

OS assignment

by Leo Richards

Submission date: 18-Apr-2025 02:00PM (UTC+0100)

Submission ID: 256018017

File name: 57111_Leo_Richards_OS_assignment_644541_444097413.docx (3.06M)

Word count: 2841

Character count: 17843



OPERATING SYSTEMS

ACCA4022



LEO RICHARDS
2418519
Cyber Security & Networking

Contents

Introduction	2
Key functions of general Operating Systems	3
Use of Windows OS CLI commands	4
Use of Linux OS CLI commands.	7
Use of Windows OS scripts (PowerShell)	9
Use of Linux OS scripts	11
Conclusion	12
References	13
Appendices	14

Introduction

Operating systems serve as the foundation of modern computing, acting as a bridge between hardware components and user applications. They manage system resources, ensure stability, and provide a structured environment for executing programs efficiently. Without an operating system, users would struggle to interact with their computers in a meaningful way, as it handles everything from memory allocation to task scheduling and peripheral device management.[1]

This document delves into the essential functions of general operating systems, examining critical components such as process management, memory distribution, and device interaction. These aspects contribute to a smooth and responsive user experience, ensuring that programs run effectively, and system resources are utilized optimally. Understanding these core functions provides insight into how different operating systems maintain efficiency and reliability across various computing environments.

Additionally, the document explores the use of command-line interfaces (CLI) and scripting in both Windows and Linux systems. CLI commands allow users to execute tasks quickly and precisely, offering more control over system operations compared to graphical interfaces. Scripting, on the other hand, enables automation, reducing manual intervention and improving administrative efficiency. By mastering these tools, users can enhance system management, automate repetitive tasks, and troubleshoot issues effectively.

By gaining a deeper understanding of operating systems, command-line functionality, and scripting techniques, users can optimize performance, improve workflow, and harness the full potential of their computing environments.[1]

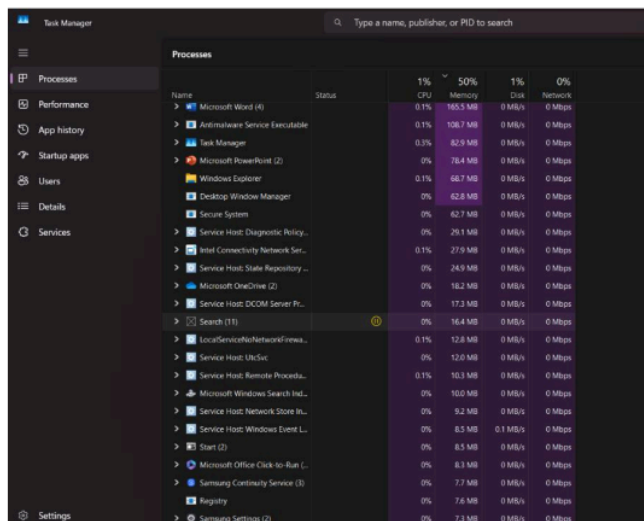
Key functions of general Operating Systems

Some of the main operating system functions are device management, memory management, process management, network management, user interface control, security and access control, error handling, logging and debugging. Depending on what system and or operating system you are running you will use different tools / apps to complete these tasks.[1]

The process management operating system handles the creation, scheduling, and termination of processes. it makes sure all running applications get enough CPU power and memory without causing drain or conflicts. If you are using a windows machine, then you will use the task manager. To access the task manager on a windows device you would use the shortcut control, alt & delete at the same time from there you will be able to monitor and control processes. however, if you are using a Linux or mac system you will need to use system monitor, or alternatively you can use the 'top' 'htop' commands in terminal [2]

Memory management manages how much RAM goes to each individual process including paging and swapping. It calculates each byte of systems memory and dictates how much memory is allocated to each process. On a windows system you would check through task manager > performance > memory. With mac open activity monitor > memory tab. And for Linux use the 'free -h' or vmstat commands.[3]

Device management helps the operating system communicate with the hardware through device drivers allowing the operating system to use accessories like keyboards, mice, speakers, printers and USB devices. Windows devices use device manager. macOS go to system settings > Hardware. Linux use the lsusb, lspci or dmesg commands



The screenshot shows the Windows Task Manager application. The 'Processes' tab is selected, displaying a list of running applications and background services. The columns shown are Name, Status, CPU, Memory, Disk, and Network. The list includes Microsoft Word (4), Antimalware Service Executable, Task Manager, Microsoft PowerPoint (2), Windows Explorer, Desktop Window Manager, Secure System, and various Windows services like Diagnostic Policy, Network Setup, and System Restore. The CPU usage is 1%, Memory is 50%, Disk is 1%, and Network is 0%.

