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

**CSSE2310: 2022 SEM1 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.

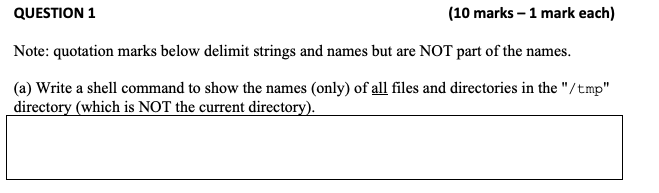
**Style.**

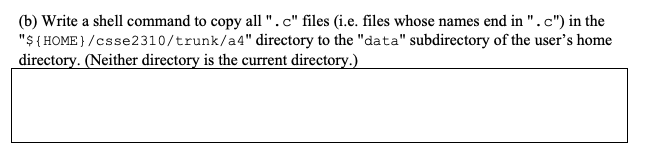
Type answers in blue beneath each question.

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.

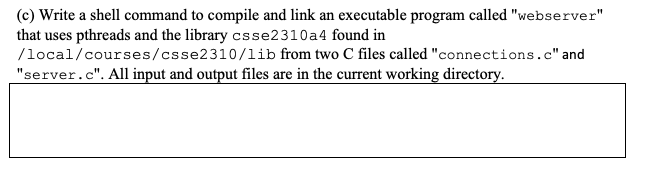
If you want some extra explanation from someone else on their answer, highlight the other person's answer and repeat the procedure above.

Feel free to leave a comment if you think the provided answer for a question is incorrect as well.



ls -a /tmp +1`

cp ~/csse2310/trunk/a4/\*.c ~/data [+2]



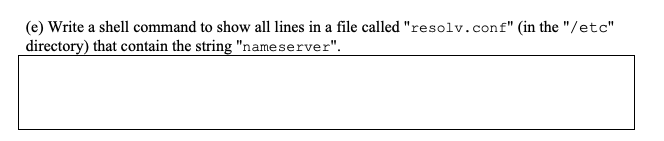
gcc -pthread -std=gnu99 -I /local/courses/csse2310/include -o webserver connections.c server.c -L /local/course/csse2310/lib -lcsse2310a4 [+1]

Above answer is correct, but the “-std=gnu99” isn’t necessary.

Also don’t need “-I” since it is not specified.

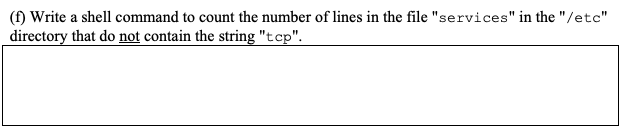


svn commit -m “Fixed bug 1234” [+1]



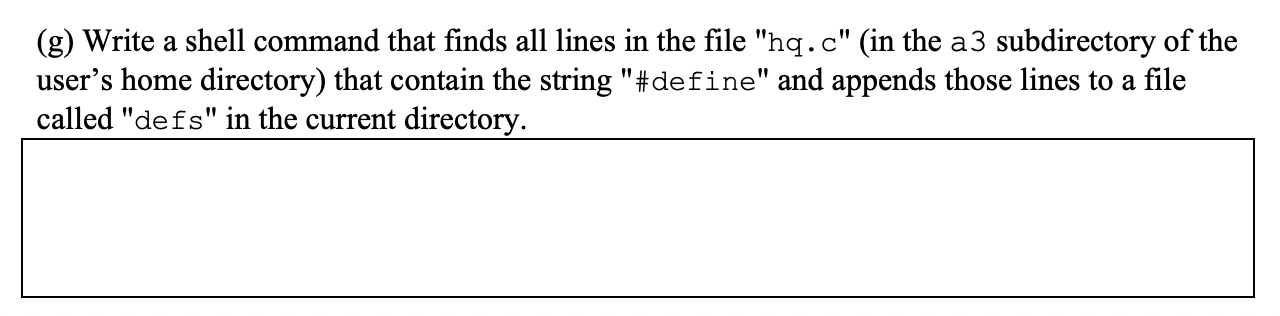
grep “nameserver” /etc/resolv.conf [+2]

cat /etc/resolv.conf | grep nameserver [+1]

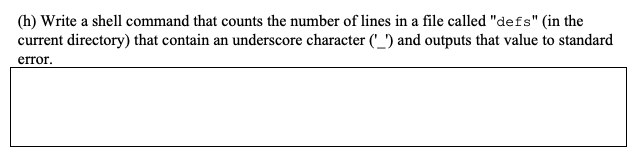


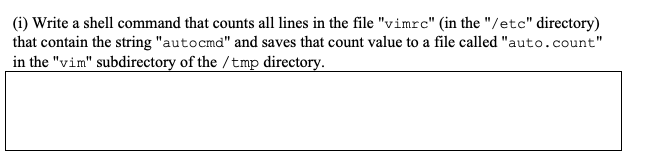
grep -c -v “tcp” /etc/services +1

grep -cv tcp /etc/services <- same thing, another way of writing it



grep “#define” ~/a3/hq.c >> defs [+2]



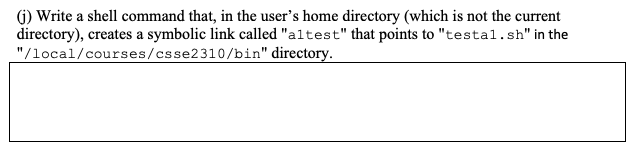
grep -c “\_” defs >&2

grep -c “autocmd” /etc/vimrc > /tmp/vim/auto.count

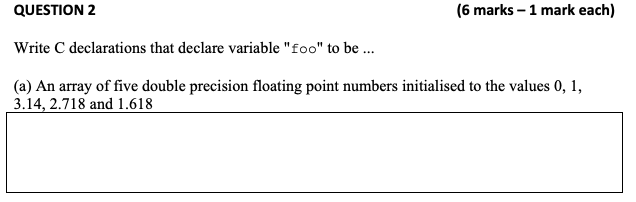
grep autocmd /etc/vimrc | wc -l > /tmp/vim/auto.count [+1] (maybe)

