Κατανεμημένα Συστήματα Επικοινωνία Client/Server Υποδοχές (Παραδείγματα)

Χάρης Μανιφάβας
Τμήμα Εφ. Πληροφορικής & Πολυμέσων
ΤΕΙ Κρήτης

Συναρτήσεις Ανάκτησης Ονομάτων

Y:\jb\TEI\Classes\L-DistributedSystems\src\inet_addr.c

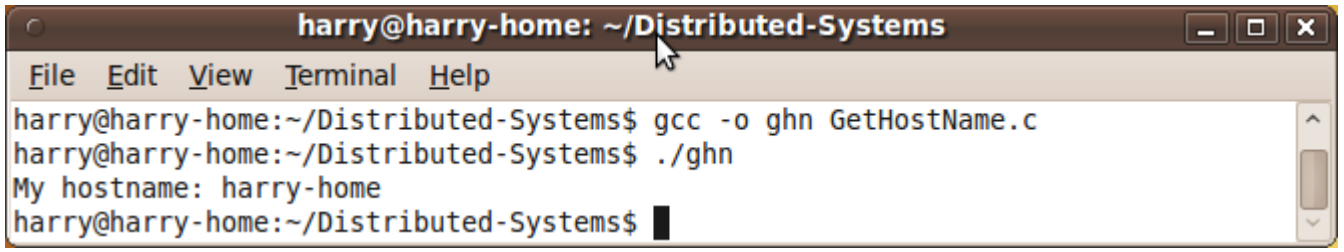
```c
#include <stdio.h>
#include <netinet/in.h>
#include <sys/socket.h>

int main()
{
struct sockaddr_in antelope;
char *some_addr;

inet_aton("10.0.0.1", &antelope.sin_addr); // store IP in antelope

some_addr = inet_ntoa(antelope.sin_addr); // return the IP
printf("%s\n", some_addr); // prints "10.0.0.1"

// and this call is the same as the inet_aton() call, above:
antelope.sin_addr.s_addr = inet_addr("10.0.0.1");
}
```

```
harry@harry-home: ~/Distributed-Systems
File  Edit  View  Terminal  Help
harry@harry-home:~/Distributed-Systems$ gcc -o inet_addr inet_addr.c
inet_addr.c: In function 'main':
inet_addr.c:16: warning: assignment makes pointer from integer without a cast
harry@harry-home:~/Distributed-Systems$ ./inet_addr
10.0.0.1
harry@harry-home:~/Distributed-Systems$
```

Συναρτήσεις Ανάκτησης Ονομάτων

Y:\jb\TEI\Classes\L-DistributedSystems\src\GetHostName.c

```c
#include <stdio.h>
#include <errno.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main()
{
  char hostname[128];

  gethostname(hostname, sizeof hostname);
  printf("My hostname: %s\n", hostname);
}
```

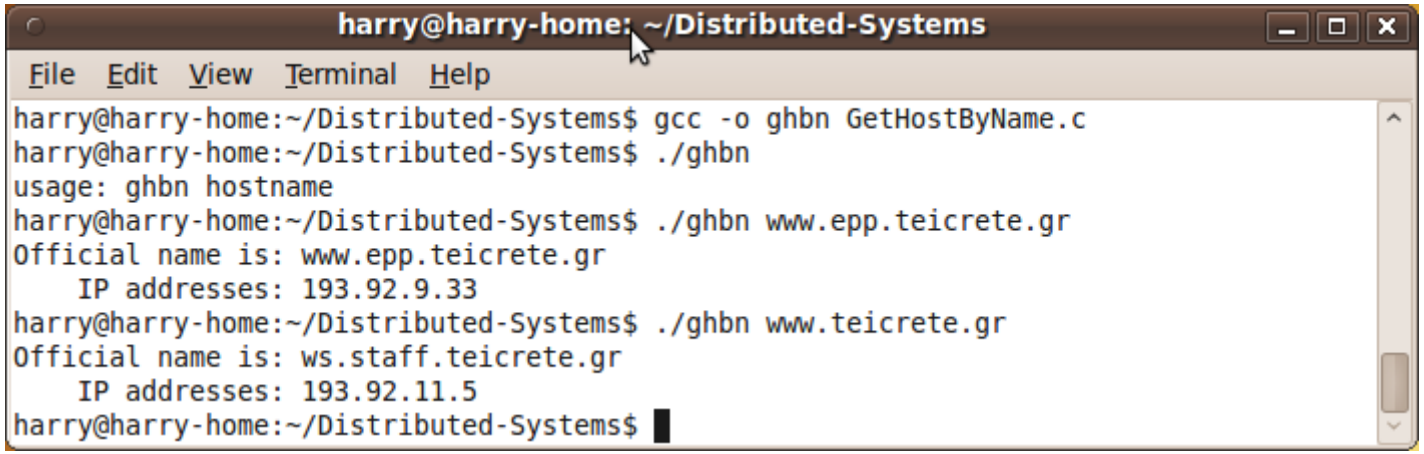


Συναρτήσε

Y:\jb\TEI\Classes\L-DistributedSystems\src\GetHostByName.c

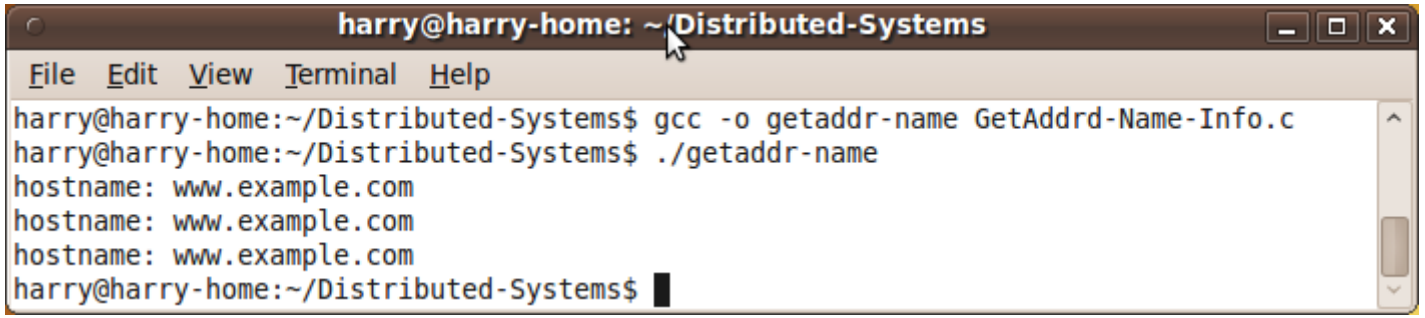
```c
// THIS IS A DEPRECATED METHOD OF GETTING HOST NAMES
// use getaddrinfo() instead!

#include <stdio.h>
#include <errno.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(int argc, char *argv[])
{
    int i;
    struct hostent *he;
    struct in_addr **addr_list;

    if (argc != 2) {
        fprintf(stderr,"usage: ghbn hostname\n");
        return 1;
    }

    if ((he = gethostbyname(argv[1])) == NULL) {   // get the host info
        herror("gethostbyname");
        return 2;
    }

    // print information about this host:
    printf("Official name is: %s\n", he->h_name);
    printf("    IP addresses: ");
    addr_list = (struct in_addr **)he->h_addr_list;
    for(i = 0; addr_list[i] != NULL; i++) {
        printf("%s ", inet_ntoa(*addr_list[i]));
    }
    printf("\n");

    return 0;
}
```

Συναρτήσεις Ανάκτησης Ονομάτων



Συναρτήσεις Ανάκτησης Ονομάτων

- getaddrinfo() and getnameinfo() functions
  - Example
    - The following example uses getaddrinfo() to resolve the domain name *www.example.com* into its list of addresses and then calls getnameinfo() on each result to return the canonical name for the address
    - In general, this will produce the original hostname, unless the particular address has multiple names, in which case the *canonical* name is returned
    - In this example, the domain name will be printed three times, once for each of the three results obtained



Συν

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

#ifndef   NI_MAXHOST
#define   NI_MAXHOST 1025
#endif

int main(void)
{
    struct addrinfo * result;
    struct addrinfo * res;
    int error;

    /* resolve the domain name into a list of addresses */
    error = getaddrinfo("www.example.com", NULL, NULL, &result);

    if (error != 0)
    {
        fprintf(stderr, "error in getaddrinfo: %s\n", gai_strerror(error));
        return 1;
    }

    /* loop over all returned results and do inverse lookup */
    for (res = result; res != NULL; res = res->ai_next)
    {
        char hostname[NI_MAXHOST] = "";

        error = getnameinfo(res->ai_addr, res->ai_addrlen, hostname, NI_MAXHOST, NULL, 0, 0);

        if (error != 0)
        {
            fprintf(stderr, "error in getnameinfo: %s\n", gai_strerror(error));
            continue;
        }

        if (*hostname)
        {
            printf("hostname: %s\n", hostname);
        }

    }

    freeaddrinfo(result);

    return 0;
}
```

UDP C/S ex1

Y:\jbt\TEI\Classes\L-DistributedSystems\src\UDP-Server-ex1.c

```c
/* fpont 12/99 */
/* pont.net  */
/* udpServer.c */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h> /* close() */
#include <string.h> /* memset() */
#include <stdlib.h> /* exit() */


#define LOCAL_SERVER_PORT 1500
#define MAX_MSG 100

int main(int argc, char *argv[]) {

  int sd, rc, n, cliLen;
  struct sockaddr_in cliAddr, servAddr;
  char msg[MAX_MSG];

  /* socket creation */
  sd=socket(AF_INET, SOCK_DGRAM, 0);
  if(sd<0) {
    printf("%s: cannot open socket \n",argv[0]);
    exit(1);
  }

  /* bind local server port */
  servAddr.sin_family = AF_INET;
  servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
  servAddr.sin_port = htons(LOCAL_SERVER_PORT);
  rc = bind (sd, (struct sockaddr *) &servAddr,sizeof(servAddr));
  if(rc<0) {
    printf("%s: cannot bind port number %d \n",
     argv[0], LOCAL_SERVER_PORT);
    exit(1);
```

UDP C/S ex1

Y:\jbi\TEI\Classes\L-DistributedSystems\src\UDP-Server-ex1.c

```
  }

  printf("%s: waiting for data on port UDP %u\n",
     argv[0],LOCAL_SERVER_PORT);

  /* server infinite loop */
  while(1) {

    /* init buffer */
    memset(msg,0x0,MAX_MSG);

    /* receive message */
    cliLen = sizeof(cliAddr);
    n = recvfrom(sd, msg, MAX_MSG, 0,
     (struct sockaddr *) &cliAddr, &cliLen);

    if(n<0) {
      printf("%s: cannot receive data \n",argv[0]);
      continue;
    }

    /* print received message */
    printf("%s: from %s:UDP%u : %s \n",
     argv[0],inet_ntoa(cliAddr.sin_addr),
     ntohs(cliAddr.sin_port),msg);

  }/* end of server infinite loop */

return 0;

}
```

UDP C/S ex1

Y:\jb\TEl\Classes\L-DistributedSystems\src\UDP-Client-ex1.c

```c
/* fpont 12/99 */
/* pont.net   */
/* udpClient.c */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h> /* memset() */
#include <sys/time.h> /* select() */

#define REMOTE_SERVER_PORT 1500
#define MAX_MSG 100


int main(int argc, char *argv[]) {

  int sd, rc, i;
  struct sockaddr_in cliAddr, ServAddr;
  struct hostent *h;

  /* check command line args */
  if(argc<3) {
    printf("usage : %s <server> <data1> ... <dataN> \n", argv[0]);
    exit(1);
  }

