H. Dip in Software Development

Computer Architecture & Technology Convergence

Final Project 2020

Aron Nutley
G00388055

Contents

Table of Figures

List of Tables

Computer Architecture and Technology Convergence Assignment

Q1: Binary Arithmetic

Feel free to use any resources you need for the tasks below, but make sure to show workings

Q1.1.

Add 11011 to 1011. Show your work (in particular, show where you get carries, and where you don't). You can check your work by translating the numbers into decimal, but I want to see the addition algorithm in base 2 instead of base ten. Hint: You can use MS Word tables to show calculations. Goto Insert->Table to insert a grid of desired size

Ans1.1

(Carry)	1	1		1	1	
		1	1	0	1	1
+			1	0	1	1
(Sum)	1	0	0	1	1	0

Table 1

$11011_{bin} = (1*2^4) + (1*2^3) + (0*2^2) + (1*2^1) + (1*2^0)$

$= (1*16) + (1*8) + (0*4) + (1*2) + (1*1)$

$= 16 + 8 + 0 + 2 + 1$

$= 27_{dec}$

$1011_{bin} = (1*2^3) + (0*2^2) + (1*2^1) + (1*2^0)$

$= (1*8) + (0*4) + (1*2) + (1*1)$

$= 8 + 0 + 2 + 1$

$= 11_{dec}$

$100110_{bin} = (1*2^5) + (0*2^4) + (0*2^3) + (1*2^2) + (1*2^1) + (0*2^0)$

$= (1*32) + (0*16) + (0*8) + (1*4) + (1*2) + (0*1)$

$= 32 + 0 + 0 + 4 + 2 + 0$

$= 38_{dec}$

$27_{dec} + 11_{dec} = 38_{dec} = 100110_{bin}$

Ans: 100110_{bin}

Q1.2.

Rewrite the following base-10numbers as 8-bit two's complement integers: -31, &-59.

Ans1.2.

31_{dec} =

		REMAINDERS	
31/2	LSB	1	
15/2		1	
7/2		1	11111_{bin}
3/2		1	
½	MSB	1	

Table 2

As 8-bit two's complement integer $31_{dec} = 00011111_{bin}$

To calculate the 8-bit two's complement integer for -31_{dec} the following steps are carried out on 00011111_{bin}.

Step 01: Invert It = 11100000

Step 02: Add 1 = 1

11100001

Therefore: $-31_{dec} = 11100001_{bin}$ as 8-bit two's complement integer

59_{dec}=

		REMAINDERS	
59/2	LSB	1	
29/2		1	
14/2		0	111011_{bin}
7/2		1	
3/2		1	
½	MSB	1	

Table 3

As 8-bit two's complement integer $59_{dec} = 00111011_{bin}$

To calculate the 8-bit two's complement integer for -59_{dec} the following steps are carried out on 00111011_{bin}.

Step 01: Invert It = 11000100

Step 02: Add 1 = 1

11000101

Therefore: $-59_{dec} = 11000101_{bin}$ as 8-bit two's complement integer.

Ans: $-31_{dec} = 11100001_{bin}$ and $-59_{dec} = 11000101_{bin}$ as 8-bit two's complement integer.

Q1.3.

What does the bit pattern 11101001 represent if you interpret it as an 8-bit two's complement integer?

Ans1.3.

In computing signed number representations are required to encode negative number in binary number systems. This is because numbers are represented only as sequences of bits in computing, unlike mathematics were negative numbers are represented by prefixing them with a minus sign ("– ").

In two's complement notation positive integers are represented by its ordinary binary representation, with additional zeros required to bring it to the necessary bit representation.

The MSB for positive integers is zero (left must be zero) as it indicates that these numbers are not negative. A two's complement 8-bit number can only represent positive integers from 0 – 127. Examples of this are as follows:

- $00000000_{bin} = 0$
- $01111111_{bin} = 127$
- $00110111_{bin} = 55$

The MSB for negative integers is one (1) as it indicates that these numbers are negative. A two's complement 8-bit number can only represent negative integers from -1 to -128. Examples are as follows:

- $11111111_{bin} = -1$
- $11001100_{bin} = -52$
- $10000000_{bin} = -128$

The figure 1 shows the numeric range of two's complement 8-bit numbers and the binary equivalent.

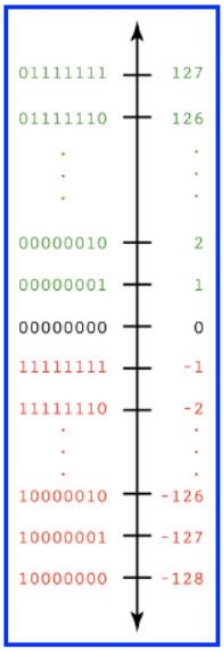

Figure 1

With the following considered we can now denote what the bit pattern 11101001_{bin} represents if you interpret it as a two's complement 8-bit number.

Ans: $11101001_{bin} = -23_{dec}$

Q1.4.

Draw up the truth table for the circuit below (inputs are X and Y and outputs are B and D). From observing the result, what function do you think this circuit performs?

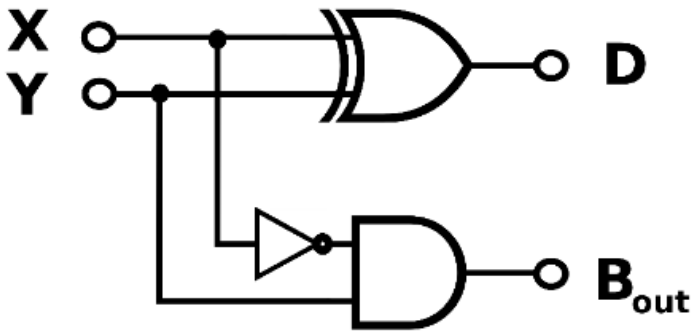

Figure 2

Ans1.4.

Truth Table:

X	Y	D	B_{out}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Table 4

Q1.5.

Draw the circuit diagram for the Boolean logic equation: (AB + C)D (Hint: you can use an online logic gate simulator such as: https://academo.org/demos/logic-gate-simulator/ and screen-capture your drawing)

Ans1.5.

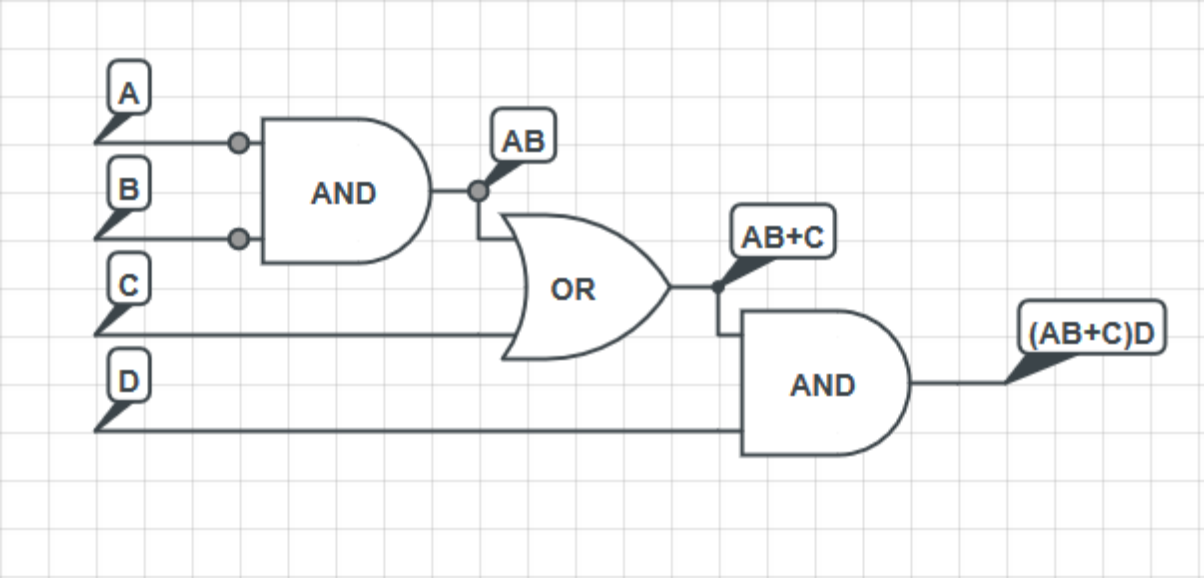

Figure 3

The image captured in Figure 3 was using software from https://www.circuitlab.com/.

Q2: Linux Assignment

Q2.1

Enter the commands below at the Linux terminal on the AWS VM (on which you completed your Linux Homework) and try to interpret the output.

For the submission:
In your own words, write a brief description of what each command does. Make sure to use Google as a resource and don't be afraid to experiment (as a normal user you cannot do much harm). Also include a screenshot of the output of the history command. Copy the screenshot into your ".docx" document for submission.

