**CS3451 INTRODUCTION TO OPERATING SYSTEMS**

**UNIT I INTRODUCTION**

**1. What is a Virtual Machine? List out the advantages of virtualization. Explain the creation of a Virtual Machine with an architecture diagram. [An] (NOV/DEC 2013)**

**Definition of Virtual Machine (VM):**

A **Virtual Machine (VM)** is a software emulation of a physical computer. It runs an operating system and applications just like a physical machine, but it is hosted within another computing environment known as the **host machine**. The VM is managed by software called a **hypervisor** or **Virtual Machine Monitor (VMM)**.

**Advantages of Virtualization:**

1. **Efficient Resource Utilization:** Multiple VMs can run on a single physical machine, making better use of CPU, memory, and storage.
2. **Isolation:** Each VM is isolated from others, enhancing security and stability.
3. **Cost Reduction:** Reduces hardware costs since multiple OSes run on one physical system.
4. **Portability:** VMs can be easily moved, copied, or restored.
5. **Testing and Development:** Safe environment for testing new applications or OS without affecting the host.
6. **Disaster Recovery:** Easier to back up and recover entire systems.
7. **Legacy Application Support:** Run older OSes or apps on modern hardware.

**Creation of a Virtual Machine – Explanation with Architecture Diagram:**

**Architecture Diagram:**

sql

CopyEdit

+-------------------------------------------+

| Guest Operating System (VM) |

| (Multiple VMs with their own OS) |

+-------------------------------------------+

| Virtual Machine Monitor (VMM) |

| / Hypervisor Layer |

+-------------------------------------------+

| Host Operating System (optional) |

+-------------------------------------------+

| Physical Hardware (CPU, Memory, etc) |

+-------------------------------------------+

**Explanation:**

* **Physical Hardware:** The real hardware of the host machine.
* **Host OS:** Optional layer, present in Type-2 hypervisors (like VMware Workstation).
* **Hypervisor/VMM:** Responsible for managing VMs, allocating resources, and ensuring isolation. Two types:
  + **Type 1 (Bare-metal):** Runs directly on hardware (e.g., VMware ESXi).
  + **Type 2 (Hosted):** Runs on top of host OS (e.g., VirtualBox).
* **Guest OS:** The OS running inside the VM (can be Linux, Windows, etc.).

**Q7. Write short notes on Operating System Services and Components. [U] (MAY/JUNE 2012)**

**Operating System Services:**

Operating systems provide essential services to ensure smooth operation and support for user applications.

1. **Program Execution:** Loads and runs programs, provides mechanisms for process control.
2. **I/O Operations:** Manages input/output devices and provides user-friendly interfaces.
3. **File System Manipulation:** Allows creation, deletion, reading, writing, and organization of files.
4. **Communication:** Enables inter-process communication (IPC) through message passing or shared memory.
5. **Error Detection:** Monitors system for potential errors and takes corrective actions.
6. **Resource Allocation:** Allocates resources like CPU, memory, and I/O devices to multiple users/processes.
7. **Security and Protection:** Prevents unauthorized access to resources and ensures data protection.

**Operating System Components:**

1. **Process Management:**
   * Handles creation, scheduling, and termination of processes.
   * Manages process synchronization and communication.
2. **Memory Management:**
   * Tracks memory usage and allocates/deallocates memory space.
   * Uses techniques like paging and segmentation.
3. **File System Management:**
   * Organizes data in files and directories.
   * Manages file permissions and access rights.
4. **Device Management:**
   * Manages device communication via drivers.
   * Handles I/O buffering and spooling.
5. **Secondary Storage Management:**
   * Handles storage allocation, free space management, and disk scheduling.
6. **Networking:**
   * Provides mechanisms for communication over network protocols (TCP/IP).
7. **User Interface (UI):**
   * Offers CLI (Command Line Interface) or GUI (Graphical User Interface) to users.
8. **Security and Protection System:**
   * Ensures user authentication and protects against malicious access.

**UNIT II PROCESS MANAGEMENT**

**1. Write in detail about several CPU Scheduling Algorithms. [An] (APRIL/MAY 2011)**

**What is CPU Scheduling?**

CPU scheduling is the process of selecting a process from the ready queue and allocating the CPU to it. It occurs when the CPU is idle or when a running process is switched out.

**Types of CPU Scheduling Algorithms:**

1. **First-Come, First-Served (FCFS):**

* **Description:** Processes are executed in the order of arrival.
* **Type:** Non-preemptive
* **Advantages:** Simple to implement.
* **Disadvantages:** Long average waiting time; suffers from **convoy effect**.

