Università degli studi di Napoli Parthenope

Laboratorio di Reti di Calcolatori

Anno:2023/2024

TRACCIA – FTP

Federico Barretta - matricola :0124/002858

Descrizione:

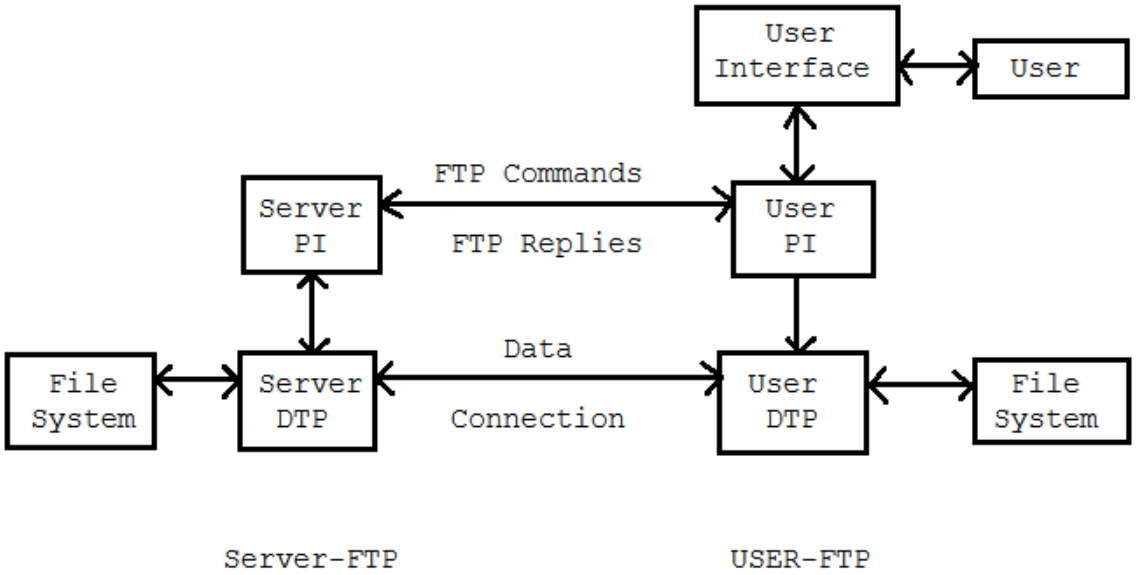
l'obbiettivo del progetto è quello di creare una connessione client/server impiegando il protocollo FTP per abilitare lo scambio di file tra il client e server.

L'utente può essere:

- Utenti Registrati, che può vedere una directory a lui assegnata contenente il nome di tutti i file a loro dedicati, la quale può farne il download e rinominarli, nella gestione della sua directory può effettuare l'upload di altri file non presenti nel server.

- Utenti Anonimi, che non essendo registrato vede una directory uguale per tutti, limitata poiché può solo vedere il nome dei file e effettuarne il download.

Descrizione e schema dell'architettura



Dettagli implementativi dei client/server:

- Avvio il server.
- Avvio del client
- Connessione tra il client e il server
- Il client effettua la scelta tra utente registrato e non registrato
- Il server legge la scelta e predispone il client per l'inserimento della password, se registrato, oppure invia un messaggio di benvenuto all'utente anonimo.
- Il server invia l'elenco di file e le possibili scelte in base al tipo di utente
- Il client risponde inserendo la scelta.
- Il server predispone per l'inserimento del nome del file su cui lavorare oppure chiude la connessione inserendo la scelta exit.

Parti rilevanti del codice:

```c
void send_file(int sockfd, int connfd, struct sockaddr_in servaddr, struct sockaddr_in cli, char *file){

  socklen_t sock_len;
  ssize_t len;
  int sent_bytes = 0;
  FILE *fd;
  off_t offset;
  int remain_data;
  struct stat file_stat;
  char file_size [256];
  char buff [MAX];



  fd = (fopen(file, "r"));



  if (fd == NULL){
    fprintf(stderr, "Error opening file --> %s", strerror(errno));
    exit(0);
  }



  /* Get file stats */
  if (fstat(fileno(fd), &file_stat) < 0){
    fprintf(stderr, "Error fstat --> %s", strerror(errno));
    exit(EXIT_FAILURE);
  }

  fprintf(stdout, "File Size: \n%ld bytes\n", file_stat.st_size);

  sock_len = sizeof(cli);

  fprintf(stdout, "Accept peer --> %s\n", inet_ntoa(cli.sin_addr));
  sprintf(file_size, "%ld", file_stat.st_size);

  /* Sending file size */
  len = send(connfd, file_size, sizeof(file_size), 0);
  if (len < 0){
    fprintf(stderr, "Error on sending greetings --> %s", strerror(errno));
    exit(0);
  }

  fprintf(stdout, "Server sent %ld bytes for the size\n", len);

  offset = 0;
  remain_data = file_stat.st_size;
  /* Sending file data */
  while (((sent_bytes = sendfile(connfd, fileno(fd), &offset, sizeof(buff))) > 0) && (remain_data > 0)){
    remain_data -= sent_bytes;
    fprintf(stdout, "1. Server sent %d bytes from file's data, offset is now : %ld and remaining data = %d\n", sent_bytes, offset, remain_data);
  }

}
```

