Gestion et surveillance des processus

Un **processus est un programme en cours d'exécution**

Pour connaître tous les processus en cours de fonctionnement

- *ps aux* : liste de tous les processus, avec PID, le terminal tty où ils ont été lancés (sinon ?), utilisateur propriétaire, ressource en CPU et mémoire, date démarrage
- *pstree* | *less* filiation des processus sous forme arborescente
- *pidof mysqld* liste des PID des processus d'un programme (numéro d'un service souvent stocké dans un fichier qui porte son nom, dans le répertoire /var/run)

connaître les ressources utilisées par les processus *top*, périodiquement mise à jour (taper q pour quitter)

Gestion et surveillance des processus

connaître l'état de la mémoire *watch -n 1 free -m*, mémoire disponible, utilisée, libre (en Mo)...

Pour modifier l'état d'un processus

- *kill -15 PID* : demande normale d'arret au processus, il peut refuser
- *killall -9 mysqld* : suppression plus radicale, en cas de processus récalcitrant
- *kill $(pidof mysqld)* supprime le processus mysqld avec *PID* obtenu par *pidof*
- *killall -HUP mysqld* : ordonne au processus de relire son fichier de configuration, ce qui évite de le relancer
- *kill -l* pour connaître la liste des signaux qu'on peut passer à kill.

Services **actuellement actifs en écoute**

Statistiques (stat) sur la configuration réseau (net) avec `netstat` : sur les connexions, les tables de routages, les connexions masquées, les interfaces, ...
`netstat -taupe | grep LISTEN` (ou `nestat -l`)

- `[--tcp|-t] [--udp|-u]` protocoles à observer
- `-a, --all` Show both listening and non-listening sockets. With the –interfaces option, show interfaces that are not up
- `-p, --program` Show the PID and name of the program to which each socket belongs.
- `-e, --extend` Display additional information. Use this option twice for maximum detail.
- `-l, --listening` Show only listening sockets. (These are omitted by default.)

Avec System V init `/etc/init.d/<service> status`
Avec Upstart init `initctl status <service>`

`nmap` est destiné pour le scan de machines distantes

Et avec SystemD ?

- le statut d'un service :
  `systemctl is-active sshd.service`
- plus de détails `systemctl status sshd.service`
- lister tous les services actifs
  `systemctl list-units --type=service`
- démarrer un service `systemctl start sshd.service`
- stopper/redémarrer/recharger la configuration un service
  `systemctl stop sshd.service` (restart/reload)
- activer un service au démarrage
  `systemctl enable sshd.service`
- le désactiver au démarrage
  `systemctl disable sshd.service`

`https://www.linuxtricks.fr/wiki/systemd-les-commandes-essentielles#paragraph_statut-d-un-service`

Avec SystemD

```
$ systemctl
UNIT                                          LOAD   ACTIVE SUB       DESC
sys-subsystem-net-devices-enp0s25.device      loaded active plugged   Ethe
sys-subsystem-net-devices-wlo1.device         loaded active plugged   Wire
apache2.service                               loaded active running   LSB:
cups.service                                  loaded active running   CUPS
systemd-journald.service                      loaded active running   Jour
systemd-modules-load.service                  loaded active exited    Load
bluetooth.target                              loaded active active    Blue
...

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

198 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.

https://www.linuxtricks.fr/wiki/systemd-les-commandes-essentielles
```

Utilisation d'un superserveur/service dispatcher (xinetd)

Le process `init` peut se charger du lancement de services réseaux (voir System V et Upstart). Il est néanmoins possible de lancer un super-serveur : xinetd, e**X**tended **I**ter**NET** **D**aemon, `xinetd.org` (remplaçant d'inetd) **pour gérer (notamment démarrer) d'autres démons en fonction des demandes de connexions et de contraintes d'accès**

Pour chaque service

Contrôler les accès : à certaines IP à certaines date/heure

Prévenir contre les attaques de déni de services : limiter le taux des connexions entrantes à certaines IP, limiter la taille des logs

Augmenter les capacités de journalisation : logs indépendants des services, suivi des connexions/déconnexion/tentatives de chaque client

Rediriger des demandes de connexion vers d'autres hotes

Interagir Utilisateur : personnaliser les bannières de connexion

Spécification dans fichier de configuration `/etc/xinetd.conf`

Sous Ubuntu, il n'est pas présent par défaut, car il faut le configurer.

Extrait exemple de /etc/xinetd.conf

Lu dans `man 5 xinetd.conf`

```
service ftp
{
    socket_type = stream
    wait = no
    user = root
    only_from = 128.138.252.1
    server = /usr/etc/in.ftpd
    server_args = -l
    instances = 4
    log_on_success += DURATION HOST USERID
    log_on_failure = HOST USERID
    access_times = 2:00-9:00 12:00-24:00
}
```

TCP wrappers (encapsuleur de services TCP)

Filtre d'accès aux services à certains hosts (autorisation/restriction)

- Historiquement, géré à travers le démon `tcpd` par un appel du superserveur, inetd dans `/etc/inetd.conf` :
  `/usr/sbin/tcpd <service>`
- Désormais, capacités de filtrage implémentées dans bibliothèque **libwrap** ; permet à une application de se passer de superserveur (par exemple pour gérer indépendamment plusieurs connexions)
- *inetd* utilisait *tcpd* et *xinetd* utilise *libwrap* (*tcpd* possible)
- Utilise 2 fichiers de configuration `/etc/hosts.allow` et `/etc/hosts.deny`
  Pour refuser toutes les demandes de connexions pour tous, on rajoute dans `/etc/hosts.deny` la ligne `ALL : ALL`
  `hosts.allow` est prioritaire sur `hosts.deny`, le premier ne devrait rien contenir
  Pour vérifier la validité des règles mises en place : `tcpdchk`