1. **Shortest Job Next (SJN) / Shortest Job First (SJF):**

* **Description:** The process with the shortest burst time is scheduled first.
* **Type:** Can be preemptive (Shortest Remaining Time First - SRTF) or non-preemptive.
* **Advantages:** Optimal average waiting time.
* **Disadvantages:** Requires knowledge of burst time; may lead to **starvation**.

1. **Round Robin (RR):**

* **Description:** Each process gets a fixed time slice (quantum) in a cyclic order.
* **Type:** Preemptive
* **Advantages:** Fair and responsive; ideal for time-sharing systems.
* **Disadvantages:** Performance depends on the quantum size.

1. **Priority Scheduling:**

* **Description:** CPU is allocated to the process with the highest priority.
* **Type:** Can be preemptive or non-preemptive.
* **Advantages:** Important tasks get more CPU time.
* **Disadvantages:** Lower-priority processes may **starve**.

1. **Multilevel Queue Scheduling:**

* **Description:** Processes are divided into different queues (foreground, background), each with its own scheduling.
* **Type:** Can use different algorithms per queue.
* **Advantages:** Special handling for different process types.
* **Disadvantages:** Rigid classification; low flexibility.

1. **Multilevel Feedback Queue Scheduling:**

* **Description:** Similar to multilevel queue but allows movement between queues.
* **Type:** Preemptive
* **Advantages:** Dynamically adjusts to process behavior.
* **Disadvantages:** Complex to implement and tune.

**Comparison Table:**

| **Algorithm** | **Preemptive** | **Starvation** | **Fairness** | **Efficiency** |
| --- | --- | --- | --- | --- |
| FCFS | No | No | Low | Low |
| SJF/SRTF | Optional | Yes | Low | High |
| Round Robin | Yes | No | High | Medium |
| Priority | Optional | Yes | Low | Medium |
| Multilevel | Optional | Yes | Medium | Medium |
| Feedback Queue | Yes | No | High | High |

**2. What is a Critical Section? Specify the Requirements for a Solution to the Critical Section Problem. [An] (NOV/DEC 2012)**

**Definition of Critical Section:**

A **critical section** is a part of the program where shared resources (like variables, files, or devices) are accessed. Multiple processes must not execute their critical sections simultaneously to avoid **race conditions**.

**Critical Section Problem:**

It refers to the challenge of designing a protocol that ensures that only one process accesses the critical section at a time in a **concurrent system**.

**Requirements for a Correct Solution:**

1. **Mutual Exclusion:**
   * Only one process can enter the critical section at a time.
   * Ensures consistency of shared resources.
2. **Progress:**
   * If no process is in the critical section and some wish to enter, one of them must be allowed to proceed.
   * No process should unnecessarily wait.
3. **Bounded Waiting:**
   * A bound must exist on the number of times other processes are allowed to enter their critical sections before a waiting process gets a turn.
   * Prevents **starvation**.

**Typical Solutions:**

* **Software:** Peterson’s algorithm, Dekker’s algorithm
* **Hardware:** Test-and-Set, Semaphores
* **OS Constructs:** Mutexes, Monitors

**3. How Monitors Help in Process Synchronization? [An] (NOV/DEC 2009)**

**What is a Monitor?**

A **monitor** is a high-level synchronization construct that combines **mutual exclusion** and the ability to wait (block) and signal (resume) threads. It’s a programming abstraction used in concurrent programming.

**Structure of a Monitor:**

A monitor typically contains:

* Shared variables
* Procedures (functions)
* Synchronization mechanisms (like condition variables)

**Working of Monitors:**

* Only **one process** can execute any of the monitor procedures at a time.
* Others attempting to enter are **blocked** until the monitor becomes available.

**Condition Variables in Monitors:**

Used for process communication inside the monitor.

* wait(condition) – A process waits and releases the monitor lock.
* signal(condition) – Wakes up a waiting process on the condition.

**Example:**

text

CopyEdit

monitor ProducerConsumer {

condition full, empty;

procedure insert(item) {

if buffer is full then wait(full);

add item to buffer;

signal(empty);

}

procedure remove() {

if buffer is empty then wait(empty);

remove item from buffer;

signal(full);

}

}

**Advantages of Using Monitors:**

1. Simplifies complex synchronization problems.
2. Ensures **mutual exclusion** automatically.
3. Avoids low-level constructs like semaphores.
4. Enhances **modularity** and code clarity.

