# **CSSE2310: 2013 exam answers**

## [UQAttic](http://uqattic.net)

## Get more out of your study time. Join UQAttic's revision chat.

#### [**Other exam papers**](https://docs.google.com/folder/d/0B6_D4T6LJ-uwYzY1YWMzNjYtMzUyZC00OTEyLWJlMjktOGExYWUwOTc4NDE3/edit)

### Please contribute to these documents.

### If you're looking for an effective way to familiarise yourself with the course material, you can't go past collaborating with fellow students. We have laboured to put these up, and so at the very least point out where you think we are wrong!

### You'll get more out of the course, you'll do better in the exam, and other students will benefit from your input as well.

### To get editing permissions, simply go to the [chatroom](http://uqattic.net) and provide us with your Google Account address.

### Style.

### Type answers in blue beneath each question.

### If you're unsure of your answer, highlight your answer text then hit Ctrl+Alt+M to create a comment beside the text. Once you're satisfied with the answer, click the "Resolve" button on the comment.

### If you want some extra explanation from someone else on their answer, highlight the other person's answer and repeat the procedure above.

### Communicate.

### Head over to [uqattic.net](http://uqattic.net/) and click "Chat Now!". You'll find a chatroom full of students just like you. Talk about a revision document (like this one) or swap prep tips. If you have your own IRC client, point it to irc.uqattic.net, port 6667, channel #attic.

This document has been has yet to have been completed

Question 1) Write shell commands to do the following:

[10 marks (2 each)]

A) Delete all files with names beginning with A and ending in .c.

~~rm –rf A\*.c~~

~~rm -rf A\*\.c ‘.’ matches any character, must use ‘\’ for an actual ‘.’~~

We don’t want the flags here as that will force deletion of directories that match the pattern too. The correct answer (I think) is:

rm A\*.c

ls | grep ^A | grep \.c$ | xargs rm <- to be as careful as possible

B) Show all lines in the file stuff which start with W:

grep ^W stuff

Cat stuff | grep “^W:”

C) A file nums consists of 4 space separated columns.

Output columns 1, 3, 4 sorted by the last column.

cut -s -d’ ‘ -f1,3,4 nums | sort -n -k3

cut -d ‘ ‘ -f1,3,4 nums | s ort -k 3

OR

sort -k 4 nums | cut -d’ ‘ -f1,3,4

OR

cat nums | cut -d’ ‘ -f1,3,4 | sort -k 3

D) Create a file c.c which is a copy of b.c.

cp b.c c.c

E) For files f 1,f 2,f 3 show all lines from any of them which contain all the words “song”, “river” and

“terrible”.

Grep “song\|river\|terrible” f[123] [JB]

Grep song f1 f2 f3 | etc.

Grep song f1 f2 f3 | grep river | grep terrible

grep song f1 f2 f3; grep river f1 f2 f3; grep terrible f1 f2 f3

or

grep “song\|river\|terrible” f1 f2 f3

Question 2) Write C to declare foo as . . . : [5 marks (1 each)]

A) an array of 12 integers.

Int foo[12];

B) a pointer to a positive integer.

Int\* foo = 3;

^ I don’t think this is correct. I think the question is more asking for:

Unsigned int \*foo;

C) another name for a small integer.

Short foo;

^ I think they are asking for a typedef, not a simple short int declaration

typedef short int foo;

typedef short foo; [JB]

D) a struct containing an integer called i and a string called s.

Struct foo {

int I;

char\* s;

}; <---This is a declaration of a struct tag, not a struct.

struct {

Int i;

Char \*s;

} foo;

[JB]

E) a pointer to a function which takes two floating point values and returns a string.

char \* (\*foo) (float I, float j);

Question 3) What is the output from the following statements [11 marks (1 each)]

A)

int a=3; int b=7 printf("%d", b/a);

2

B)

int a=3;

int b=7;

printf("%d %d", b^a, b|2);

4 7

C)

int a=3;

int b=7;

printf("%d %d", a++, --b);

3 6

D)

int a=3;

int b=7;

printf("%d", a+b\*2);

17

E)

int a=3;

int b=7;

int c=(a>b)? a : b+2;

printf("%d", c);

9

F)

int a=3; int b=7;

do {

a=a+1;

b-=1;

} while(a>b);

printf("%d %d", a, b);

4 6

G)

int a=3;

int b=7;

for (int i=0; i<2; ++i )

for (int j=i;j<4; ++j ) {

if (j>2) {

break;

}

a++;

b--;

}

printf("%d %d", a, b);

5 6

Would not compile as there is no semi colon on the third line? (He forgot to compile in C99 [JB])

^^ Answer is not even possible, since a & b are in/decremented together. Correct answer is 8 2. [BM]

H)

int a=3;

int b=7;

int c=4\*(3,7);

printf("%d", c);

28 ?

I can confirm 28.

I)

int a=3;

int b=7;

int c=12

if (b>c)

c--;

b++;

if (c & b)

c-=3;

printf("%d %d", b, c);

missing ; after int c declaration but 8 9

J)

int a[]={3,4,5};

int b=7;

int\* x=a;

int c=(\*(++x))--;

printf("%d %d %d %d", c, a[0], a[1], a[2]);

4 3 3 5

K)

char a[]="world";

char\* b=a+3;

a[1]=0;

printf("%s %s", a, b);

w ld

Qestion 4) A system has 32bit virtual addresses, 4KB pages and page table entries are 4Bytes. It uses a two level page table. [6 marks (2 each)]

A) Which pages do the following (decimal) addresses belong to?

