Unix and Shell Programming Lab

1. INTRODUCTION

1.1 UNIX:

It is a multi-user operating system. Developed at AT & T Bell Industries, USA in 1969.

Ken Thomson along with Dennis Ritchie developed it from MULTICS (Multiplexed Information and Computing Service) OS.

By1980, UNIX had been completely rewritten using C language.

1.2 LINUX:

It is similar to UNIX, which is created by Linus Torualds. All UNIX commands works in Linux. Linux is a open source software. The main feature of Linux is coexisting with other OS such as windows and UNIX.

1.2.1 SALIENT FEATURES OF UNIX:

Unix is a multi-tasking operating system – can support concurrent execution of two or more active processes. Here it may be noted that an instance of a program in execution is known as a process.

Unix is a multi-user operating system-can support more than one user to login into the system simultaneously and execute programs. For this, the Unix presents a virtual computer to every user by creating simulated processors, multiple address spaces.

Unix operating system is highly portable. Compared to other OS, it is very easy to port Unix on to different hardware platforms with minimal or no modifications at all.

1.2.2 STRUCTURE OF A LINUXSYSTEM:

It consists of three parts.

a) UNIX kernel
b) Shells
c) Tools and Applications

1.3 UNIX KERNEL:

Kernel is the core of the UNIX OS. It controls all tasks, schedule all Processes, carries out all the functions of OS. Decides when one programs tops and another starts.

1.4 SHELL:

Shell is the command interpreter in the UNIX OS. It accepts command from the userand analyses and interprets them

1.5 UNIX/LINUX FILE SYSTEM BASICS

A file system is a logical collection of files on a partition or disk. A partition is a container for information and can span **an entire hard drive if desired.**

Your hard drive can have various partitions which usually contain only one file system, such as one file system housing the **/file system** or another containing the **/home file system.**

One file system per partition allows for the logical maintenance and management of differing file systems. Everything in Unix is a file, including physical devices such as DVD-ROMs, USB devices, and floppy drives.

1.6 DIRECTORY STRUCTURE

Unix uses a hierarchical file system structure, much like an upside-down tree, with root (/) at the base of the file system and all other directories spreading from there.

A Unix filesystem is a collection of files and directories that has the following properties –

- It has a root directory (/) that contains other files and directories.
- Each file or directory is uniquely identified by its name, the directory in which it resides, and a unique identifier, typically called an **inode**.
- By convention, the root directory has an inode number of 2 and the lost+found directory has an **inode number** of 3. Inode numbers 0 and

1 are not used. File inode numbers can be seen by specifying the **-i option** to ls command.

- It is self-contained. There are no dependencies between one filesystem and another.

Si. No.	Directory & De-scription
1	/ **This is the root directory which should contain only the directories needed at the top level of the filestructure**
2	**/bin** **This is where the executable files are located. These files are available to all users**
3	**/dev** **These are device drivers**
4	**/etc**
	Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups,ethernet, hosts, where to send critical messages
5	**/lib** **Contains shared library files and sometimes other kernel-related files**
6	**/boot** **Contains files for booting the system**
7	**/home** **Contains the home directory for users and other accounts**

8	**/mnt** **Used to mount other temporary file systems, such as cdrom and floppy for the CD-ROM drive and floppydiskette drive, respectively**
9	**/proc** **Contains all processes marked as a file by process number or other information that is dynamic to thesystem**
10	**/tmp** **Holds temporary files used between system boots**
11	**/usr** **Used for miscellaneous purposes, and can be used by many users. Includes administrative commands,shared files, library files, and others**
12	**/var** **Typically contains variable-length files such as log and print files and any other type of file that maycontain a variable amount of data**
13	**/sbin** **Contains binary (executable) files, usually for system administration. For example, *fdisk* and *ifconfig*** **Utilities**

II. NAVIGATING THE FILE SYSTEM

Now that you understand the basics of the file system, you can begin navigating to the files you need. The following commands are used to navigate the system

