SERVIDOR Y CLIENTE DE CHAT MINIMO

servidor-chat.c

```
/*
 *      Servidor de chat mínimo
 *      JAMJ - Enero 2006
 *
 */

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

#define USUARIOS        10                      /* Número de usuarios concurrentes */
#define PORT            3550                    /* El puerto que será abierto */
#define BACKLOG         2                       /* El número de conexiones permitidas */
#define TAM_BUFFER      2000


int fd1;                                        /* Descriptor del servidor */
struct sockaddr_in servidor, cliente[USUARIOS]; /* Información del servidor y clientes */
int size;                                       /* Para tamaño de sockaddr_in */

int fd[USUARIOS];                               /* Descriptores de los clientes */
pthread_t h[USUARIOS];
char buffer[USUARIOS][TAM_BUFFER];
int i;

void *usuario (int id);



main()
{


   /* Información del servidor en sockaddr_in */
   servidor.sin_family = AF_INET;
   servidor.sin_port = htons(PORT);             /* Es necesario utilizar htons */
   servidor.sin_addr.s_addr = inet_addr(INADDR_ANY); /* INADDR_ANY coloca nuestra dirección IP
automáticamente */
   bzero(&(servidor.sin_zero),8);               /* Escribimos ceros en el resto de la estructura */



   /* socket() */
   if ((fd1=socket(AF_INET, SOCK_STREAM, 0)) == -1 ) {
      printf("error en socket()\n");
      exit(-1);
   }

   /* bind() */
   if(bind(fd1,(struct sockaddr *)&servidor, sizeof(struct sockaddr))==-1) {
      printf("error en bind() \n");
      exit(-1);
   }

   /* listen() */
   if(listen(fd1,BACKLOG) == -1) {
      printf("error en listen()\n");
      exit(-1);
   }
```

```
   /* Inicializamos descriptores */
   for (i=0; i<USUARIOS; i++ ) fd[i]=0;

   //size=sizeof(struct sockaddr_in);
   while(1) {      /* Aceptamos conexión */
        /* Encontramos descriptor libre */
        for (i=0; i<USUARIOS; i++)
                if (fd[i]==0) break;
        /* Atendemos usuario */
        if (i!=USUARIOS) {
                if ((fd[i] = accept(fd1,(struct sockaddr *)&cliente[i], &size))==-1) {
                        printf("error en accept()\n");
                        exit(-1);
                }
                pthread_create (&h[i], NULL, usuario, i);
        }

   }
   close (fd1);
}



void *usuario (int id){
        while (recv (fd[id], buffer[id], TAM_BUFFER, 0) != -1)
                for (i=0; i<USUARIOS; i++)
                        if (i!=id && fd[i]) send (fd[i], buffer[id], TAM_BUFFER, 0);
        fd[id]=0;
        pthread_exit (NULL);
}
```

