

**ECU178 Computer Science:
207SE Operating Systems, Security and Networks
Portfolio**

Due on Monday, December 15th, 2014

Dr Mark Elshaw

Robert Rigler : 4939377

Contents

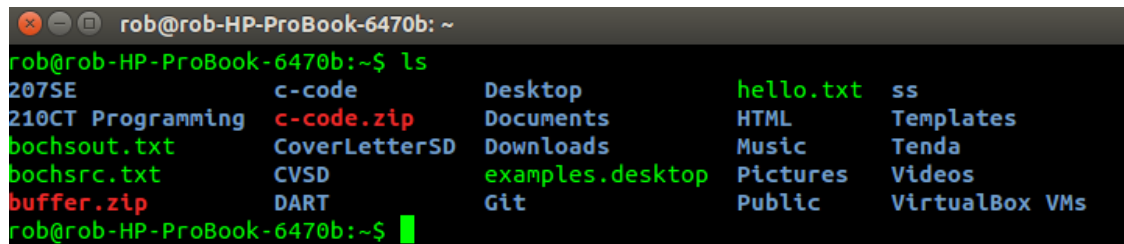
Item 1 - Linux Command Line	4
1. Make home directory read/write/executable only for myself	4
2. Download the script from http://www.centerkey.com/tree/tree.sh and make it executable	4
3. Making Directories	4
4. Display who is logged on and sort them in alphabetical order	5
5. Talk, Write and Wall	5
6. Stop bugging me	5
7. Find the number of words and lines in poem.txt	5
8. Grep the lines containing 'and' and the number of lines	6
9. Use cat to display the contents of the file	6
10. Sort the contents of poem.txt and redirect the output to poem2.txt	6
11. Sort the contents of poem.txt, reverse it, and then redirect the output to poem2.txt	6
Item 2 - Assembly Code	8
1. Right-Angled triangle	8
2. Isosceles Triangle	10
Item 3 - Bootloader	13
1. Boot pragma-linux using bochs	13
1. Make a Bootloader that displays my name	15
2. Make a Bootloader that displays a triangle of dots	16
Item 4 - Inside Proc	17
1. List the CPU Information using the Cat Command	17
2. Show a table of the interrupts on the system	18
3. Show number of CPUs, the producer of the CPUs and the CPU Model.	19
4. How the parameters that are passed to the kernel when starting up linux.	19
5. Show the name of the output devices and the number of megabytes read per second during the second sampled interval.	19
6. Menu based shell script.	20
Item 5 - Buffer tutorial	23
1. Commented version of the provided code	23
2. Evidence of compiled code	24
3. Code adaptation to show how many characters were read in total and how many times the buffer was filled	25
3a Evidence	26
4. Altering the buffer size	26
5. Adapt the code so that it is possible to compare if two files are the same.	27
5a. Evidence of comparison between review.txt and argo.txt	28
5b. Evidence of comaprison between argo.txt and reviewobserver.txt	28
Item 6 - Cache tutorial	29
1. Complete the cr_read_byte function	29
2. Prove the file is being buffered	30
3. Provide some statistics	31

Item 7 - Kernel	36
1. Description of the commands for loading and unloading Linux kernel modules.	36
2. List of the loaded modules	36
3. Description of four loaded modules	38
Item 8 - Kernel part II	39
Changing the code	39

Item 1 - Linux Command Line

1. Make home directory read/write/executable only for myself

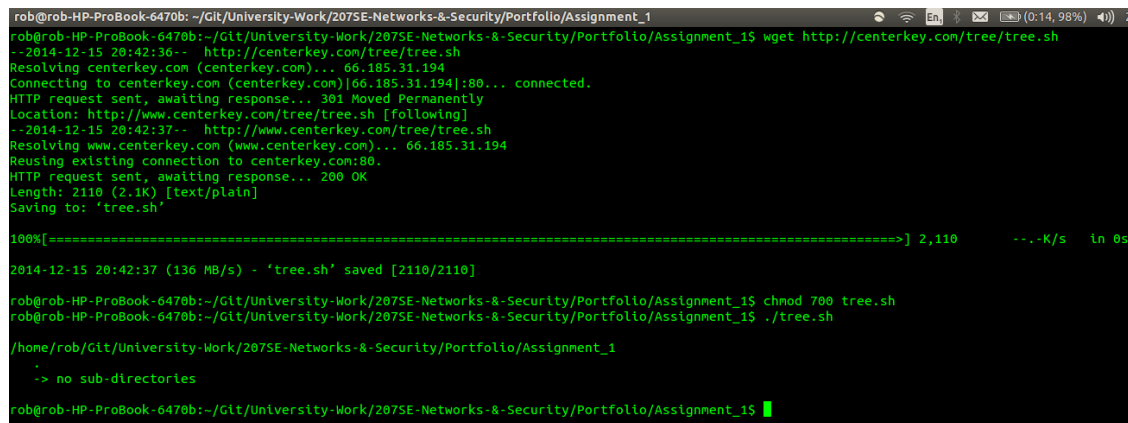
1. Linux Command: `chmod u+rwX /`



```
rob@rob-HP-ProBook-6470b: ~$ ls
207SE          c-code        Desktop       hello.txt     ss
210CT Programming c-code.zip    Documents     HTML          Templates
bochsout.txt   CoverLetterSD Downloads     Music         Tenda
bochsrc.txt    CVSD          examples.desktop Pictures       Videos
buffer.zip     DART          Git           Public        VirtualBox VMs
rob@rob-HP-ProBook-6470b: ~$
```

2. Download the script from <http://www.centerkey.com/tree/tree.sh> and make it executable

1. Linux Command: `wget http://www.centerkey.com/tree/tree.sh`
2. Linux Command: `chmod 700 tree.sh`



```
rob@rob-HP-ProBook-6470b: ~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_1
rob@rob-HP-ProBook-6470b:~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_1$ wget http://centerkey.com/tree/tree.sh
--2014-12-15 20:42:30-- http://centerkey.com/tree/tree.sh
Resolving centerkey.com (centerkey.com)... 66.185.31.194
Connecting to centerkey.com (centerkey.com)[66.185.31.194]:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://www.centerkey.com/tree/tree.sh [following]
--2014-12-15 20:42:37-- http://www.centerkey.com/tree/tree.sh
Resolving www.centerkey.com (www.centerkey.com)... 66.185.31.194
Reusing existing connection to centerkey.com:80.
HTTP request sent, awaiting response... 200 OK
Length: 2110 (2.1K) [text/plain]
Saving to: 'tree.sh'

100%[=====] 2,110 --.-K/s in 0s

2014-12-15 20:42:37 (136 MB/s) - 'tree.sh' saved [2110/2110]

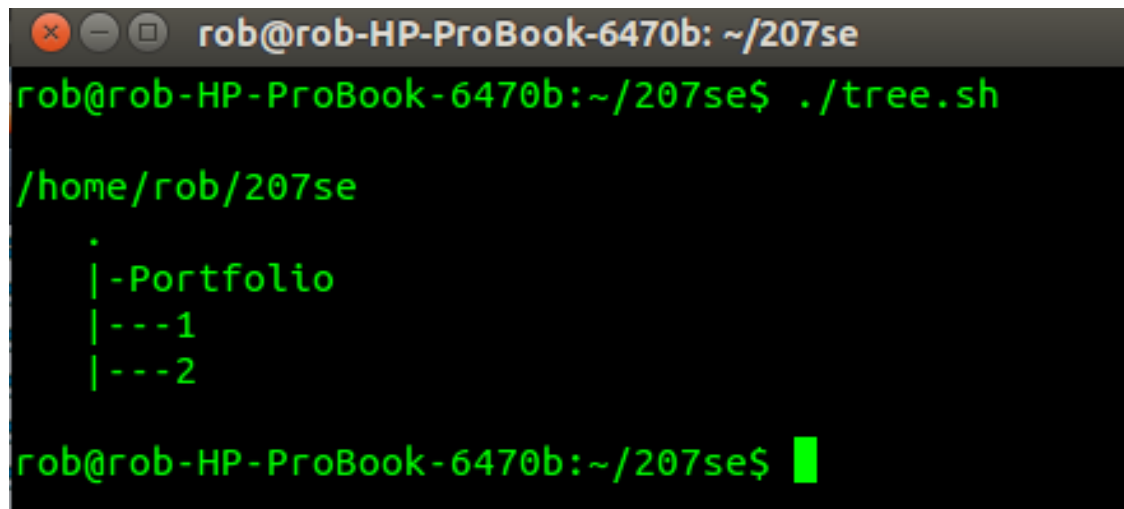
rob@rob-HP-ProBook-6470b:~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_1$ chmod 700 tree.sh
rob@rob-HP-ProBook-6470b:~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_1$ ./tree.sh

/home/rob/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_1
.-> no sub-directories

rob@rob-HP-ProBook-6470b:~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_1$
```

3. Making Directories

1. Making a directory called 207SE in Home folder
Linux Command: `mkdir 207SE`
2. Create sub-directory called Portfolio
Linux Command: `mkdir Portfolio`
3. Create numbered directories from week 1 and week 2
Linux Command: `mkdir 1`
Linux Command: `mkdir 2`
4. Transfer work from week 1 into the folder called 1
Linux Command: `mv HarvardArchitecture.txt /207SE/Portfolio/1`
5. Evidence of directory using tree.sh

A terminal window titled 'rob@rob-HP-ProBook-6470b: ~/207se'. The user enters the command './tree.sh'. The output shows a directory tree for '/home/rob/207se'. It lists a directory '-Portfolio' which contains two sub-directories, '1' and '2'.

