Fitxategi transferentziak egiteko aplikazioaren inplementazioa

“bez_fitx.h” fitxategia:

```
#define MAX_BUF 1024
#define SERVER "localhost"
#define PORT 6012

#define COM_USER  0
#define COM_PASS  1
#define COM_LIST  2
#define COM_DOWN  3
#define COM_DOW2  4
#define COM_UPLO  5
#define COM_UPL2  6
#define COM_DELE  7
#define COM_EXIT  8

#define OP_LIST   1
#define OP_DOWN   2
#define OP_UP           3
#define OP_DEL    4
#define OP_EXIT   5


char * KOMANDOAK[] =
{"USER","PASS","LIST","DOWN","DOW2","UPLO","UPL2","DELE","EXIT",NULL};
char * ER_MEZUAK[] =
{
      "Dena ondo. Errorerik ez.\n",
      "Komando ezezaguna edo ustegabekoa.\n",
      "Erabiltzaile ezezaguna.\n",
      "Pasahitz okerra.\n",
      "Arazoa fitxategi zerrenda sortzen.\n",
      "Fitxategia ez da existizen.\n",
      "Arazoa fitxategia jeistean.\n",
      "Erabiltzaile anonimoak ez dauka honetarako baimenik.\n",
      "Fitxategiaren tamaina haundiegia da.\n",
      "Arazoa fitxategia igotzeko prestatzean.\n",
      "Arazoa fitxategia igotzean.\n",
      "Arazoa fitxategia ezabatzean.\n"
};

int parse(char *status);
int readline(int stream, char *buf, int tam);
int menua();
```

“bez_fitx.c” fitxategia:

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

```
#include "bez_fitx.h"

int main(int argc, char *argv[])
{
	char buf[MAX_BUF], param[MAX_BUF];
	char zerbitzaria[MAX_BUF];
	int portua = PORT;

	int sock, n, status, aukera;
	long fitx_tamaina, irakurrita;
	struct stat file_info;
	FILE *fp;
	struct sockaddr_in zerb_helb;
	struct hostent *hp;

	// Parametroak prozesatu.
	switch(argc)
	{
		case 1:
			strcpy(zerbitzaria, SERVER);
			break;
		case 3:
			portua = atoi(argv[2]);
		case 2:
			strcpy(zerbitzaria, argv[1]);
			break;
		default:
			printf("Erabilera: %s <zerbitzaria> <portua>\n", argv[0]);
			exit(1);
	}

	// Socketa sortu.
	if((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
	{
		perror("Errorea socketa sortzean");
		exit(1);
	}

	// Zerbitzariko socketaren helbidea sortu.
	zerb_helb.sin_family = AF_INET;
	zerb_helb.sin_port = htons(portua);
	if((hp = gethostbyname(zerbitzaria)) == NULL)
	{
		herror("Errorea zerbitzariaren izena ebaztean");
		exit(1);
	}
	memcpy(&zerb_helb.sin_addr, hp->h_addr, hp->h_length);

	// Konektatu zerbitzariarekin.
	if(connect(sock, (struct sockaddr *) &zerb_helb, sizeof(zerb_helb)) < 0)
	{
		perror("Errorea zerbitzariarekin konektatzean");
		exit(1);
	}

	// Erabiltzaile eta pasahitza bidali.
	int i=0;
	do
	{
		printf("Erabiltzaile izena: ");
		fgets(param,MAX_BUF,stdin);
```

```
            param[strlen(param)-1] = 0;
            sprintf(buf,"%s%s\r\n",KOMANDOAK[COM_USER],param);

            write(sock, buf, strlen(buf));
            readline(sock, buf, MAX_BUF);
            status = parse(buf);
            if(status != 0)
            {
                  fprintf(stderr,"Errorea: ");
                  fprintf(stderr,"%s",ER_MEZUAK[status]);
                  continue;
            }

            printf("Pasahitza: ");
            fgets(param,MAX_BUF,stdin);
            param[strlen(param)-1] = 0;
            sprintf(buf,"%s%s\r\n",KOMANDOAK[COM_PASS],param);

            write(sock, buf, strlen(buf));
            readline(sock, buf, MAX_BUF);
            status = parse(buf);
            if(status != 0)
            {
                  fprintf(stderr,"Errorea: ");
                  fprintf(stderr,"%s",ER_MEZUAK[status]);
                  continue;
            }
            break;
      } while(1);

      // Begizta bat aukeren menua erakutsi eta aukera prozesatzeko.
      do
      {
            aukera = menua();       // Erakutsi aukeren menua.
            switch(aukera)
            {
                  case OP_LIST:     // Fitxategi zerrenda eskatu eta
pantailaratu.
                        sprintf(buf,"%s\r\n",KOMANDOAK[COM_LIST]);
                        write(sock,buf,strlen(buf));                    // Bidali
komandoa.
                        n = readline(sock, buf, MAX_BUF);         // Erantzuna
jaso.
                        status = parse(buf);
                        if(status != 0)
                        {
                              fprintf(stderr,"Errorea: ");
                              fprintf(stderr,"%s",ER_MEZUAK[status]);
                        }
                        else
                        {
                              // Fitxategi zerrenda jaso eta pantailaratu lerroz
lerro.
                              int kop = 0;      // Zerrendako fitxategi kopurua
kontrolatzeko.
                              printf("Zerbitzaritik jasotako fitxategi
zerrenda:\n");
                              printf("----------------------------------------
\n");
                              n = readline(sock, buf, MAX_BUF);
                              while(n > 2)
```

```
                        {
                              buf[n-2] = 0;
                              printf("%s\t\t",strtok(buf,"?"));
                              fitx_tamaina = atol(strtok(NULL,"?"));
                              if(fitx_tamaina < 0)
                                    printf("Tamaina ezezaguna\n");
                              else
                              {
                                    if(fitx_tamaina < 1024)
                                          printf("% 5ld B\n",
fitx_tamaina);
                                    else if(fitx_tamaina < 1024*1024)
                                          printf("%5.1f KB\n",
fitx_tamaina/1024.0);
                                    else if(fitx_tamaina < 1024*1024*1024)
                                          printf("%5.1f MB\n",
fitx_tamaina/(1024.0*1024));
                                    else
                                          printf("%5.1f GB\n",
fitx_tamaina/(1024.0*1024*1024));
                              }
                              kop++;
                              n = readline(sock, buf, MAX_BUF);
                        }
                        if(kop > 0)
                        {
                              printf("-----------------------------------
------\n");
                              printf("Guztira %d fitxategi
eskuragarri.\n",kop);
                        }
                        else
                              printf("Ez dago fitxategirik
eskuragarri.\n");
                        printf("----------------------------------------
\n");
                  }
                  break;
            case OP_DOWN:           // Fitxategi bat jaitsi.
                  printf("Idatzi jaitsi nahi duzun fitxategiaren izena:
");
                  fgets(param,MAX_BUF,stdin);
                  param[strlen(param)-1] = 0;
                  sprintf(buf,"%s%s\r\n",KOMANDOAK[COM_DOWN], param);
                  write(sock,buf,strlen(buf));             // Eskaera
bidali.
                  n = readline(sock, buf, MAX_BUF);  // Erantzuna jaso.
                  status = parse(buf);
                  if(status != 0)
                  {
                        fprintf(stderr,"Errorea: ");
                        fprintf(stderr,"%s",ER_MEZUAK[status]);
                  }
                  else
                  {
                        buf[n-2] = 0; //EOL ezabatu.
                        fitx_tamaina = atol(buf+2);
                        sprintf(buf,"%s\r\n",KOMANDOAK[COM_DOW2]);
                        write(sock,buf,strlen(buf));  // Eskaera
konfirmatu.
                        n = readline(sock, buf, MAX_BUF);  // Erantzuna
```

```
jaso.
                              status = parse(buf);
                              if(status != 0)
                              {
                                    fprintf(stderr,"Errorea: ");
                                    fprintf(stderr,"%s",ER_MEZUAK[status]);
                              }
                              else
                              {
                                    irakurrita = 0;
                                    if((fp = fopen(param,"w")) == NULL)      //
Fitxategia sortu diskoan.
                                    {
                                          perror("Ezin da fitxategia disko
lokalean gorde");
                                          exit(1);
                                    }
                                    while(irakurrita < fitx_tamaina)   //
Fitxategia jaso eta diskoan gorde.
                                    {
                                          n = read(sock, buf, MAX_BUF);
                                          if(fwrite(buf, 1, n, fp) < 0)
                                          {
                                                perror("Arazoa fitxategia disko
lokalean gordetzean");
                                                exit(1);
                                          }
                                          irakurrita += n;
                                    }
