Lab 1: Foundational Concepts: Number Systems, Date Formats, ASCII Codes, Disk Drives, and Computer Forensics

Objective:

This lab will introduce you to essential concepts in digital forensics, including number system conversions, date formats, ASCII code understanding, disk drive structure, file systems, and basic system verification using Linux commands. By the end of this lab, you will have a deeper understanding of the low-level operations of computers, which are crucial for digital forensics.

Part 1: Number Systems and Conversion

1. Binary to Decimal Conversion:

- Convert the binary number 1010111101 to its decimal equivalent.
- Hint: Use the positional value method to calculate the decimal equivalent of the binary number.

Binary Number: 1010111101

$1010111101_2 = (1\times2^9)+(0\times2^8)+(1\times2^7)+(0\times2^6)+(1\times2^5)+(1\times2^4)+(1\times2^3)+(0\times2^2)+(1\times2^1)+(1\times2^0)$

512+0+128+0+32+16+8+0+2+1=699

Decimal = 699

Question:

Explain the step-by-step process of converting a binary number to decimal. Provide another binary number (11011011) as an example for practice.

11011011 to decimal

$11011011_2 = (1\times2^7)+(1\times2^6)+(0\times2^5)+(1\times2^4)+(1\times2^3)+(0\times2^2)+(1\times2^1)+(1\times2^0)$

= 128+64+0+16+8+0+2+1=219

2. Hexadecimal Conversions:

- Convert the hexadecimal number 0xABCD to its decimal equivalent.
  0x ABCD = $(A*16^3) + (B*16^2) + (C*16^1) + (D*16^0)$
  - A= 10 * 4096 = 40960
  - B= 11 * 256 =2816
  - C= 12 * 16 = 192
  - D=13* 1=13

  40960+2816+2816+192+13

  =43981

Question:

Discuss the importance of hexadecimal notation in computing. Provide an example where hexadecimal notation is advantageous over decimal (e.g., in memory addressing or representing RGB colour values).

Importance of Hexadecimal Notation in Computing

- Compact representation of binary
  - 1 hex digit = 4 binary bits.
  - Example: $11111111_2 = FF_{16} = 255_{10}$.
- Memory addressing
  - Large memory addresses are easier to read in hex.
  - Example:
    - Decimal: 2147418112
    - Hex: 0x7FFF0000.
- File signatures (digital forensics)
  - Many files begin with hex "magic numbers."
  - Examples:
    - JPEG → FFD8
    - PDF → 25504446 (represents %PDF).
- Colour representation (RGB values)
  - Web and graphics use hex notation.
  - Example:
    - Red: #FF0000
    - Green: #00FF00
    - Blue: #0000FF.
- Low-level programming and debugging
  - Machine code, opcodes, and error codes are written in hex.
  - Example: instruction B8 04 00 in assembly.

Example where hex is better than decimal:

- RGB colour “sky blue”:
  - Decimal: (135, 206, 235)
  - Hex: #87CEEB → shorter, standard, and directly maps to binary.

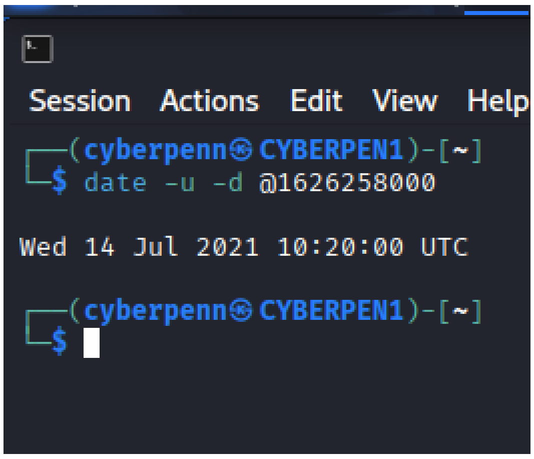
Part 2: Date Formats and Conversion

1. Unix Timestamp Conversion:

- Convert the Unix timestamp 1626258000 to a human-readable date format (in UTC).
- Hint: You can use an online converter, a Python script, or a Linux command (date).

Convert: 1626258000

Command: date -u -d @1626258000



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ date -u -d @1626258000

Wed 14 Jul 2021 10:20:00 UTC

┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ 
```

Question: Describe the Unix epoch and its significance in computing time.

- The Unix epoch (1970-01-01 00:00:00 UTC) is the baseline for measuring time in most computing systems. Its significance lies in providing a universal, consistent, and efficient method for storing and processing time, free from human formatting and time zone complications.
- Significance in Computing
  - Cross-platform standard: Used in Unix, Linux, macOS, many databases, and programming languages (C, Python, Java, etc.).
  - Simplicity in calculations: Easy to add/subtract seconds to compute durations.
    - Example: 86400 seconds = 1 day.
  - Digital forensics & logging: File systems (EXT4, NTFS), logs, and databases store times as Unix timestamps, enabling investigators to reconstruct events accurately.
  - Web and APIs: Many APIs (Twitter, GitHub, etc.) use Unix timestamps to mark events consistently across systems worldwide.

2. Date Format Differences:

- Explain the differences between the date formats MM/DD/YYYY and YYYYMM-DD.

1.

2. MM/DD/YYYY
    - Example: 07/14/2021
    - Meaning: Month / Day / Year
    - Used mainly in the United States.
    - Can cause confusion internationally (since some regions use DD/MM/YYYY).
3. YYYY-MM-DD (ISO 8601 standard)
    - Example: 2021-07-14
    - Meaning: Year - Month - Day
    - Used in international standards (ISO 8601).
    - Common in databases, programming, and APIs.

Differences

- Regional Use
    - MM/DD/YYYY → common in the US.
    - DD/MM/YYYY → common in Europe, Africa, Asia.
    - YYYY-MM-DD → universal technical standard, avoids confusion.
- Sorting & Databases
    - YYYY-MM-DD is lexicographically sortable.
        - Example: 2021-07-14 < 2022-01-01 (alphabetical sort = chronological order).
    - MM/DD/YYYY does not sort correctly as text (e.g., 12/31/2020 might come before 01/01/2021).
- Programming Standards
    - Most programming languages and databases (SQL, JSON, XML) recommend or enforce YYYY-MM-DD (ISO 8601).
    - MM/DD/YYYY is mainly for user interfaces in US-based applications.

- **Question:** When would you use each format in programming and why? Consider regional differences, programming standards, and sorting functionality in databases.
- **When to Use Each Format**
    - Use MM/DD/YYYY when:

    - Designing applications for US-based users, where this is the familiar cultural standard.
    - Example: A consumer-facing app for American users.
- Use YYYY-MM-DD when:
    - Programming, databases, APIs, and international systems.
    - Ensures universal readability, avoids ambiguity.
    - Ideal for sorting, data storage, and machine processing.
    - Example: SQL databases, JSON files, or forensic timestamp storage.

Part 3: ASCII Code Understanding

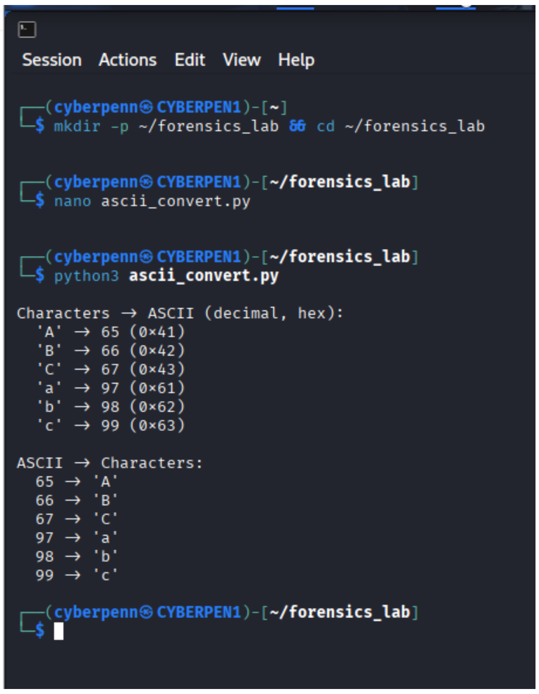
1. ASCII Table Creation:

- Create a table listing the ASCII codes for characters A, B, C, a, b, c.

Character	ASCII Decimal Code	ASCII Hex Code	ASCII Binary Code
A	65	0x41	01000001
B	66	0x42	01000010
C	67	0x43	01000011
a	97	0x61	01100001
b	98	0x62	01100010
c	99	0x63	01100011

2. Python ASCII Code Conversion:

- Write a Python script that converts ASCII codes to characters and vice versa for the characters listed above.



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ mkdir -p ~/forensics_lab && cd ~/forensics_lab

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ nano ascii_convert.py

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ python3 ascii_convert.py
Characters → ASCII (decimal, hex):
  'A' → 65 (0x41)
  'B' → 66 (0x42)
  'C' → 67 (0x43)
  'a' → 97 (0x61)
  'b' → 98 (0x62)
  'c' → 99 (0x63)

ASCII → Characters:
  65 → 'A'
  66 → 'B'
  67 → 'C'
  97 → 'a'
  98 → 'b'
  99 → 'c'

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ ▌
```

3. Extended ASCII Codes:

- Research and list ASCII codes for extended characters beyond the standard English alphabet (e.g., symbols or foreign language characters).

Character	**ASCII Code (Decimal)**	**Hex**
Ç	128	0x80
ü	129	0x81
é	130	0x82
â	131	0x83
ä	132	0x84
à	133	0x85
å	134	0x86
ç	135	0x87
ê	136	0x88
ë	137	0x89
è	138	0x8A
ñ	164	0xA4
º (masculine ordinal)	167	0xA7
± (plus/minus)	241	0xF1
÷ (division sign)	246	0xF6
© (copyright)	184	0xB8
ß (sharp S)	225	0xE1

Question:

Explain how extended ASCII characters are used in computing, such as encoding special symbols in file names or supporting internationalization.

