Hassan Zaib Jadoon
--22PWCSE2144--
: Assignment # 02:

ANSWER:-

Operating System can be categorized based on their structure or organization.

Monolithic Kernals:-

→ Pros:-

Performance:- Since the kernal runs in priveleged mode, System calls and operations can be executed with minimal overhead.

Simplicity:- Development and maintaince relatively straight forward, compared to other architectures.*

Cons:-

Stability:- A bug in one part of kernal can potentially crash the entire system.

Scalability:- Adding new features or device drivers can be complex and may require modifying the kernal directly.

Microkernel:-

Pros:

- Modularity: Most services running outside the kernel space. This makes them easier to maintain and extend.
- Reliability: By keeping the kernel minimal, the potential impact of bug is reduced.

Cons:-

Performance Overhead: Communication between user level services and the kernal adds overhead, potentially impacting performance.
Complexity: Implementing complex functionality as seperate user level services can introduce complexity in the system.

Modular Approach:-

Pros

- Flexibility: Allow for easy addition or removal of modules.
- Scalability: Facilitates to adding more modules.

Cons:-

- Intigration: Seamless intigration between modules can be challenging lead to issues.
- Complexity: Managing modules dependencies b/w modules can add complexity to system.

Layered approach:-

Pros:

- Abstraction: provides clear separation of concerns.
- Hierarchical structure offers a structured hierarchy where higher layers build upon lower layers, facilitating design and implementation.
- Isolation:- Each layer operates independently, enhancing fault tolerance and allowing for easier debugging.

Cons:

- Rigidity: Changes to one layer may require modification in multiple layers, making the system less flexible.
- Limited Communication: Communication between non-adjacent layers maybe challenging, potentially hindering system functionality.

- Hybrid kernel:-

Pros:

- Balance:- Hybrid kernels aim to strike a balance between the performance of monolithic kernel and the modularity of MK.
- Flexibility:- They offers the ability to run certain critical services in kernal space for the performance while keeping non-essential services in user space for better isolation and reliability.

Question No 02:

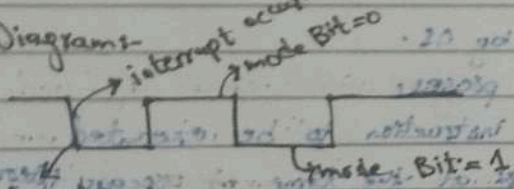
➡ Privileged Instructions:

are those that can only be executed in privileged mode also known as kernel mode.

➡ How executed by user process:

When a user process attempts to execute a privileged instruction, it triggers an exception or interrupt.

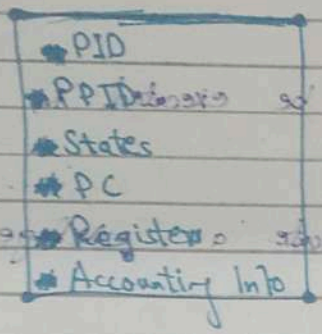
Diagram:-



➡ Question No 03:-

➡ PCB:

PCB is data structure used by the operating system to manage information about a process. It contain a various piece of information about a process. It contain a various piece of information related to the process execution state, resource usage and other important details.



➡ PID:- Unique identifier assigned to the process by OS.

➡ Process State:- Indicates the current state of process

➡ Program Counter:- stores the address of next instruction to be executed

➡ CPU Registers:- Contains the values of CPU registers at the time of process interruption

➡ Accounting Information:- Keep track of resource usage statistics, such as CPU time consumed, memory usage and I/O operations performed.

➡ Question No 04:-

Race Conditions:

Both the producer and consumer processes access the shared buffer concurrently without proper synchronization.

Buffer Overflow:

If the produced data faster than the consumer consumes it, the buffer may overflow.

Busy waiting:

Both the producer and consumer process use busy waiting loops.

Correct Code:

➡ Producer:-

```
item nextproduced;
while (1) {
    while ((in + 1) % Buffer_size) = out);
next_produced = fun();
    Buffer [in] = next_produced;
in = (in+1) % Buffer_size; }
```

➡ Consumer:-

```
item - next_consumed;
while (1) {
        while (in == out);
next_consumed = Buffer [out];
    out = (out + 1) % Buffer_Size; }
```

Question No 05:-

A process undergoes several stages throughout its existence typically referred to as a process life cycle.

New:- The process is created in this state.

Ready:- The process is loaded into main memory and is ready to be executed

Running:- The CPU executes the instructions of the process.

