Module: 207SE Operating Systems, Security and Networks

Portfolio 1

Name: James Thomas

Student ID: 9195071

Table of Contents:

Lab 1

Basic Task)

Two of the main functions of an operating system are providing a User Interface and security. The operating system can provide the user with either a text or a graphical user interface (University of Helsinki, 2019). A graphical interface is achieved through using windows and menus enabling users to navigate and perform operations with only the mouse and display. A text user interface is achieved by providing the user with an interface where a single line of commands can be written, the machine will then execute this command successfully or produce an error to aid in debugging. In the future as virtual reality and augmented reality advance and become more mainstream, we could see virtual interfaces come into use for things such as virtual musical instruments The operating system will provide ongoing security for the machine by downloading and installing security patches and system updates automatically. A lot of modern-day operating systems come with a built-in anti-virus to prevent users unwillingly infecting their machine with malicious code.

Advanced Task 2 - Python)

```
def Generator(Name, ID):

    character_list = [] #List of character in name
    Stud_len = len(ID) #Length of student ID

    counter = 1 #Set counter to 1 to avoid out of range error
    while counter <= Stud_len: #Iterate over the student ID, set column
number according to brief
        digit = int(ID[Stud_len-counter])
        if digit >= 2 and digit <= 7:
            columns = digit
            break
        else:
            counter = counter + 1
            columns = 4 #Set columns to 4 if no acceptable int found

    for i in range(0, len(Name)):
        character_list.append(Name[i]) #Take entered string and split to
list


    total_length = len(character_list) #Get length of character list

    while total_length % columns != 0: #While remainder of length/columns
is not zero add plusses to end of character list
        character_list.append("+")
        total_length = len(character_list)

    counter = 0
    while counter < total_length: #iterate over the character list, if
asterisk or plus, dont change
        for i in range(columns):
            if character_list[counter] == "*" or character_list[counter] ==
```

```
"+":
                character_list.append(character_list[counter])
                counter = counter + 1
            else: #Get ascii value of char, add current column number and
get new char, then append to the end of character list
                character = (character_list[counter])
                position = ord(character)-96
                column = i + 1
                asci = position+column+96
                if asci > 122: #If new ascii value is greater than ascii
value of z, loop to start of alphabet
                    asci = asci - 26
                new_letter = chr(asci)
                character_list.append(new_letter)
                counter = counter + 1

    total_length = len(character_list) #update length of character list
    counter = 0
    while counter < total_length: #Iterate over every item in character
list
        row_string = ""
        for i in range(columns): #Row length is the same as no. of columns
            row_string += character_list[counter] #Add chars to row sting
            counter = counter + 1
        print(row_string) #Print row string after it is the same length as
no. of columns

if __name__ == "__main__": #Get full name and student ID, exit if user
exits
    print("Type exit to exit")
    while True:
        full_name = input("Enter Full Name: ").replace(" ", "*")
        if full_name == "exit":
            exit()
        Stud_ID = input("Enter Student ID: ")
        if Stud_ID == "exit":
            exit()
        Generator(full_name, Stud_ID) #Call generator for grid output
```

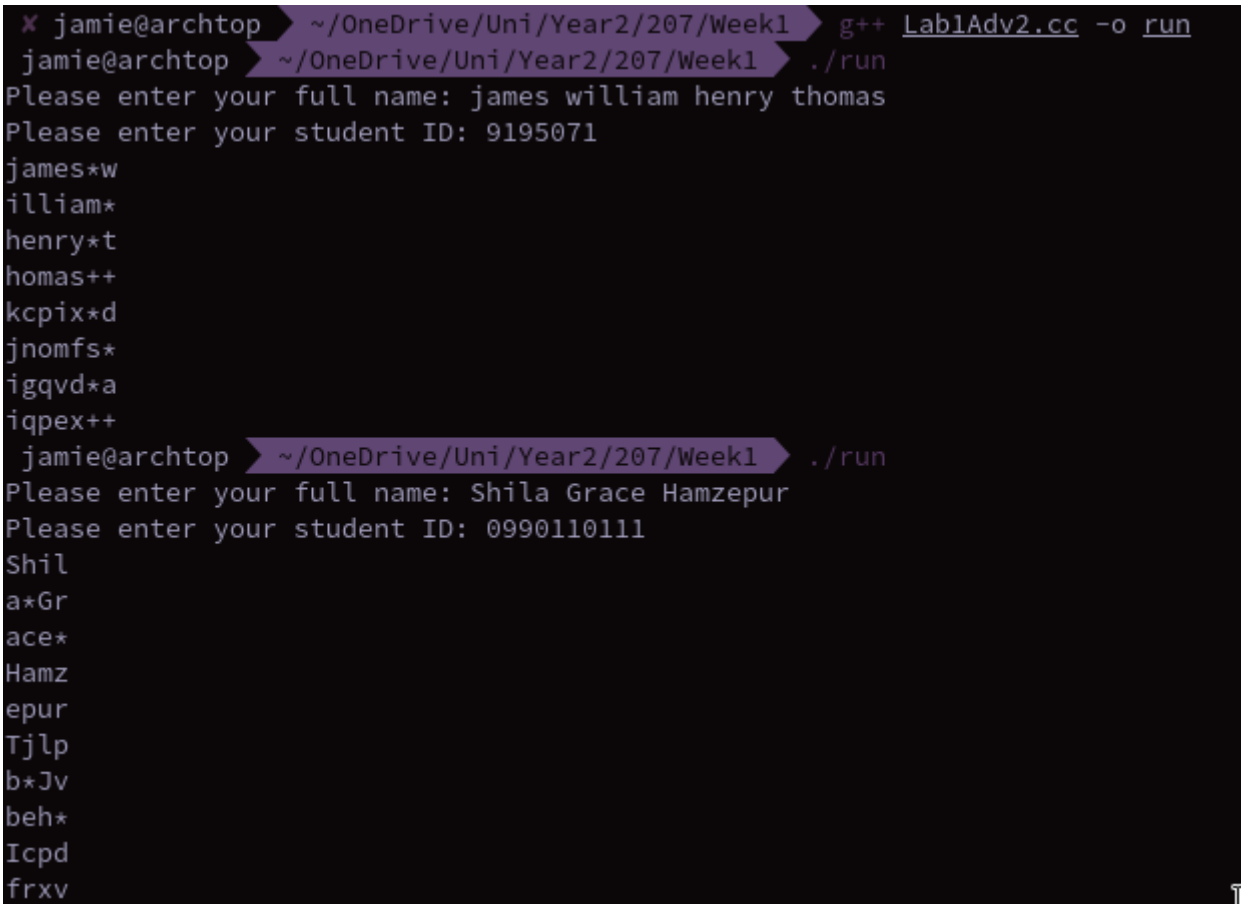
```
jamie@archtop ~/OneDrive/Uni/Year2/207/Week1 python Lab1Adv2.py
Type exit to exit
Enter Full Name: James William Henry Thomas
Enter Student ID: 9195071
James*W
illiam*
Henry*T
homas++
Kcpix*^
jnomfs*
Igqv~*[
iqpex++
Enter Full Name: Shila Grace Hamzepur
Enter Student ID: 0990110111
Shil
a*Gr
ace*
Hamz
epur
Tjlp
b*Jv
beh*
Icp~
frxv
Enter Full Name: exit
```

Advanced Task 2 - C++

```cpp
#include <fcntl.h>
#include <string>
#include <algorithm>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <iostream>

using namespace std;

int digit; //Single digits in the student ID
int columns; //Number of columns in the final output
int current_column; //Tracking where we are when deciding encrypted potion
of output
char x; //Storing current character ascii value
string temp_char = ""; //Empty string to hold current characters
string row_string; //Holds string for row output
string name; //Inputs
string student_ID;

int main(){

    cout << "Please enter your full name: ";
    getline(cin, name); //Take full name from user input
```

```
    cout << "Please enter your student ID: ";
    getline(cin, student_ID); //Take student ID from user input

    int id_length = student_ID.length(); //Get length of student ID so we
can interate over it
    int counter = 0; //Start counter at 1 so we can it
    replace(name.begin(), name.end(), ' ', '*'); //Replace whitespace with
asterisks
    while(counter <= id_length) //Iterate over the length of the student ID
    {
        char current_char = student_ID[id_length-counter]; //Grab the
current character in the name string
        digit = current_char - '0'; //Conv current character to an int
        if(digit >= 2 && digit <=7) //If in acceptable range set no. of
columns to the current character
        {
            columns = digit;
            break;
        }
        else //Inc the counter if not in acceptable range
        {
            counter++;
        }
        columns = 4; //If no numbers found from 2-7, set to 4
    }

    int name_length = name.length(); //Get length of name

    while(name_length%columns !=0) //Keep adding plusses to the end of the
string until the remainder
                                   //of the devision with the number of
columns is zero, so the output is a complete grid
    {
        name.append("+");
        name_length = name.length(); //Update name length when a plus is
added
    }

    counter = 0;
    current_column = 0;
    while(counter<name_length) //Iterate over name string to generate
encrypted string and append it to the end of name
    {
        for(int i=0; i<columns; i++) //run this loop for each column to use
the column number for encryption
        {
            if(name[counter] == '*' || name[counter] == '+') //Dont change
the character if it is a plus of asterisks
            {
                name = name  + name[counter];
                counter++;
            }
            else
            {
```

```
                temp_char = name[counter]; //Store current character
                x = temp_char.at(0); //Get ascii value of current character
                current_column = i + 1; //Get current column
                int asci = int(x) + current_column; // Generate new ascii
value of encrypted character
                if(asci > 122) //If new character has an ascii value
greater than the ascii value of 'z', loop round to the beginning of
alphabet
                {
                    asci = asci - 26;
                }
                temp_char = char(asci); //Conv ascii to char
                name.append(temp_char); //Append char to end of string
                counter++;
            }
        }
    }

    name_length = name.length(); //Update length of name string after
encryption appended to it
    counter = 0; //Reset counter
    while(counter < name_length) //Iterate over the name string
    {
        row_string = ""; //Reset row string after last one has been printed
        for(int i = 0; i < columns; i++) //take column number and take that
many chars, add to row string
        {
            temp_char = name[counter];
            row_string.append(temp_char);
            counter++;
        }
        cout << row_string << endl; //Print row string with a newline
    }
}
```



