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

Internal Assessment Test 2 – December 2021

| Sub: | Unix Programming | | | | | Sub Code: | 18CS56 | Branch: | ISE | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Date: | 17/12/2021 | Duration: | 90 min's | Max Marks: | 50 | Sem/Sec: | V A, B & C | | | OBE | |
| **Answer any FIVE FULL Questions** | | | | | | | | | MARKS | CO | RBT |
| 1a) | Write a shell script to do the following:<br>a)List of files b)Display calendar of the current month c)Today's date d)Users of the system<br>using case conditional. | | | | | | | | 06 | CO2 | L3 |
| 1b) | What is the output of the following:<br>i) ls [ijk]*doc<br>ii) [A-Z]????*<br>iii) *.[!s][!h]<br>iv) *[!0-9] | | | | | | | | 04 | CO2 | L3 |
| 2a) | File current permissions are rw_ _w_r_ _.Write chmod expression required to change for the following: (i) r _ _r_ _ _ _x (ii)rwxrwx_ _x<br>(iii) r_xr_xr_x (iv) rwxrwxr_ _ using both relative and absolute methods of assigning permissions. | | | | | | | | 08 | CO2 | L2 |
| 2b) | Recall tee command with an example. | | | | | | | | 02 | CO2 | L2 |
| 3 a) | Describe i) the system call used to create regular files ii) the system call used to terminate the connection to a file from a process. | | | | | | | | 05 | CO3 | L1 |
| 3 b) | Identify the API used to establish a connection between a process and a file | | | | | | | | 05 | CO3 | L1 |
| 4 (a) | Write a shell script to print the even numbers from 1 to n using while loop. | | | | | | | | 06 | CO2 | L3 |
| 4(b) | Discuss extended regular expressions with examples. | | | | | | | | 04 | CO2 | L2 |
| 5(a) | Discuss the API used to fetch a fixed size of block of data from a file. | | | | | | | | 5 | CO3 | L1 |
| 5 (b) | Identify the system call used to write data into a file with an example. | | | | | | | | 5 | CO3 | L2 |
| 6(a) | Describe grep family utility with one example. (Any Three) | | | | | | | | 6 | CO2 | L2 |
| 6(b) | Write a shell script to illustrate usage of shell positional parameters. | | | | | | | | 4 | CO2 | L3 |

Faculty Signature CCI Signature HOD Signature

**Scheme of Evaluation**
**Internal Assessment Test 2 – Dec 2021**

| Sub: | UNIX Programming | | | | | | | Code: | 18CS56 |
|---|---|---|---|---|---|---|---|---|---|
| Date: | 17/12/2021 | Duration: | 90mins | Max Marks: | 50 | Sem: | V | Branch: | ISE |

**Note:** **Answer Any five full questions.**

| Question # | | Description | Marks Distribution | | Max Marks |
|---|---|---|---|---|---|
| 1 | a) | Write a shell script to do the following:<br>a)List of files b)Display calendar of the current month c)Today's date d)Users of the system<br>using case conditional.<br><br>**To perform a), b) ,c) & d)** 1 mark **for each case.**<br>**To begin the program and reading input** 2 Marks. | 4 * 1M<br><br>2M | 6M | 10M |
| | b) | What is the output of the following:<br>i) ls [ijk]*doc<br>ii) [A-Z]????*<br>iii) *.[!s][!h]<br>iv) *[!0-9]<br><br>**1 Mark for each output** | 4* 1M | 4M | |
| 2 | a) | File current permissions are rw_ _w_r_ _.Write chmod expression required to change for the following: (i) r _ _r_ _ _ _x (ii)rwxrwx_ _x (iii) r_xr_xr_x (iv) rwxrwxr_ _ using both relative and absolute methods of assigning permissions.<br><br>**2 marks for each sub question, 1Mark for absolute and 1 mark for relative method writing.** | 4*2M | 8M | 10M |
| | b) | Recall tee command with an example.<br><br>**Explanation 1 Mark**<br>**Example 1 Mark** | 1 + 1 | 2M | |

