

Quinta esercitazione

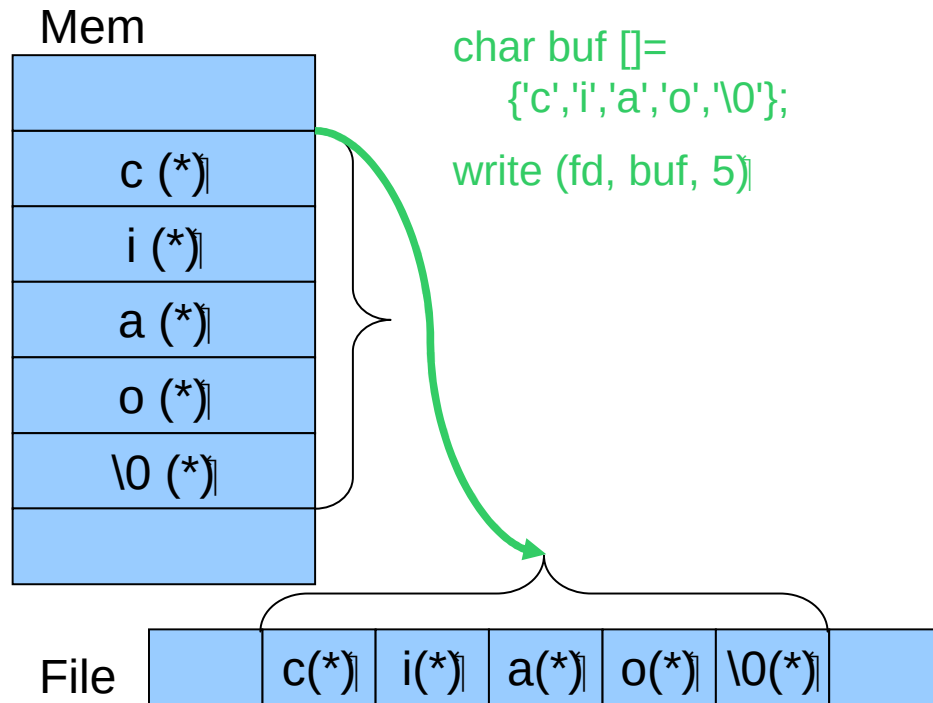
Gestione FileSystem Unix
Gestione Pipe Unix

Stefano Monti
stefano.monti6@unibo.it

Un breve richiamo...

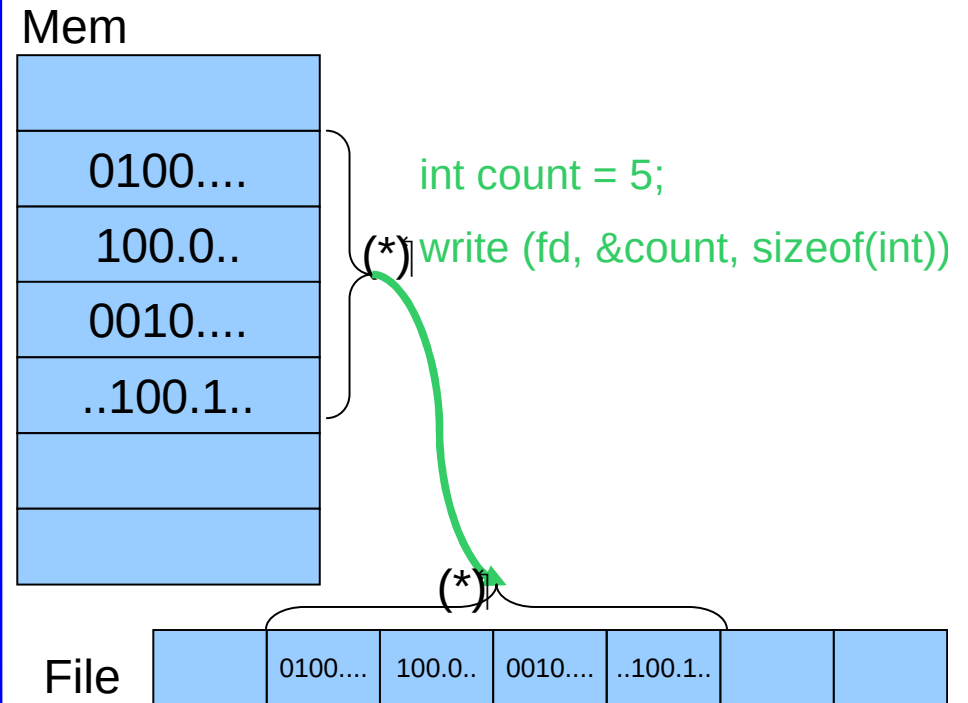
```
int read (int fd, char *buf, int n)\nint write (int fd, char *buf, int n)
```

