Systèmes d'exploitation

Module UE142 : administration de systèmes

Services du système Linux

Pierre Nerzic

IUT de Lannion

1 - Tâches planifiées

a) Présentation

Trois services permettent de lancer des commandes ou des scripts à des moments précis. Ils gèrent une sorte d'agenda. Ils diffèrent sur leur manière de travailler :

- at : ce service permet d'exécuter une commande à une date et heure donnée. Voir la commande at : at heure < script. On n'en parlera pas plus.
- cron : ce service est destiné aux machines allumées en permanence = garantissant une continuité du temps, les événements peuvent se produire au moment prévu.
- anacron : ce service fonctionne sur les machines qui ne sont pas allumées en permanence. Sur ces machines, il n'y a pas de garantie qu'un événement puisse se produire à un instant donné : si la machine est éteinte, le rendez-vous est raté. C'est pourquoi, anacron note les

dates de dernière exécution des tâches afin de savoir s'il faut les relancer.

Anacron permet de gérer des événements qui doivent se produire au moins une fois tous les *n* jours, semaines ou mois. En cas de non-exécution au moment prévu, le lancement est repoussé au prochain démarrage.

Cron permet de gérer des événements qui doivent se produire à une date précise. Si la machine est éteinte à ce moment, la tâche n'est pas déclenchée.

Anacron est destiné à fonctionner de temps en temps, par exemple au démarrage de la machine. Il s'arrête dès qu'il n'y a plus rien à faire. Au contraire, cron tourne en permanence et surveille l'agenda chaque minute et relance anacron régulièrement.

b) Organisation du système cron

Ces services sont lancés dans les runlevels ordinaires. Anacron lance les tâches qui auraient dues être lancées depuis son dernier lancement puis s'arrête. Cron travaille en tâche de fond, se réveille toutes les minutes pour examiner son agenda.

Les agendas des deux dispositifs sont structurés de la même manière : une ligne par tâche. Le début de la ligne indique les dates et heures de lancement, des jokers permettent de définir des répétitions. La fin de la ligne spécifie la commande.

/etc/anacrontab

Le calendrier de anacron est /etc/anacrontab. Voici un exemple :

```
# /etc/anacrontab: configuration file for anacron
# See anacron(8) and anacrontab(5) for details.

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

# These replace cron's entries
1 5  cron.daily     nice run-parts --report /etc/cron.daily
7 10 cron.weekly    nice run-parts --report /etc/cron.weekly
@monthly 15 cron.monthly nice run-parts --report /etc/cron.monthly
```

chaque ligne : période délai id commande

période = nombre de jours entre deux lancements de la commande

délai = nombre de minutes d'attente de anacron avant de lancer la commande

Ici on voit qu'anacron lance tous les jours run-parts /etc/cron.daily. La commande run-parts consiste à exécuter tous les scripts placés dans le répertoire indiqué : for cmde in ($1/*) ; do $cmde ; done

/etc/crontab, /etc/cron.d/* et autres crontabs

Le calendrier général de cron est /etc/crontab. Voici un exemple à étudier :

```
# m h dom mon dow user  command
17 * * * * root run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || run-parts --report
/etc/cron.daily
47 6 * * 7 root test -x /usr/sbin/anacron || run-parts --report
/etc/cron.weekly
52 6 1 * * root test -x /usr/sbin/anacron || run-parts --report
/etc/cron.monthly
30 8-18/2 * * mon-fri root rsync -ab comptes/ archive
```

Chaque ligne : m h dom mon dow user command

m, h = minute, heure de lancement, dom = jour, mon = mois, dow = jour dans la semaine (dimanche = 0 et 7, lundi = 1,... ou abbréviations anglaises thu, fri...)

user = compte qui doit lancer la commande

On peut employer le joker * qui signifie toutes les valeurs possibles pour le champ, un intervalle de deux valeurs, un chamtest -x /etc/init.d/anacron && p

suivi de */nombre* spécifie que la tâche est faite toutes les nombres unité de temps.

Dans l'exemple plus haut, crontab lance les travaux listés dans /etc/cron.hourly à la 17e minute de chaque heure et fait un rsync en semaine de 8h30 à 18h30 toutes les 2 heures.

Consulter la documentation de crontab : man 5 crontab.

