**shell commands**

NOTE: 1. You can run an executable in *gdb without having compiled with the -g flag* BUT you can’t step or see the code in gdb without the debug information in the executable.

2. Stderr doesn’t buffer so output is immediate.

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

rm –rf A\*.c

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

cat stuff | grep “^W”

Count the number of .c files in the current directory.

ls -l \*.c | wc -l

Append your username to the file *goose*

echo $USER >> goose

Count the number of processes youare running with the name *banzai*

ps -u $USER | grep bash | wc -l OR ps -u $USER | grep bash -c

\*Show all instances of vim currently running on the system

ps -e | grep ‘vim’ | tr -s ' ' OR pgrep -l vim

\*Count how many times “mouse” appears in the *ﬁrst column* of ﬁle *elephant*.

cat elephant | cut -d’ ’ -f 1 | grep mouse | wc -l

\*\*There is a file called *ids*, where each line contains an ID number followed by a comment, separated by a single tab character. Store a list of the unique ID numbers a file called *uids*.

cut -f1 ids | sort | uniq > uids

\*\*A file *nums* consists of 4 space separated columns.

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

cut -d‘ ’ -f 1,3,4 nums | sort -k 3 OR sort -k 4 nums | cut -d’ ’ -f 1,3,4

\*\*Show the ﬁlenames in the current directory which are *longer than 8 characters*.

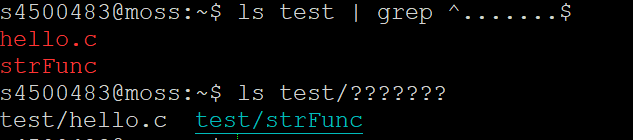
ls -ad ?????????\*

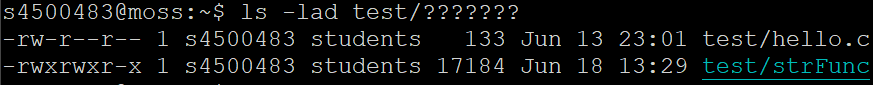
\*\*output the names (and only the names) of only the files with *un-committed modification*

svn status | grep ? | cut -c 9-

greb status, and only lines with modification(?) and then returns everything after 9th char

\*\*\*Show the names of all files from the *test* directory with names *exactly 7 chars long*.





\*\*\*Suppose you have a file *list* which contains a list of unique names (space separated), create a directory for each name in the *list*.

for name in `cat list`; do mkdir $name; done

\*\*\*Copy all files in the current directory to a new directory *../backup* and add .bak to the end of their name. eg: A.pdf would be copied to *../backup*/A.pdf.bak

mkdir ../backup && for file in \*; do cp $file ../backup/$file.bak; done

Copy all files in the current directory ending in .pdf to */dest* and add old\_ to the front of their

name. Eg: A.pdf would be copied to */dest*/old\_A.pdf.

for file in \*.pdf; do cp $file ./dest/old\_$file; done;

PATH – colon separated list of pathnames to search for commands

\*\*\*allow you to run commands from the */tmp* directory while in any directory on the system e.g. by typing *mycommand* instead of */tmp/mycommand*

export PATH=$PATH:/tmp

\*\*\*A subdirectory *bin* under your home directory containing commands. You wish to be able to r*un commands from that directory (as well as the usual commands) anywhere in the system*.

export PATH=~/bin:$PATH

\*\*\*Add /srd/bin to the list of directories to *search for commands*.

PATH=$PATH:/srd/bin

\*\*\*Output the name of the first directory which will be *searched for executables*. (Hint : is a

separator.)

echo $PATH | cut -d ':' -f

\*\*\* UID PID PPID C STIME TTY TIME CMD

uqjfenw1 58653 58652 0 23:09 pts/92 00:00:00 -bash

uqjfenw1 58654 58653 0 23:10 pts/92 00:00:00 ./do\_things

uqjfenw1 58897 58653 0 23:10 pts/92 00:00:00 vim

uqjfenw1 58899 *58654* 0 23:10 pts/92 00:00:00 *mark* <defunct>

uqjfenw1 58702 58653 0 23:10 pts/92 00:00:00 sleep

Remove *mark* from the process list above.

