Certamen Parcial

Todas las preguntas tienen igual puntaje.

1.- Haga un programa shell, supongamos nombre.sh, que tome como argumento el nombre de un usuario (su login) y muestre por pantalla el nombre completo del usuario según tabla passwd.org_dir. El formato típico de una línea de la salida generada por niscat para esta tabla es:

agv:*NP*:205:205:Agustín González V.:/home/alm2000/agv:/usr/local/bin/bash:*NP*

De modo que si invocamos el programa se tiene:
$ nombre.sh agv
Agustín González V.

```
#! /bin/bash
IFS=":"
target=$1

niscat passwd.org_dir | while :
do
  if read login junk1 junk2 junk3 nombre junk4
      then
      if test $target ==
$login
       then
       echo $nombre
       exit
      fi
  fi
done
```

2.-

a. ¿Qué diferencia existe entre exit() y _exit()?

Ambas funciones terminan el programa; sin embargo exit() hace una operación de limpieza antes de terminar invocando a las funciones que se hayan registrado con atexit(). _exit() por su parte sólo concluye el programa.

b. ¿Para qué sirve la función FILE * fdopen(int , const char *);?

Sirve para asociar un string a un descriptor de archivo. De esta forma podemos usar funciones con salida formateada como printf en lugar de write.

c. Explique una ventaja de la comunicación entre procesos a través de socket por sobre aquella con memoria compartida.

La comunicación con socket nos permite intercambiar información entre procesos ubicados tanto en la misma máquina como en máquinas remotas conectadas vía TCP/IP. La memoria compartida requiere que ambos procesos estén el la misma máquina.

d. Explique una ventaja de la comunicación entre procesos usando memoria compartida por sobre aquella vía socket.

La memoria compartida permite comunicación asincrónica, en el sentido que ambos procesos no requieren estar corriendo en forma simultánea.
La memoria compartida permite mayor throughput debido a que los datos generados por un proceso pueden ser tomados por el otro sin movimientos en memoria.

e. Explique una ventaja del servidor TCP iterativo implementado con select por sobre el servidor TCP concurrente implementado con fork.

El servidor TCP implementado con select permite intercambio de información entre los distintos clientes TCP debido a que se comparte el espacio de direcciones. Aquel hecho con fork requiere incluir otros mecanismos de comunicación entre procesos lo cual hace la implementación de este requisito algo más complejo.

3.- Se desea incorporar un reloj a una aplicación. Se dispone del comando xclock el cual muestra una ventana con la hora como se muestra en la imagen. Haga los cambios requeridos al programa mostrado para que muestre el reloj durante su ejecución.

```
#include <stdio.h>
int main(void)
{
      printf("Para su comodidad se muestra un reloj. \n");

  ...  /* resto del programa */
  ...

/* finalización del programa */
      exit(0);
}
```

```
#include <stdio.h>
#include <sys/types.h>
#include <signal.h>

int main(void)
{
  int pid;

      printf("Para su comodidad se muestra un reloj \n");
```

```
  pid=fork();
  if (pid==0)
    execlp("xclock","xclock", NULL);

/* resto del programa*/
/*
  ...
  ...
*/
/* finalización del programa */
  kill(pid,SIGKILL);
  while(pid!=wait(NULL));

      exit(0);
}
```

4.- Extienda el servidor UDP de eco (aquel que retorna todo lo que recibe) visto en clases de modo que cada vez que se presione enter en su teclado, este servidor muestre en pantalla el número de mensajes recibidos.

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>

int main(int argc, char * argv[])
{
  char buf[50];
  int s, n, len;
  struct sockaddr_in name;
  fd_set readfds, readfdsCopy;
  int numeroMensajes=0;

  s = socket(AF_INET, SOCK_DGRAM, 0);

  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(s, (struct sockaddr *) &name, len);

  FD_ZERO(&readfdsCopy);
  FD_SET(s,&readfdsCopy);
  FD_SET(0,&readfdsCopy);

  while (1) {
    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)){
        n = recvfrom(s, buf, sizeof(buf), 0,(struct sockaddr
*)&name, &len);
        if (n<= 0) break;
        sendto(s, buf, n, 0, (struct sockaddr *)&name, len);
        numeroMensajes++;
      }
      if (FD_ISSET(0, &readfds)){
        getchar();
        printf("Número de mensajes =%i\n", numeroMensajes);
      }
    }
  }
  close(s);
  exit(0);
}
```