## Table of Contain

1. Write various commands in MS Dos and Linux Operating system along with its uses and example.

⇨ **In the Operating there are use of various commands and code to run in terminal as well to do particular tasks. Each operating system use there are own command line code. Similarly in MS Dos operating system. The various command use in MS-DOS and LINUX are:**

| **Task** | **MS-DOS Command** | **Linux Command** |
|---|---|---|
| **List files/directories** | **dir** | **ls** |
| **Change directory** | **cd** | **cd** |
| **Copy files** | **copy** | **cp** |
| **Clear screen** | **cls** | **clear** |
| **Exit terminal** | **exit** | **exit** |
| **Get help** | **help** | **man** |

2. Write a program to demonstrate the concept of process creation and thread creations?

⇨ **Process is the small part of program which is runed and ready to perform the particular task in the computer. Similarly thread is the small part of the process. Due to the multiple combination of thread the process is created.**

⇨ **Here is a simple program in c programming here it is**

```
#include <stdio.h>
#inlcude <unisted.h>
#inlude <pthread.h>
Void thread_task(){
printf(“This is a  thred!\n”);
return NULL;
}
Int main(){
printf("Creating a process...\n");
   pid_t pid = fork();
   if (pid == 0) {
      printf("This is the child process!\n");
   } else {
      printf("This is the parent process!\n");
   }
   printf("Creating a thread...\n");
   pthread_t thread;
   pthread_create(&thread, NULL, thread_task, NULL);
    pthread_join(thread, NULL);
    printf("Program finished!\n");
   return 0;
}
```

**3. Write a program that simulate Round Robin scheduling algorithm. Write any  of them.**

⇨ **According to the question implementation on the basic round robin is**

```
#include <stdio.h>

void roundRobin(int processes[], int n, int burst_time[], int time_quantum) {
    int remaining_time[n], waiting_time[n], total_time = 0;
    for (int i = 0; i < n; i++) remaining_time[i] = burst_time[i];
    while (1) {
        int done = 1;
        for (int i = 0; i < n; i++) {
            if (remaining_time[i] > 0) {
                done = 0;
                if (remaining_time[i] > time_quantum) {
                    total_time += time_quantum;
                    remaining_time[i] -= time_quantum;
                } else {
                    total_time += remaining_time[i];
                    waiting_time[i] = total_time - burst_time[i];
                    remaining_time[i] = 0;
                }
            }
        }
        if (done) break;
    }
    printf("Process\tWaiting Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\n", processes[i], waiting_time[i]);
    }
}
int main() {
    int processes[] = {1, 2, 3};
    int burst_time[] = {10, 5, 8};
    int n = sizeof(processes) / sizeof(processes[0]);
    int time_quantum = 2;
    roundRobin(processes, n, burst_time, time_quantum);
    return 0;
}
```

4. Demonstrate Various attribute of files and directories.
   - ⇨ **A file is a collection of data or information stored on a computer. It can be a document, image, program, or any other type of data.**
   - ⇨ **A directory is a container used to organize files and other directories. It helps in structuring and managing files in a hierarchical manner.**

     **Files Attributes: name, types, size, permissions, owner, group, creation time, and creation time.**

     **Directory Attributes: name, type , size, permissions, owner, group, creation.**

5. **Perform Case Study on Linux and Prepare case study and summary report to understand different aspect of Linux and window OS System.**

- ⇨ Linux: Open-source, Unix-like OS. Used in servers, development, and supercomputers. It is used in servers, development, super computers, IoT.

  Its strength is free, secure, and customizable as user. It is week in Steeper learning curve, limited software support. It is best for servers, development, and tech-savvy users.
- ⇨ Windows: Proprietary OS, Dominates Desktops, Gaming and enterprise. It is basically use in productivity and enterprise. Its Strength is User-friendly, great software/hardware support. Its weak in Expensive, vulnerable to malware. It Ideal for desktops, gaming, and general users.

Key Differences:

| Aspect | Linux | Windows |
|---|---|---|
| Cost | Free | Paid |
| Customization | Highly customizable | Limited |
| Security | More secure | Vulnerable to malware |
| Performance | Lightweight, efficient for servers | Resource-intensive, good for desktops |
| User Base | Developers, Enterprises | General users, businesses |

6. What is raid? Discuss various labs of raid.

- ⇨ Radi Redundant Array of Independent Disks is a technology that combines multiple physical hard drives into a single logical unit to improve performance, reliability, or both. It is commonly used in servers and storages systems.

Level of RAID:

RAID 0 (striping)
Data is split across multiple disks for fast read/write speeds. In this raid it contains high performance. In this raid there is no redundancy, failure of on disk results in data loss.
RAID 1 (Mirroring)
Data is duplicated across two disks. It data is redundancy is high and data is safe if one disk fails. In this raid 50% storage efficiency that mean half the capacity is used for redundancy.

Raid 5 (striping with parity)
Data and parity information are distributed across there or more disks. It is good balance of performance, storge efficiency, and redundancy. It is slow write speeds due to parity calculation.

7. Write a program to simulate page replacement algorithm and write any.

- ⇨ A page replacement Algorithm is used in operating system to manage memory when a page fault occurs. A page fault happens when a program tries to access a page that is not currently in RAM.  Here is a program of the place replacement algorithm.

```
#include <stdio.h>
#include <stdbool.h>

int fifo(int pages[], int n, int capacity) {
    int frame[capacity], page_faults = 0, index = 0;
    bool isPageFault;

    for (int i = 0; i < capacity; i++) frame[i] = -1;

    for (int i = 0; i < n; i++) {
        isPageFault = true;
        for (int j = 0; j < capacity; j++) {
            if (frame[j] == pages[i]) {
                isPageFault = false;
                break;
            }
        }
        if (isPageFault) {
            frame[index] = pages[i]; // Replace the oldest page
            index = (index + 1) % capacity; // Move to the next frame
            page_faults++;
        }
    }
    return page_faults;
}

int main() {
    int pages[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2};
    int n = sizeof(pages) / sizeof(pages[0]);
    int capacity = 4;
    printf("FIFO Page Faults: %d\n", fifo(pages, n, capacity));
    return 0;
}
```