**UNIT III MEMORY MANAGEMENT**

**1. Describe the Hierarchical Paging Technique for Structuring Page Tables. (8) [An] (MAY/JUNE 2013)**

**What is Hierarchical Paging?**

Hierarchical paging is a **multi-level paging** technique used to manage large address spaces efficiently by breaking the page table into multiple levels. Instead of one large page table, the page table is split into smaller tables, reducing memory overhead.

**Need for Hierarchical Paging:**

* In systems with large address spaces (e.g., 32-bit or 64-bit), a single-level page table becomes too large.
* To reduce the size of each page table and minimize memory wastage.

**How It Works:**

Let’s consider a **32-bit address** and **4 KB page size**:

* **Page offset = 12 bits** (since 2^12 = 4096 bytes = 4 KB)
* Remaining **20 bits** are used for the page number

Split the 20-bit page number:

* **First 10 bits** → Index into **outer page table (page directory)**
* **Next 10 bits** → Index into **inner page table**

**Translation Steps:**

1. CPU generates a **virtual address**.
2. First 10 bits → Used to locate the page table from the page directory.
3. Next 10 bits → Used to locate the frame in that page table.
4. Last 12 bits → Offset added to the frame's base address to get the physical address.

**Diagram:**

pgsql

CopyEdit

Virtual Address (32-bit)

+----------+-----------+-------------+

| 10 bits | 10 bits | 12 bits |

| Dir Index| Page Index| Page Offset |

+----------+-----------+-------------+

Page Directory → Page Table → Frame → Physical Address

**Advantages:**

* Reduces memory used by page tables.
* Supports sparse address spaces.
* Efficient for large programs.

**2. What is the Cause for Thrashing? How Does the System Detect Thrashing? What Can the System Do to Eliminate This Problem? [An] (MAY/JUNE 2009)**

**What is Thrashing?**

**Thrashing** occurs when the system spends more time **swapping pages in and out of memory** than executing actual processes. It severely degrades performance.

**Causes of Thrashing:**

1. **High degree of multiprogramming.**
2. Processes do not have enough frames allocated.
3. Frequent page faults due to lack of locality of reference.
4. Overcommitment of physical memory.

**Detection of Thrashing:**

* System monitors the **CPU utilization** and **page-fault rate**.
* If CPU utilization is **low** and page-fault rate is **high**, it indicates thrashing.

**Steps to Eliminate Thrashing:**

1. **Reduce Degree of Multiprogramming:**
   * Remove or suspend processes to free up memory.
2. **Use Local Page Replacement:**
   * Allocate a fixed number of frames to each process so that it doesn't steal frames from others.
3. **Working Set Model:**
   * Allocate enough frames to each process based on its working set (recently used pages).
4. **Page-Fault Frequency (PFF) Algorithm:**
   * Control allocation based on acceptable page-fault rate. Increase or decrease frames dynamically.

**Conclusion:**

To avoid thrashing, the OS must carefully manage memory allocation and avoid overloading the system.

**3. Write in Detail About Segmentation. [U] (NOV/DEC 2009)**

**What is Segmentation?**

**Segmentation** is a memory management technique where a process is divided into variable-sized segments based on logical divisions such as functions, arrays, code, data, and stack.

**Need for Segmentation:**

* Better matches the way programs are logically organized.
* Provides protection and sharing by segment-level access control.

**Segment Table:**

Each process has a **segment table** with entries containing:

* **Base**: Starting physical address of the segment.
* **Limit**: Length of the segment.

**Address Translation:**

1. A **logical address** is given as **<segment number, offset>**.
2. The segment number is used to index the segment table.
3. Offset is added to the base address to get the **physical address**.
4. If the offset > limit, a **segmentation fault** occurs.

**Diagram:**

mathematica

CopyEdit

Logical Address → [Segment Number | Offset]

↓

Segment Table Lookup → [Base + Offset] = Physical Address

**Advantages of Segmentation:**

* Reflects the logical structure of programs.
* Facilitates code sharing and protection.
* Allows dynamic growing/shrinking of segments.

**Disadvantages:**

* May lead to **external fragmentation**.
* More complex memory management.

**UNIT IV STORAGE MANAGEMENT**

**1) Explain the Different Disk Scheduling Algorithms with Examples. [An]**

*(APRIL/MAY 2010, MAY/JUNE 2012, APRIL/MAY 2011, MAY/JUNE 2013, MAY/JUNE 2014)*

**What is Disk Scheduling?**

Disk scheduling is the process of deciding the order in which disk I/O requests are to be served. Efficient scheduling improves **disk performance** and **reduces seek time**.

