MATHÉMATIQUES ET INFORMATIQUE
Sciences
Université Paris Cité

Chapitre 5: Processus et services

Chérifa Boucetta

A.U: 2022-2023

Contenu

1. Le système de gestion de processus

2. Les services

Description et Objectifs

- UNIX est un système multi-tâches autorisant ainsi l'exécution de plusieurs tâches (ou processus) à la fois.
- Deux types de processus :
  - **Les processus système**: Ces processus ont comme propriétaire l'administrateur (root) ou un UID d'administration. Ils assurent des tâches d'ordre général et ne sont généralement arrêter qu'à l'arrêt du système.
  - **Les processus utilisateur** : Ils correspondent à chaque exécution d'un programme par un utilisateur. Le premier d'entre eux étant l'interpréteur de commande à la connexion.

Le processus

- Un **processus** représente à la fois un programme en cours d'exécution et tout son environnement d'exécution (mémoire, état, identification, propriétaire, père...).

- Les données d'identification d'un processus :

  1. **Un numéro de processus unique PID** (Process ID) : chaque processus Unix est numéroté afin de pouvoir être différencié des autres.
     - Le premier processus lancé par le système est 1 et il s'agit d'un processus appelé généralement init.
     - On utilise le PID quand on travaille avec un processus. Lancer 10 fois le même programme (même nom) produit 10 PID différents.
  2. **Un numéro de processus parent PPID** (Parent Process ID) : chaque processus peut lui-même lancer d'autres processus, des processus enfants (child process). Chaque enfant reçoit parmi les informations le PID du processus père qui l'a lancé.
     - Tous les processus ont un PPID sauf le processus 0 qui est un pseudo-processus représentant le démarrage du système (crée le 1 init).

Le processus (suite)

3. **Un numéro d'utilisateur et un numéro de groupe** : correspond à l'UID et au GID de l'utilisateur qui a lancé le processus. C'est nécessaire pour que le système sache si le processus a le droit d'accéder à certaines ressources ou non.
   - Les processus enfants héritent de ces informations.
4. **Durée de traitement et priorité** : la durée de traitement correspond au temps d'exécution écoulé depuis le dernier réveil du processus.
   - Dans un environnement multitâche, le temps d'exécution est partagé entre les divers processus, et tous ne possèdent pas la même priorité. Les processus de plus haute priorité sont traités en premier.
   - C'est l'ordonnanceur de tâches du système qui gère les priorités et les temps d'exécution.

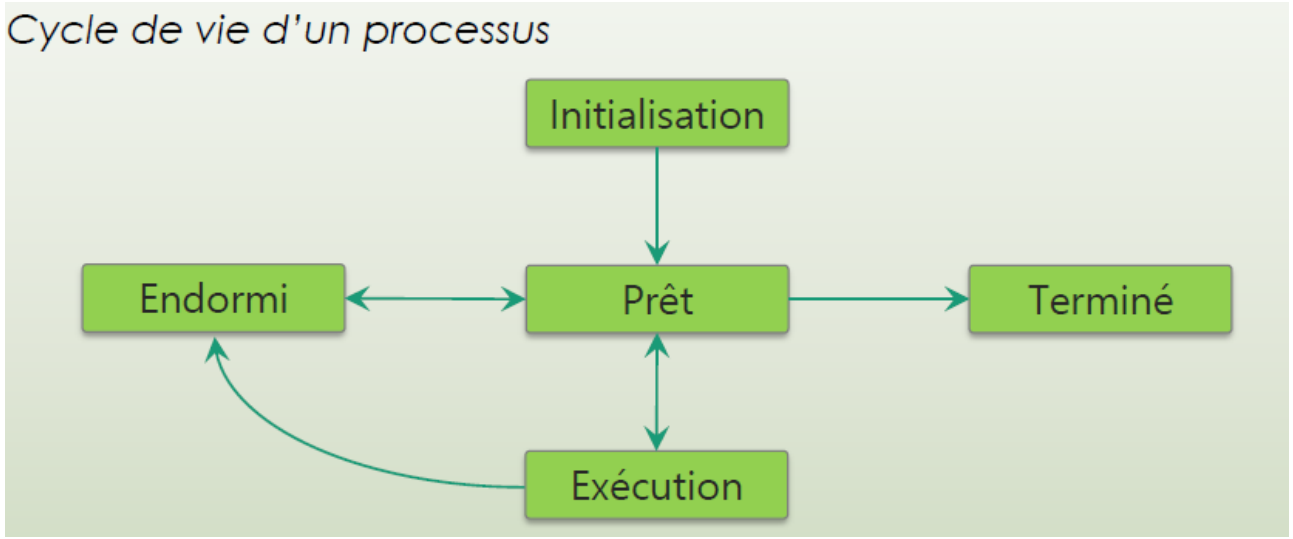
Le processus (suite)