Commands:

- echo hello world↲
- passwd↲
- date↲*
- hostname↲*
- arch↲*
- uname -a↲*
- dmesg | more↲ (you may need to press q to quit)
- uptime↲*
- whoami↲*
- who↲*
- last↲
- finger↲*
- w↲*
- top↲*(you may need to press q to quit)
- echo $SHELL↲
- echo {con,pre}{sent,fer}{s,ed} ↲
- man ls↲ (you may need to press q to quit)
- man who↲ (you may need to press q to quit)
- clear↲
- cal 2000↲
- cal 9 1752↲ (do you notice anything unusual. Why is this the case?)
- yes please↲ (you may need to press Ctrl-c to quit)
- time sleep 5↲
- history↲*

Ans2.1

- **echo hello world↲**
  The command 'echo' prints a line of text. This demonstrated below in figure 4. The command 'echo hello world' prints the line of text "hello world".

```
*** System restart required ***
Last login: Sat Apr 25 08:27:21 2020 from 64.43.184.71
aronnutley@ip-172-31-3-33:~$ echo hello world
hello world
aronnutley@ip-172-31-3-33:~$
```

Figure 4

- **passwd↲**
  The command 'passwd' changes the password for a user account. The example below (Figure 5) shows a user changing their password.
  A superuser can change the password for any user.
  In the second example (Figure 6) you can see that the user was asked to verify their current password before changing it. The superuser can bypass this step when changing another user's password.

Once a User has confirmed their password, they will then enter and confirm their new password. If the password meets the criteria of the system, it will be accepted, and the password will be updated successfully.

```
aronnutley@ip-172-31-3-33:~$ passwd
Changing password for aronnutley.
(current) UNIX password:
```

Figure 5

```
aronnutley@ip-172-31-3-33:~$ passwd
Changing password for aronnutley.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
aronnutley@ip-172-31-3-33:~$
```

Figure 6

- **date**↵
  The 'date' command is used to show the system date and time. In the example below (Figure 7) you can see the command was executed on Saturday the 25th of April 2020 at 08:48 (09:48 local time).

```
aronnutley@ip-172-31-3-33:~$ date
Sat Apr 25 08:48:11 UTC 2020
aronnutley@ip-172-31-3-33:~$
```

Figure 7

- **hostname**↵
  The 'hostname' command is used to view a computer's hostname and domain name (DNS) (Domain Name Service). The example below (Figure 8) shows this on the Linux VM.

```
aronnutley@ip-172-31-3-33:~$ hostname
ip-172-31-3-33
aronnutley@ip-172-31-3-33:~$
```

Figure 8

- **arch**↵
  The 'arch' command is used to print the computer architecture. In the example below (Figure 9) we can see that the VM is using a 'x86_64' architecture. This means that the system is operation on a 64-bit version of the x86 instruction set.

```
aronnutley@ip-172-31-3-33:~$ arch
x86_64
aronnutley@ip-172-31-3-33:~$
```

Figure 9

- **uname -a↵**

  The ‘uname’ command is a command-line utility that prints basic information about the operating system name and system hardware.

  The syntax for the ‘uname’ command is as follows:

  uname [OPTION]

  Some of these options are as follows:

  1. ‘-s’ Prints the kernel name.
  2. ‘-n’ Prints the hostname.
  3. ‘-r’ Prints the kernel release.
  4. ‘-v’ Prints the kernel version.
  5. ‘-m’ Prints the name of the machine’s hardware name.
  6. ‘-o’ Prints the name of the operating system.

  The option ‘-a’ acts as an all option and behaves the same as if the option -snrvmo has been executed together. The example below (Fig X) shows this being executed.

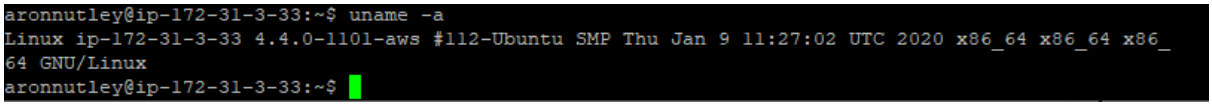

Figure 10

- **dmesg | more↵**

  The command ‘dmesg’ is used to display the kernel related messages. ‘dmesg’ stands for “display message or display driver”. ‘dmesg’ will print all messages but you will only see the latest message that fits onto the screen. If you want to analyse all the logs and display them on the page, pipe the ‘more’ command in addition to the ‘dmesg’. The syntax is ‘dmesg | more’ and the example below (Figure 11) shows this.

```
aronnutley@ip-172-31-3-33:~$ uname -a
Linux ip-172-31-3-33 4.4.0-1101-aws #112-Ubuntu SMP Thu Jan 9 11:27:02 UTC 2020 x86_64 x86_64 x86_
64 GNU/Linux
aronnutley@ip-172-31-3-33:~$ dmesg | more
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Initializing cgroup subsys cpuacct
[    0.000000] Linux version 4.4.0-1101-aws (buildd@lgw01-amd64-040) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu
1~16.04.12) ) #112-Ubuntu SMP Thu Jan 9 11:27:02 UTC 2020 (Ubuntu 4.4.0-1101.112-aws 4.4.208)
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.4.0-1101-aws root=LABEL=cloudimg-rootfs ro console=tty1 consol
e=ttyS0 nvme.io_timeout=4294967295 nvme_core.io_timeout=4294967295
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   Centaur CentaurHauls
[    0.000000] x86/fpu: xstate_offset[2]:  576, xstate_sizes[2]:  256
[    0.000000] x86/fpu: Supporting XSAVE feature 0x01: 'x87 floating point registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x02: 'SSE registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x04: 'AVX registers'
[    0.000000] x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'standard' format.
[    0.000000] e820: BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009dfff] usable
[    0.000000] BIOS-e820: [mem 0x000000000009e000-0x000000000009ffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000000e0000-0x00000000000fffff] reserved
[    0.000000] BIOS-e820: [mem 0x0000000000100000-0x000000003fffffff] usable
[    0.000000] BIOS-e820: [mem 0x00000000fc000000-0x00000000ffffffff] reserved
[    0.000000] NX (Execute Disable) protection: active
[    0.000000] SMBIOS 2.7 present.
[    0.000000] DMI: Xen HVM domU, BIOS 4.2.amazon 08/24/2006
[    0.000000] Hypervisor detected: Xen
[    0.000000] Xen version 4.2.
[    0.000000] Xen Platform PCI: I/O protocol version 1
[    0.000000] Netfront and the Xen platform PCI driver have been compiled for this kernel: unplug emulated NICs.
[    0.000000] Blkfront and the Xen platform PCI driver have been compiled for this kernel: unplug emulated disks.
               You might have to change the root device
               from /dev/hd[a-d] to /dev/xvd[a-d]
               in your root= kernel command line option
[    0.000000] HVMOP_pagetable_dying not supported
[    0.000000] e820: update [mem 0x00000000-0x00000fff] usable ==> reserved
[    0.000000] e820: remove [mem 0x000a0000-0x000fffff] usable
[    0.000000] e820: last_pfn = 0x40000 max_arch_pfn = 0x400000000
[    0.000000] MTRR default type: write-back
[    0.000000] MTRR fixed ranges enabled:
[    0.000000]   00000-9FFFF write-back
[    0.000000]   A0000-BFFFF write-combining
[    0.000000]   C0000-FFFFF write-back
[    0.000000] MTRR variable ranges enabled:
[    0.000000]   0 base 0000F0000000 mask 3FFFF8000000 uncachable
[    0.000000]   1 base 0000F8000000 mask 3FFFFC000000 uncachable
[    0.000000]   2 disabled
[    0.000000]   3 disabled
[    0.000000]   4 disabled
[    0.000000]   5 disabled
aronnutley@ip-172-31-3-33:~$
```

Figure 11

- **uptime**↵
  The 'uptime' command prints information relating to the time for which the system has been running (or up).
  In the example below (Figure 12) you can see the system has been running for the user aronnutley since 08:57:01 on the date that the command was executed (25/04/2020). The system has been up for 54 days 11 hours and 38 minutes, and that there was 1 user logged in.

```
aronnutley@ip-172-31-3-33:~$ uptime
 08:57:01 up 54 days, 11:38,  1 user,  load average: 0.00, 0.00, 0.00
aronnutley@ip-172-31-3-33:~$
```

Figure 12

- **whoami**↵
  The command 'whoami' prints the logged in user's identity. In the example below (Figure 13) you can see that the logged in user in this case is aronnutley.

```
aronnutley@ip-172-31-3-33:~$ whoami
aronnutley
aronnutley@ip-172-31-3-33:~$
```

Figure 13

- **who**↵
  The command 'who' lets you display the user's that are currently logged onto the system. In the first example below (Figure 14) the user aronnutley is the only user logged on at 08:34 UTC on the 25/04/2020. In the second example (Figure 15) you can see that additional users are logged in and the times at which they logged in.

```
aronnutley@ip-172-31-3-33:~$ who
aronnutley pts/0        2020-04-25 08:34 (64.43.184.71)
aronnutley@ip-172-31-3-33:~$
```

Figure 14

```
aronnutley@ip-172-31-3-33:~$ who
johannadeltagrimaldi pts/0        2020-04-25 13:45 (109.78.57.67)
danielclyne pts/1         2020-04-25 13:52 (212.17.34.71)
aronnutley pts/2        2020-04-25 13:11 (64.43.184.71)
aronnutley@ip-172-31-3-33:~$
```

