On-Line Applications Research Corporation (OAR).

The authors have used their best efforts in preparing this material. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. No warranty of any kind, expressed or implied, with regard to the software or the material contained in this document is provided. No liability arising out of the application or use of any product described in this document is assumed. The authors reserve the right to revise this material and to make changes from time to time in the content hereof without obligation to notify anyone of such revision or changes.

The RTEMS Project is hosted at http://www.rtems.com. Any inquiries concerning RTEMS, its related support components, its documentation, or any custom services for RTEMS should be directed to the contacts listed on that site. A current list of RTEMS Support Providers is at http://www.rtems.com/support.html.
# Table of Contents

**Preface** ......................................................... 1

1 **Configuration and Initialization** ............... 3
   1.1 Introduction .............................................. 3
   1.2 Configuration .............................................. 3
       1.2.1 Customizing the Command Set .................. 3
       1.2.2 Adding Custom Commands ...................... 3
   1.3 Initialization ............................................. 5
       1.3.1 Attached to a Serial Port .................... 5
       1.3.2 Attached to a Socket ......................... 5
   1.4 Functions ................................................. 5
       1.4.1 rtems_shell_init - initialize the shell .......... 6

2 **General Commands** .......................................... 7
   2.1 Introduction .............................................. 7
   2.2 Commands ................................................. 7
       2.2.1 alias - add alias for an existing command .......... 8
       2.2.2 date - print or set current date and time .......... 9
       2.2.3 echo - produce message in a shell script .......... 10
       2.2.4 sleep - delay for a specified amount of time .......... 12
       2.2.5 id - show uid gid euid and egid ............... 13
       2.2.6 tty - show ttyname .................................. 14
       2.2.7 whoami - print effective user id ............... 15
       2.2.8 logoff - logoff from the system ............... 16
       2.2.9 exit - exit the shell ................................ 17

3 **File and Directory Commands** .................. 19
   3.1 Introduction .............................................. 19
   3.2 Commands ................................................. 19
       3.2.1 umask - set file mode creation mask .......... 20
       3.2.2 cp - copy files ..................................... 21
       3.2.3 pwd - print work directory ...................... 25
       3.2.4 ls - list files in the directory ............... 26
       3.2.5 chdir - change the current directory .......... 28
       3.2.6 mkdir - create a directory ..................... 29
       3.2.7 rmdir - remove empty directories .......... 31
       3.2.8 chroot - change the root directory .......... 32
       3.2.9 chmod - change permissions of a file .......... 33
       3.2.10 cat - display file contents ................. 35
       3.2.11 rm - remove files .................................. 36
       3.2.12 mount - mount disk ................................ 37
       3.2.13 umount - umount disk ................................ 39
3.2.14  blksync - sync the block driver ......................... 40
3.2.15  dir - alias for ls ........................................ 41
3.2.16  cd - alias for chdir ..................................... 42

4  Memory Commands ............................................. 43
   4.1  Introduction .............................................. 43
   4.2  Commands .................................................. 43
       4.2.1  mdump - display contents of memory ................. 44
       4.2.2  wdump - display contents of memory (word) ....... 45
       4.2.3  medit - modify contents of memory ................ 46
       4.2.4  mfill - fill memory with pattern .................. 47
       4.2.5  mmove - move contents of memory ................... 49
       4.2.6  malloc - obtain information on C program heap .... 50

5  RTEMS Specific Commands ............................. 53
   5.1  Introduction .............................................. 53
   5.2  Commands .................................................. 53
       5.2.1  cpuuse - print or reset per thread cpu usage ....... 54
       5.2.2  stackuse - print per thread stack usage .......... 56
       5.2.3  perioduse - print or reset per period usage ....... 57
       5.2.4  wkspace - display information on executive workspace 59
       5.2.5  config - show the system configuration ........... 61
       5.2.6  itask - list init tasks for the system .......... 62
       5.2.7  extension - display information about extensions .. 63
       5.2.8  task - display information about tasks .......... 64
       5.2.9  queue - display information about message queues .. 65
       5.2.10  sema - display information about semaphores ...... 66
       5.2.11  region - display information about regions ...... 67
       5.2.12  part - display information about partitions ...... 68
       5.2.13  object - display information about rtems objects .. 69
       5.2.14  driver - display the rtems device driver table ... 70
       5.2.15  dname - displays information about named drivers .. 71
       5.2.16  pthread - display information about POSIX threads ... 72

6  Network Commands .......................................... 73
   6.1  Introduction .............................................. 73
   6.2  Commands .................................................. 73
       6.2.1  netstats - obtain network statistics ............. 74
       6.2.2  ifconfig - configure a network interface .......... 77
       6.2.3  route - show or manipulate the ip routing table ... 79

Function and Variable Index ............................... 81

Concept Index ............................................... 83

Command Index .............................................. 85
Preface

Real-time embedded systems vary widely based upon their operational and maintenance requirements. Some of these systems provide ways for the user or developer to interact with them. This interaction could be used for operational, diagnostic, or configuration purposes. The capabilities described in this manual are those provided with RTEMS to provide a command line interface for user access. Some of these commands will be familiar as standard POSIX utilities while others are RTEMS specific or helpful in debugging and analyzing an embedded system. As a simple example of the powerful and very familiar capabilities that the RTEMS Shell provides to an application, consider the following example which hints at some of the capabilities available:

Welcome to rtems-4.8.99.0(SPARC/w/FPU/sis)
COPYRIGHT (c) 1989-2008.
On-Line Applications Research Corporation (OAR).

Login into RTEMS

login: rtems
Password:

RTEMS SHELL (Ver.1.0-FRC):/dev/console. Feb 28 2008. 'help' to list commands.

SHLL [/] $ cat /etc/passwd
root:*:0:0:root::/:/bin/sh
rtems:*:1:1:RTEMS Application::/:/bin/sh
tty:!:2:ttty owner::/:/bin/false

SHLL [/] $ ls /dev
-rwxr-xr-x 1 rtems root 0 Jan 01 00:00 console
-rwxr-xr-x 1 root root 0 Jan 01 00:00 console_b
2 files 0 bytes occupied

SHLL [/] $ stackuse
Stack usage by thread
ID NAME LOW HIGH CURRENT AVAILABLE USED
0x00901001 IDLE 0x023d89a0 - 0x023d99af 0x023d9760 4096 608
0x0a010001 UI1 0x023d9f30 - 0x023daf3f 0x023dad18 4096 1804
0x0a010002 SHLL 0x023db4c0 - 0x023df4cf 0x023de9d0 16384 6204
0xffffffff INTR 0x023d2760 - 0x023d375f 0x00000000 4080 316

SHLL [/] $ mount -L
File systems: msdos

In the above example, the user rtems logs into a SPARC based RTEMS system. The first command is cat /etc/passwd. This simple command lets us know that this application is running the In Memory File System (IMFS) and that the infrastructure has provided dummy entries for /etc/passwd and a few other files. The contents of /etc/passwd let us know that the user could have logged in as root. In fact, the root user has more permissions than rtems who is not allowed to write into the filesystem.

The second command is ls /dev which lets us know that RTEMS has POSIX-style device nodes which can be accessed through standard I/O function calls.

The third command executed is the RTEMS specific stackuse which gives a report on the stack usage of each thread in the system. Since stack overflows are a common error in deeply embedded systems, this is a surprising simple, yet powerful debugging aid.
Finally, the last command, `mount -L` hints that RTEMS supports a variety of mountable filesystems. With support for MS-DOS FAT on IDE/ATA and Flash devices as well as network-based filesystems such as NFS and TFTP, the standard free RTEMS provides a robust infrastructure for embedded applications.

This manual describes the RTEMS Shell and its command set. In our terminology, the Shell is just a loop reading user input and turning that input into commands with argument. The Shell provided with RTEMS is a simple command reading loop with limited scripting capabilities. It can be connected to via a standard serial port or connected to the RTEMS `telnetd` server for use across a network.

Each command in the command set is implemented as a single subroutine which has a `main-style` prototype. The commands interpret their arguments and operate upon stdin, stdout, and stderr by default. This allows each command to be invoked independent of the shell.

The described separation of shell from commands from communications mechanism was an important design goal. At one level, the RTEMS Shell is a complete shell environment providing access to multiple POSIX compliant filesystems and TCP/IP stack. The subset of capabilities available is easy to configure and the standard Shell can be logged into from either a serial port or via telnet. But at another level, the Shell is a large set of components which can be integrated into the user’s developed command interpreter. In either case, it is trivial to add custom commands to the command set available.
1 Configuration and Initialization

1.1 Introduction
This chapter provides information on how the application configures and initializes the RTEMS shell.

1.2 Configuration
The command set available to the application is user configurable. It is configured using a mechanism similar to the confdefs.h mechanism used to specify application configuration.

In the simplest case, if the user wishes to configure a command set with all commands available that are neither filesystem management (e.g. mounting, formatting, etc.) or network related, then the following is all that is required:

```c
#define CONFIGURE_SHELL_COMMANDS_INIT
#define CONFIGURE_SHELL_COMMANDS_ALL
#include <rtems/shellconfig.h>
```

In a slightly more complex example, if the user wishes to include all networking commands as well as support for mounting MS-DOS and NFS filesystems, then the following is all that is required:

```c
#define CONFIGURE_SHELL_COMMANDS_INIT
#define CONFIGURE_SHELL_COMMANDS_ALL
#define CONFIGURE_SHELL_MOUNT_MSDOS
#define CONFIGURE_SHELL_MOUNT_NFS
#include <rtems/shellconfig.h>
```

1.2.1 Customizing the Command Set
The user can configure specific command sets by either building up the set from individual commands or starting with a complete set and disabling individual commands. Each command has two configuration macros associated with it.

- **CONFIGURE_SHELL_COMMAND_XXX**: Each command has a constant of this form which is defined when building a command set by individually enabling specific commands.

- **CONFIGURE_SHELL_NO_COMMAND_XXX**: In contrast, each command has a similar command which is defined when the application is configuring a command set by disabling specific commands in the set.

1.2.2 Adding Custom Commands
One of the design goals of the RTEMS Shell was to make it easy for a user to add custom commands specific to their application. We believe this design goal was accomplished. In order to add a custom command, the user is required to do the following:
• Provide a main-style function which implements the command. If that command function uses a getopt related function to parse arguments, it MUST use the reentrant form.

• Provide a command definition structure of type rtems_shell_cmd_t.

• Configure that command using the CONFIGURE_SHELL_USER_COMMANDS macro.

Custom aliases are configured similarly but the user only provides an alias definition structure of type rtems_shell_alias_t and configures the alias via the CONFIGURE_SHELL_USER_ALIASES macro.

In the following example, we have implemented a custom command named usercmd which simply prints the arguments it was passed. We have also provided an alias for usercmd named userecho.

```c
#include <rtems/shell.h>

int main_usercmd(int argc, char **argv)
{
    int i;
    printf( "UserCommand: argc=%d\n", argc );
    for (i=0 ; i<argc ; i++ )
        printf( "argv[%d]= %s\n", i, argv[i] );
    return 0;
}

rtems_shell_cmd_t Shell_USERCMD_Command = {
    "usercmd", /* name */
    "usercmd n1 [n2 [n3...]]", /* usage */
    "user", /* topic */
    main_usercmd, /* command */
    NULL, /* alias */
    NULL /* next */
};

rtems_shell_alias_t Shell_USERECHO_Alias = {
    "usercmd", /* command */
    "userecho" /* alias */
};

#define CONFIGURE_SHELL_USER_COMMANDS &Shell_USERCMD_Command
#define CONFIGURE_SHELL_USER_ALIASES &Shell_USERECHO_Alias
#define CONFIGURE_SHELL_COMMANDS_INIT
#define CONFIGURE_SHELL_COMMANDS_ALL
#define CONFIGURE_SHELL_MOUNT_MSDOS

#include <rtems/shellconfig.h>
```

Notice in the above example, that the user wrote the main for their command (e.g. main_usercmd) which looks much like any other main(). They then defined a rtems_shell_cmd_t structure named Shell_USERCMD_Command which describes that command. This command definition structure is registered into the static command set by defining CONFIGURE_SHELL_USER_COMMANDS to &Shell_USERCMD_Command.
Similarly, to add the `userecho` alias, the user provides the alias definition structure named `Shell_USERECHO_Alias` and defines `CONFIGURE_SHELL_USER_ALIASES` to configure the alias.

