

kelly casey OS assignment

by Kelly Casey

Submission date: 07-Apr-2023 03:49PM (UTC+0100)

Submission ID: 201699093

File name: 31343_Kelly_Casey_kelly_casey_OS_assignment_480410_1268976506.docx (3.86M)

Word count: 1194

Character count: 6337



OPERATING SYSTEMS

ACCA4022



APRIL 7, 2023

KELLY CASEY
2106451

Contents

Introduction	1
Key functions of general operating systems.....	2
Process management.....	2
Memory management	2
File management	2
Device management	2
Use of Windows OS CLI commands	3
Use of Linux OS CLI commands.....	8
Use of Windows OS scripts.....	12
Use of Linux OS scripts.....	16
Conclusion.....	20
Bibliography	20
Appendixes.....	21
Linux unhatched	21
Types of operating systems	21

Introduction

Operating systems are an important component of modern computing, serving as a crucial intermediary between hardware and software. As a first-year university student studying computer science, I am eager to explore the workings of operating systems, analysing their architecture, functions, and core concepts. Through this assignment, I will be looking at two types of operating systems, Windows, and Linux. By the end of this assignment, I hope to have gained a more comprehensive understanding of operating systems and their impact on computing, providing me with a solid foundation for future studies in computer science.

Key functions of general operating systems.

Modern general Operating Systems (OS) perform a wide range of functions, providing a stable and consistent platform for application software to run on. The key functions of an Operating System include process management, memory management, file management, and device management.

Process management

Process management is an important function of operating systems, its job is to schedule, and processes on the CPUs, creates user and system processes, stopping, and restarting processes, providing mechanisms for process synchronization and communication. [1]

Memory management

As the main use for a computer system is to implement programs it makes memory management a crucial component of modern operating systems, and it plays a vital role in ensuring system stability and performance. Memory management is a system which stores instructions and processes data in bytes. The CPU gets the information from the memory in accordance with the value from the program counter. [2]

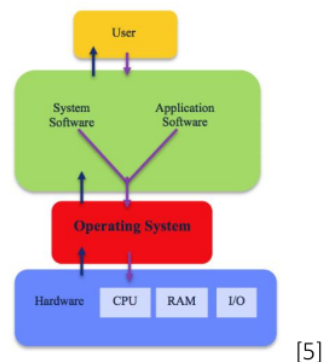
File management

File management is another important function of operating systems it deals with the organization and storage of files and data such as text, binary audio, video and so on. The file management shows us everything about each file, like when it was created, the type of file, location of file, owner, and time the file was created. Effective file management is crucial for ensuring data integrity, confidentiality, and availability. [3]

Device management

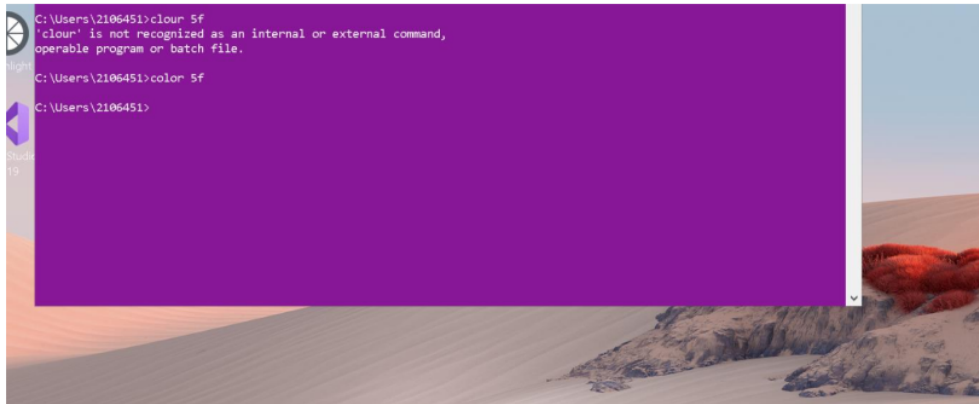
Device management is a key function of operating systems that deals with managing input/output (I/O) devices, such as keyboards, mice, printers, and storage devices. The operating system provides device drivers to interact with hardware devices and manage their resources. [4]

In conclusion, Operating Systems are essential software that provides a range of important functions for computing. Process management, memory management, file management, and device management are the fundamental functions of an Operating System.



[5]

Use of Windows OS CLI commands



```
C:\Users\2106451>color 5f
'color' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\2106451>

C:\Users\2106451>
```

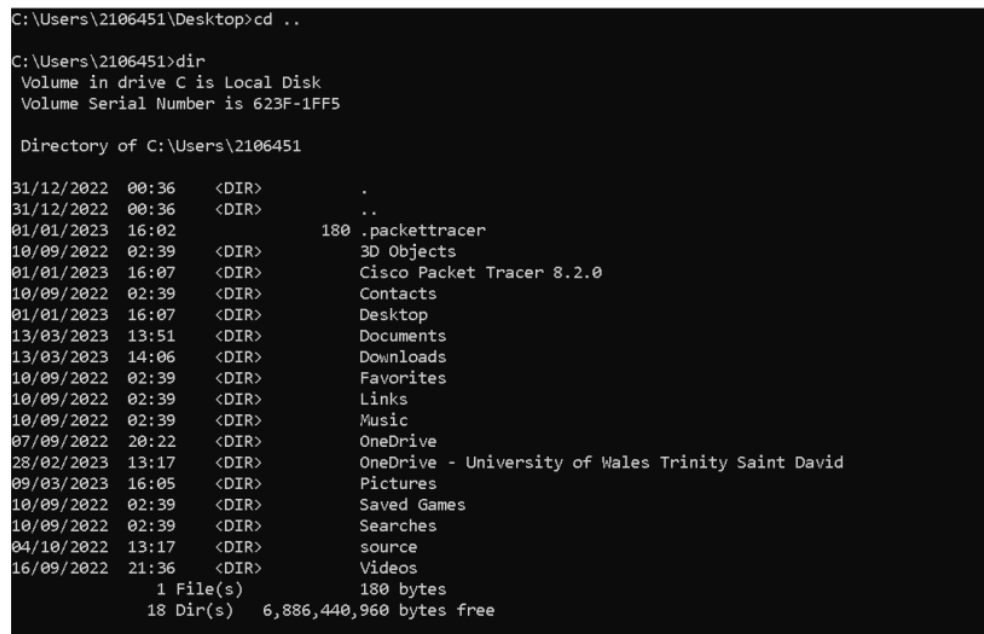
Using the word **color** allows you to change the colour of the CMD, here you can see I have changed the colour of the CMD from the original black to purple.



```
C:\Users\2106451>whoami
uni\2106451

C:\Users\2106451>
```

Using **whoami** displays the user's username. Here you can see its just my university I.D number.