Name	Status	CPU	Memory	Disk	Network
Microsoft Word (4)	Running	0.1%	165.5 MB	0 MB/s	0 Mbps
Antimalware Service Executable	Running	0.1%	108.7 MB	0 MB/s	0 Mbps
Task Manager	Running	0.3%	82.9 MB	0 MB/s	0 Mbps
Microsoft PowerPoint (2)	Running	0%	78.4 MB	0 MB/s	0 Mbps
Windows Explorer	Running	0.1%	68.7 MB	0 MB/s	0 Mbps
Desktop Window Manager	Running	0%	62.8 MB	0 MB/s	0 Mbps
Secure System	Running	0%	62.7 MB	0 MB/s	0 Mbps
Service Host: Diagnostic Policy...	Running	0%	26.1 MB	0 MB/s	0 Mbps
Intel Connectivity Network Ser...	Running	0.1%	27.9 MB	0 MB/s	0 Mbps
Service Host: State Repository ...	Running	0%	24.9 MB	0 MB/s	0 Mbps
Microsoft OneDrive (2)	Running	0%	16.2 MB	0 MB/s	0 Mbps
Service Host: DCOM Server Pr...	Running	0%	17.3 MB	0 MB/s	0 Mbps
Search (11)	Running	0%	16.4 MB	0 MB/s	0 Mbps
LocalService\NetworkDns...	Running	0.1%	12.8 MB	0 MB/s	0 Mbps
Service Host: UsbSrv	Running	0%	12.6 MB	0 MB/s	0 Mbps
Service Host: Remote Procede...	Running	0.1%	10.3 MB	0 MB/s	0 Mbps
Microsoft Windows Search Ind...	Running	0%	10.0 MB	0 MB/s	0 Mbps
Service Host: Network Store in...	Running	0%	9.2 MB	0 MB/s	0 Mbps
Service Host: Windows Event L...	Running	0%	8.5 MB	0.1 MB/s	0 Mbps
Start (2)	Running	0%	8.5 MB	0 MB/s	0 Mbps
Microsoft Office Click-to-Run ...	Running	0%	8.3 MB	0 MB/s	0 Mbps
Samsung Continuity Service (3)	Running	0%	7.7 MB	0 MB/s	0 Mbps
Registry	Running	0%	7.6 MB	0 MB/s	0 Mbps
Samsung Settings (2)	Running	0%	7.3 MB	0 MB/s	0 Mbps

(screenshot from windows task manager process management)

Use of Windows OS CLI commands

PowerShell is a robust scripting language, and command-line shell designed for system administration and automation on Windows. It enhances traditional command-line functionality with object-oriented processing, making it an invaluable tool for IT professionals.

PowerShell's built-in cmdlets streamline system management tasks. For instance, Get-Process provides details about running processes, helping users monitor CPU and memory usage. Get-Service lists system services, allowing administrators to check statuses and control essential background functions. Get-EventLog enables users to review system logs for troubleshooting and auditing purposes, while Set-ExecutionPolicy manages script execution security, ensuring protection against unauthorized scripts. [4]

The Get-Help cmdlet provides documentation on commands, making PowerShell an accessible tool for learning and reference. Additionally, common batch commands such as DIR, ECHO, PAUSE, DEL, and COPY aid in file management and automation, complementing PowerShell's capabilities.

PowerShell also supports advanced scripting, including variables, loops, conditional statements, and functions, enabling users to automate repetitive tasks efficiently. Error handling with try, catch, and finally ensures stability in scripts, preventing disruptions due to unexpected failures.

Ultimately, PowerShell is a powerful asset for Windows system administration, offering structured, efficient tools for process management, automation, and security. By mastering its features, users can optimize workflows and enhance system control with precision.[5]

