USN | | | | | | | | | | |

Internal Assessment Test 2 – Dec 2022

Sub:	UNIX PROGRAMMING					Sub Code:	18CS56	Branch:	CSE
Date:	05/12/2022	Duration:	90 mins	Max Marks:	50	Sem / Sec:	V SEM / A, B, C		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1 (a)	**Identify and explain the command used for pattern matching either in a file or in a given text. Mention different options for the same command.** **SOLUTION:** The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for global search for regular expression and print out). Syntax: grep [options] pattern [files] Options Description -c : This prints only a count of the lines that match a pattern -h : Display the matched lines, but do not display the filenames. -i : Ignores, case for matching -l : Displays list of a filenames only. -n : Display the matched lines and their line numbers. -v : This prints out all the lines that do not matches the pattern -e exp : Specifies expression with this option. Can use multiple times. -f file : Takes patterns from file, one per line. -E : Treats pattern as an extended regular expression (ERE) -w : Match whole word -o : Print only the matched parts of a matching line, with each such part on a separate output line.	[06]	CO2	L2
(b)	**Predict the output for the following:** **1. ls [ijk]*doc** **2. [A-Z]????*** **3. *.[!s][!h]** **4. *[!0-9]** **SOLUTION:** 1. Lists the filenames that start with i or j or k and end with doc. 2. Gives a pattern that starts with an Alphabet followed by any 4 characters or digits. 3. Displays all the files with any name other than .sh extension. 4. Displays a file that has zero or more occurrences of any character and does not end with a number.	[04]	CO2	L3
2 (a)	**Describe the use of Case statements with syntax in shell scripts.** **SOLUTION:** Shell supports case...esac statement which handles exactly this situation, and it does so more efficiently than repeated if...elif statements.	[05]	CO2	L2

	Syntax The basic syntax of the case...esac statement is to give an expression to evaluate and to execute several different statements based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used. case word in pattern1) Statement(s) to be executed if pattern1 matches ;; pattern2) Statement(s) to be executed if pattern2 matches ;; pattern3) Statement(s) to be executed if pattern3 matches ;; *) Default condition to be executed ;; esac			
(b)	**Write a Menu-driven shell script using case statements to print the following:** **1. List of Files** **2. Processes of the user** **3. Today's date** **4. User of the system** **5. Exit** **SOLUTION:** echo " ********** Menu **********\n 1. Listing the files 2. Processes of user 3. Today's Date 4. Users of the system 5. Quit" read choice case $choice in 1) ls ;; 2) ps -f ;; 3) date ;; 4) who ;; 5) exit ;; *) echo "Invalid Option" ;; esac	[05]	CO2	L3
3 (a)	**Explain three standard files supported by UNIX. Also, demonstrate the special files used for output redirection.** **SOLUTION:**	[10]	CO2	L2

Redirection is a feature in Linux such that when executing a command, you can change the standard input/output devices. The basic workflow of any Linux command is that it takes an input and give an output.

The standard input (stdin) device is the keyboard.
The standard output (stdout) device is the screen.
With redirection, the above standard input/output can be changed.
Output Redirection
The '>' symbol is used for output (STDOUT) redirection.

Example:
ls -al > listings
Here the output of command ls -al is re-directed to file "listings" instead of your screen.
Use the correct file name while redirecting command output to a file. If there is an existing file with the same name, the redirected command will delete the contents of that file and then it may be overwritten." If you do not want a file to be overwritten but want to add more content to an existing file, then you should use '>>' operator.

You can redirect standard output, to not just files, but also devices!
$ cat music.mp3 > /dev/audio

The cat command reads the file music.mp3 and sends the output to /dev/audio which is the audio device. If the sound configurations in your PC are correct, this command will play the file music.mp3

Input redirection
The '<' symbol is used for input(STDIN) redirection
Example: The mail program in Linux can help you send emails from the Terminal.

