

Esercizio sulle pipe

Ogni processo P_i leggerà quindi una frazione di F allo scopo di calcolare il numero delle occorrenze del carattere C nella parte di file esaminata; al termine della scansione, P_i comunicherà al padre il numero delle occorrenze di C incontrate nella frazione di file assegnatagli.

Il padre P_0 , una volta ottenuti i risultati da tutti i figli, **stamperà il numero totale di occorrenze** di C nel file F e terminerà.

Esercizio

Realizzare un programma C che, utilizzando le system call di UNIX, preveda un'interfaccia del tipo: **esame F N C**

dove:

- F rappresenta il nome assoluto di un file
- N rappresenta un intero
- C rappresenta un carattere.

Il processo iniziale P_0 deve creare un numero N di processi figli (P_1, P_2, \dots, P_N).

Ogni figlio P_i ($i=1, \dots, N$) deve leggere una parte diversa del file F : in particolare, se L è la lunghezza del file F , il figlio dovrà leggere una frazione di L/N caratteri dal file F , secondo il seguente criterio:

- P_1 leggerà la prima frazione;
- P_2 leggerà la seconda frazione;
- ...
- P_N leggerà l'ultima frazione N ;

Impostazione

- Accesso parallelo di processi a parti diverse dello stesso file: aperture separate del file da parte dei figli per evitare la condivisione dell'I/O pointer.
- Comunicazione dei figli con il padre: uso di una pipe.
- Calcolo della dimensione di un file e posizionamento dell'I/O pointer in posizione arbitraria: `lseek()`

Soluzione dell'esercizio

```
#include <fcntl.h>
#define Nmax 50
#define Mdim 8

int fd, pipefd[2], L, fraz;
char c, msg[Mdim];
void figlio(int ind, char c, char filein[]);

main(int argc, char **argv)
{
    int pid[Nmax], N, stato, i, tot, K, p;

    if (argc!=4)
    {
        printf("sintassi!\n");
        exit(-1);
    }
}
```

Sistemi Operativi A

5

```
/* padre */
close(pipefd[1]);
for(tot=0,i=0; i<N; i++)
{
    read(pipefd[0],msg, Mdim );
    K=atoi(msg);
    tot+=K;
}
printf("\n valore ottenuto: %d\n", tot);
close(pipefd[0]);
for(i=0; i<N; i++)
{
    p=wait(&stato);
    printf("terminato%d con stato %d\n", p,
    stato>>8);
}
exit(0); /* fine padre */
} /* fine main */
```

Sistemi Operativi A

7

```
N=atoi(argv[2]);
c=argv[3][0];
if ((fd=open(argv[1],O_RDONLY)<0)
{
    perror("open padre");
    exit(-2);
}
L=lseek(fd, 0, 2); /* calcolo dim. L*/
fraz=L/N;
close(fd);
if (pipe(pipefd)<0) exit(-3); /* apertura pipe */
/* creazione figli */
for(i=0;i<N;i++)
    if ((pid[i]=fork())<0)
    {
        perror("fork"); exit(-3);
    }
    else if (pid[i]==0) figlio(i,c, argv[1]);
```

Sistemi Operativi A

6

```
void figlio(int ind, char c, char filein[])
{
    int i, j, cont=0; char car;
    close(pipefd[0]);
    fd=open(filein, O_RDONLY);
    lseek(fd, ind*fraz, 0); /*posiz. inizio frazione */
    for(j=0; j<fraz; j++)
    {
        read(fd, &car, 1);
        if (car==c)
            cont++;
    }
    sprintf(msg,"%d",cont);
    write(pipefd[1],msg, Mdim);
    close(pipefd[1]);
    close(fd);
    exit(0);
}
```

Sistemi Operativi A

8