

Aidan Griffiths Operating Systems

by Aidan Griffiths

Submission date: 13-May-2026 10:54AM (UTC+0100)

Submission ID: 283351792

File name: 66131_Aidan_Griffiths_Aidan_Griffiths_Operating_Systems_684823_243174519.docx (3.55M)

Word count: 2517

Character count: 15478

Operating System Activity Document

Name: Aidan Griffiths

Student Number: 2501271

Module: Operating Systems

Course Name: Software Engineering

Contents

- Title Page
- Contents
- Operating System Functions
- Task 1
- Task 2
- Task 3
- Task 4
- Task 5
- Conclusion
- Appendix
- References

OS Functions

Processor Management: Manages the execution of processes.

Memory Management: Controls the allocation and deallocation of main memory to processes.

File Management: Organizes, stores, retrieves, and secures files on secondary storage devices, handling operations like creation, deletion, renaming, and access control.

Device Management: Coordinates communication with peripheral devices using device drivers, managing input/output operations, buffering, spooling, and efficient resource allocation.

Security and Protection: Safeguards system resources and user data from unauthorized access, employing authentication, access control, and protection mechanisms against malware and attacks.

TASK 1 – Operating system functions

Process Management

Process management is a managing process. A process is the running of a program until the life cycle of process is from the moment the program starts until it finishes. Some core functions in process management include scheduling, synchronization, deadlock handling and Inter-Process communication.

Memory Management

Memory management is how the operating system controls how data is stored and organised in main memory and secondary storage. It ensures programs get the memory required and can be cleaned (freed up) when it is no longer being used. Memory is used to allocate memory, protect processes from interfering with other memory, fragmentation handling and disk management.

Examples of Process & memory management

Task Manager

Task Manager

Type a name, publisher, or PID to search

Run new task

Processes

Name	Status	PID	Command line	1% CPU	81% Memory	0% Disk	0% Network
Apps (5)							
Brave Browser (26)				0%	1,443.1 MB	0 MB/s	0.1 Mbps
Microsoft PowerPoint		39444	"C:\Program Files\Microsoft Office\Root\Offi...	0%	270.3 MB	0 MB/s	0 Mbps
Microsoft Word		24996	"C:\Program Files\Microsoft Office\root\Offic...	0%	146.6 MB	0 MB/s	0 Mbps
Steam Client WebHelper (8)				0%	259.6 MB	0 MB/s	0 Mbps
Task Manager		39148	"C:\Windows\system32\taskmgr.exe" /4	0%	88.2 MB	0 MB/s	0 Mbps
Background processes (150)							
A Care Center User Process		14304	"C:\Program Files\AcerCCAgent\ACCUUserPS.e...	0%	0.1 MB	0 MB/s	0 Mbps
AAADSvc		5864	C:\Windows\System32\DriverStore\FileRepos...	0%	0.5 MB	0 MB/s	0 Mbps
Acer ART-AIMMX Service		5692	C:\Windows\System32\DriverStore\FileRepos...	0%	0.5 MB	0 MB/s	0 Mbps
Acer Care Center Software Co...		6220	C:\Windows\system32\AcerCCAgent.exe	0%	0.6 MB	0 MB/s	0 Mbps
Acer Device Enabling Service V2		5848	C:\Windows\System32\DriverStore\FileRepos...	0%	0.5 MB	0 MB/s	0 Mbps
Acer Device Info Software Co...		6196	C:\Windows\system32\AcerDIAgent.exe	0%	0.2 MB	0 MB/s	0 Mbps
Acer GAI Camera Service		6168	C:\Windows\System32\drivers\Acer\AcerGAI...	0%	0.5 MB	0 MB/s	0 Mbps
Acer GAI Camera Windows		32900	"C:\Windows\System32\drivers\Acer\AcerGA...	0%	1.1 MB	0 MB/s	0 Mbps
Acer Lighting Background App		28208	"C:\Windows\System32\DriverStore\FileRepo...	0%	0.2 MB	0 MB/s	0 Mbps
Acer Lighting Component		6232	C:\Windows\System32\DriverStore\FileRepos...	0%	1.9 MB	0 MB/s	0 Mbps
Acer Play Service		6148	C:\Windows\System32\DriverStore\FileRepos...	0%	1.0 MB	0 MB/s	0 Mbps
Acer Quick Access Software C...		6204	C:\Windows\system32\AcerQAAgent.exe	0%	1.1 MB	0 MB/s	0 Mbps
Acer Quick Access User Process		24984	"C:\Program Files\AcerQAAgent\AQAUUserPS...	0%	1.2 MB	0 MB/s	0 Mbps
Acer QuickPanel (3)				0%	13.8 MB	0 MB/s	0 Mbps

Command Prompt

```
Loading...

Welcome to Fedora 33 (riscv64)

[root@localhost ~]# ls
bench.py hello.c
[root@localhost ~]# ps
  PID TTY          TIME CMD
   47 hvc0      00:00:00 sh
   80 hvc0      00:00:00 ps
[root@localhost ~]#
```

```
top - 12:19:57 up 3 min, 0 users, load average: 0.00, 0.00, 0.00
Tasks: 20 total, 1 running, 19 sleeping, 0 stopped, 0 zombie
  CPU(s): 0.7 us, 0.0 sy, 0.0 ni, 99.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
  Mem: 182.0 total, 171.3 free, 4.9 used, 5.8 buff/cache
  Swap: 0.0 total, 0.0 free, 0.0 used, 172.2 avail Mem

  PID USER PR NI  VIRT  RES  SHR S  %CPU  %MEM    TIME+  COMMAND
    63 root  20   0   6024  3600 2524 R   0.7   1.7   0:00.23 top
    1 root  20   0   3136  2516 2148 S   0.0   1.4   0:00.32 init
    2 root  20   0   0      0   0   S   0.0   0.0   0:00.00 kthreadd
    3 root  20   0   0      0   0   S   0.0   0.0   0:00.00 kworker/0+
    4 root  0 -20   0      0   0   I   0.0   0.0   0:00.00 kworker/0+
    5 root  20   0   0      0   0   I   0.0   0.0   0:00.00 kworker/u+
    6 root  0 -20   0      0   0   I   0.0   0.0   0:00.00 mm_percpu+
    7 root  20   0   0      0   0   S   0.0   0.0   0:00.03 ksoftirqd+
    8 root  20   0   0      0   0   S   0.0   0.0   0:00.04 kdevtmpfs
    9 root  0 -20   0      0   0   I   0.0   0.0   0:00.00 netns
   10 root  20   0   0      0   0   S   0.0   0.0   0:00.00 oom_reaper
   11 root  0 -20   0      0   0   I   0.0   0.0   0:00.00 writeback
   12 root  0 -20   0      0   0   I   0.0   0.0   0:00.00 crypto
   13 root  0 -20   0      0   0   I   0.0   0.0   0:00.00 kblockd
   14 root  20   0   0      0   0   S   0.0   0.0   0:00.00 kswapd0
   15 root  20   0   0      0   0   I   0.0   0.0   0:00.00 kworker/0+
   32 root  20   0   0      0   0   S   0.0   0.0   0:00.00 khvcd
  42 root  20   0   1944  1472 1196 S   0.0   0.8   0:00.01 dhcpcd
  47 root  20   0   6204  2940 2380 S   0.0   1.6   0:00.28 sh
  78 root  20   0   0      0   0   I   0.0   0.0   0:00.00 kworker/u+
```

```
CPU[||||||| | |||||] 100.7% Tasks: 4, 0 thr; 1 running
Mem[||| 4.89M/182M Load average: 0.01 0.02 0.00
Swp[ 0K/0K Uptime: 00:06:33
```