5. **Répertoire de travail actif** : à son lancement, le répertoire courant (celui depuis lequel le processus a été lancé) est transmis au processus. C'est ce répertoire qui servira de base pour les chemins relatifs.
6. **Fichiers ouverts** : table des descripteurs des fichiers ouverts. Par défaut au début seuls trois sont présents : 0, 1 et 2 (les canaux standards). À chaque ouverture de fichier ou de nouveau canal, la table se remplit. À la fermeture du processus, les descripteurs sont fermés.

- On trouve d'autres informations comme la taille de la mémoire allouée, la date de lancement du processus, le terminal d'attachement, l'état du processus, les UID effectif et réel ainsi que les GID effectif et réel.

Etat d'un processus

- Durant sa vie (temps entre le lancement et la sortie) un processus peut passer par divers états ou **process state** :
  - exécution en mode utilisateur (**user mode**)
  - exécution en mode noyau (**kernel mode**)
  - en attente E/S (**waiting**)
  - endormi (**sleeping**)
  - prêt à l'exécution (**runnable**)
  - endormi dans le swap (**mémoire virtuelle**)
  - fin de processus (**zombie**)



- La liste de tous les processus avec leurs états peut être affichée avec la commande **ps –el.**

Processus-commandes

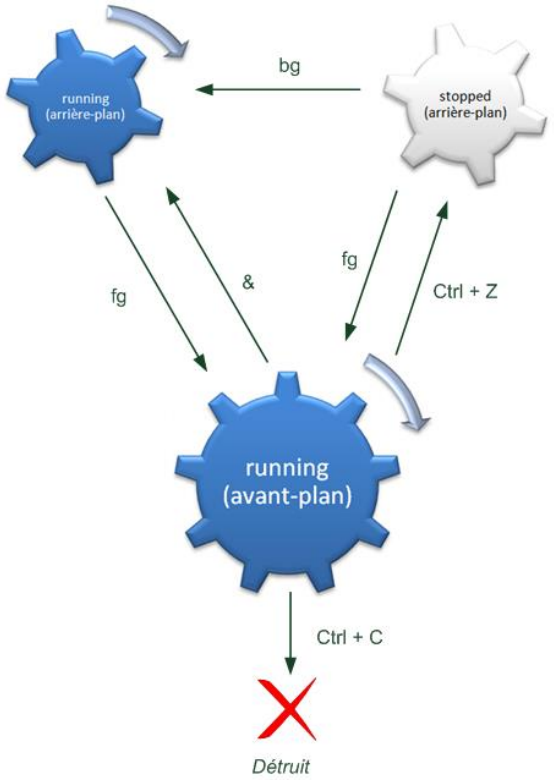
- Tous les processus ont un parent,
- Ils Définissent une arborescence de processus depuis le processus initial (**init**),
- Commandes pour les processus sont:

Commande	Action
ps	affiche les **PID** des processus d'une manière statique
top	**ps** mais avec mise à jour dynamique
jobs	liste des processus associés à un terminal
bg	Mettre un processus en arrière plan
fg	Mettre un processus en premier plan
kill	envoie un signal à un processus

Background, foreground, jobs

- Il est possible de récupérer la main sous le shell si un processus est lancé au premier plan. On peut le stopper temporairement en tapant [Ctrl] **Z** :

```
$ sleep 100
<CTRL+Z>
 [1]+  Stopped                 sleep 100
```



- Le processus est stoppé :
  - son exécution est suspendue jusqu'à ce qu'il est placé au premier plan avec la commande fg (foreground) :

```
$ fg
Sleep 100
```

Background, foreground, jobs

- Quand une commande est lancée, le premier nombre entre crochets présente le numéro de job.
  - C'est le même nombre qui apparaît quand une commande est lancée en arrière-plan. La commande **jobs** donne la liste.

```
$ jobs
[1]-  Stopped                 sleep 100
[2]+  Stopped                 sleep 100
```

