LAB4 - La gestion des processus

Table des matières

By R

Objectifs

Ce « LAB » présente les notions de base de gestions de processus. Il est basé sur la mise en parallèle de notions relatives aux commandes Shell (et Scripts Shell) avec leur équivalent en langage C.

- Documents : «Linux : la gestion des processus ».
- Notions abordées : le « scheduler », les signaux, la gestion synchrone et asynchrone, /proc
- Commandes et fichiers exploités : ps, pstree, top, trap, kill, fork(), exec().
- Travail à rendre : Vous devrez répondre directement à plusieurs questions au sein de ce document. Vous le copierez sur Moodle sous le nom : « LAB3_noms.pdf ».

Gestion des processus

.1 Shell

Lisez le document « Linux – gestion des processus »
Répondez aux questions suivantes en utilisant **une couleur distincte**.

- Dans un terminal : lancez un éditeur de texte simple (« gedit », « Vim », « Emacs », etc.). En se lançant, votre éditeur affiche (ou pas...) des informations sur le canal des erreurs. Comment feriez-vous pour qu'il n'affiche plus ces erreurs ? (commande : gedit 2>dev/null). Testez. Info : il est toujours intéressant de lancer les applications graphiques à partir d'un terminal, car cela permet de voir des dysfonctionnements souvent cachés par les interfaces graphiques (idem sous Windows).
- Pourquoi n'avez-vous plus la main sur votre terminal ? (__car il n'est pas exécuté en arrière-plan avec l'esperluette &). Lancez en aveugle la commande « id ». Fermez votre éditeur via ses menus graphiques. Cela est-il normal ? (_oui c'est normal_). Relancez-le de manière asynchrone et sans affichage des messages d'erreur (commande : gedit & 2> /dev/null).
- Définissez les 13 champs affichés par la commande « ps -faxl » (« F » : 0, « UID » :_1000_____, « PID » :2297, <<PPID>> :2159,<<PRI>> :20, <<NI>> :0,<<VSZ>> :21344 ,<<RSS>> : 3836 , <<WCHAN>> :-, <<STAT>> :R+,<<TTY>> :pst/1,<<TIME>> :0 :00,<<COMMAND>>:_ ps -faxl). Quels sont le PID et le PPID de votre éditeur ? (_2291 et 2159). Quel est le processus PPID ? (un code unique attribué sur les systèmes Unix ou Windows à tout processus lors de son démarrage. Il permet ainsi d'identifier le processus dans la plupart des commandes s'appliquant sur un processus donné)
- Affichez la liste des signaux disponible sur votre système (kill -l_). En tant que père de votre éditeur, votre shell peut lui envoyer des signaux. Fermer votre éditeur en lui envoyant le signal d'interruption « SIGINT » (commande : kill -2 2279_).
- Le PPID d'un shell correspond au processus qui gère le terminal et les entrées clavier. Quel est-il sur votre système ? (2332). Quelle commande permet de connaître les caractéristiques du terminal : (_ps -ejf |grep terminal). En analysant cette commande, vous pourrez connaître les séquences de touches associées à certains envois de signaux. Quelle séquence pour le signal SIGINT ? (kill -SIGINT [pid]). Testez.
- Lancez votre éditeur de manière synchrone (vous n'avez donc plus la main). Lancez-lui le signal de suspension (SIGSTOP) de deux manières différentes (1 : `kill -STOP pid`) (2 :CTRL + Z). Vous reprenez ainsi la main. Votre éditeur est bien toujours présent, mais comme le processus ne passe plus dans la boucle du microprocesseur, il est devenu inerte (sans activité). Vérifiez. Vous pouvez le réactiver en lui envoyant le signal (SIGCONT) de deux manières différentes (1 :kill -SIGCONT 5892) (2 :kill -18 5892). Testez. Vous pouvez réactiver tous les processus stoppés via la commande « bg » (background). INFO : vous connaissez maintenant un bon moyen de reprendre la main sur votre terminal quand vous lancez un processus graphique.
- Qu'arrivera-t-il à votre éditeur si son père meurt ?(l'éditeur ne fonctionne plus). Testez en fermant le terminal. Rouvrez un terminal et lancez-le de manière asynchrone afin qu'il ne dépende plus de son père en cas de mort subite de ce dernier (commande :nohup gedit). Vérifiez en fermant le terminal. Qui est devenu PPID de votre éditeur ? (le numéro du PPID change).