You can type the contents of the email using the standard device keyboard. But if you want to attach a File to email you can use the input re-direction operator in the following format.
This would attach the file with the email, and it would be sent to the recipient.

Error Redirection
Whenever you execute a program/command at the terminal, 3 files are always open, viz., standard input, standard output, standard error. These files are always present whenever a program is run. As explained before a file descriptor, is associated with each of these files.

File File Descriptor

Standard Input STDIN
0

Standard Output STDOUT

	1 Standard Error STDERR 2 By default, error stream is displayed on the screen. Error redirection is routing the errors to a file other than the screen. Error re-direction is one of the very popular features of Unix/Linux. Frequent UNIX users will reckon that many commands give you massive amounts of errors. For instance, while searching for files, one typically gets permission denied errors. These errors usually do not help the person searching for a particular file. While executing shell scripts, you often do NOT want error messages cluttering up the normal program output. The solution is to re-direct the error messages to a file. $ myprogram 2>errorsfile Redirection in Linux/Unix - Demystified! Above we are executing a program names myprogram. The file descriptor for standard error is 2.			

4(a)	**Demonstrate the usage of conditional statements in shell programming with the help of syntax and examples.**
	SOLUTION:
	There are total 5 conditional statements which can be used in Shell Programming.
	1. if statement
	2. if-else statement
	3. if..elif..else..fi statement (Else If ladder)
	3. if..then..else..if..then..fi..fi..(Nested if)
	4. switch statement
	
	Their description with syntax is as follows:
	
	if statement: This block will process if specified condition is true.
	Syntax:
	if [expression]
	then
	  statement
	fi
	Example:
	if grep "$1" $2
	then
	  echo "Pattern Found"
	Fi
		[06]	CO2	L2

if-else statement : If specified condition is not true in if part then else part will be execute.
Syntax:

```
if [ expression ]
then
  statement1
else
  statement2
fi
```

Example:

```
if grep "$1" $2
then
        echo "Pattern Found"
else
        echo "Pattern Not Found"
Fi
```

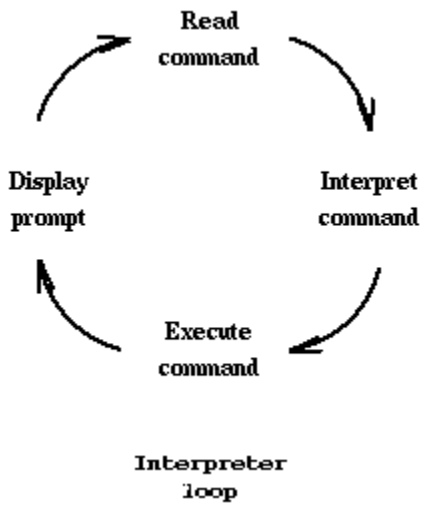
if..elif..else..fi statement (Else If ladder) :To use multiple conditions in one if-else block, then elif keyword is used in shell. If expression1 is true then it executes statement 1 and 2, and this process continues. If none of the condition is true then it processes else part.
Syntax :

```
if [ expression1 ]
then
  statement1
  statement2
  .
  .
elif [ expression2 ]
then
  statement3
  statement4
  .
  .
else
  statement5
fi
```

Nested if: Nested if-else block can be used when, one condition is satisfies then it again checks another condition. In the syntax, if expression1 is false then it processes else part, and again expression2 will be check.
Syntax:

```
if [ expression1 ]
then
  statement1
  statement2
  .
else
  if [ expression2 ]
  then
```