How Extended ASCII Characters Are Used in Computing

- Encoding special symbols in files and applications
  - Extended ASCII provides characters such as ©, ®, ±, ÷, and ° that are not available in standard ASCII.
  - These symbols are used in file names, documents, and legacy applications where special notation is required.
- Internationalization (i18n) and localization (l10n)
  - Extended ASCII adds accented letters and foreign characters (e.g., é, ñ, ü, ç).
  - This allowed early computers and operating systems to support multiple languages before Unicode became the standard.
  - For example: French ("école"), Spanish ("niño"), German ("straße").
- Legacy software and compatibility
  - Before Unicode, many programs, especially on MS-DOS and early Windows, relied on extended ASCII tables (like Code Page 437 or 850).
  - Old databases, logs, and software still store text using extended ASCII, making it relevant in digital forensics.
- Graphical text interfaces
  - Box-drawing and line characters (║, ═, ╬) were included in extended ASCII.
  - These were heavily used in command-line menus, tables, and user interfaces on DOS systems before GUI environments became common.

Part 4: Disk Drives and File Systems

1. Disk Geometry Calculations:

- Given the following disk parameters:
  - Sectors per track: 400
  - Number of heads: 12
  - Cylinders: 17,000

Tasks:

- a. Calculate the total number of sectors in the disk.

  Total Sectors =Cylinders×Heads×Sectors per Track

=17,000×12×400=

Total Sectors = 81,600,000 sectors

b. Discuss how disk geometry impacts disk performance and storage capacity.

Discussion: How disk geometry impacts performance & capacity

- Sectors per track
  - More sectors per track = higher capacity per cylinder.
  - However, denser tracks may increase risk of read/write errors if hardware precision is low.
- Number of heads
  - More heads = more platters (sides) available for data storage.
  - Increases capacity, but also increases mechanical complexity and seek times.
- Cylinders
  - More cylinders = more tracks vertically (across platters).
  - Increases total storage capacity.
- Performance Impact
  - Seek time depends on how quickly the head can move between cylinders.
  - Rotational latency depends on how many sectors per track and disk RPM.
  - Optimizing geometry helps balance capacity vs. performance.
- Modern Context
  - Today's disks use Logical Block Addressing (LBA) instead of Cylinder-Head-Sector (CHS).
  - But in digital forensics, understanding geometry is still useful for interpreting older storage devices and low-level disk analysis.

2. File Systems Overview:

File System = structure that organizes and manages how data is stored, accessed, and retrieved on a disk.

Common File Systems:

- FAT32 (File Allocation Table)
  - Simple, widely supported (USBs, SD cards).
  - Limitations: max file size = 4 GB, partition size = 2 TB.
- NTFS (New Technology File System – Windows)

    - Supports permissions, encryption, compression, journaling.
    - Used in most Windows systems, reliable for large volumes.
- EXT3/EXT4 (Linux)
    - Journaling file systems (better recovery after crashes).
    - EXT4 supports large files and high performance, standard in modern Linux.
- HFS+ / APFS (Apple)
    - HFS+ (older macOS), APFS (newer macOS & iOS).
    - APFS optimized for SSDs, encryption, and snapshots.
- exFAT
    - Designed for flash drives.
    - Supports large files (>4 GB) without the limitations of FAT32

Discussion Point:

Highlight their advantages and disadvantages, such as speed, security, and compatibility.

Discussion: Advantages & Disadvantages of FAT32, NTFS, and EXT4, FAT32

Advantages:

- Speed: Fast on small partitions and devices with low overhead.
- Compatibility: Works across almost all operating systems (Windows, Linux, macOS) and devices (USBs, cameras, embedded systems).
- Simplicity: Easy to implement, reliable for small-scale use.

Disadvantages:

- Security: No built-in permissions, encryption, or journaling → prone to corruption and unauthorized access.
- File size limit: Maximum file size = 4 GB; partition size = 2 TB.
- Performance: Slower on large disks due to outdated structure.

NTFS

Advantages:

- Security: Strong support for file permissions, encryption (EFS), and compression.

- Reliability: Journaling and MFT structure reduce corruption and enable faster recovery.
- Scalability: Supports very large files and partitions (>16 TB).

Disadvantages:

- Compatibility: Full support mainly in Windows. Linux/macOS need extra drivers to write NTFS.
- Overhead: More complex structure can slightly reduce raw speed compared to FAT32.

EXT4

Advantages:

- Speed: Optimized for Linux performance with features like extents and delayed allocation.
- Security & reliability: Journaling, permissions, and better crash recovery.
- Scalability: Handles very large files (up to 16 TB) and volumes (1 EB).

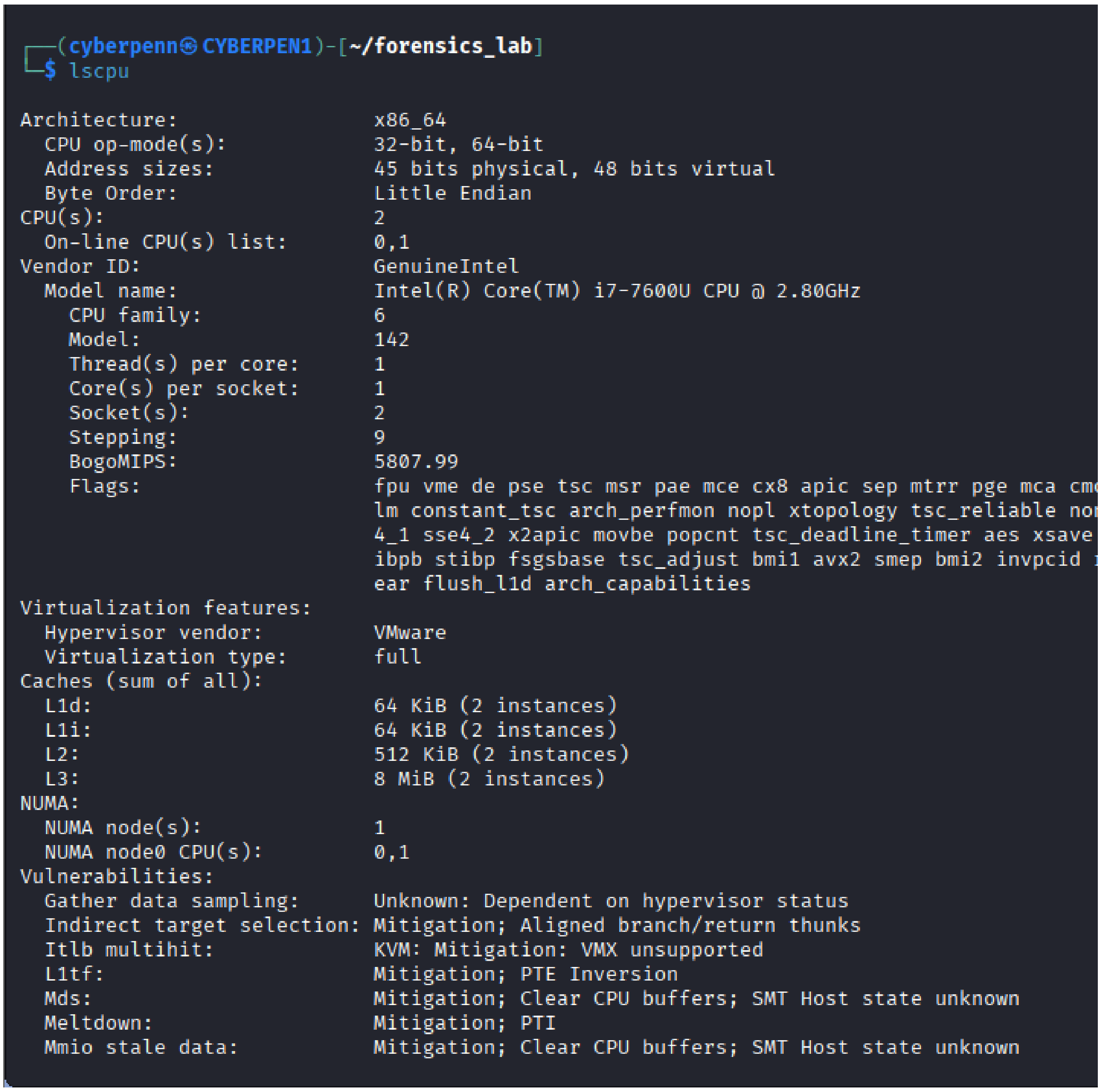
Disadvantages:

- Compatibility: Poor native support outside Linux. Windows/macOS need third-party drivers.
- Complexity: More advanced structure may be unnecessary for small removable drives.

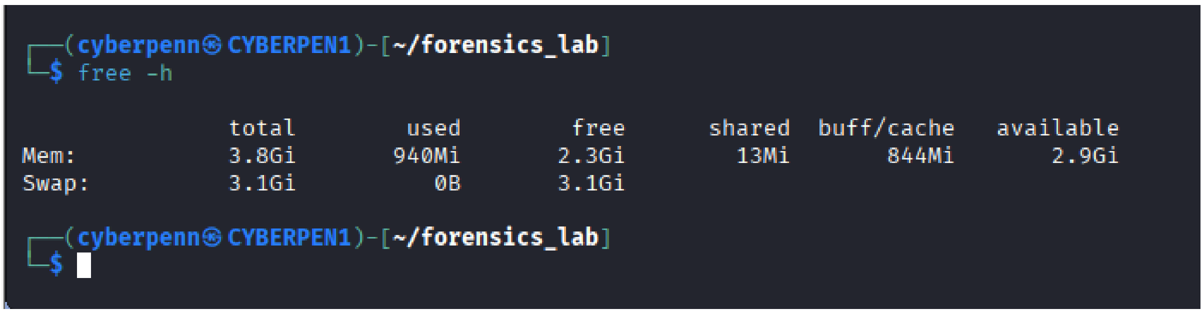
Part 5: Computer Forensics and System Verification

1. System Verification Using Linux Commands:

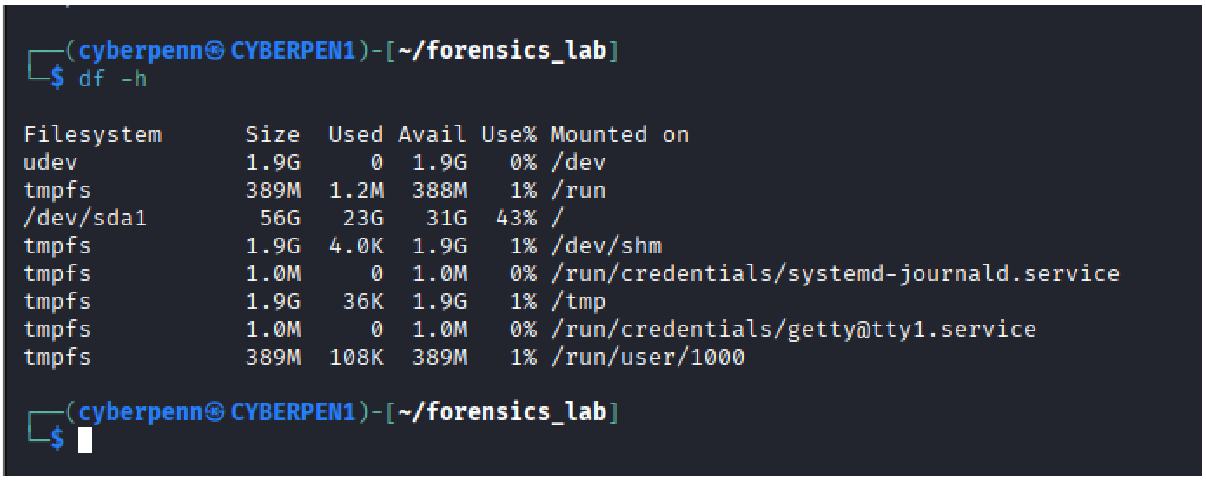
- Use Linux commands (lscpu, df, free, etc.) to verify the specifications of your computer system, including CPU details, memory, and storage.



