

Séance No 2

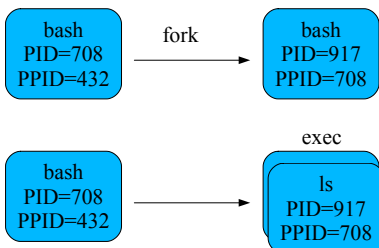
- processus
- quelques filtres classiques
- premiers scripts shell

processus

- un programme: un fichier sur disque
- un processus: un programme en cours d'exécution
 - le code exécutable du programme
 - les données de l'instance en train de s'exécuter
- programme réentrant:
 - deux instances du même programme partagent le même code exécutable
 - elles ont par contre chacune leurs données
- processus système (daemon)/utilisateur

hiérarchie de processus, recouvrement

- un processus (processus fils) est toujours créé par un autre processus (processus père):
 - fork: création d'une copie du processus père
 - exec: recouvrement par le processus fils



Hiérarchie de processus

- tout processus a un processus parent sauf le processus initial
- processus initial : init (pid 1)
- arrêté la machine: demander à init d'arrêter tous ses processus fils

ps tree

```
ns.lami
petit@dell-2:~$ ps tree
init--*--std
|-2*[automount]
|-bdflush
|-cron
|-cupd
|-dhclient-2.2.x
|-6*[getty]
|-gpm
|-icmlogd
|-inetd
|-kdm--XFree86
|-kdm---kdm_greet
|-keventd
|-khubd
|-3*[klogd]
|-klogd
|-ksoftirqd_CPU0
|-ksuapd
|-kupdated
|-lockd
|-mdrecoveryd
|-ntpd
|-portmap
```

commandes internes/externes

- commande externe: fichier exécutable:
 - création d'un nouveau processus chargé d'exécuter la commande
- commande interne:
 - exécutée par le shell (pas de création de nouveau processus)

type

- type indique si une commande est interne
- options non standard:
 - -a: indique toutes les implémentations
 - -p : indique le chemin de la commande (rien si interne)
- Exemples: testez type sur les commandes suivantes :
 - cd
 - ls
 - pwd

caractéristiques des processus

- statiques
 - PID
 - PPID
 - propriétaire
 - terminal d'attache pour les entrées-sorties
- dynamique
 - priorité
 - nice
 - consommation cpu/mémoire
 - dossier de travail

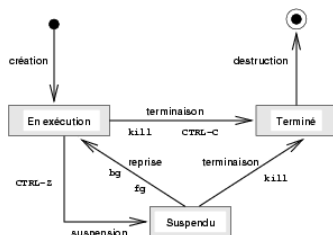
commande ps

- 2 syntaxes:
 - syntaxe Systeme V: option précédées de -
 - syntaxe BSD: options NON précédées de -
 - quelques options SysV:
 - -e ou -A: tous les processus
 - -a: tous les processus associés à un terminal
 - -H: représentation hiérarchique (forêt)
 - -f: format complet; -l: format long (encore plus détaillé)
 - -o: pour modifier le format de sortie (cf manuel)
 - -g, -p, -t, -u: n'affiche que les processus des groupe (-g), processus (-p), terminaux (-t) ou utilisateurs (-u) listés.

commande ps: exemple

Etat d'un processus

- R: exécution
- Z: zombi: il est mort mais son père ne le sait pas
- D: le processus ne peut être interrompu
- S: suspendu
- T: terminé



gestion de processus

- &
- bg
- fg
- jobs
- Ctrl-C
- Ctrl-Z

Signaux • permettent au système de communiquer avec les processus • signaux utiles – STOP: suspendre – CONT: reprendre – HUP (1): souvent: relecture configuration – KILL(9): tuer sans possibilité de traitement – INT(2): équivalent à Ctrl-C: interruption gérable. permet au processus de gérer son interruption • kill -signal PID	trap • dans une version future de ce document
avant plan/arrière plan/détachement • dans une version future de ce document (ràf)	priorité des processus • l'exécution des divers processus est gérée par un ordonnanceur (scheduler) • une priorité est définie dynamiquement • but: que chaque processus puisse avancer son exécution tout en respectant des priorités • nice: permet d'influer sur la priorité des processus – de 0 à 19 pour un utilisateur – de -20 à 19 pour root – plus le chiffre est élevé, moins le processus est prioritaire
code de retour • valeur à laquelle le processus père peut accéder • 0: terminaison normale • autre valeur: situation anormale • commande1 && commande2: la commande2 est exécutée si la commande 1 réussit • commande1 commande2: la commande2 est exécutée si la commande 1 échoue • exemple: commande test • exemple: construction if/then/else/fi	commande test • réalise des tests simple, le code de retour indique que le test est positif ou négatif • test -d /var/tmp : teste si /var/tmp est un dossier • test -x /bin/ls: teste si /bin/ls est un exécutable • test 1 = 2: teste l'égalité de deux chaînes • forme alternative : – test -d /var/tmp – [-d /var/tmp]

structure de contrôle if

- syntaxe:

```
if commande1
then
    commande2
[elif commande3
then commande4]
...
[else
    commande5]
fi
```
- si commande1 retourne 0, on exécute commande2 sinon, si commande3 retourne 0, on exécute commande4 ... sinon commande5