Lab 2

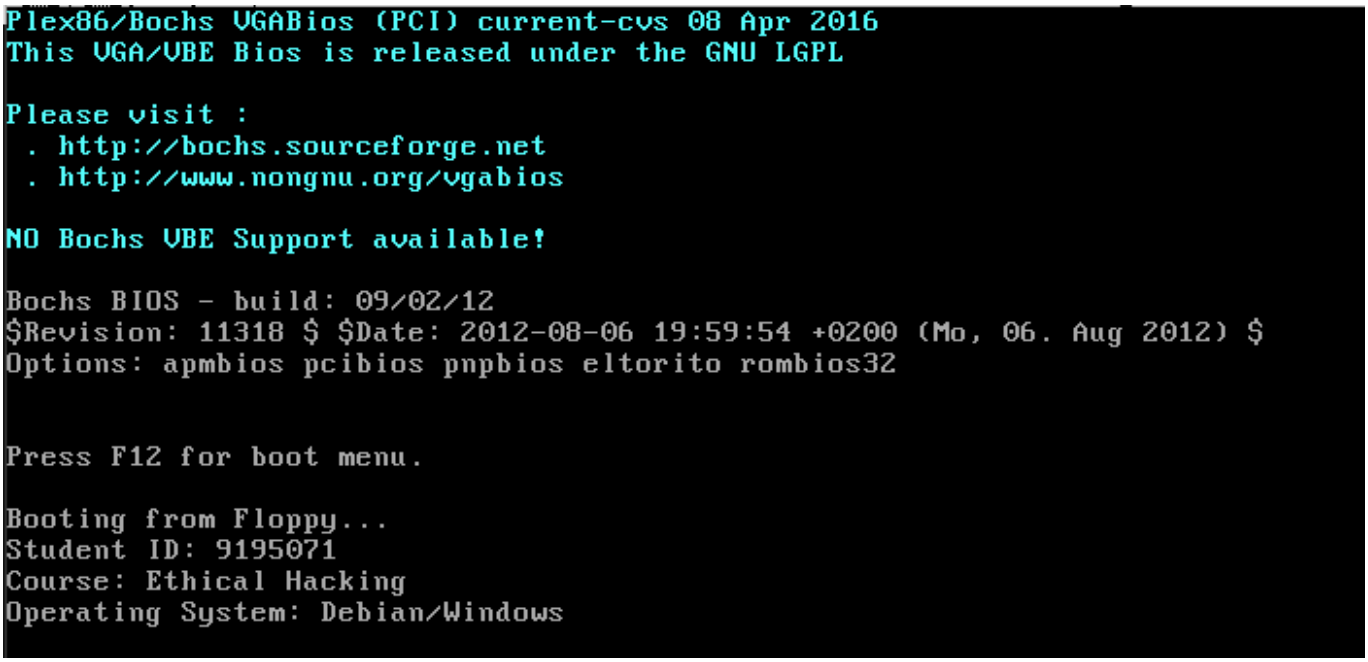
```
File Manipulation
a) mkdir ~/207SE/JT207SE2021/ && chmod 0751 ~/207SE/JT207SE2021
b) wget -O ~/207SE/JT207SE2021/tree.sh
http://www.centerkey.com/tree/tree.sh && chmod 777
~/207SE/JT207SE2021/tree.sh
c) mkdir /Portfolio1-2021 /Portfolio2-2021
d) mkdir Portfolio1-2021/P1-Lab{0..10} Portfolio2-2021/P2-Lab{11..20}
e) touch Programming.txt && mv Programming.txt ./Portfolio1-2021/P1-Lab0
f) ./tree.sh

Mixed Linux Commands
a) date && date -d '-10 year' && ncal 12 1999 && ncal -w -j
b) resolveip os-5004cem.coventry.ac.uk
c) whoami && users && w
d) ps aux && top
e) mesg -v n

Document Manipulation
a) touch Poem.txt && vim Poem.txt
b) cksum Poem.txt
c)
  i ) grep -Eviw 'he|she$' Poem.txt
  ii) grep -Eiw '(the.*also|also.*the)|(an.*also|also.*an)$' Poem.txt
d) grep -i -B 3 king Poem.txt
e) grep -Eiw 'the|an' Poem.txt | grep -Eicv 'the.*an|an.*then' && grep -
```

```
Eiwn 'the.*an|an.*then' Poem.txt
f) split -d --additional-suffix=.txt -l 4 -a 1 Poem.txt Poem
g) sort Poem.txt | rev >> Poem2.txt
h) alias SortRev="sort Poem.txt | rev >> Poem2.txt"
```

Lab 4

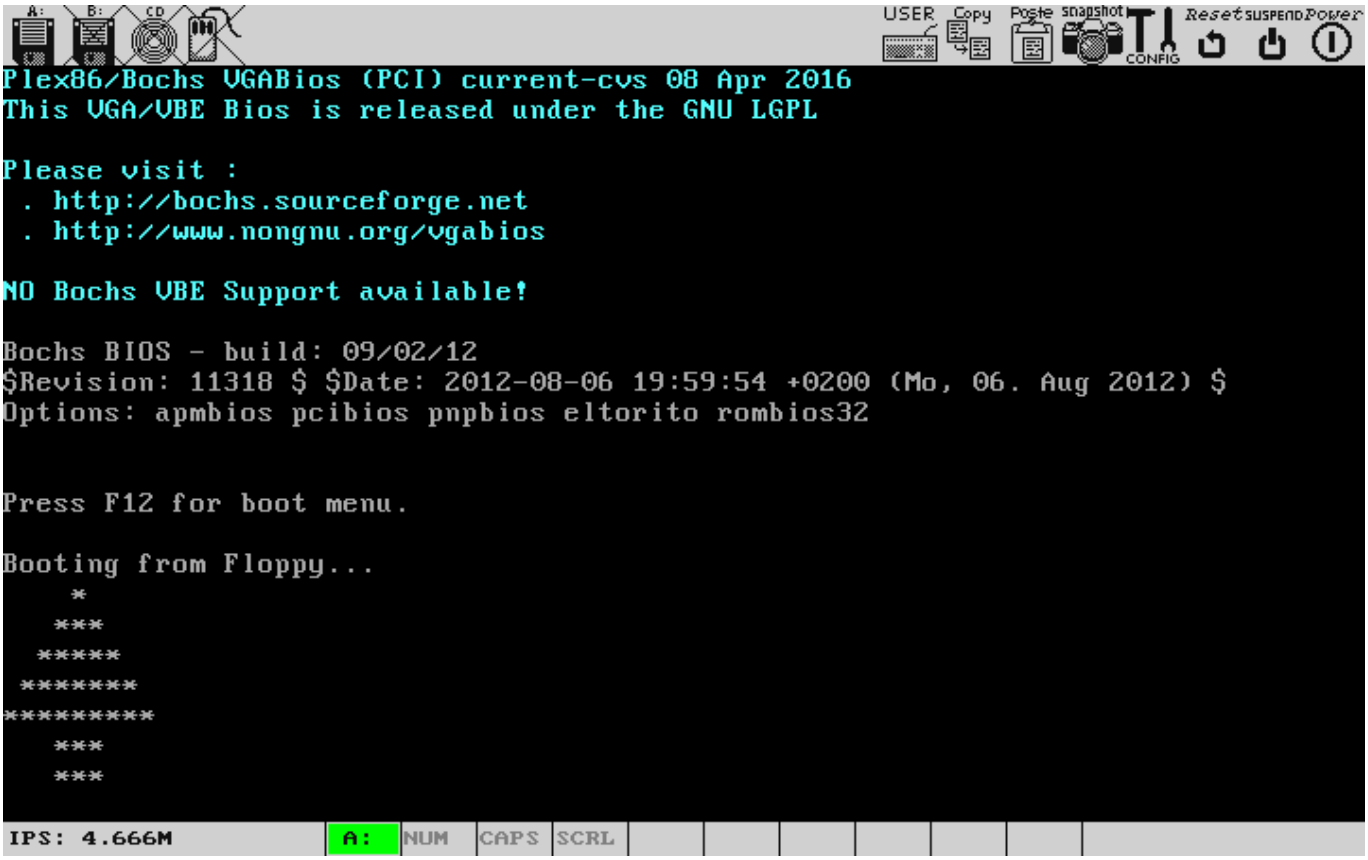
Basic Task a)

```
[BITS 16]
[ORG 0x7C00]
top:
        ;; Put 0 into ds (data segment)
        ;; Can't do it directly
        mov ax,0x0000
        mov ds,ax
        ;; si is the location relative to the data segment of the
        ;; string/char to display
        mov si, ID
        call writeString ; See below
        mov si, Course
        call writeString
        mov si, OS
        call writeString
        jmp $ ; Spin
writeString:
        mov ah,0x0E ; Display a chacter (as before)
        mov bh,0x00
        mov bl,0x07
nextchar:
        Lodsb ; Loads [SI] into AL and increases SI by one
        ;; Effectively "pumps" the string through AL
        cmp al,0 ; End of the string?
        jz done
        int 0x10 ; BIOS interrupt
        jmp nextchar
done:
        ret
        ID db 'Student ID: 9195071',13,10,0 ; Hardcoded student ID, course
and OS also hardcoded below.
        Course db 'Course: Ethical Hacking',13,10,0
        OS db 'Operating System: debian/windows',13,10,0
        times 510-($-$$) db 0
        dw 0xAA55
```