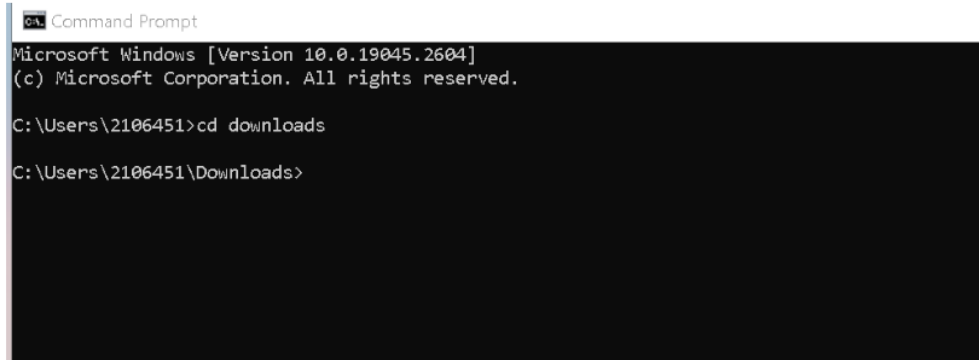
```
C:\Users\2106451\Desktop>cd ..

C:\Users\2106451>dir
Volume in drive C is Local Disk
Volume Serial Number is 623F-1FF5

Directory of C:\Users\2106451

31/12/2022  00:36    <DIR>          .
31/12/2022  00:36    <DIR>          ..
01/01/2023  16:02             180 .packettracer
10/09/2022  02:39    <DIR>          3D Objects
01/01/2023  16:07    <DIR>          Cisco Packet Tracer 8.2.0
10/09/2022  02:39    <DIR>          Contacts
01/01/2023  16:07    <DIR>          Desktop
13/03/2023  13:51    <DIR>          Documents
13/03/2023  14:06    <DIR>          Downloads
10/09/2022  02:39    <DIR>          Favorites
10/09/2022  02:39    <DIR>          Links
10/09/2022  02:39    <DIR>          Music
07/09/2022  20:22    <DIR>          OneDrive
28/02/2023  13:17    <DIR>          OneDrive - University of Wales Trinity Saint David
09/03/2023  16:05    <DIR>          Pictures
10/09/2022  02:39    <DIR>          Saved Games
10/09/2022  02:39    <DIR>          Searches
04/10/2022  13:17    <DIR>          source
16/09/2022  21:36    <DIR>          Videos
               1 File(s)              180 bytes
               18 Dir(s)  6,886,440,960 bytes free
```

Using **dir** will display a list of contents in the current directory.

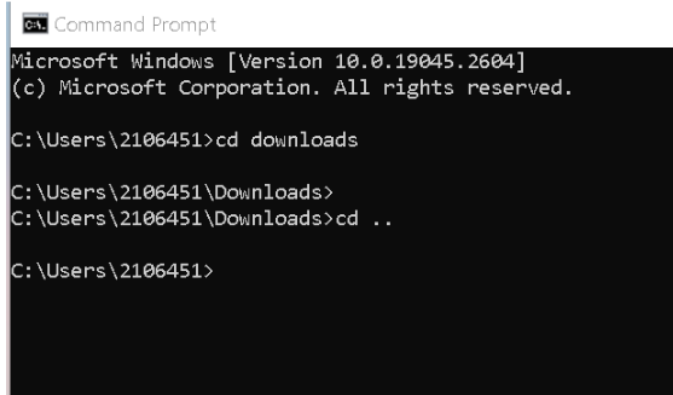


```
CA: Command Prompt
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\2106451>cd downloads

C:\Users\2106451\Downloads>
```

Using **cd** followed by the name of a folder will take you the specific directory.



```
CA: Command Prompt
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\2106451>cd downloads

C:\Users\2106451\Downloads>
C:\Users\2106451\Downloads>cd ..

C:\Users\2106451>
```

Using **cd ...** will take you back a directory.

```

Command Prompt
REPLACE      Replaces files.
RMDIR        Removes a directory.
ROBOCOPY      Advanced utility to copy files and directory trees
SET           Displays, sets, or removes Windows environment variables.
SETLOCAL      Begins localization of environment changes in a batch file.
SC            Displays or configures services (background processes).
SCHTASKS      Schedules commands and programs to run on a computer.
SHIFT         Shifts the position of replaceable parameters in batch files.
SHUTDOWN      Allows proper local or remote shutdown of machine.
SORT          Sorts input.
START         Starts a separate window to run a specified program or command.
SUBST         Associates a path with a drive letter.
SYSTEMINFO    Displays machine specific properties and configuration.
TASKLIST      Displays all currently running tasks including services.
TASKKILL      Kill or stop a running process or application.
TIME          Displays or sets the system time.
TITLE         Sets the window title for a C/D.EXE session.
TREE          Graphically displays the directory structure of a drive or
              path.
TYPE          Displays the contents of a text file.
VER           Displays the Windows version.
VERIFY        Tells Windows whether to verify that your files are written
              correctly to a disk.
VOL           Displays a disk volume label and serial number.
XCOPY         Copies files and directory trees.
WMIC          Displays WMI information inside interactive command shell.

For more information on tools see the command-line reference in the online help.

C:\Users\2106451>

