Review:
> File permissions and directories
For a directory what does the execute bit imply?

> What am I describing and where is this useful?
"Even though directory has rwx only the owner can rename or delete a subdirectory."

I logged in therefore I am , Descartes 1637

My process has a uid and euid.

If I run it under sudo which one has changed?
If I set the setuid bit which one has changed?

```
int main() { // who am i?

  struct passwd *pw;
  pw = getpwuid(getuid());

  printf("getuid: %d, Hello %s,\n",
                            getuid(), pw->pw_name);

  pw = getpwuid(geteuid());
  printf("geteuid(): %d, You are effectively %s,\n",
                           geteuid(), pw->pw_name);

  printf("Opening file %s...\n", filename);

  FILE* f = fopen(filename,"r");
  if( ! f ) quit("fopen failed");

  if( stat(filename, &s) !=0 ) quit("stat failed");
  size_t size = s.st_size;
  char* buffer = malloc(size);
  size_t bytesread = fread(buffer, 1, size, f);
  fclose(f);
  fwrite(buffer, 1, bytesread, stdout);
  free(buffer);
}
```

An example bash script

```
#!/usr/bin/env bash
OTHERUSER=$1
if [[ "$OTHERUSER" == "" ]]; then
 echo 'Specify username e.g. sshd '
 exit 1
fi

sudo chown "$OTHERUSER" secret.txt
sudo chmod 400 secret.txt

sudo rm a.out 2>/dev/null
gcc hal.c –o myprogram
sudo chown "$OTHERUSER" myprogram

ls –al
```

How do I create directories and symlinks in code?
Which of the following will fail to create a directory or symbolic link?

```
01  int main() {
02   mkdir("dir1", 0700);
03   mkdir("dir1/subdir", 0700);
04   mkdir("dir2", 0600);
05   mkdir("dir2/subdir", 0700);
06   mkdir("dir3", 0500);
07   mkdir("dir3/subdir", 0700);
08   symlink("dir1/subdir","quick1");
09   symlink("dir2/subdir","quick2");
10   symlink("dir3/subdir","quick3");
11   return 0;
12 }
```

> How do I mount and unmount a filesystem?
How is /etc/fstab used ?

> What is a loop back filesystem?

> What does a process contain? (Version 2)

virtual memory
threads, pid, ppid
open file descriptors- files,pipes,sockets
uid, euid
pwd
meta information- Total CPU time. Running status
constraints - ulimits
thread & process priority
umask

> What is RAID? Why is it necessary?

Making filesystems resilient:
RAID - "Redundant Array of Inexpensive Disks"

> Examples of virtual files in /proc:

cat /proc/sys/kernel/random/entropy_avail

hexdump /dev/random
hexdump /dev/urandom

> File Globbing

What is it?

How do you prevent it?

Who does it?

> The impossible filesystem! Fun things to do with /proc (why does it exist?)

cat /proc/meminfo
cat /proc/cpuinfo
cat /proc/cpuinfo | grep bogomips

cat /proc/meminfo | grep Swap

cd /proc/self
cat maps