

Lloyd Case Computer Architecture and Operating Systems Coursework

by Lloyd Case

Submission date: 06-Apr-2023 06:28PM (UTC+0100)

Submission ID: 201654168

File name:

23995_Lloyd_Case_Lloyd_Case_Computer_Architecture_and_Operating_Systems_Coursework_480410_1524841169.docx
(1.41M)

Word count: 2429

Character count: 12812

LLOYD CASE 1818917 L4-
COMPUTER ARCHITECTURE
AND OPERATING SYSTEMS
(ACCA4022)

Lloyd Case (1818917)

Contents

Introduction	2
Key functions of general Operating Systems.....	2
Use of Windows OS CLI commands	3
Get-Command	4
New-Item	5
Get-CloudDrive	5
Get-Process	5
Get-Help	6
Get-Process Format-List	7
Get-ChildItem	8
Get-History	8
Get-AzSubscription.....	8
Read-Host.....	9
Use of Linux OS CLI commands	9
ls	9
echo.....	9
man	9
mkdir	9
cd.....	10
top.....	10
history	10
vmstat	10
w.....	11
whoami	11
Use of Windows OS scripts	11
Variables.....	11
Arrays	11
Hash Table.....	12
User input.....	12
Functions.....	12
Use of Linux OS scripts	13
Variables.....	13
Function	13
Number	13
User input.....	14

If Statement.....	14
Conclusion.....	15
References.....	15

Introduction

In this report I will be discussing the key functions of general operating systems, the use of Windows and Linux OS CLI commands, and OS scripts for Windows and Linux. I will be analysing the key and common functions of general operating systems. The use of Windows and Linux OS CLI commands I will be using 10 batch commands for Windows and 10 Bash commands for Linux. For the scripts I will be documenting the use of the Batch/Bash script.

Key functions of general Operating Systems

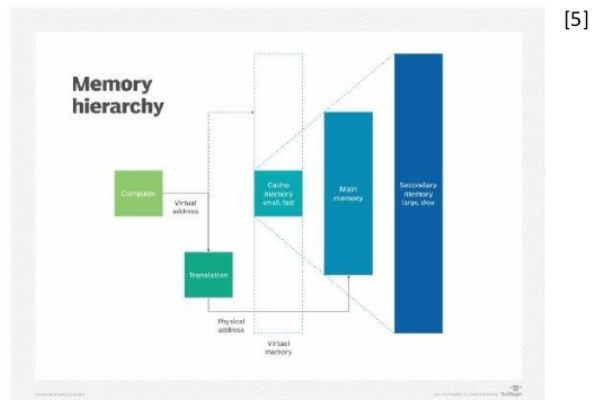
There are many key functions to an operating system such as security. Security is very important and necessary for any computer as this will allow you to have a password and to protect your data while also preventing access to hackers [1].

File management is another key function of an operating system. This allows you to manage the folders and files on the computer by being able to: create, rename, deleting, opening and closing the files [2]. You can also choose which users have access to which folders and files meaning you can create an optimised set up which allows all users to be able to access the files they need [2].

Another function is having control over your system performance, this allows you to see the health of your computer to see if it is performing well or not by tracking how fast the computer operates [3]. This can help when troubleshooting your computer because you will know what problems are occurring when a problem does occur [3].

Job accounting is another key function of an operating system, this allows the computer to keep track of what resources are being used as well as the time that is used for certain tasks that the users are using [4]. By doing this you will be able to see how much resources a user is using.

Memory management is another key function and what this does is control and coordinate the main memory (RAM) of the computer [5]. This means that the computer will use the required amount of memory (RAM) necessary for the applications and processes that are in use.



The computer can also create virtual memory (RAM), and this allows the computer to fully optimise the memory (RAM) usage and where the memory is allocated [5].

Error detecting aids allows the operating system to constantly be checking for errors which could crash your computer and cause it to shut down unexpectedly [1].

Device management is used to know and control the input and output devices that are connected to the computer [6]. This would require access to multiple resources such as RAM and file access to see if the required files are available to make sure your input and output devices are able to work correctly [6]. Devices such as a mouse and keyboard require the correct files for the computer to recognise the devices.

Use of Windows OS CLI commands

In this section I will be showcasing different commands that are available in PowerShell.

