Certamen Parcial 100 minutos.
Todas las preguntas tienen igual puntaje.

1.- Desarrolle el programa **presente.sh <login> <e-mail>.** Éste es un programa shell o script que envía un correo electrónico al **e-mail** señalado si el usuario del **login** indicado está en la máquina donde corre el script o tan pronto el usuario ingresa a la máquina.

Por ejemplo, en Aragón puedo correr:
%presente.sh cjana newton@fisica.cl
Así el señor **newton** recibirá un correo inmediatamente o tan pronto el usuario cjana se conecte a Aragón.

```
#! /bin/bash
#Busca presencia o aparición de usuario $1
smtp="smtp.elo.utfsm.cl"
until who | grep $1
do
  sleep 3
done

Mail_data()
{
echo "HELO $smtp"
echo "MAIL FROM: observador@elo.utfsm.cl"
echo "RCPT TO: $2"
echo "DATA"
echo "Subject:Alerta de presencia"
echo "From: observador@elo.utfsm.cl"
echo "To: $2"
echo "Llego el señor $1."
echo "Atte., Observador"
echo "."
}
```

Mail_data $1 $2 | telnet $smtp 25 >> /dev/null

2.- Se desea buscar el identificador de proceso más grande posible. Para esto alguien sugiere como algoritmo crear un proceso hijo, recordar su pid (identificador de proceso), terminar el proceso y luego crear uno nuevo. La idea es repetir este proceso hasta que el nuevo pid sea menor que el anterior.

Sin cuestionar la idea, haga un programa en C que arroje el máximo pid que se podría obtener programando este algoritmo.

```
#include <stdio.h>
#include <stdlib.h> /* for exit */
#include <sys/wait.h>

int main(void)
{
  pid_t pid_old, pid_last;
  pid_last=getpid();
  do {
    pid_old=pid_last;
    if ( (pid_last = fork()) < 0) {
      printf("fork error\n");
      exit(-1);
    }
    else if (pid_last == 0) {              /* child */
      exit(0);
```

```
    }
    wait(NULL);
  } while(pid_last > pid_old);
  printf("Mayor pid =  %i\n",pid_old);
  return(0);
}
```

3.- Cree un programa que use dos hilos, en uno se incrementa (aumento en uno) un millón de veces una variable entera y en otro se decrementa (disminución en uno) la misma variable un millón de veces. Si la variable comienza en cero, imprima el valor de la variable al término de ambos hilos.
No se preocupe por las actualizaciones atómicas de la variable compartida entre ambos hilos. La idea es intentar producir una carrera crítica.

La solución adjunta aprovecha el hilo principal del proceso como un hilo y crear otro hilo para reducir el valor del contador.

```
#include <stdio.h>
#include <stdlib.h> /* for exit */
#include <pthread.h>

#define ITERACIONES 1000000
int contador;

void * increment(void * arg)
{
  int i;
  for (i=0; i<ITERACIONES; i++)
    contador++;
    return(NULL);
}

int main(void)
{
  int err, i;
  contador =  0;
  pthread_t tid;

  err =  pthread_create(&tid, NULL, increment, NULL);
  if (err != 0){
    printf("can't create thread\n");
    exit(0);
  }
  for (i=0; i<ITERACIONES; i++)
    contador--;

  pthread_join(tid, NULL);
  printf("El valor final de contador es %i\n", contador);
}
```

Por completitud y para los que se preguntarán si hay cambio al imponer acceso exclusivo a la variable compartida, se presenta la solución libre de carreras críticas.

```
#include <stdio.h>
#include <stdlib.h> /* for exit */
#include <pthread.h>

#define ITERACIONES 1000000
```

```
int contador;
pthread_mutex_t mylock = PTHREAD_MUTEX_INITIALIZER;

void * increment(void * arg)
{
  int i;
  for (i=0; i<ITERACIONES; i++){
    pthread_mutex_lock(&mylock);
    contador++;
    pthread_mutex_unlock(&mylock);
  }
}

int main(void)
{
  int err, i;
  contador = 0;
  pthread_t tid;

  err = pthread_create(&tid, NULL, increment, NULL);
  if (err != 0){
    printf("can't create thread\n");
    exit(0);
  }
  for (i=0; i<ITERACIONES; i++){
    pthread_mutex_lock(&mylock);
    contador--;
    pthread_mutex_unlock(&mylock);
  }

  pthread_join(tid, NULL);
  printf("El valor final de contador es %i\n", contador);
}
```

4.- Un estudiante pregunta por el tamaño del buffer de datos utilizado por TCP para transmitir y recibir. Para ello sugiere utilizar un servidor TCP que se limita a aceptar la conexión de un cliente y luego espera indefinidamente sin leer datos de la conexión aceptada hasta su término con Control-C. Por otro lado sugiere crear un cliente TCP que envíe repetidamente datos hacia el servidor en grupos de 10 bytes imprimiendo por pantalla cada vez el valor acumulado de los bytes transmitidos. Su idea es que cuando el cliente ya no pueda transmitir más, habrá llenado los buffers de TCP de recepción del servidor y de transmisión del cliente.

Sin cuestionar mayormente la idea del estudiante, haga los programas servidor y receptor señalados.

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

#define PORTNUMBER 12345

int main(void)
{
  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);
  pause();
}

/*****************  Programa Cliente TCP    ********************/
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>   /* for printf */
#include <stdlib.h> /* for exit */
#include <string.h> /* for memcpy */

#define PORTNUMBER  12345

int main(int argc, char * argv[])
{
  int n, s, len, total;
  char buf[10];
  struct hostent *hp;
  struct sockaddr_in name;

  hp = gethostbyname("localhost"); /* localhost or another */

  s = socket(AF_INET, SOCK_STREAM, 0);

  name.sin_family = AF_INET;
  name.sin_port = htons(PORTNUMBER);
  memcpy(&name.sin_addr, hp->h_addr_list[0], hp->h_length);
  len = sizeof(struct sockaddr_in);

  connect(s, (struct sockaddr *) &name, len);
  total=0;
  while (1) {
    printf("Datos enviados %i\n",total);  fflush(stdout);
    if ((n=send(s, buf, sizeof(buf), 0)) < 0) {
      perror("send");
      exit(1);
    }
    total+=n;
  }
}
```