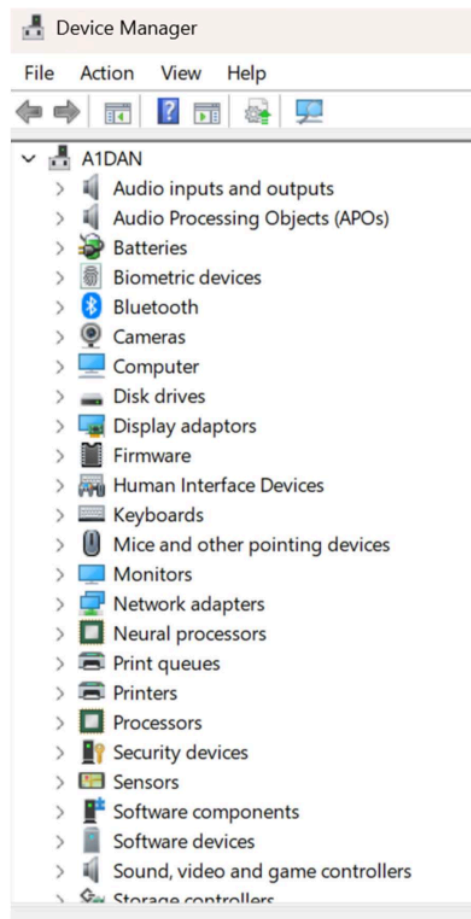
PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
82	root	20	0	3472	2564	2048	R	0.7	1.4	0:00.13	htop
1	root	20	0	3136	2516	2148	S	0.0	1.4	0:00.32	/bin/sh /sbin/init
42	root	20	0	1944	1472	1196	S	0.0	0.8	0:00.01	dhcpd
47	root	20	0	6204	2940	2380	S	0.0	1.6	0:00.29	sh -l

Device Management

Device management is how the operating system controls how the computer communicates with hardware such as printers and disks. Drivers and techniques are used to ensure efficient operation. Some components in device management include device drivers, buffering and caching. Operating systems use device drivers to communicate with devices such as GPUs, storage devices and keyboards.

Example of device management

Device Manager

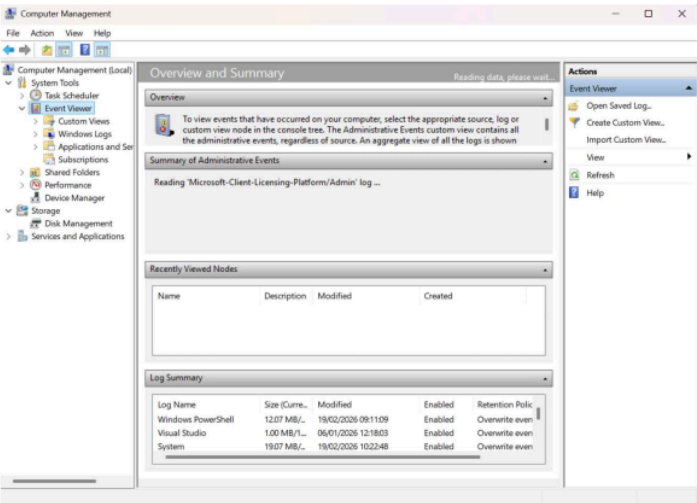
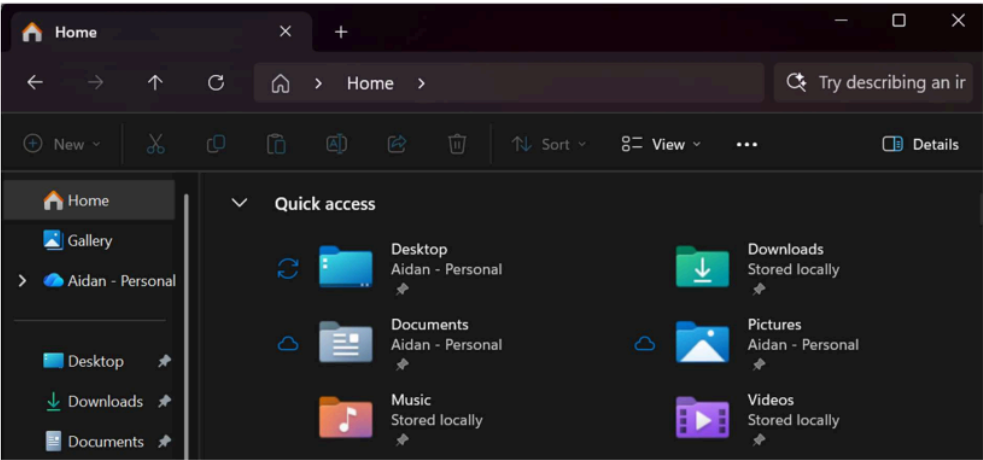


File Directory Management

File management controls how data is organised, named and stored in the computer's storage devices. This makes it easier for the user to navigate through their files of data. File directory management helps organise file attributes, file types, operations and access methods.

