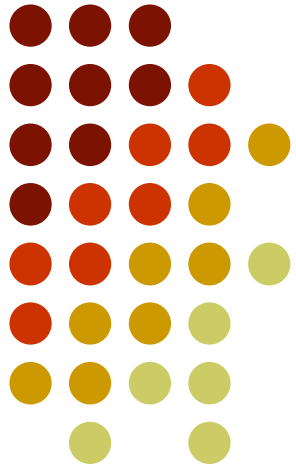
Introduction aux systèmes d'exploitation et à leur fonctionnement R1-04



Informations pratiques

patricia.mely@univ-lorraine.fr

Bureau A36

4H de cours, 18H de TP

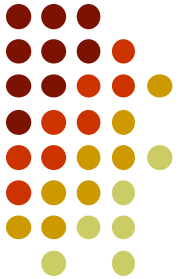
Lié à la SAE S1.03

Introduction système

- Objectifs : acquérir des notions
  - Concernant les systèmes d'exploitation (SE)
  - Ligne de commande
  - Scripts
- Unix / windows : pas de match !
- « Buttons are for idiots », Theo de Raadt, fondateur des projets NetBsd et OpenBsd
  - Automatisme, léger, adaptable

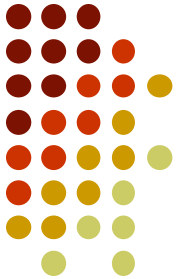
Plan du cours

- Trois chapitres :
  - Concepts de base des SE
  - Le système Unix
  - Le système Windows

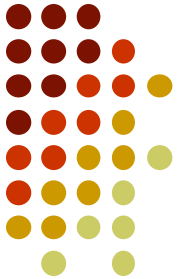


PARTIE 1
CONCEPTS DE BASE DES SE

Concept de base des SE



- Définition d’un SE
- Les rôles et les composants d’un SE
- Les processus
- Organisation des SE : fichiers et répertoires



DEFINITION D'UN SYSTÈME D'EXPLOITATION

C'est quoi ?

Définition d’un SE

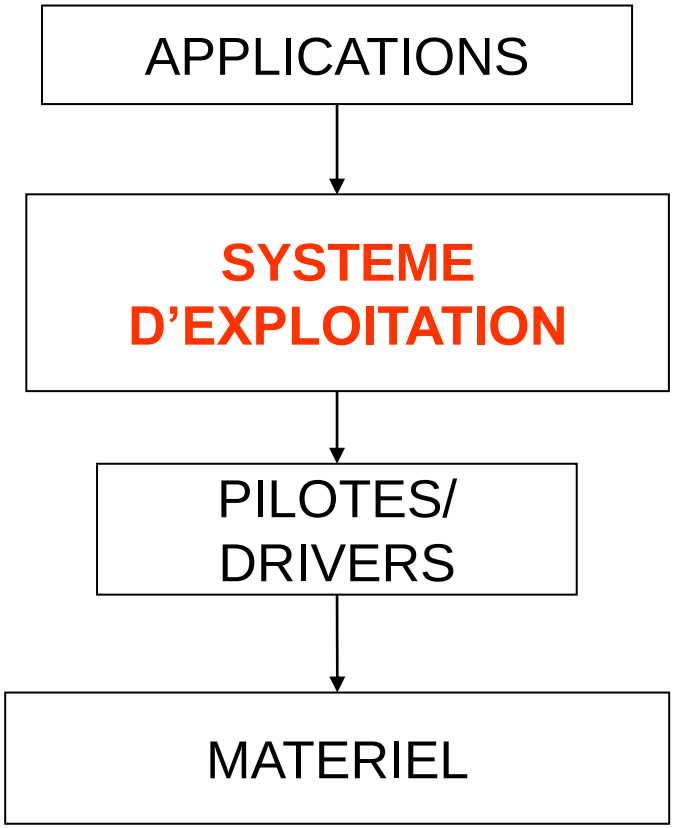
- Un système d’exploitation (OS pour Operating System en anglais) est un ensemble cohérent de logiciels, de programmes permettant d'utiliser un ordinateur et tous ses périphériques à l’utilisateur final

