MD Nastran R2.1

Installation and Operations Guide

August 15, 2007
Contents
MD Nastran R2.1 Installation and Operations Guide

Preface

List of Nastran Books  xiv
Technical Support  xv
Web  xv
Phone and Fax  xvi
Email  xvi
Training  xvi
Internet Resources  xviii

1 Introduction

Document Scope  2
Key for Readers  2
Document Structure  3
Changes to MD Nastran R2  4
MD Nastran R2  4
The nastran Command  5
Deleted, Modified, and New Keywords  5
The Directory Structure  8
Multiple Products Support  8
Multiple Computer Architecture Support  8

2 Installing MD Nastran

Overview  14
Installing MD Nastran on UNIX Systems  15
Installation Notes  15
Installation Procedures  18
Repeating a UNIX Installation  20
Installing MD Nastran on Windows Systems  22
3 Configuring MD Nastran

Overview 26

System-Specific Tuning 27
All Systems 27
AIX 27
HP-UX 29
Intel 29
Windows NT/2000 Server 29
Systems Running on Intel® Processors with HyperThreading 29

Using the “md20071” Command 30

Using the “mscinfo” Command (UNIX) 31

Managing MD Nastran Licensing 32
FLEXlm Licensing 33
Node-locked Authorization Codes 40

Activating MD Nastran Accounting 43
Enabling Account ID and Accounting Data 43
Enabling Account ID Validation 43
Securing the Accounting ID Settings and Files 49

Determining System Limits 50
HP Alpha - UNIX 50
HP-UX 51
IBM pSeries - AIX 51
Intel IA-32 - Linux 52
Intel IA-32 - Windows 52
Intel IA-64 - Linux 52
Intel x86-64 - Linux 53
NEC, SX-6, SX-8 - SUPER-UX 53
SGI R8K, R10K, R12K - IRIX64 54
Sun SPARC - Solaris 54

Specifying Parameters 55
Command Initialization and Runtime Configuration Files 55
Environment Variables 59

User-Defined Keywords 60
General Keywords 60
PARAM Keywords  61
Value Descriptors  62
Resolving Duplicate Parameter Specifications  65
Customizing Command Initialization and Runtime Configuration Files  68
Managing Remote and Distributed Hosts  74
Sample dmpdeny Implementation (AIX)  74
Limiting “memory” Requests  76
Customizing the News File  77
Customizing the Message Catalog  78
Defining a Computer Model Name and CONFIG Number  79
Generating a Timing Block for a New Computer  80
Customizing Queue Commands (UNIX)  83
Special Queues  85
Customizing the Templates  87
Keyword Reference Syntax  88
Keyword Reference Examples  89
Using Regular Expressions  92

4 Using the Basic Functions of MD Nastran

Overview  96
Using the nastran Command  97
File Types and Versioning  98
Using Filenames and Logical Symbols  99
Using the Help Facility and Other Special Functions  101
Using the Basic Keywords  102
All Systems  102
UNIX Systems  102
Queuing (UNIX)  102
Specifying Memory Sizes  104
Maximum Memory Size  105
Determining Resource Requirements  107
Estimating BUFFER SIZE 108

Using the Test Problem Libraries 109

Making File Assignments 110
ASSIGN Statement for FORTRAN Files 110
ASSIGN Statement for DBsets 110

Using Databases 113
Using the "dbs" Keyword 115
Using the ASSIGN Statement 116
Using the INIT Statement 118

Using the INCLUDE Statement 120
Specifying the INCLUDE Filename 120
Locating INCLUDE Files 122

Using the SSS Alter Library 124

Resolving Abnormal Terminations 125
Interpreting System Error Codes 125
Terminating a Job 126
Flushing .f04 and .f06 Output to Disk (UNIX) 126
Common System Errors 127

5 Using the Advanced Functions of MD Nastran

Overview 132

Using the Advanced Keywords 133
All Systems Except Primepower, SUPER-UX and UXP/V 133
AIX and Linux8664 Only 134
IRIX64 Only 134
IRIX64 and LINUX IA-64 (SGI Altix) Only 134
Solaris Only 134
SUPER-UX and Linux IA-64 (NEC Azusa) Only 134
Queuing (UNIX) 134

Using the NASTRAN Statement 136
AUTOASGN 136
BUFFPOOL, SYSTEM(114) 136
BUFFSIZE, SYSTEM(1) 137
PARALLEL, SYSTEM(107) 137
SYSTEM(128) 137
SYSTEM(198), SYSTEM(205) 137
SYSTEM(207) 137
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM(275)</td>
<td>137</td>
</tr>
<tr>
<td>Managing Memory</td>
<td>138</td>
</tr>
<tr>
<td>Managing DBsets</td>
<td>140</td>
</tr>
<tr>
<td>I/O Performance Libraries</td>
<td>140</td>
</tr>
<tr>
<td>Using the SYS Field</td>
<td>140</td>
</tr>
<tr>
<td>Using File Mapping</td>
<td>141</td>
</tr>
<tr>
<td>Using Buffered I/O</td>
<td>143</td>
</tr>
<tr>
<td>Using Asynchronous I/O</td>
<td>144</td>
</tr>
<tr>
<td>Interpreting Database File-Locking Messages (UNIX)</td>
<td>146</td>
</tr>
<tr>
<td>Interpreting the .04 File</td>
<td>149</td>
</tr>
<tr>
<td>Summary of Physical File Information</td>
<td>149</td>
</tr>
<tr>
<td>Memory Map</td>
<td>150</td>
</tr>
<tr>
<td>Day Log</td>
<td>150</td>
</tr>
<tr>
<td>User Information Messages 4157 and 6439</td>
<td>151</td>
</tr>
<tr>
<td>Memory and Disk Usage Statistics</td>
<td>152</td>
</tr>
<tr>
<td>Database Usage Statistics</td>
<td>152</td>
</tr>
<tr>
<td>Summary of Physical File I/O Activity</td>
<td>153</td>
</tr>
<tr>
<td>Running a Job on a Remote System</td>
<td>154</td>
</tr>
<tr>
<td>Installing/Running MSCRmtMgr</td>
<td>161</td>
</tr>
<tr>
<td>Running Distributed Memory Parallel (DMP) Jobs</td>
<td>163</td>
</tr>
<tr>
<td>Determining Hosts Used by DMP Jobs</td>
<td>171</td>
</tr>
<tr>
<td>Managing Host-Database Directory Assignments in DMP Jobs</td>
<td>174</td>
</tr>
<tr>
<td>Managing Files in DMP Jobs</td>
<td>175</td>
</tr>
<tr>
<td>DMP Performance Issues</td>
<td>176</td>
</tr>
<tr>
<td>Installing and Using LAM/MPI for MD Nastran on Linux</td>
<td>179</td>
</tr>
<tr>
<td>Configuring and Running SOL 600</td>
<td>185</td>
</tr>
<tr>
<td>Hardware and Software Requirements</td>
<td>185</td>
</tr>
<tr>
<td>Compatibility</td>
<td>185</td>
</tr>
<tr>
<td>Definitions</td>
<td>185</td>
</tr>
<tr>
<td>Network Configuration</td>
<td>186</td>
</tr>
<tr>
<td>Installation Notes</td>
<td>186</td>
</tr>
<tr>
<td>Solver</td>
<td>188</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>191</td>
</tr>
<tr>
<td>SOL 600 Parallel Processing on Windows</td>
<td>192</td>
</tr>
<tr>
<td>Hardware and Software Requirements</td>
<td>192</td>
</tr>
<tr>
<td>Network Configuration</td>
<td>192</td>
</tr>
<tr>
<td>Installation Notes</td>
<td>193</td>
</tr>
</tbody>
</table>
User Notes 195

Running an ISHELL Program 200
Defining Command Processor Associations 202

Using the ISHELL-INCLUDE Statement ("!") 204

Improving Network File System (NFS) Performance (UNIX) 207

Creating and Attaching Alternate Delivery Databases 208

Checkpoint Restart Facility (SGI-IRIX64) 211

6 Using the Utility Programs

Overview 214

ESTIMATE 215
Keywords 216
Rules 223
Examples 225

F04REPT 226
Options 227
Examples 228

HEATCONV 229
Keywords 229
Examples 229

MSCACT 230
Keywords 230
Examples 231
Accounting File Format 232

MSGCMP 234
Examples 234

NEUTRL 235
Keywords 235
Examples 235

OP4UTIL 236
Keywords 237

OPTCONV 239
Keywords 239
## CONTENTS

Examples 239

**PLOTPS** 240  
Keyword 240  
Examples 241  
Using the String Optimization Feature 242

RCOUT2 244  
Keywords 244  
Examples 244

RECEIVE 245  
Keywords 245  
Examples 245

TRANS 247  
Keywords 249  
Examples 249

XMONAST (UNIX) 251  
Menu Bar Commands 252  
Buttons 252  
Examples 252  
Resources 253

XNASTRAN (UNIX) 254  
Menu Bar Commands 254  
Main Window Items 254  
Resources 256

**Building the Utilities Delivered in Source Form** 257

### 7 Building and Using the Sample Programs

Overview 260

**Building and Using BEAMSERV** 261  
Building BEAMSERV 261  
Using BEAMSERV 262

**Building and Using DDLPR** 263  
Building DDLPRT 263  
Using DDLPRT 263

**Building and Using DDLQRY** 265  
Building DDLQRY 265
Using DDLQRY 265

Building and Using DEMO1 267
Building DEMO1 267
Using DEMO1 267

Building and Using DEMO2 269
Building DEMO2 269
Using DEMO2 269

Building and Using DR3SERV 271
Building DR3SERV 271
Using DR3SERV 272

Building and Using MATTST 273
Building MATTST 273
Using MATTST 273

Building and Using SMPLR 275
Building SMPLR 275
Using SMPLR 275

Building and Using TABTST 277
Building TABTST 277
Using TABTST 277

Beam Server Source Files 279
DRESP3 Server Source Files 280
MSC.Access Source Files 281

A Glossary of Terms

B Keywords and Environment Variables

Keywords 292
SYS Parameter Keywords 349
Environment Variables 352
Other Keywords 355
System Cell Keyword Mapping 360
C System Descriptions

Overview 364
Binary File Byte Ordering (Endian) 364

System Descriptions 365

Numerical Data 370
Computer Dependent Defaults 374

D Product Timing Data
Preface

- List of Nastran Books
- Technical Support
- Internet Resources
List of Nastran Books

Below is a list of some of the Nastran documents. You may order any of these documents from the MSC.Software BooksMart site at www.engineering-e.com.

<table>
<thead>
<tr>
<th>Installation and Release Guides</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Installation and Operations Guide</td>
</tr>
<tr>
<td>• Release Guide</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reference Books</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Quick Reference Guide</td>
</tr>
<tr>
<td>• DMAP Programmer’s Guide</td>
</tr>
<tr>
<td>• Reference Manual</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User’s Guides</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Getting Started</td>
</tr>
<tr>
<td>• Linear Static Analysis</td>
</tr>
<tr>
<td>• Basic Dynamic Analysis</td>
</tr>
<tr>
<td>• Advanced Dynamic Analysis</td>
</tr>
<tr>
<td>• Design Sensitivity and Optimization</td>
</tr>
<tr>
<td>• Thermal Analysis</td>
</tr>
<tr>
<td>• Numerical Methods</td>
</tr>
<tr>
<td>• Implicit Nonlinear (SOL 600)</td>
</tr>
<tr>
<td>• Explicit Nonlinear (SOL 700)</td>
</tr>
<tr>
<td>• Aeroelastic Analysis</td>
</tr>
<tr>
<td>• Superelement</td>
</tr>
<tr>
<td>• User Modifiable</td>
</tr>
<tr>
<td>• Toolkit</td>
</tr>
</tbody>
</table>
Technical Support

For help with installing or using an MSC.Software product, contact your local technical support services. Our technical support provides the following services:

- Resolution of installation problems
- Advice on specific analysis capabilities
- Advice on modeling techniques
- Resolution of specific analysis problems (e.g., fatal messages)
- Verification of code error.

If you have concerns about an analysis, we suggest that you contact us at an early stage.

You can reach technical support services on the web, by telephone, or e-mail:

Web

Go to the MSC.Software website at [www.mscsoftware.com, and click on Support](http://www.mscsoftware.com). Here, you can find a wide variety of support resources including application examples, technical application notes, available training courses, and documentation updates at the MSC.Software Training, Technical Support, and Documentation web page.
Phone and Fax

United States  
Telephone: (800) 732-7284  
Fax: (714) 784-4343  

Frimley, Camberley  
Surrey, United Kingdom  
Telephone: (44) (1276) 60 19 00  
Fax: (44) (1276) 69 11 11

Munich, Germany  
Telephone: (49) (89) 43 19 87 0  
Fax: (49) (89) 43 61 71 6

Tokyo, Japan  
Telephone: (03)-6911-1200  
Fax: (03)-6911-1201

Rome, Italy  
Telephone: (390) (6) 5 91 64 50  
Fax: (390) (6) 5 91 25 05

Paris, France  
Telephone: (03)-6911-1200  
(03)-6911-1201

Moscow, Russia  
Telephone: (7) (095) 236 6177  
Fax: (7) (095) 236 9762

Gouda, The Netherlands  
Telephone: (31) (18) 2543700  
Fax: (31) (18) 2543707

Madrid, Spain  
Telephone: (34) (91) 5560919  
Fax: (34) (91) 5567280

Email

Send a detailed description of the problem to the email address below that corresponds to the product you are using. You should receive an acknowledgement that your message was received, followed by an email from one of our Technical Support Engineers.

MSC Patran Support  
mscppatran.support@mscsoftware.com

MSC Nastran Support  
mscnastran.support@mscsoftware.com

MSC Dytran Support  
mscdytran.support@mscsoftware.com

MSC Marc Support  
mscmarc.support@mscsoftware.com

MSC Institute Course Information  
msctraining.support@mscsoftware.com

Training

The MSC Institute of Technology is the world's largest global supplier of CAD/CAM/CAE/PDM training products and services for the product design, analysis and manufacturing market. We offer over 100 courses through a global network of education centers. The Institute is uniquely positioned to optimize your investment in design and simulation software tools.
Our industry experienced expert staff is available to customize our course offerings to meet your unique training requirements. For the most effective training, The Institute also offers many of our courses at our customer's facilities.

The MSC Institute of Technology is located at:

2 MacArthur Place
Santa Ana, CA 92707
Phone: (800) 732-7211
Fax: (714) 784-4028

The Institute maintains state-of-the-art classroom facilities and individual computer graphics laboratories at training centers throughout the world. All of our courses emphasize hands-on computer laboratory work to facility skills development.

We specialize in customized training based on our evaluation of your design and simulation processes, which yields courses that are geared to your business.

In addition to traditional instructor-led classes, we also offer video and DVD courses, interactive multimedia training, web-based training, and a specialized instructor's program.

Course Information and Registration
For detailed course descriptions, schedule information, and registration call the Training Specialist at (800) 732-7211 or visit www.mscsoftware.com.
**Internet Resources**

**MSC.Software (www.mscsoftware.com)**
MSC.Software corporate site with information on the latest events, products and services for the CAD/CAE/CAM marketplace.

**Simulation Center (simulate.engineering-e.com)**
Simulate Online. The Simulation Center provides all your simulation, FEA, and other engineering tools over the Internet.

**Engineering-e.com (www.engineering-e.com)**
Engineering-e.com is the first virtual marketplace where clients can find engineering expertise, and engineers can find the goods and services they need to do their job.

**CATIASOURCE (plm.mscsoftware.com)**
Your SOURCE for Total Product Lifecycle Management Solutions.

**Process Architecture Lab (PAL) (pal.mscsoftware.com/services/pal)**
PAL is a virtual product development environment that enables PAL participants and customers to define, validate, and demonstrate advanced tools, processes, and e-business solutions.
1 Introduction

- Document Scope 2
- Document Structure 3
- Changes to MD Nastran R2 4
- The Directory Structure 8
Document Scope

The MD Nastran Installation and Operations Guide (IOG) provides instructions on how to install, customize, and use MD Nastran R2 on UNIX and Windows systems. This document assumes that you have a working knowledge of the applicable operating environments.

Note: This document includes information for systems not yet supported by MD Nastran. MSC.Software does not guarantee later support for these systems.

Key for Readers

The IOG uses certain stylistic conventions to denote user action, to emphasize particular aspects of a MD Nastran run, or to signal other differences within the text.

- **Italics**: Represent user-specified variables.
  
  **Example**: The system RC file is *install_dir/conf/nast2007rc*.

- **Courier font**: Indicates system input or output.
  
  **Example**: `$ install_dir/bin/mscid`

- **Quote marks**: Distinguish words or phrases such as lowercase keywords, commands, variables, Dbsets or file suffixes from regular text.
  
  **Example**: If “out” is not specified, MD Nastran saves the output files using the basename of the input data file as a prefix.
Document Structure

The IOG focuses on three areas of MD Nastran use and also features additional information in the form of appendixes.

Note: Chapters 2 and 3, discussing installation and configuration, are the only two chapters intended for system administrators; all other information in this document is intended for MD Nastran users.

Installation and Configuration

Chapter 2 discusses the installation of MD Nastran, while Chapter 3 demonstrates how to configure your system and MD Nastran.

Basic and Advanced Use

Chapter 4 presents the basic functions of the nastran command and provides some details on how to use system files and databases. Chapter 5 explains how to use the advanced features of the nastran command and includes information on computer resource management.

Utility and Sample Programs

The final two chapters contain information on utility and sample programs, including MSC.ACCESS and the beam server. Chapter 6 focuses on using and customizing utility programs, while Chapter 7 explains how to build and use sample programs.

Supplementary Information

In addition to these seven chapters, the IOG also includes four appendixes. Appendix A contains a glossary of terms. Appendix B reviews keywords and environmental variables. Appendix C details system descriptions, and Appendix D provides a form for product timing data.
Changes to MD Nastran R2

Modifications to MD Nastran R2 include improvements to MD Nastran capabilities, changes to the nastran command, and the utility programs.

MD Nastran R2

The directory structure for Windows has changed to install-dir\msc.software\nastran. The Unix directory structure will change in a future release.

AIX

64-bit binary only. Supports mode = i8. The 32-bit binaries have been dropped.

Fujitsu Primepower

DMP and FLEXlm are available. SMP is not available at this time. Supports mode = i8. Support for large pages (8 MB) is available by using “exe=analysis.primepower_lpg” or “exe=analysis.dmp.primepower_lpg”. We recommend that the large page binaries only be used for large problems.

Fujitsu UXPV

2005 r2 was the last release.

HPUX

64-bit binary. Supports mode = i8. The 32-bit binaries have been dropped.

HPUXIA64

Supports mode = i8.

Intel Linux IA32

LAM Daemons are now started automatically for the user.

Intel Linux IA64

DMP is available; SMP is available. Supports mode = i8.

Linux x86-64

Support for this platform has been added. Supports mode = i8.

SGI-IRIX

This 32-bit system has been dropped.
SGI-IRIX64
A Checkpoint and Restart capability (cpr) is now supported. Supports mode = i8. See “Checkpoint Restart Facility (SGI-IRIX64)” on page 211.

Solaris
64-bit binary only. The 32-bit binaries have been dropped.

UNIX
MD Nastran jobs running under LSF will now set “batch=no” and “notify=no” as defaults. These values can be overridden via the command line or RC files.

Windows
Hyperthreading. Please see “System-Specific Tuning” on page 27.

The nastran Command
The nastran command now supports automatic estimation of DBSet sizes, enhanced support for the ISHELL module and remote processing, and distributed execution.

Deleted, Modified, and New Keywords
A summary listing of the deleted, modified, and new keywords follows. See “Keywords and Environment Variables” on page 291 for a more detailed explanation of these keywords.

Table 1-1  Deleted, Modified, and New Keywords in MD Nastran R2

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Program</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoasgn</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>bfgs</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>buffpool</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>constitle</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>dbverchk</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>diaga</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>diagb</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>disksave</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>distort</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>expjid</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>f04</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>Keyword</td>
<td>Program</td>
<td>Comments</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td>f06</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>fastio</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>frqseq</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>hicore</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>hyperthreads</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>iter</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>ldqrkd</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>locbulk</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>lsymbol</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>maxlines</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>metime</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>mindef</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>minfront</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>mode</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>mperturb</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>msgbell</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>nice</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>nlines</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>nsegadd</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>punch</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>q4skew</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>q4taper</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>quadint</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>radlist</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>radmtx</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>rds</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>rdelivery</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>reexecutable</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>rgmconn</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>rmsgcat</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>redbug</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>rostype</td>
<td>nastran command</td>
<td>new</td>
</tr>
</tbody>
</table>
Table 1-1  Deleted, Modified, and New Keywords in MD Nastran R2

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Program</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>rrmtuse</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>rdirectory</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>rtimeout</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>s.rmtcmd</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>scr300</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>scr300del</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>scrsave</td>
<td>nastran command</td>
<td>new</td>
</tr>
<tr>
<td>solve</td>
<td>nastran command</td>
<td>new</td>
</tr>
</tbody>
</table>
The Directory Structure

The installation directory structure provides the following capabilities:

- Multiple versions of MSC products, such as the current and prior versions of MD Nastran.
- Multiple computer architectures, such as IBM AIX, Sun SPARC Solaris, etc.

Note: This structure does not permit both UNIX and Windows installations to share the same directory tree, e.g., on an NFS or Samba server.

Figure 1-1 shows the structure of the install_dir directory, which is selected during installation.

Multiple Products Support

The MD Nastran installation directory structure supports multiple products by using product-dependent and architecture-independent directories and files. For example, Figure 1-2 shows that the install_dir/md20071/nast directory on UNIX and install-dir/md20071\nast on Windows contains the product-dependent files for MD Nastran while the util and access directories contain the product-independent files for the various utilities and MSC.ACCESS.

Multiple Computer Architecture Support

The MD Nastran installation directory structure also supports multiple computer architectures by using architecture-dependent directories and files. All files that are dependent upon a computer architecture are isolated in a single architecture directory install_dir/md20071/arch on UNIX and
`install_dir\md20071\arch` on Windows, where `arch` is the name of the architecture, e.g., aix, alpha, hpux (see Table 3-1).

The `install_dir/md20071/nast` directory on UNIX and `install_dir\md20071\nast` directory on Windows contains news, documentation, and sample problems for MD Nastran. None of these files is architecture dependent.

The MSC.ACCESS directory (`install_dir/md20071/access` on UNIX and `install_dir\md20071\access` on Windows) contains source and make files for the MSC.ACCESS sample programs (see Figure 1-4). None of these files is architecture dependent. The DBIO library, which is architecture dependent, is
located in the architecture directory, i.e., `install_dir/md20071/arch` on UNIX and `install_dir\md20071\arch` on Windows.

Figure 1-4 Directory for access

The beam server directory (`install_dir/md20071/bmsrv` on UNIX and `install_dir\md20071\bmsrv` on Windows) contains source and make files (see Figure 1-5) for the beam server sample programs. None of these files is architecture dependent. The beam server library, which is architecture dependent, is located in the architecture directory, i.e., `install_dir/md20071/arch` on UNIX and `install_dir\md20071\arch` on Windows.

Figure 1-5 Directory for bmsrv

The utility programs directory (`install_dir/md20071/util` on UNIX and `install_dir\md20071\util` on Windows) contains source and make files (see Figure 1-6) for the utilities that are also delivered in source form. None of these files is architecture dependent.

Figure 1-6 Directory for util

The dr3 server directory (`install_dir/md20071/dr3srv` on UNIX and `install_dir\md20071\dr3srv` on Windows) contains source and make files (see Figure 1-7) for the dr3 server sample programs. None of
these files is architecture dependent. The dr3 server library, which is architecture dependent is located in the architecture directory, i.e., `install_dir/md20071/arch` on UNIX and `install_dir\md20071\arch` on Windows.

![Diagram](figure1-7.png)

**Figure 1-7** Directory for dr3srv
2 Installing MD Nastran

- Overview 14
- Installing MD Nastran on UNIX Systems 15
- Installing MD Nastran on Windows Systems 22
Overview

This chapter discusses the MD Nastran interactive installation script, and includes installation procedures for UNIX and Windows systems.

The “INSTALL.htm” and “INSTALL.txt” files, available at the top-level CD-ROM directory as soon as it is mounted, contain more detailed installation instructions, specific information about space requirements, descriptions of the other content included on the delivery media, as well as possible documentation updates that are not included in the printed or online documentation.
Installing MD Nastran on UNIX Systems

This section begins with a brief set of installation notes and general information regarding MD Nastran and the FLEXlm License Server Version 10.8. It also outlines the procedures for installing MD Nastran from local and remote CD-ROM drives. This section concludes with instructions on how to repeat a UNIX installation; this is useful when MD Nastran is being installed on a number of computers.

Installation Notes

- Environment variables that affect mscsetup include:
  - MSC_ARCH
  - MSC_BASE
  - MSC_CMDSUB
  - MSC_SETUP
  - TMPDIR.

See “Environment Variables” on page 352 for an explanation of some of these names. More complete information on the effect of these environment variables on mscsetup can be found in the script’s source.

- mscsetup is a fairly complex Korn shell script. If too many processes are running when mscsetup runs, the script may hang or generate utility errors. For best results, close or exit other applications before running mscsetup.

Note: You must have the Korn shell available as /bin/ksh.

- The disk space requirements displayed by mscsetup do not include the scratch space needed to decompress the installation files. Depending on the particular installation, up to 45 MB of additional space may be needed in the installation file system or the temporary file system. The temporary file system is defined by the
  - -t option on the mscsetup command line, for example:

  `/CDROM/mscsetup -t alternate_temporary_directory`

  - TMPDIR environment variable.
  - default temporary directory, e.g., /var/tmp on SGI systems and /tmp on others.

- The installation script identifies the system type in the first screen. If this identification is incorrect, e.g., a new computer model or a new operating system is detected, exit the script. Before restarting the script, set the environment variable MSC_ARCH to the correct architecture name as shown in Table 3-1.
If you need a FLEXlm license.dat file or a node-locked authorization code, you must do one of the following:

- Install the software and generate the “MSC.Software Contract Amendment” on the machine that will run the FLEXlm License Server (for a license.dat file) or MD Nastran (for a node-locked authorization file).
- Install the software and generate the “MSC.Software Contract Amendment” on some other machine, and manually change the hostname and ID information in the “MSC.Software Contract Amendment” to correctly identify the machine that will ultimately run the software.
- The installation procedure uses the standard gunzip and wget utilities from the Free Software Foundation. The complete gzip and wget distributions can be obtained via the Web at

http://www.gnu.org

via anonymous ftp at


or by contacting the Free Software Foundation at 51 Franklin St. - Fifth Floor, Boston, MA 02110-1301, USA.

**Note:** This is not an MSC.Software Corporation site and MSC has no control over the site’s content. MSC cannot guarantee the accuracy of the information on this site and will not be liable for any misleading or incorrect information obtained from this site.

### MD Nastran

- Any run time libraries needed by MD Nastran are included in this distribution.
- The installation test option will only be performed on the current architecture.
- You must install the “MD Nastran Utility Program Source” option if you want to customize the accounting procedures for your site.
- If you install MD Nastran in an installation base directory containing previous versions of MD Nastran, your current settings for the “authorize”, “sdirectory”, “buffsize”, and “memory” keywords will be used as defaults.
- To install MD Nastran for Distributed Memory Parallel (DMP) operations, you must select one of the following three installation schemes if you want to use more than one host in a single MD Nastran job:
  - Install MD Nastran on a filesystem that is global to every host. This provides the easiest installation and system administration, but may present network load issues when the MD Nastran is started and the delivery databases are being read.
  - Install MD Nastran on every host on host-private filesystems. This is harder to install and administer, but reduces the network load when MD Nastran is started.
• A combination of the above.

**Note:** In all cases, the `nastran` command must have the same pathname, or be in the default PATH of every host that will run a DMP job. Recall that your “.profile” and “.login” files are not used for `rcp(1)` and `rsh(1)` operations.

• With the exception of HP-UX and Intel Linux systems, you must obtain Message Passing Interface (MPI) software from your hardware vendor and install it prior to running an MD Nastran DMP job. See the “INSTALL.htm” or “INSTALL.txt” files on the CD-ROM or “DMP System Prerequisites” on page 164 to determine the MPI software requirements.

**FLEXlm License Server Version 10.8**

• In general, you should only install the FLEXlm License Server on one computer. Advanced licensing requirements may dictate more than one FLEXlm License Server.

• See “Managing MD Nastran Licensing” on page 32 for the systems supported by FLEXlm.

• If you have a FLEXlm network or counted node-lock license file, identify the name of the FLEXlm license server using “FLEXlm Server” option in the “Authorization Information” menu.

• If you have a FLEXlm uncounted node-lock license file, identify the pathname of the license.dat file using the “Authorization File” option in the “Authorization Information” menu; the file will be copied to `install-dir/flexlm/licenses/license.dat`.

• If you have a node-lock authorization code file, identify the pathname of the file using the “Authorization File” option in the “Authorization Information” menu; the file will be appended to `install-dir/conf/authorize.dat`.

• If you have a node-lock authorization code, enter the code using the “Authorization Code” option in the “Authorization Information” menu; the code will be appended to `install-dir/conf/authorize.dat`.

• The default port number for the FLEXlm license server is 1700. You must select an alternate port number if this port is already in use.

• If you want the FLEXlm license server to be automatically started at system boot time, you must run `mscsetup` as root. `mscsetup` will then be able to add an entry to your `/etc/inittab` file to start `lmgrd` at system boot time.

• After the installation is completed, see the URL

```
file:install-dir/flexlm/htmlman/flexframe.html
```

for information on configuring and using FLEXlm with MSC products. This file is part of the FLEXlm “HTML Documentation File” option.
Installation Procedures

The installation script, mscsetup, provides two installation types based on the accessibility of the CD-ROM drive to the computer running mscsetup.

- See “Installing from a Local CD-ROM” on page 18 if the CD-ROM is attached to the computer you will install the software on, or is NFS-mounted on the computer running mscsetup.
- See “Installing from a Remote CD-ROM” on page 19 if the CD-ROM is attached to another computer and is not NFS-mounted on the computer running mscsetup.

Installing from a Local CD-ROM

1. Log on to the system. You must log on to an account with root privileges to:
   - Mount the CD-ROM (not required on SGI or Sun).
   - Configure your system to automatically start the FLEXlm license server daemons at system boot time (only required if you install the FLEXlm license server software).
   - Install links in /usr/bin
   - Install the Xnastran and Xmonast app-default files.
2. If necessary, create a directory to use as the CD-ROM mount point (not required on SGI or Sun).
   
   mkdir /CDROM
3. Insert the CD-ROM and mount the CD-ROM filesystem. The device names in the following commands are examples, the actual device name on your system may differ.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fujitsu</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>HP Alpha</td>
<td>mount -rt cdfs -o noversion,rrip /dev/rr4c /CDROM</td>
</tr>
<tr>
<td>HP</td>
<td>/usr/sbin/mount -rF cdfs /dev/cd0,c1t2d0 /CDROM</td>
</tr>
<tr>
<td>Linux</td>
<td>mount /dev/cdrom /cdrom</td>
</tr>
<tr>
<td>IBM</td>
<td>mount -prv cdrfs /dev/cd0 /CDROM</td>
</tr>
<tr>
<td>SGI, Sun</td>
<td>Mounts automatically.</td>
</tr>
</tbody>
</table>
4. Run mscinfo (optional).

   HP     `/CDROM/MSCSETUP.;1’ info
   Linux  /cdrom/mscsetup info
   Sun    /cdrom/cdrom0/mscsetup info
   All Others /CDROM/mscsetup info

   If you do not want to install the software, proceed to step 7.

5. Start the installation script.

   HP     `/CDROM/MSCSETUP.;1’
   Linux  /cdrom/mscsetup
   Sun    /cdrom/cdrom0/mscsetup
   All Others /CDROM/mscsetup

6. Follow the prompts to complete the installation. See “Installation Notes” on page 15 for further information.

7. Unmount the CD-ROM.

   Linux  umount /cdrom
   SGI    eject /CDROM
   Sun    eject cdrom0
   All Others umount /CDROM

**Installing from a Remote CD-ROM**

**Note:** The “/etc/hosts.equiv” and your “.rhosts” files on the remote system must allow access from the local system.

1. Mount the CD-ROM filesystem on the remote system as described in steps 1 through 3 in “Installing from a Local CD-ROM” on page 18.

2. Log on to the system. You must log on to an account with root privileges to:
   - Configure your system to automatically start the FLEXlm license server daemons at system boot time (only required if you install the FLEXlm license server software).
   - Install links in /usr/bin
   - Install the Xnastran and Xmonast app-default files.

3. Change the working directory to /tmp or some other scratch directory.

   `cd /tmp`
4. Copy the installation files from the remote CD.

```
HP Alpha, IBM
   rsh node dd [-l user] if=file_set bs=10240 | \
   tar xvfB -
   HP
   remsh node [-l user] dd if=file_set bs=10240 | \
   tar xvfo -
Fujitsu
   rsh node dd [-l user] if=file_set bs=10240 | \
   dd obs=10240 | \
   tar xvfo -
Hitachi
   rsh node dd [-l user] if=file_set bs=10240 | \ 
   tar xvfpB -
NEC
   /usr/ucb/rsh node [-l user] dd if=file_set bs=10240 | \ 
   tar xvfo -
All others
   rsh node dd [-l user] if=file_set bs=10240 | \ 
   tar xvfoB -
```

where `node` is the network name of the remote node, `user` is an alternate user if the current user does not have remote shell privileges on `node`, and `file_set` is based on the remote system as follows:

- **HP**
  ```
  "'/CDROM/MSCSETUP.TAR;1'
  ```
- **Sun**
  ```
  /cdrom/cdrom0/mscsetup.tar
  ```
- **All others**
  ```
  /CDROM/mscsetup.tar
  ```

5. Start the interactive installation script:

```
./mscsetup
```

6. Choose option 2: *Install from a remote CD* in the “Installation Device Location” screen.

7. Follow the prompts to complete the installation. The “Installation Notes” on page 15 contain additional information.

8. Unmount the CD-ROM filesystem on the remote system as described in step 7 of “Installing from a Local CD-ROM” on page 18.

**Repeating a UNIX Installation**

You can repeat any installation using the playback file generated during every installation. A playback file can be used to reinstall MD Nastran on the same computer, or make an identical installation on another computer. The following command is used:

```
mscsetup playback-file
```

where `playback-file` is the playback file generated during a previous installation (the default playback file is `install-dir/mscsetup.pbk`).

When a playback file is used, note that:
The following environment variables will affect mscsetup: MSC_ARCH, MSC_BASE, MSC_SETUP, TMPDIR.

The architecture of every computer using the playback file must be the same as the architecture of the computer that generated the playback file.

You cannot change the installation types and user customizations.

You cannot install node-lock authorization codes using “Authorization Code” option of the “Authorization Information” menu. If you use node-lock authorization codes, you must enter the codes using one of the following methods:

1. Place the authorization codes in a file and select option 1 of the “Authorization Information” menu during the installation generating the playback file. This same file must be present during every installation using the playback file.

2. Select option D of the “Authorization Information” menu during the installation generating the playback file and manually edit the authorization code file, install-dir/conf/authorize.dat, after every installation is complete.

If the installation generating the playback file was a remote installation, and you want to mount the CD-ROM in a different system when using the playback file, the node and user can be changed with the “-r” option. For example

```
mscsetup -r node [-m /CDROM] playback-file
```

or

```
mscsetup -r user@node [-m /CDROM] playback-file
```

You only need to specify the -m option if the CD-ROM mount point changed.

You can change the installation base directory by specifying the -b option. For example

```
mscsetup -b new-install-base playback-file
```
Installing MD Nastran on Windows Systems

This section discusses the MD Nastran Windows installation. The installation notes contain information regarding performance and disk space requirements, directory structures and setup information.

Installation Notes

- You must have one of the following systems to install and run MD Nastran:
  - Intel 486DX or later processor (or compatible) running Windows NT 3.51 (or later), Windows 95, Windows 98, Windows 2000, or Windows XP, with at least 128 Megabytes RAM, 500 Megabytes available disk space to install the system, and a CD-ROM.
  
  To build the Utility Programs using the supplied source, you must also have a suitable set of compilers. Refer to “Using the Utility Programs” on page 213 and “System Descriptions” on page 365 for details.

  The default directory (called the install_dir) for MD Nastran products is “c:\msc”. This can be changed to a new or existing directory of your choice.

  The default for the MD Nastran scratch file directory is “c:\scratch”. Having this directory on a separate drive from the system swap file can help performance.

  The default program group (folder) is named MSC; you can have the icons installed in a different group if you choose. On Windows 2000 systems, this group is created as a common group if the user doing the installation has administrator authority. Otherwise, this group is created as a private group.

  To run MD Nastran from any directory, you must add the path install_dir\bin to your PATH. You can change your path in Windows by selecting the “control panel”, and then “system”. Then, click on the “Path” variable and add the following to text in the “Value” box.

```plaintext
install_dir\bin
```

Select “set”, then “OK”, and your path will be updated.

For Windows 9X, add the following text to the system AUTOEXEC.BAT file (usually C:\AUTOEXEC.BAT)

```plaintext
PATH=%PATH%;install_dir\bin
```
Installation Procedure

1. Insert the installation CD-ROM into CD drive.
2. In the file manager, open the drive containing the CD-ROM. Open the i386 directory.
3. Double click the “setup.exe” entry, and follow the instructions on the screens.
Configuring MD Nastran

- Overview 26
- System-Specific Tuning 27
- Using the "md20071" Command 30
- Using the "mscinfo" Command (UNIX) 31
- Managing MD Nastran Licensing 32
- Activating MD Nastran Accounting 43
- Determining System Limits 50
- Specifying Parameters 55
- User-Defined Keywords 60
- Resolving Duplicate Parameter Specifications 65
- Customizing Command Initialization and Runtime Configuration Files 68
- Managing Remote and Distributed Hosts 74
- Limiting “memory” Requests 76
- Customizing the News File 77
- Customizing the Message Catalog 78
- Defining a Computer Model Name and CONFIG Number 79
- Generating a Timing Block for a New Computer 80
- Customizing Queue Commands (UNIX) 83
- Customizing the Templates 87
- Using Regular Expressions 92
Overview

This chapter is intended for system administrators or anyone who needs to manage an MD Nastran installation. It starts with information on tuning your system for better performance. The chapter then concentrates on configuring MD Nastran R2 for your system. Licensing must be configured before MD Nastran will run. Other items that may require configuration include system resource limits, the command initialization file, runtime configuration files, timing blocks, and queue commands.

Two documentation conventions are used throughout the remainder of this document (typically in directory specifications). The string “install_dir” indicates the directory where MD Nastran was installed; on UNIX, this might be “/msc”, and on Windows “c:\msc”. The string “arch” indicates the MSC.Software architecture name for your computer; they are generally based on the operating system name on UNIX, while on Windows, they describe the processor. The architectures are as follows:

<table>
<thead>
<tr>
<th>Computer</th>
<th>arch</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP Alpha - UNIX</td>
<td>alpha</td>
</tr>
<tr>
<td>Fujitsu Primepower</td>
<td>solaris</td>
</tr>
<tr>
<td>HP-UX</td>
<td>hpux</td>
</tr>
<tr>
<td>IBM pSeries - AIX</td>
<td>aix</td>
</tr>
<tr>
<td>Intel Linux</td>
<td>linux</td>
</tr>
<tr>
<td>Intel Windows</td>
<td>i386</td>
</tr>
<tr>
<td>Intel x86-64</td>
<td>linux8664</td>
</tr>
<tr>
<td>Itanium Linux</td>
<td>linux64</td>
</tr>
<tr>
<td>NEC, SX-6 - SUPER-UX</td>
<td>superux</td>
</tr>
<tr>
<td>SGI R8K, R10K, R12K - IRIX64</td>
<td>irix64</td>
</tr>
<tr>
<td>Sun SPARC - Solaris</td>
<td>solaris</td>
</tr>
</tbody>
</table>

Throughout this document, while file pathnames and sample commands for Windows systems will use the standard backslash “\” directory separator character, MD Nastran also accepts pathnames using the slash “/” character as a replacement.

**Note:** The Windows command shell, CMD.EXE on Windows NT and COMMAND.COM on Windows 9X, does not accept slash “/” characters as directory separators for the command pathname, that is, the first token on the line.
System-Specific Tuning

This section presents some information on system-specific tuning that can help MD Nastran performance. Additional tuning information may be available in the “Read Me” file

install-dir/md20071/README.txt

on UNIX, or

install-dir\md20071\readme.txt

on Windows.

All Systems

All systems benefit from ensuring the I/O system is configured for the highest possible bandwidth. Setting up disk striping, or RAID-0, for use with MD Nastran databases is one of the most effective I/O performance improvements that can be made for MD Nastran.

AIX

AIX provides a utility, vmtune, that can be used by root to display and adjust AIX’s memory and paging behavior. The current values are obtained by running vmtune without options. For example,

/usr/samples/kernel/vmtune

The parameters of interest to MD Nastran tuning are

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Command Option</th>
<th>Default Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>minperm</td>
<td>-p</td>
<td>20</td>
<td>Preferred physical memory reserved for persistent storage buffers (%)</td>
</tr>
<tr>
<td>maxperm</td>
<td>-P</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>minpgahead</td>
<td>-r</td>
<td>2</td>
<td>File read-ahead (number of 4KB pages)</td>
</tr>
<tr>
<td>maxpgahead</td>
<td>-R</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>minfree</td>
<td>-f</td>
<td>120</td>
<td>Free list size (number of 4KB pages)</td>
</tr>
<tr>
<td>maxfree</td>
<td>-F</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>maxrandwrt</td>
<td>-W</td>
<td>0</td>
<td>Random writes of persistent storage buffers.</td>
</tr>
</tbody>
</table>
By default, AIX allocates 20% to 80% of physical memory for persistent storage buffers. With a memory-intensive, high-I/O bandwidth program like MD Nastran, this is too large, resulting in too few pages allocated for working sets. More appropriate values for a system primarily running MD Nastran are set with the command

```
/usr/samples/kernel/vmtune -p5 -P10
```

This sets “minperm” and “maxperm” to 5% and 10% of physical memory, respectively.

The minimum and maximum read ahead values, “minpgahead” and “maxpgahead”, are measured in 4KB pages. Proper settings for MD Nastran are a function of BUFSIZE and physical memory size. The “minfree” and “maxfree” values are the minimum and maximum number of free pages. Better settings for “average” MD Nastran workloads using the default BUFSIZE are

```
/usr/samples/kernel/vmtune -p5 -P10 -r8 -R32 -f120 -F280
```

Heavy MD Nastran workloads using larger BUFSIZE values (e.g., buffsize=32767 or larger) on a large memory system (e.g., 1GB physical memory) will benefit from a larger maximum read-ahead and free-page list, for example

```
/usr/samples/kernel/vmtune -p5 -P10 -r8 -R128 -f120 -F560
```

Users with multi-processor (SMP) system can benefit from writing persistent storage pages asynchronously by setting “maxrandwrt”. This can be added to any of the above examples,

```
/usr/samples/kernel/vmtune -p5 -P10 -W128
/usr/samples/kernel/vmtune -p5 -P10 -r8 -R32 -f120 -F280 -W128
/usr/samples/kernel/vmtune -p5 -P10 -r8 -R128 -f120 -F560 -W128
```

The vmtune command can be run at any time to change parameters, even several times during the day to suit demands of changing workloads.

The changes made by vmtune are not persistent across system restarts, you may want to set these values via an /etc/inittab entry. A sample entry is:

```
vmtune:23456:once:/usr/samples/kernel/vmtune options > /dev/console 2>&1
```

where `options` is the list of options you want to set.
HP-UX

HP-UX 11 and PA-RISC 2.0
The maximum allocatable memory is controlled by the `maxdsiz` kernel parameters. It must be large enough to accommodate the memory requests of each MD Nastran job. If this value is not large enough, MD Nastran will not be able to allocate open core memory and will terminate with the following message in the LOG file:

```
memory allocation error: unable to allocate mem words
```

where `mem` is the memory allocation request. The limit can be increased using the `sam(1M)` utility. The value is found in “Configurable Parameters” under “Kernel Parameters.”

Intel

MD Nastran makes very high memory bandwidth demands, and particular attention should be paid to the memory subsystem. A faster memory bus is more important to MD Nastran performance than a faster processor with a slower memory bus.

Windows NT/2000 Server

By default, Windows NT/2000 Server is configured to cache files as much as possible. This can cause an MD Nastran job to appear to “hang” a system running Windows NT/2000 Server.

To correct this problem, open the “Network” Control Panel applet and select the “Services” tab. Highlight “Server” and push the “Properties” button. Make sure the “Maximize Throughput for File Sharing” radio button is not selected (this is the default). Instead select either “Balance” or “Maximize Throughput for Network Applications.” Changing this option will require you to restart Windows NT/2000.

Systems Running on Intel® Processors with HyperThreading

The Intel® Pentium® 4 processor introduces a feature called HyperThreading, where a single physical processor can support more than one logical instruction stream, simulating multiple logical processors on a single physical processor. For many applications and environments, this capability may offer performance improvements over non-HyperThreading processors. If multiple MD Nastran analysis jobs are running concurrently, however, there may be performance degradations. If an installation determines this to be the case, hyperthreading should be disabled. This can be done on a permanent basis through BIOS operations or, for Windows platforms, hyperthreading may be disabled on a process by process basis using the “hyperthreads” keyword.
Using the “md20071” Command

The “md20071” command is shown as a prefix for most of the programs and commands described in this document, for example:

```
md20071 nastran ...
```

By ensuring the md20071 command is in each user’s PATH, all the commands and utilities in this release are uniformly available. The md20071 command also permits version-dependent utilities, such as TRANS, to be easily accessed.

The md20071 command is located in

```
install-dir/bin/md20071
```

on UNIX, and

```
install-dir\bin\md20071
```

on Windows.
Using the “mscinfo” Command (UNIX)

The “mscinfo” command is available on UNIX systems to display various hardware and software configuration info. This utility is run with the command

```
md20071 mscinfo
```

mscinfo will display hardware and software configuration report, including

- Hostname.
- MSCID.
- Computer Manufacturer.
- OS Name, version, and patches.
- Computer Model.
- Processor type, number, and speed.
- Window manager, Motif version, and graphics board.
- Physical and virtual memory sizes.
- Temporary directory sizes.
- Local disk sizes.

Due to the machine-dependent nature of the information, the report will vary between computer architectures.

**Note:** Root access is required to generate the complete report on some systems. If you are not root when mscinfo is run, those items requiring root access will be noted in the report.
Managing MD Nastran Licensing

**Note:** If the FLEXlm HTML documentation has been installed, additional MD-specific FLEXlm documentation can be viewed using the following URL:

```
file:install_dir/flexlm/htmlman/flexframe.html
```

Additional FLEXlm documentation can always be found at the following URL:

```
http://www.macrovision.com
```

This is not an MSC.Software Corporation site and MSC has no control over the site’s content. MSC cannot guarantee the accuracy of the information on this site and will not be liable for any misleading or incorrect information obtained from this site.

In order to run, MD Nastran requires one of the following licensing methods:

- The name of a network license server (if your computer supports FLEXlm).
- The pathname of a file containing FLEXlm licenses (if your computer supports FLEXlm).
- The pathname of a file containing one or more node-locked authorization codes.

When selecting the licensing method, MD Nastran will use the first non-null value that it finds in the following hierarchy:

1. The value of the “authorize” keyword (p. 294) on the command line.
2. The value of the MSC_LICENSE_FILE environment variable.
3. The value of the “authorize” keyword in an RC file.
4. The `install_dir/flexlm/licenses/license.dat` file, if it exists.
5. The `install_dir/conf/authorize.dat` file, if it exists.
6. The value of the LM_LICENSE_FILE environment variable.

If a non-null value cannot be found, the following User Fatal Message (UFM) is displayed by the `nastran` command:

```
*** USER FATAL MESSAGE (nastran.validate_authorize)
authorize=""     (program default)
The keyword shall not be blank or null.
```  

**UFM 3060**

If a non-null value is found for the “authorize” keyword, your MD Nastran job will be started. If the licensing information is later determined to be invalid or insufficient for the analysis, a UFM 3060 error message is printed in the .f06 file:
where \texttt{opt} is a keyword indicating the specific capability requested. The initial authorization check is for option “NAST”, subsequent checks request specific features as required by your job. Other information pertinent to this failure will be found in the LOG file.

**FLEXlm Licensing**

FLEXlm is available on the following MD Nastran platforms:

- HP Alpha -UNIX
- Fujitsu - Primepower
- HP-UX
- IBM pSeries - AIX
- Intel - Linux IA32, Linux IA64, HP-UX IA64
- Intel - Linux x86-64
- Intel - Windows
- NEC, SX-6 - SUPER-UX (client only)
- SGI R8K, R10K, R12K - IRIX64
- Sun SPARC - Solaris

Clients with network-licensed MSC software installations are encouraged to employ the most recent versions of the FLEXlm and MSC licensing daemons (lmgrd/lmutil/msc).

The binaries maintain downward compatibility, and regular upgrades are recommended, regardless of whether the current software application level required the upgrade. Updates are available at:

http://www.mscsoftware.com/support/software_updates/licserver.cfm

or from the MSC external ftp site:


MD Nastran’s implementation of FLEXlm is fully compatible with the FLEXlm implementation used by MSC.Patran 7.0 and later. Also, a license server on either UNIX or Windows can serve licenses for any number of UNIX and/or Windows systems.

FLEXlm offers two types of node-locked licensing: counted and uncounted licenses. An uncounted license does not require a license server, is the easiest to install and maintain, and offers unlimited concurrent MD Nastran jobs. A counted license requires a license server on the MD Nastran platform.
and limits the number of concurrent MD Nastran jobs. In either case you will need to determine the MSCID of the system running MD Nastran.

A FLEXlm concurrent license always requires a license server that can communicate with every computer that will run MD Nastran.

**Determining the MSCID of the FLEXlm License Server**

**Note:** Windows: The FLEXlm License Server must be accessed via TCP/IP.

If you are using a counted node-locked license or a concurrent license, the MSCID of the computer that will run the FLEXlm License Server is required. The MSCID is obtained with the command:

```
md20071 mscid
```

The command will output a line similar to

```
Please wait...
MSC ID: n
```

where \( n \) is a hexadecimal number.

**Installing a FLEXlm “license.dat” File**

A FLEXlm “license.dat” file is a text file that can be manipulated as any text file. Its default location is

```
install_dir/flexlm/licenses/license.dat
```

on UNIX, and

```
install_dir\flexlm\licenses\license.dat
```

on Windows.

**Note:** The only lines that can be altered are the SERVER, DAEMON, and comment lines; FEATURE lines, in particular, cannot be altered. On the SERVER line, the “HOSTID” field cannot be altered.

A FLEXlm license can be installed during the initial installation or any time thereafter.
UNIX

If the old license used a license server, i.e., there was a SERVER and/or DAEMON line in the file, you will need to stop and restart the FLEXlm License Server. To stop the server, enter the command

\[
\text{install_dir/bin/flexlm lmdown}
\]

It may take a few minutes for the shutdown to complete.

The new “license.dat” file is installed with the command:

\[
\text{md20071 flex license.dat}
\]

where \text{license.dat} is the new license file. This file may be an E-mail message that has been saved to disk but still contains the E-mail headers. If an existing license file is found, it will be versioned. In addition, alternate port number and options information from the SERVER and DAEMON lines will be automatically copied to the new file.

If the new license is a counted node-lock or concurrent license, restart the FLEXlm License Server with the command:

\[
\text{install_dir/bin/flexlm lmgrd}
\]

where the default log files is

\[
\text{install_dir/flexlm/lmgrd.log}
\]

An alternate log file can be specified with the “-l” option, e.g.,

\[
\text{install_dir/bin/flexlm lmgrd -l log_file}
\]

Windows

If the file was sent as an E-mail message, you will need to extract the license file portion of the message text. The actual license text is contained between the “Start of License File” and “End of License File” sentinel lines as shown:
All lines from the beginning of the file to the “Start” sentinel (inclusive), and all lines from the “End” sentinel to the end of the file (inclusive) must be deleted. You may also need to delete a “forwarding” prefix from the start of each line; this is typically the two character sequence “> ”.

Before overwriting it, you should examine the previous file to determine if any customizations were present on the SERVER or DAEMON lines. Copy these customizations to the new license file using any text editor.

**Note:** Be sure you update the `hostname`, `port`, and `pathname` lines of the SERVER and DAEMON lines to correctly reflect your installation. You cannot alter the `hostid` on the SERVER line.

### Uncounted Node-locked License

Copy the new file to

\[
install\_dir\flexlm\licenses\license.dat
\]

### Counted Node-locked or Concurrent License

Open the Control Panel applet “FLEXlm License Manager”. On the “Control” tab, select the “Stop” button to stop the FLEXlm License Server. Select the “Setup” tab to display the path name of the current license file. Copy the new file to the location shown; by default, the location is

\[
install\_dir\flexlm\licenses\license.dat
\]

Return to the “Control” tab and select the “Start” button to restart the FLEXlm License Server with the new file.
### Automatically Starting a FLEXlm Server

**UNIX**

The FLEXlm server can be automatically started at system boot time by entering one of the following lines in the “/etc/inittab” file.

<table>
<thead>
<tr>
<th>Platform</th>
<th>User</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>non-root</td>
<td><code>msclmgrd:23456:once:su user -c </code>(umask 022; install-dir/bin/flexlm lmgrd)`'</td>
</tr>
<tr>
<td></td>
<td>root</td>
<td><code>msclmgrd:23456:once:install-dir/bin/flexlm lmgrd</code></td>
</tr>
<tr>
<td>HP Alpha</td>
<td>non-root</td>
<td><code>msclmgrd:3456:once:/usr/sbin/su user -c </code>(umask 022; install-dir/bin/flexlm lmgrd)`'</td>
</tr>
<tr>
<td></td>
<td>root</td>
<td><code>msclmgrd:3456:once:install-dir/bin/flexlm lmgrd</code></td>
</tr>
<tr>
<td>Solaris</td>
<td>non-root</td>
<td><code>ml:23456:once:su user -c </code>(umask 022; install-dir/bin/flexlm lmgrd)`'</td>
</tr>
<tr>
<td></td>
<td>root</td>
<td><code>ml:23456:once:install-dir/bin/flexlm lmgrd</code></td>
</tr>
<tr>
<td>Others</td>
<td>non-root</td>
<td><code>mscl:23456:once:su user -c </code>(umask 022; install-dir/bin/flexlm lmgrd)`'</td>
</tr>
<tr>
<td></td>
<td>root</td>
<td><code>mscl:23456:once:install-dir/bin/flexlm lmgrd</code></td>
</tr>
</tbody>
</table>

**Notes:**

1. The entries in the table above should be coded in /etc/inittab as one line.
2. MSC.Software and Macrovision strongly recommend that lmgrd is not run as root. Root privilege is unnecessary and could compromise system security.

### Using FLEXlm Licensing

The following table describes various keywords that control MD Nastran’s licensing subsystem.
The “authorize” keyword is used to indicate the licensing source. The value can be any of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>@node</td>
<td>The specified node is the license server using the default port number 1700. See the description of the a.out keyword above for details.</td>
</tr>
<tr>
<td>port@node</td>
<td>The specified node is running a license server listening on the specified port.</td>
</tr>
<tr>
<td>filename</td>
<td>The specified file is used for authorization. This file may contain FLEXlm licensing information for either a node-locked or network license.</td>
</tr>
<tr>
<td>value,value,value</td>
<td>A quorum of three FLEXlm license server nodes.</td>
</tr>
<tr>
<td>value:value:...</td>
<td>UNIX: A list of FLEXlm licensing files, license server nodes, or quorums.</td>
</tr>
<tr>
<td>value:value:...</td>
<td>Windows: A list of FLEXlm licensing files, license server nodes, or quorums.</td>
</tr>
</tbody>
</table>

Examples are:

```
auth=install_dir/flexlm/licenses/license.dat
```
on a UNIX system, and

```
auth=install_dir\flexlm\licenses\license.dat
```

on a Windows system, the specified FLEXlm license file will be used. If this license file contains one or more “SERVER” lines, the file is only used to identify the server(s). If not, the file will be treated as a FLEXlm node-lock license file.

```
auth=@troll
```

Node “troll” is a FLEXlm license server using the default port number. If a port is set to ”no”, node ”troll” is a FLEXlm license server using a port number in the FLEXlm default range of 27000-27009.

```
auth=1700@troll
```

Node “troll” is a FLEXlm license server using the specified port number.

For UNIX:

```
auth=1700@banana1;1700@banana2
```

For Windows:

```
auth=1700@banana1;1700@banana2
```

Two alternate network license servers, “banana1” and “banana2”, will be used to provide network licensing services.

**Manually Starting and Stopping the FLEXlm License Server**

**UNIX**

The FLEXlm License Server is started with the command

```
install_dir/bin/flexlm lmgrd
```

where the default license and log files are
An alternate license file is specified with the “-c” option, e.g.,

```
install_dir/bin/flexlm lmgrd -c license-file
```

An alternate log file is specified with the “-l” option, e.g.,

```
install_dir/bin/flexlm lmgrd -l log-file
```

Use the following command to shut down the license server.

```
install_dir/bin/flexlm lmdown
```

or

```
install_dir/bin/flexlm lmdown -c license-file
```

It may take a few minutes for the shut down to complete.

**Note:** Do not shut down the FLEXlm license server using the kill(1) command.

**Windows NT**

To start the FLEXlm License Server, open the Control Panel applet “FLEXlm License Manager”. On the “Control” tab, select the “Start” button to start the FLEXlm License Server.

To stop the FLEXlm License Server, open the Control Panel applet “FLEXlm License Manager”. On the “Control” tab and select the “Stop” button to stop the FLEXlm License Server with the new file.

**Node-locked Authorization Codes**

The node-locked licensing system is available on all systems running MD Nastran; it remains unchanged from earlier versions.
Using Node-locked Authorization Codes

A node-locked authorization code is entered into a text file, usually install-dir/conf/authorize.dat on UNIX and install-dir\conf\authorize.dat on Windows. Any number of authorization codes for any number of computers can be present in one file. The authorize keyword is used to specify the file’s pathname, e.g.,

```
authorize=install-dir/conf/authorize.dat
```

on UNIX, or

```
authorize=install-dir\conf\authorize.dat
```

on Windows.

**Number of Users Limit (All Systems NEC)**

Node-locked licensing for MD Nastran enforces a limit on the number of users (number of seats) concurrently running MD Nastran on a single computer. This limit is defined by your contract with MSC and is encoded in the node-lock authorization code. If the maximum authorized number of jobs is already executing when a job starts, the job can wait until a seat becomes available. This wait is controlled by the “authqueue” keyword (p. 294). The default is 20, i.e., a job will wait up to twenty minutes for a seat to become available.

If a seat does not become available within the wait time, the job will terminate with the following message in the LOG file:

```
NUSR: Limit of n concurrent jobs has been reached
and queue wait period of authqueue minutes has expired.
The following jobs are currently active:
No. Username Status PID Start
--- -------- -------- ---------------
1. user active pid start_time
.
.
N usern queued pid start_time
```

where \( n \) is the maximum authorized number of concurrent jobs; \( authqueue \) is the wait time set by the “authqueue” keyword; \( user, pid, \) and \( start_time \) are the user names, process IDs, and starting times, respectively, of all MD Nastran jobs currently running or waiting to run on this computer.
Installing a Node-locked Authorization Code

An MSCID is required for the computer that will run MD Nastran. The MSCID is printed in the UFM 3060 message in the .f06 file when a run fails due to licensing problems. See “UFM 3060” on page 32. The MSCID can also be obtained with the command

```
md20071 mscid
```

This command will output a line similar to

```
Please wait...
MSC ID: n
```

where \( n \) is a hexadecimal number.

A node-locked authorization code is installed by entering the code into the authorization file using any text editor. Any number of authorization codes for any number of computers can be present in one file. The standard node-locked authorization code file is

```
install_dir/conf/authorize.dat
```

on UNIX and

```
install_dir\conf\authorize.dat
```

on Windows.

**Note:** When a job is waiting for a seat to become available, the job is consuming computer resources such as memory, swap file space, disk space, etc. Too many jobs waiting for seats could have a severe impact on the system.
Activating MD Nastran Accounting

MD Nastran provides a simple accounting package that collects usage information from each job and saves a summary of the job in the accounting directory, i.e., \texttt{install\_dir/acct} on UNIX systems and \texttt{install\_dir\acct} on Windows systems.

\begin{itemize}
  \item To activate MD Nastran accounting, set the keyword “acct=yes” in any RC file or on the command line.
  \item Placing the keyword in the system wide RC file, \texttt{install\_dir/conf/nast2007rc} on UNIX and \texttt{install\_dir\conf\nast2007.rcf} on Windows, will enable accounting for all jobs.
\end{itemize}

Instructions for generating usage summaries from the MSC accounting data are provided in the section titled “Using the Basic Keywords” on page 102.

Enabling Account ID and Accounting Data

The “acid” and “acdata” keywords are supported by the nastran command to provide hooks for a site to track additional accounting data. The “acid” keyword may be used to specify an account ID. The “acdata” keyword may be used to specify any additional accounting data needed by a site.

These keywords are activated as follows:

1. Activate accounting by putting the line “acct=yes” (page 292) in the command initialization file or a system RC file.
2. The account validation keyword, “acvalid” (page 292), can be used to validate the “acid” keyword. If “acvalid” is not defined in the command initialization file, MD Nastran will not require the “acid” keyword; if the “acvalid” keyword is defined, MD Nastran will require a valid “acid”. See “Enabling Account ID Validation” on page 43 for a complete description of this capability.

Enabling Account ID Validation

Account ID validation is enabled by defining a non-null value for the “acvalid” keyword in the command initialization file. “Specifying Parameters” on page 55 contains additional information. There are two types of account ID validation available. The nastran command’s built-in regular expression facility can be used if the account ID can be described by a regular expression (see “Using Regular Expressions” on page 92). Otherwise an external program can be used.

Validating an Account ID with a Regular Expression

To use a regular expression, the first character of the “acvalid” value must be “f” or “w” and the remainder of the value is the regular expression. The “f” indicates that an “acid” value that is not matched...
by the regular expression is a fatal error, while “w” indicates that an unmatched value is only a warning.
Note, the regular expression is always constrained to match the entire account ID string.

For the following examples, assume “acvalid=f” was set in the initialization file and an account ID is not
defined in any RC file.

```
md20071 nastran example
```

This job will fail with a message indicating an account ID is required.

```
md20071 nastran example acid=123
```

This job will be permitted to start. Since a regular expression was not defined, any non-null account ID
is valid.

For the following examples, assume “acvalid=w” is set in the initialization file and an account ID is not
defined in any RC file.

```
md20071 nastran example
```

A warning message will be issued indicating an account ID is required, but the job will be permitted to
start.

```
md20071 nastran example acid=123
```

This job will be permitted to start. Since a regular expression was not defined, any non-null account ID
is valid.

For the following examples, assume the following line is set in the command initialization file and an
account ID is not defined in any RC file:

```
acvalid=f[A-Za-z][0-9]{6}
```

This regular expression requires the account ID to be composed of a single upper- or lower-case letter
followed by six digits

```
md20071 nastran example
```

This job will fail with a message indicating an account ID is required.
Validating an Account ID with an External Program

To use an external program, the first character of the “acvalid” value must be a grave, ‘`’, and the remainder of the value is a simple command to execute the external program. The command may include keyword references but must not include pipes or conditional execution tokens.

The program must examine the account ID and write zero or more lines to its standard output indicating the result of the examination. A null output indicates a valid account ID. The non-null output is composed of two optional parts. The first part is indicated by an equal sign “=” as the first non-blank character. If this is found, the next blank delimited token is taken as a replacement account ID. With this, the external program can replace the user’s account ID with any other account ID. The second part is indicated by an “f” or “w” character. If either of these two characters are present, the remainder of the line and all remaining lines of output are taken as the body of an error message to be issued to the user. If no message text is provided, but the “f” or “w” are present, a generic message is written.

Before we discuss the external program, let’s first consider some examples of the external program’s output.

```
md20071 nastran example acid=123
```

This job will fail with a message indicating the account ID is not valid.

```
md20071 nastran example acid=2123456
```

This job will be permitted to start.

=2123456

This job will be permitted to start after the account ID is silently replaced with “Z123456”.

f

The account ID is not valid.
See your Program Manager for a valid account ID.

This job will fail with the above message.

= 2123456

w

The account ID is not valid, it has been replaced by the standard overhead charge. See your Program Manager for a valid account ID.
This job will be permitted to start after the account ID is replaced with “Z123456” and the above warning message is issued.

Sample Account Validation Programs
The account validation program can be written in any language that can process the command line. Two samples have been provided below. The Korn shell version is primarily intended for UNIX systems; the Perl version can be used on any UNIX or Windows systems that have Perl installed.

**Note:** You must have Perl installed on your system to use the Perl sample account validation program. Perl is available from numerous sources, including the URL

http://www.perl.com

This is not an MSC.Software Corporation site and MSC has no control over the site’s content. MSC cannot guarantee the accuracy of the information on this site and will not be liable for any misleading or incorrect information obtained from this site.
The Korn shell version is:

```bash
#!/bin/ksh
#
# THIS PROGRAM IS CONFIDENTIAL AND A TRADE SECRET OF MSC.SOFTWARE
# CORPORATION.  THE RECEIPT OR POSSESSION OF THIS PROGRAM DOES
# NOT CONVEY ANY RIGHTS TO REPRODUCE OR DISCLOSE ITS CONTENTS,
# SELL, LEASE, OR OTHERWISE TRANSFER IT TO ANY THIRD PARTY,
# IN WHOLE OR IN PART, WITHOUT THE SPECIFIC WRITTEN CONSENT OF
# MSC.SOFTWARE CORPORATION.
#
# Sample site-defined account validation program.
#
# usage: ksh checkac.ksh _account_file_ _account_id_
#
# If the file containing the list of valid account ID's is not specified
# or cannot be opened, report a fatal error.
#
#  if [[ $#argv -lt 1 || $#argv > 2 ]]; then
#    print "f"
#    print "Illegal usage.  See System Administrator."
#    print "f"
#    print "Account data file "$1" cannot be opened."
#    print "See System Administrator."
#  elif [[ ! -r $1 || ! -s $1 ]]; then
#    print "$1" cannot be opened."
#    print "See System Administrator."
#  elif ! [[ -z $2 ]]; then
#    acid=$(fgrep -ix $2 $1 2>/dev/null)
#    [[ -n $acid ]] && {
#      print "$acid"
#      exit
#    }
#    print "f"
#    print "The account ID is not valid."
#    print "See your Program Manager for a valid account ID."
#  fi
```

On UNIX, this program is activated with the following

```
acvalid='install-dir/bin/checkac install-dir/acct/account.dat %acid%'
```
The Perl version is:

```perl
#!/usr/local/bin/perl
#
# THIS PROGRAM IS CONFIDENTIAL AND A TRADE SECRET OF MSC.SOFTWARE
# CORPORATION. THE RECEIPT OR POSSESSION OF THIS PROGRAM DOES
# NOT CONVEY ANY RIGHTS TO REPRODUCE OR DISCLOSE ITS CONTENTS,
# SELL, LEASE, OR OTHERWISE TRANSFER IT TO ANY THIRD PARTY,
# IN WHOLE OR IN PART, WITHOUT THE SPECIFIC WRITTEN CONSENT OF
# MSC.SOFTWARE CORPORATION.
#
# Sample site-defined account validation program.
#
# usage: perl checkac.pl _account_file_ _account_id_
#
# If the file containing the list of valid account ID's is not specified
# or cannot be opened, report a fatal error.
if( $#ARGV < 0 or $#ARGV > 1 ) {
    print "f\n";
    print "Illegal usage. See System Administrator.\n";
    elsif( ! open AC, $ARGV[0] ) {
        print "f\n";
        print "Account data file \\
"; cannot be opened.\n";
        print "See System Administrator.\n"
    }
    # If no argument is specified, issue a warning and use the default
    # account ID of Z123456
    elsif( $#ARGV < 1 ) {
        print "w\n";
        print "An account ID has not been specified.\n";
        print "The standard overhead charge has been assumed.\n";
        print "See your Program Manager for a valid account ID.\n"
    } else {
        # The file is organized with one account ID per line.
        # Make sure the account ID is in the file.
        $acid = lc "$ARGV[1]";
        while( $line = <AC> ) {
            chomp $line;
            if( $acid eq lc "$line" ) {
                print "$line\n";
                exit
            }
        }
        # If we get here, the account is invalid.
        print "f\n";
        print "The account ID is not valid.\n";
        print "See your Program Manager for a valid account ID.\n"
    }
```

On Windows, this program is activated with the following

```
acvalid='perl install-dir\bin\checkac.pl install-dir\acct\account.dat %acid%'
```
Securing the Accounting ID Settings and Files

To secure the account ID settings, you must set the account ID keywords in a write-protected file and lock the values to prevent changes. For example, the following keywords can be set in the command initialization or system RC file

acct=yes
lock=acct
lock=accmd
acvalid=some-value-appropriate-to-your-site
lock=acvalid

UNIX

UNIX sites can also secure the accounting files to prevent unauthorized modification or inspection of the accounting data. This can be done by making the accounting logging program, install_dir/md20071/arch/acct, a “set uid” program.

Note: Before making install_dir/md20071/arch/acct a set-uid program, MSC.Software recommends that you carefully review the install_dir/md20071/util/mscact.c source code, ensure that you have built install_dir/md20071/arch/acct in a controlled and repeatable manner, and have performed adequate testing to ensure correct functionality.

