CRAY IOS-V Commands Reference Manual

SR–2170 8.0.3.2

Cray Research, Inc.

Portions of this product may still be in development. The existence of those portions still in development is not a commitment of actual release or support by Cray Research, Inc. Cray Research, Inc. assumes no liability for any damages resulting from attempts to use any functionality or documentation not officially released and supported. If it is released, the final form and the time of official release and start of support is at the discretion of Cray Research, Inc.

Requests for copies of Cray Research, Inc. publications should be sent to the following address:

Cray Research, Inc.
Distribution Center
2360 Pilot Knob Road
Mendota Heights, MN 55120
USA

Order desk (612) 683–5907
Fax number (612) 452–0141

Cray Research Software Documentation Map

The illustration on the following pages highlights the major body of documentation available for Cray Research (CRI) customers. The illustration is organized into categories by audience designation:

Audience	Description
End users	Those who use the UNICOS operating system, products, applications, or networking software
Application and system programmers	Those who write or modify system or application code on a CRI system for the purpose of solving computer system, scientific, or engineering problems
System administrators	Those who perform system administration tasks, such as installation, configuration, and basic troubleshooting
System analysts	Those who perform advanced troubleshooting, tuning, and customization
Operators	Those who perform operational functions, such as performing system dumps, and those who administer an operator workstation

To use the map, find the audience designation closest to your specific needs or role as a CRI system user. Note that manuals under other audiences may also be of interest to you; manuals are listed only once, underneath the audience to which they most directly apply. Some manual titles are abbreviated. The date in the map's footer tells you when the information was last revised.

For more information

In addition to the illustration, you can use the following publications to find documentation specific to your needs:

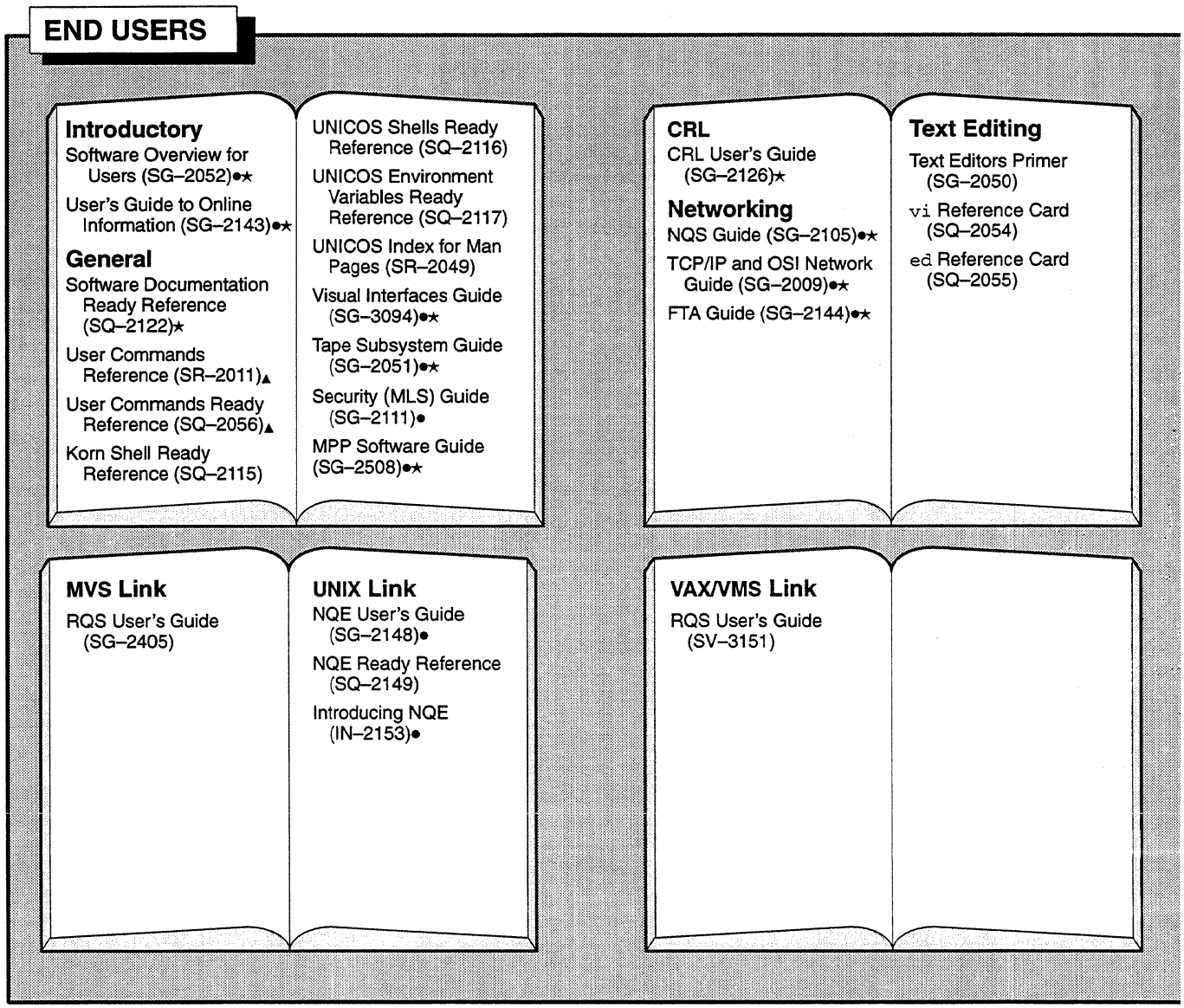
- *Software Documentation Ready Reference,* publication SQ–2122, serves as a general index to the CRI documentation set. The booklet lists documents and man pages according to topic.
- *Software Overview for Users*, publication SG–2052, introduces the UNICOS operating system, its features, and its related products. It directs you to documentation containing user-level information.
- *User Publications Catalog*, publication CP–0099, briefly describes all CRI manuals available to you, including some not shown on the map, such as training workbooks and other supplementary documentation.

Ordering

To obtain CRI publications, order them by publication number from the Distribution Center:

Cray Research, Inc.
Distribution Center
2360 Pilot Knob Road
Mendota Heights, MN 55120
USA

Order desk (612) 683-5907
Fax number (612) 452-0141



- ● Available online with CrayDoc
- ★ Available online with Docview
- ▲ Man pages available with the `man` command

10/94

APPLICATION AND SYSTEM PROGRAMMERS

Ada
Cray Ada Reference (SR–3014)★

Cray Ada Programming Guide (SR–3082)★

C
Cray Standard C Reference (SR–2074)●★

Cray Standard C Ready Reference (SQ–2076)

Cray Standard C for MPP (SR–2506)●★

CAL for CRAY Y-MP and CRAY Y-MP C90
Reference (SR–3108)●

Symbolic Machine Instructions (SR–3109)

Ready Reference (SQ–3110)

UNICOS Macros and Opdefs (SR–2403)▲

Cray Assembler for MPP
CAM Reference (SR–2510)●★

FORTRAN 77
CF77 Ready Reference (SQ–3770)

CF77 Commands and Directives (SG–3771)●★

CF77 Fortran Reference (SR–3772)●★

CF77 Optimization Guide (SG–3773)★

CF77 Message Manual (SR–3774)

Cray MPP Fortran Reference (SR–2504)●★

Fortran 90
CF90 Commands and Directives (SR–3901)●★

CF90 Fortran Language Reference (SR–3902)●★

CF90 Ready Reference (SQ–3900)

Introducing CF90 SPARC Prog. Env. (IN–2155)●

Introducing DPE (IN–2163)●

Libraries
System Libraries (SR–2080)▲

System Libraries Ready Ref. (SQ–2147)▲

Scientific Libraries (SR–2081)▲

Math Library (SR–2138)▲

Application Programmer's I/O Guide (SG–2168)▲

Application Programmer Library Ref. Manual (SG–2165)▲

Introducing CrayLibs (IN–2167)●

PVM and HeNCE Ref. (SR–2501)●★

PVM Reference Card (SQ–2512)

Loaders
Loader Reference (SR–0066)●★

SEGLDR Ready Reference (SQ–0303)

Loader for MPP
Cray MPP Loader Guide (SG–2514)●

Networking
RPC Reference (SR–2089)●★

Kerberos User's Guide (SG–2409)●★

Programming Tools
UNICOS Message System Programmer's Guide (SG–2121)●★

Compiler Information File (CIF) Reference (SR–2401)●★

CDBX Debugger Reference (SR–2091)●★

CDBX Debugger User's Guide (SG–2094)●★

CDBX Reference Card (SQ–2110)

Program Browser (`xbrowse`) (IN–2140)●

Tuning Guide to Parallel Vector Applications (SG–2182)●

MPP Apprentice Tool (IN–2511)●

Introducing Cray TotalView Debugger (IN–2502)●

Simulators
Cray MPP Simulator Guide (SG–2503)●

Source Control
USM User's Guide (SG–2097)●★

System Calls
System Calls (SR–2012)▲

X Window System
Reference (SR–2101)●★

Ready Reference (SQ–2123)

OPERATORS

OWS-E/IOS-E
OWS-E/IOS-E Reference (SR–3077)▲

OWS-E/IOS-E Ready Reference (SQ–3080)

OWS-E/IOS-E Operator's Guide (SG–3078)

OWS-E/IOS-E Administrator's Guide (SG–3079)

- ● Available online with CrayDoc
- ★ Available online with Docview
- ▲ Man pages available with the `man` command

10/94

SYSTEM ADMINISTRATORS AND ANALYSTS

UNICOS

UNICOS Installation Guide (SG–2112)

Installation Ref. Card (SQ–2411)

UNICOS Installation Tool Menus and Help Files (SG–2412)

UNICOS System Administration (SG–2113)•★

Administrator Commands Reference (SR–2022)▲

Administrator Commands Ready Ref. (SQ–2413)▲

CRL

CRL Administrator's Guide (SG–2127)★

DMF

DMF Administrator's Guide (SG–2135)★

Security and Licensing

UNICOS System Security Overview (SG–2141)★

FLEXlm Guide (SG–2181)•★

UNICOS under UNICOS

UuU Administrator's Guide (SG–2156)•★

CRAY EL Series

IOS Commands Reference (SR–2408)▲

IOS Commands Ready Ref. (SQ–2162)

UNICOS Basic Administration Guide (SG–2416)•★

UNICOS Installation Guide for CRAY Y-MP EL Systems (SG–5201)

IOS Messages (SQ–2402)

Networking

fy Driver Administrator's Guide (SG–2132)

MPP

CRAY T3D Administrator's Guide (SG–2507)•

MVS Link

RQS Administrator's Guide (SG–2406)

VAX/VMS Link

RQS Administrator's Guide (SV–3152)

UNIX Link

RQS Administrator's Guide (SG–2120)

NQE Administration (SG–2150)•

NQE Installation (SG–5236)•

Analysts

File Formats and Special Files Reference (SR–2014)▲

Data Migration MSP Writer's Guide (SN–2098)★

UNICOS Tuning Guide (SR–2099)•

UNICOS `nmake` Card (SQ–2146)

Installation and Configuration Tool Reference (SR–3090)

USCP

Front-end Protocol Internals (SM–0042)★

USCP Optimization (SN–2103)

- • Available online with CrayDoc
- ★ Available online with Docview
- ▲ Man pages available with the `man` command

10/9

Record of Revision

The date of printing or software version number is indicated in the footer. Changes in rewrites are noted by revision bars along the margin of the page.

Version	***Description***
8.0.3.2	March 1995. Original printing. UNICOS release 8.0.3.2 and CRAY J90 series IOS-V release 1.3.

Preface

This publication is for system administrators and operators of the CRAY J90 series systems. It contains IOS administrator commands that are specific to the CRAY J90 series IOS-V version 1.3 and UNICOS version 8.0.3.2. It is a helpful reference after the Cray Research UNICOS operating system and the IOS-V software are installed on your system.

Related publications

The following list includes Cray Research publications that are related to the CRAY J90 series, and they are available in the Distribution Center in Mendota Heights, Minnesota:

- *UNICOS Basic Administration Guide for CRAY J90 and CRAY EL Series*, publication SG–2416, contains an appendix that documents the differences between the EL IOS release 11.3.1 and the IOS-V release 1.3.
- *CRAY IOS-V Messages*, publication SQ–2172, contains information on conducting IOS and UNICOS dumps, recovering from a `root (/)` file system crash, and definitions of panic and warning messages.
- *UNICOS Installation Guide for the CRAY J90 Series*, publication SG–5271, contains information on how to install the UNICOS operating system and the most current release of the Cray Research J90 I/O subsystem (IOS-V) diagnostics, how to customize your configuration, and how to recover from a `root (/)` file system crash.
- *UNICOS Administrator Commands Reference Manual*, publication SR–2022, contains detailed information and examples of UNICOS administrator commands.

- *Software Overview for Users*, publication SG–2052, contains a brief introduction to Cray Research system hardware and an overview of the following topics: UNICOS, operating system features, networking and connectivity, program generation utilities and products, programming features, and applications.

The *User Publications Catalog*, publication CP–0099, describes the availability and content of all Cray Research hardware and software manuals that are available to customers.

To order a manual, either call the Distribution Center in Mendota Heights, Minnesota, at (612) 683–5907 or send a facsimile of your request to fax number (612) 452–0141. Cray Research employees may send electronic mail to `orderdsk` (UNIX system users).

Conventions

The following conventions are used throughout this manual:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
`manpage(`*x*`)`	Man page section identifiers appear in parentheses after man page names. The following list describes the identifiers: 1 User commands 1B User commands ported from BSD 2 System calls 3 Library routines, macros, and opdefs 4 Devices (special files) 4P Protocols 5 File formats 7 Miscellaneous topics 7D DWB-related information 8 Administrator commands

Convention	Meaning
`routine()`	Routine names followed by an empty set of parentheses designate a library or kernel routine; for example, `ddcntl()`. Kernel routines do not have man pages associated with them.
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.
[ ]	Brackets enclose optional portions of a command line.
...	Ellipses indicate that a preceding command-line element can be repeated.
`<KEY>`	On man pages, this convention indicates a key on the keyboard.

The following machine naming conventions may be used throughout this manual:

Term	Definition
Cray PVP systems	All configurations of Cray parallel vector processing (PVP) systems, including the following: CRAY C90 series (CRAY C916, CRAY C92A, CRAY C94, CRAY C94A, and CRAY C98 systems) CRAY C90D series (CRAY C92AD, CRAY C94D, and CRAY C98D systems) CRAY EL series (CRAY Y-MP EL, CRAY EL92, CRAY EL94, and CRAY EL98 systems) CRAY J90 series (CRAY J916 system) CRAY T90 series (CRAY T94, CRAY T916, and CRAY T932 systems) CRAY Y-MP series (CRAY Y-MP2, CRAY Y-MP4, and CRAY Y-MP8 systems)

Term	Definition
	CRAY Y-MP E series (CRAY Y-MP 2E, CRAY Y-MP 4E, CRAY Y-MP 8E, and CRAY Y-MP 8I systems)
	CRAY Y-MP M90 series (CRAY Y-MP M92, CRAY Y-MP M94, and CRAY Y-MP M98 systems)
Cray MPP systems	All configurations of Cray massively parallel processing (MPP) systems, including the CRAY T3D series (CRAY T3D MC, CRAY T3D MCA, and CRAY T3D SC systems)
All Cray Research systems	All configurations of Cray PVP and Cray MPP systems that support this release, except Cray Research Superservers, Inc. (CRS) systems
SPARC systems, including those from CRS	All SPARC platforms, including systems offered by Cray Research Superservers, Inc., that run the Solaris operating system version 2.3 or later

It is the objective of Cray Research to become compliant with IEEE Std 1003.1–1990 (POSIX.1) and IEEE Std 1003.2–1992 (POSIX.2). This manual reflects those ongoing efforts.

POSIX.2 uses *utility* to refer to executable programs that Cray Research documentation usually refers to as *commands*. Both terms appear in this document.

In this publication, *Cray Research*, *Cray*, and *CRI* refer to Cray Research, Inc. and/or its products.

Man page sections

The entries in this manual are based on a common format. The following list shows the order of sections in an entry and describes each section. Most entries contain only a subset of these sections.

Section heading	Description
NAME	Specifies the name of the entry and briefly states its function.
SYNOPSIS	Presents the syntax of the entry.
IMPLEMENTATION	Identifies the Cray Research systems to which the entry applies.
STANDARDS	Provides information about the portability of a utility or routine.
DESCRIPTION	Discusses the entry in detail.
NOTES	Presents items of particular importance.
CAUTIONS	Describes actions that can destroy data or produce undesired results.
WARNINGS	Describes actions that can harm people, equipment, or system software.
ENVIRONMENT VARIABLES	Describes predefined shell variables that determine some characteristics of the shell or that affect the behavior of some programs, commands, or utilities.
RETURN VALUES	Describes possible return values that indicate a library or system call executed successfully, or identifies the error condition under which it failed.
EXIT STATUS	Describes possible exit status values that indicate whether the command or utility executed successfully.
MESSAGES	Describes informational, diagnostic, and error messages that may appear. Self-explanatory messages are not listed.
FORTRAN EXTENSIONS	Describes how to call a system call from Fortran. Applies only to system calls.
BUGS	Indicates known bugs and deficiencies.

Section heading	Description
EXAMPLES	Shows examples of usage.
FILES	Lists files that are either part of the entry or are related to it.
SEE ALSO	Lists entries and publications that contain related information.

Online information

The following types of online information products are available to Cray Research customers:

- CrayDoc online documentation reader, which lets you see the text and graphics of a manual online. The CrayDoc reader is available on workstations. To start the CrayDoc reader at your workstation, use the `cdoc`(1) command.
- Docview text-viewer system, which lets you see the text of a manual online. The Docview system is available on the Cray Research mainframe. To start the Docview system, use the `docview`(1) command.
- Man pages, which describe a particular element of the UNICOS operating system or a compatible product. To see a detailed description of a particular command or routine, use the `man`(1) command.
- UNICOS message system, which provides explanations of error messages. To see an explanation of a message, use the `explain`(1) command.
- Cray Research online glossary, which explains the terms used in a manual. To get a definition, use the `define`(1) command.
- `xhelp` help facility. This online help system is available within tools such as the Program Browser (`xbrowse`) and the MPP Apprentice tool.

For detailed information on these topics, see the *User's Guide to Online Information*, publication SG–2143.

Reader comments

If you have comments about the technical accuracy, content, or organization of this manual, please tell us. You can contact us in any of the following ways:

- Send us electronic mail from a UNICOS or UNIX system, using the following UUCP address:

  `uunet!cray!publications`

- Send us electronic mail from any system connected to Internet, using the following Internet addresses:

  `pubs2170@timbuk.cray.com` (comments on this manual)

  `publications@timbuk.cray.com` (general comments)

- Contact your Cray Research representative and ask that a Software Problem Report (SPR) be filed. Use `PUBLICATIONS` for the group name, `PUBS` for the command, and `NO-LICENSE` for the release name.

- Call our Software Information Services department in Eagan, Minnesota, through the Technical Support Center, using either of the following numbers:

  (800) 950–2729 (toll free from the United States and Canada)

  (612) 683–5600

- Send a facsimile of your comments to the attention of "Software Information Services" in Eagan, Minnesota, at fax number (612) 683–5599.

- Use the postage-paid Reader's Comment Form at the back of the printed manual.

