# Linux Operating System (4331602) - Winter 2023 Solution

Milav Dabgar

January 16, 2024

## Question 1(a) [3 marks]

**Draw the architecture of Linux and explain various layers in brief.**

### Solution

**Answer**:
**Linux Architecture:**

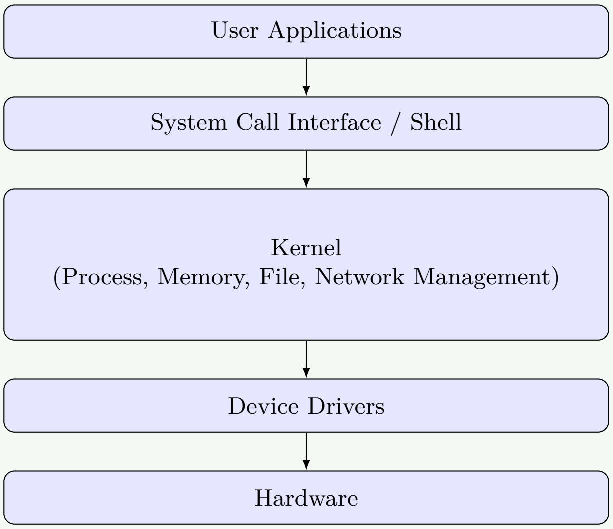
User Applications
System Call Interface / Shell
Kernel
(Process, Memory, File, Network Management)
Device Drivers
Hardware

**Figure 1.** Linux Architecture

**Layers:**

- **User Space**: Contains user applications and system utilities.
- **System Call Interface**: Provides interface between user programs and kernel.
- **Kernel Space**: Core operating system responsible for resource management.
- **Hardware**: Physical components of the computer system.

### Mnemonic

"USKDH - Users System Kernel Drives Hardware"

## Question 1(b) [4 marks]

**What is a race condition? Explain with a suitable example.**

Solution

**Answer**:

**Table 1.** Race Condition

| Aspect | Description |
|---|---|
| Definition | Situation where multiple processes access shared resources simultaneously |
| Problem | Unpredictable results depending on execution order |
| Example | Bank account balance update by simultaneous transactions |

**Example Scenario:**

1. **Process A**: Reads balance = 1000, adds 100 (ready to write 1100).
2. **Process B**: Reads balance = 1000 (before A writes), subtracts 50 (ready to write 950).
3. **Result**: Final balance could be 1100 or 950, instead of correct 1050, depending on who writes last.

Mnemonic

"RRRR - Race Results Random Resources"

## Question 1(c) [7 marks]

**List different types of Operating systems. Explain the working of multiprogramming operating systems with a suitable example.**

Solution

**Answer**:

**Types of Operating Systems:**

**Table 2.** Types of Operating Systems

| Type | Characteristics |
|---|---|
| Batch | Jobs processed in groups without user interaction |
| Time-sharing | Multiple users share system simultaneously |
| Real-time | Strict time constraints for operations |
| Distributed | Computations distributed among networked processors |
| Multiprogramming | Multiple programs kept in memory for CPU utilization |

**Multiprogramming Working:**

- **Memory Management**: Multiple jobs are kept in main memory simultaneously.
- **CPU Utilization**: When one job waits for I/O, CPU switches to another job.
- **Context Switching**: OS saves state of current job and loads state of next job.

**Example**: A user running a web browser, music player, and word processor simultaneously. While browser waits for network data, CPU executes music player instructions.

Mnemonic

"MPMP - Multiple Programs Maximize Performance"

## Question 1(c) OR [7 marks]

**List different types of Operating systems. Explain the Batch operating systems in detail.**

**Solution**

**Answer**:

**Types of Operating Systems:** (Same as above table)

**Batch Operating System:**

- **Job Collection**: Users submit jobs (program + data + control info) offline.
- **Batching**: Operator groups similar jobs into batches to speed up processing.
- **Sequential Execution**: CPU executes jobs in a batch one after another.
- **No Interaction**: User cannot interact with the job during execution.

**Advantages and Disadvantages:**

- **Pros**: High CPU utilization for similar jobs, simple to manage.
- **Cons**: Long turnaround time, difficult to debug, no real-time capability.