11111, 22222, 9001, 404040u

2 , 5 , 2 , 98

Consider, 4096 bytes in a page

Page = virtual address / page size

Page = 11111 / 4096 = 2.7

Page = 2

This means that the virtual address of 11111 is on the ge, because the numbering starts from 0.

Ceiling(virtual address/page size).

B) What causes page faults?

When an object/process/program is on disk but not in memory.

C) What causes segmentation faults?

access invalid page

~~access~~ write to read only page

Read from write only page [JB]

Question 5) [11 marks] Consider the following directory listing:

total 808

2244723 drwxr-xrwx 6 hermes base 4096 Sep 9 11:13 .

2228225 drwxrwxrwx 409 root root 319488 Sep 9 11:13 ..

2228804 -rw-r--r-- 1 hermes base 66 Sep 9 10:32 Makefile

2228798 -rw-r--r-- 1 hermes crew 83737 Sep 9 10:32 ass1\_spec.pdf

2228802 -rw-rw-r-- 1 hermes crew 17485 Sep 9 10:32 ass1\_spec.tex

2228908 -rw-r--r-- 1 hermes crew 54245 Sep 9 10:34 ass2spec.pdf

2228911 -r--rw-rw- 1 hermes base 3524 Sep 9 10:34 ass2spec.tex

2228914 -r--rw-rw- 2 hermes villans 18615 Sep 9 10:41 ass3\_spec.tex

2253442 drwxr-xr-- 2 hermes base 4096 Sep 9 10:46 fireflies

2228914 -rw-r--r-- 2 hermes base 18615 Sep 9 10:41 herring

2228920 -rw-r--r-- 1 hermes crew 340 Sep 9 10:32 marks

2228949 -rwxr-xr-x 1 hermes villans 749 Sep 9 10:32 mkres

2253440 drwx---r-x 2 hermes villans 4096 Sep 9 10:34 procmarks

2229358 -rwxr--r-- 1 hermes crew 224787 Sep 9 10:43 program

2228904 -rw-r----- 1 hermes base 932 Sep 9 10:33 stuff.txt

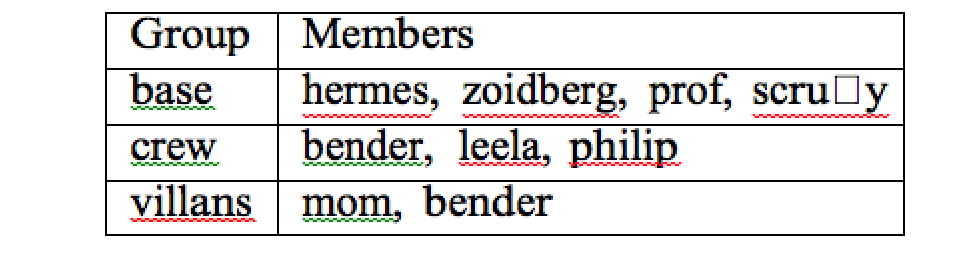
2228950 -rwxr-xr-x 1 hermes villans 19592 Sep 9 10:32 thebox

2228988 -rw-r--r-- 1 hermes uusers 9343 Sep 9 10:32 thebox.c

2228907 lrwxrwxrwx 1 hermes crew 5 Sep 9 10:38 things -> zorro

2253446 drwxr-xr-x 3 hermes villans 4096 Sep 9 11:11 toronado

2253441 drwxr-xr-x 3 hermes crew 4096 Sep 9 11:08 zorro



A) What can zoidberg do to the following: [2 marks]

• stuff.txt

read

• procmarks

Read and execute

B) Which users can modify all of the .tex files (without changing the permissions)? [2 marks]

The Crew usergroup but not hermes

C) What command(s) could mom type to execute program? [2 marks]

Assuming mom can change permissions/ownership:

Chmod o+x program

OR

Chown mom

THEN

./program

She can't as only Hermes has program execute permissions   
She could copy t he file program “cp program program2”, making her the owner of program2 and then “chmod +x program2” and execute program2, this isn’t strictly executing “program” though, only a copy of it.

D) What would change in the directory listing after hermes executed rmdir toronado

(and why?) [2 marks]

-If it was an empty directory, the directory would be deleted and the inode count will decrease.

-But, if the directory has contents within it, then it would not be deleted.

It can be seen from the reference count, which is 3, that the directory is not empty, so the directory will not be deleted and therefore nothing will be changed. Note that empty directories have a reference count of 2. [BM]

j

E) What command was used to create things? [1 marks]

Ln –s zorro things

F) Given the following commands and their output:

prompt> ls -l zorro/transport

lrwxrwxrwx 1 hermes base 21 Sep 9 11:20 zorro/transport -> ../fireflies/serenity

prompt> ls -l fireflies/serenity

-rwx--x--x 1 hermes base 1072966 Sep 9 10:46 fireflies/serenity

Can bender run ./things/transport ? (Why?)

No because bender is in a different user group, and does not have access rights.