  /* get server IP address (no check if input is IP address or DNS name */
  h = gethostbyname(argv[1]);
  if(h==NULL) {
    printf("%s: unknown host '%s' \n", argv[0], argv[1]);
    exit(1);
  }

  printf("%s: sending data to '%s' (IP : %s) \n", argv[0], h->h_name,
   inet_ntoa(*(struct in_addr *)h->h_addr_list[0]));
```

UDP C/S ex1

Y:\jb\TEI\Classes\L-DistributedSystems\src\UDP-Client-ex1.c

```
  ServAddr.sin_family = h->h_addrtype;
  memcpy((char *) &ServAddr.sin_addr.s_addr,
   h->h_addr_list[0], h->h_length);
  ServAddr.sin_port = htons(REMOTE_SERVER_PORT);

  /* socket creation */
  sd = socket(AF_INET,SOCK_DGRAM,0);
  if(sd<0) {
    printf("%s: cannot open socket \n",argv[0]);
    exit(1);
  }

  /* bind any port */
  cliAddr.sin_family = AF_INET;
  cliAddr.sin_addr.s_addr = htonl(INADDR_ANY);
  cliAddr.sin_port = htons(0);

  rc = bind(sd, (struct sockaddr *) &cliAddr, sizeof(cliAddr));
  if(rc<0) {
    printf("%s: cannot bind port\n", argv[0]);
    exit(1);
  }


  /* send data */
  for(i=2;i<argc;i++) {
    rc = sendto(sd, argv[i], strlen(argv[i])+1, 0,
    (struct sockaddr *) &ServAddr,
    sizeof(ServAddr));

    if(rc<0) {
      printf("%s: cannot send data %d \n",argv[0],i-1);
      close(sd);
      exit(1);
    }

  }

  return 1;
```

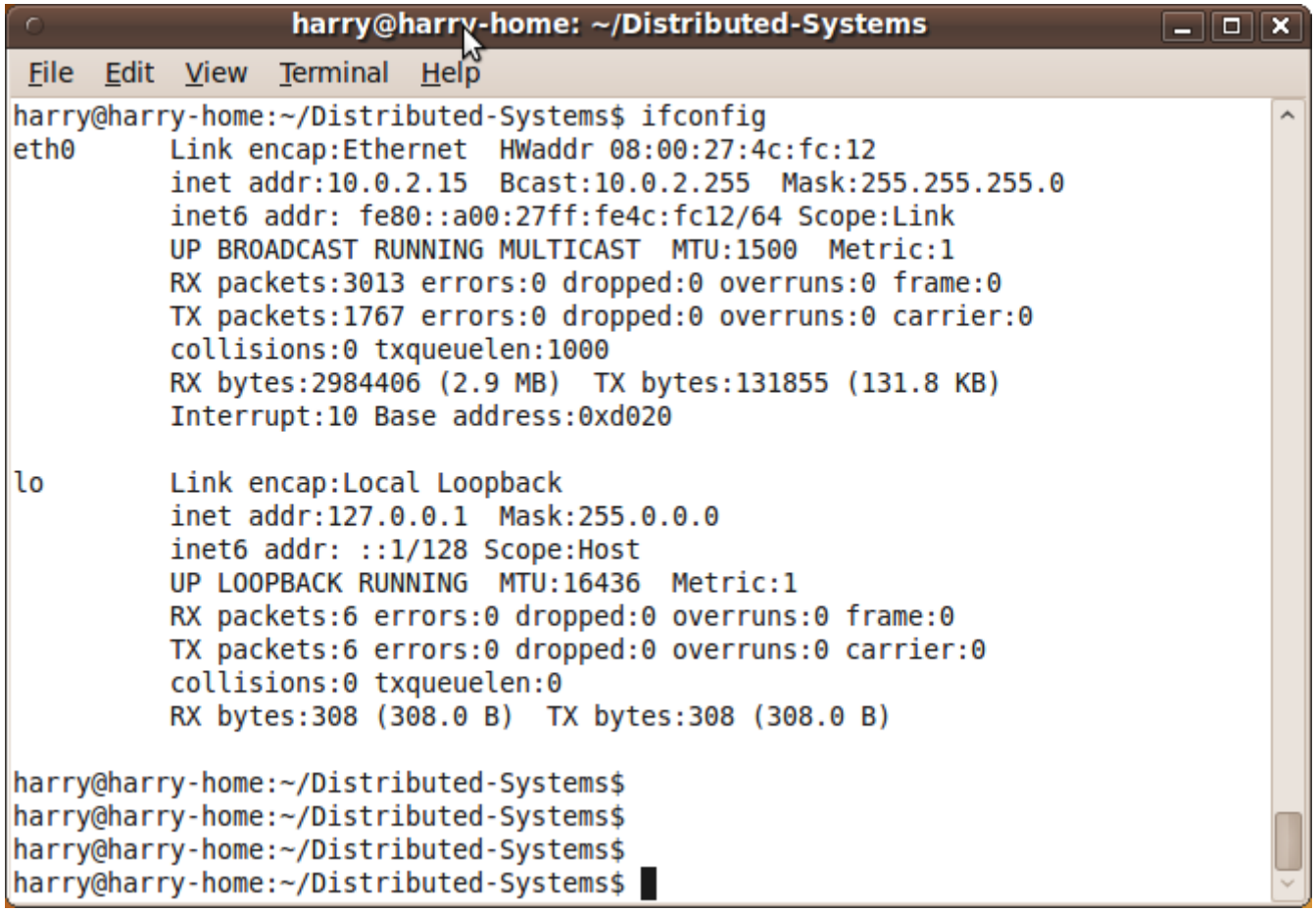
UDP C/S ex1

- UDP Client-Server (Παράδειγμα 1)
  - ifconfig
    - Shows IP address
  - whoami
    - User name
  - hostname
    - Hostname
  - arch
    - Machine architecture
  - ps
    - Process status

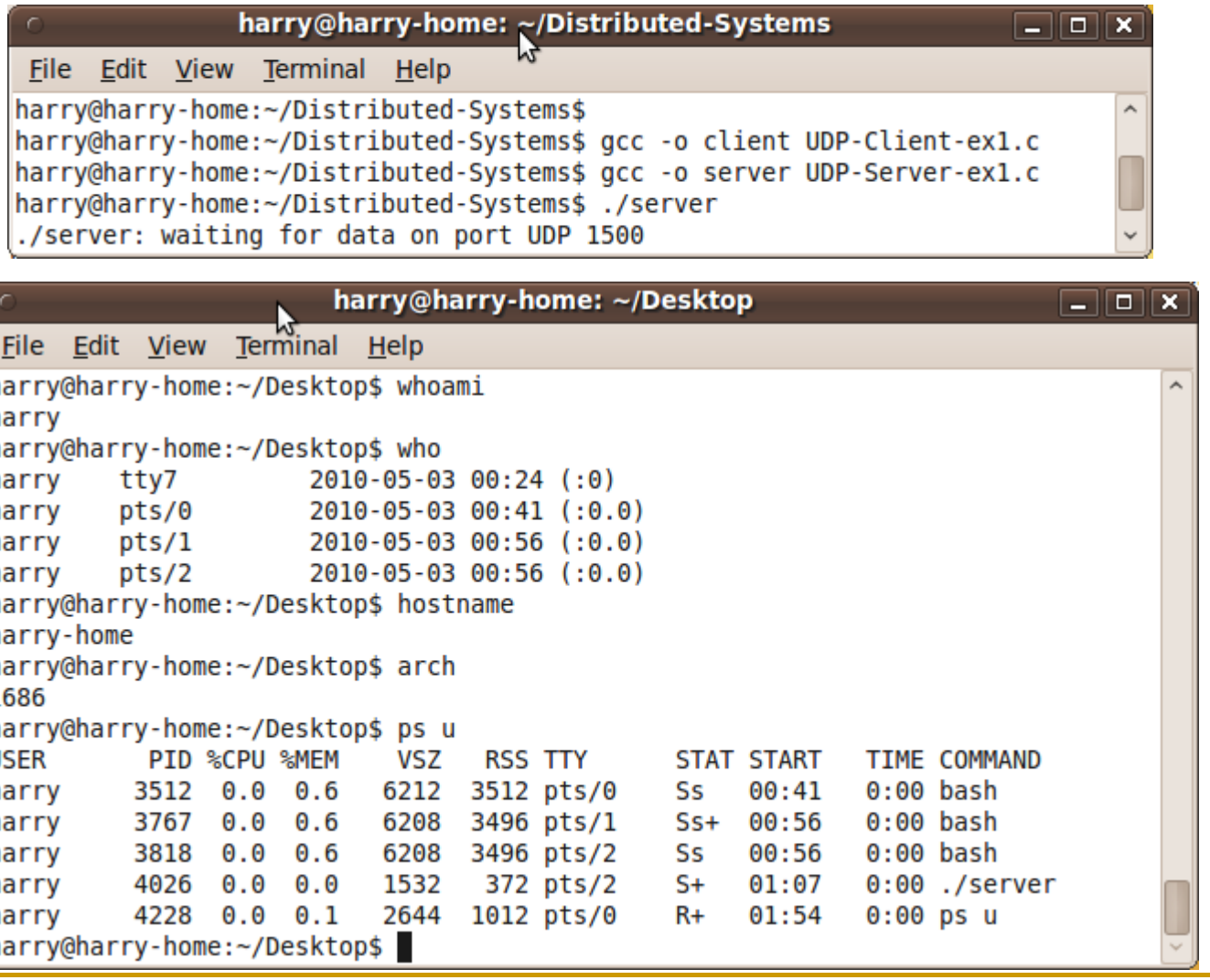
UDP C/S ex1

- UDP Client-Server (Παράδειγμα 1)
  - Αρχικά γίνεται compile ο κώδικας του Client και του Server
  - Ξεκινά ο Server
    - Που ακούει στην πόρτα 1500
  - Ξεκινά ο Client
    - Με λάθος αριθμό ορισμάτων εμπφαμίζει μήνυμα ορθής χρήσης
    - Στο παράδειγμα και οι 2 διεργασίες τρέχουν στο ίδιο μηχάνημα
    - Η διεύθυνση του Server μπρορεί να δοθεί και σαν localhost (127.0.0.1) ή με την πραγματική IP ή με το hostname
  - Επικοινωνία
    - Ο Client πέρα από το Server IP πέρνει και μία σειρά από ορίσματα (strings) που στέλνει στον Server ο οποίος τα εκτυπώνει στην οθόνη
    - Μόλις τα στείλει κάνει exit

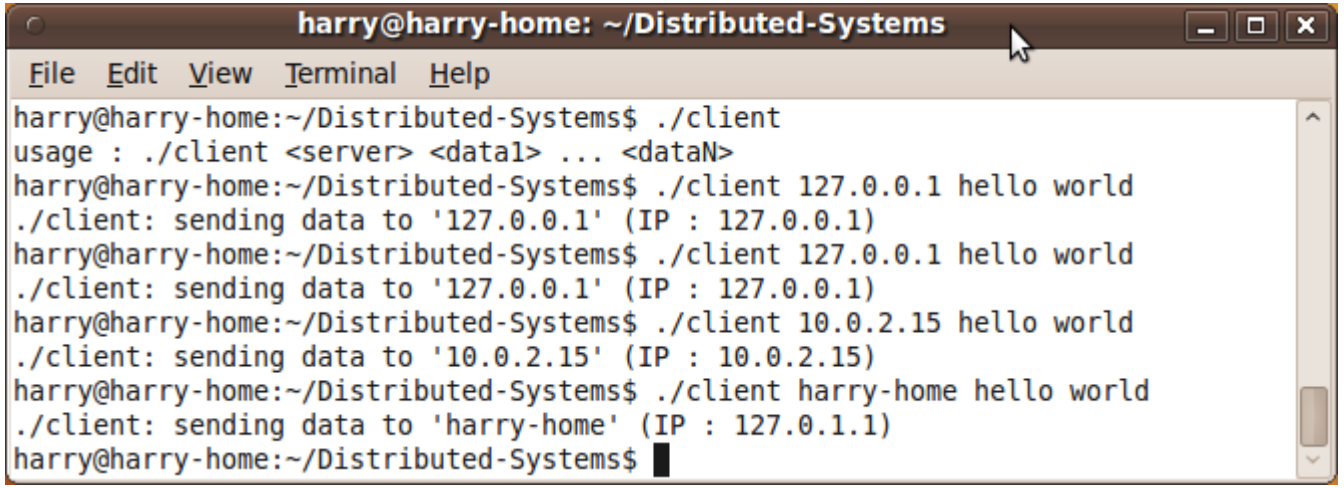
UDP C/S ex1

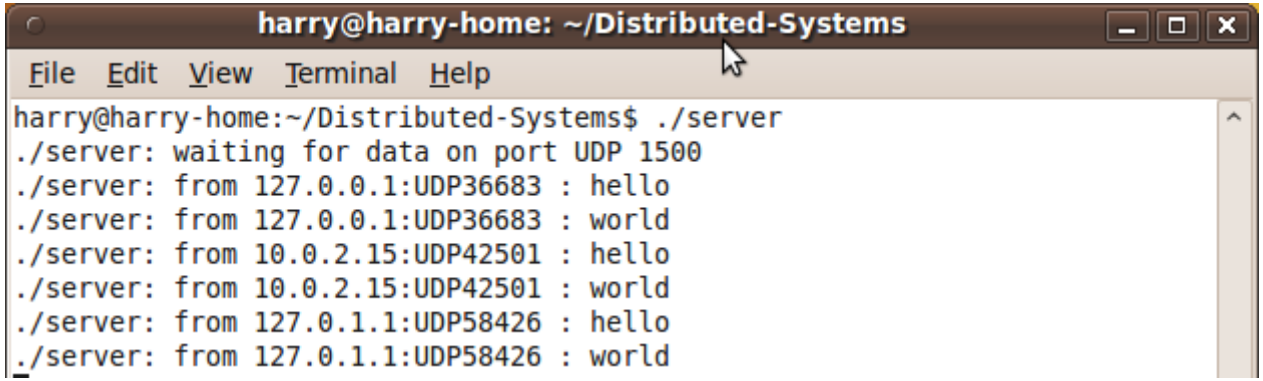

harry@harry-home: ~/Distributed-Systems
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems$ ifconfig
eth0 Link encap:Ethernet HWaddr 08:00:27:4c:fc:12
inet addr:10.0.2.15 Bcast:10.0.2.255 Mask:255.255.255.0
inet6 addr: fe80::a00:27ff:fe4c:fc12/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:3013 errors:0 dropped:0 overruns:0 frame:0
TX packets:1767 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:2984406 (2.9 MB) TX bytes:131855 (131.8 KB)
Interrupt:10 Base address:0xd020
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:6 errors:0 dropped:0 overruns:0 frame:0
TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:308 (308.0 B) TX bytes:308 (308.0 B)
harry@harry-home:~/Distributed-Systems$
harry@harry-home:~/Distributed-Systems$
harry@harry-home:~/Distributed-Systems$
harry@harry-home:~/Distributed-Systems$

UDP C/S ex1


harry@harry-home: ~/Distributed-Systems
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems$
harry@harry-home:~/Distributed-Systems$ gcc -o client UDP-Client-ex1.c
harry@harry-home:~/Distributed-Systems$ gcc -o server UDP-Server-ex1.c
harry@harry-home:~/Distributed-Systems$ ./server
./server: waiting for data on port UDP 1500
harry@harry-home: ~/Desktop
File Edit View Terminal Help
harry@harry-home:~/Desktop$ whoami
harry
harry@harry-home:~/Desktop$ who
harry tty7 2010-05-03 00:24 (:0)
harry pts/0 2010-05-03 00:41 (:0.0)
harry pts/1 2010-05-03 00:56 (:0.0)
harry pts/2 2010-05-03 00:56 (:0.0)
harry@harry-home:~/Desktop$ hostname
harry-home
harry@harry-home:~/Desktop$ arch
i686
harry@harry-home:~/Desktop$ ps u
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
harry 3512 0.0 0.6 6212 3512 pts/0 Ss 00:41 0:00 bash
harry 3767 0.0 0.6 6208 3496 pts/1 Ss+ 00:56 0:00 bash
harry 3818 0.0 0.6 6208 3496 pts/2 Ss 00:56 0:00 bash
harry 4026 0.0 0.0 1532 372 pts/2 S+ 01:07 0:00 ./server
harry 4228 0.0 0.1 2644 1012 pts/0 R+ 01:54 0:00 ps u
harry@harry-home:~/Desktop$

UDP C/S ex1


harry@harry-home: ~/Distributed-Systems
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems$ ./client
usage : ./client <server> <data1> ... <dataN>
harry@harry-home:~/Distributed-Systems$ ./client 127.0.0.1 hello world
./client: sending data to '127.0.0.1' (IP : 127.0.0.1)
harry@harry-home:~/Distributed-Systems$ ./client 127.0.0.1 hello world
./client: sending data to '127.0.0.1' (IP : 127.0.0.1)
harry@harry-home:~/Distributed-Systems$ ./client 10.0.2.15 hello world
./client: sending data to '10.0.2.15' (IP : 10.0.2.15)
harry@harry-home:~/Distributed-Systems$ ./client harry-home hello world
./client: sending data to 'harry-home' (IP : 127.0.1.1)
harry@harry-home:~/Distributed-Systems$


harry@harry-home: ~/Distributed-Systems
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems$ ./server
./server: waiting for data on port UDP 1500
./server: from 127.0.0.1:UDP36683 : hello
./server: from 127.0.0.1:UDP36683 : world
./server: from 10.0.2.15:UDP42501 : hello
./server: from 10.0.2.15:UDP42501 : world
./server: from 127.0.1.1:UDP58426 : hello
./server: from 127.0.1.1:UDP58426 : world

TCP C

Y:\jb\TEI\Classes\L-DistributedSystems\src\TCP-C-S-ex2\TCP-Server-ex2.c