Définition d'un SE

- Logiciel le plus important car il fournit :
  - Une gestion des ressources de la machine
  - Une base pour le développement et l'exécution de programmes d'applications
- Assure le démarrage, la liaison entre les ressources matérielles, l'utilisateur et les applications, l'arrêt propre de la machine

Définition d'un SE

Lorsqu'un programme désire accéder à une ressource matérielle, il ne lui est pas nécessaire d'envoyer des informations spécifiques au périphérique, il lui suffit d'envoyer les infos au SE, qui se charge de les transmettre au périphérique concerné via son pilote.



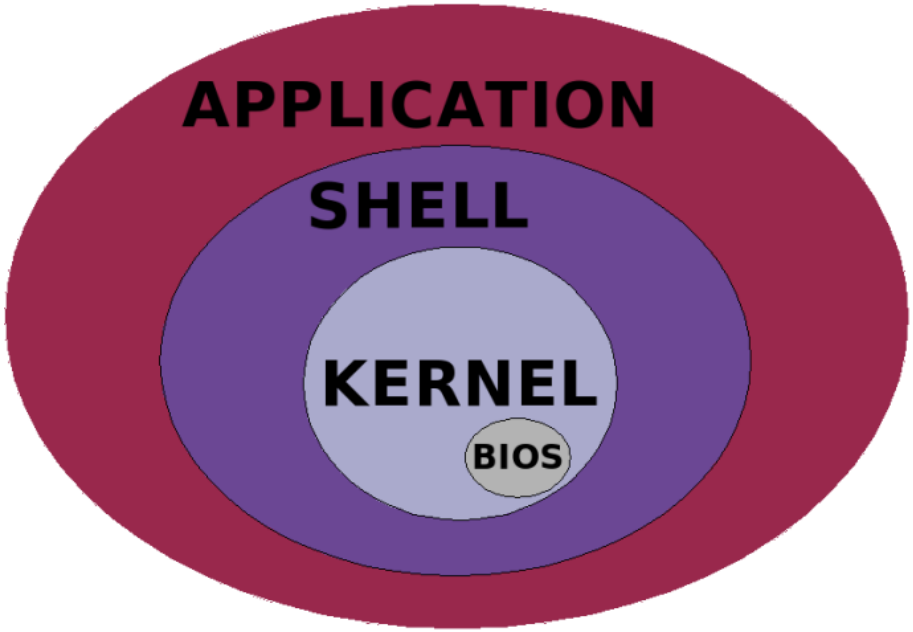
Définition d'un SE

- Exemple : partage de l'imprimante
  - Pouvoir verrouiller l'accès à l'imprimante, afin que les flots de caractères produits par les programmes désirant imprimer ne s'entrelacent pas sur le papier
  - Gérer des tampons d'impression, afin que les programmes puissent reprendre leur travail sans devoir attendre la fin effective de l'impression
  - Utilisation d'un langage de commande : pouvoir imprimer un document (lpr doc)

Définition d’un SE

La communication avec le système d'exploitation s'établit par l'intermédiaire d'un **langage de commandes** (appel à commande) **et d’un interpréteur de commandes** (logiciel interactif) → Shell

Cela permet à l'utilisateur de piloter les périphériques en ignorant tout des caractéristiques du matériel qu'il utilise.