We value your comments and will respond to them promptly.

CONTENTS

NAME

am – Alters memory

SYNOPSIS

am *address* [*parcelA*] [*parcelB*] [*parcelC*] [*parcelD*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The am command alters the contents of a 64-bit word in central memory by using the I/O channel. It accepts the following arguments:

address Relative memory address that will be altered.

parcelA Value of parcel to alter memory (most significant); default is no change.

parcelB Value of parcel to alter memory; default is no change.

parcelC Value of parcel to alter memory; default is no change.

parcelD Value of parcel to alter memory (least significant); default is no change.

NOTES

This command accesses central memory through the data channels; therefore, the CPU clock must be on.

MESSAGES

`Expected central memory address`
The first argument specified is not a valid central memory address.

`Invalid parcel` *parcel#*
The parcel value to alter memory to is not valid.

`Open of memory failed`
The open of mainframe central memory fails.

`Unable to read memory`
The read of mainframe central memory fails.

`Write to memory failed`
Writing the parcel values to memory fails.

EXAMPLES

The following command writes the value 1111 2222 3333 4444 to central memory word 1000 hexadecimal:

```
am 1000 1111 2222 3333 4444
```

SEE ALSO

dm(8) to display central memory
fm(8) to fill memory
lm(8) to load central memory

NAME

bb1test – Executes diagnostic test for I/O buffer board

SYNOPSIS

```
bb1test
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The bb1test command lets users run and control an offline I/O buffer board (IOBB) diagnostic.

To test the IOBB thoroughly, you should run this test before running the bb2test command.

WARNINGS

The bb1test command has the following limitations:

- When the operating system is running, you cannot use bb1test because a UNICOS system panic will occur, which can corrupt data.
- The bb1test command runs only on CRAY J90 series systems.
- The bb1test command does not run from the background.

SEE ALSO

bb2test(8) to execute an IOBB<-> disk test

UNICOS Administrator Commands Reference Manual, publication SR–2022, for additional UNICOS diagnostic commands

CRAY J90 Series IOS Based Tests, publication HDM–099–0 (This manual is Cray Research Proprietary; dissemination of this documentation to non-CRI personnel requires approval from the appropriate vice president and a nondisclosure agreement. Export of technical information in this category may require a Letter of Assurance.)

NAME

`bb2test` – Executes a disk I/O to and from I/O buffer board test

SYNOPSIS

```
bb2test
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `bb2test` command lets users run and control an offline I/O buffer board (IOBB) test.

To test the IOBB thoroughly, you should run the `bb1test` command.

NOTES

When running the `bb2test` command, observe the the following limitations:

- When the operating system is active, the `bb2test` command does not run.
- The `bb2test` command does not run from the boot prompt.
- The `bb2test` command does not run from the background.

SEE ALSO

`bb1test`(8) to execute a diagnostic test for IOBB

CRAY J90 Series IOS Based Tests, publication HDM–099–0 (This manual is Cray Research Proprietary; dissemination of this documentation to non-CRI personnel requires approval from the appropriate vice president and a nondisclosure agreement. Export of technical information in this category may require a Letter of Assurance.)

NAME

bg – Puts a suspended IOS command into the background

SYNOPSIS

```
bg
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The bg command starts a suspended IOS command and allows the interactive shell to run parallel with it.

EXAMPLES

If the dformat command was started in the foreground (for example, the & character was not placed at the end of the command line), and then the user entered <CONTROL-Z> to suspend it, the dformat command can resume execution in the background by entering the following command:

```
bg
```

NAME

`bootstruct` – Displays the boot environment of the IOS

SYNOPSIS

```
bootstruct
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `bootstruct` command displays the boot environment of the IOS including its network identity and that of its console server.

NAME

`cat` – Displays file

SYNOPSIS

`cat [-n]` *filename*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `cat` command displays the contents of a file on the system console disk, which is NFS mounted to the IOS. It accepts the following options:

`-n` Displays a line number with each line and first byte of that line's byte count.

filename Specifies input file.

MESSAGES

`Can't find` *<filename>*
The file name specified does not exist (cannot be opened).

NAME

`cc1test` – Executes diagnostic test for I/O buffer board and I/O channel control chip

SYNOPSIS

```
cc1test
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `cc1test` command lets users run and control an offline I/O buffer board and I/O channel control chip (IOBB and CC chip) diagnostic.

To thoroughly test the IOBB, you should run this test after the `bb1test` and `bb2test` commands.

NOTES

The `cc1test` command does not run from the background.

WARNINGS

When the operating system is running, you cannot use the `cc1test` command because a UNICOS system panic will occur, which can corrupt data.

SEE ALSO

UNICOS Administrator Commands Reference Manual, publication SR–2022, for additional UNICOS diagnostic commands

CRAY J90 Series IOS Based Tests, publication HDM–099–0 (This manual is Cray Research Proprietary; dissemination of this documentation to non-CRI personnel requires approval from the appropriate vice president and a nondisclosure agreement. Export of technical information in this category may require a Letter of Assurance.)

NAME

`cc2test` – Executes a data transfer test from central memory to I/O buffer board to central memory and verifies data

SYNOPSIS

```
cc2test
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `cc2test` command executes a data transfer test from central memory to the I/O buffer board (IOBB) and back to central memory. Test initialization includes loading and deadstarting the CPU binary file. Section initialization verifies that the I/O channels selected are connected to the IOBB being tested.

Each data pattern (85 total) is passed to the CPU program, which vector stores that pattern to central memory.

The length of the data buffer when using a 16-Mbyte IOBB is 2,096,896 D CM words. The length of the data buffer when using a 4-Mbyte IOBB is 524,032 D CM words.

The IOBB has the data pattern written to or read from location 0x200 to maximum.

After each pattern is stored to central memory, it is transferred to the IOBB by using the output command channel. After the transfer, the channel error register and IOBB status register are tested for error information.

After each pattern is transferred to the IOBB, it is transferred back to central memory by using the input command channel. After the transfer, the channel error register and IOBB status register are tested for error information.

The data in central memory is then verified by doing vector subtracts of the write buffer data from the read buffer data. A similar sequence is used to transfer address data from central memory to IOBB to central memory.

NOTES

The `cc2test` command runs only on CRAY J90 series systems.

When the operating system is active, the `cc2test` command cannot run; it must be run from the IOS prompt.

SEE ALSO

CRAY J90 Series IOS Based Tests, publication HDM–099–0 (This manual is Cray Research Proprietary; dissemination of this documentation to non-CRI personnel requires approval from the appropriate vice president and a nondisclosure agreement. Export of technical information in this category may require a Letter of Assurance.)

NAME

cd – Changes current directory

SYNOPSIS

cd *path*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The cd command changes the current directory on the system console disk, which is NFS mounted to the IOS. It accepts the following argument:

path Absolute or relative path name of the desired directory.

MESSAGES

<directory name> is not a directory
: The directory being changed to is not a directory.

Error = *<errno>*
: A VxWorks system call failed. The *errno* printed is an internal error number for debugging purposes.

No such directory!
: The directory being changed to does not exist (cannot be opened).

EXAMPLES

Example 1: The following command changes the current directory to the root directory:

```
cd /
```

Example 2: The following command changes the current directory to the boot subdirectory:

```
cd boot
```

Example 3: The following command changes the current directory to the test directory from any other directory:

```
cd /test
```

NAME

`clearlog` – Clears the statistical log data on an STK 4280 tape drive

SYNOPSIS

`clearlog rss`*CUL*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `clearlog` command clears the statistical log data on an STK 4280 tape drive.

The `clearlog` command accepts the following options:

C Specifies controller number.

U Specifies the unit number.

L Specifies the logical unit (LUN).

MESSAGES

`Cannot open tape` *<tape device>*
: An open of the specified tape device fails. This can happen if the tape device was not initialized properly during the IOS load.

`Invalid tape device name`
: The tape device specified is not of type STK 4280. Valid tape device names begin with `rss`.

`Unable to execute log command`

EXAMPLES

The following command clears the log data on drive `rss010`:

```
clearlog rss010
```

NAME

`cls` – Clears the screen display

SYNOPSIS

```
cls
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `cls` command clears the screen. It is useful in a command script for clearing data on the screen.

NAME

cmp – Performs a byte-by-byte comparison of two files

SYNOPSIS

cmp [-l] [-s] *filename1 filename2* [*skip1*] [*skip2*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The cmp command compares *filename1* and *filename2*. If you omit options, cmp makes no comment if the files are the same; if they differ, it reports the byte and line number at which the difference occurred, or, that one file is subordinate to the other.

The *skip1* and *skip2* arguments are initial byte offsets into *filename1* and *filename2*, respectively. They can be either octal or decimal; a leading 0 denotes octal.

The cmp command accepts the following option and arguments:

-l	Prints the byte number (in decimal) and the differing bytes (in octal) for all differences between the two files.
-s	Silent. Prints nothing for differing files; sets only exit codes.
filename	Specifies the name of the file(s) to be compared.
skip	Specifies at which byte the comparison should begin.

EXIT STATUS

The exit status is 0 for identical files, 1 for different files, and 2 if an error occurred.

MESSAGES

EOF on file
: The end-of-file mark is reached.

Open of file <*file name*> failed
: One of the file names specified for comparison does not exist (cannot be opened).

NAME

`conswitch` – Toggles console from IOS to UNICOS system console

SYNOPSIS

```
conswitch
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

You can execute the `conswitch` command only from the IOS. It is the equivalent of pressing `<CONTROL-a>` to toggle the console terminal from acting as the IOS console to the UNICOS console interface. This command is used in scripts to automate the transfer of control from the IOS to UNICOS.

NOTES

This command executes only in a command script file.

NAME

`count` – Counts the number of passes that a loop executes

SYNOPSIS

`count` *init*

`count` *inc*

`count` *print*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `count` command enables a counter that counts the number of passes that were executed when a loop is used.

The `count` command accepts the following arguments:

init Initializes the counter to 0.

inc Increments the counter by 1.

print Prints the current value of the counter.

NOTES

This command executes only in a command script file.

MESSAGES

`Bad argument`
The argument specified is not a valid option.

EXAMPLES

The following command line displays the count (in decimal) on the terminal screen:

```
count print
```

NAME

cp – Copies a file

SYNOPSIS

cp *source destination*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The cp command copies the contents of *filespec1* into *filespec2*.

The cp command accepts the following arguments:

source File specification of the source file.

destination File specification of the destination file.

CAUTIONS

If destination files already exist, they are overwritten.

MESSAGES

`Source file is a directory`
The source file specified to copy is an existing directory.

`Unable to open destination file`
The destination file cannot be opened.

`Unable to open source file`
The source file specified cannot be opened.

`Unable to stat source file`
The source file specified does not exist.

`Write failed on <`*filename*`> ... aborting`
An error occurred while writing the data from the source file to the new destination file.

EXAMPLES

Example 1: The following example copies contents of `test1` into a new file named `test2`:

```
cp test1 test2
```

Example 2: The following example copies file `file1` from directory `/tmp/type` to directory `/adm/type`:

```
cp /tmp/type/file1 /adm/type/file1
```

NAME

crash – Interprets IOS system dumps

SYNOPSIS

crash *filename*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The crash command reads in the IOS image created by the iosdump command (*filename*) and displays system structures, raw memory, and symbolic information, prompting users for commands.

The iosdump command creates the dump file in a format that crash expects. By default, it is kept in the /adm/dump*x*/D*date* file name convention; *x* is the IOS number on which the dump originated, and *date* is the day, month, and year the dump occurred.

To exit from the crash utility, type q and press <RETURN>.

The crash command accepts the following argument:

filename Specifies the name of the file that contains the IOS system image.

All addresses are expected to be in hexadecimal format regardless of prefixes. crash automatically determines whether the address given is an I/O buffer board (IOBB) address or IOS memory address and converts the address to a file offset accordingly. crash also recognizes a number that is simply an offset from IOS memory position 0.

The crash command accepts the following subcommands:

dc5i [-f]
: Displays the state of the DC5I controller (crash prompts you for the controller number if you do not specify it on the command line) and a history of the I/O parameter blocks that the driver has prepared for the controller in reverse chronological order.

dstage
: Displays staged disk requests.

dstat Displays the overall disk strategy numbers and the disk devices that were found on that IOS.

errpt Displays the error report.

help [*command*]
: Outputs one help line that specifies the syntax of all available dump commands.

ipi [-f]
: Displays the state of the IPI controller (crash prompts you for it if you did not specify it on the command line), along with any active I/O parameter blocks. By default, an 80-column display is generated. To display additional parameter block information, specify the -f option.

jobs Shows the last 16 user commands run (or running at time of dump) on the IOS, along with their arguments and state.

loadmap
: Lists each strategy, driver, or command that has been loaded, along with its load address and size.

`nm [-l] [*][`*symbol*`|`*address*`|driver|config|uconfig]`
Namelist command. If given a symbol or address, `nm` searches the namelist for the matching address and/or symbol pair and displays it. If preceded by a `*`, `nm` will output all symbols in the namelist that match the specified symbol pattern (for example, the `nm *pkt` command would output all symbol names that have the string `pkt`, such as `_pkt_tbl` and `_getpkt`).

The words `driver`, `config`, and `uconfig` are keywords, and they list the drivers loaded at the time of the dump, the `/config` file contents, and the UNICOS `config` file, respectively (if master IOS).

`od [-`*line count*`] -[h|o|d] [`*addr*`]`
Lists the contents of IOS memory at the *addr* specified according to the base specified (hexadecimal by default).

`packet [`*type*`] [`*addr*`]`
The *type* argument is a letter (`A`, `D`, `M`, and so on) that denotes the type of packet to be displayed, and *addr* specifies the hexadecimal address of one packet. If you specify *type* and *addr* together, `crash` tries to display the information at *addr* as a packet of type *type* if possible. If you specify only *type*, `crash` outputs only the packets found of that type. A history of the last 5120 packets are kept in the IOS; to display it, specify `packet` without arguments.

`q` Exits the `crash` command.

`s2tape`
Outputs status information on each tape attached to the small computer system interface (SCSI) adapter.

`sdisk` Outputs status information on each disk attached to the SCSI adapter.

`si2` Outputs status information on the SCSI-2 adapter.

`stape` Displays the state of each SCSI tape command issued and the device to which it was issued.

`status`
Outputs the release level of the IOS contained in the dump and the time the PANIC occurred if the dump was the result of an `IOS ASSERTION` PANIC.

`sysbuf`
Outputs the last `syslog()` messages sent to the console.

`systat`
Displays the state of the IOBB buffer pool and IOBB transfer queue at the time of the dump.

`table [-a] [-f] {pkt|fd|buf|trace}`
Displays the packet table (`pkt`) contained in IOS memory (not IOBB memory), the file descriptor table (`fd`), the IOBB buffer pool table (`buf`) (see `systat` above), or the trace table (`trace`), respectively. The trace table is defined only after an `IOS ASSERTION` PANIC.

`tcb [`*addr*`]`
The *addr* argument is the IOBB address to start listing I/O transfer control blocks (IOTCB). By default, this command starts at the beginning of the IOTCB table and outputs each control block. Each control block contains the information that the IOBB requires to complete one transfer to or from the mainframe.

`tstat` Provides a trace of tape packets from UNICOS.

`ttybuf`
Displays the tty buffer (any print statements that were queued asynchronously; that is, from an interrupt service routine (ISR) and had not been printed to the console yet).

`ver` Prints the IOS version of the IOS kernel contained in the dump.

EXAMPLES

The following example provides `crash` with a dump taken from IOS 0 (indicated by the directory in which it is found; `dump0` is IOS 0, `dump1` is IOS 1, and so on) from August 11, 1993 (indicated by the name: `D81193.0`):

```
crash  /adm/dump0/D81193.0
```

NAME

`dd5itest` – Executes a confidence test for DD-5I disk drives and controller

SYNOPSIS

```
dd5itest
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `dd5itest` test initialization reserves I/O buffer board (IOBB) write and read blocks, which are released back to the system when the test is exited.

The test consists of the following:

1. Write, read, and verify 85 canned and 100 random data patterns, using 4-Kbyte blocks and test cylinder, head group 0.
2. Verify correct head group selection.
3. Write, read, and verify 85 canned and 20 random data patterns, using 4-Kbyte blocks and test cylinder, sequential heads groups.
4. Write, read, and verify 85 canned and 20 random data patterns, using 128-Kbyte blocks and test cylinder, head group 0.
5. Generate 2048 legal random disk blocks (addresses), using random data generated from a seed number. The disk blocks (addresses) can range from cylinder 0, head group 0, to the beginning of the maintenance cylinder. All reads consists of 4-Kbytes, and read data is not verified.

NOTES

When running `dd5itest`, observe the following limitations:

- The `dd5itest` command does not run from the background.
- The `dd5itest` command does not run from the boot prompt.

SEE ALSO

`dd5iql`(8) to execute a quick-look buffered intelligent peripheral interface (IPI) drive diagnostic

CRAY J90 Series IOS Based Tests, publication HDM–099–0 (This manual is Cray Research Proprietary; dissemination of this documentation to non-CRI personnel requires approval from the appropriate vice president and a nondisclosure agreement. Export of technical information in this category may require a Letter of Assurance.)

NAME

`dd5stest` – Executes a controller comprehensive test and disk confidence test on any CRAY J90 supported SCSI disk(s)

SYNOPSIS

```
dd5stest
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

Test initialization of the `dd5stest` diagnostic includes reserving I/O buffer board (IOBB) write and read blocks, which are released back to the system when the test is exited.

The `dd5stest` diagnostic runs on board (4220) diagnostics. This can be done only if the controller is not operating. If the operating system is active, this test does not run. If the operating system is inactive, an `ioctl` call is made to the controller to run onboard diagnostics. These diagnostics are more extensive than the power-up self-test diagnostics.

During the diagnostic run time, there is no communication to the display. At the end of 20 seconds, the driver returns a run diagnostic time-out error. If a time-out error has not occurred (diagnostics are complete), a message will be displayed specifying whether all diagnostics have run with or without error.

The disk confidence portion of the test consists of the following:

1. Write, read, and verify 85 canned and 100 random data patterns, using 4-Kbyte blocks and maintenance cylinder, head 0.
2. Verify correct head selection.
3. Write, read, and verify 85 canned and 20 random data patterns, using 4-Kbyte blocks and maintenance cylinder, sequential heads.
4. Write, read, and verify 85 canned and 20 random data patterns, using 128-Kbyte blocks and maintenance cylinder, head 0
5. Generate 1024 legal random disk blocks (addresses), using random data generated from a seed number. The disk blocks (addresses) can be from cylinder 0, head 0, sector 0 to the beginning of the maintenance cylinder. All reads consist of 1 sector and read data is not verified.

NOTES

When running `dd5stest`, observe the following limitations.

- When the operating system is active, `dd5stest` runs the disk testing portion of the diagnostics. It does not run the onboard controller diagnostics.
- The `dd5stest` diagnostic does not run from the boot prompt.
- The `dd5stest` diagnostic runs only on CRAY J90 series systems.

SEE ALSO

CRAY J90 Series IOS Based Tests, publication HDM–099–0 (This manual is Cray Research Proprietary; dissemination of this documentation to non-CRI personnel requires approval from the appropriate vice president and a nondisclosure agreement. Export of technical information in this category may require a Letter of Assurance.)

