**ln If you want to edit join UQAttic group.**

**Don’t email me requesting access, read the instructions below.**

**CSS E2310: 2021 SEM1 exam answers**

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

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

[**Other exam papers**](https://drive.google.com/drive/folders/1k04K5RHL61k3ae2jJ362MPdN5xy3Bqsp)

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.

**EDITING PERMISSIONS LINK:**

<https://docs.google.com/document/d/1hCdQc0UK7jz7edo209sT59f85WLSjsRY/edit?usp=sharing&ouid=101185200039888287918&rtpof=true&sd=true>

**Style.**

Type answers in blue beneath each question.

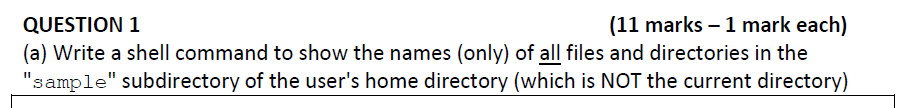
If you're unsure of your answer, highlight your answer text then hit Ctrl+Alt+M to create a comment beside the text. Once you're satisfied with the answer, click the "Resolve" button on the comment.

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.

VERSION 1: ALL GIVEN IN RED



ls -a ~/sample; or [+2]

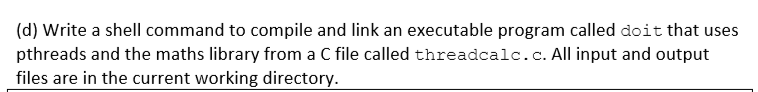
ls -a $HOME/sample; or

ls -a /home/$USER/sample <- only works if user has not changed home dir



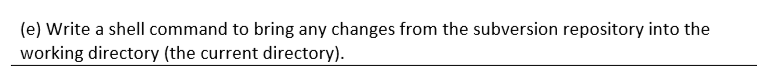
cp /tmp/\*.c ./data [+2]



mkdir breakfast

gcc -pthread -lm threadcalc.c -o doit; or

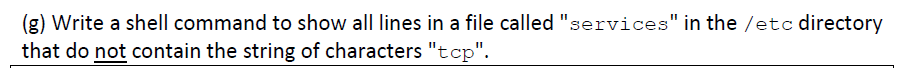
gcc -lpthread -lm threadcalc.c -o doit



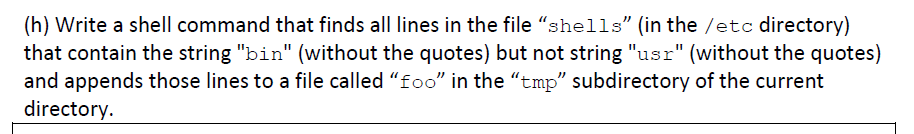
svn update



grep green colours



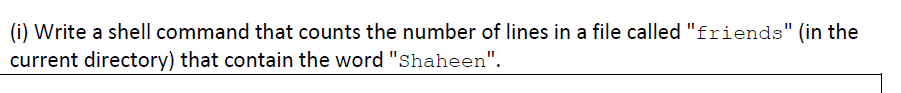
grep -v tcp /etc/services



grep bin /etc/shells | grep -v usr >> tmp/foo; or

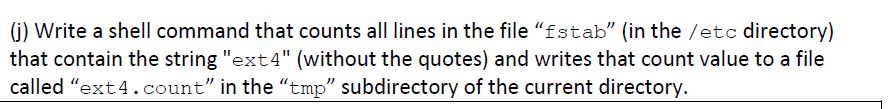
grep -v usr /etc/shells | grep bin >> tmp/foo; or

“>” or “>>”? -key word “appends” so, >>



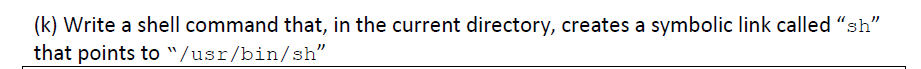
grep Shaheen friends | wc -l; or

grep -c Shaheen friends



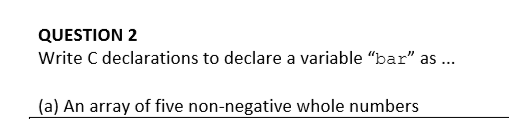
grep ext4 /etc/fstab | wc -l > tmp/ext4.count; or

grep -c ext4 /etc/fstab > tmp/ext4.count



ln -s /usr/bin/sh sh

ln -s /usr/bin/sh ./sh



unsigned int bar[5];

u\_int8\_t bar[5]? if imports allowed… # it would be uint8\_t bar[5]; but this is still wrong no? since this is just 1 byte.