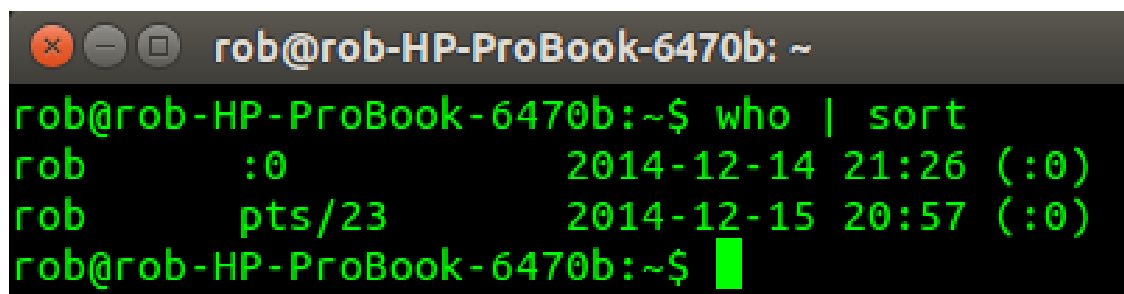
```
rob@rob-HP-ProBook-6470b: ~/207se
rob@rob-HP-ProBook-6470b:~/207se$ ./tree.sh

/home/rob/207se
.
|-Portfolio
|---1
|---2

rob@rob-HP-ProBook-6470b:~/207se$
```

4. Display who is logged on and sort them in alphabetical order

1. Linux Command: `who — sort`
2. Evidence of output:

A terminal window titled 'rob@rob-HP-ProBook-6470b: ~'. The user enters the command 'who | sort'. The output lists the current users: 'rob' with PID 0, logged in on 2014-12-14 at 21:26, and 'rob' with PID pts/23, logged in on 2014-12-15 at 20:57.

```
rob@rob-HP-ProBook-6470b:~$ who | sort
rob      :0                2014-12-14 21:26 (:0)
rob      pts/23           2014-12-15 20:57 (:0)
rob@rob-HP-ProBook-6470b:~$
```

5. Talk, Write and Wall

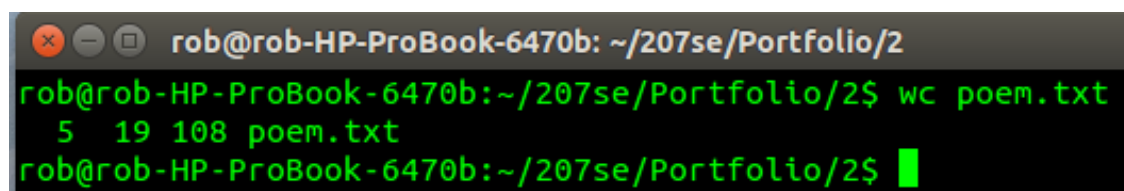
1. Write: Writes what the user types to the terminal of another specified user.
2. Talk: Opens a two-way conversation using a terminal
3. Wall: displays a message to all users on the system

6. Stop bugging me

1. 'mesg n' Will stop write, wall and talk from interrupting the user.

7. Find the number of words and lines in poem.txt

1. Linux Command. `wc poem.txt`
Evidence of output:

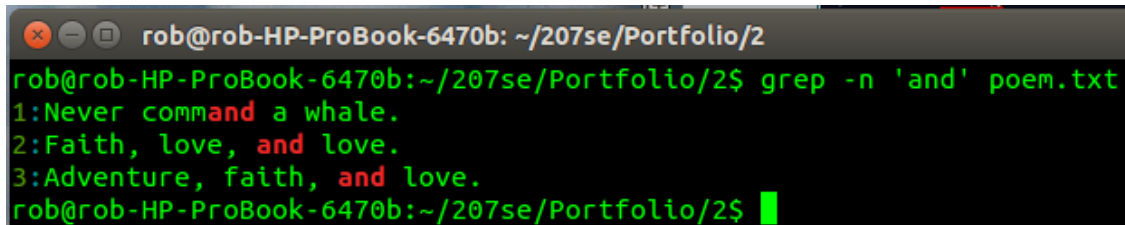
A terminal window titled 'rob@rob-HP-ProBook-6470b: ~/207se/Portfolio/2'. The user enters the command 'wc poem.txt'. The output shows '5' lines, '19' words, and '108' characters in 'poem.txt'.

```
rob@rob-HP-ProBook-6470b:~/207se/Portfolio/2$ wc poem.txt
 5  19 108 poem.txt
rob@rob-HP-ProBook-6470b:~/207se/Portfolio/2$
```

8. Grep the lines containing 'and' and the number of lines

1. Linux Command: `grep -n 'and' poem.txt`

Evidence of Output:

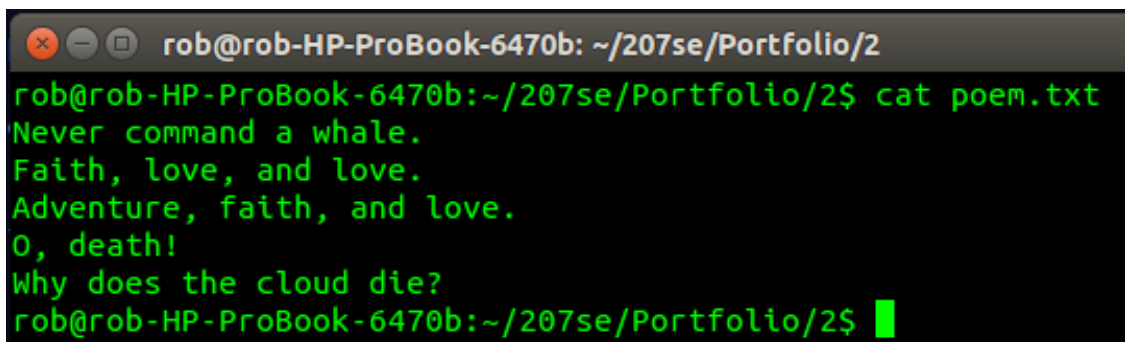


```
rob@rob-HP-ProBook-6470b: ~/207se/Portfolio/2
rob@rob-HP-ProBook-6470b:~/207se/Portfolio/2$ grep -n 'and' poem.txt
1:Never command a whale.
2:Faith, love, and love.
3:Adventure, faith, and love.
rob@rob-HP-ProBook-6470b:~/207se/Portfolio/2$
```

9. Use cat to display the contents of the file

1. Linux Command: `cat poem.txt`

Evidence of Output:

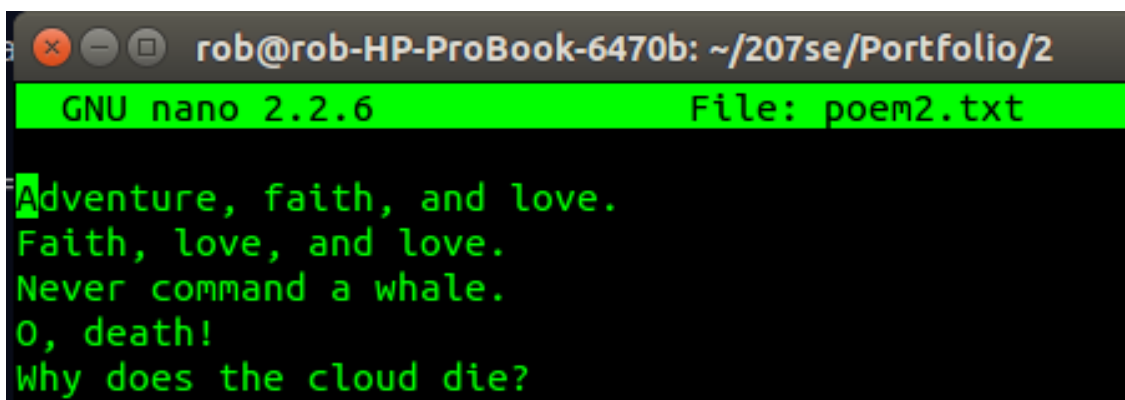


```
rob@rob-HP-ProBook-6470b: ~/207se/Portfolio/2
rob@rob-HP-ProBook-6470b:~/207se/Portfolio/2$ cat poem.txt
Never command a whale.
Faith, love, and love.
Adventure, faith, and love.
O, death!
Why does the cloud die?
rob@rob-HP-ProBook-6470b:~/207se/Portfolio/2$
```

10. Sort the contents of poem.txt and redirect the output to poem2.txt

1. Linux Command: `sort poem.txt >poem2.txt`

Evidence of Output:

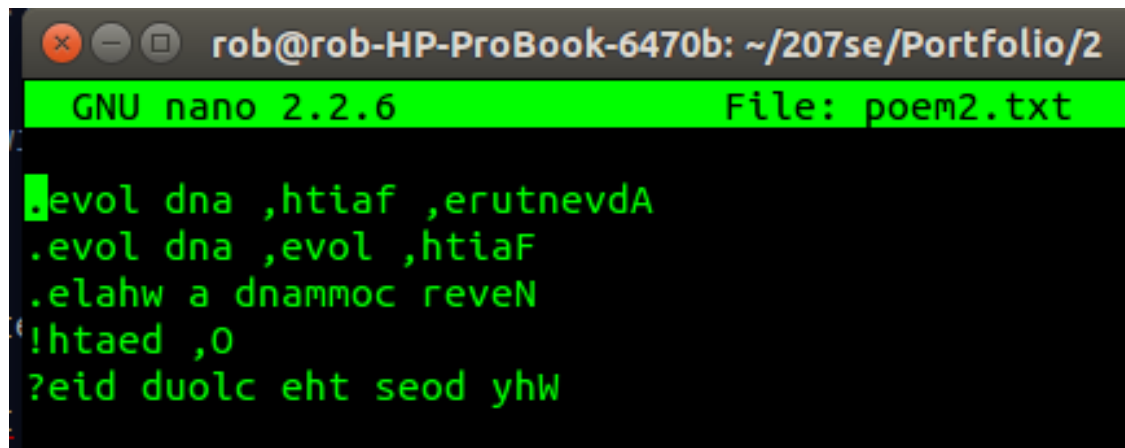


```
GNU nano 2.2.6 File: poem2.txt
Adventure, faith, and love.
Faith, love, and love.
Never command a whale.
O, death!
Why does the cloud die?
```

11. Sort the contents of poem.txt, reverse it, and then redirect the output to poem2.txt

1. Linux Command: `sort poem.txt — rev >poem2.txt`

Evidence of Output:



```
rob@rob-HP-ProBook-6470b: ~/207se/Portfolio/2
GNU nano 2.2.6 File: poem2.txt

evol dna ,htiaf ,erutnevda
.evol dna ,evol ,htiaF
.elahw a dnammoc reven
!htaed ,0
?eid duolc eht seod yhW
```

Item 2 - Assembly Code

1. Right-Angled triangle

The aim of this task was to produce a right-angled triangle of height specified by the user.