Basic Task b)

```
[BITS 16]
[ORG 0x7C00]
top:
        ;; Put 0 into ds (data segment)
        ;; Can't do it directly
        mov ax,0x0000
        mov ds,ax
        ;; si is the location relative to the data segment of the
        ;; string/char to display
        mov si, Tree
        call writeString ; See below
        jmp $ ; Spin
writeString:
        mov ah,0x0E ; Display a chacter (as before)
        mov bh,0x00
        mov bl,0x07
nextchar:
        Lodsb ; Loads [SI] into AL and increases SI by one
        ;; Effectively "pumps" the string through AL
        cmp al,0 ; End of the string?
        jz done
        int 0x10 ; BIOS interrupt
        jmp nextchar
done:
        ret
        Tree db '    *',13,10, '   ***',13,10, '  *****',13,10, ' *******',
13,10, '*********',13,10, '   ***   ',13,10, '   ***   ',13,10,0 ; Null-
terminated
        times 510-($-$$) db 0
        dw 0xAA55
```

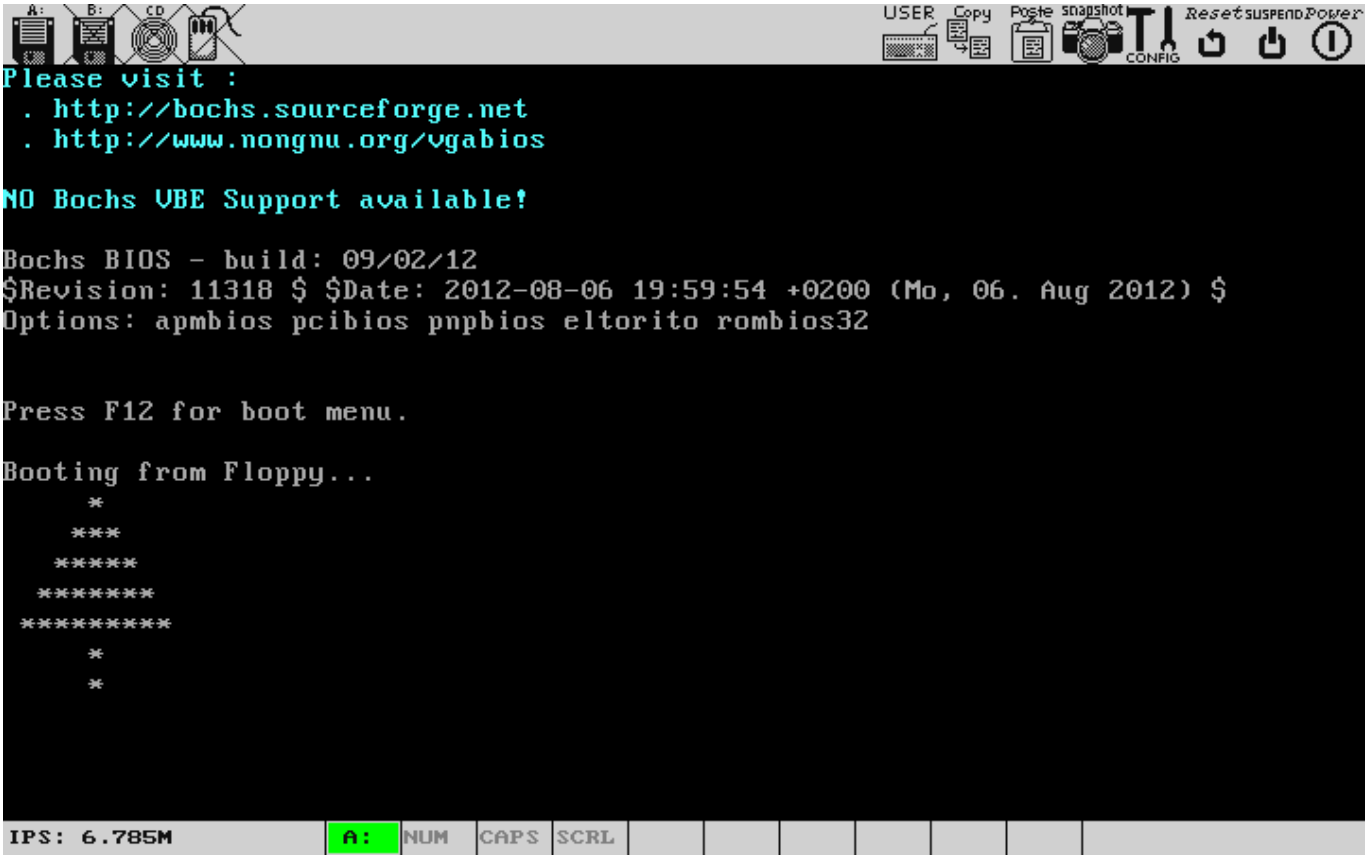


Advanced Task a)

```
[BITS 16]
[ORG 0x7C00]
top:
        ;; Put 0 into ds (data segment)
        ;; Can't do it directly
        mov ax,0x0000
        mov ds,ax
        ;; si is the location relative to the data segment of the
        ;; string/char to display
        mov cl,5 ; mov 5 into the register cl to set number of rows
        mov dl,1 ; mov 1 into dl to set no. of dots in starting row
        call Tree
        jmp $ ; Spin

Tree:
        mov dh, dl ; move starting number of dots to another memory
location so we use one as current dots needed (dh) and one as total dots in
row (dl)
        mov ch, cl ; move number of rows into another location to use as
rows remaining (cl) and spaces in row (ch)
        call spaceLoop ; print the first line of the tree
        call dotsLoop ; ^
        mov si, cr ; Print newline
        call writeString
        dec cl ; Reduce rows remaining by 1
        add dl,2 ; Increase dots in the next row by 2
        cmp cl,0 ; Check if we have printed the required amount of rows
```

```
        jne Tree ; If not, restart loop
        mov cl, 5 ; Once tree body is printed, replace cl and dl to be
number of spaces and rows respectively
        mov dl, 2 ; Number of dots is not needed for the trunk as it is 1
dot wide.
        call Trunk

spaceLoop:
        mov si, space
        call writeString ; Print a space
        dec ch ; reduce required spaces by 1
        cmp ch,0 ; see if required spaces is 0
        jne spaceLoop ; if required spaces is not 0 restart loop
        ret ; return to main code is required spaces is 0


dotsLoop:
        mov si, dot
        call writeString ; See below
        dec dh ; reduce what is in dh by 1
        cmp dh, 0 ; compare to see if what is store in dh is 0, if it is we
have printed the required number of dots
        jne dotsLoop ; if dh does not contain 0 call dotsLoop again.
        ret ; when ch contains 0 return back to main code

Trunk:
        mov ch, cl
        call TrunkSpace ; print required spaces for trunk
        mov si, dot ; print trunk dot
        call writeString
        mov si, cr ; print newline
        call writeString
        dec dl ; reduce remaining rows
        cmp dl,0 ; if not 0 print another trunk row
        jne Trunk
        ret

TrunkSpace:
        mov si, space ; print space
        call writeString
        dec ch ; reduce required spaces
        cmp ch,0
        jne spaceLoop ; if required spaces != 0 restart loop
        ret


writeString:
        mov ah,0x0E ; Display a character
        mov bh,0x00
        mov bl,0x07

nextchar:
        Lodsb ; Loads [SI] into AL and increases SI by one
        ;; Effectively "pumps" the string through AL
```

```
        cmp al,0 ; End of the string?
        jz done
        int 0x10 ; BIOS interrupt
        jmp nextchar
done:
        ret
        space db ' ',0 ; storing a space in memory
        dot db '*',0 ; storing an asterisk in memory to call
        cr db 13,10,0 ; New line
        times 510-($-$$) db 0
        dw 0xAA55
```



