# Shell Questions

**show how many files are in the current directory**

ls -1 | wc –l

[Anon] are we supposed to ignore hidden files? also this answer returns 1 more as the wc-l counts the extra line up top.

No ls -1 returns one file per line, (ls -l returns the extra line you are thinking of).

**For each user with processes running on the system print their name and how many processes are they running.**

ps -fa --no-headers | sort -k 1 | cut -d' ' -f1 | uniq -c (this gets the required information, but is ordered as count, user rather than user, count - not sure if that matters)

Tough guys do it this way:

ps -aouser | sort | uniq -c

ps -aouser --no-headers | uniq -c (problem solved).

**Show all lines in f1 f2 f3 that contain tiger**

grep tiger f1 f2 f3

**Show the second last line of a file**

tail -2 file | head -1

**Show all the lines in a file that don’t contain the word tiger**

grep –v tiger file

**Show all vim processes running on moss**

ps –ef | grep vim | grep -v grep

**Rename main.c to flip.c in svn repository.**

svn move main.c flip.c

svn commit

Also works: svn rename main.c flip.c and then commit [Jared]

**List only files with uncommitted modifications**

svn status | grep M (not correct, will also show any files that have M in its name)

Try

svn status | cut -f1 -d’ ‘ | grep M [Edd]

svn st -q [MDN]

According to the study session, this should be:

svn status | grep ‘^M’ | cut -f2 -d‘ ’ [Jenni] - ‘^’ = beginning of lin

# Declarations

**Declare foo to be:**

**a long integer**

long foo;

**an array of 12 short integers**

short foo[12];

**a struct containing a pointer to an int**

struct {

int \* var;

} foo;

**an array of strings**

char \*\* foo;

**a union containing a string, an array of strings and a char**

union {

char\* var1;

char\*\* var2;

char var3;

} foo ;

# basic coding

**What is the danger with the following line:**

**scanf(“%d %s”, &i, mystr);**

Could overflow the buffer (mystr)

mystr could point to something invalid

<http://stackoverflow.com/questions/2430303/disadvantages-of-scanf>

**The following describes a binary tree [each node has up to two “children”]**

**struct Tree\* {**

**void\* data;**

**struct Tree\* left;**

**struct Tree\* right;**

**};**

**struct Tree\* create(void\* d) {**

**struct Tree\* res=malloc(sizeof(struct Tree));**

**res->data=d;**

**res->left=0;**

**res->right=0;**

**return res;**

**}**

**void attachLeft(struct Tree\* t, struct Tree\* newt) {**

**if (t->left==0) {**

**t->left=newt;**

**}**

**}**

**What input would cause attachLeft() to misbehave?**

A cyclic link where t == newt

Also if t == NULL, according to Joel in last Friday’s study session~

**Write attachRight().**

void attachRight(struct Tree\* t, struct Tree\* newt) {

if (t->right==0 && t != newt) {

t->right=newt;

}

}

**Write struct Tree\* find(struct Tree\* root, void\* d)**

**- return the first node in the tree which stores data d. [There is some information missing which you need in order to answer this question – what is it?]**

struct Tree\* find(struct Tree\* root, void\* d) {

if(root->data == d) {

return root;

} else {

//if no children

if(!root->left && !root->right) {

return NULL;

}

struct Tree\* left, \* right;

//search left branch

if(root->left) {

left = find(root->left, d);

}

//search right branch

if(root->right) {

right = find(root->right, d);

}

if(left != NULL) {

return left;

}

if(right != NULL) {

return right;

}

return NULL

}

}

**From tute today (24/06):**

struct Tree \*find (struct Tree \*root, void \*d) {

if (root->data == d) {

return root;

}

tree \*p;

if (root->left != 0) {

p = find(root->left, d);

if (p != 0) {

return p;

}

}

if (root->right != 0) {

p = find(root->right, d);

if (p != 0) {

return p;

}

}

return 0;

}

Another alternative: (bad code style but just a subtely different approach from above)

struct Tree\* find( struct Tree\* root, void \* d) {

struct Tree\* tree = NULL;

if ((root == NULL) && (root->data == d)) {

tree = root;

} else if ((tree = find(root->left, d)) != NULL) {

} else if ((tree = find(root->right, d)) != NULL) {

}

return tree;

}

The way in which you search the tree should be specified I guess, need to know whether to do an inorder, preorder or postorder traversal.