Both work

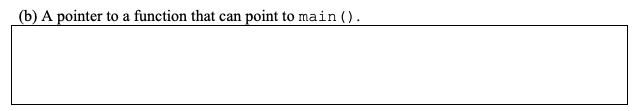
Could we also use >> to append it to a file instead of overwriting? It does not speciyfy so I reckon both can be used?



ln -s /local/courses/csse2310/bin/testa1.sh ~/a1test [+2]

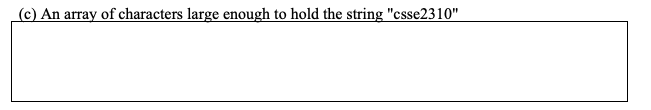


double foo[] = {0, 1, 3.14, 2.718, 1.618}; or double foo[5] = {0, 1, 3.14, 2.718, 1.618};



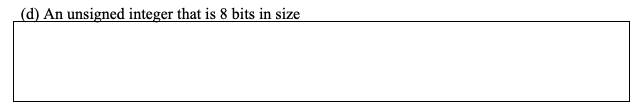
int (\*foo) (void);

Int (\*foo) (int, char\*\*) [+2]



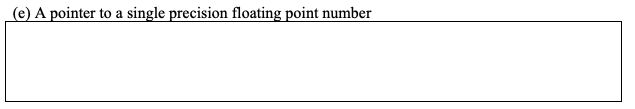
char foo[9] = “csse2310”; (-1) question asks for an array large enough, not initialised

char foo[9]; (+4)

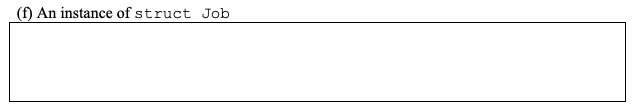


unsigned char foo;

uint8\_t foo; also works



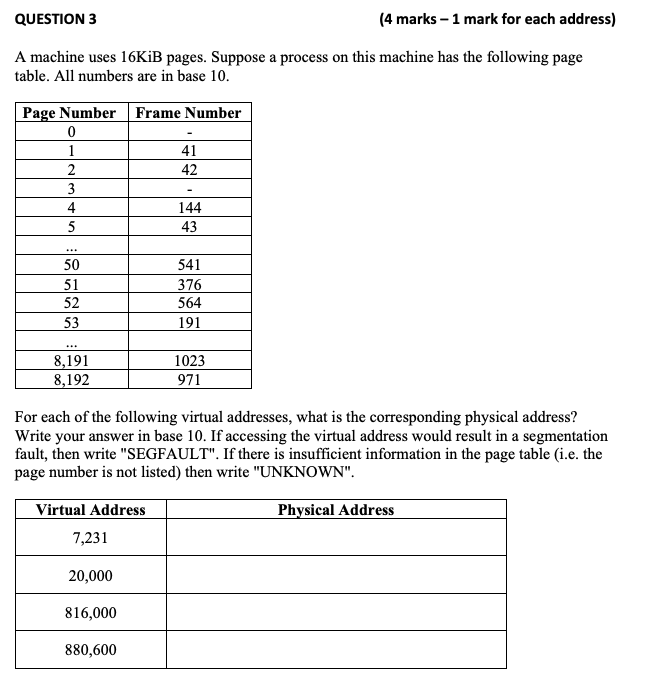
float\* foo;



struct Job foo; [+1]【+1】

Job foo;

# Don’t think can assume that “struct Job” is type aliases (typedef).

Q1 & 2 confirmed by a tutor on the cs society’s discord

b

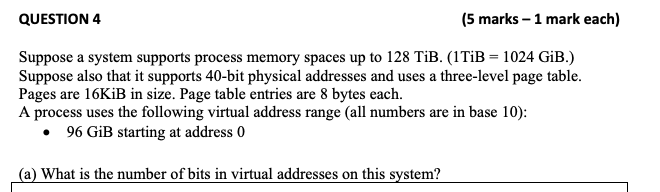
7231: SEGFAULT [5]

20000: 675360 [+6]

816000: UNKNOWN [+4]

880600: 3141592 [+6] woah its pi [+2]

Could someone explain this? Lecture 12 has a worked through example of this probably the contact, thanks



47 (11 +11 +11+14 but I’m unsure) (+4)

128 TiB = 128 \* 2^40 B (1 TiB = 2^40 B) = 2^7 \* 2^40 B = 2^47 B => ans is 47 bits (I agree with the above)

Answer is indeed 47, this question was discussed in one of the consultation session (-200000)



Entries per page = 16 KiB / 8 B = 2 Ki = 2048 entries/page

3rd level size = 2048 \* 16 KiB = 32 MiB

2nd level size = 2048 \* 32 MiB = 64 GiB

Num 3rd level page = 96 GiB / 32 MiB = 3072 pages

Total mem = (1 + 2 + 3072) \* 16 KiB = 49200 KiB (= (#Lvl 1 pages + #Lvl 2 pages + #Lvl 3 pages) \* (Page Size) [+5]

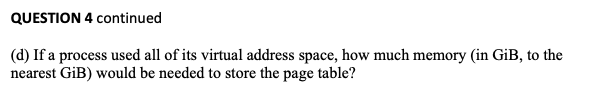
1 lvl 3 + 2 lvl 2 + 3072 pages

How come 1 + 2? One table has 2048 entries, so to fit 3072 3rd lvl tables, we need 2 2nd lvl tables.

