Scheme of Evaluation
Internal Assessment Test 4 – Feb 2022

Sub:	Unix Programming							**Code:**	18CS56
Date**:**	7/2/2022	Duration:	90mins	Max Marks:	50	**Sem:**	V	**Branch:**	ISE

Note: Answer Any five full questions.

Question #		**Description**	**Marks Distribution**		**Max Marks**
1		Describe the following commands with examples: od, mv, cp, rm, cd Each command syntax, usage and example	2M*5	10M	10M
2	a)	Write a short note on man documentation Syntax Explanation Example	1M 2M 2M	5M	10M
2	b)	Identify the contents of /etc/passwd and /etc/shadow file with respect to UNIX OS. Contents of /etc/passwd Contents of /etc/shadow	2.5M 2.5M	5M	
3	a)	Briefly describe the significance of the seven fields of ls –l command Syntax Seven field explanation	1M 4M	5M	10M
3	b)	Explain the concept of Escaping and Quoting with suitable example Escaping example and explanation Quoting example and explanation	2.5M 2.5M	5M	

4	a)	Describe for loop, also possible sources of argument list. Syntax Example Possible sources	1M 2M 2M	5M	10M
4	b)	Describe logical operations in shell programming with example. AND OR NOT	2M 2M 1M	5M	
5	a)	Discuss the API which helps a user to query or set flags with suitable example. fcntl( ) api syntax flags explanation example	1M 2M 2M	5M	10M
5	b)	Describe the functions which retrieve the file attributes of a given file with suitable example. stat( ) api syntax, explanation, example fstat( ) api syntax, explanation, example	2.5M 2.5M	5M	
6	a)	Discuss the system calls used to create and delete a new link to an existing file with suitable examples. link ( )api syntax, explanation unlink( ) api syntax, explanation Example	3M 3M 4M		10M

Q. 1 Describe the following commands with examples:
od, mv, cp, rm, cd

cp: copying a file

- cp command copies a file or a group of files.it creates an exact image of the file on the disk with the different name.
- The syntax requires atleast two filenames to be specified in the command line.
- When both are ordinary files, the first is copied to second file.

cp source file destination file

cp chap01 unit1

if destination file i.e unit1 does not exist, first it will be created before copying.if not it will be simply overwritten without any warning.

Copying a file to another directory

ex: assume there is a file named chap01 and it has to be copied to progs directory

cp chap01 progs

output: chap01 is now copied to directory named progs with the same name chap01.

rm : deleting files

The rm command deletes one or more files.

Ex 1: The following command deletes three files chap01, chap02, chap03.

$ rm chap01 chap02 chap03

Ex 2: to delete files named chap01 and chap02 under progs directory

$ rm progs/chap01 progs/chap02

Ex 3: to remove all file

$ rm*

---

mv: RENAMING FILES.

The mv command renames or moves files. It has two distinct functions:

h. It renames a file or directory

i. it moves a group of files to a different directory

To rename a file chap01 to man01

$ mv chap01 man01

mv replace the filename in the existing directory entry with the new name. No additional space is consumed on disk during renaming.

To rename a directory:

$ mv pts perdir

pts directory is renamed as perdir

od Command: DISPLAYING DATA IN OCTAL.

$ cat odfile
White space includes a
The ^G character rings a bell

$ od -b odfile

The -b option displays the octal values for each character.

000000 127 150 151 164 145 040 163 160 141 143 145 040 151 156 143 154
000000 165 144 145 163 040 141 040 011 012 124 150 145 040 007 040 143

Each line displays 16 bytes of data in octal , preceded by the offset in the file of the first byte in the line.

$od -bc odfile

The -b and -c option combined

Each line is now replaced with two.

The octal values are shown in first line and printable characters and escape sequences are shown in second line

000000	127	150	151	164	145	040	163	160	141	143	145	040	151
	W	h	i	t	e		s	p	a	c	e		i
	156	143	154										
	n	c	l										

Q.2 a) Write a short note on man documentation

THE MAN COMMAND

The man command knowing more about Unix commands and using Unix online manual pages

man is the system's manual viewer; it can be used to display manual pages, scroll up and down, search for occurrences of specific text, and other useful functions.

