

L4 CA OS assignment

by Connor Johnson

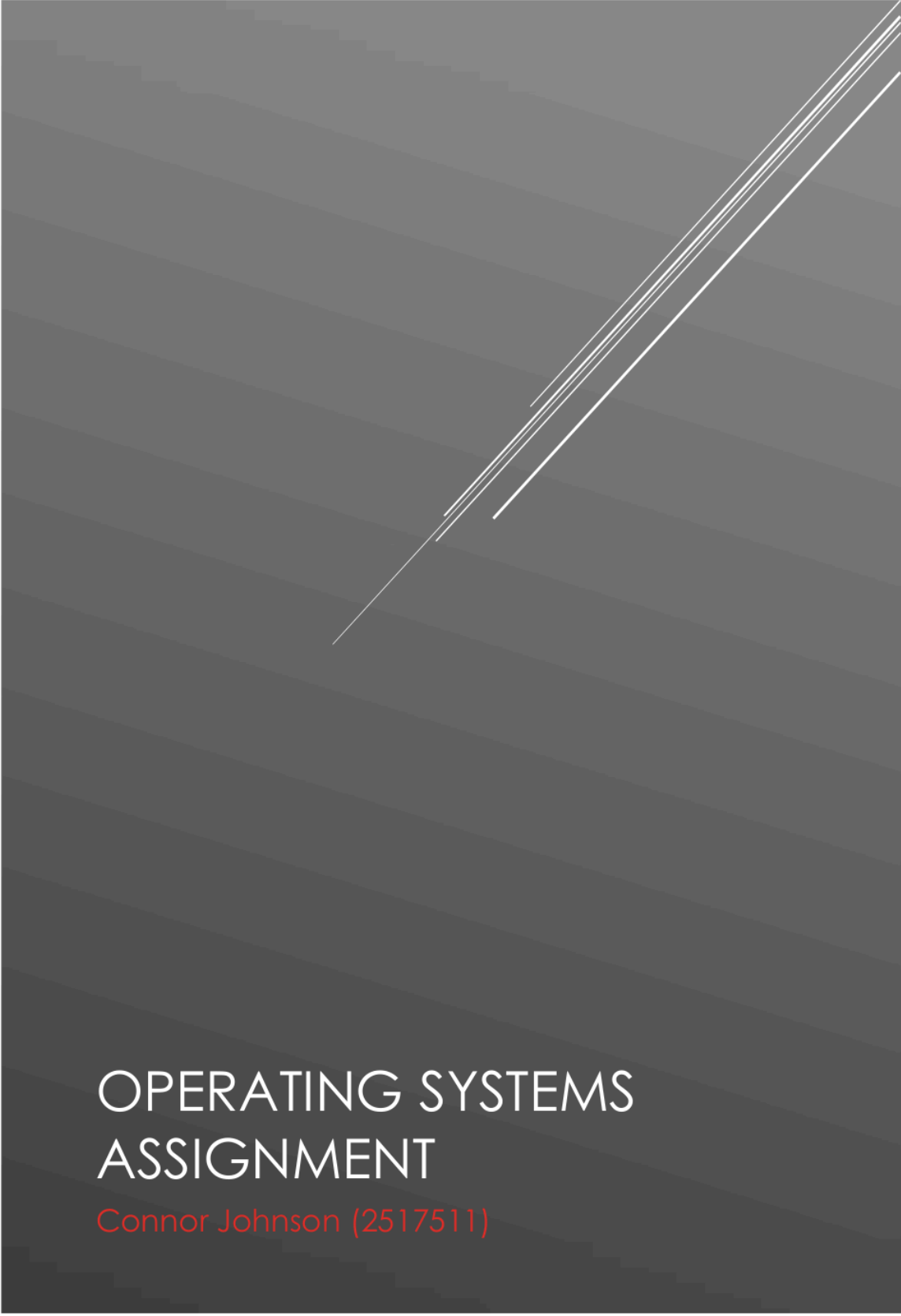
Submission date: 20-May-2026 02:58PM (UTC+0100)

Submission ID: 284191017

File name: 65019_Connor_Johnson_L4_CA_OS_assignment_684823_488147170.docx (1.61M)

Word count: 1534

Character count: 9451

The background of the slide is a dark gray rectangle. It features a series of lighter gray diagonal stripes running from the bottom-left towards the top-right. In the upper right quadrant, there is a cluster of several thin, white, slightly curved lines that appear to be trailing or moving towards the top-right corner.

