Processus et Logs du système

Roger Yerbanga
contact@yerbynet.com

Transfer 1.3 - Bobo-Dioulasso - Décembre 2012

Plan

- Les fichiers logs
- Configuration de syslog
- Surveiller les fichiers logs
- Rotation des logs
- Charge du système
- Processus
- Contrôle des processus
- Commande ps
- Processus en avant-plan et en arrière-plan
- Les jobs
- Manipulation des processus

Les fichiers logs

- Les logs sont des fichiers textes produits par le système,
- dans lesquels celui-ci raconte ce qu'il fait et ce qui lui arrive.
- Renseignent sur ce que font les programmes,
- les connexions qui arrivent au système, les personnes qui s'y connectent.

Surveiller les fichiers logs

- La plupart des logs se trouvent dans le répertoire /var/log/
- Seul l'administrateur peut les consulter
- Quelques fichiers logs très importants :
  - /var/log/messages & syslog : Contient la plupart des messages système, comme les tentatives de connexion, les déconnexions, les échecs de connexions, le démarrage des services …

Surveiller les fichiers logs

- /var/log/mail.log : Enregistre les connexions au service de messagerie.
- /var/log/cron : les messages concernant les tâches planifiées

- Lire régulièrement les logs de sa machine permet par exemple de voir si quelqu'un essaye de vous attaquer et aussi de savoir si tout se passe bien au niveau du système, du noyau, etc...

Configuration du système de log

- Certains programmes arrivent à écrire directement les logs dans des fichiers qui sont précisés lors de la configuration de ces programmes. (ex. Exim)
- D’autres programmes envoient leurs messages au système de gestion des logs (Syslog) qui se charge de les écrire dans des fichiers.
- Syslog est paramétrable avec le fichier /etc/syslog.conf

syslog.conf (1)

- Chaque ligne du fichier a la syntaxe suivante :

  Service1.Priorité1;Service2.Priorité2 Destination

- **Service** correspond au type de service.
- Les services courants :
  - auth, authpriv, security : messages d'autorisation
  - domainname : les messages DNS
  - kern : les messages du noyau
  - mail : les messages de la messagerie
  - cron : les tâches planifiées
  - gated : messages de gateway.

syslog.conf (2)

- **Priorité** représente le niveau de gravité du message.
- Les messages par priorités décroissantes :
  - panic, emerg, alert, crit
  - err, error
  - warn, warning, notice, info, debug.
- * (indique tous les niveaux de priorité)
- none (explicitement ne log pas les messages de la classe listée).

syslog.conf (3)

- **Destination** représente la destination du message qui peut être :
  - Un chemin vers un fichier texte (**/var/log/messages**)
  - Le nom d’une console pour envoyer les messages à l’écran (**/dev/tty8**).
  - Le nom d'un serveur (@**logserver.upb.bf**)
  - Le nom d'un utilisateur connecté (root,roger)

Logrotate (1)

- Répertoire /etc/logrotate.d/

```
/var/log/ufw.log
{
rotate 4                    # nombre de rotations 4
weekly                      # Par semaine
missingok                   #
notifempty
compress                    # Compression des rotations
delaycompress               # Démarrage de la compression au n° 2
sharedscripts
postrotate
invoke-rc.d rsyslog reload >/dev/null 2>&1 || true
endscript
}
```

Logrotate (2)

- /var/log/apache2/*.log {

```
weekly                                  # daily, monthly,
missingok                               # manquant OK
rotate 52                               # nombre de fichiers
compress                                # compression
delaycompress                           # à partir du 2è
notifempty                              # pas si vide
create 640 root adm                     # les droits d'accès
sharedscripts
postrotate                              # scripts à exécuter après et avant rotation
/etc/init.d/apache2 reload > /dev/null
endscript
prerotate
if [ -d /etc/logrotate.d/httpd-prerotate ]; then \
run-parts /etc/logrotate.d/httpd-prerotate; \
fi; \
endscript
}
```

Charge du système

- Le système se résume en un ensemble de processus qui s'exécutent et qui permettent d'assurer les fonctions du système.
- Les processus occupent la mémoire, utilisent le processeur, écrivent des fichiers sur le disque.
- Donc la charge du système vient des processus qu'il exécute.
- Il faut contrôler les processus.