Si. No.	Command & De-

	scription
1	cat filename Displays a filename
2	cd dirname Moves you to the identified directory
3	cp file1 file2 Copies one file/directory to the specified location
4	file filename Identifies the file type (binary, text, etc)
5	find filename dir Finds a file/directory
6	head filename Shows the beginning of a file
7	less filename Browses through a file from the end or the beginning
8	ls dirname Shows the contents of the directory specified
9	mkdir dirname Creates the specified directory

10	**more filename** **Browses through a file from the beginning to the end**
11	**mv file1 file2** **Moves the location of, or renames a file/directory**
12	**pwd** **Shows the current directory the user is in**
13	**rm filename** **Removes a file**
14	**rmdir dirname** **Removes a directory**
15	**tail filename** **Shows the end of a file**
16	**touch filename** **Creates a blank file or modifies an existing file or its attributes**
17	**whereis filename** **Shows the location of a file**
18	**which filename** **Shows the location of a file if it is in your PATH**

You can use Man page Help to check complete syntax for each command mentioned here.

III. LINUX COMMANDS IN DETAIL

1. **ls** : listing files and directories in present directory where we won't be able to view details like file types, size, modified date and time, permission and links etc.
   i. ls -l
   ii. ls -a
   iii. ls -lh
   iv. ls -r
   v. ls -R
   vi. ls -lrt
   vii. ls -l directroty_name :
   viii. Ls dir_name_1/dir_name_2/
   ix. ls ../
   x. ls ../../
   xi. ll
2. **cd** : change directory
   i. cd dir_name
   ii. cd dir_name1/dir_name2/
   iii. cd -
   iv. cd ..
   v. cd ../../
   vi. cd ~
   vii. Change to a directory containing white spaces
3. **mkdir** : creating new directories
   i. mkdir dir_name
   ii. mkdir -p dir_name1/dir_name2/
4. **rm** :
   i. rm file_name
   ii. rm -rf file_name
   iii. rm *
   iv. rm -f *

Note : Before doing rm -rf * , always check the directory you are working on and check all the files inside that. **Never do rm -rf /***

   v. rm -rf dir_name
   vi. rm -rf dir_name/*
5. **cp** : copying files and directories from source to destination
   i. cp source_filename destination_filename
   ii. cp source_file destination_directory_
   iii. cp source_file_1 source_file_2 destination_directory

iv. cp -r source_directory destination_directory
v. cp -rf source_directory destination_directory
vi. cp -rf source_directory/* destination_directory
vii. cp -i source_filename destination_filename

6. **rmdir** : removing an empty directory.
i. rmdir empty_directory_name

7. **echo** :
i. echo text/string
ii. echo text/string> filename
iii. echo text/string>>filename

8. cat :
i. cat filename
ii. cat filename1 filename2
iii. cat >filename
iv. cat filename1 > filename2
v. cat filename1 >> filename2
vi. cat filename1 filename2 > filename3

9. wc :
i. wc filename
ii. wc -l filename
iii. wc -w filename
iv. wc -c filename
v. wc -m filename
vi. wc -L filename

10. tar
i. tar -xvf filename1
ii. tar -cvf filename.tar directory_name

11. bzip2 filename.tar

12. df
i. df -h

13. chmod
i. chmod 0777 filename
ii. chmod 6444 filename1

number	Permission type
0	No permission

1	Execute
2	Write
3	execute+write
5	Read
6	read+execute
7	read+write+execute

14. chown

usage : chown <user1> filename

15. tree

- display the directories and files inside a directory in a tree like structure

usage : tree directory_name

IV. UNIX AND SHELL PROGRAMMING LAB SYLLABUS

UNIX AND SHELL PROGRAMMING LABORATORY	
Course Code: CIL37	**Credits:** 0:0:1
Pre – requisites: Nil	**Contact Hours:** 14P
Course Coordinator: Shubha Vibhu Malige	

COURSE CONTENTS

List of Lab Programs

Vi/Vim editor operations

1. Introduction to Unix and Shelling Programming, Vi/Vim editor:

Consider the following content and edit using Vi/Vim editor:

"The simplest way to understand how AI and ML relate to each other is: AI is the broader concept of enabling a machine or system to sense, reason, act, or adapt like a human. ML is an application of AI that allows machines to extract knowledge from data and learn from it autonomously.