The process outline for this task is quite simple:

1. Get the user input and assign that value to the ecx register
2. Start the outer loop, then push the outerLoop value to the stack and move the innerLoop value into ecx.
3. Start the InnerLoop and print the '*' character the number of times specified in ecx.
4. After the InnerLoop is finished, start a new line and pop the OuterLoop value off of the stack.
5. Increment the InnerLoop value by 1 (So that one more triangle is draw next iteration) and repeat the Outerloop with the value in the ecx register (which is automatically decremented by one each time the Outerloop is called).

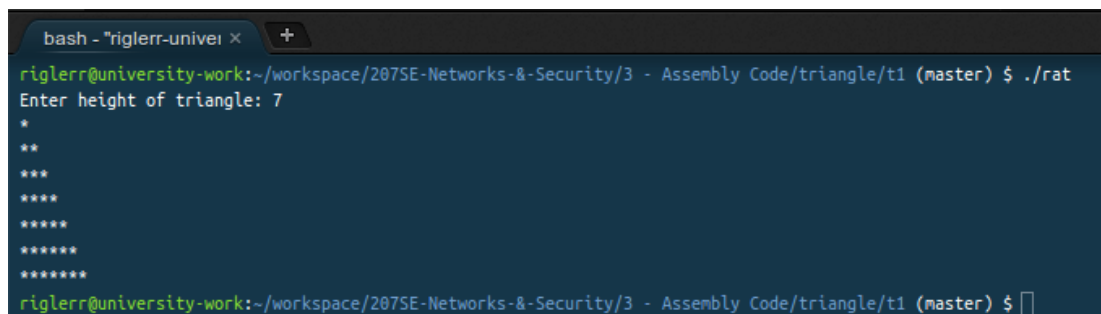
Listing 1: Right angled triangle

```
1 section .data
2
3 Prompt db 'Enter height of triangle: '
4 pLen equ $-Prompt
5 chr db '*'
6 nl:     db "  ", 0x0a ; variable to draw a new line
7 nl_len equ $-nl ; length of new line variable
8
9 section .bss
10 num resb 2 ;reserve 2 bytes for the input variable
11
12 section .text
13 global _start
14 _start:
15
16 ;Ask user for size of triangle
17 mov eax,4
18 mov ebx,1
19 mov ecx,Prompt
20 mov edx,pLen
21 int 80h
22
23 ;store the variable
24 mov eax,3
25 mov ebx,0
26 mov ecx,num
27 mov edx,2
28 int 80h
29
30 mov ecx, [num] ;dereference input and store in ecx
31 sub ecx,'0' ;convert from ascii to decimal
32 xor ch,ch ; clear upper half of ecx
33
```



```
34 mov ebx,1
35
36 lo: ;outer loop, amount of lines in triangle
37     push ecx ; push outer loop count to stack
38     mov ecx,ebx ; place inner loop count in the loop counter
39     li: ; inner loop, amount of stars in line
40
41         push ecx
42         push ebx ;push ecx and ebx to stack so they can be used in drawing
43         mov eax,4
44         mov ebx,1
45         mov ecx,chr
46         mov edx,1
47         int 80h ;draw star
48         pop ebx
49         pop ecx ; pop ecx last, so it is has the correct loop counter value
50     loop li ; end of inner loop
51
52     push ebx ; push ebx to stack, so to use ebx in starting a new line
53     mov eax, 4
54     mov ebx, 1
55     mov ecx, nl
56     mov edx, nl_len
57     int 0x80 ;draws a new line
58
59     pop ebx ;pop inner loop count off stack
60     inc ebx ;increment inner loop count (to draw 1 more triangle next iteration)
61     pop ecx ;pop outler loop count off stack to use as counter for lo
62 loop lo ;end of outer loop
63     int 80h;
64
65 mov eax,1
66 mov ebx,0
67 int 80h;exit
```

Evidence of Right angled triangle



```
bash - "riglerr-univeri x +
riglerr@university-work:~/workspace/207SE-Networks-&-Security/3 - Assembly Code/triangle/t1 (master) $ ./rat
Enter height of triangle: 7
*
**
***
****
*****
*****
*****
riglerr@university-work:~/workspace/207SE-Networks-&-Security/3 - Assembly Code/triangle/t1 (master) $
```

2. Isosceles Triangle

The process to draw this triangle is similar to the previous task, but an additional loop is needed to print the required number of spaces before drawing the asterisks.

1. Get the user input for the height of the triangle and place it in the ecx register.
2. Work out the width of the triangle using the formula $2h - 1$ where h is the height specified by the user, and place this value in a variable called height.
3. Start the outerLoop and push its value to the stack.
4. Calculate the number of spaces that need to be drawn using the formula $(width - noOfAsterisks)/2$. The noOfAsterisks is the current value of the innerLoop (which is always an odd number). And place that value in ecx to be used as the control value for the third loop.
5. When dividing the value of $(width - noOfAsterisks)$ by 2, if the value is zero (on the last line), then do not print any spaces.
6. Draw spaces ecx times.
7. Pop innerLoop value off stack and start innerLoop.
8. Start a new line, add 2 to the value of the innerLoop (The number of stars in each line increases by two each time).
9. Pop the value of the outerLoop off of the stack and start the next iteration.

Listing 2: Isosceles Triangle Code

```
1 section .data
2
3 Prompt db 'Enter height of triangle: '
4 pLen equ $-Prompt
5 chr db '*'
6 nl:      db "  ", 0x0a ; variable to draw a new line
7 nl_len  equ $-nl ; length of new line variable
8 ns: db " "; variable to draw a space
9 ns_l equ $-ns ; length of space variable
10
11 section .bss
12 num resb 2 ; reserve 2 bytes for the input variable
13 width resb 2
14
15 section .text
16 global _start
17 _start:
18
19 ; Ask user for size of triangle
20 mov eax, 4
21 mov ebx, 1
22 mov ecx, Prompt
23 mov edx, pLen
24 int 80h
25
26 ; store the variable
```

```
27 mov eax,3
28 mov ebx,0
29 mov ecx,num
30 mov edx,2
31 int 80h
32
33 mov ecx, [num] ;dereference input and store in ecx
34 sub ecx,'0' ;convert from ascii to decimal
35 xor ch,ch ; clear upper half of ecx
36
37 mov ebx,1
38 mov eax,ecx
39
40 ;this block places w=2n-1 into eax
41 push ebx ;push inner loop to stack
42 mov ebx,2 ; move 2 into ebx
43 mul ebx ; multiply eax by 2
44 sub eax,1 ; subtract 1 from eax
45 pop ebx
46 mov [width],eax ; place width value into width variable
47
48 lo: ;outer loop, amount of lines in triangle
49
50 push ecx ; push outer loop count to stack
51
52 ;This block works out the number of spaces to print before drawing
53 ;noOfSpace = (width-noOfAsterisks)/2
54 mov ecx,[width] ; make loop counter equal to the width of the triangle
55 sub ecx,ebx ;subtract number of asterisks
56 shr ecx,1 ; shift right, divides ecx by 2^1 (2)
57 jz l2 ; jump to the l2 label if the result of the division was 0
58
59 l3:
60 push ebx ; push ebx to stack, so to use ebx in starting a new line
61 push ecx
62 mov eax, 4
63 mov ebx, 1
64 mov ecx, ns
65 mov edx, ns_1
66 int 0x80 ;draws a new line
67 pop ecx ;pop inner loop count off stack
68 pop ebx
69
70 loop l3
71
72 l2:
73
74 mov ecx,ebx ; place inner loop count in the loop counter
75 li: ; inner loop, amount of stars in line
76
77 push ecx
78 push ebx ;push ecx and ebx to stack so they can be used in drawing
79 mov eax,4
```

```

80      mov ebx,1
81      mov ecx,chr
82      mov edx,1
83      int 80h ;draw star
84      pop ebx
85      pop ecx ; pop ecx last, so it is has the correct loop counter value
86      loop li ; end of inner loop
87
88      push ebx ; push ebx to stack, so to use ebx in starting a new line
89      mov eax, 4
90      mov ebx, 1
91      mov ecx, nl
92      mov edx, nl_len
93      int 0x80 ;draws a new line
94
95      pop ebx ;pop inner loop count off stack
96      ; add two to the inner loop counter,
97      ;each line always has an odd number of asterisks
98      add ebx,2
99
100
101      pop ecx ;pop outler loop count off stack to use as counter for lo
102      loop lo ;end of outer loop
103      int 80h;
104
105      mov eax,1
106      mov ebx,0
107      int 80h;exit

```

Evidence of triangle

```
bash - "riglerr-univer x +
riglerr@university-work:~/workspace/207SE-Networks-&-Security/3 - Assembly Code/triangle/t2 (master) $ ./iso
Enter height of triangle: 8
      *
     ***
    *****
   ********
  **********
 **********
 **********
 **********
 **********
riglerr@university-work:~/workspace/207SE-Networks-&-Security/3 - Assembly Code/triangle/t2 (master) $
```

Item 3 - Bootloader

This task was to create a bootloader using Assembly and boot into this bootloader using bochs. The process involved:

1. Writing and compiling the code.
2. Writing the code to the first 512 bytes of a bootable image.
3. use bochs to run the bootloader.