**void destroy(struct Tree\* root)**

**- Destroy the tree beginning at root.**

void destroy(struct Tree\* root) {

if(root->left) {

destroy(root->left);

}

if(root->right) {

destroy(root->right);

}

free(root);

}

**size\_t size(struct Tree\* root)**

**- Count the number of nodes in the tree**

size\_t size(struct Tree\* root) {

size\_t count = 1;

if(root->left) {

count += size(root->left);

}

if(root->right) {

count += size(root->right);

}

return count;

}

**For the list struct discussed in lectures write a function to compute the length of a list.**

???

<http://cslibrary.stanford.edu/103/LinkedListBasics.pdf> [Edd]

/\*

Given a linked list head pointer, compute

and return the number of nodes in the list.

\*/

int Length(struct node\* head) {

struct node\* current = head;

int count = 0;

while (current != NULL) {

count++;

current = current->next;

}

return count;

}

What I’ve got (is this okay? Could someone check, please?) [Jenni]:

//Compute length of a list

size\_t size(struct List\* list) {

size\_t size = 0;

if (list != NULL) {

size = 1;

}

if (list->next != NULL) {

size += size(list->next);

}

return size;

}

**What is foo in the following:**

**char\* const foo;**

a character pointer where the pointer is constant

**char X, \*foo;**

X is a character, foo is a character pointer

**typedef int\* foo;**

integer pointer type

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

function pointer that takes an int and char and returns an int

**const char\* const \* foo;**

a (mutable) pointer to a constant pointer to a character constant

**int\*\* (\*foo[10])();**

an array of 10 pointers to functions, each of which return a pointer to a pointer to an int pointer, and take no arguments - Should be an unspecified number of arguments [Edd]

**int \*\*(\*foo)(int[]);**

function pointer that takes an integer array and returns an array of int pointers

**What is the value of x after the following:**

**int a=1;**

**int x=0;**

**while (a++-2<4) {**

**x++;**

**}**

x = 5 according to compiler

**int a=2;**

**int x=0;**

**switch (a+a/2\*3) {**

**case 5: x=1; break;**

**case 6: x=2; break;**

**default: x+=3;**

**}**

x = 5 ← err X=1.. a/2\*3 = 3, switch(5) Clement 22/06/11

sorry, my bad... i just worked out the switch and then forgot to follow through -Steph

**for (int a=0,x=1; a<5; x+=a, a++);**

Depends if the question is in ANSI C or not. Past exams state that code is ansi-C unless otherwise declared. It’s not valid in ansi-c because of the (int...) inside the for loop, needs to be declared before it.

If we can assume it’s valid (i.e. C99), x = 11

**int a=0;**

**int b=1;**

**int x=2;**

**b+=x;**

**x=a++-1;**

**x-=a+b;**

x = -5

**int a=7;**

**int x=5;**

**x=(x^a)&(x|a);**

x = 2

## 

## File systems:

I**dentify each column in the following output from ls -li**

**27814396 -rwx--x--x 1 uqjfenw1 uusers 11335 Apr 3 18:59 moss**

inode\_num, permissions, num\_links, owner, group, size, date\_modified\_last, time\_modified, filename

n

**Suppose you have a directory called cats. What command would you need to use to allow user joel to list files in that directory?**

chmod a+r cats

→ Wait, what. Listing files requires execute and read, no? Wouldn’t it be chmod a+rx cats? [Jenni]

I agree with Jenni [Steph]

--> Checked at the study sessions today--we need x + r for ls. [Jenni]

**What extra information would you need in order to answer this question?**

who owns the directory, i can only change permissions if i own it

Possibly, which group joel belongs to for efficiency... [Jared]

**What permissions are required to execute a shell script? Why?**

Executing a shell script requires +rx to be set for the user who is executing the script. +x on its own is not enough as the shell can start a process to run the script but will not be able to read the contents of the script.