1. Get-process – The Get-Process command displays a full list of all currently running processes on your machine. It displays information such as process names, IDs, CPU usage, memory usage, and more.
2. Get-Service – This command displays a list of all the services installed on the system, along with their status (Running, Stopped, etc.). It's useful for checking whether a particular service is active or troubleshooting service-related issues.
3. Get-EventLog - Used for reading event logs on the system, which is essential for diagnosing issues, auditing activities, and reviewing system events. It can access logs like Application, System, and Security.
4. Set-ExecutionPolicy – PowerShell has an execution policy feature that restricts script execution to prevent the running of potentially malicious scripts. This command is used to change the execution policy level on a machine.
5. Get-Help - The Get-Help cmdlet provides documentation about PowerShell cmdlets, functions, workflows, aliases, and scripts. It's an essential tool for learning and reference.

```
C:\Program Files\WindowsAp  X  +  -
PowerShell 7.5.0
PS C:\Windows\System32> get-service

Status  Name                               DisplayName
-----  ---                               -
Stopped AarSvc_12eee08b                   Agent Activation Runtime_12eee08b
Stopped ALG                        Application Layer Gateway Service
Stopped AppIDSvc                   Application Identity
Running Appinfo                     Application Information
Stopped AppReadiness              App Readiness
Running AppXSvc                   AppX Deployment Service (AppXSVC)
Stopped ApxSvc                     Windows Virtual Audio Device Proxy Se...
Running AudioEndpointBuil...       Windows Audio Endpoint Builder
Running Audiosrv                  Windows Audio
Stopped autotimesvc               Cellular Time
Stopped AxInstSV                  ActiveX Installer (AxInstSV)
Stopped BcastDVRUserSer...         GameDVR and Broadcast User Service_12
```

(screenshot showing the get-service command in PowerShell)

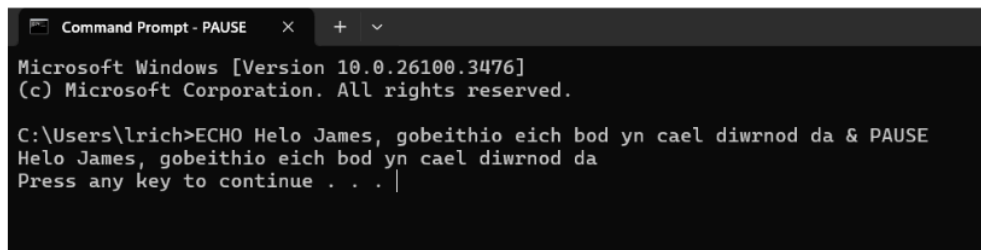
```
PS C:\Users\lrich> get-service

Status  Name                               DisplayName
-----  ---                               -
Stopped AarSvc_15baf425                   Agent Activation Runtime_15baf425
Stopped ALG                        Application Layer Gateway Service
Stopped AppIDSvc                   Application Identity
Running Appinfo                     Application Information
Stopped AppReadiness              App Readiness
Running AppXSvc                   AppX Deployment Service (AppXSVC)
Stopped ApxSvc                     Windows Virtual Audio Device Proxy ...
Running AudioEndpointBu...       Windows Audio Endpoint Builder
Running Audiosrv                  Windows Audio
Stopped autotimesvc               Cellular Time
Stopped AxInstSV                  ActiveX Installer (AxInstSV)
Stopped BcastDVRUserSer...         GameDVR and Broadcast User Service_...
Stopped BDESVC                    BitLocker Drive Encryption Service
```

(Get-Service command in PowerShell))

A few common batch commands are.

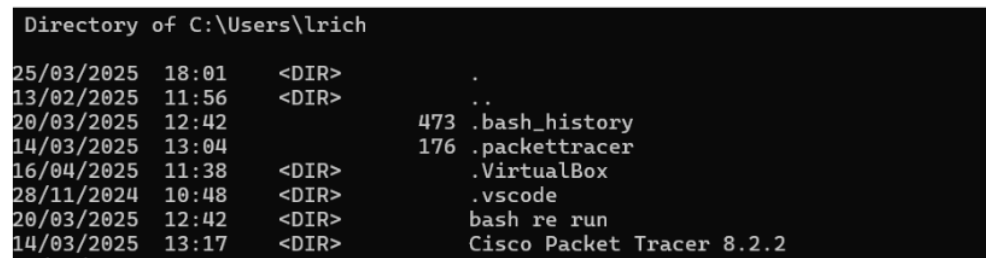
1. DIR – Displays directory contents. If you would like to see them in wide list format use /W at the end.
2. ECHO – displays messages or enables / disables command echoing.
3. PAUSE - Pauses script execution. And prompts user to press a key to continue.
4. DEL – Deletes files. Can also be used with, /F to force delete read only files and or, /Q to enable quiet mode and show no confirmation
5. COPY – copies files



```
Command Prompt - PAUSE
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\lrich>ECHO Helo James, gobeithio eich bod yn cael diwrnod da & PAUSE
Helo James, gobeithio eich bod yn cael diwrnod da
Press any key to continue . . . |
```

(Screenshot of the use of the ECHO & PAUSE commands)



