**Summary**

· A command file or script file is a file of shell commands that performs a given task.

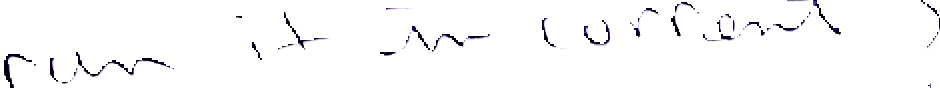
The script file must be executable in order **for** the shell to run it.

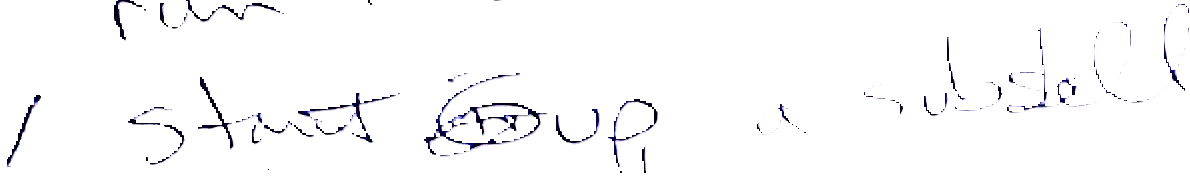
Use chmod to change the permissions of an existing file.

Use umask to control the default permissions that new files will have when they are created.

The shell uses the PATH variable to locate commands that will be executed.If a command file is not in one of the directories in the PATH, the program will not be executed.

Notes





Creating a Command File

**Lab 4**

**Objective:**

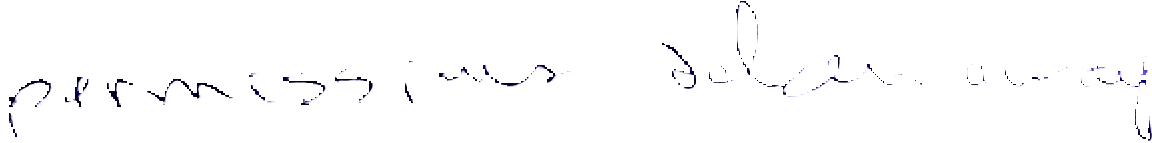
In these exercises you will create shell scripts or command files. When a file is created it is given the default file creation permissions. You will use several methods for changing the default creation mode. Finally you will execute your scripts and place them in the appropriate search path so they can be found by your shell.

**Exercises:**

1.· Create a file of zero length. Use redirection to create this file.

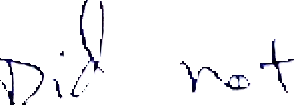
· Look at the permissions associated with this file.

·Change your umask to 0066 and create another zero length file called status.

Again look at the permissions. They should be different (-rw-- --).Explain why 

· Add the two commands date and rwho to your file.

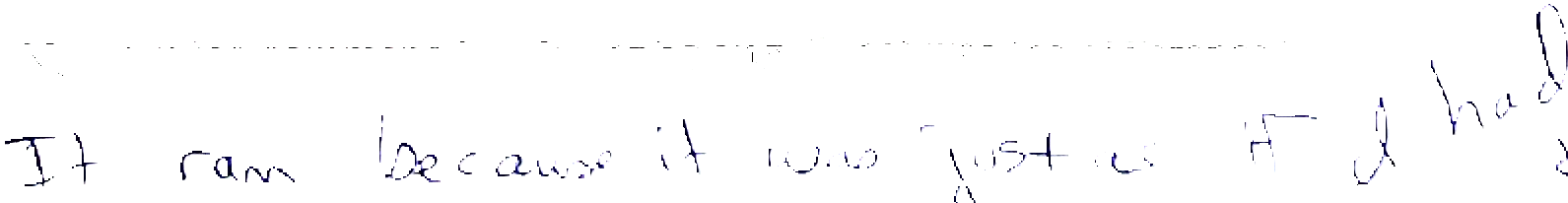
·You have created a shell script. Try and run it. Explain your message.



Now try the command bash <status. What was the difference? 





2.·Using the file status that you created in exercise #1 make the file executable using the chmod command.

