

Computronics
Peek[®] Software
(for Unix)

Computronics
4N165 Wood Dale Road
Addison, Illinois 60101 USA

Phone: 630/941-7767
Fax: 630/941-7714
Email: support@computron.com
Web: <http://www.computron.com>

Copyright © 1998 Computronics. All rights reserved.

Peek® (for Unix) Documentation Table of Contents

PEEK Feature Overview	1
PEEK and PKSH Installation and Use.....	2
General Overview of the PKSH Shell.....	2
General Overview of the PEEK Command	3
Using the Poke Capability.....	4
Commands Available Within PEEK.....	5
PEEK Command Line Options.....	6
-user.....	6
-translate	6
-no_translate.....	6
-poke.....	6
-rate	6
-cominput.....	6
-cpl.....	6
-cpl_poke	7
-password.....	7
-no_exit	7
-show	7
-clear	7
-echo.....	8
-display.....	8
-no_display	8
-full.....	8
Details of PEEK Modes and Settings.....	8
Translate Mode.....	8
Converting Between Dissimilar Terminal Types.....	9
The Polling Rate Selection	10
Poke Mode	11
Sending Messages to the Monitored User.....	12
The Status Users Command.....	12
Executing System Commands Within PEEK.....	12
Listing the Current Parameter Settings.....	13
Special Commands When Inputting User Names.....	13
Installing PEEK	14
Step 1: Loading The Distribution Media.....	14
Step 2: Installing PEEK Into The System Directories.....	15
Step 3: Run the SETUP Program.....	15
Step 4: Setting the PEEK Password.....	15
Step 5: Run the RESETPR Program.....	16
Step 6: Reset file permissions to proper values.....	16
Step 7: Set up Peekable Users	17
Step 8: Set up the program to release shared segments.....	18
PEEK Security Considerations	19
Resetting the PEEK Password.....	19
Using the PEEK Usage Audit Trail	20
Differences From the PRIMOS Version	21
Display, Remote Echo, and Full duplex Only Modes.....	21
User Command.....	22

Status Users Command.....	22
Cpl_poke Option.....	22
Line Option.....	22
Common Questions/Problems	23
Notes Regarding PEEK and Pseudo-tty's.....	23
Getting the real tty number	23
Using the \$_ttyname variable tty conditionally.....	23
Pseudo-tty's and different versions of UNIX	24
Controlling Pseudo-tty Mapping.....	24
Peek and Flow Control.....	25
The SHELL Environment Variable and Peek.....	25
The RESETPR command.....	26
Exceeding Your Licensed Number Of Peekable Users	26
Peek as an Audit Trail.....	27
Setting Up Peek As An Audit Trail For A User.....	27
Replaying Captured Peek Log Files	27
Using PKSH for Selected Devices	29
Special Notes for Different Platforms.....	30
Special Note for Users of HP/UX and Peek.....	30
Special Note for Users of IBM AIX 3.2 and Peek.....	30
Special Note for Users of Digital Unix-OSF/1 and Peek.....	31
Special Note for Users of SCO and Peek.....	32

PEEK® Feature Overview

- Provides the ability to monitor any terminal user's activities (except graphical X-Windows terminals) for security or to assist in problem solving.
- Poke mode to allow you to type commands on behalf of another user; you can type anything they can type, including an <interrupt>. This allows you to help other users out when they have trouble.
- Can monitor local users, or remotely logged on users.
- When PEEK is started, all data in the current buffers is displayed; this gives you some past history, to allow you to diagnose user problems even after they have occurred, often after they have disappeared from the monitored user's screen.
- PEEK can be run as a background process to monitor the activities of selected users.
- Output from PEEK can be captured in a file to provide an audit trail, or to allow you to capture error messages that appear on a user's terminal. If a user calls you with a strange error message, they do not need to read it to you (which is subject to error). You can PEEK on them and see the actual error message.
- A message command is included to send a message to the monitored user while they are typing.
- Translate mode to allow you to safely PEEK on users with different CRT types than yours.
- System commands and status users commands are all available without exiting PEEK.
- Minimal CPU utilization while peeking; can easily be tailored (by the user) to meet specific requirements.

PEEK and PKSH Installation and Use

PEEK® is a copyrighted product of Computronics. PEEK® is a registered trademark of Computronics. This document is also:

Copyright © 1998 Computronics. All rights reserved.

General Overview of the PKSH Shell

PKSH allows the system administrator to monitor your logon session. All characters typed or displayed on your terminal are displayed on the system administrator's screen. The system administrator may assist you by sending informative messages or by gaining control of your logon session and typing commands on your behalf.

To use the PKSH shell, simply type the command "**pksh**". When you start the program you will see no differences from your standard command shell. For example, if your default command shell was `SH`, you can continue to enter and execute `SH` commands.

To exit the shell, either enter the "**exit**" command or press the <EOF> key (<control-D>) on your keyboard. This should be done when you are at the command prompt.

General Overview of the PEEK Command

PEEK allows you to monitor the terminal activities of any user who is running the PKSH shell. All characters typed or displayed on the user's terminal are displayed. Control characters are normally echoed with a preceding circumflex (^) symbol. Thus you can monitor a user from a different type of CRT than the one the user is utilizing without worrying about confusing your terminal with various escape sequences.

To use the PEEK command, simply type the command `"/etc/peek"`. You will be asked for the login name or process id of the user to monitor. Enter the login name or process number of any terminal or remote user. You can also type the device number, such as `/dev/tty3`. If you press "S" <return> you will get a list of peekable users. If you specify a user id and more than one terminal is logged in under that id, you will be presented with a menu of possible users for you to select. (If you have all numeric user id's, precede the id with a "#". Otherwise an all numeric argument is interpreted as a process id.)

To exit the program, press either the "Q" (quit) key or the "E" (exit) key on your keyboard. This can be done at any time.

PEEK assumes that the monitored user may have a different terminal type than yours. *If your terminals are the same, and you wish control characters and escape sequences to be executed by your terminal rather than converted to printable characters, use the `"-no_translate"` option when you invoke PEEK. Simply start PEEK with the command `"/etc/peek -nt"`. This is discussed in more detail on the following pages. The `"-cvt"` option can be used to convert between dissimilar terminal types. See the section *Converting Between Dissimilar Terminal Types* for details on this option.*

The monitored user has no way of knowing that they are being monitored, so that their screen is not disturbed. The load on the system while using PEEK is very small. In fact, PEEK rarely takes over one or two percent of your CPU. Your terminal should be at the same baud rate as the user being "peeked" (or faster). This is important to prevent loss of data. If your line speed is slower than that of the monitored user, you may not see everything that the monitored user sees. (This will not affect the monitored user in any way, however.)

When PEEK is first started, the contents of the terminal output buffer are first dumped. Then, all new user input and output are displayed. *The dump of the old buffer allows you to see past input at the terminal, which can help when diagnosing problems that a user had already experienced.* The amount of "history" can be up to 4K bytes, which is the size of PEEK's internal buffer. This buffer is independent of any buffers within the UNIX operating system.