OPERATING SYSTEMS ASSIGNMENT

Connor Johnson (2517511)

Table of Contents

Introduction.....	2
Key functions of an Operating System	2
Use of Windows OS CLI commands	6
Use of Linux OS CLI Commands.....	8
Use of windows OS scripts	9
Conclusion	12

Introduction

Operating systems are essential system software that manages computer hardware, software resources and user interaction. Popular operating systems such as Microsoft Windows, macOS and Ubuntu provide functions including process, memory, file and device management and security. This assignment explores the key functions of operating systems and demonstrates the use of Windows and Linux command line tools.

An operating system or OS acts as an interface between the computer user and the computer hardware. Without an operating system a computer would not be able to run applications or manage hardware devices effectively. The operating system controls communication between software and hardware components such as the processor, memory, storage devices and input/output devices. Modern operating systems are designed to provide stability, security, and efficient resource management for both personal and business environments

One of the key functions of an operating system is process management. The operating system is responsible for creating, scheduling and terminating processes while ensuring that multiple programs can run efficiently at the same time, this is commonly referred to as multitasking. Tools such as task manager in windows or system monitor in Linux allow users to view active processes and monitor CPU usage. Efficient process management helps maintain overall system performance and prevents applications from interfering with each other.

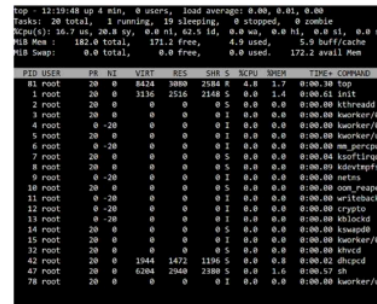


FIGURE 1 REAL-TIME PROCESS AND CPU MONITORING USING THE LINUX TOP COMMAND

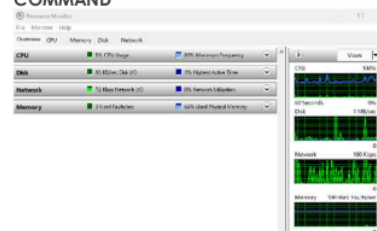


FIGURE 2 WINDOWS RESOURCE MONITOR
DISPLAYING CPU, MEMORY, DISK, AND
NETWORK USAGE

another important function is memory management. The operating system allocates ram to application and services currently running on the system, it also manages virtual memory when physical memory becomes limited. Effective memory management improves system stability and allows several applications to operate simultaneously without crashing.

operating systems can organise files into directories and provide users with methods to create, edit, move and delete files. File systems such as NTFS in windows and ext4 in Linux help manage data storage efficiently while maintaining file permissions and security. File explorer in windows and file management tools in Linux demonstrate how operating systems provide structured access to stored data.

operating systems additionally manage hardware devices with device drivers. This enables communication between the operating system and hardware components such as printers, keyboards, graphics cards and storage devices.

Security management is another essential function, including user authentication, permissions, encryption and firewall protection. These features help protect systems against unauthorised access and malware threats.

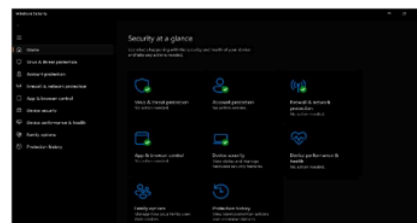


**FIGURE 3 WINDOWS SECURITY
DEMONSTRATING OPERATING SYSTEM
SECURITY FEATURES**

Modern operating systems also include command line interfaces and scripting tools that allow users and admins to automate tasks. Windows commonly uses PowerShell and Linux use bash scripting, these scripting environments provide greater control over operating system functions and improve efficiency when managing files, processes and system configurations.

