13CS356 USP LAB PROGRAMS

Pgm No	Title of the program/ Problem Statement
1	Write a program to display a file in reverse using lseek system call. If the file contains "abcdefgh" then display "hgfedcba"
	Write a program (use signal system call) a. which calls a signal handler on SIGINT signal and then reset the default action of the SIGINT signal b. which ignores SIGINT signal and then reset the default action of SIGINT signal Rewrite any one of the above programs using sigaction system call
2	Simulate stat command. Display the following : File type, File permissions, File size, inode number, time stamp, Link count Check for different file types Create links and check the output What is the significance of number of links for a directory?
	Experiment on exec as to what happens to signal pending. Is it pending in the called process? Is it delivered to the called process, if signal blocking is removed? Create the caller (1) block a signal (2) send the signal (3) exec the called process Create the called (1) check the pending signals (2) unblock the signal
3	Write a program using fork system call in which the parent displays uppercase characters A-Z and child displays lowercase characters a-z with a delay of max of 2 seconds between each character. Run the program repeatedly and see if there is racing.
	Implement client - server communication using two pipes. Client sends a string. Server makes the first char uppercase and sends it back. Client prints the modified string.
4	Write a program which takes a list of arguments from the command line. a. If the argument is a file, it should display the file contents. b. If the argument is a directory, it should display all files and subdirectories in that directory (Display the no: of files and no: of directories also) c. If there are no arguments, output should be all files and directories in the current working directory (Display the no: of files and no: of directories also) • Ignore . and .. entries while reading directory contents • Arguments can be a combination of filenames and directories.
	Generate a pattern of numbers as in snake and ladder board. Let the parent generate the odd lines (counting the first line as 1) and the child generates the even lines. Use sigusr1 to synchronize and pause to wait for a signal. You may have to make the process sleep before sending the signal to other process so that pause is reached before the signal is sent.

5	Write a program to remove all empty files in a given directory? Use appropriate system calls.
	Experiment signals on fork and exec. 1. Setup handler for the timer – start timer - fork – check who gets the timer signal (parent or child) 2. Setup handler for the timer – start timer – exec another program (Setup handler for timer in exec-ed program) – check who gets the timer signal (caller or exec-ed program)
6	Create static and dynamic libraries. Demonstrate sequence of commands for creating static and dynamic library. Show final linking with main program and output.
	Write a program to compute sum of elements in an array using multithreading. Assume size of array as 100. Threads must compute partial sums by dividing the array into equal size and each partial sum is added to a global variable sum before thread exits. Make use of mutex variable when global variable sum gets updated. Main thread must wait for all threads to complete and print total sum.
7	Write a program which takes a command with arguments on the command line and executes it. ./mypgm ls - l - i This should execute the ls cmd. Use appropriate call from exec family of functions.
	Synchronize access to shared memory segment using semaphore. Parent process creates a shared memory segment which can hold 10 integers and initializes it with random values. Create two child processes P1 and P2 by doing fork and exec. P1 and P2 should execute simultaneously. P1 modifies the shared memory segment by multiplying each integer by 2. P2 modifies shared memory segment by adding 100 to each integer. P1 should be scheduled for execution before P2 and P2 can modify shared memory segment only after P1 has modified memory segment. Use semaphore to synchronize access. Parent has to wait until P1 and P2 has finished execution and finally contents of shared memory has to be printed.
8	Demonstrate creation of orphan and zombie process. Use ps command and observe the output.
	Consider two processes P1 and P2 are printing characters from ‘a’ to ‘z’ alternately. P1 starts execution first and it will print character ‘a’ and wait for P2 (which is initially in blocked state) to print character ‘b’. Once P2 has printed character ‘b’ it will signal process P1 and wait for P1 to print next character. This should continue until all characters have been printed. Make use of two semaphores to synchronize P1 and P2.

9	Write a program to combine stdout and stderr streams into a single file.
	Write a program to perform matrix multiplication using multithreading. Each row in the resultant matrix must be computed by one thread. Row number must be passed as argument for each thread. Main thread must wait for all threads to complete and then print out the resultant matrix.
10	a. Create a file using open system call with name "hello.txt" with permissions rw-rw-r— If the file exists, do not recreate, give a message and exit. b. What happens if we open a symbolic link and read the file? How to read contents of symbolic link?
	Create a message queue with the following message structure. char cmd[5]; int val; cmd takes strings square or cube. Server receives the messages into input message queue from multiple clients. Based on the cmd, server squares or cubes the value in val and sends the message through an output message queue. Each client should receive his message.
11	Experiment on exec as to what happens to signal pending. Is it pending in the called process? Is it delivered to the called process, if signal blocking is removed? Create the caller (1) block a signal (2) send the signal (3) exec the called process Create the called (1) check the pending signals (2) unblock the signal
	Create a shared memory segment. Attach to a process. Then fork and create a new child. Examine what happens to the # number of attaches. Make the parent write a string in the shared memory. Make the child change this string to uppercase. Make the parent display the changed string.
12	Experiment with close-on-exec (FD_CLOEXEC) flag. Write a program which execs another program. Check if file descriptors remain open across exec call. Set close-on-exec flag in exec-ing process using fcntl system call and observe what happens in exec-ed process.
	Create a shared memory segment. Write a program to populate memory with 100 random numbers between 1 and 1000. Write another program to find the average, the biggest and smallest numbers in the array.