Server Operating Systems

Lecture 15
Shell Features and Environment Customisation

UNIX Shells

Bourne shell ($)

- original shell program developed by Stephen Bourne for AT&T
- mostly used by system administrators

Korn shell ($)

- superset of the Bourne shell
- developed by Stephen Korn at Bell Labs
- added features such as aliasing and history
- this course is based primarily on the Korn shell

Bash shell ($)

- updated form of Bourne shell (Bourne again shell)
- incorporates many of the added features in the Korn shell.

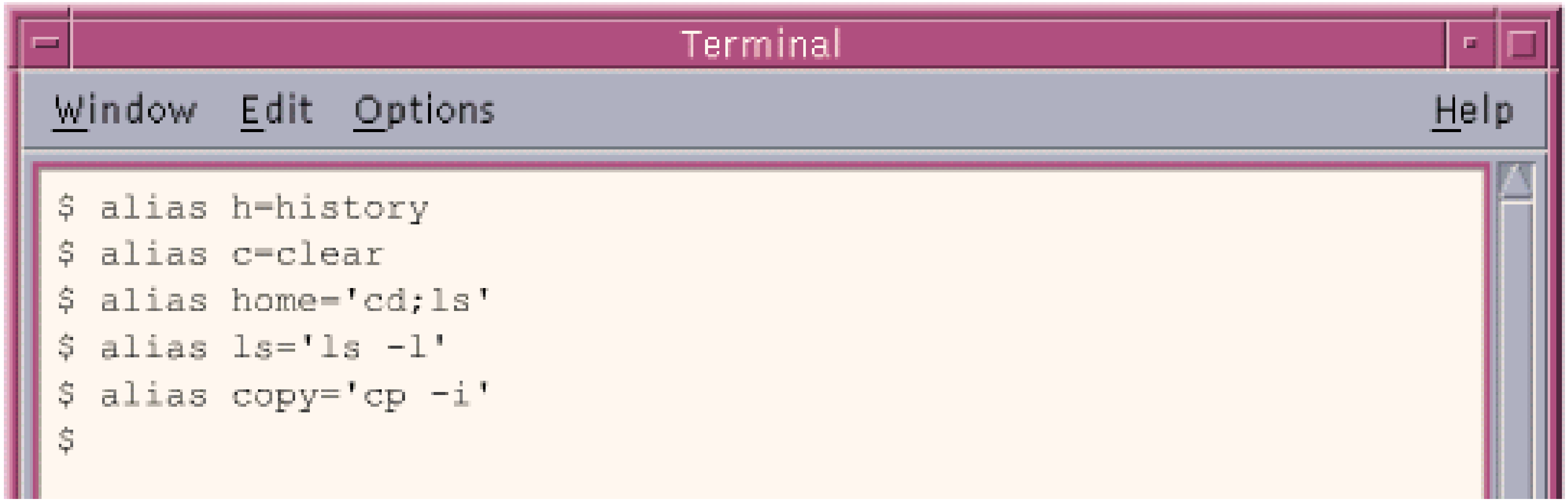
C shell (%)