Artificial Learning is the capability of a computer system to mimic human cognitive functions such as learning and problem-solving. Through AI, a computer system uses math and logic to simulate the reasoning that people use to learn from new information and make decisions

Cyber security is primarily about people, processes, and technologies working together to encompass the full range of threat reduction, vulnerability reduction, deterrence, international engagement, incident response, resiliency, and recovery policies and activities, including computer network operations, information assurance, law enforcement, etc."

a. Execute the following operations:

i. Open the file in vi/vim editor and type the above given content and save the file by name test1.txt, continue working in the opened file.
ii. Navigate through the 10^{th} line and go to save mode and write the 10^{th} line to another file named test2.txt and continue working with test1.txt
iii. Find the Keyword AI and replace AI with Artificial Learning using interactive substitution and save the current file.
iv. Delete 5^{th} line to a buffer (a), save the current file. Now switch to the file test2.txt, move to a desired location and copy the text. Toggle between previous file and current file.

v. Navigate to the first line in test.txt file, and go to 15th character towards right and move 3 lines down and 2 characters left and replace the character with M and save the current file and exit.

vi. In the test2.txt file, move to 3 words forward in the current line and take the cursor to 2 words back, delete the word in the cursor position and navigate yourself to move to the line extremes, save and escape to the shell.

vii. In the test1.txt, join first 3 lines (Join the current line with 2 lines).

viii. Delete the 7th line in current file and exit the vi editor by saving and quitting.

b. Correct the below given c program in Vi/Vim editor using vi commands and execute the corrected c program.

Sample.c(Before Correction—with errors)	Sample.c(After Correction--Right)

#include<stio.h>
#include errno.h
int test(int * message)
{
print("Errno is %8d',errno)
exit;
	#include<stdio.h>
	#include <errno.h>
	void test(char * message)
	{
	print("Errno is %8d\n",errno);
	exit(1);
	}

Shell Scripting

2. Consider the below scenarios and execute the given shell scripts.

"*Ramaiah College is having 10 departments (Say, CS, IS, AI, ML, Cyber Security, EC, Mechanical, EEE, DS, Civil) with UG and PG programs, and in each of the program student details, course details are maintained in 10 different files (such as Student Details, Course details, Curriculum, Exam, Marks, Research Activity, NBA, Placement Activities, Library Details, Extra Activities….)*."

a. Develop a shell script for the above scenario to create 10 levels of folders for the departments and inside each level(department) of folder, create 10 files in each department for maintaining student details. Display the entire hierarchy on the standard output by using tree command.

b. Develop a shell script that accepts above created filename as argument and dis-

play its creation time and permissions of the file, on the standard output.

3. a. Develop a shell script that takes a valid directory name as an argument and recursively descend all the sub-directories, finds the maximum length of any file in that hierarchy, and store the output in a file.

b. Create a shell script to find a file with particular name, (show separate outputs for both the conditions)

- if that file exists then rename the existing file and create an empty file with that name.
- if that file does not exist then create a new empty file.
- If both the conditions done together.

c. Set a cronjob for above developed scripts, that will be execute after every 30 minutes.

d. Illustrating the shell variables in a shell script.

4. a. Build a shell script to display the system space used. If it is greater than 80%, display as Low system Space and list the files having size greater than 1GB. Set up a cron job for the above developed script to execute every Monday morning 10AM.

b. Write a shell program to count number of words, characters, white spaces, and special symbols in each text and display the output on standard output. Set a cronjob to execute above script every 3rd day of week morning 9 AM.

AWK

5. a. Develop an awkscript that accepts date argument in the form of dd-mm-yy and display it in the form month, day, and year. The script should check the validity of the argument and in the case of error, display a suitable message.

   b. Develop an awkscript to delete duplicated line from a text file. The order of the original lines must remain unchanged.

   c. Set up a cron job for the above developed scripts to execute every other day evening 4 PM.