Get-Command

```
PS /home/lloyd> Get-Command
```

Cmdlet	Update-AzStorageFileServiceProperty	5.4.1	Az.Storage
Cmdlet	Update-AzStorageServiceProperty	5.4.1	Az.Storage
Cmdlet	Update-AzSupportTicket	1.0.0	Az.Support
Cmdlet	Update-AzSynapseFirewallRule	2.3.0	Az.Synapse
Cmdlet	Update-AzSynapseIntegrationRuntime	2.3.0	Az.Synapse
Cmdlet	Update-AzSynapseIntegrationRuntimeNode	2.3.0	Az.Synapse
Cmdlet	Update-AzSynapseLinkConnectionLandingZoneCredential	2.3.0	Az.Synapse
Cmdlet	Update-AzSynapseManagedVirtualNetworkConfig	2.3.0	Az.Synapse
Cmdlet	Update-AzSynapseSqlPool	2.3.0	Az.Synapse
Cmdlet	Update-AzSynapseSqlPoolAdvancedThreatProtectionSetting	2.3.0	Az.Synapse
Cmdlet	Update-AzSynapseSqlPoolVulnerabilityAssessmentSetting	2.3.0	Az.Synapse
Cmdlet	Update-AzSynapseSqlDatabase	2.3.0	Az.Synapse
Cmdlet	Update-AzSynapseSqlPool	2.3.0	Az.Synapse
Cmdlet	Update-AzSynapseSqlPoolAdvancedThreatProtectionSetting	2.3.0	Az.Synapse
Cmdlet	Update-AzSynapseSqlPoolVulnerabilityAssessmentSetting	2.3.0	Az.Synapse
Cmdlet	Update-AzSynapseWorkspace	2.3.0	Az.Synapse
Cmdlet	Update-AzTag	6.5.3	Az.Resources
Cmdlet	Update-AzureADSignedInUserPassword	0.0.0.10	AzureAD.Standard.Preview
Cmdlet	Update-AzWebRouteTable	5.5.0	Az.Network
Cmdlet	Update-AzVirtualApplianceSite	5.5.0	Az.Network
Cmdlet	Update-AzVirtualHub	5.5.0	Az.Network
Cmdlet	Update-AzVirtualHubEdgeConnection	5.5.0	Az.Network
Cmdlet	Update-AzVirtualHubVnetConnection	5.5.0	Az.Network
Cmdlet	Update-AzVirtualNetworkGatewayNatRule	5.5.0	Az.Network
Cmdlet	Update-AzVirtualRouter	5.5.0	Az.Network
Cmdlet	Update-AzVirtualRouterPeer	5.5.0	Az.Network
Cmdlet	Update-AzVirtualBian	5.5.0	Az.Network
Cmdlet	Update-AzVM	5.5.0	Az.Compute
Cmdlet	Update-AzVms	5.5.0	Az.Compute
Cmdlet	Update-AzVmsInstance	5.5.0	Az.Compute
Cmdlet	Update-AzVmsVM	5.5.0	Az.Compute
Cmdlet	Update-AzVpnConnection	5.5.0	Az.Network
Cmdlet	Update-AzVpnGateway	5.5.0	Az.Network
Cmdlet	Update-AzVpnGatewayNatRule	5.5.0	Az.Network
Cmdlet	Update-AzVpnServerConfiguration	5.5.0	Az.Network
Cmdlet	Update-AzVpnServerConfigurationPolicyGroup	5.5.0	Az.Network
Cmdlet	Update-AzVpnSite	5.5.0	Az.Network
Cmdlet	Update-AzWebAppAccessRestrictionConfig	2.13.0	Az.WebSites
Cmdlet	Update-AzWebAppTrafficRouting	2.13.0	Az.WebSites
Cmdlet	Update-FormatData	7.0.0.0	Microsoft.PowerShell.Utility
Cmdlet	Update-Help	7.3.3.500	Microsoft.PowerShell.Core
Cmdlet	Update-List	7.0.0.0	Microsoft.PowerShell.Utility
Cmdlet	Update-TypeData	7.0.0.0	Microsoft.PowerShell.Utility
Cmdlet	Use-AzHdiInsightCluster	5.0.1	Az.HdiInsight
Cmdlet	Wait-AzDataLakeAnalyticsJob	1.0.2	Az.DataLakeAnalytics
Cmdlet	Wait-AzHdiInsightJob	5.0.1	Az.HdiInsight
Cmdlet	Wait-AzRecoveryServicesBackupJob	6.3.0	Az.RecoveryServices
Cmdlet	Wait-AzSynapseSparkJob	2.3.0	Az.Synapse
Cmdlet	Wait-Debugger	7.0.0.0	Microsoft.PowerShell.Utility
Cmdlet	Wait-Event	7.0.0.0	Microsoft.PowerShell.Utility
Cmdlet	Wait-Job	7.3.3.500	Microsoft.PowerShell.Core
Cmdlet	Wait-Process	7.0.0.0	Microsoft.PowerShell.Management
Cmdlet	Where-Object	7.3.3.500	Microsoft.PowerShell.Core
Cmdlet	Write-Debug	7.0.0.0	Microsoft.PowerShell.Utility
Cmdlet	Write-Error	7.0.0.0	Microsoft.PowerShell.Utility
Cmdlet	Write-Host	7.0.0.0	Microsoft.PowerShell.Utility
Cmdlet	Write-Information	7.0.0.0	Microsoft.PowerShell.Utility
Cmdlet	Write-Output	7.0.0.0	Microsoft.PowerShell.Utility
Cmdlet	Write-Progress	7.0.0.0	Microsoft.PowerShell.Utility

