Primer Certamen
Tiempo 120 min. Al terminar, entregue sus respuestas vía AULA.

1.- (30 pts) Desarrolle el script shell nocu.sh (number of current users), el cual muestra por pantalla el número de usuarios trabajando en la máquina donde el script es ejecutado.
Ejemplo de ejecución en aragorn:
$ ./nocu.sh
5 usuarios trabajando en aragorn

Una solución es:

```
#!/bin/bash
num_usuarios=$(who | while read user rest
do
   echo $user
done | sort -u | wc -l)

echo "$num_usuarios usuarios trabajando en $(hostname)"
```

- 7 pts. Usar comando que liste usuarios actuales
- 4 pts. Filtrar usuarios
- 4 pts. Asegurar unicidad
- 7 pts. Contar el número de usuarios
- 8 pts. Mostrar resultado incluyendo nombre de la máquina

Otra solución usando last (no cuenta usuario root en consola):

```
#! /bin/bash
last -w | grep "still logged in" | while read user rest
do
 echo $user >> user$$.txt
done
echo `sort -u user$$.txt | wc -l ` usuarios trabajando en
`hostname`
rm user$$.txt
```

Ver: http://profesores.elo.utfsm.cl/~agv/elo330/2s23/evaluaciones/Solutions/

2.- (35 pts) Suponga que cuenta con el programa nocu.sh. Cada vez que éste es ejecutado, nuestra por su salida estándar el número de usuarios trabajando en la máquina. La salida tiene el siguiente formato (fijo):
<n> usuarios trabajando en <hostname>
Donde <n> es el número de usuarios y <hostname> es el nombre de la máquina donde corre el programa.
Desarrolle el programa C maxmin.c que, invocando nocu.sh como proceso hijo cada 2 segundos, muestra por su salida estándar el número máximo y mínimo de usuarios trabajando en la máquina desde el inicio de su ejecución. Use la función setitimer() para el avance del tiempo real.

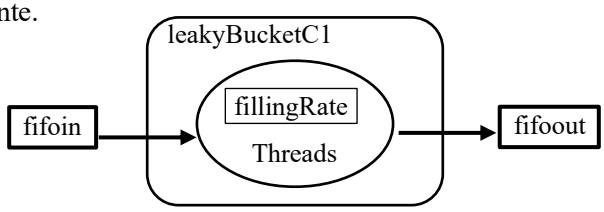

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
```

```c
#include <sys/time.h>

int max=-1;
int min=65536;  // algún valor muy grande. Otra opción es
                // programar caso especial para el primer dato.
static void compute(int signo) {
    FILE *pf;
    int nUsers;

    if ((pf = popen("./nocu.sh", "r")) == NULL) {
        perror("popen");
        exit(1);
    }
    fscanf(pf,"%i",&nUsers);
    pclose(pf);
    if (nUsers > max) max=nUsers;
    if (nUsers < min) min=nUsers;
    printf("Max=%i;\tMin=%i\n",max, min);
}

int main(void) {
    struct itimerval timerval;
    struct timeval period;
    int signo;

