U S E R ' S G U I D E

CAPBAK/UNIX

Version 3.3

Test Capture / Playback System for
UNIX

SOFTWARE RESEARCH, INC.

This document property of:

Name:_____

Company:_____

Address:_____

_____

Phone_____

SOFTWARE RESEARCH, INC.

625 Third Street
San Francisco, CA 94107-1997
Tel: (415) 957-1441
Toll Free: (800) 942-SOFT
Fax: (415) 957-0730
E-mail: support@soft.com
http://www.soft.com

Documentation: Adam Heilbrun

(Last Update April 7, 1997)
documentation/user-manuals/regression/97UNIXbooks/capbakUNIX/97capbakunix.b

Table of Contents

List of Figures

Preface

Congratulations!

By choosing the TestWorks integrated suite of testing tools, you have taken the first step in bringing your application to the highest possible level of quality.

Software testing and quality assurance, while becoming more important in today's competitive marketplace, can dominate your resources and delay your product release. By automating the testing process, you can assure the quality of your product without needlessly depleting your resources.

Software Research, Inc. believes strongly in automated software testing. It is our goal to bring your product as close to flawlessness as possible. Our leading-edge testing techniques and coverage assurance methods are designed to give you the greatest insight into your source code.

TestWorks is the most complete solution available, with full-featured regression testing, coverage analyzers, and metric tools.

Audience

This manual is intended for software testers who are using *CAPBAK/UNIX* tools. You should be familiar with the X Window System and your workstation.

Content of Chapters

Chapter 1	*SYSTEM OPERATION* explains how *CAPBAK/UNIX* operates and describes its major operating modes.
Chapter 2	*CAPBAK/UNIX INVOCATION AND USE describes how CAPBAK/UNIX* is used and how it is similar to the on-line manual pages supplied with the system.
Chapter 3	*CAPBAK/UNIX ASCII MENU OPERATION* explains the interactive menu system, which is designed to assist novices in the use of the *CAPBAK/UNIX* recording and playback tools.
Chapter 4	*FILE DESCRIPTIONS* describes *CAPBAK/UNIX* files.
Chapter 5	*CONFIGURATION FILE PROCESSING* describes how to construct or edit *CAPBAK/UNIX* configuration files.
Chapter 6	*CONDITIONAL PLAYBACK PROGRAMMING* describes how a user can create “conditional keysave files,” which behave during playback according to the results of various system calls.
Chapter 7	*USING CONDITIONAL PLAYBACK* describes some special conventions involved with playback programming.
Chapter 8	*ERROR HANDLING* describes the provisions that *CAPBAK/UNIX* has for certain kinds of error processing.
Chapter 9	*keycvt* describes **keycvt**, which in an ancillary tool to **Software Research’s** *CAPBAK/UNIX*; a keystroke capture / playback tool used primarily for software testing.

Typefaces

The following typographical conventions are used in this manual.

boldface — Introduces or emphasizes a term that refers to TestWorks' window, its sub-menus and its options.

italics — Indicates the names of files, directories, pathnames, variables, and attributes. Italics is also used for manual and book titles.

"Double Quotation Marks" — Indicates chapter titles and sections. Words with special meanings may also be set apart with double quotation marks the first time they are used.

`courier` — Indicates system output such as error messages, system hints, file output, and *CAPBAK/X*'s keysave file language.

`Boldface Courier` — Indicates any command or data input that you are directed to type. For example, prompts and invocation commands are in this text. (For instance, **stw** invokes TestWorks.)

CHAPTER 1

System Operation

This section explains how **CAPBAK/UNIX** operates and describes its major operating modes.

1.1 System Information Flow

Figure 1 shows a data flow diagram of the CAPBAK/UNIX system. Note that there are three separate types of CAPBAK/UNIX operation:

- Recording a keystroke sequence for later use.
- Recording an interactive session.
- Playing back an interactive session.

All parts of the **CAPBAK/UNIX** system can be command-line driven, and are designed to be usable with the standard UNIX pipeline and redirection facility.

In addition, a simple and friendly menu system (the user interface for the interactive version of **capbak**) assists novice users in creating special "configuration" files which record the selection of run-time parameters. It also helps in executing the programs with on-line help.

1.2 Operating Modes

As Figure 1 suggests, there are two main modes for CAPBAK/UNIX operation. In the first working mode, the CAPBAK / UNIX commands are used in a UNIX pipeline to an application. Keystrokes are captured during an initial run, and played back on successive runs.

In the second working mode the CAPBAK / UNIX system acts as a ‘‘terminal emulator" with either a special loop-back cable or a direct-connection to the user-selected ports. In this mode, an entire interactive session can be captured.

In either mode it is possible to run the commands ‘‘in the background" with the ‘‘&" operator. This makes it possible to provide for multiple parallel session simulations, using several different terminals, if desired.

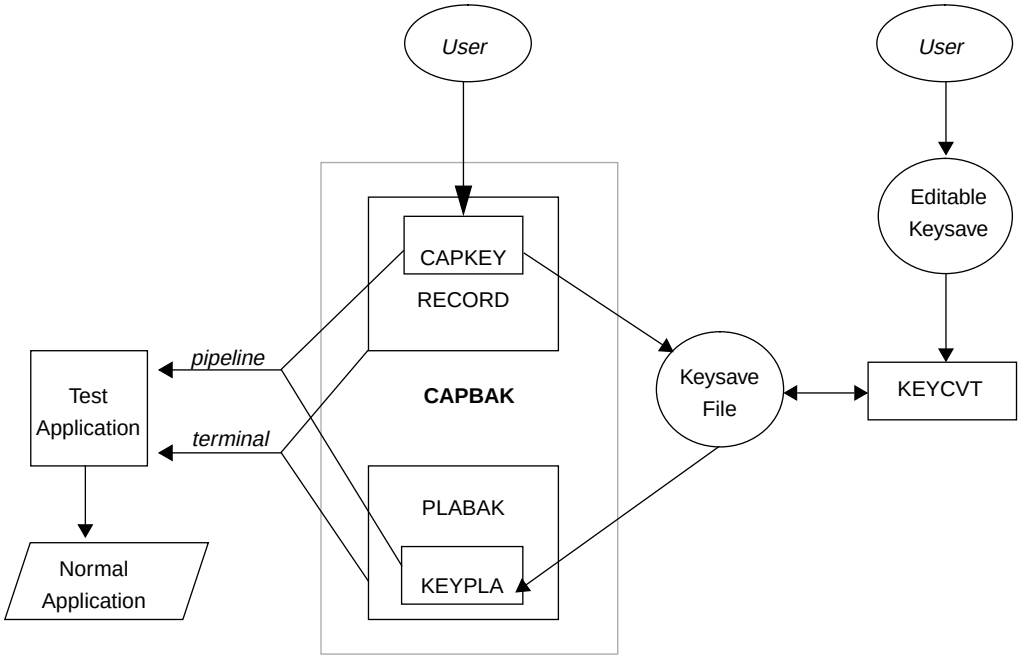


FIGURE 1 CAPBAK/UNIX System Diagram

1.3 CAPBAK/UNIX Function Descriptions

The CAPBAK/UNIX package consists of three main systems: **capbak, record,** and **plabak,** which can be used individually or with the **capbak** interactive menu system. In addition, there are several other sub-functions that can be used independently.

1.3.1 The Interactive capbak Command

The interactive menu system is designed to assist novice users in use of the CAPBAK/UNIX record and playback tools.

The menu has command-line options to select files, but its main input is from the user, who can change many options before running the record and playback commands (see below). These options can be displayed and saved to a configuration file for subsequent use.

The **capbak** command also provides on-line help with descriptions of the options and modes that can be configured.

The basic record and playback commands can be invoked from within the **capbak** menu by simply selecting the mode — record or playback — and choosing the **go** option from the main menu.

1.4 Batch Mode

Interactive **capbak** can also be run in **batch** mode, using menu commands saved in a simple text file and ‘‘redirected" into **capbak** using ‘‘<". For example, if you have recorded a session and want to play it back, the following commands could be put into a file called **cmds**:

```
mode
playback
exit
go
no
exit
exit
```

capbak would then be invoked as follows:

```
capbak < cmds
```

This means that other processes, as well as other users, can set up and run the playback tools in batch mode — an advantage for using with *SMARTS* to perform test suite management. Note that this type of batch mode processing works well with **playback** mode, but not **record**.

1.4.1 The record Command

The **record** command is a program that records keystrokes being entered at a terminal and saves them in the **SR**-standard keysave file format. It records and displays the responses from the remote machine, and saves them in a baseline file which can be used to synchronize playback. When **record** saves keystrokes it adds timing information.

1.4.2 The capkey Command

The **capkey** command allows you to capture keystrokes in a keysave file with or without actually being attached to an application, via a pipeline or terminal. No response file is generated in this case. The **capkey** command provides a way to generate keysave files independent of the user actually ‘‘doing" anything. It simply records keystrokes in a keysave file (of course, with timing information).

1.4.3 Marker Mode Operation

By permitting messages to be included in the keysave file, marker mode assists testers using *CAPBAK/UNIX* to support complex testing tasks.

Marker Mode During Recording

You can use marker mode during recording, with either **capkey** or **record**. Marker mode messages can be up to one line (80 characters) long, and can be used to log what is going on in the test session.

Marker mode messages are recorded in the generated keysave file, and can be edited after the keysave file is converted into "visible form" with **keycvt**.

In marker mode during recording the user receives the prompt:

```
Enter Marker Text (one more INTR to end program):
```

after each interrupt is typed. You can type a message of up to 80 characters (only the first 80 characters are used) and then press RETURN. If you change your mind and decide not to type in a message, simply press RETURN, and no message will be written to the keysave file.

Marker Mode During Playback

During playback with **keypla** or **plabak**, when marker mode is selected each marker message will be displayed on the screen. When marker mode is not selected, any recorded marker messages will not be displayed. They will, however, appear in the visible version of the keysave file when converted with **keycvt**. They can be useful for keysave file identification.

1.4.4 The keypla Command

The **keypla** command can be used to read a keysave file and emit the characters to the screen. It is the logical ‘‘reverse" of the capkey command.

1.4.5 The plabak Command

The **plabak** command reads a keysave file and sends the keystrokes to the remote machine, at the same speed as they were typed in during the creation of the keysave file.This playback speed can be reduced or increased either permanently, using **keycvt**, or temporarily, using a command-line option.

The remote machine is connected to the playback machine with RS-232 ‘‘straight" cable serial connections; if required, an unlimited number of ports can be connected in this way, with one *CAPBAK/UNIX* process allocated to each activity.The user selects which port to use, and which keysave file to send. The playback system thus simulates many users interacting with the remote machine.

Note that **plabak** can function (re-directing its output to a file) as a background task. Because of this, many combinations can be used provided only *one* **plabak** program is used for each port.

The keystrokes will cause the remote machine to respond by sending information or screens of data, and this is saved in a special file called a ‘‘response" file. By comparing response files using *EXDIFF*, the user can determine if the play back was successful.

1.4.6 The keycvt Utility

This utility enables the user to convert keysave files generated by *CAPBAK/UNIX* into ASCII form, and back. These files can then be edited and, if required, transported to another environment such as DOS (or vice versa). **keycvt** provides compatibility between DOS and UNIX testing.

However, only relative inter-keystroke timing will be faithfully transported across environment boundaries; absolute delay figures are environment-dependent.

1.4.7 Distinguishing capkey from record

There is an important difference between **capkey** and **record**; the difference lies in whether or not the unit that you are testing is on the same machine you are operating from.

Normally you use **capkey** to test a system which is present on the same computer you are operating from. This is done with the UNIX pipeline capability as described below. It is not possible to record logging in with **capkey** as it is with **record**. You use **record** when you wish to record all of the things you do during a terminal session, in which the computer you are connected to is, in turn, connected to a second computer containing the application under test.

It is also possible to use **record** to record a session that is being run on the same computer through use of a loopback connection. This connection involves using a standard 25-pin connector cable and a null modem (a connector with pins 2 and 3 crossed). This requires the use of two ports on the machine you are using (in addition to the port serving as the user console).

Suppose you are connected to **/dev/ttya**. During your session with **record** you designate **/dev/ttyb** as your outgoing port. Connect the loopback cable from **/dev/ttyb** to **/dev/ttyc**. Now, when you start your session you should get a login prompt, assuming your system is configured to allow logins on **/dev/ttyc**.

Note that **/dev/ttyb** should not be a login port. Record the session normally. UNIX will behave just as it would if you were connected to a terminal attached to /**dev/ttyc**. The diagram below shows how this is organized.

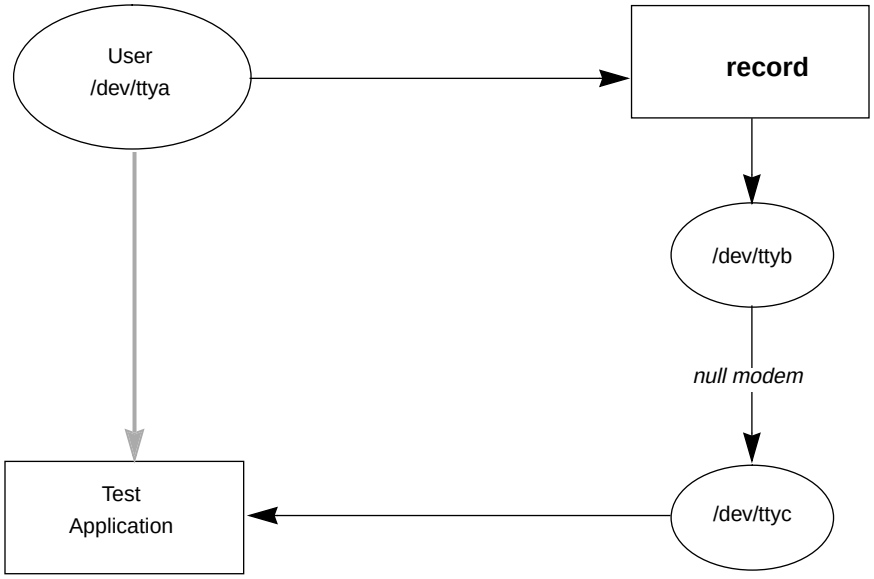


FIGURE 2 Distinguishing capkey from record

APPENDIX A provides a more detailed example of how this can be set up on a SUN-3 or SUN-4 or SUN-SPARC UNIX system.