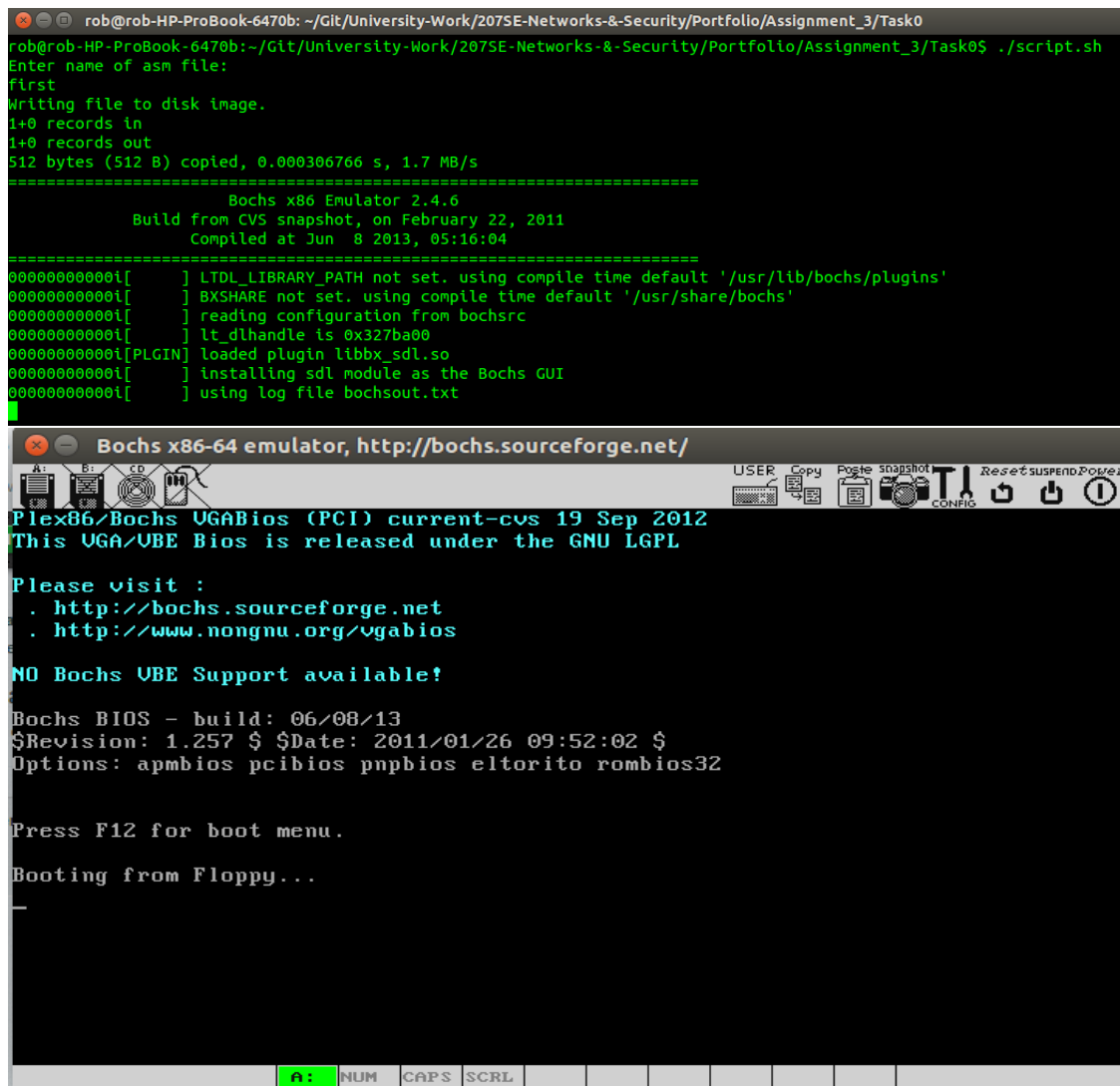
1. Boot pragma-linux using bochs

I created a Bash script to compile, create and simulate the bootloader.

Listing 3: Bash Script

```
1  #!/bin/bash
2
3
4  #User prompt to enter file name.
5  echo "Enter name of asm file: "
6
7  read asm_name  #name stored in variable
8
9  echo "Compiling file."
10
11 #appends the filename with the .asm extension
12 asm_ext="$asm_name".asm
13
14 #Using nasm to compile the file
15 nasm $asm_ext
16
17 echo "Writing file to disk image."
18
19 #Writes the file to the image
20 dd if=$asm_name bs=512 of=a.img
21
22 #starts bochs
23 bochs
```

Evidence of bash script and pragma-Linux



The screenshot shows a terminal window with the following content:

```
rob@rob-HP-ProBook-6470b: ~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_3/Task0
rob@rob-HP-ProBook-6470b:~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_3/Task0$ ./script.sh
Enter name of asm file:
first
Writing file to disk image.
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.000306766 s, 1.7 MB/s
=====
                Bochs x86 Emulator 2.4.6
                Build from CVS snapshot, on February 22, 2011
                Compiled at Jun  8 2013, 05:16:04
=====
0000000000i[      ] LTDL_LIBRARY_PATH not set. using compile time default '/usr/lib/bochs/plugins'
0000000000i[      ] BXSHARE not set. using compile time default '/usr/share/bochs'
0000000000i[      ] reading configuration from bochsrc
0000000000i[      ] lt_dlhandle is 0x327ba00
0000000000i[PLGIN] loaded plugin libbx_sdl.so
0000000000i[      ] installing sdl module as the Bochs GUI
0000000000i[      ] using log file bochsout.txt

```

Below the terminal window is the Bochs x86-64 emulator interface. The title bar reads "Bochs x86-64 emulator, http://bochs.sourceforge.net/". The main window displays the following text:

```
Plex86/Bochs VGABios (PCI) current-cvs 19 Sep 2012
This UGA/VE Bios is released under the GNU LGPL

Please visit :
. http://bochs.sourceforge.net
. http://www.nongnu.org/vgabios

NO Bochs VBE Support available!

Bochs BIOS - build: 06/08/13
$Revision: 1.257 $ $Date: 2011/01/26 09:52:02 $
Options: apmbios pcibios pnpbios eltorito rombios32

Press F12 for boot menu.

Booting from Floppy...

```

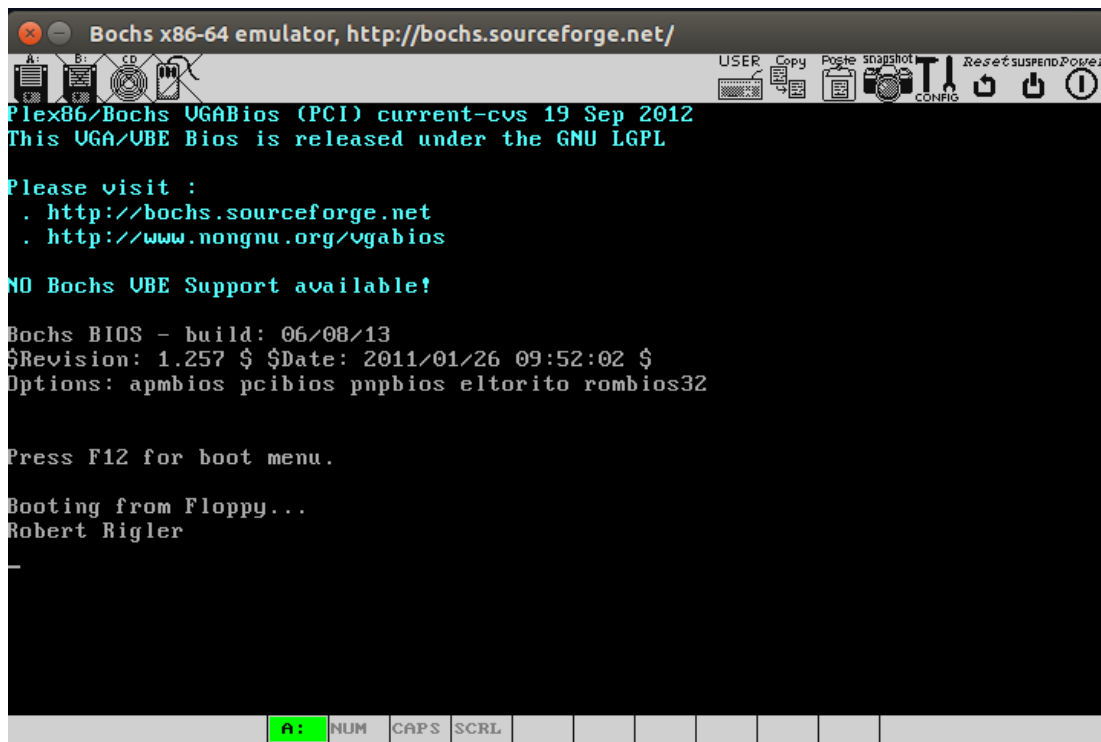
The bottom of the emulator window shows a keyboard layout with a highlighted 'a' key and other keys like NUM, CAPS, SCRL, etc.

1. Make a Bootloader that displays my name

Listing 4: Bootloader Assembly code

```
1  [BITS 16]
2  [ORG 0X7C00]
3  top:
4      mov ax,0x0000
5      mov ds,ax
6      mov si,HelloWorld
7      call writeString
8      jmp $
9
10 writeString:
11     mov ah,0x0E ;Display Character
12     mov bh,0x00
13     mov bl,0x07
14
15 nextchar:
16     lodsb
17     cmp al,0
18     jz done
19     int 0x10
20     jmp nextchar
21
22 done:
23     ret
24     HelloWorld db 'Robert Rigler',13,10,0
25     times 510-($-$$) db 0
26
27 dw 0xAA55
```

Evidence of working code



2. Make a Bootloader that displays a triangle of dots

Item 4 - Inside Proc

1. List the CPU Information using the Cat Command

Command used : *cat /proc/cpuinfo*

```
rob@rob-HP-ProBook-6470b: /proc
rob@rob-HP-ProBook-6470b:/proc$ cat /proc/cpuinfo
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 6
model          : 58
model name     : Intel(R) Core(TM) i5-3340M CPU @ 2.70GHz
stepping       : 9
microcode      : 0x16
cpu MHz        : 1200.000
cache size     : 3072 KB
physical id    : 0
siblings       : 4
core id        : 0
cpu cores      : 2
apicid         : 0
initial apicid : 0
fpu            : yes
fpu_exception  : yes
cpuid level    : 13
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx r
dtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop
_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm
2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer
aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_
shadow vnmi flexpriority ept vpid fsgsbase smep erms
bogomips       : 5387.64
clflush size   : 64
cache_alignment : 64
address sizes  : 36 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id      : GenuineIntel
cpu family     : 6
model          : 58
model name     : Intel(R) Core(TM) i5-3340M CPU @ 2.70GHz
stepping       : 9
microcode      : 0x16
cpu MHz        : 1200.000
```

2. Show a table of the interrupts on the system

Command used : `cat /proc/interrupts`

```
rob@rob-HP-ProBook-6470b: /proc
rob@rob-HP-ProBook-6470b: /proc$ cat /proc/interrupts
```

