Chapitre 6

Programmmation Shell sous UNIX / Linux

s_ouatik@yahoo.com

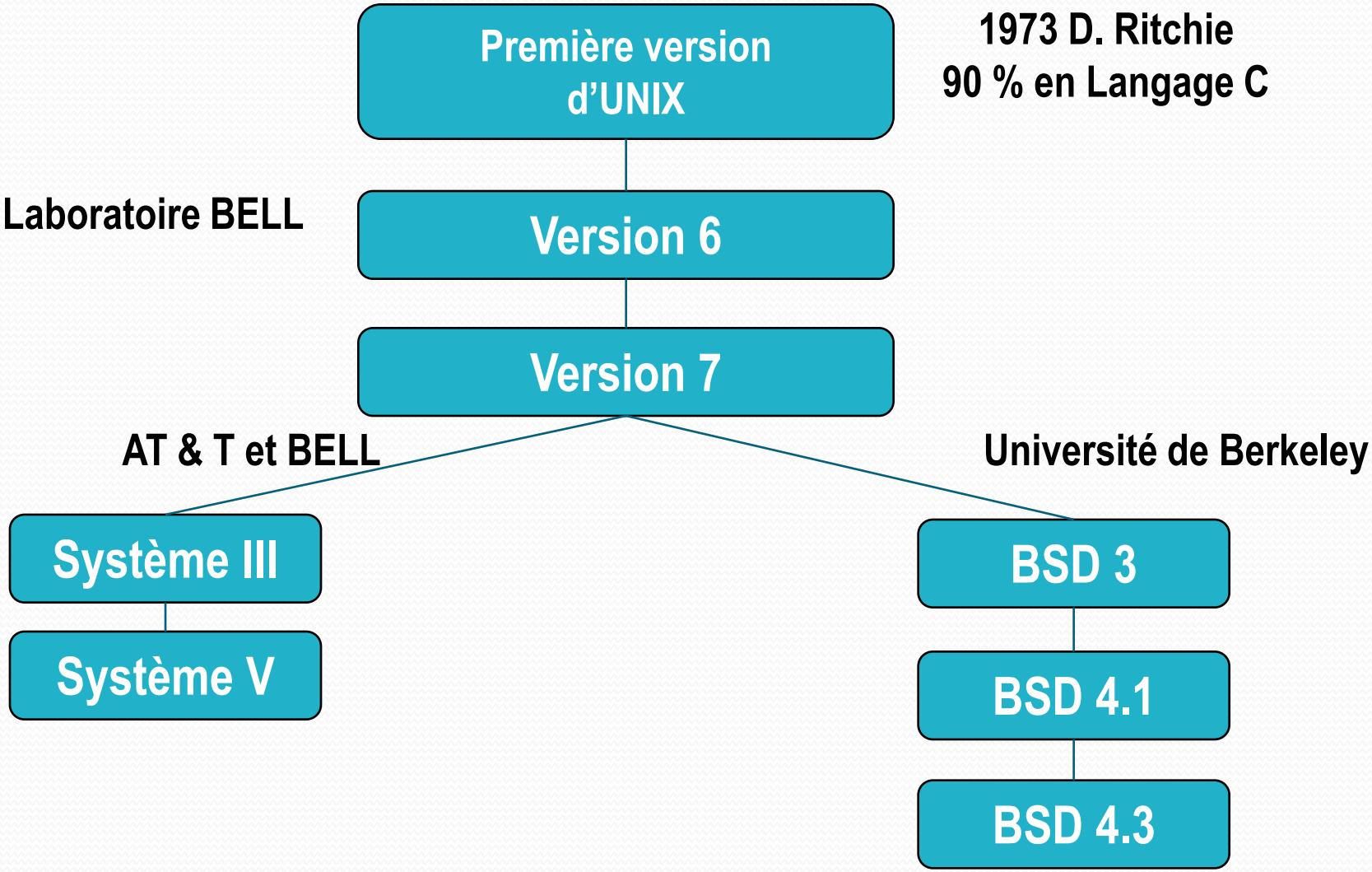
Plan

- Historique d'UNIX
- Caractéristiques d'UNIX
- SHELL : sh
  - Commandes de base
  - Fichiers de commandes: Les Scripts

Historique

- Projet du Système MULTICS (MULTiplexed Information and Computing Service) vers la fin des années 60 entre les laboratoires de BELL et General Electric
  - Un système à temps partagé
  - Ce projet a été abandonné par BELL
- Ken THOMSON, chercheur à BELL, a décidé d'écrire une version allégée de MULTICS en assembleur sur une machine PDP-7
  - Ses collègues ont surnommé cette version, par plaisanterie, UNICS (Uniplexed Information and Computing Service)
  - Le travail était impressionnant et tout le département a rejoint le projet
- UNIX a été porté sur d'autres machines PDP11-20 PDP11-45 PDP11-70
  - Réécrire UNIX dans un langage de haut niveau pour faciliter le portage sur d'autres architectures
    - Réalisation du Langage B qui a été remplacé, par la suite, par le langage C

Historique


Première version d'UNIX
1973 D. Ritchie
90 % en Langage C
Laboratoire BELL
Version 6
Version 7
AT & T et BELL
Université de Berkeley
Système III
Système V
BSD 3
BSD 4.1
BSD 4.3

