Sessional Temp topics

System Administration: Definition; Booting system; Maintaining user accounts; File systems and special files; Backups and restoration; Role and functions of a system manager, Overview of Linux operating system, Difference between LINUX and UNIX.

Show drafts

System Administration
System administration is the practice of managing and maintaining computer systems, including hardware, software, and networks. System administrators are responsible for ensuring the smooth operation of these systems, and they often work with a variety of technologies, from servers and databases to security software and networking equipment.
Booting a System
The booting process is the sequence of events that occurs when a computer system starts up. It typically involves the following steps:

1. **Power-On Self-Test (POST):** The system performs a hardware check to ensure everything is working correctly.
2. **BIOS/UEFI Bootloader:** The Basic Input/Output System (BIOS) or Unified Extensible Firmware Interface (UEFI) loads the boot loader.
3. **Boot Loader:** The boot loader loads the operating system kernel into memory.
4. **Kernel Initialization:** The kernel initializes hardware devices and system services.
5. **System Boot:** The operating system starts running, and the user interface appears.

Maintaining User Accounts
User accounts are individual logins that allow users to access system resources. System administrators are responsible for:

- **Creating User Accounts:** Assigning usernames, passwords, and permissions.
- **Modifying User Accounts:** Changing passwords, updating contact information, and adjusting permissions.
- **Deleting User Accounts:** Removing accounts that are no longer needed.

- **Managing User Groups:** Organizing users into groups to simplify permissions management.

File Systems and Special Files

A file system is a hierarchical structure for organizing and storing data on a storage device. Special files are devices or other system resources that are represented as files in the file system.

- **Common File Systems:**
  - EXT4
  - XFS
  - NTFS
  - FAT32
- **Special Files:**
  - Character devices (e.g., serial ports, modems)
  - Block devices (e.g., hard drives, SSDs)
  - Pipes
  - Sockets

Backups and Restoration

Backups are copies of data that can be used to restore the system in case of data loss. System administrators are responsible for:

- **Creating Backups:** Regularly backing up important data.
- **Storing Backups:** Storing backups securely and offsite.
- **Testing Backups:** Periodically testing backups to ensure they can be restored.
- **Restoring Backups:** Restoring data from backups in case of disasters.

Role and Functions of a System Manager

A system administrator's role involves:

- **System Installation and Configuration:** Installing and configuring hardware and software.
- **Network Administration:** Managing network infrastructure, including routers, switches, and firewalls.
- **Security:** Implementing security measures to protect the system from threats.
- **User Management:** Creating, modifying, and deleting user accounts.
- **Performance Monitoring:** Monitoring system performance and troubleshooting issues.
- **Backup and Recovery:** Implementing backup strategies and restoring data in case of failures.
- **Patch Management:** Applying software updates and security

patches.

Overview of Linux Operating System

Linux is a powerful and versatile operating system based on the UNIX kernel. It is known for its open-source nature, flexibility, and security. Linux is widely used in servers, desktops, and embedded systems.

Difference Between Linux and UNIX

While Linux is heavily influenced by UNIX, there are some key differences:

- **Licensing:** Linux is open-source, while UNIX is often proprietary.
- **User Interface:** Linux can be used with both graphical user interfaces (GUIs) and command-line interfaces (CLIs), while UNIX traditionally relied on CLIs.
- **Distribution:** Linux is available in various distributions (e.g., Ubuntu, Debian, Fedora) with different packaging systems, while UNIX is typically a single, monolithic system.
- **Community:** Linux has a large and active community that contributes to its development and support.

Shell Programming: Programming in the Borne Shell, C-Shell and Korn-Shell; Wild cards; shell programming; Shell variables; interactive shell scripts; AWK utility.

Show drafts

Shell Programming

A shell is a command-line interpreter that allows users to interact with the operating system. Shell scripts are a series of commands that can be executed sequentially or conditionally, often used to automate tasks.

Common Shells:

- **Bourne Shell (sh):** The original shell, still widely used, especially in scripts.
- **C Shell (csh):** Known for its C-like syntax, but less commonly used today.

- **Korn Shell (ksh):** Combines features of both Bourne and C shells, offering a more powerful and flexible scripting environment.

Wildcards

Wildcards are special characters used to match patterns in filenames. Common wildcards include:

- *****: Matches zero or more characters.
- **?**: Matches a single character.
- **[]**: Matches a range of characters or a specific set of characters.

Shell Programming Basics

- **Comments:** Begin with #.
- **Variables:** Declared without a specific type.
- **Control Flow:**
  - if, else, elif for conditional execution.
  - for, while, until for loops.
  - case for multiple-choice decisions.
- **Input/Output:**
  - read to read input from the user.
  - echo to print output to the console.
  - printf for formatted output.
- **Functions:** Defined using the function keyword.

Example Shell Script:

Bash

```
#!/bin/bash

echo "Enter your name:"
read name

echo "Hello, $name!"

if [ "$name" == "Alice" ]; then
  echo "Welcome, Alice!"
else
  echo "Welcome, guest!"
fi
```

Use code with caution.

Interactive Shell Scripts

Interactive shell scripts allow for user input during execution. The read command is used to prompt the user for input.

AWK Utility
AWK is a powerful text processing tool. It can be used to manipulate text files, extract information, and perform calculations.
Basic AWK Syntax:
Awk
```
BEGIN {
  # Actions to be performed before processing input
}

{
  # Actions to be performed on each line of input
}

END {
  # Actions to be performed after processing all input
}
```
Use code with caution.