CHAPTER 2

CAPBAK/UNIX Invocation and Use

This section describes how *CAPBAK/UNIX* is used, and is similar to the on-line manual pages supplied with the system.

2.1 Introduction

There are two ways to use CAPBAK/UNIX commands: in batch mode from the command-line, and in interactive mode from the capbak menu system. In either mode the commands may retrieve parameter information from a configuration file, normally called **`capbak.rc`**. In interactive mode **capbak** will assist the user in constructing such a configuration file.

2.2 Command-line Processing

Command-line processing is used to provide “batch mode" like operation of the CAPBAK/UNIX system commands. These commands can be used with or without a previously-created configuration file.

Each part of the CAPBAK/UNIX system is described in the familiar “manual page" format on the pages that follow.

2.2.1 capbak Syntax

The following command syntax, when used without arguments or flags, invokes the interactive mode version of **capbak**, described below.

Syntax

capbak -k *file* [**-b** *file*] [**-B** *baud*] [**-c** *n*|**-C**][**-d** *f*][**-e | -E**] **-f** *n* **-F** **-g** *n* -G [**-i** *file*] **l** *file* -L [**-m | -M**] [**-p** *port*] **-q** **-Q** [**-r** *file*]

Options and Parameters:

-k *file* — Use *file* to save keystrokes or play them back. This argument is required unless the keysave file is specified in the configuration file.

-b *file* — Use *file* to save program responses in record mode. This file can subsequently be used as a baseline for comparison with play back responses saved using the *-i* option (play back mode). The default is **capbak.bsl**.

-B *baud* — Baud rate. The default is 9600.

-c *n* — Delay *n* seconds after each carriage return played back; n must be an integer.

-C — Turn carriage return delay OFF.

-d *n* — Use *n* as the playback ‘‘delay" factor. Playback rate can be slowed down or speeded up. A factor of 1 causes no slowdown, or ‘‘faithful time playback". A factor of 0 results in the fastest possible playback, i.e. with zero delays between keystrokes.

The value specified is multiplied by the recorded keystroke timing, and the product will be the delay simulated on playback. The *n* parameter can be either an integer or floating point number; for example 1.5, 0.001, 20, 300, 0.5, etc. Negative values are ignored.

-e — Turn on echo mode to display what is being typed.The default is no echo.

-E — Turn echo mode OFF (default).

-f *n* — Floor time delay, an integer. This is the MINIMUM time delay between characters issued. Caution: The value used must not be larger than the ceiling value.

-F	Turns off the floor time delay described above.
-g *n*	Ceiling time delay, an integer. This is the MAXIMUM time delay between characters issued. Any delay longer than the ceiling value will be 'clipped' to that value. Caution: The value used must not be smaller than the floor value.
-G	Turns off the ceiling time delay described above.
-i *file*	Use *file* to save the responses from the remote machine in playback mode. The default is **capbak.rsp**.
l *file*	(Used for Conditional / Interpretive mode.) The name of the keystroke logfile. If this switch is present then the actual keystrokes issued (including actually issued delay values) will be placed in the named logfile. The **-l** switch works with the **-a** or **-q** switches. (See section 7.5.4).
-L	Used for Conditional / Interpretive mode. Turns off logging to the logfile described above.
-m	Display marker text entered during a record session.
-M	Turn marker text display OFF.
-p *port*	Use the specified *port*, e.g. **/dev/tty07**, **/dev/ttya**, etc.instead of the default **/dev/tty01**.
-q	Used for Conditional / Interpretive mode. Query / checkout flag. Uses the interpretive mode as with the **-a** option except that **capbak** asks the user to provide the logical values when a conditional system call is made. The actual system call text is provided, but the system call is NOT made.
-r *file*	Use *file* as the configuration file. If no file is given, **capbak** looks for the default configuration file **capbak.rc**. Note that command-line options will override configuration file options if both are active.
(no options)	Invoke **capbak** interactively, reading parameters from the default configuration file (**capbak.rc**).

Example Commands

`capbak -k keysave.ksv < cmd.file`

Reads interactive commands from cmd.file and uses **`keysave.ksv`** as the keysave file (use for playback only).

`capbak`

This command invokes the interactive mode for **`capbak`** as long as the default configuration file **`capbak.rc`** is available.

`cat cmd.file | capbak > output.file`

This feeds a **`cmd.file`** into **`capbak`** and redirects output to **`output.file`**.

Limitations

`capbak` operates as a kind of ‘‘terminal emulator" and employs the capabilities of **`capkey`** and **`keypla`** in its operation.

All limitations of these commands are inherent in the operation of **`capbak`**.

2.2.2 record Syntax

The **record** command is used to record a test session. The command-line invocation starts up a **recording** session; a **CTRL-D** or interrupt ends the session and releases the resulting keysave and response files (the key to be used for interrupt is user- programmable.)

Syntax

record ***-k*** *file* ***[-b*** *file****][-B*** *baud****][-p*** *port****][-r*** *file****]***

Options and Parameters

-k *file*	Record keystrokes in file. This argument is required unless the keysave file is specified in a configuration file.
-b *file*	Use *file* to save program responses in record mode. This file can subsequently be used as a baseline for comparison with playback responses saved with the *-i* option (playback mode). The default is **capbak.bsl**.
-B *baud*	Baud rate. The default is 9600. This argument is required unless the keysave file is specified in a configuration file.
-p *port*	Use the specified *port*, e.g. **/dev/tty07**, **/dev/ttya**, etc., instead of the default **/dev/tty01**.
-r *file*	Use *file* as the configuration file. If no file is given, **capbak** looks for the default configuration file **capbak.rc**. Note that command-line options will override configuration file options if both are active.

Example Commands

record -k keysave.ksv -p /dev/tty03 -b 4800

Use **keysave.ksv** as the keystroke file for port **/dev/tty/03** at 4800 baud.

record -b 9600 -r record.rc

Run **record** at 9600 baud, and read all other parameters from the file **record.rc**.

record Run **record** using parameters read from the configuration file **capbak.rc**.

Capture-time Behavior

Marker message mode permits a user to record messages in the keysave file that are not sent on to the application. If you press interrupt, or type in the interrupt sequence (see below) you enter **marker mode**, and *CAPBAK/UNIX* prompts you for optional marker text. Null messages, i.e. pressing RETURN without typing in any message, are ignored.

Record mode can be exited by pressing the interrupt key twice or by typing CTRL-D.

2.2.3 capkey Syntax

The command-line style invocation sets up **capkey** to record; pressing CTRL-D or a double interrupt ends the recording and releases the resulting keysave file.

Syntax

capkey -k *file* **[-r** *file***] [-e | -E]**

Options and Parameters

k *file* — Record keystrokes in *file*. This argument is required unless the keysave file is specified in the configuration file.

-r *file* — Use *file* as the configuration file. If no file is given, **capbak** looks for the default configuration file **capbak.rc**. Note that command-line options will override configuration file options if both are active.

-e — Turn on echo mode so that the user can ‘‘see" what is being typed. The default is *no* echo.

-E — Turn echo mode OFF (default).

Example Commands

capkey -k keysave.ksv

This command captures keystrokes into the keysave file **keysave.ksv**, and echoes the keystrokes to the screen.

capkey -k keysave.ksv | ed

This command captures keystrokes into the keysave file **keysave.ksv** and feeds the characters into the editor **ed**. Note that **capkey** cannot pipe to an interactive process. *CAPBAK* may be used to drive an interactive process through the **record** and **plabak** commands, with proper hardware connections.

Capture-time Behavior

When in **marker mode**, typing an interrupt will initiate a prompt for optional marker text. The material typed until the next RETURN is copied into the keysave file as a message. Null messages (i.e. a RETURN without typing a message) are ignored.

Pressing the CTRL-D key or the user-programmed interrupt sequence will terminate input and return control to the user.

capkey can be pipelined to another UNIX process; i.e. the keys you type are returned to standard output.

Limitations

capkey sends characters from the keyboard down the pipeline (or to standard output). This means that unexpected things can happen when you use the UNIX pipeline to feed certain kinds of applications. For example, if there is an error sensed by the application, in some UNIX systems this will shut down the pipeline

2.2.4 keypla Syntax

keypla reads a stated keysave file and emits the keystrokes, at the recorded intervals, to the screen. The command-line style invocation sets up **keypla** to start playback activity.

Syntax

keypla -k *file* **[-d** *n* **] [-e |-E] [-m | -M] [-r** *file***]**

Options and Parameters

-k *file* — Play back the keystrokes saved in file. This argument is required unless the keysave file is specified in a configuration file.

-d *n* — Use *n* as the playback ‘‘delay" factor. Playback rate can be slowed down or speeded up. A factor of 1 causes no slowdown, or ‘‘faithful time playback". A factor of 0 results in the fastest possible playback, i.e. with zero delays between keystrokes.

The value specified is multiplied by the recorded keystroke timing, and the product will be the delay simulated on playback. The *n* parameter can be either an integer or floating point number, for example 1.5, 0.001, 20, 300, 0.5, etc. Negative values are ignored.

-e — Turn on echo mode so that output is sent the user's screen as well as to wherever the output is piped.

-E — Turn echo mode OFF (default).

-m — Display marker text entered during a record session.

-M — Turn marker text display OFF.

-r *file* — Use *file* as the configuration file. If no file is given, capbak looks for the default configuration file **capbak.rc**. Note that command-line options will override configuration file options if both are active.

Example Commands

keypla -k my.keysave

This command plays keystrokes from **my.keysave**, and echoes them to the screen.

keypla -k my.keysave -e | ed

This command plays keystrokes from **my.keysave**, echoes them to the screen, and passes them to the ed command.

keypla -k keysave.ksv -d 0.5 | ed

This command plays keystrokes from **keysave.ksv**, echoes them to the screen using a 0.5 slowdown delay (i.e. twice as fast as they were recorded), and passes them to the **ed** command.

Playback-time Behavior

The keystrokes recorded in the keysave file are read, time delays are inserted, and the keystrokes are emitted to standard output. **keypla** can be pipelined to another UNIX process; i.e. the keys played back are returned to standard output and can be pipelined into the application under test.

2.2.5 plabak Syntax

plabak replays a previously-recorded session. The command-line invocation allows setup of the system and provides for various playback time options.

Syntax

plabak -k *file* ***[-B*** *baud****] [-c*** *n* ***| -C] [-d*** *n****] [-e| -E]***
[-i *file****] [-m | -M] [-p*** *port****] [-r*** *file****]***

Options and Parameters

-k *file*	Play back the keystrokes saved in *file*. This argument is required unless the keysave file is specified in a configuration file.
-B *baud*	Baud rate. The default is 9600.
-c *n*	Delay *n* seconds after each carriage return played back; n must be an integer.
-C	Turn carriage return delay OFF.
-d *n*	Use *n* as the playback ‘‘delay" factor. Playback rate can be slowed down or speeded up. A factor of 1 causes no slowdown, or ‘‘faithful time playback". A factor of 0 results in the fastest possible playback, i.e. with zero delays between keystrokes. The value specified is multiplied by the recorded keystroke timing, and the product will be the delay simulated on playback. The *n* parameter can be either an integer or floating point number, for example 1.5, 0.001, 20, 300, 0.5, etc. Negative values are ignored.
-e	Turn on echo mode so that the user can ‘‘see" what is being typed. The default is no echo.
-E	Turn echo mode OFF (default).
-i *file*	Use *file* to save the responses from the remote machine in playback mode. The default is **capbak.rsp**.
-m	Display marker text entered during a record session.
-M	Turn marker text display OFF.

-p *port* — Use the specified *port*, e.g. **/dev/tty07**, **/dev/ttya**, etc., instead of the default **/dev/tty01.**

-r *file* — Use *file* as the configuration file. If no file is given, capbak looks for the default configuration file **capbak.rc**. Note that command-line options will override configuration file options if both are active.

Example Commands

plabak -k my.ksv -p /dev/tty02 -b 4800

This command plays back keystrokes from **my.ksv** using port **/dev/tty02**, which is set to operate at 4800 baud.

plabak -r special.rc -p /dev/ttyb -b 1200

Keystrokes are selected from the files named in the configuration file **special.rc**. Port **/dev/ttyb** is used at 1200 baud.

Playback-time Behavior

The interrupt key or user-defined interrupt sequence can be used to terminate the playback session. The **plabak** program has many components running simultaneously and as a result there may be some delay before all the process are terminated. Files are closed, and data recorded up to that point is saved in the appropriate files.

CHAPTER 3

CAPBAK/UNIX ASCII Menu Operation

The interactive menu system is designed to assist novices in use of the *CAPBAK/UNIX* recording and playback tools.

