

0.1 Uso della chiamata POSIX `fork`

Innanzitutto è opportuno includere le definizioni POSIX.

1a $\langle \textit{Header POSIX 1a} \rangle \equiv$ (2a)
`#include <unistd.h>`

In realtà sarebbe sufficiente il $\langle \textit{Prototipo di fork 1b} \rangle$.

1b $\langle \textit{Prototipo di fork 1b} \rangle \equiv$
`pid_t fork(void);`

Definisce:

`fork`, usato nella porzione 1.

1c $\langle \textit{Altri header 1c} \rangle \equiv$ (2a)
`#include <stdio.h>`
`#include <stdlib.h>`

L'uso della `fork` è ingannevolmente semplice. L'effetto di una chiamata `fork` è la creazione in memoria di un processo pressoché identico a quello chiamante, con l'eccezione del valore di ritorno (in questo caso `x`), che assume valori differenti nel *padre* (sarà il `pid` del figlio) e nel *figlio* (sarà 0).

1d $\langle \textit{Uso di fork 1d} \rangle \equiv$ (2a)
`int x = fork();`

Definisce:

`x`, usato nella porzioni 1 e 2.

Usa `fork 1b`.

La `fork` restituisce un valore negativo nel caso fallisca e quindi non venga creato alcun processo figlio.

1e $\langle \textit{Errore nella fork 1e} \rangle \equiv$ (2a)
`if (x < 0){`
`perror("Errore nella fork:");`
`exit(1);`
`}`

Usa `fork 1b` e `x 1d`.

Si noti che le attività di padre e figlio sono scritte nello stesso programma e flusso di esecuzione: per questo motivo l'esecuzione delle rispettive istruzioni sarà tipicamente condizionato.

2a $\langle \text{forca.c 2a} \rangle \equiv$
 $\langle \text{Header POSIX 1a} \rangle$
 $\langle \text{Altri header 1c} \rangle$

```
int main(void){
   $\langle \text{Uso di fork 1d} \rangle$ 
   $\langle \text{Errore nella fork 1e} \rangle$ 
   $\langle \text{Attività del padre 2c} \rangle$ 
   $\langle \text{Attività del figlio 2b} \rangle$ 
  return 0;
}
```

Definisce:

main, mai usato.

Nel figlio x è 0: al contrario di quanto sostenuto nel famoso brocardo latino, in un sistema POSIX: *pater semper certus est*.

2b $\langle \text{Attività del figlio 2b} \rangle \equiv$ (2a)
else { $/x == 0$
 while(1) printf("Processo figlio ($x == \%d$)\n", x);
 }

Usa x 1d.

Nel padre x è il pid del figlio.

2c $\langle \text{Attività del padre 2c} \rangle \equiv$ (2a)
if ($x \neq 0$){
 while(1) printf("Processo padre ($x == \%d$)\n", x);
 }

Usa x 1d.

0.2 Indici

0.2.1 Porzioni di codice

$\langle \text{Altri header 1c} \rangle$
 $\langle \text{Attività del figlio 2b} \rangle$
 $\langle \text{Attività del padre 2c} \rangle$
 $\langle \text{Errore nella fork 1e} \rangle$
 $\langle \text{forca.c 2a} \rangle$
 $\langle \text{Header POSIX 1a} \rangle$
 $\langle \text{Prototipo di fork 1b} \rangle$
 $\langle \text{Uso di fork 1d} \rangle$

0.2.2 Identificatori

fork: 1b, 1d, 1e

main: 2a

x: 1d, 1e, 2b, 2c