**CSSE2310: 2020 SEM2 exam answers**

[**UQAttic**](https://uqattic.net)

[**Other exam papers**](https://drive.google.com/drive/folders/1i3euI6FbNv_u3fOmAtlMRzZrSUb5vYXf?usp=sharing)

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**, visit [UQAttic.net](https://uqattic.net) and provide us with your Google Account address by clicking on the “Get editing permissions” button on the front page.

**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.

1)

ls ~/work

Wrong question? Did “ls /tmp/data/”

2)

cp /tmp/\*.h ./assign2/

Different answer: cp -r /tmp/\*.c ./csse2310 (tested on moss, -r is unnecessary because of ‘\*.c’)

3)

mkdir orange

4)

gcc -o doit -pthread docalc.c

5)

svn update

6)

grep ‘cat’ creatures

Different answer: cat animals | grep dog

OR

grep dog animals

7)

grep -v ‘eagle’ birds

8)

grep ‘inet’ /etc/netconfig | grep -v ‘inet6’ >> tmp/nets

9)

grep ‘Toowong’ addresses | wc -l

10)

grep ‘Idap’ /etc/nsswitch.conf | wc -l > /tmp/Idap.count

11)

ln -s lemon citrus

Different answer:

ln -s mac apple

12)

a)

Unsigned int foo[7];

b)

char\* (\*foo)(int, int ,int)

c)

volatile char foo;

d)

double foo;

This should be: double \*foo; It’s asking for a pointer to a double

e)

bool foo[3];

f)

void (\*foo)(int);

13)

8192)

8192/ 8Kib = page 1 -> Frame 30, no offset

Frame 30 \* 8Kib = 245760?

Yes [+2].

20480)

20480 / 8Kib = page 2(.5) -> SEGFAULT

48326)

48326 / 8Kib = page 5(.8) -> 23

23 \* 8Ki.b = 188416

offset = 0.8\*8Kib = 7365

Answer = 188416 + 7365 = 195782 [+1]

86123)

86123 / 8Kib = page 10(.5) -> UNKNOWN

Yes [+2].

14)

Pages = 2^12, therefore offset = 12bits

Entries per page = 2^12 / 2^3 = 2^9 -> 3rd level = 9 bits

2nd level = how many it takes to index into 3rd = 9bits

Top = whatever's left = 9bits

i)

Max memory size fills entire virtual address space

Therefore 39 bits, 2^39

Get in GB, 2^39 / 2^30 = 2^9 Gb

ii)

2^9 entries per page, page size = 4Kib

3rd level size = 2^9 \* 4Kib = 2048Kb = 2Mib

2nd level size = 2^9 \* 2Mib = 1024Mb = 1 Gb

1200 / 2 Mib -> 600 3rd level page tables needed.

512 = max 2nd level, need 600 ----> 2 2nd level pages needed

1 Top level page

(1 + 2 + 600) \* 4 Kib is the total memory usage

603 \* 4 = 2412 Kib

iii)

2400 / 2 Mib -> 1200 3rd level pages needed

512 = max 2nd level need 1200 ---> 3 2nd level pages needed

1 Top level

1200 + 3 + 1 \* 4Kib = 1204 \* 4 = 4816 Kib

iv)

Highest possible memory address = 2^39 - 1200mib

= 5.48x10^11

5.48x 10^11 / pagesize (4Kib) = whole number (1339...blah)

We know that the address starts at a whole page index, meaning its essentially the same as the 0 index when it comes to calculating its size (addresses starting at partial pages can screw things up i think?)

Therefore its just the same as ii) except doubled (top level stays the same)

(1+4+1200) \* 4 Kib = 4820 Kib

v)