| | | | | | |
|---|---|---|---|---|---|
| 3 | a) | Describe i) the system call used to create regular files ii) the system call used to terminate the connection to a file from a process.<br><br>**2.5Marks for each sub question**<br>**Explanation, prototype and examples are expected** | 2.5M * 2 | 5M | 10M |
| 3 | b) | Identify the API used to establish a connection between a process and a file.<br><br>**Explanation 2M**<br>**prototype 1 M**<br>**Examples 2M** | 5M | 5M | |
| 4 | a) | Write a shell script to print the even numbers from 1 to n using while loop.<br><br>**Logic 2M**<br>**Usage of while loop 2M**<br>**Output write up 2M** | 2M<br>+<br>2M<br>+<br>2M | 6M | 10M |
| | b) | Discuss extended regular expressions with examples.<br><br>**Explanation 2 M**<br>**Examples 2M** | 2M<br>+<br>2M | **4M** | |
| 5 | a) | Discuss the API used to fetch a fixed size of block of data from a file.<br><br>**Identification 1Mark**<br>**Explanation 2 Marks**<br>**Prototype 1Mark**<br>**Example 1Mark** | 1+2+1+1 | 5M | 10M |
| 5 | b) | Identify the system call used to write data into a file with an example.<br>**Identification 1Mark**<br>**Explanation 2 Marks**<br>**Prototype 1Mark**<br>**Example 1Mark** | 1+2+1+1 | 5M | |

| | | | | | |
|---|---|---|---|---|---|
| 6 | a) | Describe grep family utility with one example. (Any Three)<br><br>**For each grep option with example 2Marks** | 3 * 2 = 6 | 6M | 10M |
| 6 | b) | Write a shell script to illustrate usage of shell positional parameters.<br><br>**For reading Positional parameters 2M**<br>**For displaying the same 2M** | 2M<br>+<br>2M | 4M | |

**Scheme Of Evaluation Internal Assessment Test 1 – NOV 2021**

| Sub: | UNIX Programming | | | | | | | Code: | 18CS56 |
|---|---|---|---|---|---|---|---|---|---|
| Date: | 17/12/2021 | Duration: | 90mins | Max Marks: | 50 | Sem: | V | Branch: | ISE |

**Note:** **Answer Any full five questions**

**Q. 1 a) Write a shell script to do the following:**
**a)List of files b)Display calendar of the current month c)Today's date d)Users of the system using case conditional.**

```
#!/bin/sh
#Shell script to illustrate CASE conditional – menu.sh
echo "\t MENU\n 1. List of files\n 2. Calendar 3. Today's Date\n 4. Users of System\n 5. Quit\n";
echo "Enter your option: \c";
read choice
case "$choice" in
1) ls -l ;;
2) cal ;;
3) date ;;
4) who ;;
5) exit ;;
*) echo "Invalid Option!"
esac
```

**Execution & Output:**

```
$ sh menu.sh
 MENU
    1. List of files
    2. Calendar
    3. Today's Date
    4. Users of System
    5. Quit
Enter your option: 3
Sun Jan 13 15:40:13 IST 2013
```

**Q. 1 b) What is the output of the following:**
**i) ls [ijk]*doc**
**ii) [A-Z]????***
**iii) *.[!s][!h]**
**iv) *[!0-9]**

i) ls [ijk]*doc outputs the filenames starting by i or j or k and ends with doc.
(ii)[A-Z]????* Displays Filenames with capital A-Z at least 4 characters.
(iii) Displays file names that do not have file names ending in in extention s or h
(iv)*[!0-9] Displays all file names that do not end in 0-9