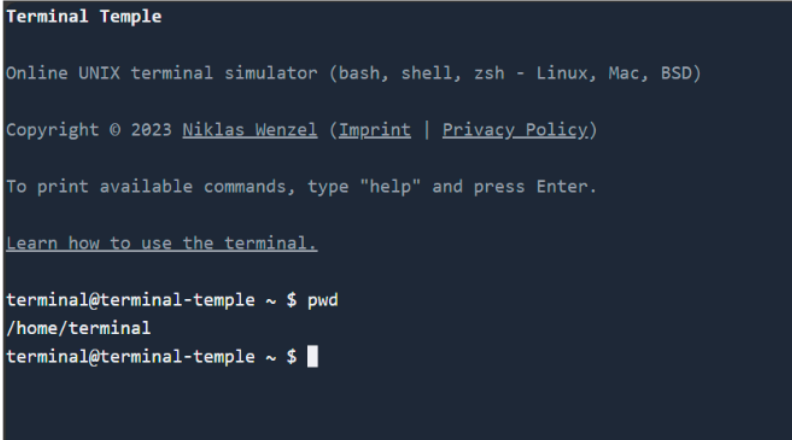
```
Directory of C:\Users\lrich
25/03/2025 18:01 <DIR> .
13/02/2025 11:56 <DIR> ..
20/03/2025 12:42 473 .bash_history
14/03/2025 13:04 176 .packettracer
16/04/2025 11:38 <DIR> .VirtualBox
28/11/2024 10:48 <DIR> .vscode
20/03/2025 12:42 <DIR> bash re run
14/03/2025 13:17 <DIR> Cisco Packet Tracer 8.2.2
```

(Screen shot of DIR command)

Use of Linux OS CLI commands.

The Bash shell is widely used in Unix-like systems for the management of files, processes, and system operations. Presented here are ten common Bash commands along with helpful switches[6]

1. PWD – shows the full working directory
2. LS – lists directory contents, there are also 2 switches you can use with this command. The -l switch lists it in long format. -a includes all hidden files
3. CD – changes the current directory.
4. MKDIR - creates a directory. The -p switch creates parent directories if needed.
5. RM – Removes files, with this command there is also the option to use 2 switches. -f to force action with no prompts and -r removes directories recursively.
6. CP – copies files and directories you can use the -r switch to make it a recursive copy. Or -v if you would like to see the verbose output
7. MV – moves or renames files.
8. ECHO – Displays text or variables.
9. CAT – Displays file content.
10. GREP – searches for text. Use the -i switch if you would like to make the search case insensitive. [6]



```
Terminal Temple

Online UNIX terminal simulator (bash, shell, zsh - Linux, Mac, BSD)

Copyright © 2023 Niklas Wenzel (Imprint | Privacy Policy)

To print available commands, type "help" and press Enter.

Learn how to use the terminal.

terminal@terminal-temple ~ $ pwd
/home/terminal
terminal@terminal-temple ~ $
```

(Screen shot of the pwd command being used in a bash shell)

Terminal Temple

Online UNIX terminal simulator (bash, shell, zsh - Linux, Mac, BSD)

Copyright © 2023 [Niklas Wenzel](#) ([Imprint](#) | [Privacy Policy](#))

To print available commands, type "help" and press Enter.

[Learn how to use the terminal.](#)

```
terminal@terminal-temple ~ $ mkdir -p newproject/james
```

```
terminal@terminal-temple ~ $ ls
```

```
Documents      images          NewDirectory    Pictures
```

```
Downloads      Music           newproject
```

```
terminal@terminal-temple ~ $ █
```

(Screen shot of the mkdir & ls commands. There is also use of the -p switch)

Use of Windows OS scripts (PowerShell)

Developed for Windows OS, PowerShell is a powerful scripting language that makes system management and administrative chores automatable. Here is an example of some of PowerShell scripting's main abilities.[7]

Variables and Data Types:

PowerShell accommodates multiple data types, including strings, integers, arrays, and objects, providing flexibility for various scripting scenarios. Variables, defined using the dollar sign (\$), store data which can be referenced and manipulated throughout a script. For instance, the statement `$JW = "JamesWilliams"` assigns a string value to the variable `$JW`. This streamlines data handling, allowing for easy reference, modification, and utilization of stored data during script execution.

Conditional Statements:

Decision making within PowerShell scripts is efficiently handled using conditional statements such as `if`, `elseif`, and `else`. These allow scripts to evaluate specific conditions and determine the appropriate action to take. Conditional statements facilitate logical comparisons, enabling scripts to branch their flow depending on variable values or the outcomes of expressions. This feature is particularly useful for implementing dynamic automation tasks.

