Informatique pour l'entreprise

Système d'exploitation

Marie Pelleau & Olivier Baldellon
marie.pelleau@univ-cotedazur.fr,
olivier.baldellon@univ-cotedazur.fr

23 janvier 2023

Plan du cours

- 4 cours
- 8 travaux pratiques
- 4 thèmes :
  - Système d'exploitation - UNIX
  - Gestionnaire de versions
  - Base de virtualisation
  - Compilation
- Contrôle des connaissances :
  - 2 Projets

Système informatique

Système informatique = Matériels + Logiciels

Matériels

- **Traitement de l'information** : ordinateurs, serveurs, etc.
- **Entrée /Sortie** : Moniteurs, scanners, imprimantes, etc.
- **Stockage de l'information** : disques et mémoires de stockage et d'archivage
- **Communication** : cartes réseaux, commutateurs, routeurs, support de transmission

Logiciels

- **Système d'exploitation** : gestion d'un ordinateur
- **Services** : Systèmes graphiques, communication réseaux, etc.
- **Logiciels** : Traitement de texte, Impression, Compilateurs, Interpréteurs, etc.

Système d'exploitation (*Operating System*)

Ensemble de programmes qui :

- Met à disposition les ressources matérielles de l'ordinateur
- Sert d'interface entre l'utilisateur et l'ordinateur
- Fait abstraction des spécificités complexes du matériel

Système d'exploitation (*Operating System*)

Plusieurs classes

- mono/multi-tâche : partage du temps de calcul du processeur entre plusieurs programmes
- mono/multi-utilisateur : plusieurs utilisateurs peuvent accéder à l'ordinateur simultanément
- distribué : gère plusieurs ordinateurs simultanément et répartit l'utilisation des ressources sur ce réseau
- embarqué : dédié à l'utilisation sur des ressources plus limitées (mémoire, capacité de calcul)
- temps-réel : assure des temps de réponse prédictibles (respect des échéances temporelles)

Historique

UNIX

- 1971 : première distribution. Multi-tâche, multi-utilisateur
- 1973 : première version portable en C (défini pour l'occasion)
- Projet GNU (1983) : objectif de développer un SE libre
- Linux (1991) : un noyau UNIX libre développé par Linus Torvald $\Rightarrow$ premier OS complet GNU/Linux libre
- Distributions GNU/Linux : Debian, Ubuntu, RedHat...
- Mac OS X (1999) : dédié aux ordinateurs Macintosh
- Android (2007) : embarqué
- RTAI, RTLinux : temps-réel

Historique

DOS/WINDOWS

- MS-DOS (1981) : pour le 1er PC d'IBM. Mono-tâche, mono-utilisateur
- Windows 1.0 (1985) : multi-tâche, mono-utilisateur
- Windows NT (1993) : multi-tâche, multi-compte mais un seul utilisateur simultanément
- Windows CE (1996) : embarqué
- Windows 7 (2009) : multi-tâche, multi-compte

Bibliographie

Systèmes d'exploitation

Andrew Tanenbaum, 3e édition, *Broché*, 2008

Unix, Linux et les systèmes d'exploitation

Michel Divay, 2e édition, *Broché*, 2004

Langages de commandes

Objectif

Apprendre à se servir efficacement d'un terminal de commandes :

- Utilisation de la ligne de commande
- Commandes classiques

Interpréteur de commandes

Interface (textuelle) entre l'utilisateur et le système

- Exécution de programmes et de commandes
- Contrôle de l'environnement
- Redirection des entrées/sorties
- Gestion de variables
- Traitement des caractères spéciaux
- Fournit un langage de programmation (script shell)

Interpréteur de commandes

Bourne Shell et dérivés (UNIX)

- sh : bourne shell (shell original)
- bash : bourne again shell
- ksh : korn shell
- zsh : Z shell

C Shell et dérivés (UNIX)

