Présentation

- Pascal PETIT
- petit@shayol.org
- tel: 01 69 47 XX XX

Définition

- SHELL: programme en mode texte assurant l'interface entre l'utilisateur et le système unix.
- S'utilise :
  - En interactif depuis une fenêtre terminal (xterm, connexion distante texte, ...) : interpréteur de commande
  - Pour réaliser des scripts (fichiers de commandes) : langage de programmation

Shell: où que je clique ?

- On ne clique pas : ça s'utilise avec une souris à 105 touches et sans boule : un clavier :-)
- L'accès est moins immédiat que celui d'une interface graphique
- Plus de liberté/possibilités qu'avec une interface graphique
- Langage de programmation: possibilité d'exprimer des requètes complexes
- Utilisation interactive ou pour écrire des fichiers de commandes (scripts)

Historique

- Les deux shells des origines sont à l'origine de deux familles de shells aux syntaxes incompatibles :
  - Le shell le plus ancien : sh ou Bourne shell écrit dans les années 70 par Steve Bourne. Tout système système unix a un shell /bin/sh qui est un bourne shell (ou un shell compatible);
  - Le csh: écrit à la même époque par Bill Joy incompatible avec le bourne shell mais offrant quelques fonctionnalités supplémentaires (historique des commandes, aliases, contrôle de tâches, ...

Historique (2)

- Ksh: korn shell (David Korn, 1983) sur la base du bourne sh. Le ksh 88 (ou +) est livré avec tous les unix commerciaux. Base de la norme IEEE Posix 1003.2;
- Tcsh: un shell évolué de la famille csh utilisé dans les années 90 comme shell interactif;
- Bash: Bourne Again sh, le shell de la FSF. Compatible posix 1003.2. Le shell de base des distribution linux.
- Zsh: un shell riche en fonctionnalités. Probablement le meilleur choix actuel en interactif.

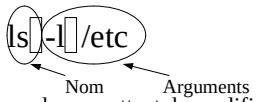
Historique (3)

- POSIX:
- SUS: Single Unix Specification: spécification suivie par les unix commerciaux (et de nombreux non commerciaux) modernes. Proche de la norme POSIX.
- se limiter à SUSv3/POSIX garantit une compatibilité maximale avec les unix utilisés de nos jours
- SUS: http://www.unix.org/what_is_unix/single_unix_specification.html
- De nos jour, il est conseillé d'utiliser un shell compatible posix/sus: ksh, bash et zsh.

Boucle d'interprétation

- Le shell est un programme qui réalise la boucle suivante :
  - Boucle :
    - Lire la ligne de commande
    - Décoder la ligne de commande
    - Exécution de la ligne de commande en créant un processus dans le cas de commandes externes
    - Attendre la fin de l'exécution du processus
    - Retourner en début de boucle

commandes simples: forme générale



- arguments:
  - paramètres optionnels permettant de modifier le comportement de la commande
  - liste des entités auxquelles doit s'appliquer la commande (nom de fichier, processus, utilisateur, ...)
- Exemples:
  - mozilla
  - mozilla -P toto www.univ-evry.fr
  - ls -lrt /etc
  - find ~ -name *.avi -exec rm -f {} \;

quelques commandes simples

- who: liste des utilisateurs ayant une session en cours sur l'ordinateur
- w: idem mais indique aussi ce qu'ils font
- date: date courante
- echo: affiche ses arguments séparés par une espace

Exemples

- Le shell va servir à lancer des commandes internes ou externes
- Exemple de session :

```
#un commentaire commence par #
# lister les fichiers présents
# dans le dossier /etc
ls -l /etc
# liste des utilisateurs connectés
# sur l'ordinateur
who
```

Erreurs

- 5 causes classiques d'erreurs
  1) syntaxe ou chemin incorrect (commande inconnue, ...)
  2) paramètres incorrects (fichier inconnu, ..)
  3) droits d'accès : permission refusée (accès à un fichier, ...)
  4) options invalides (syntaxe des options de la commande)
  5) erreur de conception : le comportement n'est pas celui attendu

•A l'aide

- le manuel
- option « --help » de certaines commandes
- la documentation de votre système d'exploitation ou du programme posant problème
  - souvent /usr/share/doc, /usr/local/share/doc
- recherche sur le WeB: quelqu'un d'autre a forcément déjà eu ce problème