Loops:

PowerShell supports several loop types, including `for`, `foreach`, and `while` loops, which automate repetitive tasks effectively. These loops allow scripts to systematically process collections of data or repeatedly execute code blocks until certain conditions are met. For example, a `foreach` loop can iterate over each item in an array or collection, performing specified actions on each element, significantly reducing the manual effort involved in repetitive administrative tasks.

Functions:

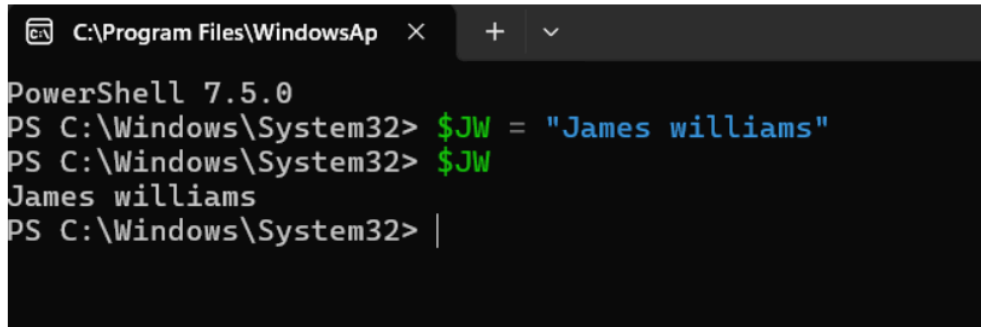
Functions in PowerShell enhance the structure, readability, and reusability of scripts. A function encapsulates logic and operations into a single named block, allowing scripts to invoke these blocks repeatedly without redundancy. This modular approach promotes clarity and maintainability in scripting, as functions can be debugged, modified, and reused across multiple scripts or various sections within a script, thus facilitating organized and efficient script development.

File Operations:

PowerShell simplifies file management operations, providing straightforward commands for reading from and writing to files. Commands such as `Get-Content` and `Set-Content` enable users to interact with textual data stored in files easily. For example, `Get-Content` reads the content of a file and outputs it to the console or another command, while `Set-Content` writes or replaces text within a file. These operations are essential for tasks involving data processing, configuration management, and automated reporting.

Error Handling:

Effective error management is crucial in robust scripting, and PowerShell offers powerful error handling capabilities through try, catch, and finally blocks. These constructs manage exceptions gracefully, ensuring scripts can handle unexpected errors without abrupt termination. The try block encloses commands that may potentially fail, while the catch block specifies actions to perform when an error occurs. Additionally, the finally block executes cleanup or finalization code regardless of whether an error occurred, maintaining script stability and reliability.

A screenshot of a PowerShell terminal window. The window title bar shows 'C:\Program Files\WindowsAp' and standard window controls. The terminal text is as follows:

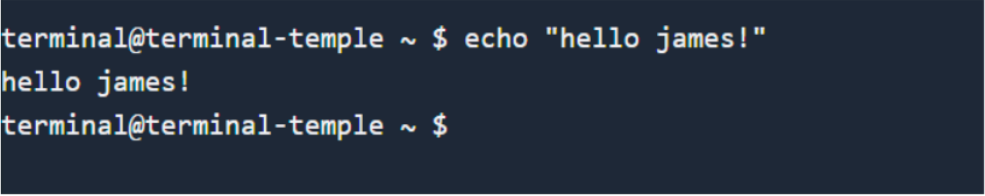
```
PowerShell 7.5.0
PS C:\Windows\System32> $JW = "James williams"
PS C:\Windows\System32> $JW
James williams
PS C:\Windows\System32> |
```

(screenshot of setting a variable and printing it on PowerShell)

Use of Linux OS scripts

Bash (Bourne Again Shell) is a powerful scripting language integral to Linux systems, enabling task automation, system administration, and efficient command execution. Below is a demonstration of key Bash scripting features. [8]

This fundamental script demonstrates the use of the echo command to print text to the terminal. It's often the first script written by beginners to familiarize themselves with Bash scripting syntax.



```
terminal@terminal-temple ~ $ echo "hello james!"  
hello james!  
terminal@terminal-temple ~ $
```

(Screenshot of the echo command on bash)

Conclusion

Operating systems are the backbone of modern computing, enabling seamless interaction between users, hardware, and applications. This document has explored essential functions of operating systems, including process management, memory allocation, and device management. These core functionalities ensure efficient resource distribution, enhance system stability, and improve overall performance across different platforms.