Cron consulte également /etc/cron.d. Il prend en compte tous les fichiers qui sont dedans.

Chaque utilisateur peut également créer son propre crontab, dans /var/spool/cron/crontabs. Il doit utiliser la commande crontab –e pour éditer, crontab –l pour afficher. L'administrateur peut etest -x /etc/init.d/anacron && mpêcher certains utilisateurs ou n'autoriser que certains utilisateurs en remplissant les fichiers /etc/cron.allow et /etc/cron.deny.

NB: quand on modifie la crontab à la main, sans utiliser la commande crontab, il faut recharger le démon cron : /etc/init.d/cron restarttest -x /etc/init.d/anacron &&

c) Configuration de cron et anacron sur Ubuntu

Chaque système configure les tâches à sa manière. Si on creuse un peu celle de l'Ubuntu, par exemple : chaque jour, à 6h25 cron vérifie qu'il y a anacron ou alors lance lui-même les tâches de /etc/cron.daily.

Le répertoire /etc/cron.d contient un crontab appelé anacron qui permet de lancer anacron tous les jours à 7h30 si la machine est allumée, et de toutes façons à chaque démarrage (/etc/rc2.d/S89anacron).

Voici /etc/cron.d/anacron :

```
# /etc/cron.d/anacron: crontab entries for the anacron package

30 7 * * * root test -x /etc/init.d/anacron && /usr/sbin/invoke-rc.d anacron start >/dev/null
```

d) Tâches lancées par cron et anacron

Un certain nombre de tâches sont effectuées régulièrement par le système cron :

- mises à jour des paquets du système
- mise à jour de la base locate et slocate (recherche de fichiers)
- rotation des logs
- effacement des caches inutiles (pages de manuel mises en forme)
- backup de certains fichiers (mots de passe)

Ces opérations sont définies dans les scripts de /etc/cron.*

On va maintenant examiner quelques uns de ces services.

2 - Syslog

a) Présentation

Cette partie s'intéresse à la surveillance du bon fonctionnement du système d'exploitation. On ne traite pas les aspects réseau (communication, sécurité) mais les aspects système (utilisation des informations).

Plusieurs aspects :

- activité des comptes et des processus,
- étude de la charge du matériel : occupation CPU, mémoire, E/S disques,
- gestion des erreurs du système.

En général des messages sont enregistrés dans des fichiers appelés logs. Des commandes permettent de les consulter et d'en extraire du sens.

b) Journaux du système

La plupart des fichiers de log sont des textes lisibles. Il n'y a pas besoin de commandes spécifiques pour les lire (grep, tail, less).

i) Enregistrer les connexions des utilisateurs

Le système enregistre les connexions des utilisateurs dans différents fichiers. Ces fichiers ne sont pas des textes ascii, mais des fichiers binaires dont les formats sont indiqués dans "/usr/include/*nom*.h" (man utmp).

/var/run/utmp : créé par init à chaque boot ; un enregistrement est créé par la commande login associée à chaque terminal ; login y place le nom du compte, le nom de la machine distante, la date de début de connexion, et n'y laisse que la date de déconnexion le cas échéant. Ce fichier est utilisé par la commande who.

/var/log/wtmp : idem, mais on trouve deux enregistrements par session : début et fin de connexion. Ce fichier n'est pas effacé automatiquement, c'est à faire manuellement ou par une directive dans rc*.d. En plus, on trouve les dates de boot et d'arrêt du système ainsi que les changements de

date. Ce fichier permet de rassembler des statistiques sur les connexions des utilisateurs.

/var/log/lastlog : rempli par la commande login (s'il n'existe pas, il faut le créer), contient les dates de connexion de chaque UID : une ligne par UID.