NAME

debug – Reports and sets the debug level on the IOS

SYNOPSIS

debug [*value*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The debug command is useful for setting various debug bit flags for message printing. If you omit *value*, debug reports the current debug value.

The debug command accepts the following argument:

value Sets the debug bit flags. The following debug bit flags are defined:

```
   0x1   UNICOS pkts                  0x400   Tape
   0x2   Console                      0x800   FDDI
   0x4   HYPERchannel                0x1000   HIPPI
   0x8   Ethernet                    0x2000   unpacket driver
  0x10   exdf driver                 0x4000   (reserved)
  0x40   SCSI commands               0x8000   (reserved)
  0x80   (reserved)                 0x10000   SI2 adapter
 0x200   Disk

0x8000   0000   General Information
```

WARNINGS

If you must use this command, you should use it in single-user mode on a relatively idle system.

Setting debug bit flags while running UNICOS can cause an extremely large number of debug messages. A large volume of output causes the IOS to panic.

For more information, consult with your system support staff.

NAME

`dflawr` – Reads Disk Flaw table

SYNOPSIS

`dflawr` b*cd* [`-l`] [`-f` *file*]

`dflawr` i*cd* [`-lr`] [`-s` *serial number*] [`-f` *file*]

`dflawr` s*cd* [`-l`] [`-f` *file*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `dflawr` command reads the Disk Flaw tables from the disk. The tables can be written to a file or displayed on the screen. The file is in the format that `dflaww` expects. If you omit the file name, the file is written automatically to the `/flaw` directory, and it is named according to the device it read and the IOS on which the device resides.

The `dflawr` command accepts the following options:

`b` Indicates a buffered intelligent peripheral interface (IPI) drive.

`i` Indicates an IPI drive.

`s` Indicates a Small Computer System Interface (SCSI) drive.

c Specifies controller number (0 to F).

d Specifies disk (0 to F).

`-l` Lists the tables to only the screen.

`-f` *file* Writes the defect lists to *file*.

`-r` Reads the sector headers on the disk to obtain the Growth Error table (GET). This option is useful to validate the correctness of the stored defect lists, or used to generate a defect list in which the stored list may have been corrupted. The DD-5I and DD-5S drive types do not support this option.

`-s` *serial number*
Specifies serial number of the drive.

NOTES

This command allows back-up capability of disk default information.

MESSAGES

`B%d%d: bogus cylinder found on this drive (%d) - bad OEM defect list!`
The cylinder read that should contain the OEM list is bad.

`Cannot get memory for defect list entry`
The program cannot allocate memory for the defect list.

`Cannot get memory for OEM defect list`
The program cannot allocate memory for the OEM defect list.

`Cannot get memory for sector IDs`
The program cannot allocate memory for the sector IDs.

`dflawr: Cannot get configuration for drive %s`
The program cannot return configuration information.

`dflawr: controller/device (%s) not found`
Device does not exist in disk table kept by the driver.

`dflawr: GET defect list not found on disk : Read of sector IDs fails0);`
A valid GET does not exist on this drive. The program cannot read the sector IDs (for the `-r` option).

`dflawr: GET defect list not found on disk : The -s option is required when no GET is found`
A valid GET does not exist on this drive. The user must enter a serial number on the command line.

`dflawr: GET not supported for device type DD_U`
The GET is not a supported feature for DD_U type drives.

`dflawr: open on drive %s failed : %s`
Open failed for listed device.

`dflawr: open on drive %s failed : %s`
Attempt to reopen drive has failed.

`dflawr: SCSI and Buffered IPI devices do not support the '-r' option`
You cannot use the `-r` option with DD-5I and DD-XS type drives.

`dflawr: unable to log the following line: %s`
A write to the `dflawr` output file failed.

`dflawr: Unknown device '%s'`
Device name entered on command line is not a valid name.

`dflawr: User supplied serial number is ignored - GET is used`
Because a valid GET with a serial number already exists, the user-entered value is ignored.

`dflawr: WARNING - OEM defect list not found`
The program cannot read the OEM defect list.

`Flaw map NOT found on cyl %d`
The flaw map cannot be read from the given cylinder.

`Flaw map NOT found on last cylinder.`
The flaw map cannot be read from the flaw cylinder.

`getdefect: unable to alloc space for GET; Aborting!`
The program cannot allocate memory for the GET.

`get_ipidef: initialize drive to read defects failed`
An attempt to initialize the defect cylinder failed.

`Growth Error Table stored on disk is invalid.`
The GET on the drive is not valid.

`iget_oemdefect: unable to uc_malloc enough space for zone table`
The program cannot allocate memory for the zone table.

`iget_physdefect: %c%d%d: cylinder %d has corrupt header`
The header for this cylinder is corrupt.

`Invalid flaw map on cyl %d, head %d`
The flaw data for the given location is not valid.

`IOCTL failed for GET_CONFIG on disk`
A call to the driver to get the configuration of the device fails.

`IOCTL failed for GET_CONFIG on IPI disk`
A call to the driver to get the configuration of the device fails.

`IOCTL failed for GET_ECONFIG on disk`
A call to the driver to get the configuration of the device fails.

`Read of sector ID %d fails`
The program cannot read the track IDs at the given sector address.

`Read of sector ID cylinder %d, track %d fails`
An attempt to read the sector ID of the given location failed.

`read_get: unable to alloc space for GET`
The program cannot allocate memory for the GET.

`read_get: unable to alloc space for tmp GET`
The program cannot allocate memory for a local copy of the GET.

`Trying cylinder %d.`
Trying alternative location.

`Unable to read GET defect list: %s`
The program cannot read the GET.

`Write to %s failed: %s`
A write to the given file failed.

EXAMPLES

Example 1: The following command reads the Disk Flaw table for disk array controller 0, disk 0. The flaw table is stored in `/flaw/ios.2/s00.flw`, which indicates the array is on IOS2.

```
dflawr s00
```

Example 2: The following command reads the Disk Flaw table from controller 0, disk 1, and it stores the data in `/flaw/ios.0/b01.flw`, indicating that the drive is on IOS0:

```
dflawr b01
```

SEE ALSO

`dflaww`(8) to read Disk Flaw table from IOS disk and write it to disk

NAME

`dflaww` – Reads Disk Flaw table from IOS disk and writes it to disk

SYNOPSIS

`dflaww b`*cd* [`-f` *file*]

`dflaww i`*cd* [`-f` *file*]

`dflaww s`*cd* [`-f` *file*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `dflaww` command reads the manufacturer's flaw table (off the IOS disk) from the specified file and writes it onto the system disk. The file is in the format that `dflaww` expects and is created by the `dflawr` command. Generally, these files are kept in the `/flaw` directory of the IOS disk and are named according to the device they reference and the IOS on which the disk resides.

The `dflaww` command accepts the following options:

`b` Indicates a buffered intelligent peripheral interface (IPI) drive.

`i` Indicates the IPI drive.

`s` Indicates an small computer system interface (SCSI) drive.

c Specifies controller number (0 to F).

d Specifies disk or bank number (0 to F).

`-f` *file* Specifies the file that `dflaww` reads.

MESSAGES

`dflaww: Cannot get configuration for drive %s`
A call to the driver to get the configuration of the device fails.

`dflaww: Cannot get memory for GET`
The program cannot allocate memory for the local copy of the GET.

`dflaww: Cannot get memory for GET defect list`
The program cannot allocate memory for the Growth Error Table (GET) defect list.

`dflaww: Cannot get memory for OEM defect list`
The program cannot allocate memory for the OEM defect list.

`dflaww: can't access file %s`
A call to determine whether the file can be accessed failed.

`dflaww: controller/device (%s) not found`
Device does not exist in disk table kept by the driver.

`dflaww: Device %s does not exist`
Device does not exist in disk table kept by the driver.

`dflaww: Does not support device type DD_U`
The `dflaww` command does not support drives of type DD_U.

`dflaww: open on drive %s failed`
The program cannot open the disk name given.

`dflaww: unable to open %s`
The program cannot open the specified file.

`dflaww: Unable to write GET defect list: %s`
A call to write the GET to the disk failed.

`Error reading flaw map %s.`
`Flaws exceed maximum of %d.`
The flaw count exceeds the maximum number of flaws allowed.

`0x%x flaws added to the GET. %s.`
A count of the flaws added to the GET is given.

`Unable to open file '%s'`
The program cannot open the specified file.

EXAMPLES

The following command reads the flaw table for disk array controller 0 and drive 2 on IOS 0 from the name `/flaw/ios.0/my.flw`:

```
dflaww  s02 -f /flaw/ios.0/my.flw
```

SEE ALSO

`dflawr`(8) to read Disk Flaw table

NAME

`dformat` – Formats disk

SYNOPSIS

`dformat b`*cd* [`-l` *level*] [`-f` *file*]

`dformat i`*cd* [`-l` *level*] [`-s` *serial number*] [`-f` *file*]

`dformat s`*cd* [`-l` *level*] [`-f` *file*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `dformat` command formats the specified disk(s) at the level requested.

The `dformat` command accepts the following options:

`b` Indicates a buffered intelligent peripheral interface (IPI) drive.

`i` Indicates an IPI drive.

`s` Indicates an Small Computer System Interface (SCSI) drive.

c Specifies controller number (0 to F).

d Specifies disk (0 to F).

`-l` *level* Specifies the format level used. If you specify level 1, `dformat` does not map or slip any defects (`dformat` writes an empty Growth Error table (GET)). If you specify level 2, `dformat` slips or maps only the OEM defect list. If you specify level 3, `dformat` slips or maps all entries currently stored in the GET (this is the default).

`-f` *file* Specifies the file to be used for generating the GET. All GET entries in the file will be slipped or mapped.

`-s` *serial number*
Specifies the drive's serial number.

NOTES

The serial number is read automatically off both the DD-5I and DD-5S disk drives.

MESSAGES

`B%d%d: bogus cylinder found on this drive (%d) - bad OEM defect list!`
The cylinder read that should contain the OEM list is bad.

`Cannot get memory for defect list entry`
The program cannot allocate memory for the defect list.

`Cannot get memory for OEM defect list`
The program cannot allocate memory for the OEM defect list.

`dformat: A serial number is required for a level-2 format`
The user must enter a serial number on the command line for a level-2 format.

`dformat: Cannot get configuration for drive %s`
The program cannot get drive configuration information from the driver.

`dformat: Cannot get configuration for drive %s`
Drive has been formatted, but GET has not been written.

`dformat: Cannot read defect list from file %s`
The program cannot read the defect list from the given file.

`dformat: Cannot read GET defect list from drive`
On a level-3 format, the GET on the drive is not valid.

`dformat: Cannot read OEM defect list from drive`
The program cannot read the OEM defect list.

`dformat: controller/device (%s) not found`
Device does not exist in disk table kept by the driver.

`dformat: %d sector %s slipped on %s`
The program has slipped the given number of sectors.

`dformat: No GET defect list on drive`
The GET on the drive is not valid.

`dformat: open on drive %s failed`
A call to reopen the drive failed.

`dformat: open on drive %s failed`
The program cannot open the drive specified.

`dformat: slip failed for sector %d: %s`
The program cannot slip the given sector.

`dformat: Slipping bad sectors on %s`
The program is slipping the flawed sectors.

`dformat: The serial number stored on the disk (%s) does NOT match the serial number in the defect file (%s)`
Self explanatory.

`dformat: unable to uc_malloc space for GET; Aborting!`
The program cannot allocate memory for the GET.

`dformat: Unable to write GET defect list: %s`
A write call to write the GET failed.

`Error reading flaw map %s.`
`Flaws exceed maximum of %d.`
The flaw count exceeds the maximum number of flaws allowed.

`Flaw map NOT found on alternate cylinder`
The flaw map cannot be read from the given cylinder.

`Flaw map NOT found on last cylinder.`
The flaw map cannot be read from the flaw cylinder.

`getdefect: unable to alloc space for GET; Aborting!`
The program cannot allocate memory for the GET.

`get_ipidef: initialize drive to read defects failed`
An attempt to initialize the defect cylinder failed.

`Growth Error Table stored on disk is invalid`
The GET on the drive is not valid.

`iget_oemdefect: unable to uc_malloc enough space for zone table`
The program cannot allocate memory for the zone table.

`IOCTL failed for GET_CONFIG on disk`
A call to the driver to get the configuration of the device failed.

`IOCTL failed for GET_CONFIG on IPI disk`
A call to the driver to get the configuration of the device failed.

`IOCTL failed for GET_ECONFIG on disk`
A call to the driver to get the configuration of the device failed.

`Invalid Flaw map on cyl %d, head %d`
The flaw data for the given location is not valid.

`read_get: unable to alloc space for GET`
The program cannot allocate memory for the GET.

`read_get: unable to alloc space for tmp GET`
The program cannot allocate memory for a local copy of the GET.

`slip failed: %s`
The program cannot slip a sector for the specified reason.

`slip: unable to malloc() space for GET; Aborting!`
The program cannot allocate memory for the GET.

`slip: unable to write the Growth Error Table`
The program cannot write the GET.

```
Trying cylinder %d.
Trying alternate location.
```
An alternate flaw cylinder is tried.

`Unable to %s the Growth Error Table`
The program cannot access the GET.

```
Unable to add slipped sector to the Growth Error Table
Number of defects exceeds maximum of %d
```
Self explanatory.

`Unable to open file '%s'`
The program cannot open the specified file.

`Unable to read GET defect list: %s`
The program cannot read the GET.

EXAMPLES

Example 1: The following command formats drive 1 on controller 0, then maps only the OEM defect list:

```
dformat  s01 -l 2
```

Example 2: The following command formats drive 1 on controller 1, then does not map:

```
dformat  b11 -l 1
```

Example 3: The following command formats drive 0 on controller 0, then maps all defects in the GET:

```
dformat  b00
```

SEE ALSO

`dstat`(8) to output activity information about the disk subsystem
`dsurf`(8) to verify disk media

NAME

dm – Displays central memory

SYNOPSIS

```
dm
dm -[l | r] -[h | o] [q] address
dm -[l | r] [q] x address
dm -[l | r] -[h | o] [q] [upper_parcel] [lower_parcel]
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The dm command reads central memory into the IOS and displays it on the IOS console by using the I/O channel. You can format display into a hexadecimal or octal representation of memory, or an exchange package format of central memory. The display of central memory is done continuously (refresh) until a dm command without options is executed, at which time the display is stopped.

To display multiple screens of different regions of central memory simultaneously, the screen is divided into two halves, each with a different central memory address (and a different display format if desired).

The display exchange package format is done on a half-screen basis, but it can be displayed on one half of the screen and allow a central memory format to be displayed on the other half.

The dm command accepts the following options:

`l` Displays on left half of the screen (default).

`r` Displays on right half of the screen.

`h` Specifies hexadecimal format.

`o` Specifies octal format (default).

`q` Quits the dm session.

`x` Displays the memory in exchange package format.

address Specifies starting address of central memory to display, or an 8-bit address when the parcel parameter is specified.

`upper_parcel`
Specifies upper 16 bits of an address.

`lower_parcel`
Specifies lower 16 bits of an address.

To display more than one area of central memory, first enter the dm command to display region A. To add region B, enter a second dm command. You can enter additional dm commands at any time.

NOTES

Because the display is refreshed at a rapid rate, the screen cursor is not always resting at the expected position on the command line. If you enter a command line, however, it will display properly.

Entering any IOS command that causes additional screen output when displaying central memory (other than a return prompt) potentially produces an unusable display. Stop the dm command first, and then restart it after entering IOS commands that generate screen output.

The default address mode is octal.

When viewing central memory on an exchange package, use the <↑> and <↓> arrow keys to scroll the display forward or backward, respectively.

Because of the way console input/output is controlled, <CONTROL-c> functionality is disabled. If you press the q key or specify dm without options, your dm session will quit.

MESSAGES

`Open failed for` *<memory device>*
Opening central memory failed.

SEE ALSO

fm(8) to fill memory
lm(8) to load central memory

NAME

ds – Loads and deadstarts a diagnostic test

SYNOPSIS

ds [*filename*[*.ext*]] [*cpu*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The ds command loads into mainframe memory a diagnostic test that has a .bin or .xxx file extension and deadstarts it. If you do not specify an extension, this command, by default, searches the current directory for the file name that has a .bin extension. If the .bin extension is unsuccessful, the file name that has an .xxx extension is searched for next. If the file is not found, an error message is displayed.

The ds command accepts the following options:

filename Specifies diagnostic test file.

.ext Specifies file extension.

cpu Specifies CPU number (0 through 15). If you do not specify a CPU number, the default is 0, which deadstarts only CPU 0.

MESSAGES

`Clock must be ON`
Self explanatory.

`CPU number must be between 0 and <Max CPUs>`
The CPU number specified on the command line is either less than 0 or greater than the maximum number of CPUs supported.

`Scan function to read status failed`
Self explanatory.

`Unable to load file <filename> ... aborting`
The file name specified could not be loaded into central memory.

NAME

`dslip` – Slips one sector

SYNOPSIS

`dslip b`*cd sector*

`dslip i`*cd sector*

`dslip s`*cd sector*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `dslip` command provides a subset of the functionality that the `dsurf`(8) command offers. Based on the sector number specified, `dslip` either slips or maps the sector or track.

First, `dslip` tries to slip the sector. If no more sectors are available on the track or a media error exists in the sector header itself, `dslip` maps the entire track to a valid one. The controller tries to salvage the data.

The `dslip` command also updates the Growth Error table (GET).

This command accepts the following options:

`b` Indicates a buffered intelligent peripheral interface (IPI) drive.

`i` Indicates an IPI drive.

`s` Indicates an Small Computer System Interface (SCSI) drive.

c Specifies a controller number (0 to F).

d Specifies a disk (0 to F).

sector Specifies a sector number from beginning of device.

MESSAGES

`dslip: block %d reported error on pass %d`
An error was detected while reading the specified sector.

`dslip: cannot get memory for flaw table`
The program cannot allocate memory for the flaw table.

`dslip: device %s does not exist`
Device does not exist in the disk table kept by the driver.

`dslip: failed: %s`
The call to slip the sector failed.

`dslip: get configuration fails for device %s`
A call to the driver to get the configuration of the device fails.

`dslip: unable to open disk %s; aborting.`
The program cannot open the disk device.

`dslip: verified data %d times; still slip? (y/n):`
The sector was read successfully.

`read_get: unable to alloc space for GET`
The program cannot allocate memory for the GET.

`read_get: unable to alloc space for tmp GET`
The program cannot allocate memory for a local copy of the GET.

`slip failed: %s`
The program cannot slip a sector for the specified reason.

`slip: unable to malloc() space for GET; Aborting!`
The program cannot allocate memory for the GET.

`slip: Unable to write the Growth Error Table`
The program cannot write the GET.

`Unable to %s the Growth Error Table`
The program cannot access the GET.

`Unable to add slipped sector to the Growth Error Table.`
`Number of defects exceeds maximum of %d`
Self explanatory.

SEE ALSO

`dsurf`(8) to verify disk media

NAME

`dstat` – Outputs activity information about the disk subsystem

SYNOPSIS

`dstat b`*cd*

`dstat i`*cd*

`dstat s`*cd*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `dstat` command briefly summarizes disk activity since the IOS was booted. Optionally, the driver of a specific disk type may request more disk-specific information.

The `dstat` command accepts the following options:

`b` Indicates a buffered intelligent peripheral interface (IPI) disk drive.

`i` Indicates an IPI disk drive.

`s` Indicates a Small Computer System Interface (SCSI) disk drive.

c Specifies controller number (0 to F).

d Specifies disk (0 to F).

SEE ALSO

`crash`(8) to analyze IOS internal information
`systat`(8) to display general IOS internal status

NAME

dsurf – Performs disk surface analysis

SYNOPSIS

dsurf b*cd* [-adfirwv] [-l *level*] [-n *blocks*] [-s *start*] [-p *passes*] [-t *count*]

dsurf i*cd* [-adfirwv] [-l *level*] [-n *blocks*] [-s *start*] [-p *passes*] [-t *count*]

dsurf s*cd* [-adfirwv] [-l *level*] [-n *blocks*] [-s *start*] [-p *passes*] [-t *count*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The dsurf command performs surface analysis on a disk, and it provides a means to pattern test a drive and a means for flawing a drive.

The dsurf command accepts the following options:

b	Indicates a buffered intelligent peripheral interface (IPI) drive.
i	Indicates an IPI drive.
s	Indicates an small computer system interface (SCSI) drive.
c	Specifies controller number (0 to F).
d	Specifies disk (0 to F).
-a	Asks before flawing (default is to flaw silently).
-d	Specifies debug mode. Errors are not flawed.
-f	Runs test until one pass completes without an error.
-i	Inhibits recheck on flawed errors (default is to recheck the space after flawing).
-r	Does not flaw errors (default is to flaw).
-w	Allows writing without prompting (default is to prompt the user for a response if the disk will be written on). This option is used for background usage.
-v	Specifies verbose mode. Its use is recommended.
-l *level*	Specifies test level. A level of 0 is a read-only test (the default). A level of 1 is an eight-pattern write and read (the patterns are 0x00, 0xFF, 0xF0, 0x0F, 0xCC, 0x33, 0xAA, and 0x55). A level of 2 is a four random-pattern write and read.
-n *blocks*	Specifies number of blocks to test (default is the entire drive).
-s *start*	Starts block address (default is 0).
-p *passes*	Specifies number of passes to run (default is 1).
-t *count*	Reads or writes I/O size in sectors (default is one track).

MESSAGES

dsurf: Cannot get configuration information on disk
: A call to the driver to get the configuration of the device fails.

dsurf: Cannot get memory for flaw table
: The program cannot allocate memory for the flaw table.

`dsurf: controller/device (%s) not found`
Device does not exist in disk table kept by the driver.

`dsurf: error %sing track %ld - checking for the bad sector`
An error was found during the I/O operation. The program then determines which sector within the range of sectors accessed is bad.

`dsurf: Invalid level %d`
The level specified on the command line is not valid.

`dsurf: Invalid number of blocks %d`
The number of blocks specified is not valid.

`dsurf: Invalid number of passes %d`
The number of passes specified is not valid.

`dsurf: I/O size of %d is invalid`
The request size specified is not valid.

`dsurf: Number of blocks entered %d is greater than %d - set to %d`
The number of blocks specifies is more than the disk capacity.

`dsurf: Open on drive %s failed: %s`
The call to open failed.

`dsurf: Slip/map could not complete for block %ld`
The call to slip the sector fails.

`dsurf: Track %ld will be slipped`
The track specified is slipped.

`dsurf: Starting block of %d is invalid`
The start block specified is not valid.

`dsurf: Unknown device '%s'`
The specified device name is not valid.

`dsurf: Verify of IPI track headers fails for %s`
The program cannot verify the track headers.

`Error [%s] found at sector %ld`
An error was enountered at the specified sector.

`read_get: unable to alloc space for GET`
The program cannot allocate memory for the Growth Error table (GET).

`read_get: unable to alloc space for tmp GET`
The program cannot allocate memory for a local copy of the GET.

`slip failed: %s`
The program cannot slip a sector for the specified reason.

`slip: unable to malloc() space for GET; Aborting!`
The program cannot allocate memory for the GET.

`slip: Unable to write the Growth Error Table`
The program was uanble to write the GET.

`Unable to %s the Growth Error Table`
The program cannot access the GET.

`Unable to add slipped sector to the Growth Error Table.`
`Number of defects exceeds maximum of %d`
Self explanatory.

EXAMPLES

Example 1: The following command reads the entire disk and prompts the user before flawing:

```
dsurf  b00  -va
```

Example 2: The following command writes four random patterns on the entire drive, then reads the drive. It does 10 passes:

```
dsurf  s01 -vp 10 -l 2
```

Example 3: The following command writes eight patterns on the entire drive, then reads the drive. It runs until no errors are found on a single pass:

```
dsurf  b11 -fvl 1
```

SEE ALSO

`dformat`(8) to format a disk
`dslip`(8) to slip sectors

NAME

echo – Displays a message

SYNOPSIS

echo [*string*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The echo command displays a specified message. It accepts the following argument:

string Character string, which is displayed on the screen when the command executes.

NOTES

The echo command is used only in a command script.

EXAMPLES

The following line prints the Debug Test Message message when the command file executes:

```
echo Debug Test Message
```

NAME

ed – Edits a text file

SYNOPSIS

ed [*file*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The ed editor is the standard text editor. If you specify the *file* argument, ed simulates an e subcommand (see the following text) on the specified file; that is, the file is read into the ed buffer so that you can edit it. The ed editor operates on a copy of the file it is editing; changes made to the copy do not affect the file until you specify a w (write) command. The copy of the text being edited resides in a temporary file called the *buffer*. Only one buffer exists.

Commands to ed have a simple and regular structure: zero, one, or two *addresses*, followed by a single-character *command*, possibly followed by parameters to that command. These addresses specify one or more lines in the buffer. Because every command that requires addresses has default addresses, you usually can omit the addresses.

Generally, only one command may appear on a line. Certain commands allow the input of text, which is placed in the appropriate place in the buffer. While ed is accepting text, it is in *input mode*. In this mode, *no* commands are recognized; all input is merely collected. To exit input mode, type a period (.) by itself at the beginning of a line, followed immediately by pressing a <RETURN>.

The ed editor supports a limited form of *regular expression* notation; regular expressions are used in addresses to specify lines and in some commands (for example, s) to specify parts of a line that will be substituted. A regular expression (RE) specifies a set of character strings. A member of this set of strings is said to be *matched* by the RE. The REs that ed allows are constructed as follows (the following one-character REs match one character):

1.1 An ordinary character (not one of those discussed in 1.2) is a one-character RE that matches itself.

1.2 A backslash (\) followed by any special character is a one-character RE that matches the special character itself. The special characters are as follows:

a. ., *, [, and \ (period, asterisk, left bracket, and backslash, respectively), which are always special, except when they appear within brackets ([ ] (see 1.4).

b. ^ (caret or circumflex), which is special at the beginning of an entire RE (see 3.1), or when it immediately follows the left pair of brackets ([ ]) (see 1.4).

c. $ (dollar sign), which is special at the end of an entire RE (see 3.2).

d. The character used to bound (such as, delimit) an entire RE, which is special for that RE (for example, see how slash (/) is used in the g command).

1.3 A period (.) is a one-character RE that matches any character except a newline character.

1.4 A nonempty string of characters enclosed in brackets (`[ ]`) is a one-character RE that matches any one character in that string. If, however, the first character of the string is a circumflex (`^`), the one-character RE matches any character except newline and the remaining characters in the string. The `^` has this special meaning only when it occurs first in the string. Use the minus (`-`) character to indicate a range of consecutive ASCII characters (for example, `[0-9]` is equivalent to `[0123456789]`). If it occurs first (after an initial `^`, if any) or last in the string, the `-` loses this special meaning. The right bracket (`]`) does not terminate such a string when it is the first character within it (after an initial `^`, if any); for example, `[ ]a-f]` matches either a bracket (`]`) or one of the letters `a` through `f`, inclusive. The four characters listed in 1.2.a stand for themselves within such a string of characters.

To construct REs from one-character REs, use the following rules:

2.1 A one-character RE is a RE that matches whatever the one-character RE matches.

2.2 A one-character RE followed by an asterisk (`*`) is a RE that matches zero or more occurrences of the one-character RE. If there is any choice, the longest leftmost string that permits a match is selected.

2.3 A one-character RE followed by `\{`*m*`\}`, `\{`*m*`,\}`, or `\{`*m*`,`*n*`\}` is a RE that matches a *range* of occurrences of the one-character RE. The values of *m* and *n* must be nonnegative integers less than 256; `\{`*m*`\}` matches *exactly m* occurrences; `\{`*m*`,\}` matches *at least m* occurrences; `\{`*m*`,`*n*`\}` matches any number of occurrences between *m* and *n*, inclusive. Whenever a choice exists, the RE matches as many occurrences as possible.

2.4 The concatenation of REs is a RE that matches the concatenation of the strings matched by each component of the RE.

2.5 A RE enclosed between the character sequences `\(` and `\)` is a RE that matches whatever the unadorned RE matches.

2.6 The expression `\`*n*, matches the same string of characters as was matched by an expression enclosed between `\(` and `\)` earlier in the same RE; *n* is a digit. The subexpression specified is the one that begins with the *n*th occurrence of `\(` counting from the left (for example, the expression `^\(.*\)\1$` matches a line that consists of two repeated appearances of the same string).

Finally, an entire RE may be constrained to match only an initial segment or final segment of a line (or both).

3.1 A circumflex (`^`) at the beginning of an entire RE constrains that RE to match an initial segment of a line.

3.2 A dollar sign (`$`) at the end of an entire RE constrains that RE to match a final segment of a line.

The string `^entire RE$` constrains the entire RE to match the entire line.

The null RE (such as, `//`) is equivalent to the last RE encountered. See also the last paragraph before the FILES section.

To understand addressing in `ed`, you must know that at any time the current line is the last line affected by a command; the exact effect on the current line is discussed under the description of each command. Addresses are constructed as follows:

1. The `.` character addresses the current line.
2. The `$` character addresses the last line of the buffer.
3. A decimal number *n* addresses the *n*th line of the buffer.
4. *'x* addresses the line marked with the mark name character *x*, which must be a lowercase letter. To mark lines, use the `k` command.

5. A RE enclosed by slashes (/) addresses the first line found by searching forward from the line following the current line toward the end of the buffer and stopping at the first line that contains a string that matches the RE. If necessary, the search wraps around to the beginning of the buffer and continues up to and including the current line so that the entire buffer is searched. See also the last paragraph before the FILES section.
6. A RE enclosed in question marks (?) addresses the first line found by searching backward from the line preceding the current line toward the beginning of the buffer and stopping at the first line that contains a string matching the RE. If necessary, the search wraps around to the end of the buffer and continues up to and including the current line. See also the last paragraph before the FILES section.
7. An address followed by a plus sign (+) or a minus sign (-) followed by a decimal number specifies that address plus (respectively minus) the indicated number of lines. You can omit the plus sign.
8. If an address begins with + or -, the addition or subtraction is taken with respect to the current line (for example, -5 is understood to mean 0.-5).
9. If an address ends with + or -, 1 is added to or subtracted from the address, respectively. As a consequence of this rule and of rule 8, the - address refers to the line preceding the current line. (To maintain compatibility with earlier versions of the editor, the ^ character in addresses is entirely equivalent to -.) Moreover, trailing + and - characters have a cumulative effect, so - - refers to the current line less 2.
10. For convenience, a comma (,) stands for the address pair 1,$; a semicolon (;) stands for the pair .,$.

Commands may require zero, one, or two addresses. Commands that do not require addresses regard the presence of an address as an error. Commands that accept one or two addresses assume default addresses when an insufficient number of addresses is specified; if more addresses are specified than such a command requires, the last one(s) is used.

Typically, addresses are separated from each other by a comma (,). They also may be separated by a semicolon (;). In the latter case, the current line (.) is set to the first address, and only then is the second address calculated. You can use this feature to determine the starting line for forward and backward searches (see rules 5 and 6). The second address of any two-address sequence must correspond to a line that follows, in the buffer, the line corresponding to the first address.

In the following list of ed commands, the default addresses are in parentheses. The parentheses are not part of the address; they show that the given addresses are the default. Generally, it is illegal for more than one command to appear on a line. However, any command (except e, f, r, or w) may be suffixed by l, n, or p; in which case, the current line is either listed, numbered, or printed, respectively, as discussed under the l, n, and p commands.

(.)a
<text>
.

(.)c — The append command reads the given text and appends it after the addressed line; . is left at the last inserted line, or, if there were none, at the addressed line. Address 0 is legal for this command: it causes the appended text to be placed at the beginning of the buffer. You can enter a maximum number of 256 characters (including the newline character). The change command deletes the addressed lines, then accepts input

<text>
. — Text that replaces these lines; . is left at the last line input, or, if there were none, at the first line that was not deleted.

(.,.)d — The delete command deletes the addressed lines from the buffer. The line after the last line deleted becomes the current line; if the lines deleted were originally at the end of the buffer, the new last line becomes the current line.

`e` *file* The `edit` command deletes the entire contents of the buffer, and then it reads in the specified file; `.` is set to the last line of the buffer. If you do not specify a file name, the currently remembered file name, if any, is used (see the `f` command). The number of characters read is typed; *file* is remembered for possible use as a default file name in subsequent `e`, `r`, and `w` commands. If *file* is replaced by `!`, the rest of the line is considered to be a shell (`sh`(1)) command whose output will be read. Such a shell command is not remembered as the current file name. See the MESSAGES section.

`E` *file* The `E` command is like the `e` command, except that the editor does not check to see whether any changes were made in the buffer since the last `w` command.

`f` *file* If you specify *file*, the file name command changes the currently remembered file name to *file*; otherwise, it prints the currently remembered file name.

`(1,$)g/` *RE* `/"`*command list*
In the global command, the first step is to mark each line that matches the given RE. Then, for every such line, the given *command list* is executed with `.` initially set to that line. One command or the first command in a list of commands appears on the same line as the global command. All lines of a multiline list, except the last line, must end with a `\`; `a`, `i`, and `c` commands and associated input are permitted. You may omit the `.` terminating input mode if it would be the last line of the *command list*. An empty *command list* is equivalent to the `p` command. The `g` and `v` commands are not permitted in the *command list*. See the BUGS section and the last paragraph before the FILES section.

`(.)i`
<text>
`.` The `insert` command inserts the given text before the addressed line; `.` is left at the last inserted line, or, if there were none, at the addressed line. This command differs from the `a` command only in the placement of the input text. Address 0 is not legal for this command. You can enter a maximum number of 256 characters per line (including the newline character).

`(.,.+1)j` The `join` command joins contiguous lines by removing the appropriate newline characters. If you specify one address, this command does nothing.

`(.)k`*x* The `mark` command marks the addressed line with name *x*, which must be a lowercase letter. The address *'x* then addresses this line; `.` is unchanged.

`(.,.)l` The `list` command prints the addressed lines in an unambiguous way: a few nonprinting characters (for example, tab and backspace) are represented by visually mnemonic overstrikes. All other nonprinting characters are printed in octal, and long lines are folded. You may append an `l` command to any command other than `e`, `f`, `r`, or `w`.

`(.,.)m`*a* The `move` command repositions the addressed line(s) after the line addressed by `a`. Address 0 is legal for `a` and moves the addressed line(s) to the beginning of the file. If address `a` falls within the range of moved lines, it is an error; `.` is left at the last line moved.

`(.,.)p` The `print` command prints the addressed lines; `.` remains at the last line printed. You may append the `p` command to any other command other than `e`, `f`, `r`, or `w` (for example, `dp` deletes the current line and prints the new current line).

`P` The editor prompts with a * for all subsequent commands. The `P` command alternately turns this mode on and off; it is initially off.

`q` The `quit` command causes `ed` to exit. An automatic write of a file is not done (see the MESSAGES section).

`Q` This command causes the editor to exit without checking whether any changes were made in the buffer since the last `w` command.

`($)r` *file* The `read` command reads in the specified file after the addressed line. If no file name is specified, the currently remembered file name, if any, is used (see `e` and `f` commands). The currently remembered file name is not changed unless *file* is the very first file name mentioned since `ed` was invoked. Address 0 is legal for `r` and reads the file at the beginning of the buffer. If the read is successful, the number of characters read is typed; `.` is set to the last line read in. If *file* is replaced by `!`, the rest of the line is considered to be a shell (`sh`(1)) command whose output will be read (for example, "`$r !ls`" appends current directory to the end of the file being edited). Such a shell command is not remembered as the current file name.

`(.,.)s/`*RE / replacement /* or
`(.,.)s/`*RE/replacement*`/g` or
`(.,.)s/`*RE/replacement*`/n` n = 1–512

The `substitute` command searches each addressed line for an occurrence of the specified RE. In each line in which a match is found, all (nonoverlapped) matched strings are replaced by the *replacement* if the global replacement indicator `g` appears after the command. If the global indicator does not appear, only the first occurrence of the matched string is replaced. If a number *n* appears after the command, only the *n*th occurrence of the matched string on each addressed line is replaced. It is an error for the substitution to fail on all addressed lines. Any character other than space or new line may be used instead of `/` to delimit the RE and the *replacement*; `.` is left at the last line on which a substitution occurred. See also the last paragraph before the FILES section.

An ampersand (`&`) that appears in the *replacement* is replaced by the string that matches the RE on the current line. To suppress the special meaning of `&` in this context, precede it by `\`. As a more general feature, the characters `\`*n* (*n* is a digit) are replaced by the text matched by the *n*th regular subexpression of the specified RE enclosed between `\(` and `\)`. When nested parenthesized subexpressions are present, *n* is determined by counting occurrences of `\(` starting from the left. When the character `%` is the only character in the *replacement*, the *replacement* used in the most recent substitute command is used as the *replacement* in the current substitute command. The `%` loses its special meaning when it is in a replacement string of more than one character or is preceded by a `\`.

To split a line, substitute a newline character into it. You must escape the newline character in the *replacement* by preceding it with a `\`. You cannot do such substitution as part of a `g` or `v` command list.

`(.,.)t`*a* This command acts just like the `m` command, except that a copy of the addressed lines is placed after address *a* (which may be 0); `.` is left at the last line of the copy.

`u` The `undo` command nullifies the effect of the most recent command that modified anything in the buffer, namely the most recent `a`, `c`, `d`, `g`, `i`, `j`, `m`, `r`, `s`, `t`, `v`, `G`, or `V` command.

`(1,$)v/`*RE/command list*

This command is the same as the global `g` command, except that the *command list* is executed with `.` initially set to every line that does not match the RE.

`(1,$)V/`*RE /*

This command is the same as the interactive global `G` command, except that the lines that are marked during the first step are those that do not match the RE.

`(1,$)w` *file*"

The `write` command writes the addressed lines into the specified file. If the file does not exist, it is created with mode 666 (readable and writable by everyone), unless your `umask` setting (see `umask`(1)) dictates otherwise. The currently remembered file name is not changed unless *file* is the very first file name mentioned since `ed` was invoked. If you do not specify file name, the currently remembered file name, if any, is used (see `e` and `f` commands); `.` is unchanged. If the command is successful, the number of characters written is typed. If *file* is replaced by `!`, the rest of the line is considered to be a shell (`sh`(1)) command whose standard input is the addressed lines. Such a shell command is not remembered as the current file name.

`($)=` The line number of the addressed line is typed; `.` is unchanged by this command.

`(.+1) <new-line>`

When an address is on a line by itself, the addressed line will be printed. A newline by itself is equivalent to `.+1p`; it is useful for stepping forward through the buffer.

A command line can consist of 512 characters per line, 256 characters per global command list, and 64 characters per file name. The limit on the number of lines depends on the amount of user memory: each line takes 1 word.

When reading a file, `ed` discards `ASCII null` characters. `ed` cannot edit files (for example, `a.out`) that contain characters not in the ASCII set (bit 8 on).

If a newline character does not terminate a file, `ed` adds a newline character and outputs a message that explains what it did.

If the closing delimiter of a RE or of a replacement string (such as, `/`) would be the last character before a new line, you may omit that delimiter, in which case, the addressed line is printed. The following pairs of commands are equivalent:

```
s/s1/s2
s/s1/s2/p

g/s1
g/s1/p

?s1
?s1?
```

CAUTIONS

You should keep reasonable editing sessions under 10 Kbytes. Lines are limited to 4096 characters.

When reading a file, `ed` discards `ASCII null` characters and all characters after the last new line. `ed` cannot edit files (for example, `a.out`) that contain characters that are not in the ASCII set (bit 8 on).

Large files generate larger editor temporary files and cost many processor cycles on entry to `ed`.

MESSAGES

The following are diagnostic messages:

`?` For command errors or if a backspace is input (in which case, you remain in command mode).

`?` *file* For an inaccessible file. (For detailed explanations, use the `help` and `Help` commands.)

If changes were made in the buffer since the last `w` command that wrote the entire buffer and the user uses the `e` or `q` command, `ed` warns the user that the `ed` buffer may be destroyed. It prints `?` and allows the user to continue editing. At this point, a second `e` or `q` takes effect.

FILES

`/tmp` Default directory for temporary work file

NAME

enstat – Displays Ethernet controller status and statistics

SYNOPSIS

enstat [-e *xxxx*]

enstat [-m *c lvl*]

enstat [-r *c*]

enstat [-s *c*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The enstat command displays Ethernet status, statistics, and error codes. If you omit arguments, enstat displays the status of all Ethernet controllers attached to the IOS.

The enstat command accepts the following options and arguments:

-e Displays the meaning of Ethernet packet error status bits.

-m Changes the automatic error logging level for a controller. Error reporting is on by default.

-r Resets controller statistics counters.

-s Displays controller statistics.

c Specifies controller number (0 to 3).

lvl Specifies the message reporting level. Valid values are on, off, or perf. Default is on.

xxxx Specifies packet status bits (hexadecimal value).

The enstat command displays the following status information:

```
Controller X at address 0xYYYYYYYY [:  controller not initialized]
Ethernet address=0xZZZZZZZZZZZZ
```

The -s option displays the following statistics (in the order listed):

Successful transmissions:*xx*
Number of packets transmitted successfully.

Successful receives:*xx*
Number of packets received successfully.

Multiple retries on xmit:*xx*
Number of multiple retries on transmissions.

Single retries:*xx*
Number of successful transmissions after one retry.

Too many retries:*xx*
Number of unsuccessful retries.

Xmit delayed due to active medium:*xx*
Number of transmissions delayed due to active medium.

Xmit chaining failed:*xx*
Number of failures due to internal chaining faults.

`Transmit data fetch failed:`*xx*
Number of failures due to internal data fetch underruns.

`Collision after xmit:`*xx*
Number of transmission collisions.

`Lost carrier:`*xx*
Number of times the carrier signal was lost.

`Xmit length > 1518:`*xx*
Number of attempts to transmit packets that are too large (babble).

`Transceiver mismatch:`*xx*
Number of times a signal from an Ethernet type 2 transceiver was not detected (not an error).

`Xmit memory errors:`*xx*
Number of internal memory errors detected.

`No receive buff available:`*xx*
Number of packets missed due to no available buffers.

`Checksum failed:`*xx*
Number of packets received with Ethernet checksums that were not valid.

`Framing error:`*xx*
Number of packets received with framing errors.

`Receive chaining failed:`*xx*
Number of failures due to internal chaining failure.

`Receive data store failed:`*xx*
Number of failures due to a buffer overrun.

`Receive memory error:`*xx*
Number of failures due to memory parity errors.

The Ethernet automatic error logging facility examines the controller's statistics every 30 seconds for any abnormalities. The controller's statistics are examined for the following conditions:

- Transmit chaining failures
- Transmit data fetch failures
- Lost carrier
- Transmitter babble
- Transmit memory errors
- Receive chaining failures
- Receive data store failures
- Receive memory errors
- Receive cyclic redundancy check (CRC) error rate greater than 5% of received packets
- Receive framing error rate greater than 5% of received packets

If any of these conditions occur during the time period, a message is displayed on the console and logged in the `/adm/syslog` file on the IOS.

In addition to reporting the above abnormalities, the `perf` message reporting level examines the Ethernet statistics for potential performance problems. The following additional conditions are reported:

- Failed packet transmit rate gerater than 5% of transmitted packets
- Multiple retry rate greater than 10% of transmitted packets

- Single retry rate greater than 30% of transmitted packets
- Transmit delay rate due to active medium greater than 10% of transmitted packets
- Collision after transmit rate greater than 10% of transmitted packets
- Backlog of output packets to IOS due to Ethernet controller delays
- Large backlog of IOBB channel requests

By default, automatic error reporting is turned on. If the messages become too numerous, the `-m` option lets you disable the generation of messages.

NOTES

The `-e` option displays the status of packets received in error. The Ethernet driver displays the status (hexadecimal value) of abnormal packets it receives. The `-e` option lets users interpret the meaning of this status.

EXAMPLES

Example 1: The following command displays the statistics for controller 0:

```
enstat -s 0
```

Example 2: The following command turns off automatic error logging on Ethernet controller 0:

```
enstat -m 0 off
```

NAME

`errpt` – Processes the error report generated by IOS kernel

SYNOPSIS

`errpt` [*filename*[*.ext*]]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `errpt` command processes data collected by the IOS error-logging mechanism and generates a report of that data in the specified file. If you do not specify a file, `errpt` displays the output on the IOS console, one error at a time. To advance to the next error, press any keyboard key. The error display stops when all errors are displayed (maximum of 64) or after you press the <q> (quit) key.

The `errpt` command accepts the following arguments:

filename Specifies IOS file name.

.ext Specifies file name extension.

This command assists in diagnosing problems that prevent the regular error logging mechanism of UNICOS to operate or to be viewed.

NOTES

Any IOS reset clears all previous error data collected.

This command displays only the last 64 error log entries.

SEE ALSO

`dstat`(8) to display disk status information

NAME

`fg` – Brings to the foreground an IOS command that is suspended or running in the background

SYNOPSIS

`fg` [*command id*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `fg` command starts a suspended command and suspends the interactive shell until that command completes.

If a command is running in the background, that command is placed in the foreground and the interactive shell is suspended until that command completes.

The `fg` command accepts the following argument:

command id
Specifies command or job identification number between 1 and 16 specified in the job output.

EXAMPLES

If the `dformat` command was started in the foreground (for example, the `&` character was not placed at the end of the command line), and then the user entered `<CONTROL-Z>` to suspend it, the `dformat` command can resume execution by doing the following:

```
fg
```

NAME

fm – Fills central memory

SYNOPSIS

fm *start count* [*parcelA*] [*parcelB*] [*parcelC*] [*parcelD*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The fm command fills mainframe central memory with the specified values by using the I/O channel.

The fm command accepts the following arguments:

start Relative address of memory to start filling.

count Number of words (in decimal) to fill.

parcelA Value to fill parcel A (most significant); default is 0.

parcelB Value to fill parcel B; default is 0.

parcelC Value to fill parcel C; default is 0.

parcelD Value to fill parcel D (least significant); default is 0.

NOTES

You must specify at least 1 parcel.

Because this command accesses central memory through the data channels, the CPU clock must be on.

MESSAGES

`Mem open failed`
Opening of mainframe central memory failed.

`Write mem failed`
Writing to mainframe central memory failed.

EXAMPLES

The following command line writes the value `123 5678 9ABC DEF0` to central memory word 100 hexadecimal through word 102 hexadecimal:

```
fm 100 3 123 5678 9ABC DEF0
```

SEE ALSO

am(8) to alter memory
dm(8) to display central memory
lm(8) to load central memory

NAME

goto – Transfers control to a command file

SYNOPSIS

goto *:label*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The goto command transfers control to a command file.

The goto command accepts the following argument:

label A string preceded by a colon (:). The first 8 characters are significant.

NOTES

This command executes only in a command script.

EXAMPLES

A command file that contains the following three lines of code prints Thanks a million until interrupted by pressing <CONTROL-c>, which kills any IOS command.

```
:AgainSam

echo Thanks a million

goto :AgainSam
```

NAME

`head` – Displays the first few lines of a specified file

SYNOPSIS

`head [-n]` *filename*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `head` command outputs the given number of lines (default 10) of the specified file.

The `head` command accepts the following arguments:

-n Specifies a line count.

filename Specifies input file.

EXAMPLES

The following example displays the first 20 lines of the `aaa` file:

```
head -20 aaa
```

NAME

help – Displays commands and their syntax

SYNOPSIS

help [*cmd*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

If *cmd* is a command name, help displays the command and its syntax. If *cmd* is a letter, help displays all commands that start with that letter and the syntax of those commands. If you do not specify an argument, help displays all commands and their syntax.

The help command accepts the following argument:

cmd Specifies command to be displayed or first letter of commands to be displayed.

EXAMPLES

The following example displays all commands that start with the letter d and the syntax of those commands:

```
$ help d

debug  debug [value]
dflawr <e|i>cd [-lr] [-s serial#] [-f filename]
dflawr scd [-l] [-f filename]
dflawr bcd [-l] [-f filename]
dflawr dcd drive [-l] [-f filename]
dflaww <s|e|i|b>cd [-f filename]
dflaww dcd drive [-f filename]
dformat <e|i>cd [-l level] [-s serial] [-f file]
dformat dcd [Bxxx] [level]
dformat <s|b>cd [-l level] [-f file]
dm -[l|r] -[h|o] [q] address
dm -[l|r] x address
dm -[l|r] -[h|o] [q] [upper_parcel] [lower_parcel]
dm
ds [filename[.ext]] cpu
dslip <s|e|i|b>cd sector\r\ndslip C: sector
dstat <s|e|d|i|b>cd
dump [-v] <s|e|d|i|b>cd sa [word_count]
```

NAME

`if` – Allows conditional transfer of control

SYNOPSIS

`if` *n* `goto` :*label*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `if` command compares *n* with the return code from the previous command. If a match exists, control is transferred to the line that immediately follows the label.

The `if` command accepts the following arguments:

n Value to compare with the return code from the previous command.

label String preceded by a colon (:); the first 8 characters are significant.

NOTES

This command executes only in a shell script.

EXAMPLES

A command file that contains the following code repeatedly reads the value of the program counter and prints it until it is equal to 1234. When the program counter equals 1234, the `Done !!!` message is printed.

```
:KeepGoing

dr P

if 1234 goto :Done

goto :KeepGoing

:Done

echo Done !!!
```

NAME

`iosdump` – Dumps the I/O processor and I/O buffer board memories to file on the NFS mounted file system

SYNOPSIS

`iosdump [-n` *filename*`] [-s` *iobbsize*`]`

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `iosdump` command saves I/O processor (IOP) and I/O buffer board (IOBB) memories to the specified file. If an IOS panic occurs, you must perform this task.

The default area dumped during an IOS dump is not a complete dump of IOS and IOBB memory. Key areas are dumped that usually are sufficient for a dump analysis. Circumstances may exist in which this may not be sufficient for a full analysis. To obtain a full dump for a particular problem, use the `-s` option and enter the memory size of the IOBB. For example, for an I/O subsystem configured with an IOBB15, you would enter `-s 4096` (4096 being the size of IOBB memory in Kbytes), and for an IOBB25, you would enter `-s 16384`. Dumps that the IOS automatically initiates will be of the default size, and you cannot control this. However, if an auto dump is taken and the IOS stops at the boot prompt, you can initiate another valid dump if you must capture the full IOBB contents. If the IOS kernel has reloaded, the IOBB contents will have been overwritten.

The `iosdump` command accepts the following options:

`-n` *filename* Specifies input file.

`-s` *iobbsize* Saves memory in Kbytes.

SEE ALSO

`crash`(8) to analyze memory information
`mfdump`(8) to dump mainframe memory and registers

NAME

`iostart` – Initiates communication between the IOS and UNICOS

SYNOPSIS

```
iostart
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `iostart` command creates the task that initiates packet communications with UNICOS, and it usually is run from the `/bin/boot` script.

MESSAGES

`IOS n ij-task: open at memory failed`
This message occurs when the `iostart` comand tries an `open()` call on the I/O buffer board (IOBB) driver and fails. The open fails when either the IOBB driver is not initialized or it encounters a catastrophe error from a previous operation and shuts itself down. To restart the IOBB driver, use the `mc` or `sc` command, or reboot the IOS.

`IOS n iostart: ij-state=x, can't execute iostart command`
This message occurs when the `iostart` command is invoked without previously entering an `mc`(8) or `sc`(8) command to reinitialize the IOS to CPU communication. It also can indicate that the `mc` or `sc` command did not initialize the system. Check that all IOSs are running properly and have established communication with the master IOS.

SEE ALSO

`lu`(8) to load UNICOS
`mc`(8) to reinitialize the CPUs and central memory
`sc`(8) to reinitialize the CPUs

NAME

`j90install` – Maintains and installs software on J90 console, IOS-V, and mainframe

SYNOPSIS

```
j90install
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `j90install` command is run on the CRAY J90 series console (Sparc Station 5). It maintains files on the Sun which enable each IOS to boot from the Sun disk. It also installs the software on both the IOS-V and CRAY J90 mainframe.

During the IOS-V install process, `j90install` calls the following commands:

`/install/waitios`
: Ensures that each IOS is loaded before continuing.

`/install/drivers`
: Configures `/config` with all strategies and drivers. It also creates `/install/net.tbl`, which is a table of all network interfaces it finds on each IOS.

`/install/iprobe`
: Creates `/install/disks.tbl`, which is a table of all disks it finds on each IOS.

`/install/autogen`
: Creates `/sys/param` from gathering information from `/install/disks.tbl` and `/install/net.tbl`.

During the UNICOS install process, `j90install` creates and calls the `/install/iboot` script. This script then calls the following:

`/install/ibootcfg`
: Creates `/install/param.ram` from `/sys/param`. This new file has a central memory file system declared and sets `rootdev` to this new central memory file system.

`/bin/lu`
: Loads the generic install kernel (`/install/unicos.ymp`) and central memory parameter file (`/install/param.ram`). It then loads the central memory file system from tape into memory and boots the kernel.

SEE ALSO

UNICOS Installation Guide for the CRAY J90 Series, publication SG-5271

NAME

jbs – Performs boundary scan interconnect test on CRAY J90 series systems

SYNOPSIS

jbs [-h] [-t *test*] [-maxpass #] [-e *info*] [-maxerr #]] [-menu]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The J90 Boundary Scan (jbs) application performs boundary scan interconnect tests on CRAY J90 series systems. The jbs application checks module interconnects for continuity; on-module interconnects and backplane interconnects are tested for every board in the system.

The jbs application accepts the following options:

-h Generates an online help display containing a synopsis and a brief description of the command options and arguments. This program exits immediately after displaying help information.

-t *test* Specifies tests to be performed. The default test level is all. The *test* argument can be one of the following:

- int Performs scan chain integrity test to verify that the scan chain is continuous.
- brd Performs scan chain integrity test and board level boundary scan tests.
- bp Performs scan chain integrity test and backplane level boundary scan tests.
- all Performs scan chain integrity test, board level boundary scan tests, and backplane level boundary scan tests.

-maxpass #
Specifies the number of passes to perform. This value must be greater than 0, and must be less than or equal to 100. The default number of passes is 1.

-e *info* Specifies the type of error information displayed to the user. The default is stan. The argument *info* can be one of the following:

- pass Pass/fail information only is displayed.
- stan Standard error information is displayed.
- ext An extended error information file is created.

-maxerr #
Specifies the number of boundary scan errors to display. To stop after the first error, specify -maxerr 1. The number of errors you specify must be greater than 0, and must be less than or equal to 10000. The default number of errors is 10000.

-menu Invokes the menu system. This option cannot be used with any other command line argument. Only pass and fail information is displayed.

When the jbs application is invoked, it performs a scan chain integrity test on every board in the system. The scan chain integrity test verifies that the boundary scan chain is intact and functional. Execution stops if an error is encountered during the integrity test. After the scan chain integrity test, jbs performs a board level boundary scan test on each board in the system. The board level boundary scan test checks on-module interconnects. The last test jbs performs is a backplane level boundary scan test. The backplane level boundary scan test checks all interconnects that pass through the backplane.

NOTES

Options are necessary only when an override of the default arguments is desired.

When running the `jbs` application, observe the following limitations:

- The `jbs` application does not run when the operating system is active.
- The `jbs` application runs only on CRAY J90 series systems.

Results of the boundary scan tests are displayed on the screen and stored in the `/adm/jbs.log` file.

SEE ALSO

CRAY J90 Series IOS Based Tests, publication HDM–099–0 (This manual is Cray Research Proprietary; dissemination of this documentation to non-CRI personnel requires approval from the appropriate vice president and a nondisclosure agreement. Export of technical information in this category may require a Letter of Assurance.)

NAME

`jcon` – Performs a remote login onto a CRAY J90 series mainframe

SYNOPSIS

```
jcon
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `jcon` command is a script you can run on the J90 series console (Sparc Station 5).

It performs an infinite loop to a remote login session. This allows for immediate reconnection to the IOS during a load or reload process.

To disconnect from the remote login, press the ~ character following immediately by <CONTROL-c>.

NAME

jconfig – CRAY J90 series configuration file builder and editor

SYNOPSIS

jconfig -iobb *devname* -edit -dump -nocr -help -ecc [on | off] -cache [on | off] -mm [on | off] -degrade [even | odd | none] -bpmt [backplane memory_types] -hwconfig [*cpu_bitmap memory_bitmap backplane memory_types*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The jconfig utility is used to build and edit CRAY J90 series system hardware configuration files. Configuration files are built, one for each processor module and one for each memory module in the system. These files are named pm[0-7].cfg for processor modules and mem[0-7].cfg for memory modules. These files are then used during master clear and deadstart sequences (see the mc(8) and ds(8) man pages).

The jconfig utility accepts the following options:

-iobb *devname*
: Specify when testing, or when running jconfig in a STCO test vehicle environment. *devname* is an IOBB device. The default device is /dev/iobb.

-edit
: jconfig runs without doing a hardware read/check. This option is useful for quick editing of .cfg file data or if the hardware is broken.

-dump
: Specified in conjunction with inline options (-ecc, -cache, -mm, -degrade), this option forces jconfig to dump ASCII versions of .cfg files in addition to the binary .cfg files. The ASCII versions have the names pm[0-7].cfg.txt and mem[0-7].cfg.txt.

-nocr
: This option can be specified only with -bpmt or -hwconfig. It allows jconfig to be fully run from the command line with no user interaction. Usually, jconfig enters a menu interface, and there are times when jconfig requests that the user press <RETURN> for confirmation. -nocr bypasses the menus and does not wait for the user to press <RETURN>. This option is useful for running jconfig from command scripts. See the -bpmt and -hwconfig option descriptions.