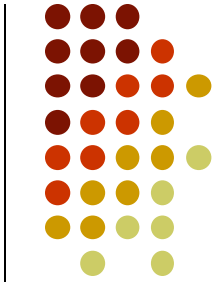


Source : linux-france.org

Bios : Basic Input Output System, recensement des ressources matérielles

LES COMPOSANTS et LES ROLES D’UN SE

Composants d'un SE



- Un SE est composé d'un ensemble de logiciels permettant de gérer les interactions avec le matériel. Parmi cet ensemble de logiciels on distingue généralement les éléments suivants :
  - Le **noyau** (en anglais **kernel**) représentant les fonctions fondamentales du système d'exploitation telles que la gestion de la mémoire, des processus, des fichiers, des entrées-sorties principales, et des fonctionnalités de communication.
  - L'**interpréteur de commande** (en anglais **shell**, traduisez «*coquille*» par opposition au noyau) permettant la communication avec le système d'exploitation par l'intermédiaire d'un langage de commandes, afin de permettre à l'utilisateur de piloter les périphériques en ignorant tout des caractéristiques du matériel qu'il utilise, de la gestion des adresses physiques, etc.
  - Le **système de fichiers** (en anglais «*file system*», noté *FS*), permettant d'enregistrer les fichiers dans une arborescence
  - Les utilitaires

Les rôles d’un SE

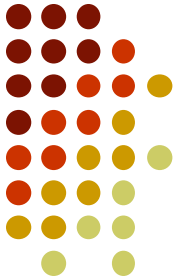
- La gestion de la mémoire : plusieurs types de mémoire, partagée
- La gestion du processeur : partagé, temps processeur en tranches, processus
- La gestion des périphériques (E/S) : ressources matérielles, gestion de ces ressource
- La gestion des fichiers : stockage, utilisation, sécurité
- La gestion des tâches.
- La gestion des pannes.
- La gestion de la sécurité

Types de SE

- Mono-utilisateur et multiutilisateurs
- Mono-tâches et multitâches
- Les SE multitâches permettent de partager le temps processeur pour plusieurs programmes ou processus, ainsi ceux-ci sembleront s'exécuter simultanément (ordonnancement des tâches)
- Un système multitâche peut permettre à plusieurs utilisateurs de travailler ensemble, il est alors dit multiutilisateurs

LES PROCESSUS

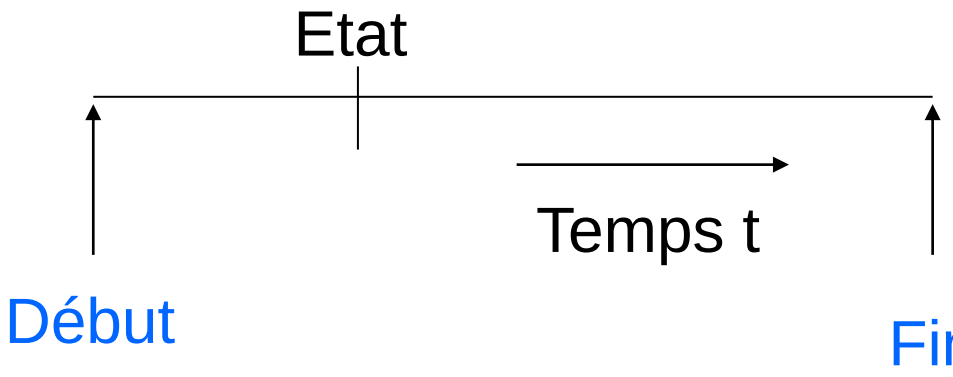
Les processus



- L’exécution d’une commande donne naissance à un processus (donc programme en train de s’exécuter)
- Définition : un processus est défini par :
  - un ensemble d'instructions à exécuter (un programme)
  - un espace mémoire pour les données de travail
  - un environnement : E/S, variables, etc
  - Son propriétaire

Les processus

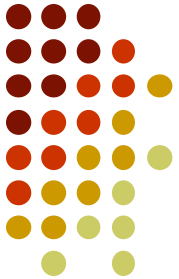
- Entité dont le noyau contrôle l'état, de la vie à la mort
- Le processeur est ainsi partagé entre plusieurs concurrents (simultanés) : système à temps partagé



Les processus

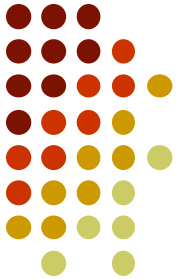
- Plusieurs états :
  - Actif : s'exécute
  - Prêt : peut s'exécuter
  - Bloqué (en attente) : attente d'une ressource (E/S)
- Le noyau découpe en tranche de temps pour permettre d'exécuter plusieurs processus