Understanding command-line interfaces (CLI) in both Windows and Linux environments offers users a deeper level of control over system operations. Windows PowerShell provides a powerful, structured approach to system administration, while Bash scripting in Linux delivers flexibility and efficiency for automation and task management. Mastery of CLI commands allows users to optimize processes, streamline workflows, and troubleshoot issues with precision.

Additionally, scripting plays a crucial role in automating repetitive tasks, reducing manual effort, and improving productivity. Windows PowerShell scripts enable robust system management, supporting variables, loops, conditional statements, and error-handling mechanisms. Linux Bash scripts leverage similar constructs, allowing users to automate operations and enhance system performance. By employing scripting techniques, administrators can implement efficient, scalable solutions tailored to their computing needs.

Ultimately, a thorough comprehension of operating systems, CLI commands, and scripting empowers users to maximize system capabilities, maintain security, and foster innovation in computing environments. Whether navigating Windows or Linux, leveraging these tools and concepts will enable professionals to achieve greater efficiency, reliability, and adaptability in their technological endeavours.

References

- [1]Espinoza, B. (n.d.). *Operating Systems: The Backbone of Modern Computing*. starkmagazine.com.
- [2]geeksforgeeks.org. (n.d.). *Memory Management in Operating System*. Retrieved from geeksforgeeks.org
- [3] tutorialspoint. (n.d.). *Operating System - Processes*. tutorialspoint. Retrieved from https://www.tutorialspoint.com/operating_system/os_processes.htm
- [4] Microsoft.Learn. What is PowerShell? [What is PowerShell? - PowerShell | Microsoft Learn](#)
- [5] GeekPedia. Mastering Advanced Scripting Techniques in Windows PowerShell [Advanced Windows PowerShell Scripting Tips](#)
- [6] stackify, 28 Bash Terminal Commands: An Essential Cheat Sheet, [28 Bash Terminal Commands: An Essential Cheat Sheet - Stackify](#)
- [7]ThinkPowerShell, Intro to PowerShell: What makes it essential for IT pros? [Intro to PowerShell: What Makes It Essential for IT Pros?](#)
- [8] DataCamp, How to Write a Bash Script: A Simple Bash Scripting Tutorial, [How to Write a Bash Script: A Simple Bash Scripting Tutorial | DataCamp](#)

Appendices

LECTURE ACTIVITY LOG

What is a OS?

A system software that manages computer hardware and software resources. And provide common services for computer programs

What does a os do

A foundation software for any device

5 common used os systems?

iOS, Windows, Linux, power shell, android.

what is linux?

a free, secure and open-source operating system (OS) that is used in many device

Common os functions.

- controlling hardware. components.
- providing a platform for software to run on.
- providing a user interface.
- managing processes.
- managing the computer's memory.
- managing input and output devices.
- managing applications.
- managing security.

Process scheduler program

Process memory management

Multi threading

Networking management is important os function

1. **/bin**: Contains essential system binaries (programs) needed to troubleshoot, recover, or repair the system, such as ls, cp, mv.

2. **/boot**: Contains bootloader files, including the Linux kernel and initial RAM disk (initrd), necessary for system startup.
3. **/dev**: Contains device files that represent hardware devices like disks, terminals, and printers. For example, `/dev/sda` represents a disk.
4. **/etc**: Holds system configuration files and settings for system-wide programs and services (e.g., `/etc/fstab` for filesystem configurations).
5. **/home**: Stores user home directories, where personal files, configurations, and settings are stored for each user (e.g., `/home/user1`).
6. **/mnt**: A mount point for temporarily mounted filesystems. It's commonly used to mount external devices or network shares.
7. **/sbin**: Contains system binaries for administrative tasks, generally used by the root user (e.g., `mount`, `ifconfig`).
8. **/usr**: Contains user programs, libraries, and documentation. It's one of the largest directories, holding applications and shared files not critical for booting or repairing the system (e.g., `/usr/bin`, `/usr/lib`).

how will ai and quantum computing change the world?

The combined power of artificial intelligence (AI) and quantum computing is expected to significantly change the world by enabling the solution of complex problems currently beyond the reach of traditional computers, leading to breakthroughs in fields like medicine, materials science, finance, and climate change, with potential for faster, more accurate analysis and decision-making across industries, while also raising concerns about security and ethical implications.

