**Lab 1**

1. Study of Unix/Linux general purpose utility command list : man, who,cat, cd, cp, ps, ls, mv,rm, mkdir, rmdir, echo, more, date, time, kill, history, chmod, chown, finger, pwd, cal, logout, shutdown commands.
2. Write C programs to simulate UNIX commands like ls, grep, etc.
3. Write a program to implement

1. Create a file

2. Read contents of a file

3. Write to a file

4. Link and unlink a file

5. Copy file

6. Read contents of a file in a reverse order

Using the system calls: open( ), close( ), read( ), write( ), lseek( ), link( ), unlink( ).

1. Determine the size of a file using the lseek command. Once you found out the size, calculate the number of blocks assigned for the file. Compare these results with the similar results obtained when using the function **stat.**
2. Write a program to change current working directory and display the inode details for each file in the new directory using the system calls: opendir( ), readdir( ), closedir( ), getcwd( ), chdir( ).

**Lab 2**

1. In this assignment we will start writing a command interpreter (Shell). The shell will give a prompt for the user to type in a command (from a set of commands), take the command, execute it, and then give the prompt back for the next command (i.e., actually give the functionality of a shell).

Your program should do the following:

Give a prompt "myshell$" for the user to type in a command

Implement the following built in commands:

(a) cd < dir > : changes the directory to "dir"

(b) pwd : prints the current directory

(c) mkdir < dir > : creates a directory called "dir"

(d) rmdir < dir > : removes the directory called "dir"

(e) ls: lists the files in the current directory. It should support both ls without any option and with the option \-l"

(f) exit: exits the shell

The commands are the same as the corresponding Linux commands by the same name. Do "man" to see the descriptions. You can use the standard system calls chdir, getcwd, mkdir, rmdir, readdir etc. to implement the calls (standard C library functions are available for these; look them up). These commands are called built in commands since your shell program will have a function corresponding to each of these commands to execute.

1. Write a C program that finds a file in a file-tree starting from a given directory. The name of the file for which we are searching for, as well as the name of the starting directory should be read from the command line. Optionally, the name of the file can be specified as a pattern using the `\*'character.
2. Write a C program that deletes a directory with all its subfolders. The name of the directory should be read from the command line.
3. Write a program that creates a child process using fork ( ) system call.
4. Write a program where child process sleeps 2 seconds while the parent process waits for the child process to exit. Note how to return of fork is used to control which code is run by parent and which by the child. (**Use if required :** fork( ), exec( ), wait( ), signal( ), kill( ), alarm( ) )

**Lab 3**

1. Write a program in C that creates a child process, waits for the termination of the child and lists its PID, together with the state in which the process was terminated (in decimal and hexadecimal).
2. Write a program that creates a child process. Parent process writes data to pipe and child process reads the data from pipe and prints it on the screen.
3. Use the ps, ps lx, ps tree and ps -aux command to display the process attributes.
4. In a C program, print the address of the variable and enter into a long loop (say using while (1)).
   1. Start three to four processes of the same program and observe the printed address values.
   2. Show how two processes which are members of the relationship parent-child are concurrent from execution point of view, initially the child is copy of the parent, but every process has its own data.
5. Implement inter process communication when

a) Two process are related

b) Two processes are not related

**Lab 4**

1. Write two programs: one called client.c, the other called server.c. The client program lists a prompter and reads from the keyboard two integers and one of the characters '+' or '-'. The read information is transmitted with the help of the system call excel to a child process, which executes the server code. After the child (server) process finishes the operation, it transmits the result to parent process (client) with the help of the system call exit. The client process prints the result on the screen and also reprints the prompter, ready for a new reading.
2. Teat the source code below:

for ( i=1;i≤10;i++) {

fork ( );

printf (“The process with the PID=%d”,getpid ( ));

}

In the next phase, modify the code, such as after all created processes have finished execution, in a file process management.txt the total number of created processes should be stored.

1. Write a program to implement signal handling.
2. Write a program to implement blocking of signal.

**Lab 5 & 6**

1. Write a C program to simulate the followings CPU scheduling algorithms to find turnaround time and waiting time for the given problem. **Assumption:** all the processes arrive at the same time, time slice t=2 sec (for RR scheduling), priority assigned to the processes as P2>P3>P1>P4 (For priority scheduling).
2. First come first serve (FCFS)
3. Shortest Job First (SJF)
4. Round Robin (RR)
5. Priority

|  |  |
| --- | --- |
| **Processes** | **Burst Time** |
| P1 | 24 |
| P2 | 3 |
| P3 | 3 |
| P4 | 7 |