- Les commandes **bg** et **fg** permettent d'agir sur ces jobs en prenant comme paramètre leur numéro. La commande **bg** est exécutée sur un job stoppé pour le relancer en arrière-plan.
  - Exemple: Le job 2 est relancé en arrière-plan :

```
$ bg 2
[2]+ sleep 100 &
$
[2]+  Done                    sleep 100
```

Liste des processus

- La commande **ps** (process status) permet d'avoir des informations sur les processus en cours.
  - Lancée seule, elle n'affiche que les processus en cours lancés par l'utilisateur et depuis la console actuelle.

```
$ ps
  PID TTY          TIME CMD
 4334 pts/1    00:00:00 bash
 5017 pts/1    00:00:00 ps
```

```
ps -f
UID          PID  PPID  C STIME TTY          TIME CMD
seb         4334 24449  0 09:46 pts/1    00:00:00 /bin/bash
seb         5024  4334  0 10:51 pts/1    00:00:00 ps -f
```

Liste des processus

- Les paramètres **-ef** donne des informations sur tous les processus en cours.

```
$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
...
seb      26431     1  0 Mar04 ?        00:00:30 kded [kdeinit]
seb      26436 26322  0 Mar04 ?        00:00:03 kwrapper ksmserver
seb      26438     1  0 Mar04 ?        00:00:00 ksmserver [kdeinit]
seb      26439 26424  0 Mar04 ?        00:00:50 kwin [kdeinit
seb      26441     1  0 Mar04 ?        00:00:28 kdesktop [kdeinit]
seb      26443     1  0 Mar04 ?        00:03:40 kicker [kdeinit]
seb      26453     1  0 Mar04 ?        00:00:00 kerry [kdeinit
seb      26454 26424  0 Mar04 ?        00:00:01 evolution
seb      26465 26424  0 Mar04 ?        00:00:11 kde-window-decorator
seb      26467     1  0 Mar04 ?        00:00:02 gconfd-2 12
seb      26474     1  0 Mar04 ?        00:00:01 knotify [kdeinit]
seb      26485     1  0 Mar04 ?        00:03:06 beagled
```

- Le paramètre **-u** permet de préciser une liste d'un ou de plusieurs utilisateurs séparés par une virgule.
- Le paramètre **-g** effectue la même chose mais pour les groupes, **-t** pour les terminaux .
- Pour plus de détails: **man ps**

Arrêt d'un processus / signaux

- Le **signal** est l'un des moyens de communication entre les processus. Lorsqu'on envoie un signal à un processus, celui-doit doit l'intercepter et réagir en fonction de celui-ci.
- La commande Kill sert essentiellement à arrêter des processus en arrière plan.