```

Using **help** will create a list of commands.

```

C:\Users\2106451>netstat

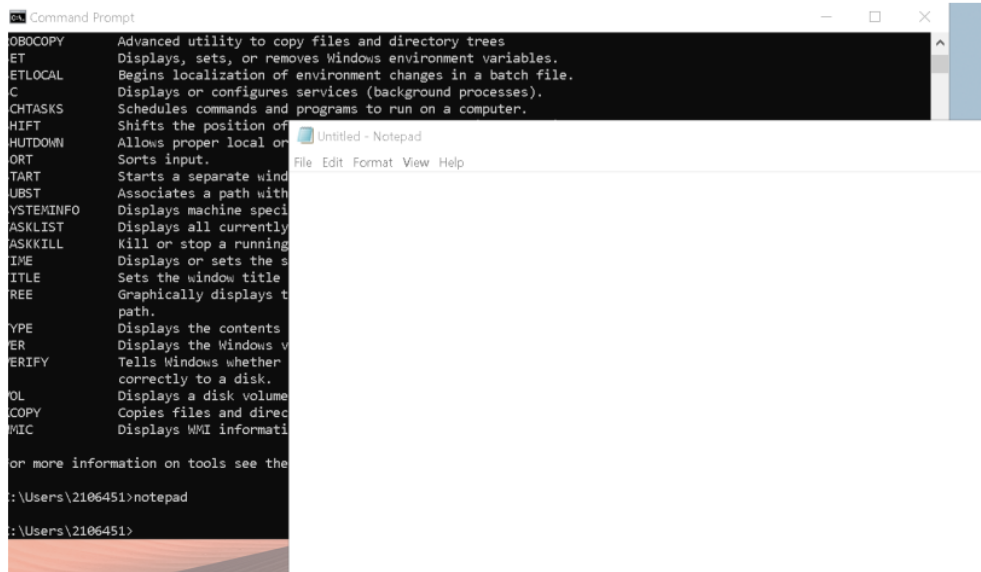
Active Connections

Proto Local Address           Foreign Address         State
TCP   127.0.0.1:49701          UOW-04395703851:49702  ESTABLISHED
TCP   127.0.0.1:49702          UOW-04395703851:49701  ESTABLISHED
TCP   127.0.0.1:49703          UOW-04395703851:49704  ESTABLISHED
TCP   127.0.0.1:49704          UOW-04395703851:49703  ESTABLISHED
TCP   192.168.0.162:49408      20.90.153.243:https    ESTABLISHED
TCP   192.168.0.162:49743      LIVINGROOM/TV:8009     ESTABLISHED
TCP   192.168.0.162:49744      192.168.0.107:8009     ESTABLISHED
TCP   192.168.0.162:49757      124:4070               ESTABLISHED
TCP   192.168.0.162:49762      LIVINGROOM/TV:8009     ESTABLISHED
TCP   192.168.0.162:49763      192.168.0.107:8009     ESTABLISHED
TCP   192.168.0.162:49792      47:https                ESTABLISHED
TCP   192.168.0.162:49803      40:https                ESTABLISHED
TCP   192.168.0.162:49808      192.168.0.15:8009      ESTABLISHED
TCP   192.168.0.162:49817      192.168.0.211:8009     ESTABLISHED
TCP   192.168.0.162:49859      we-in-f188:5228        ESTABLISHED
TCP   192.168.0.162:49887      52.111.236.14:https    ESTABLISHED
TCP   192.168.0.162:49889      52.111.236.14:https    ESTABLISHED
TCP   192.168.0.162:49934      192.168.0.15:8009      ESTABLISHED
TCP   192.168.0.162:50049      13.107.138.8:https     ESTABLISHED
TCP   192.168.0.162:50071      52.97.133.146:https    ESTABLISHED
TCP   192.168.0.162:50072      52.97.133.146:https    ESTABLISHED
TCP   192.168.0.162:50073      52.97.133.146:https    ESTABLISHED
TCP   192.168.0.162:50121      13.69.239.74:https     TIME_WAIT
TCP   192.168.0.162:50123      a-0001:https            ESTABLISHED
TCP   192.168.0.162:50125      bingforbusiness:https  ESTABLISHED
TCP   192.168.0.162:50126      20.189.173.11:https    ESTABLISHED
TCP   192.168.0.162:50127      whatsapp-cdn-shv-01-man2:https ESTABLISHED
TCP   192.168.0.162:57621      192.168.0.166:56488    ESTABLISHED

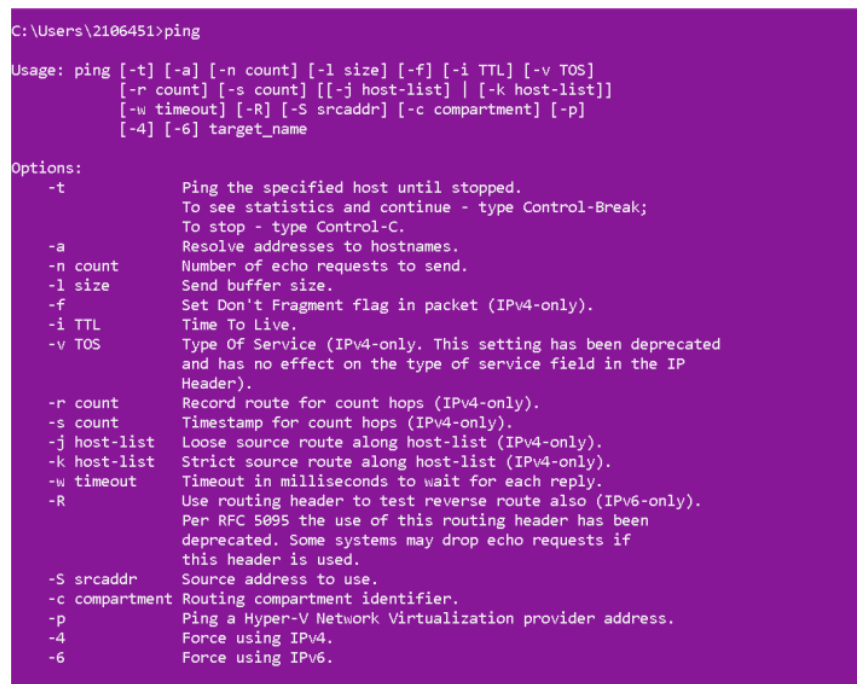
C:\Users\2106451>

```

Using **netstat** displays all network information.



