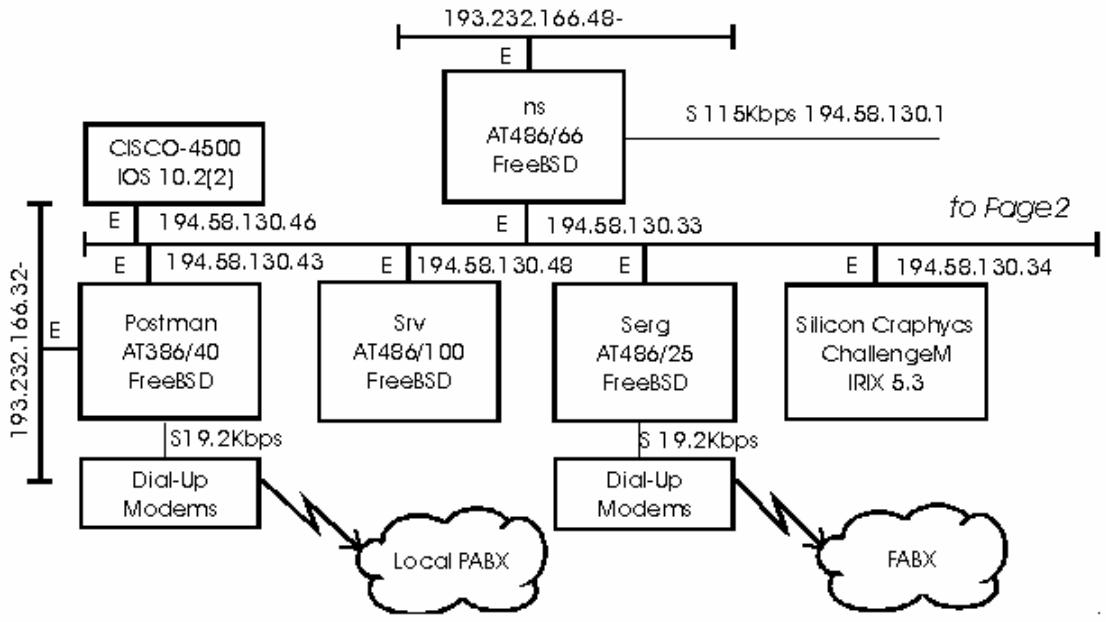
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 1999-2000

ΜΑΘΗΜΑ : ΔΙΚΤΥΑ ΥΠΟΛΟΓΙΣΤΩΝ
ΔΙΔΑΣΚΩΝ : Θ. ΑΠΟΣΤΟΛΟΠΟΥΛΟΣ

ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ CLIENT-SERVER(πελάτη-εξυπηρετητή) ΜΕ ΧΡΗΣΗ ΤΩΝ BERKELEY SOCKETS



ΟΜΑΔΑ ΕΡΓΑΣΙΑΣ(32) :
ΟΙΚΟΝΟΜΙΔΗΣ ΔΗΜΗΤΡΙΟΣ 3960070
ΠΑΠΑΪΩΑΝΝΟΥ ΜΙΧΑΛΗΣ 3960004

ΜΑΪΟΣ 2000

ΠΕΡΙΕΧΟΜΕΝΑ ***σελ***

Άσκηση 34

Γράψτε κώδικα, σε γλώσσα C, ενός προγράμματος εξυπηρετητή (server) και του αντίστοιχου προγράμματος πελάτη (client), με τη χρήση των Berkeley sockets, που θα κάνουν τα εξής:

1. Ο πελάτης θα παίρνει από το χρήστη το όνομα ενός αρχείου με ακεραίους και θα το στέλνει στον εξυπηρετητή.
2. Ο εξυπηρετητής όταν πάρει το όνομα του αρχείου, θα το ανοίγει, θα διαβάζει τους ακεραίους που περιέχει το αρχείο, θα τους ταξινομεί σε αύξουσα σειρά και θα τους γράφει ταξινομημένους σε ένα νέο αρχείο. Στη συνέχει θα στέλνει το όνομα του νέου αρχείου στον πελάτη.
3. Ο πελάτης θα εμφανίζει στο χρήστη το όνομα και το περιεχόμενο ενός νέου αρχείου που του στέλνει ο εξυπηρετητής.

Η επικοινωνία μεταξύ εξυπηρετητή και πελάτη θα γίνεται με βάση το πρωτόκολλο UDP. Ο εξυπηρετητής θα είναι επαναληπτικός (iterative).