Two level page table -> offset is still 12 bits

2nd level is still 9 bits -

1st level is 9 + whatever's left = 18 bits

2^9 entries per page, page size 4 Kib

2nd level size = 2^9 \* 4 Kib = 2 Mib

We need 1200 / 2 = 600 Mib 2nd level page tables

Need 600 2nd level page tables

Also need to store the gigantic 2^18 top level

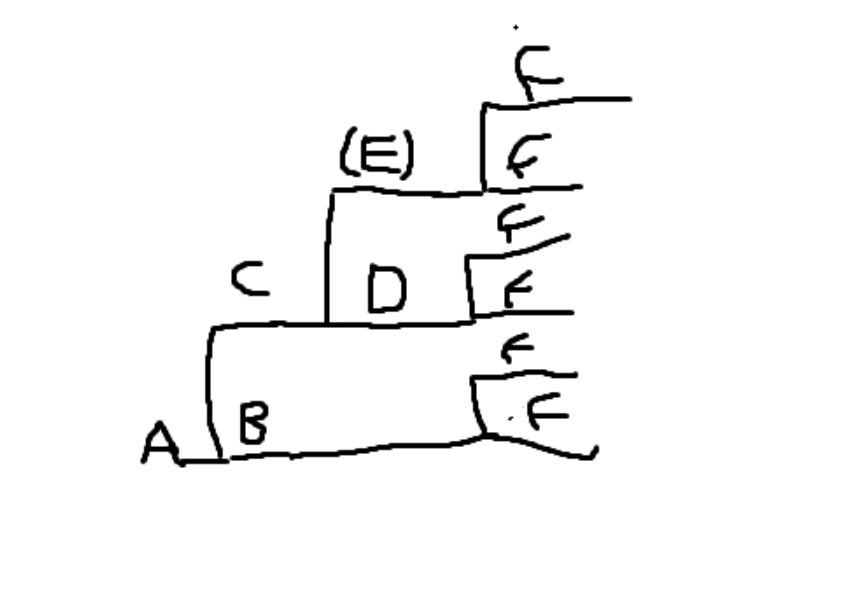
Get size of top level: 2^18 entries \* size of entries = 2^18 \* 8 Bytes = 2097152, / 1024 = 2048 Kib

600 \* 4 Kib + 2048 Kib

4448 Kib.

vi)

2^27 \* 8 Bytes / 1024^2 = 1024 MiB



15)

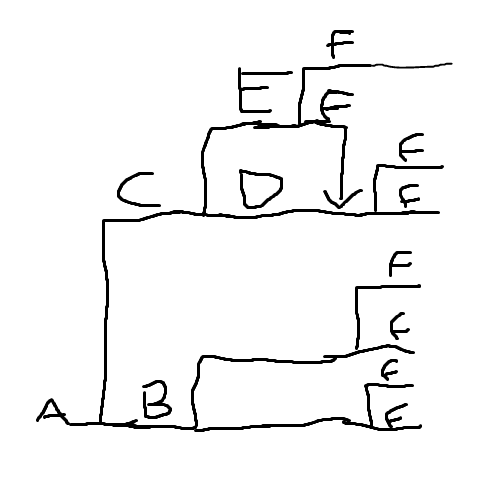
This is what I got for the forking diagram

1. False, bottom tree could finish before middle
2. True?? - someone confirm this

False. D is printed on non-buffered stream. [+1]

False - we fflush stderr BEFORE stdout

1. True, could be pushed past fork and printed twice
2. True {+1}
3. False, top/middle tree could finish before bottom

16)

This is what I got

1. 8
2. 7
3. Count up all the branches past C --> 4
4. 13, just count all the letters
5. Same as c) but for all the letters

A - 8, b - 4, c - 4, d - 2. E -2. F - 8

Total = 8+4+4+2+2+8 = 28

17)

| Network | Netmask | Broadcast Address | CIDR |
| --- | --- | --- | --- |
| A | 255.255.255.240 [+4] - this is wrong this would mean that there is an all one case to the right of the split  255.255.255.192 [+2] | 17.5.134.31 [+2]  17.5.134.63 [+2] | 17.5.134.16/28 [+3]  Corrected after broadcast address error:  17.5.134.0/26 [+2] |
| B | 255.255.254.0 [+4] | 17.5.0.0 /23  17.5.126.0  17.5.127.255 [+4] | 17.5.1.255  17.5.126.0/23 [+3] |
| C | 255.255.248.0 | 17.5.7.255 | 17.5.0.0/21 |
| D | 255.255.248.0  255.255.255.248? [+2] | 10.1.1.8 [-1000]  10.1.1.15 /29 [+3]  (Need to do bitwise NOT on subnet then OR with last octet of network)  I thought we just keep the address up until the host bits and set them all to one - same thing. Cool cool just checked your answers right  That genuinely surprises me. | 10.1.1.8/29 |

1. Find all common bits in network

23 = 00010111, 25 = 00011001, 27 = 00011011, Is always 0001xxxx

28 COMMON BITS

31 = 00011111 <- this is an all 1’s address, need to increase total by 1 bit.