The following commands may be executed (as root):

chown secure-user install_dir/md20071/arch/acct
chgrp secure-group install_dir/md20071/arch/acct
chmod ug+s install_dir/md20071/*acct
chmod o= install_dir/acct
chmod o= install_dir/acct/*

where secure-user is the userid that will own the files and secure-group is the groupid of the group that will own the files.
Determining System Limits

System resources can have a profound impact on the type and size of analyses that can be performed with MD Nastran. Resources that are too low can result in excessive time to complete a job or even cause a fatal error. The current resource limits on the local computer are obtained with the following command:

```
md20071 nastran limits
```

On UNIX, the resource limits on a remote computer that has MD Nastran installed are obtained with:

```
md20071 nastran limits node=remote_computer
```

**Note:**

1. The limits can vary among users and computers. If a queuing system such as NQS or NQE is installed, different limits may also be found on the various queues.

2. The output from the limits special function may specify “unlimited” on UNIX systems. In this context, “unlimited” means there is no limit on your use of a resource that is less than those architectural limits imposed by the processor or the operating system.

   - For example, on an IBM RISC System/pSeries, an unlimited virtual memory address space is limited by the smaller of the 2 gigabyte address space or the swap space configured in the operating system; on a HP Alpha, an unlimited virtual memory address space is only limited by the swap space configured in the operating system, i.e., it may exceed 2 gigabytes.

   - A more important interpretation of unlimited occurs when describing file size limitations. Table 4-7 lists those systems that support large files, i.e., in excess of 2 gigabytes. In this case, unlimited can mean $2^{32}-1$ (4,294,967,295) bytes if large files are not supported, or upwards of $2^{64}-1$ (18,446,744,073,709,551,615) bytes if large files are supported.

Sample output from this command for the various computers used to port MD Nastran follows.

**HP Alpha - UNIX**

Current resource limits:

- **CPU time:** unlimited
- **Virtual address space:** 1024 MB
- **Working set size:** 122072 KB
Data segment size: 131072 KB
Stack size: 2048 KB
Number of open files: 4096
File size: unlimited
Core dump file size: unlimited

**HP-UX**

Current resource limits:

CPU time: unlimited
Virtual address space: unlimited
Working set size: unlimited
Data segment size: 1048576 KB
Stack size: 8192 KB
Number of open files: 60
File size: unlimited
Core dump file size: 2047 MB

**IBM pSeries - AIX**

Current resource limits:

CPU time: unlimited
Working set size: unlimited
Data segment size: unlimited
Stack size: unlimited
Number of open files: 2000
File size: unlimited
Core dump file size: unlimited
Intel IA-32 - Linux

Current resource limits:
- CPU time: unlimited
- Virtual address space: unlimited
- Working set size: unlimited
- Data segment size: unlimited
- Stack size: 8192 KB (hard limit: 8192 KB)
- Number of open files: 1024 (hard limit: 1024)
- File size: unlimited
- Core dump file size: 0 MB

Intel IA-32 - Windows

Current resource limits:
- Physical memory: 255 MB
- Physical memory available: 192 MB
- Paging file size: 504 MB
- Paging file size available: 423 MB
- Virtual memory: 2074 MB
- Virtual memory available: 2033 MB

Intel IA-64-Linux

Current resource limits:
- CPU time: unlimited
- Virtual address space: unlimited
- Working set size: unlimited
- Data segment size: unlimited
- Stack size: 8192 KB
Number of open files: 1024
(hard limit: 1024)
File size: unlimited
Core memory available: unlimited

Intel x86-64 - Linux

Current resource limits:
CPU time: unlimited
Virtual address space: unlimited
Working set size: unlimited
Data segment size: unlimited
Stack size: 10240 KB
Number of open files: 1024
(hard limit: 1024)
File size: unlimited
Core memory available: unlimited

NEC, SX-6, SX-8 - SUPER-UX

Current resource limits:
CPU time: unlimited
Tasks: 16
Virtual address space: 1900 MB
Data segment size: 1945600 KB
Stack size: 1945600 KB
Number of open files: 256
File size: 8192 MB
File system space: unlimited
Temporary file space: 0 MB
Core dump file size: unlimited
### SGI R8K, R10K, R12K - IRIX64

Current resource limits:
- CPU time: unlimited
- Virtual address space: unlimited
- Working set size: 508016 KB
- Data segment size: unlimited
- Stack size: 65536 KB
- Number of open files: 200
- File size: unlimited
- Core dump file size: unlimited

### Sun SPARC - Solaris

Current resource limits:
- CPU time: unlimited
- Virtual address space: unlimited
- Data segment size: 2097148 KB
- Stack size: 8192 KB
- Number of open files: 64
- File size: unlimited
- Core dump file size: unlimited
Specifying Parameters

MD Nastran execution is controlled by a variety of parameters, either keywords or special MD Nastran statements, both required and optional. The purpose of this section is to describe how and where these parameters may be specified, not to describe these parameters in detail. This is done in subsequent sections. The MD Nastran parameters may be specified on the command line, in a command initialization (INI) file, in runtime configuration (RC) files and, for some parameters, from environment variables. The information from these sources is consolidated at execution time into a single set of values. Much of this information is passed to analysis processing in a "control file", built using the templates ("Customizing the Templates" on page 87). (The records in this control file are echoed to the .log file.) Examples of INI and RC files are given in the “User-Defined Keywords” on page 60 and “Customizing Command Initialization and Runtime Configuration Files” on page 68.

Command Initialization and Runtime Configuration Files

Although the purposes of the INI and RC files are somewhat different, the format of each file is the same. All INI and RC files are processed twice, once (the "first" pass) to extract parameters (keywords and other information) that are to be used for all MD Nastran jobs, and once (the "second" pass) to extract parameters specific to a particular job. This is accomplished by separating the INI and RC files into a series of "sections" identified by a "section header" and "subsections" within sections, identified by a subsection "header." There are two types of sections: "unconditional" and "conditional." Subsections are always "conditional."

- An unconditional section is one that starts with the name of the section enclosed in square brackets ("[ "\")"). Section names may not contain any embedded blanks but may be separated from the square brackets by any number of blanks. As currently implemented, there are three valid unconditional names: "General," "Solver" and "Nastran". (These section names are case-insensitive.) In addition, there is an implicit "unnamed" section that consists of all parameters in the INI or RC file that appear before the first named section or subsection. There is no special meaning assigned to any of the unconditional sections. Their use is optional; the section names are intended to be used for descriptive purposes.

- A conditional section or subsection is one that starts with an expression in the form:

  <keyword><operator><value>

enclosed in section header identification characters. For a conditional section, the section header identification characters are square brackets ("[ "\")), just as for unconditional sections. For a subsection, the section header identification characters are "less than" and "greater than" ("<", ">") characters. Keywords and values may not contain any embedded blanks but may be separated from each other and from the enclosing section header identification characters (the square brackets or "less than"-"greater than" characters) by any number of blanks. In the expression:
Parameters in unconditional sections, but not in subsections (which are always conditional) within unconditional sections, are processed on the first pass through an INI or RC file. On the second pass, these parameters are ignored (they are not reprocessed). Parameters in conditional sections and subsections are ignored on the first pass. Parameters in conditional sections and subsections whose expressions evaluate to "true" are processed on the second pass through an INI or RC file, thus allowing conditional expressions to reference all of the valid keywords. Note that for subsections within conditional sections, both the conditional expression for the section and the conditional expression for the subsection must evaluate to "true" before parameters in the subsection are processed.

In addition to parameters, INI and RC files may contain "comment" records. There are two types of comment records: ignored and printed.

- Ignored comments are records that start with a semi-colon (";") or pound sign ("#"). These records are completely ignored. When running in Windows, there is a special form of ignored comments that may be specified in an INI file (but not in RC files). These are records that start with "REM", short for "REMARK". The test for "REM" is case-insensitive.
- Printed comments are records that start with the currency symbol ("$"). These records are passed on as part of the analysis information but are otherwise ignored.
The list below specifies the INI and RC files that MD Nastran uses. Table 3-3 lists the keywords that are generally set in the unconditional sections of the command initialization file. Table 3-4 lists the keywords that are generally set in RC files.

- **Command Initialization (INI) File**
  This file is used to define keywords that are to be set whenever the nastran command is executed. Typical keywords in the unconditional sections include the installation base directory and the version of MD Nastran. Conditional sections and subsections might include keywords such as "rcmd" and "rsdirectory" in sections that are conditional upon the value of the "node" keyword.
  
  **UNIX:** `install_dir/md20071/arch/nastran.ini`
  At installation time, this name is linked to `install_dir/bin/nast2007.ini`
  
  **Windows:** `install_dir\md20071\i386\nastran.ini` or `install_dir\bin\nastran.ini`
  The file used is the first one found.

- **System RC File**
  This file is used to define parameters that are applied to all MD Nastran jobs using this installation structure. Many of the parameters that might be specified in the INI file could, alternatively, be specified in this file.
  
  **UNIX:** `install_dir/conf/nast2007rc`
  
  **Windows:** `install_dir\conf\nast2007.rcf`

- **Architecture RC File**
  This file is used to define parameters that are applied to MD Nastran jobs using this architecture.
  
  **UNIX:** `install_dir/conf\arch\nast2007rc`
  
  **Windows:** `install_dir\conf\arch\nast2007.rcf`
• Node RC File
   This file is used to define parameters that are applied to MD Nastran jobs running on this node. Alternatively, the parameters in this file could be specified in a conditional section in one of the previous files, using nodename as the value of the "s.hostname" keyword in the conditional expression.
   UNIX:  \install\dir\conf\net\nodename\nast2007rc
   Windows:  \install\dir\conf\net\nodename\nast2007.rcf

• User RC File
   This file is used to define parameters that are applied to MD Nastran jobs run by an individual user.
   UNIX:  $HOME/.nast2007rc
   Windows:  %HOMEDRIVE%\HOMEPATH\nast2007.rcf

• Local RC File
   This file should be used to define parameters that are applied to MD Nastran jobs that reside in the input data file's directory. This RC file is in the same directory as the input data file. If the "rcf" keyword (page 331) is used, this local file is ignored.
   UNIX:  .nast2007rc
   Windows:  nast2007.rcf

Please note that the UNIX shorthand ".", to refer to your or another user's home directory, cannot be used in an RC file. In addition, environment variables are only recognized within the context of a logical symbol definition.

In addition to keyword specifications, the following MD Nastran statements (from the NASTRAN and FMS Sections) may appear in RC files and conditional sections in an INI file: NASTRAN, ACQUIRE, ASSIGN, CONNECT, DBCLEAN, DBDICT, DBDIR, DBFIX, DBLOAD, DBLOCATE, DBSETDEL, DBUNLOAD, DBUPDATE, DEFINE, ECHOFF, ECHOON, ENDJOB, EXPAND, INCLUDE, INIT, PROJ, RESTART and RFINCLUDE. Except for minimal checking of the NASTRAN and PARAM statements, the syntax of these statements is not validated. These records are simply passed on for use in MD Nastran analysis processing.

Starting with MD Nastran, INI files and RC files also may contain PARAM statements that specify values that affect MD Nastran analysis processing. The values associated with PARAM names may be specified using PARAM statements in INI files and RC files or by using PARAM keywords, defined using the PARAM keywords feature as described in "User-Defined Keywords" on page 60. PARAM statements must be specified in "free-field format", i.e., in the Case Control PARAM format (PARAM{name,value}), not in Bulk Data fixed-field format. Please see "Parameters" in Chapter 5 of the MD Nastran Quick Reference Guide for more information on PARAM names and statements and their usage.
Environment Variables

Several keywords may have their values set from associated environment variables. When this is the case, the environment variable takes precedence over any INI or RC file keyword specification. A command-line specification will over-ride the environment variable specified value. This same precedence rule applies to user-defined keywords that may have their initial values taken from environment variables, as described in the next section. A list of the keywords and their associated environment variables, along with a description of each keyword, may be obtained by using the following command:

```
md20071 nastran help env
```
User-Defined Keywords

In addition to the internally defined keywords (see “Keywords” in Appendix B), MD Nastran allows users to define their own keywords. There are two classes of user-defined keywords:

- General keywords. These are intended for use in INI file or RC file conditional section clauses, in user modifications to the run template files (nastran.dmp, nastran.lcl, nastran.rmt or nastran.srv) and, for UNIX, in customized queue commands ("submit" keyword).

- PARAM keywords. These are keywords associated with a PARAM name. Using descriptive keywords to set a PARAM value may be more convenient than specifying the PARAM statement in an RC file. Also, keywords are not limited to a maximum of eight characters, as PARAM names are, and may be more descriptive of the action being affected or requested.

User-defined keywords are supported by the "help" and "whence" functions.

General Keywords

These keywords are defined in the file specified by the "0.kwds" keyword. The default file names are:

UNIX:  

\[ install\_dir/md20071/arch/nastran.kwds \]

At installation time, this name is linked to \[ install\_dir/bin/nast2007.kwds \]

Windows:  

\[ install\_dir\md20071\i386\nastran.kwds \] or  

\[ install\_dir\bin\nast2007.kwds \]

The file used is the first one found.

The records in this file consist of:

- Comment records. These are records that start with a comment character (hash, '#', semi-colon, ';', or currency symbol, '$') and are completely ignored.
- Blank or null records. These records are ignored.
- Keyword records. These records consist of the keyword name along with an optional value descriptor and comment in the form:

\[ \text{keyword\_name} : \text{value\_descriptor} \text{ comment} \]

where:

\[ \text{keyword\_name} \]

is the name to be assigned to the user keyword. This name may not contain any embedded blanks and may not be the same as any internal keyword or previously specified user-defined keyword. It is also case-insensitive except in the case when its initial value may be set from an environment variable with the same name.
value_descriptor is optional. If specified, it should be as described in “Value Descriptors” on page 62 and may not contain any embedded blanks. If this field is not present, the separating colon may be omitted. The default value descriptor is "string". This field may also specify that the initial value of this keyword be taken from an environment variable with the same name.

comment is an optional comment field. If present, it must be separated from value_descriptor or keyword_name by blanks or must begin with a comment character.

There may be any number of leading blanks in the record and before and after the separating colon.

General keywords and the values assigned to them only affect MD Nastran processing if:

- there are customized INI and RC files that have conditional sections, using these keywords in expressions, that specify other keywords and statements (e.g., NASTRAN and PARAM statements) that modify MD Nastran processing to meet the requirements of a user's site and installation.
- they are used in customized templates (“Customizing the Templates” on page 87).
- for UNIX systems, they are used in customized queue commands defined using the "submit" keyword (“Customizing Queue Commands (UNIX)” on page 83).

PARAM Keywords

These keywords are defined in the file specified by the "0.params" keyword The default file names are:

UNIX: \install_dir\md20071\arch\nastran.params

At installation time, this name is linked to \install_dir\bin\nast2007.params

Windows: \install_dir\md20071\i386\nastran.params or \install_dir\bin\nast2007.params

The file used is the first one found.

The records in this file consist of:

- Comment records. These are records that start with a comment character (hash, '#', semi-colon, ';', or currency symbol, '$') and are completely ignored.
- Blank or null records. These records are ignored.
- Keyword-name records. These records consist of the keyword name, the associated PARAM name, along with an optional value descriptor and comment in the form:
keyword_name : param_name : value_descriptor comment

where:

keyword_name is the name to be assigned to the PARAM keyword. This name is case-insensitive, may not contain any embedded blanks and may not be the same as any internal keyword, general user-defined keyword or previously specified PARAM keyword.

param_name is the PARAM name to be associated with keyword_name. This name is case-insensitive, may be a maximum of eight characters, must begin with an alphabetic character and may not contain any embedded blanks. Also, it may not be the same as any previously specified PARAM name.

value_descriptor is optional. If specified, it should be as described in Value Descriptors and may not contain any embedded blanks. If this field is not present, the separating colon may be omitted. The default value descriptor is "string".

comment is an optional comment field. If present, it must be separated from value_descriptor or param_name by blanks or must begin with a comment character.

There may be any number of leading blanks in the record and before and after the separating colons.

Keyword names that are the same as PARAM names are allowed, as long as the keyword name is not an internal or general user-defined keyword name.

Values associated with PARAM names, whether set using PARAM keywords or set using PARAM statements (statements having the form PARAM, name, value), directly affect MD Nastran analysis processing.

Value Descriptors

Value descriptors enable limited syntax checking for values assigned to general and PARAM user-defined keywords. For general keywords, they may also specify that the initial value of the keyword be set from the value associated with the environment variable having the same name as the keyword. There are two types of syntax checking available: value must be one of a list of entries or value must be numeric. Also, the two forms can be combined. These are specified as follows:

List: {"val1","val2",...,"valn"}

That is, the acceptable values are enclosed in double quotes ("" ) and separated from each other by commas. The specification, including the various acceptable values, may not contain any embedded blanks. Values are case-insensitive and any partial specification is acceptable and will be replaced by the
full value. For example, if a keyword may only have the values "preliminary", "check" and "final", the value descriptor would be:

{"Preliminary","Check","final"}

and a value specification of "Ch" would be accepted and replaced by "check".

Numeric: number

Values will be checked to see if they are valid numbers, either integer or floating point. For example, valid keyword value specifications could be: "1", "-3.247", "4.e-5", "3.75.4", "4.24x" and "-4-5" are invalid specifications.

**Note:** This checking does not support the NASTRAN "nnnseee" numeric format, where the 'e' between the number and the signed exponent ("seee") is missing.

Complex value: number,number

This format is only supported for PARAM keyword value descriptors. Values will be checked to see if they consist of two valid numeric values, separated by a comma.

Combined: {"val1","val2",...,"valn",number}

**Note:** This "combined" format does not support complex numbers.

In addition, for general keywords, if the value descriptor starts or ends with the string "env", specified in any case and separated from the rest of the value descriptor with a comma (unless the value descriptor is only "env"), the keyword value will be set using the value associated with the environment variable having the same name as the keyword. The environment variable will be subjected to the same syntax-checking rules that an INI file, RC file or command line specification would be, with a warning message generated if syntax checking fails. This occurs even if the keyword is specified on the command line. Note that, for UNIX systems, since environment variable names are case-sensitive, the keyword name must be specified exactly the same as the environment variable name. This is the only time that the keyword name is case-sensitive. For Windows systems, since environment variable names are not case-sensitive, this restriction does not apply. Keyword values set from environment variables over-ride keyword values set in INI or RC files but do not over-ride keyword values set on the command line.

If a value descriptor is omitted or is not one of these formats, no syntax checking will be performed.

**Examples:**

1. The following value descriptor would accept a value of "test", "final" or a number:
Acceptable values would be: te (replaced by test), FIN (replaced by final), 7, 14.5, 3.e-4, -5

2. The following value descriptor would accept only the strings "abc", "def", "ghi" and "glm":

\{"abc","def","ghi","glm"\}

Acceptable values would be: g (replaced by ghi), aB (replaced by abc), gl (replaced by glm), D (replaced by def)

3. The following value descriptor, only valid for a PARAM keyword, would only accept a complex number specification:

number, number

Acceptable values would be: 1, 2, 7.54, 3.14

4. The following value descriptors, only valid for a general keyword, would accept only the strings "qrs", "test", and "xyz". In addition, the value descriptor requests that the keyword value be set from the environment.

enV, \{"qrs","test","xyz"\}

or

\{"qrs","test","xyz"\}, Env

Acceptable values would be: q (replaced by qrs), xY (replaced by xyz), T (replaced by test)
CHAPTER 3
Configuring MD Nastran

Resolving Duplicate Parameter Specifications

MD Nastran processing information is obtained by scanning the various INI and RC files, the system environment, and the MD Nastran command line in the following order:

1. MD Nastran command line, first pass. Only "program options", i.e., "-x" options, are processed during this command line scan. For example, this is when the "-i ini_file_name" program option is processed.

2. Environment variables, first pass. During this pass, the only keywords whose values are set are those that may only be specified as environment variables. This includes keywords such as HOME (for UNIX), HOMEDRIVE and HOMEPATH (for WINDOWS) and PWD.

3. INI file, pass 1, if this file exists. During this pass, only unconditional sections are processed. Generally, the only keywords processed in this pass are: 0.kwds, 0.params, accmd, acvalid, rcmd, rsdirectory, sysmsg and version (although rcmd and rsdirectory probably should be in conditional sections scanned during the second pass).

4. Environment variables, second pass. During this pass, only those keywords that may only be set in global sections of the INI file or as environment variables are processed. This includes keywords such as MSC_ARCH, MSC_BASE and MSC_VERSD.

5. MD Nastran command line, second pass. The only general use keywords processed during this command line scan are: dmparallel, jid, jidpath, jidtype, node, pause, rcf, username, version and whence. The processing of other command line keywords is deferred until later command line scans.

This is the time that the user-defined keyword definition files (for both general use and PARAM keywords), if any, are processed and the keyword specifications defined by these files are added to the keywords tables. The keywords defined in these files may be used just as internal keywords are used. (See “User-Defined Keywords” on page 60.)

6. System RC file, pass 1, if this file exists. During this pass, only unconditional sections are processed.

7. Architecture RC file, pass 1, if this file exists. During this pass, only unconditional sections are processed.

8. Node RC file, pass 1, if this file exists. During this pass, only unconditional sections are processed.

9. User RC file, pass 1, if this file exists. During this pass, only unconditional sections are processed.

10. Local RC file, pass 1, if this file exists. During this pass, only unconditional sections are processed.

11. Environment variables, third pass. During this pass, only "general" user-defined keywords that have been flagged to be set from environment variables are processed. (This pass will be skipped if there are no "general" user-defined keywords.)
12. MD Nastran command line, third pass. Only “general” user-defined keywords are processed during this command line scan. (This pass will be skipped if there are no “general” user-defined keywords.)

At this point, all keyword values that can be used in conditional section expressions are known.

13. INI file, pass 2, if this file exists and has conditional sections. During this pass, only the conditional sections are processed.

14. System RC file, pass 2, if this file exists and has conditional sections. During this pass, only the conditional sections are processed.

15. Architecture RC file, pass 2, if this file exists and has conditional sections. During this pass, only the conditional sections are processed.

16. Node RC file, pass 2, if this file exists and has conditional sections. During this pass, only the conditional sections are processed.

17. User RC file, pass 2, if this file exists and has conditional sections. During this pass, only the conditional sections are processed.

18. Local RC file, pass 2, if this file exists and has conditional sections and if it is not ignored. During this pass, only the conditional sections are processed.

19. Environment variables, fourth pass. During this pass, all keywords that may be set from environment variables and that have not been processed previously are now processed.

20. MD Nastran command line, fourth pass. All keywords not processed during the previous passes are now processed. For example, this is when user-defined PARAM keyword specifications are processed.

At this point, all information necessary to generate the “control file” has been collected. This file is generated when the “script templates” (see “Customizing the Templates” on page 87) are processed.

21. NASTRAN, FMS and PARAM statements in the input file.

If duplicate keywords are encountered, the last specification found is the one used. That is, the above list specifies the precedence order, from lowest precedence (number 1) to highest (number 21). The only case in which the last keyword specification is not used is when keywords are "locked", i.e., when a specification of the form

\[ \text{lock} = \text{keyword} \]

is processed. After this "lock" request is processed, any requests to set \textit{keyword}, whether from INI files, RC files, environment variables or command line arguments, are quietly ignored. That is, processing proceeds as if any \textit{keyword} specifications specified after the "\text{lock} = \text{keyword}" request do not exist. Once a keyword has been "locked," there is no way to "unlock" it. (Note that it is valid to "lock" the \text{lock} keyword itself.)

If duplicate NASTRAN and FMS statements are encountered, they are simply passed on for use in MD Nastran analysis processing in the order in which they were encountered.
Thus, the general rule for resolution is:

- Information specified in NASTRAN input data files always takes precedence over any other values.
- Command line parameters have the next highest precedence.
- Environment variables associated with keywords and that have non-null values are next.
- RC file parameter specifications are next.
- INI file parameter specifications are last.

Generally, the only exceptions to this precedence ordering are "general" user-defined keyword specifications. The command line values take precedence over values specified in unconditional INI file and RC file sections but have lower precedence than values specified in conditional INI file and RC file sections. Because the primary purpose for general user-defined keywords is for conditional section selection, changing a general user-defined keyword in a conditional section may lead to unexpected results. Such specifications should be used with care. Also, because user-defined PARAM keywords on the command line are not processed until the last command line scan, PARAM keywords should not be used in INI file and RC file conditional section expressions since command line specified values will not be in effect when these expressions are evaluated.

Because PARAM values may be specified either using PARAM statements or using PARAM keywords, they require further explanation. PARAM statements and PARAM keywords referring to the same PARAM name are considered equivalent definitions for the PARAM name. As such, the last specification, regardless of whether it was a PARAM statement or a PARAM keyword, is the one that is used to establish the value associated with the PARAM name.
Customizing Command Initialization and Runtime Configuration Files

Table 3-3 lists the keywords that are generally set in the unconditional sections of the command initialization file.

Table 3-3  Command Initialization File Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.kwds</td>
<td>Alternate name for user-defined keywords definition file.</td>
</tr>
<tr>
<td>0.params</td>
<td>Alternate name for PARAM keywords definition file</td>
</tr>
<tr>
<td>acct</td>
<td>Enables job accounting, see “Enabling Account ID and Accounting Data” on page 43.</td>
</tr>
<tr>
<td>acvalid</td>
<td>Activates account ID validation, see “Enabling Account ID Validation” on page 43.</td>
</tr>
<tr>
<td>MSC_BASE</td>
<td>Defines the installation base directory. Normally this is defined as an environment variable by the md20071 command.</td>
</tr>
<tr>
<td>version</td>
<td>Specifies the default version of MD Nastran to be run.</td>
</tr>
</tbody>
</table>

Most of the command line keywords can be set in any of the RC files. Table 3-4 lists keywords that are generally set in the system, architecture, or node RC files:

Table 3-4  RC File Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Preferred RC File</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>accmd</td>
<td>System</td>
<td>Command line to invoke accounting logger program.</td>
</tr>
<tr>
<td>acct</td>
<td>System</td>
<td>Enables job accounting.</td>
</tr>
<tr>
<td>acvalid</td>
<td>System</td>
<td>Enables account ID (acid) validation.</td>
</tr>
<tr>
<td>authorize</td>
<td>System</td>
<td>Specifies the licensing method.</td>
</tr>
<tr>
<td>lock</td>
<td>Any</td>
<td>Prevent further changes to a keyword's value.</td>
</tr>
<tr>
<td>memory</td>
<td>Node</td>
<td>Specifies a default memory allocation</td>
</tr>
<tr>
<td>memorymaximum</td>
<td>Node</td>
<td>Specifies a maximum &quot;memory&quot; request.</td>
</tr>
<tr>
<td>ncmd</td>
<td>Architecture</td>
<td>Specifies the notify command when &quot;notify=yes&quot; is set.</td>
</tr>
<tr>
<td>news</td>
<td>System</td>
<td>Controls the display of the news file at the beginning of the .f06 file.</td>
</tr>
<tr>
<td>post</td>
<td>Architecture</td>
<td>UNIX: Specifies commands to be run after each job is completed.</td>
</tr>
</tbody>
</table>
Examples:
The following (relatively simplistic) examples illustrate how unconditional and conditional sections could be used.
Example 1:
Assumptions: There are three computer nodes, sysnode1, sysnode2 and sysnode3, that may be accessed.

On sysnode1:
- MD Nastran 2006 r1 and MD Nastran R2 are installed:
  - MD Nastran 2006 r1 is accessed using "/local/msc/bin/nast2006"
  - MD Nastran R2 is accessed using "/local/msc/bin/mdnast2007"
  - The scratch directory is /local/temp

On sysnode2:
- Only MD Nastran 2006 r1 is installed and is accessed using "/local1/msc/bin/nast2006"
- The scratch directory is /local1/temp

On sysnode3:
- MD Nastran 2006 r1 and MD Nastran R2 are installed:
  - MD Nastran 2006 r1 is accessed using "/local2/msc/bin/nast2006"
  - MD Nastran R2 is accessed using "/local2/msc/bin/mdnast2007"
  - The scratch directory is /local2/temp

All of this information could be specified in an INI file, identical on all three nodes, as follows:

```
; ; This is the MD Nastran Command Initialization File
; The default version is to be set to 2007.
; version=2007.0

; Define conditional sections giving the appropriate sdir
; values when MD Nastran is run locally.
[ s.hostname =  sysnode1 ]
  sdir=/local/temp
[ s.hostname =  sysnode2 ]
  sdir=/local1/temp
[ s.hostname =  sysnode3 ]
  sdir=/local2/temp

; Define conditional sections giving the appropriate
; remote access keywords when a "node" value,
; requesting remote execution, is specified.
[ node =  sysnode1 ]
  rsdir=/local/temp
  < version = 2006.0 >
  rcmd=/local/msc/bin/nast2006
  < version = 2007.0 >
  rcmd=/local/msc/bin/nast2007
```
[ node = sysnode2 ]
rsdir=/local1/temp
< version = 2007.0 >
rcmd=/local1/msc/bin/mdnast2007

[ node = sysnode3 ]
rsdir=/local2/temp
< version = 2006.0 >
rcmd=/local2/msc/bin/nast2006
rcmd=/local2/msc/bin/mdnast2007
< version = 2007.0 >
rcmd=/local2/msc/bin/mdnast2007

; ; This is the end of the Command Initialization file
;

Alternatively, the information could be split between an INI file and a system RC file, identical on all three nodes, as follows:

In the INI file:

; ; This is the MD Nastran Command Initialization File
; The default version is to be set to 2007.
; version=2007.0

; Define conditional sections giving the appropriate
; remote access keywords when a "node" value,
; requesting remote execution, is specified.
;
[ node = sysnode1 ]
rsdir=/local/temp
< version = 2006.0 >
rcmd=/local/msc/bin/nast2006
rcmd=/local/msc/bin/mdnast2007
< version = 2007.0 >
rcmd=/local/msc/bin/mdnast2007

[ node = sysnode2 ]
rsdir=/local1/temp
< version = 2007.0 >
rcmd=/local1/msc/bin/mdnast2007

[ node = sysnode3 ]
rsdir=/local2/temp
< version = 2006.0 >
rcmd=/local2/msc/bin/nast2006
rcmd=/local2/msc/bin/mdnast2007
< version = 2007.0 >
rcmd=/local2/msc/bin/mdnast2007

; ; This is the end of the Command Initialization file;
In the system RC file, identical on all three nodes:

`; This is the MD Nastran system RC file. ;
; Define conditional sections giving the appropriate sdir ; values when MD Nastran is run locally.

[s.hostname = sysnode1]
sdir=/local/temp

[s.hostname = sysnode2]
sdir=/local1/temp

[s.hostname = sysnode3]
sdir=/local2/temp

; This is the end of the system RC file ;

Example 2:

Assumptions: User keywords defining "run type" and "data complexity" are needed and AUTOSPC, AUTOSPCR, BAILOUT and ERROR PARAM values are to be set based on these keywords.

The nastran.kwds file could be:

`; User Keywords
Runtype:['prelim','development','final'];Analysis stage
  Level :    number       # Data complexity level

The nastran.params file could be:

; PARAM keywords
Set_AutoSPC : AutoSPC : ['Yes','No']
Set_AutoSPCR : AUTOSPCR : ['yes','no']
Bailout_Value : bailout : number
Set_Error : Error : number

Then, the system RC file could contain:

; RC file
[runttype = prelim ]
set_autospc = yes
bailout_value = -1
set_error = 0
set_autosp_cr = yes

[runttype = development ]
set_autospc=yes
bailout_value=0
set_error=-1
[runtype=final]
set_autospc=no
param,bailout,0
param,error,-1
param,autospcr,no

[level < 3]
; basic data complexity parameters
[level >= 3]
<level>8>
; advanced data complexity parameters
<level<=8>
; intermediate data complexity parameters

; End of RC file
Managing Remote and Distributed Hosts

Your site can control the hosts available to remote and distributed (DMP) jobs by creating host “accept” or “deny” utilities that list the hosts that a remote or DMP job may or may not use respectively.

For remote jobs, specified by “node=node-name”, the two utilities are install-dir/md20071/arch/rmtaccept and install-dir/md20071/arch/rmtdeny.

For DMP jobs, specified by “dmparallel=number”, the two utilities are install-dir/md20071/arch/dmpaccept and install-dir/md20071/arch/dmpdeny.

The “rmtdeny” and “dmpdeny” utilities list those hosts that cannot be used by a remote or DMP job. The “rmtaccept” and “dmpaccept” utilities lists those hosts that can be used by a remote or DMP job. At most one and only one of these utilities will be used. The nastran command will first look for the “deny” utility. If it exists and is executable, it will be run and its stdout parsed — any host listed cannot be selected by the job. If the “deny” utility does not exist, the nastran command will look for the “accept” utility. If it exists and is executable, it will be run and its stdout parsed — only those hosts listed can be selected by the job. If neither utility exists, any host will be accepted.

The required output format of these utilities is one host per line of output. For example, consider the following output:

```
banana1
banana2
```

If written by a “deny” utility, neither “banana1” nor “banana2” will be available to an MD Nastran job; if written by an “accept” utility, only these two hosts will be available.

See “Sample dmpdeny Implementation (AIX)” on page 74 for a special format supported only for the dmpdeny utility on AIX.

Sample dmpdeny Implementation (AIX)

MSC.Software has provided a default “install-dir/md20071/aix/dmpdeny” utility that converts the output of the IBM Parallel Environment for AIX command

```
jmstatus -j
```

into a form usable by the nastran command. The format of the dmpdeny output on AIX is:

```
host:adapter
```
where _host_ is the name of a host where the jmstatus output is listed as “DEDICATED”, and _adapter_ is the name of an adapter where the jmstatus output is listed as “DEDICATED”. The utility has been annotated to describe this process.

This sample implementation provides a trivial job control facility that can be used as-is, replaced with code more appropriate to your site, or removed.
Limiting “memory” Requests

The nastran command provides a “memorymaximum” keyword that permits you to specify a maximum memory request on a site-wide, per-architecture, or per-node basis. This value can be set to any legal memory size.

The default values are

memorymaximum=0.8*physical

on UNIX, and

memorymaximum=1.2*physical

on Windows. If this limit is exceeded, the nastran command will issue a UWM and reduce the memory request.

Note: As installed, the computer’s physical memory is only known on HP Alpha UNIX, Solaris, SUPER-UX, and Windows. Other systems must specify a hard limit or specify the physical memory size via the “s.pmem” keyword.

You may leave the default limits in place, or specify any value or values appropriate to your site.

It may be advisable to lock this keyword to ensure the limit is not removed. This is accomplished with the RC file entry

lock=memorymaximum

Note: Be sure you specify this line after any specification of the “memorymaximum” keyword.
Customizing the News File

MSC delivers a news file (install_dir/md20071/nast/news.txt on UNIX and install_dir\md20071\nast\news.txt on Windows) that briefly describes important new features of the release. You can also use news file to distribute information to the users of MD Nastran.

There are two ways the news file can be viewed. The most common way is by specifying “news=yes” or “news=auto” on the command line or in an RC file. This specification will cause the news file to be printed in the .f06 file just after the title page block. The other method is by using the news special function

md20071 nastran news

This will display the news file on the screen.
Customizing the Message Catalog

MD Nastran uses a message catalog for many messages displayed in the .f06 file. The standard message catalog source file is

```
install_dir/md20071/util/analysis.txt
```

on UNIX and

```
install_dir\md20071\util\analysis.txt
```

on Windows. This file may be modified to meet the needs of a site or a user. Once the changes have been made, a message catalog is generated using the command

```
md20071 msgcmp myfile
```

where “myfile.txt” is the message catalog source file. This command will generate a message catalog in the current directory with the name “myfile.msg”. The message catalog is identified with the “msgcat” keyword (p. 319), and can be tested using the command

```
md20071 nastran msgcat=myfile.msg other_nastran_keywords
```

Once the message catalog has been validated, it may be installed with the command

```
cp myfile.msg install_dir/md20071/arch/analysis.msg
```

on UNIX, or

```
 COPY myfile.msg install_dir/md20071/arch/analysis.msg
```

on Windows, where `install_dir` is the installation base directory and `arch` is the architecture of the system using the message catalog. You will need write permission to the architecture directory to do this.

**Note:** Message catalogs are computer-dependent. Table 6-1 identifies the systems that are binary compatible; binary compatible systems can use the same message file.
Defining a Computer Model Name and CONFIG Number

If the nastran command cannot identify a computer, the following message will be written to the screen before the MD Nastran job begins:

```
*** SYSTEM WARNING MESSAGE (nastran.validate_local_keywords)
 s.config=0    (program default)
    Default CONFIG value.
    A config number for this computer could not be
determined. Defining this computer in the model file
install_dir/conf/arch/model.dat, using rawid=rawid; or
defining <config> in an RC file may correct this
problem.
```

There are two possible resolutions to this warning message. The preferred solution is to create the file install_dir/conf/arch/model.dat on UNIX or install_dir/conf/arch/model.dat on Windows with the model name and configuration number of the computer. This file contains zero or more lines of the form:

```
model, proc, rawid, config
```

where

- `model` is the name of the computer model. This string should be enclosed in quote marks if it contains spaces or commas.
- `proc` is the file type of the alternate executable. This value is set to null to select the standard executable. The “system” special function reports this name.
- `rawid` is the “rawid” value reported in the above message text or by the “system” special function.
- `config` The CONFIG number used to select the timing constants. If this value is null, `rawid` is used as the CONFIG number.

Any values in this table will override the default values built into the nastran command.

An alternative solution to creating this file is to set the “config” keyword (page 297) in the node RC file; see “Customizing Command Initialization and Runtime Configuration Files” on page 68. Note, however, this will not set a model name.
Generating a Timing Block for a New Computer

MD Nastran uses timing constants to determine the fastest algorithm or “method” to perform certain numerically intensive operations. Timing constants are installed by MSC.Software for a variety of computers. If constants are not installed for your particular computer, MD Nastran will select default timing constants and display the following warning message:

```plaintext
*** USER WARNING MESSAGE 6080 (TMALOC)
THE TIMING CONSTANTS DATA BLOCK TIMEBLK NOT FOUND ON THE DELIVERY DATABASE FOR:
MACHINE = 5  CONFIG = 56  OPERASYS = 3  OPERALEV = 7  SUBMODEL = 1
LOADING DEFAULT TIMING CONSTANTS DATA BLOCK FOR:
MACHINE = 5  CONFIG = 56  OPERASYS = 3  OPERALEV = 5  SUBMODEL = 1
MODULE TIMING ESTIMATES INACCURATE AND MAY CAUSE INEFFICIENT JOB EXECUTION
```

Ignoring the message may result in excessive runtimes. Proper timing constants for a specific computer may be generated and installed by running a job that measures the timing constants of the computer and stores them in the delivery database.

Use the following steps to add timing constants for your computer to the delivery database:

1. Determine the MD architecture name of your system by consulting Table 3-1 or executing the command

   ```plaintext
   md20071 nastran system
   ```

2. Change the working directory to the architecture directory of your computer.

   ```plaintext
   cd install_dir/md20071/arch
   ```
   on UNIX, or

   ```plaintext
   cd install_dir\md20071\arch
   ```
   on Windows, where `arch` was determined in Step 1 above.

3. Copy the Structured Solution Sequence files to be modified by the gentim2 run with the commands:

   ```plaintext
   cp SSS.MASTERA gentim2.MASTERA
   cp SSS.MSCSOU gentim2.MSCSOU
   cp SSS.MSCOBJ gentim2.MSCOBJ
   ```
CHAPTER 3
Configuring MD Nastran

on UNIX, or

copy SSS.MASTERA gentim2.MASTERA
copy SSS.MSCSOU gentim2.MSCSOU
copy SSS.MSCOBJ gentim2.MSCOBJ

on Windows.

4. Issue the command

md20071 nastran DELDIR:gentim2 old=yes scratch=no batch=no

on UNIX, or

md20071 nastran DELDIR:gentim2 old=yes scratch=no

on Windows.

This command runs the job “DELDIR:gentim2.dat”, where “DELDIR” is a pre-defined logical symbol pointing to the directory containing the solution sequence source files. The value of the Bulk Data parameter “PARAM” is set to 7 by default, as shown in the partial listing of gentim2.dat below

```
NASTRAN MESH SYSTEM(124)=-1
PROJ LTC LOAD TIMING CONSTANTS
INIT MASTER,LOGICAL=(MASTERA(5000))
INIT SCRATCH(NOMEM)
TIME 2000
SOL GENTIMS
CEND
BEGIN BULK
PARAM, PARAM, 7
.
.
```

In general, the larger the value of “PARAM”, the longer the gentim2 job runs and the more accurate the timing results. If gentim2 runs for more than one hour, you may choose to reduce the value of “PARAM”. This will shorten the elapsed time of the gentim2 job.
5. If there are no errors, replace the old DBsets with the new DBsets created by the gentim2 run. Do this with the following commands:

```sh
mv gentim2.MASTERA SSS.MASTERA
mv gentim2.MSCOBJ SSS.MSCOBJ
mv gentim2.MSCSOU SSS.MSCSOU
```

on UNIX, or

```sh
copy gentim2.MASTERA SSS.MASTERA
copy gentim2.MSCOBJ SSS.MSCOBJ
copy gentim2.MSCSOU SSS.MSCSOU
```

on Windows.
Customizing Queue Commands (UNIX)

The nastran command runs an MD Nastran job by validating the command line and RC files, generating a “job script” that will run the MD Nastran executable, and running that script. When the “queue” keyword is specified, the corresponding “submit” keyword defines the command used to run the job script. The “submit” keyword (p. 338), only specified in RC files, consists of a list of queue names followed by the command definition for the queues as shown below:

```
submit=queue_list=command_definition
```

or

```
submit=command_definition
```

When specified, the `queue_list` contains one or more “queue” names separated by commas. If a queue list is not supplied (as shown in the second example), the `command_definition` applies to all queues.

The `command_definition` of the “submit” keyword value defines the command used to run a job when a “queue” keyword is specified that matches a queue name in a submit keyword’s `queue_list`. The `command_definition` can contain keyword names enclosed in percent “%” signs that are replaced with the value of the keyword before the command is run.

**Note:**

1. When defining queue commands, it may be useful to build the job script but not actually execute it. Use the “-n” option, for example

   ```
   md20071 -n nastran myjob queue=myqueue
   ```

2. The examples presented below are only intended to illustrate the “submit”, “qopt” and “queue” keywords. The examples may not work with your queuing software.

3. The Korn shell must be used to run the script generated by the nastran command.

Consider the following example:

```
submit=small,medium,large=qsub -q %queue% -x -eo -s /bin/ksh %job%
```

In this example, the “qsub” command is used to run a job when “queue=small”, “queue=medium”, or “queue=large” is specified.

Any keyword used by the nastran command may be specified in the “submit” keyword’s command definition. The most common keywords used in the command definition are:
Using the previous example, the command

```
md20071 nastran example queue=small
```

runs the job script using the command:

```
qsub -q small -x -eo -s /bin/ksh example.J12345
```

The \%queue\% keyword reference is replaced by the specified queue, and the \%job\% keyword reference is replaced by the name of the execution script.

Keyword references can also contain conditional text that is included only if the value of the keyword is not null, or matches (does not match) a regular expression. A complete description of the keyword reference syntax is described in "Keyword Reference Examples" on page 89. To check for a nonnull value, use the form

```
\%kwd:condtext\%
```

where \kwd\ is the name of the keyword and \condtext\ is the conditional text to be included. If the value of the keyword is null, the keyword reference is removed from the command. If the value of the keyword is not null, the keyword reference is replaced with the contents of \condtext\. Within \condtext\, the value of the keyword is represented by an open-close brace pair “\{\}".
For example:

```
submit=s=qsub -q %queue% %after:-a {}% -x -s /bin/ksh %job%
```

In this example, the “aft” keyword is references with conditional text. Using this example, the command

```
md20071 nastran example queue=s after=10:00
```

runs the job script using the following qsub command:

```
qsub -q s -a 10:00 -x -s /bin/ksh example.J12345
```

Using the same “submit” keyword, the command

```
md20071 nastran example queue=s
```

runs the job script using the following command:

```
qsub -q s -x -s /bin/ksh example.J12345
```

In this case, the “after” keyword was not specified and the entire contents of the %after% keyword reference was removed from the qsub command line.

**Special Queues**

When the “queue” keyword is not specified, the following three special queues are used:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Queue Name</th>
<th>Command Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>after</td>
<td>-aft</td>
<td>%at %after%</td>
</tr>
<tr>
<td>batch=yes</td>
<td>-bg</td>
<td>%nice=%^nice %j.nice:[] %job%</td>
</tr>
<tr>
<td>batch=no</td>
<td>-fg</td>
<td>%j.nice:[] %job%</td>
</tr>
</tbody>
</table>
Changing the command definitions of these queues (using the “submit” keyword) will change the way the nastran command runs a job under the “after” and “batch” keywords.

**Note:**

1. If the first character of the command is the UNIX pipe character, “|”, the contents of job script will be piped into the command.
2. The command for the “-bg” queue is always executed in the background; the “-fg” and “-aft” commands are always executed in the foreground.
CHAPTER 3
Configuring MD Nastran

87

Customizing the Templates

The nastran command relies on several templates to construct the job script (UNIX) or control file
(Windows) that is built for every MD Nastran job. Note that, for UNIX, the job script includes the
necessary commands to build the control file. Several templates are provided:

For UNIX, the following files are used. Note that the installed template files are the same for all
architectures. The file names in the arch directory are linked to files in the bin directory.

- \( \text{install_dir/md20071/arch/nastran.dmp} \) is used for DMP jobs.
  At installation time, this name is linked to \( \text{install_dir/bin/nast2007.dmp} \).
  The keyword defining this file name is 0.dmp.
- \( \text{install_dir/md20071/arch/nastran.lcl} \) is used for serial or SMP jobs run on the local system.
  At installation time, this name is linked to \( \text{install_dir/bin/nast2007.lcl} \).
  The keyword defining this file name is 0.lcl.
- \( \text{install_dir/md20071/arch/nastran.rmt} \) is used for serial or SMP jobs run on a remote system
  using the "node" keyword. At installation time, this name is linked to \( \text{install_dir/bin/nast2007.rmt} \).
  The keyword defining this file name is 0.rmt.
- \( \text{install_dir/md20071/arch/nastran.srv} \) is used for Toolkit jobs.
  At installation time, this name is linked to \( \text{install_dir/bin/nast2007.srv} \).
  The keyword defining this file name is 0.srv.

The templates provided by MSC support all versions of MD Nastran since MSC.Nastran 68.0 for all
UNIX platforms.

For Windows, two file names are listed for each template. The file used is the first one found.

- \( \text{install_dir/md20071/i386/nastran.lcl or install_dir/bin/nast2007.lcl} \) is used for serial or SMP
  jobs run on the local system.
  The keyword defining this file name is 0.lcl.
- \( \text{install_dir/md20071/i386/nastran.rmt or install_dir/bin/nast2007.rmt} \) is used for serial or SMP
  jobs run on a remote system using the "node" keyword. Currently, the remote system must be a
  UNIX system running the "rshd" daemon.
  The keyword defining this file name is 0.rmt.
- \( \text{install_dir/md20071/i386/nastran.srv or install_dir/bin/nast2007.srv} \) is used for Toolkit jobs.
  The keyword defining this file name is 0.srv.

The templates provided by MSC support all versions of MSC.Nastran since Version 70.0 for Windows
platforms.

These templates may be modified to suit your needs. For UNIX, if you modify these files, you may either
replace the link in the arch directory with your changes if your changes only affect a single architecture
or you may change the file in the bin directory if your changes are valid for all architectures. For
Windows, if you modify these files, make sure your changes are used by putting them in the i386
directory or by modifying or replacing the appropriate file in the bin directory. Alternatively, you may
use the appropriate keyword, specified either in the INI file or on the command line, to specify the location of your modified template file.

**Note:** When customizing the templates, it may be useful to build the job script or control file but not actually execute it. Use the “-n” option, e.g.,

```
md20071 nastran -n myjob
```

The name of the generated file will be echoed to stdout.

## Keyword Reference Syntax

The script templates use the keyword reference syntax that was partially introduced in the previous section. Table 3-5 provides examples.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Value</th>
<th>Side effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>%keyword%</td>
<td>Value of keyword.</td>
<td></td>
</tr>
<tr>
<td>%keyword:condtext%</td>
<td>condtext</td>
<td></td>
</tr>
<tr>
<td>%keyword=re%</td>
<td>Value of the parenthetic expression if specified in the re, otherwise the string matched by the re.</td>
<td></td>
</tr>
<tr>
<td>%keyword=re:condtext%</td>
<td>condtext if re is matched.</td>
<td></td>
</tr>
<tr>
<td>%keyword!:re:condtext%</td>
<td>condtext if re is not matched.</td>
<td></td>
</tr>
<tr>
<td>%keyword;%</td>
<td>Kill remainder of line if keyword has null value. In a case construct, the default case.</td>
<td></td>
</tr>
<tr>
<td>%keyword=re:%</td>
<td>Kill remainder of line if re does not match.</td>
<td></td>
</tr>
<tr>
<td>%keyword!:re:%</td>
<td>Kill remainder of line if re does match.</td>
<td></td>
</tr>
<tr>
<td>%keyword?:%</td>
<td>Start of case construct. See “Using Regular Expressions” on page 92.</td>
<td></td>
</tr>
<tr>
<td>%keyword&gt;cmp:context%</td>
<td>condtext if keyword is &gt; than cmp</td>
<td></td>
</tr>
</tbody>
</table>
Table 3-5  Keyword Syntax (continued)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Value</th>
<th>Side effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>%keyword&gt;=cmp:condtext%</td>
<td>condtext if keyword is ≥ than cmp</td>
<td></td>
</tr>
<tr>
<td>%keyword&lt;cmp:condtext%</td>
<td>condtext if keyword is &lt; than cmp</td>
<td></td>
</tr>
<tr>
<td>%keyword=&lt;=cmp:condtext%</td>
<td>condtext if keyword is ≤ than cmp</td>
<td></td>
</tr>
<tr>
<td>%keyword&gt;cmp:%</td>
<td>Kill remainder of line if keyword is not &gt; than cmp</td>
<td></td>
</tr>
<tr>
<td>%keyword&gt;=cmp%</td>
<td>Kill remainder of line if keyword is not ≥ than cmp</td>
<td></td>
</tr>
<tr>
<td>%keyword&lt;cmp:%</td>
<td>Kill remainder of line if keyword is not &lt; than cmp</td>
<td></td>
</tr>
<tr>
<td>%keyword=&lt;=cmp%</td>
<td>Kill remainder of line if keyword is not ≤ than cmp</td>
<td></td>
</tr>
</tbody>
</table>

**Keyword Reference Examples**

The keyword reference syntax is described using the following examples from the UNIX "install_dir/bin/nast2007.lcl" file. The same syntax is supported for the Windows control file templates.

**Unconditional Keyword Substitution**

```
export MSC_BASE=%MSC_BASE%
```

The keyword reference `%MSC_BASE%` will be replaced by the value of the "MSC_BASE" keyword.

```
export DBSDIR=%dbs=(.*)/%
```

The keyword reference `%dbs=(.*)%/` will be replaced with the value of the parenthetic regular expression. For example, given the keyword value “onedir/anotherdir/myfile”, the parenthetic expression is “onedir/anotherdir”, and the substituted line would read:

```
export DBSDIR=onedir/anotherdir
```
Conditional Keyword Substitution

%sysfield:SYSFIELD={}%

The keyword reference %sysfield:SYSFIELD={}% will be replaced by the string “SYSFIELD=keyword-value” if and only if the keyword is not null.

%dcmd=dbx:run%

The keyword reference %dcmd=dbx:run% will be replaced by “run” if and only if “dcmd=dbx” was specified. If the equal sign in the keyword reference was replaced by an exclamation mark, i.e., %dcmd!dbx:run%, then the keyword reference will be replaced by “run” if and only if “dcmd” was set to a nonnull value not equal to “dbx”.

Conditional Inclusion

%MSC_ARCH=aix:%startdate=date +%a %h %d %H:%M:%S %Z %Y
%MSC_ARCH!aix:%startdate=date

Conditional inclusion is indicated by a null conditional text string; i.e., the colon is immediately followed by a percent sign. This capability is generally used with a regular expression to include the remainder of the line if a keyword value matches or does not match a regular expression. In the first line, the remainder of the line will be included if the “MSC_ARCH” keyword contains the string “aix” while the remainder of the second line will be included if “MSC_ARCH” does not contain the string “aix”. More than one conditional inclusion keyword reference can be used on a line to create more complex tests.

%prt=y:%pdel=y:%/bin/rm %out%.f04 %out%.f06 %out%.log

The “rm” command will included if and only if “prt=yes” and “pdel=yes”.

A “case” structure is specified as follows:

...%s.model?:%
...%s.model=IP.8:% SGI_ISA=mips1; export SGI_ISA
...%s.model=IP12:% SGI_ISA=mips1; export SGI_ISA
...%s.model=IP15:% SGI_ISA=mips1; export SGI_ISA
...%s.model=:% SGI_ISA=mips2; export SGI_ISA
This sequence will result in the line

SGI_ISA=mips1

if “s.model” is “IP” followed by a single character (using the second line), or “IP12” (using the third line), or “IP15” (using the fourth line), otherwise

SGI_ISA=mips2

will be generated using the last line. Case constructs can be nested, but a keyword may only be active in one case at a time.

Greater and less-than comparisons can be used instead of regular expression matching to control conditional inclusion. These comparisons are done with integer, floating, or string values based on the types of the two values.

%a.release>68: %CONFIG=%config%

The CONFIG statement will be included if “a.release” is greater than 68.

**Nested Keyword Values**

One level of nested keywords may occur anywhere within the %.*% string. Only unconditional keywords substitutions are supported for nested keywords. Nested keywords are specified as \%keyword\%.

%dmparallel>\%maxnode\%:@@ node = %maxnode%

This sequence will cause the “@@ node ..” text to be included if the value of the “dmparallel” keyword is greater than the value of the “maxnode” keyword.
Using Regular Expressions

The regular expression syntax supported by the nastran command is compatible with the standard ed(1) regular expression syntax with the exception that only one parenthetic expression is permitted. The syntax follows.

One-character Regular Expressions

- Any character, except for the special characters listed below, is a one-character regular expression that matches itself.
- A backslash, \\, followed by any special character is a one-character regular expression that matches the special character itself. The special characters are: period, ".", asterisk, "*", and backslash "\\", which are always special except when they appear within brackets; circumflex, "^", which is special at the beginning of a regular expression or when it immediately follows the left bracket of a bracketed expression; and dollar sign "$", which is special at the end of a regular expression.
- A period, ".", is a one-character regular expression that matches any character.
- A nonempty string of characters enclosed within brackets, "[" and "]", is a one-character regular expression that matches one character in that string. If, however, the first character of the string is a circumflex, "^", the one-character regular expression matches any character except the characters in the string. The circumflex has this special meaning only if it occurs first in the string. The dash, "-", may be used to indicate a range of consecutive characters. The dash loses this special meaning if it occurs first (after an initial circumflex, if any) or last in the string. The right square bracket, "]", does not terminate such a string when it is the first character within it (after an initial circumflex, if any).

Regular Expressions

- A one-character regular expression is a regular expression that matches whatever the one-character regular expression matches.
- A one-character regular expression followed by an asterisk, "*", is a regular expression that matches zero or more occurrences of the one-character regular expression. If there is any choice, the longest leftmost string that permits a match is chosen.
- A one-character regular expression followed by "\{m\", "\{m\", or "\{m,n\}" is a regular expression that matches a ranges of occurrences of the one-character regular expression. The values of m and n must satisfy 0 ≤ m ≤ n ≤ 254; "\{m\}" exactly matches m occurrences; "\{m\}" matches at least m occurrences; "\{m,n\}" matches any number of occurrences between m and n inclusive.
- A concatenation of regular expressions is a regular expression that matches the concatenation of the strings matched by each component of the regular expression.
- A regular expression enclosed between the character sequences "\(" and "\)" defines a parenthetic expression that matches whatever the unadorned regular expression matches. Only one parenthetic expression may be specified.
The expression "\1" matches the same string of characters as was matched by the parenthetic expression earlier in the regular expression.

**Constraining Regular Expressions**

- A circumflex, "^", at the beginning of an entire regular expression constrains the regular expression to match an initial segment of a string.
- A dollar sign, "$", at the end of an entire regular expression constrains the regular expression to match a final segment of a string.
- The construction "^re$" constrains the regular expression to match the entire string.
- The construction "^$" matches a null string.
Using the Basic Functions of MD Nastran

- Overview 96
- Using the nastran Command 97
- Using the Basic Keywords 102
- Determining Resource Requirements 107
- Using the Test Problem Libraries 109
- Making File Assignments 110
- Using Databases 113
- Using the INCLUDE Statement 120
- Using the SSS Alter Library 124
- Resolving Abnormal Terminations 125
Overview

This chapter is directed to the engineer running MD Nastran, and discusses how the basic functions of MD Nastran are used. It covers using the nastran command, including file types, filenames, logical symbols, the help facility, and other functions. In addition, this chapter provides an overview of the basic keywords, outlines resource requirements, describes how to specify memory sizes, introduces the sample problem libraries, and how to make file assignments, as well as how to use databases, how to apply the INCLUDE statement, and how to resolve abnormal terminations.
Using the nastran Command

MD Nastran jobs are run using the nastran command. The basic format of this command is

```
md20071 nastran input_data_file keywords
```

where `input_data_file` is the name of the file containing the input data and `keywords` is zero or more optional keyword assignments. For example, to run an MD Nastran job using the data file example.dat, enter the following command:

```
md20071 nastran example
```

Various options to the nastran command are available using keywords described in “Keywords” on page 292. Keyword assignments consist of a keyword, followed by an equal sign, followed by the keyword value, for example:

```
md20071 nastran example scratch=yes
```

**Note:** In Windows you can use a hash mark “#” instead of the equal sign. This is useful if the nastran command is being placed in a “.bat” file.

```
md20071 nastran example scratch#yes
```

Keyword assignments can be specified on the command line or included in RC files.

There are two RC files controlled by you:

- The user RC file is used to define parameters applicable to all MD Nastran jobs you run.

  UNIX: $HOME/nast2007rc

  Windows: %HOMEDRIVE%\%HOME%\nast2007.rcf

- The local RC file should be used to define parameters applicable to all MD Nastran jobs that reside in the input data file’s directory, and is located in the same directory as the input data file. If the “rcf” keyword is used, this local RC file is ignored.

  UNIX: .nast2007rc

  Windows: nast2007.rcf
File Types and Versioning

MD Nastran’s default input and output files use the following types:

<table>
<thead>
<tr>
<th>Type</th>
<th>Type of File</th>
<th>Description of File</th>
</tr>
</thead>
<tbody>
<tr>
<td>.dat</td>
<td>Input</td>
<td>Input Data File</td>
</tr>
<tr>
<td>.f04</td>
<td>Output</td>
<td>Execution Summary File</td>
</tr>
<tr>
<td>.f06</td>
<td>Output</td>
<td>Output Data File</td>
</tr>
<tr>
<td>.log</td>
<td>Output</td>
<td>Job Log File</td>
</tr>
<tr>
<td>.op2</td>
<td>Input Output</td>
<td>OUTPUT2 File</td>
</tr>
<tr>
<td>.pch</td>
<td>Output</td>
<td>Punch File</td>
</tr>
<tr>
<td>.plt</td>
<td>Output</td>
<td>Binary Plot File</td>
</tr>
<tr>
<td>.xdb</td>
<td>Output</td>
<td>Results Database</td>
</tr>
</tbody>
</table>

Note:
1. If the input file is specified as “example” and the files “example.dat” and “example” both exist, the file “example.dat” will be chosen. In fact, it is impossible to use a file named “example” as the input data file if a file named “example.dat” exists.
2. The “jidtype” keyword may be used to specify an alternate default suffix for the input data file. For example, “jidtype=bdf” will change the default file type to “.bdf”.
3. The XDB file is not versioned.
4. The “oldtypes” keyword may be used to specify a list of additional file types that are versioned. For example, “oldtypes=xdb” will cause the XDB file to be versioned.
When a job is run more than once from the same directory, the previous output files are versioned, or given indices. The indices are integers appended to the filename; the same integer will designate files for the same job. For example,

v2401.f04  v2401.f04.1  v2401.f04.2  v2401.f04.3
v2401.f06  v2401.f06.1  v2401.f06.2  v2401.f06.3

The files listed (according to time of execution from oldest to newest) are:

v2401.f04.1  v2401.f06.1
v2401.f04.2  v2401.f06.2
v2401.f04.3  v2401.f06.3
v2401.f04  v2401.f06

**Using Filenames and Logical Symbols**

Several of the parameters used by MD Nastran, including command line arguments, initialization and RC file commands, and statements within MD Nastran input files, specify filenames. The filenames must follow your system’s standard filename conventions, with the addition that filenames can include a “logical symbol” component, i.e., the filename can be specified in either of the following forms:

```
filename
logical-symbol:filename
```

Logical symbols provide you with a way of specifying file locations with a convenient shorthand. This feature also allows input files containing filename specifications to be moved between computers without requiring modifications to the input files. Only the logical symbol definitions that specify actual file locations need to be modified.

Only one logical symbol name may be used in a filename specification. This logical symbol must be the initial component of the filename string, and it must be separated from the filename by a colon “:”. If the symbol has a non-null value, the actual filename is created by replacing the symbol name with its value and replacing the colon with a slash; otherwise, both the symbol name and the colon are left as is.

**Note:**
1. A logical symbol can be defined using any environment variable or previously defined symbol. Use the standard environment variable reference convention, i.e., “$name” or “${name}” on UNIX and “%name%” on Windows.
2. Logical symbols must be more than one character long, i.e., the filename reference “D:\temp\myfile.dat” will be interpreted on Windows as a drive reference followed by a pathname.
3. MD Nastran will accept Windows pathnames using the slash “/” character as a replacement for the backslash “\”.

For example, assume that your home RC file contains the line

SYMBOL=DATADIR=/dbs/data

on UNIX, or

SYMBOL=DATADIR=d:\dbs\data

on Windows, and a job is submitted with the command

md20071 DATADIR:nastran example

Since MD Nastran automatically sets the OUTDIR environment variable to the value of the “out” keyword, the filenames

'DATADIR:myfile.dat'
'OUTDIR:testdata.info'

will reference the files

/dbs/data/myfile.dat
./testdata.info

on UNIX and

d:\dbs\data\myfile.dat
\testdata.info

on Windows respectively, see “symbol” on page 340 for more information.

Several other symbols are automatically created by the nastran command. These include DELDIR, DEMODIR, TPLDIR, and SSSALTERDIR to access the delivery database source directory, and DEMO, TPL, and SSSALTER libraries, respectively.
Using the Help Facility and Other Special Functions

Several special functions are supported by reserved input data filenames. If these names are specified as the input data file, the nastran command will execute the special function and exit.

**Note:** If you need to use one of these reserved names as an actual input filename, you must either prefix the filename with a path or append a file type to the filename.

The special functions are invoked as follows:

**md20071 nastran help**

This request will display the basic help output. Additional help capabilities are described in the basic help output.

**md20071 nastran help keyword1 [keyword2 ...]**

This request will display help for the keywords listed on the command line.

**md20071 nastran limits**

This request will display the current UNIX resource limits.

**md20071 nastran news**

This request will display the news file.

**md20071 nastran system**

This request will display system information about the current computer.

On UNIX, these requests can be executed on a remote computer that has MD Nastran installed by also specifying the keyword “node=nodename”, for example:

**md20071 nastran system node=thatnode**
Using the Basic Keywords

The following table is a partial list of the basic keywords that may be used on the command line or placed into RC files as appropriate. More advanced keywords are listed in “Using the Advanced Keywords” on page 133, and a complete list of all keywords and their syntax is listed in “Keywords” on page 292.

All Systems

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>append</td>
<td>Combines the .f06, .f04, and .log files into a single file after the job completes.</td>
</tr>
<tr>
<td>dbs</td>
<td>Specifies an alternate name for user database files.</td>
</tr>
<tr>
<td>memory</td>
<td>Specifies the amount of memory to be used by the job.</td>
</tr>
<tr>
<td>old</td>
<td>Renames existing output files with version numbers or deletes existing output files.</td>
</tr>
<tr>
<td>out</td>
<td>Specifies an alternate name for output files.</td>
</tr>
<tr>
<td>rcf</td>
<td>Specifies an alternate name of the local RC file.</td>
</tr>
<tr>
<td>scratch</td>
<td>Indicates databases are to be deleted when job completes.</td>
</tr>
<tr>
<td>sdirectory</td>
<td>Specifies an alternate scratch file directory.</td>
</tr>
<tr>
<td>symbol</td>
<td>Defines a symbolic name and value.</td>
</tr>
</tbody>
</table>

UNIX Systems

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>after</td>
<td>Holds the job until the specified time.</td>
</tr>
<tr>
<td>batch</td>
<td>Runs the job in background or foreground.</td>
</tr>
<tr>
<td>xmonast</td>
<td>Automatically runs the Motif-based output file monitor.</td>
</tr>
</tbody>
</table>

Queuing (UNIX)

Note: These capabilities depend upon the queue submission commands defined by the “submit” keyword and your queuing system. The keywords may not work on your system.
<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>cputime</td>
<td>Specifies maximum CPU time to be allowed.</td>
</tr>
<tr>
<td>queue</td>
<td>Specifies name of queue where the job will be submitted to.</td>
</tr>
</tbody>
</table>
Specifying Memory Sizes

Several nastran keywords specify memory sizes. In all cases, the value can be specified either as the number of words (64-bit words on i8/iLP64 platforms when mode = i8, and 32-bit words on all others) or as a number followed by one of the following modifiers:

Table 4-1  Memory Size Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Size (Words)</th>
</tr>
</thead>
<tbody>
<tr>
<td>nT, nTW</td>
<td>( n \times (1024^4) )</td>
</tr>
<tr>
<td>nTB</td>
<td>( n \times \frac{(1024^4)}{bpw} )</td>
</tr>
<tr>
<td>nG, nGW</td>
<td>( n \times (1024^3) )</td>
</tr>
<tr>
<td>nGB</td>
<td>( n \times \frac{(1024^3)}{bpw} )</td>
</tr>
<tr>
<td>nM, nMW</td>
<td>( n \times (1024^2) )</td>
</tr>
<tr>
<td>nMB</td>
<td>( n \times \frac{(1024^2)}{bpw} )</td>
</tr>
<tr>
<td>nK, nKW</td>
<td>( n \times 1024 )</td>
</tr>
<tr>
<td>nKB</td>
<td>( n \times \frac{1024}{bpw} )</td>
</tr>
<tr>
<td>n<em>physical, n</em>physical</td>
<td>( n \times \text{memory}_\text{physical} )</td>
</tr>
<tr>
<td>n<em>virtual, n</em>virtual</td>
<td>( n \times \text{memory}_\text{virtual} )</td>
</tr>
</tbody>
</table>

where: \( bpw = 8 \) on i8/iLP64 supported platforms when mode = i8, and \( bpw = 4 \) on all other platforms; “physical” is the computer’s physical memory, i.e., the “RAM”; and “virtual” is the swap size on UNIX systems, and the maximum paging file size on Windows systems.

Note: In order to use the “physical” and “virtual” specifications, the computer’s physical memory and swap file size must be known to the nastran command. The nastran command always knows both these sizes on Windows systems. On UNIX systems, the physical memory is known on HP Alpha Tru64, Linux, Solaris, and SUPER-UX. The computer’s physical and virtual memory sizes can also be set via the “s.pmem” and “s.vmem” keywords respectively.
Examples are

```
md20071 nastran memory=1gb
```

Set the memory request to one gigabyte, 1024 megabytes, 1048576 kilobytes, 1073741824 bytes, 134217728 words for i8/iLP64 or 268435436 words on all other systems.

```
md20071 nastran memory=0.5xPhys
```

Set the memory request to 50% of the computer’s physical memory.

**Maximum Memory Size**

Table 4-2 lists the maximum “memory” size for MD Nastran platforms. A “memory” request larger than this value results in an error as the job starts.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>8 GB</td>
</tr>
<tr>
<td>HP UNIX</td>
<td>8 GB</td>
</tr>
<tr>
<td>HP UX-11</td>
<td>8 GB</td>
</tr>
<tr>
<td>Linux</td>
<td>2 GB</td>
</tr>
<tr>
<td>Linux64</td>
<td>8 GB</td>
</tr>
<tr>
<td>Linux8664</td>
<td>8 GB</td>
</tr>
<tr>
<td>IRIX64</td>
<td>8 GB</td>
</tr>
<tr>
<td>Solaris</td>
<td>8 GB</td>
</tr>
</tbody>
</table>

**Note:** The actual maximum value you can specify depends on several factors, including the physical memory systems and the swap file size on UNIX systems, the paging file size on Windows systems, and your virtual memory limit on most UNIX systems. You must also deduct from the maximum value the size of the executable, listed in “System Descriptions” in Appendix C, and space required for the various operating system and Fortran runtime libraries. Jobs submitted with mode=i8 on platforms that support it, have unlimited memory.
<table>
<thead>
<tr>
<th>Platform</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUPER-UX</td>
<td>8 GB</td>
</tr>
<tr>
<td>UXP/V</td>
<td>Lesser of “real” memory or 2 GB</td>
</tr>
<tr>
<td>All others</td>
<td>2 GB</td>
</tr>
</tbody>
</table>

**Note:** When running with mode = i8 or ILP64 on platforms that support it, the maximum memory is limited by system limits and virtual address space.
Determining Resource Requirements

For most models of moderate size (up to 5000 grid points for static analysis), you need not be concerned with resource requirements since the default MD Nastran parameters allocate sufficient resources. The analysis of larger models may require you to check the resource requirements and the various options that are available to manage memory and disk resources.