Examples of file directory management

File Explorer

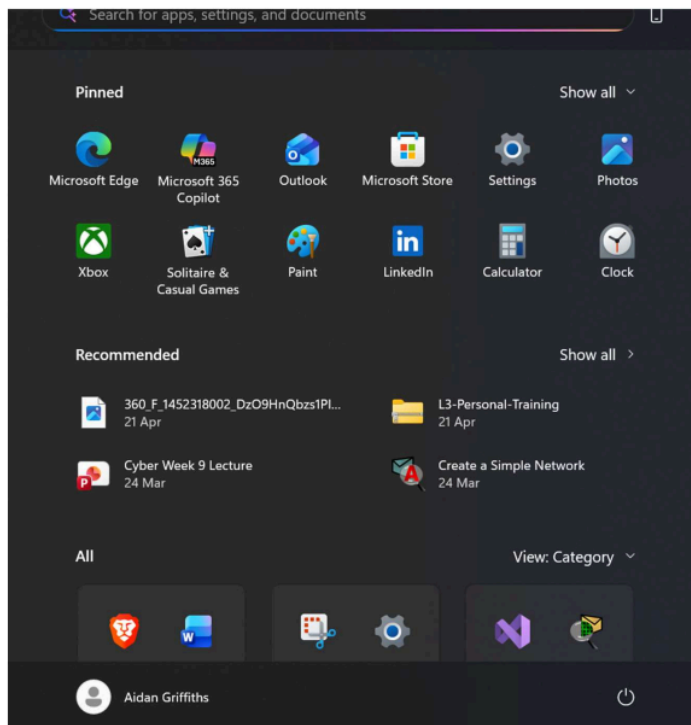


User Interface Management

The operating system itself provides the user interface which allows the user to interact with the computer's hardware and applications easier. There are different types of user interfaces. For example, Graphical User Interfaces (GUI) such as windows explorer or mac finder, Command Line Interfaces (CLI) such as PowerShell, Bash or Terminal. The user interface manages things like icons, menus, pointers, windows etc. The user interface will make the interface easier to understand by hiding the complexity of the computer and shows a friendly environment for browsing files, using programs and changing settings.

Examples of user interface management

Windows start menu

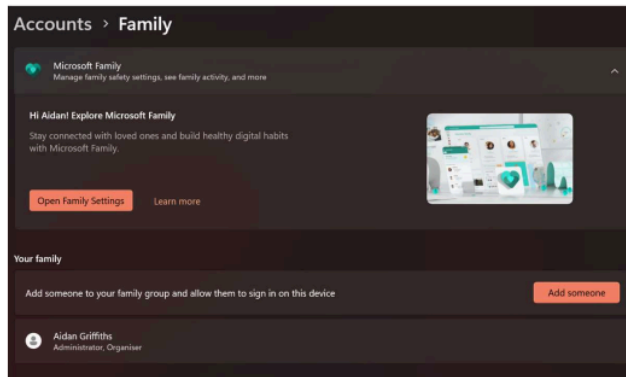


User Management

The user management allows multiple users with each of them having their own permissions, settings and profiles. The user management uses include things like creating and deleting accounts, managing passwords and authentication methods, controlling access to certain files and devices, and assigning roles.

Example of user management

Windows family users



Security & Access Control

Having a strong security ensures that only the authorised users and processes can access certain parts of the system and system recourses. OS security systems usually use things like passwords, biometrics and encryption. Windows, mac and Linux all have implanted access models to protect the files and devices on that system.

Example of Security & Access Control

Windows privacy and security settings page



Task 2

Process Management

Process management commands allow users to control and view the current running applications.

Task list

This command displays all active processes that include the memory usage and process IDs making it useful for monitoring the systems performance.

```
C:\Users\aidan>tasklist
```

Image Name	PID	Session Name	Session#	Mem Usage
System Idle Process	0	Services	0	8 K
System	4	Services	0	136 K
Secure System	236	Services	0	81,364 K
Registry	288	Services	0	41,176 K
smss.exe	732	Services	0	232 K
csrss.exe	1156	Services	0	2,492 K
wininit.exe	1356	Services	0	564 K
services.exe	1472	Services	0	7,484 K
lsass.exe	1516	Services	0	612 K
lsass.exe	1524	Services	0	14,848 K
svchost.exe	1664	Services	0	29,164 K
fontdrvhost.exe	1692	Services	0	316 K
WUDFHost.exe	1740	Services	0	2,496 K
svchost.exe	1836	Services	0	17,764 K
svchost.exe	1888	Services	0	4,208 K
WUDFHost.exe	1924	Services	0	184 K
svchost.exe	704	Services	0	3,136 K
svchost.exe	924	Services	0	896 K
landpmfservice.exe	940	Services	0	164 K
svchost.exe	1548	Services	0	3,764 K
svchost.exe	2192	Services	0	5,300 K
svchost.exe	2252	Services	0	2,512 K
svchost.exe	2288	Services	0	2,520 K
svchost.exe	2288	Services	0	5,444 K
svchost.exe	2336	Services	0	2,108 K
WUDFCompanionHost.exe	2428	Services	0	416 K
svchost.exe	2456	Services	0	3,172 K
svchost.exe	2480	Services	0	4,276 K
svchost.exe	2496	Services	0	992 K
svchost.exe	2552	Services	0	8,704 K

Task kill

This command can be used to forcefully terminate a task. The command gives the user direct control over the system activity without needing to use the task manager

```
C:\Users\aidan>taskkill /IM notepad.exe /F
SUCCESS: The process "Notepad.exe" with PID 50636 has been terminated.
C:\Users\aidan>
```