```c
/* tcpserver.c */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>


int main()
{
        int sock, connected, bytes_recieved , true = 1;
        char send_data [1024] , recv_data[1024];

        struct sockaddr_in server_addr,client_addr;
        int sin_size;

        if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
            perror("Socket");
            exit(1);
        }

        if (setsockopt(sock,SOL_SOCKET,SO_REUSEADDR,&true,sizeof(int)) == -1) {
            perror("Setsockopt");
            exit(1);
        }

        server_addr.sin_family = AF_INET;
        server_addr.sin_port = htons(5000);
        server_addr.sin_addr.s_addr = INADDR_ANY;
        bzero(&(server_addr.sin_zero),8);

        if (bind(sock, (struct sockaddr *)&server_addr, sizeof(struct sockaddr))
                                                                    == -1) {
            perror("Unable to bind");
            exit(1);
        }

        if (listen(sock, 5) == -1) {
            perror("Listen");
            exit(1);
        }

  printf("\nTCPServer Waiting for client on port 5000");
        fflush(stdout);


        while(1)
        {
```

TCP C/S ex2

Y:\jbi\TEI\Classes\L-DistributedSystems\src\TCP-C-S-ex2\TCP-Server-ex2.c

```
            sin_size = sizeof(struct sockaddr_in);

            connected = accept(sock, (struct sockaddr *)&client_addr,&sin_size);

            printf("\n I got a connection from (%s , %d)",
                   inet_ntoa(client_addr.sin_addr),ntohs(client_addr.sin_port));

            while (1)
            {
              printf("\n SEND (q or Q to quit) : ");
              gets(send_data);

              if (strcmp(send_data , "q") == 0 || strcmp(send_data , "Q") == 0)
              {
                send(connected, send_data,strlen(send_data), 0);
                close(connected);
                break;
              }

              else
                 send(connected, send_data,strlen(send_data), 0);

              bytes_recieved = recv(connected,recv_data,1024,0);

              recv_data[bytes_recieved] = '\0';

              if (strcmp(recv_data , "q") == 0 || strcmp(recv_data , "Q") == 0)
              {
                close(connected);
                break;
              }

              else
              printf("\n RECIEVED DATA = %s " , recv_data);
              fflush(stdout);
            }
          }

        close(sock);
        return 0;
}
```

TCP C

Y:\jbi\TEI\Clasees\L-DistributedSystems\src\TCP-C-S-ex2\TCP-Client-ex2.c

```c
/* tcpclient.c */

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


int main()

{

        int sock, bytes_recieved;
        char send_data[1024],recv_data[1024];
        struct hostent *host;
        struct sockaddr_in server_addr;

        host = gethostbyname("127.0.0.1");

        if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
            perror("Socket");
            exit(1);
        }

        server_addr.sin_family = AF_INET;
        server_addr.sin_port = htons(5000);
        server_addr.sin_addr = *((struct in_addr *)host->h_addr);
        bzero(&(server_addr.sin_zero),8);

        if (connect(sock, (struct sockaddr *)&server_addr,
                    sizeof(struct sockaddr)) == -1)
        {
            perror("Connect");
            exit(1);
        }

        while(1)
        {

          bytes_recieved=recv(sock,recv_data,1024,0);
          recv_data[bytes_recieved] = '\0';

          if (strcmp(recv_data , "q") == 0 || strcmp(recv_data , "Q") == 0)
          {
           close(sock);
           break;
          }

          else
```

TCP C/S ex2

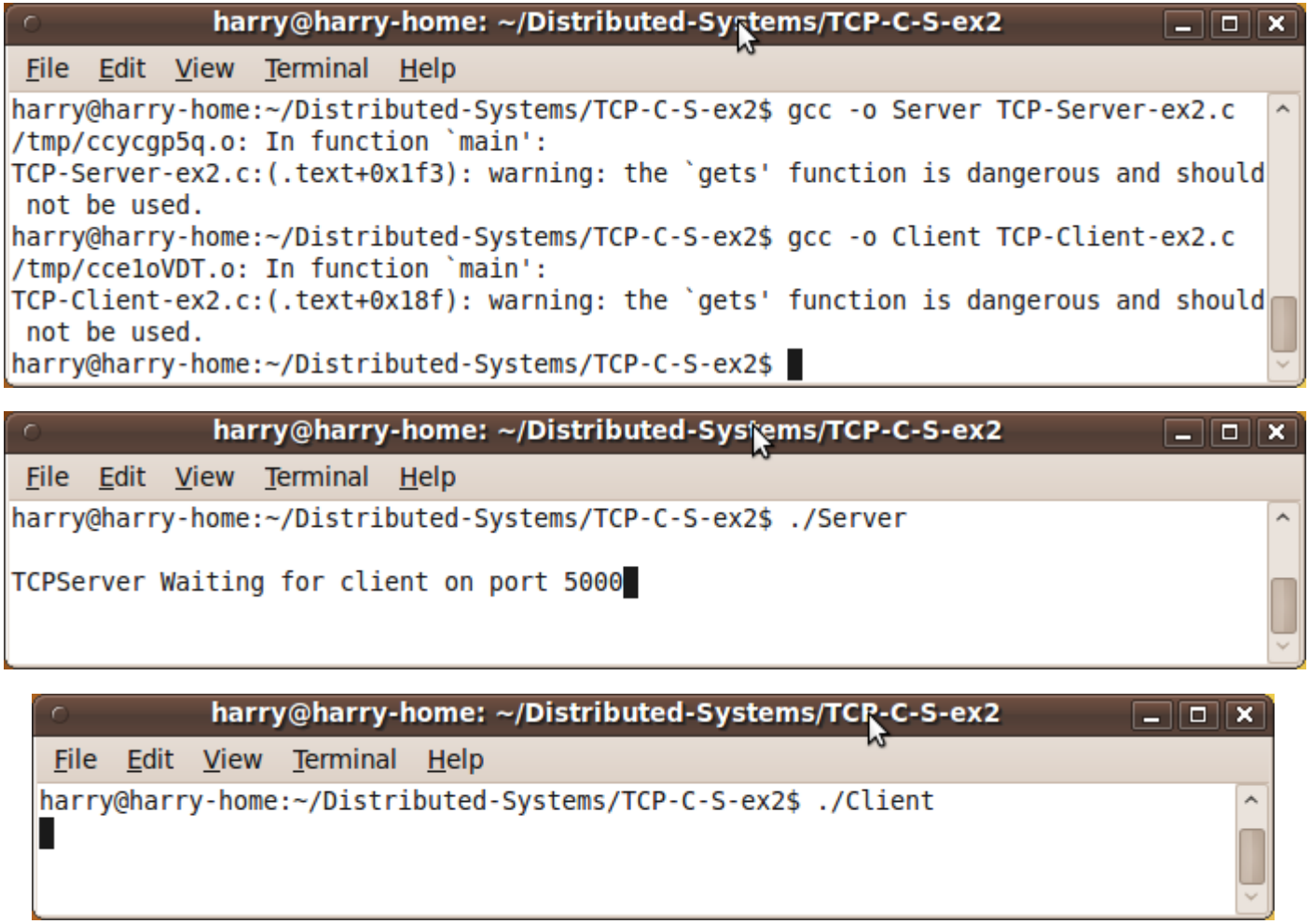
Y:\jb\TEI\Classes\L-DistributedSystems\src\TCP-C-S-ex2\TCP-Client-ex2.c