There are several tools available to assist you in determining the resource requirements of your job. Table 4-3 and Table 4-4 are the simplest tools, they present gross estimates of the memory and total disk space requirements of static analyses using default parameters with normal output requests. Other solution sequences will generally have greater requirements.

Table 4-3  Estimated Memory Requirements of Static Analyses

<table>
<thead>
<tr>
<th>Degrees of Freedom</th>
<th>Memory Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>$DOF &lt; 10000$</td>
<td>3 MW</td>
</tr>
<tr>
<td>$10000 &lt; DOF \leq 50000$</td>
<td>5 MW</td>
</tr>
<tr>
<td>$50000 &lt; DOF \leq 100000$</td>
<td>10 MW</td>
</tr>
<tr>
<td>$100000 &lt; DOF \leq 200000$</td>
<td>22 MW</td>
</tr>
</tbody>
</table>

Table 4-4  Estimated Total Disk Requirements of Static Analyses

<table>
<thead>
<tr>
<th>Degrees of Freedom</th>
<th>Total Disk Space Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>$DOF &lt; 10000$</td>
<td>90 MB</td>
</tr>
<tr>
<td>$10000 &lt; DOF \leq 50000$</td>
<td>500 MB</td>
</tr>
<tr>
<td>$50000 &lt; DOF \leq 100000$</td>
<td>1000 MB</td>
</tr>
<tr>
<td>$100000 &lt; DOF \leq 200000$</td>
<td>2000 MB</td>
</tr>
</tbody>
</table>

More detailed resource estimates can be obtained from the ESTIMATE program, described in “ESTIMATE” on page 215. ESTIMATE reads the input data file and calculates the job’s memory and disk requirements. The ESTIMATE program is most accurate in predicting the requirements of static analyses that don’t have excessive output requests. The memory requirements for normal modes analyses using the Lanczos Method are reasonably accurate; however, the disk requirements are dependent upon the number of modes. This is a value that ESTIMATE does not know. Memory and disk requirements for other solutions are less accurate.

The best estimates of the memory requirements for a job are available in User Information Message 4157, described in “User Information Messages 4157 and 6439” on page 151, but this requires an MD Nastran run.
Estimating BUFFSIZE

presents recommendations for BUFFSIZE based on model size. These values have been chosen to represent the best compromise between database access speed and storage requirements for typical problems. An excessively large BUFFSIZE can result in more I/O data transferred and wasted space in the database for smaller problems; an excessively small BUFFSIZE can result in increases I/O counts for larger problems. You may be able to achieve higher performance or smaller databases using other values.

Table 4-5  Suggested BUFFSIZE Values

<table>
<thead>
<tr>
<th>Degrees of Freedom</th>
<th>BUFFSIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$DOF \leq 100000$</td>
<td>8193</td>
</tr>
<tr>
<td>$100000 &lt; DOF \leq 200000$</td>
<td>16385</td>
</tr>
<tr>
<td>$DOF &gt; 400000$</td>
<td>32769</td>
</tr>
</tbody>
</table>

Note: The actual I/O transfer size is $(BUFFSIZE - 1) \times bpw$ where $bpw$ is 8 on i8/iLP64 and 4 on all other systems.
Using the Test Problem Libraries

Three libraries of test problems are delivered with MD Nastran.

- The demonstration problem library (DEMO) contains a selection of MD Nastran input files that are documented in the *MSC Nastran Demonstration Problem Manual*. These files are accessible via the DEMODIR symbol, or via the path `install_dir/md20071/nast/demo` on UNIX and `install_dir/md20071/nast/demo` on Windows.

- The test problem library (TPL) contains a general selection of MD Nastran input files showing examples of most of the MD Nastran capabilities. In general, these files are not documented. The files are accessible via the TPLDIR symbol, or via the path `install_dir/md20071/nast/tpl` on UNIX, and `install_dir/md20071/nast/tpl` on Windows.

The DEMO and TPL libraries contain “demoidx.dat” and “tplidx.dat” respectively. These files contain one-line descriptions of the library members. Also included are files named “tplexec” and “demoexec”, which are scripts used to run the problems on UNIX, or “tplexec.bat” and “demoexec.bat”, which are batch files used to run the problems on Windows.

If you only want to run a job from the DEMO or TPL libraries, the easiest method is to use either the “DEMODIR” or “TPLDIR” symbols, running the command from any convenient directory. For example,

```
md20071 nastran DEMODIR:d10101d
```

If you want to experiment with the file, copy the file to your own directory and then execute the problem. Note that several of the library files have “INCLUDE” files that should also be copied if they too will be modified, or they can be referenced as-is via the standard INCLUDE file processing; see “Using the INCLUDE Statement” on page 120.

Some example problems contain references to files that are qualified with the following logical symbols:

- TPLDIR
- DEMODIR
- DBSDIR
- OUTDIR

Unless they already exist in your environment as environment variables, the logical symbols DEMODIR and TPLDIR automatically point to the DEMO and TPL libraries respectively. DBSDIR and OUTDIR are always based on the “dbs” and “out” keywords respectively.
Making File Assignments

Using the ASSIGN statement, you can assign physical files used by MD Nastran to FORTRAN units or DBset files or you can modify the properties of existing or default file assignments. The ASSIGN statement is documented in the “File Management Statements” in Chapter 2 of the MD Nastran Quick Reference Guide.

ASSIGN Statement for FORTRAN Files

For FORTRAN files, the format of the ASSIGN statement is

```
ASSIGN logical-key=[{filename}]*] [UNIT=u] [[{STATUS}]=NEW|OLD|UNKNOWN]]
[[{FORM}]=FORMATTED|UNFORMATTED|LITTLEENDIAN|LITTLEENDIAN [ORDER=]<ostype>]] [DEFER]
[[{TEMP}]+] [DElete] [SYS='sys-spec']
```

Currently, there are no values of the SYS field defined for FORTRAN files on any system. For a list of the FORTRAN files and their default attributes, please refer to Table 2-1 in the “File Management Statements” in Chapter 2 of the MD Nastran Quick Reference Guide. For more information about byte-ordering within binary files (the “endian” of a file), please refer to “Binary File Byte Ordering (Endian)” in Appendix C.

ASSIGN Statement for DBsets

```
ASSIGN logical-name=[{filename}]*] [TEMP] [DELETE] [SYS='sys-spec']
```

See “Using the SYS Field” on page 140 for details on the SYS field for DBsets.

Scratch DB Set Names

The default base name for scratch DB Sets uses the base name of the input data file as a prefix; this will permit you to more easily identify the job that created specific files in the scratch directory.

UNIX:           md20071 nastran example sdir=/tmp
Windows:        md20071 nastran example sdir=c:\temp

The SCRATCH DBset names will be named “/tmp/example.T<unique>*” on the UNIX systems and “c:\temp\example.T<unique>*” on Windows systems where “<unique>” is a string created from the process ID of the nastran command and the current time.
The following tables give information about the DBALL and SCRATCH DBset default allocations.

<table>
<thead>
<tr>
<th>DBset</th>
<th>Memory</th>
<th>Type</th>
<th>Size</th>
<th>Units</th>
<th>Assignable</th>
<th>Size</th>
<th>Logical Name</th>
<th>Physical Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASTER</td>
<td>RAM</td>
<td>120000</td>
<td>Words</td>
<td></td>
<td>YES</td>
<td>8193</td>
<td>MASTER</td>
<td>dbs.MASTER</td>
<td>5000</td>
</tr>
<tr>
<td>DBALL</td>
<td>N/A</td>
<td>-</td>
<td></td>
<td></td>
<td>YES</td>
<td>8193</td>
<td>DBALL</td>
<td>dbs.DBALL</td>
<td>See Table 4-7</td>
</tr>
<tr>
<td>OBJSCR</td>
<td>N/A</td>
<td>-</td>
<td></td>
<td></td>
<td>NO</td>
<td>8193</td>
<td>OBJSCR</td>
<td>sdir.OBJSCR</td>
<td>5000</td>
</tr>
<tr>
<td>SCRATCH</td>
<td>SMEM</td>
<td>See note</td>
<td>GINO Blocks</td>
<td>YES</td>
<td>8193</td>
<td>SCRATCH</td>
<td>sdir.SCRATCH</td>
<td>See Table 4-7</td>
<td></td>
</tr>
<tr>
<td>SCRATCH</td>
<td>N/A</td>
<td>-</td>
<td></td>
<td></td>
<td>YES</td>
<td>8193</td>
<td>SCR300</td>
<td>sdir.SCR300</td>
<td>See Table 4-7</td>
</tr>
<tr>
<td>User DBset</td>
<td>N/A</td>
<td>-</td>
<td></td>
<td></td>
<td>YES</td>
<td>8193</td>
<td>DBset</td>
<td>dbs.DBset</td>
<td>25000</td>
</tr>
</tbody>
</table>

**Note:** The default SMEM value is 0 SUPER-UX and is 100 for all other platforms.

where:

**DBset** The DBset name.

**Memory** The size of open core memory (in words) of the RAM of the MASTER DBset. The size may be modified using the FMS statement, INIT MASTER (RAM = value).

**BUFFSIZE** The buffer size (words) used for I/O transfer for each DBset. This size may be changed if “YES” is in the Assignable column.

**Logical Name** The logical name of the DBset. This name may be set with the ASSIGN or INIT statement.

**Physical Name** The name of the file as known to your operating system. This name may be changed by using the ASSIGN statement.

**Size** The default maximum file size (in GINO blocks) allowed for each DBset. This size may be changed by using the INIT statement.
Table 4-6

Default DBALL and SCRATCH DBset Sizes in GINO Blocks

<table>
<thead>
<tr>
<th>Memory (MEM)</th>
<th>BUFSIZE</th>
<th>Memory (MEM)</th>
<th>BUFSIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BUFSIZE &lt; 32769</td>
<td>32769 &lt; BUFSIZE &lt; 65537</td>
<td>BUFSIZE = 65537</td>
</tr>
<tr>
<td>MEM &lt; 32 MW</td>
<td>250,000</td>
<td>250,000</td>
<td>250,000</td>
</tr>
<tr>
<td>32 MW ≤ MEM &lt; 64 MW</td>
<td>500,000</td>
<td>1,000,000</td>
<td>1,000,000</td>
</tr>
<tr>
<td>MEM ≥ 64 MW</td>
<td>1,000,000</td>
<td>2,000,000</td>
<td>2,000,000</td>
</tr>
</tbody>
</table>

Note: These values will be reduced, if necessary and without any information messages, to the maximum file size supported by the filesystem on which the file was allocated. For example:

- For Windows 95/98/ME/NT/2000/XP using a FAT filesystem, the maximum file size is 2 GB.
- For Windows 98/ME/2000/XP using a FAT32 filesystem, the maximum file size is 4 GB.
- For AIX using a non-largefile enabled JFS filesystem, the maximum file size is 2 GB.
- For AIX using a largefile enabled JFS filesystem, the maximum file size is 63.88 GB.

Default DBALL and SCRATCH DBset Sizes in GB for Specific BUFSIZE Values

<table>
<thead>
<tr>
<th>Memory (MEM)</th>
<th>BUFSIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8193</td>
</tr>
<tr>
<td>MEM &lt; 32 MW</td>
<td>7.63 GB</td>
</tr>
<tr>
<td>32 MW ≤ MEM &lt; 64 MW</td>
<td>15.26 GB</td>
</tr>
<tr>
<td>MEM ≥ 64 MW</td>
<td>30.52 GB</td>
</tr>
</tbody>
</table>
Using Databases

MD Nastran provides a database for the storage and subsequent retrieval of matrices and tables. This facility consists of several database sets (DBsets) that conform to the following specifications:

- The MD Nastran limit on the maximum number of DBsets for an analysis is 200. Your computer may have a lower limit on the maximum number of open files that a process can open. This limit is displayed as the “Number of open files” by the “limits” special function. See “Using the Help Facility and Other Special Functions” on page 101.

- Each DBset may consist of 1 to 20 physical files. Again, this is subject to the maximum number of open files that your system permits.

- The maximum size of each DBset is machine dependent. There are several factors affecting the maximum size a given file can reach. Among these are: the job’s file resource limit; the available space of the file system containing the file; the maximum file size supported by the operating system, and the BUFSIZE. On UNIX systems, the “df” command lists the maximum space and available space in a file system. Your resource limit is displayed by as the “Maximum file size” by the “limits” special function.

On a 32-bit processor running UNIX, the operating system's maximum file size has traditionally been 2 GB (actually 2**32-1 or less). In recent years, many systems have switched over to 64-bit processors or now support “large files,” i.e., a file that can exceed 2 GB. Table 4-7 lists those versions of MD Nastran that support large files.

Table 4-7  Database I/O Capabilities

<table>
<thead>
<tr>
<th>Computer</th>
<th>Large File</th>
<th>File Mapping</th>
<th>Buffered I/O</th>
<th>Async I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>Yes¹</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HP Alpha UNIX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HP-UX</td>
<td>Yes²</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Intel Linux</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Intel Linux8664</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes⁶</td>
</tr>
<tr>
<td>Intel Windows</td>
<td>Yes</td>
<td>Yes³</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IRIX64</td>
<td>Yes⁴</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Solaris</td>
<td>Yes⁵</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUPER-UX</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>UXP/V</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Notes:

1. Large files are available if the file system containing the file supports large files. See your system administrator to determine which file systems, if any, support large files.

2. Large files can only be created on file systems supporting large files (the flags value from “df -g” must show the 0x10 bit set).

3. File mapping is not available on Windows 95 and 98.

4. Large files can only be created on “XFS” file systems.

5. Large files are available on Solaris 2.6 or later if the file system containing the file supports large files. See your system administrator to determine which file systems, if any, support large files.

6. Supported on NEC and SGI IA64 only.

7. MIO is available with an MIO license from IBM.

The default database provides for five DBsets that are subdivided into two categories (scratch and permanent DBsets) as follows:

- Three DBsets are scratch DBsets that are typically deleted at the end of a run. The logical names for these DBsets are SCRATCH, SCR300, and OBJSCR.
- The remaining two DBsets have the default names of dbs.MASTER and dbs.DBAL, where dbs is set by the “dbs” keyword.

The database may be defined in two different ways:

1. Using the “dbs” keyword on the command line; see “Using the “dbs” Keyword” on page 115.

2. Using ASSIGN statements in the FMS Section of the input data file. See “ASSIGN Statement for DBsets” on page 110 and “Using the ASSIGN Statement” on page 116.
Using the “dbs” Keyword

To illustrate the use of the “dbs” keyword, see the TPL file “am762d.dat”

```
ID MSC, AM762D $ JFC 30SEP88
$ DBS=AM762D SPECIFIED WHEN JOB SUBMITTED
TIME 2
SOL 101 $ SUPERELEMENT STATICS
CEND
TITLE = EXAMPLE: SPECIFY DBS=AM762D WHEN JOB SUBMITTED AM762D
SUBTITLE = COLD START
LOAD = 11
DISPLACEMENT = ALL
ELFORCE = ALL
BEGIN BULK
CBEAM,1,1,10,20,0.,1.,0.
FORCE,11,20,,100.,1.,.8,1.
GRID,10,,0.,0.,0.,,123456
GRID,20,,10.,0.,0.
MAT1,100,1.+7,,.3
PBEAM,1,100,1.,.08,.064,.1
ENDDATA $ AM762D
```

To run this job, enter

```
md20071 nastran TPLDIR:am762d
```

The default value for “dbs” in this example is “./am762d” on UNIX and “\am762d” on Windows. The DBALL and MASTER DBsets are created in your directory as “am762d.DBALL” and “am762d.MASTER” respectively; and the output files are “am762d.f04”, “am762d.f06”, and “am762d.log”.

To restart from the previously created DBsets, use the following command:

```
md20071 nastran TPLDIR:am762r dbs=am762d
```
The input data for the restart is TPL file am762r.dat. The “dbs” keyword is set to “am762d”. The following is sample input for the am762r.dat file:

```
$ RESTART VERSION = 1 $ RESTART FROM AM762D
$ DBS=AM762D SPECIFIED WHEN JOB SUBMITTED
ID MSC, AM762R $ JFC 30S3088
TIME 2
SOL 101
CEND
TITLE = EXAMPLE: RESTART, ATTACH DATABASE VIA DBS=AM762D AM762R
SUBTITLE = RESTART WITH LARGER LOAD
SELG = ALL $ GENERATE NEW LOAD
SELR = ALL $ REDUCE NEW LOAD
LOAD = 11
DISPLACEMENT = ALL
ELFORCE = ALL
BEGIN BULK
FORCE,11,20,,100.,1.,.8,1.
ENDATA $ AM762R
```

The existing DBALL and MASTER DBsets created in your directory by the “am762d” job are used. The output files from this job are “am762r.f04”, “am762r.f06”, and “am762r.log”.

**Using the ASSIGN Statement**

This section contains two examples using the ASSIGN statement. The first example, TPL file am763d.dat shows how to use the ASSIGN statement to create the database files. The second example shows how to use the ASSIGN statement to assign database files in a restart job.

```
ASSIGN 'MASTER=DSDIR:am763d.MYMASTER'
ASSIGN 'DBALL=DSDIR:am763d.MYDBALL'
$ DBSETS CREATED WITH DIRECTORIES AND NAMES AS ASSIGNED ABOVE.
$ THIS IS ALTERNATE METHOD TO BE USED INSTEAD OF SPECIFYING DBS = AM763D
$ WHEN JOB IS SUBMITTED.
$ ID MSC, AM763D $ FILENAME CHANGED 16SEP88 -- JFC
TIME 2
SOL 101 $ STRUCTURED SUPERELEMENT STATICS WITH AUTO RESTART
CEND
TITLE = EXAMPLE: DATABASE CREATED VIA ASSIGN CARDS AM763D
SUBTITLE = COLD START.
LOAD = 11
DISPLACEMENT = ALL
ELFORCE = ALL
BEGIN BULK
CBEAM,1,1,10,20,0.,1.,0.
FORCE,11,20,,100.,1.,.8,1.
GRID,10,,0.,0.,0.,123456
GRID,20,,10.,0.,0.,123456
MAT1,100,1.,.08,.064,.1
ENDDATA
```
Before you submit this job, create a “dbs” directory in your current working directory and set the DBSDIR environment variable to “dbs” as follows:

```bash
export DBSDIR=dbs
```
in the Korn shell,

```bash
setenv DBSDIR dbs
```
in the C-shell, or

```bash
set DBSDIR=dbs
```
on Windows.

Once the DBSDIR environment variable is set, the job is submitted with the command:

```bash
md20071 nastran TPLDIR:am763d
```

The DBsets “mydball” and “mymaster” are created in the “dbs” directory with the names “am763d.MYMASTER” and “am763d.MYDBALL” respectively. The output files “am763d.f04”, “am763d.f06”, and “am763d.log” are created in the current working directory.
The second example (TPL file am763r.dat) illustrates a restart that uses the ASSIGN statement:

```plaintext
RESTART $ RESTART FROM AM763D, SAVE VERSION 1 ON DATABASE
$ ATTACH AM763D DATABASE WITH ASSIGN COMMANDS BELOW
ASSIGN MASTER='DBSDIR:am763d.MYMASTER'
ID MSC,AM763R $ FILENAME CHANGED 16SEP88 -- JFC
TIME 2
SOL 101
CEND
TITLE = EXAMPLE: RESTART, DATABASE ATTACHED VIA ASSIGN CARDS AM763R
SUBTITLE = RESTART -- ADD STRESS RECOVERY COEFFICIENTS TO PBEAM
LOAD = 11
DISPLACEMENT = ALL
ELFORCE = ALL
STRESS = ALL
BEGIN BULK
$ WITH STRUCTURED SOLUTION SEQUENCES (SOL 101+), ALL BULK DATA IS STORED
$ ON DATABASE.
$ ON RESTART, ONLY INCLUDE ADDITIONAL CARDS OR CHANGED CARDS.
$ & $ DELETE OLD PBEAM CARD ON DATABASE, ADD STRESS RECOVERY COEFFICIENTS
$ AND REPLACE AS FOLLOWS.
PBEAM,1,100,1.,.08,.064,.,,.,+PBEAM1
+PBEAM1,0.0,0.5,0.0,-0.5,0.3,0.0,-0.3,0.0,+PBEAM2
+PBEAM2,YES,0.5,1.0,.08,.064,.,,.,+PBEAM3
+PBEAM3,0.0,0.5,0.0,-0.5,0.3,0.0,-0.3,0.0
ENDDATA $ AM763R
```

To submit the above file, issue the command:

```plaintext
md20071 nastran TPLDIR:am763r
```

The DBsets “am763d.MYMASTER” and “am763d.MYDBALL” created by the previous job in the “dbs” directory are used. The output files “am763r.f04”, “am763r.f06”, and “am763r.log” are created in the current working directory.

**Using the INIT Statement**

DBsets are created using the INIT statement, which is documented in the “File Management Statements” in Chapter 2 of the *MD Nastran Quick Reference Guide*. For example,

```plaintext
INIT DBALL LOGICAL=(DBALL1(2000),DBALL2(300KB))
```
creates and allocates two members DBALL1 and DBALL2 to the DBALL DBset with a size of 2000 GINO blocks for DBALL1 and a size of 300 kilobytes for DBALL2. The size can be specified either as the number of GINO blocks or as a number followed by one of the following modifiers:

- **M or Mw**: Multiply the size by \(1024^2\), round up to a BUFFSIZE multiple.
- **Mb**: Multiply the size by \(1024^3/(bpw)\), round up to a BUFFSIZE multiple.
- **K or Kw**: Multiply the size by 1024, round up to a BUFFSIZE multiple.
- **Kb**: Multiply the size by \(1024/(bpw)\), round up to a BUFFSIZE multiple.
- **w**: Round the size up to a BUFFSIZE multiple.
- **b**: Divide the size by \(bpw\), round up to a BUFFSIZE multiple.

where \(bpw\) is 8 when mode = i8; 4 on all others. The modifier may be specified using any case combination.

**Note:** This syntax is similar to, but not the same as, the syntax described in "Specifying Memory Sizes" on page 104.
Using the INCLUDE Statement

The INCLUDE statement is used to insert a specified file into the input file. This statement is especially useful when you want to partition your input into separate files. The format is

```
INCLUDE filename
```

or

```
INCLUDE logical-symbol:filename
```

The file name must be quoted in single quotes if the name contains lowercase letters, spaces, commas, or dollar signs on UNIX; or spaces, commas, or dollar signs on Windows, for example,

```
INCLUDE 'file name'
```

Specifying the INCLUDE Filename

The filename can be continued, if necessary, on multiple lines of the input file. The filename is obtained from an INCLUDE, RFLATER, or RFINCLUDE statement as follows:

1. The filename is built up by concatenating tokens. A token is either a blank- or comma-delimited unquoted word or a quoted string (which can be continued across lines).
2. Token are separated by blanks or commas. The blanks or commas separating the tokens are ignored.
3. Statements may be continued by following the last token on a line by a comma, or specifying an incomplete quoted string (i.e., the closing quote is missing from the line). All trailing blanks on the incomplete quoted string’s initial line, all leading and trailing blanks on the incomplete quoted string’s intermediate lines, and all leading blanks on the incomplete quoted string’s final line are ignored.
4. Comments may be specified after the last filename token of a line that is not within an incomplete quoted string. The comment is started with an unquoted dollar sign “$”, and continues to the end of the current line.
5. Only the first 72 columns of a line are scanned, i.e., any characters from column 73 and onward are ignored.

These rules are best explained via some examples.

**Note:** The following examples contain a mixture of UNIX and Windows pathnames. The concepts demonstrated by each example are valid on both systems.
include datafile.dat

The filename is “DATAFILE.DAT”.

include 'c:\abc\def\ghi.include'

The filename is “c:\abc\def\ghi.include”.

include '/mydir' /level1 /level2/ 'myfile.x'

The filename is “/mydir/LEVEL1/LEVEL2/myfile.x”.

RFAlter '/mydir /level1 /level2 /level3/mydata'

The filename is “/mydir/level1/level2/level3/mydata”.

include '/proj /dept123 /sect 456 /joe/flange.bdf'

The filename is “/proj/dept123/sect 456/joe/flange.bdf”.

rfinclude c:\project
   $ A comment line
   'Data Files' \subdir\thisfile

The filename is “C:\PROJECT\Data Files\SUBDIR\THISFILE”.

The following examples illustrate what happens when comments or quotes are incorrectly placed.

include 'TPLDIR:alter.file $ comment stmt 2 $ word ' $ comment 3 ' info

The filename is “TPLDIR:alter.file Scommentstmt 2 $ word”.

include 'TPLDIR:alter.file $ comment stmt 2 $ word' $ comment 3 ' info

The filename is “TPLDIR:alter.file Scommentstmt 2 $ word”.

include 'TPLDIR:alter.file $ comment stmt 2 $ word
   ' $ comment 3 ' info

The filename is “TPLDIR:alter.file Scommentstmt 2 $ word”.

include 'TPLDIR:alter.file $ comment stmt 2 $ word
   ' $ comment 3 ' info

The filename is “TPLDIR:alter.file Scommentstmt 2 $ word”.
The filename is "proj, $ Proj Name/DEPT123/sect 456, $ Sect Name/MYFILE.DAT".

**Locating INCLUDE Files**

Once the *filename* has been obtained from the include statement and any logical symbols have been expanded, up to four filenames on UNIX systems and two filename on Windows systems will be searched for. The filename are:

1. The *filename* as specified by the include statement. If *filename* does not end in the the file type specified by the “jidtype” keyword, it is appended.
2. UNIX: The *filename* constructed immediately above, converted to lower-case, unless *filename* is already all lower-case (i.e., it was specified as a quoted string).
3. The *filename* as specified by the include statement, without the file type specified by “jidtype”.
4. UNIX: The *filename* specified above, converted to lower-case, unless *filename* is already all lower-case (i.e., it was specified as a quoted string).

For example, consider the statement

```plaintext
include File1
```

and assume “jidtype=dat” was specified or defaulted. MD Nastran will consider the following filenames on UNIX in the order specified:

```plaintext
FILE1.dat
file1.dat
FILE1
file1
```

and the following filenames on Windows in the order specified:

```plaintext
file1.dat
file1
```
For another example, consider the statement

```
include 'File1.bdf'
```

and assume “jidtype=dat” was specified or defaulted. MD Nastran will consider the following filenames on UNIX in the order specified:

```
File1.bdf.dat
file1.bdf.dat
File1.bdf
file1.bdf
```

and the following filenames on Windows in the order specified:

```
File1.bdf.dat
File1.bdf
```

If `filename` contains a directory component, MD Nastran will attempt to locate one of the four UNIX or two Windows filenames in the specified directory. If none of the names exist or are not readable, a UFM will be issued and the job will exit.

If `filename` does not contain a directory component, the default directory is the current working directory (i.e., the directory where the nastran command was run). If none of the file names exist in the current working directory, MD Nastran will look in the directory containing the file that specified the INCLUDE statement. If none of the file names exist in that directory, and the file that contained the INCLUDE statement was itself included, i.e., the INCLUDE was nested, the directory containing the parent file will be searched. This nesting will continue until the directory containing the input data file has been searched. If a file has not yet been located, the list of directories specified by the “jidpath” keyword will be searched in order. If no file can be found in any of these directories, a UFM will be issued and the MD Nastran job will exit.

**Note:** Recall that character-case is insignificant to Windows file names.
Using the SSS Alter Library

The SSS Alter directory, \textit{install\_dir/md20071/nast/sssalter} on UNIX and \textit{install\_dir/md20071/nast/sssalter} on Windows contains alters (modifications to MD Nastran solution sequences) and associated support files that represent client-requested or prototype features that are not yet implemented in MD Nastran’s standard solution sequences. These alters can be inserted using the INCLUDE statement and the SSSALTERDIR symbol. For example,

\begin{verbatim}
INCLUDE 'SSSALTERDIR:zfreqa.dat'
\end{verbatim}
Resolving Abnormal Terminations

MD Nastran generates a substantial amount of information concerning the problem being executed. The .f04 file provides information on the sequence of modules being executed and the time required by each of the modules; the .log file contains system messages. A list of known outstanding errors for MD Nastran is delivered in the file \texttt{install\_dir/md20071/doc/error.lis} on UNIX and \texttt{install\_dir/md20071/doc/error.lis} on Windows. Please consult this file for limitations and restrictions.

MD Nastran may terminate as a result of errors detected by the operating system or by the program. If the DIAG 44 is set (see the diag keyword (page 300) and the \textit{MD Nastran Quick Reference Guide}), MD Nastran will produce a dump of several key internal tables when most of these errors occur. Before the dump occurs, there may be a fatal message written to the .f06 file. The general format of this message is

\begin{verbatim}
***SYSTEM FATAL ERROR 4276, subroutine-name ERROR CODE n
\end{verbatim}

This message is issued whenever an interrupt occurs that MD Nastran is unable to satisfactorily process. The specific reasons for the interrupt are usually printed in the .f06 and/or .log file; “n” is an error code that is explained in Chapter 16 of the \textit{MSC Nastran Reference Manual}.

Whenever the System Fatal Error 4275 or 4276 is associated with a database error, further specific information is written to the .f06 file as follows:

\begin{verbatim}
bio-function ERROR - STATUS = errno, FILX = i, LOGNAME = logical, NSBUF3 = j
FILE = filename
BLKNBR = k
ERROR MESSAGE IS --
error-message-text
\end{verbatim}

The FILE and/or BLKNBR lines may not be present, depending upon the \textit{bio-function} issuing the message.

Interpreting System Error Codes

If an operating system error occurs, an attempt is made to catch the error and place the error number in the .log file. A description of these error numbers may be obtained with the following command:

\begin{verbatim}
IBM cat /usr/include/sys/errno.h
Sun man -s2 intro
Other UNIX man 2 intro
\end{verbatim}
Terminating a Job

There may be instances when a running job must be prematurely terminated; this is accomplished using one of the following procedures:

Job Running in the Foreground (batch=no on UNIX; all jobs on Windows)

Use the interrupt sequence (on NEC and Silicon Graphics systems this sequence is usually “Ctrl-\n”; on other systems “Ctrl-C”).

Job Running in the Background (batch=yes or after=time on UNIX)

Use the “ps” command to find the process ID (PID) of the MD Nastran job (i.e., the \textit{install\_dir/md20071/arch/analysis} executable) and issue the command

\texttt{kill \textit{pid}}

where \textit{pid} is the process ID.

Job Running Under NQS or NQE (queue=queue\_name on UNIX)

1. Use “\texttt{qstat -a}” to find the request-id of your job.
2. Use “\texttt{qdel request-id}” to delete a job that has not yet started; or use “\texttt{qdel -k request-id}” to kill a job that has already started where \textit{request-id} is the request ID.

Flushing .f04 and .f06 Output to Disk (UNIX)

As MD Nastran writes to the .f04 and .f06 files, the FORTRAN runtime libraries will buffer this I/O in memory to reduce the amount of time consumed by disk I/O. When the buffers are filled (i.e., MD Nastran has written a sufficient amount of information to the .f04 or .f06 file), the buffers will be flushed to the files by the FORTRAN runtime libraries. In a large job, some modules may do substantially more computation than I/O. As a result, the I/O may remain in the FORTRAN buffers (possibly for several hours) before they are written to disk.

AIX, HP-UX, IRIX64, and Linux computers support asynchronous flushing of the .f04 and .f06 files. To do this, enter the command

\texttt{kill -USR1 \textit{pid}}

where \textit{pid} is the process ID of the running MD Nastran job (i.e., the \textit{install\_dir/md20071/arch/analysis} executable). There may be a time delay between the time you issue the \texttt{kill} command and the time the files are actually updated.
Common System Errors

The most common system errors encountered during an MD Nastran job are described below.

**UNIX Disk I/O Errors**

- **ERRNO 1 (EPERM) - no permission to file (all systems).**
  
  Please check the ownership and mode of the file or directory with the “ls -l” command. Change either the ownership or permissions of the file or the directories along the path. The chgrp(1) command is used to change the group of a file, chmod(1) is used to change permissions of the file, and chown(1) is used to change ownership of the file.

- **ERRNO 27 (EFBIG) - file is too large (all systems)**
  
  This error occurs if a file's size exceeds a resource limit. The resource limits in effect during the job's execution are printed in the .log file under the heading “Current Resource Limits.” Increase the “-If” and “-IF” parameters on your qsub command if you are running NQS or NQE; ask your system administrator to increase your “Filesystem Space” limit (SUPER-UX) or “File Size” limit (all others).

- **ERRNO 28 (ENOSPC) - disk space is completely filled (all systems).**
  
  MD Nastran deletes its scratch files at termination even if the disk space fills up. Therefore, the df(1) command may show a large amount of free space even though the job failed due to lack of disk space. Both the current working directory and the scratch directory need to be checked. Move your files to a disk with more space (see the “out”, “dbs”, and “sdirectory” keywords), or delete unnecessary files from the disk.

**Inability to Allocate the Requested Amount of Memory (OPEN CORE Allocation Failed)**

- **Temporary lack of swap space (all systems).**
  
  This error may be caused by too many processes running at the same time. Decrease the number of processes or increase the available swap space.

- **The data segment of the process has exceeded the UNIX resource limit (UNIX).**
  
  The resource limits in effect during the job's execution are printed in the .log file under the heading “Current Resource Limits.” Ask your system administrator to increase your “Data Segment Size” (all), “Maximum break size” (HP-UX), or “Virtual Address Space” (all others).

- **memory allocation error: unable to allocate n words (HP-UX).**
  
  The resource limits in effect during the job's execution are printed in the .log file under the heading “Current Resource Limits.” Check your “Maximum break size”; if this is smaller than the requested memory, ask your system administrator to increase your limit.

  If your limit is large enough, the system wide “shmmax” and “maxdsize” kernel parameters may be too small. These parameters must be large enough to accommodate all simultaneously executing MD Nastran jobs plus all others users of shared memory. These values are modified using sam(1M), see “Kernel Parameters” under “Configurable Parameters”.
It may also be possible to correct these errors with the following:

- Reduce the amount of memory requested by the “memory” keyword.
- Increase the “-lm” and “-lM” parameters if you directly submitted your job to NQS or NQE using a "qsub" command.
- Increase the “prmdelta” or “ppmdelta” keyword values if you submitted your job to NQS or NQE using the nastran command’s “queue” keyword.

EAG FFIO Errors (IRIX64)
The following error message may appear on LINUX64 and IRIX64 systems when FFIO is being used:

```
eie open failure : Not enough space for cache pages
```

This message is a consequence of not having enough memory for the eie cache pages. System memory requirements are as follows:

<table>
<thead>
<tr>
<th>Description</th>
<th>Size</th>
<th>Where Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>executable</td>
<td>6.5 MW</td>
<td>“System Descriptions” on page 363</td>
</tr>
<tr>
<td>opencore memory</td>
<td>memory keyword</td>
<td>“Keywords and Environment Variables” on page 291 and “Managing Memory” on page 138</td>
</tr>
<tr>
<td>EIE buffers/ Cache</td>
<td></td>
<td>“Keywords and Environment Variables” on page 291</td>
</tr>
</tbody>
</table>

If the job was directly submitted with the “qsub” command, then the error can be avoided by increasing the NQS “lm” and “lM” parameters. The value should be at least 6.5 MW plus the value specified by the “memory” keyword plus the amount needed for eie. To determine the amount needed for FFIO, consider the following “ff_io_opts” request:

```
(eie:128:16:1:1:1:0,set:0:0)
```

This request requires an additional:

```
128(blocks/page) × 16(pages) × 512(words/block) = 1048576W=1MW
```

If the job was submitted with the nastran command’s “queue” keyword, the nastran command automatically adjusts the memory request based on the “ff_io_cachesize” keyword. The “eie open failure” message should only appear if the user modified the “ff_io_defaults” or “ff_io_opts” keywords without modifying the “ff_io_cachesize” keyword. This error can be avoided by increasing the value set by the “ppmdelta” keyword (see “ppmdelta (UNIX)” on page 326) to 6.5 MW plus the amount of memory for FFIO.
See the URL

file://install_dir/md20071/arch/ffio.html

where arch is irix64 or linux64, for a complete description of EAG FFIO.
Using the Advanced Functions of MD Nastran

- Overview 132
- Using the Advanced Keywords 133
- Using the NASTRAN Statement 136
- Managing Memory 138
- Managing DBsets 140
- Interpreting the .f04 File 149
- Running a Job on a Remote System 154
- Running Distributed Memory Parallel (DMP) Jobs 163
- Configuring and Running SOL 600 185
- SOL 600 Parallel Processing on Windows 192
- Installation Notes 193
- Running an ISHELL Program 200
- Using the ISHELL-INCLUDE Statement (!) 204
- Improving Network File System (NFS) Performance (UNIX) 207
- Creating and Attaching Alternate Delivery Databases 208
- Checkpoint Restart Facility (SGI-IRIX64) 211
Overview

This chapter discusses the NASTRAN statement, as well as how to manage MD Nastran’s internal memory allocations and databases. It also shows how to interpret performance related information in the .f04 file and some of the lower-level database messages, how to run a job on a remote system, run a DMP job, use the ISHELL module, and finally, how to create alternate delivery databases.
### Using the Advanced Keywords

The following is a partial list of the advanced keywords that may be used on the command line or placed into RC files as appropriate. More basic keywords are listed in “Using the Basic Keywords” on page 102; keywords specific to remote processing are listed in “Running a Job on a Remote System” on page 154, while keywords specific to distributed processing are listed in “Running Distributed Memory Parallel (DMP) Jobs” on page 163. Finally, a complete list of all keywords and their syntax is listed in “Keywords” on page 292.

#### All Systems

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffsize</td>
<td>Specifies the size of database I/O transfers.</td>
</tr>
<tr>
<td>bpool</td>
<td>Specifies the number of GINO blocks set aside for buffer pooling.</td>
</tr>
<tr>
<td>delivery</td>
<td>Specifies an alternate delivery database name.</td>
</tr>
<tr>
<td>exe</td>
<td>Specifies an alternate solver executable.</td>
</tr>
<tr>
<td>nastran</td>
<td>Specifies NASTRAN statements.</td>
</tr>
<tr>
<td>proc</td>
<td>Specifies an alternate solver executable file type.</td>
</tr>
<tr>
<td>rank</td>
<td>Specifies the rank size for the sparse solvers.</td>
</tr>
<tr>
<td>smem</td>
<td>Specifies the number of GINO blocks to set aside for MEMFILE portion of the SCRATCH DBset.</td>
</tr>
<tr>
<td>sysfield</td>
<td>Specifies global SYS parameters. See “Using the SYS Field” on page 140.</td>
</tr>
<tr>
<td>sysn</td>
<td>Specifies SYSTEM cell values.</td>
</tr>
</tbody>
</table>

#### UNIX Systems

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>post</td>
<td>Specifies UNIX commands to be executed after the job completes.</td>
</tr>
<tr>
<td>pre</td>
<td>Specifies UNIX commands to be executed before the job begins.</td>
</tr>
</tbody>
</table>

#### All Systems Except Primepower, SUPER-UX and UXP/V

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>parallel</td>
<td>Specifies the number of SMP tasks to use in certain numeric modules.</td>
</tr>
</tbody>
</table>
AIX and Linux8664 Only

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mio-cachesize</td>
<td>Specifies the size of the MIO cache to be used.</td>
</tr>
</tbody>
</table>

IRIX64 Only

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>spintime</td>
<td>Specifies the time to wait in a spin loop.</td>
</tr>
<tr>
<td>threads</td>
<td>Enables Dynamic Thread Management and specifies the preferred number of threads.</td>
</tr>
</tbody>
</table>

IRIX64 and LINUX IA-64 (SGI Altix) Only

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ff_io</td>
<td>Enables the FFIO high performance I/O system.</td>
</tr>
<tr>
<td>ff_io_cachesize</td>
<td>Specifies the size of the FFIO cache.</td>
</tr>
</tbody>
</table>

Solaris Only

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>sun_io</td>
<td>Enables the SUN_IO high performance I/O system and specifies the parameters.</td>
</tr>
</tbody>
</table>

SUPER-UX and Linux IA-64 (NEC Azusa) Only

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>hpio_param</td>
<td>Enables the HPIO high performance I/O system and specifies the parameters.</td>
</tr>
</tbody>
</table>

Queuing (UNIX)

Note: These capabilities are dependent upon the queue submission commands defined by the “submit” keyword and your queuing system. The keywords may not work on your system.
## Using the Advanced Functions of MD Nastran

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ppcdelta</td>
<td>Specifies the per-process CPU time limit delta.</td>
</tr>
<tr>
<td>ppmdelta</td>
<td>Specifies the per-process memory limit delta.</td>
</tr>
<tr>
<td>prmdelta</td>
<td>Specifies the per-request memory limit delta.</td>
</tr>
<tr>
<td>qclass</td>
<td>Specifies an optional queue class.</td>
</tr>
<tr>
<td>qoption</td>
<td>Specifies other queue command options.</td>
</tr>
<tr>
<td>submit</td>
<td>Defines queues and their associated submittal commands.</td>
</tr>
</tbody>
</table>
Using the NASTRAN Statement

The NASTRAN statement allows you to change parameter values at runtime.

The format of NASTRAN statements is

NASTRAN    KEYWORD1=A, KEYWORD2=B, ... KEYWORDi=I

An input file may contain more than one NASTRAN statement. A full description of these keywords is found in “The NASTRAN Statement” on page 12 of the *MD Nastran Quick Reference Guide*. A brief description of a few of the keywords follows:

**AUTOASGN**

AUTOASGN is used to determine which DBsets are automatically assigned (see the following table). The default is AUTOASGN=7, which specifies that all DBsets are to be automatically assigned.

<table>
<thead>
<tr>
<th>Value</th>
<th>Default DBsets</th>
<th>Delivery DBsets</th>
<th>DBLOCATEd DBsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>7 (Default)</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Note:**

1. Default DBsets are the user-default DBsets and any DBsets specified by INIT statements (see Table 4-7).

2. Delivery DBsets contain the Structured Solution Sequences.

3. DBLOCATEd DBsets are the DBsets specified by DBLOCATE statements. See “DBLOCATE” on page 77 of the *MD Nastran Quick Reference Guide*.

**BUFFPOOL, SYSTEM(114)**

See the “bpool” command line keyword, (page 296).
BUFFSIZE, SYSTEM(1)
See the “buffsize” command line keyword, (page 296).

PARALLEL, SYSTEM(107)
See the “parallel” command line keyword, (page 324).

SYSTEM(128)
SYSTEM(128) specifies the maximum interval of CPU time (in minutes) between database directory updates to the MASTER DBSET when the INIT MASTER(RAM) option is being used. The default is 1 minute on NEC systems and 5 minutes on all others. See “DBUPDATE” on page 85 of the MD Nastran Quick Reference Guide for more information.

SYSTEM(198), SYSTEM(205)
See the “rank” keyword, (page 329).

SYSTEM(207)
See the “LOCK” SYS field keyword, (page 342).

SYSTEM(275)
SYSTEM(275) sets the time-out for an ISHELL program to complete its work. If the value is negative (the default is -1), the ISHELL module will wait until the executable finishes, i.e., there is no time-out. If the value is positive, the ISHELL module will wait for the specified number of seconds.

If the value is zero, the ISHELL module will determine if an executable can be found, and return a zero status if found and a non-zero status if it can’t be found.
Managing Memory

Memory is dynamically allocated at runtime with the “memory” keyword of the nastran command. The memory can be partitioned in a variety of ways (see the memory map at the top of the .f04 file for the actual memory allocation used in a job). To make the most effective choice of the sizing parameters, see the following map of MD Nastran’s memory:

![Memory Map Diagram]

As can be seen in this diagram, the memory available for use by MD Nastran modules (user open core) is the amount specified by the “memory” keyword (open core size) less the space required by memory resident files and executive tables. The actual user open core is calculated as follows:

\[
\text{User Open Core} = \text{MEM} - (\text{EXEC} + \text{RAM} + \text{SMEM} \times \text{BUFFSIZE} + \text{BUFFPOOL} \times (\text{BUFFSIZE} + 10))
\]

**MEM**  
The total size of open core. There is no default. Set by the “memory” keyword, (p. 315).

**EXEC**  
The executive system work area. The size is \(70409 + 4 \times \text{BUFFSIZE}\) words.

**RAM**  
NDDL tables. The default is 30000. Set by the FMS statement INIT MASTER (RAM=value).
SMEM
The memory-resident file space for temporary database files. The default is 0 for
SUPER-UX: 100 for all others. Set by the FMS statement
INIT SCRATCH (MEM=value) or the “smemory” keyword, (p. 337).

BUFFSIZE
The maximum BUFFSIZE used for all the DBsets referenced by the job. The default is
8193. Set by the “buffsize” keyword, (p. 296).

BUFFPOOL
The buffer pool area for permanent database files. The default size is 27 on SUPER-
UX: 37 on all others. Set by the “bpool” keyword, (p. 295).

The INIT statement may be used to size MASTER and SCRATCH memory. Several examples of the
INIT statement, along with an explanation of their uses, follow:

1. If the available memory is a critical resource, then using the following selection reduces memory
requirements at the expense of increased CPU and wall-clock time.

   INIT SCRATCH (NOMEM) $ temporary database files

2. Performance gains may be made by increasing the memory-resident area for the scratch and
permanent DBset(s) as follows. Note that the default RAM is sufficiently large and need not be
increased.

   NASTRAN BUFFPOOL=70 $ increase permanent DBsets
   INIT SCRATCH (MEM=200) $ increase scratch memory

3. If disk space is critical, then all DBsets may be deleted at the end of the job by specifying “S” on
the INIT MASTER statement as follows:

   INIT MASTER(S) $ delete DBsets at end of job

   This statement is identical to specifying “scratch=yes” on the command line.

4. If disk space is critical, but data recovery restarts are required, then a database may be created that
will support data recovery restarts by setting “scratch=mini” on the command line.

   md2007 nastran example scratch=mini
Managing DBsets

I/O Performance Libraries

Several of the vendors have provided enhanced I/O libraries for use with MD Nastran database I/O. The specific keywords enabling or controlling these keywords are:

Table 5-1 I/O Performance Library Keywords

<table>
<thead>
<tr>
<th>System</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>mio-cachesize (page 318)</td>
</tr>
<tr>
<td>HP-UX</td>
<td>use_aio, (page 346)</td>
</tr>
<tr>
<td>IRIX64</td>
<td>ff_io, (page 304)</td>
</tr>
<tr>
<td>Linux IA64 (NEC Azusa)</td>
<td>hpio_param, (page 308)</td>
</tr>
<tr>
<td>Linux IA64 (SGI Altix)</td>
<td>ff_io, (page 304)</td>
</tr>
<tr>
<td>Linux x86-64 (with mio license)</td>
<td>mio-cachesize (page 318)</td>
</tr>
<tr>
<td>Solaris, Primepower</td>
<td>sun_io, (page 339)</td>
</tr>
<tr>
<td>SUPER-UX</td>
<td>hpio_param, (page 308)</td>
</tr>
</tbody>
</table>

Please see “Keywords” on page 292 for additional information on these keywords.

Using the SYS Field

The SYS field is used to specify computer-dependent parameters on ASSIGN statements. If your computer does not recognize a particular parameter, it is silently ignored. This keyword is specified as a comma separated list of keyword=value pairs. For example, file locking may be disabled on for a particular DBset with the following statement:

```
ASSIGN =DBALL=mydball.DBALL= SYS='LOCK=NO'
```

A global SYS field for all DBsets can be specified by the “sysfield” keyword, (page 133).

The following tables describe the SYS field parameters. A complete description of parameters and their syntax is available in “SYS Parameter Keywords” on page 349.

All Systems

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>lock</td>
<td>Lock database files.</td>
</tr>
</tbody>
</table>
CHAPTER 5
Using the Advanced Functions of MD Nastran

Systems Supporting File Mapping (see Table 4-7)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapio</td>
<td>Use the virtual memory system to map database files to memory.</td>
</tr>
<tr>
<td>wnum</td>
<td>Specifies the default number of maps used on database files.</td>
</tr>
<tr>
<td>wsize</td>
<td>Specifies the default size of maps used on database files.</td>
</tr>
</tbody>
</table>

Systems Supporting Buffered I/O (see Table 4-7)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffio</td>
<td>Use intermediate buffers to hold database file records</td>
</tr>
<tr>
<td>wnum</td>
<td>Specifies the default number of buffers used for database files</td>
</tr>
<tr>
<td>wsize</td>
<td>Specifies the default size of buffers used for database files</td>
</tr>
</tbody>
</table>

Systems Supporting Async I/O (see Table 4-7)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>async</td>
<td>Use intermediate buffers to hold database file records, doing predictive read-ahead</td>
</tr>
<tr>
<td>wnum</td>
<td>Specifies the default number of buffers used for database files</td>
</tr>
<tr>
<td>wsize</td>
<td>Specifies the default size of buffers used for database files</td>
</tr>
</tbody>
</table>

Using File Mapping

Notes:

1. See Table 4-8 to determine if file mapping is available on your computer.
2. IRIX64 users should use the “ff_io” parameters (page 304) as an alternative to file mapping.

File mapping is a way to tell the operating system to use the virtual paging system to process a file. From the perspective of the process, file mapping effectively changes the file I/O operations from synchronous to asynchronous because the paging functions of the operating system perform the I/O as part its normal virtual memory management. File mapping can be used for both permanent and temporary DBsets.

The “wsize” and “wnum” parameters, described in “SYS Parameter Keywords” on page 349, specify the size of the window mapping the file to memory and the number of windows or maps that will be used for each file. The larger the window, the less often it must be moved when the file is sequentially read or written. Multiple maps allow several I/O streams to be active in the same file.

File mapping is controlled globally using the “sysfield” command line keyword (page 133) and, for individual DBsets, using the ASSIGN statement SYS field or the logical-name capability of the
"sysfield" command line keyword. These same statements can be used to specify the number and size of
the windows using the "wnum" and "wsize" keywords.

As an example, if file mapping is to be enabled for all files, the "sysfield" keyword in the command
initialization or RC file or on the command line is:

sysfield=mapio=yes

If file mapping is to be disabled for all files, the "sysfield" keyword is:

sysfield=mapio=no

Enabling file mapping for all but a specified set of DBsets may be done in either of the following ways:

- using both the "sysfield" keyword and ASSIGN specifications. In the command initialization
  file, RC file, or on the command line, specify:
    
    sysfield=mapio=yes
  
  and, in the MD Nastran data file, specify:

    ASSIGN logical-name=filename, SYS='MAPIO=NO'

  for those files to be processed using normal disk I/O processing.

- using the logical-name capability of the "sysfield" keyword. In the command initialization file,
  RC file, or on the command line, specify:

    sysfield=mapio=yes,logical-name(mapio=no)

If more than one file is to be processed using normal disk I/O processing, the other logical-names may
be specified on the same "sysfield" statement, on subsequent "sysfield" statements or using the
"wildcard" capabilities of the logical-name capability. For example, if there are two user DBsets, USER1
and USER2, that are to be processed using normal disk I/O processing, specify:

sysfield=mapio=yes,user*(mapio=no)

Disabling file mapping for all but a specified set of DBsets may be done in either of the following ways:

- using both the "sysfield" keyword and ASSIGN specifications. In the command initialization
  file, RC file, or on the command line, specify:

  sysfield=mapio=no

  and, in the MD Nastran data file, specify:

  ASSIGN logical-name=filename, SYS='MAPIO=YES'

  for those files to be processed using file mapping.

- using the logical-name capability of the "sysfield" keyword. In the command initialization file,
  RC file, or on the command line, specify:

  sysfield=mapio=no,logical-name(mapio=yes)

If more than one file is to be processed using file mapping, the other logical-names may be specified on
the same "sysfield" statement, on subsequent "sysfield" statements or using the "wildcard" capabilities
of the logical-name capability. For example, if the scratch DBsets, SCRATCH and SCR300, are to be
processed using file mapping, specify:
sysfield=mapio=no,scr*(mapio=yes)

Using Buffered I/O

Notes:
1. See Table 4-7 to determine if buffered I/O is available on your computer.
2. IRIX64 and SGI IA64 Altix users should use the “ff_io” parameters (page 304) as an alternative
   to buffered I/O.
3. SUPER-UX users should use the “hpio_param” parameter (page 308) as an alternative to
   buffered I/O.

Buffered I/O instructs MD Nastran to “buffer” or use intermediate memory areas to hold records of a file
before either writing them out to disk or copying them to the MD Nastran internal areas. The primary
purpose for using buffered I/O is to increase data reuse and, in some cases, to increase the actual
read/write data lengths beyond that normally used by MD Nastran. Buffered I/O can be used for both
permanent and temporary DBsets.

The “wsize” and “wnum” parameters, described in “SYS Parameter Keywords” on page 349, specify the
size of the buffer to be used to hold file records and the number of such buffers to be used. The larger the
buffer, the less often actual physical read/write operations are needed when the file is sequentially read
or written. Multiple buffers allow several I/O streams to be active in the same file.

Buffered I/O is controlled globally using the “sysfield” command line keyword (page 133) and, for
individual DBsets, using the ASSIGN statement SYS field or the logical-name capability of the
“sysfield” command line keyword. These same statements can be used to specify the number and size
of the buffers using the “wnum” and “wsize” keywords.

As an example, if buffered I/O is to be enabled for all files, the “sysfield” keyword in the command
initialization or RC file or on the command line is:

```
sysfield=buffio=yes
```

If buffered I/O is to be disabled for all files, the “sysfield” keyword is:

```
sysfield=buffio=no
```

Enabling buffered I/O for all but a specified set of DBsets may be done in either of the following ways:

- using both the “sysfield” keyword and ASSIGN specifications. In the command initialization
  file, RC file, or on the command line, specify:

  ```
sysfield=buffio=yes
  
  and, in the MD Nastran data file, specify:
  
  ASSIGN logical-name=filename, SYS='BUFFIO=NO'
  ```

  for those files to be processed using normal disk I/O processing.

- using the logical-name capability of the “sysfield” keyword. In the command initialization file,
  RC file, or on the command line, specify:
sysfield=buffio=yes,logical-name(buffio=no)

If more than one file is to be processed using normal disk I/O processing, the other logical-names may be specified on the same "sysfield" statement, on subsequent "sysfield" statements or using the "wildcard" capabilities of the logical-name capability. For example, if there are two user DBsets, USER1 and USER2, that are to be processed using normal disk I/O processing, specify:

sysfield=buffio=yes,logical-name(buffio=no)

Disabling buffered I/O for all but a specified set of DBsets may be done in either of the following ways:

- using both the "sysfield" keyword and ASSIGN specifications. In the command initialization file, RC file, or on the command line, specify:
  sysfield=buffio=no
  and, in the MD Nastran data file, specify:
  ASSIGN logical-name=filename, SYS='BUFFIO=YES'
  for those files to be processed using buffered I/O.
- using the logical-name capability of the "sysfield" keyword. In the command initialization file, RC file, or on the command line, specify:
  sysfield=buffio=no,logical-name(buffio=yes)
  If more than one file is to be processed using buffered I/O, the other logical-names may be specified on the same "sysfield" statement, on subsequent "sysfield" statements or using the "wildcard" capabilities of the logical-name capability. For example, if the scratch DBsets, SCRATCH and SCR300, are to be processed using buffered I/O, specify:

sysfield=buffio=no,scr*(buffio=yes)

Using Asynchronous I/O

Notes:
1. See Table 4-7 to determine if asynchronous I/O ("Async I/O") is available on your computer.
2. SGI IA64 Altix users should use the “ff_io” parameters (page 304) as an alternative to asynchronous I/O.

Asynchronous I/O instructs MD Nastran to use a predictive "read-ahead" algorithm to detect read patterns within a file (either forwards or backwards) and to issue asynchronous reads to bring data in from the file in anticipation of its later use. These reads use “buffers” or intermediate memory areas to hold the records of a file before they are actually used, i.e., copied to the MD Nastran internal areas. The primary purpose for using asynchronous I/O is to have records already in buffers when they are requested by MD Nastran through the use of the predictive logic. A secondary purpose, just as for buffered I/O, is to increase data reuse and in some cases, to increase the actual read/write data lengths beyond that normally used by MD Nastran. Asynchronous I/O can be used for both permanent and temporary DBsets.

The "wsize" and "wnum" parameters, described in “SYS Parameter Keywords” on page 349, specify the size of the buffer to be used to hold file records and the number of such buffers to be used. Because the
number of buffers affects how much actual "read-ahead" is possible, it is important that the number of available buffers be as large as possible. Typically, this value should be at least twice the number of expected different read patterns in file (some MD Nastran operations may be accessing as many as four different portions of the scratch files at a time). More buffers allow more read-ahead and allow several I/O streams to be active in the same file.

Asynchronous I/O is controlled globally using the "sysfield" command line keyword (page 342) and, for individual DBsets, using the ASSIGN statement SYS field or the logical-name capability of the "sysfield" command line keyword. These same statements can be used to specify the number and size of the buffers using the "wnum" and "wsize" keywords.

As an example, if asynchronous I/O is to be enabled for all files, the "sysfield" keyword in the command initialization or RC file or on the command line is:

```
sysfield=async=yes
```

If asynchronous I/O is to be disabled for all files, the "sysfield" keyword is:

```
sysfield=async=no
```

Enabling asynchronous I/O for all but a specified set of DBsets, specifying that sixteen buffers are to be used when it is enabled, may be done in either of the following ways:

- using both the "sysfield" keyword and ASSIGN specifications. In the command initialization file, RC file, or on the command line, specify:
  ```
sysfield=async=yes,wnum=16
  ```
  and, in the MD Nastran data file, specify:
  ```
ASSIGN logical-name=filename, SYS='ASYNC=NO'
```
  for those files to be processed using normal disk I/O processing.

- using the logical-name capability of the "sysfield" keyword. In the command initialization file, RC file, or on the command line, specify:
  ```
sysfield=async=yes,wnum=16,logical-name(async=no)
  ```
  If more than one file is to be processed using normal disk I/O processing, the other logical-names may be specified on the same "sysfield" statement, on subsequent "sysfield" statements or using the "wildcard" capabilities of the logical-name capability. For example, if there are two user DBsets, USER1 and USER2, that are to be processed using normal disk I/O processing, specify:
  ```
sysfield=async=yes,wnum=16,user*(async=no)
  ```

Disabling asynchronous I/O for all but a specified set of DBsets and, for those DBsets, that sixteen buffers are to be used, may be done in either of the following ways:

- using both the "sysfield" keyword and ASSIGN specifications. In the command initialization file, RC file, or on the command line, specify:
  ```
sysfield=async=no
  ```
  and, in the MD Nastran data file, specify:
  ```
ASSIGN logical-name=filename, SYS='ASYNC=YES,WNUM=16'
```
for those files to be processed using asynchronous I/O.

- using the logical-name capability of the "sysfield" keyword. In the command initialization file, RC file, or on the command line, specify:

  sysfield=async=no,logical-name(async=yes,wnum=16)

If more than one file is to be processed using asynchronous I/O, the other logical-names may be specified on the same "sysfield" statement, on subsequent "sysfield" statements or using the "wildcard" capabilities of the logical-name capability. For example, if the scratch DBsets, SCRATCH and SCR300, are to be processed using asynchronous I/O, specify:

  sysfield=async=no,scr*(async=yes,wnum=16)

### Interpreting Database File-Locking Messages (UNIX)

All database files are locked using the operating system function “fcntl(2)”. This prevents two or more MD Nastran jobs from interfering with one another; however, this does not prevent any other program or operating system command from modifying the files.

A read-write (exclusive) lock is requested for every database file that is to be modified. A read-only (shared lock) is requested on every database file that is not modified, e.g., DBLOCATEd databases. If the lock request is denied because another MD Nastran job is using the file in a potentially conflicting manner, the following fatal error message is written to the .f06 file:

```
bio-function ERROR - STATUS = errno, FILX = i, LOGNAME = logical, NSBUF3 = j
FILE = filename
ERROR MESSAGE IS --
Unable to acquire a lock_type lock.
lock-type-explanatory-text
Process ID pid is holding a conflicting lock.
```

where lock-type-explanatory-text is:

- `lock_type` is "read-only":

This operation failed because another process already holds a read-write lock on this file.

- `lock_type` is "read-write":

This operation failed because another process already holds a read-write or read-only lock on this file.
Some systems will deny a file lock because of an internal resource limit. In these cases, the job is allowed to continue, and the following message will be written to the .f06 file:

```
bio-function WARNING - STATUS = errno, FILX = i, LOGNAME = logical, NSBUF3 = j
FILE = filename
ERROR MESSAGE IS --
Unable to acquire a lock_type lock.
```

where `computer-specific-text` is:

**AIX**

The file appears to be in a Parallel Filesystem partition, and file locking is not supported in PFS partitions.

or

The system wide maximum number of file locks has been exceeded. See ENOLCK in SC23-2198 Call and Subroutine Reference.

**HP Alpha UNIX**

The system wide maximum number of file locks has been exceeded or the file may be in a partition that does not support file lock (e.g., an NFS partition). See ENOLCK in man 2 fcntl for further information.

**HP-UX**

The file appears to be an NFS file, and remote file locking was denied. See ENOLCK in man 2 fcntl for further information.

**IRIX64**

The system wide maximum number of file locks has been exceeded. See {FLOCK_MAX} in man 2 intro.

**Solaris**

The system wide maximum number of file locks has been exceeded. See ENOLCK in man -s 2 fcntl.

**All others**

The system wide maximum number of file locks has been exceeded. See ENOLCK in man 2 fcntl.

and advisory-text is:

- `lock_type` is “read-only”

If another job modifies this file during this run, there is the potential for incorrect results to occur in this job.

- `lock_type` is “read-write”
If another job accesses this file during this run, there is the potential for the file to be damaged and/or incorrect results to occur in both jobs.

**Disabling File Locking**

File locking can be disabled by:

- Setting “sysfield=lock=no” in an RC file or on the command line; see “Using the Advanced Keywords” on page 133. This affects all DBsets in the job.
- Setting SYSTEM(207) to a nonzero value using the NASTRAN statement; see “Using the NASTRAN Statement” on page 136. This affects all DBsets in the job.