**-r---w---x 1 uqjfenw1 uusers 11335 Apr 3 18:59 moss**

**If uusers consists of richard, luke, tom and pat what can the following users do to moss?**

**richard,**

**uqjfenw1**

**peter**

* richard can write only
* uqjfenw1 can read only
* peter can execute only

**What type of files are the following:**

**1. virtual-jfenwick.PBe8Ew**

**2. magic**

**3. mov01256.mpg**

**4. virtual-jfenwick.PBe8Ew**

**5. mayavi\_user\_guide.pdf**

Not enough information given to tell in unix, as the file extension is not used for determining file type.

**What is the maximum possible size for a “unix” file system which has i-nodes with 10 direct pointers, 2 indirect pointers and 2 double indirect pointers [blocks are 4kB and each block can store 512 pointers]?**

10 direct = 10 \* 4096 = 40,960

2 indirect = 2 \* 512 \* 4096 = 4,194,304

2 double indirect = 2 \* 512 \* 512 \* 4096 = 2,147,483,648

= 2,101,288 kB

= 2,052 MB

^ Joel clarified on the newsgroup, this is correct.

**For the above filesystem how many blocks must be read to access the following bytes of the file:**

Reworked it out according to what Joel said on the newsgroup [Jenni]:

1. 200

200 < 4096, so the data will be in the first direct block

1 block must be read

2. 4096

4096 !< 4096, so the data will be in the second direct block (block 1: 0→4095, block 2: 4096→)

Because we don’t want to read 4096 bytes of the file, but only the the 4096*th* byte of the file, this will still only require 1 block read (of the 2nd direct block).

# 3. 40959

40959 !< 4096. So we do 40959/4096 = 9.99, so this byte will be stored in the 10th block. We have 10 direct blocks so this is still on 1 block read.

4. 40960

40960 !< 4096. So we do 40960/4096 = 10, so this byte will be stored in the 11th block. We have 10 direct blocks only, so this will be in the first indirect block, so it will require 1 (the block itself) + 1 (the index block) = 2 blocks to access.

5. 4235901

4235901 !< 4096. So we do 4235901/4096 = 1034.15, so this byte will be stored in the 1035th block. This will require 1035-10 = 1025 indirect blocks. 1025/512 (the number of pointers per single indirect block e.g., the number of blocks available per single indirect block) = 2.001. So the two single indirect blocks will not be sufficient; byte 4235901 will have to be stored on the first double indirect block. So to access this byte will require 1 (the block itself) + 2 (the two index blocks of the double indirect) = 3 blocks to access.

(remembering block size is 4096 bytes, also don’t count initial i-node (joel said this somewhere...)

**1. 200,**

200 < 4096 so the data will be in the first direct block, so 1

The only block which will be read to access the 200th byte is the one that the data is stored in, as it is a direct block, so the answer is 1.

**2. 4096,**

4096 is the block size, clearly 4096 bytes will fit in a block that holds 4096 bytes, but numbering starts at 0 so this is actually the first byte of block 2. These are still direct blocks, so 2 blocks will be needed.

I still don’t know about this. It will either be one or two direct blocks, depending on what starting at 0 means...

**3. 40959,**

40959/4096 = 9.99.. = 10 blocks needed. There are 10 direct blocks so this fits... just.

Since this is still direct blocks, only 10 will be needed.

**4. 40960**

40960/4096 = 10... so 10 blocks needed, right? nope, because this is the first byte of the 11th block since it starts numbering from 0. There’s only 10 indirect blocks so we’re going to have to start using a single-indirect now. We’ll only need one indirect block, all the rest of the data will be stored in the direct pointer blocks. This means the block count increases by 2: one for the single-indirect pointer block (which we need a reference to so we can access it) and one for the actual block where the data is stored. So 12 blocks total.

Here’s where it starts to get interesting... if starting at 0 doesn’t affect it, the answer will be the same as above.

If starting at 0 does affect it, the answer is one single-indirect block + one for the block the actual data is stored in = 2.

**5. 4235901**

4235901/4096 = 1034.15... = 1035 data blocks needed. There are 10 direct blocks; 1035 > 10 so we’ll be using indirect pointers as well. The first 10 blocks of the file will be saved in the direct blocks, which leaves 1025 blocks which need to be referenced through indirect pointers. There are 2\*512 = 1024 single-indirect pointers; 1025 > 1024 so just single-indirect pointers won’t be enough, we’ll need to use double-indirect pointers as well. There will only be one block (1025-1024) of actual data that needs to be accessed by a double-indirect pointer, but we’ll still need a reference to the double-indirect block that holds the pointer.

