CMR INSTITUTE OF TECHNOLOGY

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

Internal Assessment Test 2 – Nov 2017

Sub:	**Unix Shell Programming**							Code:	**15CS35**
Date:	09/11/2017	Duration:	90 mins	Max Marks:	50	Sem:	III A,B	Branch:	**ISE**

Answer Any **FIVE FULL** Questions

		Marks	OBE CO	OBE RBT
1 (a)	Explain the three modes of vi editor with a neat diagram and list the commands in each mode	[10]	CO4	L1
2 (a)	Explain Shell's interpretive life cycle.	[04]	CO4	L1
(b)	Discuss the three standard files supported by UNIX. Also explain the two special files in UNIX.	[06]	CO4	L2
3 (a)	Explain the grep command with all its options	[06]	CO4	L3
(b)	Explain what this wild cards pattern match: i)[A-Z]????* ii)*[0-9]* iii) *[!0-9] iv)*[!s][!h]	[04]	CO4	L3
4 (a)	Write significance of following commands i) cp ?????? progs ii)ls *.[xyz]* iii) echo * iv)cp foo foo* v)ls jones[0-9][0-9][0-9]	[05]	CO4	L3
(b)	Explain the Basic regular expressions and extended regular expressions with example	[05]	CO4	L2

CMR INSTITUTE OF TECHNOLOGY

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

Internal Assessment Test 2 – Nov 2017

Sub:	**Unix Shell Programming**							Code:	**15CS35**
Date:	09/11/2017	Duration:	90 mins	Max Marks:	50	Sem:	III A,B	Branch:	**ISE**

Answer Any **FIVE FULL** Questions

		Marks	OBE CO	OBE RBT
1 (a)	Explain the three modes of vi editor with a neat diagram and list the commands in each mode	[10]	CO2	L2
2 (a)	Explain Shell's interpretive life cycle	[04]	CO2	L2
(b)	Discuss the three standard files supported by UNIX. Also explain the two special files in UNIX.	[06]	CO2	L2
3 (a)	Explain the grep command with all its options	[06]	CO2	L2
(b)	Explain what this wild cards pattern match: i) [A-Z]????* ii) *[0-9]* iii) *[!0-9] iv) *[!s][!h]	[04]	CO5	L3
4(a)	Write significance of following commands i) cp ?????? progs ii) ls *.[xyz]* iii) echo * iv)cp foo foo* v)ls jones[0-9][0-9][0-9]	[05]	CO5	L3
(b)	Explain the Basic regular expressions and extended regular expressions with example	[05]	CO5	L1

5 (a)	Explain the special parameters used by the shell	[06]	CO2	L2
(b)	Differentiate between hard links and symbolic links.	[04]	CO2	L2
6 (a)	Explain the following commands with syntax and example i) head ii) tail iii) cut iv) trap v) paste vi) tee	[06]	CO2	L2
(b)	Explain with example set and shift commands in UNIX to manipulate positional parameters	[04]	CO2	L2
7 (a)	Discuss briefly sort command with options	[06]	CO2	L2
(b)	Write a shell script to add n numbers using a while loop	[04]	CO5	L3
8 (a)	Write a shell program to create option based execution on users' choice. Options include i) list of users ii) list of processes. iii) list of files iv) current date v) print current directory vi) clear screen	[05]	CO5	L3
(b)	Define a shell script .Explain the shell features of "while" and "for" with syntax.	[05]	CO5	L1

5 (a)	Explain the special parameters used by the shell	[06]	CO2	L2
(b)	Differentiate between hard links and symbolic links	[04]	CO2	L2
6 (a)	Explain the following commands with syntax and example i) head ii) tail iii) cut iv) trap v) paste vi) tee	[06]	CO2	L2
(b)	Explain with example set and shift commands in UNIX to manipulate positional parameters	[04]	CO2	L2
7 (a)	Discuss briefly sort command with options	[06]	CO2	L2
(b)	Write a shell script to add n numbers using a while loop	[04]	CO5	L3
8 (a)	Write a shell program to create option based execution on users' choice. Options include i) list of users ii) list of processes. iii) list of files iv) current date v) print current directory vi) clear screen	[05]	CO5	L3
(b)	Define a shell script .Explain the shell features of "while" and "for" with syntax.	[05]	CO5	L1

Internal Assessment Test 2 – Nov 2017

Scheme and Solutions

Sub:	UNIX Shell Programming					Sub Code:	15CS35	Branch:	CSE/ISE
Date:	9-112017	Duration:	90 min's	Max Marks:	50	Sem / Sec:	ISE (3A,B)		OBE