Historique

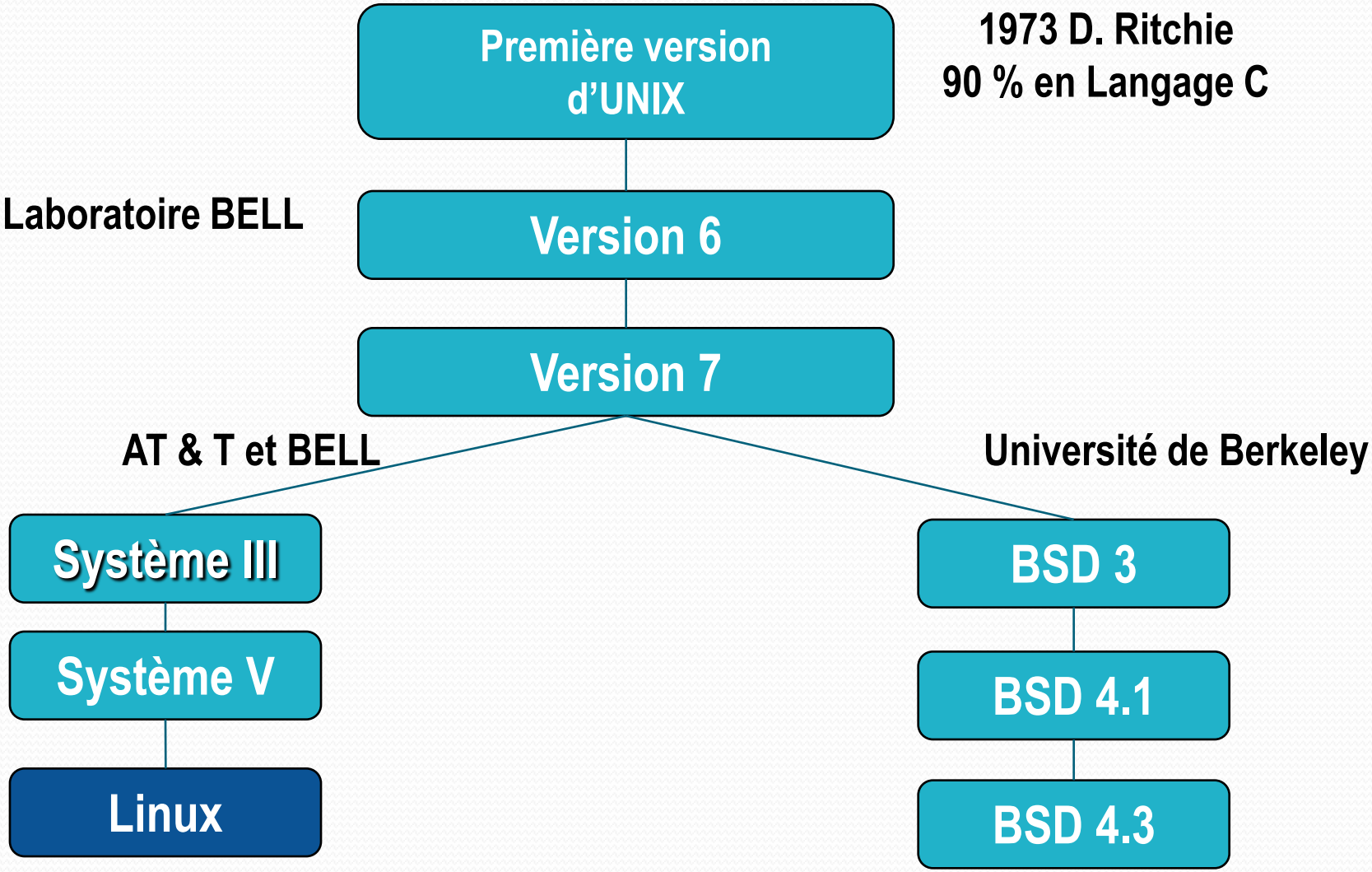
- Vers la fin des années 80, deux versions largement utilisées et sensiblement incompatibles d'UNIX
  - **System V release 3**
  - **BSD4.3** (Berkeley Software Distribution)

- Chaque constructeur ajoutait ses propres améliorations
- Cette incompatibilité a freiné le succès commercial du système
- ➔ la standardisation d'UNIX devient une nécessité

Historique

- Standardisation
  - POSIX (Portable Open System Interface eXchange) membre de IEEE
    - Définition d'un ensemble de procédures que doit fournir tout système compatible à la norme
      - Intersection des deux familles
      - Ressemble fortement à leur ancêtre version 7
    - Norme IEEE P1003.1 devenue norme ISO 9945

Historique


Première version d'UNIX
1973 D. Ritchie
90 % en Langage C
Laboratoire BELL
Version 6
Version 7
AT & T et BELL
Université de Berkeley
Système III
Système V
Linux
BSD 3
BSD 4.1
BSD 4.3

Caractéristiques d'UNIX

- Multi-tâches Multi-utilisateurs
- Sécurité (login, mot de passe)
  - Seuls les utilisateurs ayant un login et un passwd peuvent se connecter au système
- Meilleure protection
  - droits d'accès au niveau des fichiers
- Portabilité
  - Disponible pour plusieurs plateformes (Station de travail, PC, Macintosh)

Caractéristiques d'UNIX

- Modularité
  - Noyau
  - Utilitaires
- Système de fichier
  - Arborescent
  - Réparti
  - Réorganisation souple
- Traitement uniforme des périphériques
  - Un périphérique est traité comme un fichier

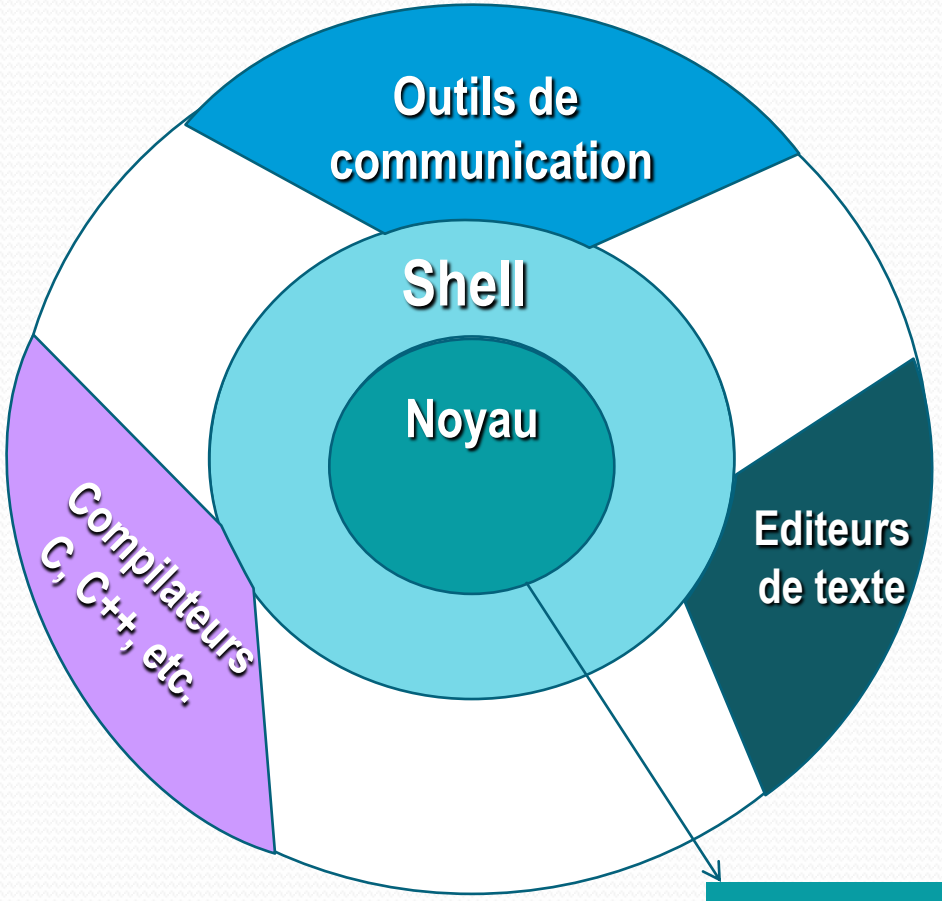
Caractéristiques d'UNIX

- Outils de communication intégrés
  - Talk, write, mail …
- Système de commandes
  - Très riche
  - Puissant
- Plusieurs interpréteurs de commandes
  - Exemples : sh, ksh, csh, …
  - Inter chargeables sans redémarrer la machine