Qu’est ce qu’un processus

- Un processus est un programme en cours d’exécution qui a besoin de ressources matérielles : l’unité centrale, la mémoire centrale et l’accès à des périphériques d’entrées/sorties.
- Ses caractéristiques statiques (ne variant pas au cours de sa vie) sont :
  - Un numéro unique : *PID* (*Process IDentifier*),
  - Un propriétaire déterminant les droits d’accès du processus aux ressources : ouverture de fichiers ...
  - Un processus parent dont il hérite la plupart des caractéristiques,
  - Un terminal d’attache pour les entrées/sorties.
- Ses caractéristiques dynamiques sont :
  - Priorité, environnement d’exécution ...
  - Quantité de ressources consommées (temps unité centrale utilisé ...)

Arborescence des processus

- Un processus est toujours créé par un autre processus appelé processus parent.
- Tout processus a un processus parent sauf le tout premier. Ce tout premier processus est appelé init et son identifiant est égal à 1 (PID = 1).
- Deux types de processus existent :
  - Les processus utilisateurs, tous issus du shell de connexion ;
  - Les processus « démons »
- « démon » est une traduction abusive de daemon signifiant deferred auxiliary executive monitor.
- Ces processus daemon assurent un service et sont souvent lancés au démarrage de la machine.

Contrôler les processus

- Permet de voir la liste des processus qui sont lancés
- Détecter les processus qui occupent de l’espace mémoire
- Les processus gourmands en CPU
- Éventuellement, détecter les processus suspects (que vous n’avez pas lancés).

La commande PS (1)

- La commande ***ps*** lancée avec les options adéquates permet de contrôler les processus
  - Exemple : ps auwx
- ***ps*** affiche les caractéristiques des processus à un instant donné.
- Par défaut, ***ps*** affiche les processus de l'utilisateur.

La commande PS (2)

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   1692   556 ?        S    Dec06   0:04 init [5]
root         2  0.0  0.0      0     0 ?        S    Dec06   0:07 [migration/0]
root         3  0.0  0.0      0     0 ?        SN   Dec06   0:03 [ksoftirqd/0]
```

Ready | ssh2: AES-256 | 5, 17 | 5 Rows, 102 Cols | VT100