Key functions of an Operating System

General purpose operating systems manage the computer's hardware and provide a platform for applications to run. Some key functions of widely used operating systems like Microsoft windows and macOS are:

Process management – 'Windows manages processes by prioritising tasks and initiating new ones using system functions like `CreateProcess()`. This enables smooth multitasking and efficient task management.

process management in macOS leverages UNIX commands `fork()` and `exec()`. These commands create and execute new processes concurrently, facilitating the simultaneous operation of multiple applications. The system efficiently determines what task should be run first.' [1]

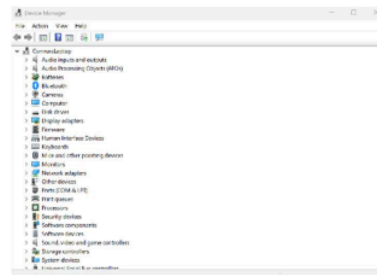
Memory management – ensures safe and efficient use of RAM, the OS assigns memory to processes as needed and reclaims it when no longer used. Systems like Windows and macOS can use disk space to simulate extra RAM, allowing more programs to run simultaneously.

File system management – windows can manage files and directories using file systems such as 'NTFS' [2] and FAT32. The operating systems organizes data into folders and controls how the files are stored, accessed, renamed, copied and deleted. Windows provides built command line tools like `dir`, `copy`, `move`, `del` and `mkdir` for file and directory management. NTFS also supports advanced features including file permissions, encryption, compression and disk quotas helping to improve security and storage efficiency.

```
PS C:\Windows\System32> dir -name
0409
1028
1029
1031
1033
1036
1040
1041
1042
1045
1046
1049
1055
2052
3082
AdvancedInstallers
```

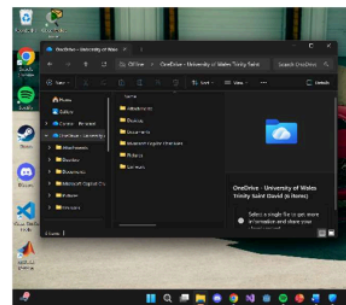
THIS SCREEN SHOT SHOWS THE (DIR - NAME) COMMAND EXECUTED IN POWERSHELL. THE (-NAME) SWITCH SIMPLIFIES THE OUTPUT WITHOUT ADDITIONAL INFORMATION SUCH AS FILE SIZE OR MODIFICATION DATE.

Device management – Microsoft windows manages devices through automatic hardware detection, local management tools and cloud-based admin features. When new hardware is connected, windows uses plug and play technology to detect the new device and automatically install suitable drivers from its system databases or through updates. Users can also manage and trouble shoot hardware manually within the Device manager which provides access to device settings and driver controls. Windows also supports Mobile device Management solutions like Microsoft Intune allowing IT admins to remotely enroll devices, manage security policies and manage software and drivers across multiple systems.



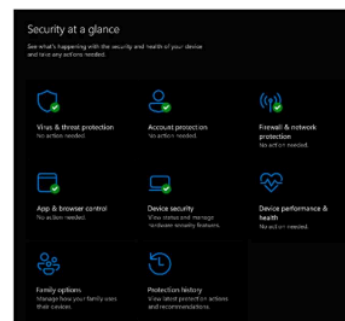
SCREENSHOT OF WINDOWS DEVICE MANAGER

User interface – windows and macOS both use graphical user interfaces or GUIs as a key function of their individual operating systems, in doing so this allows users to interact with computers through icons, menus and windows instead of command line instructions. Windows focuses on flexibility and productivity through features such as the Start menu, taskbar and file explorer whereas macOS emphasis simplicity using the dock, finder and menu bar. Both systems are designed to improve navigation, user experience and multi-tasking.[3]



SCREENSHOT SHOWING WINDOWS FILE EXPLORER, TASK BAR AND DASHBOARD.

Security and Access control – Windows and macOS operating systems have built in security systems to protect files, applications and user accounts. These systems use user authentication methods such as passwords, pins and biometric login options to control access.