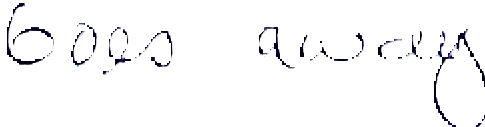
Run status

· Again look at the permissions associated with status

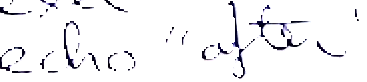
3.There are other ways to execute a shell script or command file, such as,

status 

or

exec status 



    Creating a Command File

·Try each of the above commands 

·Put each of the above commands in a separate shell script surrounded by an echo statement saying before command and after command.

Run each of the shell scripts

Explain the results

4. Create a shell script called howmany to display the current date and time and the total number of running processes. Make the file executable and run it.

5. Modify howmany to also display the names of the currently logged-on users on your network without duplicates. Then give a total number of distinct users.

6. Make howmany nonexecutable except for you and for those in your group.

7. If you create shell scripts that you use frequently, you will want to put them into one of the search directories in your PATH variable.

·Move to / etc and try and run status. What happened? 

Go to your HOME directory and make a directory called bin.

· Move status to bin.

·Now you should include $HOME/bin in your path.

PATH=$PATH:$HOME/bin

Verify it is in your path

echo $PATH

Now you can execute status regardless of where you are located within the file system.Move around and try it.

**5**

**Variables**

**Objectives**

**Determine the difference in using double quotes and** graveaccentsin **variable assignments**

·Useevalinmaking **variable assignments**

**·Make variable assignments on the command line**

Make variable assignments using the read command

Explain functionsof predefined Shell variables

Alter predefined Shell variable values

·Display and use in Shell scripts readonly Shell defined variables

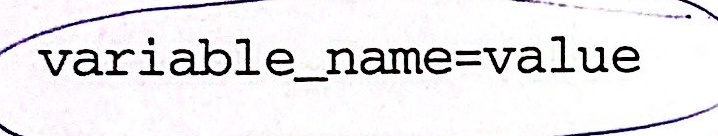


Notes

**Definition of a Shell Variable**

·A Variable is a place holder for a user-defined or system-defined **value.**

·To assign a value to a variable, use the following syntax:

The variable name is always on the leftof **the equal** sign.There **are** **NO** spaces around the equal sign.

The string should not contain the following characters:

"

;)( - < > & $

**If it is necessary** to **include the above characters in the variable** assignment,they **must be enclosed in single quotes, or preceded by a backslash.**

**To retrieve** thestored **value, you precede the variable name with a dollar** **sign“$”.**

echo $variable name

**The variable name is composed of letters, digits, and underscores, but must** **not begin with a digit. There must be no spaces in names.**

**Notes**

The shell interprets the value of a variable as a string of characters.

Data types are ignored.

The shell makes the variable substitution before the command is executed.

Q:How many Shell variables can the user define?

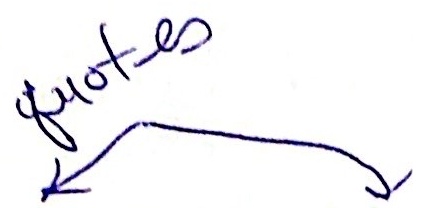
A:The number of Shell variables is essentially unlimited.

The list of special characters in the student guide have special meaning to the Shell.

Variable values may contain spaces if they are surrounded by double quotes.

**Assigning Variables Example**

**Example 1:**

$ greeting="hello world"

$ echo $greeting

hello world

$ greeting=" hello world

$ echo $greeting

hello world

$ echo "$greeting"

hello world

**Example 2:**

$ addr="Tom Jones

> Atlanta, GA 30303"

$ echo "$addr"

Tom Jones

Atlanta, GA 30303

$ echo $addr

Tom Jones Atlanta, GA 30303 

**Notes**

**Assigning Variables(read)**

**The read command is used to assign a value to a variable within a Shell** **program or at the command line. read takes its values from standard** **input.**

**Syntax:**

read variable-list

**Example:**

$ read a b c #line in Shellprogram

12345 #user input

$ echo $a #user command

1 #output display

$ echo $b #user command

2 #output display

$ echo $c #user command

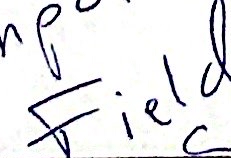
345 #output display

**Notes**

The first value is stored in the first variable. The second value is stored in the second variable.