Ces fichiers ne sont pas lisibles en clair (c'est très rare dans Unix), il faut utiliser les commandes suivantes pour afficher le contenu :

- **`who`**
- **`finger`**
- **`last [utilisateur]`**
- **`ac [-dp]`** (du paquet acct)

Certains systèmes détaillent davantage les informations des connexions. Ils offrent également d'autres commandes et avec des options spécifiques.

ii) Messages du noyau

Le noyau du système émet par la fonction printk de nombreux messages lors du boot et pendant le fonctionnement du système :

- informations sur le matériel
- erreurs et avertissements, logiciels ou matériels
- mise en route ou arrêt d'un élément du noyau

Ces messages sont placés dans des fichiers du répertoire /var/log, selon la distribution Linux, p. ex. Redhat :

- /var/log/boot.log : logs de init et des scripts rc
- /var/log/syslog : logs du démarrage initial, état du matériel

Ces messages sont également mémorisés dans un tampon circulaire qu'on peut consulter avec la commande :

- **dmesg**

Ensuite, selon le type du message, on trouve trois fichiers dans /var/log/kernel : info, warnings, errors.

iii) Messages des sous-systèmes

Certains sous-systèmes ont également leurs propres fichiers de log, par exemple XFree86 qui implémente X11.

Quand on démarre X11 de nombreuses informations sont écrites dans /var/log/XFree86.log et permettent de savoir ce qui se passe.

iv) Le système de journalisation syslog

Comme le nombre des messages est important, qu'il a fallu centraliser leur gestion afin de ne pas multiplier les fichiers, les créateurs d'Unix ont proposé le mécanisme appelé syslog.

C'est un ensemble permettant d'inscrire automatiquemenlprt les erreurs dans différents fichiers. On a :

- Le démon klogd qui s'occupe uniquement du noyau, il redirige ses messages vers syslogd quand celui-ci est actif
- le démon syslogd et son fichier de configuration /etc/syslog.conf

- des fonctions de librairie openlog(), syslog(), closelog()...
- la commande logger

Ce système est défini par la RFC3164 en tant que client-serveur, un poste émet un message vers un ou plusieurs collecteurs à travers des relais. Un message est composé d'un code-priorité, d'un en-tête, d'un corps. Le code priorité est composé lui-même de :

- une partie « type d'émetteur » (appelé entité ou facility) normalisée : kernel 0, user 1, mail 2, daemon 3, auth 4...
- un niveau de sévérité de 0 à 7 : 0 = emergency (crash imminent), 1 = alert (erreur à corriger de suite), 2 = critical (grave erreur à réparer le plus tôt possible), 3 = error (ça continue mais c'est pas bon), 4 = warning, 5 = notice, 6 = info, 7 = debug.

Le fichier de configuration spécifie ce que doit faire le collecteur avec le message. Ce fichier est lu seulement lors du lancement ou de la réception du signal HUP par le démon. Sa syntaxe est : une directive par ligne, une directive = une liste de priorités (émetteur, sévérité) TAB une action. Ex :

```
*.error      /dev/console
```

```
*.info          /var/log/messages
user.*          /var/log/user.log
```

Les jokers permettent d'économiser les touches du clavier, mais présentent l'inconvénient qu'à un message de sévérité donnée peuvent correspondre plusieurs actions. Ici, un message user.error sera à la fois écrit à la console, dans le fichier messages (car error ≥ info) et dans user.log.

L'action est :

- le nom d'un fichier auquel est concaténé le message,
- la notation @machine, pour renvoyer ces messages vers la machine,
- une liste d'utilisateurs séparés par des virgules (affichage écran).

Ce sont les programmes C qui envoient des messages au démon par les fonctions du système suivantes :

- **openlog(nom de l'auteur, options, type d'émetteur)**
- **syslog(niveau, texte)**

```
#include <syslog.h>
```

```
openlog(argv[0], LOG_PID, LOG_KERNEL);
syslog(LOG_NOTICE, "je suis là");
syslog(LOG_INFO, "hep, un peu d'attention");
closelog();
```

En shell, on peut aussi émettre des messages par la commande :

- `logger -p priorité [message]`

```
logger -p KERNEL.INFO "je suis là"
```

v) Interprétation des fichiers de log

En ce qui concerne l'interprétation, rien n'existe, il faut parcourir les fichiers à la main : grep, sort, tail, wc...

Ce sont souvent de très gros fichiers puisque tout y est inscrit.

3 - Gestion des fichiers de log

Pour éviter que les fichiers de log ne deviennent trop gros, illisibles et ingérables donc inutiles, un dispositif Unix propose de les découper en morceaux : log.0, log.1, log.2... en fonction de leur age et ensuite de faire

disparaître les fichiers les plus anciens. Il existe différents programmes qui font ce découpage, sur Linux on dispose de :

- **logrotate**

Ce logiciel se configure avec un fichier /etc/logrotate.conf. C'est presque un langage de programmation. Le lancement quotidien de ce script est en général effectué par cron.