Les processus



- Processus identifié par un numéro (PID)
- Deux types de processus : système et utilisateur
- Possibilité d’avoir un processus père et un processus fils (clonage)
- Environnement d’un processus : variables

Les processus



- Daemon ou démon
  - Processus qui s’exécute en arrière-plan plutôt que sous le contrôle direct de l’utilisateur
  - Souvent démarrés au chargement du SE
- Thread ou processus léger
  - Processus léger, correspondant à l'exécution d'un petit programme, ou d'une routine d'un programme plus gros, indépendamment de celui-ci (on parle alors de multithread). Généralement partage la même mémoire

ORGANISATION D’UN SE

Organisation des SE
Les systèmes de fichiers

- Système de fichiers (FS : File System) : stockage à long terme
- Mémoires secondaires : fichiers sous forme de blocs (suite de chiffres binaires)
- Fichiers regroupés dans des répertoires
- Souvent sous forme d'arborescence (Windows, Unix, Linux), de librairie (AS400, mainframe)
- Notion de chemin
  - Répertoire racine et sous-répertoires

Organisation des SE
Fonction du SGF

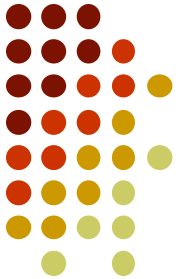
- SGF : système de gestion de fichiers
  - Manipulation des fichiers : créer, lire, modifier, supprimer un fichier
  - Allocation de la place
  - Localisation des fichiers
  - Sécurité et contrôle des fichiers

Organisation des SE : Les différents types de fichiers

- FAT : File Allocation Table
  - Assez simple, basique
  - Nombre limité des noms de fichiers
  - Tables situées à un emplacement fixe
  - Fat-12 : soit 4 096 fichiers (2^{12})
  - Fat-16 : soit 65 536 fichiers (2^{16})
  - Fat-32 : soit 268 millions de fichiers (2^{28})
  - Ex : Windows 95, Windows 98
- NTFS : New Technology File System
  - Table de fichiers maître
  - Mettre des droits spécifiques (ACL)
  - Compression des fichiers
  - Chiffrer des fichiers
  - Configurer des quotas
  - Rapidité (arbre binaire), noms de fichier
  - Ex : Windows NT4, Windows 2000/XP, Vista, Windows 7, Windows 8, Windows 10

Organisation des SE
Les différents types de fichiers

- Système de fichiers à inode (ou index)
  - UFS (Unix File System), ZFS
    - Solaris, OpenBSD
  - FFS (Fast File System)
    - Variante de UFS
    - OpenBSD
  - JFS (Journaled File System)
    - IBM Aix
  - Ext, Ext2, Ext3, Ext4, Btrfs : Linux
    - Ext2 : limite la fragmentation
    - Ext3 : journalisation
    - Ext4 : plus gros fichiers, + rapide
    - Btrfs : améliore la fragmentation, + rapide



PARTIE 2
Le système Unix

Le système Unix

- Historique Unix
- La gestion de fichiers
- Le shell :
  - Langage de commandes
  - Les commandes composées
  - Les variables d’environnement
  - Les scripts
- Les processus

HISTORIQUE UNIX

Historique Unix

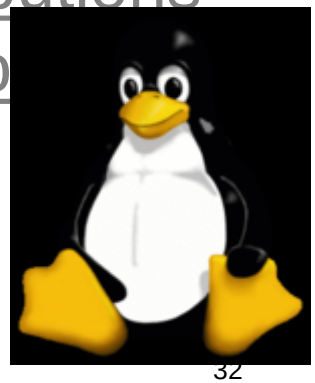
- 1969 : Ken Thompson (laboratoire de Bell) écrit la première version de ce qui devait s'appeler Unix (Unics). Fonctionne sur un DEC PDP-7
- 1971 : Ritchie et Thompson réécrivent le noyau Unix en C
- 1974-1977 : code source Unix est distribué librement aux universités
- 1978 : Unix, 7ème version. Portage sur différents architectures. Version de Kernighan et Pike, laboratoire Bell.