Each argument given to **man** is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section number, if provided, will direct **man** to look only in that section of the manual. The default action is to search in all of the available sections, following a pre-defined order and to show only the first page found, even if page ex- ists in several sections.

A man page is divided into a number of compulsory optional sections. Every command doesn't have all sections, but the first three(NAME,SYNOPSIS and DESCRIPTION) are seen in all man pages.

NAME presents the online introduction to the command

SYNOPSIS shows the syntax used by the command

DESCRIPTION provides a detailed information.

Options

-K,-- Search for text in all manual pages.

Section Numbers

The section numbers of the manual are listed below. While reading documentation, if you see a com- mand name followed by a number in parentheses, the number refers to one of these

Man Examples

$man man

View the manual page for the **man** command.

$man –s4 passwd

This displays the documentation for a configuration file from the section 4. Even this information is present in the section 1 it won't display section 1 information.

Q. 2 b) Identify the contents of /etc/passwd and /etc/shadow file with respect to UNIX OS.

The **/etc/passwd** file is the most important file in Linux operating system. This file stores essential information about the users on the system. This file is owned by the root user and to edit this file we must have root privileges.

This file contains one entry per line. That means it stores one user's information on one line. • **Username:** This field stores the usernames which are used while login into the system. The length of this field is between 1 and 32 characters.

- **Password**: This field store the password of the user. The **x** character indicates the password is stored in /etc/shadow file in the encrypted format. We can use the **passwd** command to update this field.
- **User ID(UID)**: User identifier is the number assigned to each user by the operating system to refer the users. The 0 UID is reserved for the root user. And 1-99 UID are reserved for other predefined accounts. And 100-999 are reserved by the system for administrative and system accounts/groups.
- **Group ID(GID)**: Group identifier is the number indicating the primary group of users. Most of the time it is the same as the UID.
- **User ID Info (GECOS)**: This is a comment field. This field contain**s** information like the user phone number, address, or full name of the user. This field is used by the finger command to get information about the user.
- **Home directory**: This field contains the absolute path of the user's home directory. By default**,** the users are created under the /home directory. If this file is empty**,** then the home directory of that user will be /
- **Login shell**: This field store the absolute path of the user shell. This shell is started when the user is log in to the system.

The `/etc/shadow` file contains one entry per line, each representing a user account. You can view the contents of the file.

```
mark:$6$.n.:17736:0:99999:7:::
[--] [----] [---] - [---] ----
|     |     |  |  |  |||+-----------> 9. Unused
|     |     |  |  |  ||+------------> 8. Expiration date
|     |     |  |  |  |+-------------> 7. Inactivity period
|     |     |  |  |  +--------------> 6. Warning period
|     |     |  |  +-----------------> 5. Maximum password age
|     |     |  +--------------------> 4. Minimum password age
|     |     +-------------------------> 3. Last password change
|     +-----------------------------------> 2. Encrypted Password
+-----------------------------------------> 1. Username
```

Q. 3 a) Briefly describe the significance of the seven fields of ls –l command

ls COMMAND

The ls command lists all files in the directory that match the name. If name is left blank, it will list all of the files in the directory.

Syntax

The syntax for the ls command is:

```
ls [options] [names]
```

Option	Description
-a	Displays all files.
-b	Displays nonprinting characters in octal.
-c	Displays files by file timestamp.
-C	Displays files in a columnar format (default)

To show long listing information about the file/directory:

```
$ls -l
total 1340
-rwxrwxr-x 1 vizion vizion    6961 2015-09-17 16:17 a.out
```

a. **Field 1:**

- 1st Character – File Type: First character specifies the type of the file. In the example above the hyphen (-) in the 1st character indicates that this is a normal file. Following are the possible file type options in the 1st character of the ls -l output.
  - Field Explanation
  - – normal file
  - d directory
  - s socket file
  - l link file
- **2nd to 9th character -- File Permissions:** Next 9 character specifies the files permission. Each 3 characters refers to the read, write, execute permissions for owner, group and other.

b. Field 2 – Number of links: Second field specifies the number of links for that file. In this example, 1 indicates only one link to this file ecos.c and 2 links to directory named fedora

c. Field 3 – Owner: Third field specifies owner of the file. In this example, ecos.c file is owned by username 'vizion'.

d. Field 4 – Group: Fourth field specifies the group of the file. In this example, the file ecos.c belongs to "vizion' group.

e. Field 5 – Size: Fifth field specifies the size of file. In this example, '1129' indicates the ecos.c file size.

f. Field 6 – Last modified date & time: Sixth field specifies the date and time of the last modification of the file.

g. **Field 7 – File name: The last field is the name of the file.**

Q. 3b) Explain the concept of Escaping and Quoting with suitable example

REMOVING THE SPECIAL MEANINGS OF WILDCARDS

ESCAPING and QUOTING

Escaping: providing a \(backslash character) before the wild card to remove or escape its special meaning.