**Mnemonic**

"BBBB - Batch Brings Better Business"

## Question 2(a) [3 marks]

**Draw and explain the Process life cycle.**

**Solution**

**Answer**:

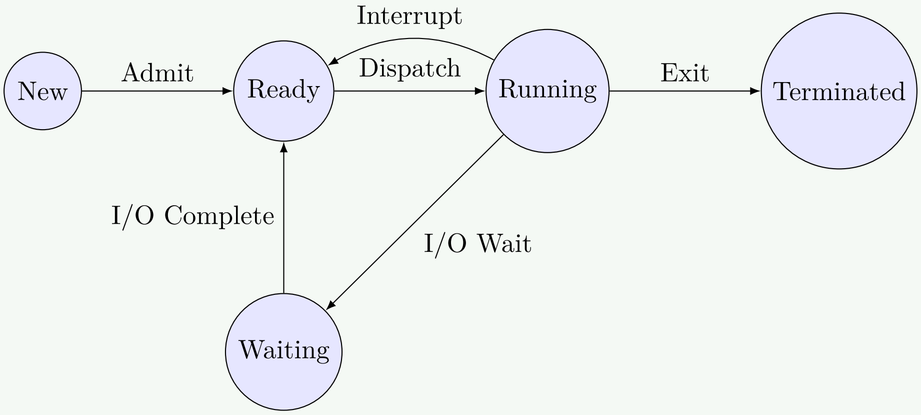

**Figure 2.** Process State Diagram

**States:**

- **New**: Process is being created.
- **Ready**: Process is waiting to be assigned to a processor.
- **Running**: Instructions are being executed.
- **Waiting**: Process is waiting for some event (I/O).
- **Terminated**: Process has finished execution.

**Mnemonic**

"NRWRT - New Ready Waiting Running Terminated"

## Question 2(b) [4 marks]

**Define deadlock and discuss necessary conditions for a deadlock to occur.**

### Solution

**Answer**:
**Deadlock**: A situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.
**Coffman Conditions (All must hold):**

1. **Mutual Exclusion**: At least one resource must be held in a non-shareable mode.
2. **Hold and Wait**: A process holds a resource while waiting for another.
3. **No Preemption**: Resources cannot be forcibly taken from a process.
4. **Circular Wait**: A set of processes $\{P0, P1, \ldots, Pn\}$ exists such that $P0$ waits for $P1$, $P1$ waists for $P2$, ..., $Pn$ waits for $P0$.

### Mnemonic

"MHNC - My Hold Never Circles"

## Question 2(c) [7 marks]

**Describe the Round Robin algorithm. Calculate the average waiting time & average turnaround time along with Gantt chart for the given data. Consider context switch = 01 ms and quantum time = 05 ms.**

### Solution

**Answer**:
**Round Robin Algorithm:** Preemptive scheduling where each process gets a small unit of CPU time (time quantum).
**Given Data:** Context Switch = 1ms, Quantum = 5ms.
**Processes:**

| Process | Arrival | Burst |
|---|---|---|
| P1 | 0 | 12 |
| P2 | 3 | 4 |
| P3 | 2 | 15 |
| P4 | 5 | 5 |

**Gantt Chart:**

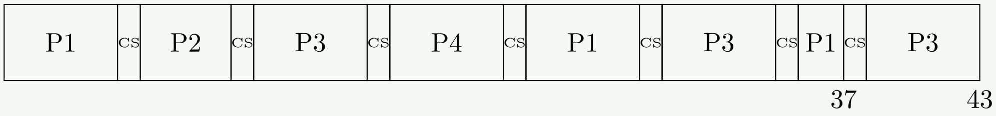

**Figure 3.** Gantt Chart (RR)

*Note: Calculated based on problem statement logic. P1(5) → P2(4) → P3(5) → P4(5) → P1(5) → P3(5) → P1(2) → P3(5).*
**Calculations:**

| **Process** | **Completion** | **Turnaround** | **Waiting** |
|---|---|---|---|
| P1 | 37 | 37 - 0 = 37 | 37 - 12 = 25 |
| P2 | 10 | 10 - 3 = 7 | 7 - 4 = 3 |
| P3 | 43 | 43 - 2 = 41 | 41 - 15 = 26 |
| P4 | 22 | 22 - 5 = 17 | 17 - 5 = 12 |

**Average Waiting Time** = (25+3+26+12)/4 = 16.5 ms **Average Turnaround Time** = (37+7+41+17)/4 = 25.5 ms

**Mnemonic**

"RRRR - Round Robin Rotates Regularly"

