Domanda A:

-- Sia dato il seguente script

```
#!/bin/bash
for i in `env`
do
  NAME=`echo $i | cut -d '=' -f 1`
  VAL=`echo $i | cut -d '=' -f 2`
  if test $NAME = "USER"
   then
     if [ $VAL = "root" ]
      then
      echo "With great power comes great responsibility"
     fi
  fi
done
```

1) che cosa stampa a video quando viene lanciato da un ipotetico utente "user", (UID 1000), assumendo che il file sia in questo stato sul filesystem?

-rwxr-xr-x 1 root root 252 Jun 10 16:38 test.sh

Lo script estrae le variabili d' ambiente, e controlla se quella chiamata USER ha valore “root”. In questo caso stampa a video “With great power comes great responsibility".
Dunque, lo script eseguito dall' utente “user” non stampa nulla a video.

2) cambierebbe qualcosa se venisse lanciato da root (UID 0)?

Per quanto detto prima, lo script stampa “With great power comes great responsibility"

Domanda B:

Quali problema presentano le seguenti regole per iptables? Come possono essere corrette?

1) iptables -A INPUT -o eth1 -s 10.0.0.2 -j DROP

2) iptables -t nat -A PREROUTING -s 192.168.0.1 -j SNAT --to 8.8.8.8

3) iptables -A INPUT -i lo -j DROP

1) La regola tenta di selezionare I pacchetti in base all' interfaccia da cui andranno ad uscire tra quelli che passano nella tabella di INPUT (ovvero quelli destinati alla macchina stessa): la selezione non ha senso. Una ragionevole correzione è sostituire la selezione sull' interfaccia di uscita (opzione -o) con quella di entrata (-i).

2) La regola sta tentando di effettuare Source Network Address Translation (SNAT) prima che la decisione di routing in entrata sia presa da netfilter, quando l' azione di SNAT va fatta dopo la decisione di routing in uscita. La regola corretta cambia -A PREROUTING in -A POSTROUTING.

3) La regola scarta tutto il traffico sull' interfaccia di loopback locale. Questa interfaccia non dovrebbe mai essere bloccata. La correzione cambia -j DROP in -j ACCEPT.

Domanda C:

-- Sia data la seguente funzione in linguaggio C:

```
double duplicate(double a){
  return 2.0*a;
}
```

E' possibile utilizzarla all' interno di un modulo per il kernel Linux?
Suggerite una possibile soluzione nel caso questo non sia fattibile.

No, non è possibile utilizzarla all' interno di un modulo per il kernel Linux, in quanto fa utilizzo di numeri a virgola mobile, che non sono supportati.

Una possibile soluzione, considerato che la funzione semplicemente duplica un valore è sostituirla con una che accetti un intero come parametro e ne restituisca uno, e dunque faccia uso di una moltiplicazione su interi, o equivalentemente, uno shift a sinistra di una posizione.

Domanda D:

Dato il seguente sorgente:

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>

int flip;
void my_handler_1(int param){
perror("Work in progress");
}
void my_handler_2(int param){
flip=1;
}
void my_handler_3(int param){
exit(0);
}

int main(int argc, char* argv[]){

struct sigaction handler_1,handler_2,handler_3,old_handler_1,old_handler_2;
handler_1.sa_handler=&my_handler_1;
handler_2.sa_handler=&my_handler_2;
handler_3.sa_handler=&my_handler_3;
sigaction(SIGQUIT,&handler_1,&old_handler_1);
sigaction(SIGUSR1,&handler_2,&old_handler_2);
```

```
while(1){
fprintf(stderr,"Yawn...\n");
if (flip){sigaction(SIGCONT,&handler_3,&old_handler_1);}
flip=0;
sleep(5);
}
return 0;
}
```

Cosa succede se al processo di cui sono dati i sorgenti vengono inviate le seguenti sequenze di segnali nell' ordine dato?

Sequenza 1	**Sequenza 2**	**Sequenza 3**	**Sequenza 4**
QUIT	QUIT	QUIT	KILL
USR1	QUIT	USR1	CHLD
TERM	QUIT	CONT	CONT

Sequenza 1: la ricezione di SIGQUIT causa la stampa di “work in progress” da parte dell' handler assegnato, quella di SIGUSR1 cambia il valore della variabile globale flip, impostandolo a 1, SIGTERM termina l' esecuzione del processo.

Sequenza 2: Viene stampato 3 volte “work in progress” per la ragione menzionata in precedenza

Sequenza 3: la ricezione di SIGQUIT causa la stampa di “work in progress” da parte dell' handler assegnato, quella di SIGUSR1 cambia il valore della variabile globale flip, impostandolo a 1, e assumendo che il processo non abbia esaurito il suo quanto di tempo, questo causa l' impostazione di un nuovo handler per SIGCONT (my_handler_3). Di conseguenza, al momento della ricezione di SIGCONT viene eseguito il nuovo handler che causa la terminazione del processo in quanto contiene una chiamata a exit()

Sequenza 4: Il processo termina al momento della ricezione di SIGKILL.

Domanda C:

Quale errore concettuale è presente nel seguente frammento di codice e come è possibile risolverlo? Indicare cosa va modificato affinché il programma seguente sia in grado di trasmettere/ricevere dati.