Kill –signal PID

  - Où signal est un numéro de signal envoyé à un processus PID

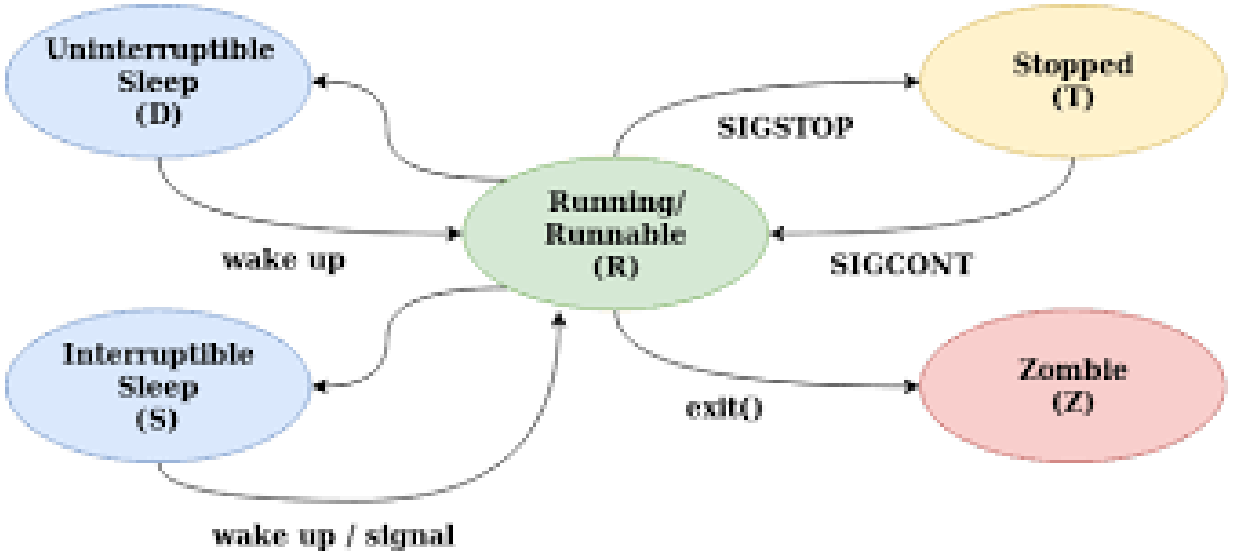
Arrêt d'un processus / signaux

- Les principaux signaux sont:

Signal	Rôle
1 (SIGHUP)	Hang Up, est envoyé par le père à tous ses enfants lorsqu'il se termine.
2 (SIGINT)	Interruption du processus demandé (touche [Suppr], [Ctrl] C).
3 (SIGQUIT)	Idem SIGINT mais génération d'un Core Dump (fichier de débuggage).
9 (SIGKILL)	Signal ne pouvant être ignoré, force le processus à finir 'brutalement'.
15 (SIGTERM)	Signal envoyé par défaut par la commande kill. Demande au processus de se terminer normalement.
18 (SIGCONT)	redémarre un procesus suspendu (avec bg ou fg)
20 (SIGTSTP)	suspend un processus (généré par control-z)

Arrêt d'un processus / signaux

```
kill -SIGINT 4805
kill -SIGTSTP 4834
kill -SIGCONT 4834
kill -SIGKILL 4867
```




```
$ sleep 100&
[1] 5187
$ kill 5187
$
[1]+  Complété                sleep 100
$ sleep 100&
[1] 5194
$ kill -9 5194
[1]+  Processus arrêté        sleep 100
```

La commande time

- La commande **time** mesure les durées d'exécution d'une commande, idéal pour connaître les temps de traitement, et retourne trois valeurs :
  - **real** : durée totale d'exécution de la commande.
  - **user** : durée du temps CPU nécessaire pour exécuter le programme.
  - **system** : durée du temps CPU nécessaire pour exécuter les commandes liées à l'OS (appels système au sein d'un programme).

```
$ time ls -lR /home
...
real    4.8
user    0.2
sys     0.5
```

Le processus de démarrage

- Le processus **init** est le père de tous les processus et possède toujours le PID 1. Il est démarré par le noyau, qui porte le PID « virtuel » 0.
- **systemd** « system daemon » est un système d'initialisation et une alternative à l'init classique. Il est le premier processus démarré par le noyau.

```
cboucetta@intranet:~$ pstree
systemd─┬─VGAuthService
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─atd
        ├─cron
        ├─dbus-daemon
        ├─irqbalance───{irqbalance}
        ├─login───bash
        ├─networkd-dispat
        ├─nmbd
        ├─polkitd───2*[{polkitd}]
        ├─rpc.gssd
        ├─rpc.idmapd
        ├─rpc.mountd
        ├─rpcbind
        ├─rsyslogd───3*[{rsyslogd}]
        ├─smbd─┬─cleanupd
        │      ├─lpqd
        │      ├─smbd
        │      └─smbd-notifyd
        ├─snapd───12*[{snapd}]
        ├─snmpd
        ├─sshd─┬─sshd───sshd───bash───pstree
        │      └─sshd───sshd───sftp-server
        ├─sssd───sssd_be
        ├─sssd_nss
        ├─sssd_pac
        ├─sssd_pam
        ├─2*[systemd───(sd-pam)]
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-network
        ├─systemd-resolve
        ├─systemd-timesyn───{systemd-timesyn}
        ├─systemd-udevd
        ├─unattended-upgr───{unattended-upgr}
        ├─upowerd───2*[{upowerd}]
        └─vmtoolsd───{vmtoolsd}
```

Contenu

1. Le système de gestion de processus

2. Les services

Systemd

- Systemd utilise des mécanismes performants faisant appel à des fonctions (API) uniquement présentes dans le noyau Linux comme.
  - la notion de cgroups qui permet de regrouper des processus dans un même groupe (chaque processus lié à un service est placé dans un même groupe).
- Les définitions des différentes unités sont présentes dans /lib/systemd/system.
- Les unités cibles ou les services devant être gérés par le système au démarrage se situent dans /etc/systemd/system, sous forme de liens symboliques ou de copies de fichiers.
  - Ces chemins peuvent être placés dans des endroits différents selon les distributions.