- Les signaux sont traités par les processus via les standards de programmation. Le « shell » est un processus comme un autre. Ses programmeurs ont prévu que l'utilisateur puisse intercepter les signaux afin de modifier l'action prévue normalement. La commande « trap » qui est interne au « shell » permet ainsi d'intercepter et de reprogrammer l'action des signaux. Ainsi, dans un « shell », la commande « trap 'ls' nro_signal » permet d'intercepter le signal numéro 'nro_signal' quand il est reçu afin de lancer la commande « ls » au lieu du traitement normalement prévu. À quelle séquence de touche correspond le signal « SIGINT » ? (ctrl + c). Reprogrammer l'action liée à ce signal afin de lancer la commande « ps ». Testez. Comment pouvez-vous réinitialiser cette programmation à sa valeur par défaut ? (_____).
- Créez un « script shell » qui affiche la liste des usagers de votre système. Cette liste est dans le fichier « /etc/passwd » (vous pouvez judicieusement exploiter la commande « cut »). Dans un terminal (et donc un shell), modifiez le comportement du signal numéro 3 afin de lancer votre script. Ouvrez un deuxième terminal. Déterminez le PID du premier terminal et envoyez-lui le signal numéro 3. Faites une capture d'écran des 2 terminaux. Supprimez l'interception du signal numéro 3.
- Quand un processus reçoit le signal Nro 1 (SIGHUP), il se ferme normalement (HUP = Hang UP = raccrocher). À partir du deuxième terminal, lancez le signal Nro 1 sur le premier terminal. Rouvrez un nouveau terminal et interceptez le signal 1 afin qu'il ne se ferme plus lors de la réception de ce signal. Testez. Quel signal faudrait-il envoyer pour fermer un processus qui ne veut plus rien savoir ? (kill -2 PID). Qui traite en fait ce signal ? (SIGKILL). INFO : C'est parfois le seul moyen de supprimer un processus planté (boucle infinie par exemple).
- Gestion interactive : Sous KDE, GNOME : <CTRL> + <ECHAP>. Sous Windows : <CTRL> + <ALT> + <Suppr>. Sous l'environnement minimaliste LXDE : commande « lxtask ». Dans un terminal texte : commande « top » (puis « h » pour connaître les interactions possibles).

.2 Création de processus en C

Dans les LAB précédents, vous avez exploité la fonction « system (commande) » de la « librairie C standard (stdlib) » qui lance un « shell » à l'intérieur duquel votre commande est exécutée. Votre programme appelant attend la fin de cet appel pour poursuivre son exécution. L'enchaînement est donc synchrone.
Vous allez maintenant exploiter les possibilités du système multiprocessus qui permet de créer un nouveau processus géré de manière asynchrone et donc traité en parallèle (fonction « fork » et « exec »).
Appuyez-vous sur le guide suivant :
cf. http://mtodorovic.developpez.com/linux/programmation-avancee/?page=page_3#L3

Réalisez un programme en C qui crée un processus fils. Le père affiche son PID et le PID de son fils et attend que vous tapiez la touche « p ». Le processus fils affiche son PID et son PPID. Il ne fait qu'attendre la réception d'un signal de son père (que vous définirez) pour mourir. Le père envoie ce signal quand vous tapez la touche « p ». Quand il s'est assuré que son fils est bien mort, il quitte aussi cette dure vie.
Réalisez une capture d'écran de son exécution.

```
srikanth@srikanth-VirtualBox:~$ gcc part-2.c -o part
srikanth@srikanth-VirtualBox:~$ ./part
        ------------
  tp4
tuer le processus
ce programme permet de tuer le processus
id  programme : 3527
information sur le   (processus) parent, id : 3527
Information sur le  (processus) fils : 3528
c'est le ppid fils id : 2210
processus Pere : je creer  un fils.
Pere: attend qu'un signal arrive
------------------------
Fils: attend qu'un  signal arrive
COMMANDE : taper la lettre p
      ------------
   >:c'est le processus fils, id : 3528
c'est le ppid fils id : 3527
processus Pere : je creer  un fils.
Pere: attend qu'un signal arrive
------------------------
Fils: attend qu'un  signal arrive
COMMANDE : taper la lettre p
      ------------
   >:s
commande saisie: s
attendre 5 seconde
Pere: attend qu'un signal arrive
------------------------
Fils: attend qu'un  signal arrive
COMMANDE : taper la lettre p
      ------------
   >:commande saisie:

attendre 5 seconde
Pere: attend qu'un signal arrive
------------------------
Fils: attend qu'un  signal arrive
COMMANDE : taper la lettre p
```

```
COMMANDE : taper la lettre p
      ------------
   >:P
commande saisie: P
attendre 5 seconde
 processus Pere :envoie un signal a son  fils.
 processus Pere : ok, le processus fils est inactif (terminer)
le processus fils  est bien mort et le pere quitte aussi cette duree de vie
1 -  programme  terminer
srikanth@srikanth-VirtualBox:~$
```