**Disk Scheduling Algorithms:**

Let’s assume the disk queue has requests for the following tracks:  
**Queue = [98, 183, 37, 122, 14, 124, 65, 67]**  
**Initial head position = 53**

**1. FCFS (First Come First Serve):**

* Services requests in the order they arrive.
* **Seek sequence:** 53 → 98 → 183 → 37 → 122 → 14 → 124 → 65 → 67
* **Total seek time:** Add absolute differences between consecutive positions.
* **Pros:** Simple, fair.
* **Cons:** Poor performance, long average seek time.

**2. SSTF (Shortest Seek Time First):**

* Selects the request closest to the current head.
* From 53, the closest is 65, then 67, 37, 14, etc.
* **Seek sequence:** 53 → 65 → 67 → 37 → 14 → 98 → 122 → 124 → 183
* **Pros:** Better performance than FCFS.
* **Cons:** May cause **starvation** for far requests.

**3. SCAN (Elevator Algorithm):**

* Moves the head in one direction and services requests until it reaches the end, then reverses.
* Direction: Assume moving towards 0.
* **Seek sequence:** 53 → 37 → 14 → 0 → then 65 → 67 → 98 → 122 → 124 → 183
* **Pros:** Fair, more uniform wait time.
* **Cons:** Long seek for edge requests.

**4. C-SCAN (Circular SCAN):**

* Head moves in one direction (e.g., toward 0), serves all requests, then jumps to end and resumes.
* **Seek sequence:** 53 → 37 → 14 → 0 → jump to 199 → 183 → 124 → 122 → 98 → 67 → 65
* **Pros:** More uniform service time.
* **Cons:** Jump adds overhead.

**5. LOOK:**

* Like SCAN but only goes as far as the last request in each direction, then reverses.
* **Seek sequence:** 53 → 37 → 14 → then back → 65 → 67 → 98 → 122 → 124 → 183
* **Pros:** Avoids unnecessary travel.

**6. C-LOOK:**

* Like C-SCAN but goes only to the last request, then jumps to the other end.
* **Seek sequence:** 53 → 65 → 67 → 98 → 122 → 124 → 183 → jump to 14 → 37
* **Pros:** More efficient than C-SCAN.

**2) Explain and Compare FCFS, SSTF, C-SCAN, and C-LOOK Disk Scheduling Algorithms with Examples. [An]**

*(NOV/DEC 2012)*

**Assume the Same Request Queue:**

**Queue = [98, 183, 37, 122, 14, 124, 65, 67], Head = 53**

**A) FCFS:**

* Order of request service: [98, 183, 37, 122, 14, 124, 65, 67]
* **Seek Time:** Large due to randomness in request order.

**B) SSTF:**

* Chooses the closest request first.
* Sequence: [65, 67, 37, 14, 98, 122, 124, 183]
* **Pros:** Improved performance.
* **Cons:** Starvation possible.

**C) C-SCAN:**

* Head moves in one direction, reaches the end, jumps to start.
* Assume direction toward 0.
* Sequence: [37, 14, 0] → Jump → [183, 124, 122, 98, 67, 65]
* **Pros:** Uniform wait time.

**D) C-LOOK:**

* Goes to last request, jumps back to lowest.
* Sequence: [65, 67, 98, 122, 124, 183] → Jump → [14, 37]
* **Efficient circular scheduling**

**Comparison Table:**

| **Algorithm** | **Approach** | **Starvation** | **Fairness** | **Seek Time** | **Movement Direction** |
| --- | --- | --- | --- | --- | --- |
| FCFS | Arrival Order | No | Medium | High | Bidirectional |
| SSTF | Shortest Seek | Yes | Low | Low | Bidirectional |
| C-SCAN | Circular Sweep | No | High | Medium | One Direction |
| C-LOOK | Circular to last | No | High | Low | One Direction |

**3) Write Short Notes on Disk Management. [U]**

*(NOV/DEC 2009)*

**What is Disk Management?**

Disk Management is a part of an operating system that handles **disk allocation, deallocation, space management**, and **disk I/O scheduling**.

**Main Functions of Disk Management:**

**A) Disk Formatting:**

* **Low-level formatting:** Prepares the physical disk by creating sectors and tracks.
* **High-level formatting:** Creates a file system (like FAT, NTFS).

**B) Partitioning:**

* Dividing a physical disk into multiple logical volumes.
* Each partition can have its own file system and OS.

**C) Free Space Management:**

* Keeps track of free blocks using:
  + Bitmaps
  + Free lists
  + Grouping methods

**D) Disk Scheduling:**

* Determines the order of servicing disk I/O requests.
* Includes FCFS, SSTF, SCAN, LOOK, etc.

**E) Disk Caching:**

* Frequently accessed data is stored in high-speed memory (cache) for faster access.

