Gruppo 17

Università degli Studi di
Bologna Scuola di Ingegneria

Esercitazione 4
Server Multiservizio: Socket C con select

Corso di Reti di Calcolatori T

Daniele Magagnoli, Giulia Martina Bal,
Antonio Cassanelli, Gianmiriano Porrazzo

Anno Accademico 2021/2022

SPECIFICHE:

L’esercitazione chiedeva la realizzazione di un server multiservizio e due client.

- Il primo servizio viene realizzato utilizzando socket senza connessione (datagram) per eliminare tutte le occorrenze di una parola in un file presente sul server remoto.

- il secondo servizio, realizzato tramite socket con connessione (stream), restituisce tutti i nomi dei file presenti nei direttori di secondo livello.

- Il primo Client, chiede ciclicamente all’utente il nome del file e la parola, invia al server la richiesta di eliminazione e riceve il numero di occorrenze eliminate.

- Il secondo Client, chiede ciclicamente all’utente il nome di un direttorio e riceve la lista di nomi di file remoti che stanno nei direttori di secondo livello.

```c
while((read(connfd, nome_dir, sizeof(nome_dir))) > 0)
{
    printf("Richiesto dir %s\n", nome_dir);

    //provo ad aprirla per vedere se esiste
    fd_dir= opendir(nome_dir);
    //se non esiste torno un messaggio di errore
    if (fd_dir==NULL){
        printf("Directory inesistente\n");
        write(connfd, "-1" , 1);
    }
    else{
         startStream=clock();
        //ciclo per ogni elemento dentro alla directory
        while ((dd = readdir(fd_dir))!= NULL)
        {
            sprintf(path1, "%s/%s",nome_dir, dd->d_name);
            //controllo se l'elemento è una directory
            if ((fd_dir2 = opendir(path1))!=NULL && dd->d_name[0] != '.')
            {
                printf("apertura di %s\n", dd->d_name);
                //ciclo ogni elemento della sotto-directory

                while((dd2 = readdir(fd_dir2)) != NULL)
                {
                    sprintf(path2, "%s/%s", path1, dd2->d_name);
                    //controllo se l'elemento è un file
                    if ((fd_dir3 = opendir(path2))==NULL && dd2->d_name[0]!= '.')
                    {   //mando il nome del file al client
                        printf("Trovato il file %s\n", dd2->d_name);
                        write(connfd, dd2->d_name, strlen(dd2->d_name)+1);
                        write(connfd, "\n", 1);
                    }
                    else{closedir(fd_dir3); continue;}
                }
                printf("chiusura %s\n", dd->d_name);
                closedir(fd_dir2);
            }
        }
        endStream=clock();
        printf("Tempo di escuzione figlio: %f\n", (float)(endStream-startStream)/CLOCKS_PER_SEC);
    }//else
    printf("chiusura %s\n", nome_dir);
    closedir(fd_dir);
    //mando un carattere speciale al client per dirgli che i file sono terminati
    write(connfd, "*", 1);
    printf("In attesa di una directory...\n");
}//while
```

SERVIZIO CON CONNESSIONE

La parte del server che realizza il servizio con connessione si occupa prima di tutto di creare un figlio, quest'ultimo si occupa ricavare i nomi dei file contenuti nei direttori di secondo livello rispetto al direttorio passatogli dal Client. Il figlio mantiene la connessione aperta fin tanto che l'operazione di lettura non viene terminata e il server invia il carattere speciale '*' per notificare al Client la fine della lettura.

```c
if (recvfrom(udpfd, &fparola, sizeof(FParola), 0, (struct sockaddr *)&cliaddr, &len)<0)
{perror("recvfrom"); continue;}


strcpy(nome_file,fparola.fileName);
strcpy(word,fparola.parola);

printf("Richiesta eliminazione della parola %s nel file %s\n", word, nome_file);

if ((fd_file = open(nome_file, O_RDONLY) )< 0) {
    printf("File inesistente\n");
    count = -1;
}
else {
    if((fdfd=open("temp", O_WRONLY|O_CREAT|O_TRUNC, 0777))<0)
        perror("open");

     startDatagram=clock();
    while ((read(fd_file, ch, sizeof(char))) > 0) {

        if (strcmp(ch, " ")==0 || strcmp(ch,"\n")==0) {
            if (strcmp(word, buffer)==0){
                count=count+1;
            }

           else{
            strcat(buffer, ch);
            write(fdfd, &buffer, strlen(buffer));
        }
            strcpy(buffer, "");
        }else
        strcat(buffer, ch);
    }
    close(fdfd);
}
endDatagram=clock();
printf("Fine ricerca parola %s, numero di volte in cui c'è stata %d\n",word, count);

printf("Tempo di escuzione byte per byte: %f\n", (float)(endDatagram-startDatagram)/CLOCKS_PER_SEC);
if (sendto(udpfd, &count, sizeof(count), 0, (struct sockaddr *)&cliaddr, len)<0)
{perror("sendto"); continue;}

count=0;
printf("Ho inviato il numero di parole eliminate\n");
```