Quoting: enclosing the wild card or even the entire pattern within quotes ('chap* '). Anyting within the quotes are left alone by the shell and not interpreted.

ESCAPING

- Placing a \ immediately before a metacharacter turns off its special meaning.
- For instance * , matches * itself. Its special meaning of matching zero or more occurrencesof character is turned off.

Ex 1:

$rm chap*

removes all the filenames starting with chap. Chap, chap01,chap02 and chap03 are removed.

$rm chap* // * metacharacter meaning is turned off

removes the filename with chap* /*name of the file itself is chap*.

QUOTING:

- This is the another way of turning off the meaning of metacharacter.
- When a command argument is enclosed within quotes, the meaning of all enclosed special characters are turned off

$rm 'chap*' // * metacharacter meaning is turned off

removes the filename with chap* /*name of the file itself is chap*.

$rm "My\ document.doc" /* To remove the file My document.doc, which has

Q. 4a) Describe for loop, also possible sources of argument list.

for : LOOPING WITH A LIST

The shells for loop differs in structure from the ones used in other programming languages.

There is no three part structure.

for *variables* in *list*
do
commands
done

The loop body also uses the keyword do and done. But the additional parameters here are variable and list. Each whitespace separated word in list is assigned to variable and commands are executed until list is executed .

Ex:

```
$for file in chap20 chap21 chap22
 do
 cp $file  {$file}.bak
 echo $file copied to $file.bak
 done
```

Output:
chap20 copied to chap20.bak
chap21 copied to chap21.bak
chap22 copied to chap22.bak

```
$echo "\$1 is $1, \$2 is $2, \$3 is $3"
```

Output: $1 is 989, $2 is 878, $3 is 779

```
$echo "The $# arguments are $*"
```

Output: The 3 arguments are 989 878 779

Q.4 b) Describe logical operations in shell programming with example.

THE LOGICAL OPERATORS && and || - CONDITIONAL EXECUTION

- The shell provides two operators that allow conditional execution.the && and ||.
- The syntax:

  cmd1 && cmd2

  cmd1 || cmd2

- Consider a file emp.lst

  $cat emp.lst

  1066| sharma | **director** |sales |03/09/66 | 7000

  1098| Kumar |**director**| production|0/08/67 | 8200

  1082|sumith| **manager**|marketing|09/09/73| 7090

- The && delimits two commands ; the command cmd2 is executed only when cmd1 succeeds.

$ grep "director" emp.lst && echo "Pattern found in file"

Output:

1066| sharma | **director** |sales |03/09/66 | 7000

1098| Kumar |**director**| production|0/08/67 | 8200

$grep " deputy manager" emp.lst || echo "Pattern not found"

Output:

Pattern not found

/* cmd1 **-deputy manager** is not found in emp.lst. Hence cmd1 fails. Therefore cmd2 "**pattern not found**" executes.

Q. 5a) Discuss the API which helps a user to query or set flags with suitable example.

❖ **fcntl**

- The fcntl function helps a user to query or set flags and the close-on-exec flag of any file descriptor.
- The prototype of fcntl is

```
#include<fcntl.h>
int fcntl(int fdesc, int cmd, …);
```

- The first argument is the file descriptor.
- The second argument cmd specifies what operation has to be performed.
- The third argument is dependent on the actual cmd value.
- The possible cmd values are defined in <fcntl.h> header.