WINDOWS SECURITY DASHBOARD SHOWING BUILT IN SECURITY FEATURES.

Use of Windows OS CLI commands

Command 1: Tasklist

```
PS C:\Windows\System32> Tasklist
```

Image Name	PID	Session Name	Session#	Mem Usage
System Idle Process	0	Services	0	0 K
System	4	Services	0	5,644 K
Secure System	284	Services	0	65,540 K
Registry	328	Services	0	39,648 K
smss.exe	962	Services	0	268 K
csrss.exe	1406	Services	0	3,948 K
wininit.exe	1616	Services	0	1,916 K
services.exe	1684	Services	0	10,092 K
LsaIso.exe	1704	Services	0	1,700 K
lsass.exe	1712	Services	0	25,148 K
svchost.exe	1840	Services	0	35,280 K
Fontdrvhost.exe	1872	Services	0	1,836 K
WUDFHost.exe	1880	Services	0	12,956 K
svchost.exe	1992	Services	0	20,448 K
svchost.exe	2000	Services	0	6,356 K
WUDFHost.exe	1064	Services	0	9,640 K
WUDFHost.exe	932	Services	0	2,408 K
svchost.exe	2384	Services	0	6,148 K
svchost.exe	2392	Services	0	1,784 K
svchost.exe	2416	Services	0	7,180 K
svchost.exe	2456	Services	0	11,112 K
svchost.exe	2476	Services	0	12,080 K
svchost.exe	2496	Services	0	10,352 K
svchost.exe	2692	Services	0	7,732 K
svchost.exe	2700	Services	0	7,600 K
svchost.exe	2708	Services	0	6,700 K

This tasklist command displays all active processes currently running on a system including process ID's and memory usage.

Command 2: dir /s

```
Directory of C:\Users\conno\.vscode\extensions\ms-vscode.powershell-2025.4.0\modules\PowerShellEditorServices\bin\Comma
nd
23/04/2026 10:27 <DIR> .
23/04/2026 10:27 <DIR> ..
23/04/2026 10:27 44,608 Mediat2.dll
23/04/2026 10:27 26,408 Microsoft.Bcl.AsyncInterfaces.dll
23/04/2026 10:27 362,684 Microsoft.CSharp.dll
23/04/2026 10:27 25,216 Microsoft.Extensions.Configuration.Abstractions.dll
23/04/2026 10:27 33,920 Microsoft.Extensions.Configuration.Binder.dll
23/04/2026 10:27 39,864 Microsoft.Extensions.Configuration.dll
23/04/2026 10:27 64,288 Microsoft.Extensions.DependencyInjection.Abstractions.dll
23/04/2026 10:27 91,440 Microsoft.Extensions.DependencyInjection.dll
23/04/2026 10:27 48,960 Microsoft.Extensions.FileSystemGlobbing.dll
23/04/2026 10:27 69,392 Microsoft.Extensions.Logging.Abstractions.dll
23/04/2026 10:27 84,560 Microsoft.Extensions.Logging.dll
23/04/2026 10:27 22,656 Microsoft.Extensions.Options.ConfigurationExtensions.dll
23/04/2026 10:27 68,994 Microsoft.Extensions.Options.dll
23/04/2026 10:27 50,488 Microsoft.Extensions.Primitives.dll
23/04/2026 10:27 40,184 Microsoft.PowerShell.EditorServices.deps.json
23/04/2026 10:27 583,200 Microsoft.PowerShell.EditorServices.dll
23/04/2026 10:27 169,480 Microsoft.PowerShell.EditorServices.pdb
23/04/2026 10:27 369,757 Microsoft.PowerShell.EditorServices.xml
23/04/2026 10:27 421,288 Microsoft.VisualStudio.Threading.dll
23/04/2026 10:27 34,776 Microsoft.VisualStudio.Validation.dll
23/04/2026 10:27 41,352 Microsoft.Win32.Registry.dll
23/04/2026 10:27 218,056 Newtonsoft.Streams.dll
23/04/2026 10:27 215,608 Newtonsoft.Json.dll
23/04/2026 10:27 324,072 OmniSharp.Extensions.DebugAdapter.dll
```

The dir command lists files and directories, the /s switch displays all files within subdirectories.

