Práctica 4

HTTP

```
# include <sys/types.h>
# include <sys/socket.h>
# include <sys/wait.h>
# include <netinet/in.h>
# include <netdb.h>
# include <errno.h>
# include <stdio.h>
# include <unistd.h>
# include <fcntl.h>
# include <signal.h>
# include <string.h>
# include <stdlib.h>
# define FICHERO  "mserver.log"
# define HEAD     "head.html"
# define TAIL     "tail.html"

static void usage(char *name)
{
   fprintf(stderr, "%s: Uso %s <port>\n", name, name);
   exit (1);
}

static void ChildHasDied(int num)
{
int   status;
pid_t pid;

   pid = wait(&status);
   if (signal(SIGCHLD, ChildHasDied) == SIG_ERR)
      fprintf(stderr, "Error al dar de alta la accion.\n");
   return;
}

static void FinishAction(int num)
{
   fprintf(stderr, "Programa finalizado correctamente.\n\n");
   exit(0);
}

static void net_server(int sock) //una vez establecida la conexion socket con el usuario
{
# define BUFF_LEN 8192
int   len;
char  formato[BUFF_LEN];
char  cadena[BUFF_LEN];
FILE *fp = NULL;

   fp = fopen(FICHERO, "w"); //abro el fichero que contiene la página WEB
   len = fcntl(sock, F_SETFD, O_NDELAY | O_NONBLOCK); //manipula el descriptor de un idchero

   do //leo el fichero en ¡¡¡ESCRITURA¡¡¡ y lo voy mostrando por consola
   {
      len = read(sock, cadena, BUFF_LEN);
      sprintf(formato, "%%%d.%ds", len, len);
      if (fp)
        {
         fprintf(fp, formato, cadena);
         printf(formato, cadena);
        }
      else
         printf(formato, cadena);
```

```
     }while (len == BUFF_LEN);
   fclose(fp);//cierro el fichero

   fp = fopen(HEAD, "r");
   if (fp)
     {
      while(fgets(cadena, sizeof(cadena), fp) != NULL)
         write(sock, cadena, strlen(cadena));

      }
   fclose(fp);


   fp = fopen(FICHERO, "r"); //abro el fichero para ller
   if (fp)
     {
      while(fgets(cadena, sizeof(cadena), fp) != NULL){ //mientras al leer una linea no sea NULL
          if (strstr(cadena, "GET") != NULL) {
            printf("%s",cadena);
          }
          if (strstr(cadena, "User-Agent") != NULL) {
            printf("%s",cadena);
          }
          write(sock, cadena, strlen(cadena)); //envio por el socket la cadena leiada
         }
     }
   fclose(fp);//Cierro el fichero con la página WEB

   fp = fopen(TAIL, "r");
   if (fp)
     {
      while(fgets(cadena, sizeof(cadena), fp) != NULL)
         write(sock, cadena, strlen(cadena));
     }
   fclose(fp);

   close(sock);
}

static int net_slisten(short port)
{
int sock, new_sock;
struct sockaddr_in sin, from;
unsigned int len = sizeof(from);
int OptVal = 1;
int OptSize = sizeof(OptVal);

   bzero((char *) &sin, sizeof (sin));
   sin.sin_family = AF_INET;
   sin.sin_port = htons(port);

   sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP); //(aceptar direcciones IPV4, Conexion basada en el protocolo TCP, )
   if (sock < 0)
      exit(-1);

   if (setsockopt(sock, SOL_SOCKET, SO_REUSEADDR,
      (char *)&OptVal, OptSize) < 0) exit(-3);

   if (bind(sock, (struct sockaddr *)&sin, sizeof(sin)) < 0) exit(-2); //bind para sasociar la direccioón del socket(ip) con el socket

   listen (sock, 4); //espera a la escucha infinita
   for (;;)
   {
      len = sizeof(from);
      new_sock = accept(sock, (struct sockaddr *)&from, &len); //acepto una nueva conexion socket
      if (new_sock < 0)
      {
         if (errno == EINTR)
            continue;
         exit(-4);
      }

      switch (fork()) //creo un hilo para que el padre siga a la escucha de nuevos usuasrios
      {
         case -1:           /* ¿error? */
            exit(-5);

         case 0:            /* proceso hijo */
```

```
            close(sock);
            return(new_sock);

         default:           /* proceso padre */
            close(new_sock);
      }
   }
}

int main(int argc, char **argv)
{
int sock;
short port;

   if (signal(SIGTERM, FinishAction) == SIG_ERR)
      fprintf(stderr, "Error al dar de alta la accion.\n");

   if (signal(SIGCHLD, ChildHasDied) == SIG_ERR)
      fprintf(stderr, "Error al dar de alta la accion.\n");

   if (argc != 2) usage(argv[0]);
   port = atoi(argv[1]);
   sock = net_slisten(port);

   if (sock == -1)
   {
     perror("mserver_http: net_slisten falló.");
     exit(1);
   }
   net_server(sock);
   return 0;
}
```

Para consultar si un puerto en concreto está libre o no:

```
netstat -an | grep 2500
```