Lab 5

Basic Task a)

```
thoma338@hvs-its-lnx01:/proc$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz
stepping        : 4
microcode       : 0xffffffff
cpu MHz         : 2294.609
cache size      : 25344 KB
physical id     : 0
siblings        : 8
core id         : 0
cpu cores       : 8
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 21
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht sy
scall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology cpuid pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 movbe
popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single pti ssbd ibrs ibpb stibp fsgsbase bmi1
hle avx2 smep bmi2 erms invpcid rtm mpx avx512f avx512dq rdseed adx smap clflushopt avx512cd avx512bw avx512vl xsaveopt xsav
ec xsaves flush_l1d
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs taa itlb_multihit
bogomips        : 4589.21
clflush size    : 64
cache_alignment : 64
address sizes   : 44 bits physical, 48 bits virtual
power management:
```

Basic Task b)

```
thoma338@hvs-its-lnx01:~$ cat /proc/interrupts
           CPU0       CPU1       CPU2       CPU3       CPU4       CPU5       CPU6       CPU7
  0:          0          0          0          0          0          0          0          0   IO-APIC   2-edge      timer
  1:          0          0          0          0          0          9          0          0   IO-APIC   1-edge      i8042
  8:          0          0          0          0          0          0          0          0   IO-APIC   8-edge      rtc0
  9:          0          0          0          0          0          0          0          0   IO-APIC   9-fasteoi   acpi
 12:          0          0          0          0          3          0          0          0   IO-APIC  12-edge      i8042
 14:          0          0          0          0          0          0          0          0   IO-APIC  14-edge      ata_pii
x
 15:          0          0          0      31149          0          0          0          0   IO-APIC  15-edge      ata_pii
x
NMI:          0          0          0          0          0          0          0          0   Non-maskable interrupts
LOC:          0          0          0          0          0          0          0          0   Local timer interrupts
SPU:          0          0          0          0          0          0          0          0   Spurious interrupts
PMI:          0          0          0          0          0          0          0          0   Performance monitoring interr
upts
IWI:         33         25         39          0         27         22         22         26   IRQ work interrupts
RTR:          0          0          0          0          0          0          0          0   APIC ICR read retries
RES:     161269     121616     211740     131221     285079     137229     148316     148825   Rescheduling interrupts
CAL:      11332      15103      20013      10389       8992      20454      12412      29827   Function call interrupts
TLB:          0          0          0          0          0          0          0          0   TLB shootdowns
TRM:          0          0          0          0          0          0          0          0   Thermal event interrupts
THR:          0          0          0          0          0          0          0          0   Threshold APIC interrupts
DFR:          0          0          0          0          0          0          0          0   Deferred Error APIC interrupt
s
MCE:          0          0          0          0          0          0          0          0   Machine check exceptions
MCP:        101        101        101        101        101        101        101        101   Machine check polls
HYP:     174196      23656    1237912       1921      19347      29355     114024      16857   Hypervisor callback interrupt
s
HRE:          0          0          0          0          0          0          0          0   Hyper-V reenlightenment inter
rupts
HVS:     516208     512801     735076     422639     435372     523775     434287     644191   Hyper-V stimer0 interrupts
ERR:          0
MIS:          0
PIN:          0          0          0          0          0          0          0          0   Posted-interrupt notification
 event
NPI:          0          0          0          0          0          0          0          0   Nested posted-interrupt event
PIW:          0          0          0          0          0          0          0          0   Posted-interrupt wakeup event
thoma338@hvs-its-lnx01:~$ cat /proc/interrupts | wc -l
30
```

> Using cat to display the interrupts file. Then repeating the command but piping to wc -l to count the lines, meaning there are 30 different interrupts in use on the system.

Basic Task c)

```
thoma338@hvs-its-lnx01:~$ grep -E -i "vendor|processor|model name" /proc/cpuinfo | sort | uniq
model name      : Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz
processor       : 0
processor       : 1
processor       : 2
processor       : 3
processor       : 4
processor       : 5
processor       : 6
processor       : 7
vendor_id       : GenuineIntel
```

> There are 8 CPUs (0-7), it's a Xeon 6140 made by Intel.

Basic Task d)

```
thoma338@hvs-its-lnx01:/proc$ awk '{ print $3, $5 }' /proc/loadavg
0.01 59034
```

Advanced Task

```
#!/bin/bash

# Each menu item is seperated into a function to keep the code clean and
easy to read.
CPU_Info (){
        cat /proc/cpuinfo # Display contents of /proc/cpuinfo
}

Interrupts(){
        cat /proc/interrupts # Display contents of /proc/interrupts
}

Load(){
        awk '{ print $3, $5 }' /proc/loadavg # Using awk to print the 3rd
and 5th columns from the loadavg output.
}

Process(){
        PID=$(ls /proc | head -n1) # Take the first file in the /proc/
directory and assign the PID to a variable.
        echo PID:
        echo $PID # Printing PID variable.
        echo Name:
        ps -p $PID -o comm= # Using ps, find the command name for that PID.
}

while true; do
        options=("Display CPU info" "Display a list of interrupts" "Display
the load average" "Display a process" "Exit") # These will be the menu
options

        PS3='Please select an option: ' # PS3 is an environmental variable
used by the select statement.
        select option in "${options[@]}"; do # Creates a menu from the
```

```
options provided in in [options]
                case $option in # Matches the selected menu item to one of
the following string patterns.
                        "Display CPU info") # If selected menu item matches
this pattern the code indented below is executed.
                                CPU_Info; break;; # Calls the CPU_info
function and breaks the first loop.
                        "Display a list of interrupts")
                                Interrupts; break;;
                        "Display the load average")
                                Load; break;;
                        "Display a process")
                                Process; break;;
                        "Exit")
                                break 2 # Break [n] breaks the nth loop
                                ;;
                        *) # Wildcard that matches if nothing else hits,
used to catch incorrect inputs.
                            echo "That's not an option is it..."
                            ;;
                esac
        done

done

echo Exiting...
```

```
thoma338@hvs-its-lnx01:~$ ./bash_menu
1) Display CPU info             4) Display a process
2) Display a list of interrupts  5) Exit
3) Display the load average
Please select an option:
```

> Menu running

```
jamie@archtop ~/OneDrive/Uni/Year2/207/Week5 ./bash_menu.sh
1) Display CPU info              3) Display the load average      5) Exit
2) Display a list of interrupts  4) Display a process
Please select an option: 1
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 165
model name      : Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz
stepping        : 2
microcode       : 0xe0
cpu MHz         : 961.357
cache size      : 8192 KB
physical id     : 0
siblings        : 8
core id         : 0
cpu cores       : 4
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
```

A section of the CPU info output

```
jamie@archtop ~/OneDrive/Uni/Year2/207/Week5 ./bash_menu.sh
1) Display CPU info              3) Display the load average      5) Exit
2) Display a list of interrupts  4) Display a process
Please select an option: 2
           CPU0       CPU1       CPU2       CPU3       CPU4       CPU5       CPU6       CPU7
  0:          7          0          0          0          0          0          0          0  IR-IO-APIC    2-edge
    timer
  1:          0          0       5628          0          0          0          0          0  IR-IO-APIC    1-edge
    i8042
  8:          0          0          0          0          0          0          0          0  IR-IO-APIC    8-edge
    rtc0
  9:          0       5388          0          0          0          0          0          0  IR-IO-APIC    9-faste
oi  acpi
 14:          0          0          0          0          0          0          0          0  IR-IO-APIC   14-faste
oi  INT3450:00
 16:          0          0          0          0          0          0          0          0  IR-IO-APIC   16-faste
oi  idma64.0, i2c_designware.0, i801_smbus
 17:          0          0          0    4242606          0          0          0          0  IR-IO-APIC   17-faste
oi  idma64.1, i2c_designware.1, snd_hda_intel:card0
 20:          0          0          0          0          0          0          0         60  IR-IO-APIC   20-faste
oi  intel_ish_ipc
 63:          0          0      93498          0          0          0          0          0  IR-IO-APIC   63-faste
oi  SYNA2B5A:00
120:          0          0          0          0          0          0          0          0  DMAR-MSI    0-edge
  dmar0
121:          0          0          0          0          0          0          0          0  DMAR-MSI    1-edge
  dmar1
```

A section of the Interrupts output

```
thoma338@hvs-its-lnx01:~$ ./bash_menu
1) Display CPU info              4) Display a process
2) Display a list of interrupts  5) Exit
3) Display the load average
Please select an option: 3
0.00 7309
```