Q. 1 A) Explain the three modes of vi editor with a neat diagram and list the commands in each mode
Listing three modes -1M
Diagram 3M
Each mode 2M*3=6M

Input Mode – Entering and Replacing Text It is possible to display the mode in which is user is in by typing,
:set showmode Messages like INSERT MODE, REPLACE MODE, CHANGE MODE, etc will appear in the last line. Pressing „i" changes the mode from command to input mode. To append text to the right of the cursor position, we use a, text. I and A behave same as i and a, but at line extremes I inserts text at the beginning of line. A appends text at end of line. o opens a new line below the current line

- r replacing a single character
- s replacing text with s
- R replacing text with R

Command mode: Press esc key to switch to **command mode** after you have keyed in text Some of the input mode commands are:
COMMAND i a I A o O r s S

Saving Text and Quitting – The **ex Mode** When you edit a file using vi, the original file is not distributed as such, but only a copy of it that is placed in a buffer. From time to time, you should save your work by writing the buffer contents to disk to keep the disk file current. When we talk of saving a file, we actually mean saving this buffer. You may also need to quit vi after or without saving the buffer. Some of the save and exit commands of the ex mode is:

Command	Action
:W	saves file and remains in editing mode
:x	saves and quits editing mode
:wq	saves and quits editing mode
:w	save as
:w!	save as, but overwrites existing file
:q	quits editing mode
:q!	quits editing mode by rejecting changes made
: sh	escapes to UNIX shell
:recover	recovers file from a crash

Q. 2 a) Explain Shell's interpretive life cycle
Each Step 1M * 4=4M

- You communicate with a UNIX system through a command program known as a ***shell***.
-
- The shell interprets the commands that you type on the keyboard.
- There are many different shells available for UNIX computers, and on some systems you can choose the shell in which you wish to work.
- You can use shell commands to write simple programs (scripts) to automate many tasks

Following commands explains shell's interpretive cycle

1. Shell issues the prompt and waits for you to enter a command.
2. After a command is issued, the shell scans command line for metacharacters and expands abbreviations to recreate a simplified command line.
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 any work while the command is running.

After the command execution, the prompt reappears and the shell returns to its waiting role to start the next cycle

Q. 2b) Discuss the three standard files supported by UNIX. Also explain the two special files in UNIX.
Three standard file 2M*3=6M
Two special file 2*2=4M

The shell associates three files with the terminal – two for display and one for the keyboard. These files are streams of characters which many commands see as input and output. When a user logs in, the shell makes available three files representing three streams. Each stream is associated with a default device: Standard input: The file (stream) representing input, connected to the keyboard.
Standard output: The file (stream) representing output, connected to the display.
Standard error: The file (stream) representing error messages that emanate from the command or shell, connected to the display.
The standard input can represent three input sources:

- The keyboard, the default source.
- A file using redirection with the < symbol.
- Another program using a pipeline.

The standard output can represent three possible destinations:

- The terminal, the default destination.
- A file using the redirection symbols > and >>.
- As input to another program using a pipeline.

A file is opened by referring to its pathname, but subsequent read and write operations identify the file by a unique number called a file descriptor. The kernel maintains a table of file descriptors for every process running in the system. The first three slots are generally allocated to the three standard streams as,
0 – Standard input
1 – Standard output
2 – Standard error
These descriptors are implicitly prefixed to the redirection symbols.
Examples: Assuming file2 doesn‟t exist, the following command redirects the standard output to file myOutput and the standard error to file myError.
$ ls –l file1 file2 1>myOutput 2>myError

To redirect both standard output and standard error to a single file use:
$ ls –l file1 file2 1>| myOutput 2>| myError OR
$ ls –l file1 file2 1> myOutput 2>& 1

/dev/null and /dev/tty : Two special files

/dev/null: If you would like to execute a command but don't like to see its contents on the screen, you may wish to redirect the output to a file called /dev/null. It is a special file that can accept any stream without growing in size. It's size is always zero.

/dev/tty: This file indicates one's terminal. In a shell script, if you wish to redirect the output of some select statements explicitly to the terminal. In such cases you can redirect these explicitly to /dev/tty inside the script.