Note that Computronics also provides a version of PEEK with a 16K buffer. Although this takes more memory per user, it allows a much larger lookback area, and is helpful at high device speeds, such as for TCP/IP connections. This 16K version is included in the standard distribution. It can be installed instead of the 4K version if desired; both versions cannot be used at the same time. Contact Computronics if you need details about this installation procedure.

Using the Poke Capability:

PEEK includes a "poke" capability. Using this mode, you can type at your keyboard and have your keystrokes passed on to the monitored user as if they were typed on their own keyboard. All characters can be poked, *including* special characters such as <interrupt> and <control-S>.

To enter poke mode from within PEEK, press the at-sign (@) on your keyboard. Unless the "no_translate" mode is on (discussed later under Translate Mode), the message "Poke mode on" will be displayed. Now, any characters that you type will be forced into the remote user's input buffer, just as if they had typed them. As mentioned above, all ASCII printable and control characters can be poked. If you press an <interrupt>, this will be simulated on behalf of the monitored user.

To exit poke mode, type the at-sign (@) followed by a carriage return. Note that you can poke an at-sign if you follow it by some character other than a carriage return. In addition, it is possible to use some character other than an at-sign to enter and exit poke mode. The poke entry/exit character can be changed by the "P" command within PEEK, or via the "-poke" command line option. This character can be changed to any other ASCII character, using the "P" command, whenever you are not in poke mode.

Commands Available Within PEEK:

When you are peeking, there are a number of commands that you can execute. They are all activated by pressing a single character on your keyboard. (Note that one character only is used). The following commands are supported within PEEK; they are discussed in detail in the sections that follow. Note that upper case letters are shown; both upper and lower case commands are acceptable, however.

C=	Change remote echo toggle.
D=	Display mode toggle.
E=	Exit Peek program.
F=	Full duplex only mode.
H=	Help message printout.
L=	List status of parameters.
M=	Message send capability.
P=	Poke mode entry/exit character change.
Q=	Quit from the Peek program (alternative to Exit).
R=	Rate of polling change.
S=	Status users display.
T=	Translate mode toggle.
U=	User change (monitored user).
!=	Prompt for a system command from within Peek.
@=	Enter/exit poke mode (character is alterable).

Note that PEEK runs under the PRIMOS operating system as well as many flavors of UNIX. The C, D and F commands are included for compatibility with the PRIMOS version of PEEK only. They are unsupported or unnecessary under UNIX because of the differences in the way PEEK is implemented on those two platforms. Their use will have no effect on the UNIX version of the program and are further discussed under the heading Differences From the PRIMOS Version.

PEEK Command Line Options:

Virtually all of the settings that can be made within PEEK can also be made via command line options when the PEEK command is first invoked. This can be extremely helpful if you often select modes other than the defaults.

The command **PEEK** may be followed by any of the options that follow. The meaning of each of these options is discussed in subsequent paragraphs. Most options accept a number of abbreviations. The alternative forms of each option are listed here, also. Note that many users wish to regularly use sets of options that are not the default. It is suggested that you set up a shell script for your use when you invoke PEEK. This script can set the desired options automatically for you.

`-user, -use, -us, -u <user>`

Monitors the specified user without asking for the name of the user to monitor. `<user>` may either be the logon name of the user, or the user's process id. It can also be the user's tty or device name.

`-translate, -tran, -tr, -t (or)`
`-no_translate, -notranslate, -notran, -not, -nt`

Selects the desired state of the translate mode. This mode can also be altered via the "T" command within PEEK. The default state when PEEK is first entered, without any command line option, is "-translate". Thus this specification is not necessary if the default is desired. The use of this mode is discussed under the heading Translate Mode.

`-poke, -pok, -po, -p <character>`

Allows the default selection of the at-sign (@) character to enter and exit poke mode to be altered. This can also be accomplished within PEEK via the "P" command. Use this command line option if you wish to poke the character string "@<return>" into the monitored user's input buffer.

`-rate, -rat, -ra, -r <value>`

Use this option, followed by a decimal number, to alter the polling rate of PEEK. The default is 100 milliseconds, or one tenth of a second. This option is equivalent to the "R" command within PEEK. More information on this setting is presented in the section The Polling Rate Selection.

`-cominput, -comi, -co, -cpl, -cp, -c`

This option must be present on the command line when PEEK is run from within a UNIX shell script. The reason for the option name is historical (it is compatible with the PRIMOS version of PEEK which may be run from within a "comi" or "cpl" program--both "comi" and "cpl" are command shells provided by the PRIMOS operating system). Note the PEEK can be run as a background process with the output redirected to an output file, if you wish to create an audit trail for a certain user. This option must be used in cases like this to tell PEEK not to look at the keyboard.

`-cpl_poke`

This option informs PEEK that it is to accept commands from an input stream which is a pipe or data file. PEEK will not try to modify the settings of the input stream. Input will be taken from the input stream and interpreted as if they were commands typed at the terminal, until EOF is encountered. When EOF is reached the program will terminate normally.

`-password, -pass, -pa, -pwd, -pw <password>`

This document discusses the use of a password on PEEK to limit the use of this command to users who know the PEEK entry password. If a password has been placed on PEEK, you will be prompted for this password when PEEK is first entered. To save time, some users may wish to use a command line option to specify this password. The password prompt is suppressed if this command line option, followed by the password, is used. We do not recommend that this password be specified within a script file, since this defeats the security provided by having a password on the PEEK command.

`-no_exit, -noexit, -noexi, -noex, -nox, -nx`

Normally Peek is exited if the monitored user logs off, or if you type in a user id for a user that is not on the system. In some situations, it would be convenient to stay in Peek no matter what happens, unless you specifically "quit". This might be handy if you are going from one user to another quickly, and logging them off as you go. Specify the `-no_exit` option to enable this mode.

`-show, -sho, -sh`

This option causes Peek to display a special shortened message of " [Poke on] " and " [Poke off] " when poke mode is started or stopped, even in the `-no_translate` mode. Normally Peek does not display any message when poke mode is entered or exited when you are in `-nt` mode, so that the screen is not altered. With this option, the abbreviated messages shown above will still appear in this situation.

`-clear, -clea, -cle, -clr, -cl`

If you start to Peek on a user and you have used the `-clear` option, your terminal screen will be cleared (using the "clear" command). It will also be cleared if you switch from one user to another. Note that the "clear" command presumes your own `TERM` variable is set properly. Without this option, no screen clear is sent when Peek is started.

`-convert, -cvt`

Use this option to cause Peek to translate escape sequences between dissimilar terminal emulations. You might also need the `-term` option to specify the type of terminal of the monitored user. See the section Converting Between Dissimilar Terminal Types for more details on this option.

Compatibility options:

PEEK runs under the PRIMOS operating system as well as many types of UNIX. The following options are unsupported or unnecessary for the UNIX versions of PEEK because of the way PEEK is implemented under UNIX. They are included for compatibility reasons only. Their use will have no effect on the UNIX version of the program. These modes are further discussed under the heading Differences From the PRIMOS Version:

`-echo, -ec, -e (or) -no_echo, -noecho, -noec, -noe`

This option is provided for compatibility with the PRIMOS version of PEEK. Under the PRIMOS version of PEEK, this option ensures that the data from remote (PRIMENET) users is properly echoed. Under UNIX, this option is unnecessary as PEEK is implemented using a different method than the PRIMOS version.

