Administration des ressources

1. Planification des tâches

Ubuntu propose trois outils de programmation de tâche correspondant à trois sortes de tâches à réaliser :

- **atd** : ce service exécute un travail donné à un moment donné, et ce, une seule fois (présent sur Ubuntu server et desktop par défaut).
- **crond** : ce service exécute un travail (ou plus) à des horaires précis suivant un intervalle de temps défini (présent sur Ubuntu server et desktop par défaut).
- **anacron** : ce démon exécute un travail (ou plus) après un laps de temps déterminé (présent sur Ubuntu desktop par défaut).

L'utilitaire de base reste pour l'administrateur le service cron.

a. Fonctionnement de cron

Le service cron travaille à partir du fichier /etc/crontab qui contient les entrées correspondant aux répertoires respectifs et contenant les scripts à exécuter :

- pour les heures : /etc/cron.hourly/
- pour les jours : /etc/cron.daily/
- pour les semaines : /etc/cron.weekly
- pour les mois : /etc/cron.monthly

Exemple de fichier /etc/crontab :

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly
)
#
```

Les quatre dernières lignes montrent la périodicité sous un schéma minute / heure / jour / mois / jour de la semaine, ce qui donne :

- la 17e minute de chaque heure pour /etc/cron.hourly
- chaque jour à 6 heures 25 pour /etc/cron.daily
- chaque semaine le dimanche à 6 heures 47 pour /etc/cron.weekly
- le premier de chaque mois à 6 heures 52 pour /etc/monthly

Les services concernant les fichiers journaux `logrotate` et `sysklogd` (voir le troisième paragraphe) se trouvent - et s'exécutent chaque jour - dans le répertoire `/etc/cron.daily`.

b. Définir une crontable personnelle

Un utilisateur crée un fichier particulier de requête appelé `crontable` avec la commande `crontab`. Ce fichier se placera dans le répertoire `/var/spool/cron/crontabs/` avec comme nom celui de l'utilisateur.

Exemple de fichier utilisateur :

```
# Exemple de tâche personnelle
01 08 * * * rm .*.*~
```

Ce qui aura pour effet de supprimer tous les fichiers cachés se terminant par un tilde (fichiers temporaires d'éditeur de texte) dans le répertoire de l'utilisateur par défaut tous les jours à 8 heures et 1 minute et si la machine est allumée. On peut définir une fréquence (intervalle ou "pas") et d'autres fonctionnalités, voir dans ce cas le manuel en ligne.

Syntaxe d'utilisation de la commande `crontab`

```
crontab -e
# édite par nano (ou VIM) et modifie la crontab active
# elle ne doit pas l'être directement par un éditeur de textes

crontab -l
# liste la crontab active

crontab -r
# arrête la crontab active
```

L'administrateur change accessoirement les droits pour autoriser ou non la commande `crontab`. Sur Ubuntu, tous les utilisateurs ont par défaut le droit de l'exécuter.

2. Surveillance du système par les processus

a. Notion de processus

Un processus se traduit par l'exécution d'un programme à un instant donné : l'exécution du shell en est un dès que vous vous connectez. À chaque processus correspond un PID (*Process IDentifier*). De plus, un processus peut lancer un autre processus : dans ce cas on parlera de processus "père" et de processus "fils". Parce que Linux est multitâche, plusieurs processus peuvent s'exécuter en même temps.

Rappel : au démarrage du système, le BIOS lance le programme de démarrage (`bootstrap`), qui charge ensuite le noyau Linux et charge enfin le programme `init`, père de tous les autres. Ce premier processus a donc un PID à 1.

Il existe une relation entre types de commandes d'un shell et processus :

- Les **commandes internes** (comme `ls`) ne donnent pas lieu à une création de nouveau processus.
- Les **commandes externes** se trouvant dans différents répertoires font l'objet de la création d'un processus fils.
- Les **commandes définies par une fonction** (ou alias) renomment en fait une commande du shell.

b. Vérification et surveillance des processus

Plusieurs commandes constituent la trousse à outils de l'administrateur.

La commande ps

Syntaxe :