```
            printf("\nRecieved data = %s " , recv_data);

            printf("\nSEND (q or Q to quit) : ");
            gets(send_data);

           if (strcmp(send_data , "q") != 0 && strcmp(send_data , "Q") != 0)
            send(sock,send_data,strlen(send_data), 0);

           else
           {
            send(sock,send_data,strlen(send_data), 0);
            close(sock);
            break;
           }

         }
return 0;
}
```

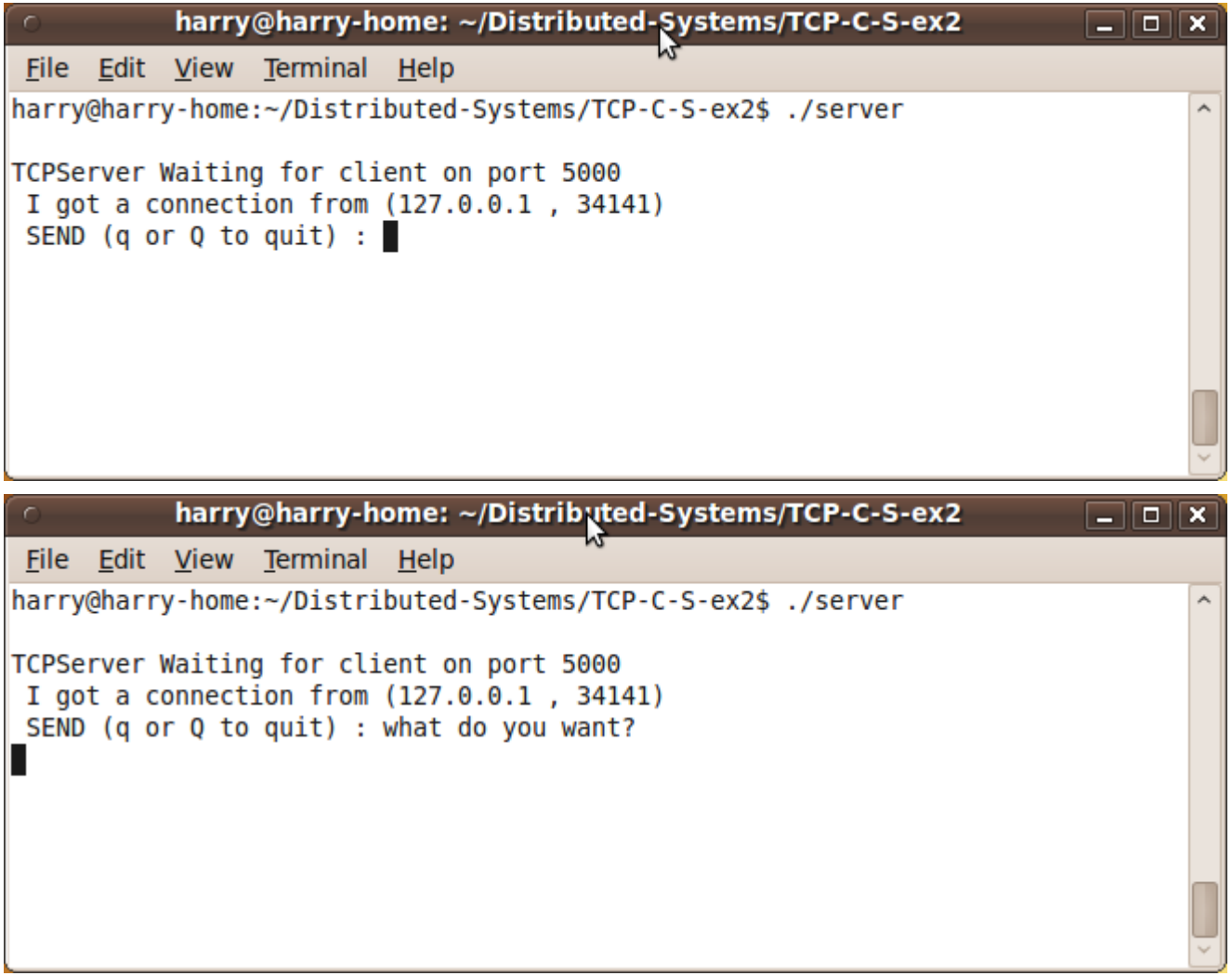
TCP C/S ex2

- TCP Client-Server (Παράδειγμα 2)
  - Αρχικά γίνεται compile ο κώδικας του Client και του Server
  - Ξεκινά ο Server
    - Που ακούει στην πόρτα 5000
  - Ξεκινά ο Client
    - Στο παράδειγμα και οι 2 διεργασίες τρέχουν στο ίδιο μηχάνημα
    - Ο Server εμφανίζει την σύνδεση που μόλις δημιουργήθηκε
  - Επικοινωνία
    - Ο Server ή ο Client μπορούν να στέλνουν μηνύματα μεταξύ τους μέχρι να πατηθεί ο χαρακτήρας q/Q στην γραμμή SEND
    - Ο παραλήπτης του μηνύματος κάθε φορά δείχνει στην οθόνη τι παρέλαβε στην γραμμή Received data
    - Ο διάλογος μπορεί να συνεχιστεί μέχρι ένας από τους δύο να κάνει quit
    - Εάν ο Server διακόψει την σύνδεση ο Client μπορεί να δημιουργήσει νέα σύνδεση
    - Για κάθε σύνδεση, η εντολή netstat δείχνει το ανοικτό socket

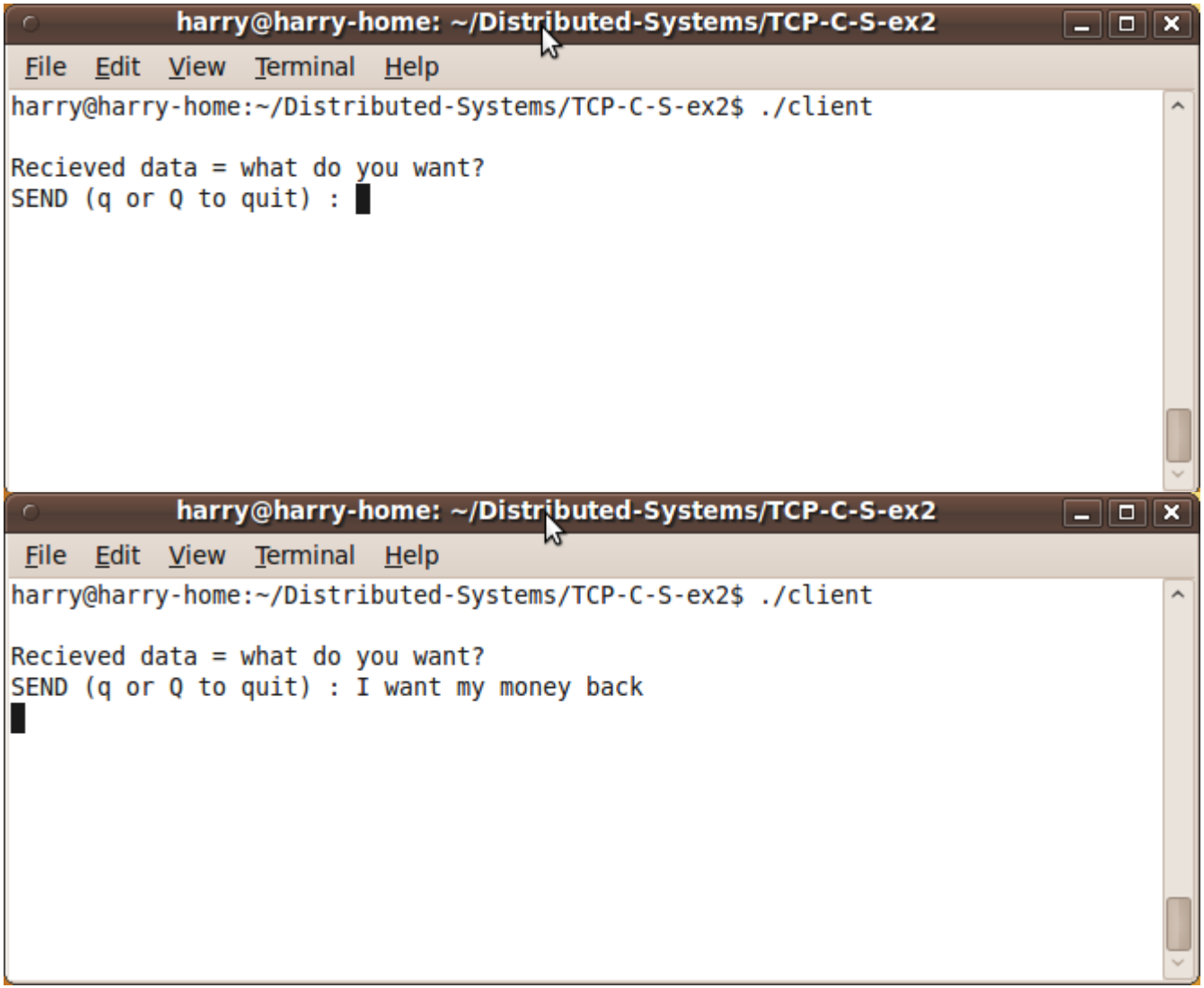
TCP C/S ex2


harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ gcc -o Server TCP-Server-ex2.c
/tmp/ccycgp5q.o: In function `main':
TCP-Server-ex2.c:(.text+0x1f3): warning: the `gets' function is dangerous and should
not be used.
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ gcc -o Client TCP-Client-ex2.c
/tmp/cce1oVDT.o: In function `main':
TCP-Client-ex2.c:(.text+0x18f): warning: the `gets' function is dangerous and should
not be used.
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$
harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./Server
TCPServer Waiting for client on port 5000
harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./Client

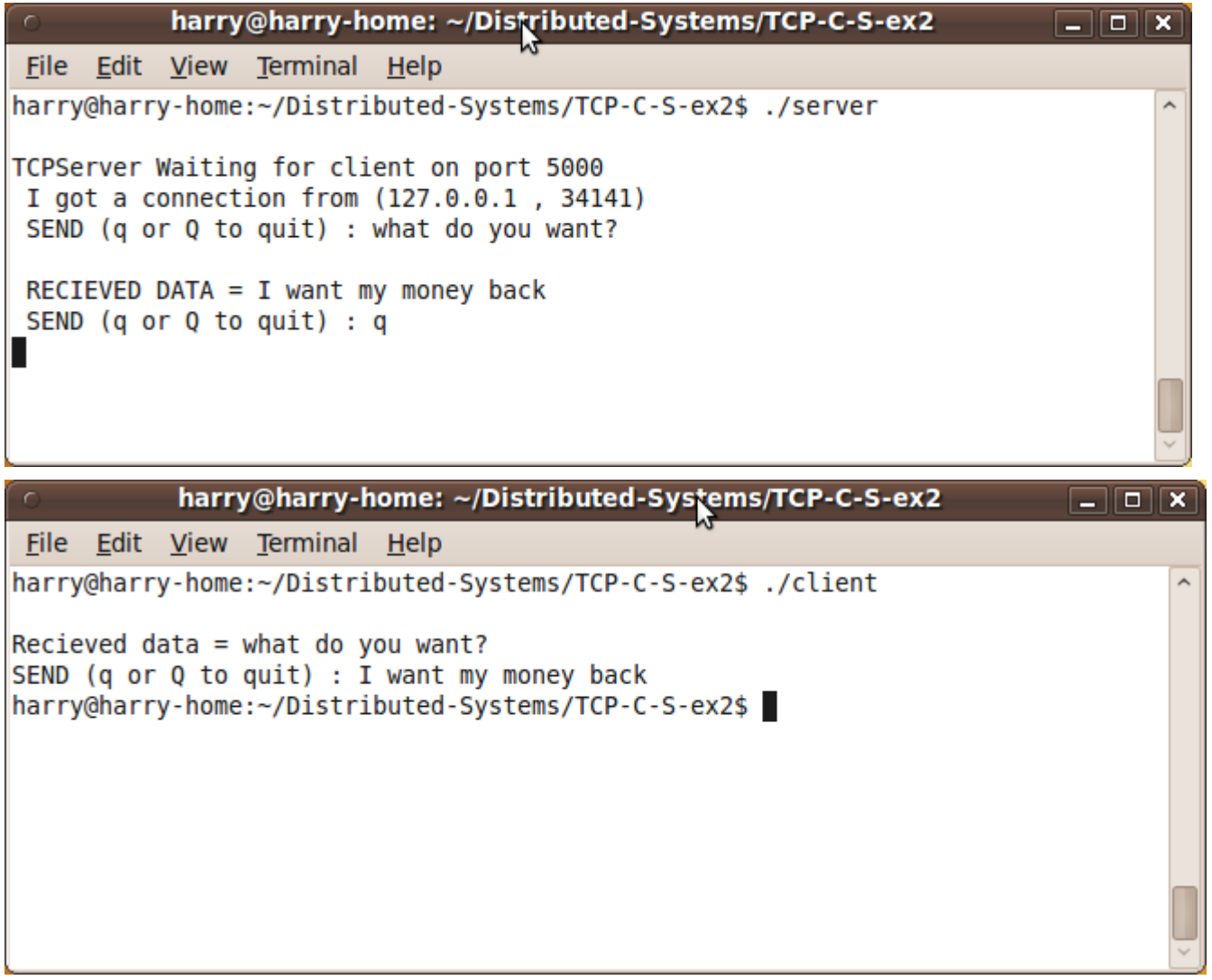
TCP C/S ex2


harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./server
TCPServer Waiting for client on port 5000
I got a connection from (127.0.0.1 , 34141)
SEND (q or Q to quit) :
harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./server
TCPServer Waiting for client on port 5000
I got a connection from (127.0.0.1 , 34141)
SEND (q or Q to quit) : what do you want?

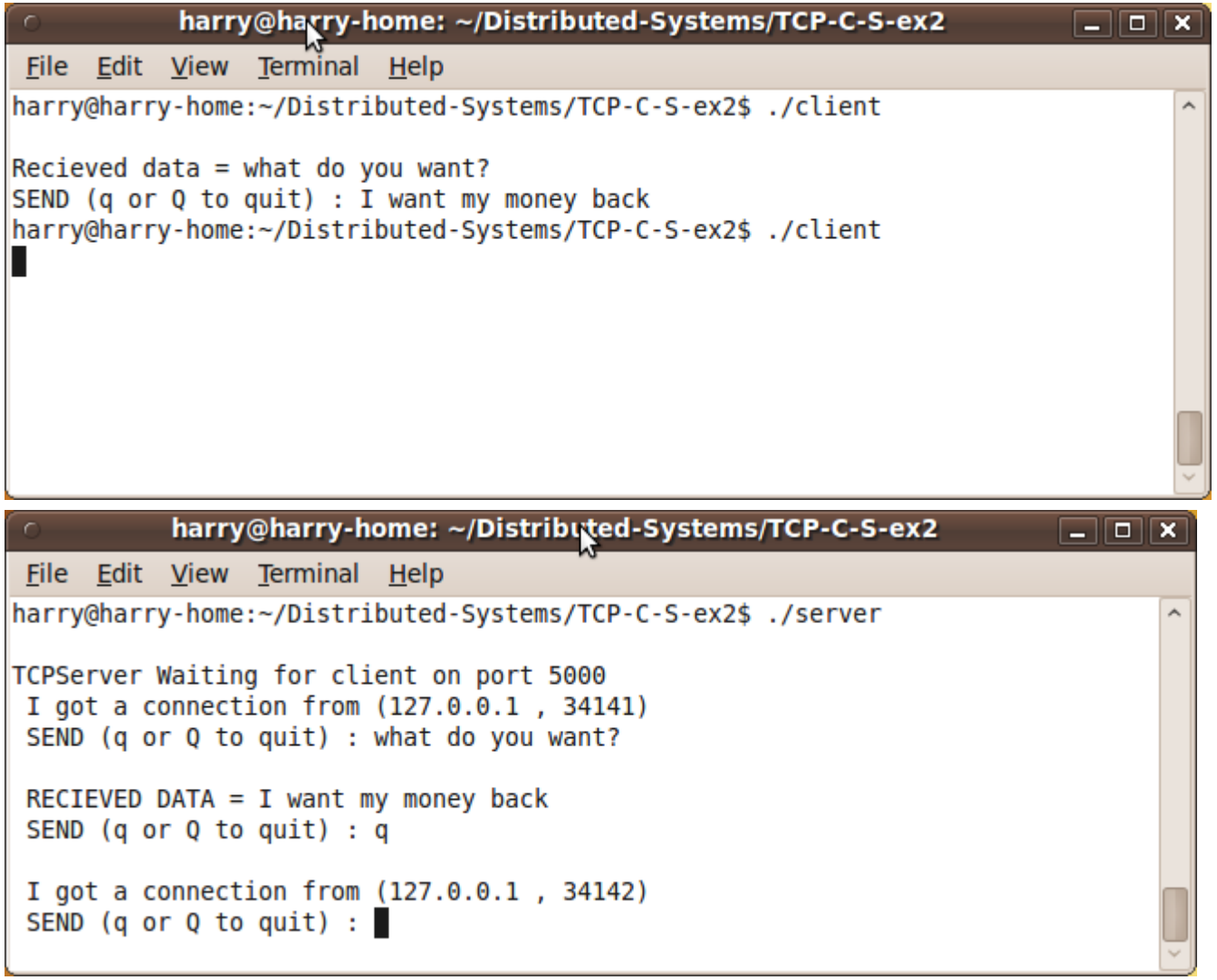
TCP C/S ex2


harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./client
Recieved data = what do you want?
SEND (q or Q to quit) :
harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./client
Recieved data = what do you want?
SEND (q or Q to quit) : I want my money back

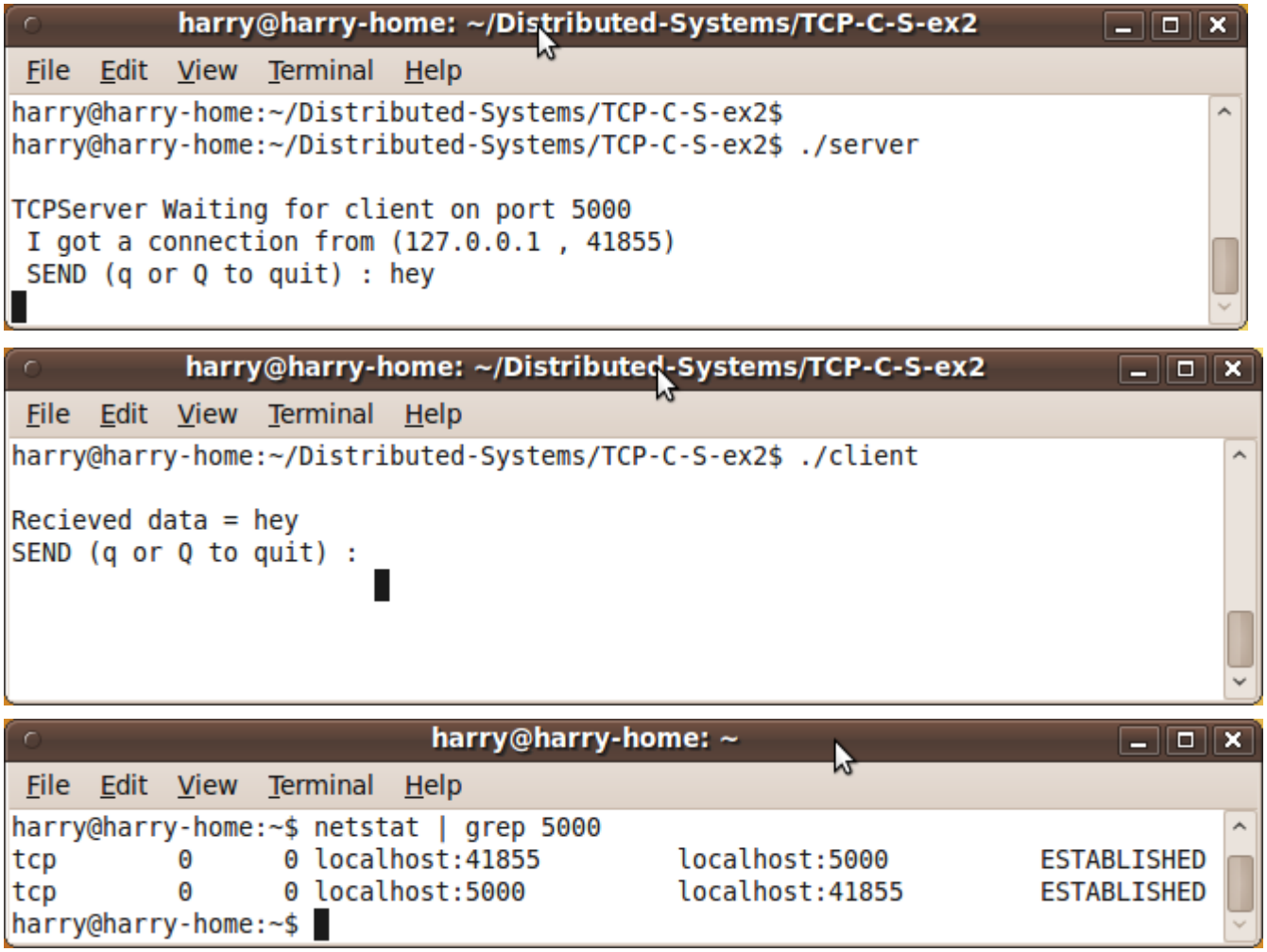
TCP C/S ex2


harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./server
TCPServer Waiting for client on port 5000
I got a connection from (127.0.0.1 , 34141)
SEND (q or Q to quit) : what do you want?
RECIEVED DATA = I want my money back
SEND (q or Q to quit) : q
harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./client
Recieved data = what do you want?
SEND (q or Q to quit) : I want my money back
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$

TCP C/S ex2


harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./client
Recieved data = what do you want?
SEND (q or Q to quit) : I want my money back
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./client
harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./server
TCPServer Waiting for client on port 5000
I got a connection from (127.0.0.1 , 34141)
SEND (q or Q to quit) : what do you want?
RECIEVED DATA = I want my money back
SEND (q or Q to quit) : q
I got a connection from (127.0.0.1 , 34142)
SEND (q or Q to quit) :

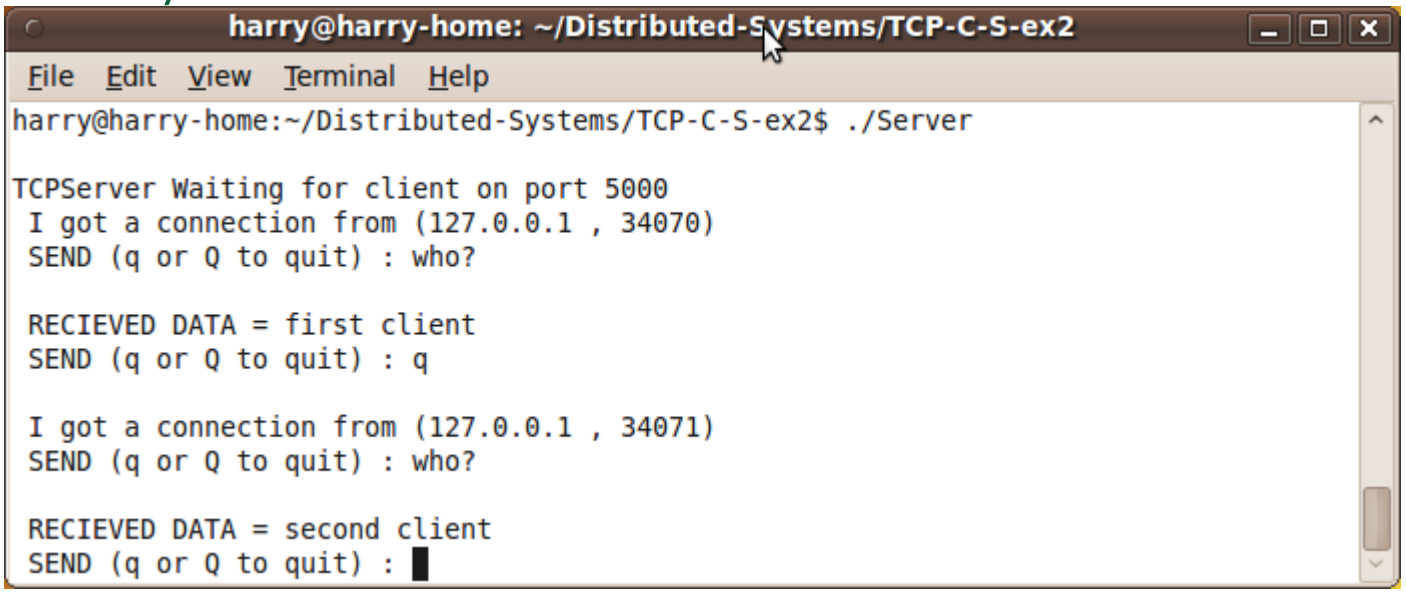
TCP C/S ex2

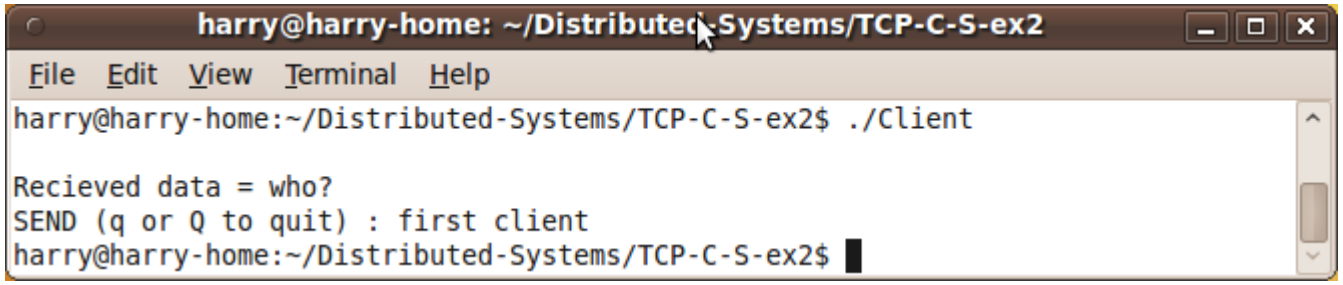

harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./server
TCPServer Waiting for client on port 5000
I got a connection from (127.0.0.1 , 41855)
SEND (q or Q to quit) : hey
harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./client
Recieved data = hey
SEND (q or Q to quit) :
harry@harry-home: ~
File Edit View Terminal Help
harry@harry-home:~$ netstat | grep 5000
tcp 0 0 localhost:41855 localhost:5000 ESTABLISHED
tcp 0 0 localhost:5000 localhost:41855 ESTABLISHED
harry@harry-home:~$

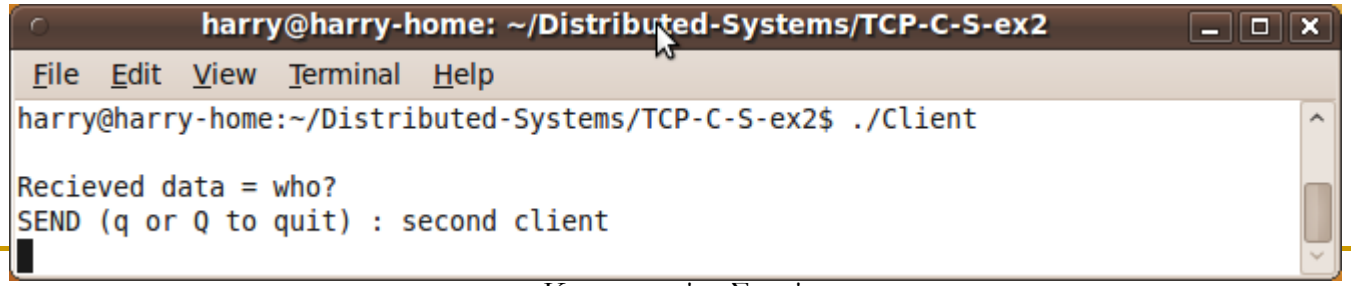
TCP C/S ex2