-help
: Prints all command line options.

If you specify the following inline options, jconfig runs without hardware check or editing capabilities. jconfig reads all .cfg files, alters the desired parameter system-wide, and updates the .cfg files.

-ecc [on | off]
: Sets SECDED error correction mode on or off, system-wide.

-cache [on | off]
: Enables or disables scalar cache, system-wide.

-mm [on | off]
: Enables or disables maintenance mode instructions, system-wide.

-degrade [even | odd | none]
: Degrades system memory so that only even sections will be accessed (even), only odd sections will be accessed (odd), or all sections will be accessed (none).

The following options allow `jconfig` to be run with less user interaction.

`-bpmt` [*backplane memory_type_codes*]
: Specifies the backplane type and memory module type codes. This option is usually specified when running from a script. Backplane types are 1x1, 2x2, 4x4, and 8x8. Memory module type codes are single-digit hexadecimal codes that are read from a sticker on each memory module's face plate. Valid codes are 0, 1, 2, 3, 4, 5, 8, 9, a, b, c, and d. The codes entered are in order starting from memory module 0. (See Example 1 in the EXAMPLES section.)

`-hwconfig` [*cpu_bitmap memory_bitmap backplane memory_type_codes*]
: This option lets you enter all hardware parameters on the command line. `jconfig` does not perform a hardware check. When `-nocr` is specified, `jconfig` runs without user interaction. The following hardware parameters can be specified with `-hwconfig`:

cpu_bitmap
: Specifies a hexadecimal bit map of all CPUs in the system. Each processor module has up to 4 CPU's on it. Bit 2^0 corresponds to CPU 0, bit 2^1 to CPU 1, and 2^31 corresponds to CPU 31. For example, for a 4x4 system with all CPU's installed, the CPU bitmap would be `ffff`.

memory_bitmap
: Specifies a hexadecimal bit map of all memory modules in the system. Bit 2^0 corresponds to memory module 0, 2^1 corresponds to memory module 1, and so on. For a 4x4 system with all memory modules installed, the memory bitmap would be `f`.

backplane
: Specifies the backplane type. Valid backplane types are 1x1, 2x2, 4x4, and 8x8.

memory_type_codes
: Specifies single digit hexadecimal codes, which are read from a sticker on the face place of each memory module. Valid codes are 0, 1, 2, 3, 4, 5, 8, 9, a, b, c, and d. The codes entered are ordered starting from memory module 0. (See Example 2 in the EXAMPLES section.)

The `.cfg` files contain all configuration data needed by CRAY J90 system configuration registers, which are contained in all CRAY J90 system ASICS' TAP controllers. The `mc` and `ds` commands read the `.cfg` files and shift the configuration data in to the system's TAP controllers.

When run in the default mode (no command line options), `jconfig` performs the following sequence:

1. Resets the CRAY J90 series system, so that it is able to do maintenance functions.
2. Reads the CRAY J90 series system to determine which slots contain modules.
3. Attempts to read an existing `.cfg` file to extract nonhardware-readable information (backplane type and memory module type codes).
4. If either step 2 or step 3 fails, the user can enter slot, backplane, and memory module type codes using an information screen that can be edited by using an editor like vi.
5. Displays main menu.

From this point, the user can edit the configuration data, view the current hardware configuration, update the system's `.cfg` files, or dump the `.cfg` files in ASCII format.

Startup

The `jconfig` utility requires the following hardware configuration information: which CPU slots have processor modules in them, which memory slots have memory modules in them, how many CPUs per Processor module, the backplane type, and memory module type codes. The CPU/memory slot information, and CPUs per processor module are readable through the maintenance channel. The backplane type and memory module type codes are not.

When invoked without -hwconfig or -bpmt, jconfig attempts to read the slot configuration. Then a search is made for a .cfg file. If a file is found, its header is read which contains, among other things, the backplane type and memory module type codes for all memory modules.

If either of these operations fails, jconfig informs you, and displays the following information screen, which you can edit to fit the actual hardware configuration:

```
CP Boards/CPUs: 1 HEX Digit Per Board, 1 Bit Per CPU On That CP.
                Example:  0000ffff  (CP 0-3, all cpus present/CP)
                          00000001  (CP 0, cpu0 only present)

MEMORY Boards:  1 Digit Per Board.  1 == Mem. Board Present, 0 == Not Present

Memory Type:    1 Digit Per Memory Board.  Get Type Code From Memory Board
                Sticker.  Valid Type Codes Are 0-5 and 8-d (hexadecimal).

Backplane Type: 8==8x8, 4==4x4, 2==2x2, 1==1x1
================================================================================

CP Boards/CPUs:  00000000  (Leftmost Digit is CP  Board 7, Rightmost is CP 0)
MEMORY Boards:   00000000  (Leftmost Digit is MEM Board 7, Rightmost is MEM 0)
Memory Type:     00000000  (Leftmost Digit is MEM Board 7, Rightmost is MEM 0)
Backplane Type:  00000000  (8==8x8, 4==4x4, 2==2x2, 1==1x1)


<h,l,k,j,CR,p> Left,Right,Up,Down,Next Line,Page   <z> Save   <ESC> Discard
                                                              [Page 1 of 1]
```

When the information is entered, using vi-like commands (see the Editing subsection), z is entered to save the configuration. At this point, jconfig asks the user for verification of what was entered. Then the main menu is displayed.

If no errors are encountered during hardware configuration sensing, jconfig asks for verification of the hardware configuration, and then displays the main menu.

Menus

The jconfig utility, invoked without the -nocr option, determines the hardware configuration and presents a menu interface. These menus let you select the type and scope of ASIC configuration register fields, which you can edit (see the Editing subsection). The following is an example of the main menu:

```
==============================MAIN_MENU==============================

<1> Edit Diagnostic Parameters
<2> Edit ALL Parameters
<3> View System Configuration
<4> Update Config File(s)
<5> Dump Configuration to ASCII File
<6> Exit

Enter # Of Choice:
```

Editing

The `jconfig` utility, invoked without the `-nocr` option, lets the user edit all fields of all ASIC configuration registers. These fields are grouped into two types:

Diagnostic — Fields that may help diagnose problems on a CRAY J90 system, or that may alter the way the CRAY J90 system runs, for example, disable SECDED.

All — Any and all configuration fields.

These fields also are grouped by scope:

System — The user can set certain fields system-wide.

Module — The user can set fields module-wide.

ASIC — The user can set fields for 1 specific ASIC, all of an ASIC type on a module, or all of an ASIC type system-wide.

Usually, the system-wide diagnostic settings are the most appropriate to use.

The edit screens display each configuration register field, an explanation of the field, and how many bits it occupies. All fields are expressed as 8-digit hexadecimal numbers. A footer at the bottom of the screen has editing instructions, the current field type and scope, and the current edit page number. The following is an example:

```
Disable Error Correction (SECDED) (1==Disable)          1  Bit:  00000000
Set Maintenance Mode System Wide  (1==Set)              1  Bit:  00000000
Disable Scalar Cache System Wide  (1==Disable)          1  Bit:  00000000
Memory Degraded (Odd or Even Sects Only,1==Degraded)    1  Bit:  00000000
Even Sections Only Or Odd Sections Only (1==even)       1  Bit:  00000000
Physical CPU # of Logical CPU 0 (0-1f)                  5 bits:  00000000
Disable 005 During 034, System-Wide PC's (1==disable)   1  Bit:  00000001
Disable 024 During 036, System-Wide PC's (1==disable)   1  Bit:  00000001
Wait On Data During 073/076, System-Wide PC's           1  Bit:  00000000
Allow 1 Instr. To VU At A Time, System-Wide PC's        1  Bit:  00000000
Allow Only 1 Port Active At A Time, System-Wide PC's    1  Bit:  00000000
Disable Instr. Chaining, System-Wide VU's (1==disable)  1  Bit:  00000000
Disable Instr. Tailgating,System-Wide VU's(1==disable)  1  Bit:  00000000
Disable 1 CP Bypass, System-Wide VU's (1==disable)      1  Bit:  00000000
Disable 2 CP Bypass, System-Wide VU's (1==disable)      1  Bit:  00000000
Logical CPU Board Number, Physical CPU Board 0 (0-7)    3 Bits:  00000000
Logical CPU Board Number, Physical CPU Board 1 (0-7)    3 Bits:  00000001
Logical CPU Board Number, Physical CPU Board 2 (0-7)    3 Bits:  00000002
Logical CPU Board Number, Physical CPU Board 3 (0-7)    3 Bits:  00000003
Logical CPU Board Number, Physical CPU Board 4 (0-7)    3 Bits:  No Board

<h,l,k,j,CR,p> Left,Right,Up,Down,Next Line,Page   <z> Save  <ESC> Discard
PARAMETER GROUP: Diagnostic System Level                      [Page 1 of 2]
```

The cursor may be moved with the same cursor keys as the `vi` editor. Pressing `z` causes the edit(s) to be saved, after which the main menu is displayed. To discard edits, hit <ESC>.

Saving and Updating `.cfg` Files

The `jconfig` utility lets the user save the current configuration into `.cfg` files using the "Update Config File(s)" option of the main menu.

The `.cfg` files are saved automatically if `-nocr` is specified.

ASCII versions of all `.cfg` files are saved if the "Dump Configuration to ASCII File" option of the main menu is used. Also, `-dump` can be specified if an inline parameter (`-ecc`, `-cache`, etc.) or `-hwconfig` or `-bpmt` is specified.

NOTES

The `jconfig` utility determines which environment is currently in place, and can be run either from the `QLOAD>` prompt (`load -q ...`) or from the `IOS>` prompt. It cannot be run from the `BOOT>` prompt.

RETURN VALUES

The `jconfig` utility returns 0 to the calling environment if there are no errors and nonzero if an error occurs.

MESSAGES

Error messages are generated if `jconfig` has trouble during the sensing of hardware configuration, or if errors are encountered during the opening, reading, or writing of files. All error messages start with `jconfig:`. Fatal errors result in messages starting with `jconfig: Aborting`.

EXAMPLES

Example 1: The following example specifies a 4x4 backplane, and type code 0 for memory module 0, 1, and 3, and a type code of 1 for memory module 2. If this option is specified with `-nocr`, `jconfig` runs without user interaction, but still performs a hardware check.

```
jconfig -bpmt 4x4 0 0 1 0
```

Example 2: This example specifies 4 processor modules with 4 CPUs on each one, 4 memory modules, a 4x4 backplane, memory module type codes 0 for memory modules 0, 1, and 3, and memory module type code 1 for memory module 2.

```
jconfig -hwconfig ffff f 4x4 0 0 1 0
```

FILES

`/sys/pm[0-7].cfg`	Module-level processor module configuration files that contain all ASIC JTAG configuration register fields for that module.
`/sys/mem[0-7].cfg`	Module-level memory module configuration files that contain all ASIC JTAG configuration register fields for that module.
`/sys/pm[0-7].cfg.txt`	ASCII version of `pm[0-7].cfg`.
`/sys/mem[0-7].cfg.txt`	ASCII version of `mem[0-7].cfg`.

SEE ALSO

ds(8), mc(8)

NAME

`jobs` – Displays user commands that are running

SYNOPSIS

```
jobs
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `jobs` command displays all currently running user commands by name and task ID.

NAME

`kill` – Kills a user command task

SYNOPSIS

`kill` *tid*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `kill` command looks for the user command task *tid* and terminates it if it is running. To obtain the task ID *tid*, use the `jobs` command, which displays the currently running commands. To kill a command running in the foreground, press <CONTROL-c>.

This command accepts the following argument:

tid Task ID; integer task identifier.

SEE ALSO

`jobs`(8) to display currently running commands

NAME

`ld` – Loads a file into central memory

SYNOPSIS

`ld` *filename*[*.ext*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

If you do not specify an extension, the `ld` command, by default, searches the current directory for the file name that has a `.bin` extension. If the `.bin` extension search is unsuccessful, the file name that has an `.xxx` extension is searched for next. If the file is not found, an error message is displayed.

The `ld` command accepts the following options:

filename Specifies the name of file that is loaded.

.ext Specifies an optional file extension.

MESSAGES

`Unable to load file <`*filename*`>`
The file specified could not be loaded into central memory.

NAME

lm – Loads central memory

SYNOPSIS

lm b*cd sa cma word_count*

lm i*cd sa cma word_count*

lm s*cd sa cma word_count*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The lm command transfers data from the specified system disk to central memory by using the I/O channel. The data is transferred into central memory through the data channel.

The lm command accepts the following options:

b	Indicates a buffered intelligent peripheral interface (IPI) drive.
i	Indicates IPI drive.
s	Indicates a small computer system interface (SCSI) drive.
c	Controller number (0 to F).
d	Indicates disk (0 to F).
sa	Starting logical sector address at which data begins on system disk.
cma	Starting central memory word address at which the data will be written.
word_count	Specifies the number of 64-bit words to write to central memory.

NOTES

Because this command accesses central memory through the data channels, the CPU clock must be on.

The sector address specified in these commands assumes a sector length of 4096 bytes. The system console must compute the correct physical sector address based on the disk type.

For the commands that write data to the disk, if the *word_count* argument is not a multiple of 512 64-bit words, the data read into the rest of the last sector will be unpredictable.

MESSAGES

lm: open failed on device <*device name*>
: An attempt to open the specified device failed.

lm: second parameter isn't a sector address
: The second parameter specified on the command line is not a valid sector address.

lm: third parameter isn't central mem addr
: The third parameter specified on the command line is not a valid central memory address.

lm: fourth parameter isn't word count
: The fourth parameter specified on the command line is not a valid integer.

lm: read of device failed: <*device name*>
: The read of the specified device failed.

`lm: write to memory failed`
An attempt to write the data from disk to central memory failed.

EXAMPLES

The following command transfers 1.3 million words of data from the SCSI disk on controller 2, unit 1, at the hexadecimal sector address 53BE to central memory address 100:

```
lm  s21 0x53BE 0x100 1300000
```

SEE ALSO

dm(8) to display central memory
sa(8) to save central memory to a binary file

NAME

`load` – Loads and boots an IOS binary image into the IOP

SYNOPSIS

`load [-n] [-q] [filename]`

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `load` command simply loads in an image that can be booted into the I/O processor (IOP) memory and tries to boot from it. It accepts either a file or a device name as input.

The `load` command accepts the following options and arguments:

–n Loads in the image, but it does not try to boot it.

`-q` Specifies a quick load of the IOS kernel to enable limited IOP or IOS maintenance to be performed. No drivers or strategies are loaded, and a reload is required to boot the operating system.

filename Specifies input file.

MESSAGES

`load: only executable from prom!`
An attempt has been made to issue the `load` command from anywhere besides PROM.

`load: open failed on <target name>`
The load of the input file specified failed.

`load: read failed on <filename> (got <num bytes> bytes); aborting!`
The file to load is not the correct size.

EXAMPLES

Example 1: The following example boots a back-up copy of the IOS kernel from `/tmp`:

```
load /tmp/ios.bak
```

Example 2: The following example boots the default IOS:

```
load
```

SEE ALSO

`lu`(8) to load a binary UNICOS image into central memory
`reload`(8) to reload an IOS from a running IOS
`reset`(8) to stop the IOS kernel

NAME

`ls` – Lists a directory

SYNOPSIS

`ls [-l] [-R] [`*dir*`] [`*filename*`[`*.ext*`]]`

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `ls` command lists either all directory entries or only those for specified files.

The `ls` command accepts the following options and arguments:

`-l`	Displays long listing, including type of file, time stamp, and number of bytes in file.
`-R`	Recursively lists subdirectories encountered.
dir	Specifies the path of the directory that will be listed. The default is the current directory.
filename	Specifies file(s) that will be listed. By default, all files are listed.
.ext	Specifies the file extension.

MESSAGES

`ls: error getting full path name`
An attempt to obtain the full path name of the file or directory failed.

`ls: %s: no such file or directory`
The file or directory to display does not exist.

`ls: Bad status on <`*filename*`>`
The file specified does not exist or for some other reason statistics on the file cannot be obtained.

NAME

lu – Loads UNICOS

SYNOPSIS

lu *file1 file2*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The lu command loads a UNICOS a.out file and a configuration file into central memory from the IOS file system in preparation for booting UNICOS. It is usually run from the /bin/boot script.

The lu command accepts the following arguments:

file1 Specifies UNICOS file.

file2 Specifies configuration file.

NOTES

In previous releases, a UNICOS kernel in a.out format was installed on a system disk that has a compatible configuration file.

MESSAGES

lu: mem open failed
An attempt to open central memory failed.

lu: open of file %s failed
An attempt to open the file to load into central memory failed.

lu: write of first file to memory failed: %s
An attempt to write the file specified into central memory failed.

lu: write of second file to mem failed: %s
An attempt to write the file specified into central memory failed.

lu: write of S4 to memory failed: %s
An attemp to set up register S4 failed.

lu: write of S7 to memory failed: %s
An attempt to set up register S7 failed.

EXAMPLES

The following command line loads the unicos and cfg files:

```
lu unicos cfg
```

The unicos file is loaded into central memory starting at address 0, with the a.out file header removed. The cfg file is loaded into central memory at the next available address following the unicos file. The first 16 addresses in central memory contain an exchange package, which is ready for execution after the unicos and cfg files are loaded.

The `lu` command modifies the S4 and S7 registers of the loaded exchange package to enumerate the last word address (LWA) of the `unicos` and `cfg` files in memory correctly.

After executing an `lu` command with valid files successfully, you can restart the CPU by using the `ds` command.

SEE ALSO

`iostart`(8) to initiate communication between UNICOS and the IOS

NAME

mc – Stops all CPU activity

SYNOPSIS

mc

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The mc command performs the initialization function for the CPU and central memory.

MESSAGES

`mc: CPUMC ioctl failed on IOBB`
An attempt to reset the master IOS CPU failed.

`mc: ENMEM ioctl failed on IOBB`
An attempt to enable memory access to the IOBB driver failed.

`mc failed, unknown arguments`
The option specified for mc is invalid. Only the diagnostics should use these options.

`mc: open failed on IOBB device`
An attempt to open central memory failed.

`mc: RESET ioctl failed on IOBB`
An attempt to issue the `RESET ioctl` call failed.

`mc: ymp_mc failed`
The command issued to clear the CPU and memory boards failed.

SEE ALSO

`iostart`(8) to initiate packet communication to UNICOS
`sc`(8) to reset all CPUs
`stat`(8) to display CPU status

NAME

mfdump – Dumps mainframe memory

SYNOPSIS

```
mfdump [-c] [-f] [-q] [-r] [-v]
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The mfdump command dumps mainframe memory to the dump device.

The mfdump command accepts the following options:

-c Checks parameters. Displays dump parameters, but it does not perform the dump.

-f Forces the dump, even if a dump already exists in the dump device.

-q Queries operator for dump parameters (see the EXAMPLES section).

-r Queries operator for the reason for the dump. The reason string cannot contain a semicolon.

-v Displays verbose internal debugging information, including dump parameters.

Dump Area Format

The dump area consists of a one-sector header that describes the disk allocation, followed by the data area.