Using **notepad** will run windows notepad.



Using **ping** tests the connectivity of the network.

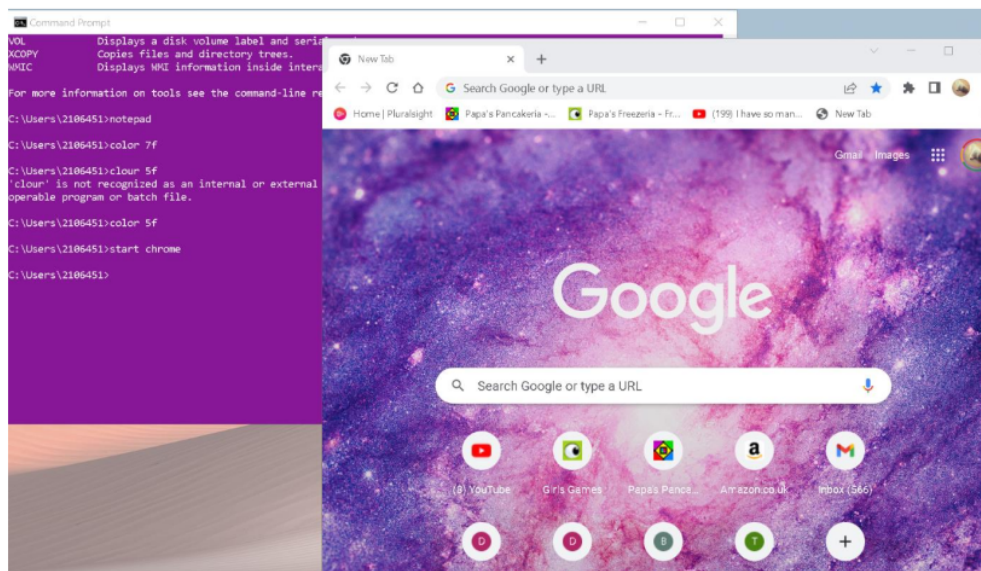
```
C:\Users\2106451>rmdir test

C:\Users\2106451>dir
Volume in drive C is Local Disk
Volume Serial Number is 623F-1FF5

Directory of C:\Users\2106451

13/03/2023  14:38  <DIR>          .
13/03/2023  14:38  <DIR>          ..
01/01/2023  16:02             180 .packettracer
10/09/2022  02:39  <DIR>          3D Objects
01/01/2023  16:07  <DIR>          Cisco Packet Tracer 8.2.0
10/09/2022  02:39  <DIR>          Contacts
01/01/2023  16:07  <DIR>          Desktop
13/03/2023  13:51  <DIR>          Documents
13/03/2023  14:06  <DIR>          Downloads
10/09/2022  02:39  <DIR>          Favorites
10/09/2022  02:39  <DIR>          Links
10/09/2022  02:39  <DIR>          Music
07/09/2022  20:22  <DIR>          OneDrive
28/02/2023  13:17  <DIR>          OneDrive - University of Wales Trinity Saint David
09/03/2023  16:05  <DIR>          Pictures
10/09/2022  02:39  <DIR>          Saved Games
10/09/2022  02:39  <DIR>          Searches
04/10/2022  13:17  <DIR>          source
16/09/2022  21:36  <DIR>          Videos
               1 File(s)          180 bytes
               18 Dir(s)  6,883,069,952 bytes free
```

Using **redir** removes a folder, as you can see, I removed one I had created one called test.



Using **start** followed by the application you want to start will start the application for you.

Use of Linux OS CLI commands.

```
localhost:~$ arp -a
? (10.5.0.1) at 6a:df:c0:58:23:cc [ether] on eth0
localhost:~$
```

Using **arp -a** will show you all active devices on your network.

```
localhost:~$ traceroute
BusyBox v1.31.1 () multi-call binary.

Usage: traceroute [-46FIlnrv] [-f 1ST_TTL] [-m MAXTTL] [-q PROBES] [-p PORT]
        [-t TOS] [-w WAIT_SEC] [-s SRC_IP] [-i IFACE]
        [-z PAUSE_MSEC] HOST [BYTES]

Trace the route to HOST

        -4,-6  Force IP or IPv6 name resolution
        -F      Set don't fragment bit
        -I      Use ICMP ECHO instead of UDP datagrams
        -l      Display TTL value of the returned packet
        -n      Print numeric addresses
        -r      Bypass routing tables, send directly to HOST
        -v      Verbose
        -f N    First number of hops (default 1)
        -m N    Max number of hops
        -q N    Number of probes per hop (default 3)
        -p N    Base UDP port number used in probes
                  (default 33434)
        -s IP   Source address
        -i IFACE Source interface
        -t N    Type-of-service in probe packets (default 0)
        -w SEC  Time to wait for a response (default 3)
        -g IP   Loose source route gateway (8 max)

localhost:~$
```

Traceroute allows you to trace the path packets go through.

```
localhost:~$ whois
BusyBox v1.31.1 () multi-call binary.

Usage: whois [-i] [-h SERVER] [-p PORT] NAME...

Query WHOIS info about NAME

        -i      Show redirect results too
        -h,-p   Server to query

localhost:~$
```

Whois will show domain information.

```
localhost:~$
```

Using **clear** will clear anything you have previously done.

```
localhost:~$ history
 0 pwd
 1 is
 2 cd home
 3 mkdir testdir
 4 is
 5 clear
 6 pwd
 7 cd/
 8 pingh
 9 ping
10 netstat
11 whois
12 who is <domain>
13 who is
14 whois
15 arp -1
16 arp -a
17 traceroute
18 history
localhost:~$
```

Using **history** will show you all commands you have previously entered.

```
localhost:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
netstat: /proc/net/tcp6: No such file or directory
netstat: /proc/net/udp6: No such file or directory
netstat: /proc/net/raw6: No such file or directory
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags     Type       State      I-Node Path
unix  2      [ ]       DGRAM      0          2961
localhost:~$
```