cliente-chat.c

```
/*
 *      Cliente de chat mínimo
 *      JAMJ - Enero 2006
 *
 */

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>                      /* netbd.h es necesitada por la estructura hostent ;-) */

#include <signal.h>

#define PORT            3550            /* El Puerto Abierto del nodo remoto */
#define TAM_BUFFER      2000            /* El número máximo de datos en bytes */

pthread_t env, rec;
char buf_teclado[TAM_BUFFER];           /* Donde almacenamos el texto a enviar  */
char buf_red[TAM_BUFFER];               /* Donde almacenamos el texto a mostrar */
int fd;

void *enviar ();
void *recibir ();




int main(int argc, char *argv[])
{
   int numbytes;
   struct sockaddr_in servidor;     /* Información sobre la dirección del servidor */
   struct hostent *he;              /* Estructura que recibirá información sobre el nodo remoto */
```

```
   if (argc !=2) {                 /* esto es porque nuestro programa necesitará la IP */
      printf("Uso: %s <Dirección IP>\n",argv[0]);
      exit(-1);
   }

   if ((he=gethostbyname(argv[1]))==NULL){      /* llamada a gethostbyname() */
      printf("gethostbyname() error\n");
      exit(-1);
   }

   /* Información del servidor en sockaddr_in */
   servidor.sin_family = AF_INET;
   servidor.sin_port = htons(PORT);             /* Es necesario utilizar htons */
   servidor.sin_addr = *((struct in_addr *)he->h_addr); /*he->h_addr pasa la información de ``*he''
a "h_addr" */
   bzero(&(servidor.sin_zero),8);               /* Escribimos ceros en el resto de la estructura */


   /* socket() */
   if ((fd=socket(AF_INET, SOCK_STREAM, 0))==-1){
      printf("socket() error\n");
      exit(-1);
   }

   /* connect() */
   if(connect(fd, (struct sockaddr *)&servidor, sizeof(struct sockaddr))==-1){
      printf("connect() error\n");
      exit(-1);
   }

   pthread_create (&rec, NULL, recibir, NULL);
   pthread_create (&env, NULL, enviar, NULL);
   pthread_join (rec, NULL);
   pthread_join (env, NULL);
   close(fd);   /* Cerramos fd */

}


void *enviar (){
        while (fgets (buf_teclado, 79, stdin) != NULL)
                send(fd,buf_teclado,TAM_BUFFER,0);
        pthread_exit (NULL);
}

void *recibir (){
        while (recv(fd,buf_red,TAM_BUFFER,0) != -1)
                fputs (buf_red, stdout);
        pthread_exit (NULL);
}
```

COLOR EN TERMINALES DE TEXTO

colores.c

```
/*

  SECUENCIA DE COLORES ANSI

        Negro       0;30     Gris oscuro    1;30
        Azul        0;34     Azul claro     1;34
        Verde       0;32     Verde claro    1;32
        Cyan        0;36     Cyan claro     1;36
        Rojo        0;31     Rojo claro     1;31
        Purpura     0;35     Purpura claro  1;35
        Marron      0;33     Amarillo       1;33
        Gris claro  0;37     Blanco         1;37


También se pueden poner colores de fondo, usando 44 para fondo azul, 41 para fondo rojo, etc. No
hay colores de fondo 'negrita'; se pueden usar combinaciones, como texto rojo claro sobre fondo
azul \033[44;1;31m, aunque parece que funciona mejor poner los colores separadamente (es decir,
\033[44m\033[1;31m). Otros códigos disponibles incluyen 4: subrayado, 5: parpadeante, 7: inverso y
8: oculto.

****************

Las secuencias de escape ANSI permiten mover el cursor por la pantalla a voluntad. Esto es más útil
para interfaces de usuario a pantalla completa generados por shell scripts, pero también se pueden
usar en prompts. Las secuencias de escape de movimientos son las siguientes:


        - Posicionar el cursor:
          \033[<L>;<C>H
          pone el cursor en la linea L, columna C.
        - Mover el cursor arriba N lineas:
          \033[<N>A
        - Mover el cursor abajo N lineas:
          \033[<N>B
        - Mover el cursor hacia adelante N columnas:
          \033[<N>C
        - Mover el cursor hacia atras N columnas:
          \033[<N>D
        - Guardar la posicion del cursor:
          \033[s
        - Restaurar la posicion del cursor:
          \033[u

*/
#include <stdio.h>

int i,j,k ;
char letra;

int main () {

        printf ("\033[%dm\033[%d;%dm COLORES EN CONSOLA (Pulsa una tecla) ", 44,0,31);
        scanf ("%c", &letra);

        for (i=40; i<48; i++) {
                printf ("\033[%d;%dH", i-30, 0);                // Mueve el cursor a linea y columna
                for (j=0; j<2; j++)
                        for (k=30; k<38; k++){
                                printf ("\033[%dm", i);         // Color de fondo
                                printf ("\033[%d;%dm HOLA", j,k);
                        }
        }
        printf ("\033[0m");                                     // Restaura los colores originales
        printf ("\033[%d;%dH", i-30, 0);                        // Seguimos en la siguiente línea
        return 0;
}
```