- based on the C programming language
- developed by Sun's Bill Joy

Reasons to Use Aliases

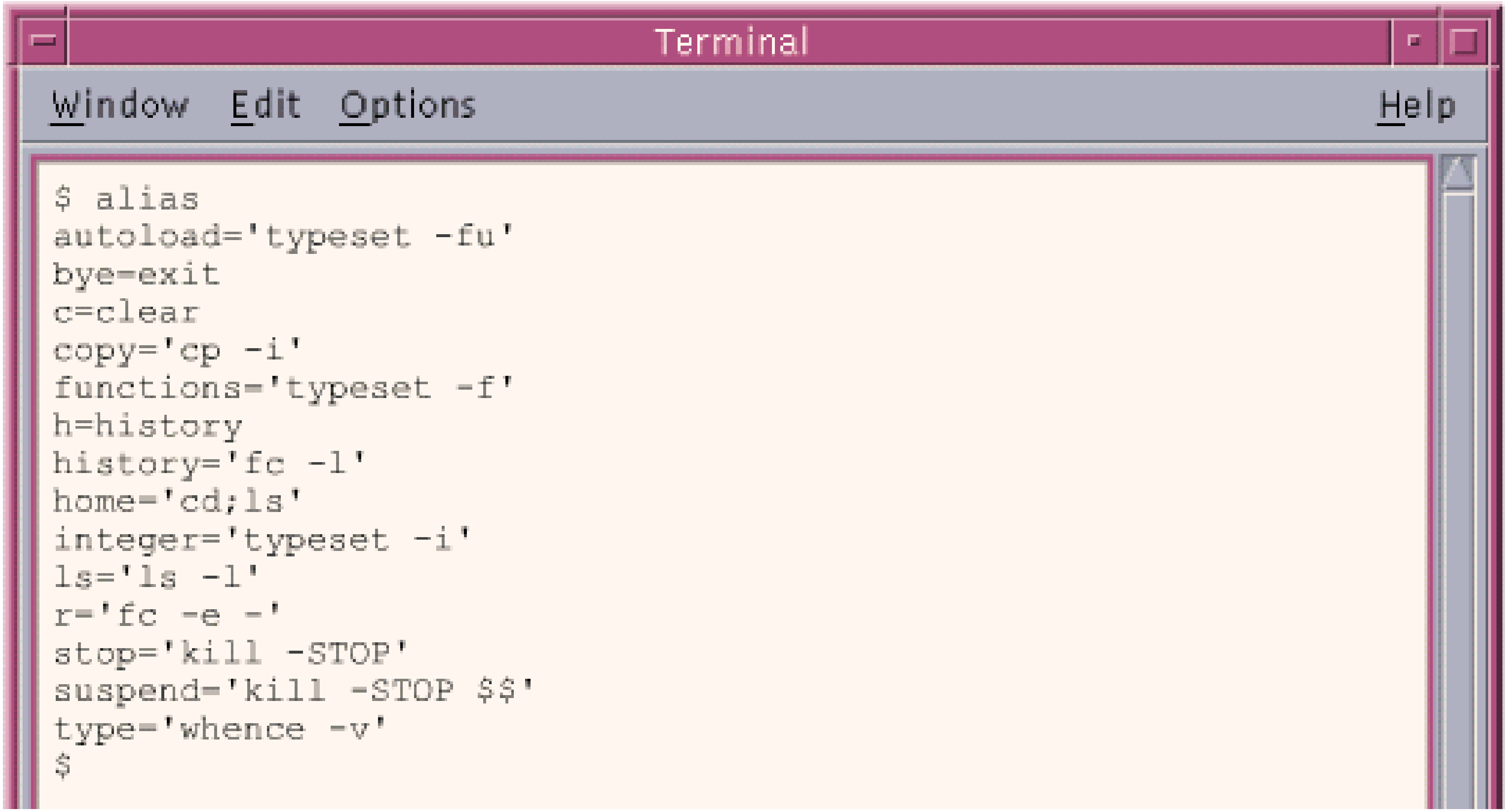
- Substitute a short command name for a longer one.
  - Type **`ll`** instead of **`ls -l`**
- Create a single command for a series of commands
- Create alternate forms of existing commands
  - Some commands can be dangerous e.g. **`rm`** Could change meaning of **`rm`** so that it always includes the **`-i`** option.

Setting Shell Aliases



```
$ alias h=history
$ alias c=clear
$ alias home='cd;ls'
$ alias ls='ls -l'
$ alias copy='cp -i'
$
```

Displaying Aliases Currently in Use



```
$ alias
autoload='typeset -fu'
bye=exit
c=clear
copy='cp -i'
functions='typeset -f'
h=history
history='fc -l'
home='cd;ls'
integer='typeset -i'
ls='ls -l'
r='fc -e -'
stop='kill -STOP'
suspend='kill -STOP $$'
type='whence -v'
$
```

Removing aliases

You can use the following command:

```
unalias aliasname
```

To temporarily bypass an alias, use a backslash:

For example, if you have used the following:

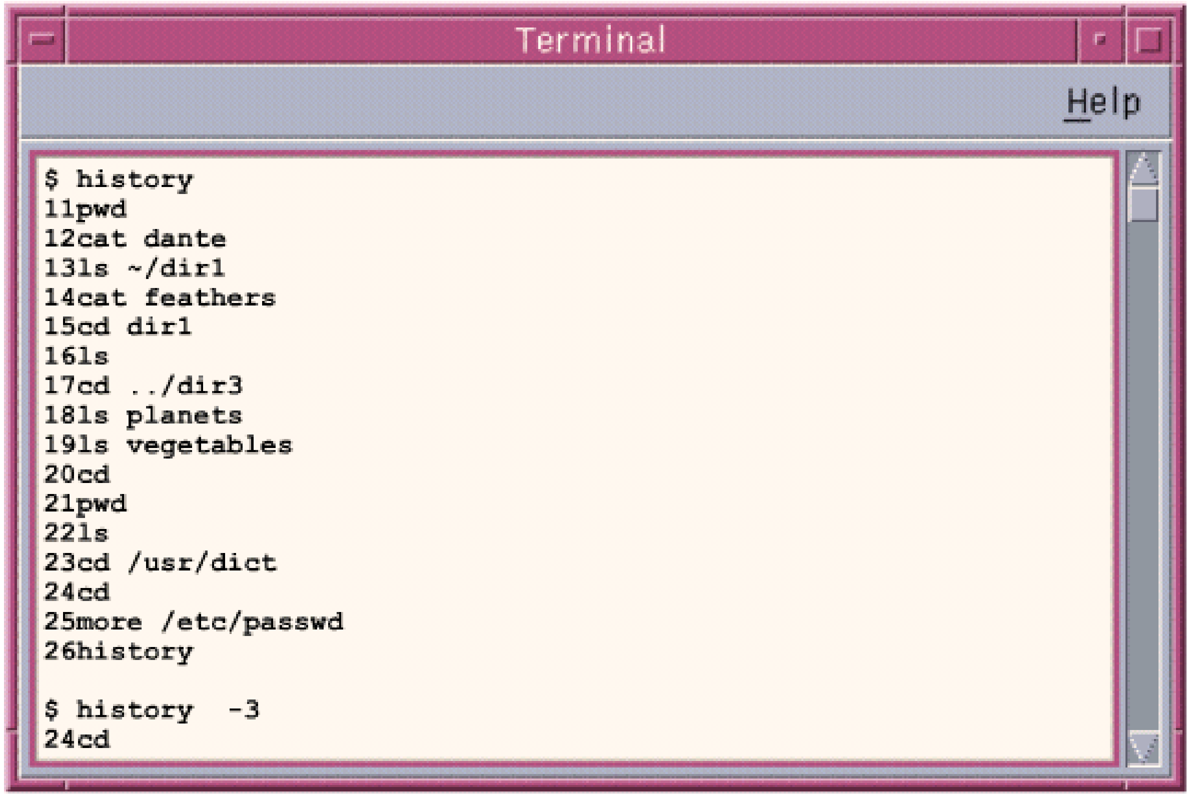
```
alias ls='ls -l'
```

Then typing:

```
\ls dir/coffees
```

will use the original form of ls

UNIX Shell History



```
$ history
11pwd
12cat dante
13ls ~/dir1
14cat feathers
15cd dir1
16ls
17cd ../dir3
18ls planets
19ls vegetables
20cd
21pwd
22ls
23cd /usr/dict
24cd
25more /etc/passwd
26history

$ history  -3
24cd
```

$ history [*options*]

By default, keeps a record of the last 128 commands but only displays the last 16 commands

Repeating Commands

Bash Shell	Korn shell	Action
! !	r	Repeats previous command
! 5	r 5	Repeats event number 5
! -2	r –2	Repeats command before last
! ls	r ls	Repeats last command beginning with ls
cd !$	No equivalent	Changes directory to the last argument of the previous command line
rm !*	No equivalent	Removes files used as arguments of previous command line

History Files

Korn shell

The history list is stored in the user's home directory in a file named

```
        ~/.sh_history.
```

By default, the Korn shell keeps a record of the last 128 commands entered.