                                    fclose(fp);
                                    printf("%s fitxategia jaso da.\n",param);
                              }
                        }
                        break;
                  case OP_UP:       // Fitxategi bat igo.
                        printf("Idatzi igo nahi duzun fitxategiaren izena: ");
                        fgets(param,MAX_BUF,stdin);
                        param[strlen(param)-1] = 0;
                        if(stat(param, &file_info) < 0)    // Fitxategiaren
tamaina lortu.
                        {
                              fprintf(stderr,"%s fitxategia ez da
aurkitu.\n",param);
                        }
                        else
                        {
                              sprintf(buf,"%s%s?%ld\r\n",KOMANDOAK[COM_UPLO],
param, file_info.st_size);
                              write(sock,buf,strlen(buf));                    //
Eskaera bidali.
                              n = readline(sock, buf, MAX_BUF);        //
Erantzuna jaso.
                              status = parse(buf);
                              if(status != 0)
                              {
                                    fprintf(stderr,"Errorea: ");
                                    fprintf(stderr,"%s",ER_MEZUAK[status]);
                              }
                              else
                              {
                                    if((fp = fopen(param,"r")) == NULL) //
```

```
Fitxategia ireki.
                                    {
                                          fprintf(stderr,"%s fitxategia ezin
izan da ireki.\n",param);
                                          exit(1);
                                    }
                                    sprintf(buf,"%s\r\n",KOMANDOAK[COM_UPL2]);
                                    write(sock,buf,strlen(buf));  // Bidalketa
konfirmatu.
                                    while((n=fread(buf,1,MAX_BUF,fp))==MAX_BUF)
      // Fitxategia bidali, tamaina maximoko blokeetan.
                                          write(sock,buf,MAX_BUF);
                                    if(ferror(fp)!=0)
                                    {
                                          fprintf(stderr,"Errorea gertatu da
fitxategia bidaltzean.\n");
                                          exit(1);
                                    }
                                    write(sock,buf,n);      // Fitxategiaren
azkeneko blokea bidali.

                                    n = readline(sock, buf, MAX_BUF);   //
Erantzuna jaso.
                                    status = parse(buf);
                                    if(status != 0)
                                    {
                                          fprintf(stderr,"Errorea: ");
                                          fprintf(stderr,"%s",ER_MEZUAK[status])
;
                                    }
                                    else
                                          printf("%s fitxategia igo da.\n",
param);
                              }
                        }
                        break;
                  case OP_DEL:            // Fitxategi bat ezabatu.
                        printf("Idatzi ezabatu nahi duzun fitxategiaren izena:
");
                        fgets(param,MAX_BUF,stdin);
                        param[strlen(param)-1] = 0;
                        sprintf(buf,"%s%s\r\n",KOMANDOAK[COM_DELE], param);
                        write(sock,buf,strlen(buf));                    //
Eskaera bidali.
                        n = readline(sock, buf, MAX_BUF);        // Erantzuna
jaso.
                        status = parse(buf);
                        if(status != 0)
                        {
                              fprintf(stderr,"Errorea: ");
                              fprintf(stderr,"%s",ER_MEZUAK[status]);
                        }
                        else
                        {
                              printf("%s fitxategia ezabatua izan da.\n",
param);
                        }
                        break;
                  case OP_EXIT:           // Sesioa amaitu.
                        sprintf(buf,"%s\r\n",KOMANDOAK[COM_EXIT]);
                        write(sock,buf,strlen(buf));                    //
```

```
Eskaera bidali.
                        n = readline(sock, buf, MAX_BUF);        // Erantzuna
jaso.
                        status = parse(buf);
                        if(status != 0)
                        {
                              fprintf(stderr,"Errorea: ");
                              fprintf(stderr,"%s",ER_MEZUAK[status]);
                        }
                        break;
            }
      } while(aukera != 5);

      close(sock);
}

/* Zerbitzaritik jasotako erantzuna aztertzen du.
* Jasotakoa OK bada 0 balioa itzuliko du eta bestela errorearen kode zenbakia.
*/
int parse(char *status)
{
      if(!strncmp(status,"OK",2))
            return 0;
      else if(!strncmp(status,"ER",2))
            return(atoi(status+2));
      else
      {
            fprintf(stderr,"Ustekabeko erantzuna.\n");
            exit(1);
      }
}

/*
* Telneteko lerro jauzi estandar bat ("\r\n") aurkitu arte datuak irakurtzen
ditu stream batetik.
* Erraztasunagatik lerro jauzia irakurketa bakoitzeko azken bi bytetan soilik
bilatzen da.
* Dena ondo joanez gero irakurritako karaktere kopurua itzuliko du.
* Fluxua amaituz gero ezer irakurri gabe 0 itzuliko du.
* Zerbait irakurri ondoren fluxua amaituz gero ("\r\n" aurkitu gabe) -1 itzuliko
du.
* 'tam' parametroan adierazitako karaktere kopurua irakurriz gero "\r\n" aurkitu
gabe -2 itzuliko du.
* Beste edozein error gertatu ezkero -3 itzuliko du.
*/
int readline(int stream, char *buf, int tam)
{
      /*
            Kontuz! Inplementazio hau sinplea da, baina ez da batere
eraginkorra.
      */
      char c;
      int guztira=0;
      int cr = 0;

      while(guztira<tam)
      {
            int n = read(stream, &c, 1);
            if(n == 0)
            {
                  if(guztira == 0)
```

```
                        return 0;
                  else
                        return -1;
            }
            if(n<0)
                  return -3;
            buf[guztira++]=c;
            if(cr && c=='\n')
                  return guztira;
            else if(c=='\r')
                  cr = 1;
            else
                  cr = 0;
      }
      return -2;
}

int menua()
{
      char katea[MAX_BUF];
      int aukera;

      printf("\n");
      printf("\t\t\t\t*********************************************\n");
      printf("\t\t\t\t*********************************************\n");
      printf("\t\t\t\t**                                         **\n");
      printf("\t\t\t\t**       1. Fitxategi zerrenda             **\n");
      printf("\t\t\t\t**       2. Fitxategia jeitsi              **\n");
      printf("\t\t\t\t**       3. Fitxategia igo                 **\n");
      printf("\t\t\t\t**       4. Fitxategia ezabatu             **\n");
      printf("\t\t\t\t**       5. Saioa amaitu                   **\n");
      printf("\t\t\t\t**                                         **\n");
      printf("\t\t\t\t*********************************************\n");
      printf("\t\t\t\t*********************************************\n");

      printf("\t\t\t\tEgin zure aukera: ");
      while(1)
      {
            fgets(katea,MAX_BUF,stdin);
            aukera = atoi(katea);
            if(aukera > 0 && aukera < 6)
                  break;
            printf("\t\t\t\tAukera okerra, saiatu berriro: ");
      }
      printf("\n");
      return aukera;
}
```