int (\*bar)(char\*, char\*);



volatile int bar;



long double bar;

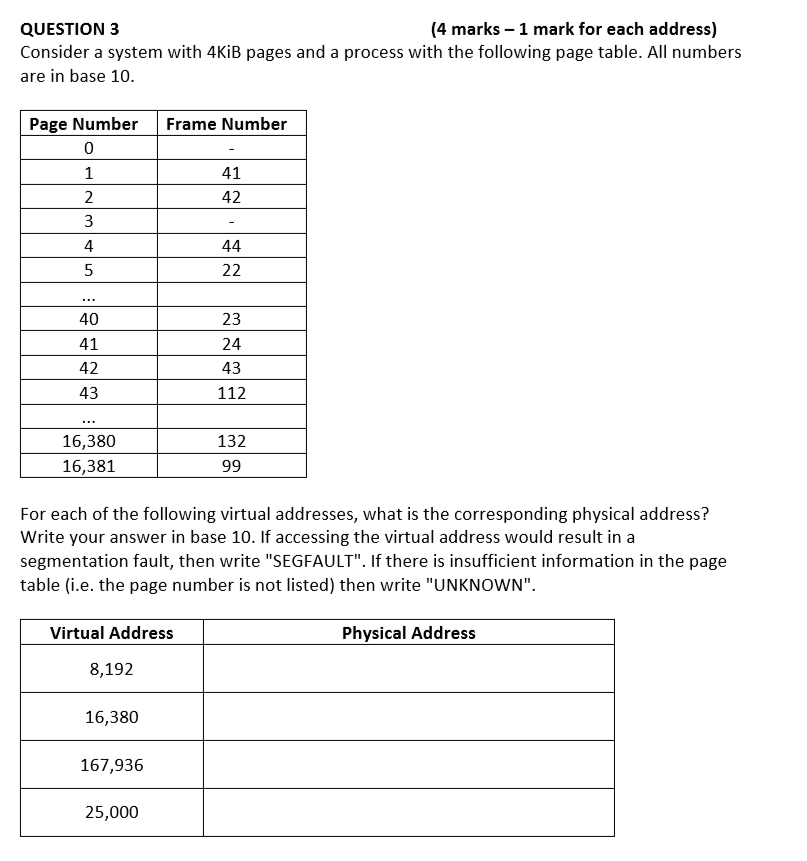
Alternative accepted answer - double bar; (approved by Peter Sutton on Ed #1442) [+1]



char \*bar[3];



void\* (\*bar)(void\*)



1. 8192/4KiB = 8192 / (4 \* 2^10 B) = 2 page number

Frame\_number(2) = 42 u

42 \* 4KiB = 42 \* 4096 = **172032** (= PA)

Is there a reason theres no offset? with offset it was 172632

1. 16,380/4KiB = 3.999<|> 3

=>>>>> bc Frame\_number(3) = - [**SEGFAULT**]

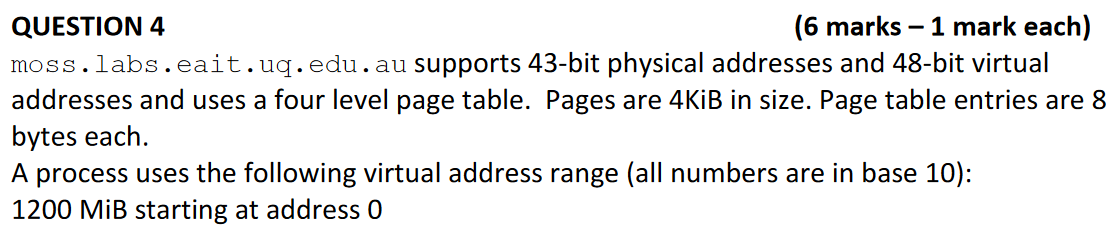
1. 167,936/4KiB = 41

Frame\_number(41) = 24

24 \* 4 KiB = **98304**

1. **UNKNOWN** because it falls past the table

Question 4





= 2^48 Bytes = 2^18 GiB = 262144 GiB

…as virtual addresses are 48-bit long and maximum memory space for a process requires virtual memory addresses.