Sed and Find

6. a. Type the below given text and save the file as 1.txt using Vi/Vim editor. Perform the below given operations.

"Python is a very popular language.
Python is easy to use. Python is easy to learn.
Python is a cross-platform language
HTML is a Markup Language
Python Programming Language
C Programming Language
Shell Programming
Perl Programming Language
Bash"

i. Replace all instances of a *Python* in a second line of 1.txt with *Perl*.

ii. Replace the last occurrence of *Programming* with *Scripting* only of a match, not other instances.

iii. Create a text file in the path /MSRIT/CSE/UG/Python.txt. Replace full path with just the filename no directory (such as Python.txt) and display it on standard output.

iv. Add string before and after the matching pattern using '\1'. In the above given text, navigate yourself to last line, you can find *Bash* keyword, Add *Learn* before *Bash* and *Programming* after *Bash* keyword.

b. Perform the following execution using find command

i. Find all the files in a current directory, whose permissions are 0777.

ii. Assign a sticky bit to all the files in a current directory.

iii. Find Directories with full permissions (777) and by using chmod command change the permissions by assigning read, write and execute permissions to

owner and only read & execute to group and others.

iv. Find last 20 days modified files, accessed files.

v. Find all the files which are modified in last 1 hour.

C Programs by using UNIX File System Calls

7. a. Develop a C program to emulate the UNIX ls –li command, which lists all the attributes of the files in a specified directory.
   b. Write a C program to remove empty files from the given directory using system calls.

8. a. Write a C program to read n characters from a file and append them back to the same file using dup2 function.

   b. Write a C program to list all files in a current directory using systemcalls
.

9. a. Create a C program to simulate the copy command in Unix (cp command)

   b. Develop a C program to simulate the ls (list) Unix command using system calls.

10. a. Understanding File Descriptors and Building a C program to create two processes P1 and P2. P1 takes a string and passes it to P2. P2 concatenates the received string with another string without using string function and sends it back to P1 for printing, send the output to standard output.

    b. Create a C program to simulate Grep Unix Command using system calls.

11. **XML Integration**

Consider the student details given below and create an XML file and save it as sample.xml.

Student Details: -

SI. No.	Student Name	USN	Department Name
1.	Alex	1RITCS001	CSE
2.	Smith	1RITDS040	DS
3.	Saliena	1RITCV051	Civil
4.	Elizabeth	1RITME011	Mechanical

Department Details: -

SI. No.	Department Name	DepartmentId	Total no of enrolled students

1.	CSE	01	170
2.	DS	02	188
3.	Civil	03	160
4.	Mechanical	04	150

a. From the above XML file, separate only student details and redirect the output to a file.
b. Replacing the tag name from name to Dept name and change globally in a shell script.

c. Read the tag value of USN and redirect the output to standard output as well as redirect to a file.

12.**Docker**

a. Docker installation and set up
b. Create a shell script to pass arguments and run in a container using Docker.
c. Creating docker file
d. Building docker image, creating a container and running the shell scripts.
e. Displaying and running docker image.

***Note: Practical sessions will be based on the contents.**

Suggested Learning Resources

Text Books:

1) Introduction to UNIX & SHELL programming, M.G. Venkatesh Murthy, Pearson Education.
2) Unix concepts and applications, Fourth Edition, Sumitabha Das, TMH.
3) Unix for programmers and users, 3rd edition, Gaham Glass & K. Ables, Pearson education.

Course Outcomes:

At the end of the course students will be able to:

1. Understand the basics of Unix concepts and commands, dockers, containers and system calls of Unix.
2. Apply the commands to incorporate changes to the file system.
3. Implement the Shell scripts based on the real time scenarios.

Course Assessment and Evaluation:

Parameter	**Marks**
CIE Test	20
Lab Record Writing + Viva+ program execution	30
Total	**50**
Final Exam will be conducted for 50 marks (SEE)	