Lettura/scrittura di file testuali



(*) ciascun byte (in memoria o su file) contiene la rappresentazione binaria del carattere corrispondente (cioè `sizeof(char) == 1`)

Lettura/scrittura di file binari



(*) i 4 byte (in memoria o su file) contengono la rappresentazione binaria dell'intero 5 (cioè `sizeof(int) == 4`)

Esercizio 1

Si realizzi un programma C usando le opportune system call Unix che abbia la seguente interfaccia

contaCaratteri <c1> <c2> <fileIn> <fileOut>

dove *c1* e *c2* sono 2 caratteri e *fileIn*, *fileOut* due nomi (assoluti) di file.

In particolare

- *fileIn* : file **di testo** presente su file system, organizzato in righe di lunghezza non nota a priori
- *fileOut* : nome di file **non presente** su file system

Specifiche

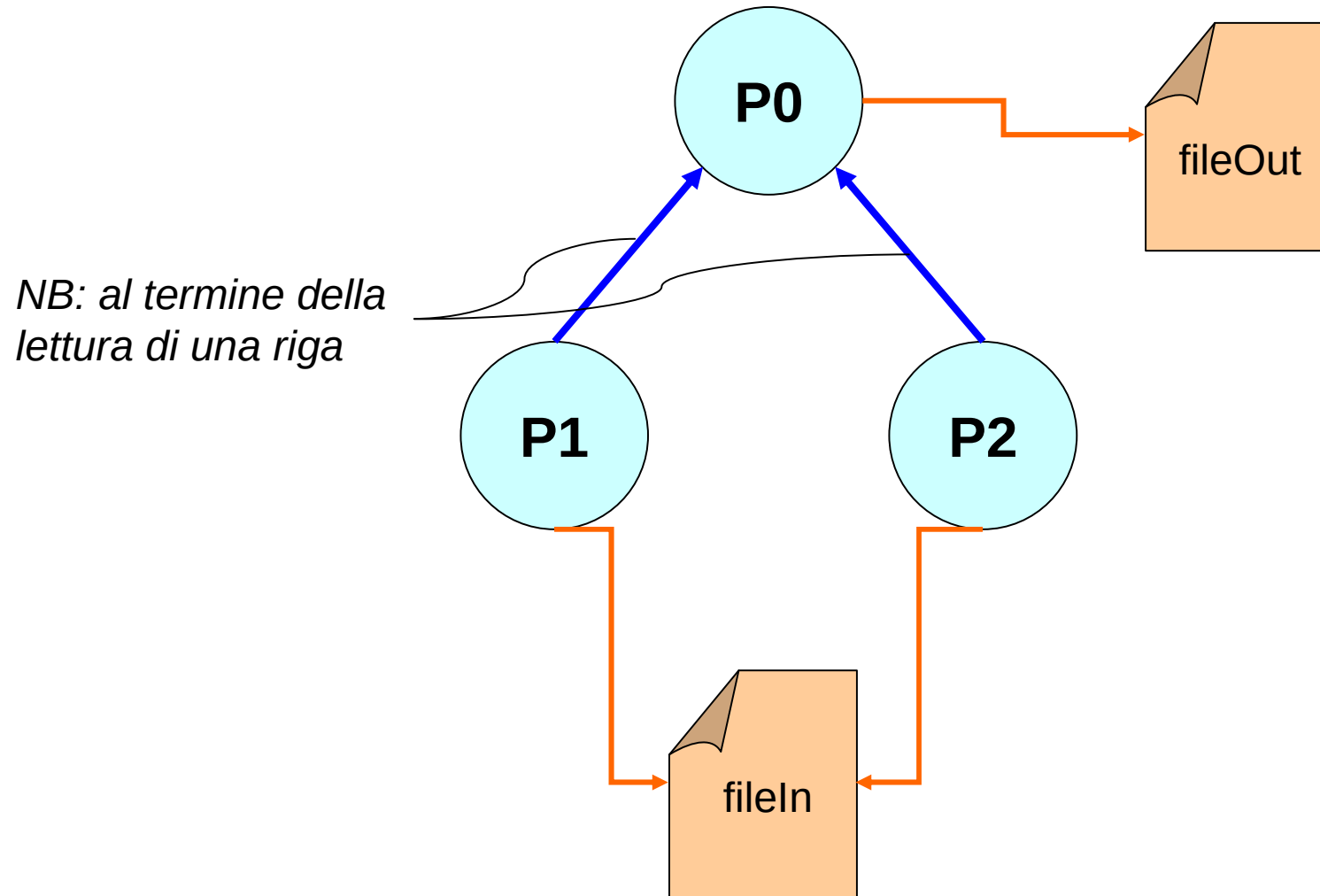
Il processo padre P0 deve generare due figli P1 e P2 (fratelli).