•Le manuel

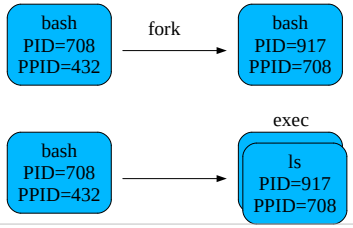
- dans une version ultérieure de ce document
- les sections du manuel: ràf
- exemples d'utilisation: ràf + exemple dans 2 sections

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

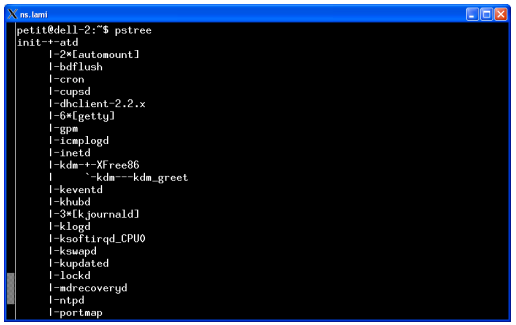
- un processus (processus fils) est toujours créé par un autre processus (processus père):
  - fork: création d'une copie du processus père
  - exec: recouvrement par le processus fils



Hiérarchie de processus

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

pstree



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 entreés-sorties
- dynamique
  - priorité
  - nice
  - consommation cpu/mémoire
  - dossier de travail

commande ps

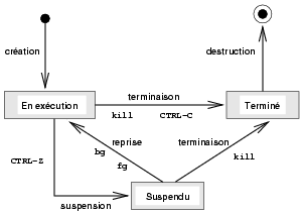
- 2 syntaxes:
  - syntaxe Systeme V: option précédées de -
  - syntaxe BSD: options NON précédées de -
  - quelques options SysV:
    - -e ou -A: tous les processus
    - -a: tous les processus associés à un terminal
    - -H: représentation hiérarchique (forêt)
    - -f: format complet;-l: format long (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

- Cliquez pour ajouter un plan

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

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 accèder
- 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

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 traîtées de gauche à droite
- du -sk /var/* > /tmp/resultat 2> /tmp/erreur
- en cas de redirection simultanée avec cette syntase: 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

<<

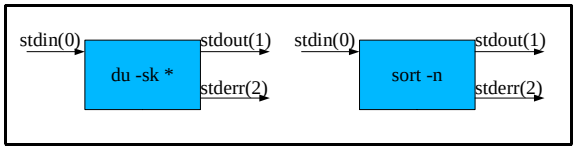
- dans une version ultérieure de ce document

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

Bilan

- A la fin de cette première séance, vous devez :
  - connaître les notions de systeme de gestion de fichiers, fichiers, dossier, chemin
  - connaître la forme générale d'une commande
  - savoir utiliser le manuel unix
  - savoir vous déplacer dans une arborescence,
  - créer/déplacer/copier des fichiers, des dossiers
  - connaître la notion de processus, d'arborescence de processus, les caractéristiques d'un processus, entrée et sortie standard
  - comprendre la notion d'enchaînement de commandes

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 triées 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
- ls | wc -l : donne le nombre de fichier du dossier courant

grep, egrep & Co

- grep chaine: sélectionne les lignes qui contiennent la chaine
- grep petit /etc/passwd: sélectionne les lignes de /etc/passwd contenant la chaîne petit
- caractères spéciaux de la commande egrep:
  - ^: début de ligne
  - $: fin de ligne
- grep '^petit:' /etc/passwd: sélectionne les lignes commençant par petit:

cut

- sélectionner certaines colonnes

tr

- permet de remplacer des caractères par d'autres
- tr 'abcdefghijklmnopqrstuvwxyz' 'nopqrstuvwxyzabcdefghijklm'

more/less

- Cliquez pour ajouter un plan

find

- Cliquez pour ajouter un plan

commandes qui ne lisent pas leur entrée standard

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

Bilan

- A la fin de cette première séance, vous devez :
  - connaître la forme générale d'une commande
  - savoir utiliser le manuel unix
  - connaître la notion de processus, d'arborescence de processus, les caractéristiques d'un processus, entrée et sortie standard
  - comprendre la notion d'enchaînement de commandes
  - comprendre et savoir utiliser sur des exemples de base les redirections et les enchaînements de commandes
  - comprendre ce qu'est un filtre
  - connaître quelques filtres de base