Bash shell

The history list is stored in the user's home directory in a file named

```
        ~/.bash_history.
```

By default, the Bash shell keeps a record of the last 1000 commands entered.

Command History in Bash

The **history** command in Bash is very similar to history in the Korn shell.

The difference is Bash does not use a dash (-) before the number of last commands to display.

Repeating Commands in the Bash Shell

The simplest way to repeat commands in the Bash shell is to press the up arrow, or Ctrl+P keys, and down arrow, or Ctrl+N keys.

These keys will scroll through the previous commands from the history list.

After displaying the command to repeat, the student presses Enter.

The Bash shell offers additional features for repeating previous command lines not available in the Korn shell.

Filename and command completion

This feature allows the user to type the first few characters of a file name or command.

The user presses a specific sequence of keys that instructs the shell to complete the remainder of the file name or command.

Korn shell uses **Esc \**

Bash shell uses **Tab** key

Another way to use the completion feature is to request the shell to display a list of files that matches the entered filename.

Korn shell uses **Esc=**

Bash shell uses **Tab Tab**, entered twice

Shell Variables

All the work that is done and processes that are executed from the time of log in to the time of log out are completed within an environment.

The environment consists of shell options that are turned on such as command line editing, noclobber, and so on.

Local (shell) variables

are variables that are available for the current shell session only.