Caractéristiques d'UNIX

- Interface graphique conviviale
  - Système de fenêtrage proche de Windows
  - Exemples : KDE, GNOME, …
- Stable
  - Plante rarement
  - Il y a toujours moyen de débloquer le processus qui bloque le système

Architecture d'UNIX



Gestion des processus

Gestion de la mémoire

Gestion des E/S

Shell

SHELL

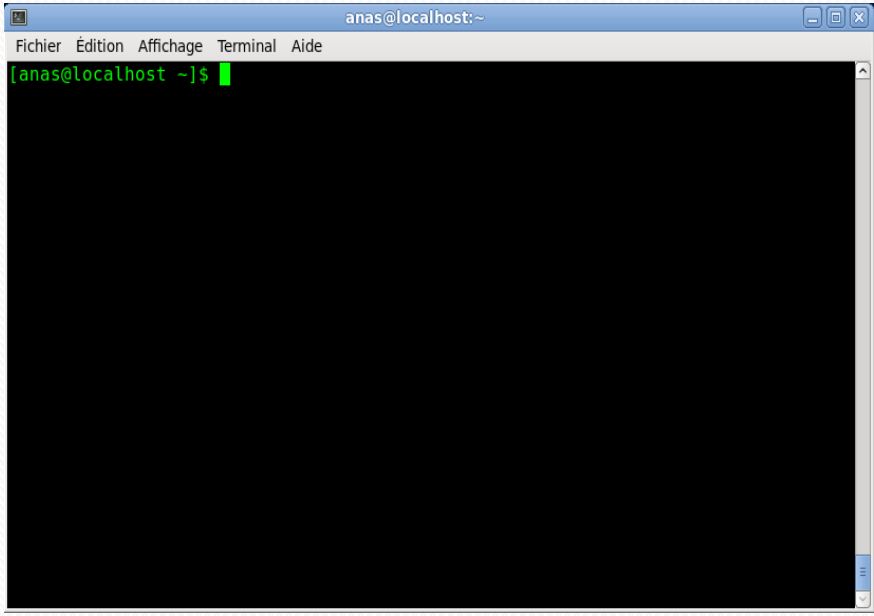
- Le shell est un interpréteur de commandes : il permet à l'utilisateur de dialoguer avec le système.
  - C'est le programme généralement exécuté lorsqu'un utilisateur se connecte. Il affiche un "prompt" (l'invite).
  - Attend les commandes de l'utilisateur.

- Le shell est aussi un langage de programmation interprété puissant.
  - Il offre à l'utilisateur un environnement composé d'un ensemble de variables et un langage de commandes.

Principaux SHELLs

- Il existe différents shells :
  - sh : Bourne Shell (shell standard unix)
  - ksh : Korn Shell
  - csh : C Shell
  - tcsh : extension de C Shell
  - bash : GNU (Bourne advanced Shell)

SHELLs



- Bash : le shell par défaut
  - Le shell bash fonctionne au sein d'un terminal.
    - Espace de saisie de commande
    - Espace d'affichage des résultats des commandes
  - Tout utilisateur de Linux et d'Unix en général a au moins un terminal ouvert en quasi-permanence.
  - L'ouverture d'un terminal (ou console) lance automatiquement le shell par défaut

Utilisation du shell

- Commandes
  - Taper une commande à l'invite du shell
- Script
  - Un fichier de commande

Utilisation du shell

- **Commandes internes et externes**

- **Il existe deux types de commandes :**
  - **Les commandes externes sont des programmes binaires présents en tant que fichiers sur votre disque dur. Quand vous exécutez la commande, ce fichier est chargé en mémoire et lancé en tant que processus.**
  - **Les commandes internes sont internes au shell et exécutées au sein de celui-ci. Ces commandes font partie du programme shell. Les commandes history ou pwd sont deux exemples de commandes internes.**

- **Vous pouvez distinguer une commande interne d'une commande externe à l'aide de la commande interne type.**

Utilisation du shell

- **Aides en ligne**

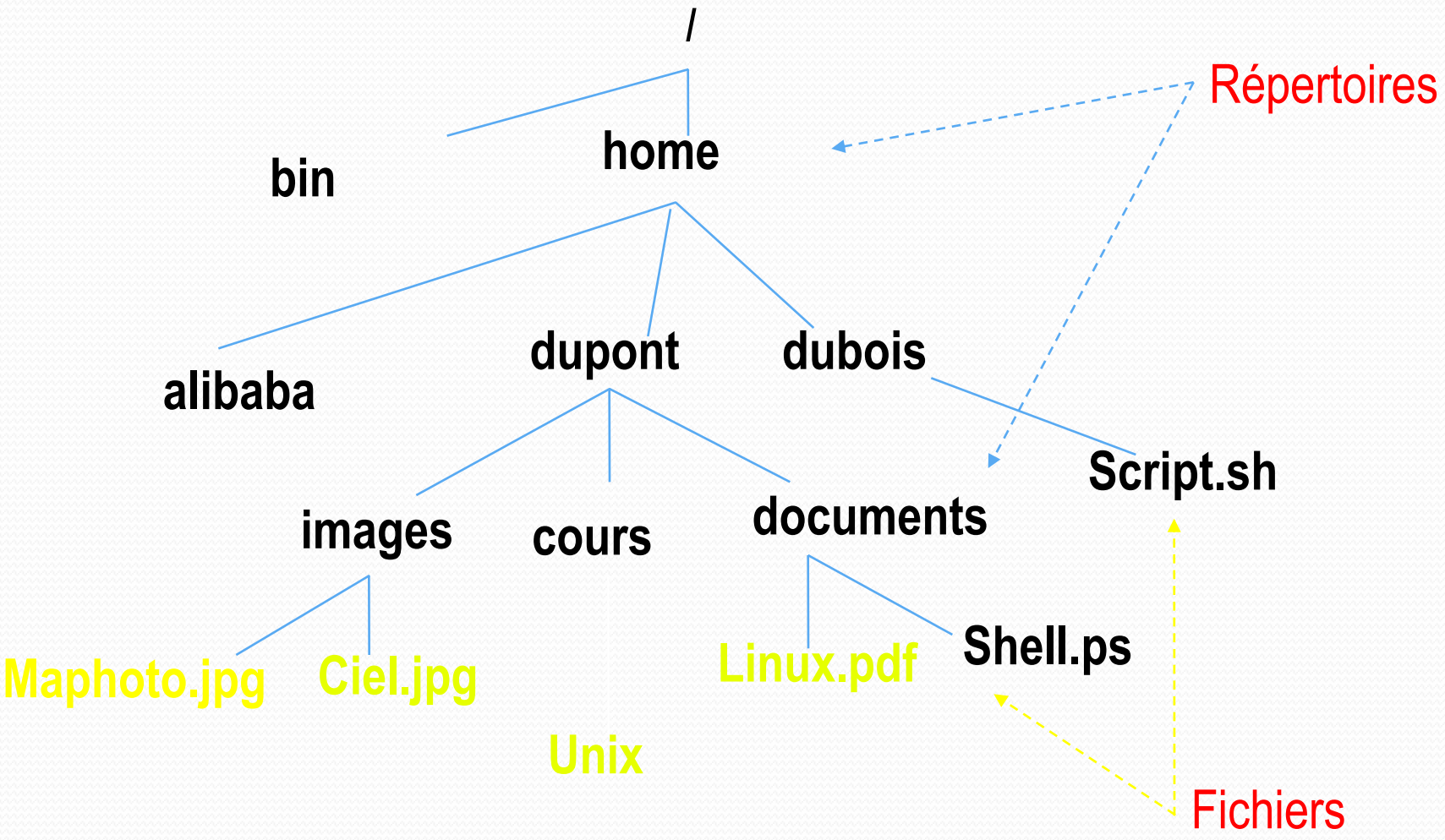
- **La commande "man" (pour manual) fournit des informations (description, options, syntaxe) sur une commande UNIX ou une application donnée .**

