Name : Muhammad Ishraf Shafiq Zainuddin

ID : 200342741

Lab Assgn : 7

Server

References: http://www.linuxhowtos.org/C_C++/socket.htm

: https://indradhanush.github.io/blog/writing-a-unix-shell-part-2/

: https://stackoverflow.com/questions/13216554/what-does-wait-do-on-unix

```
/* A simple server in the internet domain using TCP
   The port number is passed as an argument
   This version runs forever, forking off a separate
   process for each connection
*/
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/wait.h>

void dostuff(int); /* function prototype */
void error(const char *msg)
{
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[])
{
     int sockfd, newsockfd, portno, pid;
     socklen_t clilen;
```

```
struct sockaddr_in serv_addr, cli_addr;

if (argc < 2)
{
    fprintf(stderr,"ERROR, no port provided\n");
    exit(1);
}
sockfd = socket(AF_INET, SOCK_STREAM, 0);

if (sockfd < 0)
   error("ERROR opening socket");

bzero((char *) &serv_addr, sizeof(serv_addr));
portno = atoi(argv[1]);
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(portno);
if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0)
    error("ERROR on binding");

listen(sockfd,5);
clilen = sizeof(cli_addr);

char **command;
char *input;
while (1)
{
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    input = readline("socket: ");
        command = get_input(input);
    pid = fork();
    if (pid < 0)
        error("ERROR on fork");
```

```
         else if (pid == 0)
        {
             dup2(newsockfd, fflush(stdout));
             execlp(command[0], command);
             close(sockfd);
             dostuff(newsockfd);
             exit(0);
         }
         else
                while(wait(NULL)>0);
     } /* end of while */

     close(sockfd);

     return 0; /* we never get here */
}

/******** DOSTUFF() *********************
 There is a separate instance of this function
 for each connection.  It handles all communication
 once a connnection has been established.
 *****************************************/
void dostuff (int sock)
{
   int n;
   char buffer[256];

   bzero(buffer,256);
   n = read(sock,buffer,255);
   if (n < 0) error("ERROR reading from socket");
   printf("Here is the message: %s\n",buffer);
```

```
  n = write(sock,"I got your message",18);
  if (n < 0) error("ERROR writing to socket");
}
```

Client

References: http://www.linuxhowtos.org/C_C++/socket.htm

: https://stackoverflow.com/questions/35443876/executing-commands-via-sockets-with-popen

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

void error(const char *msg)
{
   perror(msg);
   exit(0);
}

int main(int argc, char *argv[])
{
   int sockfd, portno, n;
   struct sockaddr_in serv_addr;
   struct hostent *server;

   char buffer[256];
   if (argc < 3)
  {
     fprintf(stderr,"usage %s hostname port\n", argv[0]);
     exit(0);
   }
   portno = atoi(argv[2]);
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
   if (sockfd < 0)
     error("ERROR opening socket");

   server = gethostbyname(argv[1]);
   if (server == NULL)
  {
     fprintf(stderr,"ERROR, no such host\n");
     exit(0);
   }
   bzero((char *) &serv_addr, sizeof(serv_addr));
   serv_addr.sin_family = AF_INET;
   bcopy((char *)server->h_addr,
       (char *)&serv_addr.sin_addr.s_addr,
       server->h_length);
   serv_addr.sin_port = htons(portno);
   if (connect(sockfd,(struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
      error("ERROR connecting");
   printf("Please enter the message: ");
   bzero(buffer,256);
   fgets(buffer,255,stdin);
   n = write(sockfd,buffer,strlen(buffer));

   char *cp;
   char command[256];
   int acc_sock;
   FILE *pin;
   pin = popen(buffer, "r");

   while (1)
   {
           cp = fgets(command, 256, pin);
           if (cp == NULL)
              break;

          n = strlen(command);
          write(acc_sock, command, n);
       }
       pclose(pin);



       if (n < 0)
          error("ERROR writing to socket");

       bzero(buffer,256);
       n = read(sockfd,buffer,255);
       if (n < 0)
          error("ERROR reading from socket");

       printf("%s\n",buffer);
       close(sockfd);

       return 0;
   }
```