Example AWK Script:
Awk
```
{
  print $2, $1  # Swap the first and second fields
}
```
Use code with caution.

This script will swap the first and second columns of each line in the input file.
Additional Tips:
- Use set -x to trace script execution.
- Use trap to handle signals and errors.
- Consider using regular expressions for complex pattern matching.
- Leverage the power of shell pipelines to combine multiple commands.

By mastering shell programming, you can automate tasks, write efficient scripts, and streamline your workflow.

Introduction of Deadlock, methods of handling, Prevention and Avoidance, Deadlock Detection, Recovery from Deadlock, Disk

Show drafts

Deadlock
A deadlock occurs when two or more processes are waiting for each other to release resources, resulting in a stalemate. This situation can arise when resources are not allocated in a specific order or when processes hold resources while waiting for others.
Methods of Handling Deadlock
There are primarily four strategies to handle deadlocks:

1. **Deadlock Prevention:**
    - **Mutual Exclusion:** If possible, avoid situations where multiple processes need exclusive access to the same resource.
    - **Hold and Wait:** Ensure that a process acquires all necessary resources before starting execution.
    - **No Preemption:** Prevent a process from holding a resource while waiting for another.
    - **Circular Wait:** Impose an ordering on resource acquisition to avoid circular dependencies.
2. **Deadlock Avoidance:**
    - **Resource Allocation Graph:** Model the system as a graph and use algorithms to determine if a request for resources might lead to a deadlock.
    - **Banker's Algorithm:** A resource allocation algorithm that ensures that a safe state is maintained.
3. **Deadlock Detection:**
    - **Periodically check the system:** Identify cycles in the resource allocation graph.
    - **Detect deadlocks when they occur:** Use algorithms to detect deadlocks and take corrective action.
4. **Deadlock Recovery:**

    - **Process Termination:** Terminate one or more processes involved in the deadlock.
    - **Resource Preemption:** Forcibly take away resources from processes.
    - **Rollback:** Roll back processes to a previous safe state.

Disk

A disk is a non-volatile storage device used to store and retrieve data. It is typically composed of one or more platters coated with a magnetic material. Data is stored on the disk surface in the form of magnetically polarized regions.

Types of Disks:

- **Hard Disk Drives (HDDs):** Traditional mechanical storage devices that use magnetic disks to store data.
- **Solid-State Drives (SSDs):** Modern storage devices that use solid-state memory to store data, offering faster performance and greater reliability.

Disk Scheduling Algorithms:

Disk scheduling algorithms are used to optimize disk performance by determining the order in which disk requests are serviced. Some common algorithms include:

- **First-Come, First-Served (FCFS):** Processes disk requests in the order they arrive.
- **Shortest Seek Time First (SSTF):** Processes the request with the shortest seek time next.
- **SCAN:** Scans the disk from one end to the other, servicing requests along the way.
- **C-SCAN:** Similar to SCAN, but the disk head moves in one direction only.
- **LOOK:** Similar to SCAN, but the disk head moves only to the last request in each direction.
- **C-LOOK:** Similar to C-SCAN, but the disk head moves only to the last request in each direction.

By understanding these concepts, you can effectively manage and optimize disk systems in various computing environments.

Real-Time Scheduling. File Structure; Memory Management: Swapping, Demand paging, Virtual Memory, Critical Section Problem, Mutual Exclusion Problem.

Show drafts

Programming
Programming is the process of designing, writing, testing, and debugging computer programs. It involves using programming languages to create instructions that a computer can understand and execute.

Scheduling Criteria
Scheduling criteria are used to determine the order in which processes are executed on a processor. Common criteria include:

- **CPU Utilization:** Maximize CPU usage.
- **Throughput:** Maximize the number of processes completed per unit time.
- **Turnaround Time:** Minimize the time taken to complete a process.
- **Waiting Time:** Minimize the time a process spends waiting for the CPU.
- **Response Time:** Minimize the time it takes for a process to respond to a user input.

Multiple Processor Scheduling
Multiple processor scheduling involves assigning processes to multiple processors to improve performance. Common techniques include:

- **Process Scheduling:** Assigning processes to processors.
- **Load Balancing:** Distributing the workload evenly among processors.
- **Cache Coherence:** Ensuring that all processors have access to the same data.

Real-Time Scheduling
Real-time scheduling is used for systems where tasks must be completed within specific deadlines. It involves assigning priorities to tasks and ensuring that high-priority tasks are executed before low-priority tasks.

File Structure
A file system is a hierarchical structure for organizing and storing data

on a storage device. Common file structures include:

- **Hierarchical File System:** A tree-like structure with directories and files.
- **Network File System (NFS):** A distributed file system that allows multiple clients to access files on a server.

Memory Management

Memory management is the process of allocating and deallocating memory to processes. Common techniques include:

- **Swapping:** Moving processes between main memory and secondary storage.
- **Demand Paging:** Loading pages into memory only when they are needed.
- **Virtual Memory:** Creating a virtual address space that is larger than physical memory.

Critical Section Problem

The critical section problem arises when multiple processes access shared resources simultaneously. To prevent race conditions and ensure data consistency, a mechanism is needed to ensure that only one process can access the critical section at a time.

Mutual Exclusion Problem

The mutual exclusion problem is similar to the critical section problem. It involves designing a protocol to ensure that only one process can access a shared resource at a time.

By understanding these concepts, you can effectively design, implement, and optimize computer systems.