```
ps [-options]
```

Parmi les options les plus courantes :

- L'option a affiche les processus pour tous les utilisateurs.
- L'option x affiche les processus sans terminal de contrôle (services).
- L'option u affiche en regard de chaque processus le nom de l'utilisateur et son heure de lancement.

Exemple :

```
root@srvfai:~# ps -au
Warning: bad ps syntax, perhaps a bogus '-'? See http://procps.sf.net/faq.html
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      4125  0.0  0.0   1716   512 tty4     Ss+  08:29   0:00 /sbin/getty 38400 tty4
root      4126  0.0  0.1   1716   516 tty5     Ss+  08:29   0:00 /sbin/getty 38400 tty5
root      4131  0.0  0.1   1716   516 tty3     Ss+  08:29   0:00 /sbin/getty 38400 tty3
root      4132  0.0  0.0   1716   512 tty6     Ss+  08:29   0:00 /sbin/getty 38400 tty6
root      4549  0.0  0.2   2568  1212 tty1     Ss   08:30   0:00 /bin/login --
root      4550  0.0  0.3   4164  1820 tty1     R    08:30   0:00 -bash
root      7290  0.0  0.2   2568  1212 tty2     Ss   10:42   0:00 /bin/login --
max       7298  0.0  0.5   5488  2900 tty2     S+   10:42   0:00 -bash
root      7480  0.0  0.1   2644  1012 tty1     R+   10:59   0:00 ps -au
root@srvfai:~# _
```

Plus pratique, la commande ps s'utilise avec un "tube" pour éviter un affichage inutile et accéder plus rapidement à l'information. Voici un exemple pour rechercher les processus en relation avec dhcp :

```
ps -aux | grep dhcp
```

Une autre commande dérivée affiche les processus sous forme d'arborescence, ce qui permet visuellement de voir la filiation d'un processus envers un autre :

```
pstree [-options]
```

On a coutume de l'utiliser avec les options u (UID) et p (PID) :

```
root@srvfai:~# pstree -up
init(1)─┬─atd(4494,daemon)
        ├─cron(4507)
        ├─dd(4197)
        ├─dhclient3(4095,dhcp)
        ├─exim4(5850,Debian-exim)
        ├─getty(4125)
        ├─getty(4126)
        ├─getty(4131)
        ├─getty(4132)
        ├─in.tftpd(4442)
        ├─klogd(4199,klog)
        ├─login(4549)───bash(4550)───pstree(7538)
        ├─login(7290)───bash(7298,max)
        ├─portmap(3849,daemon)
        ├─rpc.idmapd(3888)
        ├─rpc.mountd(4436)
        ├─rpc.statd(3867,statd)
        ├─sshd(4221)
        ├─syslogd(4176,syslog)
        ├─udevd(2481)
        └─vmware-guestd(4354)
root@srvfai:~# _
```

La commande `top`

Syntaxe :

```
top [-options]
```

La commande ps n'offre qu'un aperçu "instantané" du système. **Top** fournit un résumé mis à jour des processus actifs et de leur utilisation en ressources. Le résultat de la commande peut être assez long (en fonction des processus ouverts) et sachez que la commande ne retourne pas au prompt (arrêt de la commande par [Ctrl] **C**. Elle actualise la page toutes les dix secondes. Comme elle utilise des ressources en temps processeur, vous ne devez l'utiliser que pour pister un processus bloqué.

Exemple :