The files have execute permissions to others, shouldn’t he be able to run it then? [+999999999999]

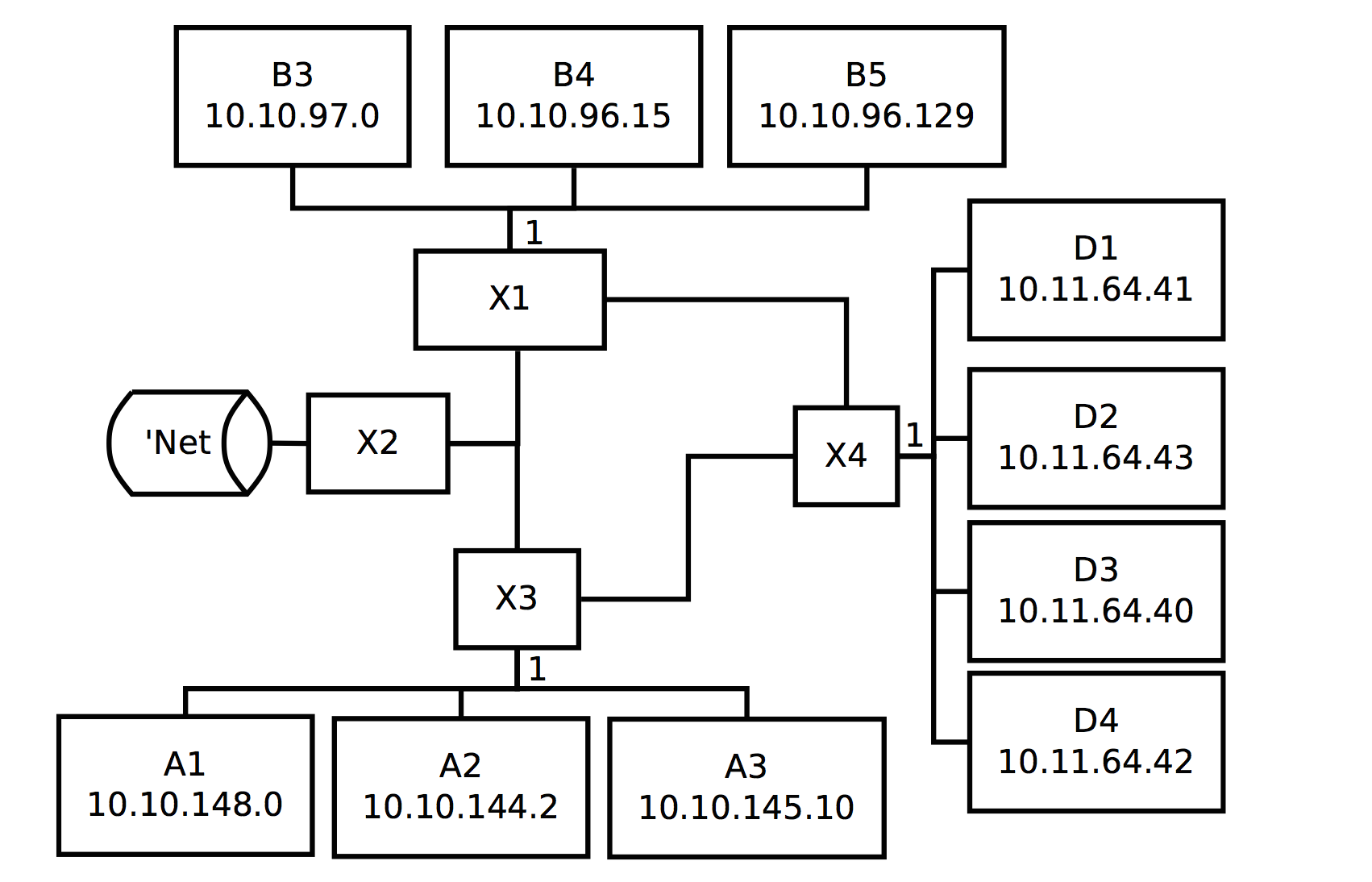
You need both read and execute to run a program. [-9999999999999999999999999999999999999]

You can execute with only ‘x’ if it’s a binary file, but you need ‘r’ to run scripts (since bash needs to read them).

But they both have read and execute in the zorro directory [+162834612893764]

Question 6) [13 marks] Consider the following network. If addresses are required but not given, you may choose an address

that makes the subnet as small as possible [in terms of number of possible hosts].



A) What are the network addresses, subnet masks and broadcast addresses for the subnets attached to the following interfaces? Use the smallest possible subnets. [5 marks]

|  |  |  |  |
| --- | --- | --- | --- |
|  | Network Address | Subnet Mask | Broadcast Address |
| X1 (interface 1) | 10.10.96.0 | 255.255.254.0 | 10.10.97.255 |
| X3 (interface 1) | 10.10.144.0 | 255.255.248.0 | 10.10.151.255 |
| X4 (interface 1) | 10.11.64.32 | 255.255.255.240 | 10.11.64.47 |

Brown = Hayden.

For X1:

So comparing these in binary we have:

These are identical for the first 7 bits (starting at the MSB). Hence the length of the subnet mask will be . Therefore the subnet mask is given by the following:

Now the network address can be found by bitwise ANDing the subnet mask with any element

from

(pick the easiest for )

Finally the broadcast address, can be determined by bitwise ORing the network address and the

bitwise inverse of the subnet mask. Hence:

For X4:

Once again, comparing these in binary yields the following:

So this one differs at the 6th (I believe I differs at the 7th, which changes all the calculations below…) bit, so the length of the subnet mask this time will be . Hence the subnet mask is:

Using the same approach above to finding the network address we come across a problem:

Notice that this network address is not unique. So the subnet mask must be modified such that it is unique. Referring back to the binary representations of and , find the first one (which will change the subnet mask) which now changes the length to . The new subnet mask will therefore be:

The correct network address will be:

The broadcast address will be:

For X3, the same process applies you should get the following:

B) What task does the bind() function perform? [1 mark]

To put it simply, bind says to the system : okay, from now on, any packet with destination {address->sun\_addr} should be

forwarded to my socket\_fd, so I can read them.