Num 3rd level page = 256 GiB / 32 MiB = 8192 pages

Now need 4 2nd level pages to store references

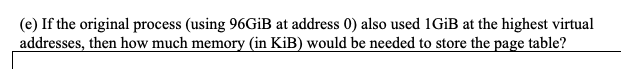
Total mem = (1 + 4 + 8192) \* 16 KiB = 131 152 KiB (+4)

Why does it need 4 2nd level pages to store references? 8192/2048 = 4

according to exam review session - 64GiB

(1 + 2048 + 2048^2) \* 16KiB f= 64GiB

At level 1, just one page, points to 2048 pages at level 2, which each point to 2048 pages at level 3. So, 1 + 2048 + 2048^2 pages. Then multiply by page size to get page table size in bytes

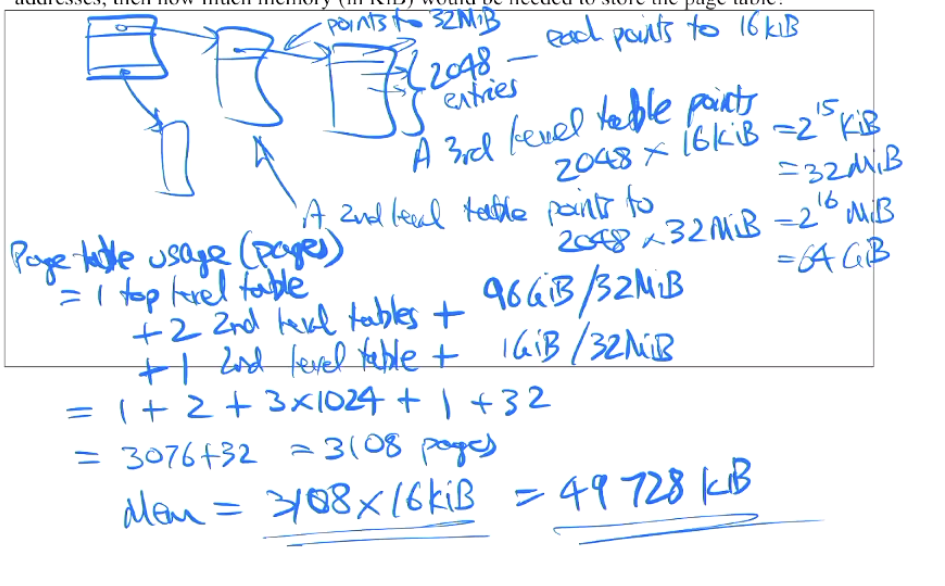


?

I don’t think storing the memory at the highest virtual addresses changes anything special.

The 1GiB of memory will need 2^30 / 2^14 = 2^16 frames to store it in.

These frames need 2^16 / 2^11 = 2^5 lvl 3 page tables

From b), we originally had 3072 lvl 3 page tables . We now have 3072 + 2^5 = 3104 lvl 3 page tables.

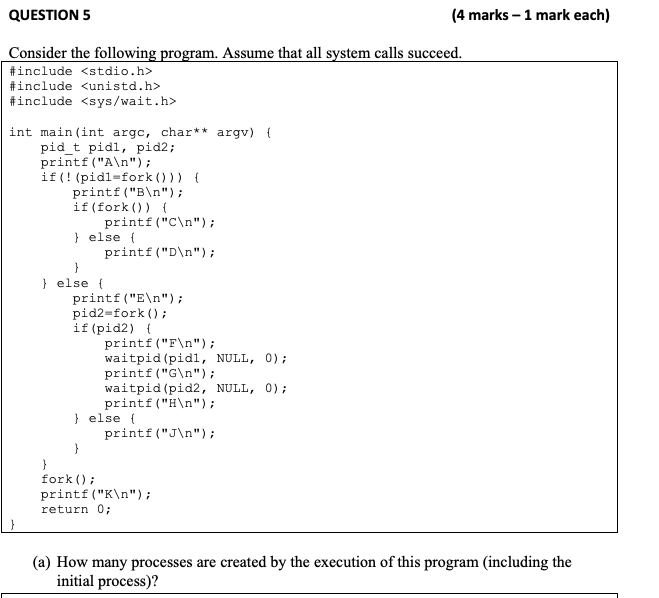
These lvl 3 page tables need 2 lvl 2 page tables to store them (This line is where you went wrong. Although we can technically fit the extra lvl 3 page tables into the two existing lvl 2 page tables, we actually need to add a new third lvl 2 page table to store them because they are at the highest address. See Exam Session 1 - 29:53 minutes in)