SERVIZIO SENZA CONNESSIONE

La parte del server che realizza il servizio senza connessione riceve dal clienti una struttura dati contenente il nome del file e la parola da eliminare. Il ciclo prevede la lettura carattere per carattere del file e servendosi di un file d'appoggio, riscrive su quest'ultimo il contenuto del primo eliminando le occorrenze della parola. Il numero delle occorrenze viene infine mandato al Client.

```c
while (gets(nome_dir)){

    if (write(sd, nome_dir, (strlen(nome_dir)+1))<0)
    {
        perror("write");
        printf("Nome del file da richiedere: ");
        continue;
    }

    printf("Richiesta dei file nei sotto-direttorii di %s inviata... \n", nome_dir);
    //continuo a leggere un byte alla volta finchè il server non mi manda '*'
     start=clock();
    while(read(sd, &buff, 1)>0 && buff!='*')
    {
        /*se il carattere che leggo è un numero negativo
        interpreto il risulatato come un fallimento*/
        if(buff == '-')
        {
            printf("Il direttorio passato non esiste.\n");
            continue;
        }
        /*altrimento lo scrivo in output*/
        else
            write(1, &buff, 1);
    }
    end=clock();
    printf("Tempo di escuzione stream: %f\n", (float)(end-start)/CLOCKS_PER_SEC);
    printf("Nome del direttorio da richiedere: ");
}//while

/*prima di terminare il client chiudo la connessione*/
printf("Chiudo connessione\n");
shutdown(sd,1); shutdown(sd, 0);
close(sd); //chiusura sempre DENTRO
```