From the lecture slide “attaches a local address to a socket”

C) X2 perfoms NAT for this network. What is NAT and why is it necessary? [2 marks]

answer 1: Network Address Translation

It is necessary because method of modifying network address information in Internet Protocol (IP) datagram packet headers while they are in transit across a traffic routing device for the purpose of remapping one IP address space into another.

For example assign a company a single IP address, then use unique private IP addresses within the company, then change the private IP address back to the companies IP address when data packet leaves the network.

answer2:

what: network address translation et. NAT performs a translation from private address to public address , so computers can access the internet.

D) To which layers do the following belong: [3 marks]

|  |  |
| --- | --- |
| Term | Layer |
| Mac Address | Data link layer |
| Socket | Session layer (SSL) / Application layer |
| IP Address | Network layer - for the internet protocol |
| Port | Transport layer |
| UDP | Transport layer |
| URL | Application layer |

Possible layers:

metaphysical, application, web, physical, network, transport, gooey caramel, putty, link, wifi.

E) What is the purpose of a gateway address?

The gateway default address is a router that is the face of a local area network which sends packets out.

When you try to access [www.google.com](http://www.google.com) , your computer will try to see if that address is in the local network before talking with your router to send the packets out of the local network.

[2 marks]

Question 7) Consider a “unix” filesystem where: [8 marks (2 each)]

• i-nodes have 10-direct pointers, 1 indirect pointer and 1 double indirect pointer.

• Blocks are 8KB

• Block pointers are 4Bytes

• blocks are numbered from 0.

A) Why is fragmentation a problem for linked filesystems but not for indexed filesystems?

Indexed filesystems are sequential so adjacent blocks store data “in order”. Linked filesystems hold pointers to data, so the data can be spread (physically) across the disk.

Random access is time consuming on a fragmented linked filesystem because each block of the sequence needs to be seeked through. In an indexed filesystem, a particular block can quickly be found by checking the index.

B) How many blocks (in total) must be accessed to read the following blocks from a file: 9, 2053, 2057

1 for 9 and 2 for 2053 and 1 for 2057 ( the pointer is stored from 2053 read) so all together, a total of 4.

C) What is the maximum possible file size for this file system?

total size = (10\*8192) + (1\*2048\*8192) + (1\*2048^2 \* 8192) = 33,570,896 kB

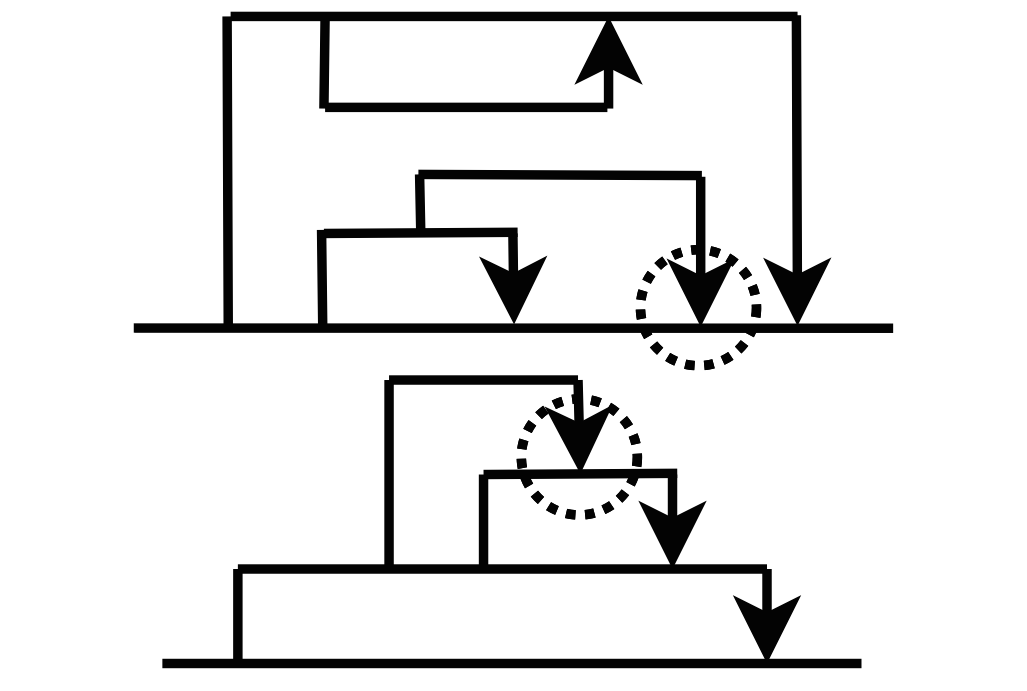
=~32.016 GiB

D) If an additional 2 double indirect pointers were added to the inodes on this filesystem, what would be the increase in maximum file size?

Increase is 2\*2048^2 \* 8192 = 67,108,864 kB

=64 GiB

Question 8) [10 marks]



A) Consider the following process fork diagrams:

The circled wait in the top diagram is possible. The circled wait in the lower diagram is not possible. Why?

if you can represent a process forking like this:



then, in the second diagram:

a new process is created.

But, the problem in the second diagram is that the child process returns to a process that hasn’t been created yet.