- **Exemple :**
  - **$ man firefox**

Organisation du SGF

- Organisation logique
  - Structure arborescente
    - Fichier : contient les données
    - Répertoire : noeud contenant des fichiers ou d'autres sous répertoires

Exemple d'arborescence du SGF



Chemin de fichier/répertoire

- Chemin d'accès
  - l'ensemble des nœuds qui mènent vers un fichier ou un répertoire séparés par '/'.
  - . : répertoire courant
  - .. : répertoire père

Chemin absolu/relatif

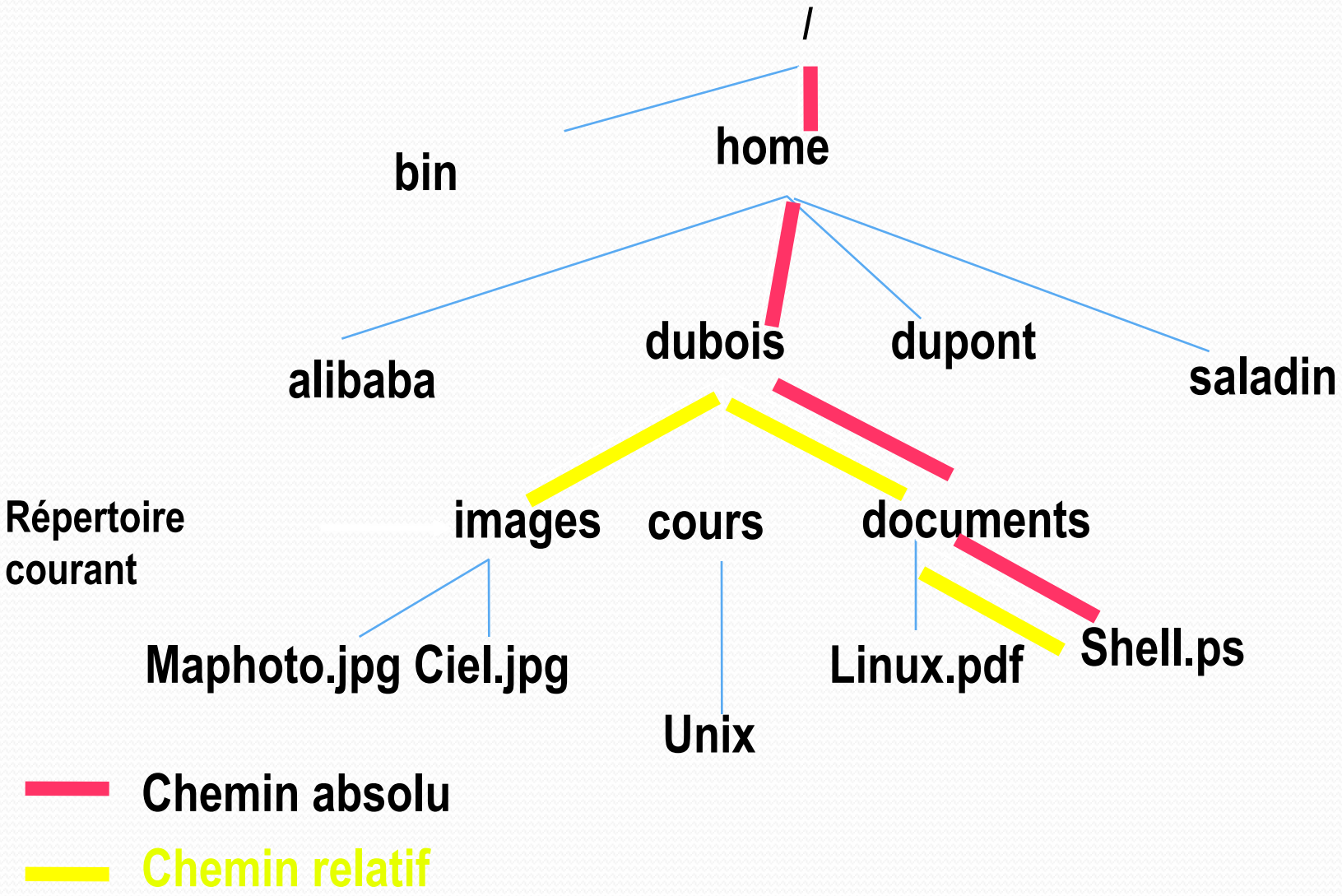
- Chemin absolu
  - commence à la racine : '/'
  - Exemple

    /***home***/***dubois***/***documents***/***shell.ps***
- Chemin relatif
  - Commence à partir du répertoire courant
  - Exemples

    ciel.jpg

    ../documents/shell.ps

Arborescence



Manipulation des fichiers et répertoires

- Identificateur (nom)
  - Suite de caractères (jusqu'à 255 caractères)
  - Sensible à la casse
  - Utiliser le caractère d'échappement '\' pour les caractères spéciaux
    - Exemple précéder le caractère espace par \
      - Nom\ fichier

Manipulation des fichiers et répertoires

- Caractères "joker"
  - Permettent d'appliquer une commande à un ensemble de fichiers dont le nom vérifie certaines contraintes (ex : le nom commence par la lettre 'p', l'extension est '.doc', ...)
    - * : remplace n'importe quelle suite de caractère (y compris la chaîne vide)
    - ? : remplace un et un seul caractère
    - [-] : définit un intervalle

