Le noyau, le chapeau du système

- Objet : Comprendre la notion de noyau
- Niveau requis :
  débutant, avisé
- Commentaires : *Pour vous cultiver dans le mode de fonctionnement des systèmes d'exploitations.*
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !
- Suivi :
  à-compléter, à-tester
  - Création par captnfab le 10/02/2014
  - Testé par ... le ...
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1)]

Introduction

Le noyau est, sans mauvais jeu de mot, une pièce centrale dans un système d'exploitation. Il est difficile de donner un exemple minimal montrant ce qu'est un noyau tellement celui-ci est omniprésent. Il est l'interface avec les périphériques, il est celui qui exécute les programmes, qui leur donne la main, les fait communiquer entre eux, il est celui qui gère la mémoire, l'alloue, la range dans la swap. Il est celui qui à plus haut niveau défini la notion de fichiers, et les décode à partir des données brutes contenues sur les périphériques de stockage, etc.

Plutôt que faire une description théorique technique du fonctionnement du noyau, des tas de livres présentant la chose existent (voir les références), nous allons explorer un peu tout ça au travers de l'action la plus banale et pourtant probablement la plus complexe faisable pour un noyau : lancer un navigateur web. Nous prendrons l'exemple du noyau Linux, mais le raisonnement devrait être le même pour tous les noyaux monolithiques (modulaires ou non)

Attention, nombre de notes de pied-de-pages sont inutiles[2)], mais donc nécessaires pour alléger la lecture...

Les processus

Pouf pouf, vous êtes devant votre shell bash, dans un terminal virtuel graphique, comme Gnome-Terminal, LXTerm ou Rxvt-unicode. Vous saisissez la commande pour lancer votre navigateur, par exemple `iceweasel 2> /dev/null &`[3)], vous validez par entrée.

Vous avez ainsi effectué un tas d'opérations complexes utilisant le noyau, mais le pire est à venir.

Vous avez lancé iceweasel, un programme, c'est-à-dire un bout de code autonome qui s'exécute dans son coin. Cette notion de programme qui tourne dans son coin est appelée **processus**[4)]. Un processus donc, c'est un objet qui contient du code, stocké en mémoire et tout un tas d'autres propriétés sur lesquelles nous reviendrons.

Mitose numérique

Comment crée-t-on un nouveau processus ? En en copiant un autre.

Eh oui. Il n'y a pas d'opération magique pour créer un nouveau processus depuis zéro. Le seul processus créé à partir de «rien» est le premier, le processus n°1 nommé init. Tous les autres sont des copies. En particulier, quand vous tapez la commande `iceweasel > /dev/null &`, la première chose que fait `bash`, c'est de se dédoubler, quasiment à l'identique. La seule différence entre l'original, le shell qui vient de lire la commande, et sa copie, qui va bientôt devenir iceweasel, c'est le contenu d'une petite variable qui dit «*est-ce que je suis le fils (la copie) ou est-ce que je suis le père (l'original)*».

Comment cela s'est-il passé ? Le programme a effectué un *appel système*, c'est à dire qu'il a passé un coup de fil direct au noyau, et lui a demandé «copie moi». Cet appel système s'appelle le **fork**[5], le noyau copie alors le processus à l'identique, à l'exception d'une variable les distinguant.

Interlude pratique: Un petit code C pour forker dans votre garage.

```
nano /tmp/test_fork.c
```

test_fork.c

```c
#include <stdio.h>
#include <unistd.h>

int main()
{
  int f=fork();   // Copie-moi !
  printf("cou");  // J'affiche « cou »

  int s;
  if(f!=0)        // Si nous sommes le pere
   {
    wait(&s);     // On attend que le fils termine
    printf("\n"); // Puis on affiche un retour a la ligne
   }
  return 0;
}
```

```
gcc -o /tmp/test_fork /tmp/test_fork.c
```

```
/tmp/test_fork
```

Répartition des richesses

En ignorant platement et sans vergogne tout le bazar qui tournait déjà sur votre ordinateur[6], nous avons désormais deux processus. À faire tourner sur un seul processeur avec un seul cœur (et oui, on ne vous l'avait pas dit, on travaille sur l'ordi de grand mère.) Quel est le problème ? Vous voyez la calculatrice qu'utilise le vendeur de l'Apple Store voisin pour calculer combien font 600 + (2×1200) ? Vous avez une bonne idée de comment faire un calcul sur cette calculatrice, et même de comment en faire deux : l'un après l'autre. Mais il n'est pas question de cela sur votre ordinateur, imaginez qu'à chaque fois que vous lancez une application toutes les autres soient interrompues jusqu'à ce que vous la fermiez[7].

Lancer deux calculs en même temps sur la même calculatrice, c'est pas fastoche. Comment faire, eh bien simplement, laisser 30 millisecondes au vendeur pour travailler sur un calcul, puis 30 millisecondes pour travailler sur l'autre, puis 30 millisecondes pour reprendre le premier calcul, etc.[8] C'est donc ce que fait le noyau, sur ses versions non-préemptives[9] il laisse un temps de calcul à un processeur sur un processus, puis au bout d'un petit laps de temps, reprend la main et la donne à quelqu'un d'autre.