“zerb_fitx.h” **fitxategia:**

```
#define MAX_BUF 1024
#define PORT 6012
#define FILES_PATH     "files"

#define ST_INIT   0
#define ST_AUTH   1
#define ST_MAIN   2
#define ST_DOWN   3
#define ST_UP           4

#define COM_USER  0
```

```
#define COM_PASS  1
#define COM_LIST  2
#define COM_DOWN  3
#define COM_DOW2  4
#define COM_UPLO  5
#define COM_UPL2  6
#define COM_DELE  7
#define COM_EXIT  8

#define MAX_UPLOAD_SIZE 10*1024*1024      // 10 MB
#define SPACE_MARGIN          50*1024*1024      // 50 MB

char * KOMANDOAK[] =
{"USER","PASS","LIST","DOWN","DOW2","UPLO","UPL2","DELE","EXIT",NULL};
char * erab_zer[] = {"anonimous","sar","sza",NULL};
char * pass_zer[] = {"","sar","sza"};
int egoera;

void sesioa(int s);
int readline(int stream, char *buf, int tam);
int bilatu_string(char *string, char **string_zerr);
int bilatu_substring(char *string, char **string_zerr);
void ustegabekoa(int s);
int bidali_zerrenda(int s);
unsigned long toki_librea();
void sig_chld(int signal);
int ez_ezkutua(const struct dirent *entry);
```