`-display, -disp, -di, -d (or)
-no_display, -nodisplay, -nodisp, -nodi, -nod, -nd`

This option is provided for compatibility with the PRIMOS version of PEEK. Under the PRIMOS version of PEEK, this option selects the desired state of the display poked data mode. Under UNIX it has no effect. Under UNIX, poked data is always displayed and cannot be suppressed.

`-full, -ful, -fu`

This option is provided for compatibility with the PRIMOS version of PEEK. Under the PRIMOS version of PEEK, this option sets full duplex mode on. This mode prevents PEEK from doing its own echo of user input while the user is in half duplex mode. Under UNIX, this option is unnecessary as PEEK always operates in full-duplex mode.

Details of PEEK Modes and Settings:

Translate Mode:

Normally, PEEK will translate all control characters that are displayed on the monitored user's terminal into a circumflex (^) followed by the character in non-control form. Thus a <control-Z> is shown as "^Z". An <escape> is a circumflex followed by a left bracket. This is the default mode, and it is essential when you are monitoring a user with a different type of terminal than yours and escape sequences are in use (for cursor positioning or to clear the screen, for example). If this were not done, and your terminals were incompatible, it is possible that the clear screen sequence of the monitored user might be the keyboard lock of your terminal. The translate mode eliminates this problem (although it will make formatted screens hard to read).

Sometimes, you may be monitoring a user with a terminal that is compatible with yours. In this case, you might wish to see their screen exactly as they see it. To accomplish this, you turn translate mode off. This can be done by invoking PEEK with the "-no_translate" command line option or via the "T" command within PEEK. When translate mode is turned off, you will not see a message when you enter or exit poke mode, or when you change the display mode setting. This is to keep your CRT screen in synchronization with the monitored user as much as possible.

Converting Between Dissimilar Terminal Types:

Version 1.9g or later of PEEK contains a terminal type conversion feature. If you wish to monitor a user who is using the same terminal type as yours, you *should* use NO_TRANSLATE mode, as in:

```
/etc/peek -nt
```

This option is always supported and is much faster than terminal type conversion. However, if the terminal type differs, and both your terminal type and the monitored users are supported types for conversion modes (see below), use the convert mode. In convert mode (that is, -cvt or -convert), PEEK will translate between dissimilar terminal types. Note that this does slow down the peeking, but in no way affects the user you are peeking at. Also, not all escape sequences can be converted. Some sequences are not available on certain terminal types, and cannot be emulated. Such sequences will be ignored. However, for most users, the screen of the user running PEEK will be quite close to the screen of the user being monitored, if not identical.

To invoke PEEK using the conversion mode, use the command:

```
/etc/peek -cvt -term <term_type>
```

Substitute the destination user's terminal type for the <term_type>.

You can also specify the destination user's type in other ways. This is the method used to determine their terminal type:

- 1) If the command line option -TERM is used, the specified terminal type will be utilized.
- 2) If the option is not used, the environment variable USER_TERM is used.
- 3) If this is not set, a file /etc/term_types is used. This file contains the device names of the users in the left column and their terminal types in the right column. Wildcards are accepted in the device names. So if tty1 is wyse50 and all other users are vt100, set up this file as:

```
/dev/tty1    wyse50  
/dev/*      vt100
```

- 4) If none of the above are used, your own TERM variable is utilized.

The terminal type of your own process running PEEK is assumed to be the setting of the TERM variable unless you set the variable OUR_TERM. This OUR_TERM variable will override the TERM variable. This is handy if you need to specify a terminal type to the PEEK program that differs from your normal TERM setting.

Supported terminal types for `-CONVERT` mode are below [synonyms in ()]. (This list is accurate for revision 1.9g. Additional types may be available in your implementation. Use the command `"/etc/peek -help"` to get a listing for your version.)

`vt100 (vt-100)`, `wyse50 (wy50,wy-50)`, `wyse60 (wy60,wy-60)`, `ibm3161`,
`ibm3163`, `adds-vp (viewpoint)`, `dg (dg413,d413)`, `dg-unix (dg413-unix)`

The Polling Rate Selection:

PEEK normally looks at your keyboard for a command and looks at the remote user for characters to be displayed every tenth of a second. This provides a PEEK capability that takes very little CPU overhead. However, PEEK may not show you every character from the monitored user's terminal with this poll rate. If the poll rate is set to zero, you will probably not miss any character from the monitored user's buffers, but the CPU usage of PEEK will increase dramatically. On the other hand, the poll rate can be increased beyond one tenth of a second to lower PEEK's CPU utilization at the expense of increasing the possibility of lost data.

The poll rate may be altered via the `"-rate"` command line option or via the `"R"` command within PEEK. In both cases, the value specified is in thousandths of a second (milliseconds). For most applications, the default of 100 (one tenth of a second) is acceptable. Any value between zero and 30,000 may be used. Zero specifies constant polling. Any value from 1 to 100 is treated as if it were 100. Other values, greater than 100, may be used in certain applications.

If the monitored user is at a very low baud rate, such as 300 baud, a polling rate as high as 300 will often be workable. At more common speeds such as 1200 baud to 19,200 baud, the default polling rate is usually adequate. If you are monitoring a user at a high baud rate, such as 38,400 baud, or via TCP/IP, and you are missing some of their screens, you can use a poll rate of zero. This will lead to excessive CPU usage of PEEK.

A better solution in the case where you are losing data from the monitored user is to use a larger `pksh` buffer. Normally, `pksh` buffers the last 4096 characters sent to each user. A special version of `pksh` that buffers the last 16,384 characters is also shipped on your distribution media, and can be installed in place of the standard version. Contact Computronics for information on using the larger `pksh` buffer.

Poke Mode:

Use the at-sign (@) (or other character, as specified via the "P" command within PEEK or the "-poke" command line option) to enter poke mode. Again, type this character, followed by a carriage return, to exit poke mode.

If you accidentally change your poke mode entry/exit character, enter poke mode, and forget what character you are using, there is no easy way to find out what that character is. Computronics' experience is that a very common poke entry/exit character is "E". Try typing "E<return>" if you are "stuck" in poke mode. (This is because of a user typing the command "peek" while they are already in PEEK.) If this does not work and you are at a user terminal, it is often easiest to simply force log out your terminal from another terminal. There is no danger of logging off a user that is in peek or in poke mode. You might also determine the poke entry/exit character by pressing each key on your keyboard followed by <return>. Only printable characters are used, and you will eventually find the proper character this way.

Sending Messages to the Monitored User:

Often it is desirable to display a message on the monitored user's terminal, without having your message executed on their behalf (that is, without using poke mode). This is helpful when you wish to provide instructions for the user, for instance. Within PEEK, enter the "M" command to utilize this capability. You will be prompted for the message to transmit. When you hit `<return>`, the message is transmitted to the remote user. Note that any control characters desired can be used in this message (unlike the PRIMOS message command).

The message can appear at any time, even in the middle of their typing. Thus you would normally not use this command if the monitored user was in the middle of a command or a full screen edit.