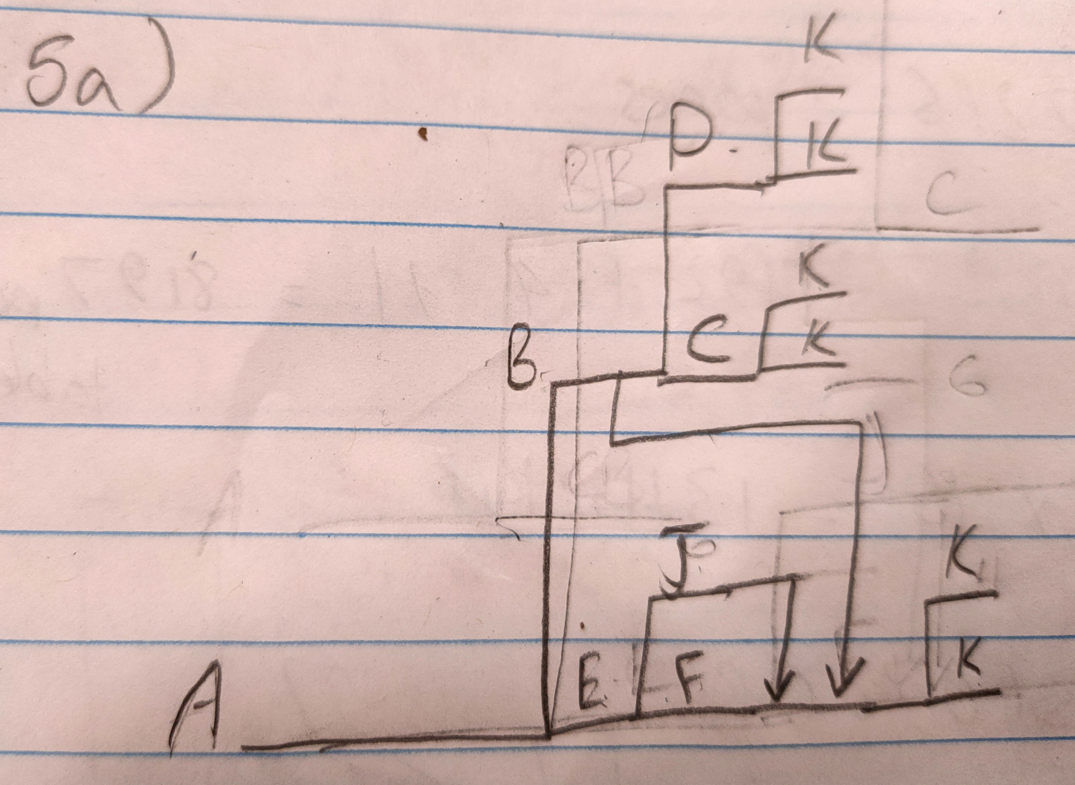
These lvl 2 page tables need 1 lvl 1 page table to store them

In total we have 3104 + 2 + 1 = 3107 pages (should be 3108)

This uses 3107 \* 16 KiB = 49 712 KiB [+1] (should be 49728) [+1]

Above is the the answer Peter Sutton had ^^



appears to be 7 processes (nevermind, point J also forks, so 8 processes)

I got 8 processes total (+4)

If someone could explain this, that would be great. I do understand that this is all the possibilities, depending on what pid 1 pid 2 and fork is, and that there seems to be no change to the pids, and thus waitpid never initializes for G and H to print. So is the the different printfs creating the processes?   
  
No, printf is just printing letters depending on which process it’s being called in. fork() is creating new processes. And some of those processes call fork again (every line upwards in the diagram is a process being forked)





6 [+1]

7 (+4) [why 7?]

2^n - 1, where n equals number of forks(for A B C) = 2^3 -1 = 7 (not that case, it depends on wait())



a, b, c, d, e, f, g, h, j, k, k, k, k, k, k = 15 -> incorrect, is actually

a, b, c, d, e, f, g, h, j, k, k, k, k, k, k, k, k = 17 (+2)

This question confused me, it seems like it should print 17 lines every time… (+1)(when a process forks, it may carry the buffer from its parent. ie not flush)

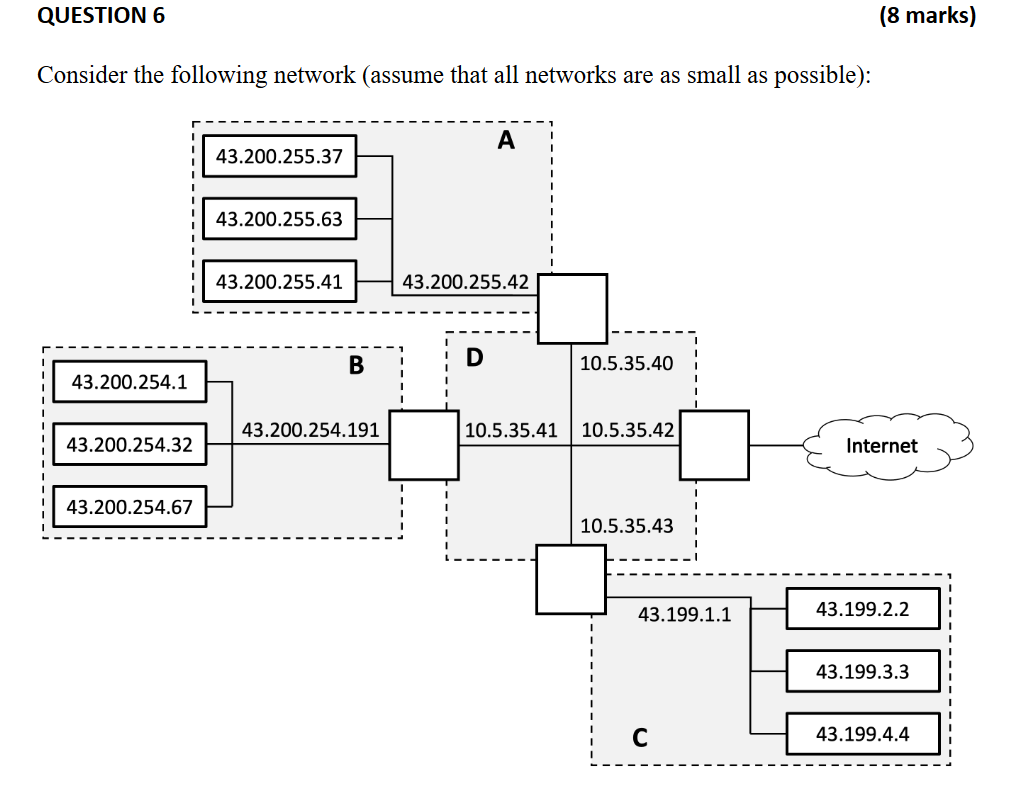
Okay but what if the forks fail? That would mean that only 4 lines can be output. Dont know if that is what they are asking but I mean technically that is the minimum amount of lines.(Generally in these type of questions, it will assume all fork() work properly, see the first line of the Question)



Stdout might be fully buffered, and the children might inherit the buffers.