“zerb_fitx.c” fitxategia:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <sys/wait.h>
#include <signal.h>
#include <sys/stat.h>
#include <dirent.h>
#include <sys/statvfs.h>

#include "zerb_fitx.h"

int main()
{
      int sock, elkarrizketa;
      struct sockaddr_in zerb_helb;
      socklen_t helb_tam;

      // Sortu socketa.
      if((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
      {
            perror("Errorea socketa sortzean");
            exit(1);
      }

      memset(&zerb_helb, 0, sizeof(zerb_helb));
      zerb_helb.sin_family = AF_INET;
```

```
	zerb_helb.sin_addr.s_addr = htonl(INADDR_ANY);
	zerb_helb.sin_port = htons(PORT);

	// Esleitu helbidea socketari.
	if(bind(sock, (struct sockaddr *) &zerb_helb, sizeof(zerb_helb)) < 0)
	{
		perror("Errorea socketari helbide bat esleitzean");
		exit(1);
	}

	// Ezarri socketa entzute socket gisa.
	if(listen(sock,5) < 0)
	{
		perror("Errorea socketa entzute socket bezala ezartzean");
		exit(1);
	}

	// Adierzi SIG_CHLD seinalea jasotzean ez dela ezer egingo. Modu honetan
amaitutako prozesu umeak ez dira zonbi egoeran geratuko.
	signal(SIGCHLD, SIG_IGN);

	while(1)
	{

		// Onartu konexio eskaera eta sortu elkarrizketa socketa.
		if((elkarrizketa = accept(sock, NULL, NULL)) < 0)
		{
			perror("Errorea konexioa onartzean");
			exit(1);
		}

		// Sortu prozesu ume bat bezeroarekin komunikatzeko.
		switch(fork())
		{
			case 0:
				close(sock);
				sesioa(elkarrizketa);
				close(elkarrizketa);
				exit(0);
			default:
				close(elkarrizketa);
		}
	}
}

/*
* Funtzio honetan kodetzen da bezeroarekin komunikatu behar den prozesu umeak
egin beharrekoa, aplikazio protokoloak zehaztu bezala.
* Parametro gisa elkarrizketa socketa pasa behar zaio.
*/
void sesioa(int s)
{
	char buf[MAX_BUF], file_path[MAX_BUF], file_name[MAX_BUF];
	int n, erabiltzaile, komando, error;
	FILE *fp;
	struct stat file_info;
	unsigned long file_size, irakurrita;
	char * sep;

	// Zehaztu uneko egoera bezala hasierako egoera.
	egoera = ST_INIT;
```

```
	while(1)
	{
		// Irakurri bezeroak bidalitako mezua.
		if((n=readline(s,buf,MAX_BUF)) <= 0)
			return;

		// Aztertu jasotako komandoa ezaguna den ala ez.
		if((komando=bilatu_substring(buf,KOMANDOAK)) < 0)
		{
			ustegabekoa(s);
			continue;
		}

		// Jasotako komandoaren arabera egin beharrekoa egin.
		switch(komando)
		{
			case COM_USER:
				if(egoera != ST_INIT)		// Egiaztatu esperotako
egoeran jaso dela komandoa.
				{
					ustegabekoa(s);
					continue;
				}
				buf[n-2] = 0;	// Lerro bukaera, edo "End Of Line"
(EOL), ezabatzen du.
				// Baliozko erabiltzaile bat den egiaztatu.
				if((erabiltzaile = bilatu_string(buf+4, erab_zer)) < 0)
				{
					write(s,"ER2\r\n",5);
				}
				else
				{
					write(s,"OK\r\n",4);
					egoera = ST_AUTH;
				}
				break;
			case COM_PASS:
				if(egoera != ST_AUTH)		// Egiaztatu esperotako
egoeran jaso dela komandoa.
				{
					ustegabekoa(s);
					continue;
				}
				buf[n-2] = 0;	// EOL ezabatu.
				// Pasahitza zuzena dela egiaztatu.
				if(erabiltzaile == 0 || !strcmp(pass_zer[erabiltzaile],
buf+4))
				{
					write(s,"OK\r\n",4);
					egoera = ST_MAIN;
				}
				else
				{
					write(s,"ER3\r\n",5);
					egoera = ST_INIT;
				}
				break;
			case COM_LIST:
				if(n>6 || egoera != ST_MAIN)  // Egiaztatu esperotako
egoeran jaso dela komandoa eta ez dela parametrorik jaso.
```