·If there are more values than variables, the remainder of the values are stored in the last variable. If the value of IFS is tabs or spaces, then the variable values are determined accordingly. Input to read on most systems can be redirected from a file.

$ read a b < file



**Assigning Variable on the Command Line**

**·Syntax:**

variable1=a

command.file **contains:**

echo $x

**Example:**

$ x=hi

$ echo $x

hi



$ x=hey command.file

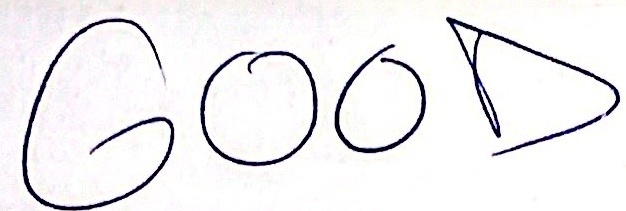
hey

$ echo $x

hi

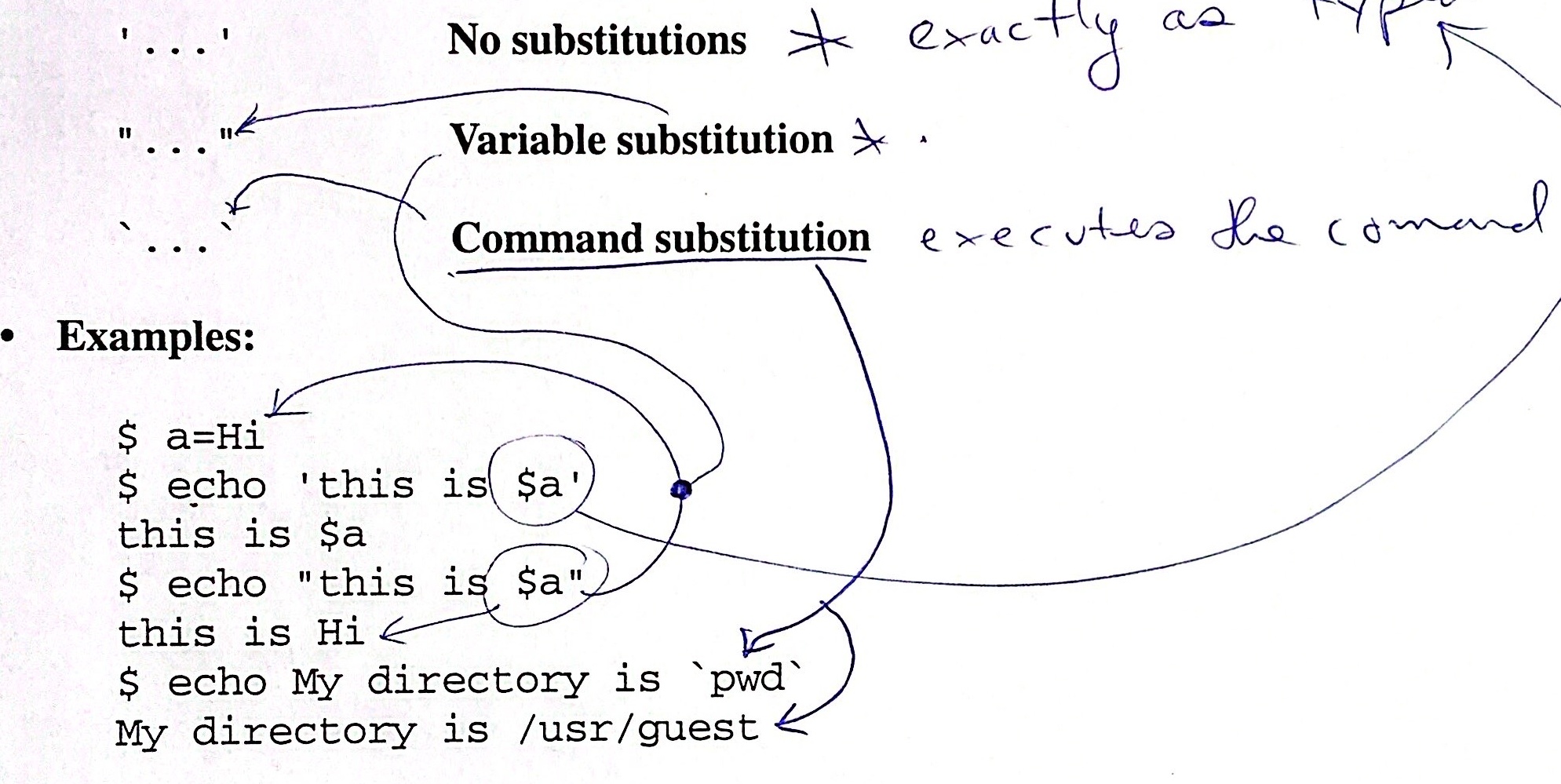
**Notes**

Sometimes it is necessary to change or assign a variable value for one command.Just preceding the command on the command line makes the variable assignments.



**The Values of Shell Variables**

**All the values of Shell variables are interpreted by the Shell as** astringof **characters.**



**Notes**

The Shell variable can be a single character or string of characters. If the string contains blanks, it must be quoted. Double quoted strings allow variable substitution. Single quotes inhibit variable substitution. This concept will be dealt with later.

Use variable assignments for frequently used strings.

$ b=/usr/bin

$ cd $b

$ pwd

The equal sign must not be preceded or followed by white space.

**Variable Assignments**

**1. Variables may be assigned in the following manner:**

$ d=bob

**The null variable can be assigned by:**

d= **or**

**2. The grave accent can be used to assign a variable with a return** stringfrom **a Shell or C program.**

$ p=`pwd

$ echo $p

**Thisiscalled *command substitution.***

3.eval **can be used to activate a command which has previously** beensettoa **variable, and it will execute it. eval is similar to evaluating a** composite **function.**

$ d=/bin/cat

$ x='$d'

$ eval $x file

**4. Variables can be assigned to contain a command. Shell variables** which **are** **set up as an abbreviation of a command cannot contain pipe symbols,redirection symbols, or the background processing symbol.**

$ s=cat

$ $s file

**Notes**

· In order to access the variable, precede the variable name with a $.

**Predefine** **d**  **Shell Variables**

MAIL **Mail delivery file**

