Gruppo 17

Università degli Studi di Bologna
Scuola di Ingegneria

Esercitazione 8
Remote Procedure Call (RPC)

Corso di Reti di Calcolatori T

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

Anno Accademico 2021/2022

SPECIFICHE DELL'ESERCITAZIONE

- L'esercitazione prevede lo sviluppo di un'applicazione Client/Server usando RPC per utilizzare delle procedure remote.
- Più nel dettaglio viene chiesto di realizzare una procedura remota per contare i caratteri, le parole e le linee di un file di testo ed un'altra per contare il numero di file, presenti in una cartella remota, la cui dimensione risulti essere maggiore di un intero indicato dal Client.

FILE XDR

```
const MAXLENGTH = 256;

struct infoFile {int nChar; int nWord; int nLine;};

struct req_dir {string nomeDir<MAXLENGTH>; int limit;};


program OPFILEPROGRAM{
    version OPFILEVERS{
        infoFile file_scan(string) = 1;
        int dir_scan(req_dir) = 2;
    } = 1;
} = 0x20000001;
```

Per prima cosa abbiamo definito le procedure remote nel file operazioni.x.

In questo file abbiamo creato due strutture con lo scopo di gestire i dati ritornati dalla procedura file_scan e inviare il nome della cartella insieme alla dimensione minima per l'input della procedura dir_scan.

FILE_SCAN

La procedura file_scan si occupa di trovare il numero di caratteri, parole e linee presenti in un file all'interno del Server, il cui nome ci viene specificato dal Client.

In caso il file non sia presente sul Client, assegniamo alla struttura il valore -1.

```
infoFile *file_scan_1_svc(char **file, struct svc_req *rp){
    static infoFile ris;
    char ch;
    int n, fd, first = 0;

    if(*file == NULL) exit(1);
    printf("File ricevuto: %s\n", *file);
    if ((fd = open(*file, O_RDONLY)) < 0)
    {
        perror("File inesistente");
        ris.nChar=-1; ris.nLine=-1; ris.nWord=-1;
    }
    else
    {
        while ((n=read(fd, &ch ,1))>0)
        {
            if(ch!=' ' && ch!='\n')
            {
                if(first==0) first = 1;
                ris.nChar++;
            }

            if(ch==' ' || ch==';' || ch=='.' || ch==':' || ch==',' || (ch=='\n' && first == 1))
                ris.nWord++;

            if(ch=='\n')
            {
                ris.nLine++;
                first=0;
            }

        }
    }
    close(n);

    ris.nLine++;
    return (&ris);
}
```

DIR_SCAN

La procedura dir_scan prende in input il nome di una cartella specificata dal Client, che si deve trovare nel file-system del Server, ed il numero minimo che indica la dimensione minima dei file presenti in tale cartella.

Il numero di file che superano tale dimensione viene inviato come risultato.

In caso di cartella inesistente viene restituito -1.

```c
int *dir_scan_1_svc(req_dir *req, struct svc_req *rp)
{
    static int result;
    DIR *dir;
    int f;
    struct dirent *d;
    char pathD[MAXLENGTH] = "", pathF[MAXLENGTH];//una variabile per il path della directory e una per i file

    printf("Parametri richiesti: directory:%s  dimMax:%d\n", req->nomeDir, req->limit);

    if((dir=opendir(req->nomeDir)) == NULL)
    {
        printf("Apertura di %s fallita.\n", req->nomeDir);
        result=-1;
    }
    else
    {
        int count=0;
        strcat(pathD, req->nomeDir); strcat(pathD, "/");

        while(d=readdir(dir))
        {
            strcpy(pathF, pathD); strcat(pathF, d->d_name);
            if((f=open(pathF, O_RDONLY))>0 && d->d_name[0]!='.')
            {
                printf("aperto il file:%s\n", d->d_name);
                if((int)lseek(f, 0, 2) > req->limit)
                    count++;
                close(f);
            }
        }
        result=count;
    }
    closedir(dir);
    return &result;

}
```