A = 7, B = 4, C = 2, D= 2, E = 3, F = 2, G = 1, H = 1, J = 1, K = 6. 29 (incorrect)

A= 8, B = 4, C = 2, D = 2, E = 4, F = 2, G = 2, H = 2, J = 2, K = 8. 36 (+2)



| Network | Netmask | Broadcast Address | CIDR |
| --- | --- | --- | --- |
| A | 255.255.255.128 [+2] | 43.200.255.127 [+2] | 43.200.255.0/25 [+2] |
|  | 255.255.255.0 [+2] | 43.200.254.255 [+2] | 43.200.254.0/24 [+2] |
| C | 255.255.248.0 [+3]  255.255.240.0  43.199.0000|0001….  43.199.0000|1000….  1111 0000 = 240 DEC | 43.199.7.255 [+3]  0000 1111 = 15 DEC [-1, should be 3 bits] | 43.199.0.0/21 [+3]  43.119.0.0/20 [-1] 1 = |001  4 = |100 so its 3 bits less than 24 |
| D | 255.255.255.240 [+2] | 10.5.35.47 [+2] | 10.5.35.32 /28 [+2] |

how did you get 25 network bits for A) ?? I got this:

43.200.255.37 = .0|01|0 0101 (last quadrant) You must consider reserved host bits i.e., you may not have all 1’s on the RHS of the host bits or 0’s for that matter

43.200.255.63 = .0|01|1 1111

43.200.255.41 = .001|0 1001

43.200.255.42 = .001|0 1010

So same bits up to the line above

8 + 8 + 8 + 3 = 27 network bits

Therefore host bits = 32-27 = 5 host bits

Netmask is 1111 1111.1111 1111.1111 1111.1110 0000

= 255.255.255.224

Broadcast add is 43.200.255.63

CIDR I got 43.200.255.32/27 <- Basically because if you use /27, the broadcast address will be in use by a device (highest address is reserved for bcast)

is this just because the BA is same as one of the addresses in A? (just saw this nice one me) yes

is this in a recording It is discussed briefly in the week 13 Contact in one of the examples. In essence you move the line you draw until there is atleast one 0 on the right hand side. E.g. .001|1 1111 becomes 0|011 1111 in your example above.

How is the answer for D’s CIDR 28 bits, shouldn’t it be 29?

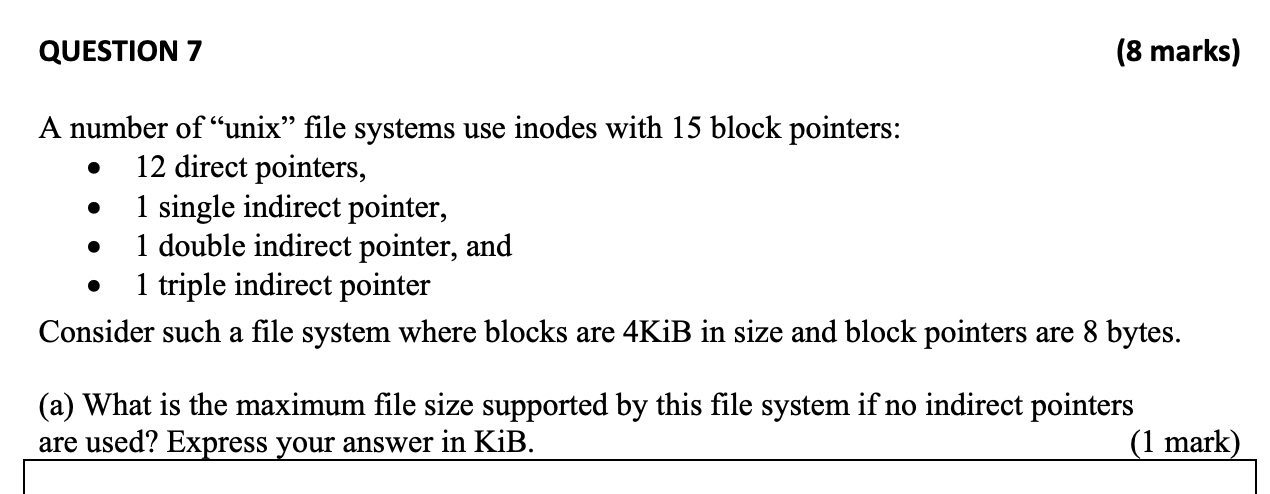
I think it should be 28, since 40 -> 00101000 and 43 -> 00101011 so you split at 0010 | 1000 and 0010 | 1011 if you split one to the right then you get 000 for 40 which isn’t valid [+1]



| Netmask | Broadcast Address | CIDR |
| --- | --- | --- |
| 255.240.0.0 | 43.207.255.255 | 43.192.0.0 /12 |

Could this be right though? As it wouldnt show the D network? According to Peter, he said the 10.x networks are private. And the above would then be correct according to my calculations aswell

Yeah I got, 192.0.0.0, 63.255.255.255, 0.0.0.0/2 Although it looks a bit ridiculous.