```
                        {
                              ustegabekoa(s);
                              continue;
                        }
                        if(bidali_zerrenda(s) < 0)
                              write(s,"ER4\r\n",5);
                        break;
                  case COM_DOWN:
                        if(egoera != ST_MAIN)        // Egiaztatu esperotako
egoeran jaso dela komandoa.
                        {
                              ustegabekoa(s);
                              continue;
                        }
                        buf[n-2] = 0; // EOL ezabatu.
                        sprintf(file_path,"%s/%s",FILES_PATH,buf+4);         //
Fitxategiak dauden karpeta eta fitxategiaren izena kateatu.
                        if(stat(file_path, &file_info) < 0)
                  // Lortu fitxategiari buruzko informazioa.
                        {
                              write(s,"ER5\r\n",5);
                        }
                        else
                        {
                              sprintf(buf,"OK%ld\r\n", file_info.st_size);
                              write(s, buf, strlen(buf));
                              egoera = ST_DOWN;
                        }
                        break;
                  case COM_DOW2:
                        if(n > 6 || egoera != ST_DOWN)           // Egiaztatu
esperotako egoeran jaso dela komandoa eta ez dela parametrorik jaso.
                        {
                              ustegabekoa(s);
                              continue;
                        }
                        egoera = ST_MAIN;
                        // Fitxategia ireki.
                        if((fp=fopen(file_path,"r")) == NULL)
                        {
                              write(s,"ER6\r\n",5);
                        }
                        else
                        {
                              // Fitxategiaren lehenengo zatia irakurri eta
errore bat gertatu bada bidali errore kodea.
                              if((n=fread(buf,1,MAX_BUF,fp))<MAX_BUF &&
ferror(fp) != 0)
                              {
                                    write(s,"ER6\r\n",5);
                              }
                              else
                              {
                                    write(s,"OK\r\n",4);
                                    // Bidali fitxategiaren zatiak.
                                    do
                                    {
                                          write(s,buf,n);
                                    } while((n=fread(buf,1,MAX_BUF,fp)) ==
MAX_BUF);
                                    if(ferror(fp) != 0)
```

```
                                    {
                                          close(s);
                                          return;
                                    }
                                    else if(n>0)
                                          write(s,buf,n);   // Bidali azkeneko
zatia.
                              }
                              fclose(fp);
                        }
                        break;
                  case COM_UPLO:
                        if(egoera != ST_MAIN)        // Egiaztatu esperotako
egoeran jaso dela komandoa.
                        {
                              ustegabekoa(s);
                              continue;
                        }
                        // Erabiltzaile anonimoak ez dauka ekintza honetarako
baimenik.
                        if(erabiltzaile==0)
                        {
                              write(s,"ER7\r\n",5);
                              continue;
                        }

                        buf[n-2] = 0;     // EOL kendu.
                        // Mezuak dauzkan bi zatiak (fitxategi izena eta
tamaina) erauzi.
                        if((sep = strchr(buf,'?')) == NULL)
                        {
                              ustegabekoa(s);
                              continue;
                        }
                        *sep = 0;
                        sep++;
                        strcpy(file_name,buf+4);      // Fitxategi izena lortu.
                        file_size = atoi(sep);        // Fitxategi tamaina lortu.
                        sprintf(file_path,"%s/%s",FILES_PATH,file_name);     //
Fitxategiak dauden karpeta eta fitxategiaren izena kateatu.
                        if(file_size > MAX_UPLOAD_SIZE)    // Fitxategi tamainak
maximoa gainditzen ez duela egiaztatu.
                        {
                              write(s,"ER8\r\n",5);
                              continue;
                        }
                        if(toki_librea() < file_size + SPACE_MARGIN)   //
Mantendu beti toki libre minimo bat diskoan.
                        {
                              write(s,"ER9\r\n",5);
                              continue;
                        }
                        write(s,"OK\r\n",4);
                        egoera = ST_UP;
                        break;
                  case COM_UPL2:
                        if(n > 6 || egoera != ST_UP)       // Egiaztatu
esperotako egoeran jaso dela komandoa eta ez dela parametrorik jaso.
                        {
                              ustegabekoa(s);
                              continue;
```