File & Directory Management

Listing Files and folders

Code: `dir /O:D /S`

The `dir` command lists files and folders in the current directory, including optional switches for sorting/showing hidden items.

```
27/04/2025 19:01      (1,099,811)  Img_1745776852755.jpg
12/08/2025 14:22      <DIR>          Work
27/11/2025 22:26      <DIR>          .
27/11/2025 22:26      <DIR>          Google Photos
29/01/2026 10:27      <DIR>          ..
12/03/2026 10:05      <DIR>          Snapchat
24/03/2026 10:49      <DIR>          Screenshots
24/03/2026 11:17      <DIR>          Camera
                3 File(s)          2,665,190 bytes

Directory of C:\Users\aidan\CrossDevice\Aidan's 523\storage\DCIM\Camera
01/05/2024 08:37      (3,099,258)  20240501_083709.jpg
22/05/2024 12:46      (5,547,427)  20240522_124632.jpg
22/05/2024 12:46      (5,269,552)  20240522_124633.jpg
22/05/2024 12:46      (5,714,924)  20240522_124636.jpg
23/05/2024 16:40      (4,178,291)  20240523_164027.jpg
24/05/2024 14:30      (1,516,666)  20240524_143020.jpg
12/06/2024 13:37      (869,654)   20240612_133709.jpg
12/06/2024 13:39      (1,053,660) 20240612_133711.jpg
12/06/2024 13:40      (1,550,151) 20240612_133943.jpg
12/06/2024 14:21      (2,050,137) 20240612_141943.jpg
14/06/2024 10:31      (892,552)   20240614_103156.jpg
25/06/2024 18:41      (2,533,578) 20240625_184153.jpg
29/06/2024 16:52      (5,379,320) 20240629_165253.jpg
29/06/2024 16:52      (5,646,294) 20240629_165256.jpg
29/06/2024 16:52      (5,076,048) 20240629_165259.jpg
29/06/2024 16:55      (5,463,264) 20240629_165520.jpg
29/06/2024 16:55      (4,027,118) 20240629_165521.jpg
29/06/2024 17:55      (4,869,436) 20240629_175519.jpg
29/06/2024 19:48      (5,596,567) 20240629_194806.jpg
29/06/2024 19:48      (5,783,710) 20240629_194809.jpg
29/06/2024 19:48      (4,900,276) 20240629_194812.jpg
29/06/2024 19:52      (17,651,950) 20240629_195234.mp4
29/06/2024 19:53      (28,601,567) 20240629_195309.mp4
29/06/2024 19:58      (19,550,480) 20240629_195754.mp4
29/06/2024 19:58      (9,872,139) 20240629_195823.mp4
```

PowerShell terminal - Removing Folders

Folders “testfolder” and “batchtest” was deleted

The `Remove-Item` command allows the user to delete files or folders including files which are protected. This command is important for automating file organisation and maintenance tasks

```
PS C:\Users\aidan> Remove-Item "testfolder"
PS C:\Users\aidan>
```

```
PS C:\Users\aidan> Remove-Item "batchtest"
PS C:\Users\aidan> |
```

Network Testing & Monitoring

Testing a connectivity to a host

The ping command checks the connectivity to a remote host and measures the response time.

```
C:\Users\aidan>ping google.com -n 5

Pinging google.com [142.250.117.139] with 32 bytes of
data:
Reply from 142.250.117.139: bytes=32 time=12ms TTL=110
Reply from 142.250.117.139: bytes=32 time=11ms TTL=110
Reply from 142.250.117.139: bytes=32 time=10ms TTL=110
Reply from 142.250.117.139: bytes=32 time=12ms TTL=110
Reply from 142.250.117.139: bytes=32 time=11ms TTL=110

Ping statistics for 142.250.117.139:
    Packets: Sent = 5, Received = 5, Lost = 0 (0% loss
),
    Approximate round trip times in milli-seconds:
        Minimum = 10ms, Maximum = 12ms, Average = 11ms

C:\Users\aidan>
```

Displays network configuration

The ipconfig /all command displays a detailed network configuration which includes the Ip addresses, DNS servers and MAC addresses.

```
C:\Users\aidan>ipconfig /all

Windows IP Configuration

    Host Name . . . . . : A1DAN
    Primary Dns Suffix . . . . . :
    Node Type . . . . . : Hybrid
    IP Routing Enabled. . . . . : No
    WINS Proxy Enabled. . . . . : No
    DNS Suffix Search List. . . . . : users.uwtsd.ac.
uk

Wireless LAN adapter WiFi 2:

    Media State . . . . . : Media disconnect
ed
    Connection-specific DNS Suffix . :
    Description . . . . . : MediaTek Wi-Fi
7 MT7925 Wireless LAN Card #2
    Physical Address. . . . . : 5A-02-05-76-F3-
BC
    DHCP Enabled. . . . . : No
    Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter WiFi 4:

    Media State . . . . . : Media disconnect
ed
    Connection-specific DNS Suffix . :
    Description . . . . . : MediaTek Wi-Fi
```

PowerShell terminal - Advanced network diagnostics

The Test-NetConnection command provides the user with deeper diagnostics by testing specific ports which makes it useful for troubleshooting web services or firewalls.

```
PS C:\Users\aidan> Test-NetConnection -ComputerName google.com -Port 443
```

```
ComputerName      : google.com  
RemoteAddress     : 142.250.117.139  
RemotePort        : 443  
InterfaceAlias    : WiFi  
SourceAddress     : 10.201.10.107  
TcpTestSucceeded  : True
```

User Management

Allows administrators to control local accounts

Displays or modifies user accounts

The net user command is used to list users or create new ones. However, the user must be in administrator or the access will be denied.

```
C:\Users\aidan>net user aidan /add
System error 5 has occurred.

Access is denied.
```

PowerShell Admin terminal

The Get-LocalUser command shows all local accounts similar to batch. Running this command in administrator showed the current user accounts. This command is important for managing permissions and securing multi-user systems.

```
PS C:\Users\aidan> Get-LocalUser

Name           Enabled Description
-----
Administrator  False  Built-in account for administering the computer/domain
aidan          True   A user account managed by the system.
DefaultAccount False  Built-in account for guest access to the computer/domain
Guest          False  Built-in account for guest access to the computer/domain
WDAGUtilityAccount False  A user account managed and used by the system for Windows Defender Application Guard scen...
```

Task Scheduling

Creating or managing scheduled tasks

Schtasks allows the user to use the /create option which allows them to automate scripts or programs at specific times.

```
C:\Users\aidan>schtasks /create /tn "BackupTask" /tr "backup.bat" /sc daily /st 18:00  
SUCCESS: The scheduled task "BackupTask" has successfully been created.
```

PowerShell terminal - Displaying scheduled tasks

The Get-ScheduledTask lists all existing scheduled tasks which enables automation of backups, monitoring tools and maintenance scripts.

```
PS C:\Users\aidan> Get-ScheduledTask | Select-Object TaskName, State
```

TaskName	State
AcerUserSensingLauncher	Ready
App Explorer	Running
BackupTask	Ready
Google Play Games Notifier	Ready
iGoAudioTaskSession	Running
OneDrive Per-Machine Standalone Update Task	Ready
OneDrive Reporting Task-S-1-5-21-1656181306-982531118-1243994317-1001	Ready
OneDrive Startup Task-S-1-5-21-1656181306-982531118-1243994317-1001	Ready
QuickPanelLauncher	Ready
StorPSCTL	Running
UbtFrameworkService	Disabled
UEIPInvitation	Ready
McAfee Windows Notification Token	Ready
Office Actions Server	Ready
Office Automatic Updates 2.0	Ready
Office Background Push Maintenance	Ready
Office ClickToRun Service Monitor	Ready
Office Feature Updates	Ready
Office Feature Updates Logon	Ready
Office Performance Monitor	Ready
Office Serviceability Manager	Ready
Office Startup Maintenance	Ready
BackgroundDownload	Ready
.NET Framework NGEN v4.0.30319	Queued
.NET Framework NGEN v4.0.30319 64	Queued
.NET Framework NGEN v4.0.30319 64 Critical	Disabled
.NET Framework NGEN v4.0.30319 Critical	Disabled
RecoverabilityToastTask	Ready
AD RMS Rights Policy Template Management (Automated)	Disabled
AD RMS Rights Policy Template Management (Manual)	Ready
PolicyConverter	Ready
VerifiedPublisherCertStoreCheck	Ready
MareBackup	Ready

Altogether, these commands show how using windows command line can provide a powerful and scripted control over the systems operations which makes them essential for system automation and administration.