Figure 15

- **last**↵
  The 'last' command displays a list of last logged in users. In the example below (Figure 16) you can see that aronnutley is the most recently logged in user and is still currently logged in. By interrogating the list in the example below (Figure 17) you can ascertain when certain users were last logged in. The 'last' command only shows the most recent users who were logged in and displays on the one screen. If you want to see additional log in's pipe the 'more' or 'less' commands.

```
aronnutley@ip-172-31-3-33:~$ last
aronnutl pts/0        64.43.184.71     Sat Apr 25 08:34   still logged in
aronnutl pts/0        64.43.184.71     Sat Apr 25 08:27 - 08:27  (00:00)
johannad pts/0        109.78.69.61     Fri Apr 24 22:05 - 22:07  (00:01)
johannad pts/0        109.78.69.61     Fri Apr 24 22:01 - 22:04  (00:03)
johannad pts/0        109.78.69.61     Fri Apr 24 21:57 - 22:00  (00:03)
johannad pts/0        109.78.69.61     Fri Apr 24 21:42 - 21:56  (00:14)
michaelc pts/1        92.251.224.141   Fri Apr 24 19:13 - 08:25  (13:11)
kateblac pts/2        40.115.105.60    Fri Apr 24 18:23 - 21:27  (03:03)
johannad pts/1        109.78.69.61     Fri Apr 24 17:51 - 18:30  (00:38)
jasminam pts/4        37.228.236.26    Fri Apr 24 17:24 - 17:26  (00:02)
kateblac pts/3        40.115.105.60    Fri Apr 24 17:14 - 20:08  (02:54)
johannad pts/1        109.78.69.61     Fri Apr 24 17:13 - 17:51  (00:37)
johannad pts/1        109.78.69.61     Fri Apr 24 17:08 - 17:13  (00:04)
johannad pts/1        109.78.69.61     Fri Apr 24 17:07 - 17:08  (00:01)
johannad pts/1        109.78.69.61     Fri Apr 24 16:45 - 17:06  (00:20)
johannad pts/1        109.78.69.61     Fri Apr 24 15:48 - 15:49  (00:00)
johannad pts/1        109.78.69.61     Fri Apr 24 15:46 - 15:48  (00:01)
johannad pts/1        109.78.69.61     Fri Apr 24 15:21 - 15:45  (00:24)
johannad pts/1        109.78.69.61     Fri Apr 24 14:20 - 15:21  (01:00)
johannad pts/1        109.78.69.61     Fri Apr 24 13:46 - 14:20  (00:33)
```

Figure 16

```
sarahmor pts/2        212.129.83.48    Wed Apr  1 12:07 - 12:13  (00:05)
bryanwal pts/1        40.115.105.60    Wed Apr  1 11:54 - 12:16  (00:22)
nnennama pts/0        37.228.233.94    Wed Apr  1 08:44 - 15:38  (06:53)
kamilabo pts/1        212.129.74.202   Wed Apr  1 08:35 - 11:38  (03:03)
nnennama pts/0        37.228.233.94    Wed Apr  1 07:06 - 08:42  (01:36)
nnennama pts/0        37.228.233.94    Wed Apr  1 06:48 - 07:04  (00:15)

wtmp begins Wed Apr  1 06:48:56 2020
aronnutley@ip-172-31-3-33:~$
```

Figure 17

- **finger**↲

  The 'finger' command is used to check the information of currently logged in users. In the example below (Figure 18) you can see that the user aronnutley is logged in. You can also see that the user is not idle and logged in on the 25/04/2020 at 08:34 UTC. When more users are logged in you can also see their details.

```
wtmp begins Wed Apr  1 06:48:56 2020
aronnutley@ip-172-31-3-33:~$ finger
Login       Name       Tty      Idle  Login Time   Office     Office Phone
aronnutley             pts/0         Apr 25 08:34 (64.43.184.71)
aronnutley@ip-172-31-3-33:~$
```

- **w**↲

  The command 'w' on the Linux OS provides a quick summary of every user logged into a computer. You can also see what activity each user is doing, and what load all this activity is imposing on the computer itself. The example below (Figure 19) is taken from the Linux VM and shows the 'w' command after being executed.

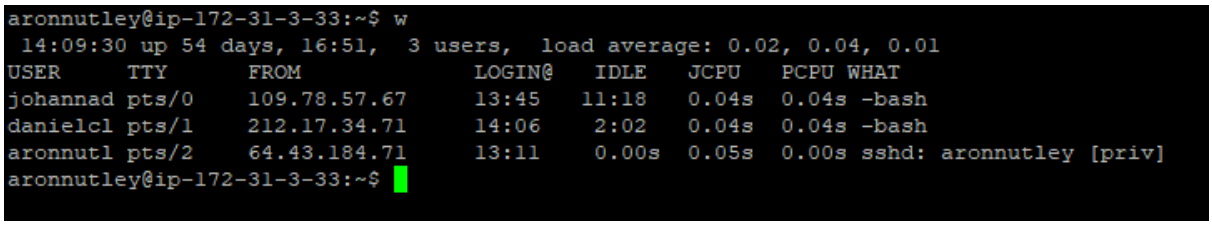

Figure 18

- **top↵**

  The 'top' command in Linux is used to show the Linux process. In the example below (Figure 20) you can see that when executed the command displays system summary information as well as a list of tasks currently being managed by the Linux kernel. The processes are listed out in columns each of which has a meaning. They are as follows:

PID	The tasks unique process ID.
USER	The username of the task's owner.
PR	The priority of the task.
NI	The nice value of the task. A positive nice value means lower priority, whereas a negative means higher. Zero in the column means priority will not be adjusted in determining a task's dispatchability.
VIRT	The total amount of virtual memory used by the task.
RES	The resident size, which is the non-swapped physical memory a task has used.
SHR	Shared Memory size (kB), which is the amount of shared memory used by a task.
%CPU	The CPU usage and the task's share of the elapsed CPU time since the last screen update. This is expressed as a percentage of total CPU time.
%MEM	Shows the memory usage. A tasks currently used share of available physical memory as a percentage.
TIME+	CPU Time
COMMAND	Displays the command line used to start a task or the name of the associated programme.

Table 5

```
aronnutley@ip-172-31-3-33:~$ top
top - 09:09:42 up 54 days, 11:51,  1 user,  load average: 0.00, 0.00, 0.00
Tasks: 114 total,   1 running, 113 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1014424 total,   170092 free,    93648 used,   750684 buff/cache
KiB Swap:        0 total,        0 free,        0 used.   688192 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
28882 aronnut+  20   0   40532   3680   3044 R   0.3  0.4   0:00.12 top
    1 root      20   0  185180   5336   3576 S   0.0  0.5   1:06.60 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.02 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:48.22 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S   0.0  0.0   0:56.99 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/0
   10 root      rt   0       0      0      0 S   0.0  0.0   0:22.94 watchdog/0
   11 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
   12 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 netns
   13 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 perf
   14 root      20   0       0      0      0 S   0.0  0.0   0:00.00 xenwatch
   15 root      20   0       0      0      0 S   0.0  0.0   0:00.00 xenbus
   17 root      20   0       0      0      0 S   0.0  0.0   0:01.14 khungtaskd
   18 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 writeback
   19 root      25   5       0      0      0 S   0.0  0.0   0:00.00 ksmd
   20 root      39  19       0      0      0 S   0.0  0.0   0:08.27 khugepaged
   21 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 crypto
   22 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kintegrityd
   23 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   24 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kblockd
   25 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 ata_sff
   26 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 md
   27 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 devfreq_wq
   30 root      20   0       0      0      0 S   0.0  0.0   0:01.81 kswapd0
   31 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 vmstat
   32 root      20   0       0      0      0 S   0.0  0.0   0:00.00 fsnotify_mark
   33 root      20   0       0      0      0 S   0.0  0.0   0:00.00 ecryptfs-kthrea
   49 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kthrotld
   50 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   51 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   52 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   53 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   54 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   55 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   56 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   57 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   58 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   59 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   60 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   61 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   62 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   63 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   64 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   65 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   66 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   67 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   68 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   69 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   70 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   71 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   72 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   73 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   74 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 nvme
```

Figure 19

- **echo $SHELL↵**
  Like with the command 'echo HelloWorld', the 'echo' command is being used print text. In the case demonstrated below (Figure 21) the command 'echo $SHELL' is being used to find out what shell is being ran. /bin/bash means the Bash Shell is being ran.

```
aronnutley@ip-172-31-3-33:~$ echo $SHELL
/bin/bash
```

Figure 20

- **echo {con,pre}{sent,fer}{s,ed} ↵**
  In this example echo is again being used to print text. In this case the shell is doing work to generate arbitrary strings. This process is known as brace expansion.
  In the example below (Figure 22) the printed test is based on what is in the curly brackets (braces), and how their content is divided with a ',' symbol. The first string in the first bracket is combined with the first string in the second and third bracket to give the string 'consents'. This process is again repeated except the first string in the first and second bracket is combined with the second string in the third bracket to give 'consented'. This pattern is repeated until the second string in the first second and third bracket are combined to give the string 'prefered'.
  In the second example below (Figure 23) you can see brace expansion for the command 'echo {0,1}{0,1}{0,1}↵'. In this example you can see that the binary digits are following the same pattern as in Figure 22.

```
aronnutley@ip-172-31-3-33:~$ echo {con,pre}{sent,fer}{s,ed}
consents consented confers confered presents presented prefers prefered
aronnutley@ip-172-31-3-33:~$
```

Figure 21

```
aronnutley@ip-172-31-3-33:~$ echo {0,1}{0,1}{0,1}
000 001 010 011 100 101 110 111
aronnutley@ip-172-31-3-33:~$
```

Figure 22

- **man ls↵**
  The 'man' command is used to display the user manual of any command. In the case below (Figure 24) the command 'man ls' has been executed. This gives the full name of the 'ls' command, a synopsis of it, and a description. Once you have finished with the command you have to press 'q' to quit.