Q. 3 a) Explain the grep command with all its options
Any one option 1M*6M

1)Option – i to ignore the case

grep -i "string" FILE

2) -n display line number

grep –n filename
3) -v excluding the lines
grep –v filename

4)-f take pattern from the file

5)grep –f pattern filename

6) -c count the line numbers

grep –c filename

7)-e multiple patterns
grep –e pattern1 –e pattern2 filename

Q. 3b) Explain what this wild cards pattern match:
i) [A-Z]????* ii) *[0-9]* iii) *[!0-9] iv) *[!s][!h]

1M*4=4

i) [A-Z]????* => Matches any filenames beginning with A-Z followed by any four characters and any number of characters

ii) *[0-9]* => Matches filenames beginning with any number of characters followed by a digit and again any number of characters

iii) *[!0-9] => Matches filenames beginning with any number of characters but not ending with digit

iv) *[!s][!h] => Matches filenames not ending with sh extension

Q.4 a)

Explain significance of following commands (2x5=10 M)

i. cp ?????? progs

Wild card ? matches any single character, hence the above command (cp) copies files whose names are six in length to progs directory

ii. ls *.[xyz]*

Wild card * matches any number of characters, hence the above command (ls) lists all the files having extension as either x, or y or z.

iii. ls jones[0-9][0- 9][0-9]

In the above command the character class[0-9] matches any digit between 0 to 9. Hence the above command lists all the files beginning with jones and having last three characters as any digit between 0 to 9.

iv. echo *

The above command lists all the file in the current directory.

v. cp foo foo*

The above command copies the file foo to file called foo*. Here the wild card * loses its meaning.

q. 4 b) Explain the Basic regular expressions and extended regular expressions with example
Basic Regular Expression 2.5M
Extended Regular Expression 2.5M

Basic Regular Expressions (BRE)

→ If an expression uses elaborate metacharac set, shell's wild-cards is termed as a regular expression.

→ Regular expressions take care of some common query and substitution requirements.

→ POSIX identifies regular expressions as belonging to two categories:

* basic
* Extended

→ Supports basic regular expressions (BRE) by default and extended regular expressions (ERE).

Table: The Basic Regular Expression (BRE) character se

Symbols (or) Expression	Matches
*	Zero or more occurrences of previous charact
g*	Nothing or g, gg, ggg etc.
.	A single character
.*	Nothing or any number of characters.
[Pqr]	A single character P, q or r.
[c1-c2]	A single character within the ASCII range represented by c_1 and c_2.
[1-3]	A digit between 1 and 3
[^pq]	A single character which is not a p, q and r.

CMR

Extended Regular Expressions (ERE) and egrep:

→ It is possible to match dissimilar pattern u a single expression.

* ### The + and ?

→ The ERE set includes two special characte + and ?

→ They are often used in place of the restrict the matching scope.

+ => matches one or more occurrences of the previous character.

? => matches zero or more occurrences of the previous character.

→ In both the cases, the emphasis is on the previous character.

→ b+ matches b, bb, bbb etc.

→ b? matcher either a single instance or nothing.

eg: $grep -E "[aA]gg?arwal" emp.lst

o/p.

2476 | anil aggarwal | manager

3564 | sudhir Agarwal | executive.

Q. 5 a) Explain the special parameters used by the shell
1M*6=6M

Variable	Description
$0	The filename of the current script.
$n	These variables correspond to the arguments with which a script was invoked. Here n is a positive decimal number corresponding to the position of an argument (the first argument is $1, the second argument is $2, and so on).
$#	The number of arguments supplied to a script.
$*	All the arguments are double quoted. If a script receives two arguments, $* is equivalent to $1 $2.
$@	All the arguments are individually double quoted. If a script receives two arguments, $@ is equivalent to $1 $2.
$?	The exit status of the last command executed.
$$	The process number of the current shell. For shell scripts, this is the process ID under which they are executing.
$!	The process number of the last background command.

Q.5 b) Differentiate between hard links and symbolic links.

Any differences 1M *4=4M

What are Hard Links

1. Hard Links have same inodes number.
2. ls -l command shows all the links with the link column showing the number of links.
3. Links have actual file contents
4. Removing any link, just reduces the link count but doesn't affect the other links.
5. You cannot create a Hard Link for a directory.
6. Even if the original file is removed, the link will still show you the contents of the file.

7.command to create hard is ln oldfile newfile

What are Soft Links

1. Soft Links have different inodes numbers.
2. ls -l command shows all links with second column value 1 and the link points to original file.
3. Soft Link contains the path for original file and not the contents.
4. Removing soft link doesn't affect anything but when the original file is removed, the link becomes a 'dangling' link that points to nonexistent file.
5. A Soft Link can link to a directory.

6. command to create soft link is ln –s oldfile newfile

Q.6 a) Explain the following commands with syntax and example

i) head ii) tail iii) cut iv) trap v) paste vi) tee

Each command 1M*6=6M

i)head- by default, prints the first 10 lines of each FILE to standard output. With more than one FILE, it precedes each set of output with a header identifying the file name. If no FILE is specified, or when FILE is specified as a dash ("**-**"), **head** reads from standard input.