Ciascun figlio deve **leggere** *fileIn*; al termine della lettura di ciascuna riga di *fileIn*, il processo figlio **comunica al padre il numero di occorrenze del carattere cercato** (rispettivamente c1 per P1 e c2 per P2) **nella riga appena letta**.

Il padre deve tenere traccia del numero di occorrenze complessive di ciascun carattere.

Una volta terminata la lettura, il padre deve **scrivere** su *fileOut* il numero complessivo di occorrenze di c1 e di c2.

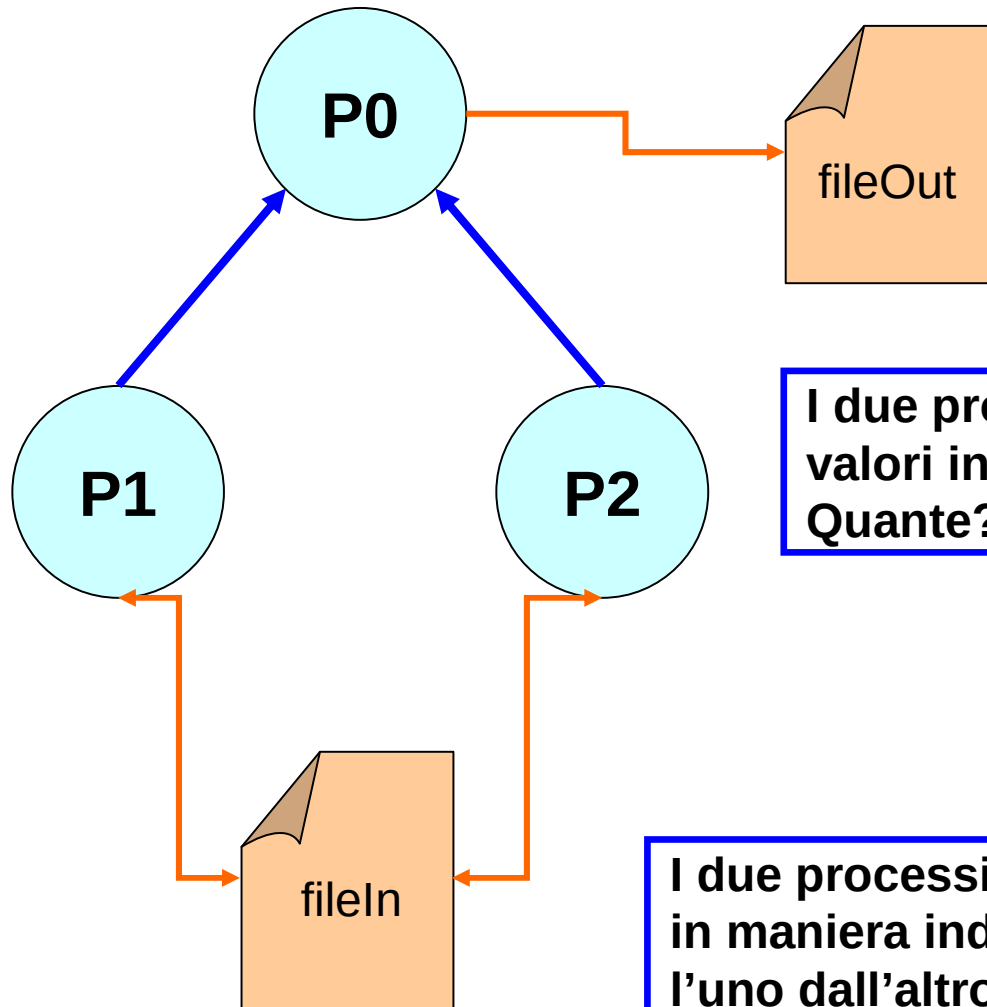
Modello di soluzione



Problematiche

- Gestione di file
 - Apertura in lettura o scrittura?
 - Apertura con IO pointer condiviso o separato?
- Gestione della comunicazione interprocesso
 - Quali (e quanti) strumenti?
- Come gestire la sincronizzazione tra processi?
 - in particolare, terminazione della lettura
 - Sono necessari segnali?