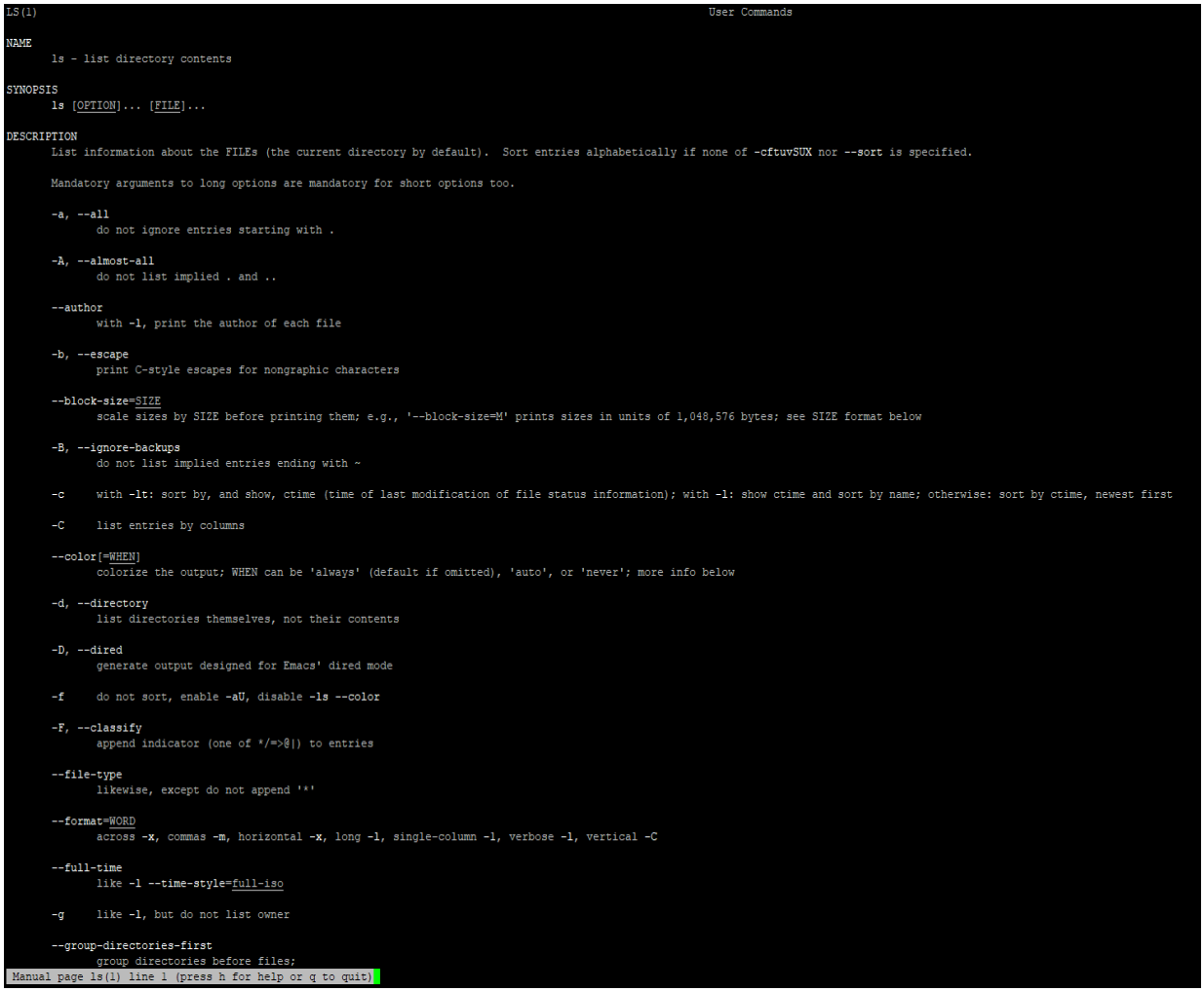
LS(1) User Commands
NAME
ls - list directory contents
SYNOPSIS
ls [OPTION]... [FILE]...
DESCRIPTION
List information about the FILEs (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.
Mandatory arguments to long options are mandatory for short options too.
-a, --all
do not ignore entries starting with .
-A, --almost-all
do not list implied . and ..
--author
with -l, print the author of each file
-b, --escape
print C-style escapes for nongraphic characters
--block-size=SIZE
scale sizes by SIZE before printing them; e.g., '--block-size=M' prints sizes in units of 1,048,576 bytes; see SIZE format below
-B, --ignore-backups
do not list implied entries ending with ~
-c with -lt: sort by, and show, ctime (time of last modification of file status information); with -l: show ctime and sort by name; otherwise: sort by ctime, newest first
-C list entries by columns
--color[=WHEN]
colorize the output; WHEN can be 'always' (default if omitted), 'auto', or 'never'; more info below
-d, --directory
list directories themselves, not their contents
-D, --dired
generate output designed for Emacs' dired mode
-f do not sort, enable -aU, disable -ls --color
-F, --classify
append indicator (one of */=>@|) to entries
--file-type
likewise, except do not append '*'
--format=WORD
across -x, commas -m, horizontal -x, long -l, single-column -1, verbose -l, vertical -C
--full-time
like -l --time-style=full-iso
-g like -l, but do not list owner
--group-directories-first
group directories before files;
Manual page ls(1) line 1 (press h for help or q to quit)

Figure 23

- **man who↵**

  Like the ‘man ls’ command the ‘man who’ command shows the manual for the command ‘who’. In the example below (Figure 25) you can see that the name of the command is given along with a synopsis and description. ‘q’ must be pressed to quit the command.

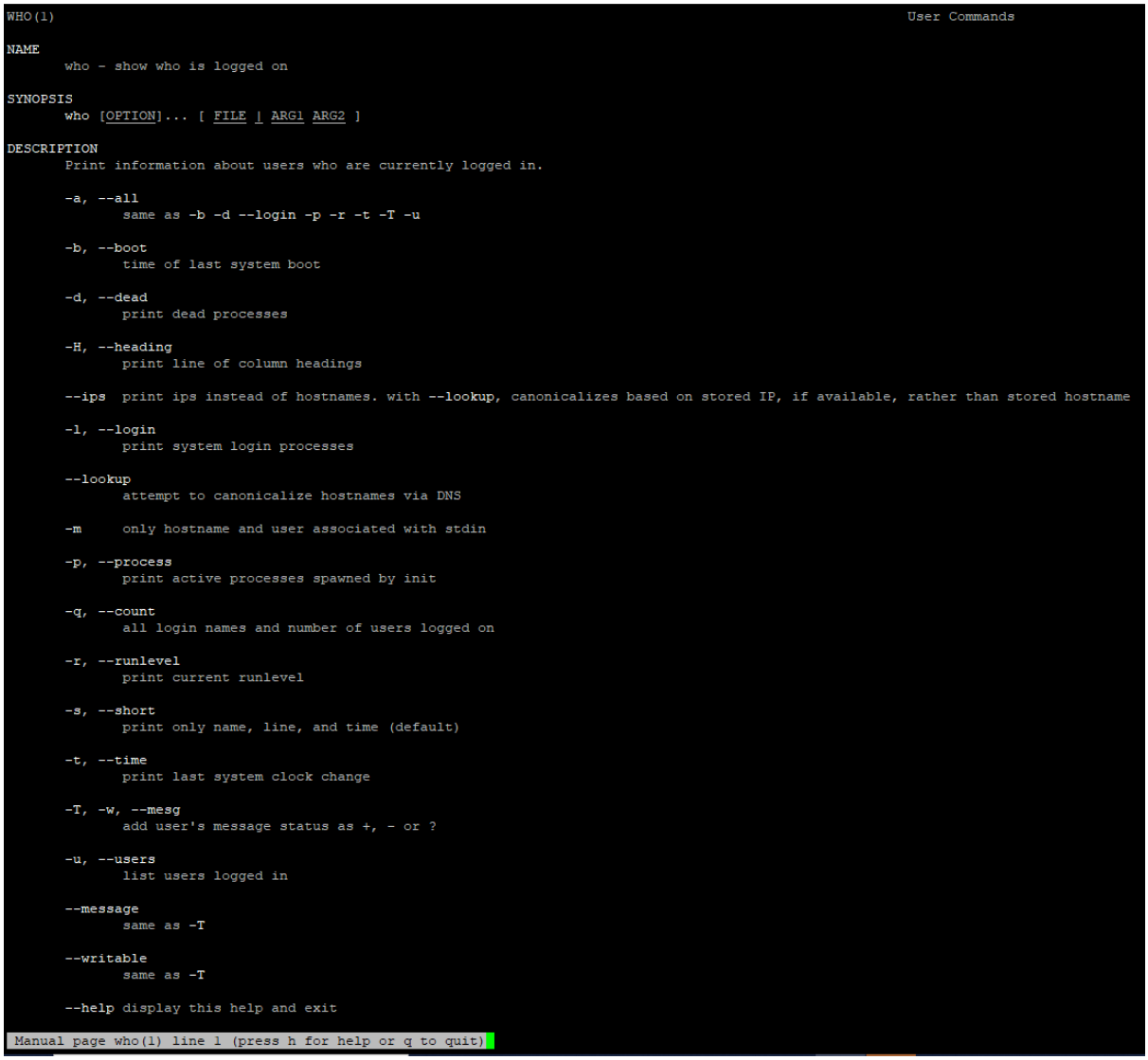

Figure 24

- **clear↵**
  The command ‘clear’ clears all text on the screen. The example below (Figure 26) shows this command after being executed.