```
                        }
                        egoera = ST_MAIN;
                        irakurrita = 0L;
                        fp=fopen(file_path,"w");           // Sortu fitxategi
berria diskoan.
                        error = (fp == NULL);
                        while(irakurrita < file_size)
                        {
                              // Jaso fitxategi zati bat.
                              if((n=read(s,buf,MAX_BUF)) <= 0)
                              {
                                    close(s);
                                    return;
                              }
                              // Ez bada errorerik izan gorde fitxategi zatia
diskoan.
                              // Errorerik gertatuz gero segi fitxategi zatiak
jasotzen.
                              // Fitxategi osoa jasotzeak alferrikako trafikoa
sortzen du, baina aplikazio protokoloak ez du beste aukerarik ematen.
                              if(!error)
                              {
                                    if(fwrite(buf, 1, n, fp) < n)
                                    {
                                          fclose(fp);
                                          unlink(file_path);
                                          error = 1;
                                    }
                              }
                              irakurrita += n;
                        }
                        if(!error)
                        {
                              fclose(fp);
                              write(s,"OK\r\n",4);
                        }
                        else
                              write(s,"ER10\r\n",6);
                        break;
                  case COM_DELE:
                        if(egoera != ST_MAIN)        // Egiaztatu esperotako
egoeran jaso dela komandoa.
                        {
                              ustegabekoa(s);
                              continue;
                        }
                        // Erabiltzaile anonimoak ez dauka ekintza honetarako
baimenik.
                        if(erabiltzaile==0)
                        {
                              write(s,"ER7\r\n",5);
                              continue;
                        }
                        buf[n-2] = 0; // EOL ezabatu.
                        sprintf(file_path,"%s/%s",FILES_PATH,buf+4);   //
Fitxategiak dauden karpeta eta fitxategiaren izena kateatu.
                        if(unlink(file_path) < 0)          // Ezabatu
fitxategia.
                              write(s,"ER11\r\n",6);
                        else
                              write(s,"OK\r\n",4);
```

```
                        break;
                case COM_EXIT:
                        if(n > 6)   // Egiaztatu ez dela parametrorik jaso.
                        {
                                ustegabekoa(s);
                                continue;
                        }
                        write(s,"OK\r\n",4);
                        return;
                }
        }
}

/*
* Telneteko lerro jauzi estandar bat ("\r\n") aurkitu arte datuak irakurtzen
ditu stream batetik.
* Erraztasunagatik lerro jauzia irakurketa bakoitzeko azken bi bytetan soilik
bilatzen da.
* Dena ondo joanez gero irakurritako karaktere kopurua itzuliko du.
* Fluxua amaituz gero ezer irakurri gabe 0 itzuliko du.
* Zerbait irakurri ondoren fluxua amaituz gero ("\r\n" aurkitu gabe) -1 itzuliko
du.
* 'tam' parametroan adierazitako karaktere kopurua irakurriz gero "\r\n" aurkitu
gabe -2 itzuliko du.
* Beste edozein error gertatu ezkero -3 itzuliko du.
*/
int readline(int stream, char *buf, int tam)
{
        /*
                Kontuz! Inplementazio hau sinplea da, baina ez da batere
eraginkorra.
        */
        char c;
        int guztira=0;
        int cr = 0;

        while(guztira<tam)
        {
                int n = read(stream, &c, 1);
                if(n == 0)
                {
                        if(guztira == 0)
                                return 0;
                        else
                                return -1;
                }
                if(n<0)
                        return -3;
                buf[guztira++]=c;
                if(cr && c=='\n')
                        return guztira;
                else if(c=='\r')
                        cr = 1;
                else
                        cr = 0;
        }
        return -2;
}