kill -9 58654

**couple reminder**

An array of 12 high precision ﬂoating point values.

double foo[12];

A pointer to a function which takes *three arbitrary pointers* and a string and returns a string

char\* (\*foo)(*void\*, void\*, void\**, char\*)

A pointer to a function which takes two parameters (*A pointer to a function which takes an array of strings and returns a string*) and returns a function pointer (*A pointer to a function which takes an array of strings and returns a string*).

typedef (char\*)(\*)(char\*\*) bar;

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

\*\*\*A pointer to a function which takes an array of function pointers (each taking a string and returning a string) and returns one of the function pointers it was passed.

(char\*) (\*(\*foo)((char\*)\*[](char\*))) (char\*)

{Grey is for foo as pointer to function

With Pink representing its parameter

The red part is for the return}

**virtual memory**

NOTE: 1. 1KiB pages := 1024 (2^10) bytes in page

1MiB := 1048576 (2^20) 1GiB := 1073741824 (2^30)

2. program, process, pointer -> memory address -> virtual memory -> page

physical memory, RAM -> memory address -> physical memory -> frame

Page = virtual address / page size

Offset = vir % page size

Phy = frame \* page size + offset

How much memory is needed for the page table using 512MiB of contiguous memory?

(multi-level, 4byte page table entry, 4KiB(2^12) pages)

512MiB = 2^29Bytes

How many pages is that?

2^29/2^12 = 2^17 pages (Memory / page size). So, we need 2^17-page table entries.

How many entries fit per page of the table?

2^12/2^2 = 2^10 = 1024 entries per page of table.

How many pages of table do we need?

2^17/2^10 = 2^7

We need an extra page for the top level of the table.

So, to store the table we need 2^7 +1 pages = (128+1) \* 4KiB.

What causes *page faults*?

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

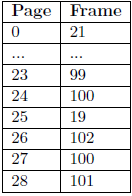
What causes *segmentation faults (SIGSEGV)*?

access invalid page OR access read only page

The process performs a read operation on its memory address 1 443 176. What signal (if any) will the process receive from the kernel?

Calculate the page = virtual address / page size

The frames were physical, and the pages were virtual. A process sending a request to the kernel would use the virtual address. If the virtual page 44 exists and so there shouldn’t be a signal, otherwise SIGSEGV signal.

There are two things which are unusual about this page table. What are they, and why are they unusual?

1. two pages map to the same physical address, wasting one page

2. page 0 is a valid page, usually reserved for error checking

**directory listing**

NOTE: 1. if *link* is readable, I also need to define whether the *target file* is readable.

2. For *.sh (script or interpreted language files) files*, you need to be able to read the file to read what commands to run

3. symbolic link, the permission to access this file depends on the file it points to

How many *subdirectories* does a folder contain?

14 – 2(the . and ..) = 12

the link count for a directory is 2 plus the number of direct subdirectories it has.

Suppose that the above filesystem is full (and no space is reserved), and then rm full (one of file) is executed. What is the *largest file* which could then be created on the filesystem (*as a regular user*)?

*No block size is known*, so the max file will be equal to what was deleted. Alternatively, the inode setup isn’t known either so maybe the max size is less than the size of full.

OR

Assuming block size is 4KiB, then the size of file “full” takes 267136 KiB, which takes 267136/4 blocks. Therefore, remove this file release 66784 blocks, the largest file could then be created is 267136 KiB.

Suppose *rm -rf /tmp/\** is executed by the *administrator* (root). How much additional diskspace

will be available? Explain your answer. (Adding up the file sizes in /tmp gives 26K.)