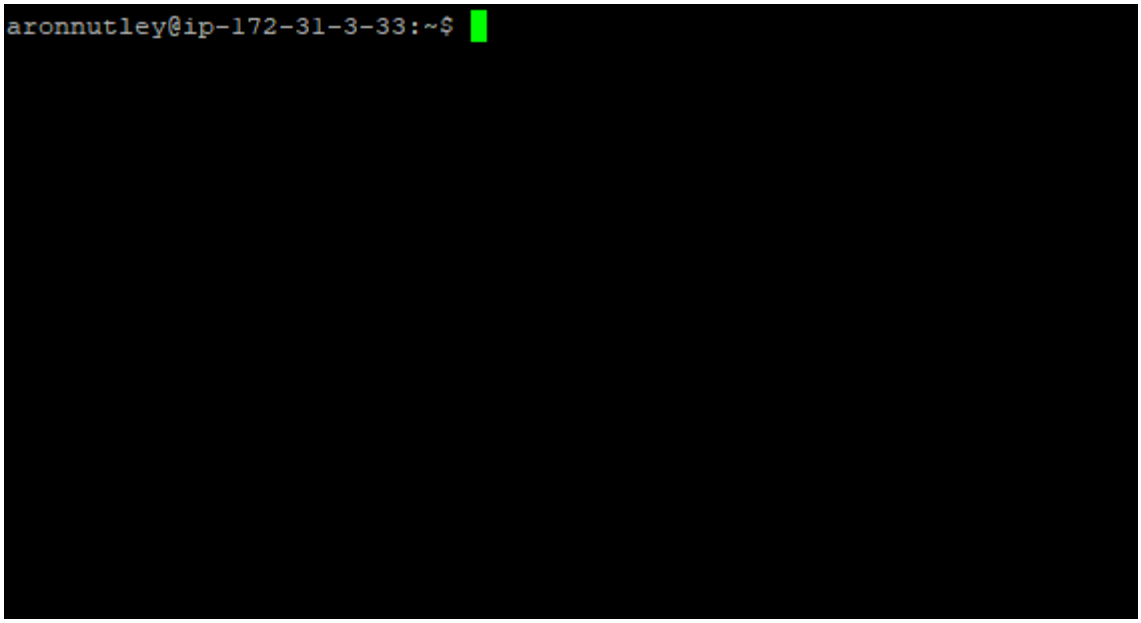

Figure 25

- **cal 2000**

  The command ‘cal 2000’ prints out a calendar for the year 2000. The example below (Figure 27) shows this command after being executed.

```
aronnutley@ip-172-31-3-33:~$ cal 2000
                              2000

      January               February                March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                   1            1  2  3  4  5              1  2  3  4
 2  3  4  5  6  7  8   6  7  8  9 10 11 12   5  6  7  8  9 10 11
 9 10 11 12 13 14 15  13 14 15 16 17 18 19  12 13 14 15 16 17 18
16 17 18 19 20 21 22  20 21 22 23 24 25 26  19 20 21 22 23 24 25
23 24 25 26 27 28 29  27 28 29              26 27 28 29 30 31
30 31

       April                  May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                   1      1  2  3  4  5  6                 1  2  3
 2  3  4  5  6  7  8   7  8  9 10 11 12 13   4  5  6  7  8  9 10
 9 10 11 12 13 14 15  14 15 16 17 18 19 20  11 12 13 14 15 16 17
16 17 18 19 20 21 22  21 22 23 24 25 26 27  18 19 20 21 22 23 24
23 24 25 26 27 28 29  28 29 30 31           25 26 27 28 29 30
30

        July                  August              September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                   1            1  2  3  4  5                 1  2
 2  3  4  5  6  7  8   6  7  8  9 10 11 12   3  4  5  6  7  8  9
 9 10 11 12 13 14 15  13 14 15 16 17 18 19  10 11 12 13 14 15 16
16 17 18 19 20 21 22  20 21 22 23 24 25 26  17 18 19 20 21 22 23
23 24 25 26 27 28 29  27 28 29 30 31        24 25 26 27 28 29 30
30 31

      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7            1  2  3  4                 1  2
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   3  4  5  6  7  8  9
15 16 17 18 19 20 21  12 13 14 15 16 17 18  10 11 12 13 14 15 16
22 23 24 25 26 27 28  19 20 21 22 23 24 25  17 18 19 20 21 22 23
29 30 31              26 27 28 29 30        24 25 26 27 28 29 30
                                            31
aronnutley@ip-172-31-3-33:~$
```

Figure 26

- **cal 9 1752↵**

  The command ‘cal 9 1752’ prints out a calendar for September 1752. What is strange about this is that the calendar skips eleven days from the 2/09/1752 to 14/09/1752, which can be seen in the example below (Figure 28). The reason this happened was because it was the time that Britain change from the Julian calendar to the Gregorian calendar, bringing it into line with most of Europe.

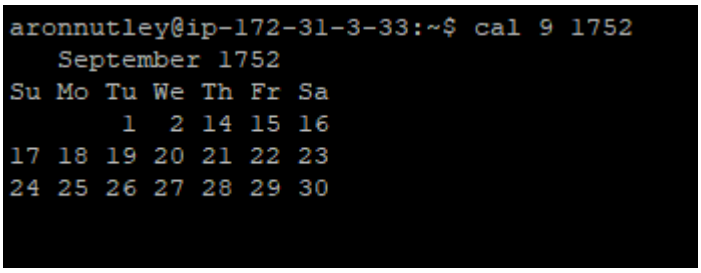

Figure 27

- **yes please↵**
  The command 'yes' outputs the same string in a constant stream. In the case of the command 'yes please', the string 'please' is constantly outputted until 'Ctrl-c' is pressed to quit. The example below (Figure 29) shows what happens after quitting which is seen on the second last line.

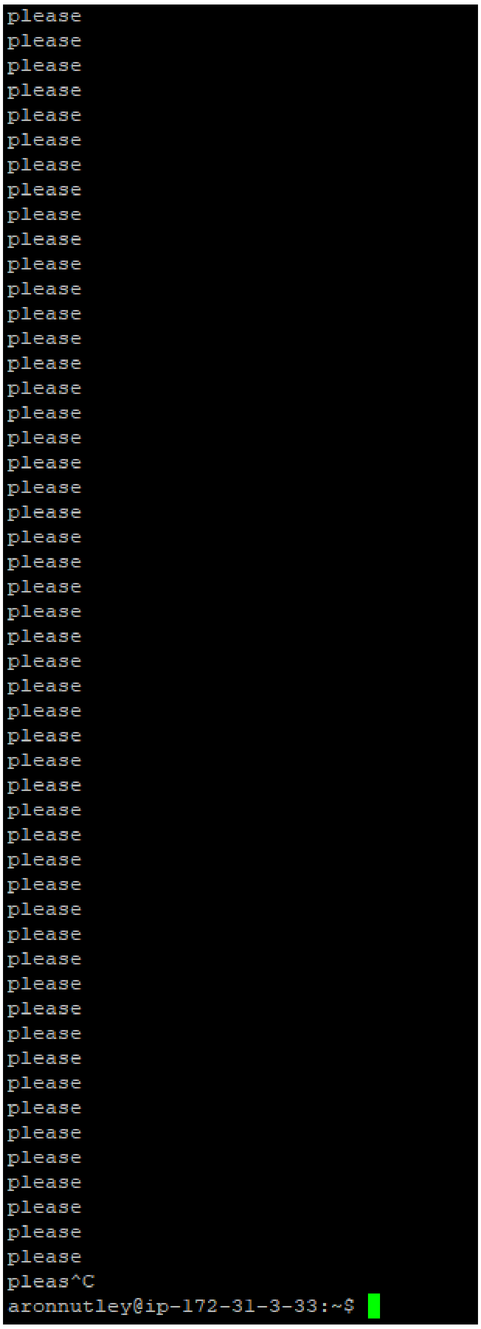

Figure 28

- **time sleep 5↲**

  The command 'time sleep 5' cases the system to sleep for 5 seconds. In the example below (figure 30) you can see this after being executed on the system.

```
aronnutley@ip-172-31-3-33:~$ time sleep 5

real    0m5.001s
user    0m0.000s
sys     0m0.000s
aronnutley@ip-172-31-3-33:~$
```