Task 3

Linux Commands

Ps aux – Shows all the running processes on the system which includes the CPU usage, memory usage, user ownership and process IDs. This command is very useful for identifying resource-heavy or suspicious processes.

```
localhost:~# ps aux
PID   USER     TIME   COMMAND
  1  root      0:00   {init} /bin/sh /sbin/init
  2  root      0:00   [kthreadd]
  3  root      0:00   [pool_workqueue_]
  4  root      0:00   [kworker/R-rcu_g]
  5  root      0:00   [kworker/R-sync_]
  6  root      0:00   [kworker/R-kvfre]
  7  root      0:00   [kworker/R-slub_]
  8  root      0:00   [kworker/R-netns]
  9  root      0:00   [kworker/0:0-eve]
 10  root      0:00   [kworker/0:0H-ev]
 11  root      0:00   [kworker/0:1-eve]
 12  root      0:01   [kworker/u4:0-ev]
 13  root      0:00   [kworker/R-mm_pe]
 14  root      0:00   [ksoftirqd/0]
 15  root      0:00   [rcu_preempt]
 16  root      0:00   [rcu_exp_par_gp_]
 17  root      0:00   [rcu_exp_gp_kthr]
 18  root      0:00   [kdevtmpfs]
 19  root      0:00   [kworker/R-inet_]
 20  root      0:00   [rcu_tasks_kthre]
 21  root      0:00   [rcu_tasks_trace]
 22  root      0:00   [oom_reaper]
 23  root      0:00   [kworker/u4:1-ev]
 24  root      0:00   [kworker/R-write]
 25  root      0:00   [kcompactd0]
 26  root      0:00   [kworker/R-kbloc]
```

Top – Displays a real-time, updating list of processes. This being very similar task manager which shows CPU load, memory usage and which processes are using and consuming most resources.

```

Mem: 10540K used, 230512K free, 8K shrd, 0K buff, 972K cached
CPU:  0% usr  0% sys  0% nic 94% idle  0% io  0% irq  4% sirq
Load average: 0.03 0.02 0.00 1/39 108

```

PID	PPID	USER	STAT	VSZ	%VSZ	CPU	%CPU	COMMAND
108	66	root	R	1644	1%	0	1%	top
15	2	root	RW	0	0%	0	0%	[rcu_preempt]
66	1	root	S	1716	1%	0	0%	sh -l
1	0	root	S	1648	1%	0	0%	{init} /bin/sh /sbin/init
76	48	dhcpcd	S	1316	1%	0	0%	dhcpcd: [BPF ARP] eth0 10
48	47	root	S	1308	1%	0	0%	dhcpcd: [privileged proxy]
47	1	dhcpcd	S	1308	1%	0	0%	dhcpcd: [manager] [ip4]
49	47	dhcpcd	S	1288	1%	0	0%	dhcpcd: [network proxy]
50	47	dhcpcd	S	1284	1%	0	0%	dhcpcd: [control proxy]
44	1	root	S	860	0%	0	0%	settime -d /
12	2	root	IW	0	0%	0	0%	[kworker/u4:0-ev]
23	2	root	IW	0	0%	0	0%	[kworker/u4:1-ev]
14	2	root	SW	0	0%	0	0%	[ksoftirqd/0]
11	2	root	IW	0	0%	0	0%	[kworker/0:1-eve]
25	2	root	SW	0	0%	0	0%	[kcompactd0]
13	2	root	IW<	0	0%	0	0%	[kworker/R-mm_pe]
17	2	root	SW	0	0%	0	0%	[rcu_exp_gp_kthr]
16	2	root	SW	0	0%	0	0%	[rcu_exp_par_gp_]
19	2	root	IW<	0	0%	0	0%	[kworker/R-inet_]
18	2	root	SW	0	0%	0	0%	[kdevtmpfs]
20	2	root	IW	0	0%	0	0%	[rcu_tasks_kthre]
22	2	root	SW	0	0%	0	0%	[oom_reaper]
21	2	root	IW	0	0%	0	0%	[rcu_tasks_trace]
24	2	root	IW<	0	0%	0	0%	[kworker/R-write]
10	2	root	IW<	0	0%	0	0%	[kworker/0:0H-ev]
7	2	root	IW<	0	0%	0	0%	[kworker/R-slab_]

Kill -9 <PID> - Forcefully terminates a process using its ID (Process ID) or PID. The -9 sends a SIGKILL signal which immediately stops any unresponsive or frozen processes.

```
localhost:~# kill -9 43
```

```

28 root      0:00 [kswapd0]
29 root      0:00 [kworker/0:1H]
30 root      0:00 [khvcd]
31 root      0:00 [kworker/u4:3-ev]
46 dhcpcd    0:00 dhcpcd: [manager] [ip4]

```

File & Directory Management

Ls -l – Lists files in long format. It shows permissions, ownership, file size and timestamps. This helps users understand file access rights and directory structure.

```
localhost:~# ls -l
total 16
-rw-r--r--  1 root    root      114 Jan 12 19:26 bench.py
-rw-r--r--  1 root    root       76 Jan 12 19:26 hello.c
-rw-r--r--  1 root    root       22 Jan 12 19:26 hello.js
-rw-r--r--  1 root    root     204 Mar 11 13:41 readme.txt
```

Ls -la – Works the same as **ls -l** but this command includes hidden files which is useful for viewing configuration files and system-hidden items.

```
localhost:~# ls -la
total 28
drwxr-xr-x  2 root    root      167 Mar 18 12:21 .
drwxrwxrwx 21 root    root     461 May  5 11:57 ..
-rw-----  1 root    root      163 May  5 12:09 .ash_history
-rw-r--r--  1 root    root      114 Jan 12 19:26 bench.py
-rw-r--r--  1 root    root       76 Jan 12 19:26 hello.c
-rw-r--r--  1 root    root       22 Jan 12 19:26 hello.js
-rw-r--r--  1 root    root     204 Mar 11 13:41 readme.txt
```

