ESERCITAZIONE 2

fork e exec

Richiami di scripting

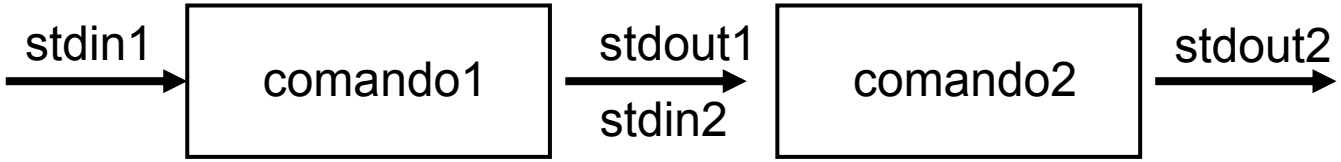
Ripassiamo:

- Differenza tra pipe e ridirezione su file
- Come si accede agli argomenti di un'invocazione
- Sintassi dell'**if**
- Sintassi del **for**

Pipe e ridirezione su file

Facciamo attenzione alla differenza tra **>>** e **|**

- **pipe**: fa apparire due comandi come uno solo, avente per stdin quello del primo, stdout quello del secondo. Il meccanismo della pipe esegue un collegamento fra lo stdout del comando a sinistra della pipe con lo stdin del comando alla destra.



- **ridirezione**: usata per riversare i dati prodotti da un comando (ad esempio la lista prodotta da ls) su un file:

ls > miofile	mette il risultato di ls nel file miofile senza presentare nulla a video
ls >> miofile	esattamente come sopra, con la differenza differenza che, se miofile esiste, l'output di ls viene aggiunto al termine del contenuto corrente

Argomenti

Si può accedere ai dati passati come argomenti mediante $, quindi se invoco

./prova.sh pippo 34 pluto

allora posso ottenere con:

- $0 → prova.sh
- $1 → pippo
- $2 → 34
- $3 → pluto

e così via…

Ricordatevi che con $# è possibile risalire al numero di argomenti passati allo script

Sintassi if

L'if ha una sintassi un po' particolare, differente da quella cui siamo abituati in C:

```
If [ condition ]
   then
      { statement }
fi
```

```
If [ condition ]
   then
      { statement }
else
      { statement }
fi
```

```
If [ condition1 ]
   then
      { statement }
elif [ condition2 ]
   then
      { statement }
else
      { statement }
fi
```

Sintassi for

Anche il for ha una sintassi differente da quella cui siamo abituati in C:

for i in <elements_list> do { statement } done	for i in 1 2 3 4 5 do echo $i done	*i* prende a ciascun ciclo il valore di uno dei valori contenuti nella lista di elementi e può essere usato come tale nello statement. Nell'esempio ad ogni iterazione viene stampato un item della lista: 1 2 3 4 5

Ancora uno script

Utilizzare lo script ps_monitor progettato la volta scorsa per creare un nuovo script che abbia la sintassi

```
./ps_user.sh [<user>]
```

Lo script deve mostrare le informazioni (tranne l'utente proprietario) relative ai tre processi meno recenti avente come proprietario <user>. Se <user> non è specificato, bisogna considerare sé stessi

Suggerimenti

- Generare l'output dello script passo passo, controllando i risultati parziali
- Solo se il risultato parziale conferma le aspettative, concatenare il nuovo comando in pipe
- Guardare il manuale di **`grep`** e **`cut`** per le selezioni/filtri su righe e colonne
- FARE MOLTA ATTENZIONE a come si imposta lo schema di selezione delle righe in uscita da ps_monitor…
  - Va filtrato il nome del proprietario
  - Va filtrato il nome completo (se indico "gian", è errato mantenere i processi di "gianluca")

Richiami (system call)

<table>
<tr><td><code>fork</code></td><td>Generazione di un processo figlio, che condivide il codice col padre e possiede struttura dati replicata</td></tr>
<tr><td><code>exit</code></td><td>Indicazione di terminazione di un processo, indicandone lo stato</td></tr>
<tr><td><code>wait</code></td><td>Raccoglimento dello stato di terminazione di un figlio; indica il pid del figlio che ha terminato la propria esecuzione e permette di capire come è terminato.
È terminato:
•Volontariamente? Con quale stato?
•Involontariamente? A causa della ricezione di quale segnale?</td></tr>
<tr><td><code>exec</code></td><td>Indicazione che il processo utilizzerà da questo momento in poi nuovo codice e una nuova struttura dati.
<table>
<tr><td><code>l</code></td><td>Gli argomenti del programma da caricare sono passati tramite una lista di parametri (terminata da NULL)</td></tr>
<tr><td>p</td><td>Il comando viene cercato all’interno della variabile d’ambiente (del processo) PATH</td></tr>
<tr><td>v</td><td>Gli argomenti del programma da caricare sono passati tramite un vettore</td></tr>
<tr><td>e</td><td>Permette di indicare il nuovo ambiente in cui eseguirà il processo</td></tr>
</table></td></tr>
</table>

Obiettivo 1/2

Scrivere un programma C che, DA UN UNICO PROCESSO PADRE, attivi su richiesta dell'utente uno dopo l'altro dei processi figli, dedicati ciascuno all'esecuzione di un comando CON PARAMETRI

