Seconda Esercitazione

System call per la gestione di processi

Stefano Monti
smonti@deis.unibo.it

System call fondamentali

`fork`	·Generazione di un processo figlio, che condivide il codice col padre e possiede dati replicati ·Restituisce il pid del figlio per il padre, 0 per il figlio o un numero <0 in caso di errore
`exit`	·Terminazione di un processo ·Si indica lo stato di terminazione; normalmente stato=0 se il processo termina correttamente, un valore diverso da 0 in caso di problemi
`wait`	·Raccolta dello stato di terminazione di un figlio (con eventuale attesa) ·Restituisce il pid del figlio terminato e permette di capire il motivo della terminazione (Volontaria? Con quale stato? Involontaria? A causa di quale segnale?)

Esempio 1 - fork e exit

- Scrivere un programma in cui il processo padre procede alla istanziazione di un numero N di figli

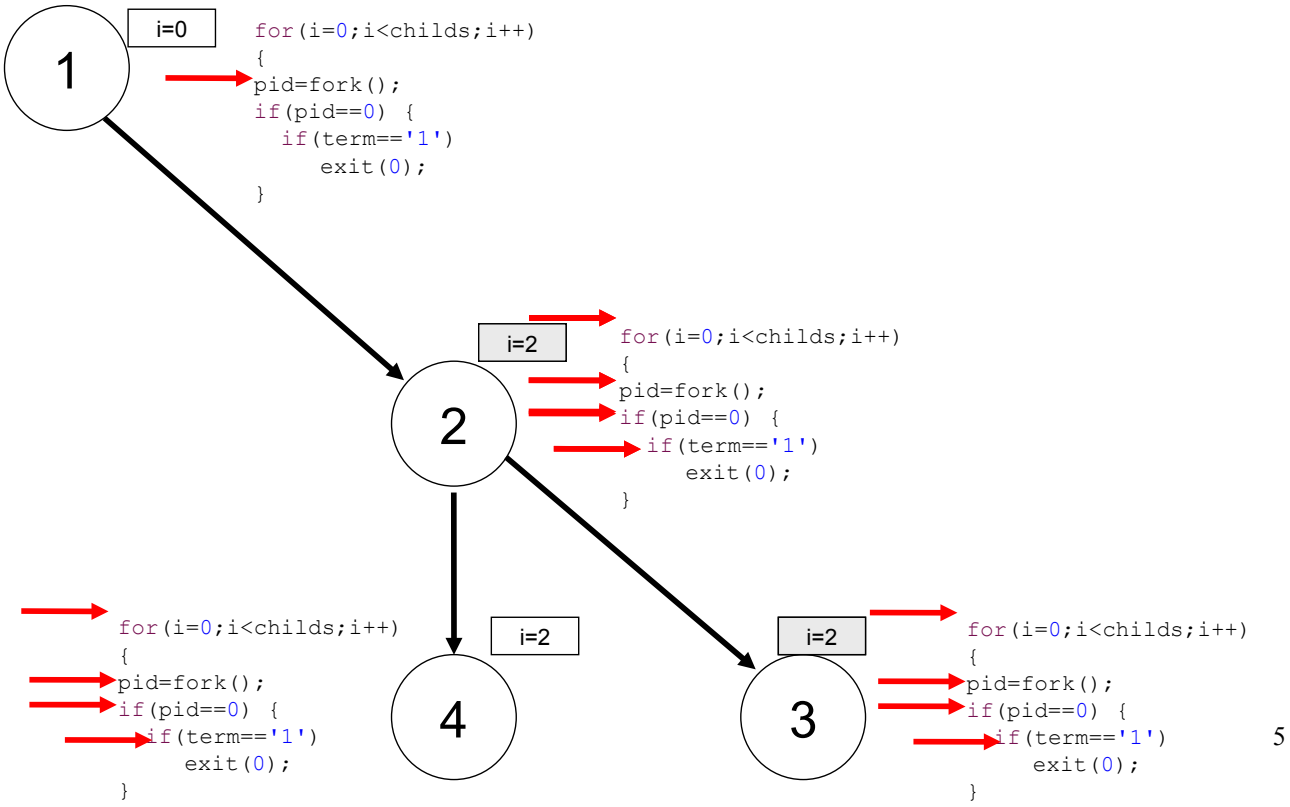
  ./generate <N> <term> (con *term* [0,1])

- Attenzione all'indicazione di terminazione dei figli
  - Cosa succede se ci si dimentica di effettuare la `exit()`?