Les services

- La configuration de Systemd se base sur des unités (units) qui ont un nom et un type.
  - **Exemples:**
    - le fichier NetworkManager.service définira l'unité de type service qui s'occupe de la gestion réseau
    - systemd-shutdownd.socket définira la socket pour l'arrêt.
- Un démon (processus) est un petit programme qui a pour but d'assurer une tâche particulière.
- Un service utilise un ou plusieurs démons :
  - **Exemples:**
    - NetworkManager gère le réseau,
    - sshd s'occupe des connexions sécurisées SSH,
    - crond s'occupera de l'automatisation des tâches répétitives etc.

Les services

- Les services finissent par le suffixe .service.
- Ils sont contrôlés par la même commande **systemctl**
  - Les unités de services sont des fichiers texte standardisés décrivant les règles et commandes liées à ce service.

service	systemctl	Description
service name start	systemctl start name	Démarre le service.
service name stop	systemctl stop name	Stoppe le service.
service name restart	systemctl restart name	Redémarre le service.
service name condrestart	systemctl try-restart name	Redémarre le service seulement s'il est démarré.
service name reload	systemctl reload name	Recharge la configuration.
service name status	systemctl status name systemctl is-active name	Indique si le service est démarré.
service --status-all	systemctl list-units --type service -all	Affiche le statut de tous les services.

Les services

- Liste des services actifs

```
# systemctl list-units --type=service
UNIT                    LOAD   ACTIVE SUB     DESCRIPTION
abrt-ccpp.service       loaded active exited   Install ABRT coredump hook
abrt-oops.service       loaded active running  ABRT kernel log watcher
abrtd.service           loaded active running  ABRT Automated Bug Reporting Tool
atd.service             loaded active running  Job spooling tools
auditd.service          loaded active running  Security Auditing Service
avahi-daemon.service    loaded active running  Avahi mDNS/DNS-SD Stack
chronyd.service         loaded active running  NTP client/server
crond.service           loaded active running  Command Scheduler
cups.service            loaded active running  CUPS Printing Service
dbus.service            loaded active running  D-Bus System Message Bus
...
```

Les services

- **Status** donne un grand nombre d'informations et notamment les traces du service ce qui peut être très utile lorsque le service ne démarre pas
- À propos des traces, utiliser **journalctl** pour une description plus détaillée.

```
# systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: disabled)
    Active: active (running) since Sat 2020-02-01 18:03:46 CET; 8min ago
      Docs: man:sshd(8)
            man:sshd_config(5)
  Main PID: 1049 (sshd)
     Tasks: 1 (limit: 4672)
    Memory: 4.6M
       CPU: 163ms
    CGroup: /system.slice/sshd.service
            +-1049 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,chacha20-poly1305
@openssh.com,aes256-ctr,aes256-cbc,aes128-gcm@openssh.com,aes128-ctr,aes128-cbc -oMACs=hma

févr. 01 18:03:46 fedora sshd[1049]: Server listening on 0.0.0.0 port 22.
févr. 01 18:03:46 fedora sshd[1049]: Server listening on :: port 22.
```

Les services

- L'activation ou la désactivation d'un service permet de spécifier son état souhaité au chargement de la cible.
- ces modifications n'auront aucune action immédiate sur le service (démarrage ou arrêt), il faudra pour cela jouer un start/stop.

systemctl	Description.
systemctl enable name	Active un service.
systemctl disable name	Désactive un service.
systemctl status name systemctl is-enabled name	Indique l'état du service.
systemctl list-unit-files --type=service systemctl list-dependencies --before / --after name	Liste des services, avec ses dépendances.

Actions système

- Les commandes **shutdown** restent toujours disponibles, tout comme **halt**, **reboot** ou **poweroff** qui sont des alias vers les commandes **systemd** associées.

Systemd	Description
systemctl halt	Stoppe le système sans éteindre la machine.
systemctl poweroff	Stoppe le système et éteint la machine.
systemctl reboot	Redémarre la machine.
systemctl suspend	Suspend l'exécution du système (économie d'énergie).

Fin du Chapitre 5