The user can configure any number of commands and aliases in this manner.

### 1.3 Initialization

The shell may be easily attached to a serial port or to the `telnetd` server. This section describes how that is accomplished.

#### 1.3.1 Attached to a Serial Port

Starting the shell attached to the console or a serial port is very simple. The user invokes `rtems_shell_init` with parameters to indicate the characteristics of the task that will be executing the shell including name, stack size, and priority. The user also specifies the device that the shell is to be attached to.

This example is taken from the `fileio` sample test. This shell portion of this test can be run on any target which provides a console with input and output capabilities. It does not include any commands which cannot be supported on all BSPs. The source code for this test is in `testsuites/samples/fileio` with the shell configuration in the `init.c` file.

```c
#include <rtems/shell.h>

void start_shell(void) {
    printf(" ================\n");
    printf(" starting shell\n");
    printf(" ================\n");
    rtems_shell_init(
        "SHLL", /* task name */
        RTEMS_MINIMUM_STACK_SIZE * 4, /* task stack size */
        100, /* task priority */
        "/dev/console", /* device name */
        0, /* run forever */
        1 /* wait for shell to terminate */
    );
}
```

In the above example, the call to `rtems_shell_init` spawns a task to run the RTEMS Shell attached to `/dev/console` and executing at priority 100. The caller suspends itself and lets the shell take over the console device. When the shell is exited by the user, then control returns to the caller.

#### 1.3.2 Attached to a Socket

TBD

### 1.4 Functions

This section describes the Shell related C functions which are publicly available related to initialization and configuration.
1.4.1 rtems_shell_init - initialize the shell

CALLING SEQUENCE:

```c
rtems_status_code rtems_shell_init (  
    char *task_name,  
    uint32_t task_stacksize,  
    rtems_task_priority task_priority,  
    char *devname,  
    tcflag_t tcflag,  
    int forever  
);  
```

DIRECTIVE STATUS CODES:

- RTEMS_SUCCESSFUL - Shell task spawned successfully
- others - to indicate a failure condition

DESCRIPTION:

This service creates a task with the specified characteristics to run the RTEMS Shell attached to the specified `devname`.

NOTES:

This method invokes the `rtems_task_create` and `rtems_task_start` directives and as such may return any status code that those directives may return.
2 General Commands

2.1 Introduction

The RTEMS shell has the following general commands:

- **alias** - Add alias for an existing command
- **date** - Print or set current date and time
- **echo** - Produce message in a shell script
- **sleep** - Delay for a specified amount of time
- **id** - show uid gid euid and egid
- **tty** - show ttynname
- **whoami** - print effective user id
- **logoff** - logoff from the system
- **exit** - alias for logoff command

2.2 Commands

This section details the General Commands available. A subsection is dedicated to each of
the commands and describes the behavior and configuration of that command as well as
providing an example usage.
2.2.1 alias - add alias for an existing command

SYNOPSIS:

    alias oldCommand newCommand

DESCRIPTION:

This command adds an alternate name for an existing command to the command set.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use alias:

    SHELL [/] $ me
    shell:me command not found
    SHELL [/] $ alias whoami me
    SHELL [/] $ me
    rtems
    SHELL [/] $ whoami
    rtems

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_ALIAS to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_ALIAS when all shell commands have been configured.

PROGRAMMING INFORMATION:

The alias is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_alias(
        int argc,
        char **argv
    );

The configuration structure for the alias has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_ALIAS_Command;
2.2.2 date - print or set current date and time

SYNOPSYS:

    date
    date DATE TIME

DESCRIPTION:

This command operates one of two modes. When invoked with no arguments, it prints
the current date and time. When invoked with both date and time arguments, it sets the
current time.

The date is specified in YYYY-MM-DD format. The time is specified in HH:MM:SS format.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

This comm

EXAMPLES:

The following is an example of how to use date:

    SHELL [/] $ date
    Fri Jan  1 00:00:09 1988
    SHELL [/] $ date 2008-02-29 06:45:32
    SHELL [/] $ date
    Fri Feb 29 06:45:35 2008

CONFIGURATION:

This command is included in the default shell command set. When building a custom
command set, define CONFIGURE_SHELL_COMMAND_DATE to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_
NO_COMMAND_DATE when all shell commands have been configured.

PROGRAMMING INFORMATION:

The date is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_date(
        int argc,
        char **argv
    );

The configuration structure for the date has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_DATE_Command;
2.2.3 echo - produce message in a shell script

SYNOPSIS:

    echo [-n | -e] args ...

DESCRIPTION:

echo prints its arguments on the standard output, separated by spaces. Unless the -n option is present, a newline is output following the arguments. The -e option causes echo to treat the escape sequences specially, as described in the following paragraph. The -e option is the default, and is provided solely for compatibility with other systems. Only one of the options -n and -e may be given.

If any of the following sequences of characters is encountered during output, the sequence is not output. Instead, the specified action is performed:

- \\b: A backspace character is output.
- \\c: Subsequent output is suppressed. This is normally used at the end of the last argument to suppress the trailing newline that echo would otherwise output.
- \\f: Output a form feed.
- \\n: Output a newline character.
- \\r: Output a carriage return.
- \\t: Output a (horizontal) tab character.
- \\v: Output a vertical tab.
- \0digits: Output the character whose value is given by zero to three digits. If there are zero digits, a nul character is output.
- \\: Output a backslash.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

The octal character escape mechanism (\0digits) differs from the C language mechanism. There is no way to force echo to treat its arguments literally, rather than interpreting them as options and escape sequences.

EXAMPLES:

The following is an example of how to use echo:

    $ echo a b c
    a b c
    $ echo

CONFIGURATION:
This command is included in the default shell command set. When building a custom command set, define \texttt{CONFIGURE\_SHELL\_COMMAND\_ECHO} to have this command included.

This command can be excluded from the shell command set by defining \texttt{CONFIGURE\_SHELL\_NO\_COMMAND\_ECHO} when all shell commands have been configured.

PROGRAMMING INFORMATION:
The \texttt{echo} is implemented by a C language function which has the following prototype:

\begin{verbatim}
int rtems_shell_rtems_main_echo(
    int argc,
    char **argv
);
\end{verbatim}

The configuration structure for the \texttt{echo} has the following prototype:

\begin{verbatim}
extern rtems_shell_cmd_t rtems_shell_ECHO_Command;
\end{verbatim}

ORIGIN:
The implementation and portions of the documentation for this command are from NetBSD 4.0.
2.2.4 sleep - delay for a specified amount of time

SYNOPSIS:

sleep seconds
sleep seconds nanoseconds

DESCRIPTION:

This command causes the task executing the shell to block for the specified number of seconds and nanoseconds.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

This command is implemented using the nanosleep() method.

The command line interface is similar to the sleep command found on POSIX systems but the addition of the nanoseconds parameter allows fine grained delays in shell scripts without adding another command such as usleep.

EXAMPLES:

The following is an example of how to use sleep:

    SHLL [/] $ sleep 10
    SHLL [/] $ sleep 0 5000000

It is not clear from the above but there is a ten second pause after executing the first command before the prompt is printed. The second command completes very quickly from a human perspective and there is no noticeable delay in the prompt being printed.

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_SLEEP to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_SLEEP when all shell commands have been configured.

PROGRAMMING INFORMATION:

The sleep is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_sleep(
        int argc,
        char **argv
    );

The configuration structure for the sleep has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_SLEEP_Command;
2.2.5 id - show uid gid euid and egid

SYNOPSIS:

id

DESCRIPTION:
This command prints the user identity. This includes the user id (uid), group id (gid),
effective user id (euid), and effective group id (egid).

EXIT STATUS:
This command returns 0 on success and non-zero if an error is encountered.

NOTES:
Remember there is only one POSIX process in a single processor RTEMS application. Each
thread may have its own user identity and that identity is used by the filesystem to enforce
permissions.

EXAMPLES:
The first example of the id command is from a session logged in as the normal user rtems:

```
/ # id
uid=1(rtems),gid=1(rtems),euid=1(rtems),egid=1(rtems)
```

The second example of the id command is from a session logged in as the root user:

```
/ # id
uid=0(root),gid=0(root),euid=0(root),egid=0(root)
```

CONFIGURATION:
This command is included in the default shell command set. When building a custom
command set, define CONFIGURE_SHELL_COMMAND_ID to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_
NO_COMMAND_ID when all shell commands have been configured.

PROGRAMMING INFORMATION:
The id is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_id(
    int argc,
    char **argv
);
```

The configuration structure for the id has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_ID_Command;
```
2.2.6 tty - show ttynname

SYNOPSIS:

tty

DESCRIPTION:
This command prints the file name of the device connected to standard input.

EXIT STATUS:
This command returns 0 on success and non-zero if an error is encountered.

NOTES:
NONE

EXAMPLES:
The following is an example of how to use tty:

    SHELL [/] $ tty
    /dev/console

CONFIGURATION:
This command is included in the default shell command set. When building a custom
command set, define CONFIGURE_SHELL_COMMAND_TTY to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_
NO_COMMAND_TTY when all shell commands have been configured.

PROGRAMMING INFORMATION:
The tty is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_tty(
        int argc,
        char **argv
    );

The configuration structure for the tty has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_TTY_Command;
2.2.7 whoami - print effective user id

SYNOPSIS:

    whoami

DESCRIPTION:

This command displays the user name associated with the current effective user id.

EXIT STATUS:

This command always succeeds.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use whoami:

    SHELL [/] $ whoami
    rtems

CONFIGURATION:

This command is included in the default shell command set. When building a custom
command set, define CONFIGURE_SHELL_COMMAND_WHOAMI to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_No_COMMAND_WHOAMI when all shell commands have been configured.

PROGRAMMING INFORMATION:

The whoami is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_whoami(
        int argc,
        char **argv
    );

The configuration structure for the whoami has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_WHOAMI_Command;
2.2.8 logoff - logoff from the system

SYNOPSIS:
logoff

DESCRIPTION:
This command logs the user out of the shell.

EXIT STATUS:
This command does not return.

NOTES:
The system behavior when the shell is exited depends upon how the shell was initiated. The typical behavior is that a login prompt will be displayed for the next login attempt or that the connection will be dropped by the RTEMS system.

EXAMPLES:
The following is an example of how to use logoff:

    SHELL [/] $ logoff
    logoff from the system...

CONFIGURATION:
This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_LOGOFF to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_LOGOFF when all shell commands have been configured.

PROGRAMMING INFORMATION:
The logoff is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_logoff(
        int argc,
        char **argv
    );

The configuration structure for the logoff has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_LOGOFF_Command;
2.2.9 exit - exit the shell

SYNOPSYS:

exit

DESCRIPTION:

This command causes the shell interpreter to exit.

EXIT STATUS:

This command does not return.