- TCP Client-Server (Παράδειγμα 2)
  - Ξεκινά ο Server
    - Που ακούει στην πόρτα 5000
  - Ξεκινά ο Client
    - Στο παράδειγμα και οι 2 διεργασίες τρέχουν στο ίδιο μηχάνημα
    - Ο Server εμφανίζει την σύνδεση που μόλις δημιουργήθηκε
  - Ξεκινά και 2ος Client !
    - Ο συγκεκριμένος Server δεν υποστηρίζει threads
    - Συγκρατεί την 2η αίτηση στην ουρά μέχρι να ολοκληρώσει την εξυπηρέτηση του 1ου Client
    - Ο Server ή ο Client μπορούν να στέλνουν μηνύματα μεταξύ τους μέχρι να πατηθεί ο χαρακτήρας q/Q στην γραμμή SEND
    - Ο παραλήπτης του μηνύματος κάθε φορά δείχνει στην οθόνη τι παρέλαβε στην γραμμή Received data
    - Ο διάλογος μπορεί να συνεχιστεί μέχρι ένας από τους δύο να κάνει quit
    - Για κάθε σύνδεση, η εντολή netstat δείχνει το ανοικτό socket

TCP C/S ex2


harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./Server
TCPServer Waiting for client on port 5000
I got a connection from (127.0.0.1 , 34070)
SEND (q or Q to quit) : who?
RECIEVED DATA = first client
SEND (q or Q to quit) : q
I got a connection from (127.0.0.1 , 34071)
SEND (q or Q to quit) : who?
RECIEVED DATA = second client
SEND (q or Q to quit) :


harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./Client
Recieved data = who?
SEND (q or Q to quit) : first client
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$


harry@harry-home: ~/Distributed-Systems/TCP-C-S-ex2
File Edit View Terminal Help
harry@harry-home:~/Distributed-Systems/TCP-C-S-ex2$ ./Client
Recieved data = who?
SEND (q or Q to quit) : second client

Ασκήσεις

- **ΑΣΚΗΣΗ 1 -- Υπηρεσία ανακοίνωσης ώρας**
  - Κάνετε τις αλλαγές που χρειάζονται στον κώδικα και στον client και serverώστε ο αριθμός του port να είναι όρισμα στο command line
    - Να μην είναι δηλαδή μία σταθερά μέσα στο πρόγραμμα (το 1500 στην περίπτωσή του κώδικα που σας δόθηκε)
  - Κάνετε τις αλλαγές που χρειάζονται στον client και server ώστε ο server να επιστρέφει στον client την ώρα που έχει το μηχάνημα στο οποίο τρέχει
    - Στον server θα πρέπει να προσθέσετε κώδικα που να ζητά την ώρα από το σύστημα, και κατόπιν να την στέλνετε μέσα από το socket στον client
    - Ο client κάνει receive τα δεδομένα από το socket και τα τυπώνει στην οθόνη του

Ασκήσεις

- **ΑΣΚΗΣΗ 2 -- Υπηρεσία παροχής νομισματικών ισοτιμιών**
  - Κάνετε τις αλλαγές που χρειάζονται στον client και server ώστε όταν ο client στέλνει ένα νόμισμα στον server, ο server να απαντά με την ισοτιμία του συγκεκριμένου νομίσματος με όλα τα υπόλοιπα νομίσματα
    - Σύμφωνα με το: http://www.x-rates.com/
    - Αν για παράδειγμα ο client στείλει USD, ο server θα απαντήσει με όλα τα στοιχεία της αντίστοιχης γραμμής του πίνακα, δηλαδή: 1 USD = 0.65 GBP, 0.75 EUR, κλπ.

	USD	GBP	CAD	EUR	AUD
	1	1.52993	0.995213	1.33149	0.931636
	0.653624	1	0.650495	0.870299	0.608939
	1.00481	1.53729	1	1.3379	0.936117
	0.751033	1.14902	0.747437	1	0.699689
	1.07338	1.64219	1.06824	1.4292	1

Friday, April 30, 2010

Ασκήσεις

- **ΑΣΚΗΣΗ 3 – Registration Server**
  - Κάνετε ότι αλλαγές χρειάζονται στον client και server έτσι ώστε ο client να μπορεί να στέλνει τα παρακάτω μηνύματα και ο server να ενεργεί ανάλογα
    - client -> server: register name – Ο server εγγράφει τον client σε μία δομή (πχ πίνακα με strings), δηλαδή τοποθετεί το όνομά του (π.χ. name) στον πίνακα
    - client -> server: unregister name - Ο server σβήνει το όνομα του client από την δομή (εφόσον είναι εγγεγραμμένος
    - client -> server : who name - Ο server επιστρέφει στον client όλα τα ονόματα τα οποία έχουν εγγραφεί σε αυτόν (Το name είναι το όνομα του client που κάνει την ερώτηση)
  - Στον server θα πρέπει να προσθέσετε κώδικα που διαχειρίζεται ένα πίνακα από strings
    - Θα χρειαστεί να χρησιμοποιήσετε κάποιες ή όλες από τις παρακάτω συναρτήσεις:
    - strcpy(), strcmp(), sprintf(), sscanf(), strcat()
  - Στον κώδικα που σας έχει δοθεί ένα msg μεταφέρεται από την μία οντότητα στην άλλη
    - Σε αυτή την άσκηση το μήνυμα θα περιέχει συγκεκριμένα πράγματα (κάποια εντολή από τον client) και το περιεχόμενο του πρέπει να ελέγχεται από τον server
    - Το μήνυμα της απάντησης του server πρέπει να περιέχει το αποτέλεσμα

Ασκήσεις

- **ΑΣΚΗΣΗ 4 -- Υπηρεσία απομακρυσμένης εκτέλεσης ενός συνόλου εντολών**
  - Στο σύστημα αυτό, ένας ή περισσότεροι servers S1, S2, ..., Sn εκτελούνται ταυτόχρονα σε κάποιους Η/Υ
  - Κάθε server μπορεί να εκτελεί ένα σύνολο εντολών, πχ
    - CMD(S1) = {SORT-UP, SORT-DOWN, REVERSE, PERMUTE}
    - CMD(S2) = {MIN, MAX}