STREAM CLIENT

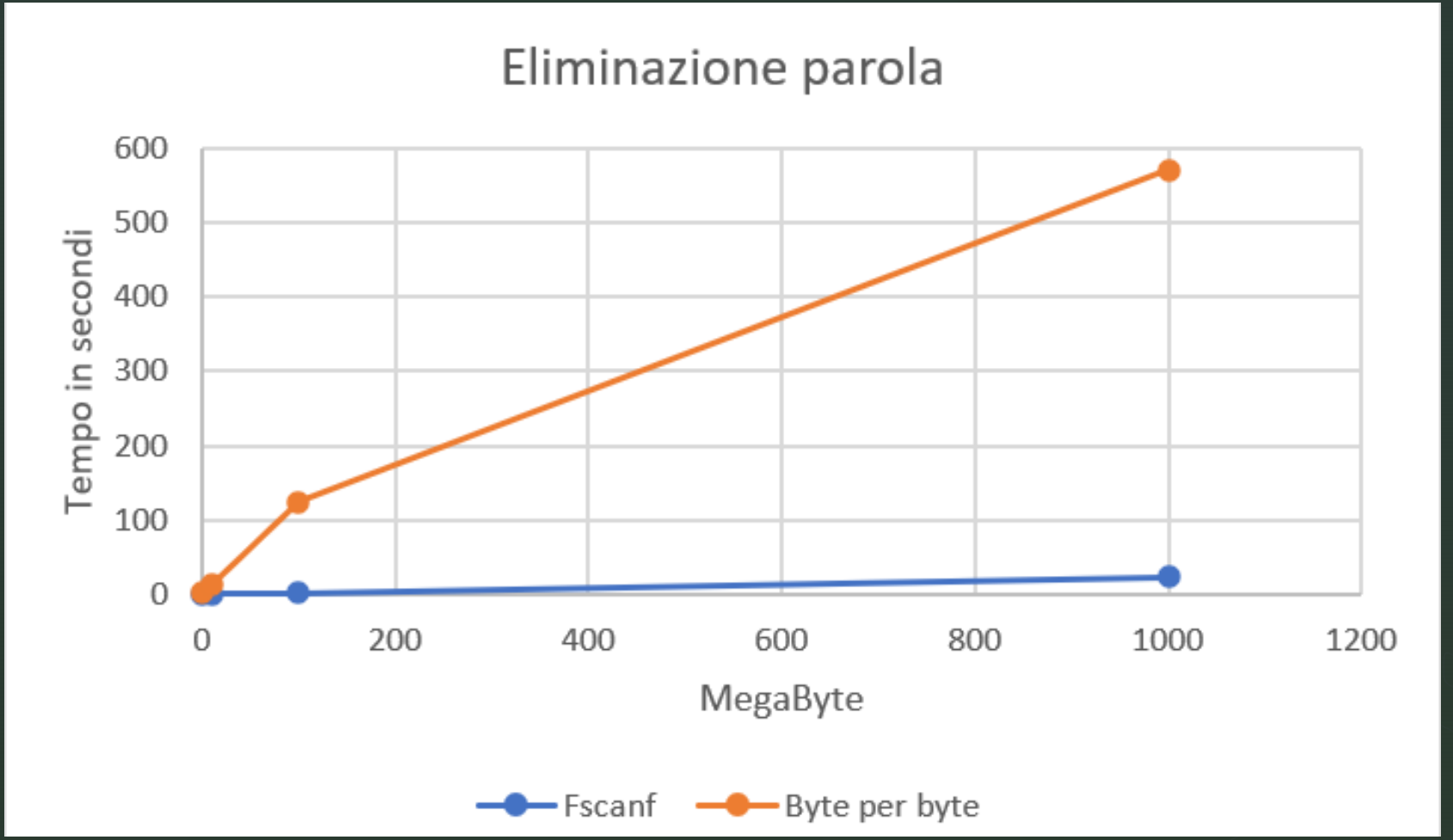
Dopo aver preso da standard input il nome di una directory, il Client con connessione, lo manda al server. Se il messaggio che riceve è un intero negativo ritorna un messaggio di errore, altrimenti lo scrive su standard output.
Il client continua a chiede il nome di una directory finché non riceve EOF.

```c
while (gets(nome_file)){
    strcpy(req.nome,nome_file);
    printf("nome: %s\n",req.nome);
    printf("Parola da eliminare: ");
    gets(parola);
    strcpy(req.p,parola);
    printf("parola: %s\n",req.p);

    /* invio richiesta */
    len=sizeof(servaddr);

    if (sendto(sd, &req, sizeof(Request), 0, (struct sockaddr *)&servaddr, len)<0){
        perror("scrittura socket");
        continue; // se questo invio fallisce il client torna all'inzio del ciclo
    }

    /* ricezione del risultato */
    printf("Attesa del risultato...\n");
    if (recvfrom(sd, &ris, sizeof(ris), 0, (struct sockaddr *)&servaddr, &len)<0){
        perror("recvfrom");
        continue; // se questa ricezione fallisce il client torna all'inzio del ciclo
    }

    if (ris<0) printf("Il file %s è scorretto o non esiste\n", nome_file);
    else printf("Nel file %s sono state eliminate %d parole\n",nome_file,ris);

    printf("Nome del file: ");

    end=clock();
    printf("Tempo di escuzione datagram: %f\n", (float)(end-start)/CLOCKS_PER_SEC);

} // while
```

DATAGRAM CLIENT

Questo client manda al server un messaggio contenente il nome di un file e una parola presi precedentemente da standard input. Quindi attende il numero di volte che il server ha trovato la parola nel file stampandolo su standard output.

CONSIDERAZIONI FINALI:



GRAZIE
PER L’ATTENZIONE