The following informational message is written to the .f06 file:

```** SYSTEM INFORMATION MESSAGE - BIO
 SYSTEM(207).NE.0 - File locking suppressed. ```

- Setting SYS=LOCK=NO on an FMS INIT statement; see “Using the SYS Field” on page 140. This only affects the specific DBset(s).
Interpreting the .f04 File

MD Nastran writes information to the .f04 file that aids in monitoring and tuning the performance of your job. An overview of the complete .f04 file can be found in Section 9.2, “Output Description,” of the MSC Nastran Reference Manual. This section contains more detailed explanations of selected portions of the .f04 file.

Summary of Physical File Information

This summary table describes the physical files used for the DBsets. A sample of this table, located near the top of the .f04 file, is shown below.

<table>
<thead>
<tr>
<th>ASSIGNED PHYSICAL FILE NAME</th>
<th>RECL (BYTES)</th>
<th>MODE</th>
<th>FLAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>/tmp/65872_57.SCRATCH</td>
<td>8192</td>
<td>R/W</td>
<td>L</td>
</tr>
<tr>
<td>/tmp/65872_57.OBJSCR</td>
<td>8192</td>
<td>R/W</td>
<td>L</td>
</tr>
<tr>
<td>/tmp/65872_57.MASTER</td>
<td>8192</td>
<td>R/W</td>
<td>L</td>
</tr>
<tr>
<td>/tmp/65872_57.DBALL</td>
<td>8192</td>
<td>R/W</td>
<td>L</td>
</tr>
<tr>
<td>/tmp/65872_57.DBALL2</td>
<td>8192</td>
<td>R/W</td>
<td>L</td>
</tr>
<tr>
<td>/MSC.msc691/aix/SSS.MASTERA</td>
<td>8192</td>
<td>R/O</td>
<td>L</td>
</tr>
<tr>
<td>/MSC.msc691/aix/SSS.MSCOBJ</td>
<td>8192</td>
<td>R/O</td>
<td>L</td>
</tr>
</tbody>
</table>

FLAG VALUES ARE --
F   FFIO INTERFACE USED TO PROCESS FILE
H   HPIO INTERFACE USED TO PROCESS FILE
L   FILE HAS BEEN LOCKED
M   FILE MAPPING USED TO PROCESS FILE
R   FILE BEING ACCESSED IN ‘RAW’ MODE

** PHYSICAL FILES LARGER THAN 2GB FILES ARE NOT SUPPORTED ON THIS PLATFORM

In this summary, “ASSIGNED PHYSICAL FILENAME” is the physical FILENAME with any symbols translated; “RECL” is the record length in bytes; “MODE” is the file access mode, R/W is read-write mode, R/O is read-only mode. The “FLAGS” column will contain various letters depending on the capabilities of the platform and user requests, the text below the table indicates flag values that are possible on the specific platform.

In this example, an INIT statement was used to create the DBALL DBset with two files using the logical names DBALL and DBALL2.

Below the summary is a message indicating if large files (see “Using Databases” on page 113) are available on this platform. On AIX, HP-UX, IRIX64, and Solaris, the actual file system containing the file must support large files; this fact is not indicated in the message.
**Memory Map**

Immediately following the “Summary of Physical File Information” is a map showing the allocation of memory. This map is also described in “Managing Memory” on page 138.

<table>
<thead>
<tr>
<th><strong>MASTER DIRECTORIES ARE LOADED IN MEMORY.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>USER OPENCORE (HICORE) = 3804612 WORDS</td>
</tr>
<tr>
<td>EXECUTIVE SYSTEM WORK AREA = 78605 WORDS</td>
</tr>
<tr>
<td>MASTER (RAM) = 30000 WORDS</td>
</tr>
<tr>
<td>SCRATCH (MEM) AREA = 204900 WORDS { 100 BUFFERS}</td>
</tr>
<tr>
<td>BUFFER POOL AREA (GINO/EXEC) = 76183 WORDS { 37 BUFFERS}</td>
</tr>
<tr>
<td>TOTAL MSC NASTRAN MEMORY LIMIT = 4194300 WORDS</td>
</tr>
</tbody>
</table>

In this table “USER OPENCORE” is the amount of memory available to the module for computation purposes; “EXECUTIVE SYSTEM WORK AREA” is the space reserved for the executive system; “MASTER (RAM)” is the space reserved to cache datablocks from the MASTER DBset; “SCRATCH (MEM) AREA” is the space reserved to cache datablocks from the SCRATCH and SCR300 DBsets; “BUFFER POOL AREA” is the space reserved for the buffer pool; “TOTAL MSC Nastran MEMORY LIMIT” is the total space allocated to MSC Nastran’s open core using the “memory” keyword.

**Day Log**

The Day Log portion of the .f04 is a DMAP execution summary. This log, in table format, contains the vast majority of the information in the .f04. The beginning of the Day Log is shown below:

```
<table>
<thead>
<tr>
<th>DAY TIME</th>
<th>ELAPSED</th>
<th>I/O MB</th>
<th>DEL_MB</th>
<th>CPU SEC</th>
<th>DEL_CPU</th>
<th>SUB_DMAP/DMAP_MODULE</th>
<th>MESSAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:32:16</td>
<td>0:16</td>
<td>13.6</td>
<td>.3</td>
<td>.8</td>
<td>.0</td>
<td>SESTATIC</td>
<td>20 IFPL</td>
</tr>
<tr>
<td>10:32:16</td>
<td>0:16</td>
<td>13.7</td>
<td>.1</td>
<td>.8</td>
<td>.0</td>
<td>IFPL</td>
<td>29 IFP1</td>
</tr>
<tr>
<td>10:32:16</td>
<td>0:16</td>
<td>13.7</td>
<td>.0</td>
<td>.8</td>
<td>.0</td>
<td>IFPL</td>
<td>39 XSORT</td>
</tr>
</tbody>
</table>
```

In the Day Log, “DAY TIME” is the time of day of the entry; “ELAPSED” is the elapsed time since the start of the job; “I/O MB” is the megabytes of I/O to the databases since the start of the job; “DEL_MB” is the delta I/O since the previous entry; “CPU SEC” is the total CPU seconds since the start of the job; “DEL_CPU” is the delta CPU since the previous entry; “SUB_DMAP/DMAP_MODULE” indicates the DMAP statement being executed; and “MESSAGES” are any messages issued by the module. “BEGN” is the start of the module and “END” is the end.

**Note:**

1. The “I/O MB” value is computed by multiplying SYSTEM(85), which is incremented by one for each GINO I/O, by BUFFSIZE. This value will lose accuracy if the DBsets do not have the same BUFFSIZE.

2. If SYSTEM(84) is set to 0, the “I/O MB” column will be the number of GINO I/Os.
3. The “I/O MB” column will be scaled by gigabytes and a “G” will be appended after each number if the value is greater than or equal to 100 000.

4. Prior to Version 69, the “I/O SEC” value was computed by multiplying SYSTEM(85) by SYSTEM(84) (a pseudo-I/O rate).

User Information Messages 4157 and 6439

The UIM 4157 text provides decomposition estimates upon completion on the preface of the decomposition module. This message has a counterpart, UIM 6439, which provides actual information from the completed decomposition process. These two messages are interspersed within the Day Log at each decomposition. The following example is from a sparse decomposition.

The most important elements of the UIM 4157 message are the “MINIMUM MEMORY REQUIREMENT”, which is an estimate of the user open core memory that will allow the decomposition to run, but with heavy spilling to disk. The “MEMORY REQR’D TO AVOID SPILL” will allow the decomposition to run in “in core”, i.e., without spilling to disk. These two values represent the extremes of memory requirements, the memory for optimal CPU performance is between the two. The “ESTIMATED MAXIMUM FRONT SIZE”, a function of the model, affects the memory estimates; the minimum memory is a function of the front size, and the memory to avoid spill is a function of the square of the front size. The “NUMBER OF NONZEROES” is the size of the input matrix, multiply this value by 8 to estimate the size of the input file in bytes. The sum of “EST. INTEGER WORDS IN FACTOR” and “EST. NONZERO TERMS” is the size of the output matrix, multiply the integer value by 8 on NEC and 4 on other machines, and the nonzero value by 8 to estimate the size of the output file in bytes. The “RANK OF UPDATE” is the number of rows that will be simultaneously updated during the decomposition. This value is set by either the “rank” keyword or SYSTEM(205).

Note: Setting SYSTEM(69)=64 will cause MD Nastran to terminate after printing UIM 4157. This can be useful for determining a job’s memory and disk space requirements.

In UIM 6439, “SPARSE DECOMP MEMORY USED” states the actual memory used in the decomposition process. Based on the execution of the module, the “SPARSE DECOMP SUGGESTED MEMORY” will result in optimal throughput performance.
Memory and Disk Usage Statistics

These tables are written after the job has completed, and indicate the maximum memory used by any sparse numerical module and the maximum disk used by any module during the job. A sample follows.

<table>
<thead>
<tr>
<th>SPARSE SOLUTION MODULES</th>
<th>MAXIMUM DISK USAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIWATER WORDS</td>
<td>SUB_DMAP NAME</td>
</tr>
<tr>
<td>DAY_TIME</td>
<td>DMAP MODULE</td>
</tr>
<tr>
<td>517786</td>
<td>04:35:44</td>
</tr>
<tr>
<td>SEKRRS</td>
<td>18 DCMP</td>
</tr>
<tr>
<td>15.625</td>
<td>04:35:48</td>
</tr>
<tr>
<td>SESTATIC</td>
<td>186 EXIT</td>
</tr>
</tbody>
</table>

In the left hand table, “HIWATER WORDS” is the maximum amount of open core used by certain sparse numerical modules; “DAY_TIME” is the time of day the module ran. “SUB_DMAP NAME” is the name of the SUBDmap; “DMAP MODULE” indicates the line number and module name that made the maximum request. Similarly, in the right hand table, “HIWATER (MB)” is the maximum amount of disk space used by any module; “DAY_TIME” is the time of day the module ran. “SUB_DMAP NAME” is the name of the SUBDmap; “DMAP MODULE” indicates the line number and module name that made the maximum request.

Database Usage Statistics

These statistics, provided in table format, summarize the I/O activity for the DBsets.

<table>
<thead>
<tr>
<th>DATABASE USAGE STATISTICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGICAL DBSETS</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>ALLOCATED (BLOCKS)</td>
</tr>
<tr>
<td>BLOCKSIZE (WORDS)</td>
</tr>
<tr>
<td>USED (BLOCKS)</td>
</tr>
<tr>
<td>%</td>
</tr>
<tr>
<td>FILE</td>
</tr>
<tr>
<td>ALLOCATED (BLOCKS)</td>
</tr>
<tr>
<td>HIWATER (BLOCKS)</td>
</tr>
<tr>
<td>HIWATER (MB)</td>
</tr>
<tr>
<td>I/O TRANSFERRED (GB)</td>
</tr>
</tbody>
</table>

This statistical table contains two parallel tables. The “LOGICAL DBSETS” table lists each DBset while the “DBSET FILES” tables lists the component files of the DBset. In these tables, “DBSET” is the name of the DBset; “ALLOCATED” is the MD Nastran DBset size limit in blocks; “BLOCKSIZE” is BUFFSIZE of the DBset minus one. “USED (BLOCKS)” and “%” are the number of blocks and percent of the DBset actually used; “FILE” is the file’s logical name associated with the DBset to the left. Additionally, “ALLOCATED” is the number of blocks allocated by MD Nastran to the file; while “HIWATER (BLOCKS)” and “HIWATER (MB)” are the number of blocks and megabytes actually used in the file. “I/O TRANSFERRED” is the amount of I/O to the file. The last line of the DBset Files table lists the “TOTAL I/O TRANSFERRED”.

In this example, the MASTER and OBJSCR DBsets are each composed of one file. The DBALL DBset is composed of two files, DBALL and DBALL2; and the SCRATCH DBset has three components, MEMFILE, SCRATCH, and SCR300.
This table can be used to determine if the DBsets and files are appropriately sized and the amount of I/O activity associated with each file. Best elapsed time performance can be obtained if the files with the greatest activity are on different physical devices (and better yet, separate I/O controllers or busses).

**Summary of Physical File I/O Activity**

This summary describes the physical file I/O for each database file.

<table>
<thead>
<tr>
<th>ASSIGNED PHYSICAL FILENAME</th>
<th>RECL (BYTES)</th>
<th>READ/WRITE COUNT</th>
<th>MAP WSIZE (NUM)</th>
<th>MAP COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>/tmp/65872_57.SCRATCH</td>
<td>8192</td>
<td>1</td>
<td>128KB (4)</td>
<td>1</td>
</tr>
<tr>
<td>/tmp/65872_57.OBJSCR</td>
<td>8192</td>
<td>378</td>
<td>128KB (4)</td>
<td>24</td>
</tr>
<tr>
<td>/tmp/65872_57.MASTER</td>
<td>8192</td>
<td>1247</td>
<td>128KB (4)</td>
<td>11</td>
</tr>
<tr>
<td>/tmp/65872_57.DBALL</td>
<td>8192</td>
<td>24</td>
<td>128KB (4)</td>
<td>1</td>
</tr>
<tr>
<td>/tmp/65872_57.SCR300</td>
<td>8192</td>
<td>1</td>
<td>128KB (4)</td>
<td>1</td>
</tr>
<tr>
<td>/MSC.msc691/aix/SSS.MASTERA</td>
<td>8192</td>
<td>162</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>/MSC.msc691/aix/SSS.MSCOBJ</td>
<td>8192</td>
<td>202</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

In this summary, “ASSIGNED PHYSICAL FILENAME”, “RECL”, and “MAP WSIZE and NUM” are repeated from the “Summary of Physical File Information” table. “READ/WRITE COUNT” is the number of GINO reads and writes that were performed on the file and “MAP COUNT” is the number of times the map window had to be remapped (these columns are only present on systems supporting mapped I/O).

This summary can be used to tune I/O performance. For mapped I/O systems, if the map count approaches the number of reads and writes, the map size and/or the number of maps should be increased. Increasing the number of maps is suggested if a module simultaneously accesses more data blocks or matrices in a file than there are windows. Increasing the size of the windows is suggested if a file contains very large data blocks or matrices. Best elapsed time performance, with or without mapping, can be obtained if the files with the greatest activity are on different physical devices (and better yet, separate I/O controllers or busses).
Running a Job on a Remote System

The nastran command offers a mechanism to run simple jobs on a computer other than the computer you are currently logged onto via the "node" keyword. (page 321). In the descriptions that follow, the "local" node or system is the computer you issue the nastran command on; the "remote" node or system is the computer named by the "node" keyword, i.e., the system where the MD Nastran analysis will run.

The method used to communicate between the local and remote nodes depends on the operating system on the remote node:

- If the remote node is a UNIX system (or a similar system such as Linux), the "rsh/rcp" communications programs may be used.
- If the remote node is a Windows system (i.e., a system running Windows NT/2000/XP), the "MSCRmtCmd/MSRmtMgr" communications programs must be used. (See "Installing/Running MSCRmtMgr" on page 161.)

Following are some general requirements for running remote jobs:

- MD Nastran must be properly installed on the remote system.
- The input data file must be accessible on the local host.
- INCLUDE files must be local-to, or visible-from, the remote system unless the "expjid" keyword is used (or taken by default).
- All default output files, i.e., those without ASSIGN statements, will be written to a directory accessible to the local host.
- In a restart, i.e., a job that uses an existing database, the DBsets must be local-to, or visible-from, the remote system.

If the rsh/rcp communications programs are to be used:

- You must have "remote execution" privileges on the remote system. That is, a password must not be required to execute a remote copy (rcp) or remote shell (rsh or remsh) command. See your system administrator for information on this. You can test this with the following command:

  remsh <node> [-l <username>] date   # HP-UX only
  rsh <node> [-l <username>] date     # All others

  where "<node>" is the name of the remote node and "<username>" is an alternative username on the remote system if your current username is not valid. For example:

  rsh node1 date

  The output from the above command should be a single line containing the current date on node1 in a format similar to:

  Tue Jul 16 15:05:47 PDT 2002
If any other output is present, please determine the source of the output and correct the problem. If you cannot eliminate the output, you will not be able to use the remote execution capabilities of the nastran command for the specified remote node.

If the MSCRmtCmd/MSCRmtMgr communications programs are to be used:

- The MSCRmtMgr program must be running on the remote system, either as an installed and started service or as a console mode program running in a Command Prompt window (started with the "-noservice" command line option). You can test this with the following command:

```
<instdir>\bin\md2007 mscrmtcmd -h <node> -i (from Windows)
<instdir>/bin/md2007 MSCRmtCmd -h <node> -i (from UNIX)
```

where "<instdir>" is the installation directory for the local MD Nastran installation and "<node>" is the name of the remote node. For example:

```
c:\msc\bin\md2007 mscrmtcmd -h node1 -i (From Windows)
/msc/bin/md2007 MSCRmtCmd -h node1 -i (From UNIX)
```

The output from the above command(s) should be a single line containing configuration information for node1 in a format similar to:

```
1@2@"C:/WINNT40/system32/cmd.exe" (If node1 is Windows)
2@1@"/bin/ksh" (If node1 is UNIX)
2@2@"/bin/bsh" (If node1 is Linux)
```

If any other output is present or if the request fails, please determine the source of the output and correct the problem. If you cannot eliminate the output, you will not be able to use the remote execution capabilities of the nastran command for the specified remote node.

**Note:** Recall that, for remote UNIX nodes, remote executions do not run a "login" shell. That is, your ".profile" or ".login" script is not executed. This is true regardless of the communications programs (rsh/rcp or MSCRmtCmd/MSCRmtMgr) being used.

If the node specified by the "node" keyword is the same as the local node, the "node" keyword is ignored and processing will continue as if "node" had not been defined.

If the local node is a Windows system and if rsh/rcp is to be used, the nastran command attempts to locate the rsh.exe program in the
directory before using the search path because "rsh" may be a "restricted shell" program installed as part of a UNIX commands package such as MKS Toolkit.

If the local node is a Windows system, MSC supplies a replacement for the standard Windows rsh.exe program, MSCrsh.exe. This program is a full functional replacement for the standard rsh.exe program with the addition of support for "Toolkit" mode. Toolkit mode requires that the stdin and stdout streams used for communications between the local and remote systems be "binary" data streams. The standard Windows rsh.exe program treats these streams as "text" streams, which is not compatible with Toolkit mode. See the MSC.visualNastran Toolkit User's Guide for more information on Toolkit mode. The nastran command will use MSCrsh.exe instead of rsh.exe if it can be found. Also, there are some circumstances where the Windows rsh.exe and rcp.exe commands do not perform reliably. MSC has found problems are more likely to occur on high-speed networks and/or when running Windows NT 4.0 SP4. Windows NT 4.0 SP5 and later systems offer more reliable rsh and rcp commands. Alternatively, use the MSCrsh.exe program.

MD Nastran does not support remote systems running the Windows 95/98/ME operating systems.

The MSCRmtCmd/MSCRmtMgr communications programs may also be used when the remote node is a UNIX system. However, there are no inherent advantages over using the rsh/rcp programs.

When running a remote job, nastran keywords are processed on both the local and remote systems. Keywords that control the job's output and interaction with you are processed on the local system. Keywords that specify information about the remote system's installation or that affect the actual analysis are processed on the remote system. MSC suggests that those keywords that specify information about the remote system's installation be defined in conditional Initialization or Runtime Configuration File sections based on the "node" keyword and that those keywords that specify information about the local system's installation be defined in conditional Initialization or Runtime Configuration File sections based the "s.hostname" keyword.

The following table lists some of the keywords that affect remote processing. These keywords are described in detail in “Keywords” on page 292.

Table 5-2  Remote Processing Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>append</td>
<td>Requests the .f06, .f04 and .log files to be concatenated.</td>
</tr>
<tr>
<td>batch</td>
<td>Requests the job is to be run in the background. (UNIX only.)</td>
</tr>
<tr>
<td>delete</td>
<td>Unconditionally deletes files after job completion.</td>
</tr>
<tr>
<td>display</td>
<td>Specifies the DISPLAY for XMONAST. (UNIX only.)</td>
</tr>
<tr>
<td>expjid</td>
<td>Specifies data file expansion on the local system.</td>
</tr>
<tr>
<td>lsymbol</td>
<td>Specifies logical symbol values to be used on the local system.</td>
</tr>
<tr>
<td>ncmd</td>
<td>Specifies an alternate notification command.</td>
</tr>
<tr>
<td>node</td>
<td>Specifies the node the job will be processed on.</td>
</tr>
</tbody>
</table>
The "sdirectory"/"rsdirectory" keywords are special, as the command line, RC files on the current host and RC files on the remote host will all be considered when establishing a scratch directory. As part of its processing, the nastran command may generate temporary files on both the local system (e.g., as part of "expjid" processing) and on the remote system (e.g., the location where temporary RC files are placed and where output files are generated). These files are placed in the "scratch" directories on the local and remote systems. If the "rsdirectory" keyword is not specified, this directory will also be used on the remote system. (Note that this is not possible if the systems are running different types of operating systems.) All other keywords specifying path/file names will only be scanned on the remote system and must specify path/file names appropriate for that system.

Once "node=remotenode" is processed, the following processing takes place:

1. Process the RC files on the local system if the "version" keyword has been defined in the command initialization file or the command line.
2. Process the RC file specified by the "rcf" keyword if it was defined on the command line.
3. Reprocess the command initialization file and any RC files if any contained conditional sections. (See “Resolving Duplicate Parameter Specifications” on page 65 for a more complete description of Command Initialization file and Runtime Configuration file processing.)

4. Determine the full pathname of the input file so that its visibility from remotenode can be tested.

5. If the "rmtdeny" utility, i.e., install-dir/md2007/arch/rmtdeny on UNIX and install-dir\md2007\arch\rmtdeny.exe on Windows, exists and is executable, run it and examine its output. If remotenode is listed, display an error and cancel the job.

6. If the "rmtaccept" utility, i.e., install-dir/md2007/arch/rmtaccept on UNIX and install-dir\md2007\i386\rmtaccept.exe on Windows, exists and is executable, run it and examine its output. If remotenode is not listed, display an error and cancel the job.

7. Verify that remotenode exists and you are able to run a command on that system. This process also determines the communications programs to be used and the remotenode operating system type. Although the nastran command can determine this information dynamically, processing may be much faster if you specify the proper information using the "rrmtuse" and/or "rostype" keywords (for example, in an INI or RC file conditional section). The information is set as follows:

8. If the "MSCRmtCmd/MSCRmtMgr" communications programs are selected (by either defining "rrmtuse=mscrmtcmd" or defining "rostype=windows" or dynamically selected), the remotenode operating system type is determined automatically.

9. If the "rsh/rcp" communications programs are selected (by either defining "rrmtuse=rsh" or defining "rostype=unix" or dynamically selected), the remotenode operating system type is assumed to be UNIX.

10. If both the local node and remote node operating system types are UNIX, create a "touch" file in the specified output file so that its visibility from remotenode can be tested.

11. If "rmdirctory" has not been defined or contains multiple values, set it as follows:

12. If "rmdirctory" has been defined but contains multiple values, change its value to the first value.

13. If "sdirectory" has not been set and the local system is Windows, set "sdirectory" to "c:\tmp" if the remotenode operating system is Windows and to "\tmp" otherwise.

14. If "sdirectory" has not been set and the local system is UNIX, set "sdirectory" to "c:\tmp" if the remotenode operating system is Windows and to the path defined by the "TMPDIR" environment variable otherwise.

15. If "sdirectory" has been set "rmdirctory" to the first (or only) value defined by "sdirectory".

16. Ensure "scratch=no" was set if the "dbs" keyword was set.

17. If the "rcmd" keyword was specified, attempt to execute that command on remotenode, displaying an error and canceling the job if it fails. Otherwise, attempt to execute the pathname of the current nastran command on remotenode. If it fails, attempt to run the basename of the current nastran command on remotenode. Display an error and cancel the job if both checks fail.

18. Run the remote nastran command identified in the previous step to determine:
19. If the directory specified by "rsdirectory" is valid.
20. If "scratch=no" is set, if the directory specified by "dbs" is valid.
21. The numeric format of the remote system.
22. The location of the TRANS program on the remote system.
   If both the local and remote modes are UNIX, the following tests are also made:
   If the input data file is visible.
   If the "touch" file is visible.
   If "expjid" was specified, if the "expjid" expand directory is visible.
23. Display an error and cancel the job if an "rsdirectory" was identified on the command line or in
    a local RC file, but does not exist on the remote node.
24. Display an error and cancel the job if the "dbs" directory was identified on the command line or
    in a local RC file, but does not exist on the remote node.
25. If a "touch" file was created above, delete it.
26. Make sure a RECEIVE executable exists on the local node if "trans=yes" was specified or
    "trans=auto" was specified and the numeric formats of the local and remote nodes differ.
27. Copy the input data file (or the expanded file if "expjid" processing was performed) to the remote
    system's scratch directory if the remote host could not see the file or if the local and remote
    operating system types are different.
28. Set the "out" to the remote system's scratch directory if the remote host could not see the output
    directory or if the local and remote operating system types are different.
29. Copy the remaining keywords on the command line that were not processed to a local RC file in
    the scratch directory on the remote node.
30. Run the job on the remote node.
31. Process the "old" and "oldtypes" keywords on the local node.
32. If the output directory was not visible from the remote node or if the local and remote operating
    system types are different, copy the output files (.f04, .f06, .log, .ndb, .pch, .plt) to the directory
    specified by the "output" keyword and delete the files from the remote node.
33. Process the "append" keyword on the local node.
34. If the output directory was not visible from the remote node or if the local and remote operating
    system types are different and if an .xdb file was created on the remote node, run the RECEIVE
    program if required by the "trans" keyword or copy the .xdb file to the directory specified by the
    "output" keyword and delete the .xdb file from the remote node.
35. Process the "notify" keyword on the local node.
Once the job has completed, the .f06, .f04, .log, .ndb, .op2, .plt, .pch and .xdb files will be present as if the job were run locally. Binary files, i.e., .op2 and .plt, will only be usable on the local node if the local and remote nodes use the same numeric format. The .xdb file will be translated via TRANSMIT and RECEIVE unless “trans=no” was specified.

**Note:** No attempt is made to copy DBset files between the local and remote systems. If this is required, you must handle this yourself and set the "dbs" keyword as required.

Several examples are provided.

```
md2007 nastran example node=othernode batch=no   (UNIX)
md2007 nastran example node=othernode            (Windows)
```

This job will run on node "othernode". The .f04, .f06, .log, .pch, .plt, and .xdb files will be brought back to the current node as if the job were run locally. (Note that Windows systems do not support the use of the "batch" keyword.)

```
md2007 nastran example node=othernode rcmd=/some/path/bin/nast2007
```

This job will also run on "othernode" (assumed to be a UNIX system) but the path to the nastran command has been specified explicitly.

```
md2007 nastran example node=othernode rcmd=d:/a/path/bin/nast2007
```

This job will also run on "othernode" (assumed to be a Windows system) but the path to the nastran command has been specified explicitly. Note the use of forward slashes ('/') in the "rcmd" value. If the local system is a Windows system, either forward slash ('/') or back slash ('\') characters may be used. If the local system is a UNIX system, forward slash ('/') characters must be used or the entire rcmd specification must be enclosed in quotes to prevent the shell from interpreting the back slash ('\') characters as "escape" characters. When the rcmd specification is used on "othernode", the forward slash characters will be changed to back slash characters as needed.

```
md2007 nastran example node=othernode dbs=/dbs
```

This job will also run on "othernode" (assumed to be a UNIX system) but will use the "/dbs/example.*" DBset files. These files must exist on "othernode" prior to running this command if this is a restart job. Once the job completes, the DBset files will be left as is.
This example will run a job on UNIX node "uxsrv" using the nastran command in the default PATH with all scratch files on the local system residing in /scratch and all scratch files on the remote system residing in /tmp. Note that the "sdir" and "rsdir" keywords could have been set in an RCF file.

This job will use the default scratch directory on "uxsrv".

This job will use the nastran command /msc/bin/nast2007 on "uxsrv".

### Installing/Running MSCRmtMgr

The MSCRmtMgr program provides the server-side communications support used by the MSCRmtCmd client-side program. That is, MSCRmtMgr provides functions equivalent to the UNIX rshd/rexec services. MSCRmtMgr is primarily intended for use on Windows NT/2000/XP systems.

For Windows systems, MSCRmtMgr may only be run on Windows NT/2000/XP. Support for the use of Windows 95/98/ME systems as remote nodes is not available. MSCRmtMgr may be run as a command-mode program or as a service, providing the same functionality in either case.

#### Running MSCRmtMgr as a Command-mode Program

This is the simplest way to run MSCRmtMgr but is the least flexible in that MSCRmtMgr must be restarted every time the operating system is restarted. In this mode, MSCRmtMgr is started in a "Command Prompt" window that is left open as long as the Windows system is to act as a server. The command is:

```
<instdir>\bin\md2007 MSCRmtMgr -noservice
```

The "-noservice" operand is required and tells MSCRmtMgr that it is not to attempt to run as a Windows service program. In this mode, MSCRmtMgr will run using the authorization and access control provided by the currently logged on user. MSCRmtMgr may be terminated using <CNTL−C> or by using the Task Manager.

#### Installing and Running MSCRmtMgr as a Windows Service

The MSCRmtMgr program may also be run as a Windows Service program. Doing this requires the Microsoft Windows Resource Kit SC.exe (Services Control) utility program, available from Microsoft, and run from a command prompt. Install MSCRmtMgr as a service using the following command:
sc create MSCRmtMgr type= own start= auto
binpath= <instdir>\md2007\i386\mscrmtmgr.exe
where:

type= own  indicates that MSCRmtMgr is to be run in its own process
start= auto indicates that MSCRmtMgr is to be automatically started at boot time.
    This may also be specified as "start= demand".
binpath= ... specifies the full path to the MSCRmtMgr program.

Note the blanks between the equal sign following the option and the actual value. These blanks are required.

Once MSCRmtMgr has been installed as a service, it may be started or stopped using the Services Administrative Tools program or using SC.exe as follows:

To start MSCRmtMgr:
sc start MSCRmtMgr -service -name "MSCRmtMgr"

To stop MSCRmtMgr:
sc stop MSCRmtMgr

If MSCRmtMgr is no longer required, it may be deleted as a service using SC.exe as follows:
sc delete MSCRmtMgr

Note that this will remove MSCRmtMgr as a service but will not actually delete the executable. It may be reinstalled as a service using the command described above.
Running Distributed Memory Parallel (DMP) Jobs

MD Nastran offers the ability to run certain solution sequences (see the MD Nastran Release Guide) in parallel using the Message Passing Interface (MPI), an industry-wide standard library for C and Fortran message-passing programs. MPI programs can be run on SMP computers, NUMA computers, distributed computers, and any collection of computers supported by the MPI package.

**Note:** Further information on the MPI standard can be obtained online at the URL

http://www.mpi-forum.org

Online documentation for the vendor’s MPI products may be found at the following URLs

Fujitsu Primepower
  http://www.fujitsu.com

HP Alpha UNIX
  http://www.hp.com/techservers/software/cmpisrc.html

HP
  http://www.hp.com/techservers/software/cmpisrc.html

IBM
  http://www.ibm.com

Linux
  http://www.lam-mpi.org/lam
  http://www.hp.com/go/hpmpi

Microsoft
  http://www.microsoft.com

NEC
  http://www.nec.com

SGI
  http://www.sgi.com/products/evaluation

Sun
  http://www.sun.com/products/hpc

These are not MSC.Software Corporation sites and MSC has no control over the site’s content. MSC cannot guarantee the accuracy of the information on these sites and will not be liable for any misleading or incorrect information obtained from these sites.
In most cases, MD Nastran uses the hardware vendor’s MPI implementation. While this usually results in the highest performance levels, it also presents a limitation—a DMP job can only run on computers supported by the vendor’s MPI package. As an example, you cannot use a mixture of IBM and Sun machines to run a single MD Nastran DMP job.

The following table lists the hardware and software prerequisites for every host that will take part in running an MD Nastran DMP job:

Table 5-3  DMP System Prerequisites

<table>
<thead>
<tr>
<th>Platform</th>
<th>Processor</th>
<th>OS</th>
<th>MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>Any</td>
<td>AIX 5.1</td>
<td>POE 3.2</td>
</tr>
<tr>
<td>AMD/Intel Linux</td>
<td>Any</td>
<td>2.418</td>
<td>LAM/MPI 7.0.4</td>
</tr>
<tr>
<td>AMD/Intel Windows</td>
<td>Opteron/EM64T</td>
<td>Windows Server 2003 x64 Edition with CCS</td>
<td>Microsoft MS-MPI</td>
</tr>
<tr>
<td>HP Alpha UNIX</td>
<td>Any</td>
<td>UNIX 5.1</td>
<td>Compaq MPI 1.9.6</td>
</tr>
<tr>
<td>HP-UX</td>
<td>PA-RISC 2.0</td>
<td>HP-UX 11.0, 64-bit</td>
<td>No additional software is required.</td>
</tr>
<tr>
<td>IRIX64</td>
<td>R8K, R10K, R12K</td>
<td>IRIX64 6.4</td>
<td>Array Services 3.3, MPT 1.3</td>
</tr>
</tbody>
</table>
Table 5-3  DMP System Prerequisites (continued)

<table>
<thead>
<tr>
<th>Platform</th>
<th>Processor</th>
<th>System Prerequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solaris</td>
<td>Processor</td>
<td>UltraSPARC</td>
</tr>
<tr>
<td></td>
<td>OS</td>
<td>Solaris</td>
</tr>
<tr>
<td></td>
<td>MPI</td>
<td>HPC 5.0</td>
</tr>
<tr>
<td>SUPER-UX</td>
<td>Processor</td>
<td>Any</td>
</tr>
<tr>
<td></td>
<td>OS</td>
<td>SUPER-UX 9.1</td>
</tr>
<tr>
<td></td>
<td>MPI</td>
<td>MPI/SX 5.0</td>
</tr>
</tbody>
</table>

In the descriptions that follow, the “local” node is the computer you issue the nastran command on, the “master” node is the first computer named by the “hosts” keyword, the “slave” nodes are the remaining systems listed in the “hosts” list.

The following are some general comments and requirements for running MD Nastran DMP jobs:

1. MD Nastran must be properly installed on, or accessible to, all the hosts listed by the “hosts” keyword.

2. With the exception of HP-UX systems, the MPI program start command (“dmprun” on HP-UX, “poe” on AIX, “mprun” on Sun, “mpiexec” on Windows, and “mpirun” on others) must be available on the PATH of the local host.

3. For Linux/UNIX systems you must have r-command access to each system you want to access in a distributed job.

You can test this with the following command:

```
remsh <node> [-l <username>] date # HP-UX only
rsh <node> [-l <username>] date # All others
```

where “<node>” is the name of the node and “<username>” is an alternate username on the remote system if your current username is not valid. For example:

```
rsh node1 date
```

The output from the above command should in a single line containing the current date on node1 in a format similar to:

```
Thu Sep 30 13:06:49 PDT 1999
```
If any other output is present, please determine the source of the output and correct the problem. If you cannot eliminate the output, you will not be able to use the distributed execution capabilities of MD Nastran.

4. On IRIX64, the use of multiple hosts requires Array Services to be configured for each possible host.

5. On AIX, you must set “resd=no” and “eulib=ip” on the command line or in an RC file. If not, the job may fail to start with the following error message:

   ERROR: 0031-149 Unable to load shared objects objects required for LoadLeveler

The following system error may be reported when the distributed job has completed:

   ERROR: 0031-636 User requested child or EOF termination of pm-command

   It can be ignored

6. On AIX running on a cluster of workstations, you must set “euidevice=ip” and “resd=yes” on the command line or in an RC file if you run on a cluster of workstations. Also, in AIX systems, the current directory must be visible to all nodes.

7. The input data file must be accessible on the local host.

8. For Windows systems, a working directory accessible to all nodes must be available for use by the CCS Job Scheduler.

9. On Windows systems, all pathnames will be converted, if necessary, to Universal Naming Convention (UNC) format for all accessibility tests. If a pathname has no equivalent UNC name, it will be considered “not accessible”. Also, if “ccsnodesmust=no” is specified, the input data file and output directory must be visible from all nodes specified by the “hosts” keyword.

10. On all systems, you must have write access to the output directory.

11. INCLUDE files must be local-to, or visible-from, each host.

12. All default output files, i.e., those without ASSIGN statements, will be written to a directory accessible to the local host.

13. The scratch directory can be a global or local file system. MSC recommends the scratch directory be local to each host, i.e., you specify per-host “sdirectory” values. See “Managing Host-Database Directory Assignments in DMP Jobs” on page 174 for more information.

14. The pathname of the nastran command must be the same on all hosts, or on the default PATH of each host, used in the analysis.

15. For Linux/Unix systems you must have “remote execution” privileges on all the hosts listed by the “hosts” keyword. That is, a password must not be required to execute a remote copy (rcp) or remote shell (rsh or remsh) command. See your system administrator for information on this.
16. For Windows systems, you must be running on the “head” node of the Windows Compute Cluster and all of the hosts listed by the “hosts” keyword must be nodes in the same Compute Cluster. Note, however, that the “head” node of the Compute Cluster does not have to be in the list of hosts specified by the “hosts” keyword.

17. On Windows systems, commands and jobs, including those run on remote hosts, are run using the Windows CCS “QueueJob” interface, waiting for the queued job to complete before continuing, since this interface does not require any special privileges. The only exception to this is when all DMP tasks are to be run on the same node and that node is the “head” node of the CCS cluster. In this case, the CCS Job Scheduler is not used, instead the DMP tasks are run using the mpiexec command directly.

18. If you execute a restart, you must specify the identical values for “dmparallel” and “hosts” as were used on the cold start.

19. In a restart, i.e., a job that uses an existing database, the DBsets must be local-to, or visible-from, the remote system.

**Note:** For Linux/UNIX systems recall that remote executions do not run a “login” shell. That is, your “.profile” or “.login” script is not executed.

When running a DMP job, nastran keywords are processed on both the local and master/slave systems. Keywords that control the job’s output and interaction with you are processed on the local system. These are:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapter_use</td>
<td>AIX: Specifies use of adapter by job.</td>
</tr>
<tr>
<td>append</td>
<td>Requests the .f06, .f04, and .log files to be concatenated.</td>
</tr>
<tr>
<td>bypass_off</td>
<td>IRIX64: Disables the HIPPI bypass, forcing all host-to-host messages through TCP.</td>
</tr>
<tr>
<td>ccsnodesmust</td>
<td>Windows: Specifies whether the hosts listed by the “hosts” keyword must be allocated by the CCS scheduler or are only a list of possible nodes.</td>
</tr>
<tr>
<td>ccsdir</td>
<td>Windows: Specifies a network-visible working directory for the Windows CCS Job Scheduler</td>
</tr>
<tr>
<td>cpu_use</td>
<td>AIX: Specifies use of CPU by job.</td>
</tr>
<tr>
<td>display</td>
<td>Linux/UNIX: Specifies the DISPLAY for XMONAST.</td>
</tr>
<tr>
<td>euidevice</td>
<td>AIX: Specifies adapter device name.</td>
</tr>
<tr>
<td>euilib</td>
<td>AIX: Specifies adapter library.</td>
</tr>
<tr>
<td>hostovercommit</td>
<td>Requests more tasks per host than CPUs.</td>
</tr>
<tr>
<td>hosts</td>
<td>Specifies list of hosts to use. Separate hosts with a comma or with the PATH separator, i.e., “;” on Linux/UNIX and “;” on Windows.</td>
</tr>
</tbody>
</table>
The “sdirectory” keyword is special, as the command line, RC files on the current host, and RC files on the each master and slave host will all be considered when establishing a scratch directory. All remaining keywords are only scanned on the master and slave systems.

Once “dmparallel=number” is processed, the following processing takes place:

1. Process the RC files on the local system if the “version” keyword has been defined in the command initialization file or the command line.

2. Process the RC file specified by the “rcf” keyword if it was defined on the command line.
3. Determine the full pathname of the input file so that its visibility from the master and each slave host can be tested. For Windows, this full pathname will be converted to UNC format, if necessary.

4. Create a “touch” file in the specified output directory so that its visibility from the master and each slave host can be tested.

5. If the “dmpdeny” utility, i.e., install-dir/msc2007/arch/dmpdeny, exists and is executable, run it, and save its output.

6. If the “dmpaccept” utility, i.e., install-dir/md2007/arch/dmpaccept, exists and is executable, run it, and save its output.

7. Ensure “scratch=no” was set if the “dbs” keyword was set.

8. Determine every possible pairing of host and sdirectory by scanning each list in a round-robin order. That is, the first host is paired with the first sdirectory, the second host with the second sdirectory, and so on.

9. For Windows systems, if any host specified using the “hosts” keyword is not the head node, then locate a working directory to be used by the CCS Job Scheduler. This working directory must be network-accessible and must be available, in write-mode, to all of the hosts specified using the “hosts” keyword. The directory is located as follows:
   a. If there is a directory specified using the “ccstempdir” keyword, it is used, converting the specified directory to UNC format, if necessary. An error will be generated if this directory cannot be converted to UNC format and processing will be terminated.
   b. If the “ccstempdir” keyword was not specified, a network-accessible directory will be searched for using the following search order:
      i. Check the directory specified by the “sdirectory” keyword.
      ii. Check the directory portion of the location specified by the “out” keyword, either the value explicitly set by the user or the internally generated location.
      iii. Check the current directory.
      iv. Check the directory specified by the “TEMP” environment variable.
      v. Check the directory specified by the “TMP” environment variable.
      If no network-accessible directory could be located using this search, an error will be generated and processing will be terminated.

Any temporary files needed by the CCS Job Scheduler and by the mpiexec program will be stored in this directory.

10. Execute the following steps for each host-sdirectory pair determined above until host-sdirectory pairs have been assigned to each of the tasks requested by the “dmparallel” keyword or no more host-sdirectory pairs are available. Steps a. through i. are executed only once per host-sdirectory pair.

   a. Verify that host exists. For Linux/UNIX systems, verify that you are able to run a command on that system. For Windows systems, verify that host is a member of the Compute Cluster.
b. If the “rcmd” keyword was specified, attempt to execute that command on host, display an error and cancel the job if it fails.

c. Otherwise attempt to execute the pathname of the current nastran command on host. If it fails, attempt to execute the basename of the current nastran command on host. Display an error and cancel the job if both checks fail. For Windows systems, these pathnames are converted to UNC format, if necessary, before they are used.

d. For Windows systems, if the “rcmd” keyword is specified, its value must be suitable for all specified hosts. Normally, this means it must be specified in UNC format. It is not automatically converted to UNC format.

e. Run the remote nastran command identified in the previous step to determine: if the input data file is visible; if the “touch” file is visible; if the “sdirectory” (if identified on the local system) exists; if the “dbs” directory (if identified on the local system) exists; the “sdirectory” value in the RC files defined on host; and finally the numeric format of host.

f. Drop this host-sdirectory pair from further consideration if a scratch directory was identified on the command line or in a local RC file and that specified a list of directories, but the sdirectory does not exist on host.

g. For Windows, if “ccsnodesmust=no” is specified, drop this host-sdirectory pair from further consideration if either the input file or the output directory is not visible on host.

h. Display an error and cancel the job if the numeric format of host differs from the numeric format of the local host or if the operating system type of host differs from the operating system type of the local host, i.e., you cannot mix Linux/UNIX hosts with Windows hosts.

i. Display an error and cancel the job if the directory specified by a “dbs” keyword on the command line or in a local RC file does not exist on host.

j. Assign the current host-sdirectory pair to the next task; save the “sdirectory” value and the per-host visibility flags and “rcmd” value.

11. Display an error and cancel the job if one or more of the tasks requested by the “dmparallel” keyword have not been assigned.

12. Delete the “touch” file created above.

13. For Linux/UNIX systems, the remaining steps are done in a background process (at possibly some time later) if “batch=yes” or “after” was specified. For Windows, if “ccsnodesmust=no” is specified, note that none of the copy steps that send files to or from hosts are done because all files must be visible.

a. Copy the input data file to the scratch directory of any host that could not see the input data file.

b. Set “out” to the host-specific scratch directory value of every host that could not see the output directory.

c. Copy the remaining keywords on the command line that were not processed to a local RC file in the “out” directory. Copy this RC file to the host-specific scratch directory on any host that could not see the output directory.

d. Process the “old” and “oldtypes” keywords on the local node.
e. Run the DMP job using the system’s MPI startup command. Note that each task will write to task-specific names. For Windows systems, the MPI startup command will be scheduled using the CCS job scheduler unless all tasks are to be run on the same node and that node is the “head” node.

f. If the master task could not see the output directory, copy the output files (.f04, .f06, .log, .ndb, .pch, .plt) from the master node to the output directory (the directory specified by the “output” keyword) and delete the files from the master node.

g. Process the “append” keyword on the local node.

h. For Linux/UNIX systems, process the “notify” keyword on the local node.

Once the job has completed, the .f06, .f04, .log, .ndb, .op2, .plt, .pch, and .xdb files from the master task will be present as if the job were run locally.

Note: No attempt is made to copy DBset files between the local and master/slave systems. If this is required, you must handle this yourself and set the “dbs” keyword appropriately.

Determining Hosts Used by DMP Jobs

The examples that follow use the Linux/UNIX syntax. Nevertheless, unless specifically noted otherwise, the discussion and examples are applicable to Windows. The only change needed in the examples to make them applicable to Windows is to replace the colon (':') separator character with a semi-colon (';') separator character. Alternatively, you may use a comma (',') as a separator character on any platform.

The nastran command uses the following hierarchy to determine the list of hosts to use:

1. The nastran command “hosts” keyword on the command line
2. System-dependent environment variable.
   AIX: MP_HOSTFILE
   HP UNIX: DMPI_HOSTFILE.
3. The nastran command “hosts” keyword in an RC file.
4. AIX: The local “host.list” file.
5. AIX: The MP_RMPOOL environment variable.
6. The local host.

Consider the following examples:

md2007 nastran example dmparallel=4

On AIX, MP_HOSTFILE environment variable, the “host.list” file, or the MP_RMPOOL environment variable will determine the hosts used by this job. On all other systems, the job will run on the local host.
This job will run on the first four available nodes from the set “node1”, “node2”, “node3”, “node4”, “node5”.

This job will read the file “my.host.list”.

The nastran command provides a simple host allocation method. If a host listed by the “hosts” keyword is unavailable, it will be skipped and the next host considered. As long as at least the number of processors specified by the “dmparallel” keyword are available on one or more of the listed hosts, the job will be allowed to run.

**Hosts (AIX)**

The “hosts” keyword can now coexist with the LoadLeveler queue submittal process if your distributed jobs must be submitted via IBM’s LoadLeveler. To submit a job via LoadLeveler, the “hosts” keyword must use the syntax “host=@queue_name”.

**Example:**

```
md2007 nastran example dmp=4 hosts=@ll
```

In this example, four hosts will be assigned by LoadLeveler after the nastran command submits the job to queue “ll”.

To use this feature, you must define queue submittal commands in an RC file using the “submit” keyword.

**Example:**

```
submit=ll=ll_submit %job%
```

The previous example nastran command will submit a job to the “ll” queue using the site’s “ll_submit” command.

---

**Note:** This uses features of the nastran command’s standard queue submittal process, but you do not set the queue keyword.
Pool Request (AIX)
A pool request can be specified using the “hosts” keyword with either of the following forms:

```
hosts=@pool1:@pool2:...:@pooln
hosts=@pool
```

where `pooli` or `pool` is a number. The second form assigns all tasks to the specified pool number. See your system administrator for information on the pools available at your site.

This job runs two tasks each on pools 1 and 2.

```
md2007 nastran example dmparallel=4 hosts=@1:@1:@2:@2
```

This job runs all tasks on pool 3.

```
md2007 nastran example dmparallel=4 hosts=@3
```

If you are using LoadLeveler Version 2.1 or greater to process your pool request you may also need to use the “maxnode” keyword. This is required if you want more than one DMP task to run on a single host.

This job runs eight tasks on four hosts from pool 1. This assumes that the hosts have at least two processors.

### nastran Command “hosts” Keyword (Distributed Jobs Under LSF)

The “hosts” keyword will default to the value set by LSF when running as a distributed job and no other value for “hosts” was set on the command line or in an RC file.

#### Example:

```
bsub -n 4 md2007 nastran example dmp=4
```
This job will use four hosts selected by LSF. Note, the number of tasks appears twice: once for use by LSF, and once for use by MD Nastran.

Managing Host-Database Directory Assignments in DMP Jobs

The performance of the disk subsystem containing the permanent and SCRATCH DBsets can have a significant impact on MD Nastran performance. In the case of a DMP job, the impact can be even greater if multiple tasks are using the same file system. For SCRATCH DBsets, there are two ways in which this can be handled: one by specifying host-specific scratch directory values in an RC file and one by specifying a list of scratch directories using the “sdirectory” keyword. For DBsets, you may specify a list of DBset locations using the “dbs” keyword. When the list method is used to specify multiple scratch directories or DBsets, the entries are paired with the “host” keyword specified host names in a round-robin order. With these capabilities, you can finely control the use of disk I/O access paths by your job.

In the case of SCRATCH DBsets, the simplest method of specifying individual directory paths for each host is to use the RC file method, reserving the “sdirectory” list method for situations where you are assigning multiple DMP tasks to a single host and you want to separate the SCRATCH DBsets, placing each on a separate file system. The following is an example of an RC file that defines the default SCRATCH directory for each node in a four-node configuration with node names “node1”, “node2”, “node3” and “node4”. This example assumes that the MD Nastran installation directory is available to and is used by each node. The following would, typically, be included in the RC file in the “conf” directory, noting that the technique is available on all platforms, where customizing the node-specific SCRATCH directory pathnames being the only change needed:

```
# Define node-specific scratch directories
[ s.hostname = node1 ]
  sdir=/nodel/scratch

[ s.hostname = node2 ]
  sdir=/node2/scratch

[ s.hostname = node3 ]
  sdir=/node3/scratch

[ s.hostname = node4 ]
  sdir=/node4/scratch
```

Note that none of the “sdirectory” keyword values should be in “list” format, that is, contain multiple directories separated by a comma or colon (Linux/UNIX) or semi-colon (Windows) unless you wish that specification to be used whenever DMP processing is requested and when you are sure that the list will match the order in which host names are specified in the “hosts” keyword.

The following examples show the effect of the round-robin ordering. These examples are the Linux/UNIX syntax. Nevertheless, unless specifically noted otherwise, the discussion and examples are applicable to Windows. The only change needed in the examples to make them applicable to Windows is to replace the colon (':') separator character with a semi-colon (';') separator character. Alternatively, you may use a comma (',') as a separator character on any platform. Also, for Windows, round-robin
ordering is ignored if “ccsnodesmust=no” is specified since the host on which a particular DMP task will run is unpredictable.

```md2007 nastran example dmparallel=4 hosts=a:b sdirectory=/aa:/ba:/ab:/bb
dbs=/aa:/ba:/ab:/bb```

This example will assign the following host-sdirectory pairs (assuming hosts “a” and “b” each have at least two processors):

<table>
<thead>
<tr>
<th>Task</th>
<th>Host</th>
<th>Scratch Directory</th>
<th>DBS Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>/aa</td>
<td>/aa</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>/ba</td>
<td>/ba</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>/ab</td>
<td>/ab</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>/bb</td>
<td>/bb</td>
</tr>
</tbody>
</table>

If directory “/ba” was not available for writing by you on host “b”, the tasks assignments would be (assuming host “a” has at least three processors):

<table>
<thead>
<tr>
<th>Task</th>
<th>Host</th>
<th>Scratch Directory</th>
<th>DBS Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>/aa</td>
<td>/aa</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>/ab</td>
<td>/ab</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>/bb</td>
<td>/bb</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>/aa</td>
<td>/aa</td>
</tr>
</tbody>
</table>

**Managing Files in DMP Jobs**

**Note:**
- AIX: If a pool host assignment is requested, the input file and output directory must be global to all hosts in the pool — this will not be validated by the nastran command.
- Windows: If “ccsnodesmust=no” is specified, the input file and output directory must be global to all requested hosts. This is validated by the nastran command after converting each to UNC format, if necessary.

When an MD Nastran DMP job is running, the input file is directly read by each MPI task that can read the file, e.g., via NFS. Each host that cannot read the input file will read a local copy of the file that is copied, via rcp(1) or scp(1) for Linux/UNIX or using MSC developed utilities on Windows, to the job’s scratch directory (“sdirectory” keyword) before the job begins.
A similar check is made for the output directory. Any host that can write to the output directory ("out" keyword) will directly write its .f04, .f06, .log and other default output files to that directory. Any host that cannot see the output directory will write its default output files to the job’s scratch directory. For Linux/UNIX systems, these files will then be copied, again via rcp(1) or scp(1) for Linux/UNIX or using MSC developed utilities on Windows, back to the output directory at the end of the job.

**Note:** The nastran command will perform these tests by converting your pathname value to an absolute pathname. As a result, a path that varies depending upon the host will be labeled as unreadable. On Windows, your pathname value will be converted to UNC format, if necessary.

If the “sdirectory” keyword is not specified on the command line or in an RC file on the local host or is not specified in list format, i.e., with multiple directory specifications separated by commas or colons (Linux/UNIX) or semi-colons (Windows), each master or slave host will use its own scratch directory. This directory is determined on the master and each slave host by examining its command initialization file and version-specific RC files if the “version” keyword was defined.

DO NOT use an ASSIGN statement for any file that will be written by MD Nastran in a Distributed Memory Parallel (DMP) job. Instead, use the “sdirectory” and “dbs” keywords to specify names of the SCRATCH and permanent DB Sets.

**DMP Performance Issues**

In addition to the normal performance issues associated with a serial or SMP job, a DMP job adds communication bandwidth as a critical performance characteristic. The basic communications channels are:

- Shared memory - SMP and NUMA systems.
- Interconnect, adapter, or switch - NUMA and distributed systems.
- High-speed special-purpose network, e.g., HIPPI - all systems.
- TCP/IP network - all systems.

The performance of any MD Nastran job is very much dependent on CPU, memory subsystem, and I/O subsystem performance. A Distributed Memory Parallel (DMP) job on an SMP or NUMA system is extremely sensitive to I/O subsystem performance since each task independently accesses the I/O subsystem.

You are especially encouraged on SMP and NUMA systems to partition your scratch directory and database assignments on DMP jobs using the “sdirectory” and “dbs” nastran command keywords.

**Example:**

```bash
md2007 nastran example dmp=4 sdir=/scr1:/scr2:/scr3:/scr4\ dbs=/dbs1:/dbs2:/dbs3:/dbs4
```
The following assignments will be made in this job:

<table>
<thead>
<tr>
<th>Task</th>
<th>sdirectory</th>
<th>dbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/scr1</td>
<td>/dbs1</td>
</tr>
<tr>
<td>2</td>
<td>/scr2</td>
<td>/dbs2</td>
</tr>
<tr>
<td>3</td>
<td>/scr3</td>
<td>/dbs3</td>
</tr>
<tr>
<td>4</td>
<td>/scr4</td>
<td>/dbs4</td>
</tr>
</tbody>
</table>

The preceding example will perform substantially better than the following job, which uses the default assignments for the “sdirectory” and the “dbs” keywords.

**Example:**

```bash
md2007 nastran example dmp=4
```

While the ultimate effect of the communications channel on job performance is dependent upon the solution sequence, for best overall job performance, you should try to use the fastest communications channels available.

Additional DMP tuning information may be available in the “Read Me” file

```bash
install-dir/md2007/README.txt
```

on UNIX, or

```bash
install-dir\md2007\readme.txt
```

on Windows.

**HP-UX Systems**

HP has created a web site that lists many tuning parameters specific to MD Nastran performance. The URL is

http://www.hp.com/go/msc

**IRIX64 Systems**

The highest communication bandwidth will be achieved if you restrict your job to a single host (e.g., do not set the “hosts” keyword, or set it to only one host) and set “bypass_off=yes” (the default). You should
only set “bypass_off=no” if you want to use the HIPPI interface for communications among multiple computers.

If your system is lightly loaded, you can also set the MPI_DSM_PPM environment variable to 1, this will limit IRIX64 to only run one process on each two-processor node. This will enable the one processor to use all the node’s available bandwidth, instead of sharing it with the other processor. For example,

```
export MPI_DSM_PPM=1
```

in the Korn shell, or

```
setenv MPI_DSM_PPM 1
```

in the C shell. Please see the mpi(1) man page for further documentation on this and other environment variables that can increase your job’s performance.

Finally, FF_IO is strongly recommended for all DMP jobs.

**Windows Systems**

Currently, MD Nastran DMP only runs on Windows Compute Cluster Systems (CCS). On these systems, DMP tasks are scheduled using the CCS job scheduler. As of the date of this documentation (March, 2007), the only parameter used by the CCS job scheduler to allocate tasks to hosts (the CCS term for “hosts” is “nodes”) is the number of processors (logical CPUs) needed for a job. For relatively small MD Nastran DMP jobs, this may not be an issue but for large MD Nastran DMP jobs, having significant disk space and memory requirements, this can be a severe restriction. The only way to force the CCS job scheduler to assign tasks to all requested hosts is to tell it that the total number of processors required for the DMP job is the sum of all processors (logical CPUs) available on all of the requested nodes. The result of this is that the CCS job scheduler will run the DMP job only when no other jobs (whether MD Nastran jobs or jobs queued by other applications) are running on the requested hosts and will not assign jobs to these nodes while the MD Nastran DMP job is running.

The way you tell MD Nastran which of the job scheduling methods to use is through the use of “ccsnodesmust” keyword. By default (“ccsnodesmust=yes”), MD Nastran tells the CCS job scheduler that an MD Nastran DMP job is to run using all requested nodes. This can be changed, however, by specifying “ccsnodesmust=no”. In this case, MD Nastran tells the CCS job scheduler that the number of processors needed is just the value of the “dmparallel” keyword. The list of nodes to be used is the value of the “hosts” keyword but there is not guarantee that the CCS job scheduler will use all of the nodes, although none the DMP tasks will be run on any nodes not in the hosts list. Note, however, that the input file and output directory must be visible from all specified hosts since the host on which a particular DMP task is run is unpredictable.

This is, perhaps, best illustrated by considering the following examples. In these examples, assume that the Windows Cluster has four nodes (named n1, n2, n3 and n4), each with two dual core CPUs and with 1,000 GB disk space available on each node in a local scratch file system.
Example 1
Four DMP jobs are to be run, each requiring approximately 100GB disk space and each to be run with “dmparallel=4” specified. Since the total resources required by these jobs does not exceed the total resources available on any single mode, the following procedure could be used:

- Submit the jobs in four separate Command Prompt windows, using the following command in each window (modifying the job name appropriately):
  msc2007 nastran <jobname> dmp=4 hosts=n1,n2,n3,n4 ccsnodesmust=no <other options as appropriate>

Each job will be submitting telling the job scheduler that four processors are required. Assuming no other jobs are running on the cluster, these jobs will be run concurrently, where the job scheduler determines which nodes are to be used for each job. Note, however, that MD Nastran will ignore any per-task “directory” or “dbs” requests since the node on which a DMP task is run is unpredictable.

Example 2
One DMP job is to be run with four DMP tasks, each task required approximately 600GB disk space. The DMP job is expected to run about 10 hours. Since it is not possible for more than one DMP task to run on a single node, each DMP task must run on a separate node. The following procedure could be used.

- Submit the job in a Command Prompt window, using the following command, noting that “ccsnodesmust=yes” is the default.
  msc2007 nastran <jobname> dmp=4 hosts=n1,n2,n3,n4 <other options as appropriate>

The job will be submitted telling the job scheduler that 16 processors (or 32 processors if hyperthreading is supported and enabled on each node) are required. If there are any other jobs, the job scheduler will not start this job until all other jobs have finished running.

It is important to note that there is no way the jobs in the two examples can be run at the same time, even though there are enough resources in the cluster to support this. The current version of the CCS job scheduler only considers “number of processors required” in scheduling jobs.

Installing and Using LAM/MPI for MD Nastran on Linux
MD Nastran on Linux systems requires LAM/MPI to run in DMP mode. A lam and hpmpi directory is available in $MSC_BASE/$MSC_VERSD/$MSC_ARCH.

Using LAM/MPI
Full documentation for LAM/MPI is included in the LAM/MPI distribution. The following is provided as a quick introduction to using LAM/MPI in MD Nastran R2 DMP operations. Unlike V2001.0.7 and prior versions of MD Nastran, LAM Daemons are now started automatically by the nastran command.
Environment Variables

The LAMHOME, PATH, and MANPATH environment variables permit easy access to LAM/MPI. Bash and PDKSH users can place the following lines in their .profile files:

```bash
export LAMHOME=/usr/local/lam-7.0.4
export PATH=$LAMHOME/bin:$PATH
export MANPATH=$LAMHOME/man:$MANPATH
```

C shell users can place the following lines in their .login files:

```bash
setenv LAMHOME /usr/local/lam-7.0.4
set path=($LAMHOME/bin $path)
setenv MANPATH $LAMHOME/man:$MANPATH
```

LAM/MPI and Your .profile or .login scripts.

As a special feature, LAM/MPI will run your .profile or .login scripts when starting up each MPI rank. You must make sure that your scripts do not require a terminal or issue any output. For example, the following command will cause recon and lamboot to fail:

```
stty erase ^H
```

The solution to this "problem" is to make sure the stty command is only issued when TERM is defined. For example, a bash or ksh user could write

```
[[ -n $TERM ]] && stty erase ^H
```

to ensure that the stty command is only executed when you are actually logging on to the node.

Running an MD Nastran DMP Job

You can start an MD Nastran DMP job using the command:

```
$ md2007 nastran example dmp=2 hosts=node1:node2
```

where "node1:node2" indicates the LAM/MPI hosts to be used in the run.
Valid output is:

MD Nastran V2007 (Intel Linux 2.2.5-15smp) Thu Mar 30 17:44:15 2003
Determining available hosts, please wait...
DMP task 1: host="beowulf21(LAM n1)" sdir="/tmp"
DMP task 2: host="beowulf22(LAM n2)" sdir="/tmp"
MD Nastran beginning distributed job example.
MD Nastran beginning child job example.T71_15.t0 on beowulf21 (master).
MD Nastran V2007 (Intel Linux 2.2.5-15smp) Wed Mar  8 18:45:08 2003
MD Nastran beginning child job example.T71_15.t1 on beowulf22.
MD Nastran job dmp_108 completed.

**Note:** The “beginning child job” lines may appear in a random order.

### Using PBS with MD Nastran and LAM/MPI

Portable Batch System (PBS) is a queuing system that can be used to submit MD Nastran serial and DMP jobs. PBS is available from:

http://pbs.mrj.com/

Once you have downloaded and installed PBS, you can use the following sample script to run an MD Nastran DMP job under PBS:
#!/bin/ksh
#
# pbs_nast: PBS script to use with MSC.Nastran V2005 and LAM/MP.
#
# Usage: qsub -lnodes=Number-Of-Nodes pbs-nast
#
# Assume the data file is located in the directory whence the qsub
# command was issued.
# dat=$PBS_O_WORKDIR/parallel.dat
# jobdat=${dat##*/}
#
# Change the working directory to the scratch directory.
# TMPDIR=/scratch
cd $TMPDIR
#
# Pull the bulk data file over.
# rcp $PBS_O_HOST:$dat .
#
# Determine the number of ranks.
# dmparallel=$(sed -n -e '$=' $PBS_NODEFILE)
#
# Build the hosts keyword value.
# hosts=" 
for node in `cet $PBS_NODEFILE`
do
hosts=$hosts":"$node
done
#
# Run the MSC.Nastran job.
# md2007 nastran $jobdat dmparallel=$dmparallel hosts=$hosts 
# scratch=yes batch=no
#
# Shut down the LAM/MP environment.
# wipe -v $PBS_NODEFILE
#
# Push the files back to the submitting host.
# jobout=${jobdat%.*}
out=${dat%/*}
rcp -p $jobout.log $PBS_O_HOST:$out
rcp -p $jobout.f04 $PBS_O_HOST:$out
rcp -p $jobout.f06 $PBS_O_HOST:$out
#
# END

Note: Be aware that in order to receive your job's stdout and stderr, your .rhosts file on the node
issuing the "qsub" command must permit access from the remote host(s).
Problems Running LAM/MPI or MD Nastran DMP Jobs

Permission denied

This error is a remote execution (r-command) problem. Possibilities include:

- Your .rhosts file on a compute node is empty or does not include an entry for the host and username you submitted your DMP job from. To correct this error, log on to the node and ensure the .rhosts file is correct.
- Your .rhosts file on a compute node does not have the correct permissions. To correct this error, log on to the node and change the permissions using the command $ chmod go= ~/.rhosts

This command will remove "group" and "other" read access to the file.

rcmd: ${Node_name}.localdomain: Success

This error may occur if an account was not set up on the compute node. Ask your administrator for assistance.

lamnodes (kinit): no running daemon

This error occurs when the LAM/MPI daemon “lamd” is not running. Please verify "lamd" is running with "tping". If "lamd" is not running, use "recon" to validate the boot schema file and "lamboot" to start the "lamd" daemon.

*** USER WARNING MESSAGE (pgm: nastran, fn: validate_dmp_hosts)

The current working directory, ${directory}, is invalid on ${node_name}(LAM nl).

This error is caused because the directory does not exist on the remote node.

*** USER INFORMATION MESSAGE (pgm: nastran, fn: validate_dmp_hosts)

${node_name}(LAM nl): Unexpected MSC Nastran remote query response:

/nast/bin/nast2007 -d0 -rN2007 frc=/scratch/example.dat drwx=/scratch fw=/scratch/example.T17_T2 drwx=/scratch

This error is caused when MD Nastran was NOT installed correctly on the remote node.
This error is an authorization problem. Likely causes are

- The license or authorization file does not include the ability to make DMP runs.
- The license or authorization file was not accessible to the first node in the hosts list.

**Alternatives to LAM/MPI on Linux**

Linux provides three methods of DMP. LAM/MPI is the default. There are also implementations of HP/MPI and Intel/MPI. There has been no performance advantages gained by either of these, but they are provided to users at the hardware vendor’s request.

To run with HP/MPI add the following parameters to the command line:

hpmpi=yes proc=hp

To run with Intel/MPI add the following parameters to the command line:

intelmpi=yes proc=intel
Configuring and Running SOL 600

Hardware and Software Requirements

By default, SOL 600 parallel runs for UNIX uses MPICH Version 1.2.5 from Argonne National Laboratory for MPI. The latter is included on the MD Nastran CD. Although no specific hardware requirements exist for MD Nastran to run in shared memory parallel or distributed memory parallel mode, it is preferable to have fast network connections between the machines if more than one machine is used. It is recommended that the network should have a speed of at least 100 MBit per second. If only two machines are to be used, you can use a hub or a cross-over cable to connect them. If more than two machines are to be used, a switch is preferable. TCP/IP is used for communications.

Compatibility

MD Nastran supports connection of homogeneous networks with machines of the same type. Two machines are compatible if they can both use the same executables. Some examples of compatible machines are:

1. Several machines with exactly the same processor type and O/S.
2. One SGI R8000/IRIX 6.5 and one SGI R10000/IRIX 6.5 machine.
3. One HP J-Class/HPUX-11.0 and one HP C-Class/HPUX-11.0.

Definitions

1. Root machine: The machine on which the job is started.
2. Remote machine: Any machine other than the root machine that is part of a distributed parallel run on the network.
3. Shared installation: MD Nastran is installed in an NFS shared directory on one machine only. Other machines can access the executables since the directory is shared.
4. Distributed installation: MD Nastran is installed on all machines. Each machine accesses its own versions of the executables.
5. Distributed execution: SOL 600 is run on multiple machines that are connected with a network. Each machine loads the executables either from shared or local directories and then executes them.
6. Shared Memory Execution: More than one processor is used to run a parallel execution of SOL 600 on the same physical computer.
7. Shared I/O: MD Nastran reads and writes data in an NFS shared directory. Each executable running on the network reads and writes to the same directory.
8. Distributed I/O: MD Nastran reads and writes data in a directory located on each machine. You must make the input available in each directory and collect the results files after the analysis.
Network Configuration

MD Nastran only needs to be installed on the root machine where the installation directory is shared via NFS (shared installation). It can also be installed on the remote Machines, which then use their own executables (distributed installation). The root machine is the one on which the SOL 600 job is started. The remote machines can be located anywhere as long as they are connected to the network. The working directory on each machine can be a shared directory on any machine on the network (shared I/O) or it can be a local directory on the hard disk of each machine in the analysis (distributed I/O). User Notes on page 195 in this chapter provide instructions for specifying the working directory to use.

Installation Notes

This part describes the specific steps needed to install and set up a network version of SOL 600.

For shared memory parallel, there are no specific installation steps necessary. When running the job, all that is necessary is to add one input line to the Bulk Data of the form:

\[ \text{PARAMARC,ID,,NP} \]

where ID is an arbitrary integer and NP is the number of processors to be used (can not exceed the number on the computer).

For distributed parallel, install MD Nastran on the root machine and, if needed, on the remote machines. MD Nastran only needs to be installed on the root machine if it is a shared installation. There is nothing special that needs to be done related to the installation itself for the network version.

In order to run parallel jobs on machines connected over the network, jobs have to be set up properly. If any of the remote hosts do not have MD Nastran installed, the installation directory on the root machine needs to be shared using NFS or some other mechanism so that all executables are available from the remote machines. Users need to be able to connect between the machines using \texttt{rlogin} without having to provide a password.

For some computer systems such as HP, SGI and IBM SP, NEC Itanium 2, it is necessary to activate hardware MPI. To do that, change to the \texttt{/usr/nastran/md2007/marc/hpuxia64/marc2007/tools} directory (for example /usr/nastran/md2007/marc/hpuxia64/marc2007/tools), make a backup copy of the original include file, then run the maintain program. At the first prompt, enter 2 (Maintenance utilities). At the next prompt, enter 2.1 (Modify MPI setting of Marc). At the next prompt, enter the desired type of MPI for your system (for example, 2.1.5 for HP MPI). If it is not obvious which version of MPI to use, enter 1.2.1 to select MPICH. Then enter 0 as many times as necessary to exit the maintain program.

IBM SP machines may require special handling. Some use various types of special network switches and forms of communication software such as IBM’s loadleveler. If your system uses loadleveler, manually edit the include file in the \texttt{/tools} directory to choose the proper switch name and change other quantities. Be sure to make a backup of the original include file before editing. If you are an IBM expert, you might be able to do this from the comments in the include file. If not, contact MSC technical support for help. Once the modified include file is setup, the IBM system will automatically choose which nodes to use, depending on the workload of the machine and a hostfile will not be required. You can bypass the loadleveler by using the Maintain program to choose Hardware MPI (which is POE for IBM) and setup
a hostfile, as described below for other computer systems. If you bypass the loadleveler, you must use a 64-bit version of Marc.

Assume that there are two machines with hostnames host1 and host2 that are to be used in a parallel job over the network. MD Nastran has been installed on host1 and the job is to be started from this machine. A hypothetical naming convention is used for shared directories where a directory name on any machine starts with /nfs/hostname, where hostname is the name of the machine on which the directory is located.

First, test the installation for single processor execution. Change to an empty directory, copy pt6003.dat from the Nastran ~/tpl directory to the empty directory. Execute the single processor job using the command

```bash
nastran pt6003 scratch=yes
```

The job should complete normally and produce a file pt6003.marc.sts with an exit code (near the end of the file) of 3004. A file named pt6003.f06 should be produced with displacements and no FATAL errors or Severe Warning messages.

Next, test the installation for shared memory multi-processor execution. Copy pt6003.dat to pt6003p.dat. Edit pt6003p.dat and add the following lines after BEGIN BULK and before ENDMODEL

```plaintext
cparam,marctemp,0
cparam,marcoutr,1
PARAMARC,123,,2
```

Save the modified input file and execute the job using the command:

```bash
nastran pt6003p scratch=yes
```

This job should produce files such as 1pt6003p.marc.dat, 1pt6003p.marc.out, 2p6003p.marc.dat, 2p6003p.marc.dat, pt6003p.marc.sts and pt6003p.f06. The pt6003p.marc.sts file should be similar to the previous pt6003.marc.sts file with exit code 3004. The pt6003p.f06 file should have displacements that are nearly identical to those in the pt6003.f06 file.

Next, test the installation for distributed memory multi-processor execution. Copy pt6003p.dat to pt6003pp.dat. Change the SOL 600 Executive Control statement to read as follows:

```plaintext
SOL 600,106 path=3 outr=f06
```

Copy pt6003p.f06 to a file named marc.pth. Edit marc.pth and search for the string “executed”. Delete all lines from the beginning up to and including that line. Skip down one line and delete all other lines so that only one line remains in the file. The one line should look similar to:

```plaintext
```

Change this line to read as follows:

```plaintext
```
Save the marc.pth file. It needs to be in the same directory as the MD Nastran input file, pt6003pp.dat.

Next, setup a hostfile (which we will name hostfile) with two processors for the two machines you wish to use. The hostfile would have the following lines:

host1 1
host2 1 shared_directory_for_files shared_directory_where_marc_is

The “shared_directory_where_marc_is” should be of the form:

/usr/nastran/md2007/marc/hpuxia64/marc2007

Run the pt6003pp model the same as was done for the pt6003p model. The same files as produced by the pt6003p model should be produced. The values in the last column (max displacement) of the pt6003pp.marc.sts file should be the same as those for pt6003.marc.sts and pt6003p.marc.sts. The displacements in pt6003pp.f06 should also be the same as for pt6003.f06 and pt6003p.f06.

Running shared memory is much easier than distributed memory parallel and should be faster as well, however, the computers cost more.

* 

This section assumes that MD Nastran has been successfully installed on at least one of two machines that are to be used in a distributed analysis, and that the appropriate SOL 600 licenses are in order. Assume that host1 is the host name of the machine on which the job is to be started (the root machine). The host name of the other machine (the remote machine) is host2.

**How to Run a Network Job**

Verify that the two machines are properly connected. From host1, access host2 with `rlogin host2`. If a password needs to be provided to do the remote login, this has to be taken care of. If the `rlogin` is not possible without providing a password, a network run will not be possible. Be sure that both host1 and host2 appear in your .rhosts files in your root directory. If they are present, see the “Troubleshooting” section.

In order to perform an analysis over a network, a “host file” needs to be created by the user. This file defines which machines are to be used, how many processes are to run on each, what working directory should be used, and where the Marc executable can be found on each machine. No specific name or extension is used for the host file except that the name `jobid.host` must be avoided since it is used internally.

**Specification of the Host File**

The host file has the following general format:

```
host1 n1
host2 n2 workdir2 installdir2
host3 n3 workdir3 installdir3
```
Each line must start at column 1 (no initial blanks). Blank lines and lines beginning with a # (number symbol) are ignored. The first entry is the host name of a machine to be used in the analysis. The root machine must be listed first and each machine must only occur once.

The second entry specifies the number of processes to run on the machine specified in the first entry. The sum of the number of processes given in the host file must equal the number of domains used. In a five-domain job, it is required that \( n_1 + n_2 + n_3 = 5 \).

The third entry specifies the working directory to use on this host. This is where the I/O for this host takes place. The results files for this machine are created in this directory.

The fourth entry specifies where the version of Marc that this host should use is located. This entry can be omitted if the name is the same on all machines (which could be a shared directory on host1 with the same name from host2 and host3). The directories in the third and fourth entries will be used from the respective host. To check the correctness of the host file specification, log in to the respective machine and list the directories as specified in the host file. For the host file given above, do:

```
rlogin host2
ls workdir2
ls installdir2
```

The second line should show the working directory to use on host2 and the third line the installation directory that will be used by host2. The different domains of the job are associated with the different machines as follows. Suppose a five-domain job test is run using a host file defined as

```
host1 2
host2 1 workdir2 installdir2
host3 2 workdir3 installdir3
```

with appropriate definitions of the third and fourth entries, indicated below. (Note: SOL 600 usually works with a Marc “single file” parallel input file that is the same as a single processor input file. Internally, Marc will create six input files associated with this job: test.dat, 1test.dat, ..., 5test.dat and move them to the correct computer locations). Domains 1 and 2 will be associated with host1, domain3 with host2 and domains 4 and 5 with host3.

### Shared I/O

Suppose a job is to be run on host1 and host2. A shared directory on host1 is to be used for I/O and from host2 its name is /nfs/host1/workdir (assuming a hypothetical naming convention for shared directories which starts with /nfs/hostname). The installation directory is assumed to have the same name on both machines. The host file for a two processor job would simply be

```
host1 1
host2 1 /nfs/host1/workdir
```

To verify the workdir given, do

```
rlogin host2 ; ls /nfs/host1/workdir
```

The directory seen should be the same one as the working directory on host1.
Distributed I/O

If the user wants to have the I/O to be local on host2, specify the host file as

```
host1 1
host2 1 /usr/people/myjob
```

The I/O on host2 will now take place in the directory `/usr/people/myjob` on the hard disk of host2. For this case, the input files are transferred to `/usr/people/myjob` on host2 before the job is started, and the results files are transferred back after the analysis for postprocessing. This transfer of files is done automatically. It is also possible to use only two entries in the host file. This requires that both the working directory and the installation directory have the same names on all machines.