  - Κάθε εντολή CMD δέχεται έναν αριθμό ορισμάτων, τα οποία είναι ακέραιοι αριθμοί, πχ
    - SORT-UP 3 1 9 5 2
    - PERMUTE 5 2 9 0
  - Κάθε εντολή CMD μπορεί να δεχτεί έναν μη προκαθορισμένο αριθμό ορισμάτων

Ασκήσεις

- **ΑΣΚΗΣΗ 4 -- Υπηρεσία απομακρυσμένης εκτέλεσης ενός συνόλου εντολών**
  - Ο binder είναι ένας άλλος τύπου server και τρέχει σε ένα αντίτυπο
    - Ο binder γνωρίζει ποιοι servers μπορούν να εκτελούν ποιες εντολές
    - Κάθε client που θέλει να εκτελέσει μία εντολή, ρωτάει πρώτα τον binder για να μάθει ποιος server μπορεί να εκτελέσει την εντολή αυτή
    - Ο binder επιστρέφει στον client μία απάντηση με τη διεύθυνση και το port του κατάλληλου server και στη συνέχεια ο client επικοινωνεί απευθείας με τον server για να εκτελέσει την εντολή

Ασκήσεις

- **ΑΣΚΗΣΗ 4 -- Υπηρεσία απομακρυσμένης εκτέλεσης ενός συνόλου εντολών**
  - Κατασκευάστε 3 servers, όπου ο πρώτος θα εκτελεί τις εντολές MIN, MAX, ο δεύτερος τις εντολές SORTU (sort up), SORTD (sort down) και ο τρίτος θα είναι ο binder
    - Θεωρείστε ότι όλες οι εντολές δέχονται 4 ακέραιους αριθμούς σαν ορίσματα
  - Παράδειγμα:
    - Εάν ο client επιθυμεί να εκτελέσει την εντολή MAX, τότε πρώτα επικοινωνεί με τον binder ο οποίος του επιστρέφει το IP address + port του server που μπορεί να εκτελέσει τη συγκεκριμένη εντολή (έστω ότι είναι ο S2)
    - Ο client κατόπιν επικοινωνεί με τον S2 στέλνοντάς του την εντολή και τα ορίσματα (έστω MAX 1 2 3 4)
    - Ο S2 απαντά στον client με το αποτέλεσμα (4 στην περίπτωσή μας) και τελικά ο client εκτυπώνει στην οθόνη το αποτέλεσμα που δέχθηκε από τον server

Ασκήσεις

- **ΑΣΚΗΣΗ 4 -- Υπηρεσία απομακρυσμένης εκτέλεσης ενός συνόλου εντολών**
  - **BINDER**
    - Η διεργασία του binder εκτελείται σε έναν Η/Υ με γνωστή IP διεύθυνση (πχ pc1.epp.teicrete.gr)
    - Κατά την εκκίνησή του, ο binder ανοίγει ένα UDP socket σε μία προκαθορισμένη πόρτα (BINDER_PORT = 5500) και δέχεται μηνύματα (UDP datagrams) τα οποία περιέχουν ως data:
    - Α) το string REGISTER COMMAND_1 COMMAND_2 COMMAND_k από κάθε server Si, που επιθυμεί να προσφέρει προς εκτέλεση το σύνολο εντολών του CMD(Si) = {COMMAND_1 COMMAND_2, COMMAND_k} (πχ REGISTER SORT)
      - Με την λήψη της εντολής ο binder εισάγει σε μία εσωτερική του δομή δεδομένων, (για κάθε COMMAND_j), το όνομα της εντολής, την IP διεύθυνση και το port του server ο οποίος την υλοποιεί, και τυπώνει τις εγγραφές στην οθόνη της μηχανής στην οποία τρέχει
    - Β) το string UNREGISTER από κάθε server Si, που επιθυμεί να διαγραφεί από την υπηρεσία
      - Με την λήψη της εντολής ο binder διαγράφει από την εσωτερική του δομή δεδομένων, όλες τις εγγραφές που έχουν την IP διεύθυνση και το port του server Si, αφού τις τυπώσει στην οθόνη
    - Γ) το string LOOKUP COMMAND_m από κάποιον client που επιθυμεί να εκτελέσει απομακρυσμένα την εντολή COMMAND_m
      - Με την λήψη της εντολής ο binder αναζητά στην εσωτερική δομή δεδομένων όλες τις εγγραφές που περιέχουν το COMMAND_m και τις τυπώνει στην οθόνη
      - Στη συνέχεια επιστρέφει ένα μήνυμα στον client (στο ίδιο IP και port από το οποίο ήρθε το datagram) με περιεχόμενο ένα string με το COMMAND_m, το IP και το port του server ο οποίος υλοποιεί την εντολή COMMAND_m serverIP serverPort
      - Αν βρει παραπάνω από μία εγγραφή (δηλαδή πολλοί servers προσφέρουν την εντολή), επιλέγει μία στην τύχη

Ασκήσεις

- **ΑΣΚΗΣΗ 4 -- Υπηρεσία απομακρυσμένης εκτέλεσης ενός συνόλου εντολών**
  - **SERVER**
    - Ο server εκτελείται σε κάποιον Η/Υ με command line argument την IP διεύθυνση του Η/Υ στον οποίο τρέχει ο binder
    - Α) Κατά την εκκίνησή του, ο server Si (1<= i <= n) επιλέγει ένα ελεύθερο port (SERVER_PORT_Si) και δημιουργεί ένα νέο UDP socket
    - Β) Κατόπιν μέσω του socket, στέλνει ένα μήνυμα REGISTER COMMAND_1 COMMAND_2, COMMAND_k με τα ονόματα όλων των εντολών που μπορεί να εκτελέσει στον binder
      - Το port του binder είναι γνωστό (BINDER_PORT = 4500)
    - Γ) Στη συνέχεια μέσω του ίδιου socket, ο server περιμένει μηνύματα (στο SERVER_PORT_Si)
      - Κάθε μήνυμα που λαμβάνει ο server Si από κάποιον client Cj περιέχει σαν data ένα string με το όνομα της εντολής που θέλει να εκτελέσει ο client μαζί με τα ορίσματά της COMMAND_j argument1 argument2 .. argumentN
    - Δ) Ο server εκτελεί την εντολή και μετά επιστρέφει με ένα μήνυμα (UDP - datagram) στον client το αποτέλεσμα (string COMMAND_j result1 result2 resultN που περιέχει το COMMAND_j και ένα πλήθος ακεραίων
    - Το IP και port του client περιέχονται στο datagram που έλαβε ο server