3.1 Overview

The menu has command-line options to select files, but its main input is from the user, who can change many options before running the recording and play back commands (see next page). These options can be displayed and saved to a configuration file for subsequent use.

The **capbak** command also provides on-line help with descriptions of the options and modes that can be configured.

The basic record and play back commands can be invoked from within the **capbak** menu by simply selecting the mode--record or playback--and choosing the **go** option from the main menu. This chapter describes how CAPBAK/UNIX is used in interactive mode.

3.2 General Interactive Mode Description

CAPBAK / UNIX provides access to the two basic primitives (described in the prior section) through use of an interactive ASCII menu scheme. Many users find this mode of operation more convenient than using the commands directly, particularly because much of the repetitive work can be handled at the **`capbak.rc`** file level.

3.2.1 Organization of CAPBAK/UNIX Menus

The organization and structure of the menus for the interactive **`capbak`** is shown in the diagram above.

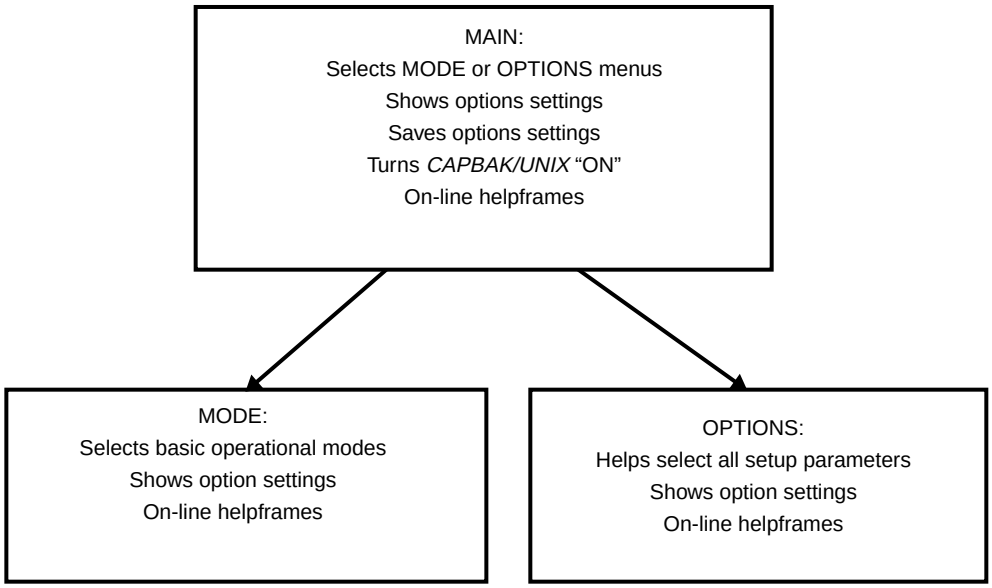


FIGURE 3 Organization of Menus

3.2.2 Invoking CAPBAK for Interactive Processing

Start up **capbak** with the command

```
capbak
```

with no parameters. **capbak** will then assume you wish the interactive mode of operation.

After *CAPBAK/UNIX* starts up, you will see the title information and the prompt: **"MAIN:".** You are then in **capbak** interactive mode.

If you wish to see the available menu options at any time type RETURN to display the options for that menu; this works for all menus.

You can display the current settings known to **capbak** at any time using the settings command, get on-line help with the **help** command, and exit the current menu using **exit**.

3.2.3 Running UNIX Commands

In addition the user may execute a system command using the "!" symbol, e.g.:

```
MAIN:!sort myfile
```

The configuration file read in the settings is automatically used. However, the settings can be changed if required (see MAIN menu, below).

3.2.4 OPTIONS Printout

The current set of options values is available from all *CAPBAK/UNIX* menus, using the **settings** command. An example of the settings display is:

```
OPTIONS:
FILES:
      Config file  = capbak.rc
     Keysave file  = t.ksv
    Response file  = t.asc
MODES:
           Marker  = ON
```

3.2.5 MAIN Menu

All commands may be abbreviated when no ambiguity exists, e.g. options can be shortened to **o** because no other command in the **MAIN** menu starts with **o**.

When **capbak** is activated the following menu options are displayed when a RETURN is typed:.

```
          mode --   Select the mode of operation
       options --   Select and set configuration parameters.
            go --   Run CAPBAK/UNIX.
          save --   Save the current settings for CAPBAK/UNIX.
      settings --   List the current settings for CAPBAK/UNIX.
    help <opt> --   Display HELP text for a command.
          exit --   Exit to the system.
```

3.2.6 MODES Menu

If the mode option is selected then the following menu is displayed when any unrecognized character followed by a RETURN is typed:

```
MODES:
              record --  Toggle the record mode on
               query --  Toggle query & play ASCII keysavefile on/off
               ASCII --  Toggle interpret & play ASCII keysave file on/
                                          off.
            playback --  Toggle the playback mode on/off.
            settings --  list current settings for CapBakUNIX.
          help <cmd> --  Display HELP text for command.
                exit --  Exit current level.
```

3.2.7 OPTIONS Menu

If the options menu is selected then the following menu is displayed when any unrecognized character followed by a RETURN is typed:

```
OPTIONS:
       config <file> --  File to read for configuration parameters.
     keysave <file> --  File in which to save the keystrokes.
     response<file> --  File in which to save the responses.
     logfile <file> --  Log file for played back keystrokes (ascii
                                          mode).
     baseline<file> --  File to use as the baseline.
        delay <opt> --  Set the delay factor to slow down playback.
         baud <opt> --  Set the baud rate for the ports.
         port <opt> --  Select which port to use.
               echo --  Echo the responses in playback mode.
           settings --  list current settingsfor CapBak/UNIX.
         help <opt> --  Display HELP text for command.
  interrupt <string>--  1 to 2 char string to interrupt record for com-
                                          ment.
  terminate <string>--  1 to 2 char string to terminate record.
 Keyboard<break/raw>--  Set user keyboard to break/raw mode.
             crdelay --  Delay before (negative) or after carriage
                                          return.
                exit --  Exit current level.
```

3.2.8 Saving Changed Option Settings

Before leaving *CAPBAK/UNIX* or running a recording or playback session, the user will be prompted to save the current settings (unless this has already been done):

```
MAIN:
Do you want to save the current settings (y/n)?:
y
```

CHAPTER 4

File Descriptions

This section describes **CAPBAK/UNIX** files. **CAPBAK/UNIX** deals with two kinds of files: keysave files, and captured (baseline or response) files.

4.1 Keysave Files

A keysave file represents the keystroke information that was recorded by use of **`record`**. The keysave files used by **`plabak`** are in binary format, and cannot be read, printed, or edited with the standard ASCII tools.

To edit keysave files, you must use the special utility, **`keycvt`**, to change the ‘‘invisible'' format to a ‘‘visible'' format.The definition of these formats is described below, but for further information please consult the *User Manual for keycvt.*

4.1.1 Keysave File Visible Format

The section below shows how a keysave file converted into ASCII format by **keycvt** appears when being edited.

Overall Structure

The visible keysave file is a sequence of character groups presented in the following format:

character character ...
character [time] character [time] ...
{special} [time] {special} [time] ...
{special} [time] character [time] ...
{{message}} ...

Note that the [*time*] field is optional ([0] is assumed).

Blanks *are* significant (this is how a sequence of blanks is recorded!). Therefore:

[0] [0] [0] [0] [0] [0]

is seen as five blanks with zero delays, and no carriage return at the end.

Character Processing

Ordinary: The ordinary ASCII character set passes through with no special treatment.

\[equals ‘‘[“ as an actual character

\{equals ‘‘{“ as an actual character

\ \equals “\”

any_other_character is ignored when converted to ‘‘invisible” format. For example: \<newline> for formatting.

[*time*] [0] means ‘‘zero delay”.

[*n*] *n* < 32767, expressed in standard delay units.

{*special*}

{F*n*}	Function Key F*n* (if used)
{CR}	Carriage Return
{LF}	Line Feed
{ESC}	Escape
{KEYPAD *n*}	Keypad *n*
{CTRL DEL}	Control-Delete, etc.
{INTR}	Interrupt Key
{SHIFT CTRL DEL}	Shift-Control-Delete
{ALT CTRL SHIFT M}	Alt-Control-Shift-M

Note that simple keys are single letters.

There can be up to four keys held down simultaneously in a {*special*} group (although this may not be legal).

{{*message*}} This is a free-text message that can be included in the keysave file visible format, which is included (but not used) in the ‘‘invisible" format.

4.1.2 Invisible Format

Sometimes it is important to know what the internal format for keysave files is. In the internal format which is actually used by **`plabak`** there are:

- 2 bytes of character information (four hex symbols). This allows for all of the unusual keystroke combinations. This supports "extended ASCII" as well as other modes.
- 2 bytes of timing information (16 bits). This is always treated as a 16-bit positive integer. The meaning varies depending on the system but usually is a count of the number of milliseconds of delay.

The internal details of keysave file representations vary with machine and environment. However, the ASCII version (see above) is universal.

Moreover, during conditional playback only the ASCII version of the file can be used (the conversion to internal format is done automatically).

A special internal format is used to store messages that may arise during the keysave generation process or may have been added while editing the visible version of the file.

4.2 Captured Files

There are two kinds of captured files produced by *CAPBAK/UNIX*: baseline files and response files. They have the same format but are used in different ways.

4.2.1 Baseline Files

The baseline files created by *CAPBAK/UNIX*'s **`record`** command have the following format:

```
marker record
response data (from remote machine)
marker record
more response data
marker record
.
.
.
etc.
```

There is one marker record inserted in the stream of output each time the record senses a RETURN key. The idea is that the RETURN should be pressed when the target system has completed its output. The marker record is used to assist in handling control flow of information during playback operations.

4.2.2 Response Files

Response files follow the same format as baseline files but do not have the marker records inserted. In effect, a response file is the normal sequence of characters issued by the target system in response to keystrokes that are recorded by **`record`**.

4.2.2.1 Response File with Terminal Control Data

The response file may contain special control sequences that are used by your UNIX system to manipulate the cursor and the display.

Because *CAPBAK/UNIX* records the response file (or the baseline file) as a complete record of what has been received by the port, this file may need some post-processing before you can "see" what the information actually looks like.

In other words, you might have to strip away the control characters and possibly map the "clear screen" command (which varies from terminal to terminal) into a "line feed" command before you can successfully compare two outputs. You may even want to have each new screen broken up into a separate file for ease of processing.

In sum, it will interpret other control sequences properly; i.e. the response file will play back.

Shown on the following pages is a sample C program that accomplishes this, assuming that you are using a WYSE 50 type terminal. For other terminal types, you will have to change the mappings so that they accomplish the intended result.

```
/*
CAPBAK/UNIX: Utility Program:  “convert.c”

This simple program maps a file that is a response file for a WYSE-50
terminal into ASCII-readable format.  This source program should be changed
in case another terminal type is being used.

Once converted, the resulting output file is safe to be processed with other
file shaping and manipulation steps.
*/

#include <stdio.h>
FILE *ip = NULL,*op = NULL;
main(argc,argv)
int argc;
char **argv;
{ int index,cc;
if (argc > 1)
{
if ((ip = fopen(argv[1],”r”)) == NULL)
{
fprintf(stderr,”Cannot open %s for input\n”,
argv[1]);
}
if (argc > 2)
{
if ((op = fopen(argv[2],”w”)) == NULL)
{
fprintf(stderr,”Cannot open %s for output\n”,
argv[2]);
}
} else op = stdout;
if (argc > 3)
for (index = 3; index < argc; ++index)
{
fprintf(stderr,”Ignoring argument %s\n”,
argv[index]);
}
} else
{
ip = stdin;
op = stdout;
}
if ((ip == NULL) || (op == NULL)) exit();
while ((cc = fgetc(ip)) != EOF)
{
if (cc == 0x1b)
{
fputc(‘\n’,op);
fputc(‘E’,op);
fputc(‘S’,op);
fputc(‘C’,op);
} else fputc(cc,op);
}
}
```

FIGURE 4 Conversion Program

CHAPTER 5

Configuration File Processing

This section describes how to construct or edit *CAPBAK/UNIX* configuration files.

5.1 Introduction

All the commands in the *CAPBAK/UNIX* system can read run-time parameters from a configuration file instead of the command-line. The default configuration filename is **`capbak.rc`**. All six commands will look for a configuration file of this name in the current working directory unless a different file is specified using the **`-r`** flag.

Using a configuration file allows the user to set various run-time parameters automatically. However, any command-line parameters which are present will override configuration file settings.

The *CAPBAK/UNIX* configuration file is a simple ASCII text file that consists of single-line entries for each parameter set. It can be created, altered, or named using the **`save`** option of interactive **`capbak`**, or alternatively, using any of the UNIX editing utilities.

5.2 Parameter Definitions

The following run-time parameters can be set in the configuration file:

`baseline` =*"file"* Save record mode responses in *file*. (default = **`capbak.bsl`**)

`baud` = *n* Set baud rate of device driver you are calling out on to *n*.
(default = **`9600`**)

`crdelay` = *n* *n* must be an integer. For positive values delay *n* seconds after each carriage return. For negative values delay *n* seconds before each carriage return.