```
exim      9168  0.0  0.0  3284  604 ?        S    12:41   0:00 spamc -s 272384 -d 127.0.0.1 -p 783
exim      9169  0.0  0.0  5868 1820 ?        S    12:41   0:00 /usr/local/exim/bin/exim -bd -q10m
exim      9170  0.0  0.0  5868 1828 ?        S    12:41   0:00 /usr/local/exim/bin/exim -bd -q10m
exim      9171 14.2  0.8 34016 29320 ?       S    12:41   0:00 spamd child
exim      9172  4.7  0.7 32848 28228 ?       S    12:41   0:00 spamd child
root      9173  0.0  0.7 30860 26432 ?       S    12:41   0:00 spamd child
exim      9175  0.2  0.0  1676  524 ?        S    12:41   0:00 /usr/lib/courier-imap/bin/pop3d Maildir
exim      9176  0.0  0.0  5592 1388 ?        S    12:41   0:00 /usr/local/exim/bin/exim -bd -q10m
exim      9177  0.0  0.0  5592 1388 ?        S    12:41   0:00 /usr/local/exim/bin/exim -bd -q10m
exim      9192  0.0  0.0  2784  620 ?        S    12:41   0:00 spamc -s 272384 -d 127.0.0.1 -p 783
exim      9194  0.0  0.0  5592 1400 ?        S    12:41   0:00 /usr/local/exim/bin/exim -bd -q10m
root      9195  0.0  0.0  3788  768 pts/1    R+   12:41   0:00 ps aux
```

PID = Identificateur de processus
PPID = PID du processus père
%CPU = Portion CPU utilisée
%MEM = Portion mémoire utilisée
VSZ = Taille mémoire Virtuelle utilisée pour tout le processus (en Ko)
RSZ = RSS (Resident Set Size) Taille mémoire physique non-swappée (en Ko)
TTY = Terminal associé
STAT = Etat
START = Temps de démarrage
TIME = Temps accumulé d'exécution

Autres commandes

- Il est possible de savoir la charge globale du système avec la commande ***uptime***.
- La commande ***top*** est du dynamique uptime+ps.
- La commande ***pstree*** permet de visualiser l'arborescence des processus.
- Les commandes du et ls pour voir les tailles des fichiers créés par les processus.

Processus en avant-plan et en arrière-plan

- Par défaut, une commande s'exécute en avant-plan (foreground).
- Par exemple, l'utilisateur tape la commande ***man*** :
  - Le shell crée un processus enfant et attend qu'il se termine.
  - Le processus enfant exécute la commande ***man***.
- Donc parent et enfant s'exécutent séquentiellement.
- Une seule commande est donc exécutée à la fois.
- Une commande peut aussi s'exécuter en arrière-plan (background).
- Par exemple, l'utilisateur entre ***tcpdump &*** **:**
  - Le shell crée un processus enfant et n'attend pas qu'il se termine.
  - Le processus enfant exécute la commande date.
- Les deux processus, parent et enfant, s'exécutent alors en même temps.

Processus en avant-plan et en arrière-plan (2)

- Il est possible de suspendre temporairement un processus en avant-plan en tapant CTRL-Z.
- Le processus suspendu pourra reprendre ultérieurement.
- Il existe deux façons de reprendre un processus suspendu :
  - En le ramenant en avant-plan par la commande ***fg*** (foreground),
  - Ou en arrière-plan par la commande ***bg*** (background).
- Par exemple, l'utilisateur lance ***tcpdump*** :
  - CTRL-Z suspend cette commande.
  - ***bg*** la reprend en arrière-plan.
  - ***fg*** reprend en avant-plan
- Un job est un processus en arrière-plan ou suspendu lié au terminal courant. Chaque job a un numéro de job :
  - ***fg %1*** : ramène le job 1 en avant-plan.
- La commande ***jobs*** permet de lister ces processus.

Manipulation des processus

- En général, un processus se termine à la fin de son exécution ; il est alors éliminé par le système d'exploitation.
- On peut arrêter un processus en avant-plan en tapant CTRL-C.
- On peut aussi terminer un processus avec la commande ***kill*** qui envoie un signal à un processus.
- Par défaut, ***kill*** envoie le signal **15** de terminaison.
- La commande ***kill*** peut aussi forcer la terminaison d'un processus en envoyant le signal **9** de destruction (SIGKILL).
- La commande ***killall*** envoie un signal à tous les processus de même nom.
- Notez que le droit de détruire un processus est réservé à son propriétaire.
- ***kill -9 1709*** : tue le processus ayant pour PID 1709
- ***kill -1 1709*** : relance le processus.
- ***Killall exim*** : tue tous les processus exim.

Crontab (1)

- Exécuter automatique de tâches planifiées (heure, minute, seconde, jour, mois)
- Une crontab par utilisateur
  - crontab -e // Permet d'éditer le fichier de crontab relatif à l'utilisateur connecté
  - crontab -e -u root //Edite la crontab de l'utilisateur root.
  - crontab -l // Affiche juste la crontab le utilisateur connecté.
  - crontab -r // Efface la crontab de l'utilisateur

Crontab (2)

- Cron du système dans /etc/cron.*
  - Indépendant des utilisateurs
  - (* = d, daily, hourly, weekly, monthly)
- /etc/cron.d contient des fichiers au format crontab pour des choses plus spécifiques.
- /etc/cron.hourly, exécuté une fois par heure
- /etc/cron.monthly, chaque mois
- /etc/cron.daily » une fois par jour

Crontab (3)

- Une entrée de crontab :
  - 51 19 * * 0 root /usr/local/bin/backupbackup
- mm hh jj MMM JDS [user] commande
- Quelques caractères utilisables :
  - ***** : tout ou chaque (chaque unité de temps).
  - **,** : permet de définir une liste. Exemple : 3,5,17
  - **-** : permet de préciser un intervalle. Exemple : de 1 à 10 se note 1-10 ==> 1,2,3,4,5,6,7,8,9
  - ***/*** : permet de définir des intervalles avec des pas différents de 1. Par exemple : */10 signifie 0,10,20,30, ... donc toutes les 10 unités de temps. Si c'est position sur les minutes, ça signifie toutes les 10 minutes.