$head myfile.txt

Display the first ten lines of myfile.txt.

$ head -15 myfile.txt

Display the first fifteen lines of myfile.txt.

$ head -n 5 myfile.txt myfile2.txt

Displays only the first 5 lines of both files.

ii) tail => tail outputs the last part, or "tail", of files

$tail myfile.txt

Outputs the last 10 lines of the file myfile.txt.

$tail myfile.txt -n 100

Outputs the last 100 lines of the file myfile.txt.

$tail -f myfile.txt

Outputs the last 10 lines of **myfile.txt**, and monitors **myfile.txt** for updates; **tail** then continues to output any new lines that are added to **myfile.txt**.

iii) cut => To divide a file into several parts (columns)

$ cat sample.txt
1;2;3;4;5;6;7;8;9

To Parse out column 2 from a semicolon (;) delimited file:

$ cat sample.txt | cut -d \; -f 2 > output.txt
$ cat output.txt
2

iv) trap => trap is a function built into the shell that responds to hardware signals and other events

$trap ‘rm $$;echo “Program Interrupted”; exit’ INT TERM
$trap ‘ ‘ 1 2 15

v)paste => The paste command merges the lines from multiple files. The paste command sequentially writes the corresponding lines from each file separated by a TAB delimiter on the unix terminal.

$cat file1
Unix
Linux
Windows

$ cat file2
Dedicated server
Virtual server

$paste file1 file2
Unix Dedicated server
Linux Virtual server
Windows

vi) tee => Reads from standard input, and writes to standard output and to files.

$ls –l file1 | tee list

Q. 6 b)Explain with example set and shift commands in UNIX to manipulate positional parameters
Set command 2M
Shift command 2M

set is a built-in function of the Bourne shell (sh), C shell (csh), and Korn shell (ksh), which is used to define and determine the values of the system environment.
Shift- This command takes one argument, a number. The positional parameters are shifted to the left by this number, *N*

$ set 100 200 300
$echo $1
100
$shift
$echo $1
200

Q. 7 a) Discuss briefly sort command with options
Each option 1M*6=6M
sort sorts the contents of a text file, line by line.

Sorting on Primary key
$ sort –t " | " –k 2 shortlist
Sorts according to 2 column

Sorting in reverse order
$ sort –t " | " –r –k 2 shortlist
Sorts in reverse order

Sorting on secondary key

$sort –t" | " –k 3,3 –k 2,2 shortlist
Promary key is third column and secondary key is second column

Numeric sort
$sort –n numfile

Removing repeated line
$sort –u file

Output is stored in sime other file
$sort –o sortedlist –k 3 shortlist

Q. 7 b) Write a shell script to add n numbers using a while loop

```
#!/bin/sh
# sum.sh : To calculate sum of n numbers
echo -n "Enter number : "
read n

# store single digit
i=1
# store number of digit
sum=0
# use while loop to caclulate the sum of all digits
while [ $i -le $n ]
  do
   sum=$(( $sum + $i )) # calculate sum of digit
       i=$(( $i + 1 ))
  done
echo  "Sum of all digit  is $sum"
```

Q. 8a) Write a shell program to create option based execution on users' choice. Options include i) list of users ii) list of processes. iii) list of files iv) current date v) print current directory vi) clear screen

Each menu 1M*5=5M

```
#!/bin/sh
# menu.sh : Uses case to offer vi-item menu

echo "            MENU\n
    1. Users of system\n 2. Processes of users\n 3. List of files\n 4. Today's date\n
    5. Current directory\n 6. Clear the screen\n 7. Quit \n Enter your option\c:"
read choice

case "$choice" in

    1) who ;;
    2) ps –f ;;
    3) ls –l ;;
    4) date ;;
    5) pwd ;;
    6) clear ;;
    7) exit ;;
    *) echo "Invalid option\n" ;;

esac
```

Q.8 b) Define a shell script .Explain the shell features of "while" and "for" with syntax

Definition 1M

A shell script is a text file that contains a sequence of commands for a UNIX-based operating system.

While loop 2M

The while loop enables you to execute a set of commands repeatedly until some condition occurs. It is usually used when you need to manipulate the value of a variable repeatedly.

```
#!/bin/sh

a=0

while [ $a -lt 10 ]
do
  echo $a
  a=`expr $a + 1`
done
```

for loop 2M

The for loop operates on lists of items. It repeats a set of commands for every item in a list.

```
#!/bin/sh

for var in 0 1 2 3 4 5 6 7 8 9
do
```

```
    echo $var
done
```