Load Average option

```
thoma338@hvs-its-lnx01:~$ ./bash_menu
1) Display CPU info              4) Display a process
2) Display a list of interrupts  5) Exit
3) Display the load average
Please select an option: 4
PID:
1
Name:
systemd
```

> Displaying a process from /proc/

```
thoma338@hvs-its-lnx01:~$ ./bash_menu
1) Display CPU info              4) Display a process
2) Display a list of interrupts  5) Exit
3) Display the load average
Please select an option: 5
Exiting...
```

> Exiting the menu

Lab 7

Basic Task a)

```
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <iostream>
#define BUF_SIZE 500
#define OUTPUT_MODE 0700
using namespace std;
int main(int argc, char *argv[])
{
        int in_file, out_file;
        int read_size = 1, write_size;
        char buf[BUF_SIZE];
        if (argc != 3)
                exit(1);
        in_file= open(argv[1], O_RDONLY);
        if (in_file  < 0)
                exit(2);
        out_file = creat(argv[2], OUTPUT_MODE);
        if (out_file < 0)
                exit(3);
        while (read_size > 0)
        {
                read_size = read(in_file, buf, BUF_SIZE);
                if (read_size <0)
                        exit(4);
                write_size = write(out_file,buf, read_size);
                if (write_size<=0)
                {
                        close(in_file);
```

```
                        close(out_file);
                        exit(5);
                        cout<<"Reading and writing from and to files is
complete"<<endl;
                }
        }
}
```

> This code takes 2 files as arguments in the command line. Multiple checks are made, first to see if the correct number of arguments have been entered, then to see if both the input and output file can be read or written to correctly, exiting if any errors are caught. In a while loop the buffer is repeatedly filled with 500 characters from the input file, the buffer is then written to the output file. This repeats until the write_size variable reaches 0, this indicates that there is no data in the buffer and the program has reached the end of the file. Once this happens the program closes the input and output files and exits. In conclusion this code works as a 'copy/paste' function between two files, the first file being copied into the second.

Basic Task b-c)

```
#include <fcntl.h>
#include <vector>
#include <string>
#include <algorithm>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <iostream>

using namespace std;

#define BUF_SIZE 500 //Setting the buffer size to 500 bytes
#define OUTPUT_MODE 0700 //Setting the permissions of the output file


int main(int argc, char *argv[])
{
    int in_file, out_file; //Initialising input and output files as
integers
    int read_size = 1, write_size; //Setting read and write size to 1 to
make sure the program goes into the while loop at line 45 and avoids
tripping line 84/90 on the first run and
    char buf[BUF_SIZE]; //Initialising a character array of the same size
as buf_size to store the characters from the input file
    if (argc != 3) //Checks to see if correct number of arguments entered,
exits with statement if caught
    {
        cout << "Please enter 3 arguments into the command line." << endl;
        exit(1);
    }
```

```
    in_file= open(argv[1], O_RDONLY); // Reading first argument given as
input file in read only mode
    if (in_file  < 0) //If the file cannot be opened this statement exits
the program
    {
        cout << "Error opening the designated file." << endl;
        exit(2);
    }

    out_file = creat(argv[2], OUTPUT_MODE);
    if (out_file < 0) //See above, with creation of output file instead
    {
        cout << "Error creating the output file." << endl;
        exit(3);
    }

    int vowel_count = 0; //Initia``lising the 4 counters used to track the
stats of the inputted file
    int total_chars = 0;
    int const_wordstart = 0;
    int times_filled = 0;
    while (read_size > 0) //Only run this loop while there is still data
being read from the file
    {
    read_size = read(in_file, buf, BUF_SIZE); //Fills the buffer as far as
it can from input file and gets the size
    cout << "Characters read from buffer: " << read_size << endl; //Outputs
no. of characters read into the buffer
    if(read_size != 0) //Increments the total times filled variable if the
read size is not 0
    {
        times_filled++;
    }
    for(int i=0; i<read_size; i++) //Iterating through each character
stored in the buffer one by one
    {
        if(buf[i] != ' ') //If the character being read isn't a space, inc
the total chars counter
        {
            total_chars++;
        }
        if(buf[i]== 'a' | buf[i]=='e' | buf[i]=='i' | buf[i]=='o' |
buf[i]=='u') //If the character being read is a vowel, inc vowel count
        {
            vowel_count++;
        }
        if(buf[i+1] == ' ') //Reading one character ahead to see if the
current character is the end of a word
        {
            auto word_start = buf[i+2]; //Storing the start of the next
word
            if(word_start != 'a' & word_start != 'e' & word_start != 'i' &
word_start != 'o' & word_start != 'u' ) //Checking if word start is not a
vowel, inc const_wordstart if it isn't
```

```
                {
                    const_wordstart++;
                }
            }
        }

        if (read_size <0) //Checking to see if the file can be read, exiting if
    it cant
        {
            cout << "Error reading data from file." << endl;
            exit(4);
        }
        write_size = write(out_file, buf, read_size); //Writes the contents of
    the buffer to the output file
        if (write_size<=0) //When the write size is 0 the program has reached
    the end of the input file
        {
            close(in_file); //Closing both files
            close(out_file);
            cout<<"Reading and writing from and to files is complete"<<endl;
    //Outputting all stats collected on the input file, then exiting
            cout << "Total vowels: " << vowel_count << endl;
            cout << "Consonents at the start of a word: " << const_wordstart <<
    endl;
            cout << "Total characters excluding spaces: " << total_chars <<
    endl;
            cout << "Times the buffer was filled: " << times_filled << endl;
            exit(5);
        }
        }

    }
```

```
jamie@archtop ~/OneDrive/Uni/Year2/207/Week7/buffer/basic ./buffer_info parasite.txt
Please enter 3 arguments into the command line.
```

> The error message generated when there are not 3 arguments given to the cli.

```
jamie@archtop ~/OneDrive/Uni/Year2/207/Week7/buffer/basic ./buffer_info parasite_wrong.txt parasite_rev.txt
Error opening the designated file.
```

> Error message generated when the input file cannot be read.

```
jamie@archtop ~/OneDrive/Uni/Year2/207/Week7/buffer/basic ./buffer_info parasite.txt parasite_rev.txt
Error creating the output file.
```

> Error message generated when the output file cannot be created.

```
jamie@archtop ~/OneDrive/Uni/Year2/207/Week7/buffer/basic ./buffer_info parasite.txt parasite_rev.txt
Characters read from buffer: 500
Characters read from buffer: 500
Characters read from buffer: 500
Characters read from buffer: 500
Characters read from buffer: 500
Characters read from buffer: 416
Characters read from buffer: 0
Reading and writing from and to files is complete
Total vowels: 858
Consonents at the start of a word: 347
Total characters excluding spaces: 2432
Times the buffer was filled: 6
```

> Full output of statistics when no errors occur.