Undetermined. *As the block size is 1K*, there may be internal fragmentation for some of the files. E.g. *temp* has 2 files: - File A (1.5 K) - File B (24.5K)

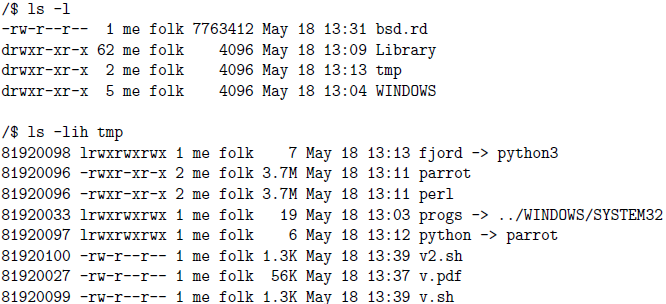
If we delete both file A & B, we would have freed 27K of space/blocks

Also note that hidden files will not be deleted. Therefore, some files will not be removed, and the corresponding disk space will not be freed.

If the current directory were removed. How much *data (file contents) would be lost*?

e.g. The files one (7602245), two (same as two\_same\_inode, 7602242) as well as three (7602247) will be lost. That’s because four has 2(link count is 2) references but the other one is not in the same directory, and one\_link is just a soft link to one so doesn’t count.

That makes 1K + 2K + 3K = 115 KB

Suppose the above filesystem is full (and no space is reserved), and then *rm v.sh v2.sh* is executed (in */tmp*). What is the *largest file* which could then be created on the filesystem (as a regular user)?

Since blocks are 1KB big, 2\*ceil(1.3KB) = 4KB

What will be the effect of executing the following commands (as me):

*ln parrot python3*

*rm parrot*

on these files (and why):

1. fjord

Removing parrot doesn't mean python3 is removed.

So, fjord can still access python3 meaning it is unaffected.

1. python

Dangling soft link

1. perl

perl is hard linked, therefore when parrot is removed, the inode counter goes down.

1. python3

nothing (bc of inode count being greater than 1)

D) Executing [2 marks]

*ls -l progs/xcode-select*

in /tmp gives:

-rwxr-xr-x 1 me me 118280 May 18 13:02 progs/xcode-select

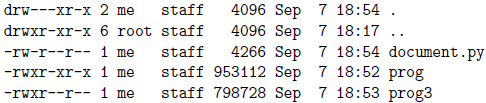
For me to successfully execute:

*progs/xcode-select*

in the */tmp* directory, what must be true about the WINDOWS/SYSTEM32 directory? (Try to be as specific as possible).

the user 'me' can execute (change directory into) WINDOWS/SYSTEM32

and thus, has the right to execute bits set on that directory

Attempting to execute *./prog3* fails (as me). Why?

Can’t access anything in a directory without x, even if you know the directory’s name.

**Network**

NOTE: 1. *The number of available hosts* can be calculated using 2^number of 0’s in in subnet

mask - 2 (minus 2 because of network and broadcast addresses) - used addresses

2. *Host bits*: number of 0’s in in subnet mask

δ = max{BX} – min{BX}

compare max{BX} and min{BX} in binary

count how many first bits are identical -> *length of the subnet mask*

e.g. if length is 23: Sm -> 11111111.11111111.11111110.00000000 -> 255.255.254.0

Now the *network address* can be found by bitwise AND the subnet mask with any element from BX

Na = AND(Sm, BX) (pick the easiest for BX)

e.g. Na = AND(255.255.254.0, 10.10.97.0) = 10.10.96.0

Finally, the *broadcast address, Ba* can be determined by bitwise OR the network address and the bitwise inverse of the subnet mask. Hence:

Ba = OR(Na, Sm-1)