Une première problématique qui se pose est celle dite de la **famine**. Prenons un exemple concret, imagé, etc. Vous vous baladez dans la rue, un jour de marché. Vous êtes juste à l'angle entre le marchand de pommes bio et l'Apple Store. Deux petites filles pleurent chacune devant un de ces magasins. Vous êtes très riche parce que libriste, elles sont très pauvres parce que leurs parents claquent tout leur argent respectivement dans les salades bio et dans les 4×4 citadins. Vous décidez d'aider une des deux petites filles aujourd'hui en lui achetant l'objet qu'elle convoite, mais n'avez pas le temps d'aider les deux. À laquelle allouez-vous vos ressources ? Celle qui en a le plus besoin ? Celle qui a besoin du plus de ressources ? Celle dont les parents sont les plus influents ? Je vous laisse choisir.

Maintenant, imaginez que la situation se reproduise chaque jour, que les deux petites filles soient des processus qui veulent du temps de calcul, et que vous êtes un noyau.

Clairement, si vous ne voulez pas qu'une des deux petites filles ne meurt de faim, il va falloir trouver un moyen de leur donner du temps de calcul à toutes les deux. Mais pour des raisons écologiques, vous allez peut-être plus souvent acheter des pommes que des ipommes.

Revenons un instant à notre vendeur. Le fait qu'il n'ait toujours pas réussi à faire une addition alors que ça fait 1h qu'il tripote sa calculatrice ne vous semble pas tout à fait normal ? Si vous voulez mon avis, le problème vient du fait que les 30ms sont un peu trop courtes, et qu'il gâche tout son temps à reprendre le calcul là où il en était. Mais le rallonger à 1h, ce ne serait pas très agréable pour jongler entre le shell et le navigateur. C'est votre avis également ? Bravo, vous pensez comme un ordonnanceur !

Des jumeaux différents

Nous avons, je vous le rappelle, deux processus : le shell, et une copie du shell qui sait qu'elle est une copie du premier. Cette copie peut demander au noyau de la réinitialiser avec un nouveau code. C'est l'appel système **exec**[10]. Il consiste en l'effacement total du code du programme lancé dans le

processus[11], le chargement en mémoire d'un autre programme, ici notre navigateur, et l'exécution de celui-ci.

Notre jumeau est donc devenu très différent de son père[12].

Interlude pratique: Votre premier shell, toujours en C. Le début d'un long projet !

```
nano /tmp/test_minishell.c
```

test_minishell.c

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int main()
{
  while(1)                                // Boucle infinie
   {
    char commande[256];
    printf("$ ");                          // Afficher un prompt minimal
    fgets(commande, 256, stdin);           // Lire une ligne
     commande[strlen(commande)-1]='\0';   // Enlever le retour à la ligne

    int f=fork();                          // Copie-moi !
    int s;
    if(f!=0)                               // Si je suis le père
     {
      wait(&s);                            // Attendre que le fils se termine
     }
    else                                   // Sinon, je suis le fils
     {
        execlp(commande, commande,NULL);   // Lancer l'exécutable reçu
      perror("Erreur");                    // Si j'arrive là, j'ai échoué
     }
   }

  return 0;
}
```

```
gcc -o /tmp/test_minishell /tmp/test_minishell.c
```

```
/tmp/test_minishell
```

Ctrl+C pour interrompre le shell. Attention, ce mini-shell n'accepte pas les commandes avec arguments.

Les interruptions matérielles

Vous revenez de la pause café ? Ça tombe bien.

Rodenticide

Click click click click click click click click click click click click click click click click click click...

Vous avez déjà entendu un Apple-user se faire utiliser par son itruc? Ça ne donne pas envie d'être une souris hein ? Eh bien ça ne donne pas envie d'être un noyau non plus.

Vous avez lancé votre navigateur, il arrive sur la page d'accueil : http://debian-facile.org et vous vous emparez de votre joystick bicéphale afin de cliquouiller sur Voir les nouvelles contributions. Pendant ce temps là, votre pointeur s'est baladé à l'écran[13] et tout comme votre redoutable fauve, *Popotte*, qui roupille dans son panier, lorsqu'il se retrouve dans votre situation, vous êtes émerveillés par la quasi-parfaite synchronicité entre les mouvements gauches de votre souris, et les déplacements du pointeur à l'écran. Certes un Pommiste trouverait ça banal, mais vous avez l'œil, vous.

En effet, vous savez que chacun des petits soubresauts que vous infligez à votre rongeur est traduit par icelui en autant d'impulsions électriques qui s'en vont chatouiller le noyau à 128ko/s. La routine du noyau en charge de traiter cette nuisance est le module **evdev**. C'est lui qui doit interpréter les errances *fenêtropommesques* pour les traduire en *clics-à-côté-de-là-où-je-voulais-et-merdouille-il-est-où-ce-pointeur-à-la-noix*. La suite ne concerne plus le noyau, mais Xorg ayant été averti à son lancement par udev de la présence de l'appendice rongeuse, la surveille au travers du fichier `/dev/input/mice`[14], et en fait profiter la carte graphique (via le noyau), le gestionnaire de fenêtre (via le noyau) et toutes les autres applications concernées, via le noyau.

Interlude pratique