Entrées-sorties

- Entrées-sorties
 - entrée standard: 0
 - sortie standard: 1
 - sortie d'erreur standard: 2



Héritage

- les descripteurs d'un processus enfant sont initialement les mêmes que ceux du processus père.
- si on ne les modifie pas, en sortie, les affichages s'entrelacent.
- il est possible de changer la valeur de entrée standard et des sorties standard et en erreur
 - pour les rediriger depuis/vers un fichier : <, >, >>, 2>, 2>>
 - pour les rediger depuis/vers un processus : |

redirection des sorties

- >: le contenu du fichier est remplacé par la sortie de la commande
- >>: la sortie s'ajoute à la fin du fichier
- exemples:
 - ls /etc > /tmp/foo.txt
 - cat ls
 - ls /usr/bin >> /tmp/foo.txt
 - du -sk /var/* > /tmp/bar.txt
 - date > /tmp/bar.txt (noter que le No d'inode est inchangé)

redirection de la sortie d'erreur standard

- commande 2> fichier
- commande 2>> fichier
- pratique pour isoler messages d'erreur et sortie
- /dev/null: le trou noir: pour éliminer les messages d'erreur
- du -sk /var/* 2> /dev/null

redirection simultanées

- on peut rediriger plusieurs descripteurs sur une même ligne de commande
- ls > /tmp/f1 2> f2
- les redirections sont traitées de gauche à droite
- du -sk /var/* > /tmp/resultat 2> /tmp/erreur
- en cas de redirection simultanée avec cette syntaxe: impérativement vers des fichiers différents

redirection en entrée

- commande < fichier
- exemples:
 - mail petit < texte
 - wc -l < /etc/passwd

redirections avancées

- &1: valeur du descripteur de fichier 1
- &2: valeur du descripteur de fichier 2
- 1>&2: l'entrée standard est redirigée vers le même fichier que la sortie standard
- sert pour des redirection simultanées vers un même fichier
- Exemples:
 - ls > /tmp/test 2>&1 #OK
 - ls 2>&1 >/tmp/test #pas OK: stderr est toujours lié au terminal

<<

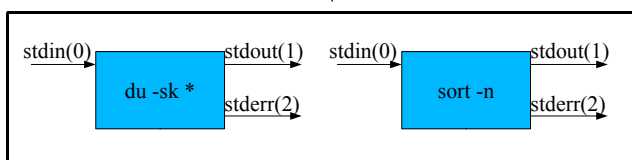
- dans une version ultérieure de ce document

fermeture d'un descripteur

- dans une version ultérieure de ce document

enchaînement de commandes

du -sk * | sort -n



L'ensemble forme une nouvelle commande

Filtres

- commande lisant leurs données sur l'entrée standard et envoyant leur sortie sur la sortie standard
- pratique pour les enchaînements de commandes
- philosophie unix: des commandes simples que l'on combine entre elles

sort

- selon SUSv3, sort a trois fonctions sur son entrée standard ou des fichiers textes constituées de lignes contenant un ou plusieurs champs :
 - trier les données (par défaut)
 - fusionner des fichiers triés en une sortie globale triée (option -m)
 - vérifier que les données sont triées (option -c)

Sort: options courantes:

- -t car_sep: permet de préciser le caractère qui sépare les champs du fichier
- -k : précise les champs sur lesquels portent le tri
- -o: indique un fichier de sortie (par défaut: sortie standard)
- -d: supprime les doublons
- -c : vérifie si un fichier est trié. Le résultat est indiqué uniquement par le code de retour: 0 si trié, 1 sinon.
- -m : fusionne des fichiers supposés déjà triés

uniq

- supprime les doublons d'une liste triée
- exemple: cat /tmp/test.txt |sort|uniq
- voir manuel pour les autres options
-

tail/head

- queue/tête d'un fichier

wc

- compte le nombre de lignes, de mots et de caractères
- wc -l : nombre de lignes
- wc -w : nombre de mots
- wc -c : nombres de caractères

grep, egrep & Co

- grep chaine: sélectionne les lignes qui contiennent la chaine

cut

- sélectionner certaines colonnes

tr

more/less

find

commandes qui ne lisent pas leur
entrée standard

- ls, who, find
- chmod, cp, mv, rm, ln, mkdir
- date
- kill
- file, type
- echo