`delay` = *factor* Use *factor* as the playback delay factor. Negative values are not allowed: see Section 3.1.1 for a full explanation of playback delay (default = **`1`**)

echo = *n* — Echo responses in playback mode(0=off, 1=on, default=**off**).

interrupt = ***c****har* — The character or pair of characters to respect as one that interrupts the current session.

keyboard char — The keyboard must be selected as either "break" or "raw". The default is "raw".

keysave =*"file"* — Use *file* as the keysave file.

marker = *n* — Set marker mode on or off during playback(0=off, 1=on, default=**off**).

port = *name* — Use port *name* to communicate with the remote machine.
(default = **/dev/tty0**)

response =*"file"* — Save playback mode responses in *file*. (default = **capbak.rsp**)

termination = *char*
The character or pair of characters that is used to cause "end of recording session."

5.2.1 Termination Character Details

The termination character definition line gives the user the capability to define the character that will be used to cause "end of recording session." I The definition can be for either a single or a double keystroke sequence.

The interrupt character definition line gives the user the capability of defining the character that will be interpreted as a session interrupt. When an interrupt is sensed, the system gives the user the chance to enter a "**marker mode**" message. Such messages can be used to enhance user-supplied flow-control methods.

When you specify a termination sequence or an interrupt sequence in the configuration file, if you want "Control" letter, you type in a circumflex ('^') followed by the letter, i.e. "^C" for "control-C".

When you are specifying the sequence interactively from within *CAPBAK* you can type, literally, the control-key with the letter or you can type 'circumflex' letter.

If you want the circumflex to be one of the letters in the termination or interrupt sequence, you have to escape it with a "\". Also, if you want a "\" as part of the sequence you have to escape it with a "\\".

5.2.2 Keyboard Mode

The keyboard mode selector must be either "break" or "raw". If **`keyboard = break`** is used, then the recorded port will be put in so-called *Cbreak mode*.

If **`keyboard = raw`** is used, then the recorded port will be placed in pure or "raw" mode. This is desirable when the testing applications put the port into an unusual configuration during start-up. No extra processing of characters is done by the UNIX operating system if used in raw mode.

5.3 Example CAPBAK/UNIX Configuration File

Below is an example of a typical *CAPBAK/UNIX* configuration file.

```
port = /dev/ttya
baud = 2400
delay = 0.5
crdelay = 1
echo = 1
keysave = "my.ksv"
baseline = "my.bsl"
response = "my.rsp"
```

Here is a slightly different configuration file. In this case, this is the order in which it is displayed from within *CAPBAK/UNIX*, and also the order in which it is written to a **`*.rc`** file under user control.

```
port   = /dev/tty7b
delay  = 1.000000
baud   = 9600
echo   = 0
crdelay= 0
termination= "^D"
interrupt= "^C"
keyboard = break
keysave  = "example.ksv"
response = "example.rsp"
baseline= "example.bsl"
ascii   = off
query = off
```

CHAPTER 6

Conditional Playback Programming

This section describes how a user can create "conditional keysave files", which behave during playback according to the results of various system calls."Play back programming" lets *CAPBAK/UNIX* users play keysave files as scripts whose behavior is modified by systemic and environmental factors.

6.1 Programming In Data and Command Modes

Conditional execution of keysave files is accomplished by designing its parser to recognize two modes of keysave file execution. Users familiar with the old keysave file format will be familiar with the **Data Mode** of operation. In this mode, text is interpreted as saved keystrokes to be played back along with timing information which is enclosed in braces '[ ]'. *CAPBAK/UNIX 3.2* and later versions also recognize a **Command Mode** of execution of keysave files. In this mode, the user can program the keysave file in order to do conditional execution based on system calls.

Wedge symbols < ... > are used to toggle back and forth between **Data Mode** and **Command Mode**. Execution of keysave files *always* begins in **Data Mode**. The "<" puts you into **Command Mode** until the next ">" takes you back into **Data Mode**. See below for examples of this.

6.1.1 Data Mode

Here in outline form are the valid syntax structures for Data Mode:

`>` Enter **Data Mode**.

This command causes *CAPBAK /UNIX* to enter Data Mode if you are in Command Mode.

`abc...` Plain text is something to be played back.

In general, unless covered by one of the exceptions shown below, characters that you see in the executable keysave file are played back to the application.

Characters can have time delays, indicated by [ ]'s, as described below.

`[a]` Playback timing information (a positive integer).

Remember, [ ]'s are optional; if no [ ]'s appear before a character then the delay value is taken as zero. Hence, if a sequence of characters appears without interspersed [ ]'s then the characters are played back as fast as possible.

Delay values are multiplied (or divided) by the `-d number` switch, which permits continuously variable time delays.

`{ }` Special Characters & Control Brackets.

{ }'s are used for special characters. There exist special character representations for {SPACE}, {TAB}, {FF}, and {LF}, as well as for various other characters, for playback. White space is ignored everywhere (but preserved for the user's convenience).

(Note: {}'s are also used to bracket control structures when in Command Mode; see below for syntax.)

6.1.2 Command Mode

The valid syntax structures for **Command Mode** are:

`<` — Enter **Command Mode**. This command causes *CAPBAK/UNIX* to enter Command Mode.

`( )` — System call text. System calls given in parentheses are executed by the UNIX Shell. The user can then test the exit code with `if` and `while` statements.

An instance of ( )'s not preceded by `if` or `while` is taken as a system call. The text is executed directly.

The logic convention used is that for "system exit codes" (see below).

`if ( )( )` — `if` statement.

The `if` statement will execute the system call. If the shell command executes without error and returns a *true* value, the following action (enclosed in brackets '{}') will be executed.

The logic convention used is that for "system exit codes" (see below).

`else ( )( )` — `else` statement.

The `else` statement may follow an '`if`' and action block. Its action (enclosed in brackets '{}') is executed only when the '`if`'s' action block is not executed.

The logic convention used is that for "system exit codes" (see below).

`while ( ) ( )` — `while` command.

The `while` statement will execute the enclosed system call. If the shell command executes without error, the following action (enclosed in brackets '{}') will be executed and the `while` statement's system call will be tried again.

The logic convention used is that for "system exit codes" (see below).

`update` — `update` command.

This command will flush the buffers of the response file.

Within an `if ( )` or `while( )` construction you can use characters freely, except for "(", ")", "{", and "}". They are used to delimit the system call and to indicate the bracketing of text that is the target of the `if` or `while`. A balancing "}" indicates the extent (scope) of the `if` or `while`.

Note that ">" and "<" are used *only* to switch between command mode and data mode.

A system call inside an `if` or `while` or within ( )'s while in command mode is always delimited by the first balancing ")" that matches the opening "(". This means that if you wish to have a system call with a "(" or a ")" you must use the escape character \ as follows: "\(" or "\)". Otherwise, the scan will end in the wrong place and the wrong system call will be issued.

If you wish to include a "\" in the system call, you must use the escaped version "\\".

Programming in *CAPBAK/UNIX* is always done in the ASCII image form of the keysave file.

6.2 Programmable Keysave File Syntax

Here is a quick summary of the syntax that a *CAPBAK/UNIX* user can type into the ASCII version of the keysave file.

Note that the syntax is easy to remember and is intentionally similar to that used in "C" language programming.

Reversed Characters

```
< > [ ] { } ( )

White space is ignored.

You may use the special sequence {LF} for
newline, {SPACE} for space (blank), etc.
```

Special Character Treatment

```
\< for <
\> for >

\( for (
\) for )

\{ for {
\} for }

\[ for [
\] for ]

\\ for \ (the usual convention for escap-
ing
the escape character).
```

Comments and Directives

```
</* Any comment. */>

<#include filename>
```

You must be in **Command Mode** to include a filename. Because you can issue a # include only from Data Mode, the included file will also be assumed to be in Command Mode.

Therefore, if you want the included file to be in Command Mode, you must make sure it begins with a "<".

Execution and Control Statements

```
<(...) [ { } ] >;
<update>
<if (...) { } [ else { } ]>;
<while (...) { } >;
<while (...) { } >;
```

Here [...] means 0 or more instances of the item enclosed. Also "..." implies that a complete block of text, including possibly other `if`s and `while`s, can be nested inside another statement.

6.3 Programmable Keysave File Command Summary

The keysave file is interpreted dynamically. There are two modes within such a keysave file: **Data Mode** and **Command Mode**.

6.3.1 General Description

Data Mode information is assumed to be keystroke information. **Command Mode** information is sent to the command interpreter. If you get **Command Mode** data into the wrong mode it will be played back as if it were keystroke information.

To illustrate this, here is a short passage which shows what is in **Command Mode** and what is in **Data Mode**. Different modes are shown at different indentation levels.

```
data {SPACE} data {SPACE} data
    <
    //* Comment *//
        while (system call)
        {
            //* Comment *//
            (command)
            >
                data {SPACE} data
            <
            (command)
            //* Comment *//
        }
    >
```

6.3.2 Detailed Description

Here is a detailed description of the behavior of the action statements available within a conditional keysave file:

`(any text)` This is a system command.

`any text` is executed by the system and the system return value is ignored. The text can be any string (up to 1024 characters) that can be interpreted by the shell.

`update` This command tells *CAPBAK/UNIX* to flush the buffers in its response files. This assures full synchronization in scripts that rely on analyzing the content of the playback response file.

`if ( text ) { true-action }`

`if ( text ) { true-action } else { false-action }`

Conditional Expression Statements. A system call is issued for `text` and the return value is used to decide between the alternative actions.

If the return code is "true" then the `{true action}` is taken. If the return code is "false" then the `{false action}` is taken. If there is no `{false action}`, then a "false" return code causes the `{true action}` to be skipped over.

Note: The logic convention used follows that employed in most system call contexts: "true" is when the system call returns a zero value, meaning that the system call was successful, and "false" is when it returns a non-zero value, meaning that the system call was unsuccessful.

`while ( text ) { loop-body }`

`While` *Statement.*

This structure causes a system call to be issued with *text*. If the return value is "true" then the *loop-body* is executed. After completion of the *loop-body* the system call *text* is issued again.

The iteration continues until the response code for the system call to *text* is "false". At that time control continues with the next activity outside of the {}'s.

If you start a `while ( ) { }` loop, and then `#include` a file which is supposed to have the closing brace of the `while ( ) { }` loop, the *CAPBAK/UNIX* system will fail. If a #include file has an entire `while ( ) { }` loop sequence, beginning with the `while` token in it, that's OK.

In other words, make sure that the balancing "}" character for a `while ( ) { }` construction is not split across file boundaries.

6.4 System Call Logic Convention

Only valid system calls are permitted as the text included in ( )'s. The validity of the system call is not checked by *CAPBAK/UNIX*.

The logic evaluations done will be based on the value that the system call returns to *CAPBAK/UNIX*. This logic permits Shell Scripts and / or Batch Files to be used. Or, the user can write a special purpose program which returns values according to the standard conventions for TRUE and FALSE. The logic convention used within *CAPBAK/UNIX* is that of **exit code status**:

```
Returned value 0:TRUE = Normal return code

Returned value non-zero:FALSE = Error
return code
```

Here is a direct quote from the Kernighan and Ritchie "C" Language text:

> "This is the convention used in the **UNIX** environment. Every command returns an **exit status** -- a value returned to the shell to indicate what happened. The exit status is a small integer; by convention, 0 means "true" (the command ran successfully) and non-zero means "false" (the command ran unsuccessfully). Note that this is opposite to the values of true and false in C."

A convenient way to check a system call's return using the Bourne shell on a Unix system is to experiment with it as follows:

```
system-call; echo $?
```

The $? contains the value of the return code of the previously-issued system command.

6.5 ask.user.c Explained

The following "C" program is the source for the **ask.user** command shown in conditional playback examples given on the following page:

```
/* Special 'ask.user' "C" program for use with conditional
execution in SMARTS, CAPBAK, and other SR tools.  Note that this
"C" program uses the SYSTEM ERROR CODE CONVENTION for
logic.

This is done to match what is used by SR's products that
evaluate system calls. */
#include <stdio.h>
main(argc,argv)
int argc;
char **argv;
{ char x;
      for (;;)
      {

      printf("ASK.USER: Exit with True or False?  (Enter T or F): ");
                  x = getchar();
                  if (x == 'f') exit(1);
                  if (x == 'F') exit(1);
                  if (x == 'n') exit(1);
                  if (x == 'N') exit(1);

                  if (x == 't') exit(0);
                  if (x == 'T') exit(0);
                  if (x == 'y') exit(0);
                  if (x == 'Y') exit(0);
                  /* printf ("OK, accepting '%c'.\n", x); */
      }
}
/* End of program. */
```

6.6 Conditional Playback Examples

The next sections give some examples of ''playback programming."