So we’ll need 1035 data blocks + 2 single-indirect pointer blocks + 1 double-indirect pointer block + 1 block that the double-indirect pointer refers to = 1039.

The only blocks we actually care about in this case are the ones which are accessed to get the data, that is the 2 indexes from the double-indirect, and the actual data block. Answer is 3. according to joel.

**Are there any circumstances when a linked filesystem would perform better than a “unix”**

**filesystem? Explain.**

A linked filesystem would be better than a unix file system if files were nearly always to be accessed sequentially and/or if the sizes of files regularly changed a lot, as sequential access is how the system is designed.

**What are two differences between symlinks and hardlinks?**

1. Can’t make hardlinks for a directory but you can make symlinks
2. Symlinks are useless once the linked item is removed whereas hard links will still show you the content

Symlinks create another, special, file which is essentially a pointer or shortcut to an existing file. The OS can recognise these special files and redirect them to the linked file. Hard links on the other hand are just another filename for the linked file, and the user will arrive at the original file without being redirected first. One other difference is that symlinks, being a separate file, have their own unique index number; hard links have the same index number as the original file. You can set a symlink to follow a different file by changing the path that users will be redirected to; changing a hard link will actually change the file itself. Similarly, if you remove a symbolic link, only the link to the original file is deleted and the file contents can still be accessed at its original location. ~~I~~f a hard link is deleted, the link count of the file decreases; the file is deleted when the link count reaches 0.

**Networks:**

**What are the network addresses of the following hosts:**

|  |  |  |
| --- | --- | --- |
| **Address** | **netmask** | AND |
| 130.102.65.27 | 255.224.0.0 | 011001  11100000 &  01100000  130.96.0.0 |
| 130.102.65.27 | 255.255.240.0 | 0100 0001  1111 0000 &  0100 0000  130.102.64.0 |
| 78.78.78.78 | 128.0.0.0 | 0100 1110  1000 0000 &  0000 0000  0.0.0.0 |
| 69.163.249.38 | 255.255.255.0 | 69.163.249.0 |

**What are the layers of the IP stack and what are their purposes?**

Application layer-In TCP/IP, the Application Layer contains all protocols and methods that fall into the realm of process-to-process communications across an Internet Protocol (IP) network. Application Layer methods use the underlying Transport Layer protocols to establish host-to-host connections.

Transport layer- Transport layer provides end-to-end communications within layers of network components and protocols.

Network layer-The Network Layer provides the functional and procedural means of transferring variable length data sequences from a source to a destination host via one or more networks while maintaining the quality of service functions. This is the layer that deals with IP addresses.

Link layer-The Data Link Layer is concerned with local delivery of frames between devices on the same LAN. This is the layer that deals with MAC addresses.

Physical layer- The Physical Layer defines the means of transmitting raw bits rather than logical data packets over a physical link connecting network nodes.

**Which layers are responsible for:**

**1. ensuring a message has not been damaged**

transport layer

**2. ensuring that messages arrive**

transport layer

**3. transferring data**

physical layer

**4. communicating between hosts**

network layer

**5. communicating between adjacent hosts**

link layer

**6. communicating between processes**

application layer

**7. communicating between processes on adjacent hosts**

application layer

**Which layers are involved [and what do they “send”] in each of the devices transferring data from a webserver(WS1) to browser running on client(C1).**

**WS1-> NAT-> Router-> Router-> C1**

Router will only do network layer

NAT converts public IP address to Private IP -> data link layer

Client will be everything up to application layer

**What network queries (may) need to be executed to transfer the webpage above?**

**From tute 24/06:**

DHCP (incase your lease time is due to be checked) :)

HTTP get

DNS lookup

(^ should be enough to answer question in exam situation)

MAC address lookup for host (possibly)

**What is the purpose of a gateway on a network?**

Identifies the device that allows local network computers to communicate with devices on other networks

communications with hosts outside the local network

**Fill in the sequence of calls needed to use TCP from the following words (duplicates allowed): accept, socket, bind, connect, listen**

*Server:*

socket

bind

listen

accept

*Client:*

socket

connect

**For each address, give a netmask for which the address is a valid host address and a netmask for which it is not a valid host address:**

Going from what Joel said on the newsgroups, this is what I’ve got. If someone could double check (--it seems...scarily simple?), that’d be legit, cheers. [Jenni]

**why cant we just make all subnets 0.0.0.0 wont that make all host adresses invalid?**

**114.78.128.0**

Invalid: 255.255.255.0 -> host = 114.78.128.0? invalid due to IP conflict?

