Certamen Parcial 90 minutos.
Todas las preguntas tienen igual puntaje.

1.- Desarrolle el programa proteja.sh dir permisoDir permisoFile. Éste es un shell o script que permite cambiar los permisos de dir y todos los subdirectorios del subárbol con raíz en dir a permisoDir y de los archivos bajo este subárbol a permisoFile.

Por ejemplo, para proteger todos los archivos de mi cuenta puedo hacer:

```
%cd                          # para quedar en home
%proteja.sh . 700 600        # y luego
%proteja.sh WWW 711  644     # para dar acceso a mi web.
```

```
#! /bin/bash
#scan directory recursively and change the diretory's permission
# under $1 to $2 and all the file's to $3
dir=$1
perD=$2
perF=$3
echo .... scanning directory $dir
chmod $perD $dir

for i in $dir/*
  do
  if test -d $i
  then
     $0 $i $perD $perF
  else
     echo ".... changing file $i"
     chmod $perF $i
  fi
done
```

Permisos Directorios: 10 pts (incluido el raíz)
Permisos archivos: 10 pts.
Trabajo en sub-árboles: 10 pts.
Estructuras de control: 3 pts.

2.- Desarrolle el programa alarma.c. Este programa se corre como:
%alarma minutos
Donde minutos es un entero no negativo.
Al correr el programa retorna al prompt del sistema inmediatamente, pero cuando ha pasado el número de minutos indicados, en la pantalla aparece por 5 segundos el reloj del comando xclock.

Por retornar pront: 10 pts.
Por esperar el número de minutos : 8 pts.
Por ejecutar xclock: 7 pts.
Por esperar 5 seg. y terminar xclock: 8 pts.

```
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(int argc, char * argv[]) {
  pid_t pid, pid2generation;

  if ( (pid = fork()) < 0)     perror("fork error in parent");
  else if (pid != 0) /* parent process*/
    exit(0);   /* this way the prompt comes back */
            /* the child process is adopted by init */

  /*child process*/
  sleep(60*atoi(argv[1]));   /* wait for N second */
  if ( (pid2generation=fork()) <0 )   /*prepare to lunch xclock */
    perror("fork error in child");
  else if (pid2generation==0)  /* grandchild process*/
    if (execlp("xclock", "xclock",(char *) 0) < 0)
      perror("execlp error");

  /* this is the child again */
  sleep(5);                    /* wait for 5 second */
  kill(pid2generation, SIGKILL);  /* close xclock */
  if (waitpid(pid2generation, NULL, 0) < 0)
    perror("wait error");
}
```

3.- Todo indica que el corta fuegos entre la red local del laboratorio de Electrónica y la red de los profesores no permite el paso del tráfico multicast. Para superar esta situación y sabiendo que Aragorn tiene una tarjeta de red con IP 172.16.0.4 y otra con IP 200.1.17.195, a usted se le pide hacer un programa que corriendo en aragorn tome el tráfico del grupo multicast 234.5.5.5, puerto 1234, de un lado de la red y lo envíe al otro y viceversa.

Por creación de sockets: 5 pts.
Por bind de sockets: 6 pts.
Por membresía de ambos sockets: 6 pts.
Por manejo de vector de descriptores 5 pts.

Por manejo de select: 5 pts.
Por re-envío: 6 pts.

```
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
#define LENGTH 1000
int main( int argc, char * argv[] ) {
  int   s_private, s_valid;
  struct sockaddr_in mc_group;
  int length;
  char buffer[LENGTH];
  int rc, n;
  struct ip_mreq mreq;  /* multicast group info structure */
  char ttl;
  fd_set readfds, readfdsCopy;

  s_private = socket (AF_INET,SOCK_DGRAM,0);
  s_valid = socket (AF_INET,SOCK_DGRAM,0);
  mc_group.sin_family = AF_INET;
  mc_group.sin_addr.s_addr = inet_addr("172.16.0.4");
  mc_group.sin_port = htons(1234);
  bind( s_private, (struct sockaddr *)&mc_group, sizeof(mc_group));
  mc_group.sin_addr.s_addr = inet_addr("200.1.17.195");
  bind( s_valid, (struct sockaddr *)&mc_group, sizeof(mc_group));

  mreq.imr_multiaddr.s_addr = inet_addr("234.5.5.5");
  mreq.imr_interface.s_addr = inet_addr("172.16.0.4");
  setsockopt(s_private,IPPROTO_IP,IP_ADD_MEMBERSHIP,(char *) &mreq,
          sizeof(mreq));
  mreq.imr_interface.s_addr = inet_addr("200.1.17.195");
  setsockopt(s_valid,IPPROTO_IP,IP_ADD_MEMBERSHIP,(char *) &mreq,
          sizeof(mreq));
  ttl = 2;
  setsockopt(s_private ,IPPROTO_IP,IP_MULTICAST_TTL,(char *) &ttl,
          sizeof(u_char));
  setsockopt(s_valid ,IPPROTO_IP,IP_MULTICAST_TTL,(char *) &ttl,
          sizeof(u_char));
  mc_group.sin_addr.s_addr= inet_addr("234.5.5.5");

  FD_ZERO(&readfdsCopy);
  FD_SET(s_private,&readfdsCopy);
  FD_SET(s_valid,&readfdsCopy);
  for(;;){
    memcpy(&readfds, &readfdsCopy, sizeof(fd_set));
```

```
    n = select(FD_SETSIZE, &readfds, (fd_set *) 0, (fd_set *) 0, NULL);
    if (n > 0) {
      if (FD_ISSET(s_private, &readfds)) {
        rc=recv(s_private, (char *)&buffer, sizeof(buffer), 0);
        sendto(s_valid, (char *)&buffer, rc, 0, (struct sockaddr*)&mc_group,
              sizeof(mc_group));
      }
      if (FD_ISSET(s_valid, &readfds)) {
        rc=recv(s_valid, (char *)&buffer, sizeof(buffer), 0);
        sendto(s_private, (char *)&buffer, rc, 0, (struct sockaddr*)&mc_group,
              sizeof(mc_group));
      }
    }
  }
}
```