```
top - 11:21:33 up  2:51,  2 users,  load average: 0.00, 0.05, 0.03
Tasks:  58 total,   1 running,  57 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    515448k total,   251408k used,   264040k free,    42576k buffers
Swap:   409616k total,        0k used,   409616k free,   174960k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 4305 root      20   0     0    0    0 S  0.3  0.0   0:00.17 vmmemctl
 7690 root      20   0  2308 1072  856 R  0.3  0.2   0:00.02 top
    1 root      20   0  1808  836  612 S  0.0  0.2   0:01.09 init
    2 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      15  -5     0    0    0 S  0.0  0.0   0:00.03 ksoftirqd/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    6 root      15  -5     0    0    0 S  0.0  0.0   0:05.16 events/0
    7 root      15  -5     0    0    0 S  0.0  0.0   0:00.02 khelper
   41 root      15  -5     0    0    0 S  0.0  0.0   0:00.02 kblockd/0
   44 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
   45 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
  108 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
  146 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  147 root      20   0     0    0    0 S  0.0  0.0   0:00.21 pdflush
  148 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kswapd0
  190 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 aio/0
 1341 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 ata/0
 1342 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 ata_aux
 1348 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 scsi_eh_0
 1351 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 scsi_eh_1
 2094 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 scsi_eh_2
 2305 root      15  -5     0    0    0 S  0.0  0.0   0:00.15 kjournald
 2481 root      16  -4  2216  652  384 S  0.0  0.1   0:00.31 udevd
 2739 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kpsmoused
 3849 daemon    20   0  1836  524  428 S  0.0  0.1   0:00.00 portmap
 3867 statd     20   0  1900  716  616 S  0.0  0.1   0:00.00 rpc.statd
 3873 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 rpciod/0
 3888 root      20   0  3652  564  316 S  0.0  0.1   0:00.00 rpc.idmapd
 4095 dhcp      18  -2  2440  772  452 S  0.0  0.1   0:00.03 dhclient3
```

Une meilleure utilisation de `top` consiste à l'effectuer dans une console différente. On comprend facilement les renseignements traitant de l'occupation processeur, mémoire et swap.

La commande `strace`

Cette commande plus difficile d'accès (plutôt pour les administrateurs chevronnés) affiche les appels système et les signaux POSIX (échanges entre le noyau et les processus). Si vous voulez en voir une démonstration, tapez :

```
strace touch essai.txt
```

La commande montre les informations pour la commande `touch` créant un fichier vide `essai.txt`. `Strace` s'utilise aussi sur des processus déjà ouverts.

La commande `lsof`

Cette commande liste les fichiers ouverts et les processus actifs :

- `lsof -i` : indique les processus de type Internet.
- `lsof -ni tcp:25` : même chose mais pour un seul protocole.
- `lsof -ni @192.168.0.1:25` : même chose mais pour une seule machine.
- `lsof -i -a -p 1234` : tous les ports réseau ouverts par le processus 1234 (`-a` est interprété comme l'opérateur ET).
- `lsof -p -u 1000` : tous les fichiers ouverts par l'utilisateur d'id 1000.

Bien sûr, il existe d'autres commandes pour faire cela, mais celle-ci est très complète.

c. "Tuer" un processus

Ubuntu stocke les processus ouverts dans le répertoire `/var/run`. La fin d'un processus se termine généralement normalement, mais il peut arriver d'avoir un processus bloqué, auquel cas il n'est pas nécessaire de redémarrer la machine. L'administrateur utilise la commande :

```
kill [-s signal]
```

On peut "tuer" un processus en ligne de commande par la commande `kill` ou `killall`. La première passe en argument le numéro de processus alors que la deuxième passe le nom de la commande. Concrètement `kill` envoie un signal à un processus et pas seulement celui qui lui demande de s'arrêter (voir la page de manuel plus de détails). Celui qui "tue" un processus est le signal 9. La commande s'écrit donc :