6.6.1 Playback Program Example 1

Here is the ASCII image of an example keysave file as it would be edited by (and seen by) the user. This example tries to show as much of the syntax and its variations as possible.

```
</*This is a comment in the ASCII keysave file, which is ignored.*/>


[10]W [20]e[1]{SPACE}[11]ar[5]e[140]{SPACE}
including {SPACE} a {SPACE} filename,{LF}[100]


</* We can include a filename at any point, and to any depth. */>
Like this[100]{LF}
<#include filename>


<if (system-call 1) {
        >
        [21]r
        [6]u[8]n
        [7]n[6]i[4]n[2]g
        [9]{SPACE}{LF}
        <if (system-call 2) {
                /* This is a simple system call with no
                action taken. */
        }
else {
        #include another.file
        /* Some other file goes into the text here. */
        >
        [6]u[8]n[7]n
        [6]i
        [4]n[2]g
        <
        }
/* The passage below plays back a fixed sequence until something
happens.  Note that comments can be anywhere inside \< \>'s, like
this one is.  The \'s are used here to prevent interpretation. */
while (system-call continues to return TRUE) {
        >
        [9]We {SPACE}are[10]{SPACE} inside {SPACE}the {SPACE} loop!
        {LF}[100]
        <
        }
>This {SPACE}is {SPACE}CAPBAK[20]/[10]U[8]N[4]
</* Any instantiated comment can go here. */>
I[5]X[53].  [19]{LF} {LF}


</* This is the end of the script */>
```

6.6.2 Playback Program Example 2

The playback program in the passage below simulates a session login and logout. The session assumes that outputs are being captured by *CAPBAK/UNIX* into the response file **RESPONSE**.

```
/* CAPBAK Demonstration Session Script.

This demonstration script illustrates basic features of the CAPBAK
system that are supported in CAPBAK/UNIX Release 3.1 and higher.

(c) Copyright 1989 by Software Research, Inc.  All Rights Reserved.  */

/* This is a keysave file that logs you in as a 'guest'. */
>
[25]{LF}
[25]{LF}
<
/* The grep of the RESPONSE file has to wait until the system
you are logging in to issues "login". */

(echo TEST >> RESPONSE)
while (notfound.grep "ogin" RESPONSE)
        {
                update
                > [15]{LF} <
                update
        }

(sleep 1
echo DEMO:  Found login in the RESPONSE file.
cp RESPONSE RESPONSE.save)
>
[10]g[10]u[10]e[10]s[10]t[50]{LF}[50]
<
while (diff RESPONSE RESPONSE.save > /dev/null)
        {
        update
        > [50] <
        update
        }
(sleep 1;
echo DEMO:  Found password in the RESPONSE file.
rm RESPONSE.save)
>
[50]g[10]u[10]e[10]s[10]t[10]{LF}[200]
[100]{LF}
<
/* Now we should be logged in. */
```

6.6.3 Playback Program Example 3

This example illustrates another style of constructing playback programs.

```
</* This is an example keysave file to demonstrate a legible style.
  * Although long sequences of command mode information can be created,

 * the main purpose of a keysave file is to store play back data.  The  * control
statements are generally short sequences, such as:


                    while(call){  ...play back data would go

                    here ...}
                    Stylistically, this form could be closely

                    preserved bythe following:
                                        <while><(call)><{> ...play back
data...<}>


          The user could think of <while>,<(call)>,<{>... as separate keywords
rather than while, (call) etc as  keywords under  command mode.
*/>
this{SPACE}is{SPACE}data{LF}
<if><(ask.user)><{>[5]t[5]r[5]u[5]e{LF}<}>
<else><{>[5]f[5]a[5]l[5]s[5]e{LF}<}>
<while><(ask.user)><{>    </* Keep system mode to a minimum,enclose in carets
                                only the shortest allowable sequences of control
                                for readability
                        */>


          this{SPACE}is{SPACE}the{SPACE}outer{SPACE}loop.{LF}
          <while><(ask.user)><{>


                    <(echo this is the inner loop)>


                    [10]data[20]{SPACE}more{SPACE}data{LF}


          <}>       </* Close inner loop */>
<}>                 </* Close outer loop */>
[10]end{SPACE}of{SPACE}data{LF}
```

CHAPTER 7

Using Conditional Playback

Playback programming involves some special conventions, described in this section.

7.1 Invoking CAPBAK/UNIX For Conditional Playback

In the **interactive mode** of operation the MODES menu is used to select conditional execution. In this case the keysave file currently known to CAPBAK/UNIX is used; this file is assumed to be in ASCII format.

For **command-line invocation**, you can use the command:

```
keypla -a -k <keysave> other-switches..
```

or

```
plabak -a -k <keysave> other-switches..
```

to obtain conditional interpretation of the **<keysave>** file.

Note that in either case the user is still required to convert a recorded non-conditional keysave file from internal to ASCII format prior to playback. The internal format of non-conditional keysave files is generally much more compact than their ASCII equivalent.

7.2 Limitations on '#include' Use

The **#include** feature has no limitations on the number of files that can be included within one another.

Difficulties will arise if: (a) a **#include** file names itself as the target of the **#include** (recursive include), and (b) if a **#include** is used in a way so that balancing "{ }"'s are not present in the same file that is used to contain the main body of a **while** loop.

7.3 Complete List of Special Characters

Here is a list of all of the special characters that *CAPBAK/UNIX* can process:

{NULL_CN}	An ASCII 0, does not get transmitted.
{ESC}	The escape character, equivalent to {CTRL [}. {CTRL A},The full set of control characters. {CTRL B},...({CTRL M} is the carriage return character '\ r'.)
{BELL}	This is equivalent to {CTRL G}.
{LF}	Line feed character. This is '\ n' and is equivalent to {CTRL J}
{FORM FEED}	Form feed character.
{SPACE}	Blank character.

7.4 Interactive 'query' Mode

To assist in analyzing a conditional keysave script *CAPBAK/UNIX* includes a switch that causes the system to read and interpret a conditional keysave file interactively.

This facility is used to "query" a keysave file that has conditional operations. This mode will also be useful in checking the syntax of a script to make sure that there are no errors.

7.5 Command Line Invocation

For interactive operation *CAPBAK/UNIX* must be invoked with a command-line call to **keycvt** using the -q switch, as follows:

```
keypla -q -k <ascii-keysave-file>
```

or

```
plabak -q -k <ascii-keysave-file>
```

where **<ascii-keysave-file>** is the candidate ASCII-format conditional keysave file.

You may also want to accelerate the analysis by using a delay factor less than 1.0; e.g. try the switch **-d 0.1**.

7.5.1 Interactive Invocation

You can also enter interpretive mode with the **query** option from the ACTIONS menu in the interactive version of **CAPBAK / UNIX**.

7.5.2 Query Mode Description

In query mode *CAPBAK/UNIX* will read the specified file and send characters to standard output — presumed to be your terminal — in a way that is similar to operation of the **keypla** or **plabak** command. Characters are emitted at the rates implied by [ ] values. Adjustments for the specified value of the delay factor are made.

```
When a conditional operation is encountered, CAPBAK/
UNIX will issue a request to the user of the format
shown here:
  ...
  ...
CAPBAK: if condition system call ENCOUNTERED, text:
"system-call text"
Should query return True or False? (t or f)
  ...
<ordinary keystrokes emitted>
  ...
  ...
```

The system will wait for the user to select **T** (or t) for True, or **F** (or f) for False.

If the conditional script contains a **while (...)** construction then this loop will continue until it is broken by a "False" response.

If the **-l logfile** switch is used then *CAPBAK/UNIX* will create a pure non-conditional keysave file that reflects the actual sequences used. This file can be used in a subsequent **non-conditional** playback session.

For example, either of the commands:

```
keypla  -q -k keysave-file  -l new-file

plabak  -q -k keysave-file  -l new-file
```

would place the outcome of the user's interactive selections in the file **new-file**. This file is suitable for use as a normal keysave file, so that a regular non-query mode call to **keypla** or **plabak** could be made.

It is important to emphasize that **new-file** that is generated in this way would **NOT** be a conditional keysave file because all of the logical selections would have been made by the user.

7.5.3 Summary of CAPBAK/UNIX Control Inputs

CAPBAK/UNIX can be controlled from the command line or from the **`*.rc`** file. Command line switches for *CAPBAK/UNIX* take precedence over **`*.rc`** file instructions.

7.5.4 Command Line Switches

Command line switches available within the *CAPBAK/UNIX* system, used *specifically* for conditional and interpretive mode, are shown below. These switches apply to all *CAPBAK/UNIX* commands: **`capbak, capkey, keypla, record,`** and **`plabak.`** Refer to another chapter for expanded lists specific to individual CAPBAK / UNIX commands (See CHAPTER 2 - "CAPBAK / UNIX Invocation and Use" on page 11.).

`-a`	Turns on the conditional / interpretive mode. The named keysave file is read and used interpretively. System calls are made and their response codes are used to determine which keystrokes are emitted. Logic with **`if`** and **`while`** and file inclusion with **`#include`** constructs are available.
`-A`	Turns off the conditional / interpretive mode described above.
`-f` *n*	Floor time delay *n*, an integer. This is the MINIMUM time delay between characters issued. Caution: The value used must not be larger than the ceiling value.
`-F`	Turns off the floor time delay described above.
`-g` *n*	Ceiling time delay *n*, an integer. This is the MAXIMUM time delay between characters issued. Any delay longer than the ceiling value will be 'clipped' to that value. Caution: The value used must not be smaller than the floor value.
`-G`	Turns off the ceiling time delay described above.
`-l` *file*	The name of the keystroke logfile. If this switch is present then the actual keystrokes issued (including actually issued delay values) will be placed in the named logfile.

The **-l** switch works with the **-a** or **-q** switches.

-L Turns off logging to the logfile described above.

-q Query / checkout flag. Uses the interpretive mode as with the **-a** option except that *CAPBAK* asks the user to provide the logical values when a conditional system call is made. The actual system call text is provided, but the system call is NOT made.

-Q Turns off the query / checkout mode described above.

7.5.5 Play Back Programming Style Note

The conditional execution feature involves, in effect, overlaying two distinct programming languages into one file: one which sorts our **if**s and **while**s, and the other which causes keystrokes to be emitted. Overlaying these two languages is difficult, and hard to do cleanly.

The "solution" chosen in this product is to have a dual-purpose input language which meets both goals, but at the expense of the user having to keep track of which mode any particular line is in: i.e. either command mode or text mode. The *CAPBAK/UNIX* system toggles back and forth between the two modes with the < >'s. When writing scripts you will have needless difficulty unless you take care to:

1. Use lots of white space to assist the eye in keeping track.
2. Avoid the use of superfluous {}'s and ()'s; use only the ones NEEDED.
3. Use **#include** liberally to invoke sub-files, but only in the proper sense:

```
<#include "filename">
```

and not:

```
#include "filename" some other command>
```

CHAPTER 8

Error Handling

CAPBAK/UNIX has provisions for certain kinds of error processing.

8.1 File Handling Errors

Here are file handling error messages that are issued by the *CAPBAK/UNIX* system:

```
capbak: Can't open fname for writing.
capbak: Can't open fname for reading.
capbak: Unable to open fname file.
capbak: Unable to open device.
```

These errors are caused by the inability of the system to create or open the file *fname*. This may be due to an invalid file name, or restricted access permissions. If the file is a device it may be due to another process tying up the port.

8.2 Command Line Errors

Here are command-line error messages issued by the *CAPBAK/UNIX* system:

```
capbak: Ignoring opt option without file name.
capbak: Ignoring opt option without baud rate.
capbak: Ignoring opt option without delay factor.
capbak: Ignoring opt option without port number.
```

Many options require that valid parameters be specified; for example:

```
-k keysave.ksv
-B 1200
```

can be used by **plabak** only if `keysave.ksv` exists, or if 1200 is an available baud rate.

CHAPTER 9

keycvt

9.1 Introduction

9.1.1 Background

keycvt (pronounced "key convert") is an ancillary tool to **Software Research's** *CAPBAK/UNIX*, a keystroke capture/playback tool used primarily for software testing. **keycvt** enables keystrokes saved by *CAPBAK* to be converted to an editable format suitable for alteration.

Edited keystrokes can then be converted back to a format suitable for reuse with *CAPBAK/UNIX*.

The ability to repeatedly perform keystroke conversions in an interactive way greatly simplifies the maintenance of keystroke files and hence the testing effort. **keycvt supports both the MS-DOS and UNIX versions of CAPBAK.**

9.1.2 Function

keycvt is used to perform any combination of the following tasks:

1. Translate the *CAPBAK/UNIX* keysave files to ASCII "visible" format.
2. Translate ASCII files to keysave files.
3. Add, alter, or remove timing delays between keystrokes.
4. Adapt the keysave files to new software revisions.When the product changes, old scripts can be updated with a minimum of effort by changing keystrokes and timing, thereby allowing existing test suites to be reused.
5. Speed up test execution. **keycvt** can strip out the timingdelays of a test session so that *CAPBAK* can replay it as fast as the application will permit.
6. Create application test scripts directly from ASCII files.Test scripts can be created with any ASCII editor and converted to *CAPBAK*-executable format with **keycvt**. This is helpful for creating application test scripts quickly.

9.1.3 Requirements

keycvt is designed to be used in conjunction with *CAPBAK/UNIX*. Note that for the **keycvt** on-line help frames to be interactively available, they must be installed in **/usr/lib/SR/capbak.hlp** .

9.1.4 Use of This Chapter

9.1.4.1 Explains how to run and use keycvt's various features.

9.1.4.2 Explains how keycvt processes its various files.

9.1.4.3 Explains how keycvt processes its configuration file.

9.1.4.4 Describes keycvt error messages and trouble-shooting procedures.

9.1.4.5 Lists special symbols used in ASCII versions of keysave files.

9.1.4.6 Includes keycvt on-line help frames.

9.2 Invocation

9.2.1 Command Line Invocation

keycvt may be invoked using command-line arguments.This makes it especially useful in batch processing. The following syntax applies to conversion from keysave to ASCII formatand vice versa.