MAILCHECK **Time Interval to check for new mail**

HOME **HOME directory**

PATH **Search path**

PS1 **Primary prompt string**

PS2 **Secondary prompt string** IFS

**Delimiter**

CDPATH **Search path for** cd **command**



**Notes**

These are variables set up by the Shell. They are originally defaulted to a standard parameter with the exception of CDPATH.

MAILCHECK is specified in seconds and the default is 600 seconds. If set to 0, the shell checks before each prompt.

·Default values are assigned to PATH, PS1, PS2, MAILCHECK,and IFS by the shell.

HOME and MAIL are set by login.

**Predefined Shell Variables (Default Values)**

**·The default values are (system dependent):**

PATH=.:/bin:/usr/bin

HOME=/usr/rice #check /etc/passwd

PS1=$

PS2=>

IFS=space,tab,newline

MAIL=/usr/spool/mail/rice

**·You can change these values to suit your own needs.**

PATH=/usr/games::/usr/bin

HOME=/usr/rice

MAIL=/usr/mail/rice



**Notes**

To remove variables, use the unset command. The unset command may not exist on all systems.

Use set to print all current values.

Most of these variables are set up in user's .profile in their HOME directory.

**·profile File**

**You can personalize your environment by setting or changing** thefollowing **variables in your.profile file**

**1.** umask

**2.** PATH



**3.** MAIL

**4.** PS1, PS2

5. TERM

**6.** HOME

7.CDPATH

**You can include:**

**flow control**

**C programs**

**Shell scripts**



**Notes**

·When you first login, the Shell reads and executes /etc/profile if it exists.It is set by the system administrator. At the exit of this program,the Shell checks your login directory for the existence of a .profile file. If it exists, the commands in it are read and executed. It sets up your local environment.

·HP-UX: Uses .vueprofile

· AIX: Uses .dtprofile

**.profile File Example**

stty erase '^H' kil1 '^U' echoe

who

PATH=.:/bin:/usr/bin:/usr/tom/bin

export PATH

PS1="`hostname`:\$PWD>"

echo "terminal?"

read x

case $x in

vt100) PS1='hi ;

ti400) PS1='ho ;

esac

export TERM

export PS1



**Notes**

Examine the contents of the .profile in your HOME directory.