Note that the UNIX message facility is not used to accomplish this functionality. Therefore, no message header is displayed on the monitored user's terminal. The standard message command within UNIX is available from within PEEK using the exclamation point command, as discussed under the heading Executing System Commands Within PEEK.

The Status Users Command:

When you are running PEEK, press the letter "S" to get a display of those users who are running the `pksh` shell. This is helpful for seeing who is on the system, especially before switching to monitor another user via the "U" command.

The "S" command within PEEK displays a list of "peekable" users. At some sites, this list is quite lengthy. This list will be displayed 24 lines at a time. After the first 24 lines, you will be asked if you wish to view "More?". If you answer "n" the display will stop. If you press `<return>` you will see all remaining users without any pauses (`<return>` stops the program from prompting for "More?"). If you answer "y" or press any other single key, you will be shown the next 24 users and then this process will be repeated.

Executing System Commands Within PEEK:

If you wish to execute a system command, type an exclamation point. You will then be prompted for the system command to execute. The command is executed when you press the `<return>` key after the command.

Note: If you are invoking Peek from a script and wish to prevent users from using this feature (and thus executing Unix commands), invoke Peek with the special option "-RESTRICT". This "!" command will not be available; the "U" command to change the monitored user number will also not be available.

Listing the Current Parameter Settings:

At any time when you are in PEEK, you can type the "L" command to get a listing of the current option settings. Your current modes, monitored user name, poke mode command character, poll rate, and the like are displayed when this "L" command is used. Here is a sample output from the "L" command:

```
User being monitored is fred.  
Translate mode is on.  
Display of poked data is on.  
Remote echo is off.  
Full duplex only mode is off.  
Poll rate in milliseconds is 100.  
Poke mode toggle character is @.
```

Special Commands When Inputting User Names

When you use the "U" command, or when you invoke PEEK without the "-user" option, you will be prompted for a user name. At this time, you can type a user logon name, a decimal process id, or one of the following special commands:

"H"	<return>	Lists commands available at this time.
"E"	<return>	Exit PEEK program.
"Q"	<return>	Alternative way to exit.
"L"	<return>	Lists current settings (the normal "L" command).
"S"	<return>	Status of users currently running PKSH.
"!"	<return>	Prompt for, and execute, any system command.

If you simply press <return> when the user prompt appears, you will exit the user input mode, and you will be returned to your previous PEEK session.

Installing PEEK

The following procedure takes about five minutes to execute, including loading the distribution media. You *must* be logged on as the `root` user to perform this procedure.

Special Note if you are upgrading Peek

If this is not a first time installation of Peek on your system, it may be difficult to install Peek while peekable users are logged onto the system. Some versions of Unix will allow an installation while Peek is in use, but most will not. Step one can be executed at any time, but it is simplest to execute the remaining steps after booting your system or at some time when there are no peekable users logged in. To see if there are peekable users, execute the command `/etc/peek` and type an "s" for the user to monitor. If no users appear, you can safely install Peek. If some users are shown, you can use poke mode to log each of those users out, and then continue with the installation.

Note that you may wish to get a license code in advance of your upgrade. To do this, run the script "display_new_code" that is provided in your PEEK distribution directory. This can be done while users are on, and will enable you to get your new code while running an older version. While this command is run, `pksh` is disabled, but only for a fraction of a second. Only users who log in during the exact instant when this command is being run might not be peekable, but no errors will occur.

Step 1: Loading The Distribution Media

Minimal disk space is required to load the PEEK program and support utilities. For most platforms, 2MB or less are used. If you decide to enable audit logging, the log files themselves may require more space, depending on how often you use the PEEK program.

PEEK will be supplied to you on either a magnetic tape or floppy diskette media. Use `cpio` to restore the program files from the media.

First, load the floppy or tape in the drive.

Now, type the following command, based on your media format: Typical device names are: `/dev/rfd0` (for a floppy), `/dev/rmt0` (for a magnetic tape under AIX), `/dev/rst1` (for a tape under SunOS4), `/dev/rmt/0` (for a tape under DG/UX, Ultrix, MIPS RISC/OS, or Solaris 2), or `/dev/rmt/0m` (for a 4mm DAT tape under HP-UX).

```
cpio -icmuvd < /dev/rfd0 (or use another device name, as appropriate)
```

Note that the normal device name for your system is shown on the label on the media that you have received. This `cpio` command will create a directory named with the version of PEEK that you have received. Thus for version 1.09g you will now have a directory called `peek109g`. In the following sections, this directory will be referred to by the generic name `peekxxx`.

Step 2: Installing PEEK Into The System Directories

Install the PEEK and PKSH commands by executing the commands:

```
cd peekxxx
./install_peek
```

This script will install PEEK into the proper directories. Note that it will ask where you wish to place "pksh". The normal location is /usr/bin, but /bin may be used if you prefer, or any other directory that is appropriate for your system.

Step 3: Run the SETUP Program (You can skip this step when upgrading Peek)

While in this same **peekxxx** directory, run the SETUP program by typing:

```
./setup -i
```

A special code will be displayed. Call, Fax, or email Computronics or your distributor with this exact code. You will be provided with a new code. When you have this code, type the command:

```
./setup
```

(Note that the command is the same as above, just the "-i" option has been left out). You will now be prompted to enter the code provided by Computronics or your distributor. This completes this step. Note that these codes must be specified exactly. There is never a zero in a code, so if you see something that looks like a zero, it is the letter O.

Step 4: Setting the PEEK Password (You can skip this step when upgrading Peek)

Before the PEEK command can be used, you must set the PEEK password.

```
/etc/peek_password
enter a zero to eighty character password (when prompted)
```

This procedure *must* be run even if you do not want to use the password feature. To disable password checking, simply press <return> when prompted for the PEEK password.

Important Note If You Enter A Password Here:

A very common question from users who have just installed Peek is: I think everything is installed right but every time I start the Peek program and answer the prompt "User?" the program just "doesn't work". The prompt "User?" is actually a prompt for the password! If the password is not entered, the program just exits a few seconds later. If you enter the proper password, the program will issue the proper "User to monitor:" prompt. The special prompt of simply "User?" will only appear if you enter a password during this step. For more details, see the section "Peek Security Considerations".

Step 5: Run the RESETPR Program

The **RESETPR** program initializes the tables maintained by PEEK to a state that indicates that there are no current PEEKable users. Furthermore, it frees up any shared memory segments that may have been in use by PEEK. Whenever you install PEEK for the first time or when you install a new version, you should execute this command. (**RESETPR** is not normally used other than in these situations).

```
./resetpr
```

Step 6: Reset file permissions to proper values

Like any program, proper access is needed to certain files for PEEK to function properly. A script file is provided that sets reasonable file permissions for the files used by PEEK. This has been set up as a script so that you can see what settings are being used. Run this program now, as well as any time you have a concern about the file permissions (this could occur if you get messages such as "Peek is not configured" or "Your Peek license has expired" for selected users):

```
./set_mode.sh
```

Step 7: Set up Peekable Users (You can skip this step when upgrading Peek)

A user who has executed the command "pksh" is "peekable". This command can be executed manually, but this is rarely done. If you have users who have access to the UNIX command level, you can simply place the command "pksh" at the end of their ".profile" or ".login" file. At most sites, this is not practical. It is desirable to configure this in a more automatic fashion, and to accommodate users who never see UNIX. This is discussed in the next few paragraphs.

