Séance de TP N°3
03. .processus - Processus

Créez votre répertoire de TP *~/l3i6/03.processus* et placez-vous dessus. Récupérez dans votre répertoire de TP le fichier *Fourniture03.tgz* sur le site de cet enseignement[1] et déballez-le. Ce répertoire contient :

- un ***Makefile.gen*** pour tous les exercices demandés,
- le script ***configure*** à exécuter pour produire un ***Makefile*** adéquate,
- ***print_exit.c*** pour imprimer le mot de statut rendu par les instructions *wait*,
- ***t_fork_wait.res*** un exemple de résultats à analyser.

Création de processus: version1 avec écritures directes.

Pour éviter les problèmes de tampons d'E/S non vidés à temps, toutes les écritures des messages se feront par un appel direct à *write*. Mais pour bénéficier des possibilités de formatage des messages, on utilisera la primitive *printf* de la bibliothèque C, mais en mode non tamponné : chaque appel de *printf* videra le tampon en appelant automatiquement *write*. Cela s'obtient en exécutant la primitive *setvbuf(3)* (*cf.* son manuel) :

setvbuf(*stdout*, **NULL**, *_IONBF*, 0);

Écrire un programme ***t_fork_wait*** qui crée deux processus fils, ***fils_1*** et ***fils_2*** et un processus ***petit_fils_1***, fils de ***fils1*** et petit-fils du processus père de départ. Chacun des quatre processus aura son code localisé dans une procédure sans argument nommée *pere*, *fils_1*, *fils_2* et *petit_fils1*. La routine *main* activera la routine *pere*. Toutes les impressions se feront sur *stdout* avec *printf*. Toutes les attentes se feront avec la routine *sleep*.
Le processus père créera ses deux fils qui exécuteront leur code placé dans les routines *fils_1* et *fils_2*. Puis le père attendra un délai de *DELAI_P* secondes, et attendra la terminaison de ses deux fils, en affichant leur identité (*FILS_1* ou *FILS_2*), leur *PID* et leur statut de terminaison par la procédure *print_exit* fournie.
Le processus ***fils_1*** affichera un message de départ indiquant son *PID* et celui de son père, créera son fils qui exécutera son code placé dans la routine *petit_fils_1*. Puis *fils_1* attendra un délai de *DELAI_F1* secondes et, sans attendre la fin de ***petit_fils_1*** affichera un message de terminaison indiquant à nouveau son *PID* et celui de son père. Il se terminera par une erreur de division par zéro précédant un *exit* de code 1. Le processus ***fils_2*** affichera un message de départ indiquant son *PID* et celui de son père, attendra un délai de *DELAI_F2* secondes, affichera un message de terminaison indiquant à nouveau son PID et celui de son père et terminera par un *exit* de code 2. Le processus ***petit_fils_1*** affichera un message de départ indiquant son *PID* et celui de son père, attendra un délai de *DELAI_PF1* secondes, affichera un message de terminaison indiquant à nouveau son *PID* et celui de son père et terminera par un *exit* de code 11.
Tester le programme ***t_fork_wait*** avec différents jeux de valeurs de délais pour respectivement le père (*DELAI_P*), le ***fils_1*** (*DELAI_F1*), le ***fils_2*** (*DELAI_F2*) et le ***petit_fils1*** (*DELAI_PF1*)

- 1, 1, 1, 1 secondes ;
- 1, 3, 3, 1 secondes ;
- 1, 1, 1, 3 secondes ;
- 3, 1, 1, 1 secondes.

Chaque expérience sera exécutée plusieurs fois. Analyser les résultats. Ceux-ci doivent faire apparaître :

- la création de processus orphelin ou zombie,
- un ordonnancement aléatoire de l'exécution des quatre processus, même avec les mêmes délais.

[1] http://deptinfo.unice.fr/~lahire/enseignement/SYSL3

Remarque. Pour ne pas avoir à modifer les valeurs des constantes de délai dans le code, pour chaque expérience, on peut donner ces valeurs sur la commande de compilation, comme dans le ***Makefile*** fourni. Mais dans ce cas, il ne faut définir ces constantes dans le programme que si la commande de compilation ne les définit pas, sinon, c'est la définition dans le programme qui l'emporte. On placera donc des directives ***#define DELAI...*** qu'entourées de directives ***#ifndef ... #endif***.
Analysez les résultats en traçant sur une feuille de papier les diagrammes de temps des différents processus.

Création de processus: version2 avec écritures tamponnées.

Modifiez le programme précédent en demandant un tamponnage forcé des écritures par *printf*.
Rappel : par défaut, *printf* vide son tampon à chaque fin de ligne si la voie *stdout* est associée à un terminal et utilise un tampon de plusieurs milliers de caractères si cette voie est associée à un disque.
On obtient un tamponnage forcé en utilisant setvbuf :

```
#define BUFSIZE 10000
char buf[BUFSIZE];
setvbuf(stdout, buf, _IOFBF, BUFSIZE);
```

Recommencez les expériences précédentes et analysez les résultats.