THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

This exam paper must not be removed from the venue

Venue	____________________
Seat Number	__________
Student Number	|__|__|__|__|__|__|__|__|
Family Name	____________________
First Name	____________________

School of Information Technology and Electrical Engineering

EXAMINATION

Semester Two Final Examinations, 2021

CSSE2310 / CSSE7231 Computer Systems Principles and Programming

This paper is for St Lucia Campus students.

Examination Duration: 120 minutes

Reading Time: 10 minutes

Exam Conditions:

This is an Open Book examination.

Casio FX82 series or a calculator on the UQ approved list permitted.

During reading time (= planning time) - students are encouraged to review and plan responses to the exam questions.

This examination paper will be released to the Library.

Materials Permitted In The Exam Venue:
(No electronic aids are permitted e.g. laptops, phones)

Blank scrap paper permitted - unlimited sheets permitted

Materials To Be Supplied To Students:

None

Instructions To Students:

Additional exam materials (e.g. answer booklets, rough paper) will be provided upon request.

Answer all questions on this exam paper – questions carry the number of marks indicated.
There are 75 marks total on the exam paper.

For Examiner Use Only

Question	Mark
1	/11
2	/6
3	/4
4	/6
5	/5
6	/9
7	/8
8	/5
9	/4
10	/17

QUESTION 1 **(11 marks – 1 mark each)**

Note: quotation marks below delimit strings and names but are NOT part of the names.

(a) Write a shell command to show the names (only) of all files and directories in the "/etc" directory (which is NOT the current directory).

(b) Write a shell command to move all ".spec" files (i.e. files ending ".spec") in the "/tmp/a4data" directory to the "data" subdirectory of the home directory.

(c) Write a shell command to make a new subdirectory of the current directory called "exam".

(d) Write a shell command to compile and link an executable program called "netserver" that uses pthreads and the maths library from two C files called "threadcalc.c" and "server.c". All input and output files are in the current working directory.

(e) Write a shell command to save any changes from the subversion working copy (the current directory) to the repository.

(f) Write a shell command to show all lines in a file called "services" (in the "/etc" directory) that contain the string "tcp".

QUESTION 1 continued

(g) Write a shell command to show all lines in a file called "`resolv.conf`" in the "`/etc`" directory that do not contain the string "`nameserver`".

(h) Write a shell command that finds all lines in the file "`colours`" (in the user's home directory) that contain the string "`r=255`" but not string "`g=255`" and appends those lines to a file called "`red`" in the "`tmp`" subdirectory of the current directory.

(i) Write a shell command that counts the number of lines in a file called "`addresses`" (in the current directory) that contain the string "`Mount Cootha`".

(j) Write a shell command that counts all lines in the file "`vimrc`" (in the "`/etc`" directory) that contain the string "`set`" and appends that count value to a file called "`set.count`" in the "`vim`" subdirectory of the current directory.

(k) Write a shell command that, in the current directory, creates a symbolic link called "`a3test`" that points to "`testa3.sh`" in the "`/local/courses/csse2310/bin`" directory.

Q1

/11

QUESTION 2 **(6 marks – 1 mark each)**

Write C declarations to declare a variable "`foo`" as ...

(a) An array of five 8-bit whole numbers (i.e. 8-bit signed integers)

(b) A pointer to a function which can be used as a signal handler.

(c) A thread ID

(d) A floating point number with an initial value of 3.14

(e) An array of integers suitable for passing to the `pipe()` system call

(f) An instance of a structure that contains two members: a character and an integer.

Q2

/6

QUESTION 3 **(4 marks – 1 mark for each address)**

`moss.labs.eait.uq.edu.au` uses 4KiB pages. Suppose a process on `moss` has the following page table. All numbers are in base 10.

Page Number	Frame Number
0	-
1	51
2	52
3	-
4	44
5	22
...	
50	5641
51	3756
52	5642
53	3743
...	
8,191	102
8,192	97

For each of the following virtual addresses, what is the corresponding physical address? Write your answer in base 10. If accessing the virtual address would result in a segmentation fault, then write "SEGFAULT". If there is insufficient information in the page table (i.e. the page number is not listed) then write "UNKNOWN".

Virtual Address	Physical Address
13,456	
20,000	
60,000	
206,495	

Q3

/4

QUESTION 4 **(6 marks – 1 mark each)**

Suppose a system uses 43-bit virtual addresses and 40-bit physical addresses and uses a three-level page table. Pages are 8KiB in size. Page table entries are 8 bytes each.
A process uses the following virtual address range (all numbers are in base 10):

- 1,000 MiB starting at address 0

(a) What is the maximum memory space for a process on this system (in GiB)?

(b) How much memory (in KiB) would be needed to store the page table for this process?

(c) If the process memory usage expanded to 10,000MiB (all starting at address 0), how much memory (in KiB) would now be needed to store the page table?

QUESTION 4 continued

(d) If the original process memory usage expanded to 10,000MiB but this was 1,000MiB starting at address 0 and 9,000MiB at the highest virtual addresses, how much memory (in KiB) would now be needed to store the page table?