Advanced Task a)

```
#include <fcntl.h>
#include <string>
#include <algorithm>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <iostream>

using namespace std;

#define BUF_SIZE 500 //Setting the buffer size to 500 bytes
#define OUTPUT_MODE 0700 //Setting the permissions of the output file

void removeStops(int file) // Function for removing stop words from the
file passed to it
{
    int read_size = 1, write_size; //Initialising read and write size to be
> 0
    char buf_new[BUF_SIZE]; //Setting up a new buffer to avoid any errors
in printing
    while(read_size > 0)
    {
        read_size = read(file, buf_new, BUF_SIZE);
        for(int j=0; j<read_size; j++) //Reading each character in buffer
one by one
        {
            /*
                Each time a space is read by the program it checks to see
if the next word is a stop word
                If the next word is a stop word the counter 'j' is skipped
forward over the word to avoid printing it to display
            */
            if(buf_new[j]==' ' & buf_new[j+1]=='a' & buf_new[j+2]==' ')
            {
                j = j+2;
            }
            if(buf_new[j]==' ' & buf_new[j+1]=='t' & buf_new[j+2]=='h' &
buf_new[j+3]=='e' & buf_new[j+4]==' ')
```

```
            {
                j = j+4;
            }
            if(buf_new[j]==' ' & buf_new[j+1]=='a' & buf_new[j+2]=='n' &
buf_new[j+3]==' ')
            {
                j = j+3;
            }
            if(buf_new[j]==' ' & buf_new[j+1]=='t' & buf_new[j+2]=='o' &
buf_new[j+3]==' ')
            {
                j = j+3;
            }

            else
            {
                cout << buf_new[j]; //If current character is not a space
followed by a stop word, print to display current character in buff
            }
        }
    }
    cout << "============================================================" <<
endl << endl << endl;
    close(file);
}

int main(int argc, char *argv[])
{
    int in_file1, in_file2, out_file; //Initialising both input files and
the output file as integers
    int read_size = 1, write_size;
    int read_size2 = 1, write_size2;
    char buf[BUF_SIZE]; //Creating two buffers, one for each input file
    char buf2[BUF_SIZE];

    //There are 5 error checks in this code: the correct amount of
arguments, open and read tests on the input files and a creation test for
the output file
    if (argc != 4)
    {
        cout << "Please enter 4 arguments into the command line." << endl;
        exit(1);
    }

    in_file1= open(argv[1], O_RDONLY);
    if (in_file1  < 0)
    {
        cout << "Error opening the designated file." << endl;
        exit(2);
    }

    in_file2 = open(argv[2], O_RDONLY);
    if (in_file2  < 0)
    {
```

```
        cout << "Error opening the designated file." << endl;
        exit(3);
    }

    out_file = creat(argv[3], OUTPUT_MODE);
    if (out_file < 0)
    {
        cout << "Error creating the output file." << endl;
        exit(4);
    }

    bool diff = false; //Stays false until the program detects the files
are different
    while (read_size > 0 | read_size2 > 0)
    {
        if(diff == true) //If diff is set to false break out of the while
loop
        {
            break;
        }
        read_size = read(in_file1, buf, BUF_SIZE); // Filling both buffers
with each input file
        read_size2 = read(in_file2, buf2, BUF_SIZE);
        int char_pos = 0; //Tracking the number of the character the code
is currently reading
        for(int i=0; i<read_size; i++) //Iterate through each character in
the buffer one by one
        {
            char_pos++;
            if(buf[i] == buf2[i])  //Checks to see if the characters in the
same position in the two files are the same
            {
                continue;
            }
            else //Catches when the characters differ
            {
                cout << "Character mismatch at character: " << char_pos <<
endl;
                cout << "File 1 contained: " << buf[i] << endl;
                cout << "File 2 contained: " << buf2[i] << endl;
                diff = true; //Set diff to true so we can break out of
while loop
                break; //Exit for loop
            }
        }

        if (read_size <0)
        {
            cout << "Error reading data from file." << endl;
            exit(6);
        }

        write_size = write(out_file, buf, read_size); //If files are
identical execute normal function of copying input file to output file
```

```
        if (write_size<=0)
        {
            close(in_file1);
            close(in_file2);
            close(out_file);
            cout<<"Reading and writing from and to files is complete"
<<endl;
            exit(7);
        }
    }
    //When the program detects the files are different this code is
executed
    //Closing and reopening each file eliminated some errors I was having
with incomplete outputs
    close(in_file1);
    in_file1= open(argv[1], O_RDONLY);
    removeStops(in_file1); //Calling removeStops for the first input file
    close(in_file2);
    in_file2 = open(argv[2], O_RDONLY);
    removeStops(in_file2); //See above, with second input file
    exit(8);
}
```

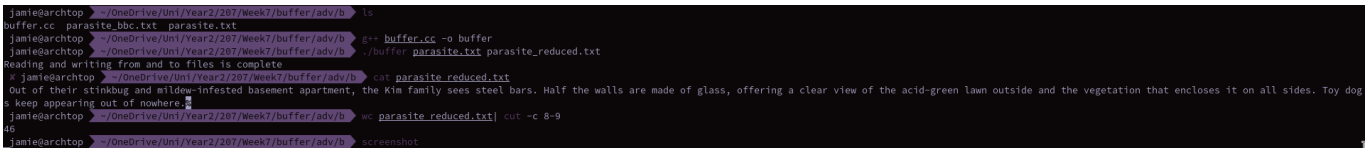
Advanced Task b)

```
#include <fcntl.h>
#include <vector>
#include <string>
#include <algorithm>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <iostream>

using namespace std;

#define BUF_SIZE 500 //Setting the buffer size to 500 bytes
#define OUTPUT_MODE 0700 //Setting the permissions of the output file


int main(int argc, char *argv[])
{
    int in_file, out_file; //Initialising input and output files as
integers
    int read_size = 1, write_size; //Setting read and write size to 1 to
make sure the program goes into the while loop at line 45 and avoids
tripping line 84/90 on the first run and
    char buf[BUF_SIZE]; //Initialising a character array of the same size
as buf_size to store the characters from the input file
    if (argc != 3) //Checks to see if correct number of arguments entered,
```

```
exits with statement if caught
    {
        cout << "Please enter 3 arguments into the command line." << endl;
        exit(1);
    }

    in_file= open(argv[1], O_RDONLY); // Reading first argument given as
input file in read only mode
    if (in_file  < 0) //If the file cannot be opened this statement exits
the program
    {
        cout << "Error opening the designated file." << endl;
        exit(2);
    }

    out_file = creat(argv[2], OUTPUT_MODE);
    if (out_file < 0) //See above, with output file instead
    {
        cout << "Error creating the output file." << endl;
        exit(3);
    }

    int total = 0;
    int count = 0;
    while (read_size > 0) //Only run this loop while there is still data
being read from the file
    {
        char buf2[BUF_SIZE] = {}; //Empty buffer on each run to avoid
duplicating outputs
        count = 0;
        read_size = read(in_file, buf, BUF_SIZE); //Fills the buffer as far
as it can from input file and gets the size
        for(int i=0; i<read_size; i++) //Iterating through each character
stored in the buffer one by one
        {
            if(total < 3) //Only want to grab 3 sentences so we dont exceed
50 words
            {
                if(buf[i] != '.' && count == 1) //If these are characters
after the first fullstop in the buffer
                {
                    buf2[i] = buf[i]; //Add current character to our output
buffer
                }
                if(buf[i] == '.' && count == 1)//Are we at the end of the
sentence we want to output?
                {
                    buf2[i] = buf[i]; //Add last fullstop to output buffer
so total output makes sense
                    count++;
                    total++; //Total sentences outputted has increased by
one
                }
                if(buf[i] == '.' && count != 1) //Any other fullstops in
```

```
the buffer also increase the count
                {
                    count++;
                }
            }
        }


        if (read_size <0) //Checking to see if the file can be read,
exiting if it cant
        {
            cout << "Error reading data from file." << endl;
            exit(4);
        }
        write_size = write(out_file, buf2, read_size); //Writes the
contents of the buffer to the output file
        if (write_size<=0) //When the write size is 0 the program has
reached the end of the input file
        {
            close(in_file); //Closing both files
            close(out_file);
            cout<<"Reading and writing from and to files is complete"
<<endl; //Outputting all stats collected on the input file, then exiting
            exit(5);
        }
    }
}
```

> This algorithm works by taking the sentence in the buffer after the first full stop and outputting this to the file. It does this until the buffer stops reading data from the input file or if the total number of sentences has been reached. As the buffer refills roughly 6 or 7 times, taking one sentence from each of these reads seemed like a good average. To reach the required amount of words for this task I set the required sentences to 3.



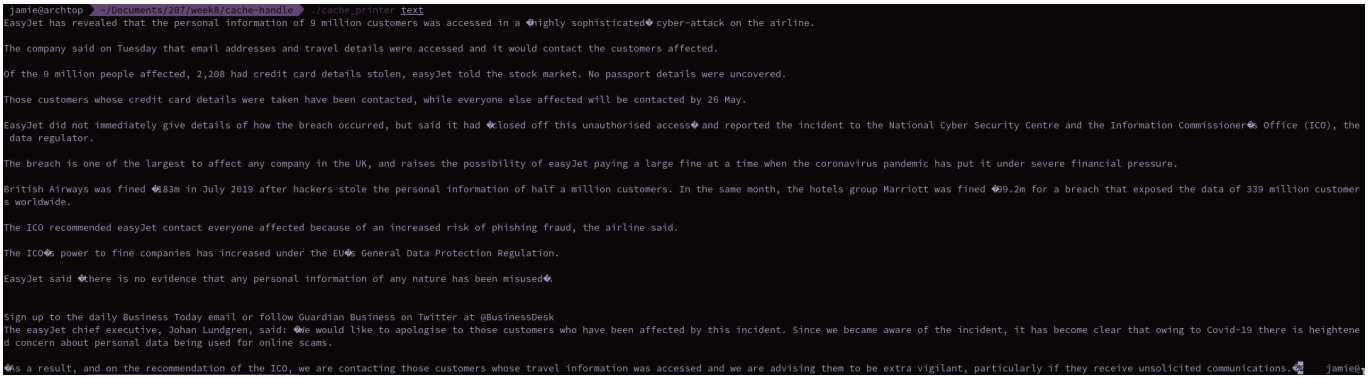
Lab 8

Basic Task a)

This is the function `return_character` from the file `buffer_handle.c`

```
char return_character(bufferCacheStruct* buff){
  char character; //Initialise character variable
  bufferCache_refill(buff); //Check if buffer needs to be refilled
  character = buff->bufferCache[buff->alongBufferCache]; //Sets character
to the current character in the buffer according to alongBufferCache
  buff->alongBufferCache++; //Set position to the next character
```

```
    return character; //Return current character
  }
```



