

Esercitazione 3

Belano Andrea, Ceccolini Gabriele, Loddo Filippo, Merenda Simone.

Specifiche del Progetto - 1

Il primo progetto prevede un Cliente e un Server.

Il Cliente invia al Server il nome di un file e attende la risposta; il Server, dopo aver ricevuto il nome di un file, verifica il numero dei caratteri della parola più lunga del file indicato e risponde al Cliente con esso se il file esiste, altrimenti risponderà con una notifica di errore per indicare che il file non esiste. Il Cliente stamperà in seguito la risposta ricevuta.

Cliente senza Connessione

Il Cliente, dopo aver creato la Socket, attende il nome di un File da inviare al Server, in seguito si mette in attesa della risposta che, se negativa, indicherà che il file non esiste, altrimenti la risposta corrisponderà al numero di caratteri della parola più lunga presente in quel file.

```
printf("Inserisci nomi di file\n");

char nomeFile[256];
len = sizeof(servaddr);

while (scanf("%s", nomeFile) != EOF) {
    if (sendto(sd, nomeFile, sizeof(nomeFile), 0, (struct sockaddr*)&servaddr, len) < 0) {
        perror("Errore nella send");
        continue;
    }
    if (recvfrom(sd, &ris, sizeof(ris), 0, (struct sockaddr *)&servaddr, &len) < 0) {
        perror("Errore nella receive");
        continue;
    }
    if (ris < 0) {
        printf("Il file non esiste\n");
    } else {
        printf("La parola di lunghezza massima ha %d caratteri\n", ris);
    }
}
```

Server senza Connessione

Il Server riceverà dal Cliente il nome di un file. Prova ad aprirlo e ne legge il contenuto per poter capire quanto misura la parola di lunghezza massima. In seguito invia la risposta al Cliente.

```
for (;;) {
    if (recvfrom(sd, nomeFile, sizeof(nomeFile), 0, (struct sockaddr *)&cliaddr, &len) < 0) {
        perror("Errore nella receive");
        continue;
    }
    if (!fork()) {
        if ((fd = open(strcat(dir, nomeFile), O_RDONLY, 0777)) < 0) {
            printf("Errore nell'apertura del file\n");
            int err = -1;
            sendto(sd, &err, sizeof(err), 0, (struct sockaddr *)&cliaddr, len);
            exit(0);
        }
        int maxLen = 0;
        int cur = 0;
        char buffer[BUFFER_SIZE];
        int bufSize;
        int i;

        while ((bufSize = read(fd, buffer, BUFFER_SIZE)) > 0) {
            i = 0;
            while (i < bufSize) {
                if (buffer[i] == '\n' || buffer[i] == ' ') {
                    i = i + maxlen + 1;
                    cur = 0;
                } else {
                    do {
                        i--;
                        cur++;
                    } while (i > 0 && buffer[i] != '\n' && buffer[i] != ' ');
                    i = i + cur + 1;
                    while (i < bufSize && buffer[i] != '\n' && buffer[i] != ' ') {
                        cur++;
                        i++;
                    }
                    if (cur > maxlen) {
                        maxlen = cur;
                    }
                }
            }
        }
        close(fd);
        if (sendto(sd, &maxLen, sizeof(maxLen), 0, (struct sockaddr *)&cliaddr, len)<0) {
            perror("Errore nella send");
        }
        exit(1);
    }
}
```

Specifiche del Progetto - 2

Il secondo progetto prevede un Cliente e un Server con connessione.
Il Cliente invia il nome di un file e il numero della riga che si vuole rimuovere al Server, quest ultimo farà gestire a un processo figlio l'eliminazione della riga richiesta e spedirà il risultato al Cliente.

Cliente con Connessione

Il Cliente, dopo aver stabilito una connessione con il Server, invia la riga da eliminare e il contenuto del file.

Attende poi fino alla ricezione della risposta contenente il file modificato.

```
printf("Inserisci il nome del file:\n");
scanf("%s", nomeFile);

if (access(nomeFile, R_OK | W_OK)) {
    printf("Errore il file non esiste o non può essere letto/scritto\n");
    exit(2);
}

riga = 0;

printf("Inserisci la riga:\n");
scanf("%d", &riga);

if (riga <= 0) {
    printf("Inserire un numero maggiore di zero\n");
    exit(3);
}

if ((fd = open(nomeFile, O_RDONLY)) < 0) {
    printf("Errore nell'apertura del file %d\n", fd);
    exit(4);
}

write(sd, &riga, sizeof(int));

while ((len = read(fd, buffer, BUFFER_SIZE)) > 0) {
    write(sd, buffer, len);
}

shutdown(sd, SHUT_WR);
close(fd);

unlink(nomeFile);

fd = open(nomeFile, O_WRONLY | O_CREAT);

while ((len = read(sd, buffer, BUFFER_SIZE)) > 0) {
    write(fd, buffer, len);
}

shutdown(sd, SHUT_RD);
close(sd);
close(fd);
```

Server con Connessione

Il Server riceve la riga e il contenuto del file dal Cliente e inizia a leggere il file fino alla riga da cancellare. Ignora quindi la riga e ricomincia a scrivere nel buffer d'ingresso il contenuto del file a partire dalla riga successiva.