NOTES:

In contrast to Section 2.2.8 [General Commands logoff - logoff from the system], page 16, this command is built into the shell interpreter loop.

EXAMPLES:

The following is an example of how to use exit:

    SHELL [/] $ exit
    Shell exiting

CONFIGURATION:

This command is always present and cannot be disabled.

PROGRAMMING INFORMATION:

The exit is implemented directly in the shell interpreter. There is no C routine associated with it.
3 File and Directory Commands

3.1 Introduction

The RTEMS shell has the following file and directory commands:

- **umask** - Set file mode creation mask
- **cp** - copy files
- **pwd** - print work directory
- **ls** - list files in the directory
- **chdir** - change the current directory
- **mkdir** - create a directory
- **rmdir** - remove empty directories
- **chroot** - change the root directory
- **chmod** - change permissions of a file
- **cat** - display file contents
- **msdosfmt** - format disk
- **rm** - remove files
- **mount** - mount disk
- **unmount** - unmount disk
- **blkssync** - sync the block driver
- **dir** - alias for ls
- **cd** - alias for chdir

3.2 Commands

This section details the File and Directory Commands available. A subsection is dedicated to each of the commands and describes the behavior and configuration of that command as well as providing an example usage.
3.2.1 umask - set file mode creation mask

SYNOPSIS:

    umask [new_umask]

DESCRIPTION:

This command sets the user file creation mask to new_umask. The argument new_umask may be octal, hexadecimal, or decimal.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

This command does not currently support symbolic mode masks.

EXAMPLES:

The following is an example of how to use umask:

    SHELL [/] $ umask
    022
    SHELL [/] $ umask 0666
    0666
    SHELL [/] $ umask
    0666

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_UMASK to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_UMASK when all shell commands have been configured.

PROGRAMMING INFORMATION:

The umask is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_umask(
        int     argc,
        char **argv
    );

The configuration structure for the umask has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_UMASK_Command;
3.2.2 cp - copy files

SYNOPSIS:

\[ \text{cp \ [-R \ [-H \ | \ -L \ | \ -P]] \ [-f \ | \ -i] \ [-pv] \ src \ target} \]

\[ \text{cp \ [-R \ [-H \ | \ -L] \ ] \ [-f \ | \ -i] \ [-NpPv] \ source\_file \ ... \ target\_directory} \]

DESCRIPTION:

In the first synopsis form, the cp utility copies the contents of the source file to the target file. In the second synopsis form, the contents of each named source file is copied to the destination target directory. The names of the files themselves are not changed. If cp detects an attempt to copy a file to itself, the copy will fail.

The following options are available:

\(-f\)

- For each existing destination pathname, attempt to overwrite it. If permissions do not allow copy to succeed, remove it and create a new file, without prompting for confirmation. (The \(-i\) option is ignored if the \(-f\) option is specified.)

\(-H\)

- If the \(-R\) option is specified, symbolic links on the command line are followed. (Symbolic links encountered in the tree traversal are not followed.)

\(-i\)

- Causes cp to write a prompt to the standard error output before copying a file that would overwrite an existing file. If the response from the standard input begins with the character ‘y’, the file copy is attempted.

\(-L\)

- If the \(-R\) option is specified, all symbolic links are followed.

\(-N\)

- When used with \(-p\), do not copy file flags.

\(-p\)

- Causes cp to preserve in the copy as many of the modification time, access time, file flags, file mode, user ID, and group ID as allowed by permissions.

- If the user ID and group ID cannot be preserved, no error message is displayed and the exit value is not altered.

- If the source file has its set user ID bit on and the user ID cannot be preserved, the set user ID bit is not preserved in the copy’s permissions. If the source file has its set group ID bit on and the group
ID cannot be preserved, the set group ID bit is not preserved in the copy’s permissions. If the source file has both its set user ID and set group ID bits on, and either the user ID or group ID cannot be preserved, neither the set user ID or set group ID bits are preserved in the copy’s permissions.

-R

If source_file designates a directory, cp copies the directory and the entire subtree connected at that point. This option also causes symbolic links to be copied, rather than indirected through, and for cp to create special files rather than copying them as normal files. Created directories have the same mode as the corresponding source directory, unmodified by the process’s umask.

-v

Cause cp to be verbose, showing files as they are copied.

For each destination file that already exists, its contents are overwritten if permissions allow, but its mode, user ID, and group ID are unchanged.

In the second synopsis form, target_directory must exist unless there is only one named source_file which is a directory and the -R flag is specified.

If the destination file does not exist, the mode of the source file is used as modified by the file mode creation mask (umask, see csh(1)). If the source file has its set user ID bit on, that bit is removed unless both the source file and the destination file are owned by the same user. If the source file has its set group ID bit on, that bit is removed unless both the source file and the destination file are in the same group and the user is a member of that group. If both the set user ID and set group ID bits are set, all of the above conditions must be fulfilled or both bits are removed.

Appropriate permissions are required for file creation or overwriting.

Symbolic links are always followed unless the -R flag is set, in which case symbolic links are not followed, by default. The -H or -L flags (in conjunction with the -R flag), as well as the -P flag cause symbolic links to be followed as described above. The -H and -L options are ignored unless the -R option is specified. In addition, these options override each subheading other and the command’s actions are determined by the last one specified.

EXIT STATUS:
This command returns 0 on success and non-zero if an error is encountered.

NOTES:
NONE

EXAMPLES:
The following is an example of how to use cp to copy a file to a new name in the current directory:

```
SHLL [/] # cat joel
```
Chapter 3: File and Directory Commands

The following is an example of how to use `cp` to copy one or more files to a destination directory and use the same basename in the destination directory:

```
SHLL [/] # mkdir tmp
SHLL [/] # ls tmp
0 files 0 bytes occupied
SHLL [/] # cp /etc/passwd tmp
SHLL [/] # ls /tmp
-rw-r--r-- 1 root root 102 Jan 01 00:01 passwd
1 files 102 bytes occupied
SHLL [/] # cp /etc/passwd /etc/group /tmp
SHLL [/] # ls /tmp
-rw-r--r-- 1 root root 102 Jan 01 00:01 passwd
-rw-r--r-- 1 root root 42 Jan 01 00:01 group
2 files 144 bytes occupied
SHLL [/] #
```

**CONFIGURATION:**

This command is included in the default shell command set. When building a custom command set, define `CONFIGURE_SHELL_COMMAND_CP` to have this command included.

This command can be excluded from the shell command set by defining `CONFIGURE_SHELL_NO_COMMAND_CP` when all shell commands have been configured.

**PROGRAMMING INFORMATION:**

The `cp` is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_cp(
    int argc,
    char **argv
);
```

The configuration structure for the `cp` has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_CP_Command;
```
ORIGIN:
The implementation and portions of the documentation for this command are from NetBSD 4.0.
3.2.3 pwd - print work directory

SYNOPSIS:

    pwd

DESCRIPTION:
This command prints the fully qualified filename of the current working directory.

EXIT STATUS:
This command returns 0 on success and non-zero if an error is encountered.

NOTES:
NONE

EXAMPLES:
The following is an example of how to use \texttt{pwd}:

\begin{verbatim}
    SHLL [/] $ pwd
    /
    SHLL [/] $ cd dev
    SHLL [/dev] $ pwd
    /dev
\end{verbatim}

CONFIGURATION:
This command is included in the default shell command set. When building a custom command set, define \texttt{CONFIGURE\_SHELL\_COMMAND\_PWD} to have this command included.

This command can be excluded from the shell command set by defining \texttt{CONFIGURE\_SHELL\_NO\_COMMAND\_PWD} when all shell commands have been configured.

PROGRAMMING INFORMATION:
The \texttt{pwd} is implemented by a C language function which has the following prototype:

\begin{verbatim}
    int rtems_shell_rtems_main_pwd(
        int argc,
        char **argv
    );
\end{verbatim}

The configuration structure for the \texttt{pwd} has the following prototype:

\begin{verbatim}
    extern rtems_shell_cmd_t rtems_shell_PWD_Command;
\end{verbatim}
3.2.4 ls - list files in the directory

SYNOPSIS:

    ls [dir]

DESCRIPTION:

This command displays the contents of the specified directory. If no arguments are given, then it displays the contents of the current working directory.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

This command currently does not display information on a set of files like the POSIX ls(1). It only displays the contents of entire directories.

EXAMPLES:

The following is an example of how to use ls:

```
SHLL [/] $ ls
drwxr-xr-x 1 root root 536 Jan 01 00:00 dev/
drwxr-xr-x 1 root root 1072 Jan 01 00:00 etc/
2 files 1608 bytes occupied
SHLL [/] $ ls etc
-rw-r--r-- 1 root root 102 Jan 01 00:00 passwd
-rw-r--r-- 1 root root 42 Jan 01 00:00 group
-rw-r--r-- 1 root root 30 Jan 01 00:00 issue
-rw-r--r-- 1 root root 28 Jan 01 00:00 issue.net
4 files 202 bytes occupied
SHLL [/] $ ls dev etc
-rwrxr-xr-x 1 rtems root 0 Jan 01 00:00 console
-rwrxr-xr-x 1 root root 0 Jan 01 00:00 console_b
```

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_LS to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_LS when all shell commands have been configured.

PROGRAMMING INFORMATION:

The ls is implemented by a C language function which has the following prototype:

```
int rtems_shell_rtems_main_ls(
    int argc,
    char **argv
);
```

The configuration structure for the ls has the following prototype:
extern rtems_shell_cmd_t rtems_shell_LS_Command;
3.2.5 chdir - change the current directory

SYNOPSIS:

    chdir [dir]

DESCRIPTION:

This command is used to change the current working directory to the specified directory. If no arguments are given, the current working directory will be changed to /.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use chdir:

    SHLL [/] $ pwd
    /  
    SHLL [/] $ chdir etc
    SHLL [/etc] $ pwd
    /etc

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_CHDIR to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_CHDIR when all shell commands have been configured.

PROGRAMMING INFORMATION:

The chdir is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_chdir(
        int argc,
        char **argv
    );

The configuration structure for the chdir has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_CHDIR_Command;
3.2.6 mkdir - create a directory

SYNOPSIS:

    mkdir dir [dir1 .. dirN]

DESCRIPTION:

This command creates the set of directories in the order they are specified on the command line. If an error is encountered making one of the directories, the command will continue to attempt to create the remaining directories on the command line.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

If this command is invoked with no arguments, nothing occurs.

The user must have sufficient permissions to create the directory. For the fileio test provided with RTEMS, this means the user must login as root not rtems.

EXAMPLES:

The following is an example of how to use mkdir:

    SHELL [/] # ls
    drwxr-xr-x  1 root root   536 Jan 01 00:00 dev/
    drwxr-xr-x  1 root root  1072 Jan 01 00:00 etc/
    2 files 1608 bytes occupied
    SHELL [/] # mkdir joel
    SHELL [/] # ls joel
    0 files 0 bytes occupied
    SHELL [/] # cp etc/passwd joel
    SHELL [/] # ls joel
    -rw-r--r--  1 root root   102 Jan 01 00:02 passwd
    1 files 102 bytes occupied

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_MKDIR to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_MKDIR when all shell commands have been configured.