Syntax

keycvt -k *ksvfile* ***-a*** *ascfile* ***-K|-A [-S*** *type factor**]***

Options

-k *ksvfile*	Use *ksvfile* for the keysave file.
-a *ascfile*	Use *ascfile* as the ASCII file.
-A	Create an ASCII file from a keysave file.
-K	Create a keysave file from a ASCII file.
-S	Set timing features according to *type* and *factor*
type	= ‘‘0" if Time is to be removed
	= ‘‘1" for Time Clipping
	= ‘‘2" for Time Expansion
	= ‘‘3" for Time Constant
	= ‘‘4" for Time Flooring
	= ‘‘5" for Time Compression
	= ‘‘6" for Selective Time Removal
factor	= magnitude of time modification(1...9)

Note that the keysave file may be either an input or an output file,depending on whether you are converting from ASCII to keysave (**-K**) or keysave to ASCII (**-A**).

Also note that the factor for Selective Time Removal (type 6) is a string of special characters separated by spaces such as:

```
keycvt  -k ksvfile  -a ascfile  -K -S6 @
{SPACE} {LF}
```

A second way to invoke **keycvt** is to put commands into a configuration fileand simply specify this file on the command line in the following manner:

```
keycvt  -r  rcfile
```

where **rcfile** is the configuration file. Chapter 4 explains how to set up configuration files for use with **keycvt**.

9.2.2 Menu Invocation

A third way to use **keycvt** is to invoke it interactively. **keycvt** is menu-driven. To start **keycvt** interactively, just enter:

```
keycvt
```

At any time the user can display the current settings using the **settings** command, get online help with the helpfP command, and exit the current menu using **exit**. In addition the user may execute a system command using the escape character **"!"**.

For example:

```
MAIN: !sort myfile
```

If a configuration file is read in, the settings in the file are automatically used. These can be changed, and the new values used and saved if required.

An example of the settings display is:

```
OPTIONS:

FILES:
            Config file     = keycvt.rc
            Keysave file    = t.ksv
            ASCII  file     = t.asc
MODES
                 Key2asc    = ON
```

The menu commands may be abbreviated when no ambiguity exists, e.g. **options** can be shortened to **o** because no other command in the MAIN menu starts with that letter.

Whenever you press a RETURN at the menu prompt, the current menu options will be displayed. If you press a RETURN, the MAIN menu will appear and prompt the user to:

`go`	Perform conversion.
`mode`	Select the mode of operation.
`options`	Select configuration parameters for **`keycvt`**.
`settings`	Display the configuration settings.
`help <opt>`	Display HELP text for a command.
`exit`	Exit to the system.

If the **`mode`** option is selected and you press a carriage return the MODES menu will appear and prompt the user to:

`key_to_ascii`	Convert a keysave file to an ASCII file.
`ascii_to_key`	Convert an ASCII file to a keysave file.
`dos_to_unix`	Convert a DOS file to a UNIX file.
`unix_to_dos`	Convert a UNIX file to a DOS file.
`settings`	Display the configuration settings.
`help <opt>`	Display HELP text for command.
`exit`	Exit current level.

If **`options`** is selected from the MAIN menu and you press a carriage return the **`keycvt`** OPTIONS menu will appear and prompt the user to:

`config <file>`	File to read for configuration parameter
`keysave <file>`	File from which to read the keystrokes.
`ascii <file>`	File in which to save the converted keystrokes.
`timing`	Modify the timing.
`settings`	Display the configuration settings.
`help <opt>`	Display HELP text for command.
`exit`	Exit current level.

If any of the first three options is selected, then in the absence of a file-name on the command line, the user will be prompted for the required parameter.

If the timing option is selected from the OPTIONS menu and you press a carriage return, the TIMING menu will appear and prompt the user to:

`remove`	Remove or ignore the timing.
`clip <1..9>`	Set the maximum time delay.
`floor <1..9>`	Set the minimum time delay.
`expand <1..9>`	Expand the time delay.
`compress <1..9>`	Compress the time delay.
`sremove`	Remove timing except after characters.
`constant <1..9>`	Set a constant time delay.
`settings`	Display the configuration settings.
`help <opt>`	Display HELP text for command.
`exit`	Exit current level.

9.3 File Processing

keycvt processes two types of files, **keysave** files created by *CAPBAK* and ASCII files created either by **keycvt** or directly by the user using a suitable editor.

There are some restrictions on the keys that can be translated and the file-names that can be used; these are described below.

Principally, **keycvt** will convert from one form to the other, and in doing so modify, create, or remove the timing delays that *CAPBAK* uses when replaying the keystrokes.

These time delays provide the faithful time recording feature, but are not necessary and may be removed or selectively removed.

9.3.1 ASCII File Format

The ASCII file equivalent of a keysave file may contain a number of special symbols enclosed in braces. This mechanism allows non-printing and unusual characters to be represented in an editable format.

For example, {BACKSPACE} is used to represent a backspace, while {CR} is used to represent a carriage return. A complete list of acceptable special symbols is contained in 10.6.

The ASCII file format is intended to be a machine- independent format for describing the time delays and keystrokes used to recreate a test session. Standard Editor files may also be processed and converted to the ASCII file format described next.

Overall Structure

The visible keysave file is a sequence of character groups in the following format:

```
character character ...
character [time] character [time] ...
{special} [time] {special} [time] ...
{special} [time] character [time] ...
{{message}} ...
```

Note that the *[time]* field is optional ([0] is assumed). Blanks are significant. For example, five blanks with zero delays would be:

```
[0] [0] [0] [0] [0] [0]
```

Character Processing

Ordinary characters: The standard ASCII character set passes through with no special treatment.

\[equals ‘‘[" as an actual character

\{ equals ‘‘{" as an actual character

\ \ equals ‘‘\"

any_other_character is ignored when converted to ‘‘invisible" format. For example: <newline> for formatting.

[*time*] [0] means ‘‘zero delay".

[n] n < 32767, expressed in standard delay units.

{*special*}

{F*n*}	Function Key Fn
{SPACE}	Space Bar
{CR}	Carriage Return
{LF}	Line Feed
{ESC}	Escape
{KEYPAD n}	Keypad n
{CTRL DEL}	Control-Delete
{INTR}	Interrupt Key
{SHIFT CTRL DEL}	Shift-Control-Delete
{ALT CTRL SHIFT M}	Alt-Control-Shift-M

A complete list of **keycvt** special symbols can be found in 10.6.

There can be up to four keys held down simultaneously in a {special} group (although this may not be legal).

{MOUSE,*n1,n2,n3,n4*}

This denotes a mouse movement.The four integers signify the following:

*n*1 = Button status for mouse function 5 and 6or undefined for function 3 and 11.

*n*2 = Button status for mouse function 3, count of button presses for mouse function 5, count of button releases for mouse function 6, or undefined for function 11.

*n*3 = Column coordinate for function 3, 5, and 6,horizontal counter for function 11.

*n*4 = Row coordinate for function 3, 5, and 6,vertical counter for function 11

{{message}} This is a free-text message that can be included in the keysave file visible format,which is included (but not used) in the ‘‘invisible" format.

9.3.2 Converting Keysave Files

Captured keystrokes are converted to ASCII format using the `key_to_asciii` mode. This option is selected or de-selected with a toggle switch from the **MODES** menu. The names of the keysave and the ASCII files to be created are selected from the options menu.

9.3.3 Creating Keysave Files

ASCII files which may include timing delays can be used to create keysave files using the `ascii_to_key mode`. The name of the ASCII file and the keysave file to be created are selected from the OPTIONS menu.

9.3.4 Playback Timing

CAPBAK/UNIX provides a playback delay feature which allows you to slow down keystroke playback, which normally is extremely fast. This feature is useful for tailoring playback sessions to particular application programs.

Playback timing can be altered in a number of other ways by using **`keycvt`** to modify keysave files. The following options are available for modifying playback timing:

```
Time Removal
Time Removal (Selective)
Time Constant
Time Clipping
Time Flooring
Time Expansion
Time Compression
```

The normal playback mode will preserve keystroke timing precisely, assuming that the recording is played on the same CPU class as the one on which it was captured. A faster CPU may result in faster playback times, but in proportion to the original recorded rates.

The faithful time recording feature records the number of 16h interrupts that have happened between keystrokes. The number is transmitted to the keysave file when the next keystroke occurs, or when the 16-bit register overflows (after approximately 30 seconds has lapsed since the last keystroke was typed, and if there has been no other keystroke). In the latter case, a ‘‘null'' symbol (and not the ‘‘bell'' symbol implemented in 2.0.7) is saved in the keysave file and the register is reset.

- **Time Removal**--Time Removal allows the user to remove all timing information.
- **Selective Time Removal**--Selective Time Removal allows the user to remove all timing information except AFTER the specified characters which can be special characters such as {SPACE} or single characters such as **@** or **e**.
- **Time Constant**--To select constant time, the user chooses a constant playback frequency from 1 to 9. The constant frequency equals 2^{n+1}16h interrupts between each keystroke, where n is a user-selected number from 1 to 9.
- **Time Clipping**--Time Clipping allows the user to select a maximum playback frequency from 1 to 9. A limit is then set on the maximum amount of time possible between each keystroke. Any frequency above this maximum will be clipped to the maximum frequency. Any frequency below it will be the real frequency as recorded during capture.
- **Time Flooring**--The Time Flooring functions analogously to the Time Clipping feature. It allows the user to choose a minimum playback frequency from 1 to 9. With this feature, the limit is set on the minimum amount of time possible between each keystroke. Any frequency below this minimum will be set to the minimum frequency. Any frequency above it will be the real frequency as recorded during capture.
- **Time Expansion**- The Time Expansion feature allows the user to choose an expansion factor n from 1 to 9. The resulting timing will be the actual time recorded during capture times 2^{n-1}. E.g., choosing a factor of 1 leaves the timing the same.
- **Time Compression**--Time Compression is the inverse of Time Expansion. It allows the user to choose a compression factor n from 1 to 9. The resulting timing will be the actual time recorded during capture divided by 2^n. Any frequency below 2^n will be the real frequency as recorded during capture.

A keysave file without timing can have timing format added to it. Use **keycvt** to convert it to an ASCII file with timing added, and then transform the ASCII file back to keysave format. In this case, the new keysave

file will have constant time counters of 256. The counters can then be modified using the command-line **-s** option with action 3, or manually by means of a text editor.

9.4 keycvt CONFIGURATION FILE

keycvt can be run using a configuration file by executing the following command:

```
keycvt  -r  rcfile
```

where rcfile is a user-created text file containing commands to control **keycvt**. ALTERNATIVELY, because **keycvt** always checks for the default configuration file **keycvt**.rc, the user may create this file, and then just enter:

```
keycvt
```

The configuration file may contain the following commands where filename is the valid disk file name, and n is the conversion factor 1...9 used by **keycvt**:

- To specify a keysave file:
  keysave = "filename"
- To specify an ASCII file:
  ascii = "filename"
- To convert a keysave file to an ASCII file:
  convert = key2asc
- To convert an ASCII file to a keysave file:
  convert = asc2key
- To modify the timing in either the keysave file or its ASCII version:

```
remove = 1
sremove = <char>
clip = n
floor = n
expand = n
compress = n
constant = n
```

Note that these options are mutually exclusive. However, **sremove** may be repeated up to 128 times and the program will keep track of all characters specified.

9.5 ERROR MESSAGES

keycvt performs a considerable amount of internal error checking. In the event that an error is detected, one or more messages are written to the screen.

9.5.1 keycvt Error Messages

Message	Meaning
Error Code 200--Bad token error	An error most likely resulting from unbalanced (missing) braces in the ASCII file.
Error Code 300- String not found error	An error resulting from an invalid special symbol used in the ASCII file (for a list of special symbols see 9.6)

TABLE 1 keycvt Error Messages

9.6 keycvt Special Symbols

In the table on the following page:

BELL	represents the CTRL-G keys.
CR	represents the carriage return key.
CUR	represents one of the cursor control indicated by its direction (LF, RT, UP, or DN).
FORM FEED	represents the CTRL-L keys.
FS	represents the CTRL- keys.
F*n*	represents function key *n*.
GS	represents the CTRL-] keys.
LF	represents the CTRL-J keys.
RS	represents the CTRL-^ keys.
SHIFT	represents the shift key.
US	represents the CTRL-_ keys.

```
{ALT B}             {CTRL F1}       {ENTER}         {F10}
{ALT C}             {CTRL F2}       {ERASE EOF}     {F11}
{ALT B}             {CTRL F1}       {ENTER}         {F10)
{ALT C}             {CTRL F2}       {ERASE EOF}     {F11}
{ALT D}             {CTRL F3}       {ERINP}         {F12}
{ALT F}             {CTRL F4}       {ESC}           {F13}
{ALT F3}            {CTRL F5}       {FIELD MARK}    {F14}
{ALT F5}            {CTRL F6}       {FORM FEED}     {F15}
{ALT F6}            {CTRL F7}       {FS}            {F16}
{ALT F9}            {CTRL F8}       {GS}            {F17}
{ALT F10}           {CTRL F9}       {HOME}          {F18}
{ALT L}             {CTRL F10}      {INDENT}        {F19}
{ALT M}             {CTRL H}        {INS}           {F20}
{ALT N}             {CTRL HOME}     {KEYPAD 0}      {F21}
{ALT V}             {CTRL I}        {KEYPAD 1}      {F22}
{ALT X}             {CTRL K}        {KEYPAD 2}      {F23}
{ALT Z}             {CTRL N}        {KEYPAD 3}      {F24}
{ASCII 128}         {CTRL O}        {KEYPAD 4}      {PG DN}
      .             {CTRL P}        {KEYPAD 5}      {PG UP}
      .             {CTRL PG DN}    {KEYPAD 6}      {RESET}
{ASCII 255}         {CTRL PG UP}    {KEYPAD 7}      {RS}
{ATTN}              {CTRL PRTSC}    {KEYPAD 8}      {SHIFT F1}
{BACKSPACE}         {CTRL Q}        {KEYPAD 9}      {SHIFT F2}
{BACKTAB}           {CTRL R}        {KEYPAD +}      {SHIFT F3}
{BELL}              {CTRL S}        {KEYPAD -}      {SHIFT F4}
{CLEAR}             {CTRL T}        {KEYPAD .}      {SHIFT F5}
{CR}                {CTRL U}        {KEYPAD _}      {SHIFT F6}
{CRSEL}             {CTRL V}        {LF}            {SHIFT F7}
{CTRL 2}            {CTRL W}        {NULL_CN}       {SHIFT F8}
{CTRL A}            {CTRL X}        {PA1}           {SHIFT F9}
{CTRL B}            {CTRL Y}        {PA2}           {SHIFT F10}
{CTRL BACKSPACE}    {CTRL Z}        {F1}            {SPACE}
{CTRL C}            {CTRL [}        {F2}            {SYSREQ}
{CTRL CR}           {CUR DN}        {F3}            {TAB}
{CTRL CUR LF}       {CUR LF}        {F4}            {TEST}
{CTRL CUR RT}       {CUR RT}        {F5}            {US}
{CTRL D}            {CUR UP}        {F6}
{CTRL E}            {DEL}           {F7}
{CTRL END}          {DUP}           {F8}
{CTRL F}            {END}           {F9}
```