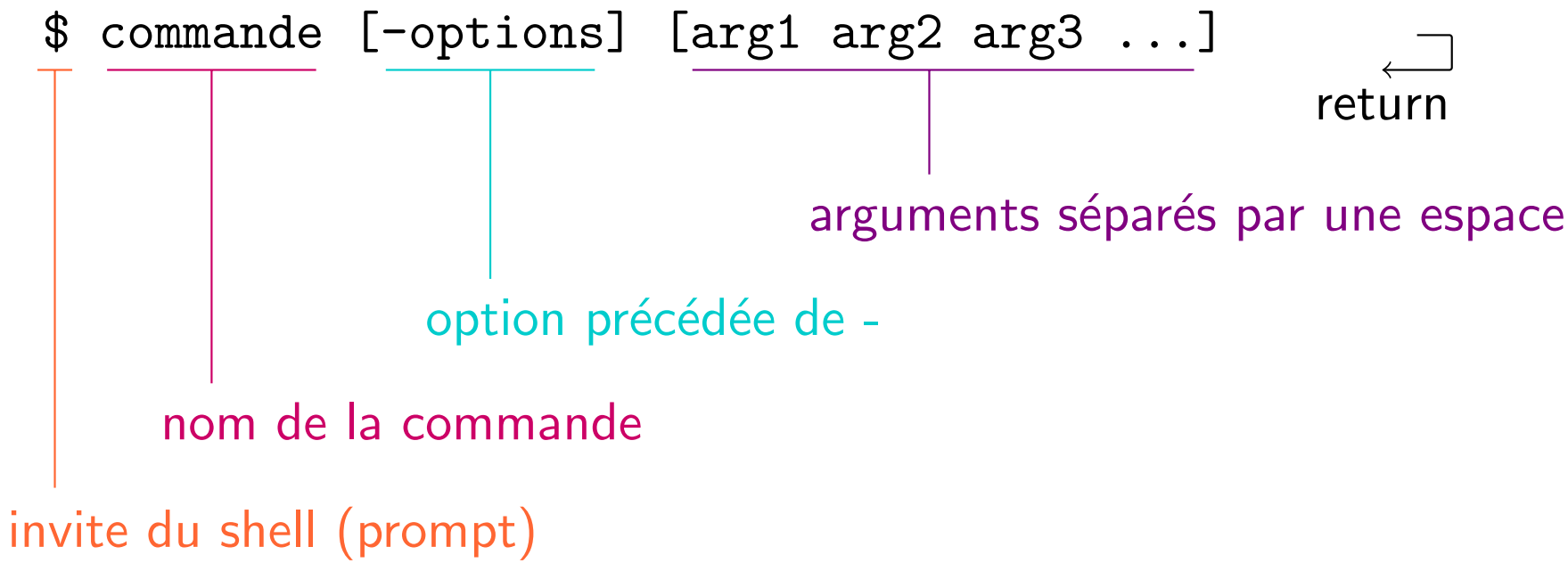
- csh : C shell, développé par Berkeley
- tcsh : C shell amélioré