**Q. 2 a) File current permissions are rw_ _w_r_ _.Write chmod expression required to change for the following: (i) r _ _r_ _ _ _x (ii)rwxrwx_ _x**
**(iii) r_xr_xr_x (iv) rwxrwxr_ _ using both relative and absolute methods of assigning permissions.**

i) r--r----x
chmod ug-w, g+r,o-r,o+x
chmod 441

ii) rwxrwx—x
chmod a+x, g+r, o-r
chmod 771

iii) r-xr-xr-x
chmod a+x,ug-w
chmod 555

iv) rwxrwxr—
chmod ug+x,g+r
chmod 774

**Q. 2 b) Recall tee command with an example.**

## Creating a tee

tee is an external command that handles a character stream by duplicating its input. It saves one copy in a file and writes the other to standard output. It is also a filter and hence can be placed anywhere in a pipeline. Example:-

The following command sequence uses tee to display the output of who and saves this output in a file as well.

**$who | tee users.lst**

Above command displays currently logged in users on standard output and writes a copy to users.lst

**Q. 3 a) Describe i) the system call used to create regular files ii) the system call used to terminate the connection to a file from a process.**

**i) the system call used to create regular files**

Creat system call is used to create new regular files.

```
#include<sys/types.h>
#include <unistd.h>
int creat(const char *pathname, mode_t mode);
```

•Returns: file descriptor opened for write-only if OK, -1 on error.
•The first argument pathname specifies name of the file to be created.
•The second argument mode_t, specifies permission of a file to be accessed by owner, group and others.

•The creat function can be implemented using open function as:

```
#define creat(path_name, mode)
open (pathname, O_WRONLY | O_CREAT | O_TRUNC, mode);
```

**ii) the system call used to terminate the connection to a file from a process.**

The close function disconnects a file from a process. Its prototype is:

```
#include <unistd.h>
int close (int fdesc);
```

- fdesc: is an integer file descriptor that refers to an opened file.
- The return value of close is zero if the call succeeds or -1 if it fails.
- The close function will deallocate system resources which reduces the memory requirement of a process.
- If a process terminates without closing all the files it has opened, the kernel will close files for the process.

**Q. 3 b) Identify the API used to establish a connection between a process and a file.**

 This is used to establish a connection between a process and a file i.e. it is used to open an existing file for data transfer function or else it may be also be used to create a new file.
 The returned value of the open system call is the file descriptor (row number of the file table), which contains the inode information.

The prototype of open function is:

```
#include<sys/types.h>
#include<sys/fcntl.h>
int open(const char *pathname, int accessmode, mode_t permission);
```

 If successful, open returns a nonnegative integer representing the open file descriptor.
 If unsuccessful, open returns –1.
 The first argument is the name of the file to be created or opened. This may be an absolute pathname or relative pathname.
 If the given pathname is symbolic link, the open function will resolve the symbolic link reference to a non symbolic link file to which it refers.
 The second argument is access modes, which is an integer value that specifies how actually the file should be accessed by the calling process.
 Generally the access modes are specified in . Various access modes are:

- O_RDONLY Open for reading file only.
- O_WRONLY Open for writing file only.
- O_RDWR Opens for reading & Writing file.

 There are other access modes, which are termed as access modifier flags, and one or more of the following can be specified by bitwise-ORing them with one of the above access mode flags to alter the access mechanism of the file.

- O_APPEND Append data to the end of file.
- O_CREAT Creat the file if doesn't exist.
- O_EXCL Generate an error if O_CREAT is also specified & the file already exists.
- O_TRUNC If file exists discard the file content & set the file size to zero bytes.
- O_NONBLOCK Specify subsequent read or write on the file should be non-blocking.

- O_NOCITY Specify not to use terminal device file as the calling process control terminal.