$ cd

$ cat .profile

umask 077

date

stty erase "^H" kill "^?" cs8 -parenb

TERMCAP=/etc/termcap

export TERMCAP

PATH=/usr/bin:/bin:.:/usr/local:

CDPATH=:/usr/rice/Letters:/usr/rice/Proposals:

/usr/rice/Article:

export PATH

TERM=vt220

export TERM

$

Your .profile file should be writable only by you, the owner. Otherwise, other users could alter it. This could be a security hole. The contents of the .profile vary from system to system.

**Predefined Shell Variable Examples**

**·Example 1: Changing your prompt**

$ PS1="hello"

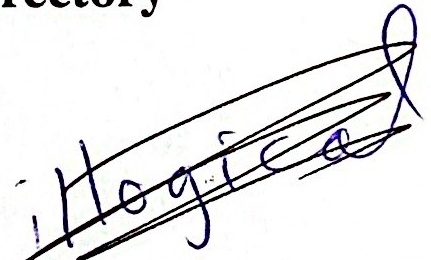
**Example 2:Change the secondary prompt**

S

$echo "today is

%Monday

%"

**Example 3: Changing HOME directory**

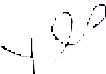
$ cd

$ pwd

$HOME=/tmp

$ cd

$ pwd

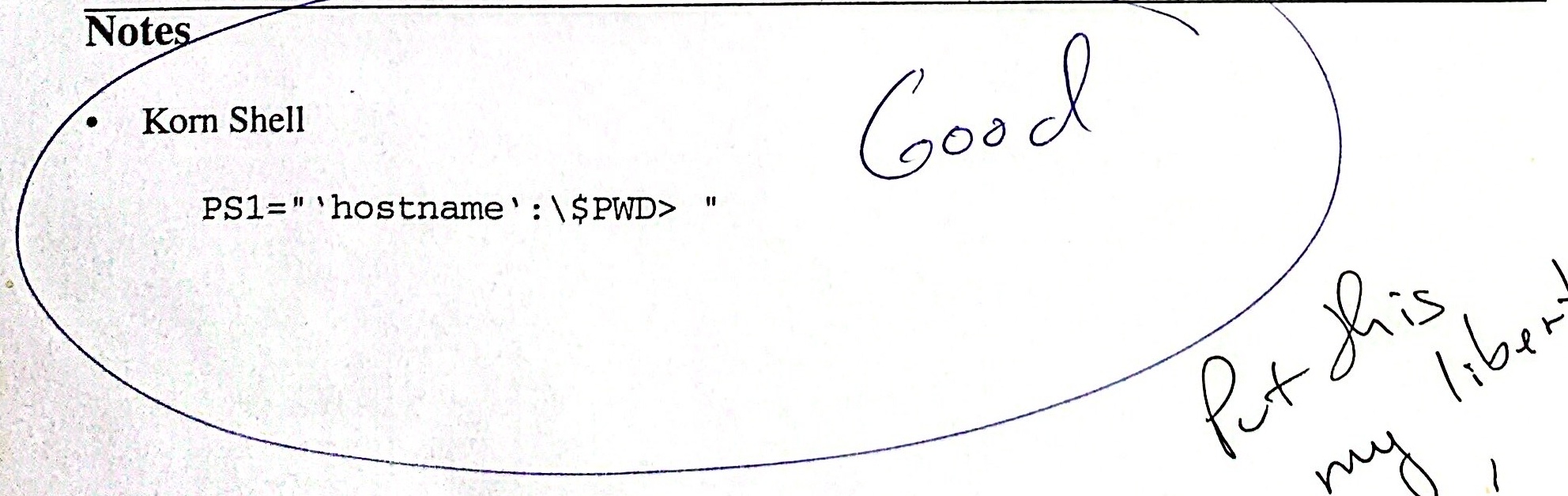


**Example 4: Your search path should include** /usr/bin **and** /bin.

/usr/sbin

$ PATH=.:/usr/bin:/bin:/usr/name/bin

$ PATH=$PATH:/etc # adding /etc to existing path



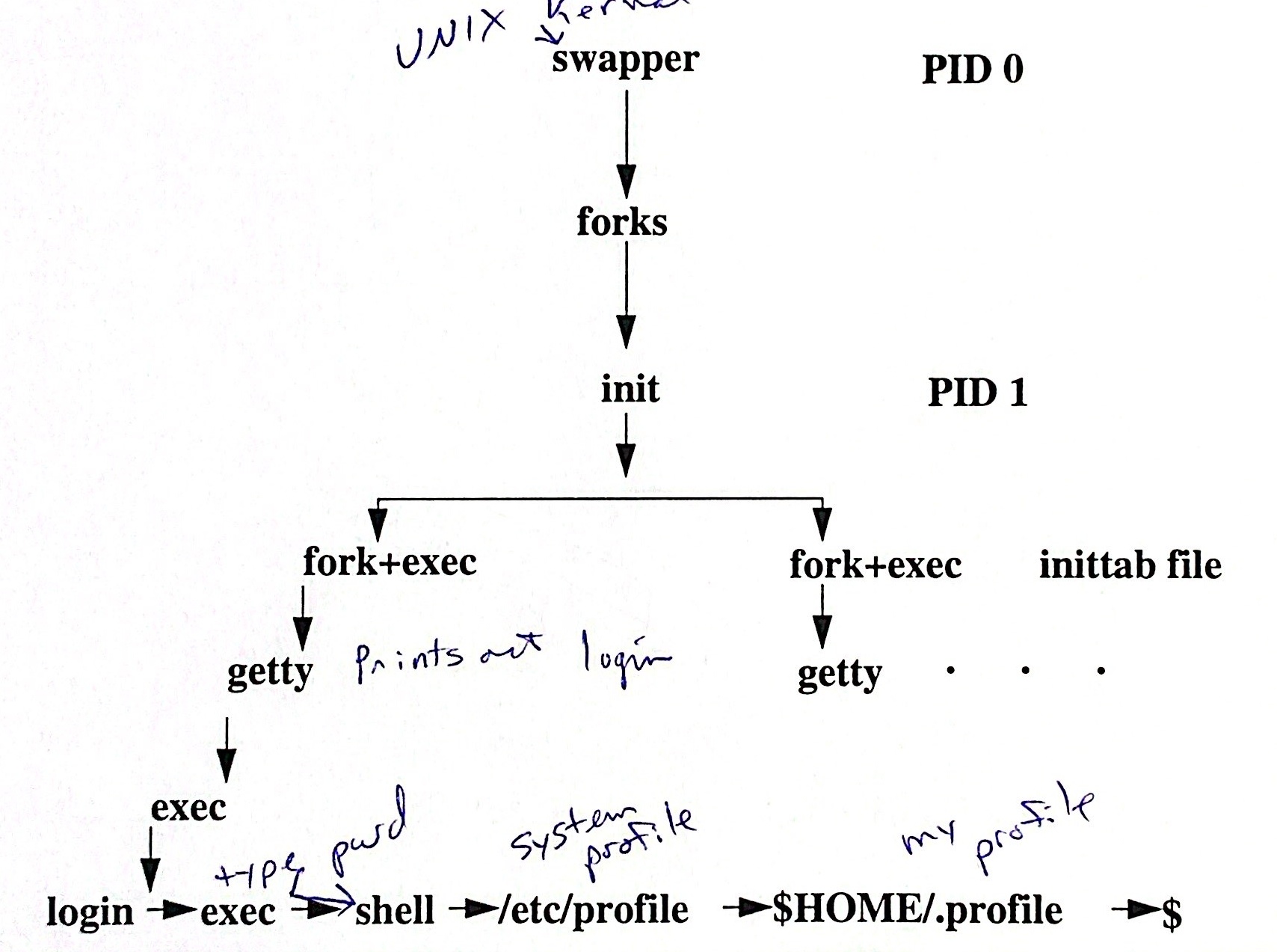


UNIX Shell Programming 5-13

© Copyright 1996-2000 GTRC, All Rights Reserved

**Initial Environment**

**·As you have seen, the system processes are created in the following order.**



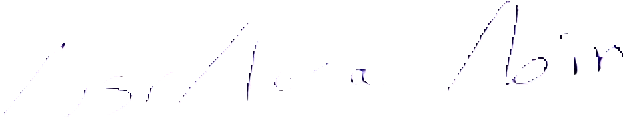
**Notes**

The Shell sets up your initial environment. The variables in your $HOME/.profile override the variables set the system file/etc/profile. 