This is the “Get-Command”, this shows every command that is available in PowerShell [7]. The list is too big to show, but here is just a short section of the list of commands.

New-Item

```
PS /home/lloyd> New-Item

cmdlet New-Item at command pipeline position 1
Supply values for the following parameters:
Path[0]: TEST 1
Path[1]: TEST 2
Path[2]: TEST 3
Path[3]: TEST 4
Path[4]: TEST 5
Path[5]:

Directory: /home/lloyd
```

UnixMode	User	Group	LastWriteTime	Size	Name
-rw-r--r--	lloyd	lloyd	4/6/2023 11:42	0	TEST 1
-rw-r--r--	lloyd	lloyd	4/6/2023 11:42	0	TEST 2
-rw-r--r--	lloyd	lloyd	4/6/2023 11:42	0	TEST 3
-rw-r--r--	lloyd	lloyd	4/6/2023 11:42	0	TEST 4
-rw-r--r--	lloyd	lloyd	4/6/2023 11:42	0	TEST 5

This is the “New-Item” command [7], this creates a new item as seen in the image above, I created 5 test items, and it shows that they were successfully created.

Get-CloudDrive

```
PS /home/lloyd> Get-Cloudrive
```

FileShareName	:	cs-1818917-student-uwtsd-ac-uk-10032000ca478103
FileSharePath	:	///csb10032000ca478103.file.core.windows.net/cs-1818917-student-uwtsd-ac-uk-10032000ca478103
MountPoint	:	/home/lloyd/cloudrive
Name	:	csb10032000ca478103
ResourceGroupName	:	cloud-shell-storage-westernaustralia
StorageAccountName	:	csb10032000ca478103
SubscriptionId	:	c5e1e442-f1da-43d8-ab8e-adfdb64cd0f5

This is the “Get-Cloudrive” command, and this shows all the details of the PowerShell I am using as I am using Azure [8].

Get-Process

```
PS /home/lloyd> Get-Process
```

NPM(K)	PM(M)	WS(M)	CPU(s)	Id	SI	ProcessName
0	0.00	133.50	7.87	21	1	node
0	0.00	502.25	49.63	119	116	pwsh
0	0.00	2.61	0.00	118	116	runuser
0	0.00	3.19	0.02	1	1	startNode.sh
0	0.00	2.53	0.00	116	116	startPwsh.sh

This is the “Get-Process” command, and this shows a list of the processes that are running, however as this is online its only what is used on their end to run this PowerShell command line to me [9].