Netstat shows statistics on network connections.

```
localhost:~$ ping
BusyBox v1.31.1 () multi-call binary.

Usage: ping [OPTIONS] HOST

Send ICMP ECHO_REQUEST packets to network hosts

    -4,-6           Force IP or IPv6 name resolution
    -c CNT          Send only CNT pings
    -s SIZE         Send SIZE data bytes in packets (default 56)
    -i SECS         Interval
    -A             Ping as soon as reply is received
    -t TTL          Set TTL
    -I IFACE/IP     Source interface or IP address
    -W SEC          Seconds to wait for the first response (default 10)
                    (after all -c CNT packets are sent)
    -w SEC          Seconds until ping exits (default:infinite)
                    (can exit earlier with -c CNT)
    -q             Quiet, only display output at start
                    and when finished
    -p HEXBYTE      Pattern to use for payload
localhost:~$
```

Ping tests your network connection.

```
localhost:~$ Generating encryption key.
pwd
/home/kellyc89
localhost:~$
```

Pwd, prints working directory.

```
localhost:~$ tree
.
├── file
└── testdir

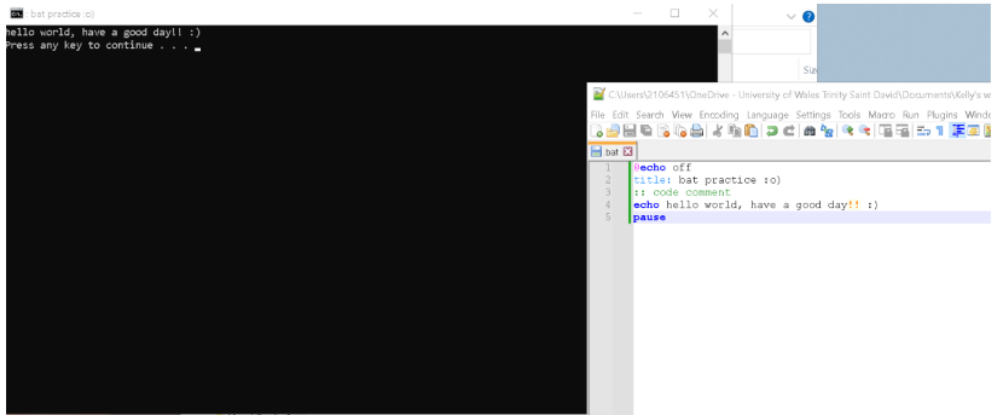
1 directory, 1 file
localhost:~$
```

Tree lists a directory tree.

```
20 history -c
localhost:~$ history
 0 pwd
 1 is
 2 cd home
 3 mkdir testdir
 4 is
 5 clear
 6 pwd
 7 cd/
 8 pingh
 9 ping
10 netstat
11 whois
12 who is <domain>
13 who is
14 whois
15 arp -1
16 arp -a
17 traceroute
18 history
19 man
20 history -c
21 history
localhost:~$
```

History – c will clear all your history.

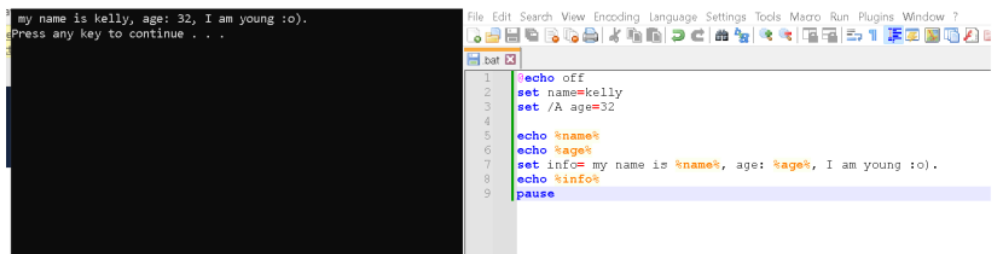
Use of Windows OS scripts.



The screenshot shows a Windows command prompt window titled "bat practice .bat" with the output: "hello world, have a good day!! :)" and "Press any key to continue . . .". To the right, a Notepad++ editor shows the following batch script code:

```
1 @echo off
2 title: bat practice :o)
3 :: code comment
4 echo hello world, have a good day!! :)
5 pause
```

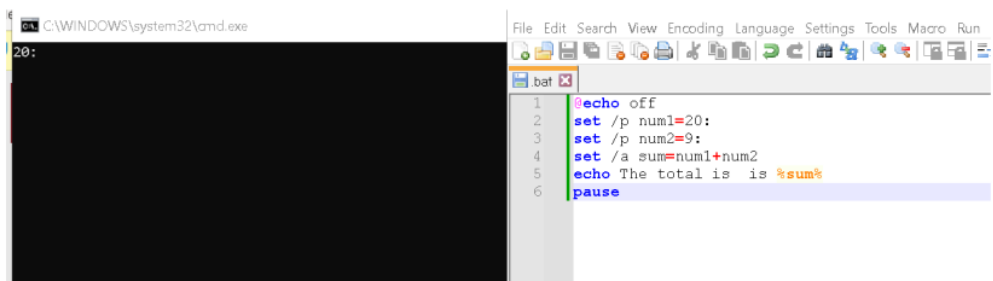
Using **echo** prints out a statement.



The screenshot shows a Windows command prompt window titled "bat practice .bat" with the output: "my name is kelly, age: 32, I am young :o)." and "Press any key to continue . . .". To the right, a Notepad++ editor shows the following batch script code:

```
1 @echo off
2 set name=kelly
3 set /A age=32
4
5 echo %name%
6 echo %age%
7 set info= my name is %name%, age: %age%, I am young :o).
8 echo %info%
9 pause
```

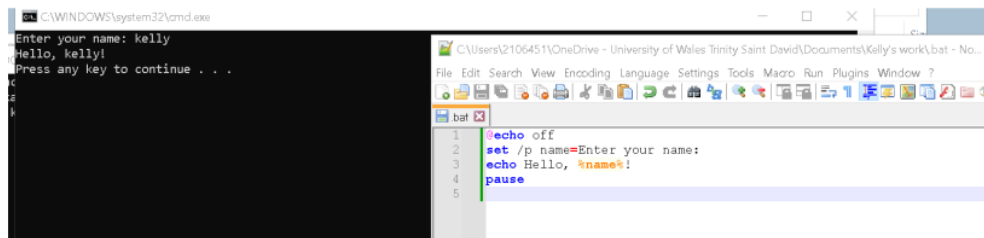
Here I have set variables of my name and age and printed it out.