Historique Unix

- 1979 : commercialisation d'Unix par ATT Université de Berkeley : BSD UNIX
- 1983 : ATT met en vente son système V
- 1984 : Richard Stallman lance le projet GNU, Unix entièrement libre
- 1987 : les différentes versions s'écartent les unes des autres
- 1988 : création des normes POSIX
- 1988 : OSF (Open System Fondation) par IBM, Hewlett-Packard, BULL, SIEMENS et Apollo

Historique Unix :

- 1991 : Linux par Linus Torvalds sur la base de Minix (Tannenbaum)
- 1996 : premières distributions GNU (Gnu is not Linux) avec Richard Stallman
- 1999 : Apple annonce Mac OS x 1.0
- 2004 : création d'Ubuntu par Mark Shuttleworth
- https://fr.wikipedia.org/wiki/Liste_des_distributions_GNU/Linux#/media/Fichier:Linux_Distribution_Timeline.svg



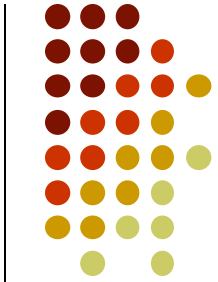
Version Unix :

- Solaris/OpenSolaris dirigé par Sun
- FreeBSD
- NetBSD, FreeBSD
- HP-UX de HP (BSD et system V)
- AIX d’IBM (system V)

Version Linux :

- Plusieurs versions de Linux :
  - Debian, éditée par une communauté de développeurs ;
  - Mandriva, éditée par la société française de même nom et impliquée dans plusieurs projets libres ;basé sur Red Hat
  - Red Hat, éditée par l'entreprise américaine du même nom qui participe également au développement de Fedora Core
  - OpenSuse
  - Ubuntu, basé sur Debian, Kubuntu
  - Linux Mint
  - etc

Norme POSIX



- **Norme POSIX (Portable Operating System Interface**)
  - famille de standards définie depuis 1988 par l'IEEE et formellement désignée IEEE 1003
  - spécifie dans près de 15 documents différents les interfaces utilisateurs et les interfaces logicielles

