Certamen Parcial 100 minutos.

Todas las preguntas tienen igual puntaje.

1.- Un profesor de enseñanza básica le pide a usted que haga algún (nos) programa (s) shell que le permita revisar las pruebas de su curso. La prueba (prueba.txt) contiene un problema por línea del tipo 45*(10-8). Cada respuesta de los alumnos fue entregada en un archivo (nombre_alumno.txt) donde hay una respuesta por línea. A usted se le pide un programa shell (nota.sh) del tipo:

% nota.sh prueba.txt nombre_alumno.txt

El cual retorna la nota obtenida por el alumno como el cuociente (Nro_preguntas_correctas/total_preguntas).

Ejemplo: Si la prueba fuera de dos preguntas, podríamos tener:

prueba.txt	Eduardo_Gonzalez.txt (uno por alumno)
45*(10-8) 4*6+20/2	90 34

Considere el uso del programa bc, el cual termina al ingresar quit, ejemplo de su uso es:

```
agustin@agustin2005 $ bc -q
45*(10-8)
90
quit
agustin@agustin2005 $
```

Una primera solución:

```
#! /bin/bash
prueba=$1
set `cat $2`                    # en valor $n tenemos respuesta a pregunta n
i=0                             # contador para el numero de preguntas
correctas=0                     # contador para el numero de respuestas correctas
bc < $prueba > /tmp/pauta$$             # generamos la pauta de correccion
while read resultado            # leemos una a una las respuesta de la pauta
  do
  if test $resultado -eq $1     # corresponde el resultado a la respuesta?
     then
     correctas=`expr $correctas + 1`
  fi
  shift                         # pase a la siguiente respuesta del alumno
  i=`expr $i + 1`
done < /tmp/pauta$$
echo $correctas "* 100 /" $i | bc
# pudo usarse:
expr $correctas \* 100 / $i     # el \ es necesario para el * no se
                                # interprete como "remplace por cualquier cosa"
rm /tmp/pauta$$
```

Otra solución:

```
#! /bin/bash
```

```
prueba=$1
alumno=$2
bc < $prueba > /tmp/pauta$$                    # respuesta correctas
diff -dB $alumno /tmp/pauta$$ >> /tmp/diff$$   # diferencia entre archivos
set `wc -l /tmp/diff$$`
num_erradas=`expr $1 - 2`                      # descontar linea inicial y final
num_erradas=`expr $num_erradas/2`              # y dividir porque se lista la
                                               # linea original y la cambiada
set `wc -l $prueba`                            # numero de preguntas
total_preguntas=$1
echo "100*($total_preguntas-$num_erradas)/ $total_preguntas" | bc
```

2.- El comando tee lee desde la entrada estándar y copia el contenido hacia la salida estándar y además un archivo. De esta forma podemos, por ejemplo, hacer un registro de la ejecución de un programa. Si ejecutamos:
% ping 200.1.17.1 | tee ping.txt
Podremos ver por pantalla el resultado del ping y además estaremos dejando una copia de este resultado en ping.txt.
Haga usted su propia implementación del comando tee, la cual llamaremos mytee. En su solución no puede hacer uso del comando tee.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int main(int argvc, char * argv[])
{
   int f;
   int n;
   char buff[256];

   f = open(argv[1], O_WRONLY | O_CREAT, 0666);
   do {
     n = read(0, buff, sizeof(buff));
     write(f, buff, n);
     write(1, buff, n);
   } while( n > 0);
   close(f);
}
```

Otra solución en bash:

```
#! /bin/bash
while read line
do
  echo $line
```

```
  echo $line >> $1
done
```

3.- Considere un servicio de eco donde el programa ***eco*** acepta conexiones en un puerto y envía de vuelta todo el tráfico que los clientes le hagan llegar.
a) ¿En qué programa -cliente o servidor- esperaría el llamado (one=1):
`setsockopt(s,SOL_SOCKET,SO_REUSEADDR,(char *) &one,sizeof(one)).`
b) ¿Qué requerimiento satisface este llamado?
c) ¿Qué pasa si omitimos tal llamado? (explique la secuencia que se debe seguir para observar algún cambio si es que lo hay).

a) En el servidor.
b) Es éste el que debe permitir el bind a un puerto que está ya en uso, como el caso de multicast, o el puerto ha sido usado recientemente por una conexión TCP y se encuentra en estado de tiempo de espera (TIME_WAIT).
c) Si se omite este llamado, al ejecutar un programa servidor, luego establecer una conexión desde un cliente y matar el cliente en forma abrupta (vía un kill –9 o control-C por ejemplo) el puerto del servidor quedará tomado y al intentar ejecutarlo nuevamente recibiremos un mensaje de error indicando que el puerto ya está en uso. Deberemos esperar poco más de un minuto para que esta situación pase y podamos volver a ejecutar el servidor con el mismo puerto.

4.- Se desea crear un servicio de chat simple. A usted le encargan la tarea de hacer el servidor del chat. chatServer <puerto> acepta conexiones en el puerto indicado. Todo lo recibido a través de una conexión es copiado en cada una de las otras, incluyendo aquella por donde llegó el mensaje.

```
void reusePort(int sock);
int EchoServe(int sockList[], int lastSock, int psd);

int main( int argc, char *argv[])
{
  int   s, ss;
  struct   sockaddr_in server;
  struct sockaddr_in from;
  int fromlen;

  fd_set readfds, readfdsCopy;
  int n,i;
  int sockList[64], lastSock=0;

    /* Construct name of socket to send to. */
  server.sin_family = AF_INET;
  server.sin_addr.s_addr = htonl(INADDR_ANY);
  server.sin_port = htons(atoi(argv[1]));

  s = socket (AF_INET,SOCK_STREAM,0);
  reusePort(s);

  if ( bind( s, (struct sockaddr *)&server, sizeof(server) ) ) {
```

```
    close(s);
    perror("binding name to stream socket");
    exit(-1);
  }

  listen(s,4);
  fromlen = sizeof(from);
  FD_ZERO(&readfdsCopy);
  FD_SET(s,&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, &readfds)) {
        printf("Accepting a new connection...\n");
        sockList[lastSock++] = accept(s, (struct sockaddr *)&from,
                                      &fromlen);
        FD_SET(sockList[lastSock-1], &readfdsCopy);
      }
      for (i=0; i < lastSock; i++) {
        if (FD_ISSET(sockList[i], &readfds))
          if (EchoServe(sockList, lastSock, i) < 0) {
            FD_CLR(sockList[i], &readfdsCopy);
            close (sockList[i]);
            sockList[i] = sockList[--lastSock];
            i--;
          }
      }
    }
  }
}

int EchoServe(int sockList[], int lastSock, int psd)
{
  char buf[512];
  int rc, i;

  for(;;){
    if( (rc=read(psd, buf, sizeof(buf))) < 0)
      perror("receiving stream  message");
    if (rc > 0){
      buf[rc]='\0';
      for (i=0; i<lastSock; i++)
        if (send(sockList[i], buf, rc, 0) <0 )
          perror("sending stream message");
      return(1);
    }
    else {
      printf("Disconnected..\n");
      return(-1);
    }
  }
}
void reusePort(int s)
{
  int one=1;
```

```
  if ( setsockopt(s,SOL_SOCKET,SO_REUSEADDR,(char *)
         &one,sizeof(one)) == -1 )
  {
    printf("error in setsockopt,SO_REUSEPORT \n");
    exit(-1);
  }
}
```