In the second diagram it appears that it wasn’t reaped by it’s parent, nor the init process. If we assume the downward arrow is the reaping(ending of the process). The first diagram appears to be reaped by init (as it’s parent has died)

[3 marks]

B) Which C function can be used to test if a child process has terminated? [1 mark]

waitpid()

C) Which function is used to send a signal to a process? [1 mark]

answer 1: Kill.

answer 2: exit

^^exit does not send a signal \*to\* a process, but rather exits the current process with the given exit code, so it is not a valid answer. [BM]

D) Which pthreads functions are used to perform the following tasks: [2 marks]

1. terminate a thread

answer1: pthread\_exit

answer 2: pthread\_cancel

2. retrieve the exit status of a thread.

the exit status can be obtained by another thread by calling pthread\_join().

E) Consider the following functions: [2 marks]

void\* f(void\* v) {

// X

void\* res=doThings(v);

// Y

return res;

}

Which functions could be called at X and Y to ensure that only one thread at a time executes

doThings()?

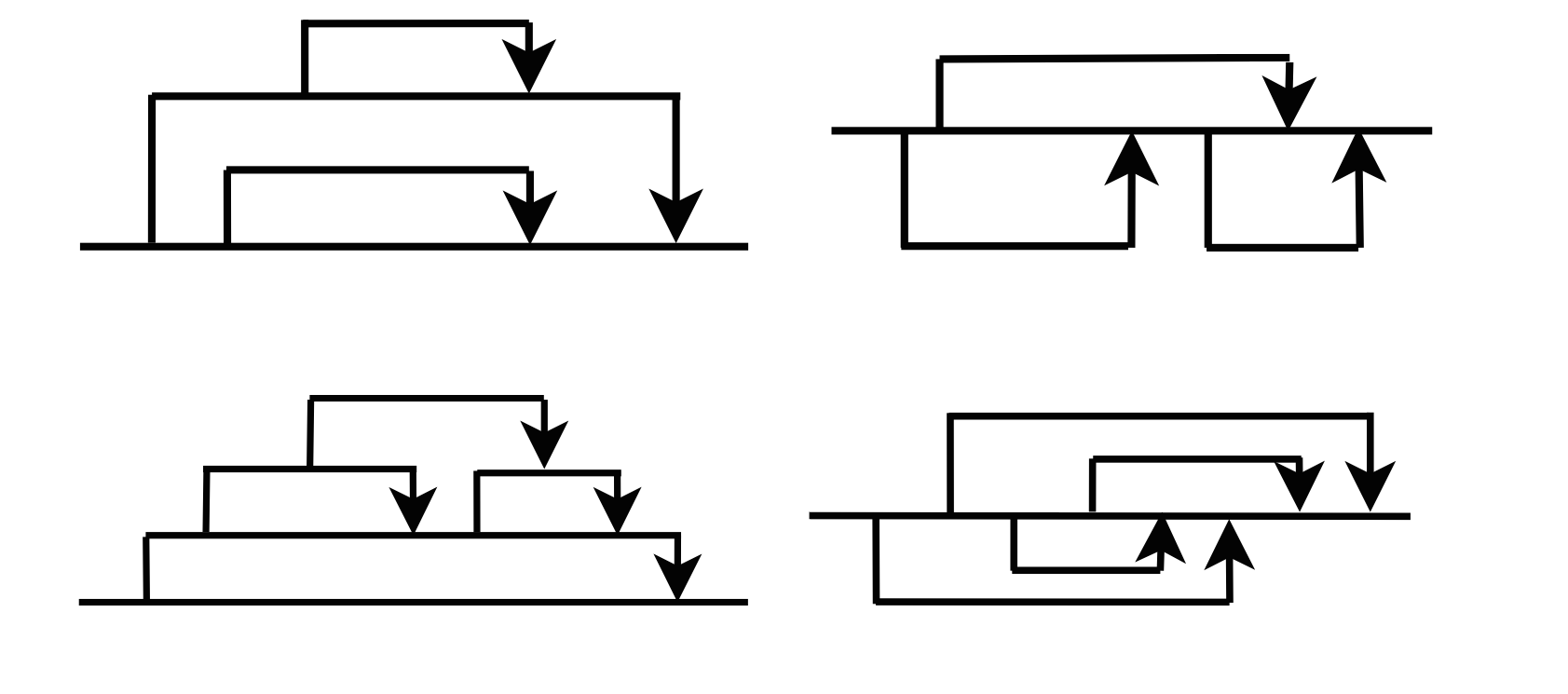
X: pthread\_mutex\_lock

Y: pthread\_mutex\_unlock

or

X: sem\_wait

Y: sem\_post



F) Which of the following pthread diagrams are possible? (Circle the possible ones) [1 mark]

All except bottom left. a thread is joined inside another thread.

Any thread can join another as long as the pthread\_t is properly passed. I believe all are possible pthread diagrams.

(See bottom of document for another possible solution to each question)

Question 9) In all of the following you may omit #includes. You may assume that all system calls succeed and that all processes exit normally. You may assume that all lines in files have 79 or fewer characters. You may assume that none of the strings to be searched for contain special charcters for grep. That is, no escaping is required. [26 marks]

A) Implement a function matchingLines which takes a string and a filename and prints (to stdout) all the lines in the file which contain the string. It should return true if the file could be opened successfully and false if not. Hint: make use of standard string functions.[4 marks]

Answer 1:

bool matchingLines(const char\* string, const char\* filename) {

This is my solution. It compiles fine, I just haven’t tested it for functionality. [Sam M.]

* changed ordering of strstr() args (they were backwards) [Matt S.]