LA GESTION DES FICHIERS SOUS UNIX

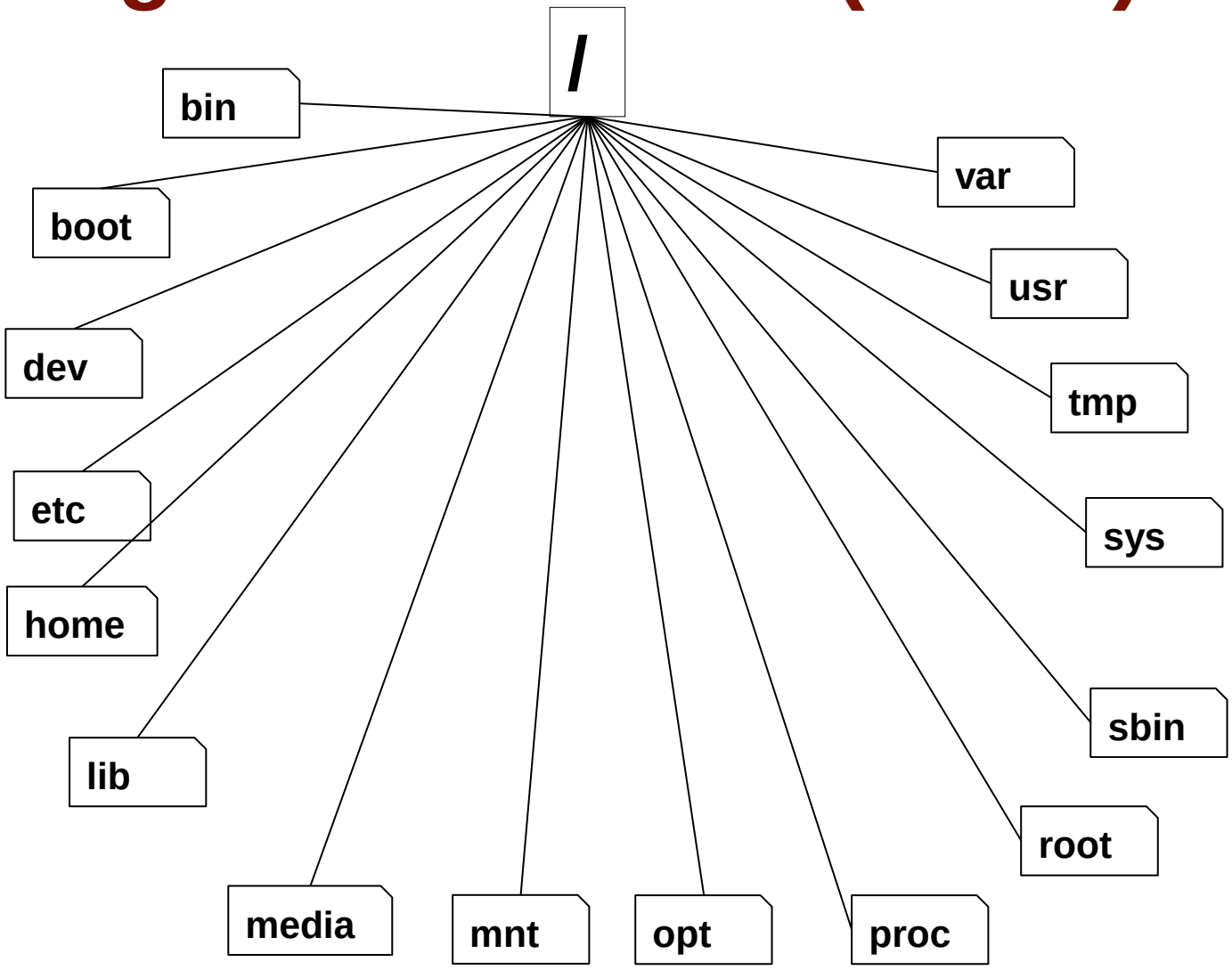
Système de gestion de fichiers

- Gère une structure d'arbre
  - La racine est désignée par /
  - Les nœuds sont les répertoires non vides
  - Les feuilles sont les fichiers
- Les chemins sont décrits avec le séparateur /
  - Exemple : /home/eleves/toto est un référence absolue
    toto/tp1.sh est une référence relative

Différents types de fichiers

- Fichiers ordinaires :
  - Données, programmes, textes
- Fichiers spéciaux
  - Correspondent à des ressources
    - /dev/hda, /dev/cdrom, /dev/null, etc
  - Les opérations de lecture/écriture sur ces fichiers activent les dispositifs physiques associés
- Fichiers répertoires
  - Permettent la structure en arbre
- Fichiers de liens symboliques
  - Alias entre fichiers

Organisation Unix (Linux)



Organisation Linux

Répertoire	Contenu
/bin	Commandes de base du système, d'administration
/boot	Fichiers du noyau Linux
/dev	Fichiers particuliers aux périphériques
/etc	Fichiers et répertoires de configuration du système
/home	Répertoires personnels des utilisateurs
/lib	Bibliothèques des ss-pgm
/media	Médias montés et éjectables comme les CDs, USB
/mnt	Répertoires de périphériques amovibles
/opt	Contient des progiciels
/proc	Contient des informations liées au système et aux processus
/root	Répertoire personnel de l'admin sous Linux
/sbin	Binaires systèmes essentiels
/sys	Fichiers système
/tmp	Fichiers temporaires
/usr	Principal répertoire du système
/var	Répertoire contenant la partie variable du système (logs, mail, spool,etc)

Les chemins

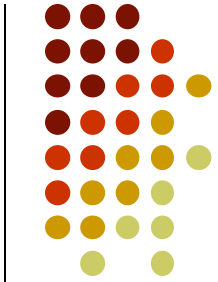
- Chemin absolu
  - Atteindre l'objet à partir de la racine
  - /home/dupont/tp/tp.txt
- Chemin relatif
  - Précise l'itinéraire à partir du répertoire de travail
  - Si je suis sur le répertoire /home/dupont → tp/tp.txt