PROGRAMMING INFORMATION:

The mkdir is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_mkdir(
        int argc,
        char **argv
    );
The configuration structure for the `mkdir` has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_MKDIR_Command;
```
3.2.7 rmdir - remove empty directories

SYNOPSIS:

    rmdir [dir1 .. dirN]

DESCRIPTION:

This command removes the specified set of directories. If no directories are provided on the
command line, no actions are taken.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

This command is implemented using the rmdir(2) system call and all reasons that call
may fail apply to this command.

EXAMPLES:

The following is an example of how to use rmdir:

    SHELL [/] # mkdir joeldir
    SHELL [/] # rmdir joeldir
    SHELL [/] # ls joeldir
    joeldir: No such file or directory.

CONFIGURATION:

This command is included in the default shell command set. When building a custom
command set, define CONFIGURE_SHELL_COMMAND_RMDIR to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_ 
NO_COMMAND_RMDIR when all shell commands have been configured.

PROGRAMMING INFORMATION:

The rmdir is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_rmdir(
        int argc,
        char **argv
    );

The configuration structure for the rmdir has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_RMDIR_Command;
3.2.8 chroot - change the root directory

SYNOPSIS:

```
chroot [dir]
```

DESCRIPTION:

This command changes the root directory to `dir` for subsequent commands.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

The destination directory `dir` must exist.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use `chroot` and the impact it has on the environment for subsequent command invocations:

```
SHLL [/] $ cat passwd
  cat: passwd: No such file or directory
SHLL [/] $ chroot etc
SHLL [/] $ cat passwd
  root:*:0:0:root::/:/bin/sh
  rtems:*:1:1:RTEMS Application::/:/bin/sh
  tty:*:2:2:tty owner::/:/bin/false
SHLL [/] $ cat /etc/passwd
  cat: /etc/passwd: No such file or directory
```

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define `CONFIGURE_SHELL_COMMAND_CHROOT` to have this command included.

This command can be excluded from the shell command set by defining `CONFIGURE_SHELL_NO_COMMAND_CHROOT` when all shell commands have been configured.

PROGRAMMING INFORMATION:

The `chroot` is implemented by a C language function which has the following prototype:

```
int rtems_shell_rtems_main_chroot(
    int argc,
    char **argv
);
```

The configuration structure for the `chroot` has the following prototype:

```
extern rtems_shell_cmd_t rtems_shell_CHROOT_Command;
```
3.2.9 chmod - change permissions of a file

SYNOPSIS:

    chmod permissions file1 [file2...]

DESCRIPTION:

This command changes the permissions on the files specified to the indicated permissions. The permission values are POSIX based with owner, group, and world having individual read, write, and execute permission bits.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

The chmod command only takes numeric representations of the permissions.

EXAMPLES:

The following is an example of how to use chmod:

    SHLL [/] # cd etc
    SHLL [/etc] # ls
    -rw-r--r-- 1 root root 102 Jan 01 00:00 passwd
    -rw-r--r-- 1 root root 42 Jan 01 00:00 group
    -rw-r--r-- 1 root root 30 Jan 01 00:00 issue
    -rw-r--r-- 1 root root 28 Jan 01 00:00 issue.net
    4 files 202 bytes occupied
    SHLL [/etc] # chmod 0777 passwd
    SHLL [/etc] # ls
    -rwxrwxrwx 1 root root 102 Jan 01 00:00 passwd
    -rw-r--r-- 1 root root 42 Jan 01 00:00 group
    -rw-r--r-- 1 root root 30 Jan 01 00:00 issue
    -rw-r--r-- 1 root root 28 Jan 01 00:00 issue.net
    4 files 202 bytes occupied
    SHLL [/etc] # chmod 0322 passwd
    SHLL [/etc] # ls
    --wx-w--w- 1 nouser root 102 Jan 01 00:00 passwd
    -rw-r--r-- 1 nouser root 42 Jan 01 00:00 group
    -rw-r--r-- 1 nouser root 30 Jan 01 00:00 issue
    -rw-r--r-- 1 nouser root 28 Jan 01 00:00 issue.net
    4 files 202 bytes occupied
    SHLL [/etc] # chmod 0644 passwd
    SHLL [/etc] # ls
    -rw-r--r-- 1 root root 102 Jan 01 00:00 passwd
    -rw-r--r-- 1 root root 42 Jan 01 00:00 group
    -rw-r--r-- 1 root root 30 Jan 01 00:00 issue
    -rw-r--r-- 1 root root 28 Jan 01 00:00 issue.net
    4 files 202 bytes occupied
CONFIGURATION:
This command is included in the default shell command set. When building a custom
command set, define CONFIGURE_SHELL_COMMAND_CHMOD to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_
NO_COMMAND_CHMOD when all shell commands have been configured.

PROGRAMMING INFORMATION:
The chmod is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_chmod(
    int argc,
    char **argv
);
```

The configuration structure for the chmod has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_CHMOD_Command;
```
3.2.10 `cat` - display file contents

**SYNOPSIS:**

```
cat file1 [file2 .. fileN]
```

**DESCRIPTION:**

This command displays the contents of the specified files.

**EXIT STATUS:**

This command returns 0 on success and non-zero if an error is encountered.

**NOTES:**

It is possible to read the input from a device file using `cat`.

**EXAMPLES:**

The following is an example of how to use `cat`:

```
SHLL [/] # cat /etc/passwd
root:*:0:0:root:::/bin/sh
rtems:*:1:1:RTEMS Application:::/bin/sh
tty::2:2:ttty owner:::/bin/false
```

**CONFIGURATION:**

This command is included in the default shell command set. When building a custom command set, define `CONFIGURE_SHELL_COMMAND_CAT` to have this command included.

This command can be excluded from the shell command set by defining `CONFIGURE_SHELL_NO_COMMAND_CAT` when all shell commands have been configured.

**PROGRAMMING INFORMATION:**

The `cat` is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_cat(
    int argc,
    char **argv
);
```

The configuration structure for the `cat` has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_CAT_Command;
```
3.2.11 rm - remove files

SYNOPSIS:

    rm file1 [file2 ... fileN]

DESCRIPTION:

This command deletes a name from the filesystem. If the specified file name was the last link to a file and there are no open file descriptor references to that file, then it is deleted and the associated space in the file system is made available for subsequent use.

If the filename specified was the last link to a file but there are open file descriptor references to it, then the file will remain in existence until the last file descriptor referencing it is closed.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use rm:

    SHLL [/] # cp /etc/passwd tmpfile
    SHLL [/] # cat tmpfile
    root:*:0:0:root::/:/bin/sh
    rtems:*:1:1:RTEMS Application::/:/bin/sh
    tty:*:2:2:TTY owner::/:/bin/false
    SHLL [/] # rm tmpfile
    SHLL [/] # cat tmpfile
    cat: tmpfile: No such file or directory

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_RM to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_RM when all shell commands have been configured.

PROGRAMMING INFORMATION:

The rm is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_rm(
        int argc,
        char **argv
    );

The configuration structure for the rm has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_RM_Command;
3.2.12 mount - mount disk

SYNOPSYS:

    mount [-t fstype] [-r] [-L] device path

DESCRIPTION:

The `mount` command will mount a block device to a mount point using the specified file system. The file systems are:

- msdos - MSDOS File System
- tftp - TFTP Network File System
- ftp - FTP Network File System
- nfs - Network File System

When the file system type is 'msdos' the driver is a "block device driver" node present in the file system. The driver is ignored with the 'tftp' and 'ftp' file systems. For the 'nfs' file system the driver is the 'host:/path' string that described NFS host and the exported file system path.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

The mount point must exist.

The services offered by each file-system vary. For example you cannot list the directory of a TFTP file-system as this server is not provided in the TFTP protocol. You need to check each file-system’s documentation for the services provided.

EXAMPLES:

Mount the Flash Disk driver to the '/fd' mount point:

    $ mount -t msdos /dev/flashdisk0 /fd

Mount the NFS file system exported path 'bar' by host 'foo':

    $ mount -t nfs foo:/bar /nfs

Mount the TFTP file system on '/tftp':

    $ mount -t tftp /tftp

To access the TFTP files on server '10.10.10.10':

    $ cat /tftp/10.10.10.10/test.txt

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define `CONFIGURE_SHELL_COMMAND_MOUNT` to have this command included.
This command can be excluded from the shell command set by defining `CONFIGURE_SHELL_NO_COMMAND_MOUNT` when all shell commands have been configured.

The mount command includes references to file-system code. If you do not wish to include file-system that you do not use do not define the mount command support for that file-system. The file-system mount command defines are:

- msdos - `CONFIGURE_SHELL_MOUNT_MSDOS`
- tftp - `CONFIGURE_SHELL_MOUNT_TFTP`
- ftp - `CONFIGURE_SHELL_MOUNT_FTP`
- nfs - `CONFIGURE_SHELL_MOUNT_NFS`

An example configuration is:

```c
#define CONFIGURE_SHELL_MOUNT_MSDOS
#ifdef RTEMS_NETWORKING
#define CONFIGURE_SHELL_MOUNT_TFTP
#define CONFIGURE_SHELL_MOUNT_FTP
#define CONFIGURE_SHELL_MOUNT_NFS
#endif
```

**PROGRAMMING INFORMATION:**

The `mount` is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_mount(
    int argc,
    char **argv
);
```

The configuration structure for the `mount` has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_MOUNT_Command;
```
3.2.13 unmount - unmount disk

SYNOPSIS:

unmount path

DESCRIPTION:

This command unmounts the device at the specified path.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

TBD - Surely there must be some warnings to go here.

EXAMPLES:

The following is an example of how to use unmount:

EXAMPLE_TBD

CONFIGURATION:

This command is included in the default shell command set. When building a custom
command set, define CONFIGURE_SHELL_COMMAND_UNMOUNT to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_
NO_COMMAND_UNMOUNT when all shell commands have been configured.

PROGRAMMING INFORMATION:

The unmount is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_unmount(
        int argc,
        char **argv
    );

The configuration structure for the unmount has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_UNMOUNT_Command;
3.2.14 blksync - sync the block driver

SYNOPSIS:

   blksync driver

DESCRIPTION:

This command XXX

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use blksync:

   EXAMPLE_TBD

CONFIGURATION:

This command is included in the default shell command set. When building a custom
command set, define CONFIGURE_SHELL_COMMAND_BLKSYNC to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_
NO_COMMAND_BLKSYNC when all shell commands have been configured.

PROGRAMMING INFORMATION:

The blksync is implemented by a C language function which has the following prototype:

   int rtems_shell_rtems_main_blksync(
       int argc,
       char **argv
   );

The configuration structure for the blksync has the following prototype:

   extern rtems_shell_cmd_t rtems_shell_BLKSYNC_Command;
3.2.15 dir - alias for ls

SYNOPSYS:

dir [dir]

DESCRIPTION:

This command is an alias or alternate name for the ls. See Section 3.2.4 [File and Directory Commands ls - list files in the directory], page 26 for more information.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use dir:

```
SHLL [/]$ dir
drwxr-xr-x 1 root root  536 Jan 01 00:00 dev/
drwxr-xr-x 1 root root 1072 Jan 01 00:00 etc/
2 files 1608 bytes occupied
SHLL [/]$ dir etc
-rw-r--r-- 1 root root  102 Jan 01 00:00 passwd
-rw-r--r-- 1 root root  42 Jan 01 00:00 group
-rw-r--r-- 1 root root  30 Jan 01 00:00 issue
-rw-r--r-- 1 root root  28 Jan 01 00:00 issue.net
4 files 202 bytes occupied
```

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_DIR to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_DIR when all shell commands have been configured.

PROGRAMMING INFORMATION:

The dir is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_dir(
    int argc,
    char **argv
);
```

