CS 17L2 NETWORKS AND OPERATING SYSTEMS LABORATORY

Set of experiments
(subject to updations and modifications)

	CYCLE I
Expt No:1 <date>	**STUDY OF SYSTEM CALLS** **AIM:** To study the system calls – create(), open(), read(), write(), close(), sleep(), exit(), unlink(), kill(), getpid(), getppid(), getuid(), getgid(), geteuid(), getegid(), fork(), system(), pipe(), mknod(), execl(), wait(),socket() **creat() system call** <Header files, syntax and description> **open() system call** <Header files, syntax and description> **read() system call** <Header files, syntax and description> **write() system call** <Header files, syntax and description> **close() system call** <Header files, syntax and description> **sleep() system call** <Header files, syntax and description> **exit() system call** <Header files, syntax and description> **unlink() system call** <Header files, syntax and description> **kill() system call** <Header files, syntax and description> **Program No: (1):** To get the process id, parent process id, real user id, real group id, effective user id, effective group id. <Header files, syntax and description of getpid(), getppid(), getuid(), getgid(), geteuid(), getegid()> Program, Execution Steps, Output Case 1: Same real user and effective user

	Case 2: Different real user and effective user **Program No: (2):** Familiarization of fork() system call <Header files, syntax and description> Program, Execution Steps, Output **Program No: (3):** Familiarization of system() system call <Header files, syntax and description> Program, Execution Steps, Output **Program No: (4):** Familiarization of pipe() system call <Header files, syntax and description> Program, Execution Steps, Output **Program No: (5):** To create a FIFO (named pipe) <Header files, syntax and description> Program, Execution Steps, Output **Program No: (6):** Familiarization of execl() system call <Header files, syntax and description> Program, Execution Steps, Output **Program No: (7):** Familiarization of wait() system call <Header files, syntax and description> Program, Execution Steps, Output Case 1: Executing without wait system call Case 2: Executing with wait system call **Program No: (8):** Familiarization of socket() system call <Header files, syntax and description> Program, Execution Steps, Output
Expt No:2 <date>	**<u>9. FILE TYPE</u>** **AIM:** To print the type of a given file <Header files, syntax and description> Program, Execution steps, Output for regular file, character special file, block special file, directory, fifo, symbolic link, socket
Expt No:3	**<u>10. FILE ATTRIBUTES</u>**

<date>	**AIM:** To accept a file name from the keyboard and display the following attributes of the file a) Access permissions b) i-node number c)Time of last file access d) Time of last file modification e) File Size <Header files, syntax and description> Program, Execution steps, Output
Expt No:4 **<date>**	**<u>11. STUDY OF SIGNALS</u>** **AIM:** To study the signals – SIGINT, SIGCHLD, SIGQUIT,SIGALRM <Header files, syntax and description> Program, Execution steps **Output** Case 1: While pressing Ctrl + C Case 2: While pressing Ctrl + / Case 3: Without pressing Ctrl + C & Ctrl + /
	<u>CYCLE II</u>
Expt No:5 **<date>**	**<u>12.INTERPROCESS COMMUNICATION USING PIPES</u>** **AIM:** To implement interprocess communication using two pipes <Header files, syntax and description> Program, Execution steps, Diagram,Output
Expt No:6 **<date>**	**<u>13. INTERPROCESS COMMUNICATION USING FIFO</u>** **AIM:** To implement interprocess communication using fifo <Header files, syntax and description> Program, Execution steps, Output
Expt No:7 **<date>**	**<u>14. FAMILIARISATION OF POSIX THREAD FUNCTIONS</u>** **AIM:** To study the basic posix thread functions – pthread_create, pthread_join, pthread_self, pthread_detach, pthread_exit <Header files, syntax and description> Program, Execution steps, Output
Expt No:8 **<date>**	**<u>15. FAMILIARISATION OF POSIX SEMAPHORE SYSTEM CALLS</u>** Implement Wait (P) and Signal (V) operations on a Semaphore using system calls

	<Header files, syntax and description> Program, Execution steps, Output
Expt No:9 **<date>**	**16. INTERPROCESS COMMUNICATION USING POSIX SHARED MEMORY** **AIM:** Write a program to create an integer variable using shared memory concept and increment the variable simultaneously by two processes. Use semaphores to avoid race conditions <Header files, syntax and description> Program, Execution steps, Output
Expt No: 10 **<date>**	**17. PRODUCER-CONSUMER PROBLEM** **AIM:** A Program to implement the Producer-Consumer problem using Semaphores and shared memory <Header files, syntax, algorithm and description> Program, Execution steps, Output
Expt No:11 **<date>**	**18. INTERPROCESS COMMUNICATION USING POSIX MESSAGE QUEUES** **AIM:** To implement interprocess communication using Message Queues **.** <Header files, syntax and description> Program, Execution steps, Output
Expt No:12 **<date>**	**19. DINING PHILOSOPHER'S PROBLEM** **AIM:** A Program to implement Dining Philosophers problem using posix threads <Header files, syntax, algorithm and description> Program, Execution steps, Output
Expt No:13 **<date>**	**20. READERS WRITERS PROBLEM** **AIM:** A Program to implement Readers-Writers problem using Semaphore. <Header files, syntax, algorithm and description> Program, Execution steps, Output
Expt No:14 **<date>**	**21. MONITORS** **AIM:** A Program to implement monitors <Header files, syntax, and description> Program, Execution steps, Output
	CYCLE III

Expt No:15 **<date>**	**22. TCP CLIENT SERVER** Design a TCP Client and server application <Header files, syntax,and description> Program, Execution steps, Output
Expt No:16 **<date>**	**23. UDP CLIENT SERVER** Design UDP Client server application <Header files, syntax, and description> Program, Execution steps, Output
Expt No:17 **<date>**	**24. PARALLEL VIRTUAL MACHINE (PVM)** Implement an algorithm using PVM <Header files, syntax, algorithm and description> Program, Execution steps, Output
	CYCLE IV
Expt No:18 **<date>**	**25. SHELL PROGRAMMING - I** (21/7/16) PROGRAM I :Four input files are given. These files represent the grades scored by a set of students in S1, and S2 semester exams and associated files to assist the task. You are required to compute the credits earned by each student, determine the number of subjects failed, calculate the SGPAs, and CGPAs. (Additional tasks later) You are not allowed to modify any of the input files, but may create additional files and alter them if necessary using shell programs. No direct data file editing is permitted. Each row of the grade file represents one student, blank lines in between if any represent an absent student. Each row(student) should be numbered in the output. Input/Data files: s1.txt s2.txt s1s2.credits.txt // as extracted from a C file ktu.gp.txt
Expt No:19 **<date>**	**26. SHELL PROGRAMMING - II** PROGRAM II

Expt No:20 **<date>**	**27. SHELL PROGRAMMING - III** PROGRAM III
Expt No:21 **<date>**	**28. SHELL PROGRAMMING - IV** PROGRAM IV
Expt No:22 **<date>**	**29. SHELL PROGRAMMING - V** PROGRAM V
Expt No:23 **<date>**	**30. LINUX INTERNALS** Remove and insert a module in Linux OS kernel
Expt No:24 **<date>**	**31. SETTING OF A LINUX LAN** Setting up of a LAN - assigning static IPs, Populating the routing table with static routing and configuring router
Expt No:25 **<date>**	**32. ADHOC BASED WIRELESS COMMUNICATION** Design an adhoc based wireless communication network
Expt No:26 **<date>**	**33. AP BASED WIRELESS COMMUNICATION** Design an infrastructure based wireless communication network