To set up a user to be peekable in a manner that still allows them to execute the full ".profile" or ".login" script and thus to be locked into a certain application, set up their initial program to be "pksh-shellname". Thus if they are to use the Korn shell, specify an initial program of "pksh-ksh". If the C-shell is to be used, specify "pksh-csh". If the Bourne shell is to be used, specify "pksh-sh", or "pksh-bsh" (depending on your system). [Do not simply specify "pksh" as the initial program. This can cause problems if the UNIX "newgrp" command is used.] One sets up the initial program via the `smit` utility (AIX), via some other utility, or via editing the file `/etc/passwd`.

AIX Note: All login shells must be listed in the file `/etc/security/login.cfg`, under AIX. Add the desired `pksh-ksh`, `pksh-sh`, etc., to this file at the appropriate place. If this is not done, `smit` will not accept this name.

Non-AIX Note: Many versions of UNIX have a file called `/etc/shells`. This file contains the names of all legal shells. Add the desired `pksh-ksh`, `pksh-sh`, etc., to this file. If this is not done, some commands may not work properly. If you do not have such a file, create one with the standard shells, plus the `pksh` shells. For details, see "man shells" on your system.

The "install_peek" script that you ran earlier will create a link for the `pksh` command to each of the following potential shells (you will need to add links to other shells, if these are utilized):

```
pksh-sh
pksh-csh
pksh-ksh
```

Note that Peek can be used even if your users do not start a normal shell, as discussed above. If the user's `passwd` entry takes them directly into an application, such as Vmark's Universe, it is possible to set up Peek to do the same thing. Any name can be specified after the "-" in the name "`pksh-xxx`". The `pksh` program will merely eliminate any characters before the hyphen, and then place the string `/bin` before the name. It will then invoke this command. Thus if the user's shell was "`pksh-xxx`", Peek would invoke a program named `"/bin/xxx"` for each user. Using links, it is possible to use `pksh` with any startup shell or program that you wish. Contact Computronics for details if you have questions about setting this up.

Thus you simply need to change a user's initial program specification to make that user peekable.

Step 8: Set up the program to release shared segments

This step is *optional*. If your users access the system via direct connect terminals, you can skip this step. Even if your users use telnet or rlogin to access the system, you may still be able to skip this step. Read the next paragraph for details.

Each Peekable user requires their own shared segment (plus Peek has one globally utilized shared segment). When a user logs off normally, this segment is freed by Peek. In some situations, users do not exit normally. In fact, they break their connection by closing their socket, or they exit via some other function that kills their processes without allowing Peek to be notified that it should clean up. The only problem that this causes is "orphaned" shared segments. You may eventually run out of shared segments in this situation.

If you are new to Peek or not sure how your users log out, it is suggested that you skip this step and wait until Peek has been running for a few days. Then, use the command "`ipcs -mo`" to look at your shared segment utilization. If you see user segments with a key of 0 and a number of users (`nattch`, the last column on the report) that is also zero, you have "orphaned" segments. If the number of such segments is large, or growing, you can use the script provided with Peek to clean them up.

Peek includes a script called "`shm_cleanup.sh`". The function of this script is to remove any unused shared segments from the system. To release unused segments, run "`shm_cleanup.sh`" under the `root` user id. This script simply looks at all shared segments in use and releases any that (1) have no users, (2) are *not* owned by root, and (3) have a segment key of 0.

If you find that this is a common problem, it is suggested that you automate this procedure. This script can be set up to be executed via a crontab entry on a regular basis. Make sure the "`root`" user id's crontab is used. A sample crontab entry is shown in the comments within `shm_cleanup.sh`. If you are not familiar with setting up a crontab entry, you may contact Computronics for assistance, or look at the information on the man page for crontab.

This completes the PEEK and PKSH installation procedure. If you wish to enable the audit logging feature, see the section Using the PEEK Usage Audit Trail below. For further information on restricting access to the PEEK program, see the separate document Peek for UNIX Access List Configuration.

PEEK Security Considerations:

Prior to using PEEK the first time, a password must be set on its use. The installation instructions discuss the procedure to follow to set a password on access to the PEEK command. When the password is enabled, you will be prompted for a password via a prompt of simply "user?". This prompt is deliberately misleading to keep users from guessing that a password is being solicited. (A prompt of "password" encourages people to try to guess the password). Type in the current password at this time. The password may be up to eighty characters in length. Lower case characters are treated as equivalent to upper case characters.

If you wish to suppress this prompt, use the "-password" command line option, followed by the correct password.

If desired, the file permissions on the `/etc/peek` program can be set to restrict the use of PEEK to a selected group of users.

A separate document, titled Peek for UNIX Access List Configuration, discusses PEEK security in more detail, including the many more extensive security capabilities afforded by PEEK. This document discusses advanced capabilities, such as controlling the list of users that a given user can peek on, and whether they will have access to poke mode. This document is shipped with all printed copies of the manual as a separate manual.

Resetting the PEEK Password:

To set a password on the PEEK command, execute the following sequence of commands:

```
/etc/peek_password  
enter a zero to eighty character password (when prompted)
```

Note that the `peek_password` command is very fast, and can be rerun as often as desired. Changing the password has no effect on the ongoing operation of the system. PEEK does not need to be reinstalled, nor does the system need to be booted when you change the password.

Using the PEEK Usage Audit Trail:

PEEK can be configured to generate an audit trail of its use. This file will contain an entry whenever a user enters or exits PEEK, and when the user enters or exits poke mode within PEEK. There is very little overhead involved with this feature, so you may wish to turn this log on to enhance your security.

To enable the audit trail, create a directory called "**peek_usage**" in the **/etc** directory. If you wish to keep the audit files in different system area, define the "**PEEK**" environment variable to contain the name of this new directory. Set the permissions to **rw** for **root** only to restrict other users from accessing this area.

```
cd /etc
mkdir peek_usage
chmod 600 peek_usage
```

The log files will be kept in this **peek_usage** directory, which may grow to a few hundred disk records, depending on how often you use PEEK at your site. The file will be named "**peek_log.mmyy**", where "**mmyy**" is the current month and year. Thus a new log file is created automatically for each month. Note that a two digit year is used, thus the log file for January 2000 is named **peek_log.0100**.

<u>Columns</u>	<u>Meaning</u>
1	Record type, one of the following values: 1 = PEEK was started by the specified user at the specified date and time. 2 = Poke mode was entered by the specified user. 3 = Poke mode was exited. 4 = PEEK was exited. 5 = PEEK was given a bad user or line number, and was exited.
3 to 6	Process id of user invoking PEEK.
8 to 39	User name of user invoking PEEK.
41 to 72	Group name of user invoking PEEK.
74 to 90	Date and time PEEK was invoked (or exited, based on the record type), in the form "mm/dd/yy hh:mm:ss".
92 to 95	Type of user being monitored. The Unix implementation of PEEK always substitutes the value of 1 for terminal users.
97	Code for assigned line versus user monitoring. The Unix implementation of PEEK always substitutes "u" for user entries. Assigned lines are not supported in the UNIX version of PEEK.
98 to 101	Process id of the user that was monitored.
103 to 134	User id of the user that was monitored.