====SOLUTION====

1. Start from determining the no. of entries per page

No. of entries per page = page size / page entry size

= 4 KiB / 8 B

= **512 entries**

2. Start from 4-th level pages

No. of 4th-level pages = 1200 MiB (*process*) / (512 \* 4KiB)

= 1200 / 2

= **600 pages**

3. Determine the number of 3-rd level pages needed to point the 600 pages

No. of 3-rd level pages = [(512 entries can point 512 pages) so we need **two** 3-rd level pages | 512 \* **2** >= 600] = **2**

4. Determine the number of 2-nd level pages needed to point to the 2 entries in the 3-rd level page

No. of 2-nd level pages = **1**

5. Determine the number of 1-st level pages needed to point to the 1 entry in the 2-nd level page

No. of 1-st level pages = **1**

Therefore:

Total number of pages = 600 + 2 + 1 + 1 = 604

Total memory needed = 604 \* 4 KiB = 2416 KiB

================

entries\_per\_page = 4KiB/8B = 4096B/8B = 512 entries\_per\_page

3rd\_level\_page\_size = = 1024MiB = 1GiB

2nd\_level\_page\_size = 512 \* 1GiB = 512GiB

num\_4th\_level\_pages = 1200MiB / 2MiB = 600 pages

Because we can only have a max of 512 entries per any page, we would need at least 2 3rd level pages to store references to 600 4th level pages

total\_mem = (1 + 1 + 2 + 600) \* 4KiB = 2416KiB

total\_mem = (num\_top\_level + num\_2nd\_level + num\_3rd\_level + num\_4th\_level) \* page\_size

—-------------------------------------------------------------------------------------

P1: do you guys get this?

P2: no… i need explanation too

P1: bahaha glad its not just me [+1]

P2: also… i seem to find the fact /8B in the first line weird, is it a conversion to bytes?

P1: pretty sure its because the questions says page table entries are 8 bytes each, so to find the num of page entries you need to divide the total size of the page by the size of a page table entry. [+2]

ps. if you guys are confused, have a look at the above solution i just made. hope it clears any confusion.

—------------------------------------------------------------------------------------

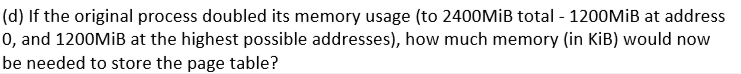


= (1200 + 3 + 1 + 1) x 4KiB = 4820 KiB

num\_4th\_level\_pages = 2400MiB/2MiB = 1200 pages

we now need at least 3 3rd level pages ->

total\_mem = (1 + 1 + 3 + 1200) \* 4KiB = 4820KiB



= (1200 + 4 + 1 + 1) x 4KiB = 4824 KiB (the third level may be 2, am unsure)

I think (1st: 1, 2nd: 2, 3rd: 4, 4th: 1200) - making 4828KiB [+2].

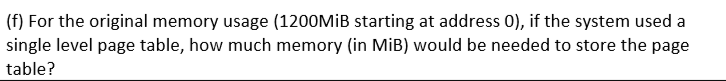
The answer is 1 for the first level table because of what Peter said regarding the first table always often being 1.

Yes you’re right. In the second exam prep session Peter does an almost identical question at 1:29 and he said 4 pages at second last level, so it would be 4828KiB +1 I also got the 2nd answerA



I have: Larger 1st level PT (12b offset, then 512 = 2^9 PTE/Page, so 9 bit for PT level 2, leaving 48 - (12+9) = 27 bits for level 1. `

Thus, 2^27 entries \* 8byte entries for level 1. We still need 600 level 2 entries, so (600 \* 4KiB + 2^27 \* 8B) = 1050976KiB [+3]

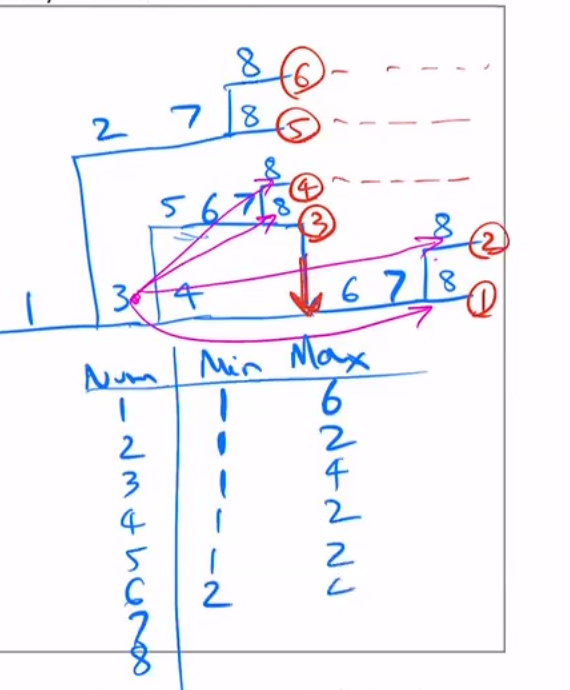


((2^(48-12) \* 8B) / 2^20 = 524288 MiB [+7]