The mkdump(8) command initializes the header. The header contains status information and describes the disk space allocated for dump data. The format of the header is as follows:

```
struct  shead   {
        word            sh_idu;
        word            sh_idl;
        struct  spart   sh_part[SH_PART];
        uint                    :32;
        uint                    :32;
        uint            sh_flag :16,
                        sh_cpus :16;
        uint                    :15,
                        sh_memh :1,
                        sh_num  :16;
};
```

When a dump is performed, the IOS fills in the sh_flag field. The cpdmp command uses the sh_flag to determine whether a dump has been taken and processed.

sh_id[ul] Dump area identifier. The valid header is *UNICOS*.

sh_part Partition descriptors; 509 entries that describe the disk allocation. An entry of 0 marks the end of the list. Only the first element of sh_part is used. It contains the total length (in sectors) of the system dump.

sh_flag Flag that is set to the system type by the IOS when a dump is taken. The cpdmp command sets the flag to CP after it copies the dump data to a system dump file.

The other fields are not used.

Dump File Format

The cpdmp command copies a system dump from the dump area to a file. To display the information in the system dump file, use the crash(8) or fdmp(8) command.

Param File Format

The mfdump command assumes the existence of the /sys/mfdumpa.arg file, which contains an ASCII description of the hardware configuration and the dump partition. You can edit this file by using the ed text editor. This file is initially set up during the install process. The format of this file is as follows:

CPUS=<*number of cpus*>
MEM=<*size of memory in megawords*>
range1=<*value*>-<*value*>
range2=<*value*>-<*value*>
range3=<*value*>-<*value*>
range4=<*value*>-<*value*>
regdump=yes|no
sysreg=yes|no
ublocks=yes|no
IOS#=<*number of the channel that has the dump partition*>
channel#=<*number of the channel that has the dump partition*>
disktype=<*disk type of the disk that has the dump partition*>
controller=<*controller number of the device that has the dump partition*>
unit=<*unit number of the device that has the dump partition*>
start=<*starting block number of the dump partition*>
length=<*length in blocks of the dump partition*>

A line beginning with # indicates a comment. White space is allowed between tokens, for example, CPUS = 1, but not within tokens. All values are decimal. The range values specify a memory range to dump. If the end range is 0, that range is not dumped. The disk type can be an integer value or one of the following mnemonics:

DD4
RD1
DDIMEM
DD5S
DD5I
DD_U
DD6S
DD7S
DD8S

The contents of a sample `mfdumpa.arg` file is as follows:

```
#comment
CPUS=4
MEM=512
range1=0-8000000
range2=0-0
range3=0-0
range4=0-0
regdump=yes
sysreg=yes
ublocks=yes
IOS#=0
channel#=16
disktype=DD5I
controller=10
unit=1
start=40000
length  =   65536
```

MDW File Description

A system dump file contains the following: a header, memory descriptors, and data area.

The header format is as follows:

```
struct  fhdr1   {
        word    fh_didu;
        word    fh_didl;
        char    fh_res[80];
        char    fh_dat[8];
        char    fh_tim[8];
        word    fh_hrp2u;
        word    fh_hrp21;
        word    pad3;
        word    fh_nmdw;
        struct  mdw1 fh_mdw1[NMDW0];
        word    pad4;
        long    fh_nxtsec;
        uint    fh_ftype:32;
        uint    fh_nxtfile:32;
};
```

`fh_did[ul]` System dump file identifier (the ASCII string `SYSDUMP`).

`fh_res` Reason the dump was taken; 80 ASCII characters.

`fh_dat` Date the dump file was created (in the format *mm/dd/yy*).

`fh_tim` Time the dump file was created (in the format *hh:mm:ss*).

`fh_hrp[ul]` File format; the ASCII string `*DUMP02*`.

`fh_nmdw` Number of memory descriptor words (MDWs). The initial 165 MDWs are allocated in the header. If additional entries are required, additional sectors are allocated; each sector has 170 MDWs and is linked to the next sector by `fh_nxtsec`.

`fh_mdw1` Array of MDWs.

`fh_nxtsec` Address of the sector that contain the next array of MDWs.

The other fields are not used.

The MDW describes the data that has been dumped and its location in the system dump file. The format of the MDW is as follows:

```
struct  mdw1    {
        uint    md_comp :1,
                md_typ  :15,
                dummy1  :16;
        uint    md_fwa  :32;
        uint    md_for  :1,
                md_sfor :1,
                md_file :5,
                        :9,
                        :16;
        uint    md_lwa  :32;
        uint    md_sc   :32;
        uint    md_sa   :32;
};
```

`md_typ` Type of data contained in this area; for a list of types, see the `sys/sdmp.h` file.

`mh_fwa` Address of origin for the data described by this MDW.

`mh_lwa` Address of origin of the last word of data.

`md_sc` Number of sectors of data contained in the system dump file.

`md_sa` Sector address of the data in the system dump file. The first two words of the sector are reserved and are not included in `md_sa`. The actual data begins after these two words.

The other fields are not used.

NOTES

The `mfdump` program executes in IOS 0. It checks for the existence of the dump disk device. If the dump disk device is on an IOS other than 0, that IOS is checked to make sure that it is running and communicating with the CPU.

The reason string entered with the `-r` option cannot contain a semicolon.

MESSAGES

The following is a list of error messages and their explanations (if necessary):

`IOS 0 is selected but channel is not 16. Enter the IOS number (0-31):`
This message is issued when the `mfdump` parameter file `mfdump.arg` has a value of 0 for the IOS number, and the channel number selected is not 16. The IOS number entered must correspond to the channel number.

`IOS` *n* `unavailable for dump to disk. Aborting.`
This message is issued when the dump disk device is on an IOS other than 0 according to the dump parameters, and the IOS is not responding to messages on the IOS network. The IOS *n* may be down, or the IOS network may be hung up. In this instance, a reboot of the IOS is needed.

`Cannot open/creat /sys/mfdump.arg. Dump config file cannot be saved.`
This message is issued when an open call fails. This indicates system console disk problems (for example, the disk is full, or a catastrophic problem exists in IOS 0). You should check the IOS file system on the system console disk for proper operation and available space.

`Write to /sys/mfdump.arg failed. Dump config file cannot be saved.`
This message is issued when a write call is made to the IOS maintenance disk. This may indicate IOS disk problems (for example, the disk is full, or a catastrophic problem exists in IOS 0). You should check the IOS file system on the disk for proper operation and available space.

`Cannot open /sys/mfdump.arg. Dump failed.`
This message is issued when an open call is made to the IOS maintenance disk to read the `mfdump.arg` file. The file may not exist; in which case, `mfdump` can be reexecuted in query mode to re-create the file.

`Read of /sys/mfdump.arg failed. Dump failed.`
This message is issued when a read call is made to the IOS maintenance disk. This may indicate IOS disk problems or a catastrophic problem in IOS 0. The `/sys/mfdump.arg` file may be bad on disk, or the IOS file system on the disk may be faulty. To remedy this problem, execute the `mfdump` command in query mode and re-create the `mfdump.arg` file.

`Cannot open memory. Dump failed.`
This message is issued when the `mfdump` command attempts an open call to the `iobb` device driver and fails. This occurs when the `iobb` is in an uninitialized state. The `mfdump` command always executes an `sc` command to the CPU, which also reinitializes the `iobb` device driver. Therefore, the failure of the open call implies that the `sc` command did not initialize the `iobb` device driver, or since that time a catastrophic event occurred that caused the `iobb` device driver to shut down.

`CPU dump truncated.`
This message is a returned error from the CPU dump program. It means that the dump partition is full.

`Invalid dump header mainframe dump failed.`
The header in the dump partition was not a legitimate dump header. Either the dump header was not initialized by using the `idmp` command or the dump parameters are in error and are directing `mfdump` to the wrong disk address.

`There is a system dump in the dump partition that has not been copied.`
`Use -F to force a dump.`
`Mainframe dump failed.`
Self-explanatory.

`IOS` *n* `is no longer responding on the IOS network.`
This message occurs during the start-up sequence in a multiple-IOS configuration, and the slave IOS does not respond on the IOS network. It indicates a problem in the slave IOS or the IOS network. To continue, reboot the IOS.

`Cannot determine disk config on IOS` *n*`. The IOS may be down.`
This message occurs during the start-up sequence in a multiple-IOS configuration, and the slave IOS is responding on the IOS network, but the return code from the slave function call is bad. This indicates a problem in the slave IOS. Perform an `iosdump`(8) of the slave and master IOS and take the output to your system support for analysis.

`IOS` *n* `Controller` *x* `Unit` *y* `not available.`
This message occurs when the master IOS tries to probe for the existence of the dump disk device, in accordance with the `mfdump.arg` parameters or the query parameters. The configuration parameters may not agree with the physical layout, or the controller or disk is not available due to a fatal error condition. Recheck the configuration or test the devices that will be available to the IOS.

`Y1 channel seems to be non-functional`
`Suggest reloading IOS and trying again.`
The Y1 channel did not respond after being reset. A hardware problem may exist. Reload the IOS and try again. If it still fails call for hardware support.

`Couldn't get registers for cpu <n>`
The `mfdump` command was unsuccessful in retrieving the B, T, and V registers for the specified CPU. The dump will continue, but these registers will not be available in the dump.

`Could not get cluster registers`
The `mfdump` command was unsuccessful in retrieving the cluster registers. The dump will continue, but these registers will not be available in the dump.

`Cannot access CPU memory. Trying 'SC' to clear the channel.`
CPU memory was inaccessible. `mfdump` tries a "soft clear" (issues the `sc` command) to clear the channel. If that still doesn't work, the dump will abort. If it works, the dump will continue, but the register and exchange package information in the dump may not be meaningful.

`mfdump: fwa: mdw's out of sync, <value> != <value>`
`mfdump: lwa: mdw's out of sync, <value> != <value>`
`mfdump: type: mdw's out of sync, <value> != <value>`
`mfdump: sc: mdw's out of sync, <value> != <value>`
An internal error exists in `mfdump`. Call your Cray Research representative for assistance.

EXAMPLES

Example 1: The following example dumps the mainframe and puts the string `System Paniced on an ORE in user code` in the "reason" field of the dump header (user input is in Courier bold):

```
IOS> mfdump -q -v System Paniced on an ORE in user code
```

Example 2: The following example dumps the mainframe and puts the string `System Paniced on an ORE in user code` in the "reason" field of the dump header (user input is in courier bold):

```
IOS> mfdump -q -v -r
mfdump: Enter reason for dump. Terminate reason with
two new line characters.
System paniced on an ORE in user code
```

FILES

`/sys/mfdump.arg`	File in which the dump parameters are stored

SEE ALSO

`dstat`(8) to display disk status
`stat`(8) to display CPU status

NAME

`mkdir` – Makes a new directory

SYNOPSIS

`mkdir` *dirname*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `mkdir` command creates a new directory on the NFS mounted file system.

The `mkdir` command accepts the following argument:

dirname Name of the new directory.

EXAMPLES

Example 1: To create a new directory called `test5` in the `results` subdirectory under the root directory, enter the following command line:

```
mkdir results/test5
```

Example 2: To change the directory to the `results` directory, use the `cd` command and enter the following command line:

```
mkdir test5
```

NAME

`mm1test` – Executes a confidence test on the IOP RAM/CACHE memory

SYNOPSIS

```
mm1test
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `mm1test` confidence test consists of the following:

1. Write, read, and verify 85 canned and 10 random data patterns, using all available RAM memory.
2. Verify partial addressing of the available RAM by writing, reading, and verifying a data pattern of address to address.
3. Repeat the preceding sequence by using the combination of RAM/CACHE.

NOTES

The `mm1test` command runs only on the CRAY J90 series systems.

SEE ALSO

CRAY J90 Series IOS Based Tests, publication HDM–099–0 (This manual is Cray Research Proprietary; dissemination of this documentation to non-CRI personnel requires approval from the appropriate vice president and a nondisclosure agreement. Export of technical information in this category may require a Letter of Assurance.)

NAME

more – Displays a file one screen at a time

SYNOPSIS

more *filename*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The more command outputs one screenful at a time of a specified file. To view the next screen, press any key on the keyboard, except the <q> key.

The more command accepts the following operand:

filename Specifies the name of the file to be viewed.

When you press the <q> key, the more command quits.

MESSAGES

more: unable to open *<dir name>*
An attempt to open the directory name specified failed.

EXAMPLES

The following example displays the aa file:

```
more aa
```

NAME

mt – Controls magnetic tape

SYNOPSIS

mt [-f *tape_dev*] *command* [*count*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The mt command performs certain functions on specified magnetic tape devices (rpd03 by default).

The mt command accepts the following options:

-f *tape_dev* Specifies the device to be activated (for example, rpd03, hrpd03).

command Specifies the command to execute on the tape device. Valid commands are as follows:

- bsf [*count*] Skips back over *count* file marks; the default is 1.
- fsf [*count*] Skips forward over *count* file marks; the default is 1.
- reten Retensions the tape.
- rewind Rewinds the tape.
- status Displays drive status.

count Specifies the number of files to skip over. This argument is valid only with the fsf argument.

RETURN VALUES

If the operation was successful, mt returns 0; otherwise, it returns –1.

MESSAGES

mt: <*command*> command not found
The options specified on the command line were not valid.

mt: device <*tape*> is not a tape device!
The tape device specified is not a valid device name.

mt: negative repeat count
A negative count for the number of file marks to forward or backward skip on tape was entered on the command line.

mt: open fails for <*tape*>
The tape device specified does not exist.

mt: status ioctl failed : <*device name*>
The issuing of an ioctl call to the specified tape device failed.

mt: Unknown drive type <*drive type*>
The tape device specified is not valid.

EXAMPLES

Example 1: The following two commands are identical; they rewind device `rpd03`:

```
mt rewind
mt -f rpd03 rewind
```

Example 2: The following commands position the tape after the first file:

```
mt -f rpd03 rewind
mt -f nrpd03 fsf 1
```

NAME

mv – Moves (renames) a file or directory

SYNOPSIS

mv *files target*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

You can use the mv utility to perform any of the following actions:

- Move (rename) one file. If the name of the new file exists, it will be overwritten.
- Move one or more files from a directory to another existing directory.
- Rename a directory.
- Move a directory.

To move one or more files, specify the current file name(s) (*files*) and the new name for the file (*target*). Do not use the same name for *file* and *target*. If *target* is not a directory, you can specify only one file before it; if it is a directory, you can specify more than one file.

If *target* does not exist, mv creates a file named *target*. If *target* exists and is not a directory, its contents are overwritten. If *target* exists and is a directory, *files* are moved to that directory.

When moving a directory, if *target* exists, mv will abort with the `Destination name already exists` message. If *target* does not exist, a new directory is created and all files and subdirectories in *files* are moved to *target*.

MESSAGES

`mv: cannot change back to` *<directory name>*
: An attempt to change back to the parent directory failed.

`mv: cannot change back to parent directory`
: An attempt to change back to the root directory failed.

`mv: Destination name already exists`
: The destination file or directory name specified already exists.

`mv: potential recursive copy - aborting`
: In moving a directory from one name to another, it was detected that the command may be in a recursive loop.

`mv: Source name does not exist`
: The target file or directory specified to move does not exist.

EXAMPLES

The following example moves (or renames) file `a` to file `b`:

```
mv  a  b
```

NAME

`nettest` – Executes a network controller confidence test

SYNOPSIS

```
nettest
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `nettest` command lets users run and control an offline network controller confidence test.

WARNINGS

When the operating system is active, you cannot execute the `nettest` command. It must be run from the IOS prompt.

SEE ALSO

UNICOS Administrator Commands Reference Manual, publication SR–2022, for additional UNICOS diagnostic commands

CRAY J90 Series IOS Based Tests, publication HDM–099–0 (This manual is Cray Research Proprietary; dissemination of this documentation to non-CRI personnel requires approval from the appropriate vice president and a nondisclosure agreement. Export of technical information in this category may require a Letter of Assurance.)

NAME

od – Displays a file by using various formats

SYNOPSIS

od [-d] [-n *num_lines*] *filename* [*offset*]

od [-h] [-n *num_lines*] *filename* [*offset*]

od [-o] [-n *num_lines*] *filename* [*offset*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The od command displays the contents of a file. The contents are formatted according to the specified numeric base parameter (-d, -x, or -o). If you omit the first argument, -x (hexadecimal) is the default. Dumping continues until an end-of-file (EOF) is reached.

The od command accepts the following options:

-d Interpets bytes as decimal.

-h Interpets bytes as hexadecimal (default).

-o Interpets bytes as octal.

-n *num_lines*
Specifies how many lines to output; *num_lines* is a numeric value.

filename Specifies the name of either a file on the IOS disk or one of the following keywords to indicate the appropriate memory:

/dev/iobb	IOBB memory
/dev/iop	IOP memory
/dev/mem	Central memory

offset Specifies number of bytes to index into the file before outputting.

EXAMPLES

Example 1: The following example displays the contents of file some.file in hexadecimal format:

```
od some.file
```

Example 2: The following example displays the first 10 lines of file some.file:

```
od -n 10 some.file
```

Example 3: The following example displays 30 lines of central memory in octal format:

```
od -on30 /dev/mem mem_address
```

Example 4: The following example displays IOBB memory:

```
od /dev/iobb mem_address
```

Example 5: The following example displays IOP memory:

```
od /dev/iop mem_address
```

NAME

`offline` – Loads and configures an offline mainframe diagnostic

SYNOPSIS

`offline [-b #] [-c #] [-d] [-k` *monitor*`] [-l #] [-m #] [-n #] [-s #]` *filename*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `offline` diagnostic loads, configures, and attaches the appropriate monitor to the offline diagnostic specified. Default configuration parameters are extracted from the IOS `CONFIG` file and set accordingly.

The `offline` diagnostic accepts the following options:

`-b #` Specifies banks of memory to test. You can set # to be in the range of 02000-0 octal.

`-c #` Specifies an octal bit mask selection of CPUs to test (CPUS). You can set # for a two-CPU system, as follows:

- 1 Test CPU 0 only.
- 2 Test CPU 1 only.
- 3 Test both CPU 0 and 1.

`-d` Disables scalar cache for all applicable diagnostics.

`-k` *monitor*
Specifies the monitor type, which can be one of the following:

```
none
ymm
yms
ymi
ysmi
ym8
```

`-l #` Specifies an octal number of clusters to test.

`-m #` Specifies central memory size (in megawords). For example, # is 32 for a 32-Mword system.

`-n #` Specifies an octal bitmask selection of physical CPUs configured (CPUN). The population of this parameter is used to partion and allocate memory for each available CPU.

`-s #` Specifies an octal bit mask section of a diagnostic.

filename Specifies an offline mainframe diagnostic to load. The `.bin` extension is appended automatically to the file name.

NOTES

Options are necessary only when an override of the default parameters is desired.

Observe the following limitations when running the `offline` diagnostic:

- When the operating system is active, the `offline` diagnostic does not run.
- The `offline` diagnostic runs only on CRAY J90 series systems.

EXAMPLES

The following command loads JSR3. The offline diagnostic configures it to test CPUs 0 and 7 that have a monitor type of YM8, 512 MWDs of memory, and 11 octal clusters. It selects three sections of the test to run. Memory is divided by the population count of the -n # value, which in this case is 2.

```
offline -c 201 -n 201 -k ym8 -m 512 -l 11 -s 7 jsr3
```

NAME

pwd – Prints current directory

SYNOPSIS

pwd

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The pwd command prints the path name of the working (current) directory.