Cp -r source_folder/ destination folder/ - Copies directories recursively, which means all files and subfolders are included. This is all commonly used for backups or duplicating project folders.

```
localhost:~# mkdir projectA
localhost:~# echo "file1" > projectA/file1.txt
localhost:~# echo "file2" > projectA/file2.txt
localhost:~# mkdir backup_projectA
localhost:~# cp -r projectA/ backup_projectA/
localhost:~# ls -l backup_projectA/
total 4
drwxr-xr-x  2 root    root      89 May  5  2026 projectA
```

Mkdir -p parent/child/grandchild – Creates nested directories in one single command. The -p option can prevent errors if the folders already exist.

```
localhost:~# mkdir -p parent/child/grandchild
localhost:~# ls -R parent
parent:
child

parent/child:
grandchild

parent/child/grandchild:
```

Network Testing & Monitoring Commands

Ping -c 4 8.8.8.8 – sends 4 ICMP packets to test internet connectivity and measure response time which is good for diagnosing network issues.

```
localhost:~# ping -c 4 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=2 ttl=115 time=340.000 ms
64 bytes from 8.8.8.8: seq=3 ttl=115 time=150.000 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 2 packets received, 50% packet loss
round-trip min/avg/max = 150.000/245.000/340.000 ms
```

Ip a – Displays all the network interfaces, IP addresses and MAC addresses. This is a more modern replacement for the ifconfig command so is essential for network configuration checks.

```
localhost:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 02:ed:c8:db:4c:39 brd ff:ff:ff:ff:ff:ff
    inet 10.5.33.20/16 brd 10.5.255.255 scope global dynamic noprefixroute eth0
        valid_lft 851sec preferred_lft 738sec
```

Netstat -tulnp – Shows all active network connections, listening ports and all processes using them. This command is useful for firewall checks and identifying running services.

```
localhost:~# netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
netstat: /proc/net/tcp6: No such file or directory
udp        0      0 0.0.0.0:68              0.0.0.0:*
49/dhpcpd: [network
netstat: /proc/net/udp6: No such file or directory
localhost:~#
```

Traceroute google.com – This command maps out the network path packets taken to reach a destination. This helps diagnose routing delays or connection bottlenecks.

```
localhost:~# traceroute google.com
traceroute to google.com (142.251.32.174), 30 hops max, 46 byte packets
 1  10.5.0.1 (10.5.0.1)  160.000 ms  150.000 ms  150.000 ms
 2  172.17.0.1 (172.17.0.1)  160.000 ms  150.000 ms  150.000 ms
 3  * * *
 4  10.84.66.14 (10.84.66.14)  200.000 ms  10.84.66.2 (10.84.66.2)  150.000 ms
10.84.66.12 (10.84.66.12)  150.000 ms
 5  143.198.253.132 (143.198.253.132)  160.000 ms  143.198.253.128 (143.198.253.
128)  150.000 ms  143.198.253.132 (143.198.253.132)  150.000 ms
 6  143.244.193.134 (143.244.193.134)  160.000 ms  143.244.193.126 (143.244.193.
126)  160.000 ms  143.244.193.124 (143.244.193.124)  150.000 ms
 7  ae12.sdr1.us-sjc1.as14061.net (143.244.227.98)  160.000 ms  ae7.sdr2.us-scj1
.as14061.net (143.244.224.14)  150.000 ms  ae11.sdr2.us-sjc1.as14061.net (143.24
4.227.100)  160.000 ms
 8  ae6.iar1.us-sjc1.as14061.net (143.244.225.49)  160.000 ms  ae5.iar1.us-sjc1.
as14061.net (143.244.225.45)  170.000 ms  ae6.iar1.us-sjc1.as14061.net (143.244.
225.49)  160.000 ms
 9  209.85.173.26 (209.85.173.26)  160.000 ms  160.000 ms  *
10  * * *
11  142.251.193.252 (142.251.193.252)  160.000 ms  142.251.241.104 (142.251.241.
104)  150.000 ms  142.251.66.108 (142.251.66.108)  150.000 ms
12  192.178.105.120 (192.178.105.120)  160.000 ms  192.178.105.112 (192.178.105.
112)  150.000 ms  66.249.94.29 (66.249.94.29)  150.000 ms
13  66.249.94.29 (66.249.94.29)  150.000 ms  192.178.99.125 (192.178.99.125)  16
0.000 ms  142.251.193.192 (142.251.193.192)  150.000 ms
14  108.170.234.26 (108.170.234.26)  170.000 ms  108.170.230.134 (108.170.230.13
4)  160.000 ms  142.251.68.52 (142.251.68.52)  170.000 ms
15  142.250.231.89 (142.250.231.89)  160.000 ms  74.125.243.89 (74.125.243.89)
160.000 ms  142.250.208.115 (142.250.208.115)  160.000 ms
16  142.251.79.37 (142.251.79.37)  170.000 ms  160.000 ms  170.000 ms
```

User Management

Sudo adduser testuser – Creates a new user account, sets a password and generates a home directory.

```
localhost:~# adduser test
Changing password for test
New password:
Bad password: too short
Retype password:
passwd: password for test changed by root
```

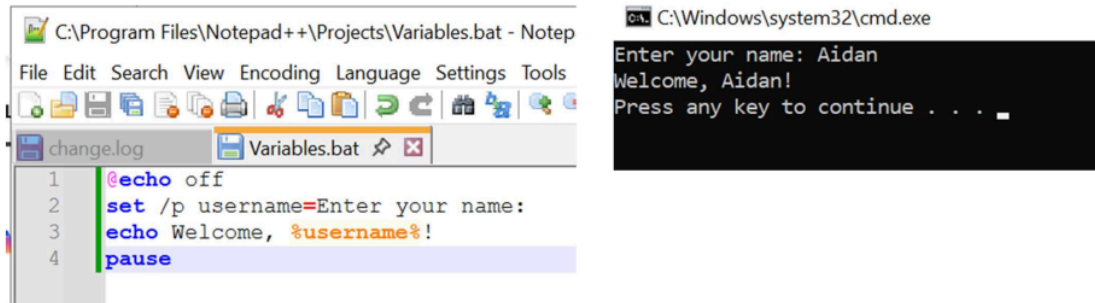
Task Scheduling

Crontab -e – Opens the system scheduler which allows automated tasks to be created. Adding a cron entry can allow a scheduling of a script to run at set intervals. The crontab -l verifies that the scheduled job has been saved successfully.

```
localhost:~# echo "echo 'Hello from uni' >>~/uni_output.txt > mycrchmod
njob.sh
> crontab -e
> ***** /home/yourusername/mycronjob.sh
> crontab -l
> ***** /home/aidan/mycronjob.sh
```