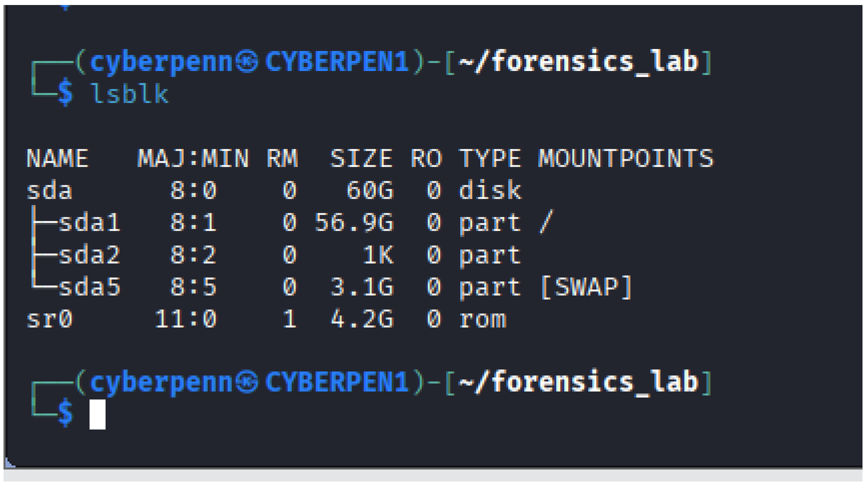
```
┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ lscpu

Architecture:             x86_64
  CPU op-mode(s):         32-bit, 64-bit
  Address sizes:          45 bits physical, 48 bits virtual
  Byte Order:             Little Endian
CPU(s):                   2
  On-line CPU(s) list:    0,1
Vendor ID:                GenuineIntel
  Model name:             Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz
    CPU family:           6
    Model:                142
    Thread(s) per core:   1
    Core(s) per socket:   1
    Socket(s):            2
    Stepping:             9
    BogoMIPS:             5807.99
    Flags:                fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cm
                          lm constant_tsc arch_perfmon nopl xtopology tsc_reliable no
                          4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave
                          ibpb stibp fsgsbase tsc_adjust bmi1 avx2 smep bmi2 invpcid
                          ear flush_l1d arch_capabilities
Virtualization features:
  Hypervisor vendor:      VMware
  Virtualization type:    full
Caches (sum of all):
  L1d:                    64 KiB (2 instances)
  L1i:                    64 KiB (2 instances)
  L2:                     512 KiB (2 instances)
  L3:                     8 MiB (2 instances)
NUMA:
  NUMA node(s):           1
  NUMA node0 CPU(s):      0,1
Vulnerabilities:
  Gather data sampling:   Unknown: Dependent on hypervisor status
  Indirect target selection: Mitigation; Aligned branch/return thunks
  Itlb multihit:          KVM: Mitigation: VMX unsupported
  L1tf:                   Mitigation; PTE Inversion
  Mds:                    Mitigation; Clear CPU buffers; SMT Host state unknown
  Meltdown:               Mitigation; PTI
  Mmio stale data:        Mitigation; Clear CPU buffers; SMT Host state unknown
```



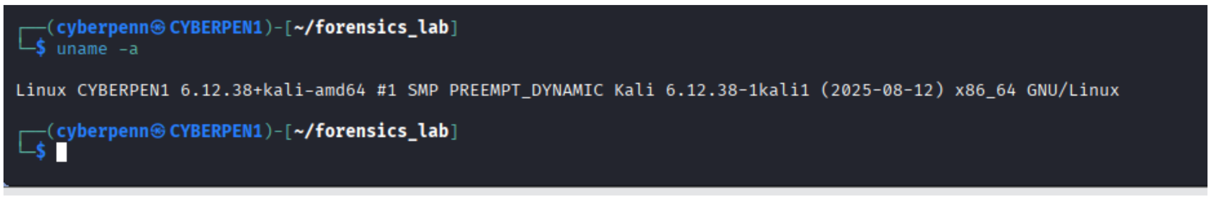
```
┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ free -h

               total        used        free      shared  buff/cache   available
Mem:           3.8Gi       940Mi       2.3Gi        13Mi       844Mi       2.9Gi
Swap:          3.1Gi          0B       3.1Gi

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ 
```



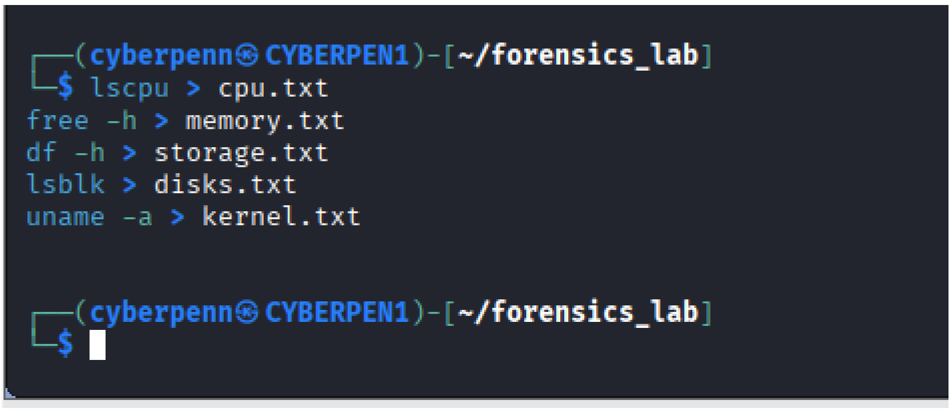
```
┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ df -h

Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G     0  1.9G   0% /dev
tmpfs           389M  1.2M  388M   1% /run
/dev/sda1        56G   23G   31G  43% /
tmpfs           1.9G  4.0K  1.9G   1% /dev/shm
tmpfs           1.0M     0  1.0M   0% /run/credentials/systemd-journald.service
tmpfs           1.9G   36K  1.9G   1% /tmp
tmpfs           1.0M     0  1.0M   0% /run/credentials/getty@tty1.service
tmpfs           389M  108K  389M   1% /run/user/1000

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ 
```



```
┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ lsblk

NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   60G  0 disk
├─sda1   8:1    0 56.9G  0 part /
├─sda2   8:2    0    1K  0 part
└─sda5   8:5    0  3.1G  0 part [SWAP]
sr0     11:0    1  4.2G  0 rom

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ 
```



```
┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ uname -a

Linux CYBERPEN1 6.12.38+kali-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.38-1kali1 (2025-08-12) x86_64 GNU/Linux

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ 
```



```
┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ lscpu > cpu.txt
free -h > memory.txt
df -h > storage.txt
lsblk > disks.txt
uname -a > kernel.txt


┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ 
```

Task:

Document the verification process step-by-step, detailing the commands used and the output observed. Include screenshots if possible.

2. Challenges in Computer Forensics:

- Question: Discuss why computer forensics can be challenging. Focus on topics like data recovery (deleted files, overwritten data), integrity verification (ensuring data hasn't been tampered with), and legal implications (e.g., maintaining chain of custody).

Challenges in Computer Forensics

Computer forensics presents several challenges and they are listed below

- **Data Recovery**
    - Deleted files may still exist in unallocated space but can be overwritten at any time.
    - SSDs with **TRIM** make deleted data nearly impossible to recover.
- **Integrity Verification**
    - Forensic investigators must prove evidence hasn't been tampered with.
    - Hashing (MD5, SHA-1, SHA-256) ensures authenticity, but even a small change alters the hash.
- **Legal Implications (Chain of Custody)**
    - Every piece of evidence must be properly documented and tracked.
    - Failure to maintain chain of custody can make evidence inadmissible in court.
- **Encryption & Privacy**
    - Full-disk encryption (BitLocker, LUKS, FileVault) may block access to evidence.
    - Legal/privacy regulations restrict how data can be collected and analyzed.
- **Data Volume**
    - Modern systems store terabytes of data.
    - Processing, indexing, and analyzing this data requires time and advanced tools.

3. Forensic Tools and Techniques:

- Question: Provide examples of tools (e.g., Autopsy, FTK Imager) and techniques (e.g., file carving, memory analysis) used in computer forensics to overcome the challenges mentioned above.

Tools

- Autopsy – GUI-based digital forensics platform; analyzes disk images, extracts browser history, recovers deleted files.
- FTK Imager – Imaging tool used to create exact bit-by-bit copies of disks while preserving integrity.
- EnCase – Commercial forensic suite widely used in law enforcement.
- Volatility – Memory forensics framework; analyzes RAM dumps (processes, open files, network connections).
- Foremost / Photorec – File carving tools that recover deleted files based on file signatures.

Techniques

- File Carving – Recovers deleted files from raw disk images by identifying file headers/footers.
- Memory Analysis – Captures volatile memory to detect running processes, malware, or encryption keys.
- Timeline Analysis – Reconstructs sequence of events (file creation, access, modification).
- Hashing – Verifies evidence integrity using algorithms like SHA-256.
- Imaging with Write-Blockers – Ensures original evidence isn’t modified during acquisition.