**F) Bad Block Management:**

* Identifies and skips bad sectors using spare blocks or remapping

**UNIT V VIRTUAL MACHINES AND MOBILE OS**

**1. Explain the Significance and Steps Involved in Setting Up Xen, VMware Software on Linux Host for Successful Virtualization.**

**[An] May/June 2015**

**Significance of Xen and VMware in Virtualization:**

**Xen:**

* An **open-source** hypervisor using **paravirtualization** and **hardware-assisted virtualization**.
* Efficiently runs multiple operating systems on the same hardware.
* Ideal for **server consolidation**, **cloud computing**, and **testing environments**.

**VMware:**

* A **proprietary virtualization platform** offering **full virtualization**.
* Supports running multiple OSes with minimal configuration.
* Widely used in **enterprise environments** for flexibility and robustness.

**Steps to Set Up Xen on a Linux Host:**

**Requirements:**

* Linux distribution (e.g., CentOS, Debian)
* Root privileges
* Virtualization-supported CPU (Intel VT-x or AMD-V)

**Installation Steps:**

1. **Update the System:**

bash

CopyEdit

sudo apt update && sudo apt upgrade

1. **Install Xen Hypervisor and Tools:**
   * For Debian/Ubuntu:

bash

CopyEdit

sudo apt install xen-hypervisor-amd64 xen-tools

* + For CentOS:

bash

CopyEdit

sudo yum install xen

1. **Modify GRUB to Boot into Xen:**
   * Set Xen as the default boot option in /etc/default/grub.
   * Update GRUB:

bash

CopyEdit

sudo update-grub

1. **Reboot into Xen Kernel:**

bash

CopyEdit

sudo reboot

1. **Verify Xen is Running:**

bash

CopyEdit

xl info

1. **Create Virtual Machines (DomU):**
   * Configure with xen-create-image.
   * Start with xl create <config\_file>.

**Steps to Set Up VMware on Linux Host:**

**Requirements:**

* Linux OS (Ubuntu, Fedora, etc.)
* VMware Workstation/Player installer (.bundle file)

**Installation Steps:**

1. **Download VMware Workstation or Player:**
   * From [VMware official site](https://www.vmware.com/).
2. **Make Installer Executable:**

bash

CopyEdit

chmod +x VMware-Workstation-Full-\*.bundle

1. **Run the Installer:**

bash

CopyEdit

sudo ./VMware-Workstation-Full-\*.bundle

1. **Follow GUI Wizard:**
   * Accept license.
   * Set preferences for network and updates.
2. **Launch VMware:**
   * From Applications Menu or:

bash

CopyEdit

vmware

1. **Create Virtual Machines:**
   * Through GUI: "Create New Virtual Machine"
   * Choose OS ISO, allocate memory, disk, and install guest OS.

**Conclusion:**

Both Xen and VMware enable **efficient use of hardware**, **secure isolation**, and **multi-OS development/testing** on a single Linux machine. Xen suits enterprise servers, while VMware excels in desktop virtualization.

**2. Briefly Discuss the Requirements to Become a Linux System Administrator.**

**[An] May/June 2015**

**Who is a Linux System Administrator?**

A **Linux System Administrator** is responsible for **installing**, **configuring**, **maintaining**, and **securing** Linux systems in an organization or server environment.

**Key Requirements:**

**1. Technical Skills:**

* **Command-Line Proficiency:**
  + Mastery of Linux shell commands (Bash)
  + File system navigation, file manipulation
* **Package Management:**
  + RPM, DPKG, APT, YUM/Zypper
* **User and Group Management:**
  + Add/delete users, set permissions, quotas
* **File Permissions and Security:**
  + chmod, chown, umask, ACLs
* **Disk Management:**
  + Mounting, partitioning, LVM, RAID setup
* **Networking:**
  + IP configuration, firewalls (iptables/firewalld), SSH, FTP
* **Service Management:**
  + systemd, starting/stopping services
* **Shell Scripting:**
  + Automate tasks using bash or Python

**2. Knowledge of Tools and Services:**

* Apache/Nginx, MySQL/PostgreSQL
* Cron jobs, rsync, tar, grep, sed, awk
* Backup and restore tools

**3. Troubleshooting & Monitoring:**

* Use top, htop, vmstat, netstat, dmesg, journalctl
* Diagnosing boot issues, service failures

**4. Certifications (Optional but Preferred):**

* **RHCSA**, **RHCE** (Red Hat)
* **LPIC-1**, **CompTIA Linux+**

**5. Soft Skills:**

* Problem-solving mindset
* Documentation and communication
* Ability to work under pressure

.