**Primer Certamen: Tiempo: 15:40 hrs - 17:10 hrs.**
**Responder un problema por página**

1.- (30 puntos) Para administrar en su espacio en disco un alumno decide borrar todos los archivos con fechas previas a la de un archivo determinado (no cuestione la decisión). Desarrolle borraOld.sh <dir> <file>, un script shell que borra todos los archivos bajo <dir> que sean más antiguos que <file>.

Ayuda: Para comparar fechas de archivos puede usar el comando **test** con la expresión condicional:
file1 -ot file2 True if file1 is older than file2, or if file2 exists and file1 does not.

```
#! /bin/bash
dir=$1
file=$2

for i in $dir/*
do
  if test –d $i
  then
    $0 $i $file
  else
    if test $i –ot $file
      then
     rm –f $i
    fi
  fi
done
```

2.- (40 puntos) Se desea comparar la rapidez para **crear y terminar procesos** con la rapidez para **crear y terminar hebras**. Para ello se pide desarrollar PH.c, un programa en C que usa como argumento el tiempo en segundos que durará la medición. PH por una parte crea procesos secuencialmente y por otra parte crea hebras secuencialmente; ambas cosas hasta que se acabe el tiempo. Cada proceso creado muestra por pantalla un número correlativo P_p, con p el número de procesos creados hasta ese momento y luego el proceso termina. Cada hebra creada debe mostrar por pantalla un número correlativo H_h, con h el número de hebras creadas hasta ese momento y luego la hebra termina.
Para reducir el tiempo total de la medición y visualizar mejor su desarrollo, se pide que ambas mediciones se hagan concurrentemente y sobre el mismo tiempo indicado en el argumento del programa.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <signal.h>
#include <pthread.h>

int done = 0;
static void  sig_alrm(int signo)
{
    done=1;
```

```
}
void doNothingFork(void) {
  return;
}
void *doNothingThread(void *null) {
  int i=0;
  pthread_exit(NULL);
}

int main(int argc, char * argv[]) {
  int pid, j, status;
  int nSeconds, p=1, h=1;

  int rc, i, detachstate;
  pthread_t tid;

  nSeconds= atoi(argv[1]);

  if (pid = fork()) {  /* Child process */
    if (signal(SIGALRM, sig_alrm) == SIG_ERR)
      exit(-1);
    alarm(nSeconds);
    while (!done) {
      if ((pid = fork()) == 0 ) {
     doNothingFork();
     exit(0);
      } else  /*** this is the parent of the fork ***/
     waitpid(pid, &status, 0);
      printf("P_%i\n",p++);
    }
    exit(0);
  }
  /* Parent process */
  if (signal(SIGALRM, sig_alrm) == SIG_ERR)
    exit(-1);
  alarm(nSeconds);
  while (!done) {
    rc = pthread_create(&tid, NULL, doNothingThread, NULL);
    if (rc) {
      printf("ERROR; return code from pthread_create()%d\n", rc);
      exit(-1);
    }
    rc = pthread_join(tid, NULL);
    if (rc) {
      printf("ERROR; return code from pthread_join() %d\n", rc);
      exit(-1);
    }
    printf("\tH_%i\n", h++);
  }
  waitpid(pid, &status, 0);
  exit(0);
}
```

3.- (30 puntos) Cree timeService <port>, un servidor TCP que envía su hora actual en formato texto tan pronto como recibe la conexión de cada cliente al puerto <port>. Luego del envío del string, timeService cierra la conexión con ese cliente y espera nuevas conexiones.

Ayuda: Para obtener la hora local puede usar:
ctime(time(null)) retorna un string con la hora actual de la forma "Wed Feb 29 15:40:00 2012\n".

```
#include <string.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <time.h>

int main(int argc, char * argv[])
{
    char * date;
    int s, n, ns, len;
    time_t t;
    struct sockaddr_in name;

    /* Create the socket. */
    s = socket(AF_INET, SOCK_STREAM, 0);
    /* Create the address of the server. */
    name.sin_family = AF_INET;
    name.sin_port = htons(atoi(argv[1]));
    name.sin_addr.s_addr = htonl(INADDR_ANY);
    len = sizeof(struct sockaddr_in);
    /* Bind the socket to the address. */
    bind(s, (struct sockaddr *) &name, len);
    /* Listen for connections.  */
    listen(s, 5);
    while (1) {
       /* Accept a connection.  */
       ns = accept(s, (struct sockaddr *) &name, &len);
       t = time(NULL);
       date = ctime(&t);
       send(ns, date, strlen(date), 0);
       close(ns);
    }
    close(s);
    exit(0);
}
```