/*
* 'string' parametroko karaktere katea bilatzen du 'string_zerr' parametroan.
```

```
'string_zerr' bektoreko azkeneko elementua NULL izan behar da.
* 'string' katearen lehen agerpenaren indizea itzuliko du, edo balio negatibo
bat ez bada agerpenik aurkitu.
*/

int bilatu_string(char *string, char **string_zerr)
{
      int i=0;
      while(string_zerr[i] != NULL)
      {
            if(!strcmp(string,string_zerr[i]))
                  return i;
            i++;
      }
      return -1;
}

/*
* 'string' parametroa 'string_zerr' bektoreko karaktere kateetako batetik hasten
den egiaztatzen du. 'string_zerr' bektoreko azkeneko elementua NULL izan behar
da.
* 'string' parametro karaktere katearen hasierarekin bat egiten duen
'string_zerr' bektoreko lehenego karaktere katearen indizea itzuliko du. Ez bada
bat-egiterik egon balio negatibo bat itzuliko du.
*/
int bilatu_substring(char *string, char **string_zerr)
{
      int i=0;
      while(string_zerr[i] != NULL)
      {
            if(!strncmp(string,string_zerr[i],strlen(string_zerr[i])))
                  return i;
            i++;
      }
      return -1;
}

/*
* Ustekabeko zerbait gertatzen denean egin beharrekoa: dagokion errorea bidali
bezeroari eta egoera eguneratu.
*/
void ustegabekoa(int s)
{
      write(s,"ER1\r\n",5);
      if(egoera == ST_AUTH)
      {
            egoera = ST_INIT;
      }
      else if(egoera == ST_DOWN || egoera == ST_UP)
      {
            egoera = ST_MAIN;
      }
}

/*
* 's' streamaren bitartez eskuragarri dauden fitxategien zerrenda bidaltzen du.
Ez bada posible fitxategiak dauden katalogoa aztertzea balio negatibo bat
itzuliko du eta bestela, zerrendako fitxategi kopurua.
*/
int bidali_zerrenda(int s)
{
```

```
      struct dirent ** fitxategi_izenak;
      int i,fitxategi_kop;
      long fitxategi_tam;
      char buf[MAX_BUF], fitxategi_path[MAX_BUF];
      struct stat file_info;

      fitxategi_kop = scandir(FILES_PATH, &fitxategi_izenak, ez_ezkutua,
alphasort);
      if(fitxategi_kop < 0)
            return -1;
      if(write(s,"OK\r\n",4) < 4)
            return -1;

      for(i=0;i<fitxategi_kop;i++)
      {
            sprintf(fitxategi_path,"%s/%s",FILES_PATH,
fitxategi_izenak[i]->d_name);
            if(stat(fitxategi_path,&file_info) < 0)
                  fitxategi_tam = -1L;
            else
                  fitxategi_tam = file_info.st_size;
            sprintf(buf,"%s?%ld\r\n",fitxategi_izenak[i]->d_name,
fitxategi_tam);
            if(write(s, buf, strlen(buf)) < strlen(buf))
                  return i;
      }
      write(s,"\r\n",2);
      return fitxategi_kop;
}

/*
* Eskuragarri dauden fitxategiak kokatuta dauden diskoan eskuragarri dagoen
tokia itzultzen du bytetan. Erroreren bat gertatuz gero balio negatibo bat
itzuliko du.
*/
unsigned long toki_librea()
{
      struct statvfs info;

      if(statvfs(FILES_PATH,&info)<0)
            return -1;
      return info.f_bsize * info.f_bavail;
}

/*
* Katalogo bat eskaneatzerakoan fitxategi ezkutuak kontuan ez hartzeko eginiko
iragazkia. Fitxategia '.' karaktereaz hasten bada 0 itzuliko du eta bestela 1.
Kontuan izan fitxategi ezkutuak adierazteko modu hau UNIX eta antzeko sistemetan
erabiltzen dela, beraz Windows sistemetan ez du esperotako portaera izango.
*/
int ez_ezkutua(const struct dirent *entry)
{
   if (entry->d_name[0]== '.')
     return (0);
   else
     return (1);
}
```