Esempio 1

```
int main (int argc, char * argv[]){
  int i,j,k,pid,status,childs;
  char term;
  childs=atoi(argv[1]);
  term=argv[2][0];
  for(i=0;i<childs;i++) {
     pid=fork();
     if(pid==0) {//codice figlio
        if(term=='1')
           exit(0);
     }
     else if(pid>0) {//codice padre
        printf("%d)child created(pid=%d)\n",
  getpid(),pid);
     }
     else perror("fork error:");
  }
}
```

Assenza della `exit()`



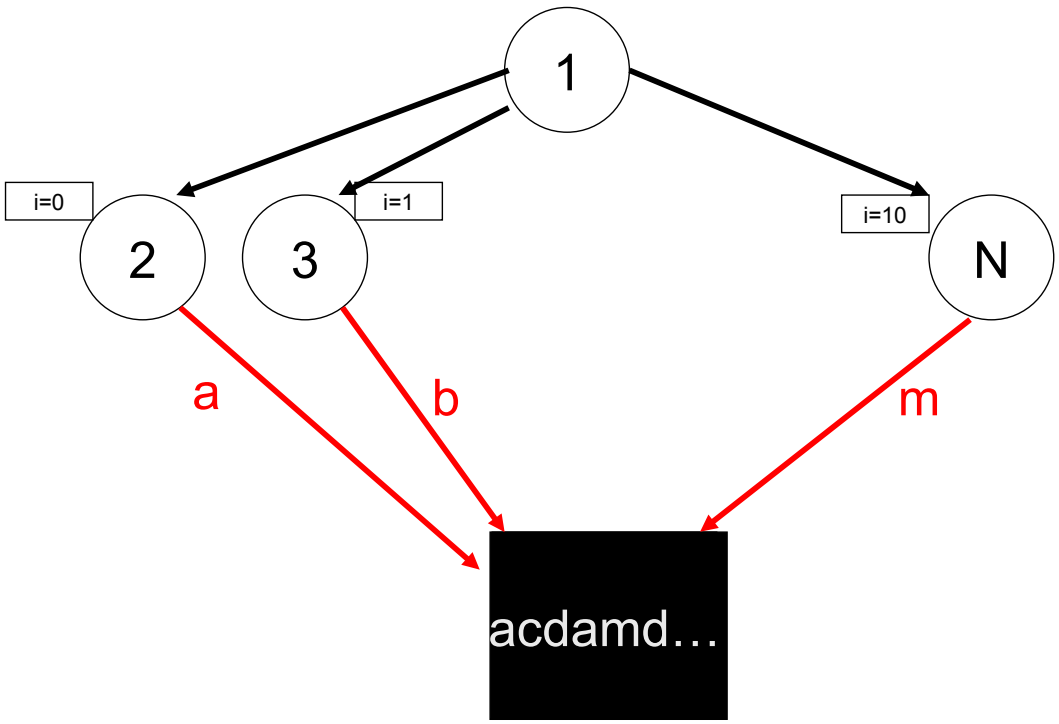
Esempio 2 - fork e wait

- Scrivere un programma in cui il processo padre procede alla istanziazione di un numero N di figli. Ciascun figlio è incaricato di stampare a terminale 10 volte una lettera dell'alfabeto, ottenuta scostandosi dalla lettera 'a' un numero di lettere pari all'ordine di generazione del figlio stesso. Si richiede un output ordinato: prima dovranno essere stampate le 10 'a', quindi le 10 'b' …
- Problema di sincronizzazione tra il padre e i figli
  - Quale grado di parallelismo?