Un exemple :

```
# 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
```

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

Quelques détails sur le fichier logrotate.conf :

- des directives globales qui s'appliquent à tous les fichiers gérés par logrotate, ce sont les mêmes directives que les suivantes, ex : weekly, compress, size...

- des directives spécifiques pour certains fichiers, la syntaxe est : nomfichier { directives spécifiques }, ces directives ne s'adressent qu'à ces fichiers.

Les directives sont :

- rotate *NB* : donne le nombre de morceaux autorisés, le dernier log sera détruit à la prochaine rotation (log devient log.0 qui devient log.1 qui devient log.2...)

- daily, weekly, monthly : donne la périodicité de rotation, en principe logrotate peut créer des fichiers vides, sauf s'il y a la directive notifempty

- size *TAILLE* : indique la taille max d'un morceau, la rotation est forcée dès que la taille dépasse cette valeur.

- create : après rotation, le fichier de log est recréé – certains logiciels exigent la présence préalable du fichier de log (ex : utmp et wtmp)

- compress : les vieux logs seront compressés avec gzip

4 - Informations et statistiques

On s'intéresse maintenant à l'obtention de statistiques sur la machine. On désire connaître la charge du système afin, par exemple, de peaufiner les réglages du noyau.

`uptime`

Indique depuis combien de temps le système fonctionne, le nombre d'utilisateurs connectés et la charge de la machine dans les dernières minutes.

De nombreux aspects du système peuvent être surveillés :

- les processus
- la mémoire
- l'activité des systèmes de fichiers et des disques

Le problème général est de limiter et d'interpréter les informations.

a) Système de fichiers /proc

L'arborescence /proc contient de nombreux éléments intéressants. On y trouve sous forme de pseudo fichiers (ils n'ont pas d'existence en tant que tels sur un disque mais sont créés à l'instant où on les consulte) le détail sur :

- la machine : état des périphériques, montages,
- les processus : commande, charge cpu, répertoire, variables, état...
- le noyau : modules, symboles...

b) Activité de la mémoire virtuelle : vmstat

Cette commande permet de déterminer comment la mémoire est utilisée, les informations sont très nombreuses. Consulter la documentation.

- **`vmstat [intervalle]`**

Elle donne des informations sur les processus : nombre dans chaque état (prêt, bloqué, swappé), sur la mémoire (pages occupées, libres, les défauts de page), les disques (transferts/s), les interruptions et l'état du CPU.

Il est intéressant de constater que cette commande utilise /proc.

c) Activité des entrées/sorties : iostat

Cette commande permet de déterminer quel est le disque le plus chargé, quel est le débit moyen... Elle donne également quelques infos recoupant celles de vmstat.

- **`iostat [disque] [intervalle]`**

Les périphériques sont indiqués par n°majeur et n°mineur et non pas par leur nom en clair. Consulter /proc/devices et /proc/partitions pour obtenir la liste des périphériques.

d) Activité sur les fichiers : lsof

Cette commande complexe permet de déterminer quels sont les fichiers ouverts et dans quels modes. On retrouve quelques informations de ps, selon les options employées : le nom du programme, le PID et propriétaire du processus, le type de l'objet ouvert, le périphérique, la taille de l'objet, l'i-node et son nom.

- **`lsof [-p PID] [fichier]`**

On peut aussi s'en servir pour déterminer quel sont les processus qui possèdent tel fichier ouvert.

e) Activité des processus

La commande top permet de visualiser rapidement quel est le processus qui consomme le plus de temps machine.

- `top`

f) Génération de statistiques

La commande sar permet de collecter de nombreuses informations sur le système pendant une certaine période. Elle possède beaucoup d'options :

- `sar [options]`

Voici les options utiles :

-b : pour obtenir des infos sur les buffers fichiers (zones de stockage des données évitant les accès disques). On peut savoir s'ils sont bien dimensionnés.

-u : activité du processeur entre les processus, les appels système, les E/S

-v : l'utilisation des tables système

g) Autres commandes

- `strace commande`

Trace tous les appels systèmes de la commande, indique les résultats.