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe" with the prompt "20:". To the right, a Notepad++ editor shows the following batch script code:

```
1 @echo off
2 set /p num1=20:
3 set /p num2=9:
4 set /a sum=num1+num2
5 echo The total is is %sum%
6 pause
```

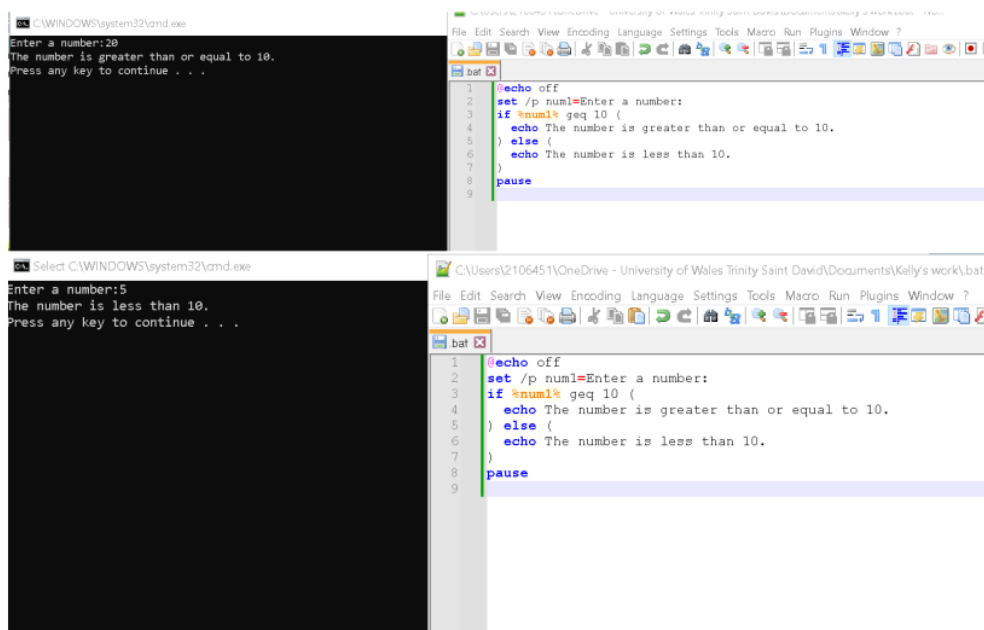
Here I have set number variables and created a sum.



The screenshot shows a Windows command prompt window on the left and a Notepad++ editor on the right. The command prompt displays the output of a batch script: "Enter your name: kelly", "Hello, kelly!", and "Press any key to continue . . .". The Notepad++ editor shows the source code of the batch script, which is a simple program that prompts the user for their name and prints a greeting.

```
1 echo off
2 set /p name=Enter your name:
3 echo Hello, %name%!
4 pause
5
```

Here I have created a simple program for the user to input their name.



This block contains two screenshots. The top screenshot shows the command prompt with the input "20" and the output "The number is greater than or equal to 10." The bottom screenshot shows the command prompt with the input "5" and the output "The number is less than 10." Both screenshots show the same batch script in the Notepad++ editor, which uses an if/else statement to check the input number against 10.

```
1 echo off
2 set /p num1=Enter a number:
3 if %num1% geq 10 {
4   echo The number is greater than or equal to 10.
5 } else {
6   echo The number is less than 10.
7 }
8 pause
9
```

Here I have created an if/else statement. Which prints out an answer depending on the input of the question.

```
C:\WINDOWS\system32\cmd.exe
What is your name? kelly
Nice to meet you, kelly.
How can I help you?
Press any key to continue . . .

C:\Users\2106451\OneDrive - University of Wales Trinity Saint David\Documents\Kelly's work\
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
bat
1 echo off
2 set /p name=What is your name?
3 if "%name%"=="Olivia" (
4     echo Hello, Olivia!
5     goto end
6 ) else (
7     echo Nice to meet you, %name%.
8 )
9 echo How can I help you?
10 :end
11 pause
12
```

```
C:\WINDOWS\system32\cmd.exe
What is your name?Olivia
Hello, Olivia!
Press any key to continue . . .

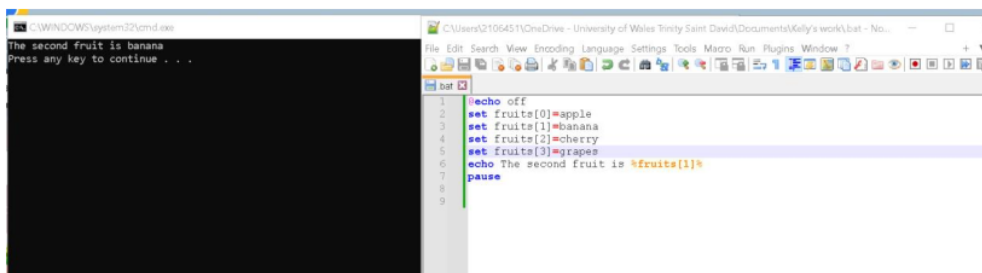
C:\Users\2106451\OneDrive - University of Wales Trinity Saint David\Documents\Kelly's work\bat - No...
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
bat
1 echo off
2 set /p name=What is your name?
3 if "%name%"=="Olivia" (
4     echo Hello, Olivia!
5     goto end
6 ) else (
7     echo Nice to meet you, %name%.
8 )
9 echo How can I help you?
10 :end
11 pause
12
```