Figure 29

- **history↲**

  The history command prints a lest of all the other commands that have been run during the session. In the case below (Figure 31) you can see that the first command that has been executed is 'ls' and the 31st is 'pwd'.

```
aronnutley@ip-172-31-3-33:~$ history
    1  ls
    2  cd Music/
    3  cd Metallica/
    4  ls
    5  pwd
    6  ls
    7  ls -al
    8  ls
    9  ls -a
   10  cddir RideTheLightning
   11  mkdir RideTheLightning
   12  mkdir Load
   13  ld
   14  ls
   15  c
   16  ls
   17  cd
   18  ls
   19  date
   20  cal
   21  uptime
   22  w
   23  finger johannadeltagrimaldi
   24  whoami
   25  df
   26  cd
   27  mkdir Music
   28  cd Music/
   29  cd
   30  ls
   31  pwd
   32  ps
   33  ls
   34  cd dir Music/
```

Figure 30

Q2.2

This is a research project. Use Google to help you identify a solution. For each of the commands marked with an *, group them into a shell script so that you can automate execution of the commands. Write the shell script using the Vim text editor. Once you have verified that the script works, add output redirection to append the output of each command to a file named as follows: **firstnameSurname.txt** (replacing firstname and surname with your own details).When writing to this text file, make liberal use of the echo command within the shell script to format the output nicely –i.e. insert blank lines or other demarcations and headings to make your file easily readable.

For the submission:
Copy and paste the contents of this auto-created ".txt" file into your ".docx" document for submission. You are also required to upload the shell script which you wrote and the text file which it generated.

Ans2.2

The text below is copy and pasted from the submitted file "aronNutley.txt":

CATC Project Q2.2

Starting ShellScript...

date

Thu Apr 30 16:27:01 UTC 2020

hostname

ip-172-31-3-33

arch

x86_64

uname -a

Linux ip-172-31-3-33 4.4.0-1101-aws #112-Ubuntu SMP Thu Jan 9 11:27:02 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux

uptime

16:27:01 up 59 days, 19:08, 15 users, load average: 0.00, 0.00, 0.00

whoami

aronnutley

who

edelcurtin pts/0 2020-04-30 11:08 (18.202.243.115)

elenamartinlopez pts/1 2020-04-30 14:55 (40.115.105.60)

leanneboyd pts/2 2020-04-30 12:01 (89.124.19.3)

edelcurtin pts/3 2020-04-30 09:16 (86.43.179.139)

aronnutley pts/4 2020-04-30 15:17 (80.233.57.148)

davidlyons pts/5 2020-04-30 16:19 (37.228.251.214)

aronnutley pts/6 2020-04-30 16:19 (80.233.57.148)

aislingconsidine pts/7 2020-04-30 14:14 (80.233.104.186)

diarmaidtoomey pts/8 2020-04-30 12:38 (109.79.164.138)

isabelabellomo pts/9 2020-04-30 15:35 (37.228.248.45)

lynnfitzgerald pts/13 2020-04-30 14:46 (212.129.78.180)

aronnutley pts/14 2020-04-30 14:23 (80.233.57.148)

lucmcauley pts/15 2020-04-30 16:06 (37.228.254.39)

lynnfitzgerald pts/16 2020-04-30 15:09 (80.233.51.101)

jasminamkaurin pts/17 2020-04-30 15:25 (37.228.250.143)

finger

Login Name Tty Idle Login Time Office Office Phone

aislingconsidine pts/7 1:07 Apr 30 14:14 (80.233.104.186)

aronnutley pts/4 1:04 Apr 30 15:17 (80.233.57.148)

aronnutley pts/6 Apr 30 16:19 (80.233.57.148)

aronnutley pts/14 1:45 Apr 30 14:23 (80.233.57.148)

davidlyons pts/5 7 Apr 30 16:19 (37.228.251.214)

diarmaidtoomey pts/8 1:19 Apr 30 12:38 (109.79.164.138)

edelcurtin pts/0 Apr 30 11:08 (18.202.243.115)

edelcurtin pts/3 Apr 30 09:16 (86.43.179.139)

elenamartinlopez pts/1 1:28 Apr 30 14:55 (40.115.105.60)

isabelabellomo pts/9 27 Apr 30 15:35 (37.228.248.45)

jasminamkaurin pts/17 Apr 30 15:25 (37.228.250.143)

leanneboyd pts/2 9 Apr 30 12:01 (89.124.19.3)

lucmcauley pts/15 Apr 30 16:06 (37.228.254.39)

lynnfitzgerald pts/13 1:40 Apr 30 14:46 (212.129.78.180)

lynnfitzgerald pts/16 24 Apr 30 15:09 (80.233.51.101)

w

16:27:01 up 59 days, 19:08, 15 users, load average: 0.00, 0.00, 0.00

USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT

edelcurt pts/0 18.202.243.115 11:08 5.00s 0.12s 0.12s -bash

elenamar pts/1 40.115.105.60 14:55 1:28m 0.05s 0.05s -bash

leannebo pts/2 89.124.19.3 12:01 9:57 0.39s 0.03s vim while.sh

edelcurt pts/3 86.43.179.139 09:16 5.00s 0.35s 0.31s ssh edelcurtin@18.202.243.115

aronnutl pts/4 80.233.57.148 15:17 1:04m 0.04s 0.04s -bash

davidlyo pts/5 37.228.251.214 16:19 7:35 0.03s 0.03s -bash

aronnutl pts/6 80.233.57.148 16:19 5.00s 0.04s 0.00s /bin/bash ./aronNutley.sh

aislingc pts/7 80.233.104.186 14:14 1:07m 0.08s 0.04s vim

diarmaid pts/8 109.79.164.138 12:38 1:19m 0.15s 0.03s vim shell_script.sh

isabelab pts/9 37.228.248.45 15:35 27:09 0.09s 0.04s vi cupoftea.sh

lynnfitz pts/13 212.129.78.180 14:46 1:40m 0.04s 0.04s -bash

aronnutl pts/14 80.233.57.148 14:23 1:45m 0.06s 0.06s -bash

lucmcaul pts/15 37.228.254.39 16:06 5.00s 0.04s 0.04s -bash

lynnfitz pts/16 80.233.51.101 15:09 24:17 0.33s 0.30s vim

jasminam pts/17 37.228.250.143 15:25 5.00s 0.05s 0.05s -bash

top

[?1h=[?25l[H[2J(B[mtop - 16:27:01 up 59 days, 19:08, 15 users, load average: 0.00, 0.00, 0.00(B[m[39;49m(B[m[39;49m[K

Tasks:(B[m[39;49m[1m 192 (B[m[39;49mtotal,(B[m[39;49m[1m 1 (B[m[39;49mrunning,(B[m[39;49m[1m 187 (B[m[39;49msleeping,(B[m[39;49m[1m 4 (B[m[39;49mstopped,(B[m[39;49m[1m 0 (B[m[39;49mzombie(B[m[39;49m(B[m[39;49m[K

%Cpu(s):(B[m[39;49m[1m 0.1 (B[m[39;49mus,(B[m[39;49m[1m 0.1 (B[m[39;49msy,(B[m[39;49m[1m 0.0 (B[m[39;49mni,(B[m[39;49m[1m 99.8

(B[m[39;49mid,(B[m[39;49m[1m 0.0 (B[m[39;49mwa,(B[m[39;49m[1m 0.0
(B[m[39;49mhi,(B[m[39;49m[1m 0.0 (B[m[39;49msi,(B[m[39;49m[1m 0.0
(B[m[39;49mst(B[m[39;49m(B[m[39;49m[K

KiB Mem :(B[m[39;49m[1m 1014424 (B[m[39;49mtotal,(B[m[39;49m[1m   108632
(B[m[39;49mfree,(B[m[39;49m[1m   207140 (B[m[39;49mused,(B[m[39;49m[1m   698652
(B[m[39;49mbuff/cache(B[m[39;49m(B[m[39;49m[K

KiB Swap:(B[m[39;49m[1m   0 (B[m[39;49mtotal,(B[m[39;49m[1m   0
(B[m[39;49mfree,(B[m[39;49m[1m   0 (B[m[39;49mused.(B[m[39;49m[1m   569364
(B[m[39;49mavail Mem (B[m[39;49m(B[m[39;49m[K

[K

[7m PID USER   PR NI   VIRT   RES   SHR S %CPU %MEM   TIME+ COMMAND
(B[m[39;49m[K

(B[m   1 root   20   0 185180   5344   3576 S 0.0 0.5   1:10.27 systemd
(B[m[39;49m[K