Valid: 255.255.0.0

**192.168.0.40**

Invalid: 255.255.0.255

Valid: 255.255.255.0

**10.0.14.20**

Invalid: 255.0.255.255

Valid: 255.255.0.0

**Give the broadcast address for the networks which the following hosts belong to:**

**10.0.0.1 mask=255.255.255.0**

10.0.0.255 [Edd]

**10.0.0.1 mask=255.255.0.0**

10.0.255.255 [Edd]

**192.168.0.102 mask=255.254.0.0**

192.169.255.255 [Edd]

**Server Client**

**Threads and processes:**

**In order to create a pipe linking two processes what constraints are there on those processes?**

must be related - ie one is child of the other

**Who can join() a thread?**

The parent of the thread generally → Do threads have parents? Furthermore, is this an actual constraint? I’m pretty sure that anyone who wants to can join a thread. Threads share all memory/resources, after all, so they can be accessed by anyone in the program. [Jenni]

I think whoever wrote this is referring to the thread that created a new thread? But I’d be inclined to agree with you Jenni, though adding that it’s *most likely* to be the ‘parent’. [Steph] Thinking about this some more I’m convinced this is a “trick” question--e.g., trying to get us to say “The thread’s parent” as if it were a process. But because it isn’t a process, I’m nearly 100% certain anything can join any thread--not necessarily the function/thread which called the pthread\_create(). [Jenni] **ETA**: Brought to you from the internet, confirmation!: “This is analogous to wait(), but for pthreads. Any thread may join any other thread in the process, there is no parent/child relationship.” (any other thread can join() a thread only restriction is that a thread can’t join itself and a thread can’t join a thread that has already been joined(talk about mind f\*\*\*)

-- By Google. Clement 24/06/11)

You guys are way over thinking this, any thread that has access to another threads thread\_id can join that thread. Usually this would just be the thread t hat created the one to be joined, as you get your thread ID from pthread\_create, but it could always pass this in as an argument to another thread. [Alex]

**Who can reap a process?**

its parent or init

I would say only its parent... if the parent terminates before reaping the process, the child process is adopted by init, so init becomes its parent

**Which function is used to reap?**

Waitpid (or wait if you don’t care about the process id you’re trying to reap, just that the process gets reaped)

**Write a C fragment to execute a program grabbe**r.

if (!fork()) execl(“/path/to/grabber/”, “grabber”, (char\*) 0);

**Why is the WEXITSTATUS macro required? That is why does wait not just send back the exit status of the child?**

I would say because wait returns the pid of the process if wait returned just the exit status we wouldn’t know which process that exit status corresponds to? → What happens if we call wait and there’re no children? Does wait return something signalling its failure, or does it just hang indefinitely? Doesn’t wait return -1 if it couldn’t reap a child? But yes, I’d say because wait reaps processes in an arbitrary order, it needs to keep track of two values--the pid of the reaped process and its exit status. [Jenni]

**What is the difference between execv and execlp?**

They take different arguments, execv takes a path and an array of arguments (with last entry NULL), execlp takes a filename and a list of arguments terminated by a null pointer.

You could also mention that execlp will search the default path or $PATH value for the specified program [Doug N]

For the above filesystem how many blocks must be read to access the following bytes of the file:

1. 200,

1 block (direct pointer)

2. 4096,

2 blocks (direct pointer)

3. 40959,

40959 / 4096 = 9.999 = 10 blocks (10 direct pointers)

4. 40960

40960 / 4096 = 10

5. 4235901

4235901 / 4096 = 1034.1 = 1035 blocks.

1035 / 512 = 2.01 = 3 index blocks

1 double indirect, 3 index blocks, 1035 blocks

= 1039 blocks