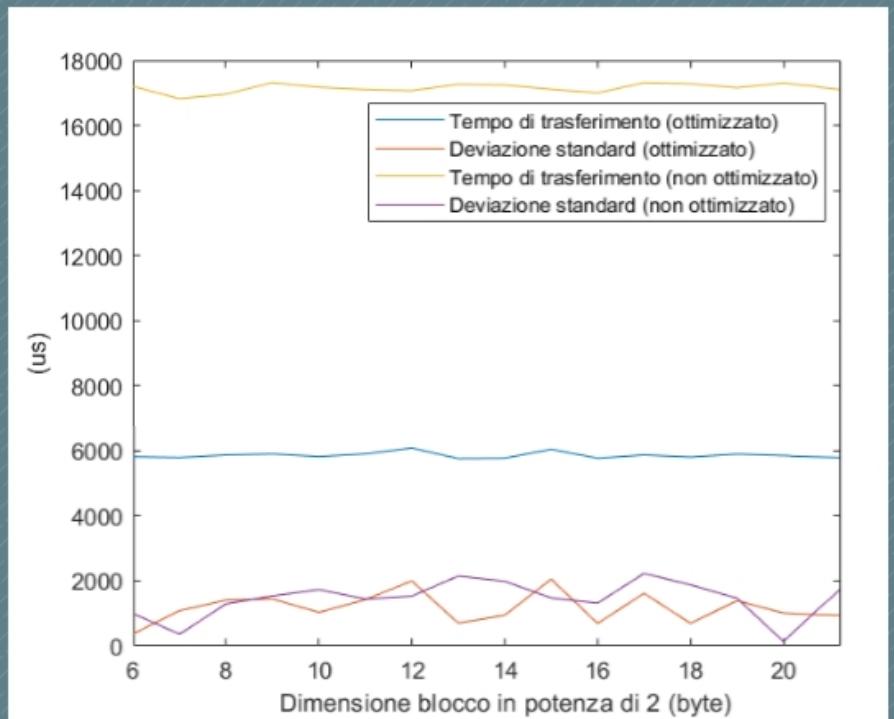
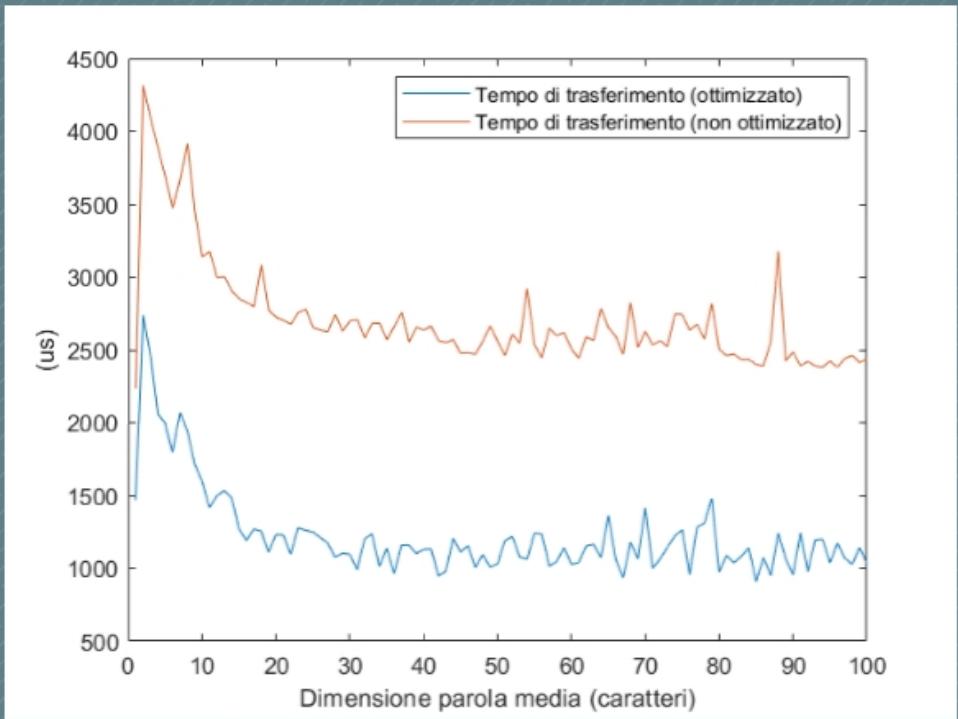
```
for (;;) {
    if((conn_sd = accept(sd,(struct sockaddr *)&cliaddr,&len)) < 0) {
        if (errno==EINTR) {
            perror("Forzo la continuazione della accept");
            continue;
        }
        else exit(1);
    }

    if (!fork()) {
        close(sd);
        int riga;
        if (read(conn_sd, &riga, sizeof(int)) <= 0) {
            printf("Errore\n");
            exit(0);
        }
        int numRighe = 1;
        int newBufSize = 0;
        int lineFlag = 1;
        while ((len = read(conn_sd, buffer, BUFFER_SIZE)) > 0) {
            for (int i = 0; i < len; i++) {
                if (lineFlag && (numRighe == riga)) {
                    newBufSize = i;
                    lineFlag = 0;
                } else if (!lineFlag && (numRighe != riga)) {
                    buffer[newBufSize] = buffer[i];
                    newBufSize++;
                }
                if (buffer[i] == '\n') {
                    numRighe++;
                }
            }
            write(conn_sd, buffer, newBufSize);
        }

        shutdown(conn_sd, SHUT_RD);
        shutdown(conn_sd, SHUT_WR);
        close(conn_sd);
    }

    close(conn_sd);
}
```

Test Ricerca



Dai test si vede che l'algoritmo ottimizzato sia fino al 101% più veloce rispetto a un algoritmo che legge carattere per carattere.

Grazie per l'ascolto

Belano Andrea, Ceccolini Gabriele, Loddo Filippo, Merenda Simone.