Lab 2: Understanding Computer Systems, Forensics Challenges, and Storage Structures

Objective:

This lab aims to provide students with a deeper understanding of computer systems, the complexities of computer forensics, hard disk drives, the boot process, and various file systems. By completing this lab, students will gain practical insights into how computers operate and the underlying structures essential for digital forensics.

Part 1: How Do Computers Work?

1. Basic Computer Architecture:

- Describe the main components of a computer system, including the CPU, RAM, motherboard, storage devices, and input/output devices.

A computer system consists of several main components that work together to process data as follows:

- CPU (Central Processing Unit): The brain of the computer that performs calculations and executes instructions. It consists of the Arithmetic Logic Unit (ALU) for calculations, the Control Unit (CU) for directing operations, and registers for temporary data storage.
- RAM (Random Access Memory): Temporary or volatile memory that stores data and instructions currently being used by the CPU. It allows fast access for running applications.
- Motherboard: The main circuit board that connects all components of the computer, allowing communication between the CPU, memory, storage devices, and peripherals through data buses and chipsets.
- Storage Devices: Permanent data storage units such as Hard Disk Drives (HDDs) or Solid-State Drives (SSDs) that keep files, programs, and the operating system even when the computer is turned off.
- Input Devices: Tools that allow users to provide data or commands to the computer, such as a keyboard, mouse, or scanner.
- Output Devices: Components that present processed information to the user, such as monitors, printers, or speakers.

Question: Explain how data flows between these components during a typical computation.

When a computation occurs, data flows through the following sequence:

- The user provides input through an input device (e.g., keyboard or mouse).
- The data is temporarily stored in RAM via the motherboard’s data buses.
- The CPU fetches the data and instructions from RAM using the Control Unit.
- The ALU processes or performs calculations on the data.

- The results are sent back to RAM or directly to an output device.
- If needed, the processed data is saved permanently to a storage device.

This continuous exchange of data between memory, CPU, and storage allows the computer to perform complex operations efficiently.

2. Functionality Overview:

- Write a brief paragraph explaining how a computer executes a program from loading it into memory to executing instructions.
    - When a computer executes a program, the operating system first loads the program from storage into the main memory (RAM). The CPU then begins a process called the fetch-decode-execute cycle**,** where it fetches an instruction from memory, decodes it to determine the required action, and executes it using the Arithmetic Logic Unit (ALU). The results are stored back in memory or sent to output devices. This cycle repeats rapidly, allowing the computer to perform complex tasks in seconds.

Part 2: Why Computer Forensics is Hard

1. Challenges in Digital Forensics:

- Discuss the reasons why computer forensics can be challenging. Consider factors like encryption, data volatility, and the potential for tampering.
    - Computer forensics involves collecting, analyzing, and preserving digital evidence from computers, mobile devices, and networks in a way that is admissible in court. However, several factors make this process complex. Digital evidence is highly volatile, meaning it can be easily altered, deleted, or lost. The widespread use of encryption makes accessing or decoding data difficult without the right keys. Additionally, the sheer volume of data, along with the risk of tampering or anti-forensic techniques, complicates evidence recovery and verification.

Question:

Identify at least three specific challenges faced by forensic analysts and propose potential solutions or tools to address these challenges.

Specific Challenges and Possible Solutions

Challenge	Description	Possible Solution / Tool
1. Data Encryption	Criminals often use strong encryption to protect data, making it inaccessible without decryption keys.	Use specialized decryption and password recovery tools such as Passware Kit, Elcomsoft Forensic Suite, or

		collaborate with law enforcement for lawful key retrieval.
2. Data Volatility	Data in RAM or cache can be lost when a system is powered off, making evidence collection time-sensitive.	Use live forensics tools like FTK Imager, Volatility Framework, or Belkasoft RAM Capturer to collect volatile data before shutdown.
3. Anti-Forensic Techniques and Data Tampering	Suspects may use tools to delete, hide, or manipulate digital traces.	Use forensic recovery tools such as Autopsy, EnCase, or X-Ways Forensics to detect manipulation and recover deleted files. Employ hashing (MD5/SHA-1) to verify integrity.

2. Case Study Analysis:

- Find a real-world example of a forensic investigation that faced significant challenges. Summarize the case and the obstacles encountered.

A notable example of a challenging forensic investigation is the **San Bernardino iPhone Case (2016)** involving the FBI and Apple Inc. After a terrorist attack, the FBI recovered an iPhone belonging to one of the suspects but was unable to access its data due to Apple's **strong encryption and security lock mechanisms**. Apple refused to create a backdoor citing privacy and security concerns, resulting in a legal and ethical standoff. Eventually, the FBI paid a third-party security firm (believed to be Cellebrite) to unlock the device.

This case highlights key forensic challenges such as **encryption barriers**, **legal constraints**, and **vendor cooperation issues**. It emphasizes the need for better collaboration between technology companies and law enforcement, as well as the importance of developing lawful, privacy-respecting forensic tools.

Part 3: Hard Disk Drives

1. Disk Drive Components: Identify and describe the main components of a hard disk drive (HDD), including platters, read/write heads, and actuators.

A **Hard Disk Drive (HDD)** is a data storage device that uses magnetic storage to store and retrieve digital information. The main components include:

- **Platters:**
  Circular disks coated with a magnetic material where data is stored. Each platter has

two surfaces (top and bottom) that can hold data. They spin at high speeds (typically 5400–7200 RPM) to allow read/write access.

- **Read/Write Heads:**
  Tiny electromagnetic devices positioned just above each platter surface. The **read head** detects the magnetic polarity on the disk surface to read data, while the **write head** changes the magnetic orientation to record data.
- **Actuator Arm and Actuator:**
  The **actuator arm** holds the read/write heads and moves them across the platter surfaces. The **actuator** (often a voice coil motor) precisely controls this movement to position the heads over the correct track during data access.
- **Spindle Motor:**
  Rotates the platters at a constant speed, enabling the heads to access data quickly as the disks spin.
- **Controller Board:**
  An electronic circuit board attached to the drive that manages communication between the HDD and the computer, handling data transfer, error correction, and caching.

2. Disk Geometry Calculation: Given the following parameters:

- Sectors per track: 500
- Number of heads: 10
- Cylinders: 12,000

Tasks: a. Calculate the total number of sectors on the disk.

Total Sectors = Sectors per Track * Number of Heads * Cylinders
=500*10*12000

Total number of sectors = 60,000,000

b. Discuss how the design of HDDs affects performance and data retrieval times.

- **Rotational Speed (RPM):**
  Higher RPMs (e.g., 7200 or 10,000) reduce the time the head waits for the right sector to rotate under it, improving **seek time** and **data transfer rates**.
- **Data Density:**
  More sectors per track allow more data to be stored in the same space, increasing **throughput** and **storage efficiency**.
- **Cache Size and Controller Efficiency:**
  Larger caches and advanced controllers enable faster temporary storage and retrieval, improving performance during frequent reads/writes.

- **Head and Actuator Precision:**
  Faster and more accurate head movements reduce **seek time**—the delay between requesting and accessing data.
- **Platter Count:**
  Multiple platters increase capacity but may slightly increase mechanical delay due to more head movements.

Part 4: Calculate Disk Partitions

1. Understanding Partitions:

- **Explain the concept of disk partitions and their purpose in a computer system.**

A **disk partition** is a logical division of a physical hard drive into separate sections, each functioning as an independent storage unit. Partitions allow an operating system to organize and manage data more efficiently.

For example:

- The **System Partition** holds the operating system and essential files.
- The **Data Partition** stores user files such as documents, videos, and applications.
- The **Recovery Partition** contains tools and backup images used to restore the system in case of failure.

2. Partition Calculation Exercise:

- If a hard drive has a total capacity of 1 TB and is divided into the following partitions:
  - System partition: 200 GB
  - Data partition: 600 GB
  - Recovery partition: 200 GB

Question:

Calculate the percentage of disk space used for each partition and the total free space available on the disk.

Solution

- **Step1:**

Total used space = (200+600+200) GB

=1000GB

This means all space are allocated and no free space

- **Step 2: Calculate the Percentage of Disk Space Used by Each Partition**

  Formular: percentage = (partition size/TotalCapacity) * 100

  - System Partition = (200/1000)*100 = 20%
  - Data Partition = (600/1000)* 100 = 60%
  - Recovery Partition = (200/1000) * 100 =20%

Part 5: PC Boot Process

1. Boot Sequence:

- Outline the steps involved in the PC boot process, from powering on the computer to loading the operating system.

Boot Sequence

The **boot process** refers to the sequence of steps a computer follows from the moment it is powered on until the operating system (OS) is fully loaded and ready for use.

Steps in the Boot Process:

1. **Power On Self-Test (POST):**
   When the computer is powered on, the **BIOS/UEFI** firmware performs a POST to check that essential hardware components (CPU, RAM, keyboard, storage devices, etc.) are functioning properly.
2. **BIOS/UEFI Initialization:**
   The BIOS (Basic Input/Output System) or UEFI (Unified Extensible Firmware Interface) initializes hardware devices and loads firmware settings stored in non-volatile memory (CMOS).
3. **Boot Device Selection:**
   The BIOS/UEFI checks the **boot order** (e.g., HDD, SSD, USB, DVD) to determine which device contains the operating system's boot loader.
4. **Loading the Bootloader:**
   Once the boot device is found, the BIOS/UEFI loads the **bootloader** (e.g., GRUB for Linux, Windows Boot Manager) into memory. The bootloader is responsible for locating and loading the operating system kernel.
5. **Loading the Operating System Kernel:**
   The bootloader transfers control to the **OS kernel**, which initializes system processes, drivers, and services necessary for operation.
6. **User Login and OS Startup:**
   Finally, the OS displays the **login screen** and loads the **user environment**, completing the boot process.