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

bool matchingLines(const char\* string, const char\* filename) {

FILE \*fp;

char buffer[100];

memset(buffer, '\0', strlen(buffer)); //memset buffer to avoid leaks

fp = fopen(filename, "r+");

if (fp == NULL) { //check for success of opening file

return false;

}

while (fgets(buffer, 100, fp) != NULL){ //Keep reading lines while they exist

if (strstr(buffer, string) != NULL) { //Check the line for string

printf("%s", buffer);

}

memset(buffer, ‘\0’, strlen(buffer)); //reset buffer for next read from file

}

fclose(fp);

return true;

}

^^ No need to use memset to clear buffer, since it is just overwritten by fgets. [BM]

Well if they end user input with a shorter string the second time it would still contain the new line character/be seen as valid, yes?

Another option would be to add a null terminator when a new line is encountered, then it’s treated as ending there regardless.

Answer 2:

#include <stdio.h> // you can ommit include statements

#include <string.h>

#include <stdbool.h>

bool matchingLines(const char\* string, const char\* filename) {

{

char temp[79];

FILE \* f;

if ((f = fopen(filename, "r")) == NULL){

return false;

}

while (fgets(temp, 79, f) && (temp != NULL))

{

if (strstr(temp, string)){

printf("%s", temp);

}

}

fclose(f);

return true;

}

}

^^ A line could be 79 characters, so buffer needs to be able to hold 79 characters for line + 1 for terminating newline + 1 for string null terminator, so buffer needs to be able to hold at least 81 characters, not 79. [BM]

Answer 3: [BM]

bool matchingLines(const char\* string, const char\* filename) {

FILE \*fp = fopen(filename, “r”);

if(!fp) {

return false;

}

char buf[81];

while(fgets(buf, 81, fp) != NULL) {

if(strstr(buf, string) != NULL) {

printf(“%s”, buf);

}

}

fclose(fp);

return true;

}

B) Implement a function grepSearch which takes a string and a filename and prints (to stdout) all the lines in the file which contain the string. It will do this by invoking the grep program (which will be on the path). It should return true if the file could be opened successfully and false if not. You may not use the functions system(), or popen(). Hint: grep has an exit status of 2 if the file can not be opened. It will never have a status higher than 2. [4 marks]

bool grepSearch(const char\* string, const char\* filename) {

* my solution, seems to work… [matt s.]
* Possible problem: This solution doesn’t redirect stderr from grep to /dev/null if a invalid file is given to grep it will output “grep: FILE: No such file or directory”
* good point - added.

bool grepSearch(const char\* string, const char\* filename) {

int pid, childPid, status, pipeFds[2];

pipe(pipeFds);

pid = fork();

if (pid != 0) { /\* parent \*/

close(STDOUT\_FILENO); //close stdout

dup(pipeFds[1]); //make stdout pipeFds[1]

close(pipeFds[0]); //dont need

childPid = wait(&status);

if (WEXITSTATUS(status) == 2) {

return false;

}

return true;

} else { /\* child \*/

close(STDIN\_FILENO);

dup(pipeFds[0]); // make stdin pipeFds[0]

int devNull = open(“/dev/null”, O\_WRONLY);

dup2(devNull, STDERR\_FILENO);

close(pipeFds[1]); //dont need

char \*arg\_list[] = {"grep", (char \*)string, (char \*)filename, NULL};

execvp("grep", arg\_list);

}

}

^^ stdout is automatically redirected to stdout of the parent, so there is no need for redirection of file descriptors. The following solution works fine:

bool grepSearch(const char\* string, const char\* filename) {

pid\_t pid;

if((pid = fork()) == 0) { // child

k("grep", "grep", string, filename, NULL);

}

// parent

int child\_status;

waitpid(pid, &child\_status, 0);

return !(WIFEXITED(child\_status) && WEXITSTATUS(child\_status) == 2);

}

Further, there is no real requirement to redirect stderr to the bitbucket because it is not stated as a specific requirement (i.e. it is undefined). [BM]

C) Implement a function matchingLinesMany which takes a string and an array of filenames (and its size) and prints (to stdout) all the lines in all the files which contain the string. All the specified files should be processed concurrently. If any of the files result in an error, then matchingLinesMany will return false otherwise return true. Hint: You should call the function from part A. For this part, you can assume it exists and works. [6 marks]

bool matchingLinesMany(const char\* string, const char\*\* filenames, int numfiles) {

Answer 1

void\* do\_thread(void \*arg) {

char\*\* args = (char \*\*)arg;

return (void \*)matchingLines(args[0], args[1]);

}

bool matchingLinesMany(const char\* string, const char\*\* filenames, int numfiles) {

pthread\_t ids[numfiles];

for (int i = 0; i < numfiles; i++) {

char\* args[2] = {(char \*)string, (char \*)filenames[i]};

pthread\_create(&ids[i], NULL, do\_thread, (void \*)args);

}

bool success = true;

for (int i = 0; i < numfiles; i++) {

bool retVal = true;

pthread\_join(ids[i], (void\*\*)&retVal);

if (retVal == false) {

success = false;

}

}

return success;

}

^^ pthread\_t ids[numfiles] is not valid C, since numfiles is a variable. pthread\_t \*ids = malloc(sizeof(pthread\_t) \* numfiles) is (with a corresponding free at the end). [BM]

^Disagree, in C99 variable sized arrays are allowed [Acebond]

Alternative answer (confirmed working):

void \* run(void \* ar){

char \*\* args= (char \*\*) ar;

matchingLines(args[0], args[1]) ;

return NULL;

}

bool matchingLinesMany(const char\* string, const char\*\* filenames, int numfiles) {

int i = 0;

char \*\* args;

pthread\_t tid[numfiles];

for(i; i < numfiles; i++){

FILE\* file = fopen(filenames[i], “w”);

if(file == NULL) {

return false;

