**Rashed Abdulla Alyammahi**

**780696944**

**1.**

a) Ready → Running

Process moves from ready to execute.

b) Running → Blocked

Switch from running to blocking to wait for an event to occur. A request must use a call from the system, the process must wait until the system responds to the request, and it takes some time for the system to respond to the request. Another cause is to wait for the I/O signal. Or have to wait for shared information from another process.

c) Blocked → Ready

While waiting for the event to complete.

**2.**

It takes 180ms to complete all three programs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Time | Process A | Process B | Process C | Notes |
| 10 | Ready | Ready | Running |  |
| 20 | Ready | Ready | Running | Process Cinitiates I/O |
| 30 | Running | Ready | Block |  |
| 40 | Running | Ready | Block | Process A initiates I/O |
| 50 | Block | Running | Block |  |
| 60 | Block | Running | Block |  |
| 70 | Block | Ready | Block |  |
| 80 | Block | Ready | Block |  |
| 90 | Block | Ready | Block |  |
| 100 | Block | Ready | Block |  |
| 110 | Block | Ready | Block |  |
| 120 | Block | Ready | Block |  |
| 130 | Block | Block | Running | Process C I/O done,Process B initiates I/O |
| 140 | Block | Block | Running |  |
| 150 | Ready | Block | Running | Process A I/O done, Process C now done |
| 160 | Running | Block | - |  |
| 170 | Running | Block | - | Process Bnow done |
| 180 | Running | - | - | Process B I/O done, Process Anow done |
| 190 | - | - | - |  |
| 200 | - | - | - |  |
| 210 | - | - | - |  |
| 220 | - | - | - |  |

**3.**

**Purpose of System Calls**:

The system call is the interface between the operating system and the user's program. When a program needs to request a service from the operating system kernel, it uses a system call.

It prevents applications in user space access directly kernel space.

**Purpose of interrupt**:

External devices are comparatively slower than CPU. So it takes a long time to wait for external devices to match the CPU speed, so we have interrupts to notify to CPU when external devices change state or have data.

**Purpose of trap**:

trap can be called as a software interrupt caused by an [excepti](https://en.wikipedia.org/wiki/Exception_handling)onal condition(break, invalid mem...). Trap occurs according to user request and its purpose for debugging.

**SYSTEM CALL and INTERRUPT**:

System call is a call to a subroutine built into the system, while Interrupt is an event, which causes the processor to temporarily hold current execution. One big difference is that system calls are synchronous, while interrupts are not. That means system calls happen at a fixed time, but interrupts can happen at any time due to an unexpected event. So when a system call occurs the processor only has to remember where to return to but in the event of an interrupt the processor has to remember both where to return to and the state of the system.

INTERRUPT and TRAP:

Interrupts are hardware interrupts, while traps are software-invoked interrupts. The presence of interrupts usually disables other hardware interrupts, traps do not. If you need to disallow hardware interrupts until a trap is served, you need to explicitly clear the interrupt flag. And often the interrupt flag on the computer affects the (hardware) interrupt as opposed to a trap. This means that clearing this flag will not prevent traps. Unlike traps, interrupts should preserve the previous state of the CPU.

**4.**

Privilege levels are used to provide protection between different components of the software stack, and attempting to perform operations that are not allowed by the current privileged mode will cause an exception. These exceptions will normally cause traps into an underlying execution environment.

**Advantage of multiple privilege levels:**

- Privilege levels provide protection between different components of the software stack, and attempts to perform operations not allowed by the current privilege mode will cause an exception. These exceptions will normally cause traps into an underlying execution environment → It’s security, minimize attack from not permitted applications.

- It’s stable when code follow the layers of privilege levels.

**Advantage of single privilege levels:**

- It’s easy when code to control all CPU and external devices without any restrictions.

- The performance is better because it takes no time to check privileged and directly execute in anywhere.