R3.05 - TP 3
Création de Processus (Suite)

Les fonctions exec()

Mise en œuvre

Avec les fonctions de la famille **exec**, un processus peut remplacer son code par celui d'un autre programme.

Cela signifie qu'après avoir appelé une des fonctions **exec** de façon appropriée, le processus verra son code remplacé par celui d'un exécutable dont le chemin sur le FS est passé en paramètre et que le processus poursuivra ensuite son exécution par le **`main()`** de cet exécutable. Donc, s'il n'y a pas d'erreur avec l'appel à une fonction **exec**, cet appel ne retournera jamais, car le code où était cet appel n'existe plus, remplacé par un autre code ! Ca peut paraître bizarre mais c'est ainsi que les processus engendrent d'autres processus pour exécuter d'autres binaires que le leur.

C'est notamment ainsi qu'un shell exécute des commandes : le Shell se **`fork()`** lui-même en un autre processus dont le code est immédiatement remplacé par celui de la commande que l'utilisateur souhaite exécuter (c'est ce qu'il a demandé de faire au Shell en tapant une commande au prompt).

Note : pour "appeler une des fonctions **exec** de façon appropriée", il faut voir le **`man`** ou l'enseignant.e mais certainement ni Google ni Stackoverflow !

Phase 1

Voici ce que vous devez réaliser dans un 1er temps.

Votre programme principal doit :

- **`fork()`**er un processus
- Dans le père : boucler sans fin sur un **`sleep(60)`** et à chaque passage afficher **`Passage num X`** où **`X`** est la valeur de la variable de boucle.
- Dans l'enfant : utilisez un **`execl()`** pour remplacer le code du programme par celui d'un autre programme (par exemple le programme **`xeyes`**). Voir la syntaxe de **`execl()`** ci-après.

Explications de **`execl()`** :

Avant tout : **`man execl`** et lisez un peu. **`execl`** est une famille de fonctions système. Ici, nous allons simplement utiliser un **`execl()`** de base.

Vous vous rappelez certainement le contenu du **`argv[]`** d'un **`main()`** ?

Dans le **`execl()`** vous avez un 1er paramètre qui est le chemin vers le programme dont le code va prendre la place, en mémoire, du code de votre programme actuel qui exécute le **`execl()`** ! Relisez le début de ce TP si besoin, ou appelez à l'aide (localement, pas sur Internet SVP)

Puis, les paramètres suivants représentent tout ce que va contenir le **`argv[]`** (y compris **`argv[0]`**). A vous de remplir tous les arguments (**`man xeyes`** pour savoir ce qu'il faudrait passer comme paramètre pour le lancer, s'il y en a besoin). N'oubliez pas, vous devez préciser à partir du 2ème paramètre de **`execl()`**, tout ce qui va rentrer dans **`argv[]`**.

Phase 2

Modifier votre programme ainsi, après le **`fork()`**, dans le père :

- Mettre en place un traitement des signaux **`SIGCHLD`**
- Sur réception de **`SIGCHLD`**, "attendre" le processus qui vient de mourir, puis en relancer un nouveau de façon identique (c'est-à-dire dans un autre processus **`fork()`**é, comme à la Phase 1), et retourner dans la boucle sans fin.

Phase 3

Modifier votre programme ainsi, après le **`fork()`**, dans le père :

- Mettre en place un traitement des signaux **`SIGUSR1`**
- Sur réception de **`SIGUSR1`**, tuer l'enfant et quitter le programme

Test

Testez votre programme qui doit afficher une fenêtre avec des yeux qui suivent la souris.

Tuez le processus enfant avec un **`kill`** dans une fenêtre de Terminal, ça doit recréer immédiatement une nouvelle fenêtre "yeux".

Envoyez un signal **`SIGUSR1`** au processus parent qui doit arrêter son fils et s'arrêter lui-même ensuite.

Affichez des messages à l'écran pour décrire tous ces événements.

Encore plus fort !

Serez-vous capable de gérer 5 fenêtres "yeux" en même temps, au lieu d'une seule ?