**453---SYSTEME D'EXPLOITATION - e-learning - Séquence 25**

**Gestion des processus – 1**

**Légende :**

Explications sur fond blanc

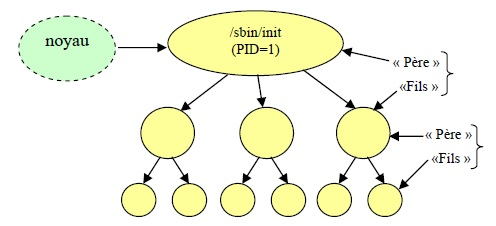
Exercice à accomplir sur fond rouge clair

Une ou plusieurs solutions possibles pour l’exécution de commande sur fond vert clair

Lignes de configuration sur fond jaune clair

Un processus est une instance de programme dont l’exécution a été lancée dans le système. À tout moment, il peut être : soit « élu » par l’ordonnanceur (réellement en cours d’exécution), soit « prêt » (dont l’exécution a été suspendue provisoirement par l’ordonnanceur), soit « bloqué » (en attente d’un événement pour être de nouveau prêt à continuer à s’exécuter).

Une application qui a été lancée consiste en un ou plusieurs processus dans le système Linux. Chaque processus correspond à un fichier exécutable qui utilise des bibliothèques, incluses dans l’exécutable (édition de liens statique) ou chargées et partagées : fichiers « .so » (« shared objects » – édition de liens dynamique). Chaque processus est créé (exécuté) par un autre processus « parent ». Le premier de tous est init, exécuté par le noyau au cours du démarrage du système. À tout moment, il y a donc un « arbre » des processus dans le système.



Chaque processus est caractérisé par : son identifiant (PID = Process IDentification), l’identifiant du processus parent (PPID = Parent PID), l’identifiant d’un groupe de processus (PGID = Process Group IDentification), le compte utilisateur sous lequel il s’exécute (UID = User IDentification), ainsi que le ou les compte(s) groupe (GID = Group IDentification), le terminal texte auquel il est associé (TTY = TeleTYpe), sa valeur modifiable (NICE) qui détermine sa priorité d’exécution, et ses arguments (CMD), en premier lieu son nom.

Chaque processus possède un espace mémoire dit d’environnement, contenant des variables d’environnement. Cet espace est hérité de processus père en processus fils.

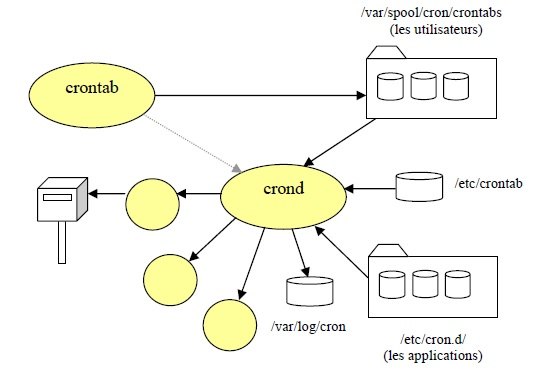
Chaque processus consomme de la mémoire. Il lui est donc alloué un espace de mémoire virtuelle. Chaque processus consomme du temps machine. Mais, la plupart du temps, un processus se trouve dans un état d’attente (non élu par l’ordonnanceur du système).

La communication interprocessus consiste en l’échange de signaux (il en existe 63, numérotés de 1 à 63). Un processus peut éventuellement ignorer un signal reçu ou déclencher en conséquence une action préprogrammée. Mais la réception par un processus d’un signal non géré entraîne sa mort (la fin de son exécution). En particulier, le signal numéro 9 ne peut être géré. On ne peut envoyer des signaux qu’à ses propres processus, à part root qui peut en envoyer à tout processus ayant un UID quelconque.