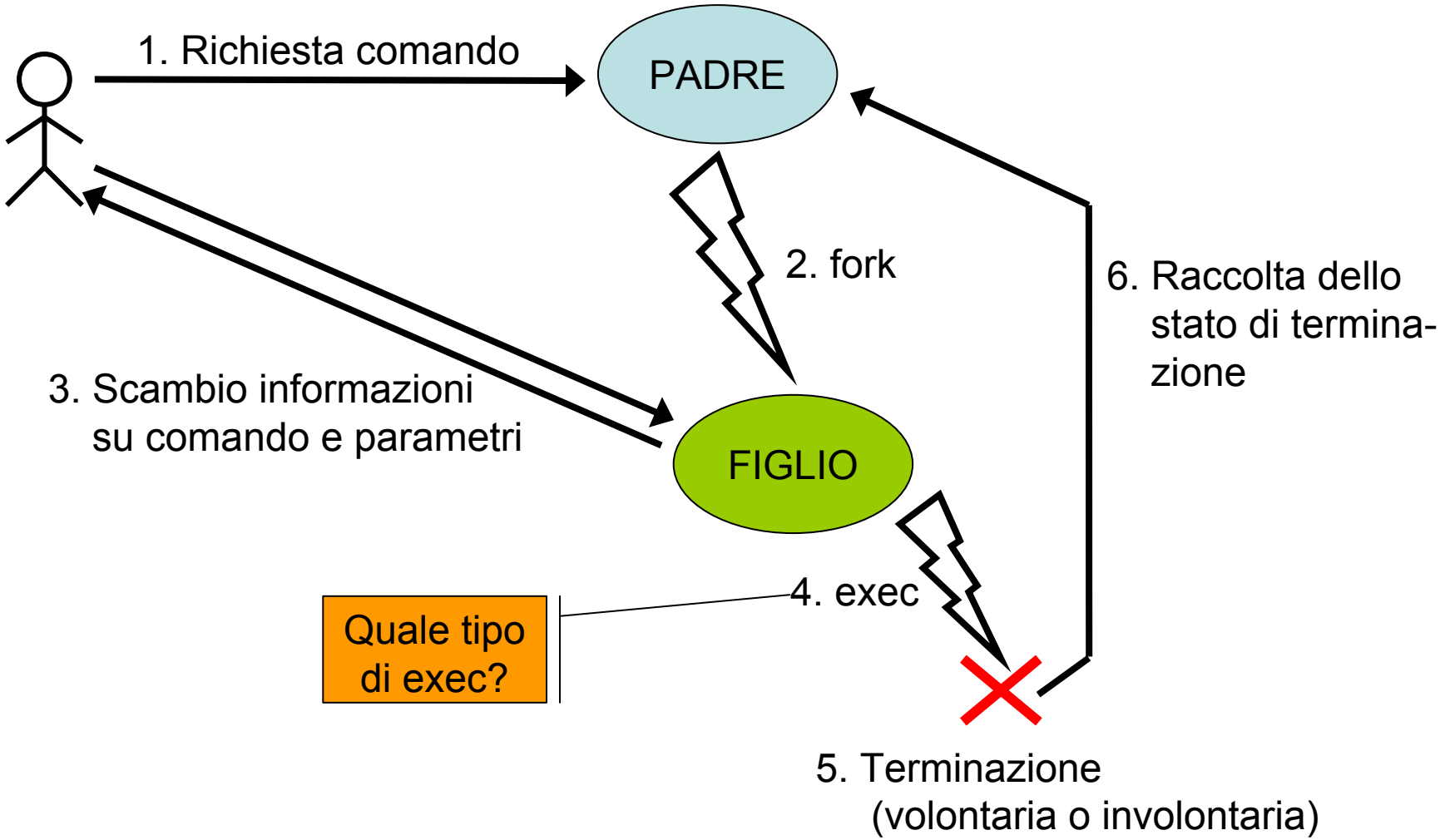
Obiettivo 2/3

- In particolare, questo programma deve:
  - iniziare un ciclo
  - all'interno del ciclo, attendere l'inserimento di un comando e dei parametri (questo può essere fatto in uno o più passi)
    - Per semplicità, fissare un numero massimo di parametri
  - una volta completato l'inserimento di un comando, mandarlo in esecuzione
  - una volta terminata l'esecuzione del comando, raccogliere la terminazione di chi l'ha eseguito, visualizzando lo stato in caso di terminazione volontaria, il segnale altrimenti
  - Ripetere il ciclo

Obiettivo 3/3

- Provare a far eseguire al figlio uno degli script progettati alla scorsa esercitazione
- Provare ad inviare un segnale, da (un altro) terminale, al processo che sta eseguendo il comando (es.: **`sleep`**)
  - Vedere come risponde il processo in risposta alla ricezione del segnale inviatogli

Schema di una richiesta



Schema di soluzione

```
main()
{
 char c;
 for(;;)
 {
  fflush(0);
  printf("\nun altro comando?[s/n]");
  scanf("%c", &c);
  if( c == 's' )
  {
   pid=fork();
   if( pid == 0 )
   {
    FIGLIO
   }
   else
   {
     PADRE: ATTENDI LA
     TERMINAZIONE DEL FIGLIO
   }
  }
  else if ( c == 'n' )
  {
   // termina
   break;
  }
  // continua nel ciclo
 }
exit(0);
}
```