1. Batch Operating Systems: These OSs handle jobs in batches without user interaction. Jobs are collected, processed together, and results are delivered later. Historically used in mainframes, these systems minimize human interaction.
(1) <https://www.geeksforgeeks.org/batch-processing-operating-system/>
2. Time-Sharing Operating Systems: Time-sharing OSs allow multiple users to access a system simultaneously by rapidly switching between tasks, giving the illusion of parallel processing. Examples include UNIX and modern versions of Linux.
(2) <https://www.scaler.com/topics/time-sharing-operating-system/>
3. Real-Time Operating Systems (RTOS): Designed for systems that require immediate processing, RTOSs manage tasks with strict timing constraints. Used in embedded systems, automotive, medical devices, and industrial controls, RTOSs prioritize predictable responses to external events.
(3) <https://www.highintegritysystems.com/rtos/what-is-an-rtos/>
4. Single-User, Single-Task Operating Systems: These OSs are designed for one user and run one task at a time. Classic examples include early versions of DOS (Disk Operating System) and some mobile OSs in simpler devices.

(4) http://techgurutechnolic.weebly.com/uploads/7/9/5/7/79577388/1_fdc_answers_unit_6.pdf

5. Single-User, Multi-Tasking Operating Systems: These OSs allow one user to run multiple applications simultaneously. Modern examples include Windows, macOS, and Linux, which support multitasking with user-friendly interfaces. (5) <https://theteacher.info/index.php/f451-fundamentals/3-1-2-software/other-software/86-single-user-multi-user-and-multi-tasking-os-notes>

6. Multi-User Operating Systems: Multi-user OSs allow multiple users to access a computer simultaneously. These are often used in servers, mainframes, and cloud environments. Examples include UNIX, Linux, and Windows Server.

(6) <https://www.geeksforgeeks.org/multi-user-operating-system/>

7. Distributed Operating Systems: These OSs manage a collection of independent computers that appear as a single system to users. They enable resource sharing across machines, such as in cloud computing. Examples include Google's Android and Hadoop in large data systems.

(7) <https://www.javatpoint.com/distributed-operating-system>

8. Network Operating Systems (NOS): A network OS is designed to manage network resources, allowing multiple computers to communicate and share data and hardware. Examples include Novell NetWare and Microsoft Windows Server.

(8) https://www.lenovo.com/us/en/glossary/nos/?orgRef=https%253A%252F%252Fwww.google.com%252F&srltid=AfmBOooV8JpbODNvf1fjDwhiDHoV6z20IATYUPkHaRkzp4h0_hebXPZq

9. Mobile Operating Systems: These are optimized for mobile devices like smartphones and tablets. They include Android, iOS, and HarmonyOS, which support touch interfaces, sensors, and power efficiency.

(9) <https://www.techtarget.com/searchmobilecomputing/definition/mobile-operating-system>

Each OS type is tailored to specific use cases, from personal devices to large-scale enterprise systems.

KEY FUNCTIONS OF GENERAL OPERATING SYSTEMS.

Right click windows icon to manage computer system

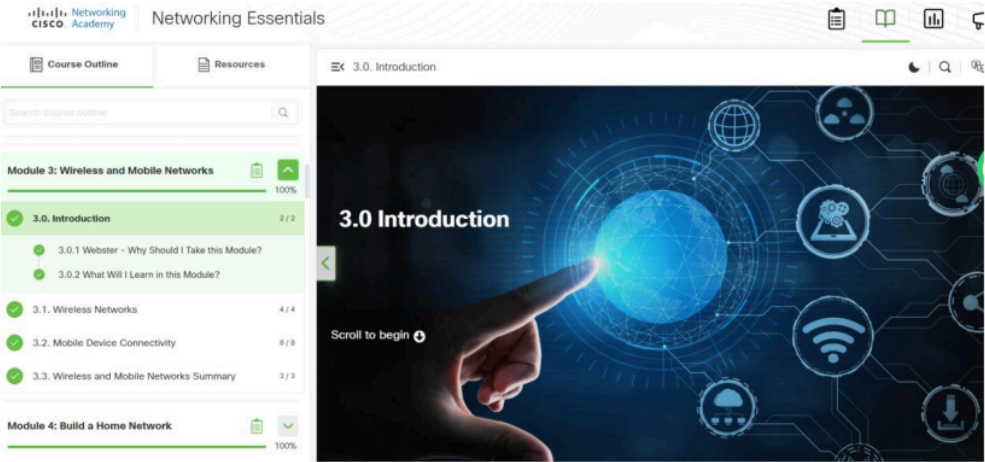
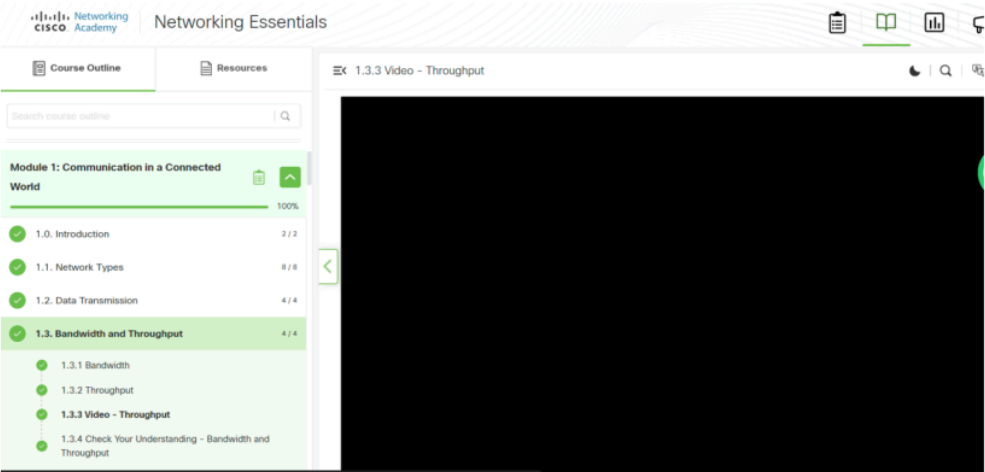
Process explorer