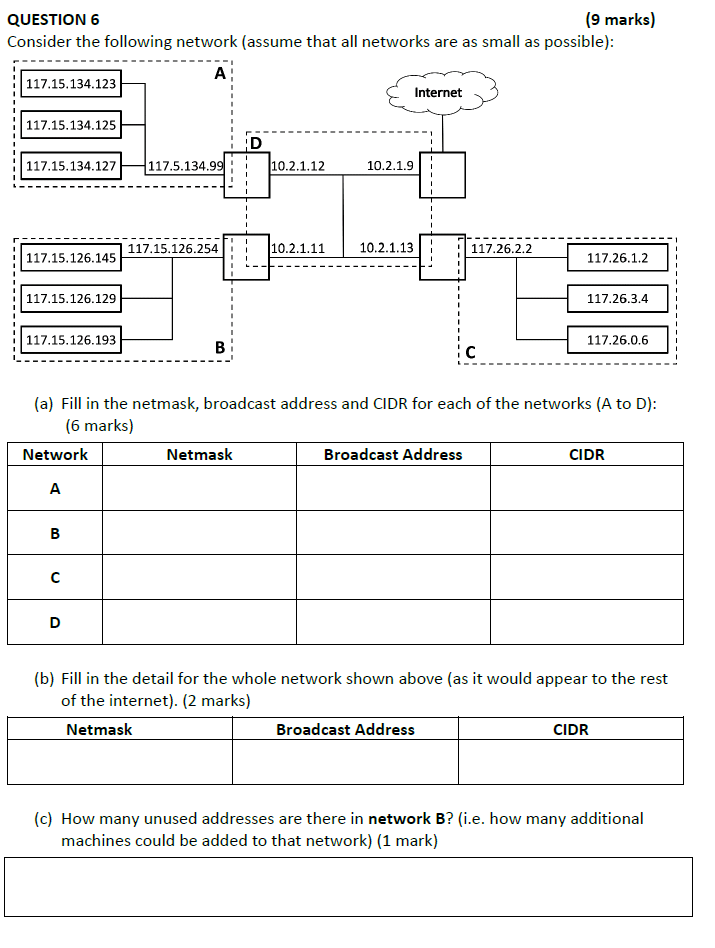
-- This should account for the offset of the virtual address??damn

x

Answers to these are in Exam Consult Recording 3 Mon Nov 15 @ 5min

image courtesy of Peter (the CC)

1. 6
2. 5 “why not 4?" not 4 cause 5 overlap if you draw a vertical line (where all the 8s are)
3. 4
4. 16
5. 32



a)

IP addresses in **red** as per Peter’s answers in Exam Prep Session 1 Recording @ 20min

IP addresses in blue are student attempted answers

| Network | Netmask | Broadcast Address | CIDR |
| --- | --- | --- | --- |
| A | **255.240.0.0** | **117.15.255.255** | **117.0.0.0/12** |
| B | 255.255.255.128 | 117.15.126.255 | **117.15.126.128/25** |
| C | 255.255.252.0 | 117.26.3.255 | 117.26.0.0/22 [+2] |
| D | **255.255.255.248** | **10.2.1.15** | **10.2.1.8/29** |

**For A:** (Assuming .15 not .5 as .5 was a typo):

Largest IP: 117.15.134.127 → 117.15.134. | 01111111

Smallest IP: 117.15.134.99 → 117.15.134. | 01100011 (Boundary from RHS till LHS matches)

CIDR: → 117.15.134.0 /24 (Number of bits on left of boundary)

Netmask: → 255.255.255.0 (Set bits on right to 1, bits on left to 0)

Broadcast Address: → 117.15.134.255 (Set bits on right to 1, bits on left unchanged)

**For B:**

**|**

Largest IP: 117.15.126.254 → 117.15.126.1**|**1111110

Smallest IP: 117.15.126.129 → 117.15.126.1**|**0000001

**|**

CIDR: → 117.15.126.128/25

(as 32 - 7 and the 128 comes form 1 before the **|**)