Get-Help

```
PS /home/lloyd> Get-Help
TOPIC
    Getting started with Powershell in Azure Cloud Shell.

SHORT DESCRIPTION
    Explains new concepts of PowerShell in Azure Cloud Shell.

LONG DESCRIPTION
    PowerShell in Azure Cloud Shell brings the familiar Powershell experience
    along with the following set of new capabilities:

AZURE DRIVE
    The Azure drive (Azure:) enables easy navigation of Azure resources such
    as Compute, Network, Storage etc. similar to filesystem navigation.

MODULES
    In addition to the built-in Powershell modules, Powershell in Azure Cloud Shell
    comes with all Az modules pre-installed.

TOOLS
    PowerShell in Azure Cloud Shell comes pre-installed with tools such as Vim, Nano,
    Git, Python, and SQLCMD. For a complete list visit https://aka.ms/cloudshell/powershell-tools

ONLINE HELP
    You can find help for PowerShell in Azure Cloud Shell online at https://aka.ms/cloudshell/powershell-docs

EXAMPLES:
    Get-CloudDrive           : List information of the Azure File storage share that is mounted as 'CloudDrive'
    Dismount-CloudDrive      : Dismounts Azure File storage share from the current session
    Get-AzCommand            : Gets all the context specific Azure commands when invoked from Azure PowerShell drive
    Invoke-AzVMCommand       : Runs commands on Azure VMs
    Enable-AzVMPSRemoting    : Enable all aspects of PowerShell remoting on the given target (NSG Rules, Target WinRM/SSH configs)
    Disable-AzVMPSRemoting   : Disable all aspects of PowerShell remoting on the given target (NSG Rules, Target WinRM/SSH configs)

SEE ALSO:
    Get-CloudDrive
    Dismount-CloudDrive
    Get-AzCommand
    Invoke-AzVMCommand
    Enable-AzVMPSRemoting
    Disable-AzVMPSRemoting
```

The “Get-Help” command is used to get some help understanding what you are doing as it will show you all different ways and different commands you can use.

Get-Process | Format-List

```
PS /home/lloyd> Get-Process | Format-List
```

```
Id      : 21
Handles :
CPU     : 8.32
SI      : 1
Name    : node
```

```
Id      : 119
Handles : 419
CPU     : 55.77
SI      : 116
Name    : pwsh
```

```
Id      : 118
Handles :
CPU     : 0
SI      : 116
Name    : runuser
```

```
Id      : 1
Handles :
CPU     : 0.02
SI      : 1
Name    : startNode.sh
```

```
Id      : 116
Handles :
CPU     : 0
SI      : 116
Name    : startPwsh.sh
```

This is the “Get-Process” command followed by the “Format-List [10]” command, and what this does is display the same info but in a different way, comparatively to the list without the “Format-List” command.

Get-ChildItem

```
PS /home/lloyd> Get-ChildItem

Directory: /home/lloyd

UnixMode User          Group          LastWriteTime         Size Name
-----
drwxr-xr-x lloyd         lloyd          4/6/2023 18:49         22 clouddrive -> /usr/cuser/clouddrive
drwxr-xr-x lloyd         lloyd          4/5/2023 23:06        4096 Microsoft
-rw-r--r-- lloyd         lloyd          4/6/2023 11:41           0 clear
-rw-r--r-- lloyd         lloyd          4/6/2023 11:41           0 clear-host
-rw-r--r-- lloyd         lloyd          4/6/2023 11:41           0 Hello
-rw-r--r-- lloyd         lloyd          4/6/2023 11:42           0 TEST 1
-rw-r--r-- lloyd         lloyd          4/6/2023 11:42           0 TEST 2
-rw-r--r-- lloyd         lloyd          4/6/2023 11:42           0 TEST 3
-rw-r--r-- lloyd         lloyd          4/6/2023 11:42           0 TEST 4
-rw-r--r-- lloyd         lloyd          4/6/2023 11:42           0 TEST 5
```

This is the “Get-ChildItem” command, and this command allows you to view the different files and folders that are available in the working directory [11].

Get-History

```
PS /home/lloyd> Get-History

Id      Duration CommandLine
--      -
1        4.478 Get-Command
2         0.003 Get-Command out-host
3       16.361 New-Item
4         0.047 clear-host
5       16.506 New-Item
6         4.778 move-item
7         0.052 mkdir
8         0.035 mkdir
9       16.532 help
10        0.075 Get-Clouddrive
11       3.279 Import-Alias
12        0.029 Get-Process
13        0.008 Get-Help
14       0.112 Start-Process Word
15       0.013 Get-Process | Format-List
16       0.019 Get-ChildItem
```

The “Get-History” command allows you to view all the previous commands you have inputted and allows you to quickly input them again via the number on the left.

Get-AzSubscription

```
PS /home/lloyd> Get-AzSubscription

Name      Id      TenantId      State
-----
Azure subscription 1 c5e1e442-f1da-43d8-ab8e-adfdb64cd0f5 4e0f11f9-046e-4505-9cb8-db2152311e21 Enabled
```

This command “Get-AzSubscription” shows the status of my current subscription to Microsoft Azure.

Read-Host

```
PS /home/lloyd> Read-Host -Prompt "Hello, how are you today?"
Hello, how are you today?: I am feeling great!
I am feeling great!
```

This command allows the user to input a prompt “Read-Host -Prompt “Hello, how are you today?” the user can then respond to the prompt as shown in the image.