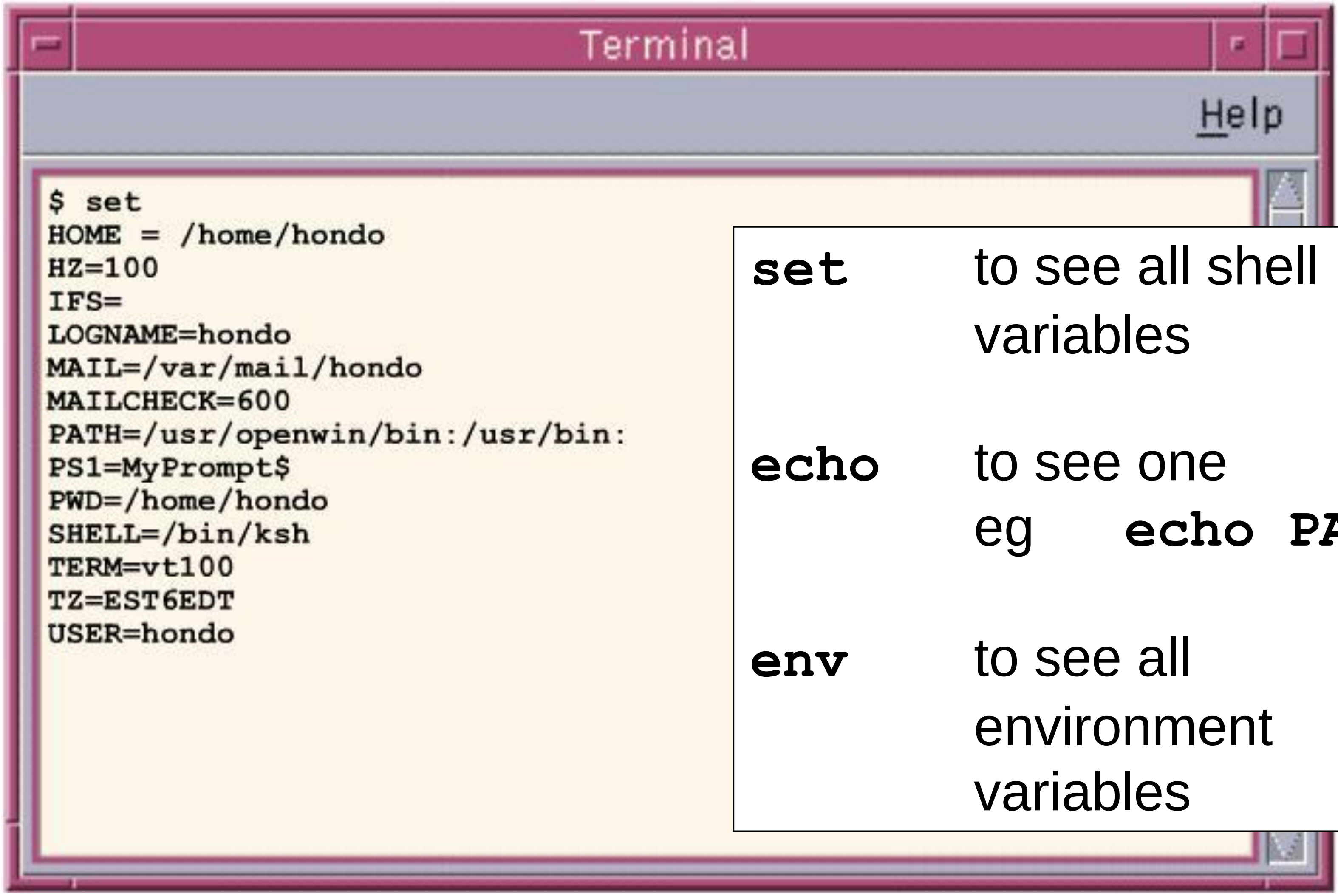
Environment (global) variables

are variables that are available to the current and all child or sub-shells that the user or the system might start.

Common Variables Set by the Shell on Login

HOME	The directory that cd changes to when no directory is given.
LOGNAME	Sets the login name of the user
PATH	A list of directories to be searched when the shell is looking for a command to execute.
SHELL	The name of the login shell
LPDEST	Name of the default printer.

Displaying Variable Values



```
$ set
HOME = /home/hondo
HZ=100
IFS=
LOGNAME=hondo
MAIL=/var/mail/hondo
MAILCHECK=600
PATH=/usr/openwin/bin:/usr/bin:
PS1=MyPrompt$
PWD=/home/hondo
SHELL=/bin/ksh
TERM=vt100
TZ=EST6EDT
USER=hondo
```

`set`	to see all shell variables
`echo`	to see one eg **`echo PATH`**
`env`	to see all environment variables