	CPU0	CPU1	CPU2	CPU3		
0:	17	0	0	0	IO-APIC-edge	timer
1:	127	965	84	114	IO-APIC-edge	i8042
5:	1	0	0	0	IO-APIC-edge	parpor
8:	0	0	0	1	IO-APIC-edge	rtc0
9:	161	666	69	45	IO-APIC-fasteoi	acpi
12:	17919	140293	11353	10149	IO-APIC-edge	i8042
16:	156	160	13	16	IO-APIC-fasteoi	ehci_h
cd:usb1, ehci_hcd:usb2						
18:	2	0	0	0	IO-APIC-fasteoi	firewl
re_ohci, mmc0						
23:	0	0	0	0	IO-APIC-edge	lis3lv
02d						
40:	0	0	0	0	PCI-MSI-edge	PCIe P
ME						
41:	0	0	0	0	PCI-MSI-edge	PCIe P
ME, pciehp						
42:	0	0	0	0	PCI-MSI-edge	PCIe P
ME						
43:	0	0	0	0	PCI-MSI-edge	PCIe P
ME						
44:	0	0	0	0	PCI-MSI-edge	xhci_h
cd						
46:	13082	7835	8087	8869	PCI-MSI-edge	ahci
47:	11	0	1	3	PCI-MSI-edge	mei_me
48:	137	253	45647	54	PCI-MSI-edge	iwlwifi
l						
49:	4755	31559	2541	2296	PCI-MSI-edge	i915
50:	1179	80	17	57	PCI-MSI-edge	snd_hd
a_intel						
NMI:	0	0	0	0	Non-maskable interrupts	
LOC:	45464	43595	48260	38011	Local timer interrupts	
SPU:	0	0	0	0	Spurious interrupts	
PMI:	0	0	0	0	Performance monitoring i	
interrupts						
IWI:	2279	1883	1820	1826	IRQ work interrupts	
RTR:	2	0	0	0	APIC ICR read retries	
RES:	20170	18531	19566	18119	Rescheduling interrupts	
CAL:	494	594	574	529	Function call interrupts	

3. Show number of CPUs, the producer of the CPUs and the CPU Model.

Command used : *grep model /proc/cpuinfo*

```
rob@rob-HP-ProBook-6470b: /proc
rob@rob-HP-ProBook-6470b:/proc$ clear

rob@rob-HP-ProBook-6470b:/proc$ grep model /proc/cpuinfo
model          : 58
model name     : Intel(R) Core(TM) i5-3340M CPU @ 2.70GHz
model          : 58
model name     : Intel(R) Core(TM) i5-3340M CPU @ 2.70GHz
model          : 58
model name     : Intel(R) Core(TM) i5-3340M CPU @ 2.70GHz
model          : 58
model name     : Intel(R) Core(TM) i5-3340M CPU @ 2.70GHz
rob@rob-HP-ProBook-6470b:/proc$ █
```

4. How the parameters that are passed to the kernel when starting up linux.

Command used : *cat /proc/cmdline*

```
rob@rob-HP-ProBook-6470b: /proc
rob@rob-HP-ProBook-6470b:/proc$ clear

rob@rob-HP-ProBook-6470b:/proc$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-3.13.0-39-generic root=UUID=cada5b07-62dd-4282-b91
b-46d583d8b2ab ro quiet splash vt.handoff=7
rob@rob-HP-ProBook-6470b:/proc$ █
```

5. Show the name of the output devices and the number of megabytes read per second during the second sampled interval.

Command used : *awk '{print \$3,\$4}' /proc/diskstats | grep sda*

```
rob@rob-HP-ProBook-6470b: /proc
rob@rob-HP-ProBook-6470b:/proc$ clear

rob@rob-HP-ProBook-6470b:/proc$ awk '{ print $3, $4}' /proc/diskstats | gre
p sda
sda 20786
sda1 166
sda2 2
sda3 162
sda4 164
sda5 14715
sda6 161
sda7 5229
rob@rob-HP-ProBook-6470b:/proc$ █
```

6. Menu based shell script.

Listing 5: Bash Script

```
1 #!/bin/bash
2
3 # DISPLAYS A MENU
4
5 while true;
6 do
7 echo "1. Display information about the CPU. "
8 echo "2. Display the interrupts system. "
9 echo "3. Display a process PID for a process on the system and its status. "
10 echo "4. exit. "
11
12 read input_variable
13 #STORES THE INPUT INTO A VARIABLE CALLED "input_variable"
14
15 echo "Your choice was $input_variable"
16
17 #CASE STATEMENT TO DIFFERENTIATE OUTPUT RESPECTIVE TO THE USER'S CHOICE.
18
19 case "$input_variable" in
20
21 1) #Display CPU info
22    echo Displaying CPU information
23    grep model /proc/cpuinfo
24    ;;
25 2) #Display the interrupts info
26    echo Displaying interrupts
27    cat /proc/interrupts
28    ;;
29 3) #Display the PID and its status.
30    echo Enter PID
31    read input2
32    ps -p "$input2"
33    ;;
34 4) #stop the script
35    break
36    ;;
37 #END CASE STATEMENT
38 esac
39 #END OF WHILE LOOP
40 done
41
42 echo "Exiting"
```

1. Output evidence of option 1:

```

rob@rob-HP-ProBook-6470b: ~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_4
rob@rob-HP-ProBook-6470b:~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_4$ ./script.sh
1. Display information about the CPU.
2. Display the interrupts system.
3. Display a process PID for a process on the system and its status.
4. exit.
1
Your choice was 1
Displaying CPU information
grep: /proc/cpuinf: No such file or directory
1. Display information about the CPU.
2. Display the interrupts system.
3. Display a process PID for a process on the system and its status.
4. exit.

```

2. Output evidence of option 2:

```

rob@rob-HP-ProBook-6470b: ~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_4
3. Display a process PID for a process on the system and its status.
4. exit.
2
Your choice was 2
Displaying interrupts

```

	CPU0	CPU1	CPU2	CPU3	
0:	18	0	0	0	IO-APIC-edge timer
1:	27081	27	5	5	IO-APIC-edge i8042
5:	1	0	0	0	IO-APIC-edge parport0
8:	0	0	0	1	IO-APIC-edge rtc0
9:	9129	1669	5	12	IO-APIC-fasteoi acpi
12:	2903458	100108	11231	10966	IO-APIC-edge i8042
16:	531	177	12	35	IO-APIC-fasteoi ehci_hcd:usb1, ehci_hcd:usb2
18:	2	1	1	0	IO-APIC-fasteoi firewire_ohci, mmc0
23:	1	0	0	0	IO-APIC-edge lis3lv02d
40:	0	0	0	0	PCI-MSI-edge PCie PME
41:	0	0	0	0	PCI-MSI-edge PCie PME, pciehp
42:	0	0	0	0	PCI-MSI-edge PCie PME
43:	0	0	0	0	PCI-MSI-edge PCie PME
44:	0	0	0	0	PCI-MSI-edge xhci_hcd
45:	110611	20107	11326	20391	PCI-MSI-edge ahci
46:	2	27	0	0	PCI-MSI-edge eth0
47:	8	6	0	0	PCI-MSI-edge mei_me
48:	848181	314	897	59	PCI-MSI-edge iwlwifi
49:	50385	1093662	1995	1916	PCI-MSI-edge i915
50:	536	118	0	0	PCI-MSI-edge snd_hda_intel
NMI:	0	0	0	0	Non-maskable interrupts
LOC:	793467	506880	689407	523114	Local timer interrupts
SPU:	0	0	0	0	Spurious interrupts
PMI:	0	0	0	0	Performance monitoring interrupts
IWI:	17017	17224	21695	15698	IRQ work interrupts
RTR:	5	1	0	0	APIC ICR read retries
RES:	129009	121889	115564	111082	Rescheduling interrupts
CAL:	774	633	606	677	Function call interrupts
TLB:	13537	11453	12688	15869	TLB shootdowns
TRM:	0	0	0	0	Thermal event interrupts
THR:	0	0	0	0	Threshold APIC interrupts
MCE:	0	0	0	0	Machine check exceptions
MCP:	93	91	91	91	Machine check polls
ERR:	0				
MIS:	0				

3. Output evidence of option 3:

```
rob@rob-HP-ProBook-6470b: ~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_4
rob@rob-HP-ProBook-6470b:~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_4$ ./script.sh
1. Display information about the CPU.
2. Display the interrupts system.
3. Display a process PID for a process on the system and its status.
4. exit.
3
Your choice was 3
Enter PID
14215
  PID TTY          TIME CMD
14215 ?                00:00:03 gnome-system-mo
1. Display information about the CPU.
2. Display the interrupts system.
3. Display a process PID for a process on the system and its status.
4. exit.
```

4. Output evidence of option 4:

```
rob@rob-HP-ProBook-6470b: ~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_4
rob@rob-HP-ProBook-6470b:~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_4$ ./script.sh
1. Display information about the CPU.
2. Display the interrupts system.
3. Display a process PID for a process on the system and its status.
4. exit.
4
Your choice was 4
Exiting
rob@rob-HP-ProBook-6470b:~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_4$ █
```

Item 5 - Buffer tutorial

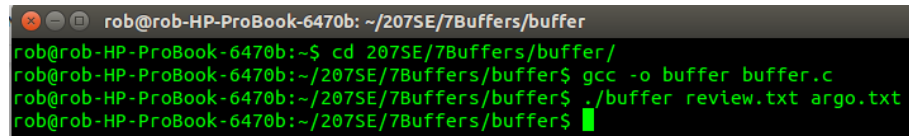
This task involved using buffers, specifically using buffers in term of reading an writing data from a file.