Shared vs. Distributed I/O

For jobs with very large post or restart files, it is sometimes more efficient to use distributed I/O. With distributed I/O, the input files and the post files are located on the host’s local disks. Marc by default automatically transfers the input files and the post files to and from the remote host if required.

Jobs with User Subroutine

User subroutines are fully supported using shared memory as well as distributed parallel.

The Fortran file with the subroutine is located in the working directory on the root machine. Marc automatically creates the executable and makes it available on all remote hosts. There is no need to modify the host file if it is correct for a job without a user subroutine. If the working directory is shared for all remote hosts and only compatible machines are used in the analysis, the user subroutine is compiled on the root machine and the executable is available in the shared working directory. If a remote host is using a local working directory, the executable will be automatically copied over to the remote machine using remote copy (`rcp`). Marc automatically knows if a directory is shared or local. If incompatible machines are used, the compilation is done on each machine separately. If a shared working directory is used, the host name is appended to the name of the executable. For local directories, the new executable is placed in the local working directory. This is all done automatically.

Solver

Solver type 6 (hardware provided sparse) is available on HP and SGI. No specific input is needed for its use in a parallel analysis. Marc makes use of the parallel features of these solvers. However, the use of a hardware solver is not recommended in a network run. The equation solution is performed on the root machine by starting multiple processes. This is done in order to utilize the parallel performance of the solver (which is using multithreading). This is efficient on a single parallel machine, but if the root machine of a network run does not have the number of processors available, it will not be efficient.

Solver types 0 (direct profile), 2 (sparse iterative), 4 (sparse direct), and 8 (multifrontal sparse) are supported in parallel. Out-of-core solution is only supported in parallel for Solver 8.

The option OOC,,1 (or OOC,0,1) which equates to MD Nastran Bulk Data entry param,marcoocc,2 is not presently supported with DDM.
Troubleshooting

Check that:

1. The network connection between the hosts is working by using the command `ping host`.
2. A remote login using the command `rlogin` can be done between the hosts without providing a password. If not, add all hosts to your `.rhosts` file in your login directory or contact your system administrator.
3. The host names used in the hostfile are correct. It should be the same as the output from the command `hostname` on the respective host.
4. The working and installation directories on the host file are correct. Log onto the remote host, change directory to these directories to verify the host file content.

Error messages:

1. The error message “semget failed...” at job start-up means that the communication environment is not clean. This can be checked with the Unix command `ipcs`. If entries belonging to specific users except `root` show up, they may need to be removed. Run the script `tools/mpiclean`.

Note: This will kill all parallel jobs currently running under the current user. Only entries belonging to the current user are deleted.

Other:

1. On some machines, sometimes there are files called `p4_shared_arena_ffff`, with `ffff` being some number, left in `/var/tmp`. These can eventually fill up the disk and should be removed.
SOL 600 Parallel Processing on Windows

Hardware and Software Requirements
The SOL 600 network version for Microsoft Windows uses MP-MPICH from University of Aachen, Lehrstuhl für Betriebssysteme. Although no specific hardware requirements exist to run SOL 600 in parallel, it is preferable that for distributed parallel processing to have fast network connections between the machines. It is recommended that the network should have a speed of at least 100 MBit per second. If only two machines are to be used, a hub or a cross-over cable can be used to connect them. If more than two machines are to be used, a switch is preferable. TCP/IP is used for communications. The O/S must be Microsoft Windows 2000, Service Pack 1 or later. It is only necessary to have a FORTRAN compiler installed if user subroutines are used. This version of SOL 600 supports Intel Fortran 8.0 as the default and Compaq visual Fortran 6.6B with Microsoft Visual Studio 6.0 as an alternate.

Definitions
1. Root machine: The machine on which the job is started.
2. Remote machine: Any machine other than the root machine that is part of a distributed run on the network.
3. Shared installation: MD Nastran is installed in a UNC shared directory on one machine only. Other machines can access the executables since the directory is shared.
4. Distributed installation: MD Nastran is installed on all machines. Each machine accesses its own set of executables.
5. Distributed execution: MD Nastran is run on multiple machines that are connected with a network. Each machine loads the executables either from shared or local directories and then executes them.
6. Shared I/O: Data is read and written to a UNC shared directory. Each executable running on the network reads/writes to the same directory.
7. Distributed I/O: Data is read and written to a directory located on each machine.
   Transfer of data files and post files between the root machine and remote machines is done automatically.
8. UNC – Uniform Naming Convention.

Network Configuration
MD Nastran only needs to be installed on the root machine where the installation directory is UNC shared (shared installation). MD Nastran can also be installed on the remote machines, which then use their own executable (distributed installation). The root machine is the one on which the job is started.

The remote machines can be located anywhere as long as they are connected to the network. The working directory on each machine can be a shared directory on any machine on the network (shared I/O) or it can be a local directory on the hard disk of each machine in the analysis (distributed I/O). The User Notes describes how to specify what working directory to use.
Installation Notes

This section describes the specific steps needed to install and set up a network version of SOL 600.

**Steps 1–4** Must be performed as Administrator or a user having administrator privileges.

**Step 1:** Install MD Nastran on the root machine. The installation directory must be shared such that it is available on the remote machines.

**Step 2:** Install MP-MPICH, which is also on the CD, on both the root machine and the remote machine(s). Note that MD Nastran only needs to be installed on the root machine but MP-MPICH must be installed on all machines used in an analysis. To install MP-MPICH, do the following:

- For the root machine, find the ~\tools directory, for example
  
  d:\msc.software\md_nastran\md2007\marc\tools

- Find the file nt-mpich.bat in the tools directory. Using notepad or some other ASCII editor, change the contents from the default specified to

  set MPI_ROOT=d:\msc.software\md_nastran\md2007\marc\mpich2

  or similar directory depending on where you installed MD Nastran

  a. Root machine:
  b. Use My Computer and find the location of the ~\mpich2\bin directory (for example)

  d:\msc.software\md_nastran\md2007\marc\mpich2\bin

c. Perform steps e to g below

  Non-Root machines:

  d. Change to the shared directory

  d:\msc.software\md_nastran\md2007\marc\mpich2\bin of the root machine

e. Type: smpd-install

  f. Enter a valid login username and password when prompted. Also enter the domain name if that is how you login to the system, or use “local” if you do not login through a domain. Note that the installation program does NOT verify that the password you entered is valid, so make sure that you enter exactly as the login password (either the local machine login or the domain login). Note: If you change your system login password you must repeat steps e and f.

**Step 3:** Make sure that the installation directory on the remote host is properly shared. Use My Computer and locate the directory to be shared. Right click on the directory and choose Sharing. Choose Share As and give it a Share Name (this is the UNC name) and click OK.
Step 4: Create a file with the shared naming information. From the Command Prompt, change directory to the tools directory in the MD Nastran installation directory (here assumed to be d:\msc.software\md_nastran\md2007\marc\tools).

\texttt{d:\msc.software\md_nastran\md2007\marc\tools}
\texttt{net share > marc.net}

This file has to be recreated each time the shared name of the installation directory is changed. The file marc.net contains the connection between the path names on the root machine and the UNC names, and can be created only by a user having administrator privileges. If this file does not exist or contains outdated information, the remote machines will not be able to find the executable on the root machine.

Step 5: Test the installation for single processor execution: Change to an empty directory, copy pt6003.dat from the Nastran ~/tpl directory to the empty directory. Execute the single processor job using the command

\texttt{nastran pt6003 scratch=yes}

The job should complete normally and produce a file pt6003.marc.sts with an exit code (near the end of the file) of 3004. A file named pt6003.f06 with displacements and no FA TAL errors or Severe Warning messages.

Step 6: If you have a multi-processor PC, test the installation for shared memory parallel execution: (If you do not have multiple processors on you machine, perform the next steps up to “execute the job” and skip to the next step. Copy pt6003.dat to pt6003p.dat. Edit pt6003p.dat and add the following lines after \texttt{BEGIN BULK} and before \texttt{ENDMODEL}

\texttt{param,marctemp,0}
\texttt{param,marcoutr,1}
\texttt{PARAMARC,123,,2}

Save the modified input file.

Execute the job using the command:

\texttt{nastran pt6003p scratch=yes}

This job should produce files such as 1pt6003p.marc.dat, 1pt6003p.marc.out, 2p6003p.marc.dat, 2pt6003p.marc.dat, pt6003p.marc.sts and pt6003p.f.06. The pt6003p.marc.sts file should be similar to the previous pt6003.marc.sts file with exit code 3004. The pt6003p.f.06 file should have displacements that are nearly identical to those in the pt6003.f.06 file.
Step 7: Test the installation for multi-processor, distributed execution.

Copy pt6003p.dat to pt6003pp.dat Change the SOL 600 e

Executive Control statement to read as follows:

SOL 600,106 path=3 outr=f06

Create a file named marc.pth which has the complete command to execute Marc in parallel. The file must contain one line that looks similar to the following:

d:\msc.software\md_nastran\md2007\marc\tools\run_marc --jid pt6003pp.marc.dat --nps 2 --v no --iam nanl --ho hostfile --b no

Save the marc.pth file. It needs to be in the same directory as the MD Nastran input file, pt6003pp.dat.

Next, setup a hostfile (which we will name hostfile) with two processors for the two machines you wish to use. The hostfile will have the following lines:

host1 1
host2 1 shared_directory_for_files shared_directory_where_marc_is

The “shared_directory_where_marc_is” needs to be of the form:

d:\msc.software\md_nastran\md2007\marc

Run the pt6003pp model using the command

nastran pt6003p scratch=yes

The same files as produced by the pt6003p model should be produced and the values in the last column (max displacement) of the pt6003pp.marc.sts file should be the same as for pt6003.marc.sts and pt6003p.marc.sts. The displacements in pt6003pp.f06 should also be the same as for pt6003.f06 and pt6003p.f06.

If the job stalls or hangs at start-up time, exit it by typing control-C in the window in which it was started. See “User Notes” in this chapter.

User Notes

This section assumes that MD Nastran, including MP-MPICH, has been successfully installed on two machines that are to be used in a distributed analysis and that the appropriate MSC licenses are in order. Assume that host1 is the host name of the root machine from which the job is to be started and the host name of the other machine (the remote machine) is host2.

How to Run a Network Job

First, make sure that the two machines are properly connected. From host1, access host2 with Network Neighborhood. If this is not possible, a network run will not be possible. See “User Notes” in this case. In order to perform an analysis over a network, a special file called a host file needs to be created by the
user. This file defines which machines are to be used, how many processes are to run on each, what working directory should be used, and where the Marc executable can be found on each machine. No specific name or extension is used for the host file except that the name $jobid$.host must be avoided since it is used internally.

**Specification of the Host File**

The host file has the following general format:

```
host1 n1
host2 n2 workdir2
host3 n3 workdir3
```

Each line must start at column 1 (no initial blanks). Blank lines and lines beginning with a # (number symbol) are ignored. The first entry is the host name of a machine to be used in the analysis. The root machine must be listed first and each machine must only occur once. The second entry specifies the number of processes to run on the machine specified in the first entry. The sum of the number of processes given in the host file must equal the number of domains used. In a five-domain job, it is required that $n1+n2+n3=5$.

The third entry specifies the working directory to use on this host. This is where the I/O for this host takes place. The MD Nastran input file for this machine must be in this directory and the results files for this machine are created in this directory.

The different domains are associated with the different machines as follows. Suppose a five-domain job *test* is run using a host file, defined as

```
host1 2
host2 1 workdir2
host3 2 workdir3
```

with appropriate definitions of the third entry, indicated below. Since SOL 600 uses a “single file” parallel input, Marc will create six input files associated with this job such as *test*.dat, 1*test*.dat, ..., 5*test*.dat. Domains 1 and 2 will be associated with host1, domain3 with host2 and domains 4 and 5 with host3.

**Shared I/O**

Suppose a job is to be run on host1 and host2. A shared directory on host1 is to be used for I/O. The UNC sharename for this directory is assumed to be *dir7*. The host file for a two processor job would simply be

```
host1 1
host2 1 "\host1\dir7"
```

To verify the work directory given, enter Network Neighborhood from host2 and click through host1 -> *dir7*. The directory seen should be the same one as the working directory on host1.

It is also possible to use only two entries for each host in the host file (host name and number of processes). This requires that the working directory is shared and that the sharing information is up to date in the file *tools\marc.net* (see above where marc.net is created).
Distributed I/O

If the user wants to have the I/O to be local on host2, specify the host file as

host1 1
host2 1 D:\users\dir5

The I/O on host2 will now take place in the directory D:\users\dir5 on the hard disk of host2. For this case, the input files are transferred to D:\users\dir5 on host2 before the job is started, and the results files are transferred back after the analysis for postprocessing. This transfer of files is done automatically.

Shared vs. Distributed I/O

For jobs with very large post or restart files, it is sometimes more efficient to use distributed I/O. With distributed I/O, the input files, and the post files are located on the host’s local disks. Marc, by default, automatically transfers the input files and the post files to and from the remote host if needed. To run a job using distributed I/O, specify a local directory in the host file:

host1 2
host2 1 d:\workdir

Jobs with User Subroutine

User subroutines are available. If local directories are used on remote hosts (distributed I/O), the new executable will be transferred automatically to the remote host if necessary.

Solver

Solver type 6 (hardware provided sparse) is not available on the Microsoft Windows platform. Solver types 0 (direct profile), 2 (sparse iterative), 4 (sparse direct), and 8 (multifrontal sparse) are supported in parallel. Out-of-core solution is only supported in parallel for Solver 8.

The option OOC,.1 (or OOC,0,1) which equates to MD Nastran Bulk Data entry param,marcoocc,2 is not presently supported with DDM.

Troubleshooting

Use the Event Viewer on the root host by selecting Start/Programs/Administrative Tools/Event Viewer to read the System, Security, and Application log files under the Log button. To read these files on a remote host, select Log/Select Computer and pick the remote host from the list. Look for log entries related to Cluster Manager Service V2. Check that:

1. Your user ID is recognized by the local or remote hosts. Also check that the password you entered during the MP-MPICH installation process is the same as that for the local system or the domain. The installation process does not verify that the password you entered is the same as the machine or domain login password.

Also note that your password must not have any spaces in it or else the MP-MPICH installation process will not handle your password correctly. Either of these types of errors will give you the error message:
Access is denied. (5)
Also note that if you change your login password you must reset your MPMPICH password by
doing the following:
   cd marc\mpich2\bin
   mpiexec -store -save ..\..\tools\setdomain.bat
Enter the username, domain, and password when prompted.
The file marc\tools\setdomain.bat is used to set the domain for the login. If you wish to
change the domain you use for the MP-MPICH login, then this file should be updated
accordingly. It has the format of:
   set domain=mydomainname

2. The remote hosts have permission to read from and write to the root host. In particular, check that
the sharing is giving full access; that is, not being restricted to read only.
3. Your licenses including parallel processing are valid.
4. The host names are valid.
5. Check that the Cluster Manager Service is running. The service is not running if you get a message
similar to:
   "C:\MSC\marc\demo_ddm\e2x1.host" does not contain any valid hosts
   Scanning network....
   Select My Computer/Control Panel/Services and look for Cluster Manager Service V2 in the list.
   Check that it has Status: Started. This must be done on all hosts. Also check that the user name
   used for this service is the same on all hosts.

Running a Parallel Job when not Connected to the Network
If you disconnect your system from the network and want to run a parallel job on that system, you will
have to install the Microsoft Loopback Adapter. Follow these steps.
   1. Go to Control Panel, Add/Remove Hardware.
   2. Select the hardware task you want to perform:
      Add/Troubleshoot a device
   3. Choose a Hardware Device:
      Add a new device
   4. Do you want Microsoft Windows to search for your new hardware?
      No, I want to select the hardware from a list
   5. Select the type of hardware you want to install:
      Network adapters
   6. Select Network Adapter:
      Manufacturers: Microsoft
      Network Adapter: Microsoft Loopback Adapter
It will now install the loopback adapter. You will have to enable/disable the loopback adapter as you remove/connect your machine to the network.

**Running a Parallel job on Windows XP System when not a Member of a Domain**

If you will be running a parallel job on a Windows XP system that is not a member of a domain, you will have to modify a registry entry.

Using `regedt32`, look for the following key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa  
"forceguest" : REG_DWORD : 00000001

If you find this key, change the REG_DWORD value to 0. The name may also appear as ForceGuest. If you do not have this registry entry, your system will function properly.

**Running a Parallel Job on Windows XP SP2**

After you install or upgrade to Windows XP SP2, the RPC protocol does not permit anonymous requests to the RPC Endpoint Mapper, but requires client requests to be authenticated. This will cause an "Access is Denied" error when you attempt to run a parallel job. To workaround this problem:

1. Run `gpedit.msc` from a command prompt.
2. Select **Computer Configuration**, expand **Administrative Templates** and expand **System**.
3. Click **Remote Procedure Call**.
4. Double click **RPC Endpoint Mapper Client Authentication**. Change the value to **Enabled**.
Running an ISHELL Program

The ISHELL module allows you to invoke your own program from DMAP to perform custom processing. Two features are provided to make running your program easier.

The first feature is the ability to construct a full named based on the up-to eight character name provided by DMAP and a list of file-type associations. MD Nastran will first attempt to find an executable in the current directory using the name as-is from the DMAP call, i.e., all upper-case. On UNIX, if this name cannot be found, another attempt is made by converting the name to all lower-case.

If a name was not found, the Command Processor Associations defined by the “ishellext” keyword will be used to construct additional names by concatenating the DMAP name with each file-type in turn until the name is found or the table is exhausted. The command processor extensions consist of pairs of file-types and commands. On UNIX systems, the default command processor associations are:

<table>
<thead>
<tr>
<th>File-Type</th>
<th>Command Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>directly execute</td>
</tr>
<tr>
<td>.sh</td>
<td>sh</td>
</tr>
<tr>
<td>.ksh</td>
<td>ksh</td>
</tr>
<tr>
<td>.csh</td>
<td>csh</td>
</tr>
<tr>
<td>.pl</td>
<td>perl</td>
</tr>
<tr>
<td>.prl</td>
<td>perl</td>
</tr>
</tbody>
</table>

On Windows, the default command processor associations are:

<table>
<thead>
<tr>
<th>File-Type</th>
<th>Command Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>.bat</td>
<td>directly execute</td>
</tr>
<tr>
<td>.exe</td>
<td>directly execute</td>
</tr>
<tr>
<td>.com</td>
<td>directly execute</td>
</tr>
<tr>
<td>.pl</td>
<td>perl</td>
</tr>
<tr>
<td>.prl</td>
<td>perl</td>
</tr>
</tbody>
</table>

Note: While this capability is similar to the Windows “File Type Associations,” it does not use that information.

These tables are processed in the order shown.

If none of the names exist in the current working directory, MD Nastran will resort to the second feature design to assist in using the ISHELL module, the “ishellpath” keyword. If this keyword is set, MD
Nastran will repeat the search described above for each of the directories listed by the keyword. To aid in using this keyword, the nastran command will set the default value for “ishellpath” as the directory containing the input data file if you have not set the keyword on the command line, via the MSCishellpath environment variable, or in an RC file.

If a file has still not been found in either the current working directory or any of the directories listed by the “ishellpath” keyword, the system PATH will be searched. Finally, if a suitable file was not found, a UFM will be issued.

A sample ISHELL job is provided by the files TPLDIR:qaishell.dat, TPLDIR:QAISHELL, and TPLDIR:qaishell.pl. The ISHELL call is

```
.ISHELL//'QAISHELL'/'S,N,IRTN/
NOINT/NOREAL/NOCMPX/NOCHAR/NOUNIT/
INT1/INT2/INT3/INT4/
REAL1/REAL2/REAL3/REAL4/
CMPL1/CMPL2/CMPL3/CMPL4/
STRING1/STRING2/STRING3/STRING4/
/UNIT1/UNIT2/UNIT3/UNIT4 $
```

For the following example, assume the nastran command provides the default value for the “ishellpath” keyword, i.e., the directory containing the input data file.

```
md2007 nastran qaishell
```

On UNIX, the following names will be checked (assuming the default command processor associations): QAISHELL, qaishell, QAISHELL.sh, qaishell.sh, QAISHELL.ksh, qaishell.ksh, QAISHELL.csh, qaishell.csh, QAISHELL.pl, qaishell.pl, QAISHELL.prl, and finally qaishell.prl. Since the file “QAISHELL” exists in the same directory as the input file, it will be found after first looking for the names in the current working directory.

On Windows, the following names will be checked (assuming the default command processor associations): QAISHELL.BAT, QAISHELL.EXE, QAISHELL.COM, QAISHELL.PL, and finally QAISHELL.PRL. Since the file “qaishell.pl” exists in the same directory as the input file, it will be found after first looking for the names in the current working directory.
Defining Command Processor Associations

The nastran command treats each specification of the “ishellext” keyword as either an addition to, modification of, or deletion from, the current definition. For example, using the default command processor associations, specifying

```
ishellext=tcl=wish
```

will add a new processor, “wish”, for the file-type “.tcl”, after the last currently defined processor. Specifying

```
ishellext=pl=
```

will delete the current association of “perl” for the file-type “.pl”. Finally,

```
ishellext=sh=ksh
```

will replace the “sh” definition for the “.sh” file type on UNIX. To change the processing order, delete the current entry and then respecify it (to append it to the end of the table). For example, to force UNIX systems to find “qaishell.pl” before “QAISHELL”, specify

```
ishellext=.=,.='
```

Note that this first deletes the null processor “.”, and then re-specifies it as “.”.

```
ishellext=.=,. '',sh=sh,ksh=ksh,csh=csh,pl=perl,prl=prl
ishellext=bat='',exe='',com='',pl=perl,prl=perl
```

These two examples are the default associations for UNIX and Windows respectively.

Special Considerations (Windows)

On Windows, all executable files must have a non-null file type; this is why the “QAISHELL” script cannot be used on Windows, even if you have a Korn shell installed.

You may need to define “CMD.EXE” on Windows NT or “COMMAND.COM” on Windows 9X as the command processor for certain “.exe” files. Examples include 16-bit compiled Basic programs.
Finally, you can use a hash mark, “#”, in place of the equals sign on Windows to facilitate setting the processor association in a “.bat” file. For example,

```
isellext#bat#",exe","com","pl#perl,prl#perl
```

is an alternate definition of the default Windows association.
Using the ISHELL-INCLUDE Statement ("!")

The ISHELL module provides a way to dynamically alter the instruction stream of a running DMAP, making it easier to integrate your own programs, and simplifying the task of customizing MD Nastran. The ISHELL-INCLUDE statement ("!") extends the ISHELL feature to the instruction stream of the input file. This capability is derived by merging the features of both the ISHELL and the INCLUDE statements (by first executing the external program and then including the output in the input stream). The format is:

```
! embedded shell command.
! continuations are indicated simply by the presence
! of another "!" in the first non-blank position of the next line.
! all characters following the "!" are passed to the appropriate
! shell for evaluation.
```

The shell (or command processor) is determined by the MSC_ISHELLEXT environment variable, or by the ishellext keyword from the command line or RC file (see “Running an ISHELL Program” for more details). On UNIX systems, the command processor associated with the null file type is used for the ISHELL-INCLUDE statement. In most cases this requires one of the following keyword assignments to be added to the command line:

```
ishellext=.=/bin/csh # for csh scripts
ishellext=.=/bin/ksh # for ksh scripts
ishellext=.=/bin/sh # for sh scripts
ishellext=.=perl # for perl scripts
```

**Note:** The ISHELL-INCLUDE statement is currently not supported for Windows.

Like the INCLUDE statement, the ISHELL-INCLUDE statement can appear anywhere in the input file. However, the output (captured from “stdout”) must be appropriate to the section in which it will be included (i.e. the final input stream must constitute a valid MD Nastran input file). Unlike the INCLUDE statement, nested ISHELL-INCLUDE statements are not supported.

The processing of an embedded shell script is done as follows:

1. The entire script is extracted and written to a temporary file.
2. If the ISHELL-INCLUDE occurs within a DMAP alter, the processing is delayed until the DMAP compiler is invoked.
3. Otherwise, the input file processing is suspended, and the external program is executed. Output from the external program is captured to another temporary file which is immediately opened and included into the input stream.
4. Once the reading of the entire output is completed, processing of the input file is resumed.

The following additional processing steps are done for an embedded shell script located within a DMAP alter:

1. The DMAP statements that are selected by the alter are extracted to an external file named: “ishell.stdin”.
2. If stdout is written to, then that output is included in the alter; otherwise, “ishell.stdin” is read. This allows an interactive program like “vi” to simply save the modified input buffer, and it is automatically included in the alter.

An immediate benefit of the ISHELL-INCLUDE statement is the ability to customize the MD Nastran job to dynamically record (and/or respond) to the run time environment. The following example captures the value of a few environment variables as comments in the f06 file:

example.dat:

```bash
echooff $ removes copy of the ishell script below from the f06
! echo "echoon" # just the results from the shell will be echoed
! echo "$"
! echo "$ License File: `printenv MSC_LICENSE_FILE`"
! echo "$ Job was run on host: `printenv HOST`"
! echo "$ Nastran Version: `printenv MSC_VERSD`"
! echo "$ Temporary Directory: `printenv MSC_SDIR`"
! echo "$ TMPDIR: `printenv TMPDIR`"
! echo "$ Scratch: `printenv MSC_SCR`"
! echo "$ User: `printenv USER`"
! echo "$ Display: `printenv DISPLAY`"
! echo "$ Base: `printenv MSC_BASE`"
! echo "$ Path: `printenv MSC_JID`"
! echo "$ Memory: `printenv MSC_MEM`"
! echo "$ Assign File: `printenv MSC_ASG`"
! echo "$ Shell: `printenv SHELL`"
! echo "$ Ishell Ext: `printenv MSC_ISHELLEXT`"
! echo "$ Ishell Path: `printenv MSC_ISHELLPATH`"
! echo "$ Ishell File: $0"
! echo "$ 
```
The example above should be executed with /bin/csh as the command processor:

> nastran example.dat scr=yes ishellext=.=/bin/csh
Improving Network File System (NFS) Performance (UNIX)

The Network File System (NFS) is software allowing file systems on remote computers to appear as if they were mounted on the local computer. There are two daemons that handle NFS traffic: “nfsd” handles file system access requests by the local computer to remotely mounted file systems; “biod” handles requests by remote computers to access local file systems.

These daemons have been designed so that multiple executing copies of each daemon increase NFS traffic capacity. Two of the possible causes of poor NFS performance are a lack of sufficient daemons to handle NFS requests made by the local computer to remotely mounted file systems (nfsd), or a lack of sufficient daemons to handle NFS requests of local file systems by remote computers (biod). The default number of daemons for nfsd and biod is typically four of each. This default is usually fine for a stand alone workstation used by one person. If you or others are accessing many remote file systems or run many MD Nastran jobs accessing file systems on file servers or remote workstations, you may need to increase the number of nfsd and biod daemons on both systems to increase NFS performance.

If you are running three or more MD Nastran jobs accessing disks on remote computers via NFS, MSC.Software recommends increasing both nfsd and biod daemons above the standard defaults. A good starting point is twelve (12) nfsd daemons and eight (8) biod daemons per CPU on client and server computers, respectively.

Your system administrator can change both system’s configurations to start additional NFS daemons. The administrator can also monitor network statistics with “nfsstat” to ensure network traffic is being handled efficiently. Additional daemon tuning may be necessary for your specific network needs.
Creating and Attaching Alternate Delivery Databases

MD Nastran uses the Structured Solution Sequences (SSS), located in `install_dir/md2007/arch` on UNIX and `install_dir/md2007/arch` on Windows, to specify the default solution sequences. You may modify and store a tailored solution sequence by creating a new delivery database. This procedure is also useful to eliminate unwanted solutions from the delivery database or add additional solution sequences.

The following files are delivered in the `install_dir/md2007/nast/del/` directory on UNIX and `install_dir/md2007/nast/del` on Windows:

<table>
<thead>
<tr>
<th>Filename</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buildsss</td>
<td>UNIX script used to build delivery database</td>
</tr>
<tr>
<td>buildsss.bat</td>
<td>Windows BAT file to build delivery database</td>
</tr>
<tr>
<td>*.dmap</td>
<td>SubDMAP source</td>
</tr>
<tr>
<td>*.dck</td>
<td>SubDMAP source that must be preprocessed by MSCFPP</td>
</tr>
<tr>
<td>*.ddl</td>
<td>NDDL source</td>
</tr>
</tbody>
</table>

Using MSC-Supplied Source

To rebuild the delivery database using the MSC-supplied source, the following procedure is used:

1. Change the working directory to an empty work directory. For example,

   ```
   cd $HOME/new-del
   ```

   on UNIX, or

   ```
   cd %HOMEDRIVE%%HOMEPATH%
ew-del
   ```

   on Windows.

2. Rebuild the delivery database.

   `md2007 buildsss`

Upon completion of this procedure, the delivery files SSS.MASTERA, SSS.MSCOBJ, and SSS.MSCSOU are created. These files are attached with the “delivery” keyword, (p. 300).

These files may be installed in the master architecture directory (if you have write access) with the command:
CHAPTER 5
Using the Advanced Functions of MD Nastran

Using Modified Source
To build a modified delivery database, use the following procedure.

1. Change the working directory to an empty work directory. For example,

   ```bash
   cd $HOME/new-del
   ```

   on UNIX, or

   ```bash
   cd %HOMEDRIVE%%HOMEPATH%\new-del
   ```

   on Windows.

2. Copy the subDMAP and NDDL source files that are to be modified to the current directory.

   ```bash
   cp install_dir/md2007/nast/del/subDMAP.dmap .
   cp install_dir/md2007/nast/del/subDMAP.dck .
   cp install_dir/md2007/nast/del/nddl.ddl .
   ```

   on UNIX, or

   ```bash
   copy install_dir\md2007\nast\del\subDMAP.dmap .
   copy install_dir\md2007\nast\del\subDMAP.dck .
   copy install_dir\md2007\nast\del\nddl.ddl .
   ```

   on Windows where subDMAP and nddl are the specific files to be modified.

3. Modify the desired subDMAP and/or NDDL source files using a text editor.

4. Rebuild the delivery database.

   ```bash
   md2007 buildsss src=
   ```
Upon completion of this procedure, the delivery files SSS.MASTERA, SSS.MSCOBJ, and SSS.MSCSOU are created. These files are attached with the “delivery” keyword (page 300).

These files may be installed in the master architecture directory (if you have write access) with the command:

```
cp SSS.* install_dir/md2007/arch
```

on UNIX, or

```
copy SSS.* install_dir\md2007\arch
```

on Windows.
Checkpoint Restart Facility (SGI-IRIX64)

Checkpoint and Restart (CPR) is a facility for saving a running process or set of processes and, at some later time, restarting the saved process or processes from the point already reached, without starting all over again. The checkpoint image is saved in a set of disk files, and restarted by reading the saved state from these files to resume execution. Processes can continue to run after checkpoint, and can be checkpointed multiple times. The process(es) that are saved by the CPR facility cannot contain unsafe checkpoint items such as open sockets or special memory devices, etc. For application softwares that do not automatically generate job restart files, a CPR checkpoint image can be the necessary fault tolerant safety net.

The CPR implementation on IRIX is based on the POSIX 1003.1m standard and extensions. System administrators can use the IRIX CPR facility to suspend and resume job execution, monitor a checkpoint, and remove statefiles. A CPR configuration and control file can also be generated by the administrator to define the desired FILE, WILL, DIR, and FORK policies that guide the job behavior at a checkpoint.

Checkpointing of sequential, shared memory parallel (SMP), distributed memory parallel (DMP), and FFIO based jobs are supported.

Complete details of the cpr(1) command and the cview graphical user interface are described in the IRIX Checkpoint and Restart Operation Guide. The users should checkpoint an entry point in the Nastran process hierarchy. An example of such for a sequential job with three FFIO child processes is shown as follows:

```
% ps -j
 PID   PGID   SID   TTY       TIME   CMD
 4847   4847   4847  ttyq      20:01  rlogind
 4895   4895   4847  ttyq      20:00  csh
 5413   5413   4847  ttyq      20:00  runme
 5416   5413   4847  ttyq      20:00  sol108.T5
 5437   5413   4847  ttyq      20:00  time
 5438   5413   4847  ttyq      20:49  analysis
 5444   5413   4847  ttyq      20:00  analysis
 5445   5413   4847  ttyq      20:00  analysis
 5446   5413   4847  ttyq      20:00  analysis
 5447   5447   4847  ttyq      20:00  ps
```

The ‘ps’ command output indicates that the entry point of the Nastran job in the runme script (UNIX PID=5413) which, in turn, executes the sol108.T5 (UNIX PID=5416) script. This job would work with a HID type checkpoint for 5413 or 5416, or a GID type checkpoint for 5413.
The runme (PID=5413) process is not recovered since it has not been included in the checkpoint hierarchy.

The CPR facility will issue an explicit message if an error is encountered in the checkpointing or restarting process. Please refer to Tables 2-2 and 2-3 of SGI’s *Checkpoint and Restart Operations Guide* for the most common causes of failure to checkpoint or restart.
Using the Utility Programs

- Overview 214
- ESTIMATE 215
- F04REPR 226
- HEATCONV 229
- MSCACT 230
- MSGCMP 234
- NEUTRL 235
- OP4UTIL 236
- OPTCONV 239
- PLOTPS 240
- RCOUT2 244
- RECEIVE 245
- TRANS 247
- XMONAST (UNIX) 251
- XNASTRAN (UNIX) 254
- Building the Utilities Delivered in Source Form 257
Overview

This chapter describes how to use the various MD Nastran utility programs. Table 6-1 groups these utilities by function.

Table 6-1 Utility Program Functions

<table>
<thead>
<tr>
<th>Utility</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESTIMATE</td>
<td>Estimates system requirements of an MD Nastran job and suggests performance improvements.</td>
</tr>
<tr>
<td>F04REPRT</td>
<td>Perl script to summarize or compare .f04 files.</td>
</tr>
<tr>
<td>HEATCONV</td>
<td>Reformats MSC Nastran Version 67 heat-transfer and optimization data files into current formats.</td>
</tr>
<tr>
<td>OPTCONV</td>
<td></td>
</tr>
<tr>
<td>MSCACT</td>
<td>Accumulates and summarizes MD Nastran accounting data.</td>
</tr>
<tr>
<td>MSGCMP</td>
<td>Compiles the message catalog.</td>
</tr>
<tr>
<td>NEUTRL</td>
<td>Converts MD Nastran plot files to PostScript or neutral format.</td>
</tr>
<tr>
<td>PLOTPS</td>
<td></td>
</tr>
<tr>
<td>RCOUT2</td>
<td>Converts neutral-format OUTPUT2 files to binary format.</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>Moves results database (XDB) files between dissimilar computers.</td>
</tr>
<tr>
<td>TRANS</td>
<td></td>
</tr>
<tr>
<td>XMONAST</td>
<td>Graphical user interface that submits and monitors MD Nastran jobs on UNIX systems.</td>
</tr>
<tr>
<td>XNASTRAN</td>
<td></td>
</tr>
</tbody>
</table>

Sections “ESTIMATE” on page 3 through “XNASTRAN (UNIX)” on page 43 describe each utility (in alphabetical order), and present applicable keywords and examples. “Building the Utilities Delivered in Source Form” on page 46 contains instructions on how to build the source code utilities.
ESTIMATE

ESTIMATE may be used to estimate the memory and disk requirements for MD Nastran jobs and make suggestions on improving the performance of these jobs. ESTIMATE will read the input data file and estimate the job’s memory and disk requirements. The ESTIMATE program is most accurate in predicting the requirements of static analyses that do not have excessive output requests. The memory requirements for normal modes analyses using the Lanczos method are reasonably accurate; however, the disk requirements are dependent upon the number of modes, this is a value that ESTIMATE cannot determine. Memory and disk requirements for other solutions are less accurate.

The basic format of the “estimate” command is

```
md20071 estimate input_file [keywords]
```

where `input_file` is the name of the data file. If the file type of the input data file is “.dat”, it may be omitted from the command line.

ESTIMATE processes keywords using the following precedence to resolve conflicts when keywords are duplicated (with 1 representing the highest precedence):

1. The Bulk Data file.
2. The command line.
3. The nastran INI and RC files (if “nastrc=yes” is specified).
4. `data-file-directory/.estimaterc` on UNIX, or `data-file-directory\estimate.rcf` on Windows, where `data-file-directory` is the directory containing the input data file.
5. `$HOME/.estimaterc` on UNIX, or `%HOMEDRIVE%%HOMEPATH%\estimate.rcf` file on Windows.
6. `estimate.ini` in the directory containing the ESTIMATE executable.

Please be aware that the Bulk Data file can only contain statements that are accepted by MD Nastran. The following keywords will be recognized by ESTIMATE when they appear in the Bulk Data file on NASTRAN statements:

```
buffpool, buffsize, real
```

**Note:** “buffsize=estimate” is NOT accepted on a NASTRAN statement.
The following Case Control statements will be recognized by ESTIMATE when they appear in the bulk data file:

adapt, method, mpc, sp

**Note:** If these statements appear multiple times, e.g., in subcases, only the first occurrence of each case control statement will be recognized.

Similarly, the nastran INI and RC files can only accept keywords that are accepted by the nastran command. The following nastran command keywords will be recognized by ESTIMATE when they appear in nastran RC files if and only if "nastrc=yes" is also set:

bpool, buffsize, memory, real, realpha, smemory, version

The full set of ESTIMATE utility keywords can ONLY appear on the ESTIMATE command line or in the ESTIMATE RC files, e.g., "estimaterc" on UNIX and "estimate.rcf" on Windows.

**Keywords**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapt</td>
<td>adapt=number Default: None Selects an ADAPT set for adaptivity jobs if an ADAPT Case Control command is not present or multiple ADAPT Case Control commands are present in the data file. By default, ESTIMATE will choose the first ADAPT found.</td>
</tr>
<tr>
<td>bpool</td>
<td>bpool=value Default: 27 (SUPER-UX) 37 (all others) Same as MD Nastran keyword, see “bpool” on page 295. This keyword cannot appear in an ESTIMATE RC file if “nastrc=yes” is specified.</td>
</tr>
<tr>
<td>buffsize</td>
<td>buffsize=number Default: 8193 Same as MD Nastran keyword, see “buffsize” on page 296. This keyword cannot appear in an ESTIMATE RC file if “nastrc=yes” is specified.</td>
</tr>
<tr>
<td>dballco</td>
<td>dballco=value Default: 1 Allows you to scale DBALL estimates. This scale factor is applied before the &quot;dballmin&quot; value, that provides a lower bound for DBALL estimates. Example: md20071 estimate example dballco=2 This will double the DBALL disk estimate and then apply the &quot;dballmin&quot; lower bound.</td>
</tr>
</tbody>
</table>
CHAPTER 6
Using the Utility Programs

Example:

```
md20071 estimate example
dballco=0.5
```

This will halve the DBALL disk estimate. An estimate less than than the lower bound specified by "dballmin" will be set to the lower bound.

dballmin

dballmin=value  Default: 1mb

Allows you to define the lower bound for all DBALL estimates. This bound is applied after the "dballco" value, that multiplies the actual estimate by a "conservatism" factor.

Example:

```
md20071 estimate example
dballmin=2mb
```

This will set the minimum DBALL disk estimate to 2 MB.

dskco

dskco=value  Default: 1

Allows you to define a factor to scale total disk estimates. This scale factor is applied before the "dskmin" value, that provides a lower bound for total disk estimates.

Example:

```
md20071 estimate example dskco=2
```

This doubles the total disk estimate and then applies the "dskmin" lower bound.

Example:

```
md20071 estimate example dskco=0.5
```

This will halve the total disk estimate. An estimate less than the lower bound specified by "dskmin" will be set to the lower bound.

dskmin

dskmin=value  Default: 1mb

Allows you to define the lower bound for all total disk estimates. This bound is applied after the "dskco" value, that multiplies the actual estimate by a "conservatism" factor.

Example:

```
md20071 estimate example dskmin=2mb
```

This will set the minimum total disk estimate to 2 MB.

enable

The "enable" keyword can be used to explicitly enable rules. This may be useful to enable a rule that was automatically suppressed when a value was assigned. For example, the following command will now calculate the estimated memory requirements for a job even though a value for memory was specified on the command line:

Example:

```
md20071 estimate example memory=5mb
enable=10
```

estimatedof

estimatedof=yes,no  Default: No

Indicates if the number of degrees of freedom are to be estimated. By default, ESTIMATE will count the DOF. This process takes time, but it is generally more accurate. Specifying "estimatedof=no" will result in a less accurate, but faster, estimate of the DOF. The presence of any MESH entries in the Bulk Data will force "estimatedof=yes".
memco

```
memco=number
```

Default: 1.0

Allows you to specify a constant factor that is either more or less conservative than the default.

Example:

```
md20071 estimate example memco=2
```

This setting will double the memory estimate.

memmin

```
memmin=value
```

Default: 16mb

Allows you to define the lower bound for all memory estimates. This bound is applied after the “memco” value, that multiplies the actual estimate by a "conservatism" factor.

Example:

```
md20071 estimate example memmin=8mb
```

This will set the minimum memory estimate to 8 MB.

memory

```
memory=size
```

Default: 4MW

Same as MD Nastran keyword, see “memory” on page 315. This keyword cannot appear in an ESTIMATE RC file if “nastrc=yes” is specified.

method

```
method=number
```

Default: None

Selects a METHOD for dynamics jobs if a METHOD Case Control command is not present or multiple METHOD Case Control commands are present in the data file. By default, ESTIMATE will choose the first METHOD found.

mode

```
mode=keyword
```

Default: suggest

Selects the program operating mode. Specifying “mode=estimate” will result in memory and disk estimates only. Specifying “mode=suggest”, the default, will estimate memory and disk requirements for the current job configuration, suggest modifications to improve the performance, and provide estimates for the memory and disk requirements of the suggested configuration. Specifying “mode=modify” does all that “mode=suggest” does plus actually make the suggested changes to your data file. See “out” to specify the new data file’s name and information on organizing your input file.

Note: If “mode=modify” is specified, and ESTIMATE detects errors in the input file or encounters valid Bulk Data that is not understood by ESTIMATE, the program will revert to “mode=suggest”.

Example:

```
md20071 estimate example
mode=estimate
```

The memory and disk requirements for the current job are displayed.

Example:

```
md20071 estimate example
```
The memory and disk requirements for the current job, suggestions for improving performance, and memory and disk requirements for the suggested configuration are displayed.

Example:

```
md20071 estimate example
mode=modify
```

The memory and disk requirements for the current job, suggestions for improving performance, and estimates of memory and disk requirements for the suggested configuration are displayed. If, and only if, modifications to “example.dat” are suggested, the original input file is versioned (given indices) and the revised data file is written to “example.dat”.

**mpc**

```
mpc=number
```

Default: None
Selects an MPC if an MPC Case Control command is not present or multiple MPC Case Control commands are present in the data file. By default, ESTIMATE will choose the first MPC found.

**nastrc**

```
nastrc=yes,no
```

Default: Yes
The “nastrc” keyword allows you to select the type of RC file processing invoked by the ESTIMATE utility. Setting “nastrc=yes”, the default, will process the standard MD Nastran RC files before the standard ESTIMATE RC files, i.e., 

```
$HOME/.estimaterc and data-file-directory/estimaterc
```

on UNIX, and

```
%HOMEDRIVE%%HOMEPATH\estimate.rcf and data-file-directory/estimate.rcf
```

on Windows, are processed. Setting “nastrc=no” will only process the standard ESTIMATE RC files.

**out**

```
out=pathname
```

Default: input filename
Specifies the name of the output file if “mode=modify” is specified and modifications of the data file are actually required. By default, the original file is versioned (given indices) and the revised data file is written to the original input file’s name. See “Using Filenames and Logical Symbols” on page 99

Example:

```
md20071 estimate example
mode=modify
```

If modifications to “example.dat” are suggested, the original input file is versioned (given indices) and the revised data file is written to “example.dat”.

Example:

```
md20071 estimate example
mode=modify \out=modified
```

The revised data file is written to “modified”.
Note:
In order to minimize the amount of data duplicated between the original input file and the modified file, MSC recommends that the Bulk Data that is not subject to modification by ESTIMATE (i.e., all Bulk Data except PARAM and EIGRL entries) be placed in an INCLUDE file.

An example of the recommended input file organization is:

```
NASTRAN statements
FMS statements
Executive
CEND
Case Control
BEGIN BULK
PARAM,...
$
EIGRL,...
$
INCLUDE file.bulk
$
ENDDATA
```

- **pause**
  
  **pause=keyword**
  
  Default: No
  
  Pause ESTIMATE before exiting to wait for the “Enter” or “Return” key to be pressed. This can be useful when ESTIMATE is embedded within another program. The values are “fatal”, “information”, “warning”, “yes”, and “no”. Setting “pause=yes” will unconditionally wait; “pause=fatal” will only wait if a fatal message has been issued by ESTIMATE; “pause=information” and “pause=warning” will similarly wait only if an information or warning message has been issued. The default is “pause=no”, i.e., do not wait when ESTIMATE ends.

- **real**
  
  **real=value**
  
  Default: See text.
  
  Same as MD Nastran keyword, see “real” on page 330. This keyword cannot appear in an ESTIMATE RC file if “nastrc=yes” is specified.

- **realdelta**
  
  **realdelta=value**
  
  Default: See text.
  
  Same as MD Nastran keyword, see “realdelta” on page 330. This keyword cannot appear in an ESTIMATE RC file if “nastrc=yes” is specified.

- **report**
  
  **report=keyword**
  
  Default: Normal
  
  Specifies the program’s report format. The “report=normal” format is intended to be read by you. The “report=keyword” format is intended to be read by a program.
**CHAPTER 6**  
**Using the Utility Programs**

**scr300co**  
\textit{scr300co=value}  
Default: 1  
Allows you to define a factor to scale SCR300 estimates. This scale factor is applied before the "scr300min" value, that provides a lower bound for SCR300 estimates.

Example:  
\texttt{md20071 estimate example}  
\texttt{scr300co=2}

This will double the SCR300 disk estimate and then apply the "scr300min" lower bound.

Example:  
\texttt{md20071 estimate example}  
\texttt{scr300co=0.5}

This will halve the SCR300 disk estimate. An estimate less than the lower bound specified by "scr300min" will be set to the lower bound.

**scr300min**  
\textit{scr300min=value}  
Default: 1mb  
Allows you to define the lower bound for all SCR300 estimates. This bound is applied after the "scr300co" value, that multiplies the actual estimate by a "conservatism" factor.

Example:  
\texttt{md20071 estimate example}  
\texttt{scr300min=2mb}

This will set the minimum SCR300 disk estimate to 2 MB.

**scratchco**  
\textit{scratchco=value}  
Default: 1  
Allows the user to define a factor to scale SCRA TCH estimates. This scale factor is applied before the "scratchmin" value, that provides a lower bound for SCRA TCH estimates.

Example:  
\texttt{md20071 estimate example}  
\texttt{scratchco=2}

This will double the SCRA TCH disk estimate and then apply the "scratchmin" lower bound.

Example:  
\texttt{md20071 estimate example}  
\texttt{scratchco=0.5}

This will halve the SCRA TCH disk estimate. An estimate less than the lower bound specified by "scratchmin" will be set to the lower bound.

**scratchmin**  
\textit{scratchmin=value}  
Default: 1mb  
Allows you to define the lower bound for all SCRA TCH estimates. This bound is applied after the "scratchco" value, that multiplies the actual estimate by a "conservatism" factor.

Example:  
\texttt{md20071 estimate example}  
\texttt{scratchmin=2mb}

This will set the minimum SCRA TCH disk estimate to 2 MB.
**smemory**  
\[ \text{smemory}= \text{size} \]  
Default: 0 (SUPER-UX)  
100 (all others)

Same as MD Nastran keyword, see “smemory” on page 337. This keyword cannot appear in an ESTIMATE RC file if “nastrc=yes” is specified.

**spc**  
\[ \text{spc}= \text{number} \]  
Default: None

Selects an SPC if an SPC Case Control command is not present or multiple SPC Case Control commands are present in the data file. By default, ESTIMATE will choose the first SPC found.

**suppress**  
\[ \text{suppress}= \text{list} \]  
Default: None

Specifies rules that are to be suppressed when “mode=suggest” or “mode=modify” is specified. See “Rules” on page 223 for the list of rules. If no value is specified, i.e., “suppress=”, then any rules previously suppressed are enabled. Multiple rules can be suppressed by using the keyword multiple times or by specifying a comma-separated list.

Example:  
```
md20071 estimate example
suppress=1
```

Suppress rule 1, the rule controlling BUFFSIZE.

Examples:  
```
md20071 estimate example suppress=1,6
md20071 estimate example suppress=1
md20071 estimate example suppress=2
suppress= 
suppress=1,6
```

Suppress rules 1 and 6.

**verbose**  
\[ \text{verbose}= \text{yes,no} \]  
Default: No

Specifies the amount of information to be displayed. Specifying “verbose=yes” will generate a much larger amount of output. The additional information includes a more detailed summary of the input file, the parameters used in estimating the memory and disk requirements, and the estimates for the original file, even when “mode=suggest” or “mode=modify” is specified.

**version**  
\[ \text{version}= \text{string} \]  
Default: 2006
Specifies the version of MD Nastran for which the estimates are to be targeted. The version will affect the estimated memory requirements and the actions of various rules, see “Rules” on page 223. This keyword cannot appear in an ESTIMATE RC file if “nastc=yes” is specified.

**wordsize**

`wordsize=number`  
Default: 32

64 if mode = i8 supported

Specifies the word size of the estimate’s target computer. By default, ESTIMATE’s calculations will be appropriate the current computer. This keyword may be used to specify estimates for a computer with a different word size. A comma-separated list of values may be specified when estimates and suggestions for multiple machines are desired. If “mode=modify” was specified, the modification are based on the last word size specified.

**Rules**

ESTIMATE has a fixed rule base that it uses to make suggestions for improvement. Any of the rules may be suppressed with the “suppress” keyword. The current rules are:

1. Set recommended BUFFSIZE.
   
   \[
   \begin{align*}
   &\text{BUFFSIZE}=8193 \quad \text{DOF} \leq 100000 \\
   &\text{BUFFSIZE}=16385 \quad 100000 < \text{DOF} \leq 400000 \\
   &\text{BUFFSIZE}=32769 \quad \text{DOF} > 400000
   \end{align*}
   \]

2. Use default BPOOL.
   
   \[
   \begin{align*}
   &\text{BPOOL}=37 \quad \text{wordsizesize} = 32 \\
   &\text{BPOOL}=20 \quad \text{wordsizesize} = 64; \text{version} < 70.5 \\
   &\text{BPOOL}=27 \quad \text{wordsizesize} = 64; \text{version} \geq 70.5
   \end{align*}
   \]

3. Suppress symmetric decomposition if not enough memory for sparse.
   
   SYSTEM(166)=0

4. Make all open core available to modules.
   
   Delete HICORE.
5. Select the sparse solver.
   - Delete SPARSE
   - Delete USPARE
   - SPARSE=1
   - USPARE=0

6. Force default rank size.
   - Delete SYSTEM(198)
   - Delete SYSTEM(205)

7. Do not sequence.
   - PARAM,NEWSEQ,-1

8. Use default Lanczos parameters.
   - EIGRL,...,V1=""
   - EIGRL,...,MAXSET=15

9. Use default SMEMORY.
   - INIT SCRATCH (MEM=100)
   - INIT SCRATCH (MEM=0)

10. Use estimated memory size.
    - memory=estimated-memory

11. Use default RAM.
    - INIT MASTER (RAM=30000)

12. Real.
    - Delete REAL.

13. Do not use Supermodule.
    - Delete PARAM,SM,YES.

14. Do not use Parallel Lanczos.
    - Delete NUMSEG.
Examples

The ESTIMATE program can be used in several ways. The default mode will make suggestions on improving the performance of MD Nastran and estimate the resource requirements of the job assuming the suggested parameters.

md20071 estimate example

To get an estimate of the job using the current parameters, use the command:

md20071 estimate example mode=estimate other_estimate_keywords

To have a new input file generated with the suggested changes, use the command:

md20071 estimate example mode=modify other_estimate_keywords

To run MD Nastran with the memory estimated by ESTIMATE, use:

md20071 nastran example memory=estimate other_nastran_keywords
F04REPRT

The F04REPRT utility is a Perl script that will summarize and/or compare .f04 files. The utility can determine the CPU time consumed by various MD Nastran modules, i.e., as a DIAG 49 replacement, or compare the relative performance of one or more jobs under various configurations.

Notes:

1. You must have Perl installed on your system to use this utility. Perl is available from numerous sources, including the URL http://www.perl.com

   This is not an MSC.Software Corporation site and MSC has no control over the site’s content. MSC cannot guarantee the accuracy of the information on this site and will not be liable for any misleading or incorrect information obtained from this site.

2. Previously, MD Nastran’s DIAG 49 provided a summary of CPU time spent in various modules. That DIAG has been removed, and replaced by this utility.

The basic format of the F04REPRT command is

\[
\text{md20071 f04reprt.pl} \ -s \ [\text{options}] \ \text{pathname} \ [\text{pathname} \ ...]
\]

or

\[
\text{md20071 f04reprt.pl} \ -d \ [\text{options}] \ \text{old1 new1} \ [\text{oldn newn} ...]
\]

where “-s” selects the summary mode, “-d” selects the comparison mode, options are zero or more of the options listed below, pathname is a path name, and oldi and newi are pathnames. If a pathname is a directory, all .f04 files in the directory are summarized/compared.

Note: Alternatively, you can run F04REPRT with the command

\[
\text{perl install-dir/md20071/util/f04reprt.pl} \ arguments ... \]

on UNIX, or

\[
\text{perl install-dir\md20071\util\f04reprt.pl} \ arguments ... \]

on Windows if perl is in your PATH. UNIX users can also use the command

\[
\text{install-dir/md20071/util/f04reprt.pl} \ arguments ... \]

if your Perl executable is /usr/local/bin/perl, or the “shbang” line was updated to the appropriate path.

Running F04REPRT without any arguments will display a help message explaining the utility’s options.
Options

-c    -c  Default: No
Indicates module times are to be accumulated in a single entry, rather than separate entries for each module occurrence

-d    -d  Default: None
Requests a comparison (difference) between each pair of oldi and newi pathnames specified on the command line. If only one pair of pathnames are specified, the “-d” is optional.

-e    Specifies that module elapsed times are to be used for ordering entries instead of CPU times.

-f    -f c  Default: Space
 Specifies a field separator character to separate field in the comparison report. This character may be inclosed in either single or double quotes to protect it from the command shell.

-m    -m number  Default: 0.05
Specifies the minimum CPU time threshold for comparisons or summaries. CPU times less than this threshold will be ignored.

-o    -o file-type  Default: None
Specifies an output file-type. If specified, each comparison or summary report will be written to a separate file in the current working directory with the name basename.ext where basename is the base name of the pathname or oldi.

If not specified, output will be written to stdout with each report separated by a form feed “Ctrl-L” character.

-r    -r number  Default: 5
Specifies the delta percentage used for “FASTER” and “SLOWER” comments in comparison (-d) output.

Any old versus new comparisons that exceed this delta from 100%, eg., (delta < 95%) or (delta > 105%), will print the appropriate comment.

-s    -s  Default: None
Requests a summary report for each pathname specified on the command line. If only one pathname is specified, the “-s” is optional.

-x    -x file-type  Default: f04
Specifies an alternate input file type.
Examples

md20071 f04reprt.pl example

If “./example” on UNIX, or “\example” on Windows, is a subdirectory of the current directory, F04REPRT will write a summary report to stdout for every .f04 file in the directory. Otherwise, if “./example.f04” on UNIX, or “\example.f04”, on Windows is a file, a summary report of the one file is written to stdout.

md20071 f04reprt.pl old new

If “old” and “new” are subdirectories of the current working directory, F04REPRT will generate lists of the .f04 files in each directory. Comparisons will be made between each pair of files with the same name in the two directories. Non-.f04 files and unpaired .f04 files, i.e., .f04 files that exist in either “old” or “new” but not both, will be ignored. Otherwise, if “old.f04” and “new.f04” are files, then a comparison of these two files will be displayed.
**HEATCONV**

HEATCONV may be used to reformat an existing heat-transfer Bulk Data file used in MSC Nastran prior to Version 68 into a format compatible with Version 68 or later. The operations performed by this program are described in the *MSC.Nastran Release Notes for Version 68*. The basic format of the “heatconv” command is

```
md20071 heatconv input_file [keywords]
```

where *input_file* is the name of the heat-transfer data file. If the file type of the old data file is “.dat”, it may be omitted from the command line.

**Keywords**

- **output**
  
  **output=pathname**
  
  Default: input_file
  
  This option specifies the name of the reformatted data file. By default, the old output file is renamed by appending the file type “.old”; the new file is the original name of the input file. If an output file is specified using this option, the original input filename is unchanged.

**Examples**

To execute the program, enter the following command:

```
md20071 heatconv example
```

The Version 68-compatible output is written to

```
example.dat
```

The original data file is renamed to example.dat.old.
MSCACT

MSCACT may be used to generate usage reports from the accounting files generated by MD Nastran when the “acct=yes” keyword is used. The basic format of the “mscact” command is

```md
md20071 mscact [keywords] acc-file [acc-file ...]
```

where `acc-file` are the names of the accounting file(s) to be summarized.

**Note:** The keywords only affect files listed after the keyword.

### Keywords

**perfile**

- **Syntax:** `perfile=yes, no`
- **Default:** No
- Specifies the summary is to be printed on a per file basis. If “perfile=yes” is specified, a summary of each file will be individually printed. By default, the summary will include all files.

**sortby**

- **Syntax:** `sortby=keyword`
- **Default:** Name
- Sort the report as specified by the keyword. The keywords are:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Sort Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>Sort by third report column.</td>
</tr>
<tr>
<td>cpu</td>
<td>Sort by second report column.</td>
</tr>
<tr>
<td>name</td>
<td>Sort by first report column.</td>
</tr>
<tr>
<td>none</td>
<td>Do not sort report; report is ordered as found in data file.</td>
</tr>
</tbody>
</table>

Setting “sortby=none” produces a report very similar to the previous versions of this utility.
CHAPTER 6
Using the Utility Programs

Examples

All of the following examples assume your current working directory is the MD Nastran accounting directory, i.e., install_dir/acct on UNIX and install_dir/acct on Windows.

To summarize accounting data across all files:

```
md20071 mscact file1 file2
file1 file2:
  Total: cpu-sec count
```

where `filei` are the filenames, `cpu-sec` is the total CPU seconds across all files, and `count` is the number of entries accumulated across all files.
To summarize accounting data from individual files:

```
md20071 mscact perfile=yes file1 file2
  file1:
    Total: cpu-sec count
  file2:
    Total: cpu-sec count
```

where `filei` is the name of each file, `cpu-sec` is the total number of CPU seconds, and `count` is the number of entries in each file.

To summarize accounting data in individual files by user:

```
md20071 mscact summary=user perfile=yes file1 file2
  file1:
    user1: cpu-sec1 count1
    user2: cpu-sec2 count2
    ...
    Total: cpu-sec count
  file2:
    user1: cpu-sec1 count1
    user2: cpu-sec2 count2
    ...
    Total: cpu-sec count
```

where `filei` are the filenames of each file, `useri` are the names, `cpu-seci` are the total CPU seconds for each user, `counti` are the number of entries accumulated for each user, `cpu-sec` is the number of total CPU seconds, and `count` is the number of entries in each file.

**Accounting File Format**

A separate file is created for each month of each year and is named

```
install_dir/acct/msclyymm.acc
```

on UNIX and

```
install_dir\acct\msclyymm.acc
```

on Windows where `yy` are the last two digits of the year and `mm` is the month (01 to 12). Each month’s file is independent of every other file.
The accounting file begins with three header records followed by detail records, one detail record for each MD Nastran job run during the given month and year. Comments, indicated by a hash mark “#” as the first character of the line, may be placed anywhere in the file after the header records.

Detail records (any non-comment line after the third line) include the following data:

1. The day the job was started (i.e., Sun., Mon., Tue., Wed., Thu., Fri., or Sat.).
2. The month the job was started (i.e., Jan., Feb., Mar., Apr., May, Jun., Jul., Aug., Sep., Oct., Nov., or Dec.).
3. The date of the month the job was started (i.e., 01 through 31).
4. The time the job was started (i.e., hh:mm:ss, where hh is 00 through 23, mm is 00 through 59, and ss is 00 through 59).
5. The time zone (i.e., the “TZ” environment variable).
6. The year the job was started (four digits).
7. The name of the user running the job.
8. The job’s output filename.
9. The analysis application, e.g., MD Nastran.
10. The version of the application (e.g., 70.5).
11. The SOL used by the job (e.g., 101, SESTATICS).
12. The total CPU time, in seconds, of the job (from the .f04 file).
13. The cumulative CPU time, in seconds, of all detail records up to and including this record.
14. The cumulative CPU time, in minutes, of all detail records up to and including this record.
15. The account ID as specified by the nastran command’s “acid” keyword.
16. The account data as specified by the nastran command’s “acdata” keyword.

**Note:** The cumulative times (fields 13 and 14) are for historical purposes only. These values are ignored.
MSGCMP

MSGCMP compiles a text message file and generates a binary message catalog. The basic format of the command is

```
md20071 msgcmp text_file [message_catalog]
```

where `text_file` is the name of an existing text message file or is “-“ to read from stdin, and `message_catalog` is the optional name of the message catalog that will be written. The type of the text file must be “.txt”. If a message catalog is not named, the message catalog will be written in the local directory as “text_file.msg”. The message catalog can be tested using the “msgcat” keyword (p. 319).

The utility can also regenerate a text file from an existing message catalog using the command

```
md20071 msgcmp message_catalog.msg [text_file]
```

where `message_catalog`.msg is the name of an existing message catalog and `text_file` is the optional name of a text file that will be written. The type of the message catalog must be “.msg” and must be entered on the command line. If a text file is not named, the text file is written to stdout.

The text source file for the standard message catalog is

```
install_dir/md20071/util/analysis.txt
```

on UNIX and

```
install_dir\md20071\util\analysis.txt
```

on Windows. The standard message catalog is

```
install_dir/md20071/arch/analysis.msg
```

on UNIX and

```
install_dir\md20071\arch\analysis.msg
```

on Windows.

Examples

The following command will compile the message catalog from a text file named “myfile.txt”

```
md20071 msgcmp myfile
```

The message catalog will be named “myfile.msg”. This catalog may be used with the nastran command

```
md20071 nastran myjob msgcat=myfile.msg other_nastran_keywords
```

Note: Message catalogs are machine dependent. “Binary File Compatibility” identifies the systems that are binary compatible; binary compatible systems can use multiple copies of the same message file.
NEUTRL

NEUTRL converts a binary-format plot file into a neutral-format plot file. The basic format of the “neutrl” command is

```
md20071 neutrl binary_plot_file
```

where `binary_plot_file` is the name of a binary plot file. If the file type of the plot file is “.plt”, it may be omitted from the command line.

**Keywords**

- **dump**
  - dump=yes,no Default: no
  - This option enables a raw print of each plot command to be made before it is processed. This print is used for debugging purposes only.

- **output**
  - output=pathname Default: `binary_plot_file.neu`
  - This option specifies the name of the neutral-format file. If “out=—” is specified, the neutral plot file is written to stdout. By default, the output file is the name of the input file with the new type “.neu”.

- **verbose**
  - verbose=yes,no Default: yes Output is a disk file.
  - no Output is stdout.
  - This option specifies whether processing messages are to be written.

**Examples**

To execute the program, enter the following command:

```
md20071 neutrl example1
```

The name of the output file is

```
example1.neu
```
OP4UTIL

OP4UTIL may be used to validate, copy or reformat binary files created using the MD Nastran OUTPUT4 module. It may also be used to “dump” the contents of any binary format file. The basic format of the “op4util” command is:

```
md20071 op4util <options> <file names>
```

This program is used as follows:

To generate a usage/help message:

```
md20071 op4util
md20071 op4util -h[elp]
md20071 op4util -?
```

To copy a file:

```
md20071 op4util -c[opy] [-v[erbose]] <from_fname> <to_fname>
```

To dump a file or files:

```
md20071 op4util -d[ump] [-v[erbose]] <fname_1> [... <fname_n>]
```

To convert a file from big-endian to little-endian or vice-versa:

```
md20071 op4util [-x[change]] [-v[erbose]] [-m nnn] <from_fname> <to_fname>
```

To convert a file from one endian format to a specified endian format:

```
md20071 op4util <endian_opt> [-v[erbose]] [-m nnn] <from_fname> <to_fname>
```

To query a file or files to determine their format:

```
md20071 op4util -q[uery] [-v[erbose]] <fname_1> [... <fname_n>]
```

To validate (test) a file or files, i.e., to check their validity as OUTPUT4 files:

```
md20071 op4util -t[est] [-v[erbose]] [-m nnn] <fname_1> [... <fname_n>]
```
Keywords

-? Requests that usage information be written to stdout. This is the same as the -h option.

-<copy> Requests the copy option. This option copies the file specified by <from_fname> to the file specified by <to_fname>, overwriting any existing file and creating a new file if it does not exist. This option does not validate <from_fname> or change its format in any way.

-d<ump> Requests the file dump option. This option lists the contents of each file, including record number and record length information, in both hexadecimal and character formats. Just as will the -query option, it checks each of the files specified by <fname_1> to <fname_n> to see if it is a valid binary file and reports its endian. The file need not be a valid OUTPUT4 file.

-h[elp] Requests that usage information be written to stdout.

-m nnn This parameter is only required when the -test, -xchange or <endian_opt> options fail because of memory allocation errors. The nnn value is the size of the memory to be used, in MB, and must be in the range 1 to 2047. The blank between the -m and the nnn value is optional.

-q[uery] Requests the file query option. This option checks each of the files specified by <fname_1> to <fname_n> to see if it is a valid binary file and reports its endian. It does not test the actual file data to see if the file is a valid OUTPUT4 file.

-t[est] Requests the file validate (test) option. This option reads each of the file specified by <fname_1> to <fname_n>, checking for a valid binary format file containing matrices in the proper OUTPUT4 format.

-v[erbose] Requests "verbose" output. Normally, the -copy, -test, -xchange, and <endian_opt> options do not generate any output and the -query and -test options only write out a single line about each of the files they process. In verbose mode, program headers and detailed file descriptions are generated, and the -test, -xchange and <endian_opt> options will list the matrices in the files along with their format and size.

-x[change] Requests the endian conversion option. <from_fname> will be checked to see if it is a valid binary file and, if it is, is copied to the file specified by <to_fname>, checking the data format as it copies the file and converting the data from the <from_fname> endian to the opposite endian. That is, if <from_fname> is a big-endian file, <to_fname> will be a little-endian file and vice-versa. On long-word systems, the integer length of the input file will be preserved. On short-word systems, long-word integer input files will be converted to short-word output files having the opposite endian from the input file.
<endian_opt> Requests the endian conversion option. This option is very similar to the
-xchange option except that the endian of <to_fname> is explicitly specified.
If <to_fname> is to have big-endian format, <endian_opt> must be
-b[igendian] or one of the following synonyms: -aix, -fuji[tsu],
-hpux, -irix, -prim[epower], -sol[aris], -supe[rux] or
-uxpv. If <to_fname> is to have little-endian format, <endian_opt>
must be -l[ittleendian] or one of the following synonyms: -alph[a],
-linu[x] or -wind[ows]. If <from_fname> already has the desired
endian format, a copy will be performed instead. On long-word systems,
<endian_opt> may have “64” appended to one of the valid options to
indicate that the -xchange output format is to be a “long-word” format. For
example, -aix64 will request a long-word big-endian format file.

If more than one processing option is specified, the last one specified is the one that will be in effect.
The default options (if one of -c, -h, -q, -t, -x, -? or -b, -l or one of their synonyms are not specified) are

-h If there are no file name options
-q If one or more than two file name options are specified
-x If exactly two file name options are specified.

Error messages such as those describing invalid command options are written to stderr.

Examples:
1. Copy file infile_1.op4 to test_file.op4:
   md20071 op4util -c infile_1.op4 test_file.op4

2. Copy file infile_big_endian.op4 to infile_little_endian.op4,
   changing its endian:
   md20071 op4util -xch infile_big_endian.op4
   infile_little_endian.op4

3. Copy file input_file.op4 to win_file.op4, forcing win_file.op4 to have
   Windows (little-endian) format. Also, generate verbose messages about the conversion process:
   md20071 op4util -wind -v input_file.op4 win_file.op4

4. Dump files input_file.op4 and input_file.op2, generating verbose messages about
   the file formats:
   md20071 op4util -dump-v input_file.op4 input_file.op2
**OPTCONV**

OPTCONV may be used to reformat an existing optimization Bulk Data file used in MSC.Nastran prior to Version 68 into a format compatible with Version 68 or later. The operations performed by this program are described in the *MSC.Nastran Release Notes for Version 68*. The basic format of the “optconv” command is

```plaintext
md20071 optconv input_file [keywords]
```

where `input_file` is the name of the dynamic-optimization data file. If the file type of the old data file is “.dat”, it may be omitted from the command line.

**Keywords**

- `output`  
  `output=pathname`  
  Default: `input-file`  
  This option specifies the name of the reformatted data file. By default, the old output file is renamed by appending the file type “.old”; the new file is the original name of the input file. If an output file is specified using this option, the original input filename is unchanged.

