HMY 360

Εργαστηριακή Άσκηση 4.1
Εισαγωγή στον προγραμματισμό των Sockets

Σκοπός της άσκησης αυτής είναι να σας δώσει μια πρώτη επαφή με τον προγραμματισμό των sockets του unix/linux. Για την άσκηση αυτή θα πρέπει να υλοποιήσετε τις δύο διεργασίες server και client που ακολουθούν. Αφού τις υλοποιήσετε εξηγήστε με συντομία τι κάνει η κάθε μια. Επίσης εξηγήστε με συντομία τι κάνουν οι πιο κάτω κλήσεις καθώς και τα ορίσματα που παίρνει η κάθε μια.

- `socket()`
- `bind()`
- `listen()`
- `accept()`
- `connect()`
- `read()`
- `write()`
- `close()`

Server

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

#include <unistd.h>
#include <errno.h>
#include <signal.h>
#include <stdio.h>

static void serve(int);

#define PORT 6060

main( int argc, char **argv )
{
     int lsd;            /* Listening socket */
     struct sockaddr_in sin;       /* Binding struct */
     int sin_size=sizeof(sin);
     int sd;                 /* Socket to accept new connexion */

     /* Create listening socket */

     if ( (lsd=socket(AF_INET, SOCK_STREAM, 0)) < 0 ) {
          fprintf(stderr, "%s: cannot create listening socket: ",
argv[0]);
          perror(0);
          exit(1);
     }

     sin.sin_family     = AF_INET;
     sin.sin_port       = htons(PORT);
     sin.sin_addr.s_addr = htonl(INADDR_ANY);

     if ( bind(lsd, &sin, sin_size) < 0 ) {
          fprintf(stderr, "%s: cannot bind listening socket: ",
argv[0]);
          perror(0);
          exit(1);
     }

     /* Initiate a listen queue */

     if ( listen(lsd, 5) < 0 ) {
          fprintf(stderr, "%s: cannot listen on socket: ", argv[0]);
          perror(0);
          exit(1);
     }
```

```
     /* Take care of the SIGPIPE signal - ignore it */
     signal(SIGPIPE, SIG_IGN);

     while ( 1 ) {
          if ( (sd=accept(lsd, &sin, &sin_size)) < 0 )
               exit(errno);
          serve(sd);
          shutdown(sd, 2);
          close(sd);
     }
}


void serve(int sd)
{
     time_t local_time;
     char *time_string;

     time(&local_time);
     time_string = ctime(&local_time);

     write(sd, time_string, strlen(time_string));

     return;
}
```

Client

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

#define BUFSIZE         1024
#define SERVER_PORT 6060

main( int argc, char **argv )
{
     int sd;                  /* Socket descriptor */
     struct sockaddr_in server;    /* Server to connect */
     struct hostent *server_host;  /* Host info */
     char buf[BUFSIZE];
     int nbytes;

     /* Create socket */
     if ( (sd=socket(AF_INET, SOCK_STREAM, 0)) < 0 ) {
          fprintf(stderr, "%s: cannot create socket: ", argv[0]);
          perror(0);
          exit(1);
     }

     /* Get info on host */
     if ( (server_host=gethostbyname(argv[1])) == NULL ) {
          fprintf(stderr, "%s: unknown host %s\n", argv[0], argv[1]);
          exit(1);
     }

     /* Set up struct sockaddr_in */
     server.sin_family = AF_INET;
     server.sin_port   = SERVER_PORT;
     bcopy((char*)server_host->h_addr, (char*)&server.sin_addr,
server_host->h_length);

     /* Connect */
     if ( connect(sd, &server, sizeof(server)) < 0 ) {
          fprintf(stderr, "%s: cannot connect to server: ", argv[0]);
          perror(0);
          exit(1);
     }

     /* Get date */

     if ( (nbytes=read(sd, buf, BUFSIZE-1)) <= 0 ) {
          fprintf(stderr, "%s: read failed: ", argv[0]);
          perror(0);
```

```
            exit(1);
        }

        buf[nbytes] = 0;
        printf("Date on host %s is: %s\n", argv[1], buf);

        close(sd);
        exit(0);
}
```