    period.tv_sec=2;
    period.tv_usec=0;
    timerval.it_interval=timerval.it_value=period;
    setitimer(ITIMER_REAL, & timerval, NULL);
    signal(SIGALRM, compute);
    while(1) pause();
    exit(0);
}
```

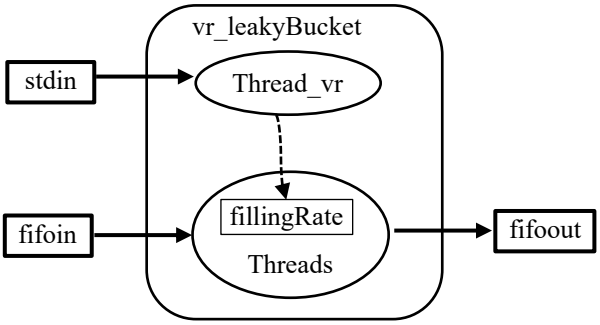
- 5 pts. Correcta invocación a setitimer para generar señal SIGALARM cada 2 segundos
- 5 pts. Correcta configuración de la función de atención de la señal.
- 10 pts. Invocar y leer desde el script usando una tubería (pipe) (Si usó pipe en lugar de popen, es OK)
- 10 pts. Comparar y actualizar valores guardados con lectura de script
- 5 pts. Imprimir salida a pantalla

Ver: http://profesores.elo.utfsm.cl/~agv/elo330/2s23/evaluaciones/Solutions/

3.- (35 pts) Se dispone del programa leakyBucketC1.c. Éste es muy similar al analizado en clases, limita la tasa de transferencia desde fifoin a fifoout a la tasa de llenado del "bucket" (fillingRate). En este programa, la tasa de llenado es constante.



A partir de leakyBucketC1.c, desarrolle el programa vr_leakyBucket.c (por variable rate leaky bucket), el cual usa una tercera hebra para leer actualizaciones de la variable fillingRate desde la entrada estándar. De esta forma el usuario puede variar la tasa de transferencia entre dos FIFOs ya creados.



La sintaxis de este programa es: $ ./vr_leakyBucket <fifoin> <fifoout>
Su tercera hebra espera por el ingreso de un valor entero, el cual usa para actualizar fillingRate.
Un ejemplo de ejecución sería (suponiendo que los FIFOs “entrada” y “salida” ya existen):
$ ./ vr_leakyBucket entrada salida
20
40
Control-C

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <sys/time.h>
#include <signal.h>
#include <unistd.h>

#define FILL_RATE 10

int bucketLevel= 0;
int BUCKET_SIZE= 1000;
int PACKET_SIZE=1; /* one char per "packet" going to fifoout */
int fillingRate= FILL_RATE; /* char per second */
// inicio código adicional
pthread_mutex_t fillRateLock = PTHREAD_MUTEX_INITIALIZER;

void * readFillRate(void * arg)
{
  int newRate;
  while (1) {
    scanf("%i",&newRate);
    pthread_mutex_lock(&fillRateLock);
    fillingRate = newRate;
    pthread_mutex_unlock(&fillRateLock);
  }
}
// fin código adicional
```

```
pthread_mutex_t bucketLock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t bucketChange = PTHREAD_COND_INITIALIZER;

void * fillBucket(void * arg)
{
    while (1) {   /* we could wait for bucket not full using
                   * another conditional variable*/
    sleep(1);
    pthread_mutex_lock(&bucketLock);
    pthread_mutex_lock(&fillRateLock);      // agregado
    if (bucketLevel+fillingRate <= BUCKET_SIZE)
      bucketLevel+=fillingRate;
    pthread_mutex_unlock(&bucketLock);
    pthread_mutex_unlock(&fillRateLock);    // agregado
    pthread_cond_signal(&bucketChange);
  }
}

int main(int argc, char * argv[])
{
  pthread_t tid;
  char c;
  int err;
  FILE *fifoin, *fifoout;

  err = pthread_create(&tid, NULL, fillBucket, NULL);
  if (err != 0){
    printf("can't create thread \n");
    exit(0);
  }
// inico agregado
  err = pthread_create(&tid, NULL, readFillRate, NULL);
  if (err != 0){
    printf("can't create thread \n");
    exit(0);
  }
  // fin agregado
  fifoin = fopen(argv[1], "r");
  fifoout= fopen(argv[2], "w");
  c = getc(fifoin);
  while (c!=EOF) {
    pthread_mutex_lock(&bucketLock);
    while ( bucketLevel < PACKET_SIZE)
      pthread_cond_wait(&bucketChange, &bucketLock);
    bucketLevel-=PACKET_SIZE;
    pthread_mutex_unlock(&bucketLock);
    putc(c,fifoout); fflush(fifoout);
    c = getc(fifoin);
  }
```

```
  exit(0);
}
```

5 pts. Creación de variable mutex para tasa de llenado
5 pts. Creación de función a invocar como hebra
5 pts. Lectura de nueva tasa desde consola
7 pts. Correcta modificación del parámetro fillingRate en thread nueva
7 pts. Correcta lectura de parámetro fillingRate en thread existente solicitando dos mutex
6 pts. Correcta creación de la hebra en main.

Ver: http://profesores.elo.utfsm.cl/~agv/elo330/2s23/evaluaciones/Solutions/

Nota: para probar este programa usted puede usar tres consolas y en cada una ejecutar:
//Consola 1
$ mkfifo salida
$ mkfifo entrada
$ cat salida
//Consola 2
$ cat <algún_archivo> > entrada
// Consola 3
$ ./vr_leakyBucket entrada salida
20
40
Control-C