(e) For the original memory usage (1,000MiB starting at address 0), if the system used a two level page table, how much memory (in KiB) would be needed to store the page table?

(f) For the original memory usage (1,000MiB starting at address 0), if the system used a single level page table, how much memory (in MiB) would be needed to store the page table?

Q4

/6

QUESTION 5 **(5 marks – 1 mark each)**

Consider the following program. Assume that all system calls succeed.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main(int argc, char** argv) {
    pid_t pid;
    printf("A\n");
    if(fork()) {
        printf("B\n");
    } else {
        printf("C\n");
        pid=fork();
        printf("D\n");
        if(pid) {
            printf("E\n");
            waitpid(pid, NULL, 0);
            printf("F\n");
        } else {
            printf("G\n");
        }
    }
    if(!fork()) {
        printf("H\n");
    }
    printf("J\n");
    return 0;
}
```

(a) How many processes are created by the execution of this program (including the initial process)?

(b) What is the maximum number of processes that could be running or runnable at any point?

(c) If standard output buffers are never flushed until a process exits, how many times will the character 'D' be output?

(d) What is the minimum number of lines of text that this program will output when run?

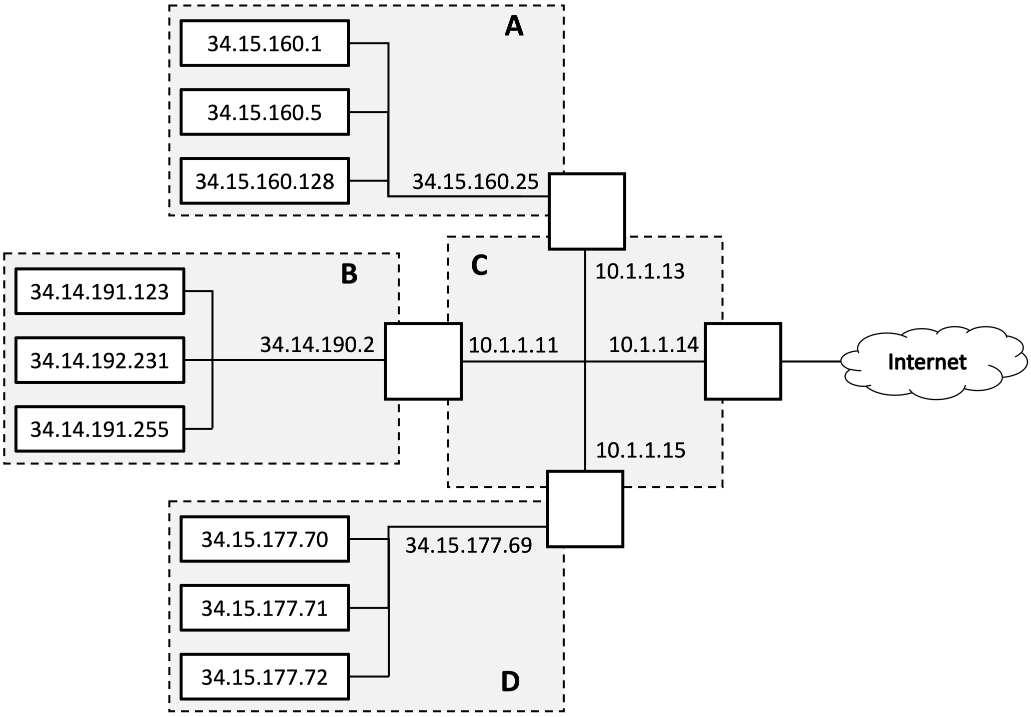
(e) What is the maximum number of lines of text that this program will output when run?

Q5

/5

QUESTION 6 **(9 marks)**

Consider the following network (assume that all networks are as small as possible):



(a) Fill in the netmask, broadcast address and CIDR for each of the networks (A to D): (6 marks)

Network	Netmask	Broadcast Address	CIDR
A			
B			
C			
D			

(b) Fill in the detail for the whole network shown above (as it would appear to the rest of the internet). (2 marks)

Netmask	Broadcast Address	CIDR

(c) How many unused addresses are there in **network D**? (i.e. how many additional machines could be added to that network) (1 mark)

Q6

/9

QUESTION 7 **(8 marks)**

Consider a "unix" file system where

- blocks are 32 KiB
- block pointers are 8 bytes
- inodes have
    - 6 direct pointers
    - 4 single indirect pointers
    - 2 double indirect pointers

(a) What is the maximum file size on this file system? Express your answer in KiB. (2 marks)

(b) What is the maximum file size that can be stored without using a double indirect pointer? Express your answer in KiB. (2 marks)

QUESTION 7 continued

(c) Assuming only the inode is cached in memory, how many blocks must be accessed to read bytes 150,000 to 200,000 (inclusive) from a file into memory? (2 marks)

(d) Some filesystems store very small files in the inode itself. What's the size of the largest file that could be stored in the inode without using a data block? State any assumptions that you make. (2 marks)

Q7
/8

QUESTION 8 **(5 marks – 1 mark each)**

Consider the following directory listing (generated by “`ls -ali`”):