The configuration structure for the dir has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_DIR_Command;
```
3.2.16 cd - alias for chdir

SYNOPSIS:
   cd directory

DESCRIPTION:
This command is an alias or alternate name for the chdir. See Section 3.2.5 [File and Directory Commands chdir - change the current directory], page 28 for more information.

EXIT STATUS:
This command returns 0 on success and non-zero if an error is encountered.

NOTES:
NONE

EXAMPLES:
The following is an example of how to use cd:

   SHLL [/] $ cd etc
   SHLL [/etc] $ cd /
   SHLL [/] $ cd /etc
   SHLL [/etc] $ pwd
   /etc
   SHLL [/etc] $ cd /
   SHLL [/] $ pwd
   /
   SHLL [/] $ cd etc
   SHLL [/etc] $ cd ..
   SHLL [/] $ pwd
   /

CONFIGURATION:
This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_CD to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_CD when all shell commands have been configured.

PROGRAMMING INFORMATION:
The cd is implemented by a C language function which has the following prototype:

   int rtems_shell_rtems_main_cd(
       int argc,
       char **argv
   );

The configuration structure for the cd has the following prototype:

   extern rtems_shell_cmd_t rtems_shell_CD_Command;
4 Memory Commands

4.1 Introduction
The RTEMS shell has the following memory commands:

- **mdump** - Display contents of memory
- **wdump** - Display contents of memory (word)
- **medit** - Modify contents of memory
- **mfill** - File memory with pattern
- **mmove** - Move contents of memory
- **malloc** - Obtain information on C Program Heap

4.2 Commands
This section details the Memory Commands available. A subsection is dedicated to each of the commands and describes the behavior and configuration of that command as well as providing an example usage.
4.2.1 mdump - display contents of memory

SYNOPSIS:

    mdump [address [length]]

DESCRIPTION:

This command displays the contents of memory at the address and length in bytes specified on the command line.

When length is not provided, it defaults to 320 which is twenty lines of output with sixteen bytes of output per line.

When address is not provided, it defaults to 0x00000000.

EXIT STATUS:

This command always returns 0 to indicate success.

NOTES:

Dumping memory from a non-existent address may result in an unrecoverable program fault.

EXAMPLES:

The following is an example of how to use mdump:

    SHELL [/] $ mdump 0x10000 32
    0x0001000000 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ...............  
    0x0001001000 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ...............  
    SHELL [/] $ mdump 0x02000000 32
    0x02000000A1 48 00 00 29 00 80 33-81 C5 22 BC A6 10 21 00 .H..)3."..!.
    0x02000010A1 48 00 00 29 00 80 33-81 C5 22 BC A6 10 21 01 .H..)3."..!.
    SHELL [/] $ mdump 0x02001000 32
    0x0200100003 00 80 00 82 10 60 00-81 98 40 00 83 48 00 00 ........'......H..  
    0x0200101084 00 60 01 84 08 A0 07-86 10 20 01 87 28 C0 02 ..'........ ...  

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_MDUMP to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_MDUMP when all shell commands have been configured.

PROGRAMMING INFORMATION:

The mdump is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_mdump(
        int argc,
        char **argv
    );

The configuration structure for the mdump has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_MDUMP_Command;
4.2.2 wdump - display contents of memory (word)

SYNOPSIS:

wdump [address [length]]

DESCRIPTION:

This command displays the contents of memory at the address and length in bytes specified on the command line.

When length is not provided, it defaults to 320 which is twenty lines of output with sixteen bytes of output per line.

When address is not provided, it defaults to 0x00000000.

EXIT STATUS:

This command always returns 0 to indicate success.

NOTES:

Dumping memory from a non-existent address may result in an unrecoverable program fault.

EXAMPLES:

The following is an example of how to use wdump:

```
SHLL [/] $ wdump 0x02010000 32
0x02010000 0201 08D8 0201 08C0-0201 08AC 0201 0874 ...............t
0x02010010 0201 0894 0201 0718-0201 0640 0201 0798 ...............t
```

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_WDUMP to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_WDUMP when all shell commands have been configured.

PROGRAMMING INFORMATION:

The wdump is implemented by a C language function which has the following prototype:

```
int rtems_shell_rtems_main_wdump(
    int argc,
    char **argv
);
```

The configuration structure for the wdump has the following prototype:

```
extern rtems_shell_cmd_t rtems_shell_WDUMP_Command;
```
4.2.3 medit - modify contents of memory

SYNOPSIS:

medit address value1 [value2 ... valueN]

DESCRIPTION:

This command is used to modify the contents of the memory starting at address using the octets specified by the parameters value1 through valueN.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

Dumping memory from a non-existent address may result in an unrecoverable program fault.

EXAMPLES:

The following is an example of how to use medit:

```
SHLL [/] $ mdump 0x02000000 32
0x02000000 A1 48 00 00 29 00 80 33-81 C5 22 BC A6 10 21 00 .H...)3.."...!.
0x02000010 A1 48 00 00 29 00 80 33-81 C5 22 BC A6 10 21 01 .H...)3.."...!.
SHLL [/] $ medit 0x02000000 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09
SHLL [/] $ mdump 0x02000000 32
0x02000000 01 02 03 04 05 06 07 08-09 00 22 BC A6 10 21 00 ............"...!.
0x02000010 A1 48 00 00 29 00 80 33-81 C5 22 BC A6 10 21 01 .H...)3.."...!.
```

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_MEDIT to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_MEDIT when all shell commands have been configured.

PROGRAMMING INFORMATION:

The medit is implemented by a C language function which has the following prototype:

```
int rtems_shell_rtems_main_medit(
    int argc,
    char **argv
);
```

The configuration structure for the medit has the following prototype:

```
extern rtems_shell_cmd_t rtems_shell_MEDIT_Command;
```
4.2.4 mfill - file memory with pattern

SYNOPSYS:

mfill address length value

DESCRIPTION:
This command is used to fill the memory starting at address for the specified length in octets when the specified at value.

EXIT STATUS:
This command returns 0 on success and non-zero if an error is encountered.

NOTES:
Filling a non-existent address range may result in an unrecoverable program fault. Similarly overwriting interrupt vector tables, code space or critical data areas can be fatal as shown in the example.

EXAMPLES:
In this example, the address used (0x23d89a0) as the base address of the filled area is the end of the stack for the Idle thread. This address was determined manually using gdb and is very specific to this application and BSP. The first command in this example is an mdump to display the initial contents of this memory. We see that the first 8 bytes are 0xA5 which is the pattern used as a guard by the Stack Checker. On the first context switch after the pattern is overwritten by the mfill command, the Stack Checker detect the pattern has been corrupted and generates a fatal error.

```
SHLL [/] $ mdump 0x23d89a0 16
0x023D89A0 A5 A5 A5 A5 A5 A5 A5 A5-FE ED F0 0D 0B AD 0D 06 ................
SHLL [/] $ mfill 0x23d89a0 13 0x5a
SHLL [/] $ BLOWN STACK!! Offending task(0x23D4418): id=0x09010001; name=0x0203D908
stack covers range 0x23D89A0 - 0x23D99AF (4112 bytes)
Damaged pattern begins at 0x023D89A8 and is 16 bytes long
```

CONFIGURATION:
This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_MFILL to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_MFILL when all shell commands have been configured.

PROGRAMMING INFORMATION:
The mfill is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_mfill(
    int argc,
    char **argv
);
```

The configuration structure for the mfill has the following prototype:
extern rtems_shell_cmd_t rtems_shell_MFILL_Command;
4.2.5 mmove - move contents of memory

SYNOPSIS:

mmove dst src length

DESCRIPTION:

This command is used to copy the contents of the memory starting at src to the memory located at dst for the specified length in octets.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use mmove:

```
SHLL [/] $ mdump 0x023d99a0 16
0x023D99A0 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 ..............
SHLL [/] $ mdump 0x02000000 16
0x02000000 A1 48 00 00 29 00 80 33-B1 C5 22 BC A6 10 21 00 ..H..)3.."......
SHLL [/] $ mmove 0x023d99a0 0x02000000 13
SHLL [/] $ mdump 0x023d99a0 16
0x023D99A0 A1 48 00 00 29 00 80 33-B1 C5 22 BC A6 A5 A5 A5 A5 ..H..)3..".....
```

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_MMOVE to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_MMOVE when all shell commands have been configured.

PROGRAMMING INFORMATION:

The mmove is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_mmove(
    int argc,
    char **argv
);
```

The configuration structure for the mmove has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_MMOVE_Command;
```
4.2.6 malloc - obtain information on C program heap

SYNOPSIS:

```
malloc [info|stats]
```

DESCRIPTION:

This command prints either information or statistics about the C Program Heap used by the `malloc` family of calls based upon the value of the first argument passed to the command.

When the subcommand `info` is specified, information on the current state of the C Program Heap is reported. This includes the following information:

- Number of free blocks
- Largest free block
- Total bytes free
- Number of used blocks
- Largest used block
- Total bytes used

When the subcommand `stats` is specified, statistics on the the C Program Heap are reported. Malloc Family Statistics must be enabled for all of the values to be updated. The statistics available includes the following information:

- Currently available memory (in kilobytes)
- Currently allocated memory (in kilobytes)
- Maximum amount of memory ever allocated (in kilobytes)
- Lifetime tally of allocated memory (in kilobytes)
- Lifetime tally of freed memory (in kilobytes)
- Number of calls to `malloc`
- Number of calls to `free`
- Number of calls to `realloc`
- Number of calls to `calloc`

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

The `CONFIGURE_MALLOC_STATISTICS` confdefs.h constant must be defined when the application is configured for the full set of statistics information to be available.

EXAMPLES:

The following is an example of how to use the `malloc` command.
Chapter 4: Memory Commands

SHLL [/] $ malloc info
Number of free blocks: 3
Largest free block: 3626672
Total bytes free: 3627768
Number of used blocks: 130
Largest used block: 1048
Total bytes used: 10136
SHLL [/] $ malloc stats
Malloc statistics
    avail:3552k allocated:9k (0%) max:10k (0%) lifetime:21k freed:12k
    Call counts: malloc:203 free:93 realloc:0 calloc:20
SHLL [/] $ malloc info
Number of free blocks: 3
Largest free block: 3626672
Total bytes free: 3627768
Number of used blocks: 130
Largest used block: 1048
Total bytes used: 10136
SHLL [/] $ malloc stats
Malloc statistics
    avail:3552k allocated:9k (0%) max:10k (0%) lifetime:23k freed:14k
    Call counts: malloc:205 free:95 realloc:0 calloc:20

Note that in the above example, the lifetime allocated and free values have increased between the two calls to malloc stats even though the amount of memory available in the C Program Heap is the same in both the malloc info invocations. This indicates that memory was allocated and freed as a side-effect of the commands.

CONFIGURATION:
This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_MALLOC to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_MALLOC when all shell commands have been configured.

PROGRAMMING INFORMATION:
The malloc is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_malloc(
    int argc,
    char **argv
);
```

The configuration structure for the malloc has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_MALLOC_Command;
```
5 RTEMS Specific Commands

5.1 Introduction

The RTEMS shell has the following rtems commands:

- **cpuuse** - print or reset per thread cpu usage
- **stackuse** - print per thread stack usage
- **perioduse** - print or reset per period usage
- **wkspc** - Display information on Executive Workspace
- **config** - Show the system configuration.
- **itask** - List init tasks for the system
- **extension** - Display information about extensions
- **task** - Display information about tasks
- **queue** - Display information about message queues
- **sema** - display information about semaphores
- **region** - display information about regions
- **part** - display information about partitions
- **object** - Display information about RTEMS objects
- **driver** - Display the RTEMS device driver table
- **dname** - Displays information about named drivers
- **pthread** - Displays information about POSIX threads

5.2 Commands

This section details the RTEMS Specific Commands available. A subsection is dedicated to each of the commands and describes the behavior and configuration of that command as well as providing an example usage.
5.2.1 cpuuse - print or reset per thread cpu usage

SYNOPSIS:

    cpuuse [-r]

DESCRIPTION:

This command may be used to print a report on the per thread cpu usage or to reset the per thread CPU usage statistics. When invoked with the -r option, the CPU usage statistics are reset.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

The granularity of the timing information reported is dependent upon the BSP and the manner in which RTEMS was built. In the default RTEMS configuration, if the BSP supports nanosecond granularity timestamps, then the information reported will be highly accurate. Otherwise, the accuracy of the information reported is limited by the clock tick quantum.