FIGURE 5 **keycvt** Symbols for Special Characters

9.7 keycvt On-Line Help Frames

```
Any menu                                                    HELP
>>> help

This menu system is designed to process user commands in a manner
which is easy to use, and easy to remember.
It is intended to be a quick reference and should
enable novice users to quickly learn the commands and options.
Help requires a parameter as indicated by <..> to specify for which
command help is required.

For further information please call our support desk at (415) 957-1441
```

```
> MAIN                                                      HELP
>>> mode

      The "mode" option displays the MODE menu, allowing the user to
      select which mode of conversion to perform.
      Each mode is described separately, and more information can be
      displayed using help.

      The following combinations are valid:

      FILE TYPE          O/S

asc2key/key2asc  +  DOX/UNIX
```

```
> MAIN                                                      HELP
>>> options

"options" displays the OPTIONS menu, allowing the user to
select which files to use for the conversion.
If the timing option is selected, another menu is displayed
 to prompt the user for the timing conversion details.

To select a file, enter the command and filename on the same
line.  Or, if only the command is specified, enter the
filename when prompted, e.g.:

            ascii myfile.txt

Each option is described separately, and more information can be
displayed using help.
```

> MAIN HELP
>>> go

The "go" command performs the file conversion as specified by the user either in the configuration file or by menu options.

Before entering the "go" command, check the current settings using the "settings" command.

If these settings are to be used again, then use the "save" option (use the "config" option in the OPTIONS menu to change the name), to copy them to the configuration file.

> Any Menu HELP
>>> settings

The "settings" command displays the current settings and options. When **keycvt** is invoked it looks for a configuration file ".rc", which can be used to automatically set all the options. The user may change any of these settings before using the "go" command. If these settings are to be used again, then use the "save" option to copy them to the configuration file. Use the "config" option in the OPTIONS menu to change the name of the configuration file. To use a different configuration file when invoking **keycvt** use the -r option, e.g.:

-r myfile

If in doubt about how to set up a configuration file with an editor use the save feature to see how **keycvt** does it. Always check the settings before using the "go" command.

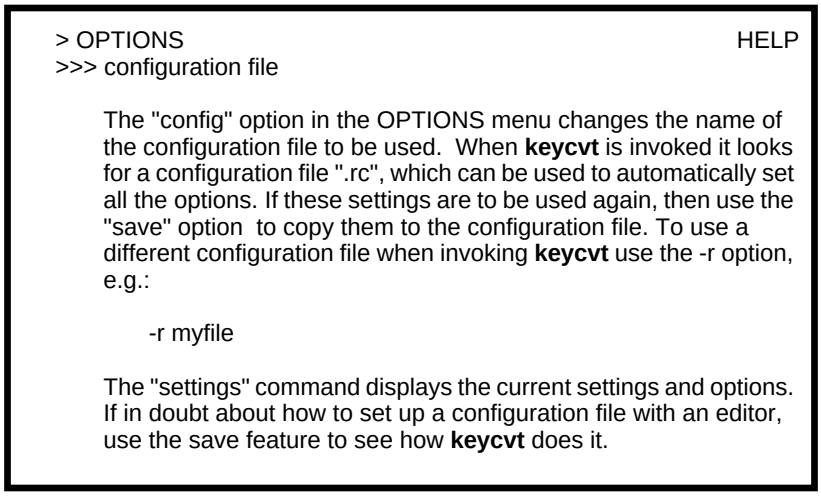
> Any menu HELP
>>> exit

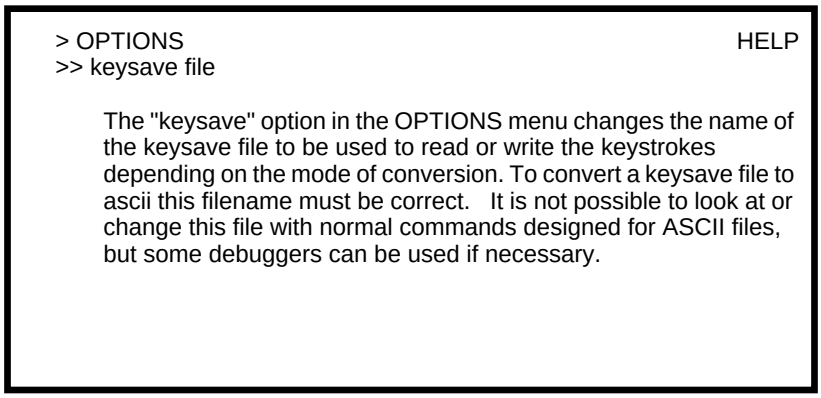
Exit will move the user to the previous menu level, and in the case of the MAIN menu back to the system.

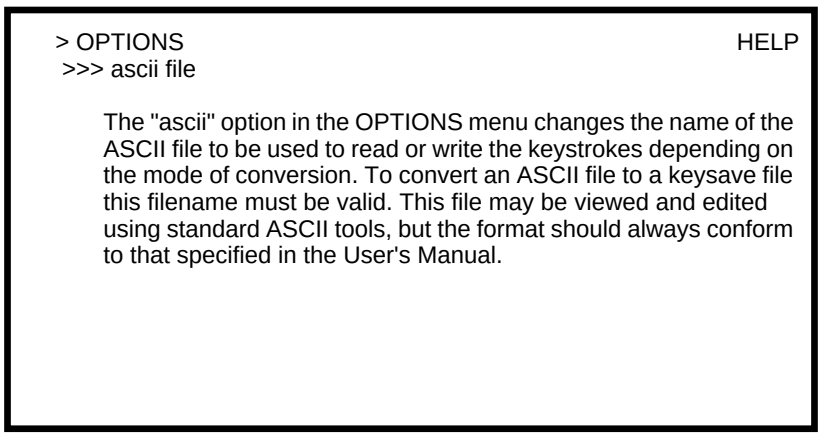
To return to the system from any menu other than MAIN, use the DEL key.

If a system command needs to be executed from within **keycvt**, use the "!" symbol, and the system command, e.g.:

MAIN:!sort myfile


> OPTIONS HELP
>>> configuration file
The "config" option in the OPTIONS menu changes the name of the configuration file to be used. When keycvt is invoked it looks for a configuration file ".rc", which can be used to automatically set all the options. If these settings are to be used again, then use the "save" option to copy them to the configuration file. To use a different configuration file when invoking keycvt use the -r option, e.g.:
-r myfile
The "settings" command displays the current settings and options. If in doubt about how to set up a configuration file with an editor, use the save feature to see how keycvt does it.


> OPTIONS HELP
>> keysave file
The "keysave" option in the OPTIONS menu changes the name of the keysave file to be used to read or write the keystrokes depending on the mode of conversion. To convert a keysave file to ascii this filename must be correct. It is not possible to look at or change this file with normal commands designed for ASCII files, but some debuggers can be used if necessary.


> OPTIONS HELP
>>> ascii file
The "ascii" option in the OPTIONS menu changes the name of the ASCII file to be used to read or write the keystrokes depending on the mode of conversion. To convert an ASCII file to a keysave file this filename must be valid. This file may be viewed and edited using standard ASCII tools, but the format should always conform to that specified in the User's Manual.

> MODE HELP
>>> key_to_ascii

The key_to_ascii option selects the mode of conversion to be from a CAPBAK keysave file to an ASCII text file. The keysave file may also be a trigger key file, in which case the delay factors are actually special codes indicating the function of the keystrokes.

This option toggles with ascii_to_key, and therefore only one mode of conversion can be performed in one "go" command.

> MODE HELP
>>> ascii_to_key

The ascii_to_key option selects the mode of conversion to be from an ASCII file to a keysave file. The ASCII file may or may not have timing, but a default time delay will be added if none exists. The keysave file may also be a trigger key file, in which case the delay factors are actually special codes indicating the function of the keystrokes.

This option toggles with key_to_ascii, and therefore only one mode of conversion can be performed in one "go" command.

> MODE HELP
>>> dos_to_unix

The dos_to_unix option is used to convert DOS keysave files into unix keysave files. This transformation is necessary when you change environments due to the subtle differences in how CAPBAK works in these two environments.

```
> MODE                                                    HELP
>>> unix_to_dos

      The unix_to_dos option is used to convert unix keysave files into
      DOS keysave files.  This transformation is necessary when you
      change environments due to the suble differences in how
      CAPBAK works in these two environments.
```

```
> OPTION                                                  HELP
 >>> timing

      The timing option selects the timing sub-menu which allows the
      user to perform one of seven commands for controlling the
      inclusion and value of timing information for both key 2 ascii and
      ascii 2 key transformations.

      Commands that may be executed from the timing sub-menu are:

      1) clip <1..9> ------- Set the maximum time delay.
      2) floor <1..9> ------ Set the minimum time delay.
      3) constant <1..9> --- Set a constant time delay.
      4) expand <1..9> ----- Expand the time delay.
      5) compress <1..9> --- Compress the time delay.
      6) remove ------------ Remove all time delays.
      7) sremove <chars> --- Remove all time delays except after chars.
```

```
> TIMING                                                  HELP
>>> clip

      The clip option allows the user to set a maximum value for ALL
      timing information included in both key 2 ascii and ascii 2 key
      transformations.  The option is executed from the timing menu by
      entering clip followed by the desired constant value (in the range
      of 1 and 9 inclusively).

Valid examples are:          Invalid examples are:

      clip 1                       clip 0
      clip 5                       clip -1
      clip 9                       clip 10
```

> TIMING HELP
>>> floor

The floor option allows the user to set a minimum value for ALL timing information included in both key 2 ascii and ascii 2 key transformations. The option is executed from the timing menu by entering floor followed by the desired constant value (in the range of 1 and 9 inclusively).

Valid examples are:	Invalid examples are:
floor 1	floor 0
floor 5	floor -1
floor 9	floor 10

> TIMING HELP
>>> constant

The constant option allows the user to set a constant value for ALL timing information included in both key 2 ascii and ascii 2 key transformations. The option is executed from the timing menu by entering constant followed by the desired constant value (in the range of 1 and 9 inclusively).

Valid examples are:	Invalid examples are:
constant 1	constant 0
constant 5	constant -1
constant 9	constant 10

> TIMING HELP
>>> expand

The expand option allows the user to expand the value for ALL timing information by a common multiplier in key 2 ascii and ascii 2 key transformations. The option is executed from the timing menu by entering expand followed by the desired multiplier (in the range of 1 and 9 inclusively).

Valid examples are:	Invalid examples are:
expand 1	expand 0
expand 5	expand -1
expand 9	expand 10

> TIMING HELP
>>> compress

The compress option allows the user to compress the value for ALL timing information by a common divisor in key 2 ascii and ascii 2 key transformations. The option is executed from the timing menu by entering compress followed by the desired divisor (in the range of 1 and 9 inclusively).

Valid examples are:	Invalid examples are:
compress 1	compress 0
compress 5	compress -1
compress 9	compress 10

> TIMING HELP
>>> remove

The remove option allows the user to remove ALL timing information for both key 2 ascii and ascii 2 key transformations. The option is executed from the timing menu by entering remove. No parameters are required.

> TIMING HELP
>>> sremove

The sremove option allows the user to remove ALL timing information except AFTER the characters specified. Up to 128 characters may be specified. The option is executed from the timing menu by entering sremove. The user will be prompted for characters.

Valid characters that may be entered are:

1) All printable characters.
2) All special characters such as {CR}, {CTRL A}, etc.

