SSN College of Engineering, Kalavakkam – 603 110
(An Autonomous Institution, Affiliated to Anna University, Chennai)
Department of Computer Science and Engineering
Continuous Assessment Test – I

Degree & Branch:	**B.E. Computer Science and Engineering**	Semester:	**3**
Subject Code & Name:	**UCS1304 UNIX AND SHELL PROGRAMMING**		
Academic Year:	**2019-2020** Batch: 2018-2022	Date:	**22-07-2019** AN
Time: 90 minutes	Answer All Questions	Maximum:	50 Marks

Part A ($6 \times 2 = 12$ marks)

1. State any four salient features of UNIX operating system. CO1, K1
   Mutiuser, mutitasking, secure, portable
2. What are the directories referred to by `/`, `~`, `.`, and `..`? CO2, K1
   / root directory, ~ home directory, . current directory, .. parent directory
3. Write a command to list the information about a directory (not the contents of the directory)? CO2, K2
   `ls -ld directory`
4. Can we remove a non-empty directory with `rmdir`? How can we remove it, with all its contents? CO2, K2
   No. We can remove `directory` with `rm -r directory`
5. After executing `chmod 567 sample`, what are the permissions on `sample`? CO2, K2
   `101 110 111 = r-x rw- rwx`
   user: read, execute; group: read, write; others: read, write, execute
6. In `vi`, what is the command to delete 2 lines starting from the current line? CO3, K2
   2 times `dd` or `2dd`

Part B ($3 \times 6 = 18$ marks)

7. The current directory has a directory `d` and a file `f`. Write commands to CO2, K2
   i) Display the absolute pathname of the current directory: `pwd`
   ii) List the contents of directory `d`: `ls d`
   iii) Link file `f` to another file `g`: `ln f g`
   iv) Copy file `f` to another file `g`: `cp f g`
   v) Move file `f` to another file `g`: `mv f g`
8. Explain the use of hard and symbolic links, with suitable examples. Under what conditions, should we use a symbolic link instead of a hard link? Your current directory has two directories `jack` and `jill`. Directory `jill` has a file `f1`. Which of the following correctly creates a link `f2` in `jack` for the target file `f1` in `jill`. CO2, K3

```
cd jack                          cd jill
ln -s ../jill/f1 f2              ln -s f1 ../jack/f2
```

Ans:

- The target (file) and a hard link both refer to the same file – same inode and same data blocks – equal status.
- A symbolic link has the pathname of the target – different inodes and different blocks. When the target is removed, the link is broken.
- We cannot create a hard link for a directory and for a file in a different filesystem.
- `ln -s ../jill/f1 f2` will store in link `jack/f2` the correct relative pathname `../jill/f1`. `ln -s f1 ../jack/f2` will store in link `jack/f2` the incorrect relative pathname `f1`. There is no `jack/f1`.

9. How are backslash, double quotes and single quotes interpreted by the shell? Explain with examples. CO4, K2

1. Backslash: (\) escape the normal meaning of characters. When escaped, metacharacters are turned into literal charaters, and certain literal characters are turned into control characters.
2. Double quotes: Metacharacters (space, newlines, *, ?, ¡, ¿) are interpreted literally. Exception: i) `$variable` ii) command substitution: `` `cmd` `` or `$(cmd)` iii) escape sequences (`\n, \t`)
3. Single quotes: No interpretation at all. The shell does not look inside a single quoted string at all.

Part C ($2 \times 10 = 20$ marks)

10. (a) Write a pattern for each line to refer to all the files in the line: (5) CO2, K3

- `file1,file2, ...,file25`: `file[1-9] file1[0-9] file2[0-5]`
- `file20,file21, ...file29`: `file2[0-9]`
- All files ending with `.c`: `*.c`
- All files starting with `f`: `f*`
- All files starting with a lowercase letter: `[a-z]*`
- All files ending with two digits: `*[0-9][0-9]`

(b) The current directory has these files: `file1, file2, file3, file4, f5,f6, f7, nfile1, nfile2, nfile441, file1a, file2a, file2b`. CO2, K3
List the files selected by each of the following patterns?

i) `file?`: `file1, file2, file3, file4`
ii) `file??`: `file1a, file2a, file2b`
iii) `file*`: `file1, file2, file3, file4, file1a, file2a, file2b`
iv) `nfile?`: `nfile1, nfile2`
v) `?file?`: `nfile1, nfile2`
vi) `?file*`: `nfile1, nfile2, nfile441`
vii) `file[a-z]`: None
viii) `file[0-9][0-9]`: None
ix) `file[0-9][a-z]`: `file1a,file2a,file2b`

OR

11. (a) With suitable commands, create the following hierarchy for directories and files. (5) CO2, K3

```
course
+--tutorial
   +--sample1
      +--file1.txt
      +--dir1
   +--sample2
      +--file2.txt
      +--dir2
```

```
mkdir -p course/tutorial/sample1/dir1
touch course/tutorial/sample1/dir1/file1.txt
mkdir -p course/tutorial/sample2/dir2
touch course/tutorial/sample1/dir2/file2.txt
```

(b) Write commands to copy file `file1.txt` using relative pathname to the current directory if the current directory is CO2, K3

i) `tutorial`: `cp sample1/file1.txt .`

ii) `dir1`: `cp ../file1.txt .`

iii) `dir2`: `../../sample1/file1.txt .`

iv) `sample2`: `cp ../sample1/file1.txt .`

(5)

12. (a) Explain how standard input, output and error can be redirected. How do we choose to overwrite or append the output to a file? How do we merge two file descriptors to a single file? (5) CO4, K2

i) `cmd < file` or `cmd 1< file`

ii) `cmd > file` or `cmd 0> file`. Use `>` to overwrite; `>>` to append.

iii) `cmd 2> file`.

iv) `cmd n> file m>&n` where `m` and `n` are the two file descriptors.

(b) Explain how pipe works. Write a pipeline of commands to find the total count of entries in the directories `/bin` and `/usr/bin` (you can use `wc -l` to count the number of lines in the standard input). (5) CO4, K2

- `cmd1 | cmd2` redirects the standard output of `cmd1` to the standard input of `cmd2`
- `cmd1` and `cmd2` run at the same time.
- `ls /bin /usr/bin | wc -l`

OR

13. Explain how the following ways of combing commands `cmd1` and `cmd2` differ, with examples: CO4, K2

i) `cmd1 | cmd2`: Output of `cmd1` is connected to the input of `cmd2`. `cmd1` and `cmd2` run concurrently.

ii) `cmd1; cmd2`: Execute `cmd1` and then, regardless of its exit status, execute `cmd2`.
iii) `(cmd1; cmd2)`: Execute `cmd1` and `cmd2` sequentially. They share the file descriptors (including `stdin` and `stdout`).
iv) `cmd1 && cmd2`: Execute `cmd1`. Only if `cmd1` succeeds, execute `cmd2`.
v) `cmd1 || cmd2`: Execute `cmd1`. Only if `cmd1` fails, execute `cmd2`.

Prepared by	Checked by	Reviewed by	Approved by
B Prabavathy, R S Milton, B Bharathi	Test Coordinator	PAC Team Member	HoD