Ασκήσεις

- **ΑΣΚΗΣΗ 4 -- Υπηρεσία απομακρυσμένης εκτέλεσης ενός συνόλου εντολών**
  - **CLIENT**
    - Ο client εκτελείται σε κάποιον Η/Υ με command line argument την IP διεύθυνση του Η/Υ στον οποίο τρέχει ο binder
    - Α) Ο client ξεκινώντας, δημιουργεί ένα νέο socket σε κάποιο ελεύθερο port
    - Β) Ο χρήστης εισάγει στο command line, το όνομα μίας εντολής με τα ορίσματά της (πχ SORT 8 1 4 0)
    - Γ) Ο client στέλνει μέσω του socket ένα μήνυμα στον binder (γνωστή IP και BINDER_PORT) για να βρει κάποιον server που υλοποιεί την εντολή αυτή και περιμένει την απάντηση (πχ LOOKUP SORT)
    - Δ) Μόλις λάβει το μήνυμα απάντησης, ο client στέλνει μήνυμα (datagram) απευθείας στον server, στο IP και port που του έδωσε ο binder, και περιμένει απάντηση
    - Το μήνυμα περιέχει σε ένα string την εντολή και τα ορίσματα που έδωσε ο χρήστης (πχ SORT 8 1 4 0)
    - Ε) Μόλις ο client λάβει την απάντηση, τυπώνει το αποτέλεσμα στην οθόνη και περιμένει νέα εντολή από τον χρήστη (βήμα Β)

Ασκήσεις

- **ΑΣΚΗΣΗ 4 -- Chat**
  - Κάνετε τις αναγκαίες προσθήκες στον κώδικα που σας έχει δοθεί ώστε στο τέλος το πρόγραμμα να πληρεί τις παρακάτω προδιαγραφές:
    - Η κατανεμημένη εφαρμογή που σας ζητείται να φτιάξετε αποτελείται από δύο μέρη
    - Το πρώτο μέρος είναι το πρόγραμμα-πελάτης (client) και το δεύτερο είναι ο εξυπηρετητής (server)
    - Ο εξυπηρετητής αναλαμβάνει να συγχρονίσει την επικοινωνία μεταξύ των προγραμμάτων-πελατών
    - Το πρόγραμμα-πελάτης είναι το κομμάτι του συστήματος που αλληλεπιδρά με τον χρήστη και αναλαμβάνει να δέχεται μηνύματα από τον χρήστη προς αποστολή και από τον εξυπηρετητή προς εμφάνιση στους υπόλοιπους χρήστες
    - Η καρδιά ενός κατανεμημένου συστήματος είναι το πρωτόκολλο επικοινωνίας μεταξύ των διαφορετικών κομματιών από τα οποία αποτελείται

Ασκήσεις

- **ΑΣΚΗΣΗ 4 -- Chat**
  - Το πρωτόκολλο αυτής της εφαρμογής είναι το ακόλουθο:
    - Μηνύματα πελάτη προς εξυπηρετητή:
    - “signin <string>”
      - Πελάτης: Το στέλνει για να γνωστοποιήσει στον εξυπηρετητή την ύπαρξή του και να δηλώσει την επιθυμία του να συμμετέχει στην συζήτηση, δηλαδή να δέχεται μηνύματα των άλλων πελατών
      - Το string είναι το όνομα με το οποίο θα είναι γνωστός ο συγκεκριμένος πελάτης
      - Την διευθυνση IP και την θύρα (port) του εξυπηρετητή την δίνει ο χρήστης όταν ξεκινάει τον πελάτη (είτε απο ορίσματα στην γραμμή εντολών είτε σαν είσοδο καθώς εκτελείται το πρόγραμμα-πελάτης)
      - Εξυπηρετητής: Προσθέτει τον πελάτη σε μια λίστα ενεργών πελατών
      - Αν λάβει μήνυμα απο πελάτη που δεν έχει προσθέσει στην λίστα και δεν είναι “signin”, το αγνοεί
      - Ο εξυπηρετητής δέχεται σαν όρισμα όπως και ο πελάτης, την θύρα που θα χρησιμοποιήσει

Ασκήσεις

- **ΑΣΚΗΣΗ 4 -- Chat**
  - Το πρωτόκολλο αυτής της εφαρμογής είναι το ακόλουθο:
    - Μηνύματα πελάτη προς εξυπηρετητή:
    - “signout <string>”
      - Ο πελάτης ενημερώνει τον εξυπηρετητή ότι επιθυμεί να βγει από το σύστημα chat
      - Το string είναι το όνομα του πελάτη
    - “who <string>”
      - Ο πελάτης ζητάει να μάθει από τον εξυπηρετητή ποιοι άλλοι πελάτες είναι εγγεγραμμένοι στο σύστημα chat
      - Το string είναι το όνομα του πελάτη.
    - “chat <string>”
      - Πελάτης: Το στέλνει όταν ο χρήστης εισαγάγει χαρακτήρες από το πληκτρολόγιο και πατήσει enter
      - Εξυπηρετητής: Διατρέχει την λίστα των ενεργών πελατών και σε κάθε ένα από αυτούς στέλνει το μύνημα “Msg <όνομα του πελάτη> says: <string>”, όπως περιγράφεται παρακάτω

Ασκήσεις

- **ΑΣΚΗΣΗ 4 -- Chat**
  - Η λίστα των ενεργών πελατών θα αποτελείται από κόμβους που θα περιέχουν δύο πεδία τουλάχιστον
    - Το όνομα του πελάτη και το socket μεσω του οποίου επικοινωνεί με αυτόν τον πελάτη
    - Ο εξυπηρετητής, μέσω επανάληψης, θα ελέγχει αρχικά το socket που έχει επικοινωνία με πιθανούς καινούργιους πελάτες και μετά ελέγχει τα sockets στην λίστα, ένα πρός ένα
    - Σε κάθε ένα socket που θα βρεί μύνημα, το χειρίζεται καταλλήλως, σύμφωνα με το παραπάνω πρωτόκολλο
  - Ο πελάτης θα πρέπει να αποτελείται από δύο νήματα (threads)
    - Το πρώτο περιμένει είσοδο του χρήστη από το πληκτρολόγιο και το δεύτερο περιμένει μήνυμα από τον εξυπηρετητή για να επεξεργαστεί.περιγράφεται παρακάτω
  - Προσοχή!
    - Άν κάποια αποστολή μηνύματος προς κάποιον πελάτη αποτύχει για οποιδήποτε λόγο, ο πελάτης αυτός πρέπει να αφαιρεθεί από την λίστα των ενεργών πελατών
    - Προσέξτε προβλήματα που μπορεί να εμφανιστούν όταν αφαιρείτε κόμβους από μια λίστα την οποία εκείνη την στιγμή διατρέχετε