1. Commented version of the provided code

Listing 6: Commented Buffer Code

```
1 #include <fcntl.h>
2 #include <stdlib.h>
3 #include <unistd.h> //Define header files
4 #include <stdio.h>
5
6 #define BUF_SIZE 500 //Define Buffer size as 500.
7 #define OUTPUT_MODE 0700 //Define file permission.
8
9 int main(int argc, char *argv[])
10 {
11     int in_fd, out_fd;
12     int rd_size = 1, wr_size;
13     char buf[BUF_SIZE]; //Declare buffer.
14
15     if (argc != 3)
16         exit(1);
17
18     in_fd = open(argv[1], O_RDONLY); //Open input file.
19     if (in_fd < 0)
20         exit(2);
21
22     out_fd = creat(argv[2], OUTPUT_MODE); //Create output file.
23     if (out_fd < 0)
24         exit(3);
25
26     while (rd_size > 0) {
27
28         rd_size = read(in_fd, buf, BUF_SIZE); // Continuously read from input file
29                                             //into buffer.
30         if (rd_size < 0)
31             exit(4);
32
33         wr_size = write(out_fd, buf, rd_size); // Continuously write from buffer into
34                                             //the output file.
35         if (wr_size <= 0) {
36             close(in_fd); //Close both of the files.
37             close(out_fd);
38             exit(5);
39         }
40     }
41 }
```

2. Evidence of compiled code

A terminal window with a black background and green text. The window title is "rob@rob-HP-ProBook-6470b: ~/207SE/7Buffers/buffer". The terminal shows the following commands and output:

```
rob@rob-HP-ProBook-6470b:~$ cd 207SE/7Buffers/buffer/  
rob@rob-HP-ProBook-6470b:~/207SE/7Buffers/buffer$ gcc -o buffer buffer.c  
rob@rob-HP-ProBook-6470b:~/207SE/7Buffers/buffer$ ./buffer review.txt argo.txt  
rob@rob-HP-ProBook-6470b:~/207SE/7Buffers/buffer$
```

Argo.txt contains the exact same text that was in review.txt

3. Code adaptation to show how many characters were read in total and how many times the buffer was filled

Listing 7: Adpated Code

```
1 #include <fcntl.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <stdio.h>
5 //Define header files
6
7 #define BUF_SIZE 500 //Define Buffer size as 500.
8 #define OUTPUT_MODE 0700 //Define file permission.
9
10 int main(int argc, char *argv[])
11 {
12     int in_fd, out_fd;
13     int buf_count=0,rd_count=0;
14     int rd_size = 1, wr_size;
15     char buf[BUF_SIZE]; //Declare buffer.
16
17     if (argc != 3)
18         exit(1);
19
20     in_fd = open(argv[1], O_RDONLY); //Open input file.
21     if (in_fd < 0)
22         exit(2);
23
24     out_fd = creat(argv[2], OUTPUT_MODE); //Create output file.
25     if (out_fd < 0)
26         exit(3);
27
28     while (rd_size > 0) {
29
30         rd_size = read(in_fd, buf, BUF_SIZE); //Continuously read from input file
31                                             //into buffer.
32         rd_count+= rd_size; //Adds rd_size to the total read count
33         if(rd_size > 0)
34             buf_count +=1; //Counts the number of times the buffer
35                           //is filled (only if rd_size is > 0
36         exit(4);
37
38         wr_size = write(out_fd, buf, rd_size); //Continuously write from buffer into
39                                             //output file.
40         if (wr_size<=0){
41             close(in_fd); //Close input file.
42             close(out_fd); //Close output file
43
44
45             printf("Number of characters read total: %d\n",rd_count );
46             //Prints how many Characters were read.
47         printf("Number of times the buffer was filled: %d\n",buf_count);
48             //Prints how many times the buffer was filled
49     }
```

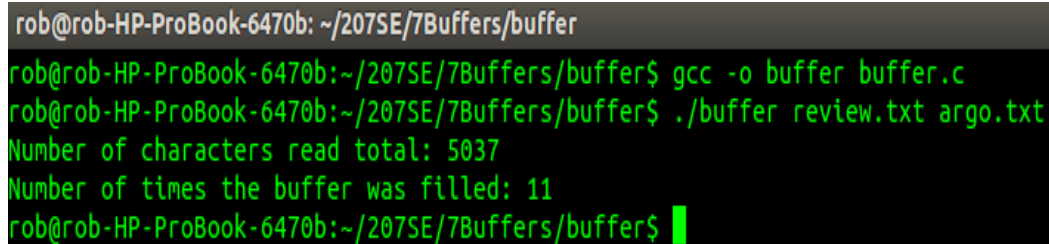
```
50 | exit(5);  
51 | }  
52 | }  
53 | }
```

Firstly I created two variables to hold the Buffer count (*buf_count*) and the character count(*rd_count*) (*line 14*).

Then to accumulate the total numbers of characters read I added the value of *rd_size* to the *rd_count* variable (*line 32*) each time the text was read into the buffer.

To count the number of times the buffer was filled, each time *rd_size* was filled and its value above 0, *buf_count* is incremented by 1(*line 34*).

3a Evidence



```
rob@rob-HP-ProBook-6470b: ~/207SE/7Buffers/buffer  
rob@rob-HP-ProBook-6470b:~/207SE/7Buffers/buffer$ gcc -o buffer buffer.c  
rob@rob-HP-ProBook-6470b:~/207SE/7Buffers/buffer$ ./buffer review.txt argo.txt  
Number of characters read total: 5037  
Number of times the buffer was filled: 11  
rob@rob-HP-ProBook-6470b:~/207SE/7Buffers/buffer$
```

4. Altering the buffer size

- Doubling the buffer size to 1000, the program filled the buffer 6 times. This is half of the original value + 1.
- Doubling the buffer size again to 2000 , the program filled the buffer 3 times which is half of 6.
- Raising the the buffer size to 10000, the program filled the buffer 1 time, indicating that the entire text was placed into the buffer.

There is a direct linear correlation between the buffer size and the amount of times that the buffer was filled.

5. Adapt the code so that it is possible to compare if two files are the same.

Listing 8: Adapted code

```
1 #include <fcntl.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <stdio.h>
5 //Define header files
6
7 #define BUF_SIZE 500 //Define Buffer size as 500.
8 #define OUTPUT_MODE 0700 //Define file permission.
9
10 int main(int argc, char *argv[])
11 {
12     int in_fd, in0_fd; // Create integers to hold file handles.
13     int rd_size = 1; // Create integer to hold the amount of bytes in the buffer.
14     char buf[BUF_SIZE]; //Declare 1st buffer.
15     char buf0[BUF_SIZE]; //Declare 2nd buffer.
16
17     if (argc != 3)
18         exit(1);
19
20
21
22     in_fd = open(argv[1], O_RDONLY); //Open 1st file.
23     if (in_fd < 0)
24         exit(2);
25
26     in0_fd = open(argv[2], O_RDONLY); //Open 2nd file.
27     if (in0_fd < 0)
28         exit(3);
29
30     while (rd_size > 0) {
31
32         int i;
33         rd_size = read(in_fd, buf, BUF_SIZE); // Read From 1st file into 1st buffer
34
35         if (rd_size < 0)
36             exit(4);
37         rd_size = read(in0_fd, buf0, rd_size); //Read from 2nd file into 2nd buffer
38
39         for (i = 0; i < BUF_SIZE; i++){//Loop through the contents of each buffer.
40
41             if (buf[i] == buf0[i])// If buffer contents are equal, go to next buffer element.
42                 continue;
43             else { //If buffer contents are not the same,
44                 //close the files and display a message
45                 //and exit the program.
46
47                 close(in_fd); //Close input file.
48                 close(in0_fd); //Close output file
49                 printf("Files are not the same. \n");
50                 exit(5);
```

```
51  
52         } //end else  
53     } //end for  
54 } //end while  
55  
56 printf("Files are the same \n");// Display this message if the files are the same  
57  
58 }//end main
```

5a. Evidence of comparison between review.txt and argo.txt

```
rob@rob-HP-ProBook-6470b: ~/207SE/7Buffers/buffer  
rob@rob-HP-ProBook-6470b:~/207SE/7Buffers/buffer$ ./bufcomp review.txt argo.txt  
Files are the same  
rob@rob-HP-ProBook-6470b:~/207SE/7Buffers/buffer$ █
```

5b. Evidence of comparison between argo.txt and reviewobserver.txt

```
rob@rob-HP-ProBook-6470b: ~/207SE/7Buffers/buffer  
rob@rob-HP-ProBook-6470b:~/207SE/7Buffers/buffer$ ./bufcomp argo.txt review_observer.txt  
Files are not the same.  
rob@rob-HP-ProBook-6470b:~/207SE/7Buffers/buffer$ █
```

Item 6 - Cache tutorial

1. Complete the cr_read_byte function

Please see the provided code in `cache_reader.c`

1. Evidence of completed working code

```
riglerr@university-work:~/workspace/207SE-Networks-&-Security/8 - Caches/cache-example (master) $ ./cache_example
On campus, they called him Keep-A-Movin' Dan, because of his cowboy vibe and because of his lifestyle, and he somehow grew to take over every conversation I had for the next six months. I pinged his Whuffie a few times, and noticed that it was climbing steadily upward as he accumulated more esteem from the people he met.

I'd pretty much pissed away most of my Whuffie -- all the savings from the symphonies and the first three theses -- drinking myself stupid at the Gazoo, hogging library terminals, pestering profs, until I'd expended all the respect anyone had ever afforded me. All except Dan, who, for some reason, stood me to regular beers and meals and movies.

I got to feeling like I was someone special -- not everyone had a chum as exotic as Keep-A-Movin' Dan, the legendary missionary who visited the only places left that were closed to the Bitchun Society. I can't say for sure why he hung around with me. He mentioned once or twice that he'd liked my symphonies, and he'd read my Ergo nomics thesis on applying theme-park crowd-control techniques in urban settings, and liked what I had to say there. But I think it came down to us having a good time needing each other.

I'd talk to him about the vast carpet of the future unrolling before us, of the certainty that we would encounter alien intelligences some day, of the unimaginable frontiers open to each of us. He'd tell me that deadheading was a strong indicator that one's personal reservoir of introspection and creativity was dry; and that without struggle, there is no real victory.

This was a good fight, one we could have a thousand times without resolving. I'd get him to concede that Whuffie recaptured the true essence of money: in the old days, if you were broke but respected, you wouldn't starve; contrariwise, if you were rich and hated, no sum could buy you security and peace. By measuring the thing that at money really represented -- your personal capital with your friends and neighbors -- you more accurately gauged your success.

And then he'd lead me down a subtle, carefully baited trail that led to my allowing that while, yes, we might someday encounter alien species with wild and fabulous ways, that right now, there was a slightly depressing homogeneity to the world.

On a fine spring day, I defended my thesis to two embodied humans and one prof whose body was out for an overhaul, whose consciousness was present via speakerphone from the computer where it was resting. They all liked it. I collected my sheepskin and went out hunting for Dan in the sweet, flower-stinking streets.

He'd gone. The Anthro major he'd been torturing with his war-stories said that they'd wrapped up that morning, and he'd headed to the walled city of Tijuana, to take his shot with the descendants of a platoon of US Marines who'd settled there and cut themselves off from the Bitchun Society.

So I went to Disney World.

In deference to Dan, I took the flight in realtime, in the minuscule cabin reserved for those of us who stubbornly refused to be frozen and stacked like cordwood for the two hour flight. I was the only one taking the trip in realtime, but a flight attendant dutifully served me a urine-sample-sized orange juice and a rubbery, pun-gent, cheese onelet. I stared out the windows at the infinite clouds while the autopilot banked around the turbulence, and wondered when I'd see Dan next.

(Taken from "Down and Out in the Magic Kingdom" by Cory Doctorow. http://craphound.com/down/)
```

