3.
```
************Makefile*************

TARGET=forme
OBJS=fileone.o filetwo.o filethree.o
CFLAGS=-g -Wall -O2
LIBS=-lm

all: ${TARGET}

${TARGET}: ${OBJS}
    ${CC} -o ${TARGET} ${OBJS} ${LIBS}

clean:
    rm -f ${TARGET} ${OBJS} core*
```

4.
```
************question4**************

#!/usr/bin/perl

use strict;
use warnings;

my $input = <STDIN>;
my @fields = split(',', $input);
if ($fields[0] eq '$GPGGA') {
    print "$fields[2]$fields[3] $fields[4]$fields[5] $fields[9]\n";
}
```

5.

Network (DDN)	Single Assignable IP (DDN)	Netmask (CIDR)	Broadcast (DDN)
a) 80.68.0.0	80.68.105.5	/17	b) 80.68.127.255
101.210.214.64	c) 101.68.105.120	d) /26	101.210.214.127

```
a)
IP in binary      --> 01010000 01000100 01101001 00000101
Netmask in binary --> 11111111 11111111 10000000 00000000

To get Network address, the IP address is bitwise '&' with the Netmask address.

        IP:        01010000 01000100 01101001 00000101
    (&)     Netmask:   11111111 11111111 10000000 00000000
    ___________________________________________________________
        Network:   01010000 01000100 00000000 00000000

So... Network (DDN) = 80.68.0.0

b)
To get Broadcast address, the IP address is bitwise '|' with the 1's compliment
of the Netmask address.

        IP:        01010000 01000100 01101001 00000101
    (|)     ~Netmask:  00000000 00000000 01111111 11111111
    ___________________________________________________________
        Broadcast: 01010000 01000100 01111111 11111111

So... Broadcast (DDN) = 80.68.127.255

c) and d)
Network in binary   --> 01100101 11010010 11010110 01000000
Broadcast in binary --> 01100101 11010010 11010110 01111111
```

```
Since last 6 bits of Network address are zero, Netmask (CIDR) must be /26.
IP address can be 80.68.105.(64-127).
Will choose a single assignable IP (DDN) as 101.68.105.120:

        IP:         01100101 11010010 11010110 01111000
    (&) Netmask:    11111111 11111111 11111111 11000000
      ___________________________________________________________
                Network:   01100101 11010010 11010110 01000000 (matches given Network)



        IP:         01100101 11010010 11010110 01111000
    (|)     ~Netmask:  00000000 00000000 00000000 00111111
      ___________________________________________________________
        Broadcast: 01100101 11010010 11010110 01111111 (matches given Broadcast)

So... Single Assignable IP (DDN) = 101.68.105.120
      Netmask (CIDR)             = /26
```

6.
a)

```
$ sqlite3 question6.db
sqlite> create table output(latitude text, longitude text, elevation text);
```

b)

```
sqlite> insert into output values('4439.3381N', '06744.4518W', '5.6');
sqlite> insert into output values('4439.3381N', '06744.4518W', '5.6');
sqlite> insert into output values('4439.3381N', '06744.4518W', '5.6');
```

c)

```
sqlite> select * from output order by rowid desc limit 200;
```

7.

```
**************question7***************

#!/usr/bin/python

import sys

linecount = 0
charcount = 0

if(len(sys.argv[1:]) == 0):
    print "Expect at least one argument"
    exit()

for f in sys.argv[1:]:
    try:
        fh = open(f,'r')

        for line in fh:
            line = line.replace(" ", "")
            linecount += 1
            charcount += len(line) - 1

        print "Total lines:", linecount
        print "Total characters:", charcount
    except:
        print "Cannot open", f
fh.close()
```

8.
a)

No, this code does not execute properly under all conditions.

b)
This code may fail because of the initialization of the desitination address 'data'. The max size 'data' can hold is 4096 bytes. If 'count' is larger than 4096 bytes, not all data will be copied from userspace. A fix would be to dynamically allocate the exact amount of memory needed for the destination address.

c)
Yes, this code successfully protects shared resources.

d)
Shared resources are protected because the mutex lock is acquired before the main process 'gpiod_set_value()' is called. The lock is then released after the process is complete.

e)
Code always protects resources.

9.
```
[0-9]{1,2}[:][0-9]{2}[ ]((am)|(AM)|(a.m.)|(A.M.)|(pm)|(PM)|(p.m.)|(P.M.))
```

10.
```
***********main.c************

#include <stdio.h>
#include <sys/stat.h>

int main(int argc, char *argv[])
{
    int i;
    long int numbytes = 0;
    struct stat fileStat;

    if (argc < 2) {
        printf("Error: expect at least one argument.\n");
    }


    for (i = 1; i < argc; i++) {
        stat(argv[i], &fileStat);
        numbytes += fileStat.st_size;
    }

    printf("Size: %ld\n", numbytes);

    return 0;
}
```

11.
a)
```
sudo chown -R $pi /usr/src
```

b)
```
ln -s /var/lib/systimer/logs/abc /usr/arm/opt/bin/foobar
```

c)
```
chmod -R go+rx /opt/ngspice
```

d)
```
grep -r --include="[0-9][0-9]" "(01)|(12)|(23)|(34)|(45)|(56)|(67)|(78)|(89)"
```

12.
My next action would be to contact the system administrator providing the reason and time I logged in with the root password. I would also notify the system administrator of the odd files connected with Trinity’s account.

13.
The cron job runs at every 45th minute of every 12th hour on every first day of the month and on every Monday and Friday.

14.
```
scp -P 666 simulation wizard@summit.ornl.gov:~/
```

15.
```
$ ps
$ man thd
$ thd --listevents
```

The process /usr/sbin/thd, triggerhappy, is a global hotkey daemon. It watches and tracks all of the system’s input devices for any key, switch, or button presses.