Dans une application complexe, il existe un processus qui joue le rôle de superviseur, chargé d’activer les autres processus. À la fin de l’exécution de ceux-ci, leur code numérique de retour est conservé par le noyau de Linux afin qu’il soit disponible pour le superviseur. Chaque processus occupe une entrée dans la table des processus du système, contenant des informations sur l’état des processus.

Les « services », comme le service « Cron », sont gérés par des « démons », c’est-à-dire des processus fonctionnant en permanence et traitant des requêtes. Leur lancement ou leur arrêt est effectué par des scripts, appelés « Run Command » (RC), et rassemblés dans le répertoire : « /etc/init.d ». Il est possible de les invoquer explicitement avec : start, stop, etc.

Voici les éléments du service « Cron » :



Le rôle du démon appelé « crond » ou « cron » est d’activer périodiquement des applications. La commande crontab permet à un utilisateur de soumettre à cron un fichier de requête (dans « /var/spool/cron/crontabs »), précisant quelles commandes il doit exécuter, à quels moments. Les requêtes des applications sont déposées lors de leur installation ou mise à jour dans « /etc/cron.d ». Les requêtes de l’administrateur sont déposées dans « /etc/crontab ». L’activité du démon est enregistrée dans « /var/log/cron ». Les fichiers « /etc/cron.allow » et « /etc/cron.deny » permettent de limiter l’usage de crontab.

Format d’une ligne d’un fichier de requête :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Minutes | Heures | Jour\_du\_mois | Mois | Jour\_de\_la\_semaine | Commande |
| (0-60) | (0-24) | (0-31) | (1-12) | (0-6) |  |

Dans un champ, « toutes les valeurs possibles » se note : « \* »

Une suite de valeurs se note : « 0,10,20,30 »

Une plage de valeurs se note : « 5-10 »

**Quelques commandes :**

PS(1)

NOM

ps - Présenter un cliché instantané des processus en cours

SYNOPSIS

ps [options]

DESCRIPTION

ps affiche des renseignements sur une sélection de processus actifs. Plusieurs

types d'options fonctionnent :

1 les options UNIX qui peuvent être regroupées et qui doivent être

précédées d'un tiret ;

2 les options BSD qui peuvent être regroupées et qui ne doivent pas être

utilisées avec un tiret ;

3 les options étendues GNU qui doivent être précédées de deux tirets.

KILL(1)

NOM

kill - Envoyer un signal à un processus

SYNOPSIS

kill [options] PID [...]

DESCRIPTION

Le signal par défaut de kill est TERM. Les options -l et -L affichent les

signaux disponibles. Les signaux particulièrement utiles sont HUP, INT, KILL,

STOP, CONT et 0. Les signaux peuvent être indiqués de 3 façons différentes :

-9 ou -SIGKILL ou -KILL. Une valeur de PID négative peut être utilisée pour

sélectionner des groupes de processus tout entiers ; regardez la colonne PGID de la

sortie de la commande ps. Un PID de valeur -1 est particulier : il indique tous

les processus sauf le processus kill lui-même et init.

SLEEP(1)

NOM

sleep - Endormir un processus pour une durée déterminée

SYNOPSIS

sleep DURÉE[UNITÉ\_DE\_TEMPS]...

sleep OPTION

DESCRIPTION

Effectuer une pause d'une DURÉE définie. L'UNITÉ\_DE\_TEMPS peut être « s » pour

des secondes (par défaut), « m » pour des minutes, « h » pour des heures ou « d » pour

des jours. Contrairement à la plupart des applications qui requièrent un nombre entier,

la DURÉE peut être ici un nombre décimal arbitraire.

CSH(1)

NAME

csh - Shell (command interpreter) with C-like syntax

SYNOPSIS

csh [-bcefimnstVvXx] [argument ...]

csh [-l]

DESCRIPTION

csh is a command language interpreter incorporating a history mechanism

(see History substitutions), job control facilities (see Jobs), interactive file name

and user name completion (see File name completion), and a C-like syntax.

It is used both as an interactive login shell and a shell script command processor.