Manipulation des fichiers et répertoires

- ls
  - Affiche le contenu du répertoire courant ou de celui passé en paramètre
  - Options
    - -l : affiche les informations complètes des fichiers et sous répertoires
    - -a : affiche les fichiers cachés
    - -R : affichage récursif
    - -i : affiche le descripteur des fichiers (i-numéro)
    - -d : n'affiche pas le contenu des répertoires

Manipulation des fichiers et répertoires

- **Information sur un i-nœud**
- **Pour obtenir quelques informations sur un i–nœud on utilise la commande stat :**
  - **Syntaxe : stat Nom_fichier ou répertoire**
  - **Exemple : $ stat file1**
  - **Options :**
    - **- f : obtenir des informations sur le système de fichier**

Manipulation des fichiers et répertoires

1	2	3	4	5	6	7
-rw-r--r--	1	root	root	44	2010-01-22 23:10	adjtime

- **Lister les fichiers et les répertoires**
  - **1 : Le premier caractère représente le type de fichier (- : ordinaire, d : répertoire, l : lien symbolique…) les autres, par blocs de trois, les droits pour l'utilisateur (rw-), le groupe (r--) et les autres (r--).**
  - **2 : Un compteur de liens physiques.**
  - **3 : Le propriétaire du fichier, généralement celui qui l'a créé.**
  - **4 : Le groupe auquel appartient le fichier.**
  - **5 : La taille du fichier en octets.**
  - **6 : La date de dernière modification avec l'heure.**
  - **7 : Le nom du fichier.**

Manipulation des fichiers et répertoires

- pwd
  - Affiche le chemin du répertoire courant
- cd chemin
  - Se déplace vers le répertoire identifié par 'chemin'
  - Exemples

    $ cd /home/dubois/doc

    $ cd ../dubois/doc

Manipulation des fichiers et répertoires

- **Créer des fichiers vides**
  - **la commande qui permet de créer des fichiers est : touch. Utilisée avec uniquement le nom d'un fichier en argument, elle crée un fichier avec une taille nulle.**
  - **Syntaxe : $ touch nom_fich1 nom_fich2**
  - **Exemple : touch fich1**

Manipulation des fichiers et répertoires

- mkdir (md) nouveau_rep
  - Crée un nouveau_rep dans
    - le répertoire courant
      - nouveau_rep est le nom du répertoire
      - Exemple

        $ mkdir stages
    - Dans le chemin indiqué par la première partie de nouveau_rep
      - Exemple
      - $ md stages
      - $ md /home/dupont/stages
      - $ mkdir ../dupont/stages

Manipulation des fichiers et répertoires

- rmdir repertoire
  - Détruit un répertoire vide
  - Exemple
    - $ *rmdir /home/dupont/temp*

Manipulation des fichiers et répertoires

- rm fichier
  - Détruit un fichier ou un répertoire non vide
  - Options
    - -r : la commande détruit de manière récursive toute la sous arborescence du répertoire
    - -i : demande la confirmation avant de supprimer le fichier
  - Exemple
    - *$ rm -r doc*
    - *$ rm -i /home/dupont/linux.pdf*

Manipulation des fichiers et répertoires

- cat fichier [fichier,…]
  - Concatène et affiche sur la sortie standard le(s) fichier(s) en paramètre
  - Exemple
    - $ cat fichier1
    - $ cat fichier1 fichier2

Manipulation des fichiers et répertoires

- more fichier
  - Affiche le contenu du fichier page par page
  - Utilisée pour les fichiers longs (contenant plusieurs pages)
    - Q : quitte la commande
    - Return : saute de ligne
    - Espace : saute de page
  - Exemple
    - $ more lettre

Manipulation des fichiers et répertoires

- head [-c nchar -n nline] fichier
  - Affiche le début du fichier
    - Par défaut les dix premières lignes
    - -c nchar : affiche les nchar premiers caractères du fichier
    - -n nline : affiche les nline premières lignes du fichier
  - Exemple
    - *$ head lettre*
    - *$head -c 280 lettre*
    - *$ head -n 5 lettre*

Manipulation des fichiers et répertoires

- tail [-/+c nchar -/+n nline] fichier
  - Affiche la fin du fichier
    - Par défaut les dix dernières lignes
    - -/+c nchar : affiche les derniers caractères du fichier

      + à partir du nchar ème caractère jusqu' à la fin du fichier

      - Les nchar derniers caractères à partir de la fin
    - -/+n nline : affiche les dernières lignes du fichier

      + à partir de la nline ème jusqu'à la fin du fichier

      - Les nline dernières lignes à partir de la fin
  - Exemple
    - *$ tail lettre*
    - *$tail -n 6 lettre* *affiche les 6 dernières lignes*
    - *$ tail +n 6 lettre* *affiche de la ligne 6 jusqu'à la fin du fichier*

Manipulation des fichiers et répertoires

- wc [-lwc] fichier
  - Compte le nombre de

    -l : lignes

    -w : mots

    -c : caractères

    du fichier
  - Par défaut les trois
  - Exemple
    - *$ wc lettre*

Manipulation des fichiers et répertoires

- cp source destination
  - Copie le fichier source dans le fichier ou répertoire destination
- mv source destination
  - Déplace le fichier source dans le répertoire destination (et le renommer)
- Exemple
  - *$ cp lettre correspondances/lettre1*
  - *$ cp lettre correspondances*
  - *$ mv lettre correspondances*

Manipulation des fichiers et répertoires

- ln source lien
  - Crée un lien physique sur le fichier source
    - Pas possible pour les répertoires
  - -s : le lien est symbolique (possible pour fichier ou dossier)
  - Exemple
    - $ ln lettre lien_lettre
    - $ ln lettre -s lien_symbolique

Manipulation des fichiers et répertoires

- Commande tar
  - Archive un ensemble de fichiers dans un seul fichier (d'extension '.tar')
    - Facilite l'organisation (moins d'encombrement dans le SGF)
    - Efficace pour envoyer, par mail, plusieurs fichiers en attachement
  - Restitue l'ensemble des fichiers à partir du fichier archive (l'opération inverse)
  - Possibilité de compression et de décompression de l'archive en appelant la commande "gzip"

Manipulation des fichiers et répertoires

- Commande tar
  - Syntaxe
    - *$ tar [options] [fichiers]*

Option	Description
-x	Extraire le contenu d'une archive.
-c	Créer une nouvelle archive.
-t	Afficher seulement la liste du contenu de l'archive, sans l'extraire.
-f *Fichier*	Indiquer le nom du fichier archive.
-v	Mode verbeux, affiche le détail des opérations.
-z	Compresser ou décompresser en faisant appel à l'utilitaire gzip.
- j	Compresser ou décompresser avec l'utilitaire bzip2.
-p	Préserver les permissions des fichiers.