Advanced Task a)

buffer_handle.c

```
#include "cache_handle.h"
#include <unistd.h>

//http://www.phim.unibe.ch/comp_doc/c_manual/C/SYNTAX/struct.html
//http://vergil.chemistry.gatech.edu/resources/programming/c-
tutorial/structs.html


int bufferCache_refill(bufferCacheStruct* buff){
  //Refills a buffer-cache
  //Only works when completely used buffer-cache
  if(buff->alongBufferCache!=buff->bufferCacheLength) //Checks if we're at
the end of the buffer or not. (if(position!=Length of buf))
    return 0;
  else{
    printf("\n-----------------------\n");
    printf("Cache Refilled!\n");
    printf("-----------------------\n");
    buff->alongBufferCache=0; //Set position in cache to the start
    int len=fread(buff->bufferCache, sizeof(char), buff->bufferCacheLength,
buff->file);//Read in new chunk on data into the cache
    if(len<buff->bufferCacheLength)
      for(int i=len;i<buff->bufferCacheLength;i++)  //If not enough data to
fill the cache, fil with EOF
    buff->bufferCache[i]=EOF;  //Accessing like an array!
    return len;
  }

}

void file_close(bufferCacheStruct* buff){
  free(buff->bufferCache);
  fclose(buff->file);
}
```

```
bufferCacheStruct* file_open(char * filename, int bufferCacheSize){
  //Info on malloc
  FILE* f;
  if ((f = fopen(filename, "r")) == NULL){
    fprintf(stderr, "Cannot open %s\n", filename);
    return 0;
  }

  bufferCacheStruct* initBufferCache=
(bufferCacheStruct*)malloc(sizeof(bufferCacheStruct));
  initBufferCache->file=f;
  initBufferCache->bufferCacheLength=bufferCacheSize;
  initBufferCache->alongBufferCache=bufferCacheSize; //Start off with no
characters, so refill will work as expected
  initBufferCache->bufferCache=(char*)malloc(sizeof(char)*bufferCacheSize);

  bufferCache_refill(initBufferCache);
  return initBufferCache;
}


int msleep(unsigned int tms){
  return usleep(tms * 1000);
}

//-----------------------------------------------------------------------
char return_character(bufferCacheStruct* buff){
  char character; //Initialise character variable
  bufferCache_refill(buff); //Check if buffer needs to be refilled
  character = buff->bufferCache[buff->alongBufferCache]; //Sets character
to the current character in the buffer according to alongBufferCache
  buff->alongBufferCache++; //Set position to the next character
  printf("%c", character);
  fflush(stdout);
  msleep(5);
  return character; //Return current character
}
```

This is an example of the output from advanced task A. Everytime the cache is refilled the message is printed. Each time a character is returned by the return_character funtion it is printed to the display along with a new line to show that one character is being read at a time. When the code runs, the usleep function is used to delay the output of each character to further display that one character is being read at a time.

Advanced Task b)

cache_printer.c

```
#include "cache_handle.h"
#include <unistd.h>
#include <stdbool.h>
#include <string.h>

int main(){
  //Open a file
  bufferCacheStruct* f = file_open("text",20);
  int VowelC=0;//Initialise counters for output
  int SentenceC=0;
  int StopC = 0;
  int EasyC = 0;
  bool end = false;
  while(end == false)//While we're not at the end of tbe file
  {
    char *arr = return_character(f);//Get array from return_character
function
    if(arr[0]!=EOF)//If we're not at the end of the file
    {
      printf("%c", arr[0]);//Print current character
      if(arr[0] == '.')//Check to see if sentence end
      {
        SentenceC++;
      }
      if(arr[1] == 'a' || arr[1] == 'e' || arr[1] == 'i' || arr[1] == 'o'
|| arr[1] == 'u')//The next character will be the start of a word, so
checking if it is a vowel
      {
        VowelC++;
      }
      if(strcmp(arr, " the")== 0 || strcmp(arr, " an")== 0 || strcmp(arr, "
a")== 0 || strcmp(arr, " to")== 0 || strcmp(arr, " and")== 0)//Check to see
if the entire array is equal to a stop word
      {
        StopC++;
      }
      if(strcmp(arr, " easyjet") == 0 || strcmp(arr, " card") == 0 ||
strcmp(arr, " breach") == 0)//Same as above but with specific words
      {
        EasyC++;
      }
    }
    else
    { //Output all counters and close the file when we reach the end
      printf("\n=========================\n");
      printf("Total words Starting with a Vowel: %i\n", VowelC);
      printf("Total Sentences in Document: %i\n", SentenceC);
      printf("Total number of stop words: %i\n", StopC);
```

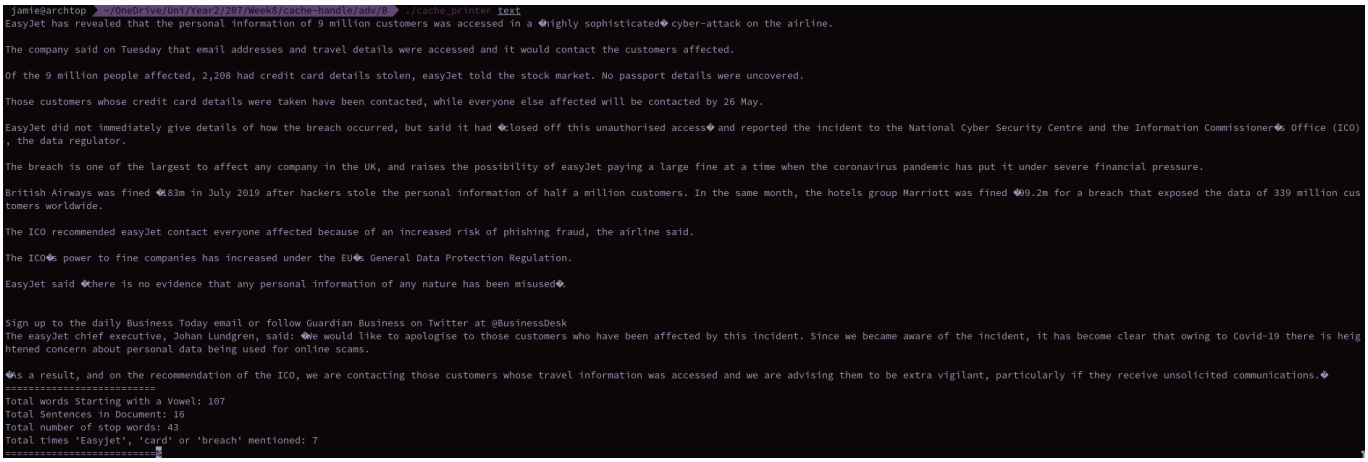
```
      printf("Total times 'Easyjet', 'card' or 'breach' mentioned: %i\n",
EasyC);
      printf("==========================");
      file_close(f);
      end = true;
    }
  }
  return 0;
}
```

cache_handle.c

```
#include "cache_handle.h"
#include <unistd.h>
#include <stdbool.h>
#include <ctype.h>
#include <stdio.h>

//http://www.phim.unibe.ch/comp_doc/c_manual/C/SYNTAX/struct.html
//http://vergil.chemistry.gatech.edu/resources/programming/c-
tutorial/structs.html


int bufferCache_refill(bufferCacheStruct* buff){
  //Refills a buffer-cache
  //Only works when completely used buffer-cache
  if(buff->alongBufferCache!=buff->bufferCacheLength) //Checks if we're at
the end of the buffer or not. (if(position!=Length of buf))
    return 0;
  else{
    buff->alongBufferCache=0; //Set position in cache to the start
    int len=fread(buff->bufferCache, sizeof(char), buff->bufferCacheLength,
buff->file);//Read in new chunk on data into the cache
    if(len<buff->bufferCacheLength)
      for(int i=len;i<buff->bufferCacheLength;i++)  //If not enough data to
fill the cache, fil with EOF
    buff->bufferCache[i]=EOF;  //Accessing like an array!
    return len;
  }

}

void file_close(bufferCacheStruct* buff){
  free(buff->bufferCache);
  fclose(buff->file);
}


bufferCacheStruct* file_open(char * filename, int bufferCacheSize){
  //Info on malloc
  //http://www.space.unibe.ch/comp_doc/c_manual/C/FUNCTIONS/malloc.html
  FILE* f;
```

```
  if ((f = fopen(filename, "r")) == NULL){
    fprintf(stderr, "Cannot open %s\n", filename);
    return 0;
  }

  bufferCacheStruct* initBufferCache=
(bufferCacheStruct*)malloc(sizeof(bufferCacheStruct));
  initBufferCache->file=f;
  initBufferCache->bufferCacheLength=bufferCacheSize;
  initBufferCache->alongBufferCache=bufferCacheSize; //Start off with no
characters, so refill will work as expected
  initBufferCache->bufferCache=(char*)malloc(sizeof(char)*bufferCacheSize);

  bufferCache_refill(initBufferCache);
  return initBufferCache;
}

int msleep(unsigned int tms){ //Convert micro seconds to milliseconds for
sleep
  return usleep(tms * 1000);
}
//-----------------------------------------------------------------------
char* return_character(bufferCacheStruct* buff){
  char character;//Init first character and temp
  char temp_char;
  bool reading;//Initialise value of whether we're reading the characters
into the array
  int count = 0;//How many chars we've put in array
  char *arr = NULL;//Init char array
  arr = (char *)malloc(100);//Using malloc to assign memory to the array so
we can access it in cache_printer.c
  bufferCache_refill(buff); //Check if buffer needs to be refilled
  character = buff->bufferCache[buff->alongBufferCache]; //Sets character
to the current character in the buffer according to alongBufferCache
if(character == ' ')
  arr[0] = character;//Assign first position in array to initial character
so it still prints to display
  if(character == ' ' || character == '\n')//If the next character is the
start of a new word
  {
    reading = true;
    count = 1;//Set count to 1 so we don't override position 0 in array
    while(reading == true)//While we're writing the next word to the array
    {
      temp_char = buff->bufferCache[buff->alongBufferCache+count]; //Grab
next char
      if(temp_char != ' ' & temp_char != '.')//If we're still in the middle
of the word
      {
        arr[count] = tolower(temp_char);//Add lowercase version of char to
array in position 'count' to avoid missing words due to case
        count++;
      }
      else//When we've reached the end
```

```
        {
          reading = false;
        }

      }
    }
    buff->alongBufferCache++; //Set position to the next character
    return arr; //Return char array of current character, or next word if
  current character is a space
  }
```