Windows DBG to debug log info

Group policy editor

System file checker tool = type CMD run as admin, type sfc / scannow

SCREENSHOTS FROM CISCO



Q

6.0. Introduction 2 /

- 21

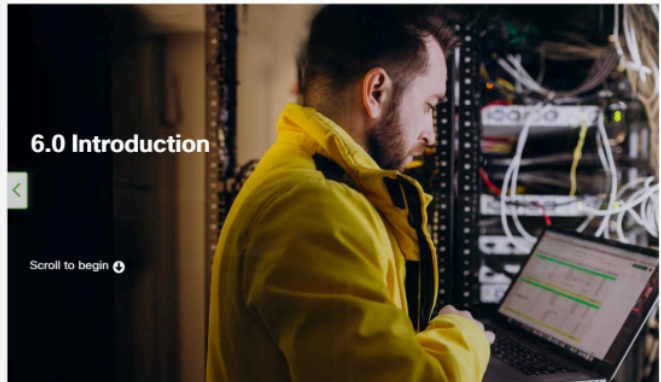
41

3/

100

100

Scroll to begin 



OS assignment

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

50/100

Total:		50
1. Engagement with Literature Skills:		
+ Good use of citations. + formatted reference list.	20	16
2. Knowledge and Understanding Skills:		
Key functions: + Very good overview of key OS functions documented. - could be extended to briefly describe 1/2 additional areas such as security management. + 1 screenshot. – Some additional screenshots showing where some of the functions could documented. Win Commands: + V. good overview of relevant commands that demonstrates an understanding of the language. Commands described could be used for additional marks in GAC4. Bash Commands: + V. good overview of relevant commands that demonstrates an understanding of the language. Commands described could be used for additional marks in GAC4. Win Scripts: + Good overview of the relevant syntax areas that demonstrates some understanding of the scripting aspect of the language. –	20	12

The syntax could be shown or used and could be extended to include 1/2 additional syntax areas. Bash Scripts: - Scripting aspects of the language not documented.		
3. Cognitive and Intellectual Skills:		
+ Evidence of good analysis and understanding of OS functions. + 1 screenshot included in the analysis. – Sec. man. Could be included and some additional screenshots would enhance the analysis. + Screenshots showing where some of the functions can be seen/monitored/managed are included in the analysis. + Conclusion.	1 0	6
4. Practical Application Skills:		
1. Win Commands: + Some use of several relevant Win. OS language commands. - could be extended to use some additional commands. 2. Bash Commands: + Some use of several relevant bash commands. - could be extended to use some additional commands. 3. Win Scripts: + Use of a variable documented. - could be extended to demonstrate the use of all syntax areas. 4. Bash Scripts: - No use of Bash script areas documented.	4 0	7

5. Transferable Skills for Life and Professional Practice:		
+ Good report format, structure, and presentation. - Report sections could be numbered. + Tutorial Activity 1 on OS Types: relevant supplementary information on OS types is included in the Appendix of the report. + Evidence of engagement with a relevant CPD course documented in the appendix. + Lecture activity evidence documented.	10	9
Plagiarism & On-time check:		/

PAGE 1
PAGE 2
PAGE 3
PAGE 4
PAGE 5
PAGE 6
PAGE 7
PAGE 8
PAGE 9
PAGE 10
PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19