Question:

What role does the BIOS/UEFI play in the boot process, and how does it interact with the operating system

The **BIOS/UEFI** acts as the bridge between the computer's hardware and the operating system, In essence, BIOS/UEFI ensures the system hardware is ready and helps the OS start smoothly. Its key roles include:

- **Hardware Initialization:** Detects and prepares essential hardware components for use.
- **System Configuration:** Stores system settings (boot order, time, device info).
- **Boot Management:** Identifies and launches the appropriate bootloader from the selected device.
- **Handoff to the OS:** Once the OS kernel is loaded, BIOS/UEFI transfers control, allowing the operating system to manage the system directly.

.

2. Boot Troubleshooting:

- Identify common issues that can occur during the boot process and provide solutions or troubleshooting steps for each.

Issue	Possible Cause	Troubleshooting Steps / Solution
No Power or No Display	Faulty power supply, loose cables, or damaged hardware.	Check power connections, test PSU, and ensure RAM/CPU are seated correctly.
POST Beep Codes or Errors	Hardware failure detected during POST (RAM, GPU, etc.).	Refer to motherboard manual for beep code meaning and replace faulty components.
"No Bootable Device Found" Error	Boot order misconfiguration or missing OS.	Check BIOS/UEFI boot order; verify that the storage drive contains a valid OS installation.
Corrupt Bootloader	Bootloader files (e.g., GRUB, Windows Boot Manager) are damaged.	Use system repair tools: Windows Recovery Environment or Linux Live USB to repair bootloader.
Blue Screen or Kernel Panic	Corrupt system files, faulty drivers, or hardware incompatibility.	Boot into Safe Mode, update or roll back drivers, and perform system restore.
Slow Boot Times	Too many startup programs or failing HDD.	Disable unnecessary startup apps, check disk health (e.g.,

		chkdsk, SMART tools), or upgrade to SSD.

Part 6: File Systems

1. File System Overview:

- Define what a file system is and its role in data management within a computer system.

A **file system** is a method and data structure used by an operating system to control how data is stored, organized, and retrieved on a storage device such as a hard drive, SSD, or USB flash drive. It defines how files are named, stored in directories, and accessed by the system and users Without a file system, the operating system would not be able to locate or interpret stored information on the disk.

The main roles of a file system include:

- **Data Organization:** Arranging files into folders and subdirectories for easy navigation.
- **Storage Management:** Keeping track of which parts of the disk are used or free.
- **Access Control:** Managing file permissions and user privileges.
- **Data Integrity:** Ensuring data is written, retrieved, and maintained without corruption.

2. Comparative Analysis:

- Compare and contrast at least three different file systems (e.g., NTFS, FAT32, EXT4) in terms of their structure, advantages, disadvantages, and use cases.

Feature	**NTFS (New Technology File System)**	**FAT32 (File Allocation Table 32)**	**EXT4 (Fourth Extended File System)**
Structure	Uses a Master File Table (MFT) to store metadata and supports journaling for reliability.	Simple table-based structure that maps clusters to files.	Uses journaling and extent-based storage for efficiency and reliability.
Maximum File Size	Up to 16 TB	4 GB (per file)	16 TB (theoretical), practical limits vary
Maximum Volume Size	256 TB	2 TB	1 EB (exabyte)
Security Features	Supports file permissions, encryption (EFS), and access control lists (ACLs).	No built-in security or permissions.	Supports Linux-style permissions and access control.

Performance	Good balance of performance and security; optimized for Windows systems.	Fast and lightweight but limited features.	Very efficient for large files and directories, excellent journaling performance.
Reliability	Journaling protects against data corruption.	Lacks journaling; prone to data loss if system crashes.	Journaling ensures quick recovery after crashes.
Compatibility	Works best on Windows; read-only support on macOS and Linux.	Compatible with almost all OSes and devices.	Native to Linux; read-only or limited support on Windows.
Ideal Use Case	Windows desktops, laptops, and servers.	USB drives, memory cards, and legacy devices.	Linux servers, modern distributions, and enterprise storage.

Question:

Which file system would you recommend for a specific application (e.g., for a personal computer, a server, or an embedded system) and why?

- **For a Personal Computer (Windows-based):**
  NTFS is recommended because it offers strong security features, file compression, large file support, and journaling for reliability. It is also optimized for Windows performance and supports modern system features like user permissions and encryption.
- **For a Server (Linux-based):**
  EXT4 is ideal due to its stability, journaling support, and efficient handling of large volumes and files. It provides excellent performance for multi-user environments and is widely supported in enterprise-grade Linux servers.
- **For an Embedded System or Portable Devices:**
  FAT32 remains suitable for small storage media (e.g., USB drives, SD cards) because of its simplicity and broad compatibility with different devices and operating systems.

Networking, and Batch Scripting

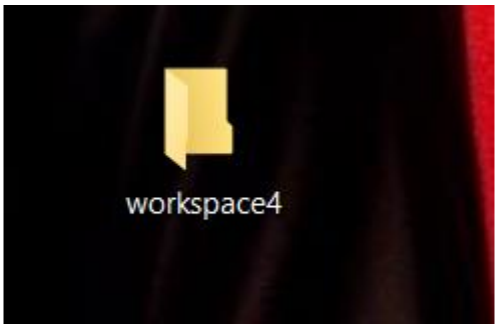
Objective:

This lab aims to provide students with practical experience in managing files and folders in the Windows operating system, understanding networking functionalities, and creating and executing batch files. By completing this lab, students will enhance their file management skills and learn how to automate tasks using batch scripts, alongside foundational networking commands.

Part 1: Create Folders and Files

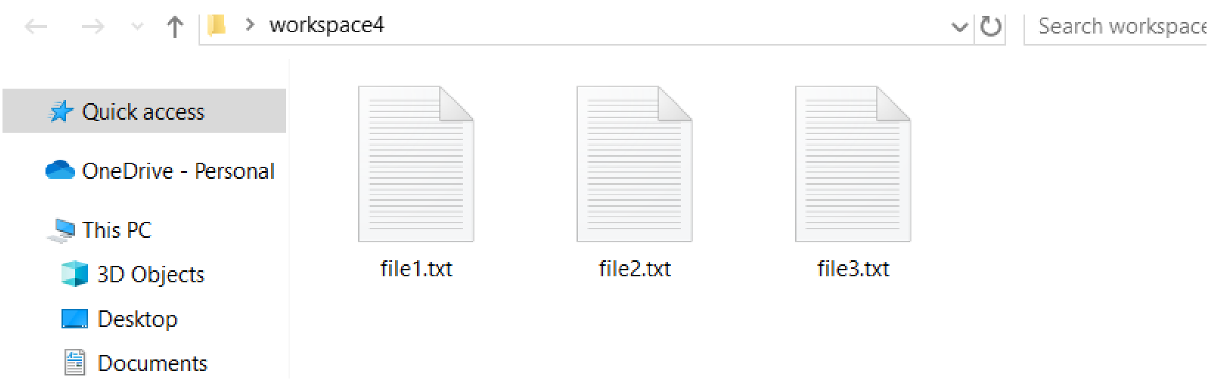
1. Folder Creation:

- Create a new folder named workspace4 on your desktop.
- Task: Verify that the folder has been created successfully.



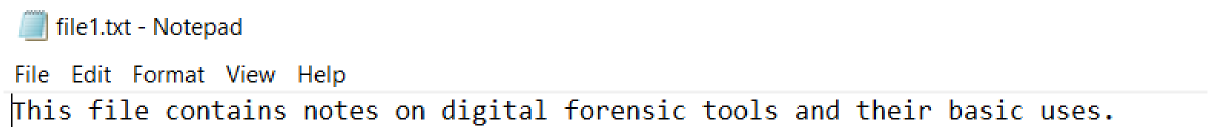
2. File Creation:

- Inside the workspace4 folder, create three text files:
- file1.txt
- file2.txt
- file3.txt

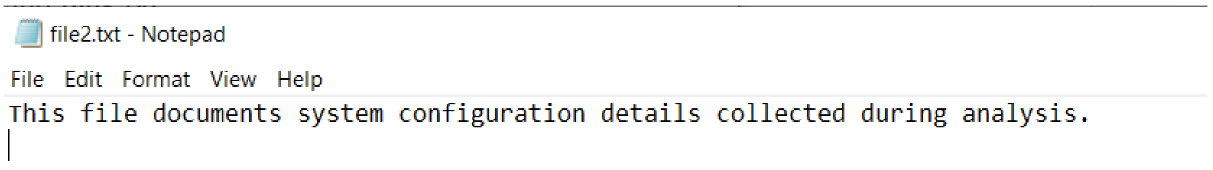


Task:

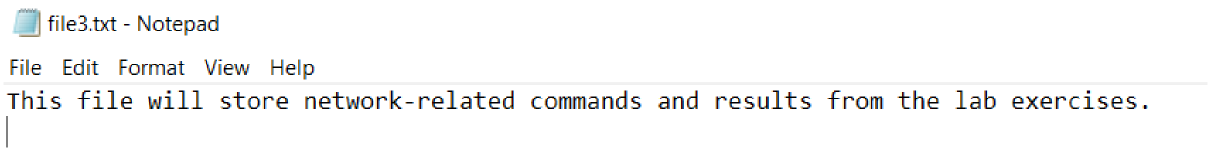
- Open each text file and write a brief description of its purpose. Save and close the files.



```
This file contains notes on digital forensic tools and their basic uses.
```



```
This file documents system configuration details collected during analysis.
```



```
This file will store network-related commands and results from the lab exercises.
```

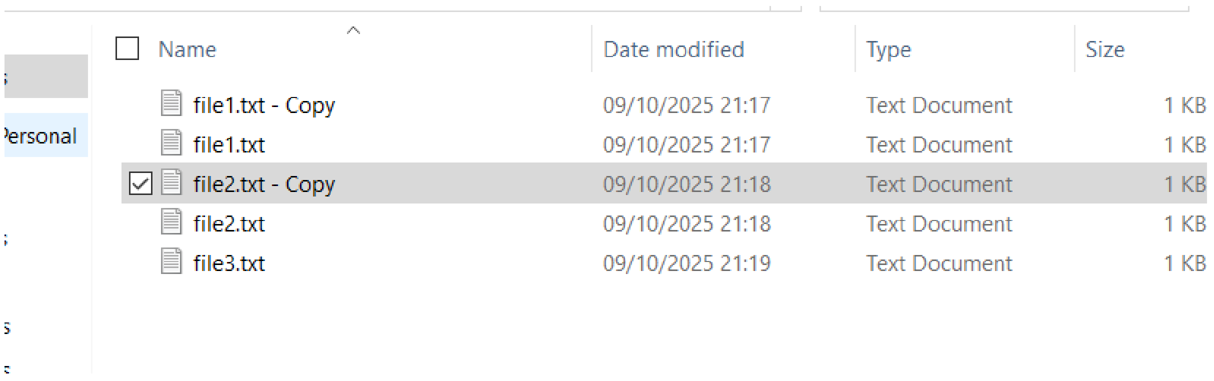
Part 2: File Copy & Deletion

1. File Copying:

- Copy file1.txt and file2.txt to the workspace4 folder.