Customizing Prompts

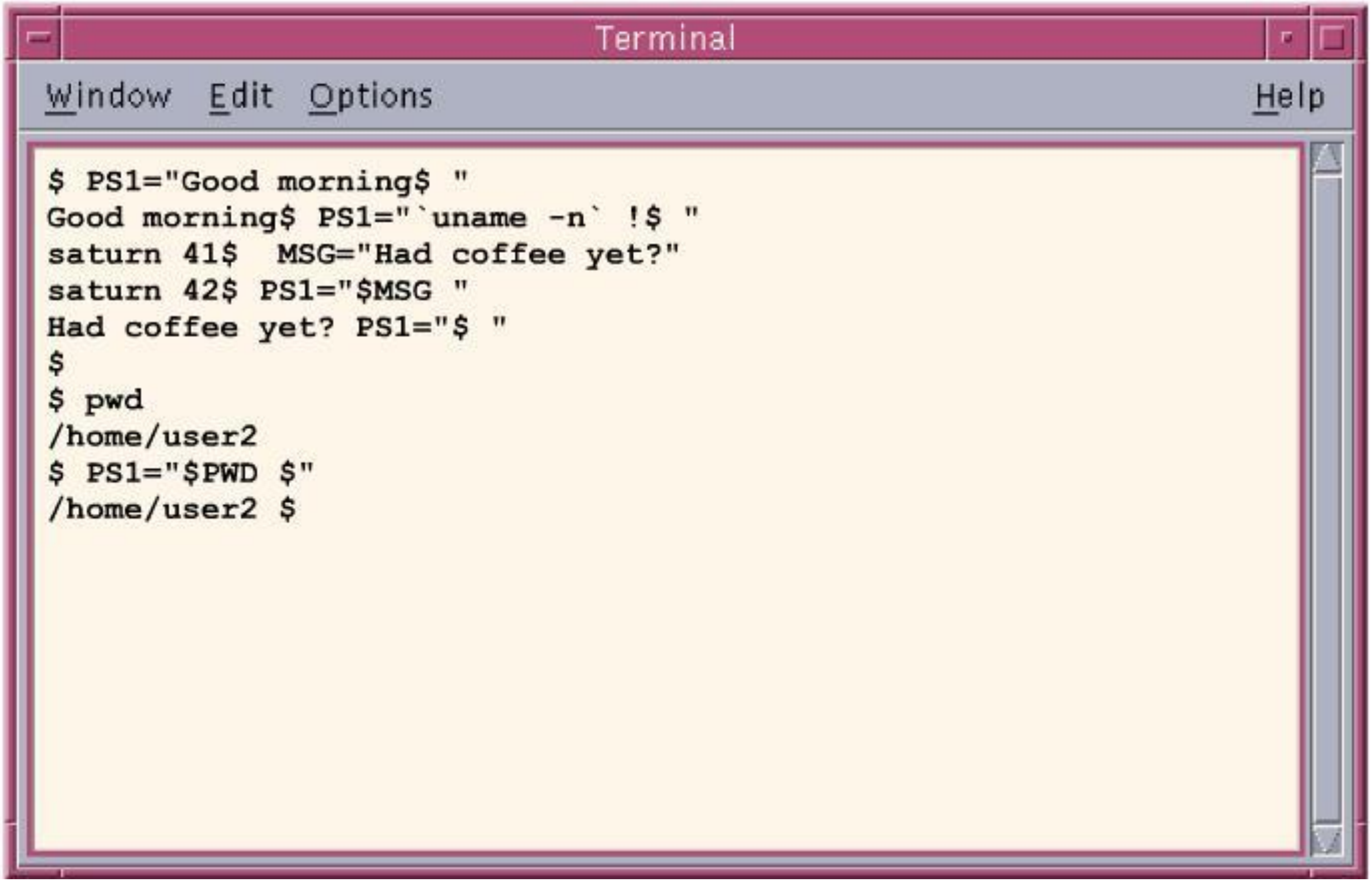
The PS1 variable is used to customise the prompt.

```
$ PS1=" `pwd` $"
/home/user02 $
```

The back quotes (`) are used to substitute the output of the command instead of interpreting it literally.

Double quotes surround the entire string.

Used if logging into several machines, you would know what machine you're on



```
$ PS1="Good morning$ "
Good morning$ PS1="`uname -n` !$ "
saturn 41$  MSG="Had coffee yet?"
saturn 42$ PS1="$MSG "
Had coffee yet? PS1="$ "
$
$ pwd
/home/user2
$ PS1="$PWD $"
/home/user2 $
```

Initialisation Files

As part of creating a user account, the system administrator chooses the login shell. The login shell determines which initialization files are read during login.

Two levels of initialization files exist.

