Università di Salerno - Dipartimento di Informatica
Tutorato di Sistemi Operativi a.a. 2017/2018
Federico Vastarini

Esempi dell'utilizzo della funzione fork()

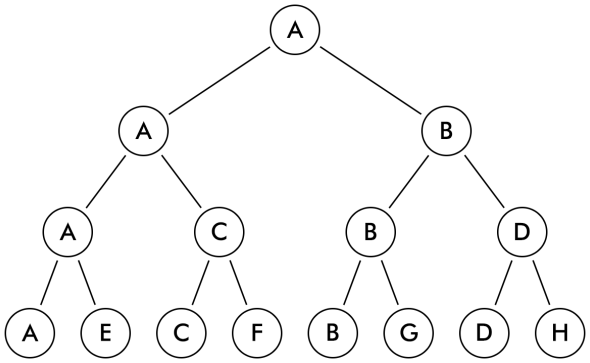
fork_01.c

Semplice esempio della funzione fork().
Un processo viene duplicato e si mostra come al seguito della duplicazione venga assegnato un nuovo process id (pid) al processo figlio.
Si ricorda che non c'è nessun ordine specifico in cui i due processi vengano eseguiti dopo la chiamata della funzione fork().

La funzione getpid() restituisce un intero rappresentante l'id del processo dal quale viene chiamata.

fork_02.c

In questo esempio viene creato un albero binario di processi.
Il primo processo A si duplica in un processo B e continua la sua esecuzione.
Al secondo passo entrambi i processi vengono duplicati secondo il seguente schema, ma non necessariamente in querst'ordine:



Da notare che:

1. Ai processi orfani, quelli di cui il nodo "parent" (il processo che li ha generati con la fork()) è terminata l'esecuzione, verrà assegnato un parent process id (ppid) = 1. Ovvero diventeranno "figli" del processo init. Questo potrebbe non accadere in alcuni sistemi operativi dove i processi orfani vengono assegnati ad uno scheduler speciale e modificato il ppid a `0`.
2. Man mano che incrementiamo il parametro n i processi iniziano ad assumere un ordine sempre più "casuale" tra un'esecuzione e l'altra. Ciò accade poiché non è stato stabilito nessun ordine particolare in cui eseguire prima l'uno o l'altro processo.
3. Inoltre se aumentiamo abbastanza n noteremo che alcuni processi figli mantengono ancora il ppid del padre. Questo accade poiché avendo un ordine casuale è possibile che il processo padre sia ancora impegnato a creare altri processi figli al momento della chiamata getppid() (con la quale si ottiene il process id del processo padre) di un figlio precedente.
4. Il parent process id del primo processo (ppid) è il pid della shell che lo ha creato. L'esecuzione tramite shell non è altro infatti che una combinazione di fork() ed exec() della shell stessa.

La funzione getppid() restituisce un intero rappresentante l'id del processo che ha generato quello dal quale viene chiamata.

fork_03.c

fork_03 utilizza il valore restituito dalla funzione fork() per distinguere il resto dell'esecuzione tra il parent process ed il child process.

La funzione fork() restituisce infatti il valore del processo figlio al processo padre e `0` al processo figlio. Nel codice che segue la funzione è quindi possibile distinguere in quale dei due processi ci si trova grazie ad un semplice if.

fork_04.c

In questa versione del programma, una volta effettuata la fork() nel ciclo while, si termina il processo parent in modo che soltanto il processo child possa continuare a ciclare. Ciò porta alla

generazione di esattamente n nuovi processi e consente anche di eseguirli nell'ordine esatto della rispettiva creazione.

Il processo child rientrerà infatti nel ciclo questa volta come processo parent ed eseguita la fork() ne genererà un altro nuovo, uscendo a sua volta dal ciclo.

Per eseguire i processi nell'ordine inverso è necessaria invece la funzione wait() da associare alla fork().

fork_05.c

In quest'ultimo esempio si evidenzia come una volta eseguita la fork, alla variabile x verranno attribuiti due nuovi indirizzi.

Anche se il valore dell'inidirizzo può sembrare lo stesso in realtà questo accade soltanto per l'indirizzo virtuale, in quanto l'indirizzo fisico al quale fa riferimento dipende dal processo. Quindi lo stesso valore utilizzato dal processo parent punterà ad un indirizzo logico differente per il processo child. In questo modo si creeranno due differenti variabili x che verranno gestite indipendentemente dai due processi. Lo "sdoppiamento" dell'indirizzo fisico rispetto a quello logico avviene soltanto nel momento in cui si accede alla variabile in scrittura.