Manipulation des fichiers et répertoires

- Commande tar
  - Exemples
    - Créer une archive
      - *$ tar –cvf archive_doc.tar /home/ali/doc*
    - Créer une archive et compression
      - *$ tar –cvzf archive_doc.tar.gz /home/ali/doc*
    - Lister le contenu d'une archive
      - *$ tar –tvf archive_doc.tar*
    - Extraire le contenu d'une archive
      - *$ tar –xvf archive_doc.tar /home/ali/cours*
      - *$ tar –xvzf archive_doc.tar.gz* (extraction dans le répertoire courant)

Manipulation des fichiers et répertoires

- Commande find
  - Cherche un fichier dans une arborescence donnée
  - Les critères de recherche peuvent porter sur :
    - Le nom
    - La dernière date d'accès
    - La dernière date de modification
    - Le type (fichier, répertoire ou lien)
    - …

Manipulation des fichiers et répertoires

- Commande find
  - Syntaxe
    - *$ find [chemin] [options]*

Option	Description
-atime *+n (-n)*	Trouve les fichiers accédés il y a plus/moins de *n* jours
-mtime *+n (-n)*	Trouve les fichiers **modifiés** il y a plus/moins de *n* jours
-name "pierre"	Trouve les fichiers dont le nom est pierre
-maxdepth *n*	Définit le niveau *n* maximum de sous répertoire à explorer
-type < d | f | l >	Indique le type de fichier à rechercher. *type* peut être: -d : pour les répertoires - f : pour les fichiers - l : pour les liens symboliques

Manipulation des fichiers et répertoires

- Commande find
  - Exemples
    - Rechercher les fichiers dont le nom commence par ‘prog’ et d’extention ‘.c’ dans le répertoire /home/ali
      - *$ find /home/ali –name " prog*.c"*
    - Rechercher les sous répertoires du répertoire /home/ali avec deux niveaux de profondeur.
      - *$ find /home/ali –type d –maxdepth 2*
    - Rechercher les fichiers du répertoire /home/ali qui ont été accédés il y a moins d’un mois.
      - *$ find /home/ali –type f –maxdepth 0 –atime -30*

Manipulation des fichiers et répertoires

- Commande grep
  - Cherche les lignes d'un fichier qui contiennent un motif (un mot ou une chaîne de caractères)
  - Affiche ces lignes sur la sortie standard
  - Syntaxe
    - *$ grep motif fichier*
  - Exemple
    - *$ grep "ali" /home/prof/notes.txt*

Edition de texte

- Il existe plusieurs commandes permettant d'éditer un fichier texte: éditeurs de texte
- nano : permet d'éditer un fichier texte en mode console
  - Syntaxe : nano nom_fichier
  - Options:
    - - A active le retour intelligent au début de la ligne
  - Exemple : $ nano fich1
- Emacs
- gedit
- Vi
- ....

Gestion des utilisateurs

- who
  - Affiche les informations sur les utilisateurs connectés
- who am i
  - Affiche les informations de l'utilisateur courant
- whoami
  - Affiche le login de l'utilisateur courant
- id
  - Affiche l'UID et le GID de l'utilisateur courant

Gestion des utilisateurs

- Commande useradd (voir aussi adduser)
  - Limité au root
  - Ajouter un nouvel utilisateur au système
  - Possibilité de spécifier quelques paramètres du nouveau compte
    - Informations concernant l'utilisateur (nom, fonction, …)
    - Durée de validité du compte
    - Groupe
    - Shell
    - …

Gestion des utilisateurs

- Commande useradd

Option	Description
-c <commentaire>	Informations concernant l'utilisateur (nom, poste).
-d <répertoire>	Chemin du répertoire personnel de l'utilisateur.
-e <date>	Date d'expiration du compte. Le format est AAAA-MM-JJ.
-f <nombre de jours>	C'est le nombre de jours suivant l'expiration du mot de passe après lequel le compte est désactivé. La valeur 0 permet de désactiver le compte dès que le mot de passe expire. La valeur -1 permet de désactiver cette caractéristique. La valeur par défaut est -1 .
-g <groupe principal>	Le nom du groupe ou le numéro du groupe de connexion initial de l'utilisateur. Le nom ou le numéro du groupe doivent exister. Le numéro de groupe par défaut est 1.
-G <groupes supplémentaires>	Les autres groupes auxquels appartient l'utilisateur (séparés par des virgules).

Gestion des utilisateurs

- Commande useradd

Option	Description
-m	Le répertoire de l'utilisateur sera créé (par défaut, cela n'est pas toujours le cas).
-k <répertoire>	À utiliser si et seulement si l'option -m est présente. Permet de copier les fichiers et répertoires contenus dans le répertoire modèle </etc/skel si non spécifié) dans le répertoire de l'utilisateur.
-p <mot de passe chiffré>	Vous pouvez saisir le mot de passe en option. Il doit être chiffré (pour récupérer la version cryptée d'un mot de passe il faut utiliser la bibliothèque *crypt).* Si le mot de passe n'est pas défini, le comportement par défaut est de désactiver le compte.
-s <chemin>	Shell lancé à la connexion de l'utilisateur.
-u	L'identifiant unique de l'utilisateur.

Gestion des utilisateurs

- Commande useradd
  - Syntaxe
    - *$ useradd [options] nouvel_utilisateur*
  - Exemple
    - Création du nouveau compte alibaba (login) appartenant au groupe Etudiants avec bash comme shell par défaut
      - *$ useradd –c "Ali BABA" –g Etudiants –s /bin/bash alibaba*

Gestion des utilisateurs

- Commande userdel
  - Supprimer un utilisateur du système
  - Syntaxe
    - *$ userdel [-r] login*
      - Rajouter l'option –r permet de supprimer le répertoire personnel de l'utilisateur en question

Gestion des utilisateurs

- Commande passwd
  - Changer le mot de passe
    - Un utilisateur normal peut changer uniquement son mot de passe
    - L'administrateur peut changer le mot de passe de tous les utilisateurs
  - Verrouiller/déverrouiller un compte
  - Supprimer un mot de passe

Gestion des utilisateurs

- Commande passwd
  - Syntaxe

Option	Description
-k	Indique que seul le mot de passe doit être mis à jour, sans toucher aux propriétés d'expiration
-1	Permet de verrouiller le compte spécifié en préfixant le mot de passe crypté par le caractère « ! ». Seul l'utilisateur *root* peut utiliser cette option.
- - stdin	Le mot de passe doit être lu à partir de l'entrée standard qui peut alors être un pipe.
-u	Déverrouille le mot de passe du compte. Seul l'utilisateur *root* peut utiliser cette option.
-d	Supprime le mot de passe d'un compte. Le champ réservé au mot de passe crypté sera supprimé dans le fichier de configuration. Seul l'utilisateur *root* peut utiliser cette option.
-S	Affiche des informations sur le statut du mot de passe pour un compte donné. Seul l'utilisateur *root* peut utiliser cette option.