(B[m   2 root   20   0   0   0   0 S 0.0 0.0   0:00.03 kthreadd
(B[m[39;49m[K

(B[m   3 root   20   0   0   0   0 S 0.0 0.0   0:51.98 ksoftirqd/0
(B[m[39;49m[K

(B[m   5 root   0 -20   0   0   0 S 0.0 0.0   0:00.00 kworker/0:0H
(B[m[39;49m[K

(B[m   7 root   20   0   0   0   0 S 0.0 0.0   0:58.28 rcu_sched
(B[m[39;49m[K

(B[m   8 root   20   0   0   0   0 S 0.0 0.0   0:00.00 rcu_bh
(B[m[39;49m[K

(B[m   9 root   rt   0   0   0   0 S 0.0 0.0   0:00.00 migration/0
(B[m[39;49m[K

(B[m   10 root   rt   0   0   0   0 S 0.0 0.0   0:25.20 watchdog/0
(B[m[39;49m[K

(B[m   11 root   20   0   0   0   0 S 0.0 0.0   0:00.00 kdevtmpfs
(B[m[39;49m[K

(B[m   12 root   0 -20   0   0   0 S 0.0 0.0   0:00.00 netns
(B[m[39;49m[K

(B[m   13 root   0 -20   0   0   0 S 0.0 0.0   0:00.00 perf
(B[m[39;49m[K

(B[m   14 root   20   0   0   0   0 S 0.0 0.0   0:00.00 xenwatch
(B[m[39;49m[K

(B[m   15 root   20   0   0   0   0 S 0.0 0.0   0:00.00 xenbus
(B[m[39;49m[K

(B[m 17 root 20 0 0 0 0 S 0.0 0.0 0:01.25 khungtaskd
(B[m[39;49m[K

(B[m 18 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 writeback
(B[m[39;49m[K

(B[m 19 root 25 5 0 0 0 S 0.0 0.0 0:00.00 ksmd
(B[m[39;49m[K

(B[m 20 root 39 19 0 0 0 S 0.0 0.0 0:08.27 khugepaged
(B[m[39;49m[K

(B[m 21 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 crypto
(B[m[39;49m[K

(B[m 22 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kintegrityd
(B[m[39;49m[K

(B[m 23 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 24 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kblockd
(B[m[39;49m[K

(B[m 25 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 ata_sff
(B[m[39;49m[K

(B[m 26 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 md
(B[m[39;49m[K

(B[m 27 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 devfreq_wq
(B[m[39;49m[K

(B[m 30 root 20 0 0 0 0 S 0.0 0.0 0:01.88 kswapd0
(B[m[39;49m[K

(B[m 31 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 vmstat
(B[m[39;49m[K

(B[m 32 root 20 0 0 0 0 S 0.0 0.0 0:00.00
fsnotify_mark
(B[m[39;49m[K

(B[m 33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 ecryptfs-
kthrea
(B[m[39;49m[K

(B[m 49 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kthrotld
(B[m[39;49m[K

(B[m 50 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 51 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 52 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 53 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 54 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 55 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 56 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 57 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 58 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 59 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 60 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 61 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 62 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 63 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 64 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 65 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 66 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 67 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 68 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 69 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 70 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 71 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 72 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 73 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
bioset
(B[m[39;49m[K

(B[m 74 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
nvme
(B[m[39;49m[K

(B[m 75 root 20 0 0 0 0 S 0.0 0.0 0:00.00
scsi_eh_0
(B[m[39;49m[K

(B[m 76 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
scsi_tmf_0
(B[m[39;49m[K

(B[m 77 root 20 0 0 0 0 S 0.0 0.0 0:00.00
scsi_eh_1
(B[m[39;49m[K

(B[m 78 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
scsi_tmf_1
(B[m[39;49m[K

(B[m 83 root 0 -20 0 0 0 S 0.0 0.0 0:00.00
ipv6_addrconf
(B[m[39;49m[K

(B[m   84 root   0 -20   0   0   0 S 0.0 0.0   0:00.00
bioset
(B[m[39;49m[K

(B[m   97 root   0 -20   0   0   0 S 0.0 0.0   0:00.00
deferwq
(B[m[39;49m[K

(B[m 249 root   0 -20   0   0   0 S 0.0 0.0   0:00.00
raid5wq
(B[m[39;49m[K[?1l>[70;1H

[?12l[?25h[K

history

```
    1  echo "CATC Project Q2.2"  >> aronNutley.txt
    2  echo "     " >> aronNutley.txt
    3  echo "Starting ShellScript..." >> aronNutley.txt
    4  echo "     " >> aronNutley.txt
    5  echo "date" >> aronNutley.txt
    6  date >> aronNutley.txt
    7  echo "     " >> aronNutley.txt
    8  echo "hostname" >> aronNutley.txt
    9  hostname >> aronNutley.txt
   10  echo "     " >> aronNutley.txt
   11  echo "arch" >> aronNutley.txt
   12  arch >> aronNutley.txt
   13  echo "    " >> aronNutley.txt
   14  echo "uname -a" >> aronNutley.txt
   15  uname -a >> aronNutley.txt
   16  echo "     " >> aronNutley.txt
   17  echo "uptime" >> aronNutley.txt
   18  uptime >> aronNutley.txt
   19  echo "     " >> aronNutley.txt
   20  echo "whoami" >> aronNutley.txt
   21  whoami >> aronNutley.txt
   22  echo "     " >> aronNutley.txt
   23  echo "who" >> aronNutley.txt
```

```
24  who >> aronNutley.txt
25  echo "    " >> aronNutley.txt
26  echo "finger" >> aronNutley.txt
27  finger >> aronNutley.txt
28  echo "    " >> aronNutley.txt
29  echo "w" >> aronNutley.txt
30  w >> aronNutley.txt
31  echo "    " >> aronNutley.txt
32  echo "top" >> aronNutley.txt
33  top >> aronNutley.txt
34  echo "       " >> aronNutley.txt
35  echo "history" >> aronNutley.txt
36  history >> aronNutley.txt
```

ShellScript Complete

Q2.3

When a user account is created on Linux machine, by default, it is public to all users on that machine. That is, anyone can view write and execute personal files of that user. The access permissions using the 'chmod' command so that your personal folder is fully accessible to you (read, write and execute privileges) and totally inaccessible to the group and all other users. Look at the lecture notes to work out how to do this. Run the "ls –l" command to verify that the permissions have been set correctly (Screenshot the result from this for submission).

Ans2.3.1

The command 'ls-l' when executed will show the directory listing and the access permissions to each. From the image below (Figure 31) you can also see that the 'chmod' command was used to change the access permissions for the user aronnutley. The user has full accessibility to the folder (read, write and execute privileges) and it is totally inaccessible to the group and all other users (apart from the Super User).

```
aronnutley@ip-172-31-3-33:/home$ ls -l
total 264
drwxr-xr-x 4 adedotunadekeye      adedotunadekeye      4096 Apr 16 23:48 adedotunadekeye
drwx------ 5 adrianodonoghue      adrianodonoghue      4096 Apr 23 20:38 adrianodonoghue
drwxr-xr-x 4 aislingconsidine     aislingconsidine     4096 Apr  3 12:32 aislingconsidine
drwxr-xr-x 4 aleksanderdziki      aleksanderdziki      4096 Mar 12 13:54 aleksanderdziki
drwxr-xr-x 4 anamarialulea        anamarialulea        4096 Apr  7 22:13 anamarialulea
drwx------ 4 aronnutley           aronnutley           4096 Apr  4 19:01 aronnutley
```

Figure 31

Q2.3.2

Because there is no GUI installed on the VM, all programs have to run instead in text mode. Use the 'lynx' text-based browser program on the VM. To run it, type:

- lynx www. google.com

In the lynx browser, search for an online IP location service to determine the city and country where the VM is located. Type the IP address of the VM into the IP locator website and it will tell you the VM's location (Screenshot the result from this for submission).

Ans2.3.2

Using the lynx browser and the search engine Google the following details about where the VM were discovered.

Location	
City:	Dublin
Postal Code:	D02
Country:	Ireland (IE)
Continent:	Europe (EU)
Coordinates:	53.3338 (lat) / -6.2488 (long)
Network	
IP address:	18.202.243.115
Hostname:	Ec2-18-202-243-115.eu-west-1.compute.amazonnaws.com
Provider:	AMAZON-02
ASN:	16509

The screen capture (Figure 32) below shows the work carried out on Linux to ascertain this information.