Strumenti IPC e gestione IO Pointer



I due processi figli devono comunicare valori interi al padre → utilizzo di pipe
Quante? Perché?

I due processi figli leggono *fileIn* in maniera indipendente
l'uno dall'altro → conviene IO Pointer separato

Generazione pipe

```
int main(int argc, char* argv[]) {  
    //--- controllo argomenti -----  
    ...  
    c1=argv[1][0];  
    c2=argv[2][0];  
  
    //--- apertura pipe da parte del padre-----  
    if (pipe(p1p0)<0){  
        printf("Errore nella creazione della pipe\n"); exit(-1);  
    } //P1 --> P0  
    if (pipe(p2p0)<0){  
        printf("Errore nella creazione della pipe\n"); exit(-1);  
    } //P2 --> P0  
  
    ...  
}
```

ATTENZIONE: perché apriamo
le pipe prima di aver generato i figli?

Generazione processi figli

```
//--- Generazione figli-----  
if((pid1=fork())<0) {  
    perror("Errore nella fork:"); exit(-2);  
}  
if( pid1 == 0) {  
    //--- Codice figlio P1-----  
}  
else{  
    if((pid2=fork())<0) {  
        perror("Errore nella fork:"); exit(-2);  
    }  
    if( pid2 == 0) {  
        //--- Codice figlio P2-----  
    }  
    else{  
        //--- Codice Padre -----  
    }  
}
```

Codice P1

```
if (pid1==0){//Codice P1
```

```
    if(((fileIn=open(argv[3],O_RDONLY))<0) {  
        perror("Error opening file:"); exit(-1); }
```

Apertura in sola lettura
Perché IO Pointer non condiviso?

```
//Chiusura dei lati di pipe non necessari
```

```
close (p1p0[0]); close (p2p0[1]); close (p2p0[0]);
```

```
//Ciclo di lettura
```

```
int byteLetti; char c; int occorrenze = 0;
```

```
while(((byteLetti=read(fileIn,&c,sizeof(char)))>0){
```

```
    if (c == '\n'){
```

```
        write(p1p0[1],&occorrenze,sizeof(int));
```

```
        occorrenze=0;
```

```
    }
```

```
    else if (c == c1)
```

```
        occorrenze++;
```

```
}
```

```
//Chiusura delle risorse
```

```
close (p1p0[1]); close(fileIn);
```

```
}
```

ATTENZIONE: perché leggiamo
carattere per carattere?

Scrittura su pipe per
comunicare al padre P0

Codice P0 (1/2)

```
else{//Codice padre
    //Apertura file in scrittura
    if((fileOut=open(argv[4],O_WRONLY|O_CREAT,0600))<0) {
        perror("Error2 opening file:");    exit(-1); }
    //Chiusura dei lati di pipe non necessari
    close (p1p0[1]); close (p2p0[1]);
    //Ciclo di letture dalle pipe
    int count_c1 = 0, count_c2 = 0;
    int byteLetti1 , byteLetti2, occorrenze1, occorrenze2;
    do{
        byteLetti1 = read(p1p0[0],&occorrenze1,sizeof(int));
        if (byteLetti1 > 0) {
            count_c1 += occorrenze1; }
        byteLetti2 = read(p2p0[0],&occorrenze2,sizeof(int));
        if (byteLetti2 > 0) {
            count_c2 += occorrenze2; }
    } while (byteLetti1 > 0 || byteLetti2>0);
```

ATTENZIONE:

- quando usciamo da questo ciclo?
- che effetti dal punto di vista della sincronizzazione?

Codice P0 (2/2)

...

//Scrittura su fileOut

char log[100];

sprintf(log,"Numero occorrenze %c : %d\n",c1,count_c1);

write(fileOut,log,strlen(log));

sprintf(log,"Numero occorrenze %c : %d\n",c2,count_c2);

write(fileOut,log,strlen(log));

//Chiusura delle risorse

close (p1p0[0]); close (p2p0[0]); close(fileOut);

//Raccolta stato terminazione dei figli

wait(&status); wait(&status);

}

Esercizio 2

Si realizzi un programma C usando le opportune system call Unix che abbia la seguente interfaccia

correggiSomme <fileIn> <fileOut>

dove *fileIn*, *fileOut* sono due nomi (assoluti) di file.