2. Prove the file is being buffered

To prove the code is being buffered. I included `printf("\n");` on line 61 in the `cache_reader.c` file . The program now starts a new line every time it reaches the end of the buffer (in this example 20).

```
riglerr@university-work:~/workspace/207SE-Networks-&-Security/8 - Caches/cache-example (master) $ ./cache_example
On campus, they call
ed him Keep-A-Movin'
  Dan, because of his
  cowboy vibe and bec
ause of his lifestyl
e, and he somehow gr
ew to take over ever
y conversation I had
  for the next six mo
nths. I pinged his W
huffie a few times,
and noticed that it
was climbing steadil
y upward as he accum
ulated more esteem f
rom the people he me
t.

I'd pretty much
pissed away most of
my Whuffie -- all th
e savings from the s
ymphonies and the fi
rst three theses --
drinking myself stup
id at the Gazoo, hog
ging library termina
ls, pestering profs,
until I'd expended
all the respect anyo
ne had ever afforded
me. All except Dan,
who, for some reaso
n, stood me to regul
```

3. Provide some statistics

To count the number of bytes read, I created a variable called *byte_tot* in the *cr_file* structure (line 13) in the *cache_reader.h* file. This variable is used in the *Refill()* method (line 8)(*cache_reader file*). Every time the *Refill()* method is called, it adds the value of *len* (which contains the number of bytes currently being read) to itself.

The amount of times the buffer was refilled, was calculated by dividing the number of bytes read from the text by the size of the buffer.

1. Evidence of completed working code, showing the statistics: Number of bytes read and Total times the buffer was refilled.

```
In deference to Dan, I took the flight in realtime, in the minuscule cabin reserved for those of
the two hour flight. I was the only one taking the trip in realtime, but a flight attendant du
gent, cheese omelet. I stared out the windows at the infinite clouds while the autopilot banked

(Taken from "Down and Out in the Magic Kingdom" by Cory Doctorow. http://craphound.com/down/)

Byte Count: 3487
Refill Count: 174
```

Listing 9: cache_example.c

```
1 #include "cache_reader.h"
2
3 //Simple file display to show how easy it is to use the cached reader functions
4
5 int main(){
6     char c;
7     int refill_count=0;
8     int byte_count=0;
9     //Open a file
10    cr_file* f = cr_open("text",20);
11
12    //While there are useful bytes coming from it
13    while((c=cr_read_byte(f))!=EOF){
14        //Print them
15        printf("%c",c);
16    }
17
18
19
20    //Then close the file
21
22    // Displaying the total number of bytes read.
23    printf("\nByte Count: %d",f->byte_tot);
24
25    //Displaying the total number of times the buffer was filled.
26    printf("\nRefill Count: %d\n",f->byte_tot/f->bufferlength);
27    (No_of_bytes / buffersize).
28    cr_close(f);
29
30    //And finish
31    return 0;
32 }
```


Listing 10: cache_reader.h

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 //The internals of this struct aren't important
5 //from the user's point of view
6 typedef struct{
7     FILE* file;           //File being read
8     int bufferlength;     //Fixed buffer length
9     int usedbuffer;       //Current point in the buffer
10    char* buffer;          //A pointer to a piece of memory
11                           // same length as "bufferlength"
12    //Integer to store the total amount of bytes that were read from the file.
13    int byte_tot;
14 } cr_file;
15
16
17
18
19 //Open a file with a given size of buffer to cache with
20 cr_file* cr_open(char* filename, int buffersize);
21
22
23 //Close an open file
24 void cr_close(cr_file* f);
25
26 //Read a byte. Will return EOF if empty.
27 char cr_read_byte(cr_file* f);
28
29
30
31 //-----
32
33 //Refill an empty buffer. Not intended for users
34 int refill(cr_file* buff);
```

Listing 11: cache_reader.c

```
1 #include "cache_reader.h"
2
3
4 //http://www.phim.unibe.ch/comp_doc/c_manual/C/SYNTAX/struct.html
5 //http://vergil.chemistry.gatech.edu/resources/programming/c-tutorial/structs.html
6
7
8 int refill(cr_file* buff){
9     //Refills a buffer
10    //Only works when completely used buffer
11    if(buff->usedbuffer!=buff->bufferlength)
12        return 0;
13    else{
14        buff->usedbuffer=0;
15        int len=fread(buff->buffer, sizeof(char), buff->bufferlength, buff->file);
16        //If we didn't fill the buffer, fill up with EOF
17        if(len<buff->bufferlength)
18            for(int i=len;i<buff->bufferlength;i++)
19                buff->buffer[i]=EOF; //Accessing like an array!
20        buff->byte_tot +=len; //Adding len to the byte total.
21        return len;
22    }
23
24 }
25
26 void cr_close(cr_file* f){
27     free(f->buffer); //clear buffer
28     fclose(f->file); //close the file
29 }
30
31
32 cr_file* cr_open(char * filename, int buffersize){
33
34     //Info on malloc
35     //http://www.space.unibe.ch/comp_doc/c_manual/C/FUNCTIONS/malloc.html
36     FILE* f;
37     if ((f = fopen(filename, "r")) == NULL){
38         fprintf(stderr, "Cannot open %s\n", filename);
39         return 0;
40     }
41
42     cr_file* a=(cr_file*)malloc(sizeof(cr_file));
43     a->file=f;
44     a->bufferlength=buffersize;
45     //Start off with no characters, so refill will work as expected
46     a->usedbuffer=buffersize;
47     a->buffer=(char*)malloc(sizeof(char)*buffersize);
48     a->byte_tot =0;
49     refill(a);
50     return a;
51 }
52
```

```
53
54
55
56 //-----
57 char cr_read_byte(cr_file* f){
58
59     char btoRet; // byte to hold the character to return.
60     if (f->usedbuffer >= f->bufferlength){ // if the buffer is all used, refill()
61         printf(" \n "); // starts a new line very time the buffer needs to be refilled.
62         refill(f);
63
64     }
65
66     else{
67         // If buffer hasn't been fully used,
68         //return the character and increase the usedBuffer position by 1.
69
70         //Place next character in the btoRet variable.
71         btoRet = f->buffer[f->usedbuffer];
72         f->usedbuffer +=1; //Move the buffer position up by 1.
73         return btoRet; //return the varibale.
74
75     }
76
77 }
```

Item 7 - Kernel

1. Description of the commands for loading and unloading Linux kernel modules.

- `lsmod` - Lists all of the available loaded kernel modules.
- `modinfo` - Displays information about a particular kernel module.
- `insmod` - Install and load a kernel module
- `rmmod` - Unload a module

2. List of the loaded modules

This list was generated by using the `lsmod` command

```
rob@rob-HP-ProBook-6470b: ~
rob@rob-HP-ProBook-6470b:~$ lsmod
Module                  Size  Used by
ctr                     13049  1
ccm                     17773  1
nvram                   14411  0
btusb                   32412  0
uvcvideo                80885  0
videobuf2_vmalloc      13216  1 uvcvideo
videobuf2_memops       13362  1 videobuf2_vmalloc
videobuf2_core         40664  1 uvcvideo
videodev               134688  2 uvcvideo,videobuf2_core
snd_hda_codec_hdmi     46368  1
snd_hda_codec_idt      54762  1
hp_wmi                  14062  0
sparse_keymap          13948  1 hp_wmi
intel_rapl              18773  0
x86_pkg_temp_thermal   14205  0
intel_powerclamp       14705  0
coretemp               13435  0
kvm                     455835  0
crct10dif_pclmul       14289  0
crc32_pclmul           13113  0
ghash_clmulni_intel    13216  0
arc4                    12608  2
aesni_intel            55624  2
aes_x86_64             17131  1 aesni_intel
lrw                     13286  1 aesni_intel
gf128mul               14951  1 lrw
glue_helper            13990  1 aesni_intel
ablk_helper            13597  1 aesni_intel
cryptd                 20359  3 ghash_clmulni_intel,aesni_intel,ablk_helper
iwlvm                  232285  0
mac80211               630653  1 iwlvm
joydev                  17381  0
serio_raw              13462  0
iwlwifi                169932  1 iwlvm
rfcomm                  69160  8
cfg80211               484040  3 iwlwifi,mac80211,iwlvm
snd_hda_intel          56451  3
snd_hda_codec          192906  3 snd_hda_codec_hdmi,snd_hda_codec_idt,snd_hda_intel
lpc_ich                 21080  0
```