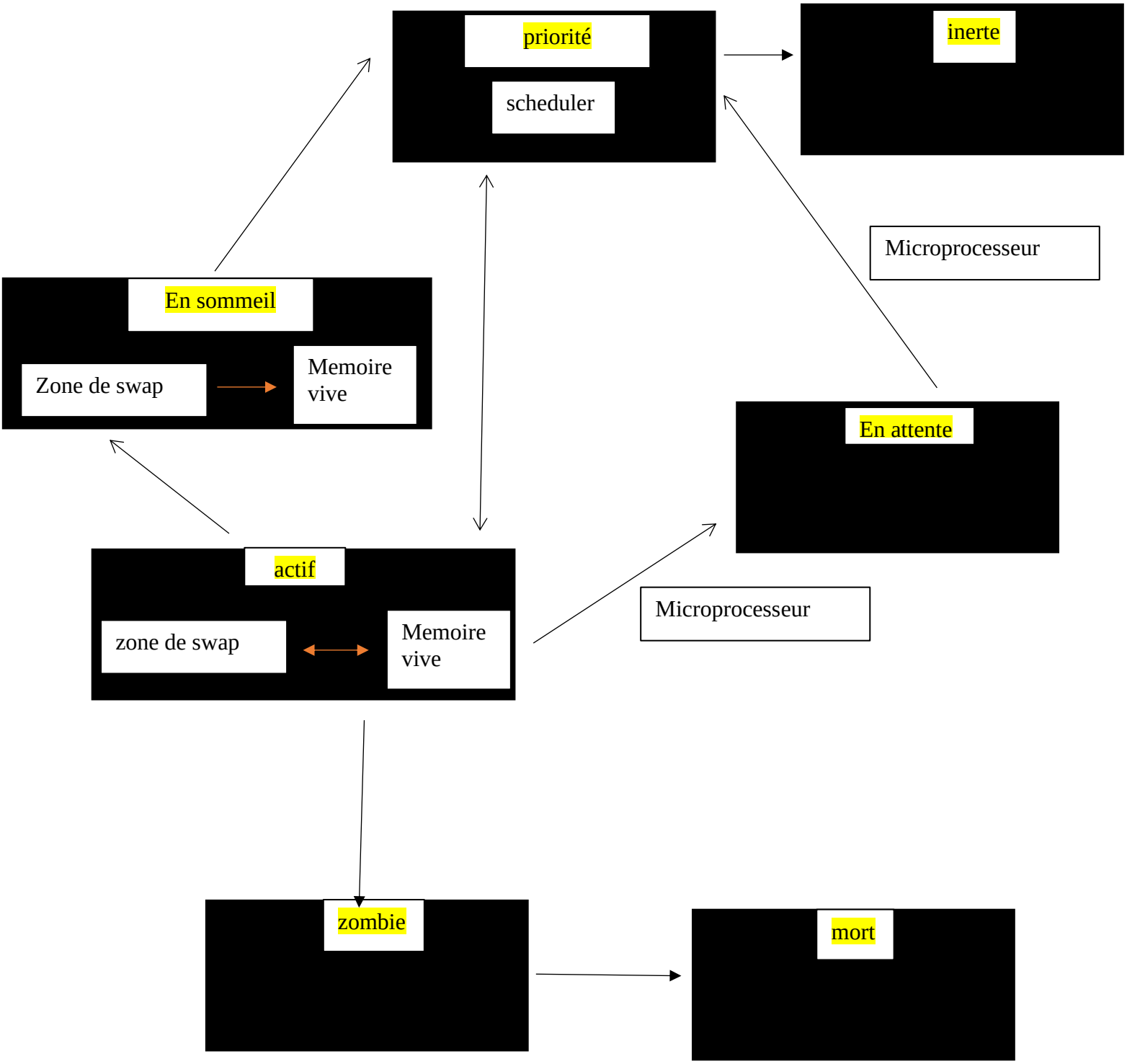
Une fois le fils créé, le père et le fils veulent écrire « je suis le (père/fils) et mon PID est ... » dans un même

fichier. Ils ne peuvent cependant pas le faire en même temps... Cherchez et développez une méthode de résolution de ce problème d'accès concurrent à une ressource (sémaphore, mutex, etc.).

Rendez ce code C sur Moodle dans un fichier part-2.c.

.3 Schéma « gestion des processus »

Réalisez ci-dessous (ou sur un document annexe) le schéma de gestion des processus sous Linux. Vous tenterez de faire apparaître les termes suivants : « microprocesseur », « mémoire vive », « zone de swap », « scheduler » « inerte » « en attente », « en sommeil », « actif », « mort », « zombie », « priorité ». Dans un court texte, vous expliquerez la vie trépidante d'un processus.



Explication la vie trépidante d'un processus :

Un processus est dirigée par des états qui diffèrent selon son activité ou si le système ou utilisateur tente de l'arrêter.

Un processus peut être suspendu, arrêté, reprendre, résilié et interrompu à l'aide de signaux.

- En cours d'exécution (actif) : Le processus est en cours d'exécution ou il est prêt à fonctionner
- En attente : Dans cet état, un processus attend qu'un événement se produise ou une ressource système.
- En sommeil : dans cet état, un processus a été arrêté, généralement en recevant un signal. Par exemple, un processus qui est débogué.
- Zombie : c’est un processus mort, il a été arrêté, mais il a toujours une entrée dans la table de processus.