12 \* 4 = 48 KiB (+3)



4 \* 1024/8 = 512 pointers/block

512 + 512^2 + 12 = 262668 blocks

262668 \* 4 = 1050672 KiB [+2]

Can someone explain why the single indirect pointer and double indirect pointer are not themselves counted as blocks? They are blocks however they are not data blocks and therefore cannot store data. See <https://premaseem.files.wordpress.com/2016/02/linux_inode_diagram.gif?w=544&zoom=2>



4\*1024/8 = 512 pointers/block

512 + 512^2 + 512^3 + 12 = 134480396 blocks

134480396 \* 4 = 537921584 KiB [+1]



15 \* 8 = 120 bytes [+1]



Block num = FLOOR(addr/block size)

48001/4096 = 11 blocks

2148000/4096 = 524 blocks

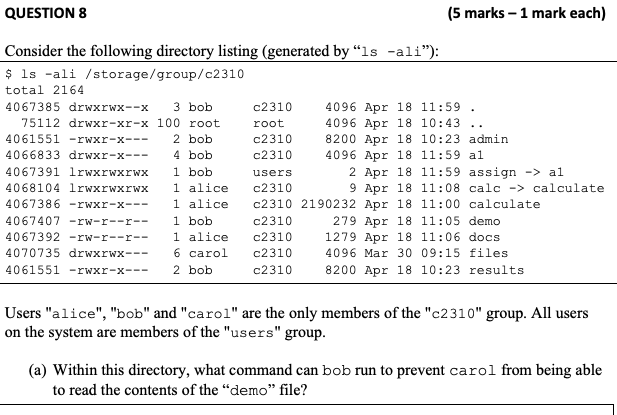
No. of blocks read = 524 - 11 = 514 + num index blocks 524-11 = 513? It is inclusive so you need to always add 1: 524 - 11 + 1 = 514



Number of index blocks = 3 (first single indirect from B12-B523, then double indirect for B524-B1035)

(514 + 3) = 517 block accesses [+2]

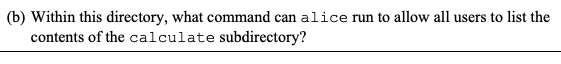
**^2022 sem 2 PPL!!!!!!!!!!!!!!!!!!!!!!!!!!Why are we not counting the triple indirect pointer blocks??Uhm i think we dont need to bc we found the sufficient number of bytes within the double indirect block [+1]**



unsure: chmod g-r demo (+4)

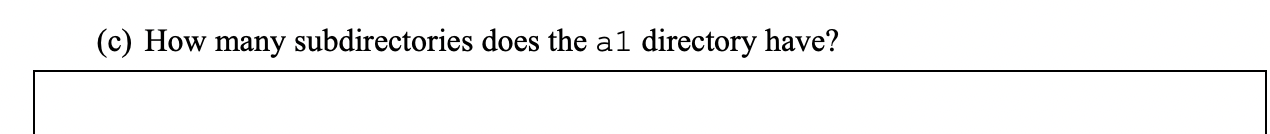
Should we not also use chmod o-r? No because carol is only affected by the most “specific” one of the 3 rules, ie. group in this case - so changing others’ perms has no effect.

Maybe ‘chmod go-r demo’ because carol is also part of users which is others. [+1]



chmod o+r calculate (+1)

chmod a+r calculate(+1) o is other users not in file group



link count = 4, minus “.”, minus “a1” So 4 - 2 = 2 subdirectories

Agree with the answer above but I thought it was 4 minus ‘.’ and minus ‘..’ to get 2 [+1], I think that is in fact right, but . and .. is always there



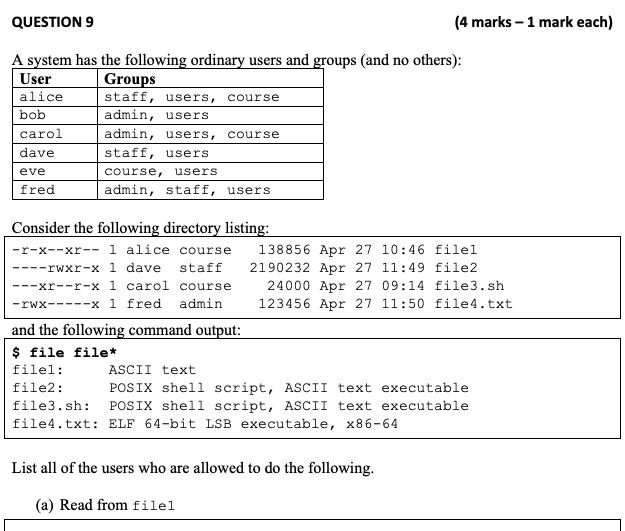
0 - there still exists one link to the file, so removing does nothing (+1)

Still removes the directory entry, judging by the other directories in the current directory it looks like directory entries are 4096 bytes [-1 for the reason above]

Directories themselves are at least 4096 bytes, not individual entries. That should be more than enough for all the dirents actually used, so I vote for 0.



bob, alice, carol (look at row for admin, see that owner has r + x, and has given group members r + x - unsure if the owner needs r + x to give the group members r + x) (+1) Why do you look at the rights for admin? I can tell there is a link from assign to a1, but i dont understand how that has anything to do with the admin filez - i think that was a typo, they meant a1 not admin.



Alice, bob, dave, fred (+4)

It says read for world/everyone, doesnt that mean that all of them can read the file?

It’s *other*, not *everyone*. Group rules override other rules, and owner rules override group rules.

Gotcha, thank you very much



Alice, fred (+2)

Dave is owner who cannot write, but he is also in the group that can write. ????Confused here

As far as i understand it, you can only have one set of permissions per person. So since dave is the owner, they are only affected by the ‘owner’ permissions. Similarly, a group member’s permissions are unaffected by what permissions the owner has

I think it should be Dave too as it is part of the staff group.

Dave cannot write since owner doesn’t have write perms. The first rule that applies to you, going from left to right, is the only one that applies. +1



Carol, bob, dave, fred (+3) (THIS IS WRONG)

Bob, dave, fred (Carol cannot “run” as they don’t have read perms on a shell script) (+5)

This is the reason why they gave us the output of “file file\*”. So we know that this is a shell script, so read perms are required.



Fred, alice, dave, eve (+4)

Again, since the others group cannot read the file, shouldn’t it just be Fred?