Command 3: Taskkill /PID 24172 /F

```
C:\Users\conno>taskkill /PID 24172 /F
SUCCESS: The process with PID 24172 has been terminated.
C:\Users\conno>
```

The taskkill command terminates a running process using its process ID and the /F switch forces the process to close.

Connor Johnson (2517511)

Command 4: ipconfig

```
C:\Users\conno>ipconfig

Windows IP Configuration

Wireless LAN adapter Wi-Fi 3:

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

Wireless LAN adapter Wi-Fi 4:

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

Wireless LAN adapter Wi-Fi:

    Connection-specific DNS Suffix  . : home.arpa
    IPv6 Address. . . . . : 2a0e:1d47:c405:7b00:73eb:f1fb:5d4e:58e0
    IPv6 Address. . . . . : 2a0e:1d47:c405:7b00:d1b4:c596:f1b:6e99
    Temporary IPv6 Address. . . . . : 2a0e:1d47:c405:7b00:e90e:7f09:9d3a:24dd
    Link-local IPv6 Address . . . . . : fe80::f5ba:1e26:507c:5072%16
    IPv4 Address. . . . . : 192.168.1.41
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : fe80::eefc:2fff:fe05:4d57%16
    192.168.1.1
```

The ipconfig command is used to display detailed information about the network.

Command 5, 6: net user, whoami

```
C:\Users\conno>net user

User accounts for \\CONNORSLAPTOP

-----
Administrator        conno                DefaultAccount
Guest                 WDAGUtilityAccount
The command completed successfully.

C:\Users\conno>whoami
connorslaptop\conno
```

The net user command lists all user accounts configured on this computer.

The whoami command displays the username of the currently logged in account.

Command 7: ping google.com

```
C:\Users\conno>ping google.com

Pinging google.com [2a00:1450:4009:c15::65] with 32 bytes of data:
Reply from 2a00:1450:4009:c15::65: time=17ms
Reply from 2a00:1450:4009:c15::65: time=16ms
Reply from 2a00:1450:4009:c15::65: time=17ms
Reply from 2a00:1450:4009:c15::65: time=16ms

Ping statistics for 2a00:1450:4009:c15::65:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 16ms, Maximum = 17ms, Average = 16ms
```

The ping command tests connectivity between the local computer and a remote host.

Command 8: schtasks /query

```
C:\Users\connor>schtasks /query

Folder: \
TaskName      Next Run Time      Status
=====
NVIDIA App SelfUpdate {82FE1952-8186-46C N/A      Ready
OneDrive Per-Machine Standalone Update T 14/05/2026 02:24:15 Ready
OneDrive Reporting Task-S-1-5-21-1185330 14/05/2026 01:42:51 Ready
OneDrive Startup Task-S-1-5-21-1185330 N/A      Ready
XRiteColorAssistantStartup N/A      Running

Folder: \GoogleSystem
TaskName      Next Run Time      Status
=====
INFO: There are no scheduled tasks presently available at your access level.

Folder: \Lenovo
TaskName      Next Run Time      Status
=====
INFO: There are no scheduled tasks presently available at your access level.

Folder: \Lenovo\InController
TaskName      Next Run Time      Status
=====
INFO: There are no scheduled tasks presently available at your access level.
```

The schtasks command displays scheduled tasks currently configured on the system.