**Example of Changing·profile**

**1.Change your** .profile **file.** 

**2. Change your** eraseandki11 **characters using the** stty **command.**

**3. Add a new directory to your search path.** 

4. **Set the** umask **command so that you alone have read and write** whenfiles **are created.**

**5. Run the** .profile **script.**

**Notes**

1. vi .profile

2. stty erase '^x'stty kill #

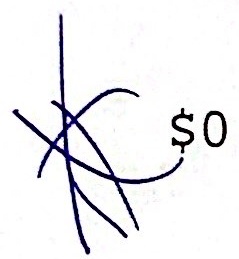
3. PATH=$PATH:/usr/local/bin

4. umask 066

5. . .profile

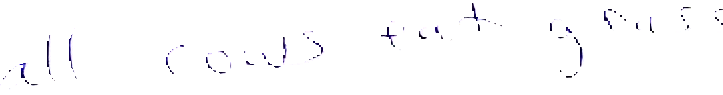
**Shell Defined Variables(readonly)**

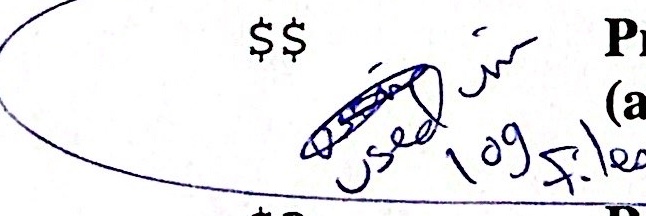
$\* “$1, $2, ...”positional parameters (arguments) passed **from** the command file

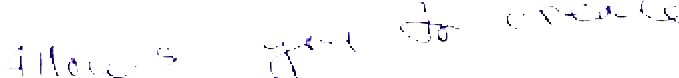
$@ “$1”“$2”... positional parameters passed from the command file

Command name

(procedure being executed)

$# Number of Shell file arguments 

Process-ID number of this Shell

(a string of 5 digits) 

**Return** value after any program dies

(exit value: zero successful, nonzero unsuccessful)

 Options passed to the Shell

 PID of last background taskinvoked

(a string of 5 digits)

**Positional** parameter n,when 0<n≤9.

**Notes**

More detail on these variables will be presented in later sections.

· To see current settings, use the echo command.

Example: $ echo $#

$ echo $$

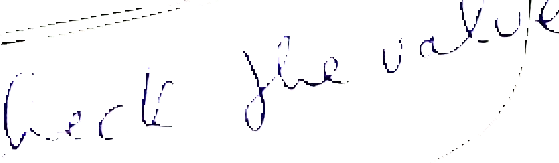
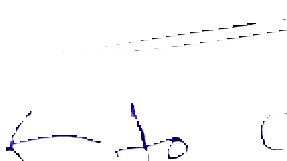
$ echo $?

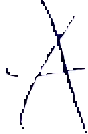
$ echo $!

These variables will be used in next chapter.

**Summary**

**Shell variables hold values. Variable names should begin with an alphabetic** **or underscore.Variable values can be of any length.**

**Variable assignments have the form** name=value.There **must not be any** **spaces around the =.** Useecho$name **to get the value of a variable.**

 treated completely **literally.**Variablevalues **can be used inside double quotes. Inside** singlequotes textis

Grave quotes,`...` **are** **used** **for** **command** substitution. The command inside the quotes is run, **and** **the** **results** **are** **substituted** back into the command line.

Predefined variables control **prompting, checking** for mail,command searching,HOME directory,etc.

The .profile file **in** **the** HOME directory is used for setting initial values of shell variables. This file is executed after /etc/profileand .bashrc.

The shell provides a number of read-only special variables, such as $$,the current process ID.

Variables can be changed temporarily by assigning values on the same line with a command.

Notes

**Lab 5**

**Objective:**

One of the most important components in programming is the ability to make assignments to variables and then retrieve the stored values. The shell provides multiple ways to make these assignments. You will now experiment in making variable assignment and retrieval.