```

rob@rob-HP-ProBook-6470b: ~
snd_page_alloc      18710  2  snd_pcm,snd_hda_intel
snd_seq_midi        13324  0
i915                784207  4
snd_seq_midi_event  14899  1  snd_seq_midi
wmi                 19177  1  hp_wmi
snd_rawmidi         30144  1  snd_seq_midi
snd_seq             61560  2  snd_seq_midi_event,snd_seq_midi
drm_kms_helper      55071  1  i915
drm                 303102  5  i915,drm_kms_helper
snd_seq_device      14497  3  snd_seq,snd_rawmidi,snd_seq_midi
i2c_algo_bit       13413  1  i915
tpm_infineon        17372  0
snd_timer           29482  2  snd_pcm,snd_seq
snd                 69322  17  snd_hwdep,snd_timer,snd_hda_codec_hdmi
eq_device,snd_seq_midi
video               19476  1  i915
hp_accel            26012  0
lis3lv02d           20156  1  hp_accel
soundcore           12680  1  snd
mei_me              18627  0
hp_wireless         12637  0
input_polldev       13896  1  lis3lv02d
mei                 82276  1  mei_me
mac_hid             13205  0
binfmt_misc         17468  1
parport_pc          32701  1
ppdev               17671  0
lp                  17759  0
parport             42348  3  lp,ppdev,parport_pc
psmouse             106714  0
firewire_ohci       40409  0
ahci                 25819  3
e1000e              254433  0
sdhci_pci           23172  0
libahci             32716  1  ahci
sdhci               43015  1  sdhci_pci
firewire_core       68769  1  firewire_ohci
ptp                 18933  1  e1000e
crc_itu_t           12707  1  firewire_core
pps_core            19382  1  ptp
rob@rob-HP-ProBook-6470b:~$ git pull

```

3. Description of four loaded modules

My selected modules are ccm, nvram, btusb, uvcvideo

- **ccm**

The 'ccm' module is the Clock Control Module, it controls the hardware clocks on the motherboard which consists of two crystal oscillators and a control chip. The 'ccm' ensures that the proper time is kept, so that any devices connected to the clock are using the correct time.

- **nvram**

The 'nvram' module is the Non-Volatile memory module. This driver allows the user to access the contents of the RAM in real time.

- **btusb**

The 'btusb' module is a generic Bluetooth USB driver. This driver controls the USB Bluetooth devices connected to the computer.

- **uvcvideo**

The 'uvcvideo' module is a UVC (USB Video Class) driver which controls webcams that are compliant to the UVC specification. It makes sure that the device is compatible with the video streaming functionality on the Universal-Serial-Bus.

Item 8 - Kernel part II

This task involves changing the code in `cache_reader.h` and `cache_reader.c` to use the kernel system calls `open()`, `read()`, and `close()` instead of the `c` alternatives `fopen()`, `fread()`, and `fclose()`.

Changing the code

The changes I made are:

1. In `cache_reader.h`, in the `cr_file` structure (*line 7*), I changed the type of *file* from:
`FILE*`
To:
`int`
This is because the system calls use integer file descriptors instead of a `FILE` type.
2. The line 16 of the `cache_reader.c` file has been changed from:
`int len=fread(buff->buffer, sizeof(char), buff->bufferlength, buff->file);`
To:
`int len=read(buff->file,buff->buffer, buff->bufferlength);`
3. The line 30 of the `cache_reader.c` file has been changed from:
`fclose(f->file);`
To:
`close(f->file);`
4. The line 38 of the `cache_reader.c` file has been changed from:
`if ((f = fopen(filename, "r")) == NULL)`
To:
`int f;`
`f = open(filename, O_RDONLY);`
`if (f <0)`

Listing 12: `cache_reader.c`

```
1 #include "cache_reader.h"
2 #include <stdlib.h>
3 #include <fcntl.h>
4 #include <unistd.h>
5
6 //http://www.phim.unibe.ch/comp_doc/c_manual/C/SYNTAX/struct.html
7 //http://vergil.chemistry.gatech.edu/resources/programming/c-tutorial/structs.html
8
9 int refill(cr_file* buff){
10     //Refills a buffer
11     //Only works when completely used buffer
12     if(buff->usedbuffer!=buff->bufferlength)
13         return 0;
14     else{
15         buff->usedbuffer=0;
16         int len=read(buff->file,buff->buffer, buff->bufferlength);
17         int i;
18         //If we didn't fill the buffer, fill up with EOF
19         if(len<buff->bufferlength)
20             for(i=len;i<buff->bufferlength;i++)
```

```

21         buff->buffer[i]=EOF; //Accessing like an array!
22         buff->byte_tot +=len; //Adding len to the byte total.
23         return len;
24     }
25
26 }
27
28 void cr_close(cr_file* f){
29     free(f->buffer);
30     close(f-> file);
31 }
32 //-----
33 cr_file* cr_open(char * filename, int buffersize){
34
35     //Info on malloc
36     //http://www.space.unibe.ch/comp_doc/c_manual/C/FUNCTIONS/malloc.html
37     int f;
38     f = open(filename, O_RDONLY);
39     if ( f < 0){
40         fprintf(stderr, "Cannot open %s\n", filename);
41         return 0;
42     }
43
44     cr_file* a=(cr_file*)malloc(sizeof(cr_file));
45     a->file=f;
46     a->bufferlength=buffersize;
47     //Start off with no characters, so refill will work as expected
48     a->usedbuffer=buffersize;
49     a->buffer=(char*)malloc(sizeof(char)*buffersize);
50     a->byte_tot =0;
51     refill(a);
52     return a;
53 }
54 //-----
55 char cr_read_byte(cr_file* f){
56
57     char btoRet; // byte to hold the character to return.
58     // if the buffer is all used, refill()
59     if (f->usedbuffer >= f->bufferlength){
60
61         // starts a new line very time the buffer needs to be refilled.
62         // printf(" \n ");
63         refill(f);
64     }
65     // If buffer hasn't been fully used, return the chracater
66     // and increase the usedBuffer position by 1.
67     else{
68         //Place next character in the btoRet variable.
69         btoRet = f->buffer[f->usedbuffer];
70
71         f->usedbuffer +=1; //Move the buffer position up by 1.
72         return btoRet; //return the varibale.
73     }

```



```
74     }  
75  
76 }
```

Listing 13: cache_reader.h

```
1  #include <stdio.h>  
2  #include <stdlib.h>  
3  
4  //The internals of this struct aren't important  
5  //from the user's point of view  
6  typedef struct{  
7      int file;          //File being read  
8      int bufferlength; //Fixed buffer length  
9      int usedbuffer;    //Current point in the buffer  
10     char* buffer;      //A pointer to a piece of memory  
11                        // same length as "bufferlength"  
12     //Integer to store the total amount of bytes that were read from the file.  
13     int byte_tot;  
14 } cr_file;  
15  
16  
17  
18  
19 //Open a file with a given size of buffer to cache with  
20 cr_file* cr_open(char* filename, int buffersize);  
21  
22  
23 //Close an open file  
24 void cr_close(cr_file* f);  
25  
26 //Read a byte. Will return EOF if empty.  
27 char cr_read_byte(cr_file* f);  
28  
29  
30  
31 //-----  
32  
33 //Refill an empty buffer. Not intended for users  
34 int refill(cr_file* buff);
```

Evidence of the adapted code:

```

rob@rob-HP-ProBook-6470b: ~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_7/Cache
rob@rob-HP-ProBook-6470b:~/Git/University-Work/207SE-Networks-&-Security/Portfolio/Assignment_7/Cache$ ./cache_example
On campus, they call'd him Keep-A-Movin' Dan, because of his cowboy vibe and bec'ause of his lifestyle, and he somehow gr'ew to take over ever' co
nversation I had'ed for the next six months. I pinged his Whuffle a few times, 'and noticed that it 'was climbing steadily upward as he accum'ulated m
ore esteem f'rom the people he me't.

I'd pretty much 'missed away most of 'my Whuffle -- all th'e savings from the s'mphonies and the fl'ist three theses -- 'rinking myself stup'id at th
e Gazoo, hogg'ing library termina's, pestering profs,'until I'd expended 'til the respect any'o'e had ever afforded'me. All except Dan,'who, for so
me reaso'n, stood me to regul'r beers and meals a'nd movies.

I got to' feeling like I was 'someone special -- n'ot everyone had a ch'nn as exotic as Keep'A-Movin' Dan, the l'egendary missionary 'who visited the
only' places left that we'e closed to the Bit'chun Society. I can't say for sure why h'e hung around with m'e. He mentioned once'or twice that he'd
'liked my symphonies,' and he'd read my E'r'onomics thesis on a'pplying theme-park c't'owd-control techniq'ues in urban setting', and liked what I 'ad
to say there. Bu't I think it came do'wn to us having a go'd time needling eac'h other.

I'd talk t'o him about the vast' carpet of the futur'e unrolling before u's, of the certainty 'that we would encoun'er alien intelligen'es some day,
of the' unimaginable fronti'ers open to each of 'us. He'd tell me tha't deadheading was a 'strong indicator tha't one's personal res'ervoir of introspe
ct'ion and creativity w'as dry; and that wit'out struggle, there'is no real victory.'

This was a good fl'ght, one we could ha'be a thousand times 'without resolving. I'd get him to conced' that Whuffle recap'ured the true essen'e of
money: in the 'ld days, if you wer'e broke but respect'e, you wouldn't star'e; contrariwise, if'you were rich and h'ated, no sum could b'uy you sec
urity and 'peace. By measuring 'the thing that money'really represented -- your personal cap'tal with your frien's and neighbors -- 'you more accur
ately 'gauged your success.'

And then he'd lead'me down a subtle, c'refully baited trai' that led to my all'wing that while, ye's, we might someday 'encounter alien spec'es wi
th wild and fa'bulous ways, that ri'ght now, there was a'slightly depressing' homogeneity to the 'world.

On a fine sp'ing day, I defended'my thesis to two em'bodied humans and on' prof whose body wa' out for an overhau', whose consciousness was prese
nt via s'peakerphone from the' computer where it w'as resting. They all'liked it. I collect'ed my sheepskin and 'went out hunting for'Dan in the swee
t, f'lower-stinking street's.

He'd gone. The 'nthro major he'd be'en torturing with hi' war-stories said t'hat they'd wrapped u' that morning, and 'e'd headed to the w'alled cit
y of T'juan', to take his shot 'with the descendants'of a platoon of US 'Marines who'd settle'd there and cut them'selves off from the 'Bitchun Socie
ty.

So'I went to Disney Wo'ld.

In deference t'o Dan, I took the fl'ght in realtime, in'the minuscule cabin'reserved for those 'f us who stubbornly'refused to be froze' and stac
ked like c'ardwood for the two 'our flight. I was t'he only one taking t'he trip in realtime,' but a flight attend'ant dutifully served'me a urine-sa
mple-s'ized orange juice an' a rubbery, pungent' cheese omelet. I s'ared out the window' at the infinite cl'ouds while the autop'ilot banked around
t'he turbulence, and w'ondered when I'd see'Dan next.

```

The output shows the contents of the file "text.txt". You may notice the strange character that appears every time the buffer is refilled.