Use of Linux OS CLI Commands

Command 1: ps aux

```
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1   0.0  0.0  100308 10876 ?        Ss   13:19   0:05 /sbin/init
root         2   0.0  0.0   3120  2176 ?        Ss   13:19   0:00 /init
root         7   0.0  0.0   3120  1792 ?        Ss   13:19   0:00 pland --control-socket 7 --l
root        76   3.5  0.1  39472 14976 ?        Ss   13:19   0:00 /lib/systemd/systemd-journald
root       104   4.3  0.0  23228  5888 ?        Ss   13:19   0:00 /lib/systemd/systemd-udev
root       119   0.0  0.0   4492  1284 ?        Ss   13:19   0:00 snapfuse /var/lib/snapd/snap
root       124  44.7  0.0   4720  1476 ?        R    13:19   0:02 snapfuse /var/lib/snapd/snap
root       125   6.5  0.0   5028  1208 ?        R    13:19   0:01 snapfuse /var/lib/snapd/snap
root       126   0.0  0.0   4624   772 ?        Ss   13:19   0:00 snapfuse /var/lib/snapd/snap
root       127   0.0  0.0   4492   772 ?        Ss   13:19   0:00 snapfuse /var/lib/snapd/snap
root       128   6.2  0.0   4948  1432 ?        R    13:19   0:01 snapfuse /var/lib/snapd/snap
systemd+   135   5.2  0.1  25532 13128 ?        Ss   13:19   0:01 /lib/systemd/systemd-resolve
root       170   0.1  0.0   4304  2560 ?        Ss   13:19   0:00 /usr/sbin/cron -f -P
message+   171   0.6  0.0   8620  4480 ?        Ss   13:19   0:00 dbus-daemon --system --addr
root       184   3.6  0.2  30100 18688 ?        Ss   13:19   0:00 /usr/bin/python3 /usr/bin/ne
syslog     185   0.5  0.0  222400  5376 ?        Ssl  13:19   0:00 /usr/sbin/rsyslogd -n -MHOM
root       188  16.4  0.3 1467084 32356 ?        Ssl  13:19   0:01 /usr/lib/snapd/snapd
root       192   2.2  0.0  15344  7552 ?        Ss   13:19   0:00 /lib/systemd/systemd-logind
root       271   4.7  0.0   4780  3200 ?        Ss   13:19   0:00 /bin/bash /snap/ubuntu-deskt
root       273   1.7  0.2 107220 21120 ?        Ssl  13:19   0:00 /usr/bin/python3 /usr/share/
root       281   0.0  0.0   3236  2176 hvc0    Ss+  13:19   0:00 /sbin/agetty -o -p -- lu --
root       282   0.0  0.0   3120  1028 ?        Ss   13:19   0:00 /init
root       283   0.1  0.0   3136  1156 ?        R    13:19   0:00 /init
lsac       285   0.8  0.0   6084  4864 pts/0    Ss   13:19   0:00 -bash
root       286   0.3  0.0   7516  4480 pts/1    Ss   13:19   0:00 /bin/login -f
root       289   0.0  0.0   3192  2176 tty1     Ss+  13:19   0:00 /sbin/agetty -o -p -- lu --
lsac       333   3.4  0.1  17056  9344 ?        Ss   13:19   0:00 /lib/systemd/systemd--user
lsac       335   0.0  0.0  103360  5064 ?        S    13:19   0:00 (sd-pam)
lsac       371   1.0  0.0   6120  4736 pts/1    Ss   13:19   0:00 -bash
root       485   5.8  0.0  14692  6656 ?        Ss   13:19   0:00 /lib/systemd/systemd-timedat
root       492  22.2  0.2 25496 17324 ?        D    13:19   0:01 /snap/ubuntu-desktop-install
lsac       502   0.0  0.0   7480  3072 pts/0    R    13:19   0:00 ps aux
```

The ps aux command displays all running processes including CPU usage, memory usage and process ID.