e.g. Ba = OR(10.10.96.0, 0.0.1.255) = 10.10.97.255

min: 10.17.21.2 max: 10.17.21.95

2: 00000010 95: 01011111

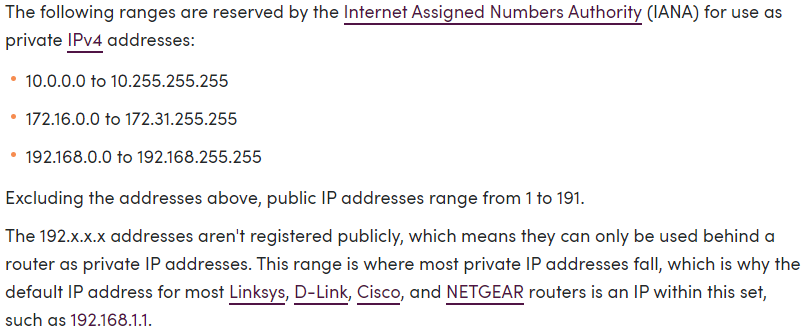
length of the subnet mask: 24+1=25

11111111.11111111.11111111.10000000

Sm: 255.255.255.128

Na = AND(255.255.255.128, 10.17.21.2) = 10.17.21.0

Ba = OR(Na, Sm-1) = OR(10.17.21.0, 0.0.0.127) = 10.17.21.127



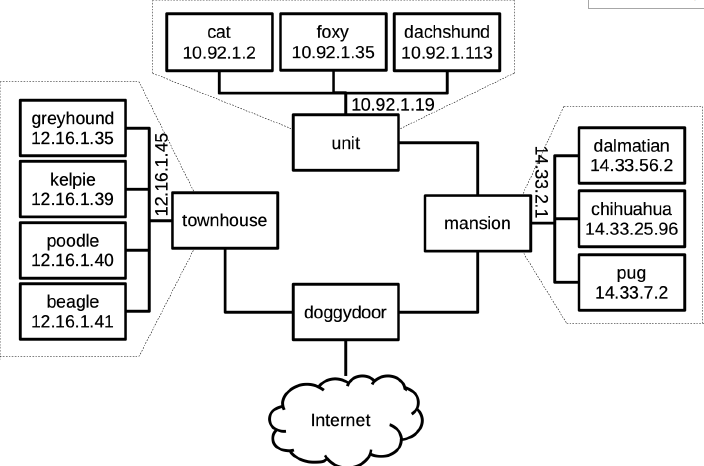
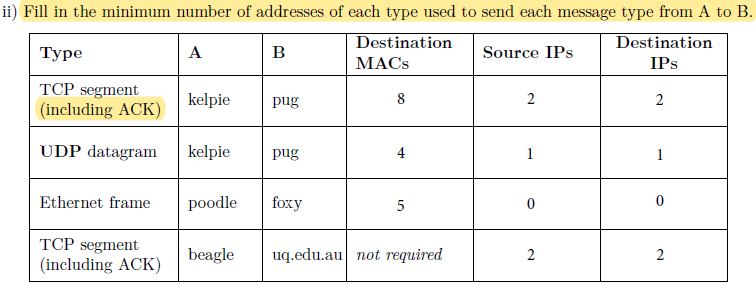
NOTE: 1. TCP: send message back *always.*

2. TLB: Memory cache for pages and page to frame references.

a) one level: if in the TLB 0, otherwise 1 page read

b) two level: if in the TLB 0, otherwise 2 pages read

*number of memory access*: 1 access if in the TLB, 3 if *two level*, 2 if *one level*

*Kelpie* -> dest MAC adder *townhouse* -> dest MAC adder *doggydoor* -> dest MAC adder *mansion* -> dest MAC adder *pug* -> dest IP dest port / source IP source port -> *pug*

Kelpie -> pug MACs: 4 sourceIP:1 destIP:1 (send message)

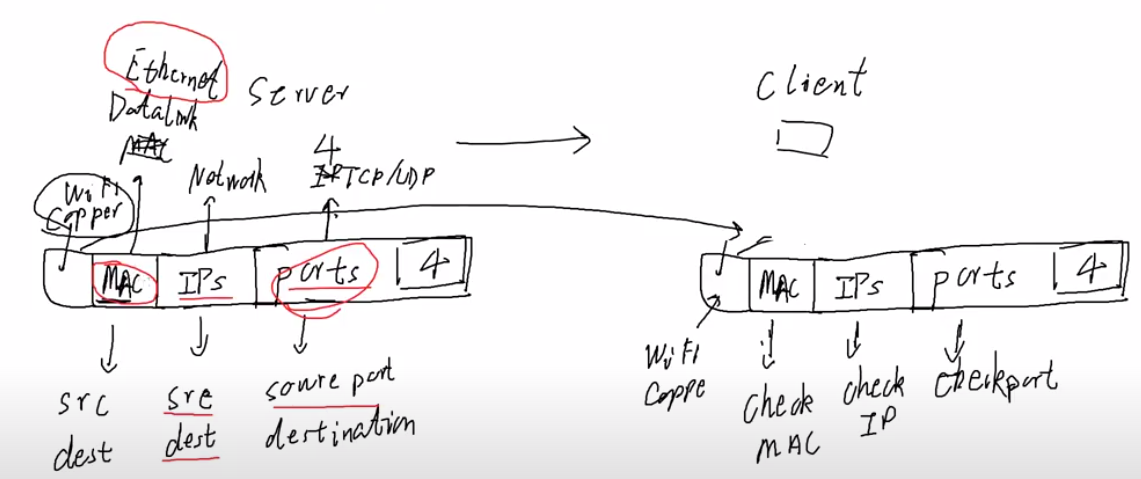
(ACK) pug -> kelpie MACs: 4 sourceIP:1 destIP:1 (reply ACK)