APT(8)

NOM

apt - Interface en ligne de commande

SYNOPSIS

apt [-h] [-o=chaîne\_de\_configuration] [-c=fichier\_de\_configuration]

[-t=publication\_cible] [-a=architecture] {list | search | show | update |

install paquet [{=numéro\_version\_paquet | /publication\_cible}]... |

remove paquet... | upgrade | full-upgrade | edit-sources |

{-v | --version} | {-h | --help}}

DESCRIPTION

apt (Advanced Package Tool) est un outil en ligne de commande pour gérer les paquets.

Il fournit une interface en ligne de commande au système de gestion de paquets. Voir

aussi apt-get(8) et apt-cache(8) pour davantage d'options en ligne de commande.

APT-GET(8)

NOM

apt-get - Utilitaire APT pour la gestion des paquets - interface en ligne de commande

SYNOPSIS

apt-get [-asqdyfmubV] [-o=chaîne\_de\_configuration] [-c=fichier\_de\_configuration]

[-t=publication\_cible] [-a=architecture] {update | upgrade | dselect-upgrade |

dist-upgrade | install paquet [{=numéro\_version\_paquet | /publication\_cible}]... |

remove paquet... | purge paquet... |

source paquet [{=numéro\_version\_paquet | /publication\_cible}]... |

build-dep paquet [{=numéro\_version\_paquet | /publication\_cible}]... |

download paquet [{=numéro\_version\_paquet | /publication\_cible}]... |

check | clean | autoclean | autoremove | {-v | --version} | {-h | --help}}

DESCRIPTION

apt-get est le programme en ligne de commande pour la gestion des paquets. Il peut être

considéré comme l'outil de base pour les autres programmes de la bibliothèque APT.

Plusieurs interfaces utilisateur existent, comme aptitude(8), synaptic(8) et wajig(1).

APTITUDE(8)

NOM

aptitude - Interface évoluée pour le gestionnaire de paquets

SYNOPSIS

aptitude [<options>...] {autoclean | clean | forget-new | keep-all | update}

aptitude [<options>...] {full-upgrade | safe-upgrade} [<paquets>...]

aptitude [<options>...] {build-dep | build-depends | changelog | download |

forbid-version | hold | install | markauto | purge | reinstall | remove | show |

unhold | unmarkauto | versions} <paquets>...

aptitude extract-cache-subset <répertoire-sortie> <paquets>...

aptitude [<options>...] search <motifs>...

aptitude [<options>...] {add-user-tag | remove-user-tag} <étiquettes> <paquets>...

aptitude [<options>...] {why | why-not} [<motifs>...] <paquet>

aptitude [-S <nom-fonct>] [--autoclean-on-startup | --clean-on-startup | -i | -u]

aptitude help

DESCRIPTION

aptitude est une interface en mode texte pour le gestionnaire de paquets

de Debian GNU/Linux. Elle permet à l'utilisateur de connaître la liste des paquets et

de réaliser des tâches d'administration comme l'installation, la mise à jour ou

la suppression de paquets. Ces tâches peuvent être réalisées en mode « interactif »

ou à partir de la « ligne de commande ».

ACTIONS EN LIGNE DE COMMANDE

Le premier argument qui ne commence pas par un tiret (« - ») sera considéré comme étant

la commande que le programme doit réaliser. Si aucune commande n'est donnée,

aptitude démarrera en mode interactif.

CRONTAB(1)

NOM

crontab - Maintenir les fichiers crontab pour les utilisateurs individuels (Vixie cron)

SYNOPSIS

crontab [ -u utilisateur ] fichier

crontab [ -u utilisateur ] [ -i ] { -e | -l | -r }

DESCRIPTION

crontab est le programme utilisé pour installer, désinstaller ou afficher le contenu

des tables permettant de piloter le fonctionnement du démon cron(8) de Vixie Cron.

Chaque utilisateur dispose de sa propre crontab, et bien que celles-ci se trouvent

dans /var/spool/cron/crontabs, elles ne sont pas conçues pour être modifiées directement.