Netmask: → 255.255.255.128

(as the 1 before the **|** and the 255 because of all digits becoming 1 to the left **|** and 0 to the right)

Broadcast Address: → 117.15.126.255

(because 1**|**1111111 = 255 and the rest is unchanged)

…

b) Peter’s answers in Exam Prep Session 1 Recording @ 23min

| Netmask | Broadcast Address | CIDR |
| --- | --- | --- |
| 255.224.0.0 | 117.31.255.255 | 117.0.0.0/11 |

c) According to Peter’s working in Exam Prep Session 1 Recording @ 1hr

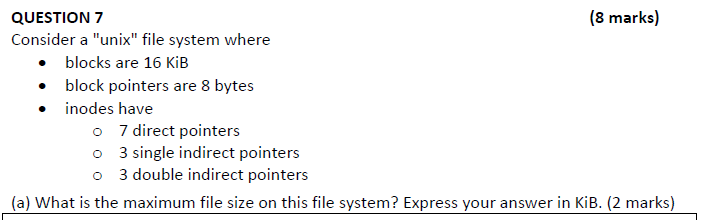
total bits = 32

network bits = 25

remaining host bits = 32 - 25 = 7

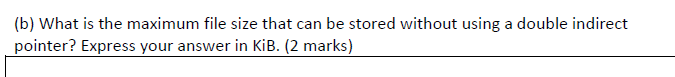
max number of hosts = 2^7 - 2 = 126

Given 4 addresses already assigned in the network → unused address = 126 - 4 = 122



= (7 + (2^11 x 3) + (2^11)^2 x 3) x 16KiB

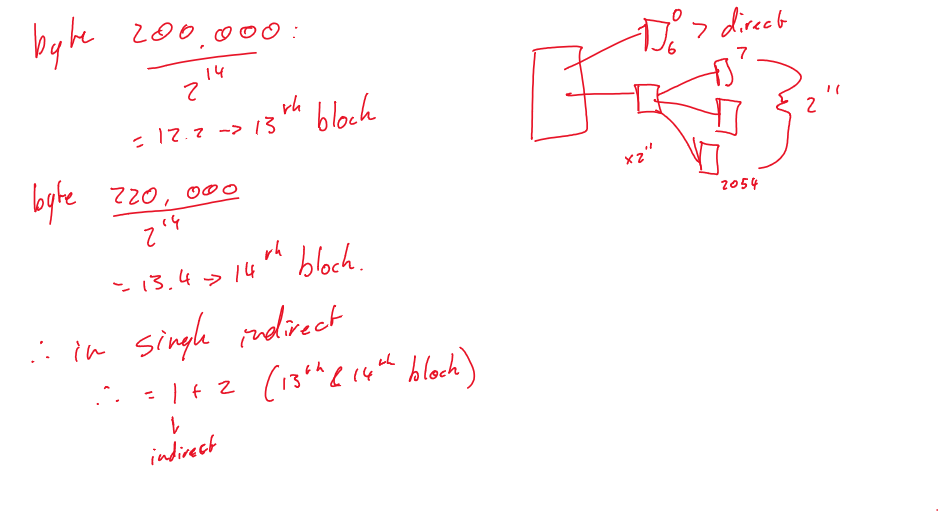
= 201,425,008 KiB



All direct and single indirect

=( 7 + 2^11 x 3)x 16Kib

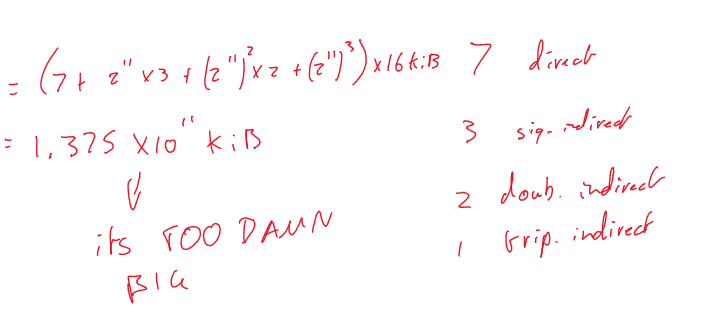
= 98416KiB



This ends up being correct but Peter uses the floor of the address/blocksize not the ceiling so should be 12th and 13th block not 13th and 14th block