Here I have carried on with the if else statement and added a goto, which depending on the input of the user will take you to a different part of the code.

```
C:\WINDOWS\system32\cmd.exe
Counting from 1 to 5:
1
2
3
4
5
List of fruits:
apple
banana
cherry
Press any key to continue . . .

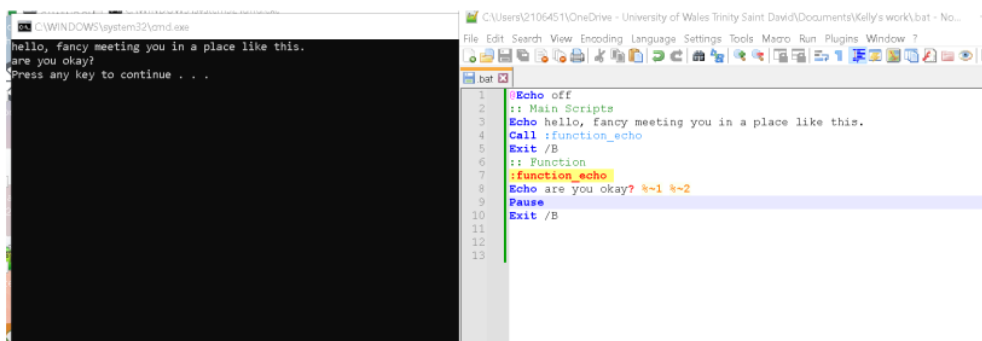
C:\Users\2106451\OneDrive - University of Wales Trinity Saint David\Documents\Kelly's work\bat - No...
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
bat
1 echo off
2 echo Counting from 1 to 5:
3 for /L %i in (1,1,5) do (
4     echo %i
5 )
6 echo List of fruits:
7 for %f in (apple banana cherry) do (
8     echo %f
9 )
10 pause
11
```

Here I have used a forloop which loops things making the code much easier so you don't have to keep writing things many times



```
1 | echo off
2 | set fruits[0]=apple
3 | set fruits[1]=banana
4 | set fruits[2]=cherry
5 | set fruits[3]=grapes
6 | echo The second fruit is %fruits[1]%
7 | pause
8 |
9 |
```

Here I have used an array. Arrays are like little boxes of memory that hold a single piece of data. Here I have an array of fruit, asking which. the second array will produce that single piece of data.



```
1 | Echo off
2 | :: Main Script
3 | Echo hello, fancy meeting you in a place like this.
4 | Call :function_echo
5 | Exit /B
6 | :: Function
7 | :function_echo
8 | Echo are you okay? %~1 %~2
9 | Pause
10 |
11 |
12 |
13 |
```

Here I have created a function. Functions are pieces of code that can be used over and instead of having to retype it every time.

Use of Linux OS scripts

```
1 echo "Hello everyone, are you having a good day ?"
> bash main.sh
Hello everyone, are you having a good day ?
>
```

Here I have used echo which simply prints out your message.

```
1 #!/bin/bash
2
3 echo "What is your name?"
4 read name
5
6 echo "Hello, $name! How are you today?"
7
8 echo "Enter the first number:"
9 read num1
10 echo "Enter the second number:"
11 read num2
12 sum=$((num1 + num2))
13 echo "The sum of $num1 and $num2 is $sum."
14
> bash main.sh
What is your name?
kelly
Hello, kelly! How are you today?
Enter the first number:
305
Enter the second number:
6
The sum of 305 and 6 is 311.
>
```

Here I have used variables, sum and user input to create this simple program.

```
1 #!/bin/bash
2
3 num1=7
4 num2=8
5
6 let sum=num1+num2
7 let diff=num1-num2
8 let product=num1*num2
9 let quotient=num1/num2
10
11 echo "The sum of $num1 and $num2 is $sum."
12 echo "The difference between $num1 and $num2 is $diff."
13 echo "The product of $num1 and $num2 is $product."
14 echo "The quotient of $num1 and $num2 is $quotient."
> bash main.sh
The sum of 7 and 8 is 15.
The difference between 7 and 8 is -1.
The product of 7 and 8 is 56.
The quotient of 7 and 8 is 0.
>
```

Here I have used let, which allows the user to do math.

```
1 #!/bin/bash
2
3 echo "What is your name?"
4 read name
5
6 echo "Hello, $name! How are you today?"
7
8 echo "How old are you?"
9 read age
10
11 echo "what is your favourite TV program?"
12 read program
13
14 echo "nice, i like that too!"
```

```
> bash main.sh
What is your name?
kelly
Hello, kelly! How are you today?
How old are you?
33
what is your favourite TV program?
greys anatomy
nice, i like that too!
>
```

Now ive created a simple program to allow the user to put in their put in their own input

```
1 #!/bin/bash
2
3 favorite="Cadburys"
4
5 read -p "Guess my favorite chocolate bar: " guess
6
7 if [[ "$guess" == "$favorite" ]]; then
8     echo "You guessed correctly!"
9 else
10     echo "Sorry, that's not my favorite."
11 fi
12
```

```
> bash main.sh
Guess my favorite chocolate bar: bounty
Sorry, that's not my favorite.
>
```

```
1 #!/bin/bash
2
3 favorite="Cadburys"
4
5 read -p "Guess my favorite chocolate bar: " guess
6
7 if [[ "$guess" == "$favorite" ]]; then
8     echo "You guessed correctly!"
9 else
10     echo "Sorry, that's not my favorite."
11 fi
12
```

```
> bash main.sh
Guess my favorite chocolate bar: Cadburys
You guessed correctly!
>
```

Here I have used if else statements to get the user to guess the answer to my question. It will then print out a reply depending on if they are correct or not.

```

1 #!/bin/bash
2
3 echo "What is your name?"
4 read name
5
6 echo "Hello, $name! How are you today?"
7
8 echo "Enter the first number:"
9 read num1
10 echo "Enter the second number:"
11 read num2
12 sum=$((num1 + num2))
13 echo "The sum of $num1 and $num2 is $sum."
14

```

```

> bash main.sh
What is your name?
kelly
Hello, kelly! How are you today?
Enter the first number:
305
Enter the second number:
6
The sum of 305 and 6 is 311.
> 

```

```

1 #!/bin/bash
2
3 echo "What is your favorite color?"
4 read color
5
6 case $color in
7     red)
8         echo "Red is a nice color."
9         ;;
10    blue)
11        echo "Blue is a calming color."
12        ;;
13    green)
14        echo "Green is a natural color."
15        ;;
16    *)
17        echo "I'm sorry, I don't know that color."
18    ;;
19