Total 8 2 2

UDP packet do not need to reply ACK

Ethernet 5 destination MACs 0 0 (cuz ethernet layer (mac) is higher than layer for IPs)

TCP segment to uq.edu.au 2 source IP 2 Destination IP



There is an upper limit on the number of nodes between L2 and X2 if they are to communicate.

What determines this upper limit?

The *time-to-live(TLL)* which is the number of hops that a packet is permitted to travel before being discarded.

If X1 and X2 had their *IP addresses changed to 192:168:1:1 and 192:168:1:2 respectively*; what else would need be true about the network shown above?

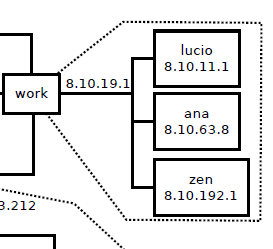
192.168.0.0 are private IP addresses. Therefore, there will needed to be *an additional router to NAT these addresses* if X1 and X2 want to communicate to the internet.

X2 performs NAT for this network. *What is NAT and why is it necessary*?

what: network address translation

why: all computers are given a private address on a network (10.x.x.x, 172.16.x.x,192.168.x.x). These IP addresses cannot be used to access the internet. *NAT performs a translation from private address to public address, so computers can access the internet.*

Network monitoring shows that connections to *lucio are preceded by UDP traffic to zen*. Give

a likely explanation for this.

I think it’s because zen is a DNS server containing the information about lucio’s IP, so incoming messages ask zen using UDP what lucio’s IP is before they talk to lucio.

\*A student's server has crashed, and they quickly restart it, however the *server is unable to use the port they previously used*. What have they forgotten to do?

Forgot to use the setsockopt() function with SO\_REUSEADDR as an option

**Network layer**

|  |  |
| --- | --- |
| layer1: **physical** | Air & EM waves, voltage & wire |
| layer2: data **link** | MAC address, Ethernet frame, Wi-Fi, infrared, Carrier pigeon |
| layer3: **network** | IP address, Routing |
| layer4: **transport** | Port, UDP, TCP |
| layer5: **application** | Socket, URL, HTTP, SSH, Putty, Netcat |

Which layer(s) are affected by a cable change from copper to a WiFi connection?

physical, possibly link (if different hardware)

Name a reliable transport layer protocol: TCP

purpose of a **network gateway**: Router interface connected to the local network. It’s purpose is to send packets out of and receive to the local network.

Why is *fragmentation*(**数据分片**) 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 dats, so the data can be spread (physically) across the disk.

To break a 4.51.16.0/21 block into networks, each containing 31 machines (hosts). What is the maximum number of such networks they could have?

Number of network addresses = 2^(32-21) = 2048

Each subnetwork should hold 31 machines (31 different IP address), including broadcast and network addresses this is 33. Requires 6 bits per subnetwork (does not fit 2^5).

Number of subnetworks =floor(2048 / (2^6)) = 32 networks

**“UNIX” filesystem**

Space that direct pointers take => dir \* block\_size = 36,864 bytes

+ (indir \* Pointers\_per\_block \* block\_size) = 2134016 bytes so the (indir) indirect pointers have a range from 36864 (incl) to 2134015 bytes

+ (double\_in \* Pointers\_per\_block^2 \* block\_size) = 539004928 bytes. Range = 2134016 (incl) to 539004927 bytes (counting bytes start at 0).