*Run*, not *read*. [they try to trick you with .txt, but the file is an executable and doesn’t need to be read in order to be executed, unlike shell scripts]

Answer is correct see: https://unix.stackexchange.com/questions/440082/does-execution-of-a-file-need-read-permission



gcc -Wall -I/csse2310/include sigcat.o sigtable.o -o sigcat (not sure where these fit, if at all – hq.c jobtable.h jobtable.c jobtable.h)

Not too confident in this one but here’s what I got:

gcc -Wall -l/csse2310/include -c hq.c <- I missed this one

gcc -L/csse2310/lib -lcsse2310a3 hq.0 jobtable.o -o hq

gcc -Wall -I/csse2310/include -c sigcat.c -o sigcat.o

gcc -Wall -I/csse2310/include -c sigtable.c -o sigtable.o

gcc -Wall -I/csse2310/include sigcat.o sigtable.o -o sigcat Why are we not using the default Link here? -L/csse2310… instead of the -Wall ….?

Could you explain where the first 3 came from? I’m a bit confused there, seeing as the gcc flag appears to only be called once. Are those just different answers

gcc -Wall -I/csse2310/include -c hq.c -o hq.o

gcc -L/csse2310/lib -lcsse2310a3 hq.o jobtable.o -o hq

gcc -Wall -I/csse2310/include -c sigcat.c -o sigcat.o

gcc -Wall -I/csse2310/include -c sigtable.c -o sigtable.o

gcc -Wall -I/csse2310/include sigcat.o sigtable.o -o sigcat Why are we not using -L/… instead of -I/… To me that seems like the logic thing to do since it is similar as hq?

hq.o needs to be rebuilt because jobtable.h is newer than it.

Experimental result: (substitute “/local/courses” with “/”)

gcc -Wall -I/csse2310/include -c -o hq.o hq.c

gcc -L/local/courses/csse2310/lib -lcsse2310a3 hq.o jobtable.o -o hq

gcc -Wall -I/csse2310/include -c -o sigcat.o sigcat.c

gcc -Wall -I/csse2310/include -c -o sigtable.o sigtable.c

gcc -Wall -I/csse2310/include sigcat.o sigtable.o -o sigcat

According to exam revision session - need to recompile those which are ‘outdated’:

[Recompile hq.o] gcc -Wall -I/csse2310/include -c hq.c

[recompile hq] gcc -o hq hq.o jobtable.o -L/csse2310/lib -lcsse2310a3

[recompile sigcat.o] gcc -Wall -I/csse2310/include -c sigcat.c

[recompile sigtable.o] gcc -Wall -I/csse2310/include -c sigtable.c

[recompile sigcat] gcc -Wall -I/csse2310/include sigcat.o sigtable.o -o sigcat



gcc -Wall -I/csse2310/include -g sigcat.o sigtable.o -o sigcat

gcc -Wall -I/csse2310/include -g -c sigcat.c -o sigcat.o

gcc -Wall -I/csse2310/include -g -c sigtable.c -o sigtable.o

gcc -Wall -I/csse2310/include -g sigcat.o sigtable.o -o sigcat

Again, don’t know where the first 2 came from. What exactly is a -c flag?

Purple is correct, the first two are because the target is older than one of the prerequisites.

From the gcc man page,

“-c Compile or assemble the source files, but do not link. The linking stage simply is not done. The ultimate output is in the form of an object file for each source file.“

Experimental result:

gcc -Wall -I/csse2310/include -g -c -o sigcat.o sigcat.c

gcc -Wall -I/csse2310/include -g -c -o sigtable.o sigtable.c

gcc -Wall -I/csse2310/include -g sigcat.o sigtable.o -o sigcat



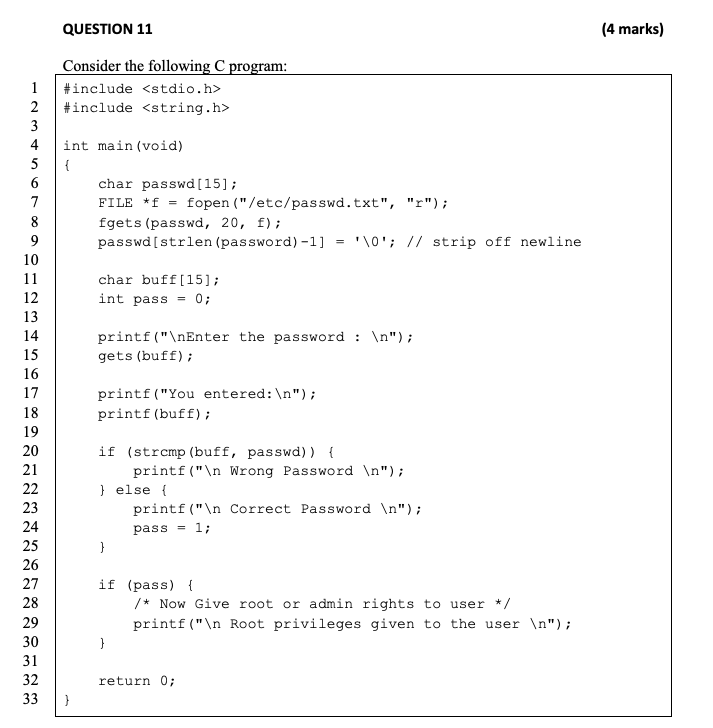
rm -f sigcat.o sigtable.o hq.o jobtable.o hq.o jobtable.o sigcat.o sigtable.o ??