}

fclose(file);

args = (char \*\*) malloc(2\*sizeof(char \*\*));

args[0] = string;

args[1]= filenames[i];

pthread\_create(&tid[i], NULL, run, (void \*) args);

}

i = 0;

for(; i < numfiles; i++){

pthread\_join(tid[i], NULL);

}

}

^^ Does not meet requirement to return true/false based upon any files resulting in errors. [BM]

D) Implement a function grepSearchMany which takes a string and an array of filenames (and its size) and prints (to stdout) all the lines in all the files which contain the string. All the specified files should be processed concurrently. If any of the files result in an error, then grepSearchMany will return false otherwise return true. Hint: You should call the function from part B. For this part, you can assume it exists and works. [6 marks]

bool grepSearchMany(const char\* string, const char\*\* filenames, int numfiles) {

bool grepSearchMany(const char\* string, const char\*\* filenames, int numfiles) {

int childPid, status, pids[numfiles], pipefds[numfiles][2];

for (int i = 0; i < numfiles; i++) {

pipe(pipefds[i]);

pids[i] = fork();

if (pids[i] != 0) {

//parent

dup(pipefds[i][1]);

close(pipefds[i][0]);

} else {

//child

close(STDIN\_FILENO);

dup(pipefds[i][0]);

close(pipefds[i][1]);

if (grepSearch(string, filenames[i])) {

exit(0);

} else {

exit(2);

}

}

}

bool success = true;

for (int i = 0; i < numfiles; i++) {

childPid = wait(&status);

if (WEXITSTATUS(status) == 2) {

success = false;

}

}

return success;

}

^^ int pids[numfiles] is not valid C, since numfiles is a variable. int \*pids = malloc(sizeof(int) \* numfiles) is (with a corresponding free at the end). [BM]

Can anyone confirm is the above ^^ is true, I thought it was fine to use variables in that manor as long as they are not returned?

It’s fine, we’re using C99. Google ‘variable length arrays’ for details.

E) Write a program which takes the following arguments: [6 marks]

• The character M or G.

• a string

• a sequence of filenames to process.

It will output all the lines in any of the files which contain the string. It will do this by calling

matchingLinesMany (if the character is ‘M’) or grepSearchMany (if the character is ‘G’). For example:

./a.out M fish jungle ocean aquarium

Would search the files jungle, ocean, aquarium for lines containing fish.

For this part, you may assume that matchingLinesMany() and grepSearchMany() exist and function correctly.

|  |  |  |
| --- | --- | --- |
| Condition | Exit status | Message (to stderr) |
| No errors | 0 |  |
| Incorrect number of parame-  ters or invalid character | 1 | Bad params. |
| matchingLinesMany() or  grepSearchMany() returns  false | 2 | Bad file. |

int main(int argc, char\*\* argv) {

int main(int argc, const char \*\*argv) {x

if (argc < 4) {

fprintf(stderr, "Bad Params.");

return 1;

}

if (strcmp(argv[1], "M") == 0) {

if(!matchingLinesMany(argv[2], argv + 3, argc - 3)) {

fprintf(stderr, "Bad File.");

return 2;

}

} else if(strcmp(argv[1], "G") == 0) {

if(!grepSearchMany(argv[2], argv + 3, argc - 3)) {

fprintf(stderr, "Bad File.");

return 2;

}

} else {

fprintf(stderr, "Bad Params.");

return 1;

}

return 0;

}

Guthers’s (on UQCS slack) attempted answers (compiles - behaves how I think it should)

Change it if you have a problem with them

(A note on question E - as I was using main for testing I had to improvise so it just passes argv and argc to the function E() for the answer to question E)