cmd value	Use
F_GETFL	Returns the access control flags of a file descriptor fdesc
F_SETFL	Sets or clears access control flags that are specified in the third argument to fcntl. The allowed access control flags are O_APPEND & O_NONBLOCK
F_GETFD	Returns the close-on-exec flag of a file referenced by fdesc. If a return value is zero, the flag is off; otherwise on.
F_SETFD	Sets or clears the close-on-exec flag of a fdesc. The third argument to fcntl is an integer value, which is 0 to clear the flag, or 1 to set the flag
F_DUPFD	Duplicates file descriptor fdesc with another file descriptor. The third argument to fcntl is an integer value which specifies that the duplicated file descriptor must be greater than or equal to that value. The return value of fcntl is the duplicated file descriptor

- The fcntl function is useful in changing the access control flag of a file descriptor.
- For example: after a file is opened for blocking read-write access and the process needs to change the access to non-blocking and in write-append mode, it can call:

```
int cur_flags=fcntl(fdesc,F_GETFL);
int rc=fcntl(fdesc,F_SETFL,cur_flag | O_APPEND | O_NONBLOCK);
```

The following statements change the standard input og a process to a file called FOO:

```
int fdesc=open("FOO",O_RDONLY);   //open FOO for read
close(0);                                  //close standard input
if(fcntl(fdesc,F_DUPFD,0)==-1)
perror("fcntl");                    //stdin from FOO now
char buf[256];
int rc=read(0,buf,256);             //read data from FOO
```

Q. 5 b) Describe the functions which retrieve the file attributes of a given file with suitable example.

❖ **stat, fstat**

- The stat and fstat function retrieves the file attributes of a given file.
- The only difference between stat and fstat is that the first argument of a stat is a file pathname, where as the first argument of fstat is file descriptor.
- The prototypes of these functions are

```
#include<sys/stat.h>
#include<unistd.h>

int stat(const char *pathname, struct stat *statv);
int fstat(const int fdesc, struct stat *statv);
```

- The second argument to stat and fstat is the address of a struct stat-typed variable which is defined in the
  - <sys/stat.h> header.
- Its declaration is as follows:

```
struct stat
{
dev_t     st_dev;     /* file system ID */
ino_t     st_ino;     /* file inode number */
mode_t    st_mode;    /* contains file type and permission */
nlink_t   st_nlink;   /* hard link count */
uid_t     st_uid;     /* file user ID */
gid_t     st_gid;     /* file group ID */
dev_t     st_rdev;    /*contains major and minor device#*/
off_t     st_size;    /* file size in bytes */
time_t    st_atime;   /* last access time */
time_t    st_mtime;   /* last modification time */
time_t    st_ctime;   /* last status change time */
};
```

- The return value of both functions is
  - 0 if they succeed
  - -1 if they fail
  - *errno* contains an error status code
- The lstat function prototype is the same as that of stat:

```
int lstat(const char * path_name, struct stat* statv);
```

- We can determine the file type with the macros as shown.

Q. 6 a) Discuss the system calls used to create and delete a new link to an existing file with suitable examples.

link

- The link function creates a new link for the existing file.
- The prototype of the link function is

```
#include <unistd.h>
int link(const char *cur_link, const char *new_link);
```

- If successful, the link function returns 0.
- If unsuccessful, link returns –1.
- The first argument cur_link, is the pathname of existing file.
- The second argument new_link is a new pathname to be assigned to the same file.
- If this call succeeds, the hard link count will be increased by 1.
- The UNIX ln command is implemented using the link API.

unlink

- The unlink function deletes a link of an existing file.
- This function decreases the hard link count attributes of the named file, and removes the file name entry of the link from directory file.
- A file is removed from the file system when its hard link count is zero and no process has any file descriptor referencing that file.
- The prototype of unlink is

```
#include <unistd.h>
int unlink(const char * cur_link);
```

- If successful, the unlink function returns 0.
- If unsuccessful, unlink returns –1.
- The argument cur_link is a path name that references an existing file.
- ANSI C defines the rename function which does the similar unlink operation.
- The prototype of the rename function is:

The UNIX mv command can be implemented using the link and unlink APIs as shown:

```
#include <iostream.h>
#include <unistd.h>
#include<string.h>
int main ( int argc, char *argv[ ])
{
        if (argc != 3 || strcmp(argv[1],argcv[2]))
                cerr<<"usage:"<<argv[0]<<""<old_link><new_link>\
                n";
        else if(link(argv[1],argv[2]) == 0)
                return unlink(argv[1]);
        return 1;
}
```