Lab 10

Basic Task

cache_handle.h

```
#include <stdio.h>
#include <stdlib.h>

//The internals of this struct aren't important
//from the user's point of view
typedef struct{
  int file;         //File being read
  int bufferCacheLength;  //Fixed buffer-cache length
  int alongBufferCache;     //Current point in the buffer-cache
  char* bufferCache;       //A pointer to a piece of memory
                      //  same length as "cacheLength"
} bufferCacheStruct;

//Open a file with a given size of cache with
bufferCacheStruct* file_open(char* filename, int bufferCacheSize);

//Close an open file
void file_close(bufferCacheStruct* buff);

//Read a byte.  Will return EOF if empty.
char return_character(bufferCacheStruct* buff);
```

```
//-------------------------------------------------------

//Refill an empty buffer.  Not intended for users
int bufferCache_refill(bufferCacheStruct* buff);
```

cache_handle.c

```
#include "cache_handle.h"
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

//http://www.phim.unibe.ch/comp_doc/c_manual/C/SYNTAX/struct.html
//http://vergil.chemistry.gatech.edu/resources/programming/c-
tutorial/structs.html


int bufferCache_refill(bufferCacheStruct* buff){
  //Refills a buffer-cache
  //Only works when completely used buffer-cache
  if(buff->alongBufferCache!=buff->bufferCacheLength) //Checks if we're at
the end of the buffer or not. (if(position!=Length of buf))
    return 0;
  else{
    buff->alongBufferCache=0; //Set position in cache to the start
    int len=read(buff->file, buff->bufferCache, buff-
>bufferCacheLength);//Read in new chunk on data into the cache
    if(len<buff->bufferCacheLength)
      for(int i=len;i<buff->bufferCacheLength;i++)  //If not enough data to
fill the cache, fil with EOF
    buff->bufferCache[i]=EOF;  //Accessing like an array!
    return len;
  }

}

void file_close(bufferCacheStruct* buff){
  free(buff->bufferCache);
  close(buff->file);
}


bufferCacheStruct* file_open(char * filename, int bufferCacheSize){
  //Info on malloc
  //http://www.space.unibe.ch/comp_doc/c_manual/C/FUNCTIONS/malloc.html
  int f;
  if ((f = open(filename, O_RDONLY)) == -1){
    fprintf(stderr, "Cannot open %s\n", filename);
    return 0;
  }
```

```
  bufferCacheStruct* initBufferCache=
(bufferCacheStruct*)malloc(sizeof(bufferCacheStruct));
  initBufferCache->file=f;
  initBufferCache->bufferCacheLength=bufferCacheSize;
  initBufferCache->alongBufferCache=bufferCacheSize; //Start off with no
characters, so refill will work as expected
  initBufferCache->bufferCache=(char*)malloc(sizeof(char)*bufferCacheSize);

  bufferCache_refill(initBufferCache);
  return initBufferCache;
}
//-------------------------------------------------------------------
char return_character(bufferCacheStruct* buff){
  char character; //Initialise character variable
  bufferCache_refill(buff); //Check if buffer needs to be refilled
  character = buff->bufferCache[buff->alongBufferCache]; //Sets character
to the current character in the buffer according to alongBufferCache
  buff->alongBufferCache++; //Set position to the next character
  return character; //Return current character
}
```

```
jamie@archtop ~/OneDrive/Uni/Year2/207/Week10/cache-handle/basic ./cache_printer text
EasyJet has revealed that the personal information of 9 million customers was accessed in a highly sophisticated cyber-attack on the airline.

The company said on Tuesday that email addresses and travel details were accessed and it would contact the customers affected.

Of the 9 million people affected, 2,208 had credit card details stolen, easyJet told the stock market. No passport details were uncovered.

Those customers whose credit card details were taken have been contacted, while everyone else affected will be contacted by 26 May.

EasyJet did not immediately give details of how the breach occurred, but said it had closed off this unauthorised access and reported the incident to the National Cyber Security Centre and the Information Commissioner Office (ICO), the
 data regulator.

The breach is one of the largest to affect any company in the UK, and raises the possibility of easyJet paying a large fine at a time when the coronavirus pandemic has put it under severe financial pressure.

British Airways was fined 83m in July 2019 after hackers stole the personal information of half a million customers. In the same month, the hotels group Marriott was fined 99.2m for a breach that exposed the data of 339 million customer
s worldwide.

The ICO recommended easyJet contact everyone affected because of an increased risk of phishing fraud, the airline said.

The ICO s power to fine companies has increased under the EU s General Data Protection Regulation.

EasyJet said there is no evidence that any personal information of any nature has been misused.


Sign up to the daily Business Today email or follow Guardian Business on Twitter at @BusinessDesk
The easyJet chief executive, Johan Lundgren, said: We would like to apologise to those customers who have been affected by this incident. Since we became aware of the incident, it has become clear that owing to Covid-19 there is heightene
d concern about personal data being used for online scams.

As a result, and on the recommendation of the ICO, we are contacting those customers whose travel information was accessed and we are advising them to be extra vigilant, particularly if they receive unsolicited communications. jamie@
```

Advanced Task a)

```
#define _GNU_SOURCE
#include "cache_handle.h"
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

//http://www.phim.unibe.ch/comp_doc/c_manual/C/SYNTAX/struct.html
//http://vergil.chemistry.gatech.edu/resources/programming/c-
tutorial/structs.html


int bufferCache_refill(bufferCacheStruct* buff){
  //Refills a buffer-cache
  //Only works when completely used buffer-cache
  if(buff->alongBufferCache!=buff->bufferCacheLength) //Checks if we're at
```

```
the end of the buffer or not. (if(position!=Length of buf))
    return 0;
  else{
    buff->alongBufferCache=0; //Set position in cache to the start
    int len=read(buff->file, buff->bufferCache, buff-
>bufferCacheLength);//Read in new chunk on data into the cache
    if(len<buff->bufferCacheLength)
      for(int i=len;i<buff->bufferCacheLength;i++)  //If not enough data to
fill the cache, fil with EOF
    buff->bufferCache[i]=EOF;  //Accessing like an array!
    return len;
  }

}

void file_close(bufferCacheStruct* buff){
  free(buff->bufferCache);
  close(buff->file);
}


bufferCacheStruct* file_open(char * filename, int bufferCacheSize){
  //Info on malloc
  //http://www.space.unibe.ch/comp_doc/c_manual/C/FUNCTIONS/malloc.html
  int f;
  if ((f = open(filename, O_RDONLY | O_DIRECT | O_SYNC)) == -1){
    fprintf(stderr, "Cannot open %s\n", filename);
    return 0;
  }

  bufferCacheStruct* initBufferCache=
(bufferCacheStruct*)malloc(sizeof(bufferCacheStruct));
  initBufferCache->file=f;
  initBufferCache->bufferCacheLength=bufferCacheSize;
  initBufferCache->alongBufferCache=bufferCacheSize; //Start off with no
characters, so refill will work as expected
  initBufferCache->bufferCache=(char*)malloc(sizeof(char)*bufferCacheSize);

  bufferCache_refill(initBufferCache);
  return initBufferCache;
}
//---------------------------------------------------------------------
char return_character(bufferCacheStruct* buff){
  char character; //Initialise character variable
  bufferCache_refill(buff); //Check if buffer needs to be refilled
  character = buff->bufferCache[buff->alongBufferCache]; //Sets character
to the current character in the buffer according to alongBufferCache
  buff->alongBufferCache++; //Set position to the next character
  return character; //Return current character
}
```

> From my understanding of the task, adding the O_DIRECT and possibly O_SYNC as flags for the read function should remove the effects of cacheing in the program. When the code is run with these flags set it produces a 'double free or curruption (out)' error.

References

University of Helsinki. 2019. Operating system and user interface. Student's Digital Skills. https://blogs.helsinki.fi/students-digital-skills/1-introduction-to-the-use-of-computers/1-1-computer-functionality/operating-system-and-user-interface/