In figura è mostrata la funzione di invio file implementata lato server. Apre il file in lettura(“r”) , controlla se la connessione è andata a buon fine, viene utilizzata la funzione “fstat” per recuperare informazioni sul del file da inviare, effettua la send in cui invia le informazioni del file e poi invio il file con il metodo sendfile della libreria <sys/sendfile.h> che è una libreria di sistema.

```c
int sockfd, connfd, len;
char *file;
struct sockaddr_in servaddr, cli;

// socket create and verification
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd == -1) {
  printf("socket creation failed...\n");
  exit(0);
}
else
  printf("Socket successfully created..\n");
bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);

// Binding newly created socket to given IP and verification
if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
  printf("socket bind failed...\n");
  exit(0);
}
else
  printf("Socket successfully binded..\n");

// Now server is ready to listen and verification
if ((listen(sockfd, 5)) != 0) {
  printf("Listen failed...\n");
  exit(0);
}
else
  printf("Server listening..\n");
len = sizeof(cli);

// Accept the data packet from client and verification
connfd = accept(sockfd, (SA*)&cli, &len);
if (connfd < 0) {
  printf("server accept failed...\n");
  exit(0);
}
else
  printf("server accept the client...\n");
```

Nella seconda figura evidenziamo la connessione. Utilizziamo una socket per creare la connessione con il metodo socket della libreria di sistema <sys/socket.h>. identifichiamo utilizzando servaddr come variabile assegnata alla struttura per identificare il tipo di:

- Protoccolo internet con AF_INET
- Permetti la connessione con indirizzi esterni con INNADR_ANY
- Settiamo la porta con PORT(un define che definisce la porta utilizzata)

Tramite il Metodo Bind istauriamo la connessione con il client e con il metodo accept abilitiamo la connesione

```c
void edit_file_name(char * old_file_name, char *new_file_name){

  char c_name[MAX];

  FILE *fd = fopen("rg.txt", "r+");
  FILE *f_temp = fopen("temp.txt", "w");
  if (fd == NULL || f_temp == NULL){
    fprintf(stderr, "Failed to open file --> %s\n", strerror(errno));
    exit(0);
  }


  while (fscanf(fd, "%s" , c_name) != EOF){

    if (strncmp(c_name, old_file_name, sizeof(old_file_name)) == 0){

      if (rename(old_file_name, new_file_name) == 0)
        printf("Name changed into %s\n", new_file_name);
      else{
        fprintf(stderr, "Error --> %s\n", strerror(errno));
        exit(0);
      }
      fprintf(f_temp, "%s\n", new_file_name);
    } else
      fprintf(f_temp, "%s\n", c_name);
  }



  fclose(fd);
  fclose(f_temp);

  remove("rg.txt");
  rename("temp.txt", "rg.txt");
}
```

Apriamo i due file con fopen il primo in lettura e scrittura(“r+”) il secondo solo in scrittura usandolo come un file temporaneo. Apriamo il ciclo che ci permetterà di effettuare la rinomina, usiamo fscanf per leggere da File(fd), utiliazziamo strncmp per confrontare le stringhe con la certezza di non avere caratteri in eccesso effettuiamo la rename del file, se fallisce si ferma, se va a buon fine si fa fprintf del file inserendogli il nuovo nome. Chiudiamo i file, utilizziamo la remove per eliminare il vecchio file e la rename per rinominare il file temporaneo e crearne uno nuovo.

Manuale Utente

Compilazione dei file:

- Dalla cartella client
  - Comando da utilizzare “make client”
- Dalla cartella server
  - Comando da utilizzare “make server”

Pulizia dei file di compilazione

- Dalla cartella client e server
  - Comando “make clean”

Istruzioni per l’esecuzione

- Dalla cartella server
  - ./server
- Dalla cartella client
  - ./client