Linux un système multi tâches

Cet article a pour vocation de donner à un débutant Linux un aperçu sur le concept du multi-tâche, les méthodes et organisations utilisées et l'utilité globale de sa mise en oeuvre sous Linux.

En lien avec le concept du multi-tâches, nous allons donc : * faire un point sur les capacités des organes de votre PC, * aborder le concept (par comparaison à celui du mono-tâche), définir l'utilité économique et technique de son usage, * vous faire entrevoir, de manière globale et non exhaustive, la complexité des méthodes d'organisation que le multi-tâches met en jeu et leur gestion par le système d'exploitation Linux.

Les organes de votre PC :

La puissance d'un processeur, ou encore sa "capacité" à fournir l'exécution des instructions en mémoire communément repérée par le type de technologie de celui-ci, associé à la fréquence d'horloge gérant le cycle mémoire (par exemple chez Intel : Pentium IV à 2,6 GHz) est une des composantes de la capacité de traitement de votre PC. Mais bien d'autres éléments matériels entrent en ligne de compte, la vitesse (fréquence d'horloge) du bus de transfert d'informations, la vitesse de rafraîchissement de la mémoire vive, sa technologie, la capacité de transfert des organes périphériqhes (disques, disquettes, imprimantes...)... etc.

Ceci est lié au fait que les cartes mères actuelles sont élaborées autour de jeux de composants spécialisés. Ceux-ci, par le jeu de micro-programmes exécutent de manière désynchronisée (par rapport au CPU - Central Processeur Unit-), les entrées sorties entre la mémoire centrale et les différents ports (disquette, clavier, souris,... USB, IDE, ... mémoires SDRAM, DIMM, ...). Le CPU lui, n'effectue que "le travail noble de calcul" dans sa mémoire cache (mémoire très rapide en proximité) ou en mémoire centrale. Puisque tout est calcul pour votre CPU : il ajoute, décale et compare en permanence des zéros et des uns inscrits dans des registres, répondant par là à des instructions. C'est là, la tâche noble...

En fait, dans **un système mono-tâche**, le processeur qui exécute un programme en mémoire, passe le plus clair de son temps à attendre :

- attendre que la recherche d'informations sur disque soit faite,
- attendre que la mémoire vive soit rafraîchie,
- attendre votre réponse au clavier,
- attendre une écriture sur disquette,
- attendre...

Un jour, après avoir utilisé de manière intensive le logiciel MS Excel (je ne maîtrise pas la politique informatique de mon entreprise ;-)) durant une bonne journée de 9h00, j'ai eu la curiosité de regarder sur mon micro, le temps CPU qui avait été consommé. Si j'avais un évident besoin de reconstituer ma force de travail, le CPU lui, n'avait lui été utilisé que 13 minutes (et c'est énorme comme usage !), j'étais presque déçu, mais lui avait attendu.

Cela démontre que si la puissance du CPU est vitale dans certaines séquences d'exécution des programmes en mémoire (tant que le flot des instructions en cours de traitement n'est pas interrompu), dès que le système enchaîne des opérations physiques ou de transfert, il attend…

Le traitement multi-tâches optimise l'usage de cette ressource chère qu'est le Processeur Central (le CPU, représente entre le tiers et la moitié du coût d'un PC habituel !). A l'instar de ce qui existe sur les systèmes Unix, Linux organise le parallélisme de l'exécution des tâches présentes en mémoire à un instant "t". En fait, à cet instant "t" le processeur central n'exécute qu'une tâche (il peut cependant y en avoir plusieurs CPU sur une machine, mais ne compliquons pas…).

C'est **« l'ordonnanceur »** (ensemble de programmes du système d'exploitation) qui gère le fait que soit attribué, à tout programme en mémoire (qu'il soit utilisateur ou système), et à un instant « t » la ressource CPU. En fait par un algorithme complexe mêlant des attributions de tranches de temps, des priorités d'exécution, prenant en compte l'arrêt d'exécution par attente de ressource externe, cet ordonnanceur va conjointement optimiser l'accès aux ressources ainsi que la fluidité de l'exécution des programmes en mémoire centrale.

Le plus souvent, dans le cadre d'un usage courant, c'est l'attente de ressources externes qui va déclencher la mise à disposition du CPU a un autre programme, présent en mémoire centrale prêt à utiliser les disponibilités en puissance de calcul.

En fait, dès qu'une tâche en cours d'exécution en mémoire centrale a besoin d'une ressource externe (pour exemple : attente d'une entrée au clavier par l'utilisateur), le CPU initialise un ensemble d'actions lui permettant d'aller exécuter une autre tâche (qui attend, prête à redémarrer) tout en conservant les paramètres nécessaires à la reprise future de la tâche quittée (dans l'exemple : reprise qui sera possible quand l'utilisateur aura entré sa donnée au clavier et que celle-ci sera disponible pour traitement). Il va sans dire que pour l'utilisateur le programme s'exécute sans visibilité de discontinuité.