Microsoft

- DOS : command.com
- Windows : cmd.exe

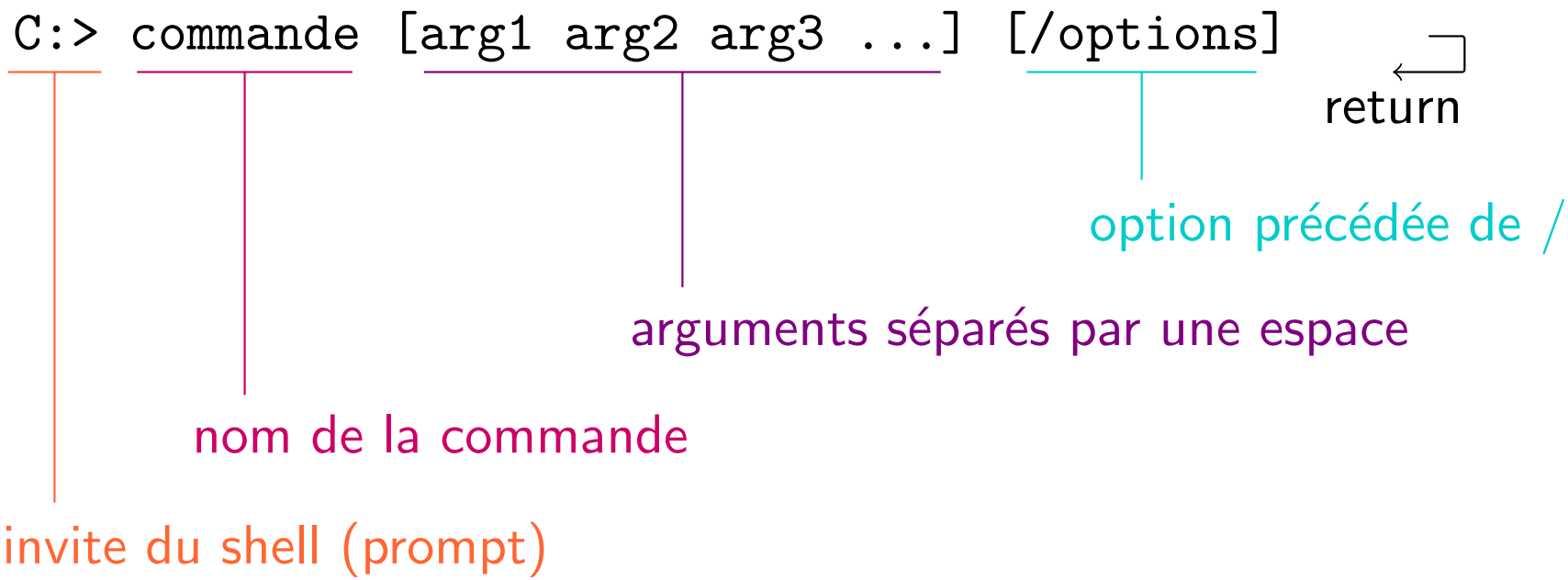
Format d'une commande

Format général d'une commande (UNIX)

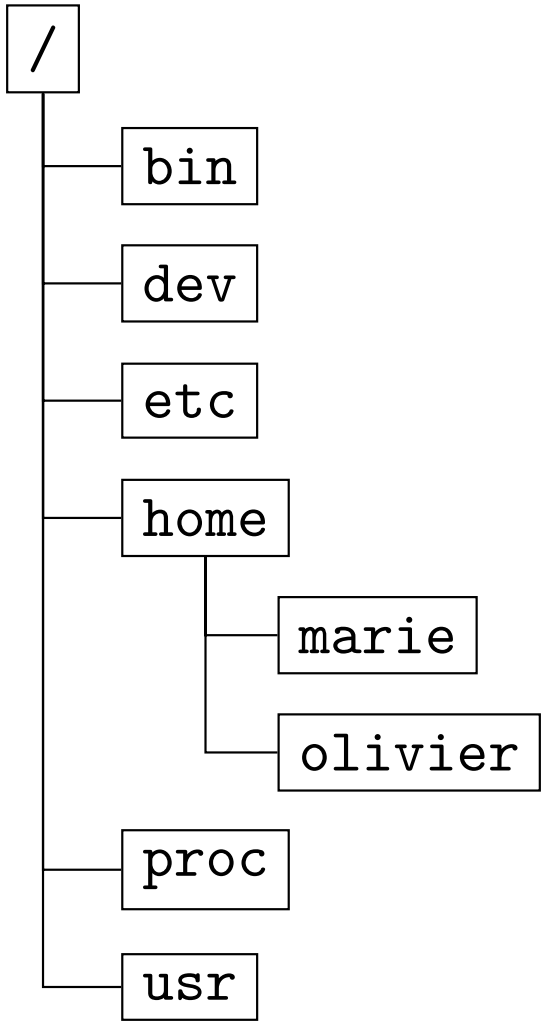


Format d'une commande

Format général d'une commande (DOS)



Arborescence Linux typique



- `/` : la racine
- `/bin` : commandes système
- `/etc` : configuration
- `/home` : répertoires de connexion des utilisateurs

Les caractères spéciaux du SHELL

Chemins

- `~` désigne le répertoire `home` de l'utilisateur
- `.` désigne le répertoire courant
- `..` désigne le répertoire parent

Expressions : les jokers

- `?` 1 caractère quelconque
- `*` 0, un ou plusieurs caractères
- `[ ]` caractère parmi un ensemble
- `[^ ]` caractère en dehors d'un ensemble

Lister les fichiers

Lister les fichiers du répertoire courant

- `$ ls`

Lister les fichiers du répertoire `/bin`

- `$ ls /bin`

Lister les fichiers `.pdf` du répertoire courant

- `$ ls *.pdf`

Exercice 1

Que font ces commandes :

- `$ ls /bin/m*`
- `$ ls /bin/m[a-g]*`
- `$ ls /bin/m[^a-g]*`
- `$ ls /bin/m[^a-g]??[a-z]`

Création/renommage de fichier/répertoire

Renommer un fichier

- `$ mv <fich1> <fich2>`

Création de répertoire(s)

- `$ mkdir [-p] <rep>[/<rep>]`

l'option `-p` : crée les répertoires parents si besoin

Renommer un répertoire

- `$ mv <rep1> <rep2>`

Déplacer un fichier

- `$ mv [-fiu] <fich1> <fich2>`

Diverses commandes sur les fichiers

Supprimer des répertoires

- `$ rmdir [-if] -r <rep1> ... <repn>`
- `$ rm -rf <rep1> ... <repn>`

Copier plusieurs fichiers vers un répertoire

- `$ cp [-if] <fich1> ... <fichn> <rep>`

Copier un répertoire récursivement

- `$ cp [-if] -r <rep1> ... <repn> <rep>`

Supprimer des fichiers

- `$ rm [-if] <fich1> ... <fichn>`
  - `-i` : confirmation avant destruction
  - `-f` : force la destruction des fichiers sans autorisation d'écriture