## Question 2(a) OR [3 marks]

### Differentiate: CPU bound process v/s I/O bound process.

**Solution**

**Answer**:

**Table 3.** CPU vs I/O Bound

| **Aspect** | **CPU Bound** | **I/O Bound** |
|---|---|---|
| Activity | High CPU computations | Frequent I/O operations |
| Burst Time | Long CPU bursts | Short CPU bursts |
| Wait States | Less frequent | Frequent waiting for I/O |
| Examples | Scientific calculation | Data processing, File copy |

**Mnemonic**

"CIC - CPU Computes I/O Interacts"

## Question 2(b) OR [4 marks]

### Define Critical Section and discuss the general structure of a critical section solution.

**Solution**

**Answer**:
**Critical Section**: Code segment where shared resources (variables, files) are accessed. Only one process should execute in CS at a time.
**General Structure:**

```
do {
    entry section   // Request permission
       critical section
    exit section    // Release permission
       remainder section
} while (true);
```

**Requirements:**

- **Mutual Exclusion**: Only one process in CS.
- **Progress**: If CS is empty, selection of next process cannot be postponed indefinitely.
- **Bounded Waiting**: Waiting time for entry must be limited.

**Mnemonic**

"ECER - Entry Critical Exit Remainder"

## Question 2(c) OR [7 marks]

### Describe the SJF algorithm. Calculate the average waiting time and average turn-around time along with Gantt chart for the given data.

Solution

**Answer**:
**Shortest Job First (SJF)**: Non-preemptive algorithm where process with shortest CPU burst is scheduled first.
**Execution Order**: P1(8), P2(4), P3(9), P4(5). Order based on arrival and burst: 1. t=0, P1 arrives (8). Runs 0-8. 2. t=8, P2(4), P3(9), P4(5) available. P2 shortest. Runs 8-12. 3. t=12, P4(5), P3(9) avail. P4 shortest. Runs 12-17. 4. t=17, P3(9) runs. Runs 17-26.
**Gantt Chart (SJF):**

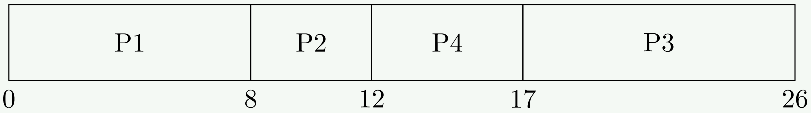

**Figure 4.** Gantt Chart (SJF)

**Calculations:**

| Process | Arr | Burst | Comp | TAT | Wait |
|---|---|---|---|---|---|
| P1 | 0 | 8 | 8 | 8 | 0 |
| P2 | 3 | 4 | 12 | 9 | 5 |
| P4 | 6 | 5 | 17 | 11 | 6 |
| P3 | 5 | 9 | 26 | 21 | 12 |

**Avg Wait** $= (0 + 5 + 6 + 12)/4 = 5.75$ ms **Avg TAT** $= (8 + 9 + 11 + 21)/4 = 12.25$ ms

Mnemonic

"SJSS - Shortest Jobs Start Soon"

## Question 3(a) [3 marks]

### Explain two-level directory structure.

Solution

**Answer**:

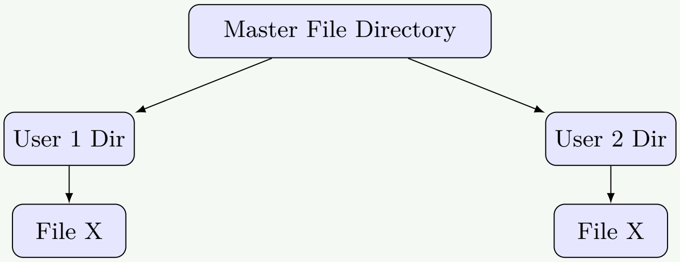

**Figure 5.** Two-level Directory

**Features:**
- Separate directory for each user (UFD).
- Solves name collision problem (different users can have same filenames).
- Provides isolation between users.

**Mnemonic**

"TTTT - Two Tiers Tackle Troubles"

## Question 3(b) [4 marks]

### Explain the different file operations.

**Solution**

**Answer**:

**Table 4.** File Operations

| **Operation** | **Description** |
|---|---|
| Create | Allocates space and creates directory entry |
| Open | Loads file metadata into memory for access |
| Read | Reads data from current position |
| Write | Writes data to current position |
| Delete | Releases space and removes directory entry |
| Close | Frees internal resources |

**Mnemonic**

"CORWCD - Create Open Read Write Close Delete"

## Question 3(c) [7 marks]

### List the different file allocation methods and explain contiguous allocation with necessary diagram.

**Solution**

**Answer**:
**Methods**: Contiguous, Linked, Indexed.
**Contiguous Allocation:** File occupies set of consecutive blocks on disk. Directory entry stores start block and length.