Use of Linux OS CLI commands

In this section I will be showcasing different commands that are available in Bash.

ls

```
lloyd [ ~ ]$ ls
clear  clear-host  clouddrive  Hello  Microsoft  'TEST 1'  'TEST 2'  'TEST 3'  'TEST 4'  'TEST 5'
```

The command used here is “ls” this simply list all the files in a certain directory [12].

echo

```
lloyd [ ~ ]$ echo "Hi this is Lloyd Case 1818917"
Hi this is Lloyd Case 1818917
```

The “echo” command displays what the user has input in the brackets after the echo, such as “echo “Hi this is Lloyd Case 1818917””. This will then be displayed [12].

man

```
lloyd [ ~ ]$ man
What manual page do you want?
For example, try 'man man'.
lloyd [ ~ ]$ man man
```

```
MAN(1)                                Manual page utils

NAME
    man - an interface to the system reference manuals

SYNOPSIS
    man [man options] [function] name ...
    man -k [man options] cman ...
    man -k [man options] [section] term ...
    man -f [whatis options] name ...
    man -l [man options] file ...
    man -M [man options] name ...

DESCRIPTION
    man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the available sections following a pre-defined order (see DEFAULTS), and to show only the first page found, even if page exists in several sections.

    The table below shows the section numbers of the manual followed by the types of pages they contain.

    1 Executable programs or shell commands
    2 System calls (functions provided by the kernel)
    3 Library calls (functions within program libraries)
    4 Special files (usually found in /dev)
    5 File formats and conventions, e.g. /etc/passwd
    6 Games
    7 Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7), man-pages(7)
    8 System administration commands (usually only for root)
    9 Kernel routines (non standard)
```

This command allows you to get help and see the manual which displays what specific commands do in case you need help figuring out which command does what. This displayed the whole manual for Bash commands [12].

mkdir

```
lloyd [ ~ ]$ mkdir test1
```

The “mkdir” is a command used when you want to create a new directory, in the image I created a directory named test1 [13].

cd

```
lloyd [ ~ ]$ cd test1
lloyd [ ~/test1 ]$
```

In this image I am using the “cd” command which is used to navigate the directory [12], in this case the directory is test1 and in the image we can see that I was successfully able to access the directory.

top

```
top - 12:42:50 up 2:16, 0 users, load average: 0.26, 0.35, 0.29
tasks: 12 total, 1 running, 11 sleeping, 0 stopped, 0 zombie
%Cpu0 :  2.7/1.3      4[ ]  %Cpu1 :  1.4/0.7      2[ ]
GiB Mem : 38.2/13.8
GiB Swap: 0.0/0.0

  PID USER      PR  NI    VIRT    RES    %CPU    %MEM     TIME+ S COMMAND
    1 root        20   0    7.6m    3.9m    0.0    0.1   0:00.01 S /bin/bash ./linux/startNode.sh
   21 root        20   0   902.2m 140.2m    0.7    3.6   0:03:77 S   node --max-http-header-size=16000 app.js
  116 root        20   0    7.6m    3.0m    0.0    0.1   0:00.00 S   /bin/bash /usr/cloudshell/linux/startPush.sh lloyd
  118 root        20   0    7.4m    2.9m    0.0    0.1   0:00.00 S   /bin/runuser lloyd --session-command /usr/bin/push -Hologo
  119 lloyd       20   0   13.8g 372.3m    0.7    9.5   0:12:51 S   /usr/bin/push -Hologo
  232 root        20   0    7.6m    3.2m    0.0    0.1   0:00.00 S   /bin/bash /usr/cloudshell/linux/startPush.sh lloyd
  234 root        20   0    7.4m    3.0m    0.0    0.1   0:00.00 S   /bin/runuser lloyd --session-command /usr/bin/push -Hologo
  235 lloyd       20   0   13.8g 374.8m    0.7    9.5   0:09:05 S   /usr/bin/push -Hologo
  261 root        20   0    7.6m    3.1m    0.0    0.1   0:00.00 S   /bin/bash /usr/cloudshell/linux/start.sh lloyd
  263 root        20   0    7.4m    3.0m    0.0    0.1   0:00.00 S   /bin/runuser -u lloyd /bin/bash
  264 lloyd       20   0    7.9m    4.3m    0.0    0.1   0:00.01 S   /bin/bash
  302 lloyd       20   0    7.1m    3.0m    0.0    0.1   0:00.00 S   top
```