Ο εξυπηρετητής θα εκτελείται στο παρασκήνιο και θα παίρνει ως πρώτη παράμετρο τον αριθμό του port στο οποίο θα δέχεται τις αιτήσεις των πελατών. Για παράδειγμα:

```
sort_server 6050 &
```

Στη γραμμή εντολών του πελάτη θα δίνεται ως πρώτη παράμετρο το όνομα ή η IP διεύθυνση της μηχανής που εκτελείται ο εξυπηρετητής και ο αριθμός του port στο οποίο δέχεται τις αιτήσεις. Για παράδειγμα:

```
sort_client dias.aueb.gr 6050
```

Α. Περιγραφή του αντικειμένου της εργασίας

Η εργασία αυτή εκπονήθηκε στα πλαίσια του μαθήματος «Δίκτυα Υπολογιστών» του Στ' Εξαμήνου του τμήματος Πληροφορικής του Ο.Π.Α. με διδάσκοντα καθηγητή τον κ. Θ. Αποστολόπουλο.
Αντικείμενο της εργασίας είναι η υλοποίηση σε γλώσσα C μιας εφαρμογής ταξινόμησης ακεραίων αριθμών σε αρχιτεκτονική πελάτη – εξυπηρετητή (*client-server*) με την χρήση των *Berkeley sockets*. Η εφαρμογή αυτή αποτελείται από το πρόγραμμα του πελάτη (client) sort_client.c, το πρόγραμμα του εξυπηρετητή (server) sort_server.c και ένα αρχείο δηλώσεων, το inet.h. Η επικοινωνία μεταξύ του εξυπηρετητή και του πελάτη γίνεται με βάση το πρωτόκολλο **UDP**. Ο εξυπηρετητής είναι επαναληπτικός (*iterative*), δηλαδή διαχειρίζεται την αίτηση του πελάτη, στέλνει την απάντηση και τέλος κλείνει την σύνδεση με τον πελάτη.

Β. Βήματα λειτουργίας της εφαρμογής

1. Ο πελάτης παίρνει από το χρήστη μέσω του *stdin* το όνομα του αρχείου με τους ακεραίους αριθμούς προς ταξινόμηση και το στέλνει στον εξυπηρετητή.
2. Ο εξυπηρετητής λαμβάνει το όνομα του αρχείου από τον πελάτη, το ανοίγει, διαβάζει τους ακεραίους που περιέχει το αρχείο, τους ταξινομεί με την μέθοδο *quicksort* σε αύξουσα σειρά και τους γράφει ταξινομημένους σε ένα νέο αρχείο του οποίου το όνομα έχει το μορφότυπο (*format*) sort_file_[όνομα αταξινόμητου αρχείου]. Στη συνέχεια στέλνει το όνομα του νέου αρχείου στον πελάτη.
3. Ο πελάτης λαμβάνει από τον εξυπηρετητή το όνομα του νέου αρχείου και εμφανίζει στον χρήστη μέσω του stdout το όνομα και το περιέχομενο του νέου ταξινομημένου αρχείου.

Γ. Τρόπος εκτέλεσης της εφαρμογής

Ο εξυπηρετητής εκτελείται στο παρασκήνιο και λαμβάνει ως πρώτη παράμετρο τον αριθμό του port στο οποίο δέχεται τις αιτήσεις των πελατών. Για παράδειγμα:

```
sort_server 6050 &
```

Στη γραμμή εντολών της μηχανής του πελάτη δίνεται ως πρώτη παράμετρος το όνομα ή η IP διεύθυνση της μηχανής που εκτελείται ο εξυπηρετητής και ο αριθμός του port στο οποίο αυτός δέχεται τις αιτήσεις. Για παράδειγμα:

```
sort_client dias.aueb.gr 6050
ή  sort_client 195.251.250.242 6050
```

Δ. Listing του κώδικα

αρχείο inet.h

```
/* Definitions for UDP client-server sorting programs */

#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#define MAXLINE 80
#define MAXVECTOR 1000
```