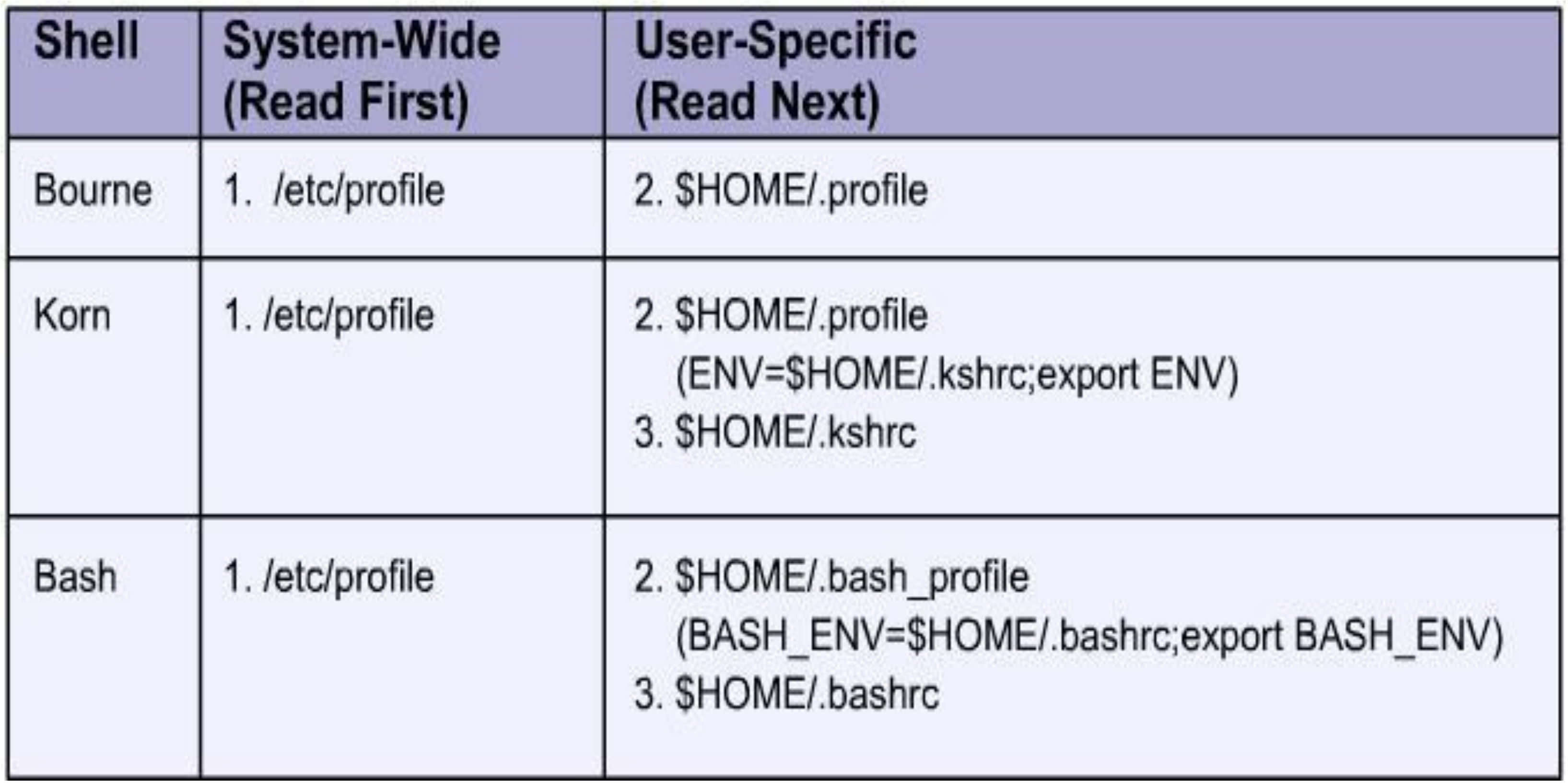
The first level is system wide.

System wide initialization files are maintained by a system administrator and reside in the /etc directory.

The second level is the user specific initialization files that reside in a user's home directory.

During the log in process the system wide initialization file is read first, and the user specific files are read next

Initialisation File Names



Shell	System-Wide (Read First)	User-Specific (Read Next)
Bourne	1. /etc/profile	2. $HOME/.profile
Korn	1. /etc/profile	2. $HOME/.profile (ENV=$HOME/.kshrc;export ENV) 3. $HOME/.kshrc
Bash	1. /etc/profile	2. $HOME/.bash_profile (BASH_ENV=$HOME/.bashrc;export BASH_ENV) 3. $HOME/.bashrc

.kshrc	Korn shell run control
.bashrc	Bash run control

Functions of the /etc/profile File

- Exports Environment Variables
- Exports PATH for Default Command Path
- Sets TERM to default Terminal Type
- Displays contents of /etc/motd File
- Sets Default File creation permissions
- Checks for mail

Functions of User Initialisation Files

The ~/.profile file usually contains one time only commands and variable definitions. Read once at login.

The .bashrc file is read each time a new shell or subshell is started.

Either might contain:

- Set Default Prompt
- Define default printer
- Set Default Permissions for new files
- Set Default Terminal type
- Set New Mail Location
- Set NoClobber
- Set Command PATH
- Define user command aliases