This is the “top” command, and this command is used to show what processes you are running and the resources they take, allowing you to monitor if they are working correctly and that there is no memory overflow [13].

history

```
lloyd [ ~ ]$ history
 1 ls
 2 echo "Hello World"
 3 echo "Hi this is Lloyd Case 1818917"
 4 man
 5 man man
 6 clear
```

The “history” command is used in the same way as it is in PowerShell in that it displays the history of all the previous commands you have used [12].

vmstat

```
lloyd [ ~ ]$ vmstat
procs -----memory----- --swap-- ----io---- -system-- -----cpu-----
 r b swpd free buff cache si so bi bo in cs us sy id wa st
 5 1  0 119068 268192 1997188  0  0 219 157 141 829 2 2 93 3 0
```

The “vmstat” command shows the user a statistically detailed report of the virtual memory that is currently in use [14].

w

```
lloyd [ ~ ]$ w
12:53:28 up 2:26, 0 users, load average: 1.11, 0.66, 0.42
USER      TTY      LOGIN@  IDLE   JCPU   PCPU WHAT
```

The “w” command shows which users are currently logged in and also displaying their current activity. In the image there are no users as there aren’t any in the specified group [14].

whoami

```
lloyd [ ~ ]$ whoami
lloyd
```

The “whoami” command, very simply displays your username to show you that you are you [14].

Use of Windows OS scripts

In this section I will be showcasing different scripts that are available in PowerShell.

Variables

```
PS /home/lloyd> $username = "Hello my name is lloyd"
PS /home/lloyd> echo $username
Hello my name is lloyd
```

This script is for variables [15]. Shown in the image above is a variable in action with the \$username. The variable \$username is being used to hold the information “Hello my name is Lloyd”. When the echo command is used, we can see that everything in \$username is being displayed.

Arrays

```
PS /home/lloyd> $ArrayTest = (
>> 1,
>> 2,
>> 3
>> )
```

This is a script demonstrating the use of arrays in PowerShell. In this we can see that I have created a list of the numbers 1,2 and 3 [16]. Using the variable \$ArrayTest we can store the table of numbers.

```
PS /home/lloyd> echo $Arraytest
1
2
3
```

This is the array in an echo command, so its being displayed and this is what it looks like, as we can see its simple as it is just a list that goes down in a line displaying all that was in the \$Arraytest variable.

Hash Table

```
PS /home/lloyd> $hashtable = @{  
>> NAME = "Lloyd"  
>> STID = 1818917  
>> AGE = 21  
>> }
```

A hash table is used to display information in a table. Using the by using \$hashtable I am able to write out the code for what I want displayed in the table. NAME, STID and AGE are all variables that were used to display my name, student ID and age [17].

```
PS /home/lloyd> $hashtable  
  
Name      Value  
----      -  
AGE       21  
STID      1818917  
NAME      Lloyd
```

As shown above the \$hashtable is being displayed and it is showing the information in a nicely organised table that displays the name and the value.

User input

```
PS /home/lloyd> $userAge = Read-Host "Age"  
Age: 21
```

This script allows for user input [18]. In this example you are asked your age, you will then be prompted to input your age.

```
PS /home/lloyd> Write-Output "Your age is: ${userAge}"  
Your age is: 21
```

After writing the "Write-Output" command we can then see that the age number has been stored and we can see that it now displays your age.

Functions

```
PS /home/lloyd> function TheSum {  
>> $Sum = 5 * 10  
>> Write-Output $Sum  
>> }
```

Lastly is functions, this is another key feature in Windows scripting as it allows the user to store multiple commands within a single variable[18]. In this example we can see that it allows for a sum to be completed.

```
PS /home/lloyd> TheSum  
50
```

As the image previously showed, the equation was $5 * 10$ and the program was able to work out the sum and through inputting "Thesum" (the name of the function) we can see that the number 50 is correctly displayed.

Use of Linux OS scripts

In this section I will be showcasing different scripts that are available in Bash.

Variables

```
FIRST='Hello, my name is '  
Name='Lloyd'  
Age=21  
  
echo $FIRST $Name and I am $Age years old
```

The first script is with variables [19]. Variables are used to set what will be displayed. In the example above we can see that FIRST will display "Hello, my name is" with the use of the echo command as well as the \$FIRST to actually display that specific variable. The same is true for Name and Age, they have a set variable, and, in the echo, command are set by \$Name and \$Age.

```
> bash main.sh  
Hello, my name is Lloyd and I am 21 years old
```

This is then what is displayed once you run the script.