EXAMPLES:

The following is an example of how to use cpuuse:

    SHLL [/>] $ cpuuse
    CPU Usage by thread
            ID   NAME     SECONDS   PERCENT
    0x09010001  IDLE   49.745393   98.953
    0x0a010001   UI1    0.000000   0.000
    0x0a010002   SHLL   0.525928   1.046
    Time since last CPU Usage reset 50.271321 seconds
    SHLL [/>] $ cpuuse -r
    Resetting CPU Usage information
    SHLL [/>] $ cpuuse
    CPU Usage by thread
            ID   NAME     SECONDS   PERCENT
    0x09010001  IDLE    0.000000   0.000
    0x0a010001   UI1    0.000000   0.000
    0x0a010002   SHLL  0.003092 100.000
    Time since last CPU Usage reset 0.003092 seconds

In the above example, the system had set idle for nearly a minute when the first report was generated. The cpuuse -r and cpuuse commands were pasted from another window so were executed with no gap between. In the second report, only the shell thread has run since the CPU Usage was reset. It has consumed approximately 3.092 milliseconds of CPU time processing the two commands and generating the output.
CONFIGURATION:
This command is included in the default shell command set. When building a custom command set, define `CONFIGURE_SHELL_COMMAND_CPUUSE` to have this command included.

This command can be excluded from the shell command set by defining `CONFIGURE_SHELL_NO_COMMAND_CPUUSE` when all shell commands have been configured.

PROGRAMMING INFORMATION:
The `cpuuse` is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_cpuuse(
    int argc,
    char **argv
);
```

The configuration structure for the `cpuuse` has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_CPUUSE_Command;
```
5.2.2 stackuse - print per thread stack usage

SYNOPSIS:

stackuse

DESCRIPTION:

This command prints a Stack Usage Report for all of the tasks and threads in the system. On systems which support it, the usage of the interrupt stack is also included in the report.

EXIT STATUS:

This command always succeeds and returns 0.

NOTES:

The STACK_CHECKER_ON confdefs.h constant must be defined when the application is configured for this command to have any information to report.

EXAMPLES:

The following is an example of how to use stackuse:

```shell
SHELL [/] $ stackuse
Stack usage by thread
ID  NAME  LOW  HIGH  CURRENT  AVAILABLE  USED
0090000000000000 IDLE 0x023d99af - 0x023d99af 0x023d9760 4096 608
0090000000000001 UI1 0x023d99af - 0x023d99af 0x023dad18 4096 1804
0090000000000002 SHLL 0x023d99af - 0x023d99af 0x023de9d0 16384 5116
0010000000000000 INTR 0x023d99af - 0x023d99af 0x00000000 4080 316
```

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_STACKUSE to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_STACKUSE when all shell commands have been configured.

PROGRAMMING INFORMATION:

The stackuse is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_stackuse(
   int argc,
   char **argv
);
```

The configuration structure for the stackuse has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_STACKUSE_Command;
```
5.2.3 perioduse - print or reset per period usage

SYNOPSIS:

   perioduse [-r]

DESCRIPTION:

This command may be used to print a statistics report on the rate monotonic periods in
the application or to reset the rate monotonic period usage statistics. When invoked with
the -r option, the usage statistics are reset.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

The granularity of the timing information reported is dependent upon the BSP and the
manner in which RTEMS was built. In the default RTEMS configuration, if the BSP
supports nanosecond granularity timestamps, then the information reported will be highly
accurate. Otherwise, the accuracy of the information reported is limited by the clock tick
quantum.

EXAMPLES:

The following is an example of how to use perioduse:

   SHELL [/] $ perioduse
   Period information by period
   --- CPU times are in seconds ---
   --- Wall times are in seconds ---
   ID   OWNER  COUNT  MISSED  CPU TIME  WALL TIME
       MIN/MAX/AVG  MIN/MAX/AVG
   0x42010001 TA1  502  0  0:000039/0:042650/0:004158  0:000039/0:020118/0:002848
   0x42010002 TA2  502  0  0:000041/0:042657/0:004309  0:000041/0:020116/0:002848
   0x42010003 TA3  501  0  0:000041/0:041564/0:003653  0:000041/0:020003/0:002814
   0x42010004 TA4  501  0  0:000043/0:044075/0:004911  0:000043/0:020004/0:002814
   0x42010005 TA5   10 0  0:000065/0:005413/0:002739  0:000065/1:000457/0:041058

   SHELL [/] $ perioduse -r
   Resetting Period Usage information
   SHELL [/] $ perioduse
   --- CPU times are in seconds ---
   --- Wall times are in seconds ---
   ID   OWNER  COUNT  MISSED  CPU TIME  WALL TIME
       MIN/MAX/AVG  MIN/MAX/AVG
   0x42010001 TA1   0  0
   0x42010002 TA2   0  0
   0x42010003 TA3   0  0
   0x42010004 TA4   0  0
   0x42010005 TA5   0  0

CONFIGURATION:

This command is included in the default shell command set. When building a custom com-
mand set, define CONFIGURE_SHELL_COMMAND_PERIODUSE to have this command included.
This command can be excluded from the shell command set by defining `CONFIGURE_SHELL_NO_COMMAND_PERIODUSE` when all shell commands have been configured.

**PROGRAMMING INFORMATION:**
The `perioduse` is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_perioduse(
    int argc,
    char **argv
);
```

The configuration structure for the `perioduse` has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_PERIODUSE_Command;
```
5.2.4 wkspace - display information on executive workspace

SYNOPSIS:

\texttt{wkspace}

DESCRIPTION:

This command prints information on the current state of the RTEMS Executive Workspace reported. This includes the following information:

- Number of free blocks
- Largest free block
- Total bytes free
- Number of used blocks
- Largest used block
- Total bytes used

EXIT STATUS:

This command always succeeds and returns 0.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use \texttt{wkspace}:

\begin{verbatim}
  SHELL [/] $ wkspace
  Number of free blocks: 1
  Largest free block: 132336
  Total bytes free: 132336
  Number of used blocks: 36
  Largest used block: 16408
  Total bytes used: 55344
\end{verbatim}

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define \texttt{CONFIGURE\_SHELL\_COMMAND\_WKSPACE} to have this command included.

This command can be excluded from the shell command set by defining \texttt{CONFIGURE\_SHELL\_NO\_COMMAND\_WKSPACE} when all shell commands have been configured.

PROGRAMMING INFORMATION:

The \texttt{wkspace} is implemented by a C language function which has the following prototype:

\begin{verbatim}
int rtems\_shell\_rtems\_main\_wkspace(  
    int argc,  
    char **argv  
);
\end{verbatim}
The configuration structure for the \texttt{wkspace} has the following prototype:

\begin{verbatim}
extern rtems_shell_cmd_t rtems_shell_WKSPACE_Command;
\end{verbatim}
5.2.5 config - show the system configuration.

SYNOPSIS:
config

DESCRIPTION:
This command display information about the RTEMS Configuration.

EXIT STATUS:
This command always succeeds and returns 0.

NOTES:
At this time, it does not report every configuration parameter. This is an area in which user submissions or sponsorship of a developer would be appreciated.

EXAMPLES:
The following is an example of how to use config:

INITIAL (startup) Configuration Info
------------------------------------------------------------------------------
WORKSPACE start: 0x23d22e0; size: 0x2dd20
TIME usec/tick: 10000; tick/timeslice: 50; tick/sec: 100
MAXIMUMS tasks: 20; timers: 0; sems: 50; que’s: 20; ext’s: 1
partitions: 0; regions: 0; ports: 0; periods: 0

CONFIGURATION:
This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_CONFIG to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_CONFIG when all shell commands have been configured.

PROGRAMMING INFORMATION:
The config is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_config(
    int argc,
    char **argv
);
```

The configuration structure for the config has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_CONFIG_Command;
```
5.2.6 *itask* - list init tasks for the system

**SYNOPSIS:**

```
itask
```

**DESCRIPTION:**

This command prints a report on the set of initialization tasks and threads in the system.

**EXIT STATUS:**

This command always succeeds and returns 0.

**NOTES:**

At this time, it includes only Classic API Initialization Tasks. This is an area in which user submissions or sponsorship of a developer would be appreciated.

**EXAMPLES:**

The following is an example of how to use *itask*:

```
SHLL [/] $ itask
# NAME ENTRY ARGUMENT PRIO MODES ATTRIBUTES STACK SIZE
------------------------------------------------------------------------------
  0 UI1 [0x2002258] 0 [0x0] 1 nP DEFAULT 4096 [0x1000]
```

**CONFIGURATION:**

This command is included in the default shell command set. When building a custom command set, define `CONFIGURE_SHELL_COMMAND_ITASK` to have this command included.

This command can be excluded from the shell command set by defining `CONFIGURE_SHELL_NO_COMMAND_ITASK` when all shell commands have been configured.

**PROGRAMMING INFORMATION:**

The *itask* is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_itask(
    int argc,
    char **argv
);
```

The configuration structure for the *itask* has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_ITASK_Command;
```
5.2.7 extension - display information about extensions

SYNOPSIS:
extension [id [id ...]]

DESCRIPTION:
When invoked with no arguments, this command prints information on the set of User
Extensions currently active in the system.
If invoked with a set of ids as arguments, then just those objects are included in the
information printed.

EXIT STATUS:
This command returns 0 on success and non-zero if an error is encountered.

NOTES:
NONE

EXAMPLES:
The following is an example of using the extension command on a system with no user
extensions.

    SHELL [/] $ extension
    ID   NAME
    ------------------------------------------------------

CONFIGURATION:
This command is included in the default shell command set. When building a custom com-
mand set, define CONFIGURE_SHELL_COMMAND_EXTENSION to have this command included.
This command can be excluded from the shell command set by defining CONFIGURE_SHELL_-
NO_COMMAND_EXTENSION when all shell commands have been configured.

PROGRAMMING INFORMATION:
The extension is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_extension(
        int argc,
        char **argv
    );

The configuration structure for the extension has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_EXTENSION_Command;
5.2.8 task - display information about tasks

SYNOPSYS:

    task [id [id ...] ]

DESCRIPTION:

When invoked with no arguments, this command prints information on the set of Classic API Tasks currently active in the system.

If invoked with a set of ids as arguments, then just those objects are included in the information printed.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use the task on an application with just two Classic API tasks:

    SHELL [/] $ task
    ID   NAME  PRIO  STAT  MODES  EVENTS  WAITID  WAITARG  NOTES
    -----------------------------------------------
    0a010001  UI1   1  SUSP  P:T:nA  NONE
    0a010002  SHELL 100  READY  P:T:nA  NONE

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_TASK to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_TASK when all shell commands have been configured.

PROGRAMMING INFORMATION:

The task is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_task(
        int argc,
        char **argv
    );

The configuration structure for the task has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_TASK_Command;
5.2.9 queue - display information about message queues

SYNOPSIS:

    queue [id [id ... ]]

DESCRIPTION:

When invoked with no arguments, this command prints information on the set of Classic API Message Queues currently active in the system.

If invoked with a set of ids as arguments, then just those objects are included in the information printed.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

NONE

EXAMPLES:

The following is an example of using the queue command on a system with no Classic API Message Queues.