Command 2: top

```
top - 13:33:54 up 15 min, 1 user, load average: 0.17, 0.11, 0.11
Tasks: 33 total, 1 running, 32 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.4 us, 0.5 sy, 0.0 ni, 99.1 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
Mem Mem : 7941.2 total, 6659.6 free, 468.3 used, 813.3 buff/cache
Mem Swap: 2048.0 total, 2048.0 free, 0.0 used, 7319.4 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR   S %CPU  %MEM    TIME+  COMMAND
    1 root        0     0 101488 12156  8188  S   1.0   0.1   0:18.51 system
 590 root        0     0 44128  37608 10112  S   0.7   0.5   0:12.44 python3
    2 root        0     0   3120   2176  2048  S   0.0   0.0   0:00.17 init-systemd(Ub
    7 root        0     0   3120   1792   1792  S   0.0   0.0   0:00.00 init
   76 root       19    -1 47668 14976 14080  S   0.2   0:00.06 system-journal
 104 root        0     0 23228  5888  4480  S   0.1   0:00.03 system-udev
 119 root        0     0  4492  1284   1152  S   0.0   0:00.00 snapfuse
 124 root        0     0  4728 1452   1152  S   0.0   0:10.23 snapfuse
 125 root        0     0  4760 1284   896  S   0.0   0:02.88 snapfuse
 126 root        0     0  4624   772   640  S   0.0   0:00.00 snapfuse
 127 root        0     0  4492   772   640  S   0.0   0:00.00 snapfuse
 128 root        0     0  4968  1024   1024  S   0.0   0:00.51 snapfuse
 135 systemd+    0     0 25532 13128  9088  S   0.2   0:01.11 system-resolve
 170 root        0     0  4304  2560  2432  S   0.0   0:00.03 cron
 171 message+     0     0  8620  4480  3968  S   0.1   0:00.22 dbus-daemon
 184 root        0     0 30100 18688 10112  S   0.2   0:00.69 networkd-dispat
 185 syslog      0     0 222400  5376  4224  S   0.1   0:00.12 rsyslogd
 188 root        0     0 1467084 31896 19840  S   0.4   0:12.53 snapd
 192 root        0     0 15344  7552  6656  S   0.1   0:00.47 system-logind
 271 root        0     0  4780  3200  3072  S   0.0   0:00.01 subiquity-serve
```

The top command provides a real time view of running processes and system resource usage.

Command 3: ping

```
l4soac@JMGWL4Azure:~$ ping google.com
PING google.com (142.250.140.139) 56(84) bytes of data.
^C
--- google.com ping statistics ---
16 packets transmitted, 0 received, 100% packet loss, time 15353ms
```

The ping command test connectivity to a remote host.

Command 4: whoami

```
l4soac@JMGWL4Azure:~$ whoami
l4soac
```

The whoami command displays current logged in user.

Command 5: adduser testuser

```
l4soac@JMGWL4Azure:~$ sudo adduser testuser
[sudo] password for l4soac:
Adding user `testuser' ...
Adding new group `testuser' (1001) ...
Adding new user `testuser' (1001) with group `testuser' ...
Creating home directory `/home/testuser' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for testuser
Enter the new value, or press ENTER for the default
  Full Name []: test
  Room Number []: 55
  Work Phone []: 5678
  Home Phone []: 8
  Other []:
Is the information correct? [Y/n] y
```

The adduser command creates a new user account on the system.

Use of windows OS scripts

Windows PowerShell is a command-line scripting language used in the Windows operating system for automation, system administration, and performance monitoring. It allows users to run commands that interact directly with system processes, files, and hardware resources. PowerShell is widely used because it enables efficient management of operating system functions through scripting rather than manual navigation of graphical tools.

The script used in this task demonstrates process monitoring within the Windows operating system. It begins by displaying a heading using the Write-Host command, which outputs text to the terminal to improve readability and structure. The main functionality of the script is achieved using the Get-Process command, which retrieves a list of all currently running processes on the system.

The output of Get-Process is then passed through a pipeline (|) to the Where-Object command. This command filters the results based on a condition, in this case selecting only processes where CPU usage is greater than 10. This demonstrates how PowerShell can be used to analyse system performance and identify resource-intensive applications.

The filtered results are then passed to Select-Object, which is used to display only specific properties of each process, including the process name, CPU usage, and process ID. This helps to simplify the output and make it easier to interpret. Finally, Format-Table -AutoSize is used to display the data in a structured table format, improving readability.