**LABORATOIRE – Gestion des processus – 1**

**Les exercices des séquences 25 à 26 s’enchaînent.**

**Il faut donc les accomplir dans l’ordre prévu.**

**Dans la distribution Debian GNU/Linux :**

Connectez-vous dans un terminal sous le compte « root »

Affichez le manuel des commandes ps, kill et sleep

man ps

man kill

man sleep

Listez exhaustivement les processus, puis affichez le PID de cron

ps -e

ps -e | grep cron

Sans l’option -e, ps ne liste que les processus associés au terminal courant.

Exécutez la commande sleep pour 300 secondes en tâche de fond, puis affichez en full-format les caractéristiques principales des processus associés au terminal courant et repérez les champs d’information affichés, ainsi que la tâche de fond

sleep 300 &

ps -f

Affichez les caractéristiques de la tâche de fond

ps <pid de sleep 300 &>

ps -p <pid de sleep 300 &>

ps -f <pid de sleep 300 &>

ps -f -p <pid de sleep 300 &>

Tuez la tâche de fond

kill -TERM <pid de sleep 300 &>

Affichez en full-format les caractéristiques principales de tous les processus dans le système, en filtrant

ps -ef | more

S’il n’est pas encore installé, installez le C shell csh

apt-get install csh

aptitude install csh

Affichez le manuel de la commande csh

man csh

Activez une session C shell, repérez son PID puis tuez ce C shell

csh

ps

kill <pid du C shell>

kill -9 <pid du C shell>

kill était un premier essai inopérant, car un shell ignore le signal TERM envoyé par défaut ; kill -9 est par contre opérant sur un shell.

Affichez le manuel de l’outil apt

man apt

Affichez le manuel de l’utilitaire apt-get

man apt-get

Nettoyez le référentiel local des paquets récupérés, afin de libérer de l’espace disque

apt-get clean

S’il n’existe pas encore, créez le compte utilisateur guest, commentaire = « invité », ainsi que son répertoire de connexion

useradd -c 'invité' -m guest

Mettez « guest » comme mot de passe à l’utilisateur guest

passwd guest

guest

guest

Dans le même terminal, connectez-vous sous le compte « guest »

su - guest

Affichez le manuel de la commande crontab

man crontab

Vérifiez si l’utilisateur guest possède un crontab

crontab -l

Créez, sous le compte « guest », un crontab qui accomplit les tâches suivantes :

● Ecrire, toutes les minutes, la date et l’heure à la fin du fichier /tmp/date-heure.log

● Ecrire, tous les quarts d’heure, de 9 h à 16 h 45, du lundi au vendredi, les caractéristiques principales de tous les processus du système à la fin du fichier /tmp/processus.log

crontab -e

\* \* \* \* \* date >> /tmp/date-heure.log 2>&1

0,15,30,45 9-16 \* \* 1-5 ps -ef >> /tmp/processus.log 2>&1

2>&1 signifie que la sortie d’erreurs est fusionnée avec la sortie standard (c’est-à-dire, dans ce cas-ci, que les messages d’erreur éventuels sont aussi écrits dans le fichier vers lequel on redirige en ajout).

Visualisez le crontab de l’utilisateur guest

crontab -l

Vérifiez l’effet de la première requête dans le crontab de l’utilisateur guest

tail -f /tmp/date-heure.log

<Attendez les deux ou trois écritures suivantes>

L’option -f de la commande tail permet de garder le fichier ouvert en attente de nouvelles écritures. Ctrl+C arrête l’exécution de la commande.

A l’heure 0, 15, 30 ou 45, vérifiez l’effet de la seconde requête dans le crontab de l’utilisateur guest

less /tmp/processus.log

Supprimez le crontab de l’utilisateur guest

crontab -r

Visualisez l’effet de cette suppression

crontab -l

Déconnectez-vous du compte guest

exit

Déconnectez-vous