	statement3 . fi fi switch statement: case statement works as a switch statement if specified value match with the pattern then it will execute a block of that particular pattern. When a match is found all of the associated statements until the double semicolon (;;) is executed. A case will be terminated when the last command is executed. If there is no match, the exit status of the case is zero. Syntax: case in Pattern 1) Statement 1;; Pattern n) Statement n;; esac Example Programs Example: echo " ********** Menu **********\n 1. Listing the files 2. Processes of user 3. Today's Date 4. Users of the system 5. Quit" read choice case $choice in 1) ls ;; 2) ps -f ;; 3) date ;; 4) who ;; 5) exit ;; *) echo "Invalid Option" ;; esac			
(b)	**Consider the following conditions and write shell code using the test command and its shortcut for the same.** **1. Variable num is equal to 0** **2. check whether a given string is Null** **3. Variable num1 is greater than num2.** **4. Comparing two strings to find whether they are similar or not.** **SOLUTION:** 1. test “$num” -eq 0 ; [$num -eq 0] 2. test -z “$var” ; [-z "$var"]	[04]	CO2	L3

	3. test "$num1" -gt "$num2" ; ["$num1 -gt "$num2"] 4. test "$str1" == "$str2" ; ["$var1 == "$var2"]			
5 (a)	**Explain Shell's Interpretive life cycle.** **SOLUTION:** Shell Interpretive Cycle 1. Shell issues the prompt and waits for you (user) to enter a command. 2. After a command is entered, the shell scans command line for meta-characters and expands abbreviations to recreate simplified commands. 3. It then passes on the command line to kernel for execution. 4. The shell waits for the command to complete and normally can't do anything while the command is running. 5. After the command gets executed the prompt reappear and the whole cycle is repeated again.  	[04]	CO2	L1
(b)	**Consider the string str = " This is UNIX Programming Exam". Predict and depict the changes in the positional parameters when the following commands are executed sequentially** **1. set $str** **2. shift** **3. shift** **SOLUTION:** set $str: $1 = This; $2 = is; $3 = UNIX; $4 = Programming; $5 = Exam; shift: $1 = is; $2 = UNIX; $3 = Programming; $4 = Exam shift: $1 = UNIX; $1 = Programming; $3 = Exam	[06]	CO2	L3
6 (a)	**Explain the looping statements along with the syntax in shell scripts** **(i) For (ii) While** **SOLUTION:** The Looping Statements in Shell Programming are:	[05]	CO2	L2

	1. while Loop 2. For Loop while Loop: Here command is evaluated and based on the result loop will executed, if command raise to false then loop will be terminated Syntax : while condition ; do Statement(s) done Example: a=0 while ["$a" -lt 10] do echo $a done for Loop: The for loop operate on lists of items. It repeats a set of commands for every item in a list. Syntax : for var in 0 5 do statements done Example: for file in Demo1.sh Demo2.sh Demo3.sh Demo4.sh; do cp $file ${file}.bak echo $file Copied to $file.bak done			
(b)	**A user wants to create a backup for five files named File1.sh, File2.sh, File3.sh, File4.sh and File5.sh in the same folder. Write a shell script to perform the above problem.** **SOLUTION:** for file in Demo1.sh Demo2.sh Demo3.sh Demo4.sh; do cp $file ${file}.bak echo $file Copied to $file.bak done	[05]	CO2	L3

CO PO Mapping

Course Outcomes		Modules covered	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	Explain Unix Architecture, File system and use of Basic Commands	M1	3	2	3	2	2	-	0	0	0	0	0	0	0	0	0	0
CO2	Illustrate Shell Programming and to write Shell Scripts	M2	3	2	3	2	2	0	0	0	0	0	0	0	0	0	1	0
CO3	Categorize, compare and make use of Unix System Calls	M3M4 M5	3	2	3	2	2	0	0	0	0	0	0	0	0	2	0	0
CO4	Build an application/service over a Unix system.	M1, M2, M3, M4, M5	3	2	3	2	2	0	0	0	0	0	0	0	0	2	2	0

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium

PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Develop applications using different stacks of web and programming technologies				
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems				
PSO3	Apply software engineering methods to design, develop, test and manage software systems.				
PSO4	Develop intelligent applications for business and industry				