Esempio 2

```
for(i=0;i<CHILDS;i++){
 pid=fork();
 if(pid==0){ //codice figlio
 c='a'+i;
 for(j=0;j<ITERATION;j++){
    printf("%c",c); fflush(0);
    for(k=0;k<ACTIVE_WAIT;k++); //Ciclo di attesa attiva
   }//for
    printf(" \n");
    exit(0);
} else if(pid>0) {//codice padre
   wait(&status);
} else perror("fork error:");
}
```


Assenza della `wait()`

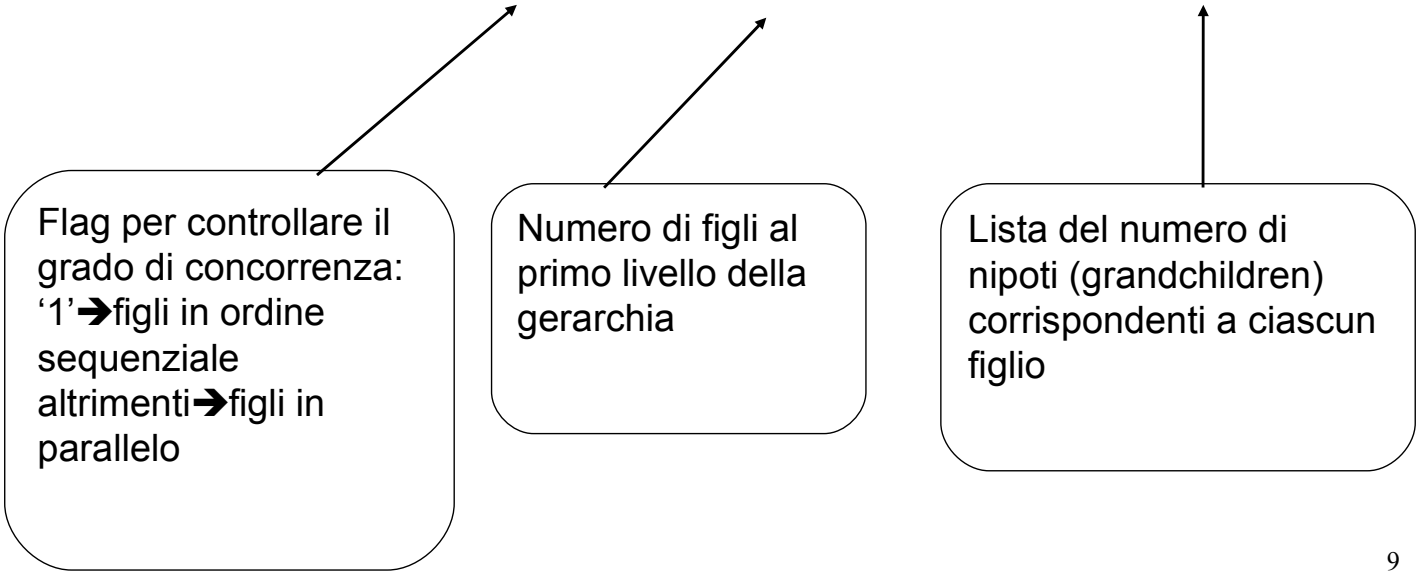


Standard output

Esercizio 1

- Scrivere un comando che abbia la sintassi

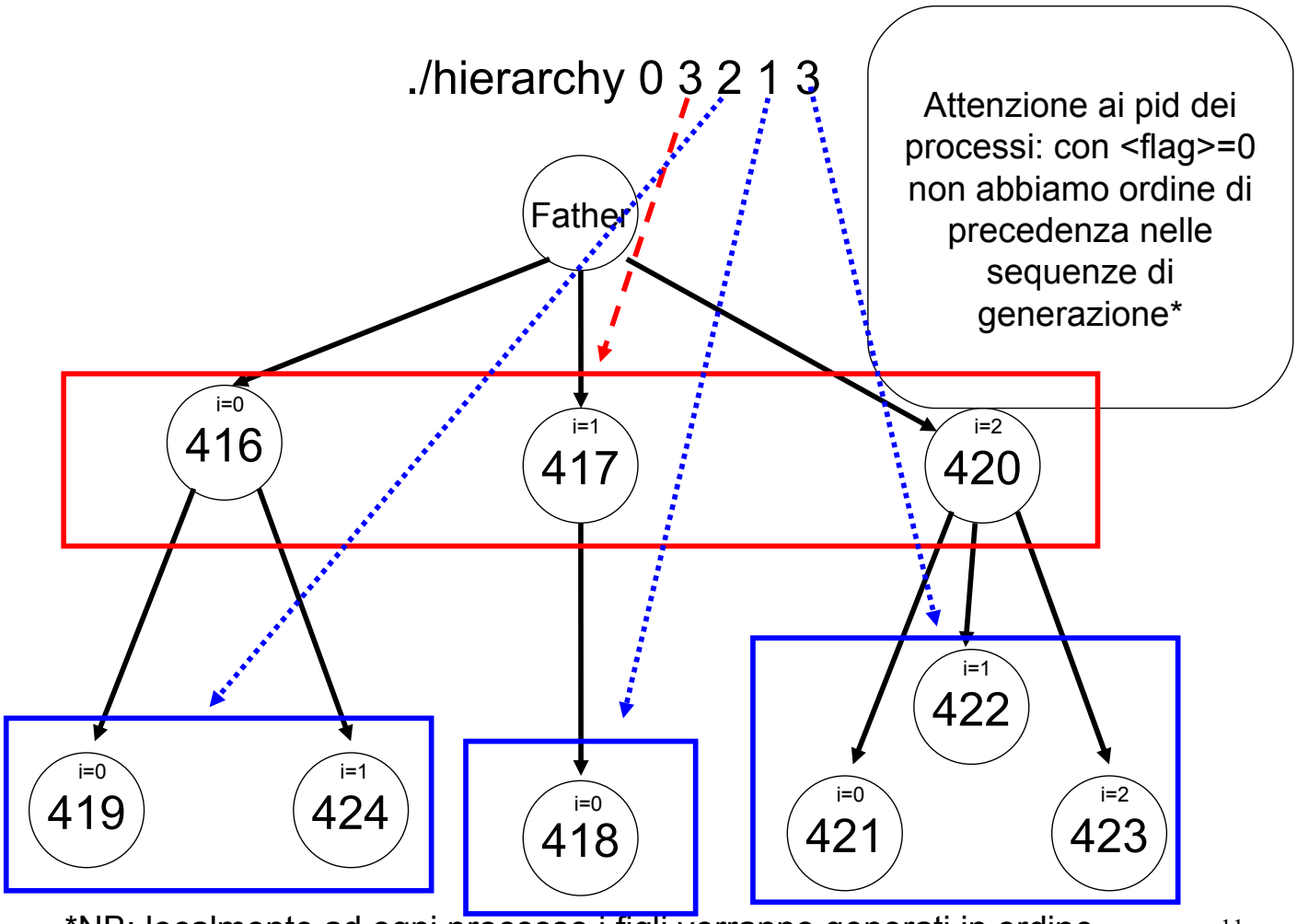
`hierarchy <flag> <#_cldr> <#_grandcldr>*`



Esercizio 1

In particolare

- Il processo padre genera un numero di figli pari al valore indicato dall'argomento <#_cldr>
- Ciascun figlio genera un numero di figli (nipoti del processo padre) pari al valore corrispondente nella sequenza individuata dagli argomenti <#_grandcldr>*
- Il parallelismo è determinato dal flag
  - " Come fa il padre a gestirlo?



*NB: localmente ad ogni processo i figli verranno generati in ordine comunque