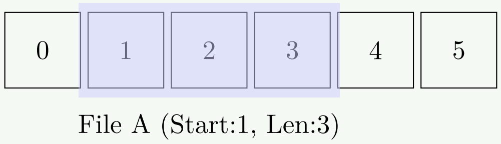

**Figure 6.** Contiguous Allocation

**Pros**: Simple (start, length), High performance (fast sequential access). **Cons**: External fragmentation, Difficult to grow file.

**Mnemonic**

"CCCC - Contiguous Creates Continuous Clusters"

## Question 3(a) OR [3 marks]

### Describe the types of file structures.

Solution

**Answer**:

- **Sequential**: Records stored in order. Simple but slow search.
- **Direct/Random**: Records accessed by key/index. Fast access.
- **Indexed**: Separate index file points to data records.

Mnemonic

"SDI - Sequential Direct Indexed"

## Question 3(b) OR [4 marks]

### Explain the different file attributes.

Solution

**Answer**:

**Table 5.** File Attributes

| **Attribute** | **Description** |
|---|---|
| Name | Human-readable identifier |
| Type | Format of file (.txt, .exe) |
| Size | Current file size |
| Location | Pointer to file location on device |
| Protection | Access control info (R/W/X) |
| Time/Date | Info for creation, mod, usage |

Mnemonic

"NTSLPT - Name Type Size Location Permissions Time"

## Question 3(c) OR [7 marks]

### List the different file allocation methods and explain linked allocation with necessary diagram.

Solution

**Answer**:
**Linked Allocation:** Files stored in non-contiguous blocks. Each block contains pointer to next block.

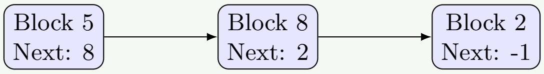

**Figure 7.** Linked Allocation

**Directory Entry**: Stores pointer to first and last blocks.
**Pros**: No external fragmentation, easy file growth. **Cons**: Slow random access (must traverse chain), pointer

overhead.

Mnemonic

"LLLL - Links Lead Logical Locations"

## Question 4(a) [3 marks]

**Define Program threats and explain its types.**

Solution

**Answer**:
**Program Threats**: Malicious code embedded in program.

- **Trojan Horse**: Appears useful but does damage (steals login).
- **Trap Door**: Secret entry point left by designer.
- **Logic Bomb**: Code that explodes (executes) under specific conditions.
- **Virus**: Code that embeds itself into other programs.

Mnemonic

"TTLV - Trojan Trap Logic Virus"

## Question 4(b) [4 marks]

**Explain System Authentication.**

Solution

**Answer**:
**Authentication**: Verification of user identity.
**Methods:**

1. **Passwords**: Secret string known to user.
2. **Biometrics**: Fingerprint, retina scan, face ID.
3. **Smart Cards**: Physical token with embedded chip.
4. **Two-Factor**: Combining two methods (e.g., Password + OTP).

Mnemonic

"PBST - Passwords Biometrics Smartcards Two-factor"

## Question 4(c) [7 marks]

**Explain Access Control List in detail.**

Solution

**Answer**:
**Access Control List (ACL)**: A list associated with each object (file/resource) specifying which domains (users/processes) can access it and how.
**Structure:** File X: $(UserA, Read), (UserB, Read/Write), (GroupC, Execute)$
**Pros**:

- Precise control over individual objects.
- Easy to revoke permissions for specific users.

**Cons**:

- Searching ACL can be slow.
- Managing ACLs for many files is complex.

**Mnemonic**

"ACLU - Access Controls Limit Users"

## Question 4(a) OR [3 marks]

**Define System threats and explain its types.**

**Solution**

**Answer**:

**System Threats**: Target the environment/OS itself.

- **Worm**: Independent program that spreads through networks consuming resources.
- **Port Scanning**: Detecting open ports to find vulnerabilities.
- **Denial of Service**: Overwhelming system to prevent legitimate use.

