Sistemi Operativi – Lab 3 03.11.15 - A.A. 2015/2016 - Prof. L. Sterpone

Goal: Utilizzo delle system call fork, wait e waitpid.

Esercizio 1 (system call fork e comandi di shell per processi):
Un programma riceve due valori interi sulla riga di comando, n e t. Il programma (processo padre) deve eseguire 2 figli e terminare. Ciascun figlio deve eseguire a sua volta due figli e terminare. Il procedimento deve andare avanti sino a quando sono in esecuzione i 2^n processi sulle foglie dell'albero. Questi ultimi aspettano per t secondi e visualizzano (a video) un messaggio di terminazione. Si noti che ogni processo (ad albero) ne esegue altri due. Solo quelli sulle foglie dell'albero dormono e visualizzano un messaggio.

Qual è l'ordine di terminazione dei processi? E' sempre lo stesso? Come si possono riconoscere (ppid)?
Una volta eseguito il programma:
A. cercare il pid dei vari processi con il comando ps
- dalla stessa shell
- da un'altra shell

B. terminarne l'esecuzione di alcuni processi con il comando kill eseguito
- dalla stessa shell
- da un'altra shell.

Esercizio 2 (system call fork e wait): Si desidera scrivere un programma C che, utilizzando n-1 processi che si sincronizzano con la system call wait, visualizzi tutti gli elementi di un vettore.
Piu' in dettaglio, il programma deve effettuare le seguenti operazioni:
1. ricevere sulla riga di comando un valore intero n.
2. allocare dinamicamente un vettore di interi di dimensione n, leggere da tastiera n valori interi e memorizzarli nel vettore.
3. visualizzare (a video) gli elementi del vettore a partire dall'elemento n e terminando con l'elemento 0.
Questa operazione deve essere fatta clonando il processo n-1 volte.
Ogni processo clone visualizza un solo elemento del vettore e si sincronizza con gli altri processi mediante la system call wait in modo da visualizzare gli elementi nell'ordine voluto (n-1, n-2, ..., 0).

Suggerimento
Il processo padre P1 crea un processo figlio P2 e si mette in sua attesa.
Il processo padre P2 crea un processo figlio P3 e si mette in sua attesa.
...
Visualizza l'elemento n-1 del vettore e termina.
Visualizza l'elemento n-2 del vettore e termina.
...
Visualizza l'elemento 0 del vettore e termina..

Esercizio 3 (system call fork e waitpid): Si scriva un programma in linguaggio C in grado di generare un processo padre che crea N figli e aspetta la fine della loro esecuzione. Modificare il programma per fare in modo che il processo padre aspetti solo gli M < N ultimi figli creati.

Suggerimento per la prima parte: si utilizzi la waitpid con parametro -1 e nessun parametro macro (i.e., waitpid(-1,&status,0). Utilizzare la macro WIFEXITED ed identificare il valore di terminazione del processo figlio con WEXITSTATUS.