```
int main(int argc, char *argv[]) {
  int socket_fd;
  struct sockaddr_in name;
  struct hostent* hostinfo;
  if ( (socket_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    perror("Socket creation error");
    return -1;
  }
  hostinfo = gethostbyname ("127.0.0.1");
  errorlog(hostinfo,NULL);
  name.sin_addr = INADDR_ANY;
  name.sin_port = htons (9999);
  listen(socket_fd,10);
  /*trasmissione/ricezione dati*/
  return 0;
}
```

Il codice precedente crea un socket TCP con l' intenzione di agire come server (l' indirizzo di ingresso è impostato a INADDR_ANY e viene chamata la funzione listen()), senza legarsi al socket. É necessario invocare la funzione bind con I seguenti parametri appena prima della listen:

```
bind(socket_fd, (struct sockaddr * ) name, sizeof(name))
```

Domanda D:

Indicate quali regole iptables scattano e, quante volte, su una macchina configurata con la seguente lista di comandi :

iptables -A INPUT -s 192.168.3.3 -j LOG

iptables -A INPUT -i lo -j ACCEPT

iptables -P FORWARD ACCEPT

iptables -A FORWARD -i lo -j ACCEPT

iptables -A INPUT -s 192.168.3.4/24 -j RATEEST

iptables -A FORWARD -d 46.59.1.2 -j ACCEPT

iptables -I INPUT 2 -s 192.168.3.3 -j DROP

iptables -I INPUT 2 -s 192.168.3.3 -j QUEUE

iptables -P OUTPUT ACCEPT

iptables -A INPUT -d 192.168.3.42 -j ACCEPT

iptables -A INPUT -s 192.168.3.7 -j DROP I

ptables -A INPUT -s !192.168.3.3 -j ACCEPT

iptables -P INPUT DROP

iptables -A FORWARD -s 192.168.3.4/24 -j RATEEST

iptables -A FORWARD -s 192.168.3.4/24 -j ACCEPT

iptables -A FORWARD -p tcp --dport 53 -j LOG

Assumendo che passino i seguenti pacchetti:

1 - 192.168.3.7:12943 -> 192.168.3.42:123, TCP

2 - 192.168.3.42:12943 -> 192.168.3.42:80, TCP

3 - 192.168.3.3:12943 -> 8.8.8.8:53, TCP

4 - 192.168.3.3:12943 -> 192.168.3.4:80, TCP

5 - 46.59.1.2:12943 -> 192.168.3.42:80, TCP

L' host abbia 2 interfacce di rete, una esposta con indirizzo pubblico 131.175.1.2/24, l' altra verso una rete interna con indirizzo 192.168.3.42/24.

Il forwarding dei pacchetti ip è abilitato.

Indicate anche quali regole sono inutili (ovvero quali regole potete rimuovere senza cambiare il comportamento della macchina), indipendentemente dal traffico che transita.

Allo scopo di risolvere questo esercizio, è utile riordinare le regole di fornite, dividendole per tabella, e sostituendo gli inserimenti ad un indice preciso con delle azioni di append.

Tabella INPUT:

iptables -A INPUT -s 192.168.3.3 -j LOG

iptables -A INPUT -s 192.168.3.3 -j QUEUE

iptables -A INPUT -s 192.168.3.3 -j DROP /* inutile: questa regola scarta pacchetti che non arriveranno mai fino a qui perchè accodati verso userspace dalla precedente*/

iptables -A INPUT -i lo -j ACCEPT

iptables -A INPUT -s 192.168.3.4/24 -j RATEEST

iptables -A INPUT -d 192.168.3.42 -j ACCEPT

iptables -A INPUT -s 192.168.3.7 -j DROP

iptables -A INPUT -s !192.168.3.3 -j ACCEPT /*questa regola rende inutile la policy DROP in quanto accetta tutti I pacchetti, fatto salvo per quelli con sorgente 192.168.3.3 che sono già gestiti a monte.*/

iptables -P INPUT DROP

Tabella FORWARD:

iptables -A FORWARD -i lo -j ACCEPT

iptables -A FORWARD -d 46.59.1.2 -j ACCEPT

iptables -A FORWARD -s 192.168.3.4/24 -j RATEEST

iptables -A FORWARD -s 192.168.3.4/24 -j ACCEPT

iptables -A FORWARD -p tcp --dport 53 -j LOG

iptables -P FORWARD ACCEPT

Tabella OUTPUT:

iptables -P OUTPUT ACCEPT

1 - 192.168.3.7:12943 -> 192.168.3.42:123, TCP – Il pacchetto viene analizzato dalla tabella di INPUT in quanto destinato alla macchina e scartato dalla regola iptables -A INPUT -s 192.168.3.7 -j DROP

2 - 192.168.3.42:12943 -> 192.168.3.42:80, TCP - Il pacchetto viene analizzato dapprima in uscita dalla tabella di OUTPUT, in quanto esce dalla macchina , dove viene accettato dalla policy, Successivamente viene analizzato dalla tabella di input, dove viene accettato da iptables -A INPUT -i lo -j ACCEPT

3 - 192.168.3.3:12943 -> 8.8.8.8:53, TCP - Il pacchetto viene analizzato dalla catena di FORWARD, dove fa scattare la regola iptables -A FORWARD -p tcp --dport 53 -j LOG che causa il fatto che venga tenuta traccia del suo passaggio nei log di sistema, per poi essere accettato dalla policy.

4 - 192.168.3.3:12943 -> 192.168.3.4:80, TCP Il pacchetto viene analizzato dalla catena di FORWARD, dove viene accettato dalla policy.

5 - 46.59.1.2:12943 -> 192.168.3.42:80, TCP l pacchetto viene analizzato dalla tabella di INPUT in quanto destinato alla macchina e accettato dalla regola iptables -A INPUT -d 192.168.3.42 -j ACCEPT

Domanda E:

Assumendo che un programma contenga nel proprio sorgente la seguente dichiarazione di funzione:

```
void my_handler(int param){
        fprintf(stderr,"Received Kill Signal"); }
```

E la seguente invocazione di primitiva:

```
[...]
sigaction(SIGKILL,&handler,&old_handler);
[...]
```

Assumendo che il programma giri con Process ID 1024, che effetto ha il seguente comando?
kill -SIGTERM 1024

Il comando termina l' esecuzione del processo: esso invia un segnale di terminazione regolare per cui non è stato istanziato un handler diverso da quello di default.

Domanda F:

Una macchina in vostro possesso non pare essere in grado di raggiungere altri host via Wi-Fi. Esaminando la configurazione di rete della macchina tramite ip ottenete I seguenti risultati :

```
[root@grayswandir tmp]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
   inet6 ::1/128 scope host
     valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
   link/ether 00:1f:16:1e:03:67 brd ff:ff:ff:ff:ff:ff
   inet 192.168.42.82/24 brd 192.168.42.255 scope global eth0
   inet6 fe80::21f:16ff:fe1e:367/64 scope link
     valid_lft forever preferred_lft forever
3: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000
   link/ether 00:21:6a:40:f5:a8 brd ff:ff:ff:ff:ff:ff
   inet 192.168.42.105/32 scope global wlan0
```

1. Osservando l' output dei comandi fornito, quale appare essere il problema?
2. Indicate come risolverlo (il comando da eseguire)

1) l' interfaccia di rete wifi è attualmente spenta: si noti “state DOWN” nella sua descrizione
2) ip link set dev wlan0 up

Domanda G:

1. Qual è la condizione che consente il verificarsi della sindrome dell' apprendista stregone?

La sindrome dell' apprendista stregone si verifica quando, in una trasmissione via UDP con meccanismo di ACK a livello applicativo per cui valga la regola “alla ricezione dell' n-esimo ACK , invia l' n+1 esimo pacchetto di dati”, il ritorno di un ACK avviene dopo che è stato ritrasmesso il suo corrispondente pacchetto dati.

2. Descrivere il fenomeno, le possibili soluzioni e dire se la condizione non desiderata tende a risolversi autonomamente.

La ritrasmissione prematura causa la trasmissione di un ACK duplicato, a cui seguirà una intera trasmissione duplicata di dati. La condizione indesiderata è tipicamente causata da un eccesso di traffico di rete che rallenta il ritorno del primo ACK come descritto in precedenza. Considerato che UDP non è dotato di controllo di congestione, la condizione non si risolve autonomamente.
Le soluzioni in questo caso sono mirate a evitare la ritrasmissione duplicata: ad esempio, è possibile tenere traccia dei pacchetti trasmessi per ogni client socket che li richiede.

3. Quale problema si presenta nel tentare di usare il seguente frammento di codice all' interno di un modulo per kernel Linux?

```
float a,b,c;
a=b+c;
```

Il frammento di codice fa uso di float: l' uso dei numeri a virgola mobile non è consentito in kernelspace.