|  |
| --- |
| #include <stdlib.h>  #include <stdio.h>  #include <stdbool.h>  #include <string.h>  #include <unistd.h>  #include <sys/types.h>  #include <sys/wait.h>  #include <pthread.h>  //Question 9 A  bool matchineLines(const char\* string, const char\* filename) {  FILE\* file = fopen(filename, "r");  if (file == NULL) {  return false;  }  char\* buffer = malloc(sizeof(char)\*80);  while(fgets(buffer, 80, file) != NULL) {  //check if string is in the like  if (strstr(buffer,string) != NULL) {  printf("%s", buffer);  }  }  return true;  }  //Question 9 B  bool grepSearch(const char\* string, const char\* filename) {  pid\_t pid = fork();  if (pid == 0) {  //Child  execlp("grep", "grep", string, filename, NULL);  }  int status;  waitpid(pid,&status, 0);  if (WEXITSTATUS(status) == 2) {  return false;  }  return true;  }  //Question 9 C - lets hope A works  void\* matchineLines\_thread(void\* arg) {  char\*\* args = (char\*\*)arg;  return (void \*)matchineLines(args[0],args[1]);  }  bool matchineLinesMany(const char\* string, const char\*\* filenames, int numfiles) {  pthread\_t TIDs[numfiles];  for (int i=0; i<numfiles; i++) {  const char\* args[2] = {string,filenames[i]};  pthread\_create(&(TIDs[i]),NULL, &matchineLines\_thread, (void\*)args);  }  for (int i=0; i<numfiles; i++) {  bool result;  pthread\_join(TIDs[i], (void\*)&result);  if (!result) {  return false;  }  }  return true;  }  //Question 9 D  bool grepSearchMany(const char\* string, const char\*\* filenames, int numfiles) {  pid\_t pids[numfiles];  for (int i=0; i<numfiles; i++) {  pids[i] = fork();  if (pids[i] == 0) {  //child  execlp("grep", "grep", string, filenames[i], NULL);  }  }  for (int i=0; i<numfiles; i++) {  int status;  waitpid(pids[i],&status, 0);  if (WEXITSTATUS(status) == 2) {  return false;  }  }  return true;  }  //Question 9 E - dosent need to be done with a struct, but makes for a good example  typedef struct {  char func;  const char\* string;  const char\*\* filenames;  int numfiles;  } argStruct;  void\* E\_Thread(void\* arg) {  argStruct\* as = (argStruct\*)arg;  if (as->func == 'M') {  matchineLinesMany(as->string,as->filenames,as->numfiles);  } else if (as->func == 'G') {  grepSearchMany(as->string,as->filenames,as->numfiles);  }  }  int E(int argc, char \*\*argv) {  if ((argc < 2) || (strlen(argv[1]) != 1)){  return 1;  }  char\* funcStr = argv[1];  char func = funcStr[0];  if ((func != 'M') && (func != 'G')) {  return 1;  }  char\* filenames[argc - 3];  for (int i=3; i< argc; i++) {  filenames[i-3] = malloc(sizeof(argv[i]));  strncpy(filenames[i-3],argv[i],sizeof(argv[i]));  }  argStruct\* as = malloc(sizeof(argStruct));  as->func = func;  as->string = argv[2];  as->filenames = filenames;  as->numfiles = (argc - 3);  pthread\_t tid;  pthread\_create(&tid,NULL,&E\_Thread, (void\*)as);  bool result;  pthread\_join(tid,(void\*)&result);  if (!result) {  return 2;  }  return 0;  }  int main(int argc, char \*\*argv) { //Mainly used to debug other questions  //for(int i=0; i<argc; i++) {  // printf("(%d) = %s\n", i, argv[i]);  //}  if (argc == 1) {  //Q9A  bool Q9A1, Q9A2, Q9A3;  Q9A1 = matchineLines("bar","Q9A.txt");  Q9A2 = matchineLines("foo","Q9A.txt");  Q9A3 = matchineLines("bar","FAIL");  printf("Q9 A1 Expected Result = true | Actual = %s\n", Q9A1?"true":"false");  printf("Q9 A2 Expected Result = true | Actual = %s\n", Q9A2?"true":"false");  printf("Q9 A3 Expected Result = false | Actual = %s\n", Q9A3?"true":"false");    //Q9B  bool Q9B1, Q9B2, Q9B3;  Q9B1 = grepSearch("bar","Q9B.txt");  Q9B2 = grepSearch("foo","Q9B.txt");  Q9B3 = grepSearch("bar","FAIL");  printf("Q9 B1 Expected Result = true | Actual = %s\n", Q9B1?"true":"false");  printf("Q9 B2 Expected Result = true | Actual = %s\n", Q9B2?"true":"false");  printf("Q9 B3 Expected Result = false | Actual = %s\n", Q9A3?"true":"false");  //Q9C  const char\* fileArr1[2] = {"Q9A.txt", "Q9B.txt"};  const char\* fileArr2[2] = {"Q9A.txt", "FAIL"};  bool Q9C1, Q9C2, Q9C3, Q9C4, Q9C5, Q9C6;  Q9C1 = matchineLinesMany("bar",fileArr1,1);  Q9C2 = matchineLinesMany("foo",fileArr1,1);  Q9C3 = matchineLinesMany("bar",fileArr1,2);  Q9C4 = matchineLinesMany("foo",fileArr1,2);  Q9C5 = matchineLinesMany("foo",fileArr2,1);  Q9C6 = matchineLinesMany("foo",fileArr2,2);  printf("Q9 C1 Expected Result = true | Actual = %s\n", Q9C1?"true":"false");  printf("Q9 C2 Expected Result = true | Actual = %s\n", Q9C2?"true":"false");  printf("Q9 C3 Expected Result = true | Actual = %s\n", Q9C3?"true":"false");  printf("Q9 C1 Expected Result = true | Actual = %s\n", Q9C4?"true":"false");  printf("Q9 C2 Expected Result = true | Actual = %s\n", Q9C5?"true":"false");  printf("Q9 C3 Expected Result = false | Actual = %s\n", Q9C6?"true":"false");  //Q9D  bool Q9D1, Q9D2, Q9D3, Q9D4, Q9D5, Q9D6;  Q9D1 = grepSearchMany("bar",fileArr1,1);  Q9D2 = grepSearchMany("foo",fileArr1,1);  Q9D3 = grepSearchMany("bar",fileArr1,2);  Q9D4 = grepSearchMany("foo",fileArr1,2);  Q9D5 = grepSearchMany("foo",fileArr2,1);  Q9D6 = grepSearchMany("foo",fileArr2,2);  printf("Q9 D1 Expected Result = true | Actual = %s\n", Q9D1?"true":"false");  printf("Q9 D2 Expected Result = true | Actual = %s\n", Q9D2?"true":"false");  printf("Q9 D3 Expected Result = true | Actual = %s\n", Q9D3?"true":"false");  printf("Q9 D1 Expected Result = true | Actual = %s\n", Q9D4?"true":"false");  printf("Q9 D2 Expected Result = true | Actual = %s\n", Q9D5?"true":"false");  printf("Q9 D3 Expected Result = false | Actual = %s\n", Q9D6?"true":"false");  } else {  int result = E(argc,argv);  printf("E returned code %d\n", result);  }  } |