Task:

- Confirm that the files have been copied correctly.



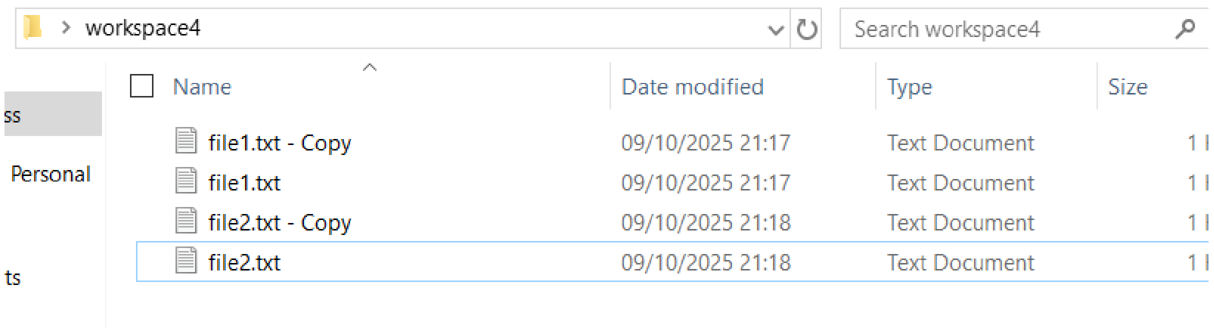
	Name	Date modified	Type	Size
	file1.txt - Copy	09/10/2025 21:17	Text Document	1 KB
	file1.txt	09/10/2025 21:17	Text Document	1 KB
✓	file2.txt - Copy	09/10/2025 21:18	Text Document	1 KB
	file2.txt	09/10/2025 21:18	Text Document	1 KB
	file3.txt	09/10/2025 21:19	Text Document	1 KB

2. File Deletion:

- Delete file3.txt from the workspace4 folder.

Task:

- Ensure that the file has been deleted and does not appear in the folder.



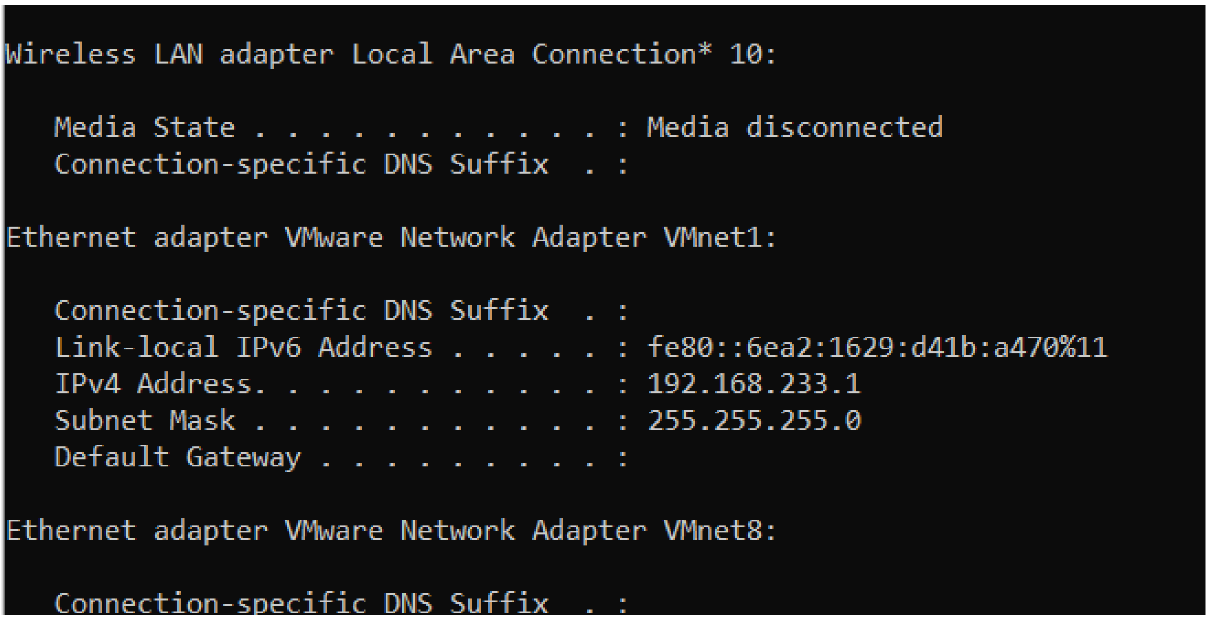
workspace4 | Search workspace4

	Name	Date modified	Type	Size
	file1.txt - Copy	09/10/2025 21:17	Text Document	1
	file1.txt	09/10/2025 21:17	Text Document	1
	file2.txt - Copy	09/10/2025 21:18	Text Document	1
	file2.txt	09/10/2025 21:18	Text Document	1

Part 3: Networking and Searching for Information

1. Networking Commands: Use the following commands in the Command Prompt to gather information about your network:

- ipconfig: Display your current IP address and network configuration.
    - The system's **local IP address** is `192.168.200.228`, assigned to the Wi-Fi adapter.
    - The **default gateway** is `192.168.200.247`, which is the router connecting the PC to the internet.
    - The **subnet mask** (`255.255.255.0`) shows that all devices with IPs starting with `192.168.200` are on the same network.
    - Several **virtual adapters** (OpenVPN, VMware) appeared as disconnected, indicating inactive or virtual network interfaces.



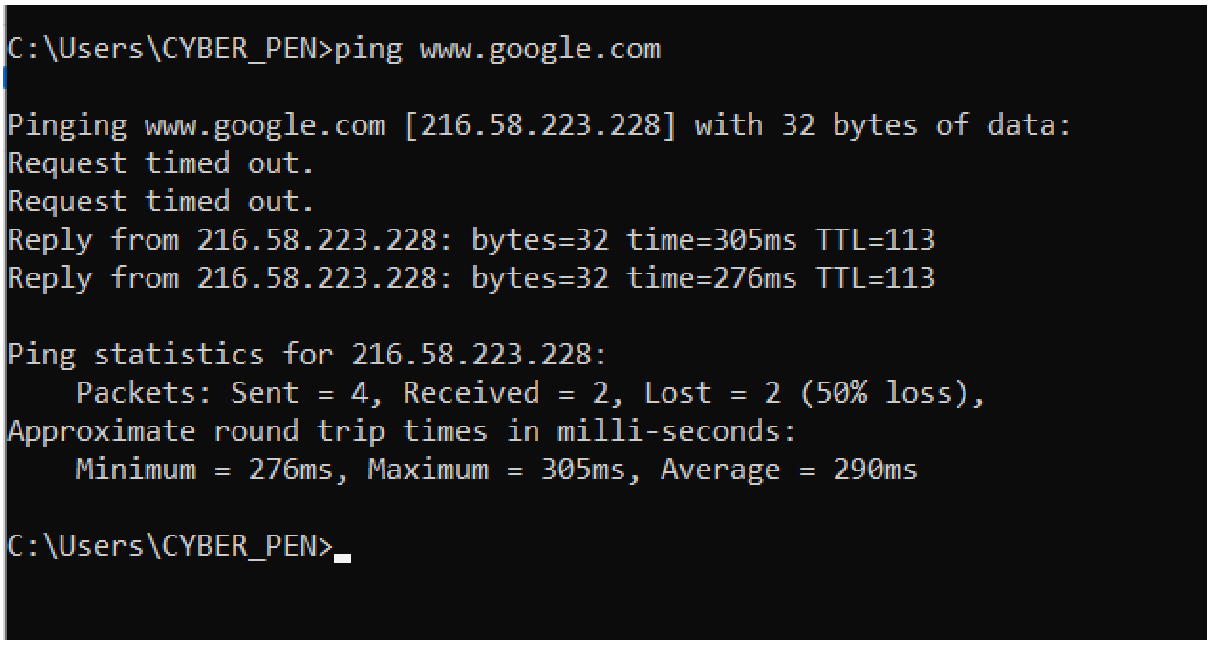
```
Wireless LAN adapter Local Area Connection* 10:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter VMware Network Adapter VMnet1:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::6ea2:1629:d41b:a470%11
   IPv4 Address. . . . . . . . . . . : 192.168.233.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Ethernet adapter VMware Network Adapter VMnet8:

   Connection-specific DNS Suffix  . :
```

- ping www.google.com: Check the connectivity to Google's servers.

Information Gathered:

- The system successfully connected to Google's server at IP **216.58.223.228**.
- **50% packet loss** indicates an unstable network connection, possibly due to congestion or VPN interference.
- The **average latency (290 ms)** is relatively high, meaning slower network response times.
- Confirms **partial internet connectivity**.