In particolare

- *fileOut* : nome di file non presente su file system
- *fileIn* : file **binario** presente su file system contenente un numero N non noto a priori di triplette di interi:

A	B	C	A	B	C	A	B	C	...
---	---	---	---	---	---	---	---	---	-----

Ad esempio:

1	3	4	78	49	264	40	56	96	...
---	---	---	----	----	-----	----	----	----	-----

Specifiche

Il processo padre P0 deve generare due figli P1 e P2 (fratelli)

Il processo P2 deve

- **leggere** i primi due interi (A,B) di ogni tripletta in *fileIn*
- **comunicare** a P1 tale coppia di interi letti

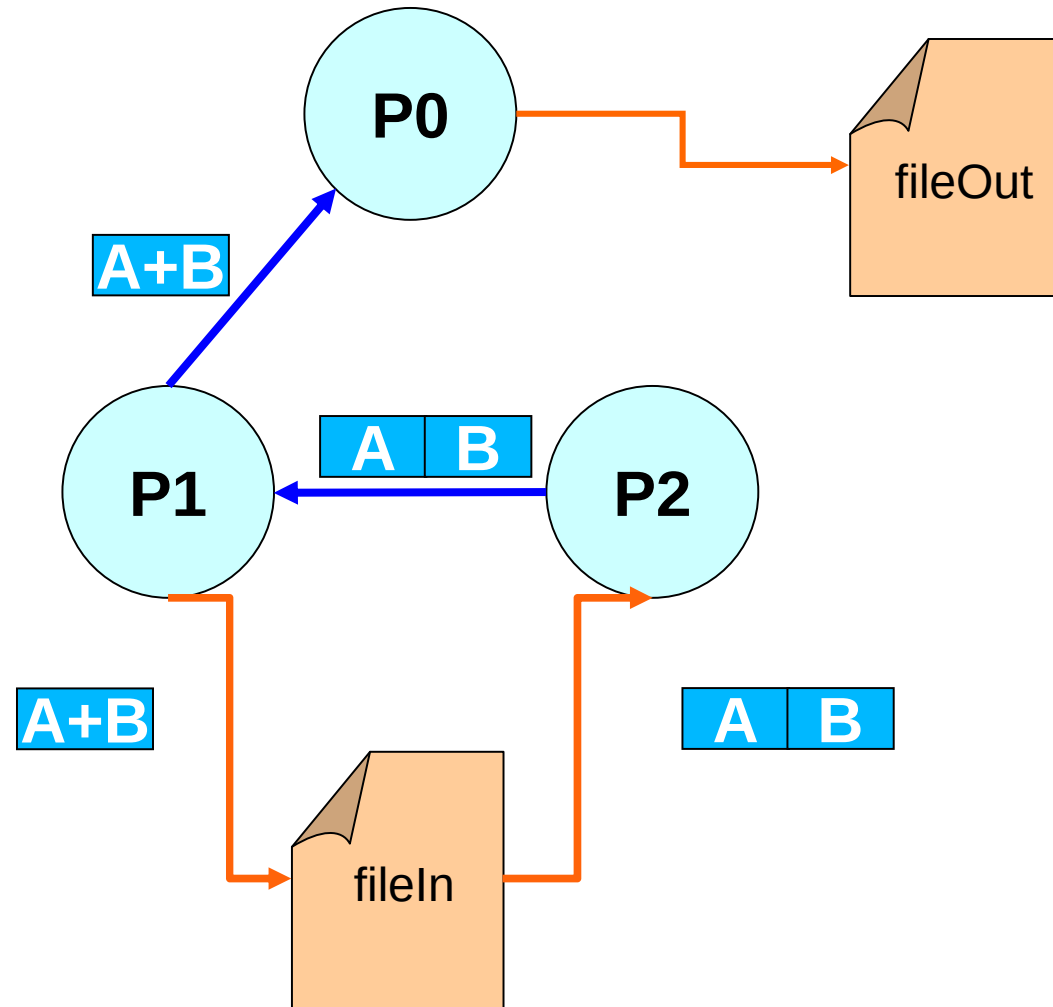
Il processo P1 deve

- sommare ciascuna coppia di interi ricevuti da P2
- **scrivere** il valore della somma così ottenuta **al posto del relativo elemento C** della tripletta
- **comunicare** a P0 il valore di tale somma

Il processo P0 deve

- tenere traccia del massimo valore di somma ricevuto da P1
- al termine della elaborazione dei figli, **scrivere** tale valore su *fileOut*

Modello di soluzione



Problematiche

- Gestione di file
 - Apertura in lettura o scrittura?
 - Apertura con IO pointer condiviso o separato?
- Gestione della comunicazione interprocesso
 - Quali (e quanti) strumenti?
- Come gestire la sincronizzazione tra processi?
 - in particolare, terminazione della lettura
 - sono necessari segnali?