Laboratorio in C su Processi e POSIX IPC (Inter Process Communications)

Decima lezione di laboratorio

NOTA BENE: E INSISTO !!!!!

Usare il comando **man** nomefunzionedilibreria per ottenere informazioni sull'uso di una specifica funzione di libreria standard del linguaggio C e, in generale, informazioni del man.

In caso che uno stesso nome di funzione sia presente in piu' sezioni del man, occorrera' specificare il **numero della sezione** del man a cui fare riferimento, ovvero:

1. Executable programs or shell commands
2. System calls (functions provided by the kernel)
3. Library calls (functions within program libraries)
4. Special files (usually found in /dev)
5. File formats and conventions eg /etc/passwd
6. Games
7. Miscellaneous (including macro packages and conventions), e.g. **man(7), groff(7)**
8. System administration commands (usually only for root)
9. Kernel routines [Non standard]

Ad esempio, per ottenere informazioni sull'uso della funzione **open** per i file o della **waitpid** :

man 2 open **man 2 waitpid**

Ad esempio, per ottenere informazioni sull'uso della funzione **fopen** per i file :

man 3 fopen

Esercizio 21: es21_memcondivisafiglio.c

creazione di processo figlio e uso di memoria condivisa

Scrivere un programma, in linguaggio C, che effettua le seguenti operazioni:

il processo principale crea un segmento di memoria, condivisibile tra piu' processi e di dimensione 512 byte, e lo inizializza riempiendolo di caratteri 'B' con uno '\0' in fondo.

Successivamente il processo crea un processo figlio, il quale deve modificare il contenuto del segmento di memoria condiviso riempiendolo di caratteri 'K' con uno '\0' in fondo, e infine terminare se stesso. In caso di qualche errore il processo figlio restituisce un codice di errore diverso da 0, se tutto va bene il processo figlio termina restituendo 0.

Il processo padre deve attendere che il processo figlio abbia completato il suo compito, controllare che il figlio sia terminato normalmente e verificare il codice di errore restituito dal figlio.

Se il figlio ha restituito codice di errore 0 allora il padre deve stampare il contenuto del segmento di memoria condivisa. Se invece il figlio ha restituito codice di errore diverso da 0 il padre deve stampare un avviso contenente il codice di errore ottenuto.

Infine, il processo padre deve eliminare quel segmento condiviso.

Si consiglia di utilizzare, come esempi da cui prendere spunto, i files fork.c e acrossProcessBoundary.c

Esercizio 21bis: modifica minimale del programma precedente

Nel figlio, aggiungere una attesa di 60 secondi con una sleep(60);

Durante quei 60 secondi, in un'altra shell usare il comando ps per scoprire il process identifier del processo figlio.

Uccidere il processo figlio utilizzando il comando kill -9 PIDFIGLIO

Verificare come si comporta il processo padre.

Esercizio 22: es22_passaggioparametriafiglio.c

creazione di molti processi figli e passaggio di parametri sfruttando la duplicazione dello spazio di memoria

Scrivere un programma, in linguaggio C, che effettua le seguenti operazioni:

il processo principale deve creare 10 processi figli, passando a ciascuno di questi un indice intero univoco, crescente da 0 a 9.

I figli stampano l'indice e terminano.

Terminata la creazione, il processo padre deve attendere la terminazione dei 10 figli, in un ordine qualunque.

Suggerimento: Non create un segmento di memoria condivisa per passare delle informazioni a ciascun processo figlio, non serve, c'e' un modo decisamente piu' semplice, addirittura banale.

Esercizio 23: es23_staffetta.c

passaggio di testimone tra pthread

Il main di un programma crea 20 pthread che eseguono la funzione staffetta. Poi si mette in attesa della terminazione dei 20 pthread. Fatto questo il main crea altri 20 pthread e si rimette in attesa della loro terminazione, e cosi' via all'infinito.

La funzione staffetta, operando in mutua esclusione, deve assegnare, ad una variabile globale float di nome **fglob** inizializzata ad 1111, il valore della radice quadrata della variabile fglob stessa.

Dopo l'assegnamento, la funzione deve aspettare fino a che un totale di 5 pthread hanno incominciato l'esecuzione della funzione staffetta ed effettuato l'assegnamento. Ogni volta che 5 pthread hanno effettuato l'assegnamento allora i 5 pthread possono proseguire e terminare la loro esecuzione.

Esercizio 24: un po' di preliminari con la bash

Vedere cosa succede lanciando il comando

bash -c ps aux

Vedere cosa succede, invece, lanciando il comando (notare i due apici singoli)

bash -c 'ps aux'

Esercizio 25: es25_zombie.c

eseguire in un figlio un comando bash per vedere i figli Zombie

Scrivere un programma, in linguaggio C, che effettua le seguenti operazioni:

il processo principale deve creare 10 processi figli, passando a ciascuno di questi un indice intero univoco, crescente da 0 a 9. I figli stampano l'indice e terminano.

Terminata la creazione, il processo padre deve attendere la terminazione di **9 qualunque dei suoi 10 figli**, e stampare il codice di errore restituito da ciascun figlio.

Successivamente il processo padre deve attendere due secondi e creare un nuovo processo a cui far eseguire il comando (notare gli apici)

bash -c ' ps aux | grep **nomeeseguibileprocessopadre** '

dove **nomeeseguibileprocessopadre** e' il nome del vostro eseguibile

Fatto questo, il processo padre deve attendere la terminazione degli ultimi due suoi processi figli ed, infine, terminare.

Guardare l'output del comando ps e riconoscere i figli zombie.

SOLUZIONI

Esercizio 21

http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es21_memcondivisafiglio.tgz

Esercizio 22

http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es22_passaggioparametriafiglio.tgz

Esercizio 23

http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es23_staffetta.tgz

Esercizio 24

il comando **bash -c** usa il solo argomento che segue il **-c** come comando da eseguire.

Quindi se scrivo bash -c ps aux viene eseguito solo ps senza argomenti, che visualizza solo i processi lanciati dalla bash corrente.

Invece, se scrivo bash -c 'ps aux' viene eseguito il comando ps aux cioè ps con argomento aux, che visualizza tutti i processi.

Esercizio 25

http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es25_zombie.tgz