Let's look at a few sample log entries (extra blanks have been removed for clarity):

```
1 15 root    other    11/05/97 13:12:59 1 u0001 sysadmin
2 15 root    other    11/05/97 13:13:03 1 u0001 sysadmin
1 27 myuser  other    11/05/97 13:13:18 1 u0015 root
3 15 root    other    11/05/97 13:13:27 1 u0001 sysadmin
4 15 root    other    11/05/97 13:13:32 1 u0001 sysadmin
4 27 myuser  other    11/05/97 13:13:55 1 u0015 root
```

In this example, the following actions occurred: process 15, logged in as user id "root" under the group "other", invoked PEEK at 13:12:59 on the specified date. They entered poke mode 4 seconds later. Fifteen seconds later, user "myuser" invoked PEEK to look at user "root". Nine seconds later, at 13:13:27, root exited poke mode, and 5 seconds later they exited PEEK. Finally, at 13:13:55 "myuser" exited PEEK.

Note that multiple log entries may be written for a single PEEK session. An entry is written when the user number monitored by PEEK changes, which will happen when you use the "U" command while in PEEK.

These log files are standard ASCII, editable and printable files, You may wish to write your own reporting programs to work with these files or to match up started and stopped entries, for example.

Differences From the PRIMOS Version

PEEK runs under the PRIMOS operating system for the 50 series of minicomputers as well as several versions of UNIX. The particulars of how PEEK is implemented under UNIX is quite different from its PRIMOS counterpart. This causes some of the original PEEK options and commands to become obsolete. Also, the operation of some of these commands has changed.

Display, Remote Echo, and Full duplex Only Modes

Options and commands related to display, remote echo, and full duplex only modes, are included only for compatibility with the PRIMOS version of PEEK. Their use has no effect on the UNIX version of the program.

The PRIMOS version of PEEK accesses the user's input and output buffers in order to provide the peek and poke functionality. The UNIX version does not process the I/O buffers directly. Instead, the user runs a program which invokes a secondary command shell. Output from the secondary command shell is piped back to the recording program for display on the user's screen. A copy of the output is sent to the system administrator if they are monitoring the user's session using PEEK.

The PRIMOS version of the program has to be concerned as to whether the user is local or remote, if NTS is being used, or if the user program is in half or full duplex. The display, remote echo, and full duplex commands are required to ensure characters are displayed properly on the monitoring user's screen. Under certain circumstances it is possible that user input could be echoed twice, or not at all. The display, remote echo, and duplex options correct this problem.

The UNIX version of PEEK implements data capture in an entirely different way. With the UNIX version of PEEK, screen data is always displayed properly regardless of what mode a program may be in.

User Command

Users must run the `pksh` program to allow monitoring of their logon sessions. Sessions are identified by the user's logon name and process id rather than their terminal line number. PEEK for Unix accepts logon names, device names, or process id's in response to the `user` command or `-user` option.

Status Users Command

The "`status users`" command lists only those users that are running the `pksh` shell. For a complete listing of all the users on the system, use the exclamation point (!) to issue an equivalent UNIX command (e.g. "`ps -ef`").

Cpl_poke Option

Instead of providing a separate program, the `-cpl_poke` command line option has been provided to allow access to poke mode from within a UNIX shell script. Its use was described earlier in this document.

Line Option

The PRIMOS version of PEEK allows monitoring of assigned lines as well as user sessions. Since, under UNIX, the `pksh` program must be run by a logged in process, it is not possible to monitor assigned lines using the UNIX version of PEEK.

Common Questions/Problems

Notes Regarding PEEK and Pseudo-tty's

Getting the real tty number

When you invoke "pksh" (or "pksh-ksh" or another name) you are assigned a "pty" number. This is a pseudo-tty device. Some sites would like to know a user's actual user number, because they use this number in some fashion. This might be for specifying a terminal type to software packages, for extra security, or for any other reason. PEEK will automatically set up an environment variable when it is invoked, which is set to the user's real tty number. This variable is called "**_ttyname**". If a variable by this name already exists, the value of it is replaced by the user's real tty name. If it does not exist, it is added to the environment when "pksh" is started. Thus if you log in using "pksh" and your real tty number was 4, you would have a variable set automatically, just as if you had specified the following lines in your ".profile" file:

```
_ttyname=/dev/tty4  
export _ttyname
```

Your programs or scripts can then use this variable in any manner you wish.

Using the **\$_ttyname** variable conditionally

In a script, you may want to use the `_ttyname` variable to determine the underlying device name for a peekable user, but what happens if a user is not peekable and thus this variable is not set?

In the Bourne or Korn shell, there is a simple solution. To reference the variable conditionally, use this expression:

```
${_ttyname:-`tty`}
```

This syntax will return the variable `_ttyname` if the user is peekable, or the value of the command "tty" if the user is not peekable. Alternatively, if you use the following syntax, the same value will be returned, but the variable `_ttyname` will automatically be set to the value of the "tty" command if the user is not peekable. Thus you can use the name `_ttyname` later in the same script, whenever needed.

```
${_ttyname:=`tty`}
```

Pseudo-tty's and different versions of UNIX

Under some versions of UNIX, such as HP-UX or SunOS, pseudo-tty's are handled differently than under other implementations (such as AIX). On some systems with limited pty resources, PEEK will look for pseudo-tty's named as `"/dev/ptym/pty x y"` where x is a letter and y is a hex digit. PEEK will look for a free pseudo-tty by looking through all groups of potential pty's, regardless of how they are numbered. We suggest you configure your system with enough pseudo-tty's for all possible peekable users as well as any other users of pseudo-tty's, such as Ethernet users. This is set via the `"npty"` configuration parameter, or some similar kernel configuration parameter. Note also that Peek uses a shared memory segment for each user, and one global segment. You may need to alter the maximum number of shared memory segments configured for your system.

Controlling Pseudo-tty Mapping

Peekable users show up as using a pseudo-tty, as discussed earlier. Normally, these "pty's" are allocated dynamically. Some sites run applications programs where it is desirable to form an association between the pty name and the actual device name. Peek allows the customization of the correspondence between the allocated pty's and each user. Normally, this translation is handled automatically (either by the `pksh` program or by Unix, depending on the platform).

If you would like to define your own associations, set up a file called `"/etc/peek_pty_list"`. This file contains a series of lines with two fields on each. The left field is a tty (or pty) name, while the right field is the corresponding pty to use. The full name must be used in both cases. The fields are separated by a space or a tab. Use the type of device name used by your system (this varies for each "flavor" of Unix). For instance, under many versions of Unix, one could use the following line:

```
/dev/tty1/0 /dev/ptyp7
```

This would force the specified tty to use the pseudo-tty named `/dev/ptyp7`. Peek will dynamically allocate pty's if no `"peek_pty_list"` file exists, or if there is no entry for a given terminal. If you are unsure of the types of device names used on your Unix platform, log in without Peek, execute the command `"tty"`. Then start the `"pksh"` program and execute the `"tty"` command again. You will see an example of each type of device name.

