Primer Certamen: Inicio: 12:00 hrs - 13:30 hrs.
Todas las preguntas tienen igual puntaje.

1.- Al ejecutar who en aragorn se tiene algo del tipo:

```
agv@aragorn 10:14 PM ~$ who
fsilvav  pts/0        Oct 16 20:41 (63-236-21-190.adsl.terra.cl)
glecaros pts/1        Oct 16 11:54 (pabiertas1.ciac.usm.cl)
arivera  pts/4        Oct 16 20:31 (pc-108-134-239-201.cm.vtr.net)
```

Se pide hacer el script origen.sh que permita mostrar por pantalla el número de usuarios que trabajan en aragorn que están conectados con proveedor de servicios vía terra y aquellos conectados vía vtr.
Por ejemplo, si para este caso en aragorn ejecutamos el programa, tenemos:
$ origen.sh
Usuarios terra: 1
Usuarios vtr: 1

Hay múltiples soluciones, dos de ellas:
a) Invocando who dos veces:

```
#! /bin/bash
#Count Terra and vtr users working on the machine
echo "Usuarios terra: `who | grep terra.cl | wc -l`"
echo "Usuarios vtr: `who | grep vtr.net | wc -l`"
```

b) invocando who sólo una vez (y sin archivos temporales)

```
#! /bin/bash
#Count terra and vtr users working on the machine
mkfifo union$$
echo "Usuarios terra: `grep terra.cl union$$ | wc -l` " &
echo "Usuarios vtr: `who | tee union$$ | grep vtr.net | wc -l`"
rm union$$
```

Michael observó que la pregunta pide número de usuarios por cada proveedor. La solución previa cuenta dos veces un usuario con dos conexiones. Se tomó así con puntaje total, como hice la pauta inicialmente. La versión estricta según lo pedido sería, por ejemplo:

```
#! /bin/bash
#Count Terra and vtr users working on the machine
echo "Usuarios terra: `who | grep terra.cl |  awk '{print $1}' | sort -u | wc -l`"
echo "Usuarios vtr: `who | grep vtr.net | awk '{print $1}' | sort -u | wc -l`"
```

2.- Haga un programa que cada un segundo muestre por pantalla el número de bytes recibidos por la interfaz de red eth0.

Este programa es un parte de su última tarea.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <string.h>
#include <signal.h>

void sig_alarm(int signo)
{   return;
}

int main(void) {
```

```
    FILE *pf;
    char line[50];
    struct itimerval itval;

    signal(SIGALRM, sig_alarm);  /* with alarm, we ensure output every 1 second */
    itval.it_value.tv_sec = 1;
    itval.it_value.tv_usec = 0;
    itval.it_interval = itval.it_value;
    setitimer(ITIMER_REAL, &itval, NULL);

    while(1){
      pf = popen("/sbin/ifconfig eth0|grep \"RX bytes\" ", "r");
      fgets(line, sizeof(line), pf);
      strtok(line,":"); /* skip first text */
      printf("Número de bytes recibidos = %s\n", strtok(NULL," "));
      pclose(pf);
      pause();
    }
}
```

La solución vía script usando sleep, fue aceptada pues nada se decía sobre programa en C. Pero esta solución es menos buena debido a la imprecisión en el tiempo (no importante en este caso).
Era algo del tipo:

```
#! /bin/bash
#Display every second, the number of bytes sent by the interface 0
while :
do
  /sbin/ifconfig eth0 | grep "RX bytes" | awk -F ":" '{print $2}' | awk '{print $1}'
  sleep 1
done
```

3.- La función gethostbyaddr(..) y gethostbyname(...) no garantizan ser seguras cuando son llamadas desde un hilo. Es decir no garantiza “re-entrancia”, su invocación por parte de dos hilos en forma concurrente, puede conducir a un resultado errado. Cree la función equivalente myGethostbyaddr que sea segura al ser invocada por hilos de un proceso.

```
#include <sys/socket.h>
#include <netdb.h>
#include <pthread.h>

struct hostent *
myGethostbyaddr(const void *addr, int len, int type)
{
  static pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
  int error;

  if (error = pthread_mutex_lock(&lock))
    return NULL;
  struct hostent * ret = gethostbyaddr(addr, len, type);
  pthread_mutex_unlock(&lock);
  return ret;
}
```

4.- Para estudiar el comportamiento de partida lenta de TCP un estudiante propone lo siguiente. **Haga un programa servidor TCP que corra en puerto 12345 y descarte todo el tráfico que llegue. Por otro lado haga cliente TCP que envíe paquetes de 1024 bytes y muestre por pantalla los segundos y microsegundos desde el inicio del programa por cada paquete enviado. Asuma que el cliente se correo según:**

$ client <servidor> <puerto>

Así el estudiante espera que los tiempos resultantes deberían reflejar el incremento experimentado por la ventana para el control de congestión.

```
/*Servidor*/
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <stdio.h>

#define PORTNUMBER  12345

int main(void)
{
    char buf[1400];
    int s, ns, len;
    struct sockaddr_in name;

    s = socket(AF_INET, SOCK_STREAM, 0);
    name.sin_family = AF_INET;
    name.sin_port = htons(PORTNUMBER);
    name.sin_addr.s_addr = htonl(INADDR_ANY);
    len = sizeof(struct sockaddr_in);

    if( bind(s, (struct sockaddr *) &name, len))
      printf("bind error");
    listen(s, 5);
    ns = accept(s, (struct sockaddr *) &name, &len);
    while (recv(ns, buf, sizeof(buf), 0) > 0);
    close(ns);
    close(s);
    exit(0);
}

/*Client*/
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#define PORTNUMBER  12345

int main(int argc, char * argv[])
{
  int s, len;
  char buf[1024];
  struct hostent *hp;
  struct sockaddr_in name;
```

```
  struct timeval tv_init, tv;

  gettimeofday(&tv_init, NULL);
  hp = gethostbyname(argv[1]);
  s = socket(AF_INET, SOCK_STREAM, 0);
  name.sin_family = AF_INET;
  name.sin_port = htons(atoi(argv[2]));
  memcpy(&name.sin_addr, hp->h_addr_list[0], hp->h_length);
  len = sizeof(struct sockaddr_in);
  connect(s, (struct sockaddr *) &name, len);
  while (send(s, buf, sizeof(buf),0) > 0) {
    gettimeofday(&tv, NULL);
    if (tv.tv_usec < tv_init.tv_usec){
      tv.tv_sec--;
      tv.tv_usec+=1000000;
    }
    printf("Número se sec:usec = %i : %i\n", tv.tv_sec-tv_init.tv_sec,
          tv.tv_usec-tv_init.tv_usec);
  }
  close(s);
  exit(0);
}
```