Flux d'entrée/sortie

Sous UNIX un processus est créé avec 3 flux (canal de communication) :

- Entrée standard (`stdin`, flux numéro 0) : lecture des données d'entrée, utilisé par exemple par `scanf`
- Sortie standard (`stdout`, flux numéro 1) : écriture des données de sortie, utilisé par exemple par `printf`
- Sortie erreur (`stderr`, flux numéro 2) : écriture des données d'erreur, utilisé par exemple par `perror`

Exemple

- `stdout` avec la commande `ls`
- `stderr` avec la commande `ls` (avec un répertoire inexistant)

Redirection des entrées/sorties

Il est possible de changer la destination des flux d'entrée/sortie

- `<` redirige `stdin`
- `>` redirige `stdout`
- `2>` redirige `stderr`
- `>>` ajoute `stdin` (en fin de fichier)
- `2>>` ajoute `stderr` (en fin de fichier)
- `>&` redirige `stdout` et `stderr`
- `2>&1` redirige `stderr` sur `stdout`

Exemple

- `ls /bin/m* > bin_commands`
- `ls /bin/p* > bin_commands`
- `ls /bin/l* >> bin_commands`

Les variables

Affectation

- `<var>=<valeur>`

Désigner la valeur d'une variable

- `$<var>`

Affecter une variable d'environnement

- `export <var>=<valeur>`

Les variables prédéfinies

Quelques variables d'environnement Unix

`HOME`	répertoire racine de l'utilisateur
`SHELL`	nom du shell utilisé par l'utilisateur
`PATH`	répertoire des commandes
`UID, USER`	identité de l'utilisateur
`MANPATH`	chemin pour trouver les pages du manuel
`PS1`	invite utilisée par le shell

Manuel d'une commande

`man <commande>`

Le man décrit (entre autre)

- La manière d'appeler la commande (Synopsis)
- Le rôle de la commande et toutes ses options

Remarque

Vous ne connaissez pas une commande ?
Vous ne connaissez pas une option ?
Utilisez le man !

Fichier

`touch [-option] [-r référence | -t date] fichier`

- `touch` modifie la date d'accès et la date de modification d'un fichier
- les fichiers n'existant pas sont créés, leur contenu est vide ⇒ souvent utilisé pour créer des fichiers vides

`cat > <fich>`

- Saisie d'un fichier au clavier
- Fin de saisie avec Ctr + D

Affichage

Afficher le contenu d'un fichier

- `cat <fich>`

Afficher page par page

- `more <fich>`

Afficher le contenu de plusieurs fichiers

- `cat <fich1> ... <fichn>`

Afficher un texte

- `echo <chaine>`

Filtres

Donner les dernières lignes d'un fichier

- `tail [-n number] <fich>`

Par défaut `n` vaut 10

Donner les premières lignes d'un fichier

- `head [-n number] <fich>`

Par défaut `n` vaut 10

Compter le nombre de lignes, de mots et de caractères des fichiers

- `wc [options] <fich>`

Filtres

grep

`grep [options] <expression> [<fichier>]`

- Permet de sélectionner des lignes de fichier contenant un texte ou motif (cf expressions régulières) donné
- Quelques options (voir `man` pour d'autres options)
  - `-i` pas de différence entre majuscule et minuscule
  - `-c` compte le nombre de lignes
  - `-v` inverse le résultat

Expressions régulières

- `.` un caractère quelconque
- `*` 0 à n fois le caractère qui précède
- `[ ]` un des caractères entre crochets
- `[^ ]` un des caractères qui n'est pas entre crochets
- `^` début de ligne
- `$` fin de ligne

Exemple

- `a*`
- `ab*`
- `.*`
- `^[a-z][0-9]*$`

Séquence de commandes

Séquence simple ;

- `echo "Commandes en k : " ; ls /bin/k*`

Séquence avec redirection |

La sortie d'une commande est redirigée vers l'entrée de la suivante

- `ls /bin/k* | wc`

Autres

Connexion sur un ordinateur hôte (distant) de manière sécurisée

- `ssh [-l username] host`

Récupérer un fichier d'un serveur

- `wget`

Passer en mode super-utilisateur

- `su` ou `sudo`

Changer les droits d'accès

- `chmod`

Connaitre les dernières commandes utilisées

- `history`