Date.....

OSSP Tutorial-1

Q1 Two design goals that may contradict each other in building an operating system are resource utilization and responsiveness.

Resource Utilization → This aims to efficiently use hardware resources like CPU and memory to maximise system throughput and efficiency. It involves running background tasks & computations to make the most of resources.

Responsiveness → This focusses on providing quicky & timely responses. It ensures that user-initiated tasks are processed promptly.

Heavy resource utilization can lead to delayed responsiveness. When the system is busy with resources-intensive tasks, user interactions might be sluggish, causing frustation.

Q2 Kernel mode and user mode are distinct privilege levels in an operating system. Kernel mode permits direct access to hardware and privileged instructions, serving critical tasks. User mode confines applications' resources access, bolstering security and stability. This separation prevents unauthorised access, sustains system stability, enables efficient resource mangement and facilitates ~~et~~ controlled error handling.

Date.....

Q3. a) Disable all interrupts → This instruction should be allowed only in kernel mode. Disabling interrupts can have a significant impact on the system's functionality.

b) Read the time-of-day clock → This instruction doesn't necessarily need to be restricted to kernel mode. User-mode instructions might need access to the current time for various purposes.

c) Set the time-of-day clock → This instruction should be allowed only in kernel mode. This should be handled by OS's kernel to prevent unauthorised changes.

d) Change the memory map → This instruction should be allowed only in kernel mode. It includes reconfiguring memory allocation and management. Allow user-application to manipulate could lead to security vulnerabilities, data corruption, & system instability.

Q4. A trap instruction, also known as a software interrupt, is a command used by programs to request services or perform privileged operations from the operating system. It's a controlled way to switch from user mode to kernel mode, allowing access to protected resources and functions. When a program executes it, it triggers a transition to the operating system's handler, which carries out the requested operation in a secure and controlled manner. This mechanism enhances security and prevents unauthorised access to critical system resources.

Spiral

Date.....

Q5. a) CPU → Time Multiplexing is commonly used to share the CPU among multiple resources.

b) Memory → Both time and space multiplexing can be used for memory sharing.

c) Disk → Disk resources can be shared using time multiplexing, where multiple users or process take turns accessing the disk. Disk partitioning is an example of space multiplexing.

d) Network card → It often use both time & space multiplexing.

e) Printer → It can be shared using both time & space multiplexing.

f) Keyboard & Display → They are usually shared using time multiplexing.

Real time scenarios →

1. Time Multiplexing → Telephone networks, Satellite Television.
2. Space Multiplexing → Wireless Internet Access Points, Data centers.

Q6 It is not essential for the library procedure and system call to have the same name. Both are important in their own ways: the library procedure enhances code organisation and reusability, while the system call provides essential interactions between user-level programs and the operating system. Clarity in ~~mean~~ naming for both is valuable for code comprehension and maintainability.

Date.....

Q7. → Steps 1-3 → Push parameters nbytes, &buffer, fd.
Step 4 → Call the library procedure read.
Step 5 → The library procedure put a code for read in register.
Step 6 → Call trap function, trap to the kernel.
Step 7 → The kernel locates and dispatches the system call handler via a table of pointers ~~and~~ to system call handlers indexed on system call no.
Step 8 → The system call handler runs
Step 9 → Returns to the user space library procedure.
Step 10 → The library procedure returns to the user program.
Step 11 → The SP is incremented (pop up, as stack grows downward) to clean up the stack.

Q8 I will choose Windows →
Reasons →

1. Software Compatibility → Many software application to be used are designed for Windows, for programming, design and tasks.

2. Ease of use → Windows is more user friendly.

3. Gaming → Windows offers better compatibility with a wide range of games.

4. Microsoft office → There is heavy use of MS office applications & Windows provides seamless integration & performance for these tools.

5. Commercial Software Support → Some commercial softwares are better optimised for Windows, ensuring smooth funct.

Spiral

Date.....

Q9 1) Batch Operating System →
- Executes jobs in sequence
- No user interaction
- Efficient resource utilisation.
- Limited multitasking.

2) Interactive OS →
- Supports direct user ~~Integ~~ interaction.
- Allows multitasking.
- Prioritizes quick user responses.
- Manages foreground & background tasks.

3) Time Sharing OS →
- Shares CPU among multiple users.
- Combines interactive and multitasking features.
- Ensures fairness & equity.
- Involves frequent context switching.

4) Real Time OS →
- Focusses on precise timing.
- Manages tasks based on deadlines.
- Aims for minimal execution time variability.
- Can be hard or soft real-time.

5) Distributed OS →
- Operates in a networked environment.
- Enables resource sharing among interconnected comp.
- Provides transparency to users.
- Manages concurrency and co-ordination across nodes.

Spiral

Date.....

Q1a. Advantages of Open-Source Operating Systems→

1. Customizability → Developers & tech-savvy user can tailor the OS to fit specific needs.
2. Transparency → Security-conscious users can review the code for vulnerabilities.
3. Community Support → Users can seek help from active open-source communities.
4. Cost Savings → Individuals & small businesses can save money as open-source OS are free.
5. Innovation → Collaboration among developers leads to faster feature development.

Disadvantages of Open-source Operating Systems →

1. Complexity → Non-technical users may struggle with installation & maintenance.
2. Lack of Support → official support channels might be lacking, affecting non-tech users.
3. Compatibility → Compatibility issues with certain software & hardware can arise.
4. Fragmentation → Diversity of distributions can lead to fragmentation & management complexities.
5. Commercial Software → Limited support for several commercial software.