Note that all users need read access to the `peek_pty_list` file, so execute the following command after creating this file:

```
chmod ugo+r /etc/peek_pty_list
```

Note for users of Peek on Silicon Graphics (SGI), Dynix, OSF/1 (Digital Unix), and NCR/AT&T systems: The <code>peek_pty_list</code> feature is not available on these platforms, due to the design of these Unix implementations. This feature can be used on all other platforms supported by Peek.
--

Peek and Flow Control

Flow control is handled by PEEK in the following fashion: If you invoke "pksh" with flow control enabled, it will be handled directly by PEEK. If it is off, it will be left off. This will cause one side effect: If you invoke a program that wants to treat <control-S> and <control-Q> as normal text characters, rather than using their normal flow control interpretations, you may need to disable these options before invoking "pksh". For example, use the command "stty -ixon -ixoff" before invoking "pksh". This will cause these characters to pass through as text characters. This is admittedly very rarely done, but PEEK will work properly whether you use these characters for flow control or not. (If PEEK were to always pass on the flow control characters, response to a <control-S> would be very slow. In many situations, buffering and system delays could cause the response to a <control-S> to be delayed a great deal, sometimes by several thousand characters.)

The SHELL Environment Variable and Peek

The environment variable "**SHELL**" is set to the user's shell name in all flavors of Unix. When you invoke the Peek Shell and it invokes another shell, this variable would normally have a value such as "/usr/bin/pksh-ksh". Certain commands on certain platforms (such as the "script" command on AIX) do not like this name and will not function properly. The variable "SHELL" is automatically set by Peek to the normal value without the "pksh-" prefix. That is, the variable would be set to "/usr/bin/ksh" in the example above.

The RESETPR command

The "**resetpr**" command can be used to reset the tables used by Peek. It is a good idea to execute this command whenever the system is booted, although this is not absolutely necessary. This command asks for permission to perform the reset. If you are executing this command from a script file, run the program with the "**-nq**" option to suppress this question. That is, utilize the command "**resetpr -nq**". There is also a debug option, "**-d**", that displays a detailed dump of the status of Peek. This is often helpful to Computronics when problems occur.

"When I am Peekable Certain Commands do not Work Properly"

Some users notice that most programs work fine, but a few behave differently or don't work when a user is running pksh. If this is the case, check for a file called "shells" in the directory "/etc". This file should contain the names of all legal shells (such as /bin/sh, /bin/pksh-sh, /usr/bin/ksh, etc.) If there is no such file, set one up and give all users read access to this file. This will often solve problems with commands that do not work if you are peekable.

Exceeding Your Licensed Number Of Peekable Users

Peek will not provide an indication to a user if they log in and exceed the number of Peekable users that you have licensed. This is to eliminate any confusion on the part of your users.

A file in /etc called **peek.log** is created when Peek is installed. This file logs the date and time when a user logs in that exceeds the maximum number of peekable users. This file is for your use to determine if you need to increase the licensed number of peekable users.

Peek as an Audit Trail

Setting Up Peek As An Audit Trail For A User

Users often ask about using Peek to capture the activities of a user. One can capture the activities of a user by simply using output redirection. For example,

```
peek -user john > /tmp/john.log
```

All of the activities of user "john" will be placed in the file "/tmp/john.log". Let's take this a step further. If we want to set up a user's .profile or .login to automatically create such a log of their activities, we need to perform two steps:

- 1) Create a file called "/etc/peek_on_me" containing this one line (any file name can be used):

```
/etc/peek -u $_ttyname -comi > logfile &
```

In the above, substitute your desired output file pathname for the "logfile". Type the line exactly as shown, with the ampersand at the end.

- 2) In the user's .profile (Bourne or Korn shell) or .login (C-shell), place the following single line:

```
sh >/dev/null 2>&1 /etc/peek_on_me
```

Type this line exactly as shown.

That's all you need to do! When the other user logs in, the "peek_on_me" script will be started as a background job. It in turn will invoke Peek using another background job. These two processes are required to prevent Peek from erroring out with a message like "You can't Peek on yourself." Note that the first background job will log off immediately, the second one will run until this user logs out themselves.

Replaying Captured Peek Log Files

A handy use for Peek is to capture an audit trail of the activities of a user. This is useful for security purposes as well as for debugging and training. You can capture the output of a Peek session either with output redirection or by using the tee command:

```
/etc/peek -u userid -nt > /tmp/logfile
```

The above command will start a Peek session monitoring the designated user id while storing their output into the specified logfile. This command is best used as a background job since no output is sent to your screen when Peek is used this way; all output goes to the file. All data, including control characters, is written to this file when the "-nt" option is specified.

```
/etc/peek -u userid -nt | tee /tmp/logfile
```

This command is similar, but is handiest when used interactively. The output still goes to your screen as well as to the logfile, using Unix's standard "tee" command.

Now that you have a log file, you can use `more`, `cat` or `vi` to view this file. However, if you wish to look at the screens and to have the display pause when new screens start, you can use the "peek_replay" command provided with Peek version 1.9a or later. Here's how:

Set the environment variable `PEEK_BREAK` to the character strings where you wish the replay to pause. These are typically clear screen sequences but might vary for different applications. The `PEEK_BREAK` variable is set like any other variable, depending on the shell you are using. Thus in the Bourne or Korn shell, use the `=` command and an `export` command. In the C-shell use the `setenv` command.

The variable is set using a syntax like the `termcap/terminfo` entries, with multiple sequences delimited by a ":". The standard special characters are supported. These characters are `\e` for escape, `\n` for new line, `\t` for tab, `\r` for return, `\a` for bell, `\b` for backspace, `\f` for form feed, `\\` for a `\`, and `\xxx` for octal character "xxx". You can also specify control characters with a leading "^". Thus `^Z` would represent a `<control-Z>`.

The standard default if you don't set `PEEK_BREAK` is:

```
PEEK_BREAK=\e[H:\e[1;1H
```

This specifies that Peek Replay will pause when the output file contains a home cursor sequence on an ANSI terminal, either an `<escape>[H` or an `<escape>[1;1H`. For most applications you will want to set a different sequence that matches the characters used by your specific software.

Now, to display the log file, execute the command:

```
/etc/peek_replay /tmp/logfile
```

The Peek Replay command assumes a 24 line screen. To use a different size, specify the number of lines before the file name, as in the following entry for a 48 line window:

```
/etc/peek_replay -48 /tmp/logfile
```

While replaying the file, the program will pause at the specified `PEEK_BREAK` characters, as well as when the specified number of lines per page are reached. If you wish to exit the program at this time, just press "q" or `<control-C>` or `<delete>`. Any other key will cause the next section of the log file to be displayed.