Function

```
username()  
{  
    echo "Lloyd"  
}  
echo "Your name is ${username}"
```

This is a function script. This is used for when you want multiple lines of code to work within a single variable. In this example "username" is used to hold echo "Lloyd", then this is displayed at the end when the variable \${username} is used [20].

```
> bash main.sh  
Your name is Lloyd
```

This is what will be displayed, as you can see the function has worked correctly and does display what was in the username function.

Number

```
Number1=5  
Number2=10  
Answer=$(( $Number1 + $Number2 ))  
echo Answer = "$Answer"
```

This is a script for displaying numbers. As you can see it is very simple and follows the same syntax as the other scripts shown, where a \$ is needed in front of the variable to display in the echo statement at the end. This script will allow the numbers to be added together to display 15.

```
> bash main.sh
Answer = 15
```

This is what it displays, as you can see it has correctly done the maths which was 5+10, as it gave the answer 15.

User input

```
echo "How old are you?"
read Age
echo "Wow, you are very brave for a $Age year old!"
```

This is a script of using user input [17]. This is a great way to make a sort of game if you wanted to (a text based one) as you are able to input whatever you like. In this script you can see it asks for your age and you will simply input your age.

```
> bash main.sh
How old are you?
21
Wow, you are very brave for a 21 year old!
```

Displayed here is the code in action and an age displayed along with the statement displayed after you enter your age.

If Statement

```
echo "What year was I born? "
read YearBorn
if [ $YearBorn == 2002 ];
then
echo "Hooray! You got it!."
else
echo $YearBorn is unfortunately incorrect."
fi
```

This is an if statement [19], this allows you to have two different outcomes to your coding. In the example above you can see that it asks to guess what year I was born, you can see in the coding that the correct answer is 2002.

```
> bash main.sh
What year was I born?
2002
Hooray! You got it!.
```

If correct this will be displayed.

```
> bash main.sh
What year was I born?
2000
2000 is unfortunately incorrect.
```

If incorrect this will be displayed.

Conclusion

In conclusion I have discovered that both the Windows and Linux Batch/Bash are very similar and sometimes use the same syntax, however PowerShell was much more difficult for me to understand due to some syntax being different between the two, with Linux seemingly simplifying the process it seems.

I have learned that there are many things an operating system must have for it to be safe and secure, with Windows everything is set up for you, if you were to get a Linux machine you must make sure that you have all the security features are in place otherwise you could be susceptible to an attack.

References

- [1] "Functions of Operating Systems," Geeks for Geeks, 13 March 2023. [Online]. Available: <https://www.geeksforgeeks.org/functions-of-operating-system/>. [Accessed 29 March 2023].
- [2] M. Narang, "Functions of Operating Systems," Shiksha Online, 17 February 2023. [Online]. Available: <https://www.shiksha.com/online-courses/articles/functions-of-operating-system/#Functions-of-operating-system>. [Accessed 29 March 2023].
- [3] "Functions of Operating Systems," BYJU, 2023. [Online]. Available: <https://byjus.com/gate/functions-of-operating-system-notes/#what-are-the-functions-of-operating-systems>. [Accessed 29 March 2023].
- [4] B. Priya, "What are the functions of the operating system?," Tutorials Point, 25 November 2021. [Online]. Available: <https://www.tutorialspoint.com/what-are-the-functions-of-the-operating-system>. [Accessed 29 March 2023].
- [5] R. Sheldon, "memory management," WhatIs, [Online]. Available: <https://www.techtarget.com/whatis/definition/memory-management#:~:text=Memory%20management%20is%20the%20process,to%20carry%20out%20their%20operations..> [Accessed 2 April 2023].
- [6] "Device Management in Operating System," Java T Point, [Online]. Available: <https://www.javatpoint.com/device-management-in-operating-system>. [Accessed 2 April 2023].
- [7] D. Scripto, "Table of Basic PowerShell Commands," Microsoft, 11 June 2015. [Online]. Available: <https://devblogs.microsoft.com/scripting/table-of-basic-powershell-commands/>. [Accessed 3 April 2023].
- [8] "Persist files in Azure Cloud Shell," Microsoft, 30 March 2023. [Online]. Available: <https://learn.microsoft.com/en-us/azure/cloud-shell/persisting-shell-storage>. [Accessed 3 April 2023].
- [9] "Get-Process," PDQ, [Online]. Available: <https://www.pdq.com/powershell/get-process/>. [Accessed 3 April 2023].