CLIENT

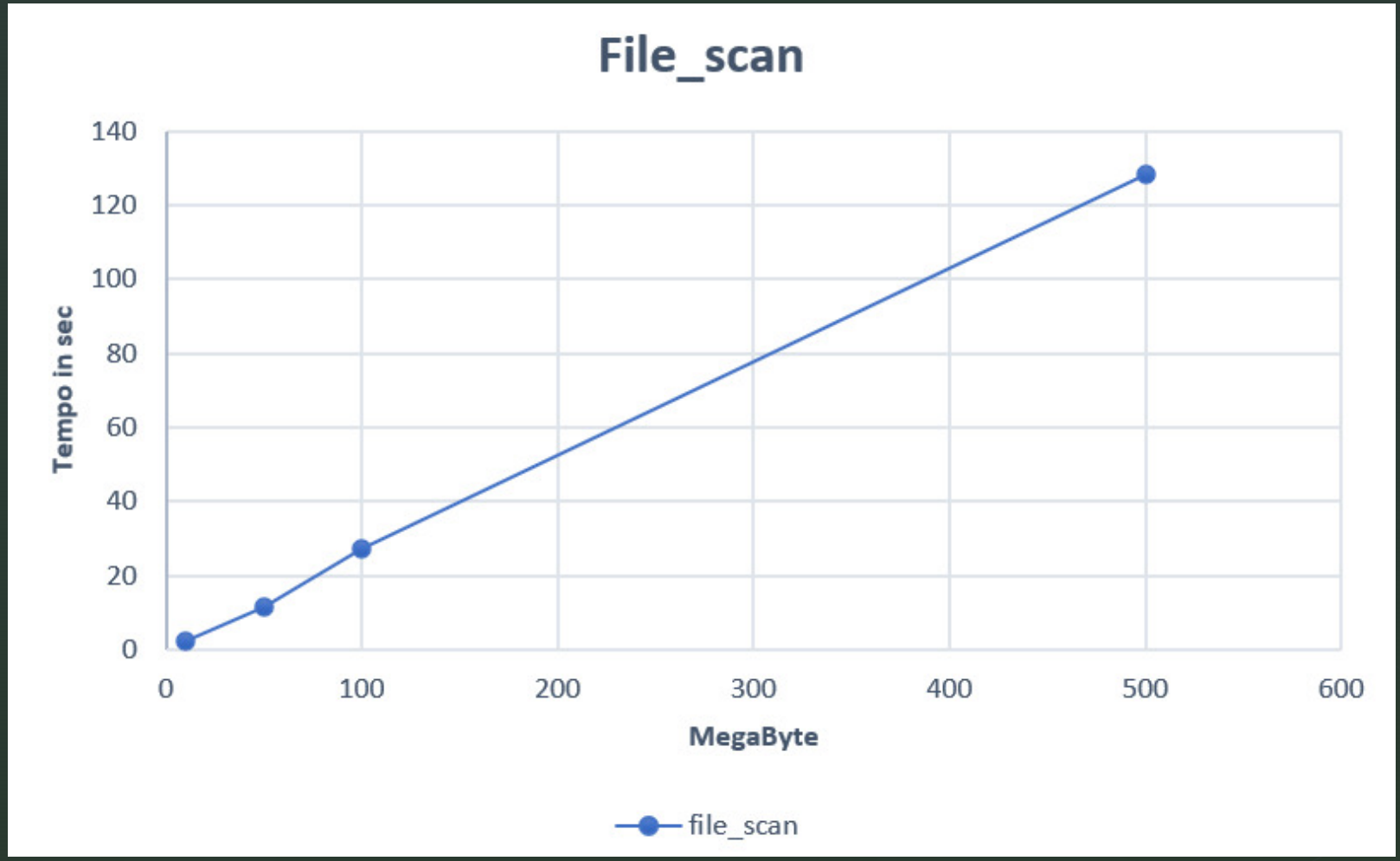
Il Client è un filtro che interagisce con l'utente fin quando non riceve ^D (Unix) o ^Z (Windows).

Ad ogni ciclo il Client chiede all'utente quale procedura vuole svolgere, per poi far inserire all'utente stesso i parametri necessari.

Alla fine della procedura viene stampato il risultato o un messaggio di errore.

```
while((s=getc(stdin))!= EOF)
{
    while(getc(stdin) != '\n');//per pulire il buffer

    if (s != 'F' && s != 'D')
    {
        printf("Il tipo di operazione deve essere 'F' o 'D'\n");
        printf("Quale servizio vuoi utilizzare? (ctrl+D per terminare)\n");
        continue;
    }

    if(s=='F')
    {
        printf("Inserisci il nome di un file:\n");

        if(gets(input)==EOF)
            break;

        fileName=input;
        ris = file_scan_1(&fileName, cl);

        if (ris == NULL)
        {
            clnt_perror(cl, server);
            printf("Errore nell'analisi del file.\n\n");
            printf("Quale servizio vuoi utilizzare? (F= file scan, D= dir scan)\n");
            continue;
        }

        if(ris->nChar == -1)
            printf("File not found.\n");
        else
            printf("Risultato ricevuto da %s: caratteri %d , parole %d, righe %d\n", server, ris->nChar, ris->nWord, ris->nLine);

    }//case:F
    else
    {

        printf("Inserisci il nome di un direttorio:\n");
        if(gets(input)==EOF)
            break;

        req=malloc(strlen(input)+sizeof(int));

        //salvo il nome della directory in un array secondario
        strcpy(dir, input);
        req->nomeDir=dir;

        printf("Scegli una soglia minima: \n");

        if(gets(input)==EOF)
            break;

        for(int i = 0; input[i] != '\0'; i++){
            if(input[i] < '0' || input[i] > '9'){
                printf("Integer required.\n");
                printf("Quale servizio vuoi utilizzare? (ctrl+D per terminare)\n");
                continue;
            }
        }
        req->limit=atoi(input);

        if(req->limit<0)
        {
            printf("Intero negativo.\n");
            printf("Quale servizio vuoi utilizzare? (ctrl+D per terminare)\n");
            continue;
        }
        res = dir_scan_1(req, cl);


        if (res == NULL)
        {
            clnt_perror(cl, server);
            printf("Quale servizio vuoi utilizzare? (ctrl+D per terminare)\n");
            continue;
        }
        if(*res == -1)
            printf("Apertura del direttorio fallita.\n");
        else
            printf("File che superano %d in %s: %d\n", req->limit, req->nomeDir, *res);

        free(req);
    }//case:D
    printf("Quale servizio vuoi utilizzare? (ctrl+D per terminare)\n");
}//while
```

CONSIDERAZIONI FINALI



GRAZIE PER L'ATTENZIONE