Waiting:- The process cannot proceed further until some event occur.

Termination:- The process finishes its execution and releases any allocated resource.

➡ Transition B/w stages are prompted by various factors.

⇒ System Calls:-

Process may transition b/w states in response to system calls

⇒ Interrupt:-

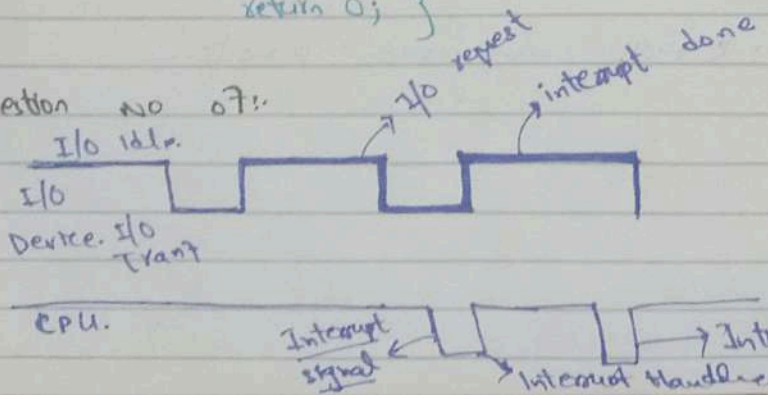
Hardware or software interrupts can cause transition.

Question No 06:-

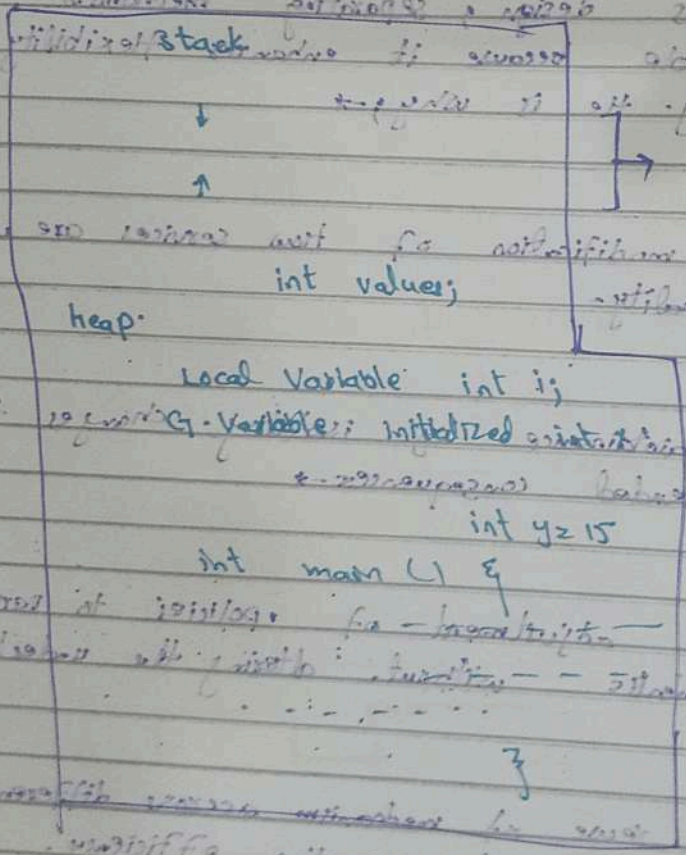
```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
```

```
int main() {   int i, x, n;
               n=3
for (i=0; i<3; i++) {
                        x= fork();
    if (x>0) {
                        break; }
    else if {
        if (i==0) {     /bin/
                        execl ("ps", "ps", NULL); }
        else if (i==1) {  /bin/
                        execl ("ps", "ps", NULL); } }
wait (NULL);
return 0; }
```

Question No 07:



Question No 08:



➡ Question NO 10:-★

In Operating Systems design, separating mechanism from policy is desireable because it enhances flexibility, maintainability, and adapability. He is why;-★

Flexibility.

Allows independant modification of how services are used and the underlying functionality.

➡ Improves Maintainability:

➡ Simplifies system maintainance by isolating changes to policies reducing the risk of unintended consequences.-★

➡ Facilitates Adaptability:-

Enables dynamic adjustment of policies to varying conditions or evolving requirements without altering the underlying mechanism.

➡ Promotes Reuseability:- Encourage reuse of mechanism accross different contexts applications reducing development time and promoting efficiency.

signal ← → Interrupt Handler