`peek_replay` also accepts input from a pipe, so you can execute sequences like the following:

```
cat file_to_display | /etc/peek_replay
```

Using PKSH for Selected Devices

Usually, sites configure all users as peekable, or they select peekable users by user id. This is done by placing "pksh-" in the shell name of selected users in `/etc/passwd`. Sometimes, however, it is desirable to control who is peekable by the device they are on. This might be the case if you wished to Peek only on modem users, or users in a certain department.

Beginning at revision 1.9c of Peek, it is possible to set up Peek so that you will become a peekable user only when you log in on selected devices. This document describes the procedures to follow to set this up.

First, set up Peek as described in the Peek documentation. Make your selected user id's (or all user id's) peekable using any standard method. The most common method is to specify a shell name in `/etc/passwd` that includes the pksh program. For instance, if your startup shell is normally:

```
/bin/ksh
```

Change this to:

```
/bin/pksh-ksh
```

The above procedure will make these user id's always peekable. To set this up so that only selected devices are peekable, create a file called `/etc/peek_devices`.

In this file, place the device names of all users that you wish to make peekable. The full device name must be used, as in `"/dev/tty17"`. Use the output of the `"tty"` command as a guide for the proper names on your system. Place one entry per line in this file. Entries can be specified in any order. When the `/etc/peek_devices` file exists, you will only be peekable if you run `pksh` *and* if your device name is specified explicitly or via a wildcard in this file.

The `"/etc/peek_devices"` file supports wildcards. So, if you wish to make all users who log in on tty's `tty00` through `tty09` peekable, you could specify:

```
/dev/tty0*
```

If you want a range of devices, this can also be specified in the standard Unix fashion, such as:

```
/dev/tty0[1-5]
```

Make sure all users have read access to the `/etc/peek_devices` file. This is set automatically if you execute the `set_mode.sh` script provided with Peek.

Special Notes for Different Platforms

Special Note for Users of HP/UX and Peek

The clone device

The Hewlett-Packard HP/UX version of Peek will use a clone device if one has been configured. This speeds up the time it takes to log in when a user is using "pksh" and there are many other users on the system. If you are running HP/UX version 9.0 or later, you can use this clone device to speed up the startup of pksh. You need to set up this feature under HP/UX on a one time basis. Simply execute the following command while logged in as root:

```
mknod /dev/ptym/clone c 16 0xffffffff
```

This command only needs to be executed once. (The above command is discussed in the manual "Readme Before Installing or Updating HP-UX, HP 9000 Series 800", on page 1-23.)

Changing the Default Shell for New Users added via SAM

Some users like to make the pksh shell the default shell for new users added via SAM. This will work at HP/UX 9 or 10.

In the file `/usr/sam/lib/C/ug.ui` look for the string "Start-up Program". You will see a section that begins:

```
text_edit ug_anu_sp { label "Start-up Program"
```

The next line reads "default /bin/sh". Just change the path to the desired path, such as `/usr/bin/pksh-ksh`, and that will be the new default when adding a user in SAM.

Special Note for Users of IBM AIX 3.2 and Peek

Problem for IBM RS/6000 users (AIX) with users occasionally "locking up" and using excessive CPU time: Several changes were made in Peek beginning at version 1.6i to eliminate problems with Peek and heavy load situations when pksh was first started. On AIX (and only on AIX) it was possible for the second process that is started by Peek to be taking all available cpu time on the system. This process was also not killable, even with "kill -9". This difficulty was due to problems in AIX with the use of pseudo-ttys. Version 1.6i of Peek (or later) was designed to work around this AIX bug. To avoid problems, IBM suggest that you run AIX version 3.2.5, or at least install IBM's latest ptydd (patches for pty's).

Special Note for Users of *Digital Unix-OSF/1* and Peek

IMPORTANT NOTE: On this platform, you must use the special command `vipw` to edit the `/etc/passwd` file, or use the `mkpasswd` command after editing the file in another manner. If you simply edit the `/etc/passwd` file directly, it will appear that your changes are not working. This is because Digital Unix uses a "cached" version of the password file. The "`vipw`" command updates this cache for you. See "`man vipw`" if you need more details.

OSF/1 and Adding Pseudo-terminals

- First you need to set up the pseudo-devices, unless they already exist. Check in the directory `/dev/pts`. Verify that there are sufficient devices there for the number of users who will be peekable at once, as well as for any Telnet sessions (this does not apply to LAT connections).
- Assuming you need additional devices, execute the `MAKEDEV` script to create them. Each `pty` specification below creates 16 devices, that is, `pty0` creates `pts/0` through `pts/15`, `pty1` creates `pts/16` through `pts/31`, etc.

```
cd /dev
./MAKEDEV pty0 pty1 pty2...
```

Alternatively, `./MAKEDEV PTY_1` will create a block of 384 `pty` devices at once.

- The scripts above create BSD and SystemV type devices. To use the SystemV devices, one merely needs to remove the BSD devices from the system. This is true even if you did not need to run the scripts above. To do this, execute the following commands (this comes from the *DEC OSF/1 Release Notes*, Order#: AA-PS2BD-TE, page 4-12, section 4.4.1):

```
cd /dev
./SYSV_PTY
```

- One more note is that the kernel is configured to support 255 devices by default. To increase this, edit the appropriate configuration file and relink the kernel:

```
pseudo-device rpty #           (this is for streams)
pseudo-device pty #           (this is for clist)
```

Special Note for Users of SCO and Peek

Increasing the number of pts devices under SCO

The PEEK software uses `pts` devices under SCO. This requires that the streams module be loaded, which is an option specified via `sysadmsh`. You may also need to add additional `pts` devices. To do this, follow this procedure:

In `/etc/conf/cf.d/mdevice`, edit these lines:

```
ptm      -      Scio    ptm      0      42      1      32      -1
ldterm   -      Si      ldtr     0      0       1      32      -1
ptem     -      Si      ptem     0      0       1      32      -1
```

Change the 32's to the number of devices needed, such as 256:

```
ptm      -      Scio    ptm      0      42      1      256     -1
ldterm   -      Si      ldtr     0      0       1      256     -1
ptem     -      Si      ptem     0      0       1      256     -1
```

To increase the number of `pts` units, edit `/etc/conf/sdevice.d/ptm`. Change the 2nd field to the desired number of units. You cannot make this larger than 256. Then edit `/etc/conf/node.d/pts`. Add lines similar to the ones that already exist:

```
pts    pts016    c    16
pts    pts017    c    17
...
pts    pts255    c    255 (as many as you need)
```

Now, you need to increase `ldterm` and `ptem` to the same number of devices. To do this, edit the files `/etc/conf/sdevice.d/ldterm`, and `/etc/conf/sdevice.d/ptem`.

Finally, tune kernel parameters `NUMTIM` or `NUMTRW` with `configure(ADM)`. You can do this via `sysadmsh`, using the sequence `sysadmsh -> system -> configure -> kernel -> parameters -> (option 11 Streams Data)`

After making these changes you will need to relink the kernel (this can be done with `sysadmsh` also) and reboot.

Copyright ©1998 Computronics. All rights reserved.
Peek® is a registered trademark of Computronics.

```
(revision 22.1b/16-Jan-91)
(revision 1.0a/23-Jun-92)
(revision 1.6k/15-Nov-94)
(revision 1.9g/12-Nov-97)
```