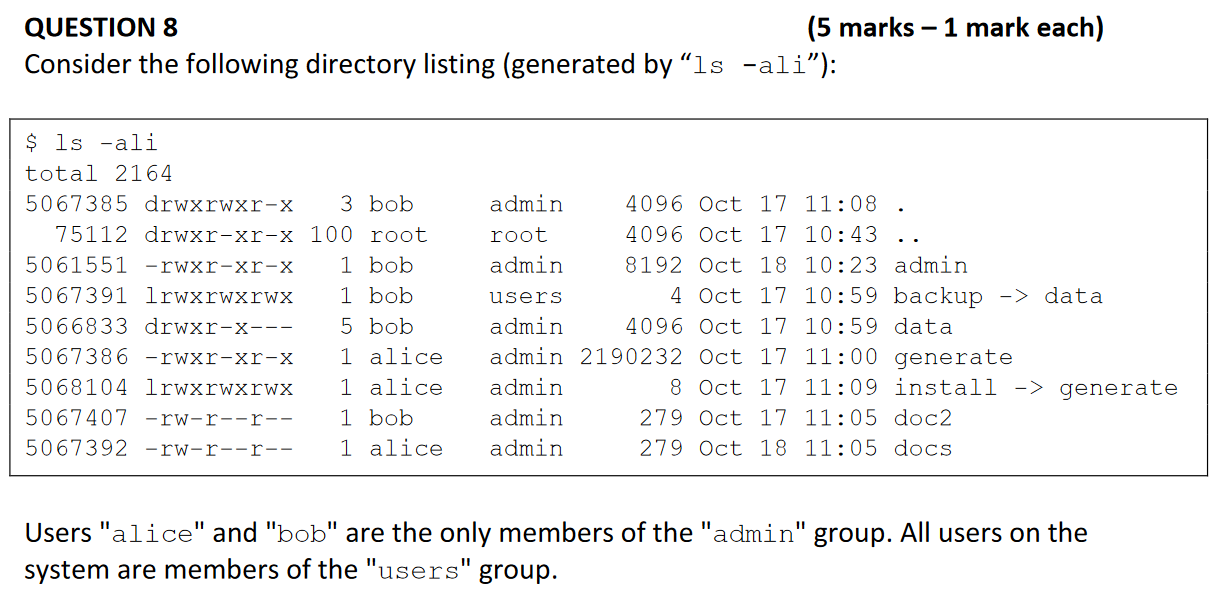
We should use floor for the lower block number, ceiling for the higher block number. This gives us blocks 12 to 14 inclusive, which must be handled through a single single indirect pointer, which gives us 4 block accesses.

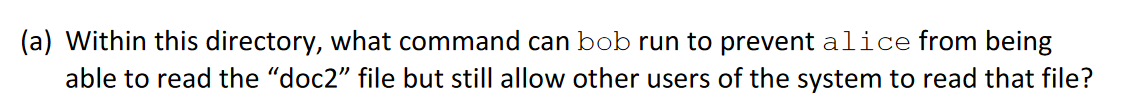




1.376\*10^11 KiB (with the rounded number)

<https://www.youtube.com/watch?v=DpF7Bgkv2QU> +4 gr





chmod g-r doc2

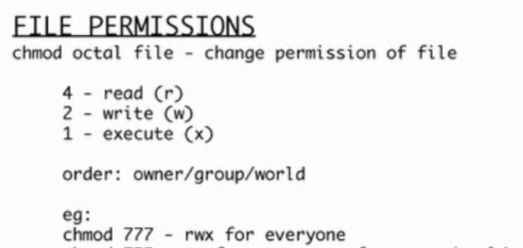
—--------------------------------------------

or (if this is helpful)

chmod 604 doc2

6 0 4 because

* owner has r w = (4 + 2 + 0)
* group (alice) has none now = (0 + 0 + 0)
* world has r = (4)



—-----------------------------------------------



chmod g-rx, o+rx data(I think you need both read and run permission to list something cuz it’s a shell script(?)) [+2]

For directories, x acts like access permission, and r acts like ls (read) permission.

can it be chmod o+r,,x data ??

chmod 705 data;