NAME

`readlog` – Reads the statistical log data on an STK 4280 tape drive

SYNOPSIS

`readlog rss`*CUL* `[-il] [-f` *file*`]`

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `readlog` command reads the statistical log data on an STK 4280 tape drive. It either writes the data to a file or displays it to the screen.

The `readlog` command accepts the following options:

C Specifies controller number.

U Specifies the unit number.

L Specifies the logical unit (LUN).

`-i` Lists the improved cartridge recording capability (ICRC) format, which means that data compression and compaction are supported.

`-l` Lists to screen.

`-f` *file* Writes the log data to *file* (default is `/adm/read.log`).

NOTES

This command does not return correct data unless the drive is online and ready (tape loaded).

MESSAGES

`readlog: Cannot open tape <`*device name*`>`
An attempt to open the specified tape device failed.

`readlog: Invalid tape device name`
The device name specified is not valid. It must be of type STK 4280.

`readlog: This device does not support the '-' format`
The option `-i` was specified, but it is not supported for the device specified.

`readlog: Unable to execute log command`
Execution of the log command failed.

`readlog: Unable to open output file <`*filename*`>`
The creation of the log file failed.

`readlog: Writing log data to <`*filename*`>`
Writing to the log data file failed.

EXAMPLES

The following command reads from drive `rss010` and writes to the default file:

```
readlog rss010
```

NAME

`reload` – Initiates the reboot of the IOS

SYNOPSIS

`reload` [*filename*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `reload` command renames the specified file `/reboot`. The `reload` command then resets the VME, which results in a reboot of the IOP from PROM.

If you omit *filename*, `load` tries to load in the `/ios/ios` file.

The `reload` command accepts the following option:

filename Specifies the input file.

EXAMPLES

Example 1: The following command reboots a back-up copy of the IOS kernel from `/tmp`:

```
reload /tmp/ios.bak
```

Example 2: The following command reboots with the default IOS:

```
reload
```

SEE ALSO

`load`(8) to load and boot an IOS binary image into the IOP
`lu`(8) to load a UNICOS binary image into central memory
`reset`(8) to reset the IOS

NAME

`reset` – Resets the IOS

SYNOPSIS

```
reset
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `reset` command stops execution of the IOS by first flushing any buffers out to the IOS disk, and then it resets the VME bus. This returns control to PROM.

UNICOS also is stopped, and the CPU halted until another IOS kernel is booted.

SEE ALSO

`load`(8) to load an IOS kernel from a boot state
`lu`(8) to load a UNICOS binary image into central memory
`reload`(8) to initiate the reboot of the IOS from a running IOS

NAME

`rlogin` – Invokes the remote login

SYNOPSIS

`rlogin sn`*xxxx*`-IOS`*n*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

You can use the `rlogin` command to connect a system console to any IOS or from the master IOS to a slave IOS. `rlogin` is useful to run diagnostics on a slave IOS.

The `rlogin` command accepts the following option:

`sn`*xxxx*`-IOS`*n*
Specifies slave IOS to which connection is being made. The IOS name consists of the following:

`sn`*xxxx*	Specifies the mainframe serial number.
`IOS`*n*	Specifies the IOS number.

NOTES

The use of `rlogin` is the only way to execute interactive commands on a slave IOS.

EXAMPLES

The following example shows a connection being made to the slave IOS, IOS1, on machine serial number sn9005.

```
rlogin sn9005-IOS1

sn9005-ios1>
```

NAME

rm – Removes files and directories

SYNOPSIS

rm [-r] *file1* [*file2 file3* ...]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The rm command removes all files listed on the command line. Directories are removed only when you specify the -r option (to remove empty directories, see the rmdir(8) command).

The rm command accepts the following options:

-r Removes directories recursively.

file# Specifies name(s) of file(s) to be removed.

MESSAGES

rm: %s: is a directory
The file name specified to delete is a directory.

rm: %s: no such file or directory
The directory or file name specified to remove does not exist.

rm: Can't remove ROOT directory
An attempt was made to remove the ROOT (/) directory.

rm: error getting full path name
An attempt to obtain the full path name for the file to remove failed.

rm: error reading %s
While trying to remove all files in the specified directory, an error occurred reading the file name.

rm: opendir failed on %s
An attempt to open the specified file failed.

rm: removing file %s failed
An error occurred while trying to remove the specified file.

EXAMPLES

The following command removes the aa file and the /tmp/xx directory:

```
rm -r aa /tmp/xx
```

SEE ALSO

rmdir(8) to remove an empty directory

NAME

`rmdir` – Removes a directory

SYNOPSIS

`rmdir` [*path/*]*dirname*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `rmdir` command removes a directory from the hard disk. It accepts the following arguments:

path/ Specifies the path to the new directory.

dirname Specifies the name of the new directory.

NOTES

You can remove a subdirectory only if it is empty; that is, if it contains only the special entries (.) and (..).

You can remove only one subdirectory at a time.

You cannot remove the root directory and the current directory.

MESSAGES

`rmdir: cannot access <`*dirname*`>`
The directory name specified does not exist or for some other reason cannot be accessed.

EXAMPLES

To remove a directory called `test5` in the `results` subdirectory under the root directory, enter the following command:

```
rmdir results/test5
```

NAME

`script` – Executes a script of IOS commands

SYNOPSIS

`script` [*-x*] *filename*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `script` command interprets an ASCII file as IOS commands and executes each line of the specified file. If a user enters the name of an ASCII file that contains IOS commands at the `IOS>` prompt, the IOS invokes `script` automatically.

The `script` command accepts the following option and argument:

-x Debug flag; `script` prints each line it is about to execute.

filename Specifies file on which to execute `script`.

NOTES

The first line of a `script` file must contain the `#!` characters.

MESSAGES

`Error on line <line number>`
An error occurred on the specified line number of the script file.

`script: Bad magic number. Scrips must have '#!' as first two characters.`
The script does not contain the required '#!' as the first two characters.

`script: unable to open %s`
The opening of the specified script file failed.

`script: wild cards not allows in scripts`
Self explanatory.

EXAMPLES

The following command interprets the `/bin/boot` file:

```
script /bin/boot
```

NAME

`stat` – Displays the CPU and program states

SYNOPSIS

`stat` [*n*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `stat` command reads scan status for each CPU that is configured in at IOS boot time, and a composite display of all CPU statuses is presented.

The `stat` command accepts the following argument:

n Specifies the number of times the `stat` function is executed before returning the IOS prompt. If you omit *n*, `stat` executes repeatedly until you press <CONTROL-c>.

To declare the presence of known CPUs in the configuration file (`/config`), use the following keyword/value pair: `NCPUS=`*n*. You must set the `NCPUS` keyword to reflect the slot number where a CPU can be found. Valid entries range from `NCPUS=0` for one CPU found in slot 0 to `NCPUS=0123` to declare a CPU in all four possible slots.

The `ARCH` and `NCPUS` keywords are required in the `/config` file for the IOS `stat` command to work properly. For CRAY J916 systems, you must declare the `ARCH` keyword as `ARCH=J90`.

SEE ALSO

`clk`(8) to turn clock on or off
`mc`(8) to stop all CPU activity
`sc`(8) to reset all CPUs

NAME

`systat` – Outputs various IOS system-related information

SYNOPSIS

```
systat
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `systat` command displays the current status of various parts of the IOS. The display includes I/O buffer board (IOBB) buffer pool numbers, queued IOBB packets, open file descriptors, amount of IOS memory available, slave IOS information, if applicable, and so on.

The IOS network status table portion of the `systat` display gives information about your master and slave IOSs if applicable.

The "state" field of this table describes the state of the appropriate IOS. The possible states and their meanings follow:

State	Meaning
NOT AVAILABLE	The IOS has never responded.
BOOTPROM	The IOS is sitting at the boot prompt.
BOOTING	The IOS is actively booting.
LOADING	The IOS kernel is being loaded.
RUNNING	The IOS is available and running.

The "network status" field indicates whether the slave IOS has recently responded to the master IOS. The possible states and their meanings follow:

State	Meaning
active	The slave IOS is currently active.
1+ min late	The slave IOS has not responded in at least 1 minute.
5+ min late	The slave IOS has not responded in at least 5 minutes.

The `load cmd sent` field indicates whether the appropriate IOS has been issued the `load` command.

EXAMPLES

The following example shows the output of the `systat` command.

```
IOS0>systat
Buffer Pool Status:
buf size:      128   bytes, tot:           16,   free:    16,   used:        0   (reserved)
buf size:     1024   bytes, tot:            1,   free:     1,   used:        0
buf size:     3096   bytes, tot:            2,   free:     2,   used:        0   (reserved)
buf size:     4096   bytes, tot:           21,   free:    21,   used:        0
buf size:    32768   bytes, tot:            2,   free:     2,   used:        0
buf size:    49408   bytes, tot:            1,   free:     1,   used:        0   (reserved)
buf size:   131072   bytes, tot:          123,   free:   123,   used:        0
buf size:   147456   bytes, tot:            2,   free:     2,   used:        0   (reserved)

getblks:         0                  relblks:   0             waiting :   0
waited: :        0                exact fits:   0             big fits:   0

Transfers To Mainframe: 0 queued (5120 max; 0 queued IDX pkts)
Open file descriptors : 8

IOS network status:
IOS  state    network status    load cmd sent  last pkt rec'd
---  -------  ----------------  -------------  ----------------

0    RUNNING  active            Yes       TUE APR 26 17:32:55 1994
1    BOOTING  active            Yes       TUE APR 26 17:32:54 1994
```

SEE ALSO

`crash`(8) to display IOS internal information
`dstat`(8) to display disk information

NAME

`table` – Displays current status of various IOS system tables

SYNOPSIS

`table [-a]` *table_name*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `table` command lets users display various system tables. The display is an approximation because the information may have changed immediately before or during output to the screen.

If you specify `table` without arguments, the various tables will display.

The `table` command accepts the following option and argument:

`-a` Specifies all table entries. This option can create a lot of output because each entry in the table is output whether it is in use. By default, only entries that are currently in use are output (except for small tables).

table_name Specifies the name of table to be displayed (`pkt`, `fd`, or `loadmap`).

NAME

tar – Archives tape files

SYNOPSIS

tar [*key*] [*files*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The tar command saves and restores files on magnetic tape and disk files, and the *key* argument controls its actions.

The tar command accepts the following arguments:

key A string of characters that contains one function letter (c, t, or x) and possibly followed by one or more function modifiers (b, f, or v).

The *key* argument can be one of the following options:

c (Creation of a new archive) Starts writing at the beginning of the archive, rather than after the last file.

t (Table) Lists the names and other information for the specified files each time that they occur on the archive. The listing is similar to the format that the ls -l command produces. If you do not specify a *files* argument, all names on the archive are listed.

x (Extract) Extracts the specified *files* from the archive. If a specified file matches a directory whose contents was written onto the archive, this directory is (recursively) extracted. You must use the file or directory's relative path when appropriate; otherwise, tar does not find a match. The owner, modification time, and mode are restored (if possible). If you do not specify a *files* argument, the entire contents of the archive is extracted. If several files with the same name are on the archive, the last file overwrites all earlier ones.

You can use the following options in addition to the option that selects the desired function:

b (Blocking factor) Causes tar to use the block argument as the blocking factor for tape records. The default and maximum value is 20. The block size is determined automatically when reading tapes created on block special devices (keyletters x and t).

f (File) Causes tar to use the device argument as the name of the archive.

v (Verbose) Displays the name of each file it treats, preceded by the function letter. With the -t function, -v gives more information about the tape entries than just the name. Usually, tar does its work silently.

files Files or directories that will be dumped or restored. In all cases, appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

BUGS

You cannot request the *n*th occurrence of a file.

The tar command does not copy empty directories or special files.

EXAMPLES

Example 1: To extract files from the cartridge tape, enter the following command:

```
tar -xvf rpd03
```

Example 2: To extract only the `td.c` file from a cartridge tape, enter the following command:

```
tar -xvf rpd03 td.c
```

NAME

`test` – Returns value of program counter or status of flag

SYNOPSIS

```
test p
test pm
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `test` command returns the value of the CRAY J90 system program counter or the status of the PMATCHED flag.

The `test` command accepts the following options:

`p` Specifies the program counter.

`pm` Specifies the PMATCHED flag.

NOTES

The `test` command executes only in a command file.

For the PMATCHED flag, 1 equals matched, and 0 equals no match.

EXAMPLES

Example 1: The following command line returns the value of the program counter:

```
test p
```

Example 2: The returned value from Example 1 can then be used in an `if` statement following the `test` statement in a command file, as in the following example:

```
test pm

if 0 goto :notmatched

echo matched

:notmatched

echo notmatched
```

NAME

`time` – Sets and displays the real-time clock

SYNOPSIS

`time` [*mm/dd/yy hh:mm:ss*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `time` command sets and displays the real-time clock in the IOS. If you omit arguments, the system date and time are returned.

The `time` command accepts the following arguments:

mm/dd/yy Specifies month, day, and year.

hh:mm:ss Specifies hours, minutes, and seconds.

NOTES

The separator is a slash for month, day, and year; a colon separates hours, minutes, and seconds. You must specify 2 digits in all fields.

When you boot the system, the IOS real-time clock is used to set UNICOS time.

NAME

`tp1test` – Executes a confidence test on tape handlers

SYNOPSIS

```
tp1test
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `tp1test` command lets users run and control an offline tape diagnostic.

The tape handler confidence test consists of the following:

1. Write, read, and verify blocks of canned data patterns (85) and 20 random patterns.
2. Verify tape mark control.
3. Write, read, and verify random data by using preselected block length and record counts.

NOTES

Test sections 1 and 2 use both I/O processor (IOP) and I/O buffer board (IOBB) for data transfers to and from tape. The remaining test sections use only IOBB.

When running `tp1test`, observe the following limitations:

- The `tp1test` command does not run from the boot prompt.
- The `tp1test` command runs only on CRAY J90 series systems.

SEE ALSO

UNICOS Administrator Commands Reference Manual, publication SR–2022, for additional UNICOS diagnostic commands

CRAY J90 Series IOS Based Tests, publication HDM–099–0 (This manual is Cray Research Proprietary; dissemination of this documentation to non-CRI personnel requires approval from the appropriate vice president and a nondisclosure agreement. Export of technical information in this category may require a Letter of Assurance.)

NAME

`version`, `ver` – Displays version number of the IOS software or PROM firmware

SYNOPSIS

```
version
```

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

If entered from the IOS prompt, the `version` command displays the version level of the IOS you are currently running, along with the date and time stamp that indicates when it was built.

If entered from the boot prompt, the `version` command displays the version level of the IOS PROM firmware that is currently running, along with the Cray Research part number of that PROM version.

NAME

`wait` – Waits several seconds before executing next command in command buffer

SYNOPSIS

`wait` [*seconds*]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

You can use the `wait` command only in a command script file.

The `wait` command accepts the following argument:

seconds Number of seconds; default is 10.

EXAMPLES

The following command causes a 15-second `wait` before the next command executes in the command script file:

```
wait 15
```

NAME

what – Extracts SCCS version from a file

SYNOPSIS

what *filename*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The what command searches each *filename* for the Source Code Control System (SCCS) version and prints it to the standard output. The @(#) pattern is assumed to be the start of the version string.

The what command accepts the following operand:

filename Specifies file to be searched.

EXAMPLES

The program.c file contains the following:

```
char file_version[] = "@(#)file version information";
```

The program.c file was compiled to yield program.o and program, with the following command:

```
what program.c program.o program
```

This command produced the following:

```
program.c :
     file version information

program.o:
     file version information

program:
     file version information
```

NAME

whatmic – Displays microcode level(s) at the IOS prompt

SYNOPSIS

whatmic [*device*] [-s]

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The whatmic command displays one of the following:

- A file of all IOS controller and device microcode levels, including the IOS PROM firmware level.
- Microcode level of a specific device.

The whatmic command accepts the following options:

device Displays the microcode level of the specified device. whatmic rpd03 displays the microcode level of the DAT tape drive. whatmic i00 displays the microcode level of IPI drive 0.

-s Saves the /adm/mic_code.log microcode file that is built at IOS load time in the /adm/mic_code.sav file.

By default, the whatmic command displays the microcode file built at IOS load time. The file is located in the mic_code.log file in the /adm directory.

If you specify the *device* name line operand, whatmic functions the specified device for its microcode information rather than retrieving this information from the mic_code.log file. Avoid functioning a device for microcode information when that device is under the control of UNICOS.

NAME

`which` – Searches for specified file name

SYNOPSIS

`which` *filename*

IMPLEMENTATION

CRAY J90 series IOS-V

DESCRIPTION

The `which` command searches the same IOS disk directories that the IOS kernel does when it tries to locate a file name to execute. If the file is found, `which` prints out the full path to it.

Index

INDEX

Reader's Comment Form

CRAY IOS-V Commands Reference Manual SR–2170 8.0.3.2

Your reactions to this manual will help us provide you with better documentation. Please take a moment to complete the following items, and use the blank space for additional comments.

List the operating systems and programming languages you have used and the years of experience with each.

Your experience with Cray Research computer systems: _____0-1 year _____1-5 year _____5+years

How did you use this manual: _____in a class _____as a tutorial or introduction _____as a procedural guide _____as a reference _____for troubleshooting _____other

Please rate this manual on the following criteria:

	Excellent			Poor
Accuracy	4	3	2	1
Appropriateness (correct technical level)	4	3	2	1
Accessibility (ease of finding information)	4	3	2	1
Physical qualities (binding, printing, illustrations)	4	3	2	1
Terminology (correct, consistent, and clear)	4	3	2	1
Number of examples	4	3	2	1
Quality of examples	4	3	2	1
Index	4	3	2	1

Please use the space below for your comments about this manual. Please include general comments about the usefulness of this manual. If you have discovered inaccuracies or omissions, please specify the number of the page on which the problem occurred.

Name _____
Title _____
Company _____
Telephone _____
Today's date _____

Address _____
City _____
State/Country _____
Zip code _____
Electronic mail address _____

Cut along this line

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

ATTN: Software Information Services
655 LONE OAK DR BLDG F
EAGAN MN 55121-9957

Fold

Reader's Comment Form

CRAY IOS-V Commands Reference Manual SR–2170 8.0.3.2

Your reactions to this manual will help us provide you with better documentation. Please take a moment to complete the following items, and use the blank space for additional comments.

List the operating systems and programming languages you have used and the years of experience with each.

Your experience with Cray Research computer systems: _____0-1 year _____1-5 year _____5+years

How did you use this manual: _____in a class _____as a tutorial or introduction _____as a procedural guide _____as a reference _____for troubleshooting _____other

Please rate this manual on the following criteria:

	Excellent			Poor
Accuracy	4	3	2	1
Appropriateness (correct technical level)	4	3	2	1
Accessibility (ease of finding information)	4	3	2	1
Physical qualities (binding, printing, illustrations)	4	3	2	1
Terminology (correct, consistent, and clear)	4	3	2	1
Number of examples	4	3	2	1
Quality of examples	4	3	2	1
Index	4	3	2	1

Please use the space below for your comments about this manual. Please include general comments about the usefulness of this manual. If you have discovered inaccuracies or omissions, please specify the number of the page on which the problem occurred.

Name _____
Title _____
Company _____
Telephone _____
Today's date _____

Address _____
City _____
State/Country _____
Zip code _____
Electronic mail address _____

Cut along this line

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

ATTN: Software Information Services
655 LONE OAK DR BLDG F
EAGAN MN 55121-9957

Fold