```

```

> bash main.sh
What is your favorite color?
orange
I'm sorry, I don't know that color.
> 

```

Using case statements creates a value and test it against several conditions. I've used it here to create a simple program about favourite colours.

```

1 #!/bin/bash
2
3 count=1
4 while [ $count -le 20 ]
5 do
6     echo "Count: $count"
7     count=$((count+1))
8 done
9
10 fruits=("apple" "banana" "orange" "kiwi" "grapes"
11         "strawberry")
12 for fruit in "${fruits[@]}"
13 do
14     echo "Fruit: $fruit"
15 done
16

```

```

> bash main.sh
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
Count: 6
Count: 7
Count: 8
Count: 9
Count: 10
Count: 11
Count: 12
Count: 13
Count: 14
Count: 15
Count: 16
Count: 17
Count: 18
Count: 19
Count: 20
Fruit: apple
Fruit: banana
Fruit: orange
Fruit: kiwi
Fruit: grapes
Fruit: strawberry
> 

```

Here I have used while loop to count numbers up to 20, using this stops the user having to type every single number.

I then used a for loop in an array. An array stores a single piece of data in boxes.

```

1  #!/bin/bash
2
3  show_car_info() {
4      local make=$1
5      local model=$2
6      local year=$3
7      local color=$4
8      echo "Make: $make"
9      echo "Model: $model"
10     echo "Year: $year"
11     echo "Color: $color"
12 }
13
14
15 calculate_car_age() {
16     local current_year=$(date +%Y)
17     local car_year=$1
18     local age=$((current_year - car_year))
19     echo $age
20 }
21 echo "Enter car information:"
22 echo -n "Make: "
23 read make
24 echo -n "Model: "
25 read model

```

```

15 calculate_car_age() {
16     local current_year=$(date +%Y)
17     local car_year=$1
18     local age=$((current_year - car_year))
19     echo $age
20 }
21 echo "Enter car information:"
22 echo -n "Make: "
23 read make
24 echo -n "Model: "
25 read model
26 echo -n "Year: "
27 read year
28 echo -n "Color: "
29 read color
30
31 show_car_info $make $model $year $color
32
33 age=$(calculate_car_age $year)
34
35
36 echo "The car is $age years old."
37

```

```

> bash main.sh
Enter car information:
Make: volkswagen
Model: beetle
Year: 1970
Color: yellow
Make: volkswagen
Model: beetle
Year: 1970
Color: yellow
The car is 53 years old.
>

```

Here I have created a program to calculate the age of a car using functions.

Conclusion

In conclusion, operating systems are essential software that manage and the hardware and software components of a computer. Scripting is a powerful tool that can be used to automate tasks, increase productivity, and customize the operating system experience. Bash scripting is commonly used on Linux, while Bat scripting is commonly used on Windows. Personally, I found bash scripting preferable. Scripting allows users to write code to automate tasks, perform system maintenance, and manage files and directories.

Bibliography

- [1] "java point," [Online]. Available: <https://www.javatpoint.com/process-management-in-os>. [Accessed 5 4 2023].
- [2] "geeksforgeeks," 9 1 2023. [Online]. Available: <https://www.geeksforgeeks.org/memory-management-in-operating-system/>. [Accessed 5 4 2023].
- [3] "scaler," [Online]. Available: <https://www.scaler.com/topics/file-management-in-operating-system/>. [Accessed 5 4 2023].
- [4] "bcasstudy," [Online]. Available: <https://bcasstudyguide.com/unit-4-device-management/>. [Accessed 5 4 2023].
- [5] "univeristy of wollongong australia," [Online]. Available: <https://www.uow.edu.au/student/learning-co-op/technology-and-software/operating-systems/>. [Accessed 5 4 2023].

Appendixes

Linux unhatched

NDG Linux Unhatched English 0323b

⬆️⬆️⬇️

Welcome to NDG Linux Unhatched!

Your progress ?

NDG Linux Unhatched

⬆️

N

Course content

N

Exam

✓

✓

End of Course Feedback

⬆️

N

End of course feedback

✓

User report - Kelly casey

Overview report User report

Grade item	Calculated weight	Grade	Range	Percentage	Feedback	Contribution to course total
NDG Linux Unhatched English 0323b						
N Help & Resources	-		-			-
N Course content	-		-			-
N Exam	10.00 %	8.33	0-10	83.33 %		8.33 %
N End of course feedback	90.00 %	90.00	0-90	100.00 %		90.00 %
N Course total	-	98.33	0-100	98.33 %		-

Types of operating systems

[types of OS.docx](#)

FINAL GRADE

GENERAL COMMENTS

Instructor

84/100

Total:		84
1. Engagement with Literature Skills:		
+ Some good use of citations. + formatted reference list. - Some additional sources and citations, plus a book/academic paper could also be included in the reference list, to gain the extra marks in this category.	20	10
2. Knowledge and Understanding Skills: 4e		
Key functions: + Very good overview of key OS functions documented. - could be extended to include security. 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. Bash Scripts: + Good overview of the syntax that demonstrates an understanding of the scripting aspect of the language. - Citations and corresponding	20	18

reference sources could be used to support the CLI task answers.		
3. Cognitive and Intellectual Skills: 2e		
Evidence of some good analysis and understanding of OS functions and the relevant languages is clearly documented. + Conclusion. - Citations and corresponding reference sources could be used to support the code analysis.	10	9
4. Practical Application Skills: 10e		
1. Bat Commands: + V. good use of relevant Bat commands. - could be extended to include some additional 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. 4. Bash Scripts: + Good use of Bash scripts to demonstrate knowledge of the core syntax.	40	40
5. Transferable Skills for Life and Professional Practice: 5+1+2+2		
+ Good report format, structure, and presentation. - Report sections could be numbered. + Tutorial Activity 1 – OS Types: relevant supplementary information on OS types is included in the Appendix of the report, - as a word document that is not accessible.	10	7

+ Evidence of
engagement with the Linux
Unhatched course 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

PAGE 18

PAGE 19

PAGE 20

PAGE 21