**Examples**

To execute the program, enter the following command:

```plaintext
md20071 optconv example
```

The Version 68-compatible output is written to

```plaintext
example.dat
```

The original data is renamed to example.dat.old.
PLOTPS

PLOTPS reads plotting commands from a single MD Nastran binary- or neutral-format plot file and produces a file that can be printed on a PostScript device. The basic format of the “plotps” command is

```
md20071 plotps input_plot_file [keywords]
```

where *input_plot_file* is the name of the plot file generated by MD Nastran or NEUTRL. A neutral-format plot file can be read from stdin by specifying “-” as the filename. The plot file type “.plt” does not have to be specified on the command line.

**Keyword**

- **begin**
  ```
  begin=number
  Default: 1
  
  end=number
  Default: 999999
  ```
  Plots a selected range of plot frames.

- **color**
  ```
  color=yes,no
  Default: No
  ```
  Enables or disables color pens. Setting “color=no”, the default, will assign a solid line to pen 1 and various dashed lines to pens 2, 3, and 4. Setting “color=yes” will assign black to pen 1, red to pen 2, green to pen 3, and blue to pen 4. All text and axes will always be written with a solid black pen.

- **cscale**
  ```
  cscale=number
  Default: 1.0
  ```
  Specifies a scale factor for all characters and special symbols on the plot. By default, characters and special symbols are 9 points (about 0.125 inch). The scale value, if specified, is also applied to characters and special symbols. The “cscale” value is critical to the correct imaging of the plot if “optimizestrings=yes” was specified. In general, you must specify the same “cscale” value as was specified in the original MD Nastran job that generated the PLT file.

- **dump**
  ```
  dump=yes,no
  Default: No
  ```
  Enables a raw print of each plot command before it is processed. This print is used for debugging purposes only.

- **format**
  ```
  format=keyword
  Default: Binary
  ```
  Specifies the input file format. If the file type of the input file is “.neu” or the plot file is read from stdin, then “format=neutral” is assumed.

- **height**
  ```
  height=number
  Default: 10.0 inches
  ```
  Specifies the printable page height. The actual page is assumed to be 1 inch larger.
### optimizestrings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimizestrings=</td>
<td>Indicates the string optimization feature is to be enabled. This feature can result in a substantial reduction in plot file size, printer memory requirements, and print speed. If “optimizestrings=no” is set, PLOTPS will draw each character individually, at the expense of PS file size and the memory and time needed by your PostScript printer to image the file.</td>
<td></td>
</tr>
<tr>
<td>optimizestrings=yes.no</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

### output

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>output=</td>
<td>Specifies the name of the PostScript output file. If a neutral-format plot file is read from stdin, the default output filename is “plotps.ps”. If “out=—” is specified, the PostScript output is written to stdout. By default, the output file is named the name of the input file with the new type “.ps”.</td>
<td></td>
</tr>
<tr>
<td>pathname</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### rotate

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotate=</td>
<td>Controls the orientation of the generated image. If “rotate=automatic” is specified, the program orients the image so that the long direction of the image is aligned with the long direction of the page. If “rotate=no” is specified, the image is generated with the horizontal axis aligned with the bottom edge of the page. If “rotate=yes” is specified, the image is generated with the horizontal axis aligned with the right edge of the page.</td>
<td></td>
</tr>
<tr>
<td>keyword</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### scale

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>scale=</td>
<td>Specifies a scale factor for all elements of the plot.</td>
<td></td>
</tr>
<tr>
<td>number</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### verbose

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose=</td>
<td>Specifies whether processing messages are to be written.</td>
<td></td>
</tr>
<tr>
<td>yes.no</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### width

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>width=</td>
<td>Specifies the printable page width. The actual page is assumed to be 1 inch larger.</td>
<td></td>
</tr>
<tr>
<td>number</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Note:

The program will not attempt to print a multipage image if this option is used to enlarge the image beyond the size of the available page.

### Examples

To translate a binary-format plot file named example1.plt into PostScript, use

```
md20071 plotps example1
```
The name of the output file is

example1.ps

To translate a neutral-format plot file named example2.neu into PostScript, use

md20071 plotps example2.neu

The name of the output file is

example2.ps

**Using the String Optimization Feature**

When the string optimization feature functions correctly, you can realize a substantial reduction in the size of the PostScript file and a commensurate reduction in the memory and time needed by your PostScript printer to image the file. However, there are some cases where the feature does not function correctly, and generates an incorrect plot image.

The “cscale” value used in the MD Nastran job that generated the PLT file is critical to the correct operation of the “optimizestrings” feature. In general, you need to specify the same value in the PLOTPS run. There are some cases, however, where the value should be left at the default, i.e., 1.0. You can determine this by imaging and printing the first frame of the PLT file with the following two commands:

```bash
md20071 plotps plt-file end=1 out=value.ps cscale=cscale-value
md20071 plotps plt-file end=1 out=default.ps
```

where `plt-file` is the MD Nastran PLT file and `cscale-value` is the CScale value used in the MD Nastran job that generated the file. A visual comparison of the two PostScript images will identify the correct setting. In general, it will be the first command, i.e., the one that set the CScale value to the MD Nastran job’s value.
A summary of PostScript file sizes and “cscale” values is presented below for several TPL files:

<table>
<thead>
<tr>
<th>File name</th>
<th>PLT File Size</th>
<th>PS File Size</th>
<th>&quot;cscale&quot; Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>V70</td>
<td>opt=no</td>
</tr>
<tr>
<td>d10112r</td>
<td>102272</td>
<td>137889</td>
<td>53129</td>
</tr>
<tr>
<td>hd15901</td>
<td>57152</td>
<td>68644</td>
<td>27613</td>
</tr>
<tr>
<td>pt1031</td>
<td>81216</td>
<td>100844</td>
<td>39345</td>
</tr>
<tr>
<td>v14501q</td>
<td>15040</td>
<td>24343</td>
<td>10123</td>
</tr>
</tbody>
</table>

Of these files, only “d10112r” used the default “cscale” value to image correctly.
RCOUT2

RCOUT2 is used to convert a neutral-format OUTPUT2 file generated by MD Nastran into a binary-format OUTPUT2 file. Since MD Nastran can read and write binary-format and neutral-format OUTPUT2 files, this utility is generally used to construct a binary OUTPUT2 file for a third-party program that can only read a binary OUTPUT2 file. The basic format of the “rcout2” command is

```
md20071 rcout2 neutral_output2_file [keywords]
```

where **neutral_output2_file** is the name of the neutral-format OUTPUT2 file. If the file type of the OUTPUT2 file is “.on2”, it may be omitted from the command line.

**Note:** Prior to MSC.Nastran V70, the types “.neut” and “.out2” were used. V70 changed them to the more portable “.on2” and “.op2”, respectively.

**Keywords**

**output**

output=pathname  Default:  neutral_file.op2

This option specifies the name of the binary OUTPUT2 file. By default, the output file is the name of the input file with the new type “.op2”.

**Examples**

To execute the program, enter the following command:

```
md20071 rcout2 example
```

The name of the output file is

```
example.op2
```
CHAPTER 6
Using the Utility Programs

RECEIVE

RECEIVE converts a neutral results database file (NDB) into a binary results database file (XDB). The basic format of the “receive” command is

```
md20071 receive neutral_xdb_file [keywords]
```

where `neutral_xdb_file` is the name of the NDB file. If “-” is specified as the neutral format database file, the file is read from stdin. If the file type of the NDB file is “.ndb”, it may be omitted from the command line.

**Note:** Prior to MSC.Nastran V70, the file type was “.ntrl”; V70 changed this to the more portable “.ndb”.

### Keywords

**output**

```
output=pathname
```

Default: `neutral_xdb_file.xdb`

This option specifies the name of the binary results database file. By default, the output file is the name of the input file with the new type “.xdb”. If the neutral format database file was read from stdin, the default output filename is “receive.xdb”. A binary XDB file cannot be written to stdout.

**verbose**

```
verbose=yes,no
```

Default: Yes

Output is a disk file

NoOutput is stdout.

This option specifies whether processing messages are to be written.

### Examples

To execute the program, enter the following command:

```
md20071 receive example
```

The name of the output file is

```
example.xdb
```
On UNIX systems, an XDB file can be transferred directly from a remote system with the following command:

**HP-UX**

```bash
$ remsh node md20071 trans binary_xdb_file out=- \|
   md20071 receive - out=binary_xdb_file
```

**SUPER-UX**

```bash
$ /usr/ucb/rsh node md20071 trans binary_xdb_file out=- \|
   md20071 receive - out=binary_xdb_file
```

**All Others**

```bash
$ rsh node md20071 trans binary_xdb_file out=- \|
   md20071 receive - out=binary_xdb_file
```

See the rsh(1) man page for further information.
TRANS

A results database file (XDB) may be exchanged between computer systems that have binary file compatibility as displayed in Table 6-2. Otherwise, the TRANS utility is required. TRANS converts an XDB file that is generated by MD Nastran to an equivalent character file that can be sent across a network to another computer. RECEIVE converts the character file back into an XDB file for postprocessing.

Binary File Compatibility

The following table lists the compatibility of binary files between various computer systems supported by current or previous versions of MSC products. Note that not all of these combinations have been tested by MSC. Please report any compatibility problems encountered to your MSC representative.

<table>
<thead>
<tr>
<th>MD Nastran</th>
<th>Architecture</th>
<th>Postprocessor Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IEEE Byte Order Word Size HP Alpha Compaq VAX HP IBM pSeries SGI Sun SPARC Intel</td>
<td></td>
</tr>
<tr>
<td>Cray UNICOS</td>
<td>No Big 64 TR TR TR TR TR TR TR</td>
<td></td>
</tr>
<tr>
<td>Cray IEEE T90 UNICOS</td>
<td>Yes Big 64 TR TR TR TR TR TR TR</td>
<td></td>
</tr>
<tr>
<td>HP Alpha UNIX</td>
<td>Yes Little 32 Copy TR TR TR Copy Copy Copy Copy Copy</td>
<td></td>
</tr>
<tr>
<td>Compaq Alpha OpenVMS</td>
<td>Yes Little 32 TR TR TR Copy Copy Copy Copy Copy</td>
<td></td>
</tr>
<tr>
<td>Compaq VAX OpenVMS</td>
<td>No Little 32 TR Copy TR TR TR TR TR</td>
<td></td>
</tr>
<tr>
<td>Fujitsu VPP VX UX/PU</td>
<td>Yes Big 32 TR Copy TR Copy Copy Copy Copy Copy</td>
<td></td>
</tr>
<tr>
<td>HP HP-UX, SPP-UX</td>
<td>Yes Big 32 TR Copy TR Copy Copy Copy Copy Copy</td>
<td></td>
</tr>
<tr>
<td>HP C-Series ConvexOS</td>
<td>Yes Big 32 TR Copy TR Copy Copy Copy Copy Copy</td>
<td></td>
</tr>
<tr>
<td>Hitachi S-Series HS-OIF/1-MJ</td>
<td>No Big 32 TR TR TR TR TR TR</td>
<td></td>
</tr>
<tr>
<td>IBM S.390 MVS/SA, VM</td>
<td>No Big 32 TR TR TR TR TR TR</td>
<td></td>
</tr>
<tr>
<td>IBM pSeries AIX</td>
<td>Yes Big 32 TR Copy Copy Copy Copy Copy Copy</td>
<td></td>
</tr>
</tbody>
</table>

Table 6-2: Binary File Compatibility
## MD Nastran Architecture

<table>
<thead>
<tr>
<th>MD Nastran</th>
<th>IEEE</th>
<th>Byte Order</th>
<th>Word Size</th>
<th>HP Alpha</th>
<th>Compaq VAX</th>
<th>IBM pSeries</th>
<th>SGI</th>
<th>Sun SPARC</th>
<th>Intel</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEC SUPER-UX</td>
<td>Yes</td>
<td>Big</td>
<td>64</td>
<td>TR</td>
<td>TR</td>
<td>TR</td>
<td>TR</td>
<td>TR</td>
<td>TR</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>Big</td>
<td>32</td>
<td>Copy³</td>
<td>Copy</td>
<td>Copy</td>
<td>Copy</td>
<td>Copy</td>
<td>Copy³</td>
</tr>
<tr>
<td>SGI IRIX, IRIX64</td>
<td>Yes</td>
<td>Big</td>
<td>32</td>
<td>TR</td>
<td>Copy³</td>
<td>TR</td>
<td>Copy</td>
<td>Copy</td>
<td>Copy</td>
</tr>
<tr>
<td>Sun SPARC Solaris</td>
<td>Yes</td>
<td>Big</td>
<td>32</td>
<td>TR</td>
<td>Copy³</td>
<td>TR</td>
<td>Copy</td>
<td>Copy</td>
<td>Copy³</td>
</tr>
<tr>
<td>Intel Linux, Linux64 Linuxx8664 Windows</td>
<td>Yes</td>
<td>Little</td>
<td>32</td>
<td>Copy</td>
<td>TR</td>
<td>TR</td>
<td>TR</td>
<td>TR</td>
<td>Copy</td>
</tr>
</tbody>
</table>

### Notes:

1. Copy³ indicates that using the 2001.0.1 or later released version of the DBIO library, the XCB files produced by MD Nastran can be transferred between the systems without using the TRANS and RECEIVE programs.

2. Copy indicates that XDB files can be transferred between the systems without using TRANS and RECEIVE.

3. TR indicates that XDB files must be transferred between the systems using TRANS and RECEIVE.

4. With MSC.Nastran V2005, SUPER-UX switched from an ILP-64 build to a LP-64 build. This changed the data type from 64-bits to 32-bits, making it binary compatible with other 32-bit, big-endian, IEEE systems, and incompatible with earlier SUPER-UX versions.

The first column on the left of the table lists various platforms that run MD Nastran. The second and third columns list basic architectural features of the computer, specifically whether the computer conforms to ANSI/IEEE Standard 754-1985 (the *IEEE Standard for Binary Floating-Point Arithmetic*) and byte ordering (big endian or little endian) used by the computer. The remaining columns list postprocessor platforms.

### Running TRANS

TRANS converts a binary results database file (XDB) into a neutral results database file (NDB) that may be copied to any other computer. The basic format of the “trans” command is

```
md20071 trans binary_xdb_file [keywords]
```
where \texttt{binary\_xdb\_file} is the name of the XDB file. An XDB file cannot be read from stdin. If the file type of the XDB file is "\.xdb", it may be omitted from the command line.

## Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
</table>
| \texttt{alphabet} | \texttt{alphabet=number} Default: 64  
Choose the 48- or 64-character conversion table. |
| \texttt{output} | \texttt{output=pathname} Default: \texttt{binary\_xdb\_file.ndb}  
This option specifies the name of the neutral format database file. If "\texttt{out=-}" is specified, the neutral-format database file will be written to stdout. By default, the output file name is the input file name with the new type "\.ndb". (Prior to MSC.Nastran V70 the type "\.ntrl" was used; V70 changed this to the more transportable "\.ndb"). |
| \texttt{verbose} | \texttt{verbose=yes, no} Default: \texttt{yes}  
\texttt{Output is a disk file}  
\texttt{No Output is stdout.}  
This option specifies whether processing messages are to be written. |

## Examples

To execute the program, enter the following command:

```
md20071 trans example
```

The name of the output file is

```
exmaple.ndb
```
On UNIX systems, an XDB file can be transferred directly to a remote system with the following commands:

**HP-UX**
```
md20071 trans binary_xdb_file out=- \r
    remsh node [-l user] md20071 receive - out=binary_xdb_file
```

**SUPER-UX**
```
md20071 trans binary_xdb_file out=- \r
    /usr/ucb/rsh node [-l user] md20071 receive - \r
    out=binary_xdb_file
```

**All others**
```
md20071 trans binary_xdb_file out=- \r
    rsh node [-l user] md20071 receive - out=binary_xdb_file
```

See the `remsh(1)` or `rsh(1)` man pages for further information.
CHAPTER 6
Using the Utility Programs

XMONAST (UNIX)

XMONAST is a simple OSF/Motif GUI to monitor MD Nastran jobs. The Motif runtime libraries along with an X-capable terminal/monitor are required to run XMONAST. The basic format of the “xmonast” command is:

```
md20071 xmonast list_of_files &
```

XMONAST is a point-and-click text file viewer that can view the output of your MD Nastran job as it progresses. The viewer can be started in three ways:

- From the command line:

  ```
  md20071 xmonast list_of_files &
  ```

  where `list_of_files` are text files that will be displayed. XMONAST will read stdin if “-” is specified.

- From the nastran command (with manual termination of XMONAST):

  ```
  md20071 nastran input_file... xmon=yes
  ```

  where only the `.log` file will be displayed. From the nastran command (with automatic termination of XMONAST when the MD Nastran job ends):

  ```
  md20071 nastran data_file ... xmon=kill
  ```

  where only the `.log` file will be displayed.

See the “xmonast” keyword (page 348) for more details on these methods.

The selected files will be displayed in scrollable windows. Once the entire file as it currently exists has been displayed, XMONAST will enter an infinite loop waiting for additional text. This process will continue until the “Exit” push button is selected, or until the MD Nastran job has completed if XMONAST is started from the nastran command with “xmon=kill”.

You may temporarily suspend updates to the scrollable windows (e.g., to browse the output) by selecting the “Pause Output” push button. To resume output, select the same button, now labeled “Continue Output”.

If a `.log` file is being displayed, the “Kill Job” push button may be used to cancel a running MD Nastran job. This will send an interrupt kill signal (SIGKILL) to your MD Nastran job. Unless started by “xmon=kill”, you may still scroll through the output data files after terminating a job.
To exit XMONAST, select the “Exit” push button or select “File --> Exit” from the menu bar.

**Menu Bar Commands**

<table>
<thead>
<tr>
<th>Button</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Re-Open Files</td>
<td>Rereads the input files from the beginning (does not function for stdin).</td>
</tr>
<tr>
<td>Exit</td>
<td></td>
<td>Writes various resources to “$HOME/Xmonast” and exits XMONAST.</td>
</tr>
<tr>
<td>Kill</td>
<td>Sure Kill</td>
<td>Sends signal SIGKILL (9) to the MD Nastran job. This command is only enabled if an MSC.Nastran Version 68.1 (or later) .log file is being displayed in one of the panes. If more than one .log file is being displayed, only one of the jobs will be killed.</td>
</tr>
<tr>
<td>Help</td>
<td>Online Documentation</td>
<td>Starts the online documentation application. The command used by XMONAST to start the application is specified by the “Xmonast*docname” resource. The default value is “mne”, i.e., the MD Nastran Encyclopedia (a separately installed product).</td>
</tr>
<tr>
<td></td>
<td>Program Version...</td>
<td>Displays the program version in a pop-up window. Press the “OK” button to dismiss the pop-up window.</td>
</tr>
</tbody>
</table>

**Buttons**

<table>
<thead>
<tr>
<th>Button</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pause Output</td>
<td></td>
<td>Suspends output to the panes so that they can be examined. The button will change to “Continue Output” while the output is paused. Pressing “Continue Output” will resume output to the panes.</td>
</tr>
<tr>
<td>Kill</td>
<td></td>
<td>Sends signal SIGKILL (9) to the MD Nastran job. The button is only enabled if an MSC.Nastran V68.1 (or later) .log file is being displayed in one of the panes. If more than one .log file is being displayed, only one of the jobs will be killed.</td>
</tr>
<tr>
<td>Exit</td>
<td></td>
<td>Writes the current resource settings to “$HOME/Xmonast” and exits.</td>
</tr>
</tbody>
</table>

**Examples**

To monitor the .f06, .f04, and .log files of an already running job named example, use:

```
md20071 xmonast example.f06 example.f04, example.log &
```
To run an MD Nastran job named example in the background and monitor the .log file as the job progresses, use:

```
md20071 nastran example batch=yes xmonast=yes
```

XMONAST will continuously display the .log file until the “Exit” push button is selected.

**Resources**

The default resource file, “/usr/lib/X11/app-defaults/Xmonast”, and, if it exists, your resource file, “$HOME/Xmonast”, are read at application startup. Your resource file is completely rewritten if XMONAST is terminated using the “File->Exit” menu item or the “Exit” button at the bottom of the window. Your resource file is not written if you terminate XMONAST using the “window->Close” menu item. Documentation of the XMONAST resources can be found in the standard MD resource file, “install_dir/md20071/arch/Xmonast”.
XNASTRAN (UNIX)

XNASTRAN is a simple OSF/Motif Graphical User Interface to submit MD Nastran jobs. The Motif runtime libraries along with an X-capable terminal/monitor are required to run XNASTRAN. The basic format of the “xnastran” command is:

```
md20071 xnastran &
```

The XNASTRAN command allows you to select the input file, set job options (i.e, command line keywords), and submit the job to MD Nastran.

**Menu Bar Commands**

<table>
<thead>
<tr>
<th>File</th>
<th>Exit</th>
<th>Exits XNASTRAN.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td>MD Nastran Version...</td>
<td>Allows you to enter the “MD Nastran Version Label” defining the product name, the default is “MD Nastran V2006” and the “Run Command” that submits a job, the default is ‘/usr/bin/md20071 nastran’. The “Select File” button will bring up a standard file selection tool allowing you to find the run command file. The “Accept” button will accept the changes and cancel the dialog, the “Cancel” button will cancel the dialog with making any changes, and the “Help” button will bring up a help window; select the “Close” button to dismiss the help dialog.</td>
</tr>
<tr>
<td>System Default</td>
<td></td>
<td>Resets various defaults, including the window size, the “MD Nastran Version Label”, and the “Run Command”.</td>
</tr>
<tr>
<td>Save</td>
<td></td>
<td>Writes all the current settings, including the various entries, to the “$HOME/Xnastran” resource file. These values will be reloaded the next time you enter XNASTRAN.</td>
</tr>
<tr>
<td>Help</td>
<td>Online Documentation</td>
<td>Starts the online documentation application. The command used by XNASTRAN to start the application is specified by the “Xnastran*docname” resource. The default value is “mne”, i.e., the MD Nastran Encyclopedia (a separately installed product).</td>
</tr>
<tr>
<td>Program Version...</td>
<td></td>
<td>Displays the program version in a pop-up window. Select the “OK” button to dismiss the pop-up window.</td>
</tr>
</tbody>
</table>

**Main Window Items**

Each item in the main window includes a “Help” button. Selecting the help item will bring up a short help dialog; the dialog is dismissed with the “Close” button.
The items in the main window are listed below as they appear from the top of the window to the bottom.

**Input Data File**
This subpane allows you to enter the name of the input file using the keyboard or with a file selection tool if the “Select File” button is selected.

**Scratch Directory**
This subpane allows you to enter the name of the scratch directory using the keyboard or with a directory selection tool if the “Select Directory” button is selected. This sets the “sdirectory” keyword.

**Database Prefix**
This subpane allows you to enter the prefix of the database files using the keyboard. This sets the “dbs” keyword; if the text field is empty, the “dbs” keyword is not set.

**Monitor Output**
This subpane allows you to start the XMONAST utility to monitor the .f06, .f04, and .log files. This sets “xmon=yes” or “xmon=no”.

**Background Process**
This subpane allows you to run the job in the background. This sets “batch=yes” or “batch=no”.

**Combine Files**
This subpane allows you to append the .f06, .f04, and .log files into a single OUT file. This sets “append=yes” or “append=no”.

**Delete Databases**
This subpane allows you to delete the user databases at the completion of the MD Nastran job or select a “mini” database. This sets “scratch=yes”, “scratch=no”, or “scratch=mini”.

**Display News**
This subpane allows you to display the MD Nastran system news in the .f06. This sets “news=yes” or “news=no”.

**Print Output Files**
This subpane allows you to print the .f06, .f04, and .log files at the completion of the MD Nastran job. This sets “prt=yes” or “prt=no”.

**Send Notification**
This subpane allows you to receive notification when the MD Nastran job completes. This sets “notify=yes” or “notify=no”.

**Save Previous**
This subpane allows you to version old output files before the MD Nastran job begins. This sets “old=yes” or “old=no”.

**Output Prefix**
This subpane allows you to enter the prefix of the output files using the keyboard. This sets the “out” keyword; if the text field is empty, the “out” keyword is not set.

**Start Time**
This subpane allows you to select a job starting time using the keyboard. This sets the “after” keyword; if the text field is empty, the “after” keyword is not set.

**Queue Name**
This subpane allows you to select the starting time of the job using the keyboard. This sets the “after” keyword.
Advanced Keywords
This subpane allows you to enter any additional keywords using the keyboard. You must enter the complete text of any keywords to be set. If the text field is empty, no additional keywords are set.

Memory Size
This subpane allows you to enter the memory allocation using the keyboard. The pop-up menu allows you to select the units modifier, i.e., none, “Kb”, “Kw”, “Mb”, “Mw”, “Gb”, “Gw”. This sets the “memory” keyword.

Submit MD Nastran
This button submits the job using the parameters displayed in the window.

Resources
The default resource file, “/usr/lib/X11/app-defaults/Xnastran”, and, if it exists, your resource file, “$HOME/Xnastran”, are read at application startup. Your resource file is completely rewritten if you select the “Setup->Save” menu item. Documentation of the XNASTRAN resources can be found in the standard MD resource file, “install_dir/md20071/arch/Xnastran”.
Building the Utilities Delivered in Source Form

Several of the utilities (i.e., PLOTPS, NEUTRL, RCOUT2, and MSCACT) are delivered in source and executable form. The source code allows these utilities to be customized or built for other platforms. A script and makefile are provided to build and install these utilities. The script determines the architecture of current platform and invokes the make utility to perform the actual compilation, link, and installation.

The utility program source files are located in

```
install_dir/md20071/util
```

on UNIX and

```
install_dir\md20071\util
```

on Windows. This directory is an optional component of the MD Nastran installation. This directory includes the following files:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ld.f</td>
<td>Source for RCOUT2 Utility Routines.</td>
</tr>
<tr>
<td>libfmsc.F</td>
<td>Source for FORTRAN Utility Library Routines.</td>
</tr>
<tr>
<td>makefile</td>
<td>Makefile to Build Source Utility Programs.</td>
</tr>
<tr>
<td>mattst.F</td>
<td>Source for Sample OUTPUT4 File Reader TABTST (see “Building and Using TATBST” on page 277).</td>
</tr>
<tr>
<td>mscact.c</td>
<td>Source for MSC Accounting Programs.</td>
</tr>
<tr>
<td>neutrl.F</td>
<td>Source for NEUTRL Utility.</td>
</tr>
<tr>
<td>ngtarg.F</td>
<td>Source for Command Line Utilities.</td>
</tr>
<tr>
<td>plotps.F</td>
<td>Source for PLOTPS Utility.</td>
</tr>
<tr>
<td>rcout2.F</td>
<td>Source for RCOUT2 Utility.</td>
</tr>
<tr>
<td>tabtst.F</td>
<td>Source for Sample OUTPUT2 File Reader MATTST (see “Building and Using MATTST” on page 273).</td>
</tr>
<tr>
<td>util</td>
<td>Script to Build Source Utility Programs.</td>
</tr>
</tbody>
</table>

Three steps are required to build and install the source utilities. Make sure that you are in the utility program source directory, i.e., `install_dir/md20071/util` on UNIX and `install_dir\md20071\util` on Windows.
1. The first step compiles and links all of the source utility programs. Enter the command

```
md20071 util build
```

If only one utility is to be built, use the name of the utility (i.e., “mscact,” “neutrl,” “plotps,” or “rcout2”) instead of “build”. For example,

```
md20071 util plotps
```

will only build the PLOTPS utility.

2. After the programs are generated in the current directory, you can install the executable programs into the architecture directory for your computer (i.e., `install_dir/md20071/arch` on UNIX and `install_dir\md20071\arch` on Windows). Enter the command

```
md20071 util install
```

3. The third step deletes all object files and temporary files created by the “make” process. Enter the command

```
md20071 util clean
```

The building and installation process can be repeated if you want to build the utilities for other computer architectures at your site.

To build the utilities on another computer that does not have MD Nastran installed, copy the complete utilities directory to the other computer. Since the `md20071` command will not be available, you must run the `util` script directly. Before you do, however, set the environment variable `MSC_ARCH` to the name of a supported architecture as shown in Table 3-1. The “install” option cannot be used.
Building and Using the Sample Programs

- Overview 260
- Building and Using BEAMSERV 261
- Building and Using DDPRT 263
- Building and Using DDLQRY 265
- Building and Using DEMO1 267
- Building and Using DEMO2 269
- Building and Using DR3SERV 271
- Building and Using MATTST 273
- Building and Using SMPLR 275
- Building and Using TABTST 277
- Beam Server Source Files 279
- DRESP3 Server Source Files 280
- MSC.Access Source Files 281
Overview

This chapter describes how to build and use the various MD Nastran sample programs. The sample programs are grouped by function as follows:

<table>
<thead>
<tr>
<th>Program</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEAMSERV</td>
<td>Implements user-defined bar and beam elements for MD Nastran.</td>
</tr>
<tr>
<td>DDLPPRT</td>
<td>Reads and displays XDB results database files. These sample programs are part of MSC.Access and demonstrate how to use the database library routines.</td>
</tr>
<tr>
<td>DDLQRY</td>
<td></td>
</tr>
<tr>
<td>DEMO1</td>
<td></td>
</tr>
<tr>
<td>DEMO2</td>
<td></td>
</tr>
<tr>
<td>SMPLR</td>
<td></td>
</tr>
<tr>
<td>DR3SERV</td>
<td>Implements user-defined responses for MD Nastran.</td>
</tr>
<tr>
<td>MATTST</td>
<td>Reads and displays OUTPUT2 and OUTPUT4 files.</td>
</tr>
<tr>
<td>TABST</td>
<td></td>
</tr>
</tbody>
</table>

Descriptions on building and using the sample programs follow in alphabetical order.
Building and Using BEAMSERV

BEAMSERV implements a user-defined beam element for MD Nastran.

**Note:** The sample beam server source code is only provided as a simple example illustrating basic concepts. It is not intended to be a complete or usable program.

Unlike the other sample programs, a beam server is not a stand-alone program that runs from the command line. Instead, the beam server is started and controlled by MD Nastran. In the current implementation, communications between MD Nastran and the beam server are accomplished through pipes, with MD Nastran reading and writing BEAMSERV’s stdout and stdin units, respectively.

**Notes:**

1. The MD Nastran job invoking the beam server and the beam server itself may run on different computers but they have to be network mounted.
2. Your program may not read from stdin (FORTRAN logical unit 5) nor write to stdout (FORTRAN logical unit 6).
3. The beam server cannot write to the .f06, .f04, or .log files of the MD Nastran job that started the beam server.
4. Debugging must be accomplished by writing to a disk file, or connecting to the running beam server executable with a debugger (this may not be available on all systems, and debug compiler options should be used).

Building BEAMSERV

The DR3SERV program source files are located in the directory

```
install_dir/md20071/bmsrv
```

on UNIX and

```
install_dir\md20071\bmsrv
```

on Windows (see “Beam Server Source Files” on page 279).

To build the program, change the working directory to the bmsrv directory and enter the command:

```
md20071 bmsrv build
```
If you do not have write access to `install_dir/md20071/beamsrv`, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
md20071 ./beamsrv build
```

**Using BEAMSERV**

MD Nastran is made aware of the beam server by the “gmconn” keyword and an external evaluator connection file. Entries in the connection file for piped communications are formatted as follows:

```
evaluatorname,pipe,pathname
```

where `evaluatorname` is the evaluator name defined on the CONNECT FMS statement and `pathname` is the pathname of the beam server executable.

**Note:** The evaluator name on the CONNECT FMS statements and in the external evaluator connection file must match exactly, including character case. To use a mixed or lower case group name, the name on the CONNECT FMS statement must be in quote marks; the name in the external evaluator connection file is never quoted.

To use the sample beam server and data file, create the file “samp_eval” with the following line:

```
LOCBMLS,pipe,pathname
```

where `pathname` is the pathname of the beam server built above, e.g.,

- `install_dir/md20071/arch/beamserv` or `./beamserv` on UNIX
- `install_dir/md20071\arch\beamserv` or `.\beamserv` on Windows.

MD Nastran is then run using the following command:

```
md20071 nastran sample gmconn=samp_eval
```
Building and Using DDLPRF

**DDLPRF** illustrates the mass retrieval of data from the MSC.Access Data Definition Language (DDL) database.

### Building DDLPRF

The DDLPRF program source code is in the file “ddlprt.F” (see “DRESP3 Server Source Files” on page 280). To build the program, change the working directory to the access directory and type the command:

```plaintext
md20071 access ddlprt
```

If you do not have write access to the source directory, `install_dir/`md20071/access on UNIX and `install_dir/md20071/access` on Windows, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```plaintext
md20071 ./access ddlprt
```

on UNIX, or

```plaintext
md20071 .\access ddlprt
```

on Windows. Note, the directory specification is required in this circumstance.

### Using DDLPRF

DDLPRF is run with the “ddlprt” command. The format of the “ddlprt” command is

```plaintext
md20071 ddlprt [ddl_xdb_file] [keywords]
```

If the DDL XDB file is not specified, the program uses the default MSC.Access DDL file, `install_dir/md20071/arch/dbc.xdb` on UNIX and `install_dir/md20071/arch/dbc.xdb` on Windows. The optional keywords are:
To execute the program, enter the command

```
md20071 ddlpert
```

The program displays the filename, version, and compilation date of the DDL file as well as the names of the print and table of contents files. Once these files are generated, the program exits. The print and table of contents files may then be printed once DDLPRRT has completed.
**Building and Using DDLQRY**

DDLQRY illustrates the interactive retrieval of data from the MSC.Access Data Definition Language (DDL) database.

**Building DDLQRY**

The DDLQRY program source code is in the file “ddlqry.F” (see “DRESP3 Server Source Files” on page 280). To build the program, change the working directory to the access directory and type the command:

```
md20071 access ddlqry
```

If you do not have write access to the source directory, `install_dir`md20071/access on UNIX or `install_dir`md20071/access on Windows, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
md20071 .access ddlqry
```

on UNIX, or

```
md20071 .access ddlqry
```

on Windows. Note, the directory specification is required in this circumstance.

**Using DDLQRY**

DDLQRY is run with the “ddlqry” command. The format of the “ddlqry” command is

```
md20071 ddlqry [ddl_xdb_file]
```

If a file is not specified, the program uses the default MSC.Access DDL file, `install_dir/md20071/arch/dbc.xdb` on UNIX and `install_dir/md20071/arch/dbc.xdb` on Windows.

The program displays the filename, version, and compilation date of the DDL file and prompts you for the name of a DDL object:

```
Enter Object Name (null to quit)
```
After you enter the name of each object, the format of the object is displayed. The program repeats the prompt until a blank line is entered.
Building and Using DEMO1

DEMO1 prints information about a results database (XDB) file produced by MD Nastran.

Note: The sample program source code is only provided as a simple example illustrating basic concepts. It is not intended to be a complete or usable program.

Building DEMO1

The DEMO1 program source code is in the file "demo1.f" (see “DRESP3 Server Source Files” on page 280). To build the program, change the working directory to the access directory and type the command:

    md20071 access demo1

If you do not have write access to the source directory, install_dir/md20071/access on UNIX or install_dir\md20071\access on Windows, copy the entire directory to another location, change the working directory to the new location, and issue the command:

    md20071 .\access demo1

on UNIX, or

    md20071 .\access demo1

on Windows. Note, the directory specification is required in this circumstance.

Using DEMO1

DEMO1 is run using the “demo1” command. The installed version of the program is run with the command:

    md20071 demo1

You are prompted for the input graphics database filename.

Enter the database path name:
Running MD Nastran with a101x.dat (in \texttt{install\_dir/md20071/access}) produces a101x.xdb that may be used as input to this program.
Building and Using DEMO2

DEMO2 prints information about a results database (XDB) file produced by MD Nastran.

Note: The sample program source code is only provided as a simple example illustrating basic concepts. It is not intended to be a complete or usable program.

Building DEMO2

The DEMO2 program source code is in the file “demo2.f” (see “DRESP3 Server Source Files” on page 280). To build the program, change the working directory to the access directory and type the command:

```
md20071 access demo2
```

If you do not have write access to the source directory, `install_dir/md20071/access` on UNIX or `install_dir\md20071\access` on Windows, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
md20071 ./access demo2
```

on UNIX, or

```
md20071 .\access demo2
```

on Windows. Note, the directory specification is required in this circumstance.

Using DEMO2

DEMO2 is run using the “demo2” command. The installed version of the program is run with the command:

```
md20071 demo2
```

You are prompted for the input graphics database filename.

```
Enter the database path name:
```
Running MD Nastran with a61x.dat (in install_dir/md20071/access) produces a101x.xdb that may be used as input to this program.
Building and Using DR3SERV

DR3SERV implements user-defined responses for MD Nastran.

**Note:** The sample DRESP3 server source code is only provided as a simple example illustrating basic concepts. It is not intended to be a complete or usable program.

Unlike the other sample programs, a DRESP3 server is not a stand alone program that runs from the command line. Instead, the DRESP3 server is started and controlled by MD Nastran. In the current implementation, communications between MD Nastran and the DRESP3 server are accomplished through pipes, with MD Nastran reading and writing DR3SERV’s stdout and stdin units, respectively.

**Notes:**

1. The MD Nastran job invoking the DRESP3 server and the DRESP3 server itself may run on different computers but they have to be network mounted.
2. Your program may not read from stdin (FORTRAN logical unit 5) nor write to stdout (FORTRAN logical unit 6).
3. The DRESP3 server cannot write to the .f06, .f04, or .log files of the MD Nastran job that started the DRESP3 server.
4. Debugging must be accomplished by writing to a disk file, or connecting to the running dresp3 server executable with a debugger (this may not be available on all systems, and debug compiler options should be used).

Building DR3SERV

The DR3SERV program source files are located in the directory

```
install_dir/md20071/dr3srv
```

on UNIX and

```
install_dir\md20071\dr3srv
```

on Windows (see “DRESP3 Server Source Files” on page 280).

To build the program, change the working directory to the dr3srv directory and enter the command:

```
md20071 dr3srv build
```
If you do not have write access to `install_dir/md20071/dr3srv`, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
md20071 ./dr3srv build
```

**Using DR3SERV**

MD Nastran is made aware of the DRESP3 server by the “gmconn” keyword and an external evaluator connection file. Entries in the connection file for piped communications are formatted as follows:

```
evaluatorname,pipe,pathname
```

where `evaluatorname` is the evaluator name defined on the CONNECT FMS statement and `pathname` is the pathname of the DRESP3 server executable.

**Note:** The evaluator name on the CONNECT FMS statements and in the external evaluator connection file must match exactly, including character case. To use a mixed or lower case group name, the name on the CONNECT FMS statement must be in quote marks; the name in the external evaluator connection file is never quoted.

To use the sample DRESP3 server and data file, create the file “samp_eval” with the following line:

```
myrsp,pipe,pathname
```

where `pathname` is the pathname of the DRESP3 server built above, e.g., `install_dir/md20071/arch/dr3serv` or `./dr3serv` on UNIX and `install_dir/md20071\arch\dr3serv` or `\dr3serv` on Windows.

MD Nastran is then run using the following command:

```
md20071 nastran sample gmconn=samp_eval
```
Building and Using MATTST

MATTST reads a binary format OUTPUT4 matrix.

**Note:** The sample program source code is only provided as a simple example illustrating basic concepts. It is not intended to be a complete or usable program.

**Building MATTST**

The MATTST program source code is in the file “mattst.f” (see “Building the Utilities Delivered in Source Form” on page 257). To build the program, change the working directory to the util directory and type the command:

```
md20071 util mattst
```

If you do not have write access to the source directory, `install_dir/md20071/util` on UNIX or `install_dir\md20071\util` on Windows, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
md20071 ./util mattst
```

on UNIX, or

```
md20071 .\util mattst
```

on Windows. Note, the directory specification is required in this circumstance.

**Using MATTST**

MATTST is run with the “mattst” command. The installed version of the program is run with the command:

```
md20071 mattst
```

You are prompted for the number of matrices.

```
Please enter the number of matrices:
```
You are prompted for the input filename.

Please enter the INPT4 FILENAME:

You are prompted for the output binary filename.

Please enter the output binary filename:

You are prompted for the output text filename.

Please enter the output text filename:

Running the MD Nastran job “DEMODIR:um54.dat” produces a file, “um54.f11”, that may be used as input to this program.
Building and Using SMPLR

SMPLR reads a results database (XDB) file produced by MD Nastran.

**Note:** The sample program source code is only provided as a simple example illustrating basic concepts. It is not intended to be a complete or usable program.

### Building SMPLR

The SMPLR program source code is in the file “smplr.f” (see “DRESP3 Server Source Files” on page 280). To build the program, change the working directory to the access directory and type the command:

```bash
md20071 access smplr
```

If you do not have write access to the source directory, `install_dir/md20071/access` on UNIX or `install_dir\md20071\access` on Windows, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```bash
md20071 ./access smplr
```

on UNIX, or

```bash
md20071 .\access smplr
```

on Windows. Note, the directory specification is MSC.Access in this circumstance.

### Using SMPLR

SMPLR is run using the “smplr” command. The installed version of the program is run with the command:

```bash
md20071 smplr
```

You are prompted for the input filename.

**Enter the database name to process:**
Running MD Nastran with a101x.dat (see “DRESP3 Server Source Files” on page 280) produces a101x.xdb that may be used as input to this program.
Building and Using TABTST

TABTST reads a binary format OUTPUT2 file (do not confuse this program with RCOUT2, described in “RCOUT2” on page 244).

**Note:** The sample program source code is only provided as a simple example illustrating basic concepts. It is not intended to be a complete or usable program.

Building TABTST

The TABTST program source code is in the file “tabtst.f” (see “Building the Utilities Delivered in Source Form” on page 257). To build the program, change the working directory to the util directory and type the command:

```
md20071 util tabtst
```

If you do not have write access to the source directory, `install_dir`md20071/util on UNIX or `install_dir`md20071/util on Windows, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
md20071 ./util tabtst
```

on UNIX, or

```
md20071 .\util tabtst
```

on Windows.

**Note:** The directory specification is **required** in this circumstance.

Using TABTST

TABTST is run with the “tabtst” command. The installed version of the program is run with the command:

```
md20071 tabtst
```
You are prompted for the input filename.

Please type the INPUT2 filename:

You are prompted for the output filename.

Please type the output filename:

Running the MD Nastran job “TPLDIR:tabtsta.dat” produces a file, “tabtsta.f11”, that may be used as input to this program.
Beam Server Source Files

The BEAMSERV program source files are located in the beam server source directory, i.e.,`install_dir/md20071/bmsrv` on UNIX and `install_dir\md20071\bmsrv` on Windows. This directory is an optional component of the MD Nastran installation. Table 7-1 lists files contained in this directory.

Table 7-1  Beam Server Sample Program Source Files

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bmsrv</td>
<td>UNIX script to build the sample beam server program.</td>
</tr>
<tr>
<td>brtucd.F</td>
<td>Source for sample beam server subroutine BRTUCD.</td>
</tr>
<tr>
<td>brtugd.F</td>
<td>Source for sample beam server subroutine BRTUGD.</td>
</tr>
<tr>
<td>brtuid.F</td>
<td>Source for sample beam server subroutine BRTUID.</td>
</tr>
<tr>
<td>brtupd.F</td>
<td>Source for sample beam server subroutine BRTUPD.</td>
</tr>
<tr>
<td>bsbrcd.F</td>
<td>Source for sample beam server subroutine BSBRCD.</td>
</tr>
<tr>
<td>bsbrgd.F</td>
<td>Source for sample beam server subroutine BSBRGD.</td>
</tr>
<tr>
<td>bsbrid.F</td>
<td>Source for sample beam server subroutine BSBRID.</td>
</tr>
<tr>
<td>bsbrpd.F</td>
<td>Source for sample beam server subroutine BSBRPD.</td>
</tr>
<tr>
<td>bsbrt.F</td>
<td>Source for sample beam server subroutine BSBRT.</td>
</tr>
<tr>
<td>bscon.F</td>
<td>Source for sample beam server subroutine BSCON.</td>
</tr>
<tr>
<td>bsgrq.F</td>
<td>Source for sample beam server subroutine BSGRQ.</td>
</tr>
<tr>
<td>bsmgs.F</td>
<td>Source sample beam server subroutine BSMSG.</td>
</tr>
<tr>
<td>makefile</td>
<td>Makefile to build the sample beam server program.</td>
</tr>
<tr>
<td>main.c</td>
<td>Source for sample beam server main program.</td>
</tr>
<tr>
<td>mevbrd.F</td>
<td>Source for sample beam server subroutine MEVBRD.</td>
</tr>
<tr>
<td>msbrcd.F</td>
<td>Source for sample beam server subroutine MSBRCD.</td>
</tr>
<tr>
<td>msbrgd.F</td>
<td>Source for sample beam server subroutine MSBRGD.</td>
</tr>
<tr>
<td>msbrid.F</td>
<td>Source for sample beam server subroutine MSBRID.</td>
</tr>
<tr>
<td>sample.dat</td>
<td>MD Nastran sample data file.</td>
</tr>
</tbody>
</table>
DRESP3 Server Source Files

The DR3SERV program source files are located in the DRESP3 server source directory, i.e., 
install_dir/md20071/dr3srv on UNIX and install_dir\md20071\dr3srv on Windows. This directory is an
optional component of the MD Nastran installation. Table 7-2 lists files contained in this directory.

Table 7-2  DRESP3 Server Sample Program Source Files

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dr3srv</td>
<td>UNIX script to build the sample DRESP3 server program.</td>
</tr>
<tr>
<td>makefiles</td>
<td>Makefile to build the sample DRESP3 server program.</td>
</tr>
<tr>
<td>r3sgtr.F</td>
<td>Sample source for the DRESP3 server.</td>
</tr>
<tr>
<td>r3svld.F</td>
<td>Sample source for the DRESP3 server.</td>
</tr>
<tr>
<td>r3svls.F</td>
<td>Sample source for the DRESP3 server.</td>
</tr>
</tbody>
</table>
MSC.Access Source Files

The MSC.Access sample source files are located in the MSC.Access source directory `install_dir/md20071/access` on UNIX and `install_dir/md20071/access` on Windows. This directory is an optional component of the MD Nastran installation. Table 7-3 lists files contained in this directory.

Table 7-3  MSC.Access Sample Program Source Files

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a101x.dat</td>
<td>MD Nastran data file.</td>
</tr>
<tr>
<td>access</td>
<td>Script to build MSC.Access sample programs.</td>
</tr>
<tr>
<td>ddlprt.F</td>
<td>Demonstration database dictionary print program.</td>
</tr>
<tr>
<td>ddlqry.F</td>
<td>Demonstration database dictionary query program.</td>
</tr>
<tr>
<td>demo1.F</td>
<td>Source for sample MD Nastran database reader.</td>
</tr>
<tr>
<td>demo2.F</td>
<td>Source for sample MD Nastran database reader.</td>
</tr>
<tr>
<td>makefile</td>
<td>Makefile to build MSC.Access sample programs.</td>
</tr>
<tr>
<td>smplr.F</td>
<td>Source for sample MD Nastran database reader.</td>
</tr>
</tbody>
</table>
A  Glossary of Terms
3060  A User Fatal Message indicating that authorization to run MD Nastran has been denied (see “Using the “mscinfo” Command (UNIX)” on page 31).

6080  A User Warning Message indicating that timing blocks must be generated for your computer (see “Generating a Timing Block for a New Computer” on page 80).

acct  MD accounting file directory, “install_dir/acct” on UNIX and “install_dir\acct” on Windows. Also, the program (install_dir/md20071/arch/acct on UNIX and install_dir\md20071\arch\acct.exe on Windows) that updates the current month’s accounting data file. See MSCACT for the program source.

architecture RC file  The RC file “install_dir/conf/arch/nast2007rc” on UNIX and “install_dir\conf\arch\nast2007.rcf” on Windows. See Table 3-1 for a listing of architecture names.

archive  A test problem library (install_dir/misc/archive on UNIX and install_dir\misc\archive on Windows) that contains test decks that are no longer part of either the DEMO or TPL libraries. These files may be incompatible with MSC.Nastran V70.5 or may use features that are no longer supported.

ASSIGN  A File Management Section (FMS) statement that is used to assign physical files to DBsets or FORTRAN files.

authorize  Command line and RC file keyword that is used to set the authorization code required to run MD Nastran.

basename  The part of a pathname exclusive of the directory and file type (e.g., the basename of /temp/myfile.dat is “myfile”).

buffer pool  A disk cache of GINO blocks.

BUFFPOOL  The NASTRAN statement keyword that sets the size of the buffer pool (see “Using the NASTRAN Statement” on page 136).

BUFFSIZE  One plus the number of words in a GINO physical record. Also, the NASTRAN statement keyword that sets the default buffer size (see “Using the NASTRAN Statement” on page 136).

conf  The MD configuration file directory (install_dir/conf on UNIX and install_dir\conf on Windows) contains the system, architecture, and node RC files and other site-specific files.

counted license  A counted license is a FLEXlm license that limits the number of concurrent executions of MD Nastran. Counted licenses always require a FLEXlm license server.

daemon  A UNIX program that runs in the background and provides services to the operating system and to users. Daemons are generally started when the system is bootstrapped and terminate when the system shuts down.
dat  Default input data file type.

DBALL  Default DBALL DBset file type. The DBALL DBset contains your model and results.

DBset  Database file set.

DDLPRRT  Utility program that prints the contents of the results database (XDB) data definition language database (install_dir/md20071/arch/dbc.xdb on UNIX and install_dir\md20071\arch\dbc.xdb on Windows) and illustrates the batch recovery of the data definition language.

DDLQRY  Utility program that prints the contents of the results database (XDB) data definition language database (install_dir/md20071/arch/dbc.xdb on UNIX and install_dir\md20071\arch\dbc.xdb on Windows) and illustrates the interactive recovery of the data definition language.

del  Delivery database library.

DEMO  The demonstration problem library (install_dir/md20071/nast/demo on UNIX and install_dir\md20071\nast\demo on Windows) contains a selection of MD Nastran input files that are documented in the MSC Nastran Demonstration Problem Manual.

DEMO1  Sample program that prints information from a graphics database file.

DEMO2  Sample program that prints information from a graphics database file.

DMAP  Direct Matrix Abstraction Program, which is the programming language of the MD Nastran solution sequences.

DMP  Distributed Memory Parallel. In MD Nastran, DMP execution is enabled by the “dmparallel” keyword.

doc  Documentation file directory.

EAG FFIO  Engineering Applications Group Flexible File I/O, an asynchronous database I/O library on IRIX64 and linux64 systems. See the ff_io keyword, (p. 304)

ESTIMATE  Utility that estimates memory and disk requirement of a data file and make suggestions on improving the performance of MD Nastran.

F04  The F04 file is created by MD Nastran and contains a module execution summary as well as a database information summary. The F04 file has the file type “.f04”.

F06  The F06 file is created by MD Nastran and contains the numerical results of the analysis. The F06 file has the file type “.f06”.

file locking  A mechanism to prevent multiple MD Nastran jobs from interfering with one another. For example, two jobs attempting to write to the same DBset interfere with one another, whereas two jobs reading the delivery database do not interfere with one another.
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file mapping</td>
<td>A mechanism to use the system’s virtual paging system to access a file. MD Nastran can use file mapping to access GINO files. See Table 4-7 for a listing of systems that support file mapping.</td>
</tr>
<tr>
<td>FMS</td>
<td>File Management Section of the input file, which is used to attach and initialize DBsets and FORTRAN files.</td>
</tr>
<tr>
<td>gentim2</td>
<td>MD Nastran job that determines the timing constants for your computer.</td>
</tr>
<tr>
<td>GINO</td>
<td>The MD Nastran database subsystem.</td>
</tr>
<tr>
<td>GINO block</td>
<td>A block of data transferred by GINO.</td>
</tr>
<tr>
<td>HEATCONV</td>
<td>Utility program that converts pre-MSC.Nastran V68 heat-transfer data files to the MSC.Nastran Version 68 format.</td>
</tr>
<tr>
<td>HPIO</td>
<td>An asynchronous database I/O library on SUPER-UX systems. See the “hpio_param” keyword, (p. 308).</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers, Inc. A professional society. The floating point formats and, to a lesser extent, algorithms used on most MD Nastran computers are defined by IEEE Standard 754.</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>A general MD Nastran input file statement that inserts an external file into the input file. INCLUDE statements may be nested.</td>
</tr>
<tr>
<td>INIT</td>
<td>The INIT statement is part of the File Management Section (FMS) and is used to create a temporary or permanent DBset.</td>
</tr>
<tr>
<td>large file</td>
<td>A file on a 32-bit system that can be 2 gigabytes or larger. All files on a 64-bit system can be large files. See Table 4-7 for a listing of systems that support large files.</td>
</tr>
<tr>
<td>local RC file</td>
<td>The RC file “.nast2007rc” on UNIX and “nast2007.rcf” on Windows in the directory containing the input data file.</td>
</tr>
<tr>
<td>LOG</td>
<td>The LOG file is created by MD Nastran and contains system information as well as system error messages. The LOG file has the file type “.log”.</td>
</tr>
<tr>
<td>MASTER</td>
<td>Default MASTER DBset file type. The MASTER DBset contains the names of other database members and indices.</td>
</tr>
<tr>
<td>MATTST</td>
<td>Sample program that reads the OUTPUT4 matrix files.</td>
</tr>
<tr>
<td>memory</td>
<td>Command line keyword that is used to define the amount of memory allocated for open core.</td>
</tr>
<tr>
<td>MPI</td>
<td>Message Passing Library. An industry-standard library for message passing programs.</td>
</tr>
<tr>
<td>MPL</td>
<td>The module properties list is a table that defines the properties of DMAP modules.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MSC.ACCESS</td>
<td>FORTRAN-callable subroutine library that reads and writes results database (XDB) files.</td>
</tr>
<tr>
<td>MSCACT</td>
<td>Utility program that generates accounting reports. The source for this utility and the</td>
</tr>
<tr>
<td></td>
<td>accounting file update program are maintained in the same file on UNIX and on Windows.</td>
</tr>
<tr>
<td>MSGCMP</td>
<td>Utility program that compiles a text file to create a message catalog.</td>
</tr>
<tr>
<td>NAO</td>
<td>The Network Authorization Option of MD Nastran. The implementation in MSC.Nastran Version</td>
</tr>
<tr>
<td></td>
<td>70.5 is not compatible with earlier versions of NAO.</td>
</tr>
<tr>
<td>ndb</td>
<td>Default neutral-format results database file type.</td>
</tr>
<tr>
<td>neu</td>
<td>Default neutral-format plot file type. Only created by NEUTRL.</td>
</tr>
<tr>
<td>NEUTRL</td>
<td>Utility program that converts binary plot (.plt) files to neutral plot (.neu) files.</td>
</tr>
<tr>
<td>node RC file</td>
<td>The RC file “install_dir/conf/net/nodename/nast2007rc” on UNIX and “install_dir/conf/net/</td>
</tr>
<tr>
<td></td>
<td>nodename/nast2007.rcf” on Windows.</td>
</tr>
<tr>
<td>NUSR</td>
<td>The node-locked license enforcement of the maximum number of users concurrently running MD</td>
</tr>
<tr>
<td></td>
<td>Nastran. See “Enabling Account ID Validation” on page 43 for additional information.</td>
</tr>
<tr>
<td>on2</td>
<td>Default neutral-format OUTPUT2 file type.</td>
</tr>
<tr>
<td>op2</td>
<td>Default binary-format OUTPUT2 file type.</td>
</tr>
<tr>
<td>open core</td>
<td>Amount of working memory in words.</td>
</tr>
<tr>
<td>OPTCONV</td>
<td>Utility program that converts pre-MSC.Nastran V68 optimization and design-sensitivity data</td>
</tr>
<tr>
<td></td>
<td>files to the MSC.Nastran Version 68 format.</td>
</tr>
<tr>
<td>pch</td>
<td>Default punch file type.</td>
</tr>
<tr>
<td>PLOTPS</td>
<td>Utility program that converts binary (.plt) or neutral (.neu) plot files to PostScript (.ps)</td>
</tr>
<tr>
<td>plt</td>
<td>Default binary-format plot file type.</td>
</tr>
<tr>
<td>ps</td>
<td>Default PostScript plot file type.</td>
</tr>
<tr>
<td>RC file</td>
<td>Runtime configuration file that is used by MD Nastran to control execution parameters.</td>
</tr>
<tr>
<td>RCOUT2</td>
<td>Utility program that converts a neutral OUTPUT2 (.np2) file to a binary OUTPUT2 (.op2) file.</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>Utility program that converts neutral results database (.neu) files to binary results</td>
</tr>
<tr>
<td></td>
<td>database (XDB) files.</td>
</tr>
<tr>
<td>RFA</td>
<td>Rigid-format alter library, “install_dir/md20071/nast/rfa” on UNIX and “install_dir/md20071/</td>
</tr>
<tr>
<td></td>
<td>nast/rfa” on Windows. (This directory is now empty.)</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SCR300</td>
<td>Default SCR300 DBset file type.</td>
</tr>
<tr>
<td>SCRATCH</td>
<td>Default SCRATCH DBset file type.</td>
</tr>
<tr>
<td>sdir</td>
<td>Keyword that is used to set the directory for temporary scratch files produced by MD Nastran.</td>
</tr>
<tr>
<td>SMEM</td>
<td>Scratch memory area for memory-resident database files.</td>
</tr>
<tr>
<td>smemory</td>
<td>Command line keyword to set SMEM.</td>
</tr>
<tr>
<td>SMP</td>
<td>Shared Memory Parallel. In MD Nastran, SMP execution is enabled by the “parallel” keyword.</td>
</tr>
<tr>
<td>SMPLR</td>
<td>Sample program that reads graphics database files.</td>
</tr>
<tr>
<td>SSS</td>
<td>Structured Solution Sequences. The delivery database files (SSS.MASTERA, SSS.MCSOU, and SSS.MSCOBJ) are found in “install_dir/md20071/arch” on UNIX and “install_dir/md20071/arch” on Windows; the source files are found in “install_dir/md20071/nast/del” on UNIX and “install_dir/md20071/nast/del” on Windows.</td>
</tr>
<tr>
<td>SSSALTER</td>
<td>Additional alter and error corrections library, “install_dir/md20071/misc/sssalter” on UNIX and “install_dir/md20071/misc/sssalter” on Windows.</td>
</tr>
<tr>
<td>SUN_IO</td>
<td>An asynchronous database read library on Solaris systems. See the “sun_io” keyword, (p. 339).</td>
</tr>
<tr>
<td>SYS</td>
<td>An ASSIGN statement parameter that is used to specify special machine-dependent information. File locking and file mapping of database files are controlled through the SYS parameter.</td>
</tr>
<tr>
<td>sysfield</td>
<td>The global SYS parameter that can be specified on the command line or in an RC file.</td>
</tr>
<tr>
<td>SYSTEM(x)</td>
<td>System cells that are used by MD Nastran to control analysis parameters.</td>
</tr>
<tr>
<td>TAPBTST</td>
<td>Sample program that reads binary-format OUTPUT2 files.</td>
</tr>
<tr>
<td>TPL</td>
<td>The test problem library (TPL, install_dir/md20071/nast/tpl on UNIX and install_dir/md20071/nast/tpl on Windows) contains a general selection of MD Nastran input files showing examples of most of the MD Nastran capabilities, in general, these files are not documented.</td>
</tr>
<tr>
<td>TRANS</td>
<td>Utility program that converts binary results database (.XDB) files to neutral results database (.neu) files.</td>
</tr>
<tr>
<td>type</td>
<td>The part of the pathname exclusive of the directory and basename (e.g., the file type of myfile.dat is “.dat”).</td>
</tr>
<tr>
<td>UFM</td>
<td>A User Fatal Message that describes an error severe enough to terminate the program.</td>
</tr>
</tbody>
</table>
UFM 3060
A User Fatal Message indicating that authorization to run MD Nastran has been denied (see “Using the mscinfo Command (UNIX)” on page 31).

UIM
A User Information Message that provides general information.

uncounted license
An uncounted license is a FLEXlm license that allows any number of concurrent executions of MD Nastran on a given node. An uncounted license does not require a FLEXlm license server.

user RC file
The RC file “$HOME/nast2007rc” on UNIX and “%HOMEDRIVE%HOMEPATH%/nast2007.rcf” on Windows.

util
Utility program library, “install_dir/md20071/util” on UNIX and “install_dir/md20071/util” on Windows.

UWM
A User Warning Message that warns of atypical situations. You must determine whether a problem exists in the analysis.

UWM 6080
A User Warning Message indicating that timing blocks must be generated for your computer (see “Generating a Timing Block for a New Computer” on page 80).

version
A file is “versioned” by appending a dot followed by a version number to the file’s name. The latest version of a file does not have a version number, all earlier versions do, with the oldest having the smallest version number and the latest having the highest version number.

XDB
The XDB file is created by MD Nastran and contains results information for use by various post-processing programs. See the “POST” parameter in “Parameters” on page 645 of the MD Nastran Quick Reference Guide for further information on generating XDB files. XDB files are not versioned. The XDB file has the file type “.xdb”.

CHAPTER A
Glossary of Terms
Keywords and Environment Variables
**Keywords**

The following is a complete list of the keywords that may be used on the command line or placed into RC files as appropriate.

Keywords that use yes/no values accept partial specification and case-independent values. For example, “yes” may be specified as “y”, “ye”, or “yes” using uppercase or lowercase letters.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acct</td>
<td>acct=yes,no Default: No Indicates solution accounting is to be performed. The new “lock” keyword may be used to ensure that all jobs have solution accounting enabled. For example, the following RC file lines force all jobs to use accounting: Example: acct=yes lock=yes The first line turns accounting on. The second line ensures accounting is on for every job; see the “lock” keyword for more details.</td>
</tr>
<tr>
<td>acdata</td>
<td>acdata=string Default: None Specifies site defined accounting data. See your system administrator to determine if and how this keyword is to be used. See “Enabling Account ID and Accounting Data” on page 43 for additional information.</td>
</tr>
<tr>
<td>acid</td>
<td>acid=string Default: None Specifies the site defined account ID for this job. See your system administrator to determine if and how this keyword is to be used. See “Enabling Account ID and Accounting Data” on page 43 for additional information.</td>
</tr>
<tr>
<td>acvalid</td>
<td>acvalid=string Default: None Note: This keyword can only be set in the command initialization file, see the sections titled “Enabling Account ID and Accounting Data” on page 43 and “Specifying Parameters” on page 55. Indicates account ID validation is to be performed. If “acvalid” is not defined, or is null, then no checks are made of the account ID. If “acvalid” is defined, then account ID validation is performed. “Enabling Account ID and Accounting Data” on page 43 contains more information on defining this keyword.</td>
</tr>
</tbody>
</table>
adapter_use

Specifies how the node's adapter is used in the IBM Parallel Environment for AIX. The legal values are “dedicated” and “shared”.

The default is “adapter_use=dedicated” if “euilib=us”, otherwise it is “adapter_use=shared”.

This keyword may also be set with the MPAdapterManager environment variable. The environment variable overrides the RC files; the command line overrides the environment variable.

after

Holds the job's execution until the time specified by time. See the description of the “at” command in your system documentation for the format of time.

Example:

md20071 nastran example
after=10:00

The job is held until 10:00 AM.

append

Combines the F04, F06, and LOG files into a single file after the run completes. If “no” is specified, the files are not combined. If “yes” is specified, the files are combined into one file with the type “.out”.

Example:

md20071 nastran example
append=yes

The F04, F06, and LOG files are combined into a file named “example.out”.

application

Specifies the application to be run.

Note: This keyword should always be set to “NASTRAN”, and may only be specified on the command line or in the command initialization file. See “Specifying Parameters” on page 55.

attdel

Controls automatic assignment of the delivery database. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.

autoasgn

Controls automatic assigning of DBsets. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.
**authinfo**

<table>
<thead>
<tr>
<th>authinfo=number</th>
<th>Default: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the amount of information written to the LOG during authorization processing. Values greater than zero indicate additional information is to be written.</td>
<td></td>
</tr>
</tbody>
</table>

**authorize**

<table>
<thead>
<tr>
<th>authorize=spec</th>
<th>Default: UNIX: install_dir/conf/authorize.dat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows: install_dir/conf/authoriz.dat</td>
<td></td>
</tr>
<tr>
<td>Selects the licensing method for MD Nastran. The spec can take on several forms. They include:</td>
<td></td>
</tr>
</tbody>
</table>

- **authorize=FLEXlm-license-spec**
  FLEXlm licensing has been selected. Please see “Automatically Starting a FLEXlm Server” on page 37 for information on specifying a FLEXlm license.

- **authorize=pathname**
  This specifies either a FLEXlm license file, see the above reference, or a node-lock authorization code, see “Using Node-locked Authorization Codes” on page 41 for information on specifying a node-locked authorization code. If only a directory is specified, the program assumes that either “authorize.dat” or “license.dat” is in the specified directory.

**Example:**

```
md20071 nastran example
auth=myauthfile
```

The job runs using the node-locked authorization code in “myauthfile”.

**authqueue**

<table>
<thead>
<tr>
<th>authqueue=number</th>
<th>Default: 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>All systems except NEC: Specifies the time in minutes to wait for a seat to become available. If the seat becomes available before this specified time period expires, the job will be allowed to continue. If not, the job will be terminated.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:**

When a job is waiting for a seat to become available, it consumes computer resources such as memory, swap file space, disk space, etc. Too many jobs waiting for licenses could have a severe impact on the system.

**Example:**

```
md20071 nastran example
auth=myauthfile
```

The job runs using the node-locked authorization code in “myauthfile”. If a seat is not available within 20 minutes of the start of the job, the job terminates.
Example:  
md20071 nastran example  
  \[ \text{auth=myauthfile} \]  
  \[ \text{authqueue=10} \]  

The job is run using the node-locked authorization code in “myauthfile”. If a 
seat is not available within 10 minutes of the start of the job, the job will be 
terminated.

**batch**  
**batch=**\[yes, no\]  
Default: Yes  

(UNIX) Indicates how the job is to be run. If “yes” is specified, the job is run as a 
background process. If “no” is specified, the job is run in the foreground. If the 
“aft” or “queue” keywords are specified, the batch keyword is ignored. Jobs 
submitted with “batch=yes” will run under nice(1).

**Note:** If the job is already running in an NQS or NQE batch job, the default is “no”.

Example: 
md20071 nastran example  
\[ \text{batch=no} \]  

The job is run in the foreground.

**bfgs**  
**bfgs=**\[number\]  
Default: 0  

Selects strategies of BFGS updates for the arc-length methods in non-linear 
analysis. See the “\text{nastran Command and NASTRAN Statement}” in Chapter 1 
of the \text{MD Nastran Quick Reference Guide} for more information on this 
keyword.

**bpool**  
**bpool=**\[value\]  
Default: \[27 (SUPER-UX); 37 (all others)\]  

Specifies the number of GINO and/or executive blocks that are placed in buffer 
pool.

Example:  
md20071 nastran example  
\[ \text{bpool=100} \]  

Space for 100 GINO buffers is reserved for the buffer pool.

**buffpool**  
**buffpool=**\[number\]  
Default: \[27 (SUPER-UX); 37 (all others)\]  

Specifies the number of GINO and/or executive blocks that are placed in the 
buffer pool. This keyword is a synonym for the “bpool” keyword. See the 
description of the “bpool” keyword for more information.
**buffsize**

`buffsize=value`  
Default: 8193

Specifies the physical record size, in words (1 word = 8 bytes when mode = i8; 4 bytes on all others), of all MD Nastran DBsets except those specified with INIT statements and MSCOBJ. The physical I/O size is BUFFSIZE-1 words.

If “buffsize=estimate” is specified, ESTIMATE will be used to determine `value`.

See “Estimating BUFFSIZE” on page 108 for recommended BUFFSIZE values based on model size.

BUFFSIZE must reflect the maximum BUFFSIZE of all DBsets attached to the job including the delivery database, which is generated with a BUFFSIZE of 8193. If you generate your own delivery database, this default may be different. The maximum value of BUFFSIZE is 65537 words. BUFFSIZE must be one plus a multiple of the disk block size. The disk default block size may be determined with the “system” special function described in “Using the Help Facility and Other Special Functions” on page 101; specific block size information may be obtained from your system administrator.

Example:
```
md20071 nastran example
buffsize=16385
```

The BUFFSIZE is set to 16385 words.

**bypass_off**  
(bypass_off=yes,no  
Default: Yes)

(IRIX64) Disables the HIPPI bypass, forcing all host-to-host MPI messages to be sent through TCP. You should only set “bypass_off=no” if you want to use the HIPPI interface for communications among multiple computers.

See SGI’s mpi(1) man page or MPI documentation for additional information.

**ccstempdir**  
(ccstempdir=directory  
Default:)

(Windows) Specifies a network-visible working (scratch) directory for use by the Windows CCS Job Scheduler. If this keyword is not specified, a search will be made for a suitable directory by looking at: the directory specified by the “sdirectory” keyword, the directory portion of the location specified by the “out” keyword, the current directory, the directory specified by the “TEMP” environment variable or the directory specified by the “TMP” environment variable. A directory is network-visible if it is specified in UNC format or it if can be converted to a UNC name. Processing is terminated if none of these locations specified a network-visible directory and if a suitable directory was not specified using this “ccstempdir” keyword.
### config

`config=number`  
Default: Computer dependent

Specifies the configuration (CONFIG) number used by MD Nastran to select timing constants. You can change this value to select the timing constants of a different computer model. A configuration number of zero is considered undefined by the `nastran` command. See “Defining a Computer Model Name and CONFIG Number” on page 79 and “Generating a Timing Block for a New Computer” on page 80 for additional information.