APPENDIX A

Port Configuration: Sun 3 & Sun 4

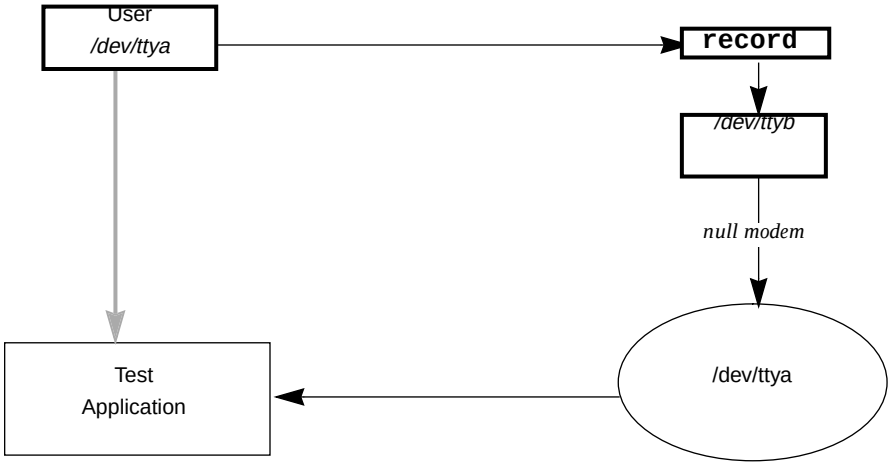
A.1 Port Configuration: SUN-3 & SUN-4

For capture and playback on the same machine, CAPBAK/UNIX on the Sun 3 and 4 requires the use of two serial ports in addition to the port serving as the user console. The two serial ports are linked together in a ''loopback'' or ''wraparound'' fashion, i.e. out one port and into the other. The first port and process drives the second. This first process runs CAPBAK (and possibly SMARTS), and the second runs the system under test. The first process logs into the second.

The physical connection is a standard RS232 cable. A null modem is required.

For this arrangement to work, one port must be a dial*in* port (''loginable'') and the other a dial*out* port (not ''loginable'') — communications to the dialout must not cause the login demons to start a login process.

The diagram on the next page describes an example arrangement with **`ttyb`** used as the dialout port, **`ttya`** as the dialin port, and the user at the system console.



Configure the ports as follows, as superuser:

1. In the file /etc/ttys, set the login bits for the two ports to be 'O' for dialout, and '1' for dialin. The login bit is the first digit of each tty entry. For example, using /dev/ttyb as the dialout and /dev/ttya as the dialin:

```
12ttya
02ttyb
```

2. Reinitialize the ports. This can be done, without rebooting the entire system, using the following command:

```
kill -1 1
```

3. The tty device files must have read permission. For example, the following command will make ttya readable by "user"

```
chmod u+r /dev/ttya
```

Use the **g** or **a** options instead of **u** to widen the range of use.

APPENDIX B

Summary of Command-Line Options

Option	Sample Key Argument	Explanation	Command				
			CAPBAK	CAPKEY	KEYPLA	PLABAK	RECORD
Keysave File	-k temp.ksv	File to save keystrokes in	l	l	l	l	l
Baseline File	-b tem.bsl	File to save baseline responses. Default = capbak.bsl	l				l
Baud Rate	-B 1200	Baud rate. Default = 9600	l			l	l
CR Delay	-c 2 -c	Delay n seconds after each carriage return. Turn delay off (default).	l			l	
Playback Delay	-d 1.5	Delay or speed up playback by specified factor	l		l	l	
Echo	-e -E	Turn echo ON. Turn echo OFF (default).		l	l	l	
Response File	-i temp.rap	File to save remote responses. Default= capbak.rsp.	l			l	
Marker Display	-m -M	Display text entered in marker mode. Turn display OFF (default).	l		l	l	
Port Name	-p /dev/ttya	Port used to send/receive. Default = /dev/tty0	l			l	l
Configuration File	-r config.rc	Read parameters from named configuration file. Default = capbak.rc	l	l	l	l	l

APPENDIX C

On-Line Help Facility

The interactive mode of capbak provide an on-line help facility. From any interactive mode menu, you can get help by typing:

help *command-name*

capbak responds by showing the user a screen of data describing how to use the selected commands. The available help frames are shown on the following pages.

C.1 General Help Frames

```
        > CAPBAK/UNIX                                                HELP
>>> ? (General help)

CAPBAK/UNIX provides four types of commands to record and play back software tests. The
interactive version allows the user to invoke these four functions. It is also possible to call the
CAPBAK/UNIX functions with command-line options.
The CAPBAK/UNIX commands are:
        capbak---to invoke the interactive menu processor
        capkey---to generate a keysave file outside an active session
        keypla---to play back a session to standard output
        record---to record a session
        plabak---to play back a previously recorded session.

In interactive mode, type “help <command name>” to get help screens for: baseline, commands,
convert, go, help, keycvt, keypla, keysave, marker-mode, mode, options, plabak, record,
response. ? gets this frame. You can use any fragment for help frames, e.g. h[elp] he [lp].

CAPBAK/UNIX                                                          HOO
```

```
> CAPBAK/UNIX                                                        HELP
>>> HELP

The CAPBAK/UNIX system commands are:

        capbak---to invoke the interactive menu processor
        capkey---to generate a keysave file outside an active session
        keypla---to play back a session to standard output
        record---to record a session
        plabak---to play back a previously recorded session.

In interactive mode, type “help <command name>” to get help screens (like this one) one most
topics. For detailed information please consult the CAPBAK/UNIX User Manual.


CAPBAK/UNIX                                                          H12
```

C.2 MAIN Menu Help Frames

> MAIN HELP
>>>mode MENU

The "mode" option selects the MODE menu which is use the choose the mode of operation for CAPBAK/UNIX. The options are:

record	--record a test session
playback	--play back a test session.
marker	--turn the marker mode on. (Marker mode not yet implemented in interactive mode must be turned on from command-line or rc file).

If the "go" option is selected without the setting mode, the user will be asked to enter the appropriate information.

The values entered as options can be displayed with the "settings" command.

CAPBAK/UNIX HO1

> MAIN HELP
>>> options MENU

The "options" option selects the OPTIONS menu which is use the choose various parameters for CAPBAK/UNIX. The options are:

keysave,response,baselinechange the name of the file

delay	--slow down a test play back session
baud	--change the line speed.
port	--select the line
echo	--echo the responses during play back.

If the "go" option is selected without the setting mode, the user will be asked to enter the appropriate information.

The values entered as options can be displayed with the "settings" command.

CAPBAK/UNIX HO2

> MAIN HELP
>>> go

The “go” option activates play back or record as selected in the MODE menu.

If the “go” option is selected without setting the mode, the user will be asked to enter the appropriate information.

The record or play back programs are then executed, using the current settings. The INTR or ^D key may be used to stop execution. INTR may also be used the enter marker text.

If the current settings have not already been saved, the user is prompted to do so.

CAPBAK/UNIX HO3

C.3 MODE Menu Help Frames

> MODE HELP
>>> record

The "record" option activates record program by setting the record mode on. In record mode, keystrokes typed by the user are saved in the keysave file, and all responses from the machine under test are saved in the baseline file.

The record program is executed, using the current settings. The INTR key or ^D may be used to stop the program. INTR may also be used to enter marker text.

If the current settings have not already been saved, the user is prompted to do so.

The values entered as options can be displayed with the "settings" command.

CAPBAK/UNIX HO4

> MODEHELP
>>> playback

The "playback" option activates the plabak program.

In plabak mode, keystrokes in the keysave file are sent to the remote machine, and all responses from the machine are saved in the response file.

The plabak program is executed, using the current settings. The INTR key may be used to stop the program.

If the current settings have not already been saved, the user is prompted to do so.
The values entered as options can be displayed with the "settings" command.

CAPBAK/UNIX HO5

> MODE HELP
>>> convert

The “convert” option selects the mode where plabak expect to process a keysave file created using CAPBAK/DOS. In this mode, all keystrokes typed by the user are played back with the timing adjusted for the UNIX machine.

If the current settings have not already been saved, the user is prompted to do so.

The values entered as options can be displayed with the “settings” command.

This mode is NOT available on CAPBAK/UNIX at present.

CAPBAK/UNIX HO7

> MODE HELP
>>> marker mode operation

The “marker” option activates the record program and sets the marker mode on. In this mode the user can record a message within the keysave file for later use as documentation, and possibly for advanced methods of flow control.

The user is prompted after each INT key for the text of the marker mode message.

NOTE: Marker mode is not yet implemented in interactive mode, and must be turned on from the command line or the rc file.

CAPBAK/UNIX HO8

C.4 OPTIONS Menu Help Frames

> OPTIONS HELP
>>> keysave

The "keysave" option selects the name of the keysave file to which all keystrokes should be saved in record mode and read from in play back mode.
This file is not in human readable form, but can be translated to editable ASCII using the KEYCVT utility. KEYCVT can then be used to translate edited ASCII versions back to keysave format. KEY-CVT can also be used to change the play back speed.

The filename may be specified on the command line with the-k flag.

The values entered as options can be displayed with the "settings" command.

CAPBAK/UNIX HO6

> OPTIONS HELP
>>> response

The "response" option selects the name of the file in which all responses from the remote machine are to be saved in play back mode.

The response file may be specified on the command line with the -k flag.

The values entered as options can be displayed with the "settings" command.

CAPBAK/UNIX HO9

> OPTIONS HELP
>>> baseline

The “baseline” option selects the file to which all responses from the remote machine are to be saved in record mode.

The baseline file has special marker records between responses which are used to maintain flow control in play back mode. This file can be edited using the standard editors.

The filename may be specified on the command line or the user will be prompted for the name to use.

The values entered as options can be displayed with the “settings” command.

CAPBAK/UNIX H10

C.5 System Command Help Frames

```
> CAPBAK/UNIX                                                    HELP
>>> capkey

Syntax
        capkey -k <file> [-r <file>]
Options
        -k <file>       Record keystrokes in <file>. This argument is required unless the keysave
                        file is specified in the configuration file.
        -r <file>       Use <file> as the configuration file. If no file is given, capbak looks for the
                        default configuration file "capbak.rc". Note that command-line options will
                        override configuration file options if both are open.
        -e                                                          Turn echo ON.
        -E              Turn echo OFF

CAPBAK/UNIX                                                        H13
```

```
> CAPBAK/UNIX                                                    HELP
>>> keypla

Syntax
                keyplay -k <file> [-d <factor>] [-e [-m -M] [-r <file> ]
Options
        -k <file>       Keysave file to play back keystrokes from.
        -d <factor>     Delay or speed up play back by specified factor.
        -e              Turn echo ON.
        -E              Turn echo OFF (default).
        -m              Display text entered in marker mode.
        -M              Turn display OFF (default).
        -r <file>       Use <file> as configuration file. Default= "capbak.rc".


CAPBAK/UNIX                                                        H14
```

```
> CAPBAK/UNIX                                                    HELP
>>> keycvt

This utility enables the user to convert keysave files generated either by CAPBAK (for DOS) or
CAPBAK/UNIX into ASCII format, and then back to keysave format. These files can then be
edited and, if required, transported to another environment such as DOS (or vice-versa).

KEYCVT. provides a compatibility bridge between DOS and UNIX testing.




CAPBAK/UNIX                                                      H15
```

```
> CAPBAK/UNIX                                                    HELP
>>> capbak
The capbak command invokes the interactive mode of CAPBAK.

Syntax
        capbak -k <file> [-b <file>] [-B <baud>] [-c n\ -C]
         [-d<factor>] [-e\ -E] [-i <file>] [-m/-M] [-p <port>} {-r <file>}
Options
        -k <file        >                                   File to save keystrokes in.
        -b <file>       File to save record responses in. Default = “capbak.bsl”.
        -B<baud>        Baud rate. Default =9600.
        -c<n>           Delay n seconds after each carriage return.
        -d <factor>     Delay or speed up playback by specified factor
        -e              Turn echo ON.
        -E              Turn echo OFF(default).
        -e <file>       File to save play back responses. Default = “capbak.rsp”
        -m              Display text entered in marker mode.
        -M              Turn marker text display OFF (default)
        -p <port>       Port used to send/receive. Default=/dev/tty0
        -r <file>       use <file> as configuration file. Default = capbak.rc”

CAPBAK/UNIX                                                      H16
```

```
> CAPBAK/UNIX                                                    HELP
>>> record

The record command is used to record a test session. The command-line style invocation sets up
record; a ^D or INTR ends the session and releases the resulting keysave and response files.
Syntax
        record -k <file> [-b <file>] [-B<baud>] [-p<port> [-r                    <file>
Options
        -k <file        File to save keystrokes in.
        -b <file>       File to save responses in. Default = "capbak.bsl".
        -B <baud>       Baud rate. Default=9600
        -p <port>       Port used to send/receive. Default =
                        /dev/tty0.
        -r <file>       use <file> as configuration file. Default="capbak.rc"

CAPBAK/UNIX                                                      H17
```

```
> CAPBAK/UNIX                                                    HELP
>>> plabak

The plabak command replays a previously-recorded session.
Syntax
        record -k <file> [-B<baud>] [-c n/-C] [-d <factor>] [-e /-E] [-i <file>] [-m/-M][-p<port> [-r
<file>
Options
        -k <file        Keysave file to play back keystrokes in.
.       -B <baud>       Baud rate. Default=9600
        -c <n>          Delay n seconds after each carriage return.
        -d <factor>     Delay or speed up play back by specified factor.
        -e              Turn echo ON
        -E              Turn echo OFF (default).
        -i <file>       File to save play back responses. Default= "capbak.rsp"
        -m              Display text entered in marker mode.
        -M              Turn marker text OFF (default).
        -p <port>       Port used to send/receive. Default =
                        /dev/tty0.
        -r <file>       use <file> as configuration file. Default="capbak.rc"

CAPBAK/UNIX                                                      H18
```

C.6 Additional Help Frames

> Global HELP
>>> opt

The menu selections sometimes require parameters, and if so they can be specified on the same line as the command.

In many cases if the option is not specified, the user will be asked to enter the appropriate information.

The values entered as options can be displayed with the “settings” command.

CAPBAK/UNIX H11

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W