27 COMMON BITS

But now, if we calculate the broadcast address, its still .31, which is still used by the network

Increase total by 1 bit again-> 26 COMMON BITS

The broadcast address is now .63 -> we’re safe

CIDR = 17.5.134.00000000 / 26 (keep all bits until host bits, set them to 0)

Broadcast = 17.5.134.00011111 (keep all bits until host bits, set them to 1)

Netmask = 255.255.255.11110000 (set all network bits to 1, set host bits to 0)

1. 17.5.01111110.45

17.5.01111111.193 // if end was .255, we’d have to increase to 22 bits (all 1s address)

17.5.01111110.193

Up to 23 bits are common

1. 17.6.00000001.2

17.6.00000011.4

17.6.00000101.6

00000xxx is common

Up to 21 bits are common

1. 10.1.1.00001011

10.1.1.00001100

10.1.1.00001101

10.1.1.00001110

00001xxx is common

Up to 29 bits are common **Why only 21 common bits?**

Whoops counted wrong - just to confirm you got 29 when you counted again? Yeah, 32 - 3 bits. Was confused because of C), just thought it was the same as that but that also includes another 8 bits on the end.

Thank fack ill correct c as well now, did the broadcast address wrong

Do you know how to do network translation for 17 part 2? They’re all private ip addresses but you need to answer how they appear to the internet, idk how

17 part 2)

Don’t know how to do this - the 10. Addresses are private so what exactly is the broadcast address?

Same as the method for ABCD, except you ignore the network D because it is 10.0.0.0 (private network)

| Network | Netmask | Broadcast Address | CIDR |
| --- | --- | --- | --- |
| Internet | 255.252.0.0 | 17.7.255.255 | 17.4.0.0/14[+1] (I had /13) |

17 part 3)

In part A theres 32 - 28 = 4 host bits. 3 of those are already taken up, and theres 2 addresses we can’t use. So 2^4 -5 = 11 more machines

<Including 17.5.134.31 > I got 2^(32 - 28) - 4 - 2 = 10 Machines [+2]

Aren’t there 26 network bits, so 6 host bits. 3 devices + 1 router + 2 for broadcast etc.

64-6=58

18)

Max file size = all pointers are full

Direct = 7 \* 4 Kib

Number of pointers per block = 4 Kib / 8 = 512 pointers

Single indirect = 3 \* 512 \* 4 Kib

Double indirect = 2 \* 512^2 \* 4Kib

Total Size = (7 + 3 \* 512 + 2 \* 512^2) \* 4 Kib

= 525831 \* 4096 = 2153803776 bytes [+2]

19)

All pointers up to double are filled

Direct = 7 \* 8 Kib

P per block = 8 Kib / 8 = 1024 pointers

Single Indirect = 4 \* 1024 \* 8 Kib

Total size = (7 + 4\*1024)\*8Kib = 32824Kib

Isnt it a different file system with 6 direct pointers?

Direct = 6 \* 8 KiB

Single Indirect = 4 \* 1024 \* 8 KiB

Total Size = 32816 KiB [+3]

20)

What block number is that? 237741 / 4 Kib = Block number 58

Split blocks into numbers

Direct = 0 - 5

Pointers per block = 4096 / 8 = 512

Single = 6 - 517

Therefore it accesses 1 single indirect block, which leads to a direct block.

Therefore 2 accesses ? (+2)

Same as above but should be 237741 / 16KiB = Block no. 19

So should be single indirect which is 2 accesses

I believe it is 1 access since it is in RAM (-2)

I would assume that it is the same as the example since the inode is not a block per say

21)

1. Chmod g-x generate
2. Chmod o+r data

Chmod g-r data

Chmod o+r g-r data // this fails on my machine - ‘cannot access ‘g-r’ no such file.

Chmod o+r data ; chmod g-r data

Chmod o+r, g-r data. Alternatively, Chmod 714 data (if you feel fancy). [+2]

1. Num of links = 4 (I see 5 in the q) - 2 for . and .., therefore 2 subdirectories, right?

5 - 2 = 3 ? [+2]

1. Docx is a hard link, as it has 2 links to it. Same inode number as doc2

ln doc2 docx

1. Docz is a soft link, so use the -s option to the ln command  
   ln -s docs docz

→ I agree with these commands, but arent they all called ‘file’?

22)

1. Alice + bob [+2]
2. Dave + bob + carol + eve

Dave Bob Carol [+2] Eve is project so she can’t

1. Is it a standard executable or a shell script???

Eve Carol Bob [+1]

23)