    ◦



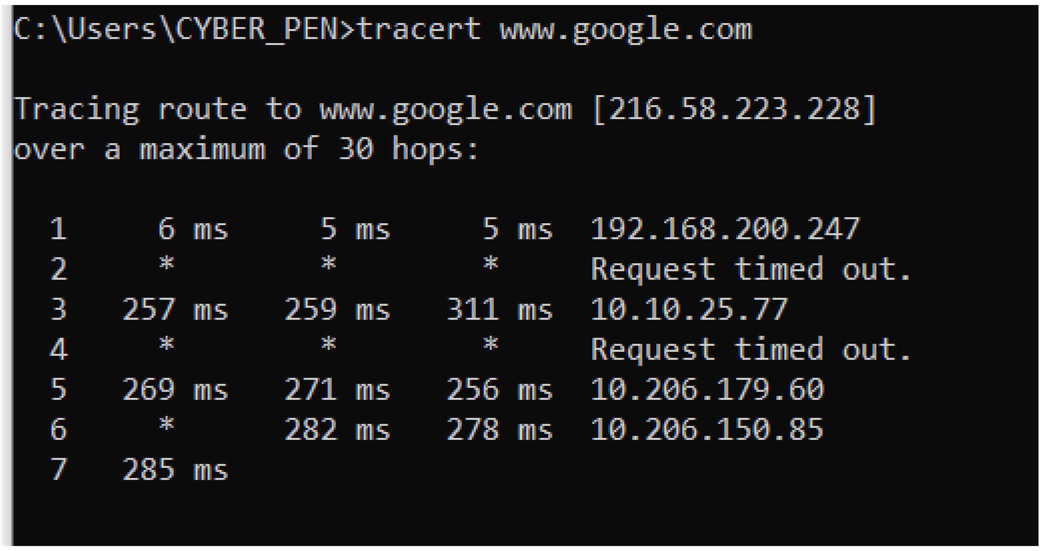
```
C:\Users\CYBER_PEN>ping www.google.com

Pinging www.google.com [216.58.223.228] with 32 bytes of data:
Request timed out.
Request timed out.
Reply from 216.58.223.228: bytes=32 time=305ms TTL=113
Reply from 216.58.223.228: bytes=32 time=276ms TTL=113

Ping statistics for 216.58.223.228:
    Packets: Sent = 4, Received = 2, Lost = 2 (50% loss),
Approximate round trip times in milli-seconds:
    Minimum = 276ms, Maximum = 305ms, Average = 290ms

C:\Users\CYBER_PEN>
```

- tracert www.google.com: Trace the route packets take to reach Google

Information Gathered:

- The first hop (`192.168.200.247`) is the **router (default gateway)**.
- The remaining hops show the **path through various network routers** (10.x.x.x and 172.x.x.x are internal ISP and Google network routers).
- Asterisks (`*`) mean some routers blocked ICMP responses for security reasons.
- The command verifies the **route and delay each hop takes** to reach Google’s servers

.

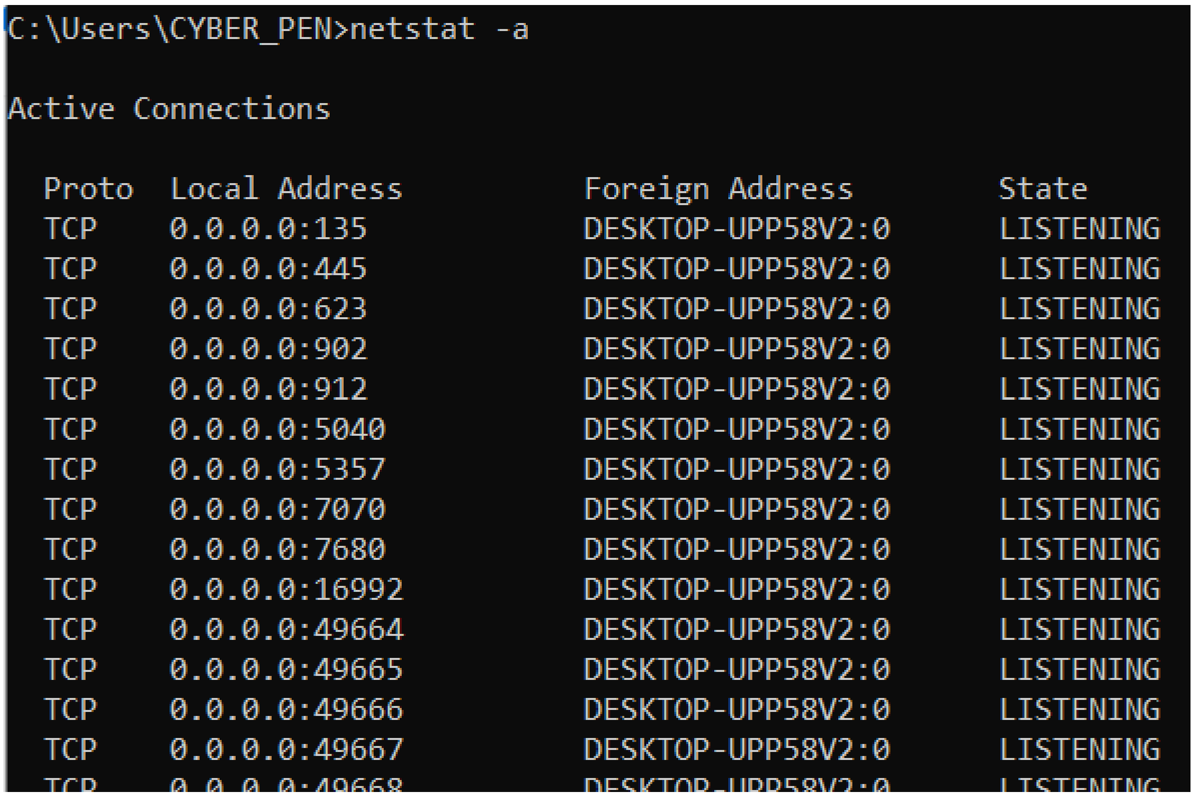


```
C:\Users\CYBER_PEN>tracert www.google.com

Tracing route to www.google.com [216.58.223.228]
over a maximum of 30 hops:

  1     6 ms     5 ms     5 ms  192.168.200.247
  2     *        *        *     Request timed out.
  3   257 ms   259 ms   311 ms  10.10.25.77
  4     *        *        *     Request timed out.
  5   269 ms   271 ms   256 ms  10.206.179.60
  6     *      282 ms   278 ms  10.206.150.85
  7   285 ms
```

- netstat -a: Show active connections and listening ports.

Information Gathered:

- The **LISTENING** ports (e.g., 135, 445) indicate Windows services like **Remote Procedure Call (RPC)** and **file sharing**.
- The **ESTABLISHED** connections (e.g., `5228`, `https`) show active communication with **Google and other web servers**.
- This helps identify **active network sessions** and potential **suspicious connections**, which is vital for forensic analysis.



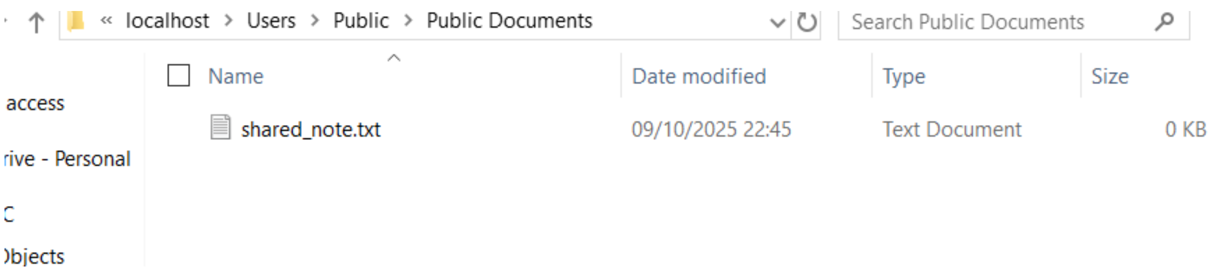
```
C:\Users\CYBER_PEN>netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:445            DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:623            DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:902            DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:912            DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:5040           DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:5357           DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:7070           DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:7680           DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:16992          DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:49664          DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:49665          DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:49666          DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:49667          DESKTOP-UPP58V2:0      LISTENING
  TCP    0.0.0.0:49668          DESKTOP-UPP58V2:0      LISTENING
```

Task:

- Document the output of each command and explain what information you gathered.

2. Network Access:

- Access a shared network folder (if available) or create a simple text file within your local network that can be accessed by another user.



localhost > Users > Public > Public Documents

Name	Date modified	Type	Size
shared_note.txt	09/10/2025 22:45	Text Document	0 KB

Task:

- Describe how you accessed the network folder and any challenges encountered.

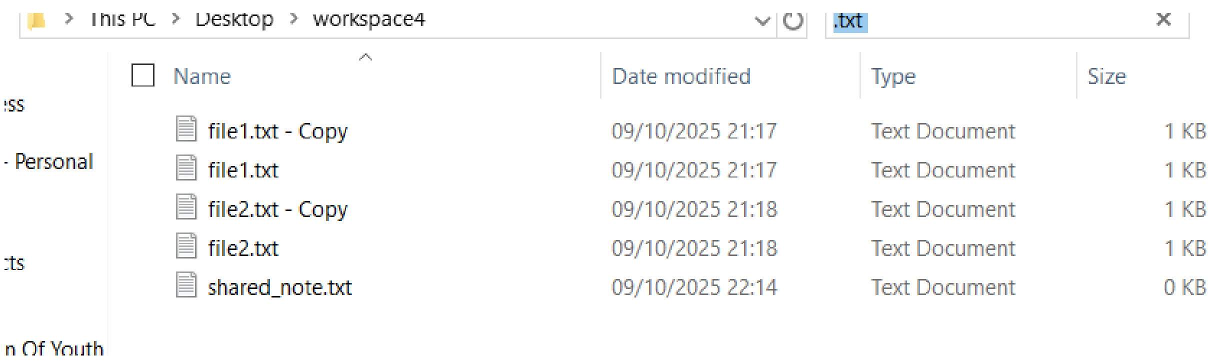
Description:
I created a text file named `shared_note.txt` and shared it on my local network. After enabling **Network Discovery** and **File and Printer Sharing** in the Control Panel, I accessed

the file using its network path (`\\CYBERPEN1\Users\Public\shared_note.txt`). Another user on the same Wi-Fi network could open and edit the file successfully.

Challenges:
Initially, network discovery was off, preventing access. After enabling sharing settings, the issue was resolved.

3. Searching for Information:

- Use the Windows search feature to locate all .txt files within the workspace4 folder.



This PC > Desktop > workspace4 | .txt

Name	Date modified	Type	Size
file1.txt - Copy	09/10/2025 21:17	Text Document	1 KB
file1.txt	09/10/2025 21:17	Text Document	1 KB
file2.txt - Copy	09/10/2025 21:18	Text Document	1 KB
file2.txt	09/10/2025 21:18	Text Document	1 KB
shared_note.txt	09/10/2025 22:14	Text Document	0 KB

Task:

- List the steps you took to perform the search and the results obtained.