    SHELL [/] $ queue
      ID   NAME  ATTRIBUTES  PEND  MAXPEND  MAXSIZE
      -----------------------------------------------

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_QUEUE to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_QUEUE when all shell commands have been configured.

PROGRAMMING INFORMATION:

The queue is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_queue(
        int argc,
        char **argv
    );

The configuration structure for the queue has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_QUEUE_Command;
5.2.10 sema - display information about semaphores

SYNOPSIS:

    sema [id [id ... ] ]

DESCRIPTION:

When invoked with no arguments, this command prints information on the set of Classic API Semaphores currently active in the system.

If invoked with a set of objects ids as arguments, then just those objects are included in the information printed.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use sema:

    SHELL [/] $ sema

    ID NAME ATTR PRICEIL CURR_CNT HOLDID
    -------------------------------
    1a010001  LBI0  PR:BI:IN  0   1  00000000
    1a010002  TRmi  PR:BI:IN  0   1  00000000
    1a010003  LBI00 PR:BI:IN  0   1  00000000
    1a010004  TRia  PR:BI:IN  0   1  00000000
    1a010005  TRoa  PR:BI:IN  0   1  00000000
    1a010006  TRxa  <assoc.c: BAD NAME>  0   0  09010001
    1a010007  LBI01 PR:BI:IN  0   1  00000000
    1a010008  LBI02 PR:BI:IN  0   1  00000000

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_SEMA to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_SEMA when all shell commands have been configured.

PROGRAMMING INFORMATION:

The sema is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_sema(
        int argc,
        char **argv
    );

The configuration structure for the sema has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_SEMA_Command;
5.2.11 region - display information about regions

SYNOPSIS:

    region [id [id ... ] ]

DESCRIPTION:
When invoked with no arguments, this command prints information on the set of Classic API Regions currently active in the system.

If invoked with a set of object ids as arguments, then just those object are included in the information printed.

EXIT STATUS:
This command returns 0 on success and non-zero if an error is encountered.

NOTES:
NONE

EXAMPLES:
The following is an example of using the region command on a system with no user extensions.

    SHELL [/] $ region
    ID  NAME  ATTR  STARTADDR  LENGTH  PAGE_SIZE  USED_BLOCKS
    -----------------------------------------------

CONFIGURATION:
This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_REGION to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_REGION when all shell commands have been configured.

PROGRAMMING INFORMATION:
The region is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_region(
      int argc,
      char **argv
    );

The configuration structure for the region has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_REGION_Command;
5.2.12 part - display information about partitions

SYNOPSYS:

    part [id [id ... ] ]

DESCRIPTION:

When invoked with no arguments, this command prints information on the set of Classic API Partitions currently active in the system.

If invoked with a set of object ids as arguments, then just those objects are included in the information printed.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

NONE

EXAMPLES:

The following is an example of using the part command on a system with no user extensions.

    SHELL [/] $ part

    ID  NAME  ATTR  STARTADDR  LENGTH  BUF_SIZE  USED_BLOCKS
    ---------------------------------------------

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_PART to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_PART when all shell commands have been configured.

PROGRAMMING INFORMATION:

The part is implemented by a C language function which has the following prototype:

    int rtems_shell_rtems_main_part(
        int argc,
        char **argv
    );

The configuration structure for the part has the following prototype:

    extern rtems_shell_cmd_t rtems_shell_PART_Command;
5.2.13 object - display information about rtems objects

SYNOPSIS:

    object [id [id ...] ]

DESCRIPTION:

When invoked with a set of object ids as arguments, then a report on those objects is printed.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use object:

```
SHLL [/] $ object 0a010001 1a010002
```

```
ID   NAME   PRIO   STAT    MODES EVENTS WAITID WAITARG NOTES
------------------------------------------------------------------------------
0a010001 UI1    1 SUsP  P:T:nA NOne
ID   NAME   ATTR   PRICEIL CURR_CNT HOLDID
------------------------------------------------------------------------------
1a010002 TRmi   PR:BI:IN 0 1 00000000
```

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_OBJECT to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_OBJECT when all shell commands have been configured.

PROGRAMMING INFORMATION:

The object is implemented by a C language function which has the following prototype:

```
int rtems_shell_rtems_main_object(
    int argc,
    char **argv
);
```

The configuration structure for the object has the following prototype:

```
extern rtems_shell_cmd_t rtems_shell_OBJECT_Command;
```
5.2.14 driver - display the rtems device driver table

SYNOPSIS:

    driver [ major [ major ... ] ]

DESCRIPTION:

When invoked with no arguments, this command prints information on the set of Device Drivers currently active in the system.

If invoked with a set of major numbers as arguments, then just those Device Drivers are included in the information printed.

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use driver:

```bash
SHLL [/] $ driver
Major   Entry points
0        init: [0x200256c]; control: [0x20024c8]
         open: [0x2002518]; close: [0x2002504]
         read: [0x20024f0]; write: [0x20024dc]
1        init: [0x20023fc]; control: [0x2002448]
         open: [0x0]; close: [0x0]
         read: [0x0]; write: [0x0]
SHLL [/] $
```

CONFIGURATION:

This command is included in the default shell command set. When building a custom command set, define `CONFIGURE_SHELL_COMMAND_DRIVER` to have this command included.

This command can be excluded from the shell command set by defining `CONFIGURE_SHELL_NO_COMMAND_DRIVER` when all shell commands have been configured.

PROGRAMMING INFORMATION:

The `driver` is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_driver(
    int argc,
    char **argv
);
```

The configuration structure for the `driver` has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_DRIVER_Command;
```
5.2.15 dname - displays information about named drivers

SYNOPSIS:

dname

DESCRIPTION:
This command XXX

WARNING! XXX This command does not appear to work as of 27 February 2008.

EXIT STATUS:
This command returns 0 on success and non-zero if an error is encountered.

NOTES:
NONE

EXAMPLES:
The following is an example of how to use dname:

EXAMPLE_TBD

CONFIGURATION:
This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_DNAME to have this command included.

This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_DNAME when all shell commands have been configured.

PROGRAMMING INFORMATION:
The dname is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_dname(
    int argc,
    char **argv
);
```

The configuration structure for the dname has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_DNAME_Command;
```
5.2.16 pthread - display information about POSIX threads

SYNOPSIS:
   pthread [id [id ...] ]

DESCRIPTION:
When invoked with no arguments, this command prints information on the set of POSIX API threads currently active in the system.

If invoked with a set of ids as arguments, then just those objects are included in the information printed.

EXIT STATUS:
This command returns 0 on success and non-zero if an error is encountered.

NOTES:
This command is only available when the POSIX API is configured.

EXAMPLES:
The following is an example of how to use the task on an application with four POSIX threads:

\[
\text{SHLL [\text{/}] $ pthread}
\]
\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
\text{ID} & \text{NAME} & \text{PRI} & \text{STATE} & \text{MODES} & \text{EVENTS} & \text{WAITID} & \text{WAITARG} & \text{NOTES} \\
\hline
0b010002 & Main & 133 & READY & P:T:nA & NONE & 43010001 & 0x7b1148 \\
0b010003 & ISR & 133 & Wcvar & P:T:nA & NONE & 43010003 & 0x7b1148 \\
0b01000c & & 133 & READY & P:T:nA & NONE & 33010002 & 0x7b1148 \\
0b01000d & & 133 & Wmutex & P:T:nA & NONE & 33010002 & 0x7b1148 \\
\hline
\end{array}
\]

CONFIGURATION:
This command is part of the monitor commands which are always available in the shell.

PROGRAMMING INFORMATION:
This command is not directly available for invocation.
6 Network Commands

6.1 Introduction
The RTEMS shell has the following network commands:

- \texttt{netstats} - obtain network statistics
- \texttt{ifconfig} - configure a network interface
- \texttt{route} - show or manipulate the IP routing table

6.2 Commands
This section details the Network Commands available. A subsection is dedicated to each of the commands and describes the behavior and configuration of that command as well as providing an example usage.
6.2.1 netstats - obtain network statistics

SYNOPSIS:

    netstats [-Aimfpcut]

DESCRIPTION:

This command is used to display various types of network statistics. The information
displayed can be specified using command line arguments in various combinations. The
arguments are interpreted as follows:

- **-A**  print All statistics
- **-i**  print Inet Routes
- **-m**  print MBUF Statistics
- **-f**  print IF Statistics
- **-p**  print IP Statistics
- **-c**  print ICMP Statistics
- **-u**  print UDP Statistics
- **-t**  print TCP Statistics

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

NONE

EXAMPLES:

The following is an example of how to use **netstats**:

The following is an example of using the **netstats** command to print the IP routing table:

```
[1] $ netstats -i

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway/Mask/Hw</th>
<th>Flags</th>
<th>Refs</th>
<th>Use</th>
<th>Expire</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>192.168.1.14</td>
<td>UGS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>eth1</td>
</tr>
<tr>
<td>192.168.1.0</td>
<td>255.255.255.0</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>eth1</td>
</tr>
<tr>
<td>192.168.1.14</td>
<td>00:A0:C8:1C:EE:28</td>
<td>UHL</td>
<td>1</td>
<td>0</td>
<td>1219</td>
<td>eth1</td>
</tr>
<tr>
<td>192.168.1.51</td>
<td>00:1D:7E:0C:DO:7C</td>
<td>UHL</td>
<td>0</td>
<td>840</td>
<td>1202</td>
<td>eth1</td>
</tr>
<tr>
<td>192.168.1.151</td>
<td>00:1C:23:B2:0F:BB</td>
<td>UHL</td>
<td>1</td>
<td>23</td>
<td>1219</td>
<td>eth1</td>
</tr>
</tbody>
</table>
```

The following is an example of using the **netstats** command to print the MBUF statistics:

```
[1] $ netstats -m

************ MBUF STATISTICS ************
mbufs:2048 clusters: 128 free: 63
 drops: 0 waits: 0 drains: 0
free:1967 data:79 header:2 socket:0
pcb:0 rtable:0 htable:0 atable:0
soname:0 soopts:0 ftable:0 rights:0
ifaddr:0 control:0 oobdata:0
```
The following is an example of using the `netstats` command to print the interface statistics:

```bash
$ netstats -f
************* INTERFACE STATISTICS *************
***** eth1 *****
Ethernet Address: 00:04:9F:00:5B:21
Address: 192.168.1.244  Broadcast Address: 192.168.1.255  Net mask: 255.255.255.0
Flags: Up Broadcast Running Active Multicast
Send queue limit: 50  length: 1  Dropped: 0
  Rx Interrupts: 889  Not First: 0  Not Last: 0
  Giant: 0  Non-octet: 0
  Bad CRC: 0  Overrun: 0  Collision: 0
  Tx Interrupts: 867  Deferred: 0  Late Collision: 0
  Retransmit Limit: 0  Underrun: 0  Misaligned: 0
```

The following is an example of using the `netstats` command to print the IP statistics:

```bash
$ netstats -p
************ IP Statistics ************
total packets received 894
packets rcvd for unreachable dest 13
datagrams delivered to upper level 881
total ip packets generated here 871
```

The following is an example of using the `netstats` command to print the ICMP statistics:

```bash
$ netstats -c
************ ICMP Statistics ************
Type 0 sent 843
number of responses 843
Type 8 received 843
```

The following is an example of using the `netstats` command to print the UDP statistics:

```bash
$ netstats -u
************ UDP Statistics ************
```

The following is an example of using the `netstats` command to print the TCP statistics:

```bash
$ netstats -t
************ TCP Statistics ************
connections accepted 1
connections established 1
segs where we tried to get rtt 34
times we succeeded 35
delayed acks sent 2
total packets sent 37
data packets sent 35
data bytes sent 2618
ack-only packets sent 2
total packets received 47
packets received in sequence 12
bytes received in sequence 307
rcvd ack packets 35
bytes acked by rcvd acks 2590
times hdr predict ok for acks 27
times hdr predict ok for data pkts 10
```
CONFIGURATION:
This command is included in the default shell command set. When building a custom command set, define CONFIGURE_SHELL_COMMAND_NETSTATS to have this command included.
This command can be excluded from the shell command set by defining CONFIGURE_SHELL_NO_COMMAND_NETSTATS when all shell commands have been configured.