### constitle

`constitle=yes, no`  
Default: Yes

(Windows) Specifies whether or not the console title bar is to be modified to have the job identification. This keyword is only applicable to Windows systems.

### cputime

`cputime=cputime`  
Default: None

(UNIX) The following capability is dependent upon the queue submission commands defined by the “submit” keyword and your queuing system. The capability or examples may not work on your system.

Note: Specifies the maximum amount of CPU time that the complete job is permitted to use when the “queue” keyword is used. This time includes the execution of the driver program, the MD Nastran executable, plus any commands specified by the “pre” and “post” keywords. See your system’s queuing documentation for the format of `cputime`.

The value can be specified as either “hours:minutes:seconds”, “minutes:seconds”, or “seconds”; it will always be converted to seconds by the `nastran` command.

Example:  
```
md20071 nastran example \
queue=small cputime=60
```

This example defines the maximum CPU time for the complete job as 60 seconds.

Example:  
```
md20071 nastran example \
queue=small cpu=1:15:0
md20071 nastran example \
queue=small cpu=75:0
md20071 nastran example \
queue=small cpu=4500
```

These examples all define the maximum CPU time for the complete job as one hour and fifteen minutes.
**cpu_use**

`cpu_use=keyword`  
Default: See text

(AIX)

Specifies how the node’s CPU is used in the IBM Parallel Environment for AIX. The legal values are “unique” and “multiple”.

The default is “cpu_use=unique” if “eulib=us”, otherwise it is “cpu_use=multiple”.

This keyword may also be set with the `MP_CPU_USE` environment variable. The environment variable overrides the RC files; the command line overrides the environment variable.

**cpyinput**

cpyinput=0,1  
Default: 0

Indicates the input data file is to be copied to a temporary file before processing. Setting cpyinput=1 will emulate the old MD Nastran behavior of copying the file, this will consume additional time and disk resources.

**dballco**

dballco=value  
Default: 1

Allows you to scale DBALL estimates. This scale factor is applied before the "dballmin" value, that provides a lower bound for DBALL estimates.

Example:

```
md20071 estimate example
dballco=2
```

This will double the DBALL disk estimate and then apply the "dballmin" lower bound.

Example:

```
md20071 estimate example
dballco=0.5
```

This will halve the DBALL disk estimate. An estimate less than than the lower bound specified by "dballmin" will be set to the lower bound.

**dballmin**

dballmin=value  
Default: 1mb

Allows you to define the lower bound for all DBALL estimates. This bound is applied after the "dballco" value, that multiplies the actual estimate by a "conservatism" factor.

Example:

```
md20071 estimate example
dballmin=2mb
```

This will set the minimum DBALL disk estimate to 2 MB.

**dbs**

dbs=pathname  
Default: .

Creates database files (see Table 4-7) using an alternate file prefix. If “dbs” is not specified, database files are created in the current directory using the basename of the input data file as the prefix. If the “dbs” value is a directory, database files are created in the specified directory using the basename of the input data file as the filename.

**Note:**

If “dbs” is specified and “scratch=yes” is specified, a warning will be issued and “scratch=no” assumed.
In the following examples, assume the current directory includes sub-directories “mydir” and “other”, and that an “example.dat” exists in both the current directory and “other”. That is, ./example.dat, ./mydir, ./other, and ./other/example.dat exist on UNIX; and \example.dat, \mydir, \other, and \other\example.dat exist on Windows.

Example:       md20071 nastran example
Database files are created in the current directory with the name “example”, e.g., ./example.DBALL on UNIX; and \example.DBALL on Windows.

Example:       md20071 nastran other/example
Database files are created in the “other” directory with the name “example”, e.g., ../other/example.DBALL on UNIX and \other\example.DBALL on Windows.

Example:       md20071 nastran example
dbs=myfile
Database files are created in the current directory with the name “myfile”, e.g., ./myfile.DBALL on UNIX and \myfile.DBALL on Windows.

Example:       md20071 nastran example
dbs=mydir
Database files are created in the mydir directory with the name “example”, e.g., ./mydir/example.DBALL on UNIX and \mydir\example.DBALL on Windows.

Example:       md20071 nastran example
dbs=mydir/myfile
Database files are created in the mydir directory with the name “myfile”, e.g., ./mydir/myfile.DBALL on UNIX and \mydir\myfile.DBALL on Windows.

Example:       md20071 nastran example
dmp=4 host=a:b:c:d
dbs=/aa:/bb:/cc:/dd
This example will set the “dbs” directory to “/aa” on host a, “/bb” on host b, “/cc” on host c, and finally “/dd” on host d.

**Note:** The use of distinct per-task database directories can have a significant impact on elapsed time performance of DMP jobs on SMP and NUMA systems.

**dbverchk**

| dbverchk | dverchk=0, 1 | Default: 0 (check is performed) |

Specifies whether or not database version checking is to be skipped. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the *MD Nastran Quick Reference Guide* for more information on this keyword.
delete

delete=yes, no, all, jid, list

Default: No

Note:
This keyword is only intended to be used when MD Nastran is running in server mode or is embedded within another application. The deletion occurs before the post commands are run.

Unconditionally delete files after an MD Nastran job completes. Specifying "delete=yes" will delete the F04, F06 and LOG files when the job completes; "delete=all" will delete the F04, F06, LOG, NDB, OP2, PCH, PLT and XDB files when the job completes. You can also specify a list of file types, e.g., "delete=f04,log,plt" will only delete the F04, LOG and PLT files. Note that, on UNIX systems, this list of file types is case-sensitive. That is, "delete=master" will delete files with an extension of "master" but not files with an extension of "MASTER" and "delete=MASTER" will delete files with an extension of "MASTER" but not files with an extension of "master".

Example: md20071 nastran example
deadle=op2,plt

After the MD Nastran job has completed, the "example.op2" and "example.plt" files will be unconditionally deleted. These files are normally kept if they are not empty.

Example: md20071 nastran example
deadle=plt MASTER DBALL

After the MD Nastran job has completed, the "example.plt", "example.MASTER" and "example.DBALL" files will be unconditionally deleted. Normally, the "example.plt" file will be kept if it is not empty and the "example.MASTER" and "example.DBALL" files are kept unless "scratch=yes" was specified.

delivery
delivery=pathname

Default: MSCDEF

Specifies an alternate delivery database option. See “Creating and Attaching Alternate Delivery Databases” on page 208 for further information on alternate delivery databases.

The special pathname “MSCDEF” indicates the standard MD Nastran delivery database.

Example: md20071 nastran example
delivery=mysss

The job runs using a solution sequence from the delivery database “mysss.MASTERA”.

diag
diag=flag,flag,...

Default: None

Sets MD Nastran diagnostics. This keyword may also be set with the DIAG Executive Control Statement. See “DIAG” on page 110 in the MD Nastran Quick Reference Guide for information on the default value and legal values for this keyword. The diagnostics set using this keyword are in addition to any diagnostics set with the DIAG statement in the input file.
Example:  

```
md20071 nastran example
diag=5
```

The MD Nastran job is run with DIAG 5 set.

**diaga**

`diaga=number`  
**Default:** None  

Set MD Nastran diagnostic flags 1-32. The value specified over-rides any previous "diag=n" values where n is in the range 1 to 32. These diagnostics are set in addition to any diagnostics set via the Executive Control "DIAG" statement in the input data file. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the **MD Nastran Quick Reference Guide** for more information on this keyword.

**diagb**

`diagb=number`  
**Default:** None  

Set MD Nastran diagnostic flags 33-64. The value specified over-rides any previous "diag=n" values where n is in the range 33 to 64. These diagnostics are set in addition to any diagnostics set via the Executive Control "DIAG" statement in the input data file. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the **MD Nastran Quick Reference Guide** for more information on this keyword.

**disksave**

`disksave=number`  
**Default:** 0 (no save)  

Specifies Lanczos High Performance Option controlling whether or not the matrix/vector multiply is saved in a scratch file. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the **MD Nastran Quick Reference Guide** for more information on this keyword.

**display**

`display=display_name`  
**Default:** Current display  

(UNIX) Specifies a display for XMONAST. This keyword may also be set with the DISPLAY environment variable. The environment variable overrides the RC files; the command line overrides the environment variable.

**distort**

`distort=number`  
**Default:** 0 (terminate run)  

Specifies element distortion fatal termination override. Applies to all p-elements and the TETRA h-elements. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the **MD Nastran Quick Reference Guide** for more information on this keyword.

**dmparallel**

`dmparallel=number`  
**Default:** 0  

(See Table 5-3) Specifies the number of tasks for a Distributed Memory Parallel (DMP) analysis. This value may only be set on the command line.

The value must be null or zero to cancel DMP processing, or a number greater than zero to enable DMP processing.

See “Running Distributed Memory Parallel (DMP) Jobs” on page 163 for additional information.
Example:

md20071 nastran example
dmp=4

The job is run with four DMP tasks.

dskco
dskco=value  Default: 1

Allows you to define a factor to scale total disk estimates. This scale factor is applied before the "dskmin" value, that provides a lower bound for total disk estimates.

Example:

md20071 estimate example
dskco=2

This doubles the total disk estimate and then applies the "dskmin" lower bound.

Example:

md20071 estimate example
dskco=0.5

This will halve the total disk estimate. An estimate less than the lower bound specified by "dskmin" will be set to the lower bound.

dskmin
dskmin=value  Default: 1mb

Allows you to define the lower bound for all total disk estimates. This bound is applied after the "dskco" value, that multiplies the actual estimate by a "conservatism" factor.

Example:

md20071 estimate example
dskmin=2mb

This will set the minimum total disk estimate to 2 MB.

euidevice
euidevice=device-name  Default: css0

(AIX) Specifies the communications adapter to use in the IBM Parallel Environment for AIX. This keyword is used when "euilib=ip" has been specified. The specified device must exist as a character special device in /dev.

The default is “euidevice=css0”.

This keyword may also be set with the MP_EUIDEVICE environment variable. The environment variable overrides the RC files; the command line overrides the environment variable.
euilib

Specifies the CSS library implementation to use in the IBM Parallel Environment for AIX. Setting "euilib=us" will select the User Space (US) CSS; "euilib=ip" will select the Internet Protocol (IP) CSS.

The default is "euidevice=css0".

This keyword may also be set with the MP_EUILIB environment variable. The environment variable overrides the RC files; the command line overrides the environment variable.

executable

Specifies the name of an alternate solver executable. This keyword overrides all architecture and processor selection logic. If a directory is not specified by pathname and the file does not exist in the current directory, the default architecture directory is assumed.

Example: md20071 nastran example
exe=analysis.um

The job runs using the executable “analysis.um”. Since a directory was not specified, this file must exist in either the current directory or install_dir/md20071/arch on UNIX or install_dir/md20071/arch on Windows.

expjid

Specifies whether or not the input file is to be "expanded" or not, that is, whether or not the input file is to be read and all "include" files processed. If "expjid=no" is specified, the input file will be used directly.

If "expjid=yes" is specified, the input file will be expanded and stored in the location specified by "out", with an extension of "exp" added.

If "expjid=pathname" is specified, the input file will be expanded and stored in the location specified by pathname. If pathname specifies a directory, the expanded file will be stored using the base name of the input file, with an extension of "exp" added. If pathname specifies a file name without an extension, and extension of "exp" will be added.

If "expjid=auto" is specified (or taken by default):

- If "node" is specified, the input file will be expanded only if it is not visible from the remote node.
- If "node" is not specified, the input file will not be expanded, i.e., processing will be as if "expjid=no" was specified.

If the input file is expanded:
• If "node" is specified, the expanded file will be copied (if necessary) to the remote node for processing and will be deleted from both the remote and local nodes at the completion of processing.

• If "node" is not specified, processing will terminate without actually invoking the MD Nastran analysis program and without storing any other files.

**f04**

f04 = number  
Default: 4  
Specifies FORTRAN unit number for Execution Summary Table. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.

**f06**

f06 = number  
Default: 6  
Specifies FORTRAN unit number for standard output file. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.

**fastio**

fastio = number  
Default: 0 (UNPACK/PACK)  
Specifies Lanczos High Performance Option controlling input/output in orthogonalization/normalization routines. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.

**fbsmem**

fbsmem = number  
Default: See the MD Nastran Quick Reference Guide.  
Reserves memory for faster solution of the Lanczos method of eigenvalue extraction. This keyword may also be set with the “sys 146” command line keyword. See the MD Nastran Quick Reference Guide for information on the default value and legal values for this keyword.

**fbsopt**

fbsopt = number  
Default: See the MD Nastran Quick Reference Guide  
Selects the forward-backward substitution methods. This keyword may also be set with the “sys 70” command line keyword. See the MD Nastran Quick Reference Guide for information on the default value and legal values for this keyword.

**ff_io**

ff_io = yes, no, append  
Default: Yes  
(SGI/Altix - Linux/IA64 IRIX64)  
*Note:* Because of the difficulty in setting the FF_IO_OPTS value, especially the striping partitions, you are strongly urged to remove any FF_IO_OPTS settings you may have been using.
Indicates EAG FFIO is to be enabled. EAG FFIO can provide a substantial elapsed-time performance increase.

If “ff_io=yes” is set and “ff_io_opts” is not set, a default value for the FF_IO_OPTS environment variable will be determined. This value will: include both the default permanent and scratch DBsets; use the cache size specified by the “ff_io_cachesize” keyword; consider the device geometries of the disks containing the “dbs” and “sdirectory” directories.

If “ff_io=append” is set, the calculated FF_IO_OPTS value will be appended to the user’s FF_IO_OPTS value.

If “ff_io=no” is specified, any values for FF_IO_OPTS and FF_IO_DEFAULTS will be suppressed, and EAG FFIO will be disabled.

Additional documentation supplied by Cray Research, Inc. on EAG FFIO can be found in the HTML files “install_dir/md20071/irix64/ffio.html”, “install_dir/md20071/linux64/ffio.html”.

The default parameters are share=1 and stride=1. The values for max_lead, alloc, and set (i.e., cbiks and cbits) are based on disk device geometry; page_size and num_pages are based on BUFFSIZE and “ff_io_cachesize”.

The minimum cache size is 512 000 words.

The size is specified as a memory size, see “Specifying Memory Sizes” on page 104.

Example:
```
md20071 nastran example \\
  ff_io=yes \\
  ff_io_cachesize=2mw
```

The job is run with a 2 MW EAG FFIO cache.
**ff_io_defaults**

ff_io_defaults=string

Default: None

(SGI/Altix - Linux/IA64 IRIX64)

Specifies the EAG FFIO default options to be used. This value must be a valid FFIO specification string; no error checking is performed before MD Nastran starts.

This keyword may also be set by the FF_IO_DEFAULTS environment variable. The environment variable overrides the RC files, and the command line overrides the environment variable.

**ff_io_opts**

ff_io_opts=string

Default: See “ff_io”

(SGI/Altix - Linux/IA64 IRIX64)

Specifies the EAG FFIO options to be used. This value must be a valid EAG FFIO specification string; no error checking is performed before MD Nastran starts.

This keyword may also be set by the FF_IO_OPTS environment variable. The environment variable overrides the RC files, and the command line overrides the environment variable.

**frqseq**

frseq=number

Default: 0 (equal segments)

Specifies Lanczos High Performance Option controlling exponent for rational function for segment boundaries. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.

**gmconn**

gmconn=pathname

Default: None

Specifies the name of the external evaluator connection file. External geometric and bar or beam element evaluators may be specified. See the MSC Nastran Version 69 Release Guide for additional information on external bar or beam elements. Also, see “Using BEAMSERV” on page 262 for information on running an MD Nastran job using a beam server.

Example:

```
md20071 nastran example
gmconn=mybeamserver
```

The job is run with the external evaluators specified in “mybeamserver”.

---

**Note:** Because of the difficulty in setting the FF_IO_OPTS value, especially the striping partitions, you are strongly urged to remove any FF_IO_OPTS settings you may have been using.
Keywords and Environment Variables

hicore

```
hicore=memory_size
```

Default: Dependent on "memory" and other keywords

Specifies maximum working memory. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.

hostovercommit

```
hostovercommit=yes,no
```

Default: No

Allows this job to assign more tasks to a host than processors. This does **not** prevent other MD Nastran jobs or users from using the processors. See also the “hosts” keyword below.

If “hostovercommit=no” is specified, at most one task will be assigned for each processor on the host, i.e., a four processor system can only have four tasks assigned.

If “hostovercommit=yes” is specified, tasks are assigned to hosts in a round-robin order until all tasks are assigned, without regard to the number of processors on the host.

**Note:** Assigning more tasks to a host than it has processors will impact the elapsed-time performance of your DMP job.

In the following examples, assume that host1 and host2 each have two processors.

**Example:**
```
md20071 nastran example
dmp=6 
hosts=host1:host2
hostovercommit=no
```

The job will not be started because a total of only four processors are available on host1 and host.

**Example:**
```
md20071 nastran example
dmp=6 
hosts=host1:host2
hostovercommit=yes
```

The job will be allowed to start, with three tasks each assigned to host1 and host2.

hosts

```
hosts=host:host:...
```

Default: See text

hosts=host:host:...

hosts=filename
Defines the list of candidate hosts to be used for a DMP analysis. This list is scanned in a round-robin order until all tasks have been assigned to a host. If “hostovercommit=no” is specified, at most one task will be assigned for each processor on the host, i.e., a four processor system can only have four tasks assigned.

Multiple hosts are specified in the standard manner for the PATH environment variable, that is “hosts=host1:host2:...” on UNIX and “hosts=host1:host2:...” on Windows.

On AIX, the default is “/host.list”, a file containing the names of the hosts; on other systems, the default is the current system.

See “Running Distributed Memory Parallel (DMP) Jobs” on page 163 for additional information.

In the following examples, assume that the current host, host1, and host2 each have two processors.

Example:

```
md20071 nastran example
dmp=2
```

If this command is executed on an AIX system, the ./host.list file will be used to determine the list of hosts; on all other systems, the job will be run on the current host.

Example:

```
md20071 nastran example
dmp=3 \
hosts=host1:host2
```

The first and third tasks will be assigned to host1, the second task will be assigned to host2.

Example:

```
md20071 nastran example
dmp=3 \\nhosts=myhostfile
```

The file ./myhostfile on UNIX and .\myhostfile on Windows will be read to determine the list of hosts to use.

**hpio_param**

(SUPER-UX, Linux IA64 (NEC Azusa))

```
hpio_param=string
```

Default: None

Specifies the HPIO control string. The control string is composed of one or more filename-options pairs of the form:

```
```
where:

- **file_template**: Blank separated list of filename templates, there is no default. Examples are “*DBALL*” to match all files ending in “DBALL” and “*DBALL SCR*” to match all files ending in “DBALL” and all files with “SCR” anywhere in the name.

- **p1**: Number of cache pages for each file; the default is 5.

- **p2**: The size of each cache page; the default is “8m” or 8 MB. This value is the number of bytes, or a number followed by “k” for KB, or “m” for MB. If no unit is supplied, the default is “k” for KB.

- **p3**: The maximum number of cache pages to read ahead. The default and minimum is 1. This value must be less than p1. A rule of thumb is $p_3 < 0.5 \times p_1$.

- **p4**: “nolog” or “log”, the default is “nolog”. Note, the “log” option is intended for tuning and debugging purposes only, it can generate large amounts of output.

- **p5**: XMU cache working directory, there is no default. The file system containing this directory must be an SFS/H file system.

- **p6**: Number of XMU cache pages, the default is 5. This value must be greater than 0 if p5 is specified.

- **p7**: Number of buffers for XMU cache access, the default is 2.

**Note:** If invalid XMU fields are specified, a message will be printed in the LOG file and will continue without using HPIO.
The additional main memory consumed by the HPIO facility is:

\[(p1 + 1) \times p2 \times n_{files} \text{ bytes}\]

without the XMU, or

\[(p1 + 1 + p7 + 2) \times p2 \times n_{files} \text{ bytes}\]

with the XMU, where \(n_{files}\) is the number of files matched by file_template.

The space on the XMU consumed by the HPIO facility is

\[p6 \times p2 \text{ bytes}\]

Example:

```
md20071 nastran example \ hpio_param='*SCR* 
(9::1)'
```

The job is run with HPIO enabled for all files with SCR in their name, e.g., sdir/example.SCRATCH and sdir/example.SCR300. HPIO will allocate ten cache pages \((p1 = 9)\) of 8 MB per page per file, the cache will read ahead one page \((p3 = 1)\). Assuming no other files use HPIO, i.e., no other filenames used by the job match the template "*SCR*", an additional 160 MB of memory will be required by this job.

**Note:**

XMU cache is only supported on the SX-4 platform. It is not applicable to MSC.Nastran 2001 or later releases.

**hyperthreads**

(Windows)

Species whether or not Intel® HyperThreads logical processors are to be used or not. This keyword is ignored unless HyperThreading is enabled (in the BIOS) and is supported by the operating system. Currently, only WindowsXP Professional and later systems support the full capabilities of HyperThreading, although Windows 2000 Server will utilize HyperThreading logical processors. Normally, "hyperthreads=no" should only be used on systems where multiple MD Nastran jobs are run concurrently or where "parallel=2" is specified for a single MD Nastran job. For those systems, it may be useful to specify the "hyperthreads=no" keyword in the install_dir/conf\nast2007.rcf file.
### ishellext

**Default:** See text.

Defines command processor associations for ISHELL executables. Each value is specified as "`file-type=processor`" where `processor` is the executable used by MD Nastran to execute an ISHELL program with the specified `file-type`. See “Running an ISHELL Program” on page 200 for information on using an ISHELL program and the default list of processors.

Specify two consecutive quotes, e.g., `ishellext=ksh=''` to specify a null processor, that is, to directly execute the ISHELL program.

**Note:** You will need protect the quotes from the shell if specified on the command line.

Specify a null `file-type` to define a processor for files without a file type.

Specify `"="`` to specify a null `file-type` and a null `processor`.

Specifying a `file-type` already defined in the table will replace the previous entry; specifying a `file-type` not yet defined in the table will append the new entry to the end of the table, that is, it will be processed last.

**Note:** On Windows, all executable files must have a non-null `file-type`. This is why "TPLDIR:QAISHELL" executable cannot be used on Windows, but "TPLDIR:qaishell.pl" can.

On Windows NT, it may be necessary to define "CMD.EXE" as the processor for certain `.EXE` files, e.g., 16-bit compiled Basic program. This can be done with "ishellext=exe=cmd"

Up to twenty associations can be defined.

This keyword may also be set with the `MSC_ISHELLEXT` environment variable. The environment variable overrides the RC files; the command line overrides the environment variable.

**Example:**

```plaintext
md20071 nastran example \ 
  ishellext=tcl=wish,sh=ksh
```

This example will add one association and replace another. If the ISHELL program name exists with the file type `.tcl`, the wish executable will be used; if the ISHELL program name exists with the file type `.sh`, the ksh executable will be used. Since neither processor specification included a pathname component, the system PATH will be searched for the executables.

---

### ishellpath

**Default:** See text.

**Default:** See text.

ishellpath=`value:value:..`

Example:

```plaintext
md20071 nastran example \ 
  ishellext=tcl=wish,sh=ksh
```

This example will add one association and replace another. If the ISHELL program name exists with the file type `.tcl`, the wish executable will be used; if the ISHELL program name exists with the file type `.sh`, the ksh executable will be used. Since neither processor specification included a pathname component, the system PATH will be searched for the executables.
Defines a list of directories to search for the ISHELL program if a suitable ISHELL program doesn’t exist in the current working directory. If this list is exhausted before finding a suitable ISHELL program, the standard PATH is searched. Multiple paths are specified in the standard manner, that is “ishellpath=/dir1:/dir2:...” on UNIX and “ishellpath=\dir1\dir2;...” on Windows.

If you have not set a value for “ishellpath”, the value will be set to the directory containing the input data file, this automatically handles the common case where the ISHELL program is located in the same directory as the input data file referencing it.

This keyword may also be set with the MSC_ISHELLPATH environment variable. The environment variable overrides the RC files; the command line overrides the environment variable.

Example:  
```
md20071 nastran
TPLDIR:qaishell
```

Assuming no RC file set “ishellpath” and the environment variable MSC_ISHELLPATH was not defined, the “ishellpath” value will be set to the directory referenced by “TPLDIR:”. MD Nastran will attempt to locate the ISHELL program in the current working directory, the TPL directory, or in the PATH.

Example:  
```
md20071 nastran example
ishellpath=bin
```

This example assumes either the current working directory or the bin subdirectory contains the ISHELL program.

**iter**  
iter=yes, no  
Default: No (do not execute iterative solver)

Controls execution of iterative solver. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.

**ja**  
ja=yes, no  
Default: No

(SUPER-UX) Enable job accounting using the ja(1) utility. See the ja(1) man page for additional information on this utility.

Example:  
```
md20071 nastran example
ja=yes
```

The job is run with the job accounting system enabled.

**jid**  
jid=pathname  
Default: None

Specify the name of the input data file. An input file must be defined on the command line. Any command line argument that does not have a keyword is assumed to be the input file; only the last filename is used.
Keywords and Environment Variables

CHAPTER B

Example:  
```
md20071 nastran this that example
```

The input file “example.dat” is used; the tokens “this” and “that” are ignored.

**Note:**

If the input file is specified as “example” and the files “example.dat” and “example” both exist, the file “example.dat” will be chosen. In fact, it is **impossible** to use a file named “example” as the input data file if a file named “example.dat” exists.

**jidpath**

<table>
<thead>
<tr>
<th>jidpath</th>
<th>jidpath=path-spec</th>
<th>Default: None</th>
</tr>
</thead>
</table>

Specify a list of directories to search if the input data file or any INCLUDE file does not specify a pathname component and does not exist in the current directory.

This keyword may also be set by the MSC_JIDPATH environment variable. The environment variable overrides the RC files, and the command line overrides the environment variable.

**UNIX example:**
```
md20071 nastran example
jidpath=$HOME
```

**Windows Example:**
```
md20071 nastran example \ 
jidpath=%HOMEDRIVE%\HOMEPA TH%
```

These will find the file “example.dat” or “example” if it is located in either the current working directory or your home directory.

Multiple directories are specified using the standard syntax for the PATH environment variable. For example:

**UNIX example:**
```
md20071 nastran example \ 
jidpath=/models/a:/models/b
```

**Windows Example:**
```
md20071 nastran example \ 
jidpath=\models\a;\models\b
```

Your specification of this value in RC files can include environment variable references. On UNIX, use the standard shell “$name” or “${name}” syntax; on Windows use the standard “%name%” syntax.

**jitdtype**

<table>
<thead>
<tr>
<th>jidtype</th>
<th>jidtype=file-type</th>
<th>Default: dat</th>
</tr>
</thead>
</table>

Specify an alternate default file-type of the input data file and any INCLUDE files.

**Example:**
```
md20071 nastran example 
jidtype=bdf
```
This example will set the default file type to “bdf”, i.e., the nastran command will look first for a file named “example.bdf”, and if that is not found for the file “example”; if neither file is found, an error will be reported.

If you have not defined a value for the “jidtype” keyword, the nastran command will set the keyword to the actual file type of the input data file.

Example:
```
md20071 nastran example.bdf
```

The nastran command looks for “example.bdf.dat”, if that file does not exist, it then looks for “example.bdf”. Assuming that file exists, and no other value for “jidtype” has been defined, the nastran command sets “jidtype=bdf”.

**ldqrkd**

*ldqrk=number*

Default: 0 (Version 68+ method)

Selects the differential method for CQUAD4 and CTRIA3 elements. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the *MD Nastran Quick Reference Guide* for more information on this keyword.

**locbulk**

*locbulk=number*

Default: 0 (RESTART FMS statement)

Specifies that special Bulk Data processing is in effect. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the *MD Nastran Quick Reference Guide* for more information on this keyword.

**lock**

*lock=keyword*

Default: None

The “lock” keyword can be used by a site or a user to prevent modification of a keyword’s value.

For example, the following RC file lines will force all jobs to use accounting by setting the “acct” keyword on and then preventing the keyword from being changed later in an RC file, or on the command line:

Example:
```
acct=yes
lock=acct
```

Once these lines are read, any attempt to set the “acct” keyword later in the same RC file, in an RC file read after this file, or on the command line will be silently ignored. See “RC File Keywords” on page 68 for information on RC file and command line processing.

The “lock” keyword may appear anywhere a keyword is accepted. The lock keyword itself can be locked with “lock=lock”.

Example:
```
authorize=license-spec
lock=authorize
```

Once these lines are read, any attempt to set the “authorize” keyword later in the same RC file, in an RC file read after this file, in the environment via “MSC_LICENSE_FILE” or “LM_LICENSE_FILE”, or on the command line will be silently ignored.
<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsymbol</td>
<td>lsymbol=name=string</td>
<td>None  This keyword has the same general function and syntax as the &quot;symbol&quot; keyword except that it defines a &quot;local&quot; symbolic (or logical) name. Symbols defined using this keyword will not be passed to remote hosts, i.e., to hosts specified by the &quot;node&quot; keyword. When the &quot;node&quot; keyword is not specified, this keyword is synonymous with the &quot;symbol&quot; keyword.</td>
</tr>
<tr>
<td>massbuf</td>
<td>massbuf=number</td>
<td>See the MD Nastran Quick Reference Guide. Sets half the number of buffers to set aside for storing the mass matrix in memory. This keyword may also be set with the &quot;sys199&quot; command line keyword. See the MD Nastran Quick Reference Guide for information on the default value and legal values for this keyword.</td>
</tr>
<tr>
<td>maxnode</td>
<td>maxnode=number</td>
<td>Value of dmparallel parameter (AIX) Specifies the maximum number of hosts to be used when a pool request is being used. It is required if you want more than one DMP task to be assigned to a single host in pool. The default pool processing will only assign one DMP task to each host.</td>
</tr>
<tr>
<td>maxlines</td>
<td>maxlines=number</td>
<td>999999999 Specifies the maximum number of output lines. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.</td>
</tr>
<tr>
<td>memmin</td>
<td>memmin=value</td>
<td>16mb Allows you to define the lower bound for all memory estimates. This bound is applied after the “memco” value, that multiplies the actual estimate by a “conservatism” factor. Example: md20071 estimate example memmin=8mb This will set the minimum memory estimate to 8 MB.</td>
</tr>
<tr>
<td>memory</td>
<td>memory=size</td>
<td>estimate This keyword may also be set with the MP_NODES environment variable. The environment variable overrides the RC files; the command line overrides the environmental variable.</td>
</tr>
</tbody>
</table>

**Note:** See “Determining Resource Requirements” on page 107 for information on estimating a job’s memory requirements.
Specifies the amount of open core memory to allocate. If “memory=estimate” is specified, ESTIMATE will be used to determine size. Otherwise, the size is specified as a memory size, see “Specifying Memory Sizes” on page 104.

If a value was not assigned to the “memory” keyword, or if “memory=estimate” was specified and ESTIMATE failed to provide an estimate, the nastran command will use the value specified by the “memorydefault” keyword. If the “memorydefault” value is null, the nastran command will issue a fatal error and the job will end.

Example:  
```
md20071 nastran example
memory=25mw
```

The job is run using an open core memory size of 25 MW, or 25600 KW, or 26214400 words.

Example:  
```
md20071 nastran example
memory=0.5xPhysical
```

If run on Windows, the job is run using an open core memory size of half the computer’s physical memory. If run on UNIX and the computer’s physical memory was not defined using the “s.pmem” keyword, the job will fail.

memorydefault

memorydefault=size  
Default: 8mw

Specifies the default memory size if a null value was defined for the “memory” keyword, or “memory=estimate” was defined and the ESTIMATE utility failed to provide an estimate.

Note:  
If a null value is defined for “memorydefault” and it is used as described above, the job will not start.

memorymax

memorymax=size  
Default: UNIX: 0.8*physical  
Windows: 1.2*physical

Specifies the maximum memory size that may be requested. Any request in excess of this will be limited to the “memorymaximum” value. See “Specifying Memory Sizes” on page 104 for MD Nastran’s maximum memory limits.

Note:  
If size includes a reference to “physical” or “virtual”, and the value is not known, the “memorymaximum” value will be silently ignored.

In the following examples, assume “memorymaximum=1gb” was set in an RC file.

Example:  
```
md20071 nastran example
memory=900mb
```

The job is run using an open core memory size of 900MB.

Example:  
```
md20071 nastran example
memory=1200mb
```

The job is run using an open core memory size of 1GB, i.e., the “memorymaximum” value set in the RC file.
**mergeresults**

*mergeresults=yes,no* 
Default: Yes

Specifies the results from each DMP task are to be merged into the standard files from the master host.

Setting “mergeresults=yes” will cause the output from all tasks to appear in the output files for the master task. That is, as if the analysis were run with one task.

Setting “mergeresults=no” will cause the output from each tasks to appear task-specific output files. That is, each file will need to be examined to get all results.

**Note:**

If “mergeresults=no” is specified in a static run the results of the individual domains will not be sent back to the master and the system solution will not be obtained.

The keyword “mergeresults” has no affect on a solution 103 or 111 run.

The only circumstances where “mergeresults=no” is recommended is where xdb files are requested and intended to be attached using MSC.Patran in solution 108.

In solution 108, if “mergeresults=no” is specified and “slaveout=yes” is not specified, then the results of the slave processors will be lost.

In solution 108, it is possible to get a through-put advantage by saving communication between the master and slaves when “mergeresults=no” and “slaveout=yes” is specified.

**metime**

*metime=number* 
Default: -1

Minimum time for execution summary table message. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the *MD Nastran Quick Reference Guide* for more information on this keyword.

**mindef**

*mindef=number* 
Default: 1 (do not check)

Indefinite Mass Matrix Check flag. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the *MD Nastran Quick Reference Guide* for more information on this keyword.

**minfront**

*minfront=number* 
Default: Machine dependent

Set the rank minimum front size in the sparse modules. See the “nastran Command and NASTRAN Statement” on page 1 of the *MD Nastran Quick Reference Guide* for more information on this keyword. This value may also be set with the “rank” keyword.
mio_cachesize

Specifies the size of mio cache to be used.

Default: 0

(AIX)

mode

Changes the default INTEGER mode for a platform from the default to the specified value, where “ilp64” is equivalent to “i8”, “lp64” and “ilp32” are equivalent to “i4” and “base” or “no” specify that the default is to be used. If alternate INTEGER mode is not supported on a particular platform and if this keyword is specified, “User Information Message” is used and this keyword is ignored. This keyword may only be specified in the initialization file or on the command line.

Example: mode=i8

Specifies that “i8” INTEGER mode is to be used for this job.

rmode

Specifies the remote node INTEGER mode. This keyword value is passed to a remote node as its “mode” keyword value. This keyword is ignored unless “node” is specified.

Example: rmode=i8

Specifies the “mode=i8” is to be passed to the remote node specified using the “node” keyword.

mperturb

Set the perturbation factor for indefinite mass matrix. See the nastran Command and NASTRAN Statement on page 1 of the MD Nastran Quick Reference Guide for more information on this keyword.

Default: 1 (do not perturb)

mpyad

Selects/deselects multiplication method selection. This keyword may also be set with the “sys66” command line keyword. See the MD Nastran Quick Reference Guide for information on the default value and legal values for this keyword.

Default: See the MD Nastran Quick Reference Guide.

msgbell

Specifies whether or not the job completion string will include an audible message (“bell” sound) or not. “Yes” or “bell” says that three “bell” sounds will be appended to the job completion string. “No” suppresses the bell sounds.

Default: Yes
### msgcat

**Default:**
- UNIX: `install_dir/md20071/arch/analysis/msg`
- Windows: `install_dir/md20071/arch/analysis/msg`

The “msgcat” keyword specifies an alternate message catalog containing the message text used for many MD Nastran messages. A site or user can modify the message file to include message text that is more appropriate to their operations, compile the new catalog using the MSGCMP utility, and invoke the new catalog using this keyword.

**Example:**
```
md20071 nastran example
msgcat=mycat.msg
```

This example will use the file “mycat.msg” as the message catalog. See the sections titled “Customizing the Message Catalog” on page 78 and “MSGCMP” on page 234 for additional information.

**Note:**
Message catalogs are computer-dependent, “Binary File Compatibility”, identifies the systems that are binary compatible; binary compatible systems can use the same message file.

### nastran

**Default:** None

Specifies a value for the NASTRAN statement.

**Note:**
This keyword can only be specified in an RC file. If the last character of the keyword value is a comma, or a quote or parenthetic expression is open, the next line in the RC file is considered a continuation. The statement will continue until the quote or parenthetic expression is closed and a line that is not ended by a comma is found.

### ncmd

**Default:** `print msg | write user tty`

Specifies an alternate job completion notification command (see the “notify” keyword). If this keyword is being set on the command line, and `command` contains embedded spaces, enclose `command` in quotes.

If the specified command contains the two-character sequence `{}`, the sequence is replaced by the text “MD Nastran job name completed”.

**Note:**
The following example may not work on your system. The “mail(1)” utility on HP-UX does not accept the “-s” option.

**Example:**
```
md20071 nastran example
notify=yes \ncmd="print {} | \mail -s {} $(whoami)"
```
At the end of the job, mail is sent to the user submitting the job. The braces in the “ncmd” value are replaced by the job completion text, and the modified command is run:

```
print "MSC/NASTRAN job example completed" \ 
mail -s "MSC/NASTRAN job example completed" user
```

Windows example: 
```
md20071 nstran example "ncmd=echo
done"
```

The word “done” will be printed in the command window when the job completes.

**newhess**

Default: See the *MD Nastran Quick Reference Guide*

Requests the complex eigenvalue method. This keyword may also be set with the “sys108” command line keyword. See “EIGC” on page 1437 of the *MD Nastran Quick Reference Guide*, and the *MSC Nastran Numerical Methods User’s Guide* for information on the default value and legal values for this keyword.

**news**

Default: Yes

Displays the news file (`install_dir/md20071/nast/news.txt` on UNIX and `install_dir/md20071/nast/news.txt` on Windows) in the F06 file. If “auto” is specified, the news file is only displayed if it has been modified since the last time it was displayed for you. If “yes” is specified, the news file is displayed in the F06 file regardless of when it was last changed. If “no” is specified, the news file is not displayed in the F06 file.

Example: 
```
md20071 nastran example
news=yes
```

The news file is displayed in the F06 file after the title page block.

**Note:**

The news file can also be displayed on the terminal by using the command:
```
md20071 nastran news
```
Keywords and Environment Variables

nice  nice=yes,no  Default: no (Windows)
                 yes (UNIX, batch=yes)
                 no (UNIX, batch=no)

Species whether or not the priority of the MD Nastran analysis process should be reduced or not.

For Windows systems, "nice=yes" means that the priority of the analysis process will be changed to one level below standard command priority.

For UNIX systems, the default behavior, as determined by the initial settings for the "-fg" and "-bg" special queue names described in "Customizing Queue Commands (UNIX)" on page 83 is as follows: "nice=yes" means that the analysis process will be run by the "nice" command regardless of the setting of the "batch" keyword, "nice=no" means that the analysis process will not have its priority reduced, regardless of the setting of the "batch" keyword. This behavior may be modified if the default definitions of the "-bg" and "-fg" queue names are changed.

nlines  nlines=number  Default: 50

Specifies number of lines printed per page of output. See the "nastran Command and NASTRAN Statement" in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.

node  node=nodename  Default: None

Executes the job on the specified node. This node may be either a UNIX node or Windows NT/2000/XP node. See "Running a Job on a Remote System" on page 154 for additional information. This keyword may only be specified on the command line.

Use the "username" keyword to specify an alternate user name on the remote node.

Example:

md20071 nastran example
node=othernode

The job is run on the computer named "othernode".

If the remote node is a UNIX node, rsh/rcp processing must be enabled.

If the remote node is a Windows node running Windows NT/2000/XP, the MSCRmtMgr program must be running on that node, either as a started service or as a program running in a command prompt window.

Windows nodes running Windows 95/98/ME may not be used as remote nodes.

notify  notify=yes,no  Default: Yes

Sends notification when the job is completed. See the "ncmd" keyword to define an alternate notification command.
Note: If the job is queued using the “queue” keyword, or the job is already running in an NQS batch job, the default is “notify=no”.

Example: md20071 nastran example notify=yes

nsegadd

\text{nsegadd=number}  \text{Default: 2}

Number of segments in the element error table in adaptive analysis. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.

numseg

\text{numseg=number}  \text{Default: See text.}

Sets the number of segments for the Lanczos High Performance Option. See “EIGRL” on page 1450 of the MD Nastran Quick Reference Guide for information on the default value and legal values for this keyword.

Note: In a DMP job, the default is the number of tasks specified by the “dmparallel” keyword.

old

\text{old=yes,no}  \text{Default: Yes}

Saves previous copies of the F04, F06, LOG, OP2, OUT, PCH, and PLT output files using sequence numbers (additional user-specified file types can be versioned with the “oldtypes” keyword). Sequence numbers are appended to the keyword filename and are separated by a period.

If “yes” is specified, the highest sequence number of each of the output files is determined. The highest sequence number found is incremented by one to become the new sequence number. Then, all current output files that do not include sequence numbers are renamed using the new sequence number as a type.

Example: md20071 nastran example old=yes

For example, assume your current working directory contains the following files:

v2401.datv2401.f04.1v2401.f04v2401.f06v2401.logv2401.log.1v2401.f04v2401.f04.2v2401.f04.1v2401.log.1v2401.log.3
Apparently, the user ran the job four times, but deleted some of the files, e.g., v2401.f04.3, v2401.f06.2, and v2401.f06.3. When the job is run again with “old=yes”, the files are renamed as follows: v2401.f04 is renamed to v2401.f04.4, v2401.f06 is renamed to v2401.f06.4, and v2401.log is renamed to v2401.log.4. The sequence number 4 is used because it is one greater than the highest sequence number of all of the selected files (the highest being v2401.log.3).

**oldtypes**

oldtypes=list

Default: None

Specifies additional file types that will be subject to versioning and deletion via the “old” keyword. The items in the list may be separated by either spaces or commas; they should not include the leading “.”. You may specify file types that do not exist.

Example: md20071 nastran example
oldtypes=xdb,mytype

The files “example.xdb” and “example.mytype” will be subject to versioning or deletion as specified by the “old” keyword.

This keyword may also be set by the MSC.OLDTYPES environment variable. The environment variable overrides the RC files, and the command line overrides the environment variable.

**out**

out=pathname

Default: .

Saves the output files using a different file prefix or in a different directory. If “out” is not specified, the output files are saved in the current directory using the basename of the input data file as a prefix. If the “out” value is a directory, output files are created in the specified directory using the basename of the input data file as the filename.

In the following examples, assume the current directory includes sub-directories “mydir” and “other”, and that an “example.dat” exists in both the current directory and “other”. That is, ./example.dat, ./mydir, ./other, and ./other/example.dat exist on UNIX; and \example.dat, \mydir, \other, and \other\example.dat exist on Windows.

Example: md20071 nastran example
or: md20071 nastran other/example

Output files are created in the current directory with the name “example”, e.g., ./example.f06 on UNIX and \example.f06 on Windows.

Example: md20071 nastran example
out=myfile

Output files are created in the current directory with the name “myfile”, e.g., ./myfile.f06 on UNIX and \myfile.f06 on Windows.
Example: md20071 nastran example
out=mydir

Output files are created in the mydir directory with the name “example”, e.g.,
./mydir/example.f06 on UNIX and \mydir\example.f06 on Windows.

Example: md20071 nastran example
out=mydir/myfile

Output files are created in the mydir directory with the name “myfile”, e.g.,
./mydir/myfile.f06 on UNIX and \mydir\myfile.f06 on Windows.

parallel
(parallel=value) Default: 0

SMP processing reduces elapsed time at the expense of increased CPU time. The default is 0, which specifies no SMP processing. If "parallel=1", the parallel algorithms are used on one processor.

Note: If you need to vary the number of SMP CPUs during a job, you must set either the “parallel” keyword or SYSTEM(107) on a NASTRAN statement to the maximum number of SMP CPUs that will be requested. Some systems cannot process a DMAP request for CPUs in excess of this initial value.

Example: md20071 nastran example
parallel=2

The job is run in SMP mode on a maximum of two CPUs.

pause
pause=keyword Default: no

Pause the nastran command before exiting to wait for the “Enter” or “Return” key to be pressed. This can be useful when the nastran command is embedded within another program. The values are “fatal”, “information”, “warning”, “yes”, and “no”. Setting “pause=yes” will unconditionally wait; “pause=fatal”, “pause=warning”, and “pause=information” will only wait if a fatal, warning, or information message has been issued by the nastran command. The default is “pause=no”, i.e., do not wait when the nastran command ends.
post

post=command_string

Default: None

Runs the specified command after the job has completed and after the F06, F04, and LOG files have been concatenated if “append=yes” is specified. For UNIX, the command must be a valid Korn shell command. The command may pipe the output of one command into another. If the specified command contains embedded spaces, enclose the entire command_string in quotes. Each occurrence of the “post” keyword will be concatenated together to form a sequence of commands. Specify a null value, i.e., “post=” to erase all of the previously entered commands. Typical uses of this keyword are to run postprocessing programs or to compress the output files to save space.

UNIX example:

md20071 nastran example
post='gzip example*' 

At the end of the job, the command “gzip example*” is run to compress all files beginning with “example”.

The value of the “out” keyword is available for use by the “post” keyword. The example “post” keyword could also have been written as post='gzip SMSC_OUT.*'. If app=yes was specified, post='gzip SMSC_OUT.out' would only compress the output file.

Windows example:

md20071 nastran example
post="print example.*"

At the end of the job, all files named “example.*” will be printed. The output of the post command(s) will be displayed on the command shell window.

See the “Environment Variables” on page 352 for a list of environment variables that may be used in the post command.

Note:

In order to allow the “post” keyword to operate on the output files, the standard output from the post commands is not written to the output files.

ppcdelta

ppcdelta=time

Default: None

Note:

The following capability is dependent upon the queue submission commands defined by the “submit” keyword and your queuing system. The capability or examples may not work on your system.

Specifies the amount of time to subtract from the specified CPU time to determine the per-process CPU time limit. This subtraction will ensure that MD Nastran does not consume all of the time allocated to the job.

The value can be specified as either “hours:minutes:seconds”, “minutes:seconds”, or “seconds”, and will always be converted to the number of seconds.

Example:

md20071 nastran example \ queue=small cpu=1000 ppcdelta=5
The job is submitted to the small queue with a total CPU time limit of 1000 seconds; the MD NASTRAN job will be limited to 995 seconds.

**ppmdelta**

(UNIX)

Default: 105% of executable size

Note: The following capability is dependent upon the queue submission commands defined by the “submit” keyword and your queuing system. The capability or examples may not work on your system.

Specifies the amount of memory to add to the “memory” value to determine “ppm”, the per-process memory value. The per-process limit is the total amount of memory that each process may acquire. This includes the executable, open core memory (via the “memory” keyword), disk file buffers, and etc. (IRIX64 systems also include EAG FFIO cache).

The *size* is specified as a memory size, see “Specifying Memory Sizes” on page 104.

If *size* is less than 1000, then “ppmdelta” equals *size* divided by 100 and multiplied by the size of the executable, i.e., 105 specifies the default 105% of executable size.

If *size* is greater than 1000, but less than the size of the executable, then “ppmdelta” equals *size* plus the executable size.

If *size* exceeds the size of the executable, then “ppmdelta” equals *size*.

Example:

```
md20071 nastran example \
queue=small mem=100m 
ppmdelta=10m
```

The job is submitted to the small queue with a open core size of 100 MW, and a per-process memory limit of 110 MW.

**pre**

Default: None

Runs the specified command before the job begins. For UNIX, the command must be a valid Korn shell command. The command may pipe the output from one command to another. If the specified command contains embedded spaces, enclose the entire *command* in quotes. Each occurrence of the “pre” keyword will be concatenated together to form a sequence of commands. Specify a null value, i.e., “pre=” to erase all of the previously entered commands.

UNIX example:

```
md20071 nastran example \ 
pre="print Job beginning |\ 
mail $(whoami)"
```

Sends mail to the submitting user immediately before beginning the job.

Windows example:

```
md20071 nastran example 
pre="dir example.*"
```
At the end of the job, a directory listing of all files named “example.*” will be displayed in the LOG file.

See “Environment Variables” on page 352, for a list of environment variables that may be used in a “pre” command.

**prmdelta** (UNIX)

| prmdelta=size | Default: 5120 |

**Note:**
The following capability is dependent upon the queue submission commands defined by the “submit” keyword and your queuing system. The capability or examples may not work on your system.

Specifies the amount of memory to add to the specified “ppm” value to determine “prm”, the per-request or per-job memory value. The per-job limit is the total amount of memory that all processes in the job may acquire. This includes the MD Nastran process plus any other concurrent or parent processes. The minimum value is 5120.

The *size* is specified as a memory size, see “Specifying Memory Sizes” on page 104.

Example:
```
md20071 nastran example \
queue=small prmdelta=10k
```

The per-job memory limit is 10 KW larger than the per-process memory limit.

**processor**

| processor=file_type | Default: Computer dependent |

Specifies the file type of the solver executable. On some computers, MD Nastran provides more than one executable. The baseline executable has the filename “analysis” on UNIX and “analysis.exe” on Windows. Other, advanced-architecture executables are named “analysis.file_type” on UNIX and “analysis.file_type.exe” on Windows, e.g., “analysis.power2” on AIX or “analysis.ultra” on Solaris systems. The nastran command will select the correct executable based on the current computer. In some cases, it may be desirable to use one of the other executables. For example, to run the baseline executable on an advanced system, specify “proc=”. To run an advanced-architecture on a new computer not correctly identified by the nastran command, specify “proc=file_type”.

**Note:**
This keyword overrides the processor selection logic. Specification of an incompatible executable may cause errors or incorrect operations.

**punch**

| punch=number | Default: 7 |

Specifies FORTRAN unit number for PUNCH file. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the *MD Nastran Quick Reference Guide* for more information on this keyword.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>q4skew</td>
<td>Minimum allowable value of skew for the CQUAD4 element. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword. Default: 30.0</td>
</tr>
<tr>
<td>q4taper</td>
<td>Maximum allowable value of taper for the CQUAD4 element. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword. Default: 30.0</td>
</tr>
<tr>
<td>qclass</td>
<td>Defines an optional queue class that can be used in the definition “submit” keyword. It is also used to define the class used when submitting DMP jobs to the AIX LoadLeveler. Default: None (UNIX)</td>
</tr>
<tr>
<td>qoption</td>
<td>Defines the options to add to the queue submittal command. See the “submit” keyword. Default: None (UNIX)</td>
</tr>
<tr>
<td>quadint</td>
<td>Specifies quadratic or linear interpolation for line search method in nonlinear analysis. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword. Default: 0 (quadratic)</td>
</tr>
<tr>
<td>queue</td>
<td>The following capability is dependent upon the queue submission commands defined by the “submit” keyword and your queuing system. The capability or examples may not work on your system. Default: None (UNIX)</td>
</tr>
</tbody>
</table>

**Example:**

```
md20071 nastran example \
queue=small qoption=-mu
```

The job is run with the additional job submission parameter “-mu” if the keyword reference %qopt% was included in the queue’s command definition.
Specifies the name of the queue to use for job submittal. This keyword requires the submit keyword to define the available queues and queue submittal commands. See the “submit” keyword.

Example:  
```
md20071 nastran example
queue=small
```

This example submits the job to the small queue.

**radlst**

```radlst``` = ```number```  
Default: 0

Print radiation area summary. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the *MD Nastran Quick Reference Guide* for more information on this keyword.

**radmtx**

```radmtx``` = ```number```  
Default: 0

Type of radiation exchange coefficients. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the *MD Nastran Quick Reference Guide* for more information on this keyword.

**rank**

```rank``` = ```number```  
Default: See “System Descriptions” on page 363

Sets both SYSTEM(198) and SYSTEM(205) to the specified value. SYSTEM(198) and SYSTEM(205) set the minimum front size and number of rows that are simultaneously updated, respectively, in sparse symmetric decomposition and FBS. The sparse solver will build a front, a `k k` sub matrix, until `k` is at least as large as SYSTEM(198). Once a sufficiently large front has been built, it is updated `m` rows at a time, where `m` is the value of SYSTEM(205).

For best performance, `SYSTEM(205) ≥ SYSTEM(198)`. The optimal values for these system cells is problem and processor dependent; the default values for these system cells are set to processor-dependent values.

The actual value used for SYSTEM(205) may be found in the F04 file in the text of USER INFORMATION MESSAGE 4157 as the RANK OF UPDATE value. See Table C-18 for the default values of these system cells.

**rdbs**

```rdbs``` = ```pathname_prefix```  
Default: .

Remote Node alternate user database prefix. Overrides “scratch=yes” and “dbs=”. If the prefix is a directory, ’jid-basename’ is appended. The default on the remote node is “dbs=./’jid-basename’”. This keyword is ignored unless “node=” is specified.
**rdelivery**

```
rdelivery=pathname, MSCDEF  Default: MSCDEF
```
Remote Node alternate delivery database prefix or "MSCDEF". This keyword overrides all MSC-supplied solution sequences. See “Creating and Attaching Alternate Delivery Databases” on page 208 for further information on alternate delivery databases. If a directory is not specified, the default delivery database directory is assumed. The default is “rdelivery=MSCDEF. This keyword is ignored unless "node=" is specified.

**real**

```
real=size  Default: See text.
```
Specifies the amount of open core memory that certain numerical modules will be restricted to. This keyword may be used to reduce paging, at the potential expense of spilling. The keyword may also be set with the "sys81" keyword. See the *MD Nastran Quick Reference Guide* for further information.

The *size* is specified as a memory size, see “Specifying Memory Sizes” on page 104.

On UNIX systems, the default is “0”. On Windows systems, the default is calculated using "realdelta".

**realdelta**

```
realdelta=size  Default: 12MB
```
(Windows) Specifies the difference between physical memory and the “real” parameter if neither “real” nor “sys81” were set.

The *size* is specified as a memory size, see “Specifying Memory Sizes” on page 104.

If *size* is greater than 1000, the value is subtracted from the physical memory size.

If *size* is less than 1000, it is assumed to be a percentage of the physical memory size.

Example:

```
md20071 nastran example
realdelta=50
```

The “real” value will be set to 50% of the physical memory if no value has been assigned to “real” or SYSTEM(81).
### Keywords and Environment Variables

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resd</td>
<td>Use the Resource Manager to allocate nodes.</td>
</tr>
<tr>
<td></td>
<td>This keyword may also be set by the MP_RESD environment variable. The</td>
</tr>
<tr>
<td></td>
<td>environment variable overrides the RC files, and the command line overrides</td>
</tr>
<tr>
<td></td>
<td>the environment variable.</td>
</tr>
<tr>
<td>rexecutable</td>
<td>Remote Node alternate solver executable. This keyword overrides all</td>
</tr>
<tr>
<td></td>
<td>architecture and processor selection logic. If a directory is not specified,</td>
</tr>
<tr>
<td></td>
<td>the default executable directory is assumed. This keyword is ignored unless</td>
</tr>
<tr>
<td></td>
<td>“node=” is specified.</td>
</tr>
<tr>
<td>rcf</td>
<td>Specifies the name of the local RC file. If this keyword is not specified,</td>
</tr>
<tr>
<td></td>
<td>the .nast2007rc file on UNIX and nast2007.rcf on Windows located in the</td>
</tr>
<tr>
<td></td>
<td>input data file’s directory is used.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>md20071 nastran example</td>
</tr>
<tr>
<td></td>
<td>rcf=nast.rcf</td>
</tr>
<tr>
<td></td>
<td>The nastran command will process ./nast.rcf on UNIX, or \nast.rcf on Windows in lieu of the default local RC file ./nast2007rc on UNIX and</td>
</tr>
<tr>
<td></td>
<td>\nast2007.rcf on Windows.</td>
</tr>
<tr>
<td>rcmd</td>
<td>Specifies the path of the nastran command on the remote system when remote</td>
</tr>
<tr>
<td></td>
<td>processing has been requested via the “node” keyword. If this value is not</td>
</tr>
<tr>
<td></td>
<td>set, the nastran command will first try its own absolute path on the remote</td>
</tr>
<tr>
<td></td>
<td>system, if this fails, the path will be removed, i.e., the default PATH of</td>
</tr>
<tr>
<td></td>
<td>the remote system will be used.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>md20071 nastran example \</td>
</tr>
<tr>
<td></td>
<td>rcmd=/msc/bin/nast2007</td>
</tr>
<tr>
<td></td>
<td>The pathname of the nastran command on the remote system is explicitly</td>
</tr>
<tr>
<td></td>
<td>defined as /msc/bin/nast2007. If this file does not exist, or is otherwise</td>
</tr>
<tr>
<td></td>
<td>not executable, the job will fail.</td>
</tr>
<tr>
<td>rgmconn</td>
<td>Remote Node Geometric evaluator connection file. See the description of the</td>
</tr>
<tr>
<td></td>
<td>“gmconn” keyword for more detailed information. This keyword is ignored</td>
</tr>
<tr>
<td></td>
<td>unless “node=” is specified.</td>
</tr>
<tr>
<td>rmpool</td>
<td>Specifies the pool ID to be used when LoadLeveler Version 2.1 or greater</td>
</tr>
<tr>
<td></td>
<td>queue submittal is being used to run a DMP job.</td>
</tr>
</tbody>
</table>
This keyword may also be set with the MP_RMPOOL environment variable. The environment variable overrides the RC files; the command line overrides
the environment variable.

**rmsgcat**

```
rmsgcat=pathname
```

Default: See msgcat= keyword

Remote Node binary message catalog path name. If a directory is not
specified, the default executable directory is assumed. This keyword is ignored
unless "node=" is specified.

**rdebug**

```
rdebug=yes, no, number
```

Default: Yes

This keyword controls what, if any, debug settings ("-d" options) are
propagated to a remote node. "Yes" will send all current debug flags (except
for "SHELL" and "RSHELL"), "no" will not pass any current debug flags.
Specifying a number will set the remote debug flags to that value, where a
value of "0" is equivalent to "no". This keyword is ignored unless "node=" is
specified.

**rostype**

```
rostype=windows, nt, 1, unix, linux, 2
```

Default: None

Specifies the remote node operating system type. "Windows", "NT" and "1"
are equivalent. "Unix", "Linux" and "2" are equivalent. If this keyword is not
specified, the nastran command will attempt to determine the remote node
operating system type dynamically. This type code is used to determine the
format of the remote commands used, for example, to test for file existence or
to delete temporary files on the remote node. Also, if "rrmtuse" is not
specified, this keyword will determine what communications programs are
used, where "Windows" is equivalent to "rrmtuse=mscrmtcmd" and "Unix" is
equivalent to "rrmtuse=rsh". This keyword is ignored unless "node=" is
specified.

**rrmtuse**

```
rrmtuse=mscrmtcmd, 1, rsh, 2
```

Default: None

Specifies which communications programs are to be used to access the remote
node. "Mscrmtcmd" and "1" are equivalent. "Rsh" and "2" are equivalent. If
"rsh" is specified, the remote node will be assumed to be a UNIX system. If
this keyword is not specified and if the "rostype" keyword is specified,
"mscrmtcmd" will be assumed if the "rostype" value is "windows" and "rsh"
will be assumed if the "rostype" value is "unix". This keyword is ignored
unless "node=" is specified.

**rmdir**

```
rmdir=pathname
```

Default: See sdirectory= keyword

Remote Node directory for scratch files. This is the default directory for user
database files if "scratch=yes". If this keyword is not specified, the
"sdirectory" value is used. Please see the description of the "sdirectory"
keyword for the default value. This keyword is ignored unless "node=" is
specified.
**rttimeout**

*rttimeout = number*  
Default: 60

Specifies the timeout value, in seconds, to be used by "MSCRmtCmd" (or the program defined by the "s.rmtcmd" keyword) in accessing a remote node. This keyword is ignored unless "node" is specified.

**s.rmtcmd**

*s.rmtcmd = pathname*  
Default: MSCRmtCmd

Specifies the full pathname to the MSC Remote command used to communicate with Windows or UNIX/Linux systems. This keyword may only be specified in the Initialization file or on the command line. This keyword is ignored unless "node" is specified.

**scr300**

*scr300 = number*  
Default: 2 (create)

Requests creation of SCR300 partition on SCRATCH DBset. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.

**scr300co**

*scr300co = value*  
Default: 1

Allows you to define a factor to scale SCR300 estimates. This scale factor is applied before the "scr300min" value, that provides a lower bound for SCR300 estimates.

Example:

```
md20071 estimate example
scr300co=2
```

This will double the SCR300 disk estimate and then apply the "scr300min" lower bound.

Example:

```
md20071 estimate example
scr300co=0.5
```

This will halve the SCR300 disk estimate. An estimate less than the lower bound specified by "scr300co" will be set to the lower bound.

**scr300del**

*scr300del = number*  
Default: 100

Sets minimum number of blocks of SCR300 partition of SCRATCH DB set at which it is deleted. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.

**scr300min**

*scr300min = value*  
Default: 1mb

Allows you to define the lower bound for all SCR300 estimates. This bound is applied after the "scr300co" value, that multiplies the actual estimate by a "conservatism" factor.

Example:

```
md20071 estimate example
scr300min=2mb
```

This will set the minimum SCR300 disk estimate to 2 MB.
<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
<th>Default</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>scratch</td>
<td>scratch=yes,no,mini</td>
<td>No</td>
<td>Deletes the database files at the end of the run. If the database files are not required, “scratch=yes” can be used to remove them preventing cluttering of the directory with unwanted files. If “mini” is specified, a reduced size database that can only be used for data recovery restarts will be created. See Chapter 12 of the <em>MSC Nastran Reference Manual</em> for further details on the “mini” database.</td>
<td>md20071 nastran example scratch=yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>All database files created by the run are deleted at the end of the job in the same way as the FMS statement INIT MASTER(S).</td>
<td></td>
</tr>
<tr>
<td>scratchco</td>
<td>scratchco=value</td>
<td>1</td>
<td>Allows the user to define a factor to scale SCRA TCH estimates. This scale factor is applied before the “scratchmin” value, that provides a lower bound for SCRA TCH estimates.</td>
<td>md20071 estimate example scratchco=2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This will double the SCRA TCH disk estimate and then apply the “scratchmin” lower bound.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This will halve the SCRA TCH disk estimate. An estimate less than the lower bound specified by “scratchmin” will be set to the lower bound.</td>
<td></td>
</tr>
<tr>
<td>scratchmin</td>
<td>scratchmin=value</td>
<td>1mb</td>
<td>Allows you to define the lower bound for all SCRA TCH estimates. This bound is applied after the “scratchco” value, that multiples the actual estimate by a “conservatism” factor.</td>
<td>md20071 estimate example scratchmin=2mb</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This will set the minimum SCRA TCH disk estimate to 2 MB.</td>
<td></td>
</tr>
<tr>
<td>scrsave</td>
<td>scrsave=number</td>
<td>0</td>
<td>Lanczos High Performance Option that controls reuse of scratch files in segment logic. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the <em>MD Nastran Quick Reference Guide</em> for more information on this keyword.</td>
<td></td>
</tr>
</tbody>
</table>
sdball

**sdball=size**

Default: Computer dependent

Specifies an alternate default size for the DBALL DBset. The computer-dependent default is listed in "Computer Dependent Defaults" on page 374. This default is overridden by an INIT FMS statement. If the value "sdball=estimate" is specified, ESTIMATE will be used to determine a suitable default.

The size is specified as the number of blocks (BUFFSIZE words long) or the number of words or bytes followed by one of the modifiers: "T", "TW", "TB", "G", "GW", "GB", "M", "MW", "MB", "K", "KW", "KB", "W", "B". See "Specifying Memory Sizes" on page 104 for a description of these modifiers.

**Note:** No attempt is made to verify if the DBALL DBset can ever grow to the size specified by this keyword.

Example:

```
md20071 nastran example
sdball=1024gb
```

Defines the default size of the DBALL DBset as 1 TB.

Example:

```
md20071 nastran example
sdball=.5tb
```

Defines the default size of the DBALL DBset as .5TB or 512GB.

**sdirectory**

**sdirectory=directory**

Default: See text.

Specifies the directory to use for temporary scratch files created during the run. MD Nastran can create very large scratch files, the scratch directory should contain sufficient space to store any scratch files created during a run. You must have read, write, and execute privileges to the directory.

UNIX: The default value is taken from the TMPDIR environment variable if it is set to a non-null value. Otherwise the computer’s default temporary file directory is chosen; this is usually /tmp, but on IRIX64 systems, it is /var/tmp.

Windows: The default value is taken from the TEMP environment variable.

UNIX example:

```
md20071 nastran example
sdir=/scratch
```

Scratch files are created in the /scratch directory.

Windows example:

```
md20071 nastran example
sdir=d:\scratch
```

Scratch files are created in the d:\scratch directory.
If a DMP run was selected with $dmpParallel \geq 1$, unique task-specific scratch directories may be set for each host using the standard PATH separator, i.e., ":" on UNIX and ";" on Windows, to separate entries. The directories will be paired with each host in a round-robin order, that is, the list will be reused if more tasks than directories are specified.

See “Running Distributed Memory Parallel (DMP) Jobs” on page 163 for additional information.

UNIX example:
```
md20071 nastran example
dmp=4 \
sdir=/scratch1:/scratch2
```

In this example, /scratch1 will be used for the first and third tasks, while /scratch2 will be used for the second and fourth tasks.

<table>
<thead>
<tr>
<th>slaveout</th>
<th>slaveout=\text{yes, no}</th>
<th>Default: No</th>
</tr>
</thead>
</table>

Specifies the output files from the slave nodes are to be copied back to the local node.

<table>
<thead>
<tr>
<th>smaster</th>
<th>\text{smaster=\text{size}}</th>
<th>Default: Computer dependent</th>
</tr>
</thead>
</table>

Specifies an alternate default size for the MASTER DBset. The computer-dependent default is listed in “Computer Dependent Defaults” on page 374. This default is overridden by an INIT FMS statement.

The size is specified as the number of blocks (BUFFSIZE words long) or the number of words or bytes followed by one of the modifiers: "T", "TW", "TB", "G", "GW", "GB", "M", "MW", "MB", "K", "KW", "KB", "W", "B". See “Specifying Memory Sizes” on page 104 for a description of these modifiers.

<table>
<thead>
<tr>
<th>Note</th>
<th>No attempt is made to verify if the MASTER DBset can ever grow to the size specified by this keyword.</th>
</tr>
</thead>
</table>

Example:
```
md20071 nastran example
smaster=1024gb
```

Defines the default size of the MASTER DBset as 1 TB.

Example:
```
md20071 nastran example
smaster=.5tb
```

Defines the default size of the MASTER DBset as .5TB or 512GB.
**smemory**  
**smemory=value**  
Default: 0 (SUPER-UX)  
100 (all others)  
Specifies the amount of space in open core to reserve for scratch memory.

The size is specified as the number of blocks (BUFFSIZE words long) or the number of words or bytes followed by one of the modifiers: “T”, “TW”, “TB”, “G”, “GW”, “GB”, “M”, “MW”, “MB”, “K”, “KW”, “KB”, “W”, “B”. See “Specifying Memory Sizes” on page 104 for a description of these modifiers. The value specified using this keyword may be overridden by the FMS statement INIT SCRATCH (MEM=value).

Example:  
```
md20071 nastran example
smem200
```
This example reserves 200 GINO blocks for scratch memory.

Example:  
```
md20071 nastran example
smem=4mw
```
This example reserves 4,194,304 words for scratch memory.

Example:  
```
md20071 nastran example
smem=2.5mw
```
This example reserves 2,621,440 words for scratch memory.

**solve**  
**solve=number**  
Default: -1 (print up to 50 messages)  
Controls matrix decomposition. See the “nastran Command and NASTRAN Statement” in Chapter 1 of the MD Nastran Quick Reference Guide for more information on this keyword.

**sparse**  
**sparse=number**  
Default: See QRG.  
Sparse matrix method selection. This keyword may also be set with the “sys126” command line keyword. See the MD Nastran Quick Reference Guide for information on the default value and legal values for this keyword.

**spintime**  
**spintime=value**  
Default: IRIX64:10000000  
(IRIX64) Specifies the number of times to wait in a spin-wait loop before blocking the thread. MD Nastran slave threads (i.e., the threads that will run MD Nastran sub tasks) spin wait until there is work to do. This makes them immediately available when a parallel region is reached. However, spin waiting wastes processor resources. After a specified spin-wait time has elapsed, the threads block themselves using a system call.

Note that blocking is transparent to MD Nastran: blocked threads are automatically unblocked when a parallel region is reached. Once a thread is blocked, another system call is required to activate it again. This makes the response time much longer when starting up a parallel region. If the value is set to zero, the slave threads will block themselves immediately.
This keyword may also be set by the MSC_SPINTIME environment variable. The environment variable overrides the RC files, and the command line overrides the environment variable.

### ssr

**ssr=size**

Default: Computer dependent

Specifies an alternate default size for the SCRATCH DBset. The computer-dependent default is listed in “Computer Dependent Defaults” on page 374. This default is overridden by an INIT FMS statement. If the value “ssr=estimate” is specified, ESTIMATE will be used to determine a suitable default.

The size is specified as the number of blocks (BUFFSIZE words long) or the number of words or bytes followed by one of the modifiers: “T”, “TW”, “TB”, “G”, “GW”, “GB”, “M”, “MW”, “MB”, “K”, “KW”, “KB”, “W”, “B”. See “Specifying Memory Sizes” on page 104 for a description of these modifiers.