TASK 4

Variable



The screenshot displays two windows. On the left is a Notepad++ window titled 'C:\Program Files\Notepad++\Projects\Variables.bat - Notep'. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, and Tools. The toolbar shows various icons for file operations. The file list at the bottom includes 'change.log' and 'Variables.bat'. The script content is as follows:

```
1 @echo off
2 set /p username=Enter your name:
3 echo Welcome, %username%!
4 pause
```

On the right is a Command Prompt window titled 'C:\Windows\system32\cmd.exe'. It shows the output of the batch script:

```
Enter your name: Aidan
Welcome, Aidan!
Press any key to continue . . . .
```

This screenshot shows how the batch script uses variables to store and reuse the user's input. The scripts start with a set username which prompts the user to enter a value. Whatever the user enters is stored in that variable called %username%. The next script line will then output the message Welcome, %username% and replaces the %username% with the entered variable. This shows how batch scripts can interact with the user and use inputs to create the programs output. Variables are a fundamental of scripting and allows data to be reused, manipulated and captured.

Goto command

```
1  @echo off
2
3  echo Welcome to Aidan's Goto command!
4  echo.
5
6  :menu
7  echo 1. Say Hello
8  echo 2. Show Date
9  echo 3. Exit
10 set /p choice=Enter your choice (1-3):
11
12 :: Validate input
13 if "%choice%"=="1" goto hello
14 if "%choice%"=="2" goto showdate
15 if "%choice%"=="3" goto end
16
17 echo Invalid choice. Please try again.
18 echo.
19 goto menu
20
21 :hello
22 echo Hello, User!
23 echo.
24 goto menu
25
26 :showdate
27 echo Today's date is: %date%
28 echo.
29 goto menu
30
31 :end
32 echo Goodbye!
33 pause
```

Welcome to Aidan's Goto command!

1. Say Hello
2. Show Date
3. Exit

Enter your choice (1-3): 1
Hello, User!

1. Say Hello
2. Show Date
3. Exit

Enter your choice (1-3): 2
Today's date is: Thu 03/12/2026

1. Say Hello
2. Show Date
3. Exit

Enter your choice (1-3): 3
Goodbye!
Press any key to continue . . .

The goto command provides the user with options and control the program flow. The script shows a simple menu with 3 options for the user to choose. Depending on the input, the script will jump to the labelled sections such as the options above being hello, showdate and end all based on the user entering "1", "2", or "3". Each label has its own set of code which allows the script to behave like a menu program. This shows how batch scripts can create interactive tools that can respond to the users' selections and navigate between different areas of the script.

If Statements

If statements allow decision making based on certain conditions

Example:

```
Enter your username:Aidan
Welcome, Aidan!
Enter your age:19
You are an adult.
Press any key to continue . . . |
```

```
@echo off
set /p username=Enter your username:
echo Welcome, %username%!

set /p age=Enter your age:
if %age% GEQ 18 (
    echo You are an adult.
) else (
    echo You are under 18.
)

pause
```

This screenshot highlights a conditional logic which uses the “if” command. The script will ask the user for their age and the script will compare it using the “if %age% GEQ 18”. If the condition/the age entered is true which means the user is 18 or above, the script outputs “You are an adult”, otherwise it will output “You are under 18”. This shows how batch scripts can make decisions based on the user’s input.

Functions (CALL and Labels)

```
Enter your username:Aidan
Welcome, Aidan!
Enter your age:19
You are an adult.
Hello, Aidan!
Press any key to continue . . .
```

```
call :greet Aidan

pause

:greet
echo Hello, %1!

pause
```

This screenshot shows how batch scripts can use functions using call and labels. The script above calls :greet and passes the parameter, accessed using %1. That allows the code to be reused in a modular way which is like functions in other programming languages

In conclusion, batch scripting demonstrates the essential automation skills which enables user interaction, decision making, looping and modular code which makes windows command line tasks more efficient.

Task 5

Shebang and Basic Output

```
aidan@A1DAN:~$ #!/bin/bash
echo "This is my first Bash script!"
echo "Bash is running correctly."
This is my first Bash script!
Bash is running correctly.
```

This screenshot shows the fundamental structure of a bash script using the shebang and the echo command. The shebang will tell the system which interpreter should execute the script which ensures consistent behaviour regardless of the user's default shell. The script then uses echo to output a simple message which demonstrates how bash handles the basic output operations.

Variables and User Input

```
aidan@A1DAN:~$ #!/bin/bash
read -p "Enter your name: " username
echo "Welcome, $username!"
Enter your name: Aidan
Welcome, Aidan!
```

This screenshot shows how the bash script stores and reuses data using variables. Variables in bash don't require a type of declaration and are always assigned using name=value. The user input is taken using the read command and the stored value is accessed using the \$variable. This aligns with the standard bash behaviour which is described in multiple references which then emphasises variables as important for dynamic scripts behaviours.

IF Statements

```
aidan@A1DAN:~$ #!/bin/bash
read -p "Enter your age: " age

if [[ $age -ge 18 ]]; then
    echo "You are an adult."
elif [[ $age -eq 17 ]]; then
    echo "Almost there – one more year!"
else
    echo "You are under 18."
fi
Enter your age: 19
You are an adult.
```

This screenshot shows conditional logic using the if statements, elif statements and else statements. Bash support string and numeric comparisons which allows scripts to make decisions based off the user's input. The syntax follows the standard conditional format documented in the bash tutorials where the conditions are enclosed in the `[[ ]]` brackets and are then evaluated before being executed to the corresponding code block

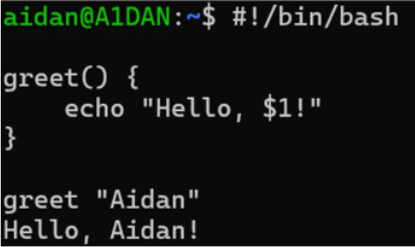
FOR Loop

```
aidan@A1DAN:~$ #!/bin/bash
echo "Listing all .txt files in this directory:"

for file in *.txt; do
    echo "Found file: $file"
done
Listing all .txt files in this directory:
Found file: *.txt
```

This screenshot highlights a for loop which allows bash to iterate over files, lists or commands. An example is looping through all the .txt files in the directory by using the “for file in *.txt; do... done”. Bash loops are used for automation tasks such as system monitoring, bash operations and file processing.