Attention : notion de répertoire courant (de travail) et de répertoire de login ($HOME)

Notion d'utilisateur et de groupe

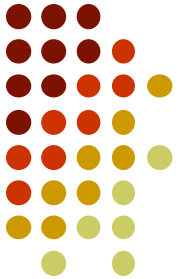
- Il existe un super-utilisateur : **root**
  - Tous les pouvoirs
  - Accès à toutes les ressources
  - *root* porte le numéro id 0
  - En général, seul l'administrateur possède le mot de passe *root*
- Notion de groupe : un utilisateur appartient à un groupe (droits communs)
  - GID : Group Identification User
- Un fichier a toujours un propriétaire (user)
  - UID : User Identification User

Gestion des utilisateurs



- **sudo** permet d'exécuter une commande en tant qu'un autre utilisateur (prendre les droits root)
- **adduser** ajout d'un nouvel utilisateur
- **passwd** modifie le mot de passe
- **deluser** détruit un utilisateur
- **who** utilisateurs connectés au système
- **finger** idem who avec plus de détail
- **last** derniers utilisateurs connectés
- **lastb** dernières tentatives de connexion qui ont échoué

Attributs d’un fichier :

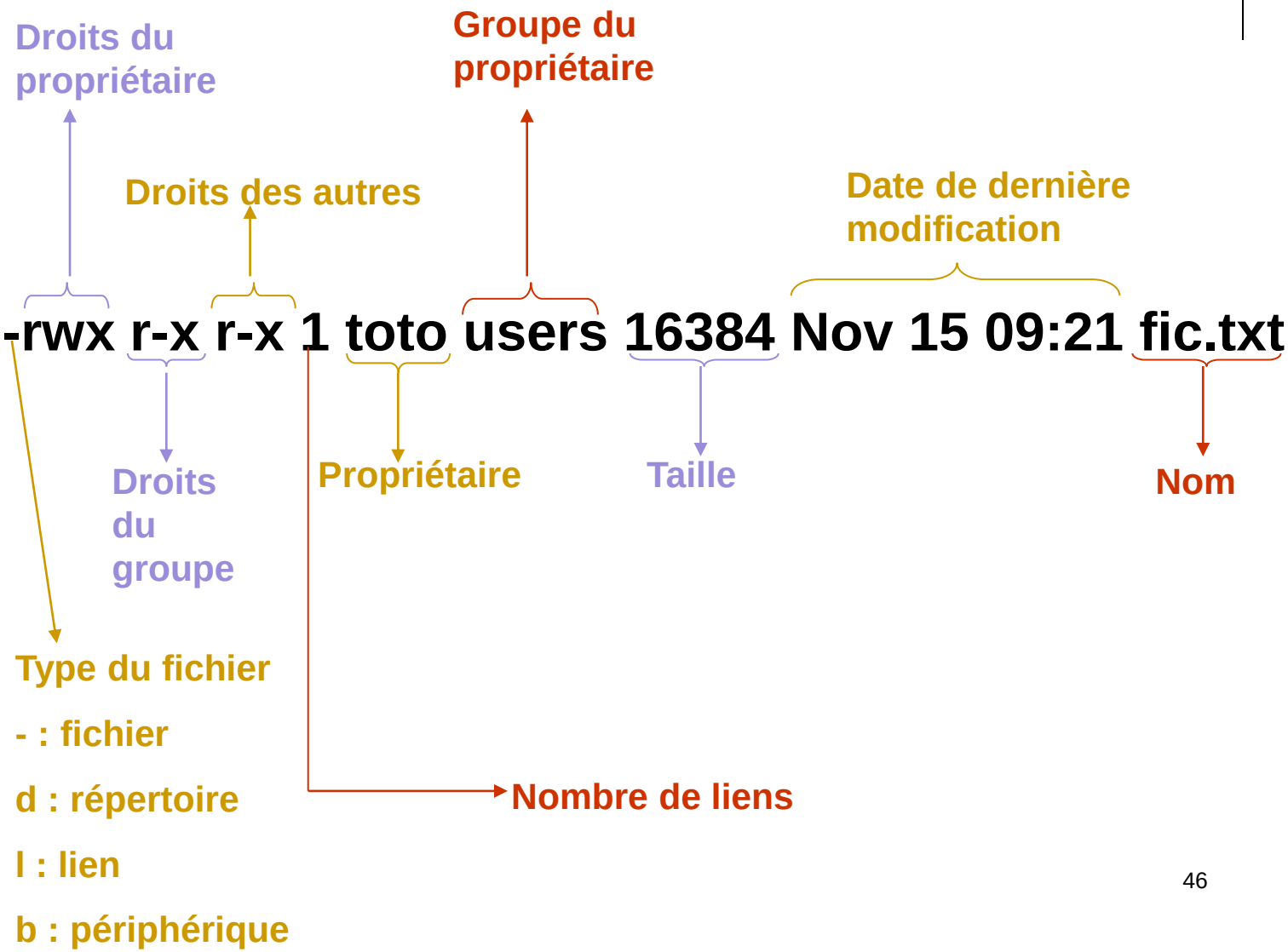


- Un fichier (sens large) possède un ensemble d’attributs :
  - L’UID de son propriétaire
  - Le GID de son groupe propriétaire
  - Les droits pour les différents catégories d’utilisateurs

Droits d’accès

- Trois types de droits sur un fichier Unix :
  - La lecture (r)
  - L’écriture (w)
  - L’exécution (x)
  - r, w, x appelés aussi flags ou drapeaux
- Sur un fichier donné, ces 3 flags peuvent être définis
  - Pour son propriétaire (user)
  - Pour le groupe auquel appartient le propriétaire (group)
  - Pour tous les autres (others)
- Seuls root et son propriétaire peuvent changer ses permissions d’accès et son appartenance

Droits d'accès : ls -l



Droits d'accès : les commandes