1. Repeat the problem 1 with an assumption that processes arrived as per the given arrival times:

|  |  |  |
| --- | --- | --- |
| **Processes** | **Burst Time** | **Arrival Time** |
| P1 | 24 | 0 |
| P2 | 3 | 2 |
| P3 | 3 | 1 |
| P4 | 7 | 3 |

1. **For RR Scheduling in problem 1 & 2:** Vary the time slice/quantum of RR scheduling from 1 to 3sec(in steps of 1 sec) and plot a graph showing how the average turnaround time for processes vary with time slice/quantum. Also, plot a graph showing how the average waiting time for processes varies with time slice/quantum.
2. Write a C program to simulate multi-level queue scheduling algorithm considering the following scenario. All the processes in the system are divided into two categories – system processes and user processes. System processes are to be given higher priority than user processes. The priority of each process ranges from 1 to 3. Use fixed priority scheduling for all the processes.

|  |  |  |
| --- | --- | --- |
| **Processes** | **Burst Time** | **User/System Process** |
| P1 | 3 | User Process |
| P2 | 2 | System Process |
| P3 | 5 | User Process |
| P4 | 1 | System Process |

**Home Assignment**

1. You need to program the following scheduling algorithms (Preferably in C): FCFS, SJF (with preemption) and Round Robin Scheduling

Each program will read from a file containing a list of processes with pertinent data for the process. The program will simulate the execution of the processes. It will print out the time taken by each process to complete (turn around time) and the wait time and compute the average turn around time for all processes to execute as well as the standard deviation of that average.

**Inputs:** A file name from the keyboard, read the file for the pertinent data, and a possible time slice size (depending on which algorithm is used). The file containing the information on the processes will have each process on a separate line. The processes will be in the file in the order in which they arrive at the OS. Each line will have a process name that will be a string. Following that will be the arrival time of the process. The arrival time will be in reference to the previous process. Following this will be the total execution time. Next will be the elapsed time between I/O interrupts (system calls), next will be the time spent waiting and processing the I/O and finally the priority of the process as an integer (smaller values will have higher priority). It will look like this

P1 0 20.0 1.5 5.0 2

P2 2 15.0 2.0 6.0 1

P3 6 27.0 1.8 3.5 4

p4 4 36.0 2.1 2.6 3

. . . .

x x x x x

The xx indicates the end of the data.

**Outputs:** A prompt for which file is to be read. A prompt for the amount of time for a time-slice. A list of each process and the time it took for it to complete. Then an average time for processes to complete. Finally the standard deviation for the average time for processes to complete.

**Example:**

The name of the file to be read: < filename >

process name turn around time total wait time

? ? ?

? ? ?

The average time for processes to complete was ???

The standard deviation for the average process completion time was ???

**For RR Scheduling:** The time slice if required for your algorithm will be 3. You may assume that a swap (context switching time) is small enough that it can be ignored. Vary the time slice/quantum of RR scheduling from 1 to 10sec (in steps of 1 sec) and plot a graph showing how the average turnaround time for processes vary with time slice/quantum. Also, plot a graph showing how the average waiting times for processes vary with time slice/quantum.

**Lab 7 & 8**

1. Write a program to simulate producer-consumer problem using semaphores.
2. Write a program to simulate the concept of Dining-Philosophers problem using semaphores.
3. Write a program to simulate reader-writer problem using semaphores.
4. **Server Socket Management**. Servers can be designed to limit the number of open connections. For example, a server may wish to have only N socket connections at any point in time. As soon as N connections are made, the server will not accept another incoming connection until an existing connection is released. Write a program using semaphores to synchronize server activity to limit the number of concurrent connections.
5. **The Sleeping-Barber Problem**. A barbershop consists of a waiting room with n chairs and a barber room with one barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy but chairs re available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a program to coordinate the barber and the customers using semaphore.

**Home Assignment**

1. Write a program to simulate producer-consumer problem using monitors.
2. Write a program to simulate the concept of Dining-Philosophers problem using monitors.
3. Write a program to simulate reader-writer problem using monitors.
4. **The Cigarette-Smokers Problem.** Consider a system with three smoker processes and one agent process. Each smoker continuously rolls a cigarette and then smokes it. But to roll and smoke a cigarette, the smoker needs three ingredients: tobacco, paper and matches. One of the smoker processes has paper another has tobacco and the third has matches. The agent has an infinite supply of all three materials. The agent places two of the ingredients on the table. The smoker who as the remaining ingredient then makes and smokes a cigarette, signaling the agent on completion. The agent then puts out another two of the three ingredients, and the cycle repeats. Write a program to synchronize the agent and the smokers using semaphore.