- [10] "Format-List," Microsoft, [Online]. Available: <https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/format-list?view=powershell-7.3>. [Accessed 3 April 2023].
- [11] "Get-ChildItem," SS64, [Online]. Available: <https://ss64.com/ps/get-childitem.html>. [Accessed 3 April 2023].
- [12] C. Wilson, "Bash cheat sheet: Top 25 commands and creating custom commands," Educative, 27 August 2019. [Online]. Available: <https://www.educative.io/blog/bash-shell-command-cheat-sheet>. [Accessed 4 April 2023].
- [13] G. Myrianthous, "16 Bash Commands Data Scientists Must Know," Built In, 18 October 2022. [Online]. Available: <https://builtin.com/data-science/bash-commands>. [Accessed 4 April 2023].
- [14] "An A-Z Index of the Linux command line: bash + utilities.," ss64, [Online]. Available: <https://ss64.com/bash/>. [Accessed 4 April 2023].
- [15] Microsoft, [Online]. Available: <https://learn.microsoft.com/en-us/powershell/scripting/samples/sample-scripts-for-administration?view=powershell-7.3>. [Accessed 6 April 2023].
- [16] R. Mens, "PowerShell Scripting – Get started with this Ultimate Guide," Lazy Admin, 9 January 2023. [Online]. Available: <https://lazyadmin.nl/powershell/powershell-script/>. [Accessed 6 April 2023].
- [17] L. Reynolds, "Bash Scripting Tutorial," Linux Config, 4 September 2022. [Online]. Available: <https://linuxconfig.org/bash-scripting-tutorial>. [Accessed 6 April 2023].
- [18] C. Perlton, "The Perfect PowerShell Script," Inedo Blog, 29 March 2021. [Online]. Available: <https://blog.inedo.com/powershell/perfect-script/>. [Accessed 6 April 2023].
- [19] "Bash scripting cheatsheet," Dev Hints, [Online]. Available: <https://devhints.io/bash>. [Accessed 6 April 2023].
- [20] K. Pykes, "How to Write a Bash Script: A Simple Bash Scripting Tutorial," Data Camp, November 2022. [Online]. Available: <https://www.datacamp.com/tutorial/how-to-write-bash-script-tutorial>. [Accessed 6 April 2023].

Lloyd Case Computer Architecture and Operating Systems Coursework

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

77 /100

Instructor

Total:		77
1. Engagement with Literature Skills:		
+ Very good use of citations. + formatted reference list. - A book/academic paper could also be included in the reference list, and multiple citations used in places, to gain the extra marks in this category.	20	18
2. Knowledge and Understanding Skills: 4e		
Key functions: + Very good overview of key OS functions documented. Bat Commands: + V. good overview of relevant commands that demonstrates an understanding of the language. Bash Commands: + V. good overview of relevant commands that demonstrates an understanding of the language. Bat Scripts: + Good overview of the syntax that demonstrates an understanding of the scripting aspect of the language. – A couple of additional scripts could be used. Bash Scripts: + Good overview of the syntax that demonstrates an understanding of the scripting aspect of the language. – Several additional scripts could be used.	20	16

3. Cognitive and Intellectual Skills: 2e		
Evidence of some good analysis and understanding of OS functions and the relevant languages is clearly documented. – could be extended in some script areas as mentioned above, to gain the extra marks in this category. + Conclusion.	1 0	7
4. Practical Application Skills: 10e		
1. Bat Commands: + V. good use of relevant Bat commands. 2. Bash Commands: + V. good use of relevant Bash commands. 3. Bat Scripts: + Good use of Bat scripts to demonstrate knowledge of the core syntax. - could be extended in 1/2 areas to demonstrate some additional knowledge of the core syntax. 4. Bash Scripts: + Good use of Bash scripts to demonstrate knowledge of the core syntax. - could be extended in several areas to demonstrate some additional knowledge of the core syntax.	4 0	32
5. Transferable Skills for Life and Professional Practice: 5+1+2+2		
+ Good report format, structure, and presentation. - Report sections could be numbered. - Try to use third person tense and avoid using 'I' when writing a report. - Tutorial Activity 1 – OS Types: relevant supplementary information on OS types is not included in the Appendix of the report.	1 0	4

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

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