Example:
To illustrate the use of the above flags, the following example statement opens a file called /usr/SRP/usp for read and write in append mode: int fd=open("/usr/SRP/usp",O_RDWR | O_APPEND,0);

**Q. 4 a) Write a shell script to print the even numbers from 1 to n using while loop.**

```
echo "enter n value as range to calculate odd and even numbers."
read n
i=1
while [ $i -le $n ]
do
if [ `expr $i % 2` -eq 0 ]
then
echo even=$i
else
echo odd=$i
fi
i=`expr $i + 1`
done
```

**Q. 4 b) Discuss extended regular expressions with examples.**

## EXTENDED REGULAR EXPRESSION (ERE) AND egrep

**ERE** make it possible to match dissimilar patterns with a single expression.

**grep** uses ERE characters with **-E** option.

**egrep** is another alternative to use all the ERE characters without **-E** option.

This **ERE** uses some additional characters set shown in below table-

| Expression | Significance |
|---|---|
| ch+ | Matches one or more occurrences of character ch |
| ch? | Matches zero or one occurrence of character ch |
| Exp1 \| Exp2 | Matches exp1 or exp2 |
| GIF \| JPEG | Matches GIF or JPEG |
| (x1\|x2)x3 | Matches x1x3 or x2x3 |
| (hard\|soft)ware | Matches hardware or software |

**Examples:**

+ - Matches one or more occurrences of the previous character

? - Matches zero or one occurrence of the previous character.

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

2476|anil aggarwal |manager |sales |01/05/59|5000

3564|sudhir Agarwal |executive |personnel |06/07/47|7500

- **Matching Multiple Patterns( |, ( and ) )**

**$ grep -E 'sengupta|dasgupta' emp.lst**

2365|barun sengupta |director |personnel |11/05/47|7800

1265|s. n. dasgupta |manager |sales |12/09/63|5600

**$ grep -E '(sen|das)gupta' emp.lst**

2365|barun sengupta |director |personnel |11/05/47|7800

1265|s. n. dasgupta |manager |sales |12/09/63|5600

**Q. 5 a) Discuss the API used to fetch a fixed size of block of data from a file.**

The read function fetches a fixed size of block of data from a file referenced by a given file descriptor. The prototype of read function is:

```
#include <sys/types.h>
#include <unistd.h>
        size_t read(int fdesc, void *buf, size_t nbyte);
```

- If successful, read returns the no. of bytes actually read, on error it returns –1.
- The first argument is an integer, fdesc that refers to an opened file.
- The second argument, buf is the address of a buffer holding any data read.
- The third argument specifies how many bytes of data are to be read from the file.
- The size_t data type is defined in the header and should be the same as unsigned int.

**Q. 5 b) Identify the system call used to write data into a file with an example.**

✓ The write system call is used to write data into a file.

✓ The write function puts data to a file in the form of fixed block size referred by a given file descriptor.

✓ The prototype of write is

```
#include <sys/types.h>
#include<unistd.h>
size_t write(int fdesc, const void *buf, size_t size);
```

If successful, write returns the number of bytes actually written.

✓ If unsuccessful, write returns –1.

✓ The first argument, fdesc is an integer that refers to an opened file.

✓ The second argument, buf is the address of a buffer that contains data to be written.

✓ The third argument, size specifies how many bytes of data are in the buf argument.

**Example:**

**Program:**

```
#define _POSIX_SOURCE
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
#undef _POSIX_SOURCE
#include<stdio.h>
#include<string.h>

int main()
{
  char fn[]="Nirmal", text[]="He is a student of 5th sem CSE";
  int fd;

  if ((fd = creat(fn, S_IRUSR | S_IWUSR)) < 0)
    perror("creat() error");
  else
 {
    write(fd, text, strlen(text));
    close(fd);
  }
return(fd);
}
```

**Q. 6 a) Describe *grep* family utility with one example. (Any Three)**

grep: Searching for a pattern

The command uses the following syntax:

$grep options pattern filename(s)

Examples:

```
$ grep “sales” emp.lst
2233|a. k. shukla |g. m. |sales |12/12/52|6000
1006|chanchal singhvi |director |sales |03/09/38|6700
1265|s. n. dasgupta |manager |sales |12/09/63|5600
2476|anil aggarwal |manager |sales |01/05/59|5000
```

here, grep displays 4 lines containing pattern as “sales” from the file emp.lst.