```
kill -9 n°_de_PID
```

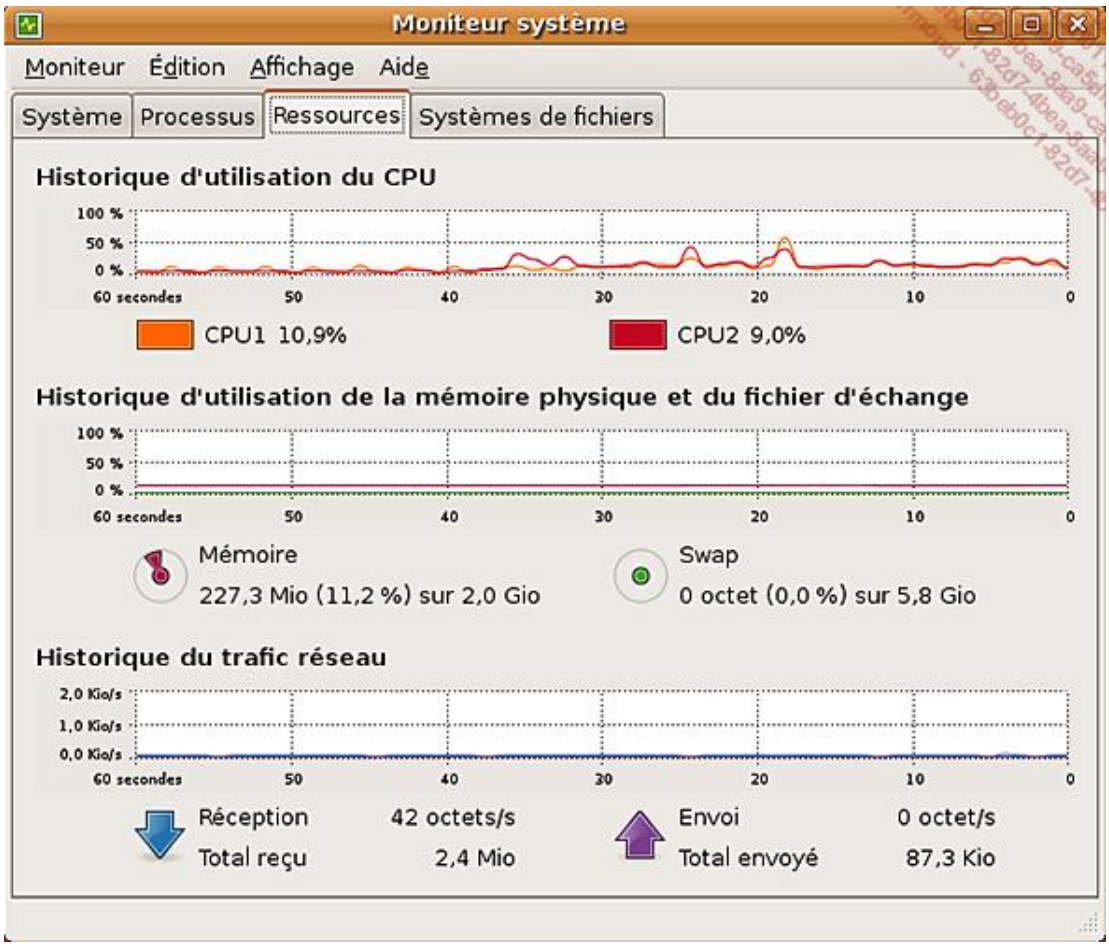
Attention de bien identifier le bon processus ! Outre les instabilités engendrées par une manipulation erronée, la réussite en dépend. Par exemple, dans le cas du serveur WEB Apache, plusieurs processus sont en jeu :

```
root      8215  0.0  0.5  10472  2584 ?        Ss   11:41   0:00 /usr/sbin/apache2 -k start
www-data  8216  0.0  0.3  10244  1784 ?        S    11:41   0:00 /usr/sbin/apache2 -k start
www-data  8218  0.0  0.4 231808  2400 ?        Sl   11:41   0:00 /usr/sbin/apache2 -k start
www-data  8222  0.0  0.4 231808  2404 ?        Sl   11:41   0:00 /usr/sbin/apache2 -k start
root      8535  0.0  0.1   2644  1008 tty1     R+   11:41   0:00 ps aux
```

Sur les quatre processus pour Apache, seul celui sous l'identité du `root` compte (processus père). C'est donc celui-là qu'il faut détruire par un `kill -9 8215`.

d. Moniteur du système en mode graphique

Sous GNOME, l'application **gnome-system-monitor** (menus **Système - Administration - Moniteur système**) rend plus agréable les opérations de surveillance et de gestion des ressources systèmes :



3. Surveillance du système par les journaux

a. Consignation des évènements

Ubuntu consigne les évènements de son activité par l'intermédiaire de deux services généraux : `syslogd` et `klogd`. La commande `ps aux | grep logd` montre (parmi d'autres lignes) les deux processus en cours :

```
syslogd 5256 0.0 0.0 1936 684 ? Ss 07:46 0:00 /sbin/syslogd -u syslo
...
klogd 5314 0.0 0.0 3860 2776 ? Ss 07:46 0:00 /sbin/klogd -P /var/run
```

Les fichiers journaux se trouvent dans le répertoire `/var/log` :