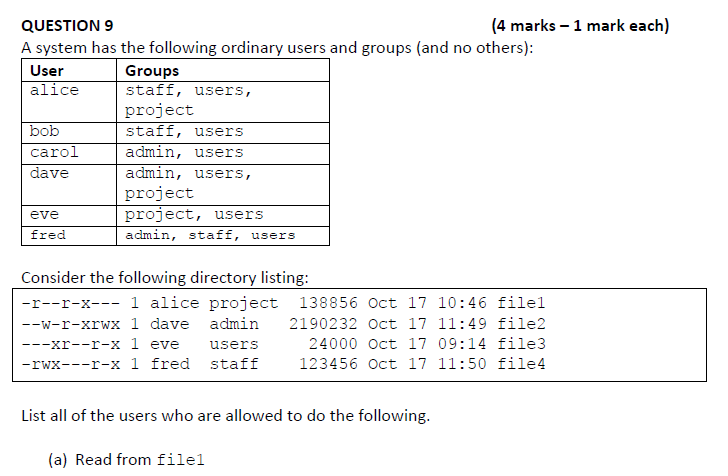
3 (5-2)

3

ln doc2 docx



ln -s docs docz



alice, dave, eve



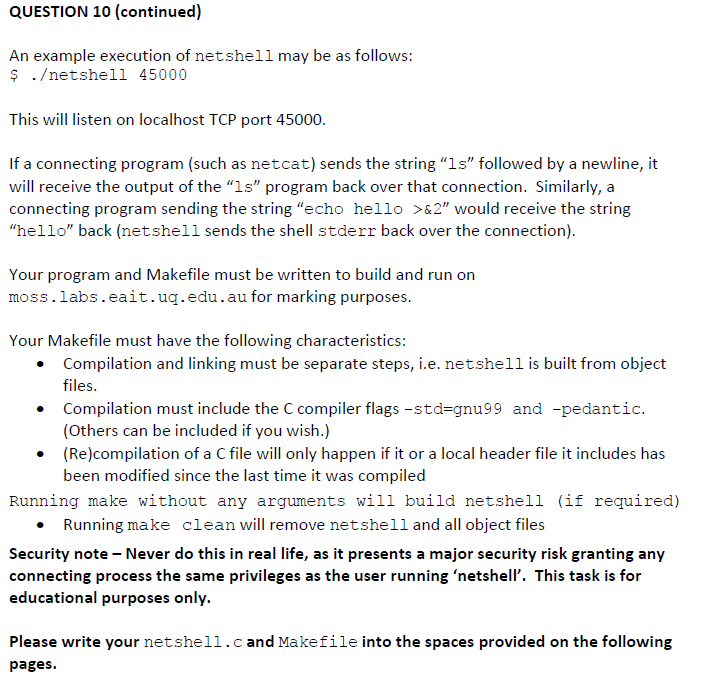
dave, eve, bob, alice



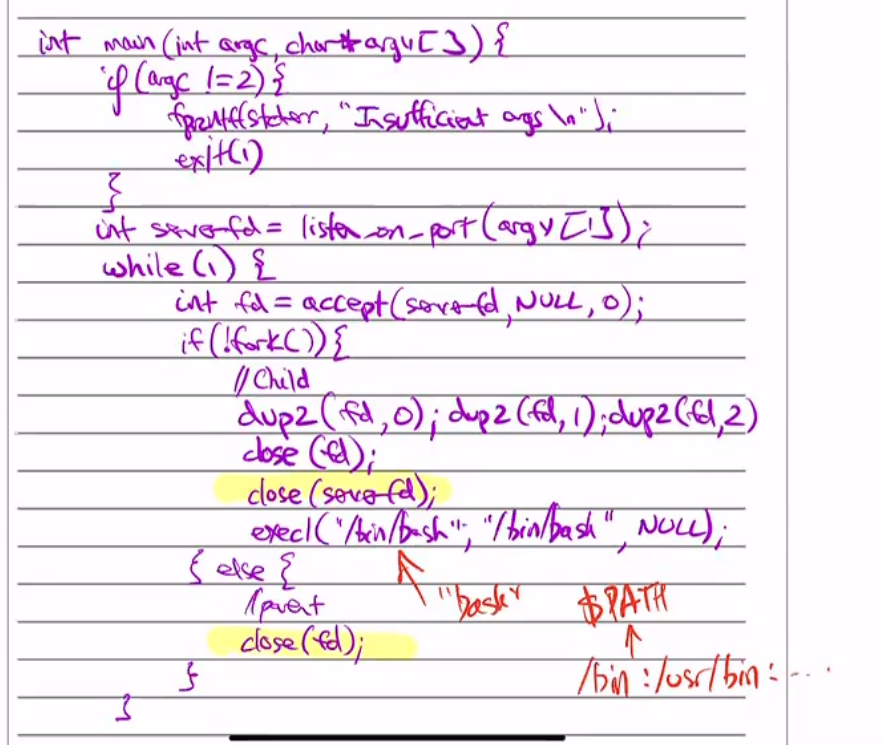
eve



fred, eve, dave, carol

9

This is just an attempt, there are still a couple flaws in the program: for one, there is some superfluous code in it, and I didnt link the .h file and the listen\_on\_port properly (I just threw it into the netshell.c code).