Le multi-tâche, est donc performant. Par contre l'utilisation de ce concept nécessite d'organiser le flot des tâches en machine (en l'occurrence le PC) et cette organisation en plus d'être chronophage, consomme des ressources machines (CPU, mémoire, disque, …). Il est donc nécessaire d'optimiser cette méthode de travail en s'assurant de sa performance.

De manière très globale, pour fonctionner, le multi-tâche nécessite (entre-autres !) :

- le découpage des programmes en tâches,
- la gestion des tâche susceptibles d'être exécutées à un instant « t », la gestion des priorités d'exécution entre les tâches (les tâches système sont prioritaires par rapport aux tâches utilisateur),
- la gestion de la présence en mémoire de plusieurs tâches correspondant à plusieurs programmes, l'indépendance de ces tâches entre elles,
- la gestion de l'optimisation de l'utilisation de la mémoire centrale,
- la création d'un système de communication entre les processus, la gestion de la fin d'un processus,

- … etc…

Pour faire simple, malgré la complexité, et permettre de commencer à appréhender la finesse de ce système Linux, regardons quelques phases de l'organisation « multi-tâches » décrite ci-dessus.

Phase 1

Le découpage d'un programme en tâches est des plus simples. Une tâche n'est qu'une suite d'instructions d'un programme ne provoquant aucune attente du CPU. Par exemple : une suite d'instructions entre « retour d'informations disponible en mémoire centrale prêtes à être exécutées » et « besoin d'une nouvelle information disponible sur un support externe à la mémoire centrale ».

Phase 2

L'ordonnanceur gère les priorités des tâches entre elles et (entre autres) les paramètres essentiels permettant aux tâches remises en mémoire centrales de redémarrer au bon endroit.
L'ordonnanceur est lui-même un programme qui :

- s'active en fonction des divers événements affectant les tâches en machine (tâche en retour prête à être exécutée, tâche en attente de ressources…),
- donne la main pour exécution à l'une ou l'autre tâche après lui avoir restitué un environnement de fonctionnement attendu,
- récupère le niveau de priorité d'une tâche dans le PID (voir 3) la décrivant et gère en conséquence l'accès des tâches concurrentes au CPU,
- …

Pour l'ordonnanceur, à un instant « t », un processus peut être :

- Running : prêt ou en cours d'exécution,
- Stopped : suspendu,
- Waiting : en attente d'une ressource ou d'un événement extérieur,
- Zombie : terminé mais dont les constantes d'environnement sont conservées,

Phase 3

Linux maintient toutes les informations nécessaires à la gestion des tâches des programmes en exécution dans des structures d'information chaînées entre elles (les **PID : Process Identifiers**).
Dans une arborescence ayant pour origine le processus initial (Init) les PID étant chaînés entre eux, il suffit de suivre cette arborescence pour connaître la liste des processus actifs à un instant « t ».
Dans un souci d'efficacité une table permet un accès direct à l'un d'entre-eux en fonction de son PID.

Un processus possède des droits d'accès et un groupe d'utilisateurs d'appartenance permettant de définir les autorisations d'accès à l'espace disque et aux ressources physiques.

Phase 4

Une tâche qui a la main à un moment donné **peut avoir besoin de plus de mémoire centrale**. Pour ce faire le système peut être amené à mettre sur disque l'image exacte des tâches en attente avec leur contexte d'usage (ensemble des éléments nécessaires à la correcte reprise suivante : résultats des calculs intermédiaires, variables utilisées et leur valeur, ...). C'est **le Swapping**, les informations en mémoire centrale sont mises sur disque permanent dans la partition appelée Swap, c'est une opération de permutation. Celle-ci est définie, construite et organisée lors de l'implantation initiale puis à chaque initialisation du système.

L'information sur disque sera rappelée en mémoire (nouvelle permutation), pour exécution, lors de la disponibilité de la ressource attendue et lorsque la tâche en cours sera à son tour en attente de ressource... Quitte d'ailleurs, en cas de besoin à la swapper : juste retour des choses !

Phase 5

Tout processus (à part le processus initial) est créé par un processus parent. Afin qu'ils vivent en harmonie, il est donc nécessaire que ces processus communiquent entre eux. Un des canaux de communication entre processus s'établit par **des signaux**. Chaque signal étant repéré dans l'environnement du processus par son rang dans l'espace mémoire qui est alloué au signaux et par la valeur de cet espace mémoire. Par exemple :

Lorsque le processus fils se termine normalement, il envoie le signal « SIGTERM avec 15 pour valeur » décrivant son arrêt logiciel normal au processus parent. Une commande d'arrêt programme (Ctrl+C) saisie par l'utilisateur sera transmise au processus par un signal « SIGINT avec 2 pour valeur » qui provoquera l'arrêt de la tâche de ce programme décrite dans le processus en cours.

Phase 6

etc. ...

Voilà quelques caractéristiques et méthodes que met en oeuvre Linux pour assurer l'optimisation de l'usage des ressources de votre PC. Ces méthodes étaient mises en oeuvre il y a quelques dizaines d'années sur les très grosses machines (appelées Mainframes), Linux a été écrit d'emblée pour optimiser et rationnaliser l'usage des ressources de votre PC.

source: Framasoft