Variable substitution is affected by enclosing the variable within grave quotes, single quotes,and double quotes. You need to discover how these symbols affect your variables.

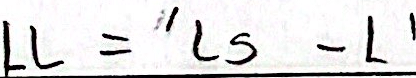
**Exercises:**

1. Some variables are assigned a default value by the shell. Use echo to determine predefined Shell variables such as MAIL, PATH, HOME, PS1,PS2, CDPATH,etc.

If a parameter is not set, then assign to it an appropriate value.

2. You use 1s -1 quite frequently. You want to find a short cut for displaying the output. Each time you type $11 you want the output displayed as if you typed 1s 1. What assignment must be made?

What is the difference if you assign 11='1s -1 'and then echo $11?

3. Use the read command to make 3 variable assignments. Enter more values than 3 variables.echo the variable values one-by-one.

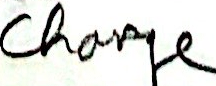
4. You can make variable assignments on the command line precedig the command. Create a shelI script that utilizes the variable values in the command file.

Example: Shell Script called process

ps -ax1| grep $p | sort +1

or variable command line assignment

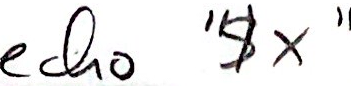
$p=p2process

5. Assign a variable to the letter x.

Using echo enclose your variable within single quotes and display the output.

Using echo enclose your variable within double quotes and display the output

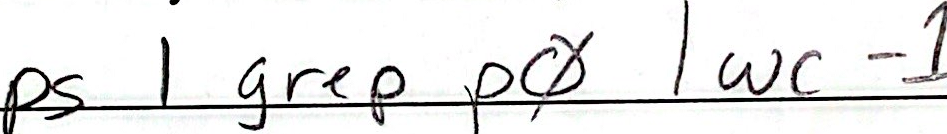
 What was the difference?

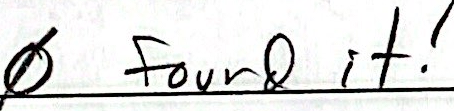
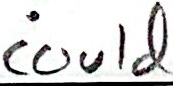
6. Alter your .bashrc file such that your system prompt always displays the current terminal.(Use tty to determine the terminal.) Cause the change to take place immediately without having to logoff then login.

7. Assign a value to the variable number such that each time you type eval $number the response is the processes associated with your tty..

8. Use grep in three different search situations. At the end of each search display $?. Explain the results. 

a. Search for a pattern that exists in a file in your current directory.

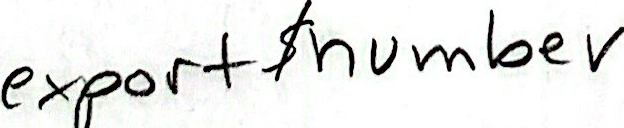
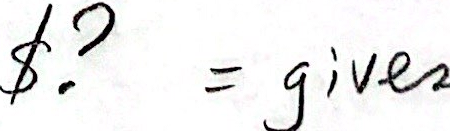


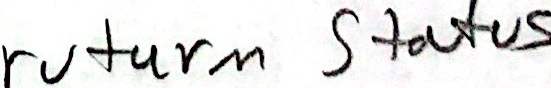
b. Search for a pattern that does not exist in any file in your current directory.   

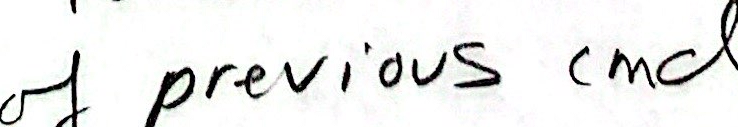
c. Search using an invalid option. 

9. The shell variables $$, $#, $0, $?, and $1 are readonly variables and you cannot change them.Create a shell script called variable that describes each of the variables and displays its value.







**6**

**Parameters**

Objectives

·Set and use positional parameters

·Change the contents of a positional parameter

·Use variable substitution to substitute default values

·Use variable substitution to generate error messages

·Use variable substitution to assign default values to undefined variables



Notes

UNIX Shell Programming 6-2

© Copyright 1996-2000 GTRC, All Rights Reserved