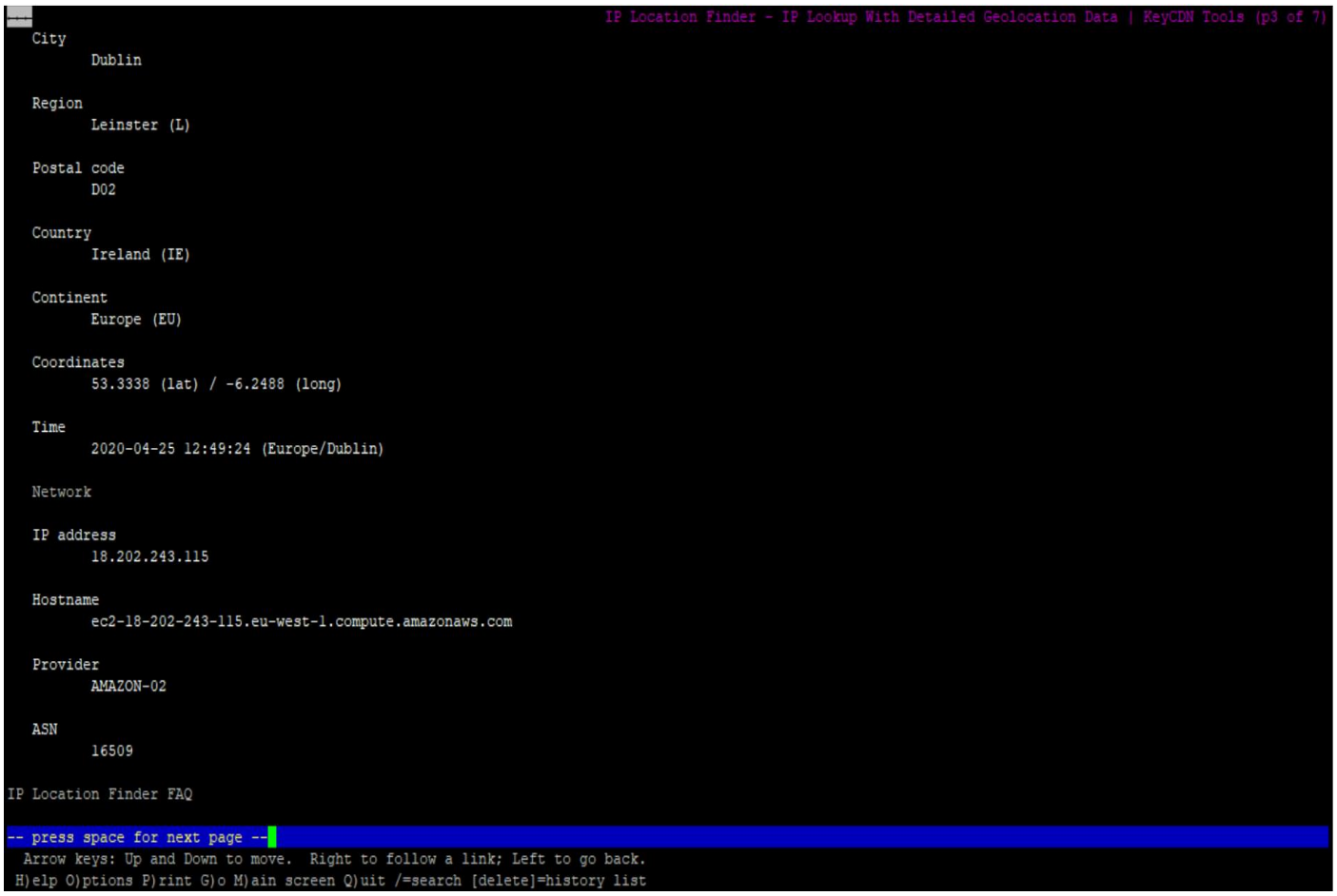
IP Location Finder - IP Lookup With Detailed Geolocation Data | KeyCDN Tools (p3 of 7)
City
Dublin
Region
Leinster (L)
Postal code
D02
Country
Ireland (IE)
Continent
Europe (EU)
Coordinates
53.3338 (lat) / -6.2488 (long)
Time
2020-04-25 12:49:24 (Europe/Dublin)
Network
IP address
18.202.243.115
Hostname
ec2-18-202-243-115.eu-west-1.compute.amazonaws.com
Provider
AMAZON-02
ASN
16509
IP Location Finder FAQ
-- press space for next page --
Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list

Figure 32

Q2.4

This is a research project. Use Google to help you identify a solution.

The objective of the task is to (using the Vim text editor)write a shell script program that behaves like an Irish person offering a cup of tea.

If the user types ‘y’ to the offer, the program displays “Great, I’ll make tea now” to the console.

If the user types ‘n’ to the offer, the program asks the user “Are you sure” 4 more times before giving up. If at any point during the 4 follow up offers, the user changes their mind and presses ‘y’, the computer will print out “Great, I’ll make tea now” to the console.

In addition to shell scripting, this assignment examines your ability to use ‘while loops’ and ‘if statements’ correctly. It also examines your ability to research and locate the information required online.

For the submission:

Capture a screenshot of the program in operation. Copy and paste the screenshot into your document for submission. Also, copy and paste the shell script code into your Word document for submission.

Ans2.4

The shell script for this Question is called “aronNutleyTea2.sh”. In the image (Figure 33) below you can see that the script it designed to ask the user; “Would you like a cup of tea?”.

The user will ‘y’ (for yes) or ‘n’ (for no) to answer the question.

If the user inputs ‘y’ the script will then print “Great, I’ll make tea now.” This can be seen on line two and three.

If the user inputs anything other than ‘y’ or ‘n’ the script will print “It’s a ‘y’ of ‘n’ question.” This can be seen in the example below then the user inputs “gibberish”

If the user inputs ‘n’ the script with print “Go on have a cup!” and prompt the user to answer again.

If the user inputs ‘n’ four more times the script will then print “Don’t say I didn’t ask” and then terminate.

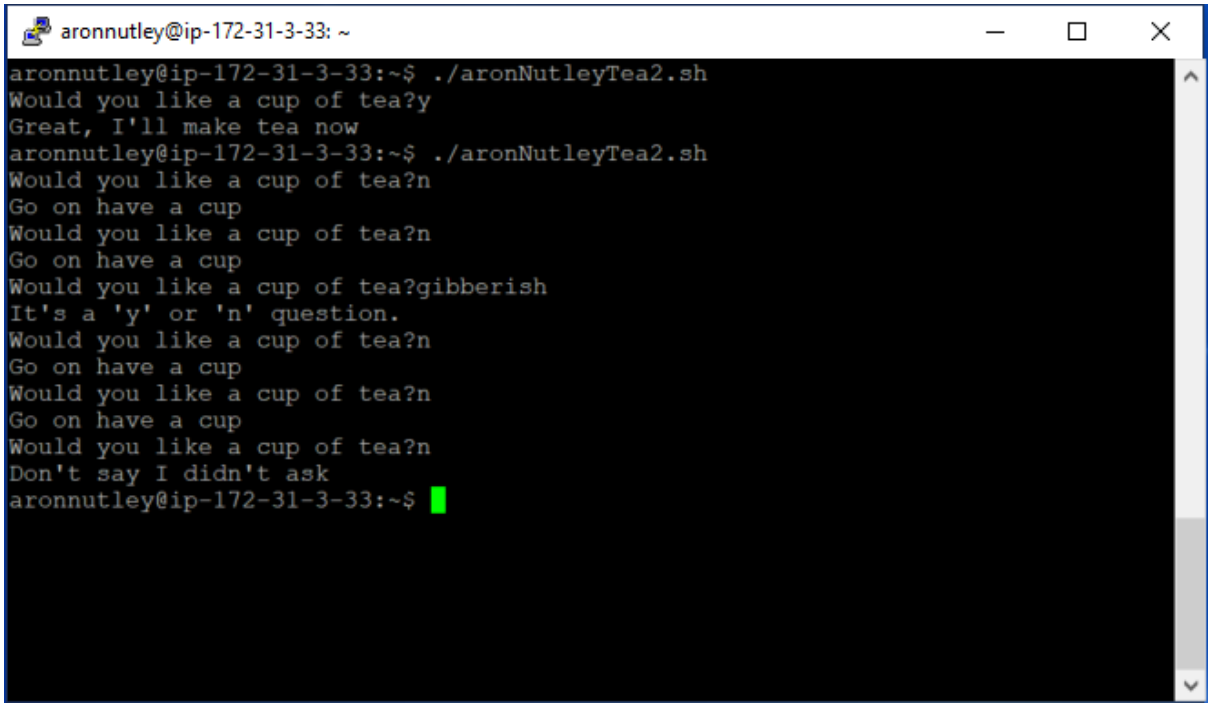
aronnutley@ip-172-31-3-33: ~
aronnutley@ip-172-31-3-33:~$ ./aronNutleyTea2.sh
Would you like a cup of tea?y
Great, I'll make tea now
aronnutley@ip-172-31-3-33:~$ ./aronNutleyTea2.sh
Would you like a cup of tea?n
Go on have a cup
Would you like a cup of tea?n
Go on have a cup
Would you like a cup of tea?gibberish
It's a 'y' or 'n' question.
Would you like a cup of tea?n
Go on have a cup
Would you like a cup of tea?n
Go on have a cup
Would you like a cup of tea?n
Don't say I didn't ask
aronnutley@ip-172-31-3-33:~$

Figure 33

Ans 2.4 Shell Script

```
#!/bin/bash

# CATC Assignment Q2.4

#Author Aron Nutley

_ANSWER=""
_COUNT=0 #variable to store no answers



while [ ${_COUNT} -lt 10 ] ; do #while loop opened

        read -p "Would you like a cup of tea?" _ANSWER

        if [ "${_ANSWER}" == "y" ]; then
                echo "Great, I'll make tea now."
                break #User only needs to say 'y' once and script will terminate

        elif [ "${_COUNT}" == 4 ]; then
                echo "Don't say I didn't ask!"
                break #User will be asked a total of 5 times before script terminates with above
statement

        elif [ "${_ANSWER}" == "n" ]; then
                echo "Go on have a cup!"
                _COUNT=$((${_COUNT} +1))

        else
                echo "It's a 'y' or 'n' question." #echo statement to pick up errors
```

```
        fi

done
```

Glossary of Terms

TERM	MEANING
DNS	Domain Name Service
LSB	Least Significant Bit
MSB	Most Significant Bit

Table 6