Gestion des utilisateurs

- Commande passwd
  - Syntaxe
    - *$ passwd [options] [login]*
    - Si le paramètre login n'est pas spécifié alors, le changement s'applique pour le compte courant. Le système demande de saisir l'ancien et le nouveau mot de passe.
  - Exemple
    - *$ passwd*
    - *$ passwd –d alibaba*

Gestion des utilisateurs

- Commande su
  - Changement momentané de l'identité dans la même session
    - Même utilisateur (qui a plusieurs compte)
    - Un autre utilisateur (connexion rapide)
  - Syntaxe
    - *$ su [-] [login]*
    - Sans login permet de se connecter en tant que root
    - Dans ce cas Le mot de passe de utilisateur *root* est demandé par le système
    - Le tiret '–' permet de récupérer l'environnement de l'utilisateur en question
  - Exemple
    - *$ su* *(se connecter en tant que administrateur, pour sortir exit)*
    - *$ su – alibaba*
- Pour exécuter une commande en tant que *root* sans changer d'*user* :
- sudo *commande*.
- Le mot de passe de utilisateur lançant sudo est demandé par le système.

Gestion des utilisateurs

- Commande groupadd (voir aussi addgroup)
  - Crée un nouveau groupe
  - Syntaxe
    - *$ groupadd [options] nouveau_groupe*

Option	Description
-g	Permet de spécifier le GID du nouveau groupe
-f	Stoppe la commande si le groupe ou le GID existe déjà

Gestion des utilisateurs

- Commande groupdel
  - Supprime un groupe
    - Aucun utilisateur ne doit avoir ce groupe comme principal.
  - Syntaxe
    - *$ groupdel groupe*

Gestion des processus

- ***ps***
  - ***Affiche les informations des processus actifs***
  - ***Options***
    - -l : affiche les informations complètes des processus
    - -x : affiche tous les processus actifs (d'autres utilisateurs)
    - -u : affiche les processus d'un utilisateur donné
  - Exemple
    - $ ps -l
    - $ ps -u dupont

Gestion des processus

- ***kill -signal ident_processus***
  - ***Envoi un signal à un processus***
  - Exemple
    - *$ kill -9 33456*

Droits d’accès aux fichiers

- Déterminent les types d’opérations qu’un utilisateur ou une classe d’utilisateurs peuvent effectuées.
- Chaque fichier peut avoir ses propres droits d’accès
- Système de protection très puissant

Droits d’accès aux fichiers

- Types d’utilisateurs
  - Propriétaire (owner)
  - Groupe du propriétaire (ou du fichier)
  - Les autres utilisateurs (others)

Droits d'accès aux fichiers

- Types d'autorisation des fichiers
  - r : droit d'accès en lecture
  - w : droit de modification
  - x : droit d'exécution (limité aux fichiers exécutables)

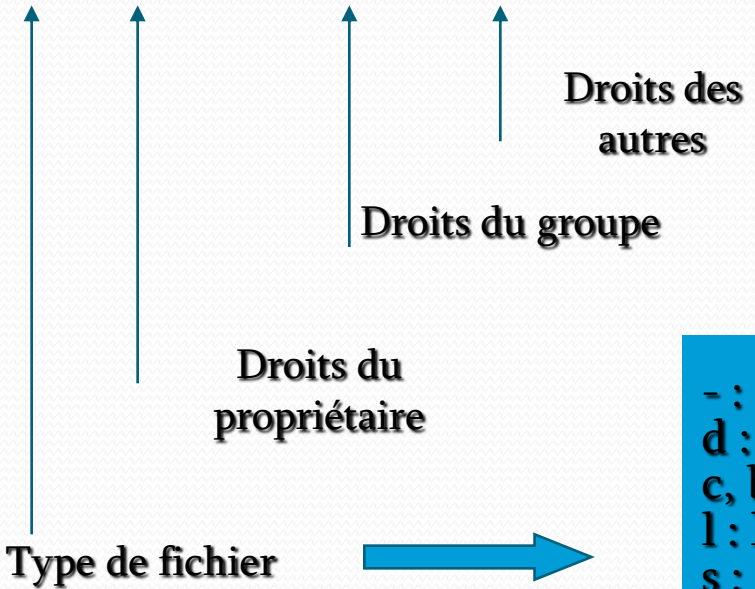
Droits d’accès aux fichiers

- Types d’autorisation des répertoires
  - r : droit d’afficher le contenu
  - w : droit d’ajout et de suppression
  - x : droit d’exploration

Droits d'accès aux fichiers

- Exemple

-	rwx	rw-	--x	monexe
-	rwx	rw-	r--	document



- : fichier ordinaire
d : répertoire
c, b : fichier périphérique
l : lien symbolique
s : socket
p : tube

Droits d'accès aux fichiers

- chmod mode fichier
  - Mode = utilisateurs/permission
    - Exemple
      - *$chmod u+x fich1*
      - *$chmod g-w fich1*
      - *$chmod +r fich1*
  - Mode = chiffres
    - Exemple
      - *$ chmod 754 fich1*

Droits d'accès aux fichiers

- chown nouveau_util fichier
  - Change le propriaitaire du fichier
    - Exemple

      $ chown dubois fich1

- chgrp nouveau_grp fichier
  - Change le groupe du fichier
    - Exemple

      $ chgrp telecom fich1

Redirection des E/S

- > : la sortie standard est redirigée vers un fichier (ecrasement de son contenu s'il existe déjà)
- < : les entrées de la commande proviennent d'un fichier
- >> : la sortie standard est insérée à la fin d'un fichier
- 2> : la sortie d'erreur est redirigée vers un fichier
- 2>> : la sortie d'erreur est insérée à la fin d'un fichier

Le pipe |

- Permet la communication des entrées/sorties entre les processus
- Schéma
  - cmd1|cmd2| cmd3|….
  - Le résultat de cmd1 sera l'entrée de cmd2, le résultat de cmd2 sera l'entréee de cmd3 et ainsi de suite.
- Exemple
  - *$ ls –l | grep "prog"*

script

Définition

Un script est un fichier texte contenant des commandes. L'ordre et le choix de l'exécution des commandes sont déterminés par un ensemble de structures (test, boucles, …)

Etapes d'un script

- Création
  - Créer le fichier de commandes avec un éditeur de texte
  - Sauvegarder le fichier
- Exécution
  - Rendre ce fichier exécutable
    - chmod +x
    - Appelé directement comme n'importe quelle commande
  - Exécuter le Shell sur ce fichier
    - $ sh script

Choix du shell

- Spécifie le shell qui exécutera le script
  - Rajouter une ligne tout au début du script
    - #! nomshell
  - Exemple
    - #! /bin/bash

Commentaire

- Tout texte précédé du caractère '#'
- N'a aucun effet sur l'exécution du script
- Décrit, explique ou commente le script ou une de ses parties
- Exemple

