

Questions for Resource Container and Lottery Scheduling:

1. *Briefly explain the difference among 'protection domain', 'resource principal', and 'execution/scheduling unit'.*

Process is the protection domain. We need to avoid activities from different protection domains to overwrite/see each other's state.

Resource principle is about resource allocation. Entities that belong to the same resource principle will share the same resource allocation and scheduling priority.

Execution/scheduling unit is thread in OSes that support threads. It is the unit that will occupy CPU at any moment.

2. *Suppose we need to schedule applications with the same priority but different numbers of threads. What is the resource-container's solution to achieve fairness among these applications? What is the solution under lottery scheduling?*

Resource container: create two resource containers, one for each application, and give them equal priority. Whenever a thread is created, bind it with its corresponding application's resource container.

Lottery scheduling: create two currencies, one for each application. Assign the same amount of base tickets to these two currencies. Suppose each thread with tickets issued by corresponding application currency.

Questions for Scheduler Activation: *Compare user-level thread and kernel-level thread: name two advantages of user-level thread over kernel-level thread and two advantages of kernel-level thread over user-level thread.*

Advantages of user-level threads:

- It has less overhead on operations like synchronizations, thread creation, etc.
- It has more flexibility regarding scheduling algorithm and others.

Disadvantage of user-level threads:

- Kernel is not aware of the existence of multiple threads. One user-thread's I/O could block the whole group of user threads.
- In M:N (M user threads sharing N kernel threads) mode, Kernel cannot tell the priority difference among the M user threads. Bad kernel scheduling decision is inevitable.

Questions for Monitors: *Name one difference between the design of Hoare-Monitor and Mesa-Monitor.*

The biggest difference is on their designs of condition variables.

In Hoare-Monitor, the signaling process is blocked; one waiting process immediately wakes up and executes. The invariant associated with the condition variable is GUARANTEED to be satisfied when the waiting process wakes up.

In Mesa-Monitor, the signaling process continues its execution after the signal. The waiting process might wake up and finds out the condition it is waiting for is still unsatisfied.

Questions for Unix:

1. *Give two examples of UNIX design that help the performance of disk file access.*

The open-file table caches the translation from file path/name to inode.

Buffer cache in physical memory significantly decreases the number of disk accesses.

Questions for Multics:

1. *The virtual memory system of Multics used both segmentation and paging. What is the difference between a segment and a page?*

Each segment is a logical unit in the system. It serves as a sharing unit, as well as protection unit. It is easy for developers to specify one sharing/protection property for the whole segment in their program. Different segments can have different sizes. Every segment can consist of multiple pages.

Page is a better unit for physical memory management. Every page has a fixed size. It occupies consecutive physical memory/disk space.

2. *When two processes share a segment, does that segment have the same segment-ID in the two processes?*

No. The segment ID is determined when the segment is loaded into a process.

Questions for THE and Nucleus: *List three differences between the design of THE and Nucleus?*

THE:

monolithic kernel, hierarchical design;
using semaphore for synchronization.
Supporting a fixed number of processes

Nucleus:

micro-kernel.

Many operating system functionalities are implemented as processes running on the micro-kernel.

- Using message passing

- Supporting dynamic creation of processes

- Nucleus is more reliable (the `kernel` is smaller, some resource-management component's crash will not affect other part of the system, message passing naturally puts more isolation among processes than shared memory schemes)

- Nucleus can provide more flexibility in system design

- Nucleus has worse performance than monolithic kernel