**Mnemonic**

"WPD - Worm Port DoS"

## Question 4(b) OR [4 marks]

**Discuss the needs and goals of protection in OS.**

**Solution**

**Answer**:

**Needs:**

- Prevent malicious misuse of system.
- Ensure resources are used fairly.
- Protect user data integrity and confidentiality.

**Goals:**

- **Availability**: Resources available to auth users.
- **Integrity**: Data not modified unauthorizedly.
- **Confidentiality**: Data not viewed unauthorizedly.

**Mnemonic**

"CIA - Confidentiality Integrity Availability"

## Question 4(c) OR [7 marks]

**Discuss various operating system security policies and procedures.**

**Solution**

**Answer**:

**Policies:**

- **User Policy**: Strong passwords, regular changes.
- **Access Policy**: Least privilege principle.
- **Data Policy**: Encryption of sensitive data.

**Procedures:**

- **Auditing**: Monitoring logs for suspicious activity.
- **Backups**: Regular data backup for recovery.
- **Updates**: Patching OS to fix vulnerabilities.
- **Intrusion Detection**: Systems to detect attacks in real-time.

**Mnemonic**

"APPI - Access Password Policy Incident"

## Question 5(a) [3 marks]

**Explain the following commands: (i) pwd (ii) cd (iii) comm**

**Solution**

**Answer**:

**Table 6.** Commands

| Command | Purpose |
|---|---|
| `pwd` | Print Working Directory. Shows current path. |
| `cd` | Change Directory. Navigate to different folder. |
| `comm` | Compare two sorted files line by line. |

**Mnemonic**

"PCC - Pwd Cd Comm"

## Question 5(b) [4 marks]

**Write a shell script to concatenate the contents of two files in a third file.**

**Solution**

**Listing 1.** Concatenate Files

```bash
#!/bin/bash
# Script to concatenate two files

echo "Enter first filename:"
read f1
echo "Enter second filename:"
read f2
echo "Enter output filename:"
read f3

if [ -f "$f1" ] && [ -f "$f2" ]; then
```

```
    cat "$f1" "$f2" > "$f3"
    echo "Files merged into $f3"
else
    echo "Files not found"
fi
```

**Mnemonic**

"CCCC - Cat Combines Content Correctly"

## Question 5(c) [7 marks]

**Write a shell script to find the sum of all the individual digits in a given 5 digit number.**

**Solution**

**Listing 2.** Sum of Digits

```
#!/bin/bash
# Sum of 5 digits

echo "Enter 5 digit number:"
read n

if [ ${#n} -ne 5 ]; then
    echo "Please enter 5 digits"
    exit 1
fi

sum=0
while [ $n -gt 0 ]
do
    rem=$((n % 10))
    sum=$((sum + rem))
    n=$((n / 10))
done

echo "Sum of digits: $sum"
```

**Mnemonic**

"SSSS - Sum Separates Single Symbols"

## Question 5(a) OR [3 marks]

**Explain the following commands: (i) man (ii) mkdir (iii) grep**

**Solution**

**Answer**:

**Table 7.** More Commands

| Cmd | Purpose |
|---|---|
| `man` | Manual. Displays help/manual for commands. |
| `mkdir` | Make Directory. Creates new folder. |
| `grep` | Global Regular Expression Print. Search text in files. |

Mnemonic

"MMG - Manual Make Grep"

## Question 5(b) OR [4 marks]

**Write a shell script to generate and display Fibonacci series.**

Solution

**Listing 3.** Fibonacci Series

```bash
#!/bin/bash
# Fibonacci Series

echo "Enter N:"
read n
a=0
b=1

echo -n "$a $b "

for (( i=0; i<n-2; i++ ))
do
    c=$((a + b))
    echo -n "$c "
    a=$b
    b=$c
done
echo ""
```

Mnemonic

"FFFF - Fibonacci Follows Forward Formula"

## Question 5(c) OR [7 marks]

**Write a shell script to determine whether a given string is palindrome.**

Solution

**Listing 4.** Palindrome Check

```bash
#!/bin/bash
# Palindrome Check

echo "Enter string:"
read str
```

```
len=${#str}
rev=""

for (( i=$len-1; i>=0; i-- ))
do
    rev="$rev${str:$i:1}"
done

if [ "$str" == "$rev" ]; then
    echo "Palindrome"
else
    echo "Not Palindrome"
fi
```

### Mnemonic

"PPPP - Palindromes Proceed Perfectly Parallel"