**Note:** No attempt is made to verify if the SCRATCH DBset can ever grow to the size specified by this keyword.

| Example: | md20071 nastran example ssr=1024gb |
| Example: | md20071 nastran example ssr=.5tb |

Defines the default size of the SCRATCH DBset as 1 TB.

Defines the default size of the SCRATCH DBset as .5TB or 512GB.

### subcomplex

**subcomplex=name**

Default: Current complex

(SPP-UX) Specifies the name of the sub complex where the MD Nastran job is to be run. This may be desirable if specific sub complexes have been configured to run MD Nastran.

Example: | md20071 nastran example subcomplex=crunch |

This example runs the MD Nastran job on the sub complex named “crunch”.

### submit

**submit=[list=]definition**

Default: None

(UNIX) Defines the command and options used to run a job when the “queue” keyword is specified. The “submit” keyword, only specified in RC files, consists of an optional queue list, followed by the command definition for the specified queues as shown below:

```markdown
submit=list=command
submit=command
```

When specified, the list contains one or more “queue” names separated by commas. If a queue list is not supplied, the command applies to all queues.
The *command* section of the "submit" keyword value defines the *command* used to run a job when a "queue" keyword is supplied that matches a queue name in the *list*. The *command* can contain keyword names enclosed in percent "%" signs that are replaced with the value of the keyword before the command is run. A complete description of the *command* is found in "Customizing Queue Commands (UNIX)" on page 83.

**sun_io**

<table>
<thead>
<tr>
<th>sun_io</th>
<th>Default: None</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Solaris)</td>
<td>Enables Sun’s enhanced library for database I/O.</td>
</tr>
<tr>
<td>(Primepower)</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** For maximum performance, the striped file system containing the files subject to SUN_IO should be created with the Veritas File Manager. The BUFFSIZE should match the interleave size of the disk stripe.

The control string is composed of one or more filename-options pairs of the form:

\[
p_1, p_2, p_3 : \text{file-templates}
\]

where:

- **p1**: Number of I/O threads; default is \(\max(n_{pp}, 8)\). Setting \(p_1=0\) will select default of no read-ahead.
- **p2**: Number of read-ahead buffers per thread. \(p_2 \geq 0\). Setting \(p_2=0\) will select default of 4 MB.
- **p3**: Read-ahead threshold. \(p_3 \geq 0\). Setting \(p_3=0\) will select the default of 256 KB.
- **file_templates**: Colon separated list of filename templates, there is no default. Examples are "*DBALL*" to match all files ending in "DBALL" and "*DBALL:*SCR*" to match all files ending in "DBALL" and all files with "SCR" anywhere in the name.
For each of the filenames listed in file-templates, $p_1$ pages, each of $p_2 \times \text{BUFFSIZE}$ words, will be read ahead if the number of consecutive reads exceeds $p_3$.

The additional main memory consumed by the SUN_IO facility is:

$$p_1 \times p_2 \times \text{BUFFSIZE} \times n_{\text{files}}$$

where $n_{\text{files}}$ is the number of files matched by file_templates.

This keyword may also be set by the MSC_SUN_IO environment variable. The environment variable overrides the RC files, and the command line overrides the environment variable.

Example:

```
md20071 nastran example
    'sun_io=*SCR*'
```

This example uses the defaults for $p_1$, $p_2$, and $p_3$ on the SCRATCH and SCR300 files.

Example

```
md20071 nastran example \
    'sun_io=2,3:*DBALL:*SCR*'
```

This example creates 2 I/O threads, each reading 3 buffers ahead for the DBALL, SCRATCH, and SCR300 files.

symbol=\text{name}=\text{string}

Default: None

Defines a symbolic (or logical) name used in ASSIGN and INCLUDE statements and in command line arguments. This keyword may be specified in initialization or RC files and on the command line. The symbol definition may include references to previously defined symbols or environment variables using the standard "$\text{name}$" or "${\text{name}}" syntax on UNIX or \%\text{name}\% syntax on Windows. For convenience, the character separating the "symbol" and "name" specification and the "name" and "string" specification may be either an equal sign ("=") or a hash mark ("#"). The use of a hash mark allows this keyword to be specified as an argument to a Windows .bat file.

If "node" is specified, symbolic names defined using this keyword are not used on the local system. Instead the specified values are passed to the remote system. This means that any pathnames must be valid on the remote system. Use the "lsymbol" keyword to specify symbolic names for the local system.

If "node" is not specified, symbolic names defined using the "lsymbol" keyword are processed as if they were defined using the "symbol" keyword.

Symbolic names are processed in the order they are encountered while processing the initialization and RC files and the command line. If a duplicate symbolic name is encountered, the new value replaces the previously specified value.
Symbolic names must be 16 characters or less, the value assigned to the symbolic name must be 256 characters or less. If the symbolic name used in an ASSIGN or INCLUDE statement or in command line arguments is not defined, it is left in the filename specification as is.

For example, many of the TPL and DEMO input data files have ASSIGN statements such as the following:

```
ASSIGN 'MASTER=DBSDIR:abc.master'
```

The string "DBSDIR:" specifies a symbolic name that is to be replaced by another string. The replaced string is defined by the "symbol=" keyword (or "lsymbol=" keyword if "node" was not specified) in an initialization or RC file, on the command line, or as environment variable. For example,

(UNIX)
```
symbol=DBSDIR=/dbs
```

(Windows)
```
symbol=DBSDIR=d:\dbs
```

When the previous ASSIGN statement is processed, the filename assigned to the logical name MASTER is /dbs/abc.master on UNIX and d:\dbs\abc.master on Windows. An alternate way of defining symbolic names is through the use of environment variables. For example, typing the following command

```
export DBSDIR=/dbs
```

at a Korn shell prompt, or

```
setenv DBSDIR /dbs
```

at a C-shell prompt, or

```
set DBSDIR=d:\dbs
```

at a Windows shell prompt, is equivalent to the "symbol" keyword definition.

**Note:** If a symbolic name is defined by both a symbol statement in an RC file and by an environment variable, the symbol statement value will be used.
The section titled “Environment Variables” on page 352 contains a list of environment variables that are automatically created by the nastran command. Of particular interest to the logical symbol feature are the OUTDIR and DBSDIR variables. These variables refer to the directory that will contain the output files (set using the “out” keyword) and the directory that will contain the permanent database files (set using the “dbs” keyword), respectively.

**sysfield**

```
sysfield=option,option,...
```

Default: None

Defines a global SYS value that is applied to Dbsets. Each option must have one of the following formats:

```
keyword=value
```

or

```
LNAMEXP(keyword=value,keyword=value,...)
```

where:

```
LNAMEXP = specifies a logical name expression using the UNIX/Windows file name specification format
```

Characters may be specified in any case. Internally, they are converted to upper-case before they are used.

Most characters in a substitution pattern match themselves but you can also use some special pattern-matching characters in the pattern.

These special characters are:

```
* = Matches any string, including the null string.
? = Matches any one character.
[...] = Matches any one of the characters enclosed in the square brackets.
```
\[!...\] = Matches any \textit{one} of the characters enclosed in the square brackets.

Matches any one character \textit{other than} one of the characters that follow the exclamation mark within square brackets.

Inside square brackets, a pair of characters separated by a \texttt{-} (minus) specifies a set of all characters that collate within the range of that pair, as defined by the ASCII collating sequence, so that \([a-dy]\) is equivalent to \([abcdy]\).

\texttt{keyword=value} = Specifies a keyword and value to be used for the Dbset file. If the entry is part of an option qualified by an \texttt{LNAMEXP} expression, the keyword and value will only be used for a Dbset file whose logical name is selected by the expression specified by \texttt{LNAMEXP}. Otherwise, the keyword and value will be used for all Dbset files. Note that a null \texttt{LNAMEXP} expression will match any logical name.

The "sysfield" keyword may be specified more than once. The options are processed in the order specified on the various specifications. If multiple "keyword=value" options specify the same keyword, the last one encountered is the one that is used. You may use the "whence" keyword to see the "sysfield" keyword values. Also, the "sysfield" keywords are listed in the LOG file.

See the sections titled “Using the SYS Field” on page 140 or “SYS Parameter Keywords” on page 349 for details on the valid keyword options.

Example: \texttt{md20071 nastran example sysfield=lock=no}

This example disables file locking for all Dbsets.

Example: \texttt{md20071 nastran example sysfield=lock=no \ sysfield=scr*(mapio=yes,lock=yes)}

This example disables file locking for all files and then enables filemapping ("mapio=yes") and turns file locking back on for Dbsets whose logical names start with "SCR". The end result is that file locking is disabled for all Dbsets except those whose logical names start with "SCR" and file mapping and file locking are enabled for Dbsets whose logical names start with "SCR".
**sysn**

**sysn=value**  
Default: None

Sets the **SYSTEM(n)** cell to **value**. This keyword may be repeated any number of times. All non-repeated cells are used, but only the last repeated cell is used. If there is a "name" associated with the **SYSTEM(n)** value, that keyword will also be set to **value**. The System Cell number to System Cell name equivalence is listed in the "**nastran Command and NASTRAN Statement**" in Chapter 1 of the *MD Nastran Quick Reference Guide*. The form "**system(n)=value**" may be used, but the entire keyword-value string must be quoted when used on a UNIX command line.

Example:  
```
md20071 nastran example
sys114=200
```

or  
```
md20071 nastran example
"system(114)=200"
```

These examples set **SYSTEM(114)** to 200. The second example shows how to quote the parenthetic form. Also, in this example, since **SYSTEM(114)** has the name "BUFFPOOL", the value of the "buffpool" keyword is also set to 200.

**t3skew**

**t3skew=number**  
Default: 30.0

Controls minimum vertex angle for **TRIA3** elements at which User Warning Message 5491 is issued. See the *MD Nastran Quick Reference Guide*, Section 1, The NASTRAN Statement, for more information on this keyword.

**tetraar**

**tetraar=number**  
Default: 100.0

Specifies maximum allowable aspect ratio of longest to shortest edge for the **CTETRA** element. See the "**nastran Command and NASTRAN Statement**" in Chapter 1 of the *MD Nastran Quick Reference Guide* for more information on this keyword.

**threads**

**threads=value**  
Default: None

*(IRIX64)* A number of features are provided in the IRIX64 version of MD Nastran that allow sophisticated users to override multiprocessing defaults and tailor a job's parallelism to their particular requirements.

Threads are used by IRIX to implement MD Nastran tasks. For maximal performance, there should be one thread per MD Nastran task and one processor per thread. An excess number of threads will not help performance; if there are more MD Nastran tasks than threads or more threads than processors, a longer elapsed time will result.

The Dynamic Thread Management feature is available only in the MD Nastran Rank-N sparse solver (see the "rank" keyword). The Rank-N sparse solver is used widely in linear static analysis and Lanczos eigenvalue analysis jobs. Other MD Nastran parallel modules will run with a constant number of threads specified by the PARALLEL keyword.
The “threads” keyword specifies the suggested number of threads to be maintained by the Dynamic Thread Management feature. Setting a value for “threads” causes the runtime library to create an additional asynchronous “monitor” process that periodically awakens to monitor system load. When idle processors exist, this monitor process increases the number of threads up to the maximum that is specified by the “parallel” keyword. As the system load increases, the monitor process decreases the number of threads, possibly to as few as one. If “threads” has not been set, this feature is disabled and the constant number of threads specified via the “parallel” keyword will be used.

This keyword may also be set by the MP_SUGNUMTHD environment variable. The environment variable overrides the RC files, and the command line overrides the environment variable.

**thread_max**

*thread_max=value*  
Default: parallel

(IRIX64) Specifies an upper bound on the number of threads that a job will use when “threads” is also set. The value must satisfy the relation

\[ \text{thread}_{\text{min}} \leq \text{thread}_{\text{max}} \leq \text{parallel} \]

where parallel is the value specified by the “parallel” keyword.

This keyword may also be set by the MP_SUGNUMTHD_MAX environment variable. The environment variable overrides the RC files, and the command line overrides the environment variable.

**thread_min**

*thread_min=value*  
Default: 1

(IRIX64) Specifies a lower bound on the number of threads a job will use when “threads” is also set. The value must satisfy the relation

\[ 1 \leq \text{thread}_{\text{min}} \leq \text{thread}_{\text{max}} \]

This keyword may also be set by the MP_SUGNUMTHD_MIN environment variable. The environment variable overrides the RC files, the command line overrides the environment variable.

**thread_verbose**

*thread_verbose=yes,no*  
Default: No

(IRIX64) Controls the output of informational messages. If “thread_verbose=yes” is set, the monitoring process will write messages to the LOG file whenever it changes the number of threads.

This keyword may also be set by the MP_SUGNUMTHD_VERBOSE environment variable. The environment variable overrides the RC files, the command line overrides the environment variable.

**trans**

*trans=yes,no,auto*  
Default: no (local)

auto (remote)

If the “node” keyword is not specified, this keyword indicates the XDB file is to be translated to a neutral-format file using the TRANS utility. The output file will have the file type “.ndb”.

---

**Keywords and Environment Variables**

CHAPTER B

345
UNIX only: If the “node” keyword is specified, this keyword indicates how an
XDB file is to be copied back to the local node. If “trans=auto” is specified,
the XDB file will be copied using TRANS/RECEIVE if the two computers use
different floating point formats or by a binary copy if the floating point formats
are the same. If “trans=yes” is specified, the XDB is always copied using
TRANS on the remote node and RECEIVE on the local node (this may be
needed if the floating point formats are identical but the file formats are not).
If “trans=no” is specified, the XDB file will not be copied back.

Example:
```
md20071 nastran example
trans=yes
```

This example will run MD Nastran and then convert the XDB file, if written,
to neutral format using TRANS.

UNIX example:
```
md20071 nastran example
node=othernode \
trans=yes
```

This example will run MD Nastran on node othernode and copy the XDB file
back using TRANS/RECEIVE.

**use_aio**

(HP-UX 11)

Enables HP’s enhanced library for database I/O. Setting “use_aio=yes” will
enable the library for all *.SCRATCH and *.SCR300 files, using $n_{cpu} - 1$
threads to control the asynchronous read-aheads.

This keyword may also be set by the USE_AIO environment variable. The
environment variable overrides the RC files, and the command line overrides
the environment variable. Setting the environment variable to any non-null
value is equivalent to “use_aio=yes”; unset the environment variable to set
“use_aio=no”.

Example:
```
md20071 nastran example
use_aio=yes
```

This example will run MD Nastran on with HP’s AIO library enabled.

The library is controlled by a number of environment variables. They include:

**AIO_FLIST**
Comma-separated list of filenames. The default is “*.SCRATCH, *.SCR300”.

**AIO_THREADS**
Maximum number of concurrent I/O
threads per file. The default is $n_{cpu} - 1$.

**AIO_BUFFERS**
Maximum number of I/O buffers per
file. The default is $n_{cpu} - 1$.

**AIO_PATDEPTH**
Number of I/Os to detect sequential
access. The default is 3.
username  
username = name  
Default: Current user name

(UNIX) Specifies an alternate username on the remote host when the “node” keyword is specified. This keyword may only be specified on the command line.

Example: 
```
md20071 nastran example 
node=othernode 
user=fred
```

This example will run MD Nastran on node othernode as user “fred”.

usparse  
usparse = number  
Default: See the description below.

Unsymmetric sparse matrix method selection. This keyword may also be set with the “sys209” command line keyword. See the MD Nastran Quick Reference Guide for information on the default value and legal values for this keyword.

version  
version = version_number  
Default: Latest installed version.

Specifies the version number. The keyword may only be specified on the command line or in the command initialization file.

Example: 
```
md20071 nastran example 
version=68.2
```

This example will run MSC.Nastran V68.2 assuming it has been installed in the same installation base directory as this version of MD Nastran.

whence  
whence = keyword_list  
Default: None

Displays value and source for listed keywords. This keyword may be used to determine a keyword's value and where it was set. An input datafile (JID) is optional; the job will not be run. If the "node" keyword is specified, the request will be passed to the remote node for processing. Otherwise, information will be displayed for the local node. If multiple “whence” keywords are specified, the keywords in the various keyword lists will be concatenated, except that if a null list is specified, all existing keywords in the accumulated list will be deleted. Any keywords in the "keyword_list" that have the format "sys n" will attempt to return the value associated with the System Cell name associated with system cell n, if possible. The entries in the "keyword_list" may also request information about a PARAM name. These entries have the format "p:name", where "name" is the PARAM name (not the name of the associated PARAM keyword, if any).

Normally, the output is two lines for each keyword. The first line specifies the "source", i.e., from where the keyword value is obtained; the second line specifies the keyword and its value. The only exception is when the keyword is "symbol", "system" or "j.params". In these cases, there will be multiple lines of keyword value information. If an unknown keyword is specified, a "User Warning Message" will be generated and the keyword will be ignored.
Example:  
```bash
md20071 nastran iter=yes
whence=sys1,bpool
whence=sscr,iter
```

Assuming that none of these values is modified in configuration files, the output from this request is:

MD Nastran V2007.0 (...) ...

$ internal default
   sysi=8193
$ internal default
   bpool=37
$ internal default
   sscr=250000
$ command line[1]
   iter=yes

**xhost**

xhost=yes,no  Default: No

(UNIX) Indicates if the xhost(1) command is to be run. The xhost(1) command may be required if the “node” keyword and either “xmon=yes” or “xmon=kill” are specified. The argument to xhost(1) will be the node specified by the “node” keyword. This keyword is ignored if the “node” keyword is not specified.

**xmonast**

xmonast=yes,no,kill  Default: No

(UNIX) Indicates if XMONAST is to be run to monitor the MD Nastran job. If “xmonast=yes” is specified, XMONAST will be automatically started; you must manually exit XMONAST when the MD Nastran job has completed. If “xmonast=kill” is specified, XMONAST will start and will automatically exit when the MD Nastran job has completed.

Example:  
```bash
md20071 nastran example
xmon=kill
```

This example runs the XMONITOR utility while the MD Nastran job is running. Once the job completes, the XMONITOR program is automatically terminated.
### SYS Parameter Keywords

#### async
- **Keyword:** async
- **Default:** No
- **Syntax:** async=yes,no,must
- **Usage:** This keyword specifies the file is to be read using asynchronous I/O. If “async=yes” is specified and a memory allocation operation fails, then unbuffered disk I/O will be used. If “async=must” is specified and a memory allocation operation fails, then a fatal error will be issued and the job terminated. See “Using Asynchronous I/O” on page 144 for further information.

#### buffio
- **Keyword:** buffio
- **Default:** No
- **Syntax:** buffio=yes,no,must
- **Usage:** This keyword specifies the file is to be buffered. If “buffio=yes” is specified and a memory allocation operation fails, then unbuffered disk I/O will be used. If “buffio=must” is specified and a memory allocation operation fails, then a fatal error will be issued and the job terminated. See “Using Buffered I/O” on page 143 for further information.

#### lock
- **Keyword:** lock
- **Default:** No for Delivery DBsets, Yes for all others.
- **Syntax:** lock=yes,no
- **Usage:** Specifies the file is to be locked when it is opened. Locking a file prevents two or more MD Nastran jobs from interfering with one another; however, this does not prevent any other program or operating system command from modifying the file.

**UNIX**
- System(207) can also be used to globally control DBset locking. Setting system(207)=1 will disable locking unless overridden for a specific file by SYS=LOCK=YES on an ASSIGN FMS statement. Setting system(207)=0 will enable locking of read-write DBsets unless overridden for a specific file by SYS=LOCK=NO on an ASSIGN FMS statement.

#### mapio
- **Keyword:** mapio
- **Default:** No
- **Syntax:** mapio=yes,no,must
- **Usage:** This keyword specifies the file is to be mapped. If “mapio=yes” is specified and a mapping operation fails, then normal disk I/O will be used. If “mapio=must” is specified and a mapping operation fails, then a fatal error will be issued and the job terminated. See “Using File Mapping” on page 141 for further information.
report  

Requests that a summary report about the number of file operations and other information about the I/O processing done for a particular file be written to the file defined by stderr when the file is closed. In addition, if TIMING=YES is specified, this report will contain timing information about the various steps involved in the I/O processing. If ASYNC=YES, BUFFIO=YES or MAPIO=YES, the report will contain additional information about the processing specific to these methods.

timing  

Requests that operation timing be enabled for the file. This timing information will be included in the .f04 file and, if REPORT=YES is also in effect, in the report written to stderr.

wnum  

Specifies the number of windows or buffers that will be maintained for each mapped, buffered or asynchronous I/O file. The use of multiple windows or buffers permits multiple I/O streams to target a file (e.g., simultaneously reading one matrix and writing another) without forcing an excessive number of window remap operations or buffered read/writes. The number must be between 1 through 32 inclusive, values outside of this range are ignored without acknowledgement.

wsize  

File Mapping. Specifies the size of the window mapping the file into memory. The window is that portion of the file that is visible through the map. If the window is the same size as the file, then the entire file is visible. If the window is smaller than the file, then any portion of the file within the window or windows can be directly accessed; the rest of the file cannot be accessed until a window is remapped to include the desired file location. The default is 128KB or 4*BUFFSIZE, whichever is larger.
Buffered I/O. Specifies the size of the buffer read from or written to disk. If the buffer is the same size as the file, then the entire file is memory resident. If the buffer is smaller than the file, then any portion of the file within the buffer or buffers can be directly accessed; the rest of the file cannot be accessed until a buffer is read to include the desired file location. The default is 4*BUFFSIZE or 64K, whichever is larger.

Asynchronous I/O. Specifies the size of the buffer used to hold data read from disk. If the buffer is the same size as the file, then the entire file is memory resident. If the buffer is smaller than the file, then any portion of the file within the buffer or buffers can be directly accessed; the rest of the file cannot be accessed until a buffer is read to include the desired file location. The default is 8*BUFFSIZE or 64KB, whichever is larger.

The total window or buffer size (WNUM value * WSIZE value) is limited to 25% of the available address space or, for Windows, to 25% of the physical memory. The address space limit is displayed by the “limits” special function, see “Using the Help Facility and Other Special Functions” on page 101, as the “Virtual Address Space” limit. If the address space limit or physical memory cannot be determined for a particular platform, a value of 64MB for 32-bit pointer systems and 8GB for 64-bit pointer systems is used as the 25% limit value. If “wsize=0” is specified for a read-only file, the entire file will be mapped or buffered into memory, subject to the 25% address space limit. (The 25% limit can be overridden if the numeric value is specified as a negative number. The 25% test will be suppressed and the actual window size value will be the absolute value of the specified numeric value. It is the user’s responsibility to ensure that the specified value is valid and does not cause performance problems.)

The size is specified as a memory size, see “Specifying Memory Sizes” on page 104.

If size is less than the file’s BUFFSIZE, then size is multiplied by BUFFSIZE.
Environment Variables

The following environment variables affect the execution of the nastran command.

Table B-1  Environment Variables Affecting the nastran Command

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPLAY</td>
<td>UNIX: The default display for xmonast.</td>
</tr>
<tr>
<td>DMPI_HOSTFILE</td>
<td>HP Alpha UNIX: Alternate means to set the “hosts” keyword.</td>
</tr>
<tr>
<td>FF_IO_DEFAULTS</td>
<td>IRIX64, Linux64: Alternate means to set the “ff_io_default” keyword.</td>
</tr>
<tr>
<td>FF_IO_OPTS</td>
<td>IRIX64, Linux64: Alternate means to set the “ff_io_opt” keyword.</td>
</tr>
<tr>
<td>HOME</td>
<td>UNIX: The user’s home directory.</td>
</tr>
<tr>
<td>HOMEDRIVE</td>
<td>Windows: The user’s home drive.</td>
</tr>
<tr>
<td>HOMEPATH</td>
<td>Windows: The user’s home directory.</td>
</tr>
<tr>
<td>HPIO_PARAM</td>
<td>SUPER-UX: Alternate means to set the “hpio_param” keyword.</td>
</tr>
<tr>
<td>LM_LICENSE_FILE</td>
<td>Alternate means to set the “authorize” keyword.</td>
</tr>
<tr>
<td>LOGNAME</td>
<td>UNIX: The user ID.</td>
</tr>
<tr>
<td>MP_ADAPTER_USE</td>
<td>AIX: Alternate means to set the “adapter_use” keyword.</td>
</tr>
<tr>
<td>MP_BYPASS_OFF</td>
<td>IRIX64: Alternate means to set “bypass_off” keyword.</td>
</tr>
<tr>
<td>MP_CPU_USE</td>
<td>AIX: Alternate means to set the “cpu_use” keyword.</td>
</tr>
<tr>
<td>MP_EUIDevice</td>
<td>AIX: Alternate means to set the “euidevice” keyword.</td>
</tr>
<tr>
<td>MP_EUILIB</td>
<td>AIX: Alternate means to set the “eulib” keyword.</td>
</tr>
<tr>
<td>MP_HOSTFILE</td>
<td>AIX: Alternate means to set the “hosts” keyword.</td>
</tr>
<tr>
<td>MP_PROCS</td>
<td>AIX: Alternate means to set the “dmparallel” keyword.</td>
</tr>
<tr>
<td>MP_RESD</td>
<td>AIX: Alternate means to set the “resd” keyword.</td>
</tr>
<tr>
<td>MP_SUGNUMTHD</td>
<td>IRIX64: Alternate means to set the “threads” keyword.</td>
</tr>
<tr>
<td>MP_SUGNUMTHD_MAX</td>
<td>IRIX64: Alternate means to set the “thread_max” keyword.</td>
</tr>
<tr>
<td>MP_SUGNUMTHD_MIN</td>
<td>IRIX64: Alternate means to set the “thread_min” keyword.</td>
</tr>
<tr>
<td>MP_SUGNUMTHD_VERBOSE</td>
<td>IRIX64: Alternate means to set the “threadVERBOSE” keyword.</td>
</tr>
<tr>
<td>MSC_ARCH</td>
<td>Specifies the MD Nastran architecture.</td>
</tr>
<tr>
<td>MSC_BASE</td>
<td>If set, the script will use this directory as the install_dir.</td>
</tr>
<tr>
<td>MSC_ISHELLEXT</td>
<td>Alternate means to set the “ishellext” keyword.</td>
</tr>
</tbody>
</table>
The following environmental variables are available for use by the “pre” and “post” keywords.

Table B-1  Environment Variables Affecting the nastran Command (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBSDIR</td>
<td>The directory part of MSC_DBS, i.e., the directory that will contain the permanent database files.</td>
</tr>
<tr>
<td>DELDIR</td>
<td>Directory containing the solution sequence source files (install_dir/md20071/nast/del on UNIX and install_dir/md20071/nast/del on Windows).</td>
</tr>
<tr>
<td>DEMODIR</td>
<td>Directory containing DEMO library (install_dir/md20071/nast/demo on UNIX and install_dir/md20071/nast/demo on Windows).</td>
</tr>
<tr>
<td>JIDDIR</td>
<td>Directory containing the input file.</td>
</tr>
<tr>
<td>MSC_APP</td>
<td>yes,no</td>
</tr>
</tbody>
</table>
Table B-2  "Pre" and "Post" Keyword Environment Variables (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSC_ASG</td>
<td>MSC use only.</td>
</tr>
<tr>
<td>MSC_ARCH</td>
<td>The actual architecture used by the nastran command.</td>
</tr>
<tr>
<td>MSC_LICENSE_FILE</td>
<td>Licensing value.</td>
</tr>
<tr>
<td>MSC_BASE</td>
<td>The actual install_dir used by the nastran command.</td>
</tr>
<tr>
<td>MSC_DBS</td>
<td>Default prefix of permanent databases.</td>
</tr>
<tr>
<td>MSC_EXE</td>
<td>Executable path.</td>
</tr>
<tr>
<td>MSC_JID</td>
<td>Input data file path.</td>
</tr>
<tr>
<td>MSC_MEM</td>
<td>Open core memory size in words.</td>
</tr>
<tr>
<td>MSC_OLD</td>
<td>yes,no</td>
</tr>
<tr>
<td>MSC_OUT</td>
<td>Prefix of F06, F04, and LOG files.</td>
</tr>
<tr>
<td>MSC_SCR</td>
<td>yes,no</td>
</tr>
<tr>
<td>MSC_SDIR</td>
<td>Default prefix of scratch databases.</td>
</tr>
<tr>
<td>MSC_VERSD</td>
<td>MSC use only.</td>
</tr>
<tr>
<td>OUTDIR</td>
<td>Output file directory.</td>
</tr>
<tr>
<td>SSSALTERDIR</td>
<td>Directory containing SSS alters</td>
</tr>
<tr>
<td></td>
<td>(install_dir/md20071/nast/sssalter on UNIX and install_dir/md20071/nast/misc/sssalter on Windows).</td>
</tr>
<tr>
<td>TEMP</td>
<td>Windows: Temporary directory.</td>
</tr>
<tr>
<td>TMPDIR</td>
<td>UNIX: Temporary directory.</td>
</tr>
<tr>
<td>TPLDIR</td>
<td>Directory containing TPL library</td>
</tr>
<tr>
<td></td>
<td>(install_dir/md20071/nast/tpl on UNIX and install_dir/md20071/nast/tpl on Windows).</td>
</tr>
</tbody>
</table>
Other Keywords

The following keywords are available for use by the nastran command and script templates. You will generally not need to set or use these values.

Table B-3 Other Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pathname of the nastran command.</td>
</tr>
<tr>
<td>0.acceptdeny</td>
<td>Pathname of accept/deny utility used in this job.</td>
</tr>
<tr>
<td>0.dmp</td>
<td>DMP job template pathname.</td>
</tr>
<tr>
<td>0.dmpaccept</td>
<td>Pathname of dmpaccept utility.</td>
</tr>
<tr>
<td>0.dmpdeny</td>
<td>Pathname of dmpdeny utility.</td>
</tr>
<tr>
<td>0.ini</td>
<td>Command initialization file pathname.</td>
</tr>
<tr>
<td>0.kwds=filename</td>
<td>Pathname of User-defined general keywords file</td>
</tr>
<tr>
<td>0.lcl</td>
<td>Local job template pathname.</td>
</tr>
<tr>
<td>0.params=filename</td>
<td>Pathname of User-defined PARAM keywords file</td>
</tr>
<tr>
<td>0.rmt</td>
<td>Remote job template pathname.</td>
</tr>
<tr>
<td>0.rmtaccept</td>
<td>Pathname of rmtaccept utility.</td>
</tr>
<tr>
<td>0.rmtdeny</td>
<td>Pathname of rmtdeny utility.</td>
</tr>
<tr>
<td>0.srv</td>
<td>Server job template pathname.</td>
</tr>
<tr>
<td>0.tmplt</td>
<td>Alternate template pathname, overrides local/remote template selection logic.</td>
</tr>
<tr>
<td>a.addall=list</td>
<td>Comma separated list of extensions to be added to the j.all list</td>
</tr>
<tr>
<td>a.addapp=list</td>
<td>Comma separated list of extensions to be added to the j.app list.</td>
</tr>
<tr>
<td>a.addofp=list</td>
<td>Comma separated list of extensions to be added to the j.ofp list.</td>
</tr>
<tr>
<td>a.addold=list</td>
<td>Comma separated list of extensions to be added to the j.old list.</td>
</tr>
<tr>
<td>a.appdir</td>
<td>Application specific base pathname relative to MSC_BASE.</td>
</tr>
<tr>
<td>a.altmode</td>
<td>The INTEGER mode associated with the alternate architecture.</td>
</tr>
<tr>
<td>a.altmodedir</td>
<td>The directory name associated with the alternate architecture.</td>
</tr>
<tr>
<td>a.archdir</td>
<td>Architecture specific base pathname relative to MSC_BASE.</td>
</tr>
<tr>
<td>a.estimate</td>
<td>ESTIMATE executable filename relative to “a.archdir”.</td>
</tr>
<tr>
<td>a.exedir</td>
<td>Directory part of any file name specified by “executable”.</td>
</tr>
<tr>
<td>a.flex</td>
<td>Pathname of default FLEXlm license file.</td>
</tr>
<tr>
<td>a.fms</td>
<td>Comma-separated list of FMS keywords recognized in RC files.</td>
</tr>
<tr>
<td>a.k</td>
<td>Multiplier for K factor.</td>
</tr>
<tr>
<td>Keyword</td>
<td>Purpose</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------</td>
</tr>
<tr>
<td>a.msgcat</td>
<td>Pathname of default message catalog.</td>
</tr>
<tr>
<td>a.news</td>
<td>News filename relative to “a.appdir”.</td>
</tr>
<tr>
<td>a.port</td>
<td>Default FLEXlm port number.</td>
</tr>
<tr>
<td>a.rc</td>
<td>RC file basename. User RC files are prefixed by “.&quot;.</td>
</tr>
<tr>
<td>a.receive</td>
<td>RECEIVE executable filename relative to “a.archdir”.</td>
</tr>
<tr>
<td>a.release</td>
<td>Release number, same as MD Nastran version number.</td>
</tr>
<tr>
<td>a.sbcm</td>
<td>Pathname of default node-locked authorization code file.</td>
</tr>
<tr>
<td>a.solver</td>
<td>Solver executable filename relative to “a.archdir”.</td>
</tr>
<tr>
<td>a.sss</td>
<td>Delivery database filename relative to “a.archdir”.</td>
</tr>
<tr>
<td>a.tier</td>
<td>MSC internal variable.</td>
</tr>
<tr>
<td>a.touch</td>
<td>News file touch pathname.</td>
</tr>
<tr>
<td>a.trans</td>
<td>TRANS executable filename relative to “a.archdir”.</td>
</tr>
<tr>
<td>a.urc</td>
<td>File name of default User RC file.</td>
</tr>
<tr>
<td>a.xmonitor</td>
<td>XMONAST executable filename relative to “a.archdir”.</td>
</tr>
<tr>
<td>d.hosts</td>
<td>Blank separated list of per-task hostnames</td>
</tr>
<tr>
<td>d.jidvis</td>
<td>Blank separated list of per-task JID visibility flags.</td>
</tr>
<tr>
<td>d.outvis</td>
<td>Blank separated list of per-task output directory visibility flags.</td>
</tr>
<tr>
<td>d.rcmds</td>
<td>Blank separated list of per-task “rcmd” values.</td>
</tr>
<tr>
<td>d.sdirs</td>
<td>Blank separated list of per-task “sdirectory” values.</td>
</tr>
<tr>
<td>d.tid</td>
<td>DMP task ID.</td>
</tr>
<tr>
<td>dcmd</td>
<td>Debugger.</td>
</tr>
<tr>
<td>debug</td>
<td>Run solver under debugger.</td>
</tr>
<tr>
<td>j.all</td>
<td>Blank separated list of file types to be deleted at job completion if “delete=all” is specified.</td>
</tr>
<tr>
<td>j.app</td>
<td>Blank separated list of file types to be appended at job completion if “append=yes” is specified.</td>
</tr>
<tr>
<td>j.base</td>
<td>Job basename.</td>
</tr>
<tr>
<td>j.command</td>
<td>Job submittal command string.</td>
</tr>
<tr>
<td>j.dir</td>
<td>Job directory.</td>
</tr>
<tr>
<td>j.env</td>
<td>Job environment variable list.</td>
</tr>
<tr>
<td>j.expbase</td>
<td>Generated &lt;expjid&gt; base name.</td>
</tr>
<tr>
<td>j.expdir</td>
<td>Generated &lt;expjid&gt; directory.</td>
</tr>
</tbody>
</table>
### Table B-3  Other Keywords (continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>j.expjid</td>
<td>Generated <code>&lt;expjid&gt;</code> file name.</td>
</tr>
<tr>
<td>j.mode</td>
<td>Generated effective “mode” value in effect. Will be one of “”, “i4” or “i8”.</td>
</tr>
<tr>
<td>j.modedir</td>
<td>The directory associated with the “j.mode” value. NULL unless “j.mode” is one of “i4” or “i8”.</td>
</tr>
<tr>
<td>j.msg</td>
<td>Job completion message.</td>
</tr>
<tr>
<td>j.nascar</td>
<td>List of NASTRAN entries.</td>
</tr>
<tr>
<td>j.news</td>
<td>News file pathname.</td>
</tr>
<tr>
<td>j.nice</td>
<td>Nice command to be used for commands, set based on “nice” keyword. (UNIX Only).</td>
</tr>
<tr>
<td>j.ofp</td>
<td>Blank separated list of file types to be deleted at job completion if and only if they are empty.</td>
</tr>
<tr>
<td>j.old</td>
<td>Blank separated list of file types to be versioned or deleted under the “old” keyword.</td>
</tr>
<tr>
<td>j.out</td>
<td>Appended output file type.</td>
</tr>
<tr>
<td>j.params</td>
<td>Generated list of PARAM statements. Contains the result of INI file, RC file and command line PARAM processing</td>
</tr>
<tr>
<td>j.rcfiles</td>
<td>Comma-separated list of RC files.</td>
</tr>
<tr>
<td>j.server</td>
<td>MD Nastran server flag</td>
</tr>
<tr>
<td>j.shell</td>
<td>Shell debugging flag.</td>
</tr>
<tr>
<td>j.startdate</td>
<td>Job start date-time string.</td>
</tr>
<tr>
<td>j.title</td>
<td>Title of XMONAST icon.</td>
</tr>
<tr>
<td>j.tty</td>
<td>TTY name.</td>
</tr>
<tr>
<td>j.type</td>
<td>Space separated list of file types to be versioned.</td>
</tr>
<tr>
<td>j.unique</td>
<td>Job unique name.</td>
</tr>
<tr>
<td>job</td>
<td>Job script filename, created in out directory.</td>
</tr>
<tr>
<td>log</td>
<td>Pathname of LOG file.</td>
</tr>
<tr>
<td>msgdest</td>
<td>System message destination.</td>
</tr>
<tr>
<td>nprocessors</td>
<td>Number of processors.</td>
</tr>
<tr>
<td>ppc</td>
<td>Per-process CPU time limit.</td>
</tr>
<tr>
<td>ppm</td>
<td>Per-process memory limit.</td>
</tr>
<tr>
<td>prm</td>
<td>Per-request memory limit.</td>
</tr>
<tr>
<td>PWD</td>
<td>Current working directory.</td>
</tr>
</tbody>
</table>
Table B-3  Other Keywords (continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>r.altmode</td>
<td>The INTEGER mode associated with the remote mode alternate architecture.</td>
</tr>
<tr>
<td>r.altmodedir</td>
<td>The directory name associated with the remote node alternate architecture.</td>
</tr>
<tr>
<td>r.argv</td>
<td>List of arguments to be processed on rmt/dmp host.</td>
</tr>
<tr>
<td>r.expvis</td>
<td>&quot;expjid&quot; visibility flag for remote job. Value is &quot;yes&quot; or &quot;no&quot;.</td>
</tr>
<tr>
<td>r.jidvis</td>
<td>JID visibility flag.</td>
</tr>
<tr>
<td>r.oscode</td>
<td>Remote system operating system code, 1 = Windows, 2 = UNIX.</td>
</tr>
<tr>
<td>r.outvis</td>
<td>Output directory visibility flag.</td>
</tr>
<tr>
<td>r.rmtcode</td>
<td>Remote communications protocol, 1 = MSCRmtCmd, 2 = rsh/rcp.</td>
</tr>
<tr>
<td>r.rshell</td>
<td>Remote node Shell pathname. Only used when &quot;r.rmtcode&quot; is 1.</td>
</tr>
<tr>
<td>s.arch</td>
<td>System architecture name.</td>
</tr>
<tr>
<td>s.block</td>
<td>Words per disk block.</td>
</tr>
<tr>
<td>s.bpw</td>
<td>Bytes per word.</td>
</tr>
<tr>
<td>s.clock</td>
<td>CPU clock frequency.</td>
</tr>
<tr>
<td>s.config</td>
<td>CONFIG number.</td>
</tr>
<tr>
<td>s.cpu</td>
<td>CPU name.</td>
</tr>
<tr>
<td>s.hostname</td>
<td>Simple hostname.</td>
</tr>
<tr>
<td>s.model</td>
<td>System model name.</td>
</tr>
<tr>
<td>s.modeldata</td>
<td>Pathname of site specific model data.</td>
</tr>
<tr>
<td>s.nproc</td>
<td>Number of processors.</td>
</tr>
<tr>
<td>s.numeric</td>
<td>Encoded numerical format.</td>
</tr>
<tr>
<td>s.os</td>
<td>OS name.</td>
</tr>
<tr>
<td>s.osv</td>
<td>OS version.</td>
</tr>
<tr>
<td>s.pmem</td>
<td>Physical memory, in MB. Only known on HP Alpha UNIX, Solaris, SUPER-UX, and Windows.</td>
</tr>
<tr>
<td>s.proc</td>
<td>Default processor subtype.</td>
</tr>
<tr>
<td>s.rawid</td>
<td>Raw configuration number.</td>
</tr>
<tr>
<td>s.rcp</td>
<td>Remote file copy command.</td>
</tr>
</tbody>
</table>
### Table B-3  Other Keywords (continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>s.rsh</td>
<td>Remote shell command.</td>
</tr>
<tr>
<td>s.type</td>
<td>System description.</td>
</tr>
<tr>
<td>s.vmem</td>
<td>Virtual memory, in MB. Only known on Windows.</td>
</tr>
<tr>
<td>tcmd</td>
<td>Timing command.</td>
</tr>
</tbody>
</table>
## System Cell Keyword Mapping

The following table lists the System Cell Name - System Cell Number equivalence used by MD Nastran when processing the `sysn` and `whence` keywords:

<table>
<thead>
<tr>
<th>System Cell Name</th>
<th>System Cell Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>attdel</td>
<td>124</td>
</tr>
<tr>
<td>autoasgn</td>
<td>133</td>
</tr>
<tr>
<td>bfgs</td>
<td>145</td>
</tr>
<tr>
<td>buffpool</td>
<td>114</td>
</tr>
<tr>
<td>buffsize</td>
<td>1</td>
</tr>
<tr>
<td>chexaint</td>
<td>212</td>
</tr>
<tr>
<td>config</td>
<td>28</td>
</tr>
<tr>
<td>cordm</td>
<td>204</td>
</tr>
<tr>
<td>cpyinput</td>
<td>305</td>
</tr>
<tr>
<td>dblamkd</td>
<td>155</td>
</tr>
<tr>
<td>dbverchk</td>
<td>148</td>
</tr>
<tr>
<td>diaga</td>
<td>25</td>
</tr>
<tr>
<td>diagb</td>
<td>61</td>
</tr>
<tr>
<td>disksave</td>
<td>193</td>
</tr>
<tr>
<td>distort</td>
<td>213</td>
</tr>
<tr>
<td>f04</td>
<td>86</td>
</tr>
<tr>
<td>f06</td>
<td>2</td>
</tr>
<tr>
<td>fastio</td>
<td>194</td>
</tr>
<tr>
<td>fbsmem</td>
<td>146</td>
</tr>
<tr>
<td>fbsopt</td>
<td>70</td>
</tr>
<tr>
<td>frqseq</td>
<td>195</td>
</tr>
<tr>
<td>hicore</td>
<td>57</td>
</tr>
<tr>
<td>iter</td>
<td>216</td>
</tr>
<tr>
<td>ldqrkd</td>
<td>170</td>
</tr>
<tr>
<td>lochulk</td>
<td>143</td>
</tr>
<tr>
<td>massbuf</td>
<td>199</td>
</tr>
<tr>
<td>maxset</td>
<td>263</td>
</tr>
</tbody>
</table>
### Table B-4 System Cell Name -- System Cell Number

<table>
<thead>
<tr>
<th>System Cell Name</th>
<th>System Cell Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>metime</td>
<td>20</td>
</tr>
<tr>
<td>mindef</td>
<td>303</td>
</tr>
<tr>
<td>minfront</td>
<td>198</td>
</tr>
<tr>
<td>mperturb</td>
<td>304</td>
</tr>
<tr>
<td>mpyad</td>
<td>66</td>
</tr>
<tr>
<td>newhess</td>
<td>108</td>
</tr>
<tr>
<td>nlines</td>
<td>9</td>
</tr>
<tr>
<td>nsegadd</td>
<td>200</td>
</tr>
<tr>
<td>numseg</td>
<td>197</td>
</tr>
<tr>
<td>parallel</td>
<td>107</td>
</tr>
<tr>
<td>punch</td>
<td>64</td>
</tr>
<tr>
<td>q4skew</td>
<td>190</td>
</tr>
<tr>
<td>q4taper</td>
<td>189</td>
</tr>
<tr>
<td>quadint</td>
<td>141</td>
</tr>
<tr>
<td>radlst</td>
<td>88</td>
</tr>
<tr>
<td>radmtx</td>
<td>87</td>
</tr>
<tr>
<td>real</td>
<td>81</td>
</tr>
<tr>
<td>scr300</td>
<td>142</td>
</tr>
<tr>
<td>scr300del</td>
<td>150</td>
</tr>
<tr>
<td>scrsave</td>
<td>196</td>
</tr>
<tr>
<td>solve</td>
<td>69</td>
</tr>
<tr>
<td>sparse</td>
<td>126</td>
</tr>
<tr>
<td>t3skew</td>
<td>218</td>
</tr>
<tr>
<td>tetraar</td>
<td>191</td>
</tr>
<tr>
<td>usparse</td>
<td>209</td>
</tr>
</tbody>
</table>
System Descriptions

- Overview 364
- System Descriptions 365
- Numerical Data 370
- Computer Dependent Defaults 374
Overview

This appendix presents quantitative information for evaluating the processing requirements of MD Nastran. It includes system descriptions, numerical data, and information on computer dependent defaults.

Binary File Byte Ordering (Endian)

The term "endian" refers to the byte ordering for numeric data used by a particular computer architecture. "Big-endian" specifies that the most significant byte (MSB) of a data element is stored at the lowest byte address, while "little-endian" specifies that the least significant byte (LSB) of a data element is stored at the lowest byte address. Most UNIX platforms, e.g., almost all except HP Alpha, are big-endian machines, while all Intel x86 and compatible platforms, e.g., Intel Pentium and AMD Athlon and Opteron, including those running both Windows and Linux, are little-endian machines. Some architectures can be run in either endian mode. For example, the Intel Itanium processor runs in big-endian mode when running HP-UX and in little-endian mode when running Linux or Windows. The diagrams in Section C.3 illustrate the difference between big-endian and little-endian for both integer and floating point data.
System Descriptions

Table C-1  System Description – HP Alpha – UNIX

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Model(s)</td>
<td>EV4, EV5, EV6, EV7</td>
</tr>
<tr>
<td>Installed Timing Constants</td>
<td>GS140, 500/500, PW500a, ES40, ES45</td>
</tr>
<tr>
<td></td>
<td>3000/800, 3000/500, 2100-4/275, 2100-5/300, 4100-5/400, PW5000</td>
</tr>
<tr>
<td>Operating System(s)</td>
<td>Tru64 UNIX 5.1A-1885</td>
</tr>
<tr>
<td>Compilers</td>
<td>f90: Digital FORTRAN V5.5-2602</td>
</tr>
<tr>
<td></td>
<td>cc: Compaq C V6.4-014</td>
</tr>
<tr>
<td>Compiler Options</td>
<td>-O4 -tune ev6 -assume noaccuracy -assume dummy -math_library</td>
</tr>
<tr>
<td></td>
<td>fast -om</td>
</tr>
<tr>
<td>Word Length</td>
<td>32 bits</td>
</tr>
<tr>
<td>Build Type</td>
<td>LP-64</td>
</tr>
<tr>
<td>Memory Management</td>
<td>Virtual</td>
</tr>
</tbody>
</table>

Table C-2  System Description – HP-UX

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Model(s)</td>
<td>PA-RISC 2.0</td>
</tr>
<tr>
<td>Installed Timing Constants</td>
<td>250, 710, 712, 715, 720, 730, 735, 778, 800, 819, 889, 2200</td>
</tr>
<tr>
<td>Operating System(s)</td>
<td>PA-RISC 2.0:HP-UX 11.0</td>
</tr>
<tr>
<td>Compilers</td>
<td>PA-RISC 2.0, HP-UX 11.0: f90 2.6.5</td>
</tr>
<tr>
<td>Compiler Options</td>
<td>PA-RISC 2.0, HP-UX 11.0: +ppu +02 +save</td>
</tr>
<tr>
<td></td>
<td>+Olibcalls +DS2.0W +DA2.0W</td>
</tr>
<tr>
<td></td>
<td>+Onolimit +Odataprefetch</td>
</tr>
<tr>
<td>Word Length</td>
<td>32 bits/64-bits</td>
</tr>
<tr>
<td>Build Type</td>
<td>LP-64 (HP-UX 11.0)/ILP-64</td>
</tr>
<tr>
<td>Memory Management</td>
<td>Virtual</td>
</tr>
</tbody>
</table>
Table C-3  System Description – Intel – Linux

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Model(s)</td>
<td>Intel and Intel-compatible</td>
</tr>
<tr>
<td>Installed Timing Constants</td>
<td>Pentium II 400 MHz, Pentium 4 IA64 733 MHz, Intel Nacona, AMD/Opteron</td>
</tr>
<tr>
<td>Operating System(s)</td>
<td>Linux 2.4.18 (Kernel)</td>
</tr>
<tr>
<td>Compiler</td>
<td>IA32: INTEL 8.1</td>
</tr>
<tr>
<td></td>
<td>IA64: INTEL 8.1</td>
</tr>
<tr>
<td></td>
<td>X8664: INTEL 9.1</td>
</tr>
<tr>
<td>Compiler Options</td>
<td>IA32: -02 -nbs -pad_source -save -zero</td>
</tr>
<tr>
<td></td>
<td>IA64: -02 -nds -pad_source -save -zero -ftz -tpp2 -Qdyncom XNSTRN</td>
</tr>
<tr>
<td>Compiler Options (continued)</td>
<td>X8664: -02 -nbs -w -290 -cm -WB</td>
</tr>
<tr>
<td></td>
<td>-pad -source -W0 -save -zero -Qdyncom XNSTRN</td>
</tr>
<tr>
<td>Word Length</td>
<td>32 bits</td>
</tr>
<tr>
<td>Build Type</td>
<td>IA32: ILP-32</td>
</tr>
<tr>
<td></td>
<td>IA64: LP-64</td>
</tr>
<tr>
<td>Memory Management</td>
<td>Virtual</td>
</tr>
<tr>
<td>Size of Executable</td>
<td>IA32: SMP: 56 MB, DMP: 57 MB</td>
</tr>
<tr>
<td></td>
<td>IA64: SMP: 102 MB, DMP: 104 MB</td>
</tr>
<tr>
<td></td>
<td>X8664: SMP: 52 MB, DMP: 53 MB</td>
</tr>
</tbody>
</table>

Table C-4  System Description – Intel – Windows

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Model(s)</td>
<td>Intel and Intel-compatible</td>
</tr>
<tr>
<td>Installed Timing Constants</td>
<td>Pentium II 400 MHz</td>
</tr>
<tr>
<td>Operating System(s)</td>
<td>Windows NT 4.0, 2000, XP</td>
</tr>
<tr>
<td>Compiler</td>
<td>Intel FORTRAN Compiler 8.1</td>
</tr>
<tr>
<td></td>
<td>Intel C++ Compiler 8.1</td>
</tr>
</tbody>
</table>
Table C-4  System Description – Intel – Windows

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler Options</td>
<td>ifl /nbs/w/w-990/cm/WB/G3/02/MT/Quppercase /W0</td>
</tr>
<tr>
<td></td>
<td>icl /MT/G3/02/W0</td>
</tr>
<tr>
<td>Word Length</td>
<td>32 bits</td>
</tr>
<tr>
<td>Build Type</td>
<td>ILP-32</td>
</tr>
<tr>
<td>Memory Management</td>
<td>Virtual</td>
</tr>
</tbody>
</table>

Table C-5  System Description – IBM pSeries – AIX

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Model(s)</td>
<td>Power3, Power4, Power5</td>
</tr>
<tr>
<td>Installed Timing Constants</td>
<td>320H, 375, 390, 39H, 43P, 530, 560, 580, 590, 950, 980E, 990</td>
</tr>
<tr>
<td>Operating System(s)</td>
<td>AIX 5.1</td>
</tr>
<tr>
<td>Compiler</td>
<td>xlf 8.1.1.4</td>
</tr>
<tr>
<td>cc 6.0.0.7</td>
<td></td>
</tr>
<tr>
<td>Compiler Options</td>
<td>xlf90 -q64 -qfixed -qsave -qlist -c -gexname</td>
</tr>
<tr>
<td></td>
<td>-qstrict -03 -qflttrap=ov:zero:inv:imp -qsigtrap</td>
</tr>
<tr>
<td></td>
<td>-qarch=pwr3 -qtune=pwr4</td>
</tr>
<tr>
<td></td>
<td>cc -c -q64 -qstrict -03 -qflttrap=ov:zero:inv:imp</td>
</tr>
<tr>
<td></td>
<td>-qarch=pwr3 -qtune=pwr4</td>
</tr>
<tr>
<td>Word Length</td>
<td>32 bits/64 bits</td>
</tr>
<tr>
<td>Build Type</td>
<td>LP-64/ILP-64</td>
</tr>
<tr>
<td>Memory Management</td>
<td>Virtual</td>
</tr>
</tbody>
</table>

Table C-6  System Description – NEC, SX8 – SUPER-UX

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Model(s)</td>
<td>SX-5, SX-6, SX-8</td>
</tr>
<tr>
<td>Installed Timing Constants</td>
<td>SX-5</td>
</tr>
<tr>
<td>Operating System(s)</td>
<td>SUPER-UX 14.1</td>
</tr>
<tr>
<td>Compiler</td>
<td>f90 2.0 Rev. 303a</td>
</tr>
<tr>
<td></td>
<td>SXC++ 1.0 Rev. 67</td>
</tr>
</tbody>
</table>
Table C-6  System Description – NEC, SX8 – SUPER-UX (continued)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler Options</td>
<td>-c -float0 -Cvopt -Wf,-Pe,-L,mrgmsg,transform,nomsg,-pvct1,nomsg, noassume,vwork=stack</td>
</tr>
<tr>
<td>Word Length</td>
<td>32 bits</td>
</tr>
<tr>
<td>Build Type</td>
<td>LP-64</td>
</tr>
<tr>
<td>Memory Management</td>
<td>Real</td>
</tr>
</tbody>
</table>

Table C-7  System Description – SGI R8K, R10K, R12K – IRIX64

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Model(s)</td>
<td>R8K, R10K, R12K</td>
</tr>
<tr>
<td>Installed Timing Constants</td>
<td>IP21, IP27</td>
</tr>
<tr>
<td>Operating System(s)</td>
<td>IRIX 6.5</td>
</tr>
<tr>
<td>Compilers</td>
<td>F90 7.4.2m</td>
</tr>
<tr>
<td>Compiler Options</td>
<td>-G 0 -O3 -TENV:X=0 -static -mips4 -64</td>
</tr>
<tr>
<td>Word Length</td>
<td>32 bits/64 bits</td>
</tr>
<tr>
<td>Build Type</td>
<td>LP-64/ILP-64</td>
</tr>
<tr>
<td>Memory Management</td>
<td>Virtual</td>
</tr>
</tbody>
</table>

Table C-8  System Description – Sun SPARC – Solaris

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Model(s)</td>
<td>UltraSPARC</td>
</tr>
<tr>
<td>Installed Timing Constants</td>
<td>UltraSPARC</td>
</tr>
<tr>
<td>Operating System(s)</td>
<td>UltraSPARC:Solaris 8</td>
</tr>
<tr>
<td>Compilers</td>
<td>F90 S1S11</td>
</tr>
<tr>
<td>Compiler Options</td>
<td>f90 -c -f77 -xrecursive -fast -Bstatic -w</td>
</tr>
<tr>
<td></td>
<td>-xtarget=ultra3 -xarch=v9a</td>
</tr>
<tr>
<td></td>
<td>-04 -ftrap=invalid,overflow,division</td>
</tr>
<tr>
<td></td>
<td>cc -xarch=v9a -c -O</td>
</tr>
<tr>
<td>Word Length</td>
<td>32 bits/64 bits</td>
</tr>
<tr>
<td>Build Type</td>
<td>LP-64/ILP-64</td>
</tr>
<tr>
<td>Memory Management</td>
<td>Virtual</td>
</tr>
</tbody>
</table>
Table C-9  System Description – Fujitsu PrimePower - Solaris

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Model(s)</td>
<td>PrimePower Series</td>
</tr>
<tr>
<td>Installed Timing</td>
<td>400, 600</td>
</tr>
<tr>
<td>Operating System(s)</td>
<td>Solaris 8.0</td>
</tr>
<tr>
<td>Compilers</td>
<td>Fujitsu f90 5.3</td>
</tr>
<tr>
<td>Compiler Options</td>
<td>-Bstatic -0 -KV9</td>
</tr>
<tr>
<td>Word Length</td>
<td>32 bits/64 bits</td>
</tr>
<tr>
<td>Build Type</td>
<td>LP-64/ILP-64</td>
</tr>
<tr>
<td>Memory Management</td>
<td>Virtual</td>
</tr>
</tbody>
</table>

Table C-10  System Description - IA64 - HP-UX

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Model(s)</td>
<td>Intel Itanium, Itanium2</td>
</tr>
<tr>
<td>Installed Timing Constants</td>
<td>2000, 2733, 4000, 4900, 5300, 5400, 5500, 5600</td>
</tr>
<tr>
<td>Operating System(s)</td>
<td>HP-UX 11.23</td>
</tr>
<tr>
<td>Compilers</td>
<td>f90 2.8 cc 6.02</td>
</tr>
<tr>
<td>Compiler Options</td>
<td>+ppu +save +02 +DStanium2 +DD64</td>
</tr>
<tr>
<td>Word Length</td>
<td>32 bits/64 bits</td>
</tr>
<tr>
<td>Build Type</td>
<td>LP-64/ILP-64</td>
</tr>
<tr>
<td>Memory Management</td>
<td>Virtual</td>
</tr>
</tbody>
</table>
# Numerical Data

Table C-11  Numerical Data – 32-bit, big endian, IEEE (All but HP Alpha, and Intel)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER Bit Representation</td>
<td>0 1 31</td>
</tr>
<tr>
<td></td>
<td>S Integer</td>
</tr>
<tr>
<td>REAL Bit Representation</td>
<td>0 1 8 9 31</td>
</tr>
<tr>
<td></td>
<td>S Exponent</td>
</tr>
<tr>
<td></td>
<td>Mantiss</td>
</tr>
<tr>
<td>Exponent Range for a REAL Number</td>
<td>±38</td>
</tr>
<tr>
<td>Precision of a REAL Variable</td>
<td>6 digits (24 bits)</td>
</tr>
<tr>
<td>DOUBLE PRECISION Bit Representation</td>
<td>0 1 11 12 31</td>
</tr>
<tr>
<td></td>
<td>S Exponent</td>
</tr>
<tr>
<td></td>
<td>Mantissa</td>
</tr>
<tr>
<td></td>
<td>Mantissa (continued)</td>
</tr>
<tr>
<td>Exponent Range for a DOUBLE PRECISION Number</td>
<td>±308</td>
</tr>
<tr>
<td>Precision of a DOUBLE PRECISION Variable</td>
<td>15 digits (53 bits)</td>
</tr>
</tbody>
</table>

Table C-12  Numerical Data – 32-bit, little endian, IEEE (HP Alpha, Intel)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER Bit Representation</td>
<td>31 30 0</td>
</tr>
<tr>
<td></td>
<td>S Integer</td>
</tr>
<tr>
<td>REAL Bit Representation</td>
<td>31 30 23 22 0</td>
</tr>
<tr>
<td></td>
<td>S Exponent</td>
</tr>
<tr>
<td></td>
<td>Mantiss</td>
</tr>
<tr>
<td>Exponent Range for a REAL Number</td>
<td>±38</td>
</tr>
</tbody>
</table>
### Table C-12  Numerical Data – 32-bit, little endian, IEEE (HP Alpha, Intel) (continued)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision of a REAL Variable</td>
<td>6 digits (24 bits)</td>
</tr>
<tr>
<td>DOUBLE PRECISION Bit Representation</td>
<td>63 62 52 51 32</td>
</tr>
<tr>
<td></td>
<td>S  Exponent  Mantissa</td>
</tr>
<tr>
<td></td>
<td>31 0</td>
</tr>
<tr>
<td></td>
<td>Mantissa (continued)</td>
</tr>
<tr>
<td>Exponent Range for a DOUBLE PRECISION Number</td>
<td>±308</td>
</tr>
<tr>
<td>Precision of a DOUBLE PRECISION Variable</td>
<td>15 digits (53 bits)</td>
</tr>
</tbody>
</table>

### Table C-13  Numerical Data – 64-bit, big endian

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER Bit Representation</td>
<td>0 1 63</td>
</tr>
<tr>
<td></td>
<td>S  Integer</td>
</tr>
<tr>
<td>REAL Bit Representation</td>
<td>0 1 15 16 63</td>
</tr>
<tr>
<td></td>
<td>S  Exponent  Mantissa</td>
</tr>
<tr>
<td>Exponent Range for a REAL Number</td>
<td>±2644</td>
</tr>
<tr>
<td>Precision of a REAL Variable</td>
<td>14 digits (48 bits)</td>
</tr>
<tr>
<td>DOUBLE PRECISION Bit Representation</td>
<td>0 1 15 16 63</td>
</tr>
<tr>
<td></td>
<td>S  Exponent  Mantissa</td>
</tr>
<tr>
<td></td>
<td>64 79 80 127</td>
</tr>
<tr>
<td></td>
<td>(Unused)  Mantissa (continued)</td>
</tr>
</tbody>
</table>
Table C-13  Numerical Data – 64-bit, big endian

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponent Range for a DOUBLE PRECISION Number</td>
<td>±2466</td>
</tr>
<tr>
<td>Precision of a DOUBLE PRECISION Variable</td>
<td>28 digits (96 bits)</td>
</tr>
</tbody>
</table>

Table C-14  Numerical Data – 64-bit, big endian (IEEE T90)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER Bit Representation</td>
<td>0 1 63</td>
</tr>
<tr>
<td></td>
<td>S Integer</td>
</tr>
<tr>
<td>REAL Bit Representation</td>
<td>0 1 11 12 63</td>
</tr>
<tr>
<td></td>
<td>S Exponent Mantiss</td>
</tr>
<tr>
<td>Exponent Range for a REAL Number</td>
<td>±308</td>
</tr>
<tr>
<td>Precision of a REAL Variable</td>
<td>15 digits (53 bits)</td>
</tr>
<tr>
<td>DOUBLE PRECISION Bit Representation</td>
<td>0 1 15 16 63</td>
</tr>
<tr>
<td></td>
<td>S Exponent Mantiss</td>
</tr>
<tr>
<td></td>
<td>64 127 Mantissa (continued)</td>
</tr>
<tr>
<td>Exponent Range for a DOUBLE PRECISION Number</td>
<td>±4932</td>
</tr>
<tr>
<td>Precision of a DOUBLE PRECISION Variable</td>
<td>33 digits (112 bits)</td>
</tr>
</tbody>
</table>

Note: IEEE Standard 754 does not define a 128-bit floating point value; the format varies among computer manufacturers.
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER Bit Representation</td>
<td></td>
</tr>
<tr>
<td>REAL Bit Representation</td>
<td></td>
</tr>
<tr>
<td>Exponent Range for a REAL Number</td>
<td>±308</td>
</tr>
<tr>
<td>Precision of a REAL Variable</td>
<td>15 digits (53 bits)</td>
</tr>
<tr>
<td>DOUBLE PRECISION Bit Representation</td>
<td></td>
</tr>
<tr>
<td>Exponent for a DOUBLE PRECISION Number</td>
<td>±4932</td>
</tr>
<tr>
<td>Precision of a DOUBLE PRECISION Variable</td>
<td>33 digits (112 bits)</td>
</tr>
</tbody>
</table>
Computer Dependent Defaults

These tables list the computer-dependent default values for MD Nastran. The default rank values are listed in Table C-17.

Table C-15  Computer-Dependent Defaults, SUPER-UX

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Input File Settings</th>
<th>Command Line Settings</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFPOOL</td>
<td>NASTRAN BUFFPOOL=n</td>
<td>bpool=n</td>
<td>27</td>
<td>GINO Blocks</td>
</tr>
<tr>
<td>BUFFSIZE</td>
<td>NASTRAN BUFFSIZE=n</td>
<td>bufsize=n</td>
<td>8193</td>
<td>Max: 65537</td>
</tr>
<tr>
<td>BUFFSIZE Increment</td>
<td>NASTRAN SYSTEM(136)=n</td>
<td>sys136=n</td>
<td>128</td>
<td>Words</td>
</tr>
<tr>
<td>DBALL Size</td>
<td>INIT DBALL , LOGICAL= DBALL(n)</td>
<td>sdball=n</td>
<td>1000000</td>
<td>GINO Blocks</td>
</tr>
<tr>
<td>DBS Update Time</td>
<td>NASTRAN SYSTEM(128)=n</td>
<td>sys128=n</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Lanczos HPO</td>
<td>NASTRAN SYSTEM(193)=n</td>
<td>sys193=n</td>
<td>1</td>
<td>Recompute</td>
</tr>
<tr>
<td>Lanczos HPO</td>
<td>NASTRAN SYSTEM(194)=n</td>
<td>sys194=n</td>
<td>1</td>
<td>Read/Write</td>
</tr>
<tr>
<td>MAXSET</td>
<td>NASTRAN MAXSET=n</td>
<td>sys263=n</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>SCRA TCH Size</td>
<td>INIT SCRATCH , LOGICAL=(logname(n)), SCR300=(logname(n))</td>
<td>sscr=n</td>
<td>1000000</td>
<td>GINO Blocks</td>
</tr>
<tr>
<td>SMEM</td>
<td>INIT SCRATCH (MEM=n)</td>
<td>smem=n</td>
<td>0</td>
<td>GINO Blocks</td>
</tr>
<tr>
<td>Sparse Ordering Method</td>
<td>NASTRAN SYSTEM(206)=n</td>
<td>sys206=n</td>
<td>0</td>
<td>Choose best reordering</td>
</tr>
</tbody>
</table>

Table C-16  Computer-Dependent Defaults, All Others

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Input File Settings</th>
<th>Command Line Settings</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFPOOL</td>
<td>NASTRAN BUFFPOOL=n</td>
<td>bpool=n</td>
<td>37</td>
<td>GINO Blocks</td>
</tr>
<tr>
<td>BUFFSIZE</td>
<td>NASTRAN BUFFSIZE=n</td>
<td>bufsize=n</td>
<td>8193</td>
<td>Max: 65537</td>
</tr>
<tr>
<td>BUFFSIZE Increment</td>
<td>NASTRAN SYSTEM(136)=n</td>
<td>sys136=n</td>
<td>128</td>
<td>Words</td>
</tr>
<tr>
<td>DBALL Size</td>
<td>INIT DBALL , LOGICAL= DBALL(n)</td>
<td>sdball=n</td>
<td>250000</td>
<td>GINO Blocks</td>
</tr>
<tr>
<td>DBS Update Time</td>
<td>NASTRAN SYSTEM(128)=n</td>
<td>sys128=n</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Lanczos HPO</td>
<td>NASTRAN SYSTEM(193)=n</td>
<td>sys193=n</td>
<td>0</td>
<td>Save</td>
</tr>
</tbody>
</table>
Table C-16  Computer-Dependent Defaults, All Others (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Input File Settings</th>
<th>Command Line Settings</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lanczos HPO</td>
<td>NASTRAN SYSTEM(194)=n</td>
<td>sys194=n</td>
<td>0</td>
<td>Pack/Unpack</td>
</tr>
<tr>
<td>MAXSET</td>
<td>NASTRAN MAXSET=n</td>
<td>sys263=n</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>SCRATCH Size</td>
<td>INIT SCRATCH , LOGICAL=(logname(n)), SCR300=(logname(n))</td>
<td>sscr=n</td>
<td>250000</td>
<td>GINO Blocks</td>
</tr>
<tr>
<td>SMEM</td>
<td>INIT SCRATCH (MEM=n)</td>
<td>smem=n</td>
<td>100</td>
<td>GINO Blocks</td>
</tr>
<tr>
<td>Sparse Ordering</td>
<td>NASTRAN SYSTEM(206)=n</td>
<td>sys206=n</td>
<td>4</td>
<td>Prefer Extreme</td>
</tr>
<tr>
<td>Method</td>
<td></td>
<td></td>
<td></td>
<td>reordering</td>
</tr>
</tbody>
</table>

Table C-17  Computer-Dependent Default Rank Values

<table>
<thead>
<tr>
<th>Computer Type</th>
<th>Model</th>
<th>SYS198</th>
<th>SYS205</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>All</td>
<td>8</td>
<td>24</td>
</tr>
<tr>
<td>HP Alpha UNIX</td>
<td>All</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>HP-UX</td>
<td>All</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>Intel, Linux and Windows (32 bit)</td>
<td>All</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>IRIX64</td>
<td>R8K, R10K, R12K</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>R10K, R12K</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>Linux 64 bit</td>
<td>All</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>Linux x8664</td>
<td>All</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>Solaris</td>
<td>UltraSPARC</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SUPER-UX</td>
<td>All</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>UXP/V</td>
<td>All</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>
Product Timing Data
If User Warning Message 6080 is printed in the .f06 file, please fill out this form and mail it along with machine-readable copies of the gentim2.f04, gentim2.f06, gentim2.log, and gentim2.pch files (see Generating a Timing Block for a New Computer, 56) to MSC.Software at the address below.

Client:  
Site:  
Computer:  
Model:  
Submodel:  
Operating System:  
Operating Level:  

Thank you.

MD Nastran Client Support  
MSC.Software Corporation  
815 Colorado Blvd.  
Los Angeles, CA 90041