 # ceci est un commentaire

variables

- Identificateur
  - Nom composé de caractères
  - Certains caractères sont interdits ($,#,...)
- Types
  - Numérique
  - Chaîne de caractères
- Accès au contenu
  - Précéder l'identificateur par le caractère : '$'

Variables d'environnement

- SHELL : le shell utilisé
- USER : nom de l'utilisateur
- UID : identificateur de l'utilisateur
- PATH : chemin des répertoires contenant les fichiers exécutables
- HOME : chemin du répertoire d'accueil
- PWD : chemin du répertoire courant
- HOSTNAME : nom de la machine
- …

Variables de positionnement

- $0 : le nom du script
- $# : le nombre de paramètres du script
- $n : le $n^{ème}$ paramètre du script
- $* : liste de tous les paramètres du script
- shift -n : ote les n premiers paramètres
  - Par défaut le premier paramètre

Lecture

- Lecture (read)
  - read permet de lire une ou plusieurs variables à partir de l'entrée standard
  - Syntaxe
    - read var1 [var2, …]
    - Si plusieurs variables à la fois, le contenu saisi sera réparti sur les variables, dans l'ordre, avec espace comme séparateur.
  - Exemples
    - read a
    - read n a

Affichage

- Affichage (echo)
  - echo permet d'afficher une expression sur la sortie standard
  - L'expression peut être :
    - Une ou plusieurs variables (précédées du caractère '$')
    - Une chaîne constante (de préférence entre guillemets)
    - Combinaison des deux
  - Syntaxe
    - echo <expression>
  - Exemples
    - echo $a
    - echo "la valeur de a est : $a"

Affectation

- Affectation : L'opérateur =
  - Syntaxe
    - Ident_variable = <expression>
  - Exemples
    - n=10
    - Nom="dupont"

Opérations arithmétiques

- Commande expr
  - Permet d'exécuter les opérations arithmétiques de base
  - syntaxe
    - ***`expr var1 op var2`***
    - ***op : +,-,*,/,%***
  - **Exemple**
    - ***n=`expr $a + $b`***

Executions des commandes

- *En séquence : ;*
  - *Chaque commande est suivie du caractère ';'*
    - *commande1;commande2;...*
- *En parallèle : &*
  - *Chaque commande est suivie du caractère '&'*
  - *Cmd1 & cmd2 & ....*
- *Pipe : |*
  - *La sortie d'une commande est l'entrée d'une autre*
  - *cmd1 | cmd2 | ...*

Expressions logiques

- ***Commande test***
  - ***Comparaison des variables numériques***
    - ***test v1 opérateur v2***

      -eq : =

      -ne : ≠

      -lt : <

      -le: <=

      -gt : >

      -ge : >=

Expressions logiques

- ***Commande test***
  - ***Comparaison des chaînes de caractères***
    - ***test v1 opérateur v2***

      = : égalité

      != : différence
    - ***Exemple***
      - ***test $USER = 'dupont'***
      - ***test $SHELL='bash'***

Expressions logiques

- ***Commande test***
  - ***Teste si une variable est vide***
    - ***test -z $var : teste si var est vide***
    - ***test -n $var : teste si var n'est pas vide***

Expressions logiques

- ***Commande test***

  - ***Teste sur les fichiers test op nom_fich***

-f : vérifie si nom_fich est un fichier

-s : vérifie si nom_fich est un fichier non vide

-d : vérifie si nom_fich est un répertoire

-r : vérifie si nom_fich est un fichier accessible en lecture

-w : vérifie si nom_fich est un fichier accessible en écriture

-x : vérifie si nom_fich est un fichier accessible en exécution

Expressions logiques

- ***Commande test***
  - ***Combinaison des expressions logiques***
    - ***-a : et logique***
    - ***-o : ou logique***
    - ***! : non logique***
  - ***Exemple***
    - ***test -f nomfichier –a –r nomfichier***

Instruction if/else

- Permet de choisir entre deux blocs de commandes
  - Syntaxe :

```
if condition
    then
        liste_commandes1
else
        liste_commandes2
fi
```

Instruction if/else

- Choix d'un seul bloc si une condition est vérifiée
  - Syntaxe :

```
if condition
    then
        liste_commandes
fi
```

Instruction if/else

- Imbrication de if/else
  - Syntaxe :

```
if condition1
then
liste_commandes1
 elif condition2
        then
        liste_commandes2
      else
        liste_commandes3
 fi
```

Instruction if/else

Exemple

Le script Compare deux nombres passées en paramètres. Si le nombre de paramètres est différent de deux, alors, il affiche un message d'erreur

```
if test $# -ne 2
   then
   nm=`expr 2 - $#`
   echo $nm PARAMETRES MANQUANTS
else
   if test $1 -lt $2
     then
     echo $1 '<' $2
   elif test $1 -gt $2
        then
             echo $1 '>'  $2
     else
       echo $1 '=' $2
   fi
fi
```

Instruction case

- Choisir un bloc de commandes parmi plusieurs
  - Syntaxe :

```
case param in
        cas1[| cas] ...) liste_commandes1 ;;
        cas 2) liste_commandes2 ;;
        ....
esac
```

Instruction case

- Exemple

Affichage du jour passé en paramètre en anglais

```
case $1 in
        lundi|LUNDI) echo "monday" ;;
        mardi) echo "tusday";;
        mercredi) echo "wednesday";;
        jeudi) echo "thursday" ;;
        vendredi) echo "friday";;
        samedi) echo "saturday";;
        dimanche) echo "sunday";;
esac
```

Instruction while

- Exécuter une liste de commandes tant qu'une condition soit vérifiée
  - Syntaxe :

```
while commande
do
liste_commandes
    done
```

Instruction while : exemple

Exemple

Concaténation des fichiers en paramètres dans le fichier "global.sh"

```
while test $# -ne 0
do
  if test -r $1  -a -f  $1
      then
      cat $1 >> global.sh
   fi
   shift
done
```

Instruction until

- Exécuter une liste de commandes jusqu'à ce qu'une condition soit vérifiée
  - Syntaxe :

```
until commande
do
liste_commandes
    done
```

Instruction until : exemple

Exemple

Concaténation des fichiers en paramètres dans le fichier "global.sh"

```
until test $# -eq 0
do
  if test -r $1  -a -f  $1
      then
      cat $1 >> global.sh
   fi
   shift
done
```

Instruction for

- Exécuter une liste de commandes pour chaque paramètre dans une liste
  - Syntaxe

```
for param in liste
do
liste_de commandes
done
```

Instruction for

Exemple

Copier les fichiers du répertoire désigné par le premier paramètre dans un répertoire à créer et qui aura comme nom le deuxième paramètre

```
if test $# -ne 2
   then
   echo PARAMETRE MANQUANT
else
   mkdir $2
   for fic in `ls $1`
   do
     if test -r $1/$fic
  then
  cp $1/$fic  $2
     fi
   done
fi
```