```
$ ls -ali /storage/group/c2310
total 2164
5067385 drwxrwx--x   3 bob     c2310    4096 Nov 18 11:59 .
  75112 drwxr-xr-x 100 root    root     4096 Nov 18 10:43 ..
5061551 -rwxr-x---   2 bob     c2310    8200 Nov 18 10:23 admin
5068104 lrwxrwxrwx   1 alice   c2310       9 Nov 18 11:08 calc -> calculate
5067386 -rwxr-xr-x   1 alice   c2310 2190232 Nov 18 11:00 calculate
5067407 -rw-r--r--   1 bob     c2310     279 Nov 18 11:05 demo
5067392 -rw-r--r--   1 alice   c2310    1279 Nov 18 11:06 docs
5066833 drwxr-x---   5 bob     c2310    4096 Nov 18 11:59 exam
5067391 lrwxrwxrwx   1 bob     users       4 Nov 18 11:59 final -> exam
5070735 drwxrwx---   6 alice   c2310    4096 Oct 30 09:15 resources
5061551 -rwxr-x---   2 bob     c2310    8200 Nov 18 10:23 results
```

Users "`alice`" and "`bob`" are the only members of the "`c2310`" group. All users on the system are members of the "`users`" group.

(a) Within this directory, what command can `bob` run to prevent `alice` from being able to access files in the “`exam`” directory (but make no other permissions changes)?

(b) Within this directory, what command can `alice` run to allow all users other than `bob` to list the contents of the `resources` subdirectory?

(c) How many subdirectories does the `resources` subdirectory have?

(d) If the disk block size is 4096 bytes, how much disk space (in bytes) will be saved if the file `results` is removed?

(e) Which users will be able to successfully run “`ls -al /storage/group/c2310/calc`” ?

Q8

/5

QUESTION 9 **(4 marks – 1 mark each)**

A system has the following ordinary users and groups (and no others):

User	Groups
alice	admin, users, course
bob	staff, users
carol	staff, users, course
dave	staff, users
eve	course, users
fred	admin, staff, users

Consider the following directory listing:

```
-r-x--xr-- 1 alice course   138856 Nov 17 10:46 file1
--w-r-xrwx 1 dave  staff   2190232 Nov 17 11:49 file2
---xr--r-x 1 eve   course    24000 Nov 17 09:14 file3.sh
-rwx---r-x 1 fred  admin    123456 Nov 17 11:50 file4
```

List all of the users who are allowed to do the following.

(a) Read from `file1`

(b) Write to `file2`

(c) Run shell script `file3.sh`

(d) Run `file4`

Q9

/4

QUESTION 10 **(17 marks)**

Write a C program `execpipe.c` that when compiled to an executable (`./execpipe`) has the behaviour described below (13 marks). You must also write a Makefile that will build this program (4 marks).

The program is run with four or more command line arguments as follows:

```
./execpipe infile outfile cmd1 cmd2 [optional args for cmd2]
```

The contents of file `infile` are used as the standard input to command `cmd1` (found in the user's PATH). The standard output of that command becomes the standard input to command `cmd2` (also found in the user's PATH) which is run with any additional command line arguments provided. The standard output of `cmd2` is saved to the file named `outfile`. (`infile`, `outfile`, `cmd1` and `cmd2` are replaced by whichever names are given on the `execpipe` command line.)

The behaviour of the program is the same as running

```
cmd1 < infile | cmd2 [optional args for cmd2] > outfile &
```

in a shell session. (The `&` at the end of the command line indicates that this runs in the background, i.e. `execpipe` won't wait for `cmd1` and `cmd2` to finish before exiting.)

For example, running

```
./execpipe /etc/services ./out.txt sort grep -v tcp
```

is the same as running this command in the background in a shell session:

```
sort < /etc/services | grep -v tcp > ./out.txt
```

Your program does not need to do any error checking (and no marks are awarded for this). You can assume that

- 4 or more arguments will be provided to execpipe
- The given `infile` can be opened for reading
- The given `outfile` can be opened for writing
- Both `cmd1` and `cmd2` are executable programs that can be found on the user's PATH
- All system calls succeed

Your Makefile must have the following characteristics:

- Compilation and linking must be separate steps, i.e. `execpipe` is built from an object file.
- Compilation must include the C compiler flags `–std=gnu99` and `-pedantic`. (Others can be included if you wish.)
- Running make without any arguments will build `execpipe` (if required)
- Running `make clean` will remove `execpipe` and all object files

Your program and Makefile must be written to build and run on `moss.labs.eait.uq.edu.au` for marking purposes but you can omit including headers in `execpipe.c` (i.e. there is no need to show any `#include` lines).

Write your `execpipe.c` and `Makefile` into the spaces provided on the following pages.

QUESTION 10 continued

(a) execpipe.c

QUESTION 10 continued

(a) execpipe.c continued

QUESTION 10 continued

(a) execpipe.c continued

QUESTION 10 continued

(b) Makefile

END OF EXAMINATION

Q10

/17

This page is provided for rough or additional working and will not be marked unless an earlier answer explicitly refers to a continuation of an answer on this page.

This page is provided for rough or additional working and will not be marked unless an earlier answer explicitly refers to a continuation of an answer on this page.