Functions

A terminal window with a black background and white text. The prompt is 'aidan@AIDAN:~\$' followed by the command '#!/bin/bash'. Below this, a function 'greet()' is defined with a body containing 'echo "Hello, \$1!"'. After the function definition, the command 'greet "Aidan"' is entered, and the output 'Hello, Aidan!' is displayed.

```
aidan@AIDAN:~$ #!/bin/bash  
  
greet() {  
    echo "Hello, $1!"  
}  
  
greet "Aidan"  
Hello, Aidan!
```

This screenshot shows how bash supports reusable code through functions which is defined using “name() {...}” and can be called like a regular command. Functions allow scripts to be maintainable and easier to expand. This is highlighted in bash scripting as a key part of writing structured and efficient scripts.

Conclusion

The use of both Windows and Linux scripting has shown how important command-line automation is across modern operating systems. By using Bash and bash scripts, it's clear that scripting is a technical and practical tool for reducing repetitive tasks, improving efficiency and understanding how operating systems behave beneath the graphical user interface. Using hands-on experimentation, the differences and similarities between the two script types are much easier to understand.

Windows Batch scripting shows how simple commands, variables, menu-driven structures and conditions can automate administrative tasks. Features like "if", "set", "goto" and "for" loops highlight how Windows handle user input, iteration and branching. The examples show that even lightweight scripting language can support meaningful automation and system interaction.

Linux bash scripting expands on these areas with a more powerful and flexible command-line interface. Bash supports structured programming through functions, advanced conditions and looping. The use of shebang, variables and user inputs shows how bash scripts integrate directly with the Linux shell. This shows the importance of scripting Linux-based systems where automation and command-line driven workflows are central to system administration.

Overall, the practical exercises on both operating systems highlight the value of scripting as a core in computing. Knowing and understanding how to write, debug and execute scripts will provide a strong foundation for future work in networking, software development, system administration and cyber security. The tasks completed in this task show technical understanding and the ability to apply scripting concepts in real scenarios.

Appendix

Computing relies on several categories of operating systems which are all designed to meet different performance, reliability and usability requirements. One of the most common categories is the General-Purpose Operating System which includes systems such as Windows, Linux and macOS. These operating systems are designed to maximise the overall system efficiency, support multitasking and provide a stable friendly user interface. The flexibility of this type of operating system makes it suitable for laptops, desktops and servers.

References

Task 1

GeeksforGeeks (2018). *Functions of Operating System*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/operating-systems/functions-of-operating-system/>.

Institute, Teachers. (2025). *Operating Systems Explained: Functions, Types, and Examples • Teachers Institute*. [online] Teachers Institute. Available at: <https://teachers.institute/computer-in-education/operating-systems-functions-types-examples/>.

IBM (2025). *Operating system*. [online] Ibm.com. Available at: <https://www.ibm.com/think/topics/operating-systems>.

Task 2

Xelu86 (2025). *net user*. [online] Microsoft.com. Available at: <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/net-user>.

JasonGerend (2026). *Windows commands*. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands>.

All commands were found in windows learn in the link above

Task 3

Manual, L. (2026). *ps(1) - Linux manual page*. [online] man7.org. Available at: <https://man7.org/linux/man-pages/man1/ps.1.html>.

Manual, L. (2026b). *top(1) - Linux manual page*. [online] man7.org. Available at: <https://man7.org/linux/man-pages/man1/top.1.html>.

Manual, L. (2026a). *kill(1) - Linux manual page*. [online] man7.org. Available at: <https://man7.org/linux/man-pages/man1/kill.1.html>.

Manual, L. (2025). *ping(8) - Linux manual page*. [online] Man7.org. Available at: <https://man7.org/linux/man-pages/man8/ping.8.html>.

Task 4

JasonGerend (2026b). *Windows commands*. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands>.

ss64.com. (n.d.). *An A-Z Index of Windows CMD commands - SS64.com*. [online] Available at: <https://ss64.com/nt/>.

Task 5

GeeksforGeeks (2022). *Bash Scripting Introduction to Bash and Bash Scripting*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/linux-unix/bash-scripting-introduction-to-bash-and-bash-scripting/>.

GeeksforGeeks (2022b). *Bash Scripting Working of Bash Scripting*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/linux-unix/bash-scripting-working-of-bash-scripting/>.

Devhints.io cheatsheets. (2000). *Bash scripting cheatsheet*. [online] Available at: <https://devhints.io/bash>.

GeeksforGeeks (2022a). *Bash Scripting Functions*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/linux-unix/bash-scripting-functions/>.

GeeksforGeeks (2017). *Introduction to Linux Shell and Shell Scripting*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/linux-unix/introduction-linux-shell-shell-scripting/>.

FINAL GRADE

GENERAL COMMENTS

77 /100

Total: 77

**1. Engagement
with Literature Skills:**

- No use of citations. + Formatted reference list. - IEEE format not used. 20 10

**2. Knowledge
and Understanding Skills: 4e**

Key functions: + 20 17

Good overview of key OS functions documented. - could be supported by citations. + Good use of screenshots used to show where some of the functions can be seen/monitored/managed documented.

Win Commands: +
Good overview of relevant commands that demonstrates an understanding of the language.

Bash Commands: + Good overview of relevant commands that demonstrates a sound understanding of the language.

Win Scripts: + Good overview of the syntax that demonstrates an understanding of the scripting aspects of the language.

Bash Scripts: + Good overview of the syntax that demonstrates a clear understanding of the scripting aspects of the language.

– some commands and scripts could be supported by citations to demonstrate research and enhanced learning.

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

+ Evidence of some good analysis and understanding of OS functions. 10 8

+ Screenshots showing where some of the functions can be seen/monitored/managed are included in the analysis. – could be supported by citations.

+ Conclusion.

4. Practical Application Skills: 10e

1. Win Commands: + Good use of relevant Win. language commands. 40 37

2. Bash Commands: + Good use of relevant Bash commands.

3. Win Scripts: + Good use of Win scripts to demonstrate applied knowledge of the core syntax.

4. Bash Scripts: + Good use of Bash scripts to demonstrate

applied knowledge of the core syntax.

- some commands and scripts could be supported by citations to demonstrate research and enhanced applied practical learning.

5. Transferable Skills for Life and Professional Practice:
4+2+2+1+2

+ Report 10 5
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 is included in the Appendix of the report. – could be extended.

- 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

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