netshell.c

#include <sys/types.h>

#include <sys/socket.h>

#include <netdb.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

//for duping and execing

#include <fcntl.h>

#include <unistd.h>

#define INITIAL\_BUFFER\_SIZE 80

int listen\_on\_port(char\* port);

int process\_connection(int fdServer);

int listen\_on\_port(char\* port) {

struct addrinfo\* ai = 0;

struct addrinfo hints;

memset(&hints, 0, sizeof(struct addrinfo));

hints.ai\_family = AF\_INET;

hints.ai\_socktype = SOCK\_STREAM;

getaddrinfo("localhost", port, &hints, &ai);

int server\_fd = socket(AF\_INET, SOCK\_STREAM, 0);

bind(server\_fd, (struct sockaddr\*)ai->ai\_addr,

sizeof(struct sockaddr));

listen(server\_fd, 1);

return server\_fd;

}

int process\_connection(int fdServer) {

int fd; //this will become the client fd

struct sockaddr\_in fromAddr;

socklen\_t fromAddrSize;

fromAddrSize = sizeof(struct sockaddr\_in);

// Block, waiting for a new connection.

fd = accept(fdServer, (struct sockaddr\*)&fromAddr, &fromAddrSize);

return fd;

}

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

int fdServer, fdClient;

//check num of arguments

if (argc != 2) {

fprintf(stderr, "invalid num of arguments\n");

exit(1);

}

// listen on the given port

fdServer = listen\_on\_port(argv[1]);

while(1) {

//receive a connection

fdClient = process\_connection(fdServer);

if (!fork()) { //CHILD

//setup filefd's

dup2(fdClient, 1);

dup2(fdClient, 2);

dup2(fdClient, 0);

close(fdClient);

//exec a bash terminal

execlp("/bin/bash", "/bin/bash", NULL);

} else { //PARENT

close(fdClient);

}

}

return 0;

}

MAKEFILE:

CC = gcc

CFLAGS = -Wall -pedantic -std=gnu99 -g

# -pthread -lm (not required)

.PHONY: all clean careful

.DEAFULT\_GOAL := all

all: netshell

careful: CFLAGS += -Werror

careful: all

#build netshell with netshell.c

netshell: netshell.o

$(CC) $(CFLAGS) netshell.o -o netshell

netshell.o: netshell.c

$(CC) $(CFLAGS) -c netshell.c

clean:

rm -f netshell \*.o

Here is another attempt which seems to work on moss:

**Makefile**:

CC=gcc

CCFLAGS=-Wall -pedantic -std=gnu99

.PHONY: all

.DEFAULT\_GOAL: all

netshell: netshell.o listen.o

$(CC) $(CCFLAGS) $^ -o $@

netshell.o: netshell.c listen.h

listen.o: listen.c listen.h

clean:

rm -f \*.o netshell

all: netshell

—-----------------------------------

some idiot deleted it.. restored from previous version now

**netshell.c**:

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

#include "listen.h"

void process\_connections(int fdServer) {

int fd;

struct sockaddr\_in fromAddr;

socklen\_t fromAddrSize;

char buffer[2048];

ssize\_t numBytesRead;

while (1) {

fromAddrSize = sizeof(struct sockaddr\_in);

fd = accept(fdServer, (struct sockaddr\*)&fromAddr, &fromAddrSize);

int pid = fork();

if (pid == 0) {

printf("Entering child\n");

close(fdServer);

dup2(fd, STDIN\_FILENO);

dup2(fd, STDOUT\_FILENO);

dup2(fd, STDERR\_FILENO);

close(fd);

execlp("/bin/bash", "/bin/bash", NULL);

}

}

}

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

if (argc != 2) {

fprintf(stderr, "Usage error\n");

exit(1);

}

int fdServer = listen\_on\_port(argv[1]);

process\_connections(fdServer);

return 0;

}