Name: POV SIDANAK

ID: e20190865

Group: I4-GIC(B)

TP02(Process & Thread)

1. Process is an executing software program that forms the basis for all computation. It is distinct from computer code, although it shares similarities. Unlike the program, which is often considered a passive entity, a process is an active entity that actively performs tasks.

2. The attributes characteristics of a process are:

- Process ID: is a unique identifier assigned by the operating system

- Process State: can be ready, running, waiting, etc.

- CPU register: Like the Program Counter, processes must be stored in various CPU registers for execution in the running state (CPU registers must be saved and restored when a process in swapped in and out of CPU)

- Accounts information: This comprises CPU use for process execution, time constraints, and execution ID, among other things

- I/O status information: It indicates whether a process is engaged in performing I/O operations, such as reading from or writing to a file, waiting for user input, or waiting for data from an external device.

- CPU scheduling information: Process priority and additional scheduling information are required for the process to be scheduled.

3. Process Life Cycle in Operating System

- Start: When a process is started/created first, it is in this state.

- Ready: The process is waiting for the processor to be assigned to it. Processes in the ready state are awaiting assignment of a processor by the operating system so that they can begin or resume execution. This state can be entered when a process is initially started or while it is already running, but may be interrupted by the scheduler to allocate the CPU to another process.

- Running: When a processor is assigned to a process by the operating system scheduler, the process state gets set to running, and the processo r begins executing the instructions of the process.

- Waiting: If a process requires a resource that is not currently available, such as user input or an unavailable file, it enters the waiting state, where it remains until the resource becomes available.

- Terminated or exit: Once a process has finished executing or has been terminated by the operating system, it is moved to the terminated state, where it remains until it is removed from the main memory.

- Suspended Ready: When the ready queue becomes full, some processes are moved to

- suspended ready state.

- Suspended Block: When waiting queue becomes full.

4. Thread known as a single sequential flow of activities being executed in a process; it is also known as the thread of execution or the thread of control. Now, thread execution is possible within any OS's process. Apart from that, a process can have several threads.

5. Multithreading offers several advantages:

- Enhanced throughput of the system: When a process is divided into multiple threads, with each thread function considered as a separate job, the system's throughput can increase as more jobs are completed per unit of time.

- Communication: Communication between multiple threads is easier, as they share a common address space, allowing for direct access to shared memory. In contrast, communication between processes often requires specific inter-process communication (IPC) techniques to exchange data or synchronize operations.

- Resource sharing: Resources like code, data, and files can be shared among all threads within a process. Note: stack and registers can't be shared among the threads. Each thread has its own stack and registers.

- Effective utilization of multiprocessor system: When a single process contains multiple threads, they can be scheduled to run on separate processors concurrently, resulting in faster overall process execution.

- Faster context switch: Context switch time between threads is generally lower compared to process context switch. Process context switching typically incurs higher overhead on the CPU due to the need to save and restore larger amounts of process-specific information.

- Responsiveness: In a process divided into multiple threads, if one thread completes its execution, its output can be immediately returned without waiting for the entire process to finish.

6. The difference between thread and process

> The main distinction is that threads within the same process share a common memory space, whereas processes have their own separate memory spaces. Threads are not fully independent like processes, as they share code section, data section, and OS resources (such as open files and signals) with other threads. However, like processes, each thread has its own program counter (PC), register set, and stack space.

7. Following the code below, there are one A, two B and four C are printed.

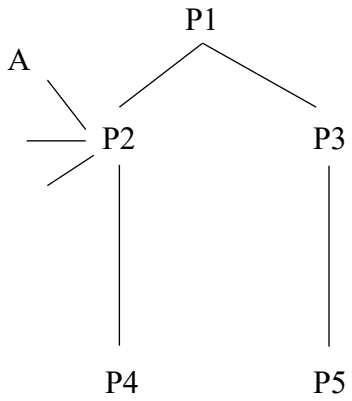
```c
#include <stdio.h>
#include <unistd.h>
int main(int argc,char *argv[]){
    printf("A\n");
    fork();
    printf("B\n");
    fork();
    printf("C\n");
    return 0;
}
```

8. Following the code below

```c
#include <stdio.h>
#include <unistd.h>

int main(){
      printf("A\n");
      pid_t pid=fork();
      if(pid==0) {
         printf("B\n");
      } else {
         printf("C\n");
      }
      printf("D\n");
      fork();
      return 0;
}
```

a. There are 5 processes created.
b. Draw the process hierarchy created by executing this code.



c. For each process in the hierarchy, indicate its output sequence (the order in which it executes printf)