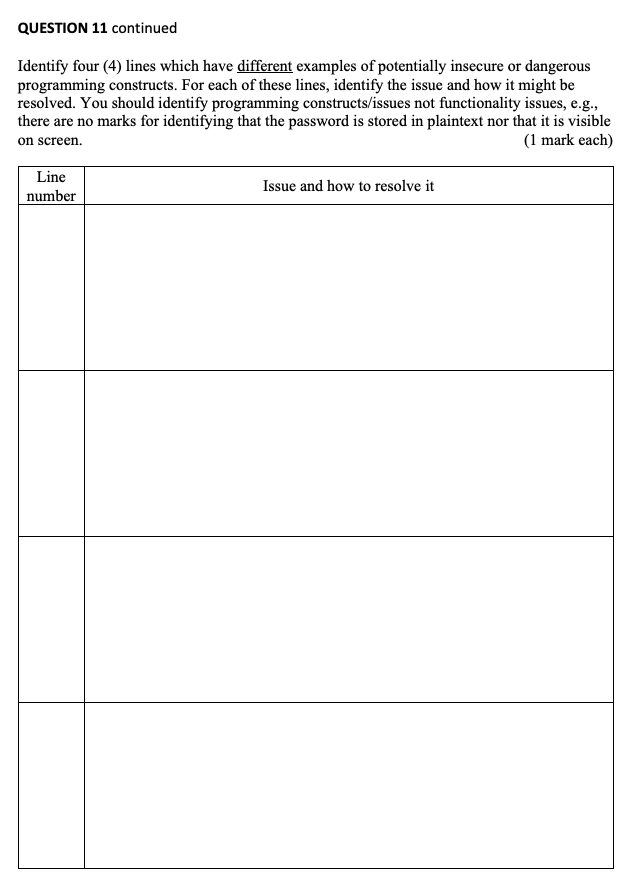
rm -f sigcat.o sigtable.o hq.o jobtable.o hq sigcat(+2)

(+1)

Experimental result:

rm -f sigcat.o sigtable.o hq.o jobtable.o hq sigcat

aAs



| Line Number | Issue and how to resolve it |
| --- | --- |
| 8 | Writing 20 characters into an array of length 15. Likely to cause a buffer overflow.  Replace 20 with 15, or increase length of array to 20 to fix this (use #define constants instead to avoid magic numbers) (+1) |
| 9 | Array probably isn’t null-terminated, since fgets probably tried to put the null character at index 19. This would cause strlen to not work correctly, potentially looping forever until SEGFAULTing password variable is actually not defined, not sure if this is intentional or not. |
| 15 | gets is depreciate and shouldn’t be used, highly likely to cause buffer overflows since it doesn’t know how many bytes it will read. Use fgets intstead +2 |
| 27 | Highly likely that the memory used by pass has been corrupted from the buff overflow from gets. This would mean pass is likely non-zero, even if password was incorrect. Could fix this by putting this logic into the else statement at line 22, or by being more specific and saying if (pass == 1) { …I would argue this is the same as the other buffer overflow attacks and shouldn’t count as the question asks for different examples. |
| 20 | 18 (-1) strncmp is worse here. Take the password “test”, with n = 4, then the user can enter any string that starts with “test” and be granted access. But also same thoughts as above box  No, the n would not be 4, it would be 20, which is the number of characters got by fgets(). n does not have to equal strlen(“test”). Thus, str**n**cmp works. |
| 18 | printf (buff) is dangerous, if the user inputs a ‘%’ in buff. The system can mistake ‘%’ for a special character like %d for integers. (+2) specifically passing a user controlled string as a format string is big no no. Can be used to sniff the stack for the password  Solution: just use printf(%s, buff); |

I apologise for what you’re about to read

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

char\*\* cmd1args = malloc(sizeof(char\*));

char\*\* cmd2args = malloc(sizeof(char\*));

cmd1args[0] = argv[1];

int i;

for(i = 2; i < argc; i++) {

if (strcmp(“==”, argv[i])) {

break;

}

cmd1args = realloc( cmd1args, i \* sizeof(char\*));

cmd1args[i - 1] = argv[i];

}

i++;

for(int j = 0; j < argc - i; j++) {

cmd2args = realloc(cmd2args, (j + 1) \* sizeof(char\*));

cmd2args[j] = argv[i + j];

}

int fd[2];

pipe(fd);

int read = fd[0];

int write = fd[1];

pid\_t cmd1;

if (!(cmd1 = fork())) {

// Child

close(2); // close stderr

close(read); // don’t need to read

dup2(write, 1); // redirect stdout

close(write);

execvp(cmd1args[0], cmd1args);

}

pid\_t cmd2;

if (!(cmd2 = fork())) {

// Child

dup2(read, 0); // redirect stdin

close(read);

execvp(cmd2args[0], cmd2args);

}

waitpid(cmd1, 0, 0);

int status;

waitpid(cmd2, &status, 0);

if (WIFEXITED(status)) {

return WEXITSTATUS(status);

}

return 0;

}

THIS IS THE SOLUTION PROVIDED DURING THE EXAM HELP SESSION

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

int n = 2;

while (strcmp(argv[n], “==”) != 0) {

n++;

}

argv[n] = NULL;

n++;

int fd[2];

pipe(fd);

if (!fork()) {

// Child

close(2);

dup2(fd[1], 1);

close(fd[1]);

close(fd[0]);

execvp(argv[1], argv + 1);

} else {

Inde // Parent

dup2(fd[0], 0);

close(fd[0]);

close(fd[1]);

execvp(argv[n], argv + n);

}

}

Same as green just better on the eyes:

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

int n = 2;

while (strcmp(argv[n], “==”) != 0) {

n++;

}

argv[n] = NULL;

n++;

int fd[2];

pipe(fd);

if (!fork()) {

// Child

close(2);

dup2(fd[1], 1);

close(fd[1]);

close(fd[0]);

execvp(argv[1], argv + 1);

} else {

// Parent

dup2(fd[0], 0);

close(fd[0]);

close(fd[1]);

execvp(argv[n], argv + n);

}

}