colores.sh

```
#!/bin/bash
#
# Este fichero saca por pantalla un monton de codigos de color
# para demostrar que hay disponible. Cada linea es un color con
# fondo negro y gris, con el codigo en medio. Funciona sobre
# fondos blancos, negros y verdes (2 dic. 98)
#
echo "  Sobre gris claro:         Sobre negro:"
echo -e "\033[47m\033[1;37m  Blanco        \033[0m\
 1;37m \
\033[40m\033[1;37m  Blanco        \033[0m"
echo -e "\033[47m\033[37m  Gris Claro    \033[0m\
   37m \
\033[40m\033[37m  Gris Claro    \033[0m"
echo -e "\033[47m\033[1;30m  Gris          \033[0m\
 1;30m \
\033[40m\033[1;30m  Gris          \033[0m"
echo -e "\033[47m\033[30m  Negro         \033[0m\
   30m \
\033[40m\033[30m  Negro         \033[0m"
echo -e "\033[47m\033[31m  Rojo          \033[0m\
   31m \
\033[40m\033[31m  Rojo          \033[0m"
echo -e "\033[47m\033[1;31m  Rojo Claro    \033[0m\
 1;31m \
\033[40m\033[1;31m  Rojo Claro    \033[0m"
echo -e "\033[47m\033[32m  Verde         \033[0m\
   32m \
\033[40m\033[32m  Verde         \033[0m"
echo -e "\033[47m\033[1;32m  Verde Claro   \033[0m\
 1;32m \
\033[40m\033[1;32m  Verde Claro   \033[0m"
echo -e "\033[47m\033[33m  Marrón        \033[0m\
   33m \
\033[40m\033[33m  Marron        \033[0m"
echo -e "\033[47m\033[1;33m  Amarillo      \033[0m\
 1;33m \
\033[40m\033[1;33m  Amarillo      \033[0m"
echo -e "\033[47m\033[34m  Azul          \033[0m\
   34m \
\033[40m\033[34m  Azul          \033[0m"
echo -e "\033[47m\033[1;34m  Azul Claro    \033[0m\
 1;34m \
\033[40m\033[1;34m  Azul Claro    \033[0m"
echo -e "\033[47m\033[35m  Púrpura       \033[0m\
   35m \
\033[40m\033[35m  Purpura       \033[0m"
echo -e "\033[47m\033[1;35m  Rosa          \033[0m\
 1;35m \
\033[40m\033[1;35m  Rosa          \033[0m"
echo -e "\033[47m\033[36m  Cyan          \033[0m\
   36m \
\033[40m\033[36m  Cyan          \033[0m"
echo -e "\033[47m\033[1;36m  Cyan Claro    \033[0m\
 1;36m \
\033[40m\033[1;36m  Cyan Claro    \033[0m"
```

dialogo1.sh

```
#!/bin/sh
#
# Diálogo sí/no

dialog \
--backtitle "Diálogo sí/no" --title "Alerta" --colors --no-shadow \
--yesno  "\n\Z2\Zu\ZrProseguir\n" 7 60


sel=$?
case $sel in
    0) echo "Haz elegido proseguir";;
    1) echo "Haz elegido no proseguir";;
    255) echo "Cancelado por el usuario: [ESC]";;
esac
```

dialogo2.sh

```
#!/bin/sh
#
# Diálogo de menú


dialog \
--backtitle "Diálogo de menú" --title "Menú principal" \
--menu "Pulsa [UP] [DOWN],[Enter] para seleccionar" 15 50 3 \
Fecha/Hora "Mostrar Fecha y Hora" \
Calendario "Ver calendario " \
Editor "Iniciar editor vi " \
2>/tmp/menuitem.$$


menuitem=`cat /tmp/menuitem.$$`
opt=$?

case $menuitem in
        Fecha/Hora) date;;
        Calendario) cal;;
        Editor) vi;;
esac
```