αρχείο sort_server.c

```
/*********************************************************
***************
 * Program name: sort_server.c
*
 * Programmers : Dimitris Oikonomidis (e-mail:
p3960070@dias.aueb.gr)  *
 *               Mixalis Papaiwannou  (e-mail:
p3960004@dias.aueb.gr)  *
 * This program is the iterative server using UDP
protocol. Its task   *
 * is to sort the contents of a text-file containing
integer numbers,  *
 * whose name receives from the client, and creates a new
sorted file  *
 * of integer, whose name sends to the client.
                *
 * It is invoked from the command line with :
sort_server [port] &    *
 * It consists of the main function, which calls the
function dg_sort. *
 * It includes the header file inet.
*

*********************************************************
*************/

#include "inet.h"
```

```
dg_sort(int sockfd, struct sockaddr *pcli_addr, int
maxclilen);

main (int argc, char *argv[])
{
int      sockfd;
u_short  port;
struct sockaddr_in serv_addr, cli_addr;

port=atoi(argv[1]);

/*  Open a UDP socket  */
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0))<0)
        { printf("Server:can't open datagram socket\n");
          exit(1); }

/*  Bind local address  */
bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family      = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port        = htons(port);
if (bind(sockfd, (struct sockaddr *) &serv_addr,
sizeof(serv_addr))<0)
        { printf("Server:can't bind local address\n");
          exit(1); }

dg_sort(sockfd, (struct sockaddr *) &cli_addr,
sizeof(cli_addr));

close(sockfd);
}


/* Function name: dg_sort
         *
 * Receives the filename from client and reads the
 contents of the    *
 * textfile. It converts the ASCII characters to integers
 with which  *
 * it fills a vector of integer. It sorts the vector
 using the        *
 * quicksort method and writes the numbers in a new file
 of integers  *
 * and sends the name of the new file to the client.
              *
 * It calls the qsort function.
         */

dg_sort(int sockfd, struct sockaddr *pcli_addr, int
maxclilen)
{
```

```
int   clilen, length, i;
char  oldfilename[MAXLINE], newfilename[MAXLINE],
num[MAXLINE];
FILE  *oldfp, *newfp;
int   vector[1000];

void  qsort(int v[], int left, int right);

for (;;) {
  clilen=maxclilen;

  /*  Read from the socket the name of the file to be
sorted  */
  if ((length=recvfrom(sockfd, oldfilename, MAXLINE, 0,
pcli_addr, &clilen))<0)
        { printf("Server:recvfrom error\n");
          exit(1); }

  /*  Open the unsorted file for reading  */
  if ((oldfp=fopen(oldfilename, "r"))==NULL)
        { printf("Server:error opening file\n");
          exit(1); }

  for (i=0;i<1000;i++) vector[i]='\0';
  i=0;

  /*  Read numbers from file and fill the vector of
integers  */
  for (;;){
        fgets(num, MAXLINE, oldfp);
        if (feof(oldfp)) break;
     vector[i]=atoi(num);
        printf("%d\n",vector[i]);
        i++;
        }
  fclose(oldfp);

  /*  Sort vector of integers  */
  qsort(vector, 0, i-1);

  /*  Create the name of the new file  */
  strcpy(newfilename, "sort_file_");
  strcat(newfilename, oldfilename);

  /*  Open new file for writing  */
  if ((newfp=fopen(newfilename,"w"))==NULL)
        { printf("Server:error opening new file\n");
          exit(1); }
  /*  Write the sorted vector of integers to the file  */
  for (i=0; vector[i]!='\0'; i++)
     fwrite(&vector[i], sizeof(int), 1, newfp);
```

```
  /*  Write the name of the new sorted file to the socket
*/
  length=strlen(newfilename);
  if (sendto(sockfd,newfilename,length,0,pcli_addr,
clilen)!=length)
        { printf("Server:sendto error\n");
          exit(1); }
  fclose(newfp);
  } /* end for */
}


/* Function name : qsort
*
 * It implements the quicksort sorting method by sorting
 *
 * v[left]...v[right] ascending.
 *
 * It calls the swap function.
*/

void qsort(int v[], int left, int right)
{
 int  i, last;
 void swap(int v[], int i, int j);

 if (left >= right) return;
 swap(v, left, (left+right)/2);
 last=left;
 for (i=left+1; i<=right; i++)
     if (v[i]<v[left])
        swap(v, ++last, i);
 swap(v, left, last);
 qsort(v, left, last-1);
 qsort(v, last+1, right);
}


/* swap function : swaps v[i], v[j] */
void swap(int v[], int i, int j)
{
   int temp;

   temp=v[i];
   v[i]=v[j];
   v[j]=temp;
}
```