```
root@muaddib:~# ls /var/log/
acpid           dmesg           lastlog         samba
apache2         dmesg.0         lpr.log         scrollkeeper.log
apparmor        dmesg.1.gz      mail.err        syslog
apt             dmesg.2.gz      mail.info       syslog.0
aptitude        dmesg.3.gz      mail.log        syslog.1.gz
auth.log        dmesg.4.gz      mail.warn       syslog.2.gz
auth.log.0      dpkg.log        messages        syslog.3.gz
boot            exim4           messages.0      udev
bootstrap.log   faillog         messages.1.gz   unattended-upgrades
btmp            fontconfig.log  mysql           user.log
cups            fsck            mysql.err       user.log.0
daemon.log      gdm             mysql.log       wtmp
daemon.log.0    installer       mysql.log.1.gz  wvdialconf.log
debug           kern.log        mysql.log.2.gz  Xorg.0.log
debug.0         kern.log.0      news            Xorg.0.log.old
dist-upgrade    kern.log.1.gz   pycentral.log
root@muaddib:~#
```

Le répertoire comporte plusieurs autres journaux car, pour des raisons de lisibilité, les consignations se rapportant à d'autres services font l'objet d'un ou de plusieurs fichiers séparés. Ainsi, au lieu de "fouiller" dans le fichier `syslog`, l'administrateur accède directement aux informations de l'application. Certaines gèrent même leurs journaux sans passer par le démon `syslog`.

Ceci se gère par la configuration dans le fichier `/etc/rsyslog.d/50-default.conf` dont voici un extrait :

```
#  Default rules for rsyslog.
#
#                       For more information see rsyslog.conf(5)
and /etc/rsyslog.conf

#
# First some standard log files.  Log by facility.
#
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log

#
# Logging for the mail system.  Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info                       -/var/log/mail.info
mail.warn                       -/var/log/mail.warn
mail.err                        /var/log/mail.err
```

Peu de modifications ou changements sont à apporter dans ce fichier, ainsi que la configuration par défaut dans `/etc/default/syslogd`. Cela dépend essentiellement de l'application. Aussi, vous vous reporterez au manuel de celle-ci. La structure d'une ligne suit la base :

```
fonction.priorité          fichier
```

Dans l'exemple, les évènements du service de courrier (mail) comportent trois priorités : `info`, `warn` et `err`. Le tiret devant les fichiers de logs des deux premiers informe le démon `syslog` de ne pas synchroniser le fichier après chaque message. L'astérisque comme priorité lui indique d'inscrire tout ce qui concerne la fonction. Il n'est pas contradictoire d'avoir les deux comme le montre l'exemple `mail`. Tous les messages seront inscrits dans `mail.log` et une ventilation de ceux-ci ce fera dans les trois autres par priorité.

La pratique courante de l'administrateur consiste à ouvrir une console supplémentaire et à taper la commande :

```
tail -f /var/log/syslog
```

Cela ouvrira la fin du fichier sans le refermer. Toute nouvelle inscription s'affichera en temps réel : ce qui est, pour

un administrateur, très utile.

Pour être complet, sachez qu'en mode graphique un utilitaire **gnome-system-log** affiche ou surveille les journaux système par les menus **Système - Administration - Visionneur de journaux système**.

b. Archivage des fichiers journaux

De par l'obligation légale dans certaines activités de garder trace des informations et simplement du fait que l'administrateur puisse les revoir, les fichiers journaux sont archivés suivant des règles définies pour le programme `logrotate` dans `/etc/logrotate.conf` :

```
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

# system-specific logs may be configured here
```

Ubuntu transforme donc les fichiers journaux en archive (`numéro.gz`) avec, on le voit, une fréquence hebdomadaire qui n'excède pas 4 semaines. Les fichiers `wtmp` et `btmp` utilisés par les programmes `login`, `shutdown`, etc. font l'objet d'un archivage séparé. Vous pouvez mettre vos propres règles d'archivage là où le message le spécifie.

L'exécution de `logrotate` s'effectue par `cron` et, vu la fréquence, plus exactement par `cron.weekly`.