Overall, this script demonstrates key features of Windows operating system scripting, including process management, data filtering, and output formatting. It shows how PowerShell can be used to monitor system performance and retrieve important information about running processes. This type of scripting is useful for system administrators who need to quickly identify applications consuming system resources and manage system efficiency effectively.

```
1 Write-Host "High CPU Processes"
2
3 Get-Process |
4 Where-Object {$_.CPU -gt 10} |
5 Select-Object Name, CPU, Id |
6 Format-Table -AutoSize
7
8
```

Name	CPU	Id
chrome	22.81	800
chrome	63.44	1628
chrome	12.25	8268
chrome	36.84	2472
chrome	61.56	728
chrome	684.11	204
chrome	236.69	672
chrome	4780.59	816
chrome	11.41	844
chrome	488.58	736
chrome	466.16	544
chrome	868.61	496
chrome	1459.41	448
chrome	82.08	668
chrome	2019.69	872
chrome	11.64	816
chrome	65.89	136
chrome	68.25	828
chrome	35.12	936

FIGURE 4 POWERSHELL SCRIPT SHOWING HIGH CPU PROCESS MONITORING CODE

Conclusion

In conclusion, this assignment demonstrates the key functions of modern operating systems, including process management, memory management, file handling, device management, and security. Using both Windows PowerShell and Linux Bash scripting, this shows that operating systems provide powerful tools for monitoring and managing system resources efficiently.

The Windows PowerShell script shows how processes can be filtered and analysed based on CPU usage, allowing users to identify resource-intensive applications and monitor system performance.

Overall, both operating systems provide effective command-line tools that support system administration and automation. These scripting environments improve efficiency by allowing users to perform complex tasks quickly without relying solely on graphical interfaces.

References

- [1] v. r. mirabueno. "Process-Memory-and-Storage-Management-Windows-vs-macOS.pdf." slideshare.net. Accessed: May 7 2026. [Online]. Available: <https://www.slideshare.net/slideshow/process-memory-and-storage-management-windows-vs-macos-pdf/277653754>
- [2] Microsoft. "NTFS overview." [Online]. Available: https://learn.microsoft.com/en-us/windows-server/storage/file-server/ntfs-overview?utm_source=chatgpt.com
- [3] Microsoft. "User interface overview." [Online]. Available: https://learn.microsoft.com/en-za/windows/apps/develop/user-interface?utm_source=chatgpt.com

L4 CA OS assignment

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

40/100

Total: 40

**1. Engagement
with Literature Skills:**

+ Some use of citations. + Well formatted IEEE reference list. – could be extended.

20 5

- could include 1/2 books/journal papers.

2. Knowledge and Understanding Skills: 4e

Key functions: + 20 9

Good overview of key some OS functions documented. +

Screenshots showing where some of the functions can be seen/monitored/managed are documented.

Win Commands: +

Good overview of some relevant commands that demonstrates an understanding of the language in all key areas. – Several additional commands could be documented and described here.

Bash Commands: + Overview of some relevant commands demonstrates some understanding of the language. – Some additional commands could be documented and described here.

Win Scripts: + Useful overview of some PS code. - Scripting aspects of the language have not been described.

Bash Scripts: - Not documented.

3. Cognitive and Intellectual Skills: 4+4+2

+ Evidence of some good analysis and understanding of OS functions.

10 9

+ Screenshots showing where some of the functions can be seen/monitored/managed are included in

the analysis.

+ Conclusion.

4. Practical

Application Skills: 10e:

Comm.: 7e + 3 all areas.

Sc: 7 + 3 Citations, Customisation & combo.

1.

40 14

Win Commands: +

Good use of some relevant Win. language commands. - could be extended to use several additional commands.

2. Bash Commands: + Some good use of relevant Bash commands. - could be extended to use some additional commands.

3. Win Scripts: + Some use of PS code documented. - Core script syntax areas not demonstrated.

4. Bash Scripts: - Not documented.

5. Transferable

Skills for Life and Professional Practice:

4+2+2+1+2

+ Report

10 3

format, structure, and presentation quite good.

- Report sections could be numbered.

- Each report section could be briefly introduced.

- Tutorial Activity 1 on OS

Types: relevant supplementary information on OS types not included in the Appendix of the report.

- Evidence of engagement with a relevant CPD course not documented in the appendix.

- Lecture activity evidence not documented.

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