Steps Taken

1. I opened **File Explorer** on my Windows system.
2. I navigated to the folder named **workspace4**, where my project files are located.
3. In the **search bar** at the top-right corner of the File Explorer window, I typed:
4. `*.txt`

   This command filters and displays all text files within the folder.

5. Windows automatically searched through the folder and its subfolders to locate all `.txt` files.
6. After the search completed, I reviewed the results list displayed.

Results Obtained

The search returned all .txt files present in the **workspace4** folder.
The results included files such as:

- file1.txt
- file2.txt
- shared_note.txt (the file I created for network sharing)

Each file was displayed with its file path, size, and modification date, confirming that the files exist and are accessible in the workspace4 directory.

Part 4: Creating a Batch File

1. Batch File Creation:

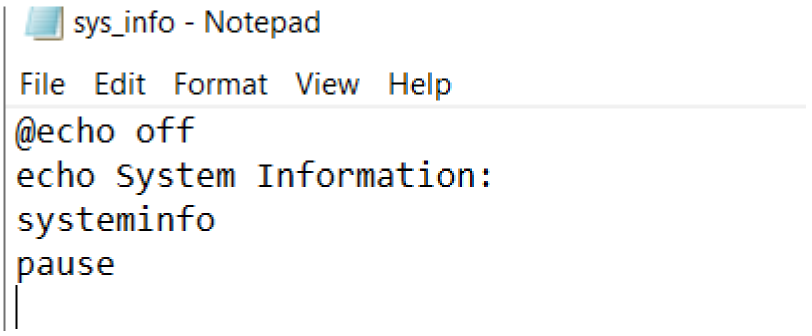
- Create a batch file named sys_info.bat with the following content:

2. @echo off

3. echo System Information:

4. systeminfo

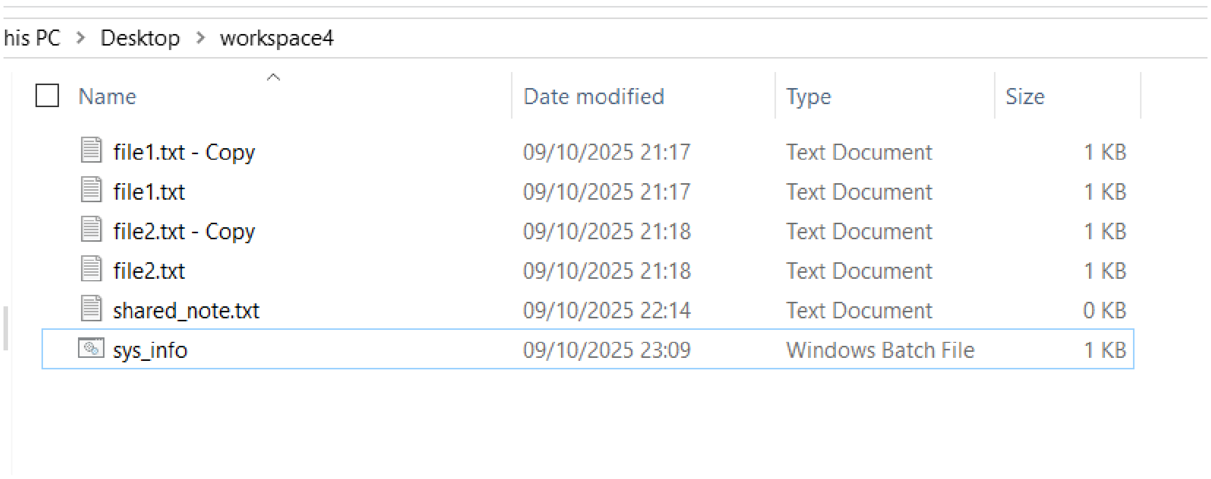
pause



```
@echo off
echo System Information:
systeminfo
pause
```

Task:

- Save the batch file in the workspace4 folder.



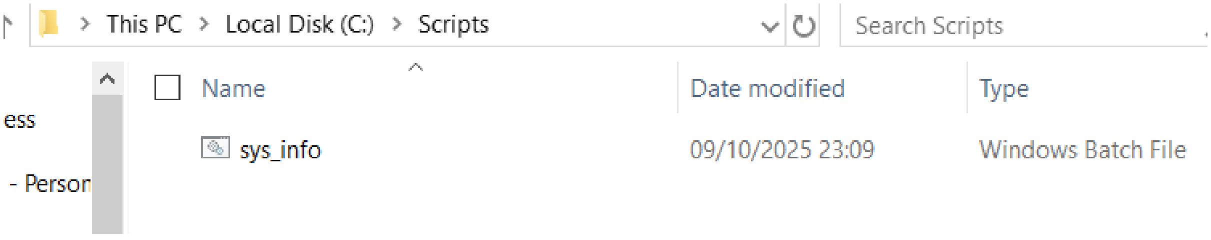
Name	Date modified	Type	Size
file1.txt - Copy	09/10/2025 21:17	Text Document	1 KB
file1.txt	09/10/2025 21:17	Text Document	1 KB
file2.txt - Copy	09/10/2025 21:18	Text Document	1 KB
file2.txt	09/10/2025 21:18	Text Document	1 KB
shared_note.txt	09/10/2025 22:14	Text Document	0 KB
sys_info	09/10/2025 23:09	Windows Batch File	1 KB

5. Making the Batch File Callable:

- To make sys_info.bat callable from any folder, follow these steps:

a. Create a new folder to store batch files (if not already done), e.g., C:\Scripts.

b. Copy the sys_info.bat file from workspace4 to C:\Scripts.

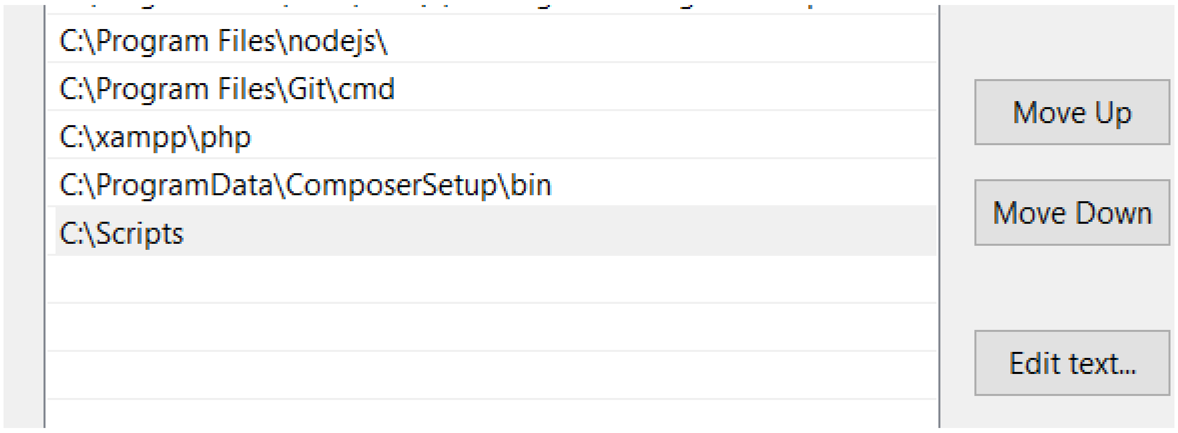


This PC > Local Disk (C:) > Scripts

Name	Date modified	Type
sys_info	09/10/2025 23:09	Windows Batch File

c. Modify the Path Variable:

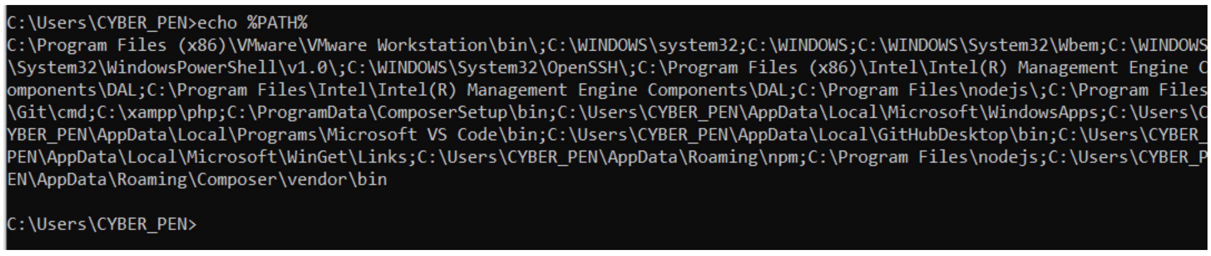
- Add the new folder (C:\Scripts) to the system PATH environment variable:
- Right-click on This PC or Computer and select Properties.
- Click on Advanced system settings.
- In the System Properties window, click on the Environment Variables button.
- In the System variables section, find and select the Path variable, then click Edit.
- Click New and add C:\Scripts to the list.
- Click OK to save the changes.



C:\Program Files\nodejs\
C:\Program Files\Git\cmd
C:\xampp\php
C:\ProgramData\ComposerSetup\bin
C:\Scripts

Task:

- Confirm that the path has been successfully modified.



```
C:\Users\CYBER_PEN>echo %PATH%
C:\Program Files (x86)\VMware\VMware Workstation\bin\;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\WINDOWS
\System32\WindowsPowerShell\v1.0\;C:\WINDOWS\System32\OpenSSH\;C:\Program Files (x86)\Intel\Intel(R) Management Engine C
omponents\DAL;C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\nodejs\;C:\Program Files
\Git\cmd;C:\xampp\php;C:\ProgramData\ComposerSetup\bin;C:\Users\CYBER_PEN\AppData\Local\Microsoft\WindowsApps;C:\Users\C
YBER_PEN\AppData\Local\Programs\Microsoft VS Code\bin;C:\Users\CYBER_PEN\AppData\Local\GitHubDesktop\bin;C:\Users\CYBER_
PEN\AppData\Local\Microsoft\WinGet\Links;C:\Users\CYBER_PEN\AppData\Roaming\npm;C:\Program Files\nodejs;C:\Users\CYBER_P
EN\AppData\Roaming\Composer\vendor\bin

C:\Users\CYBER_PEN>
```

6. Executing the Batch File:

- Open the Command Prompt and type sys_info.bat from any folder.

Task:

- Document the output you receive and verify that the batch file executed successfully.