αρχείο sort_client.c

```
/*********************************************************
***************
 * Program name: sort_client.c
          *
 * Programmers : Dimitris Oikonomidis (e-mail:
p3960070@dias.aueb.gr)  *
 *               Mixalis Papaiwannoy  (e-mail:
p3960004@dias.aueb.gr)  *
 * This program is the client using UDP protocol. Its
task is to read  *
 * the name of the unsorted file and send it to the
server and print to*
 * standard output the content of the sorted file, whose
name receives *
 * from server.
*
 * It is invoked from the command line with:
*
 * sort_client [machine IP address/name] [port]
                *
 * It consists of the main function, which calls the
function dg_cli.  *
 * It includes the header file inet.h.
*

*********************************************************
*************/

#include "inet.h"

void dg_cli(int sockfd, struct sockaddr *pserv_addr, int
servlen);

main(int argc, char *argv[])
{
int     sockfd;
u_short port;
struct sockaddr_in  serv_addr,cli_addr;
struct hostent  *hptr;        /*  result of host name
lookup  */

if (argc!=3)
        { printf("Usage: sort_client [machine IP
address/name] [port]\n");
          exit(1); }
```

```
port=atoi(argv[2]);

/*  Fill in the "serv_addr" structure with hte address of
the server  */
bzero((char*) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
/*  Check whether the name or the IP address of the
server machine is
 *  given  */
if ((hptr=gethostbyname(argv[1]))!=NULL)
     bcopy((char *)hptr->h_addr, (char
*)&serv_addr.sin_addr, hptr->h_length);
else serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
serv_addr.sin_port = htons(port);

/*  Open a UDP socket  */
if ((sockfd=socket(AF_INET, SOCK_DGRAM, 0))<0)
        { printf("Client:can't open datagram socket\n");
          exit(1); }

/*  Bind any local address  */
bzero((char *) &cli_addr,sizeof(cli_addr));
cli_addr.sin_family      = AF_INET;
cli_addr.sin_addr.s_addr = htonl(INADDR_ANY);
cli_addr.sin_port        = htons(0);
if (bind(sockfd, (struct sockaddr *) &cli_addr,
sizeof(cli_addr))<0)
        { printf("Client:can't bind local address\n");
          exit(1);}

dg_cli(sockfd, (struct sockaddr *) &serv_addr,
sizeof(serv_addr));

close(sockfd);
exit(0);
}


/* Function name: dg_cli
*
 * It reads from input name of file to be sorted and
prints the name*
 * and content of new sorted file
*/

void dg_cli(int sockfd, struct sockaddr *pserv_addr, int
servlen)
{
int  length, num;
char filename[MAXLINE];
FILE *fp;
```

```
/*  Read from stdin the name of the file of integers to
be sorted  */
printf("Give full-path of integer-file to be sorted:");
gets(filename);
length=strlen(filename);
/*  Write the name of the file to the socket  */
if (sendto(sockfd, filename, length, 0, pserv_addr,
servlen)!=length)
        { printf("client:sendto error on socket\n");
          exit(1); }

/*  Read the name of the new sorted file from the socket
*/
if ((length=recvfrom(sockfd, filename, MAXLINE, 0,
(struct sockaddr *) 0, (int *) 0))<0)
        { printf("Client:recvfrom error\n");
          exit(1); }

/*  Open the new sorted file for reading  */
if ((fp=fopen(filename,"r"))==NULL)
        { printf("Client:error opening file\n");
          exit(1); }

/*  Print to standard output the content of the file  */
printf("New sorted file is: %s\n",filename);
printf("File content:\n");
for(;;) {
     fread(&num, sizeof(int), 1, fp);
     if (feof(fp)) break;
     printf("%d\n",num);
     }
if (ferror(fp))
        { printf("Client:error reading file\n");
          exit(1); }
fclose(fp);
}
```

Ε. ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

- **Δοκιμαστικά Δεδομένα**

Περιεχόμενα του αρχείου test1:

```
37
52
25
1
61
66
16
72
71
42
```

- **Αποτελέσματα εκτέλεσης – ταξινόμηση των περιεχόμενων ακεραίων του αρχείου test1 σε αύξουσα σειρά (όπως εμφανίζονται στο stdout):**

```
1
16
25
37
42
52
61
66
71
72
```

Screenshot εκτέλεσης του προγράμματος με δοκιμαστικά δεδομένα