PROGRAMMING INFORMATION:
The netstats is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_netstats(
    int argc,
    char **argv
);
```

The configuration structure for the netstats has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_NETSTATS_Command;
```
6.2.2 ifconfig - configure a network interface

SYNOPSIS:

```bash
ifconfig
ifconfig interface
ifconfig interface [up|down]
ifconfig interface [netmask|pointtopoint|broadcast] IP
```

DESCRIPTION:
This command may be used to display information about the network interfaces in the system or configure them.

EXIT STATUS:
This command returns 0 on success and non-zero if an error is encountered.

NOTES:
Just like its counterpart on GNU/Linux and BSD systems, this command is complicated. More example usages would be a welcome submission.

EXAMPLES:
The following is an example of how to use `ifconfig`:

```bash
************ INTERFACE STATISTICS ************
***** eth1 *****
Ethernet Address: 00:04:9F:00:5B:21
Address:192.168.1.244 Broadcast Address:192.168.1.255 Net mask:255.255.255.0
Flags: Up Broadcast Running Active Multicast
Send queue limit:50 length:1 Dropped:0
Rx Interrupts:5391 Not First:0 Not Last:0
   Giant:0 Non-octet:0
   Bad CRC:0 Overrun:0 Collision:0
   Tx Interrupts:5256 Deferred:0 Late Collision:0
   Retransmit Limit:0 Underrun:0 Misaligned:0
```

CONFIGURATION:
This command is included in the default shell command set. When building a custom command set, define `CONFIGURE_SHELL_COMMAND_IFCONFIG` to have this command included.

This command can be excluded from the shell command set by defining `CONFIGURE_SHELL_NO_COMMAND_IFCONFIG` when all shell commands have been configured.

PROGRAMMING INFORMATION:
The `ifconfig` is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_ifconfig(
    int argc,
    char **argv
);
```

The configuration structure for the `ifconfig` has the following prototype:
extern rtems_shell_cmd_t rtems_shell_IFCONFIG_Command;
6.2.3 route - show or manipulate the ip routing table

SYNOPSIS:

route [subcommand] [args]

DESCRIPTION:

This command is used to display and manipulate the routing table. When invoked with no arguments, the current routing information is displayed. When invoked with the subcommands add or del, then additional arguments must be provided to describe the route.

Command templates include the following:

route [add|del] -net IP_ADDRESS gw GATEWAY_ADDRESS [netmask MASK]
route [add|del] -host IP_ADDRESS gw GATEWAY_ADDRESS [netmask MASK]

When not provided the netmask defaults to 255.255.255.0

EXIT STATUS:

This command returns 0 on success and non-zero if an error is encountered.

NOTES:

Just like its counterpart on GNU/Linux and BSD systems, this command is complicated. More example usages would be a welcome submission.

EXAMPLES:

The following is an example of how to use route to display, add, and delete a new route:

$ route
Destination Gateway/Mask/Hw Flags Refs Use Expire Interface
default 192.168.1.14 UGS 0 0 0 eth1
192.168.1.0 255.255.255.0 U 0 0 1 eth1
192.168.1.14 00:A0:C8:1C:EE:28 UHL 1 0 1444 eth1
192.168.1.51 00:1D:7E:0C:D0:7C UHL 0 10844 1202 eth1
192.168.1.151 00:1C:23:B2:0F:BB UHL 2 37 1498 eth1
[/] $ route add -net 192.168.3.0 gw 192.168.1.14
[/] $ route
Destination Gateway/Mask/Hw Flags Refs Use Expire Interface
default 192.168.1.14 UGS 0 0 0 eth1
192.168.1.0 255.255.255.0 U 0 0 1 eth1
192.168.1.14 00:A0:C8:1C:EE:28 UHL 2 0 1498 eth1
192.168.1.51 00:1D:7E:0C:D0:7C UHL 0 14937 1202 eth1
192.168.1.151 00:1C:23:B2:0F:BB UHL 2 96 1399 eth1
192.168.3.0 192.168.1.14 UGS 0 0 0 eth1
[/] $ route del -net 192.168.3.0 gw 192.168.1.14
[/] $ route
Destination Gateway/Mask/Hw Flags Refs Use Expire Interface
default 192.168.1.14 UGS 0 0 0 eth1
192.168.1.0 255.255.255.0 U 0 0 1 eth1
192.168.1.14 00:A0:C8:1C:EE:28 UHL 1 0 1498 eth1
192.168.1.51 00:1D:7E:0C:D0:7C UHL 0 15945 1202 eth1
192.168.1.151 00:1C:23:B2:0F:BB UHL 2 117 1399 eth1
**CONFIGURATION:**
This command is included in the default shell command set. When building a custom command set, define `CONFIGURE_SHELL_COMMAND_ROUTE` to have this command included.

This command can be excluded from the shell command set by defining `CONFIGURE_SHELL_NO_COMMAND_ROUTE` when all shell commands have been configured.

**PROGRAMMING INFORMATION:**
The route is implemented by a C language function which has the following prototype:

```c
int rtems_shell_rtems_main_route(
    int argc,
    char **argv
);
```

The configuration structure for the route has the following prototype:

```c
extern rtems_shell_cmd_t rtems_shell_ROUTE_Command;
```
# Function and Variable Index

## C

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIGURE_MALLOC_STATISTICS</td>
<td>50</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_ALIAS</td>
<td>8</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_BLKSYNC</td>
<td>40</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_CAT</td>
<td>35</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_CD</td>
<td>42</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_CHDIR</td>
<td>28</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_CHMOD</td>
<td>34</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_CHROOT</td>
<td>32</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_CONFIG</td>
<td>61</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_CP</td>
<td>23</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_CPUSER</td>
<td>55</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_DATE</td>
<td>9</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_DIR</td>
<td>41</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_DNAME</td>
<td>71</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_DRIVER</td>
<td>70</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_ECHO</td>
<td>11</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_EXTENSION</td>
<td>63</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_ID</td>
<td>13</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_IFCONFIG</td>
<td>77</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_ITASK</td>
<td>62</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_LOGOFF</td>
<td>16</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_LS</td>
<td>26</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_MALLOC</td>
<td>51</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_MDUMP</td>
<td>44</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_MEDIT</td>
<td>46</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_MFILL</td>
<td>47</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_RM</td>
<td>36</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_RMDIR</td>
<td>31</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_ROUTE</td>
<td>80</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_SEMA</td>
<td>66</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_SLEEP</td>
<td>12</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_STACKUSE</td>
<td>56</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_TASK</td>
<td>64</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_TTY</td>
<td>14</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_UMASK</td>
<td>20</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_UNMOUNT</td>
<td>39</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_WDUMP</td>
<td>45</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_WHOAMI</td>
<td>15</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_COMMAND_WKSPACE</td>
<td>59</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_CHDIR</td>
<td>28</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_CHMOD</td>
<td>34</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_CHROOT</td>
<td>32</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_DATE</td>
<td>9</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_DIR</td>
<td>41</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_DNAME</td>
<td>71</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_DRIVER</td>
<td>70</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_ECHO</td>
<td>11</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_EXTENSION</td>
<td>63</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_ID</td>
<td>13</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_IFCONFIG</td>
<td>77</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_ITASK</td>
<td>62</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_LOGOFF</td>
<td>16</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_LS</td>
<td>26</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_MALLOC</td>
<td>51</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_MDUMP</td>
<td>44</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_MEDIT</td>
<td>46</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_MFILL</td>
<td>47</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_MXDIR</td>
<td>29</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_MMOVE</td>
<td>49</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_MOUNT</td>
<td>37</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_NETSTATS</td>
<td>76</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_OBJECT</td>
<td>69</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_PART</td>
<td>68</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_PERIODUSE</td>
<td>57</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_PWD</td>
<td>25</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_QUEUE</td>
<td>65</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_REGION</td>
<td>67</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_RM</td>
<td>36</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_RMDIR</td>
<td>31</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_ROUTE</td>
<td>80</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_SEMA</td>
<td>66</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_SLEEP</td>
<td>12</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_STACKUSE</td>
<td>56</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_TASK</td>
<td>64</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_TTY</td>
<td>14</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_UMASK</td>
<td>20</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_UNMOUNT</td>
<td>39</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_WDUMP</td>
<td>45</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_WHOAMI</td>
<td>15</td>
</tr>
<tr>
<td>CONFIGURE_SHELL_NO_COMMAND_WKSPACE</td>
<td>59</td>
</tr>
</tbody>
</table>

## R

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtems_shell_init</td>
<td>6</td>
</tr>
<tr>
<td>rtems_shell_rtems_main_alias</td>
<td>8</td>
</tr>
<tr>
<td>rtems_shell_rtems_main_blksync</td>
<td>40</td>
</tr>
<tr>
<td>rtems_shell_rtems_main_cat</td>
<td>35</td>
</tr>
<tr>
<td>rtems_shell_rtems_main_cd</td>
<td>42</td>
</tr>
<tr>
<td>rtems_shell_rtems_main_chdir</td>
<td>28</td>
</tr>
<tr>
<td>rtems_shell_rtems_main_chmod</td>
<td>34</td>
</tr>
<tr>
<td>rtems_shell_rtems_main_chroot</td>
<td>32</td>
</tr>
<tr>
<td>rtems_shell_rtems_main_chroot</td>
<td>32</td>
</tr>
</tbody>
</table>
Concept Index

initialization ........................................ 6
# Command Index

## A
- alias .............................................. 8

## B
- blksync ........................................... 40

## C
- cat ............................................... 35
- cd ............................................... 42
- chdir ............................................ 28
- chmod ........................................... 33
- chroot .......................................... 32
- config .......................................... 61
- cp ............................................... 21
- copen .......................................... 54

## D
- date .............................................. 9
- dir ............................................... 41
- dname .......................................... 71
- driver ........................................... 70

## E
- echo ............................................. 10
- exit ............................................ 17
- extension ..................................... 63

## I
- id ............................................... 13
- ifconfig ....................................... 77
- itask .......................................... 62

## L
- logoff .......................................... 16
- ls ................................................ 26

## M
- malloc ........................................... 50
- mdump .......................................... 44
- medit .......................................... 46
- mfill ........................................... 47
- mkdir .......................................... 29
- mmove .......................................... 49
- mount .......................................... 37

## N
- netstats ........................................ 74

## O
- object .......................................... 69

## P
- part ............................................. 68
- periduse ....................................... 57
- pthread ........................................ 72
- pwd .............................................. 25

## Q
- queue .......................................... 65

## R
- region .......................................... 67
- rm ............................................... 36
- rmdir .......................................... 31
- route .......................................... 79

## S
- sema ............................................ 66
- sleep ........................................... 12
- stackuse ....................................... 56

## T
- task ............................................. 64
- tty ............................................. 14

## U
- umask .......................................... 20
- unmount ....................................... 39

## W
- wdump .......................................... 45
- whoami ......................................... 15
- wkspace ....................................... 59