Ασκήσεις

- **ΑΣΚΗΣΗ 4 -- Chat**
  - Ο server πλέον θα πρέπει όχι μόνο να αποθηκεύει τα ονόματα των clients που έχουν εγγραφεί (όπως στον Registration Server) αλλά και τις διευθύνσεις τους έτσι ώστε να μπορούν να χρησιμοποιηθούν στο μέλλον
  - Η διεύθυνση του client υπάρχει στο receive στον κώδικα του server
  - Η δομή που θα χρησιμοποιήσετε θα περιλαμβάνει λοιπόν δύο στοιχεία για κάθε client, το όνομα και την διεύθυνσή του
  - Στο τέλος ο server θα κάνει ένα loop στη δομή και θα παίρνει μία μία τις διευθύνσεις, θα τη βάζει στο send, μαζί με το μήνυμα, και θα το στέλνει στον κάθε client ξεχωριστά

netstat

- **netstat** (**net**work **stat**istics)
  - A command-line tool that displays network connections (both incoming and outgoing), routing tables, and a number of network interface statistics
  - It is available on Unix, Unix-like, and Windows NT-based operating systems
  - It is used for finding problems in the network and to determine the amount of traffic on the network as a performance measurement
  - http://ibgwww.colorado.edu/~lessem/psyc5112/usail/man/linux/netstat.8.html

netstat

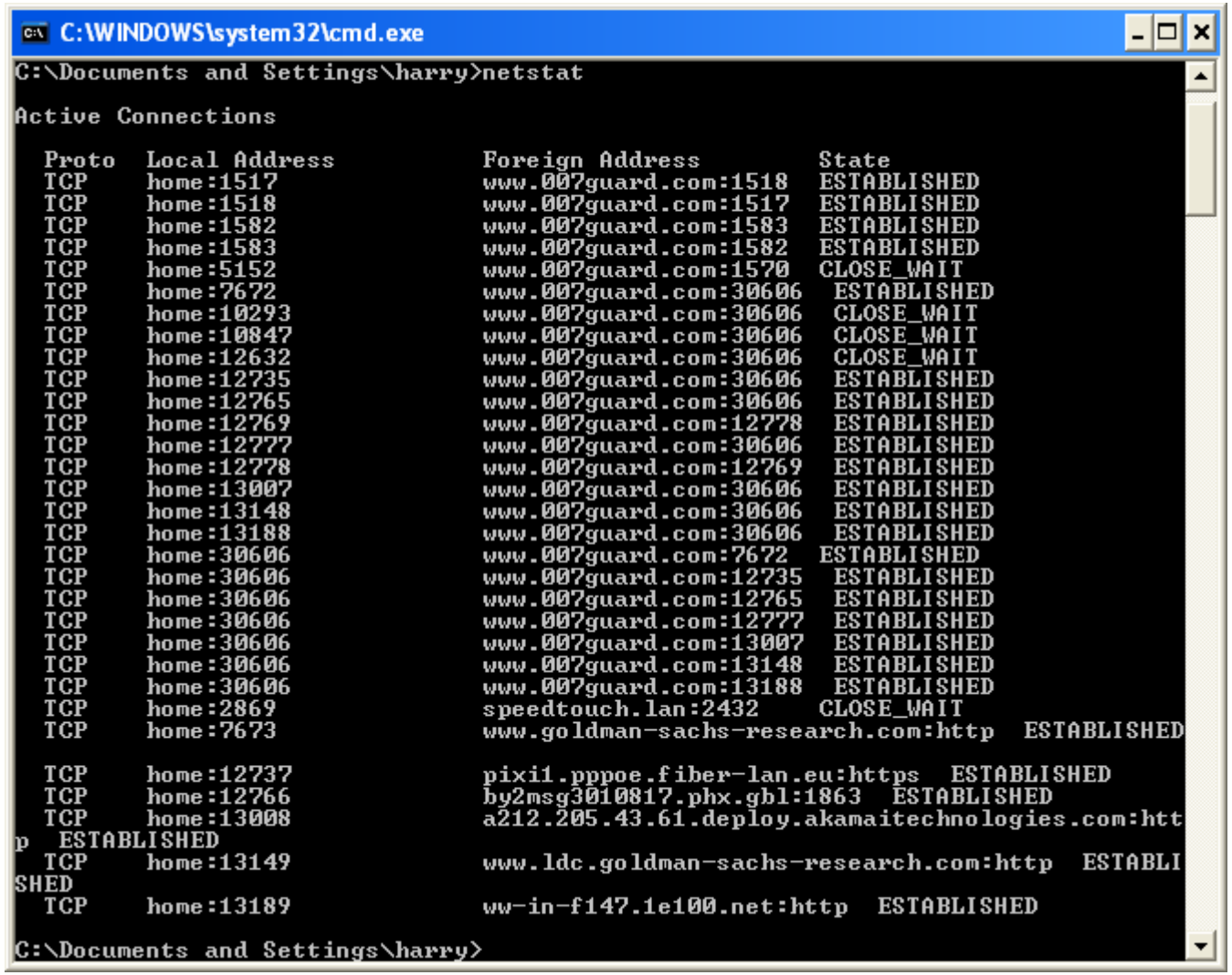
- Computer processes that provide application services are called servers
  - They create sockets on start up that are in *listening state*
  - These sockets are waiting for initiatives from client programs
  - For a listening TCP socket, the remote address presented by netstat may be denoted 0.0.0.0 and the remote port number 0

- A TCP server may serve several clients concurrently
  - By creating a child process for each client and establishing a TCP connection between the child process and the client
  - Unique *dedicated sockets* are created for each connection
  - These are in *established* state, when a socket-to-socket virtual connection or virtual circuit (VC), also known as a TCP session, is established with the remote socket, providing a duplex byte stream
  - Other possible TCP socket states presented by the netstat command are Syn-sent, Syn-Recv, Fin-wait1, Fin-wait2, Time-wait, Close-wait and Closed which relate to various start up and shutdown steps

netstat

- A UDP socket cannot be in an established state, since UDP is connectionless
  - Therefore, netstat does not show the state of a UDP socket
  - A UDP server does not create new child processes for every concurrently served client, but the same process handles incoming data packets from all remote clients sequentially through the same socket
  - This implies that UDP sockets are not identified by the remote address, but only by the local address, although each message has an associated remote address

netstat


C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\harry>netstat
Active Connections
Proto Local Address Foreign Address State
TCP home:1517 www.007guard.com:1518 ESTABLISHED
TCP home:1518 www.007guard.com:1517 ESTABLISHED
TCP home:1582 www.007guard.com:1583 ESTABLISHED
TCP home:1583 www.007guard.com:1582 ESTABLISHED
TCP home:5152 www.007guard.com:1570 CLOSE_WAIT
TCP home:7672 www.007guard.com:30606 ESTABLISHED
TCP home:10293 www.007guard.com:30606 CLOSE_WAIT
TCP home:10847 www.007guard.com:30606 CLOSE_WAIT
TCP home:12632 www.007guard.com:30606 CLOSE_WAIT
TCP home:12735 www.007guard.com:30606 ESTABLISHED
TCP home:12765 www.007guard.com:30606 ESTABLISHED
TCP home:12769 www.007guard.com:12778 ESTABLISHED
TCP home:12777 www.007guard.com:30606 ESTABLISHED
TCP home:12778 www.007guard.com:12769 ESTABLISHED
TCP home:13007 www.007guard.com:30606 ESTABLISHED
TCP home:13148 www.007guard.com:30606 ESTABLISHED
TCP home:13188 www.007guard.com:30606 ESTABLISHED
TCP home:30606 www.007guard.com:7672 ESTABLISHED
TCP home:30606 www.007guard.com:12735 ESTABLISHED
TCP home:30606 www.007guard.com:12765 ESTABLISHED
TCP home:30606 www.007guard.com:12777 ESTABLISHED
TCP home:30606 www.007guard.com:13007 ESTABLISHED
TCP home:30606 www.007guard.com:13148 ESTABLISHED
TCP home:30606 www.007guard.com:13188 ESTABLISHED
TCP home:2869 speedtouch.lan:2432 CLOSE_WAIT
TCP home:7673 www.goldman-sachs-research.com:http ESTABLISHED
TCP home:12737 pixi1.pppoe.fiber-lan.eu:https ESTABLISHED
TCP home:12766 by2msg3010817.phx.gbl:1863 ESTABLISHED
TCP home:13008 a212.205.43.61.deploy.akamaitechnologies.com:http ESTABLISHED
TCP home:13149 www.ldc.goldman-sachs-research.com:http ESTABLISHED
TCP home:13189 ww-in-f147.1e100.net:http ESTABLISHED
C:\Documents and Settings\harry>