**grep options:**

The below table shows all the options used by grep.

| Option | Significance |
|---|---|
| -i | Ignores case for matching. |
| -v | Doesn't display lines matching expression |
| -n | Displays line numbers along with lines |
| -c | Displays count of number of occurrences |
| -l | Displays list of filenames only. |
| -e exp | Matches multiple patterns |
| -f filename | Takes patterns from file, one per line |
| -E | Treats patterns as an ERE |
| -F | Matches multiple fixed strings |

Examples:

When you look for a name but are not sure of the case, use the -i (ignore) option.

```
$ grep -i 'agarwal' emp.lst

3564|sudhir Agarwal |executive |personnel |06/07/47|7500
```

This locates the name Agarwal using the pattern agarwal.

**Deleting Lines (-v):**

The -v option selects all the lines except those containing the pattern.

It can play an inverse role by selecting lines that does not containing the pattern.

```
$ grep -v 'director' empl.lst

2233|a. k. shukla |g. m. |sales |12/12/52|6000

5678|sumit chakrobarty |d. g. m. |marketing |19/04/43|6000

5423|n. k. gupta |chairman |admin |30/08/56|5400
```

**Displaying Line Numbers (-n):**

The -n(number) option displays the line numbers containing the pattern, along with the lines.

```
$ grep -n 'marketing' emp.lst
3: 5678|sumit chakrobarty |d. g. m. |marketing |19/04/43|6000
11: 6521|lalit chowdury |director |marketing |26/09/45|8200
14: 2345|j. b. saxena |g. m. |marketing |12/03/45|8000
15: 0110|v. k. agrawal |g. m. |marketing |31/12/40|9000
```

here, first column displays the line number in emp.lst where pattern is found

**Counting lines containing Pattern (-c):**

How many directors are there in the file emp.lst?

The -c(count) option counts the number of lines containing the pattern.

```
$ grep -c 'director' emp.lst

4
```

- ✓ Matching Multiple Patterns (-e):

  With the -e option, you can match the three agarwals by using the grep like this:

```
$ grep -e "Agarwal" -e "aggarwal" -e "agrawal" emp.lst

2476|anil aggarwal |manager |sales |01/05/59|5000
3564|sudhir Agarwal |executive |personnel |06/07/47|7500
0110|v. k. agrawal |g. m. |marketing |31/12/40|9000
```

✓ Taking patterns from a file (-f):
You can place all the patterns in a separate file, one pattern per line.
Grep uses -f option to take patterns from a file:

```
$ cat patterns.lst
director
manager
chairman

$ grep -f patterns.lst emp.lst
        9876|jai sharma |director |production |12/03/50|7000
        2365|barun sengupta |director |personnel |11/05/47|7800
        5423|n. k. gupta |chairman |admin |30/08/56|5400
        1006|chanchal singhvi |director |sales |03/09/38|6700
        1265|s. n. dasgupta |manager |sales |12/09/63|5600
        2476|anil aggarwal |manager |sales |01/05/59|5000
        6521|lalit chowdury |director |marketing |26/09/45|8200
```

**Q. 6 b) Write a shell script to illustrate usage of shell positional parameters.**

```
#!/bin/sh
#Shell Script to illustrate usage of shell positional parameters - sample.sh
echo "The Script Name is: $0"
echo "Number of arguments specified is: $#"
echo "The arguments are: $*"
echo "First Argument is: $1"
echo "Second Argument is: $2"
echo "Third Argument is: $3"
echo "Fourth Argument is: $4"
echo "The arguments are: $@"
```

Execution & Output:

```
$ sh sample.sh
welcome to hit nidasoshi [Enter]
The Script Name is: sample.sh
Number of arguments specified is: 4
The arguments are: welcome to hit nidasoshi
First Argument is: welcome
Second Argument is: to
Third Argument is: hit
Fourth Argument is: nidasoshi
The arguments are: welcome to hit nidasoshi
```