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

Pointers per block: block size / block pointers size= 256 blocks

Total no. of pointers: (dir) + (256)\*(indir) + (256^2)\*(double\_in) = 131,593

Size = no. of pointers \* block size = 131,593 \* block\_size = 539,004,928 bytes or

539,004,928/ 2^10 = 526,372 KiB

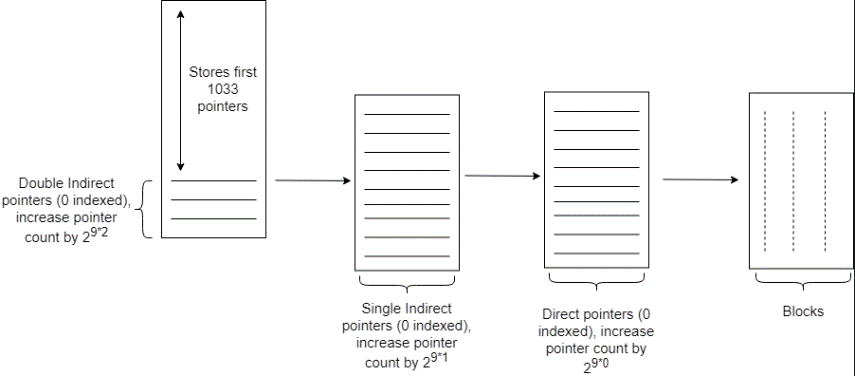
ALT: 4 KiB \* [(dir) + (indir)\*256 + (double\_in)\*256^2] = 526,372 KiB

New modified quantities (9 direct, 2 single\_indirect, 1 double\_indirect, 1 triple\_indirect)

Max size = 4 KiB \* [(dir) + (indir)\*256 + (double\_in)\*256^2 + (tri\_in)\*256^3] = … KiB

Difference = … - 526372 = 66846720 KiB

How many blocks (in total) must be accessed to read the following blocks from a file:

0, 1, 2, 4012, 8009

Block 0 ⇒ Direct block 0

Block 1 ⇒ Direct block 1

Block 2 ⇒ Direct block 2

Block 4012

We want to find: l1, l2, l3 in the equation 4012 = 1033 + (l1 \* 22\*9 + l2 \* 21\*9 + l3 \* 20\*9) Where l1, l2, l3 represent the double indirect, single indirect and direct pointer count respectively. We want to use the largest (non-negative) values for l1, l2, l3. Clearly l1=0 since l1 increases the pointer count by 2^18. So, 4012 = 1033 + (0\*2^(2\*9) + l2\*2^(1\*9) + l3\*2^(0\*9))

2978 = l2\*2^(1\*9)+ l3\*2^(0\*9)

Largest value for l2 is l2 = floor(2978/2^9)=5

Meaning 2978 = 5219+ l3\*2^(0\*9) ⇒ l3=418

Thus, Double Indirect [0] -> Single Indirect [5] -> Direct [418] ⇒ Block 4012

Similar reasoning Double Indirect [0] -> Single Indirect [13] -> Direct [350] ⇒ Block 8009

*Total amount of blocks accessed = 1+1+1+3+2 = 8*

**fork()**

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?

In the lower diagram it appears that it wasn’t reaped by its parent, nor the init process.

If we assume the downward arrow is the reaping(ending of the process). The top diagram appears to be reaped by init (as its parent has died)

Which C function can be *used to test if a child process has terminated*?

waitpid

Which function is *used to send a signal to a process*? kill

Which pthreads functions are *used to terminate a thread*? pthread\_exit or pthread\_cancel

*retrieve the exit status of a thread*? pthread\_join

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

Write a shell command or C code to *send a SIGINT to a process with pid= 517*.

*(note use kill -l to see a list a signal name)*

kill(517, 2); OR kill -2 517 OR kill -SIGINT 517

The current process has a child with pid=451. Write a code fragment to *print out its exit*

*status.*

int status;

waitpid(451, &status, 0);

if (WIFEXITED(status)) printf("%d\n", WEXITSTATUS(status));