```
od -t x1 -w3 /dev/input/mice
```

Bougez votre souris. Ctrl+C pour quitter.

Pouf. Vous avez atteint le lien, et l'avez subtilement éraflé du clic, afin de visiter la page des nouvelles contributions.

Tennis de table

Rappels

Internet, vous connaissez. Aussi, vous savez que quand vous demandez à votre navigateur d'aller visiter une adresse comme http://wiki.debian-facile.org/?do=recent, il va commencer par résoudre le nom d'hôte « wiki.debian-facile.org » en une adresse IP. La résolution DNS peut se faire de différente manière, par exemple en lisant le fichier `/etc/hosts`[15] ou en interrogeant un serveur DNS renseigné dans `/etc/resolv.conf`[16]. L'ordre dans lequel cette recherche est faite étant fixé par les fichiers `/etc/nsswitch.conf`[17] et `/etc/host.conf`[18]. Pour `wiki.debian-facile.org`, il est probable que la résolution ne puisse pas être faite par votre fichier hosts. Le serveur DNS sera alors interrogé avec la requête suivante :

```
wiki.debian-facile.org.              IN      A
```

Ce à quoi le serveur DNS répondra gaiement :

```
wiki.debian-facile.org. 18731   IN      CNAME   newdf.debian-facile.org.
newdf.debian-facile.org. 78939  IN      A       88.191.244.114
```

Ce qui signifie que `wiki.debian-facile.org` est un alias qui pointe vers l'enregistrement `newdf.debian-facile.org` dont l'adresse IPv4 est 88.191.244.114.

Essayez vous-même :

```
apt-get install dnsutils
```

```
dig -t A wiki.debian-facile.org
```

Puis il va envoyer une requête HTTP GET afin de demander la page au serveur.

```
GET /?do=recent HTTP/1.1
Host: wiki.debian-facile.org
Referer: http://example.com/votre/adresse/initiale
User-Agent: Votre/Navigateur
```

Le serveur répondra alors

```
HTTP/1.1 200 OK
Date: Tue, 11 Feb 2014 11:16:59 GMT
Content-Type: text/html
Content-Length: Taille de la page HTML
Last-modified: Tue, 11 Feb 2014 1:40:05 GMT

<html>
```

```
…
</html>
```

Où les points de suspension représentent le contenu de la page HTML.

Ce qui signifie que la page a bien été trouvée (**200 OK** par opposition à par exemple **404 Not Found**), que c'est une page HTML, etc. Bien sûr, cette page fait référence à des images et des feuilles de styles que le navigateur devra également récupérer, par le même procédé.

Interlude pratique:

```
echo "GET / HTTP/1.1
Host: wiki.debian-facile.org

" | nc debian-facile.org 80 | less
```

Ce qui se traduit en bon françoy par *Envoyer la requête HTTP du echo sur le port 80 de la machine derrière le nom `debian-facile.org` et en afficher la réponse grace au pager `less`.* (q pour quitter)

Le noyau et le réseau

La pendule de grand-mère

1)
N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !
2)
Cette note-ci, par exemple, est inutile.
3)
Voir man 4 null pour comprendre qui est `/dev/null`.
4)
processus: de l'anglais *process* signifiant *danser à la queue-leu-leu*, comme le font les processionnaires du pin des poils desquelles, à l'instar de ceux des geeks, je me méfierais si j'étais vous.
5)
fork: de l'anglais *fourchette*, parce que le flot d'exécution se divise en deux, ça fait donc une petite fourche. Voir man 2 fork. Cela dit, une fourchette anglaise à deux doigts, c'est bien la preuve que les *british* mangent des escargots eux aussi.
6)
Réjouissez vous, ce n'est rien à côté de votre voisin sous Ubuntu.
7)
J'avais une coloc, c'était un peu ça, mais je m'égare…
8)
Avouez que vous aimeriez voir le tableau.
9)
Parfois, quand vous êtes partis pour bosser pendant 1h, et qu'au bout de deux minutes vous recevez un message sur IRC, vous êtes interrompus et changez de tâche plus tôt que prévu. Bravo, vous êtes

compilé avec préemption.
10)

Voir man 2 exec
11)

Un processus est bien plus qu'un code en cours d'exécution, il possède un espace mémoire alloué, un propriétaire, une entrée et deux sorties standard, etc.
12)

Hein ?
13)

Constitué uniquement d'une coquille et d'un surfer, ne l'oublions pas...
14)

C'est le rôle du paquet `xserver-xorg-input-evdev`.
15)

voir man hosts
16)

voir man resolv.conf
17)

voir man nsswitch.conf
18)

voir man host.conf