- **chown** changer le propriétaire
- **chgrp** changer le groupe propriétaire
- **chmod** changer les droits d'accès
- Portée :
  - a : tous les utilisateurs
  - u : le propriétaire du fichier
  - g : tous les membres du groupe
  - o : tous les autres utilisateurs
- Opérande:
  - = : octroie expressément le droit d'accès
  - + : ajoute le droit d'accès
  - - : retire le droit d'accès
- Octal
  - r=4, w=2, x=1

Droits d'accès : exemples

- Changement de propriétaire
  - *chown user1 fic1* change le propriétaire de fic1 en user1
- Changement de groupe
  - *chgrp grp1 fic1* change le groupe de fic1 en user1
- Changement des droits d'accès :
  - chmod u+x fic2 ajoute au propriétaire le droit d'exécution au fichier fic2
  - chmod o+rwx fic2
  - chmod 541 fic3 donne exactement les droits :
    5 : 4+1 (lecture + exécution) au propriétaire
    4 : lecture pour le groupe
    1 : exécution pour les autres

Quelques fichiers particuliers

- /etc/passwd : informations relatives aux utilisateurs (accessible en lecture à tous)
- /etc/group : liste des groupes et utilisateurs
- /etc/shadow : mot de passe hashé – correspondance avec /etc/passwd
- /etc/services : liste des services définis sur la machine
- /dev/null : fichier poubelle

Quelques fichiers particuliers

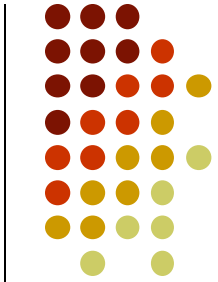
- /etc/fstab : liste des systèmes de fichiers qui peuvent être montés
- /etc/hosts : fichier de correspondance @IP et nom
- /var/log/messages : fichier de messages
- /etc/sysconfig/network : interfaces réseau

LE SHELL UNIX
Langage de commande

Le shell

- Il permet à l'utilisateur de dialoguer avec le système (langage de commandes)
- Programme exécuté lorsqu'un utilisateur se connecte
- C'est aussi un langage de programmation interprété puissant (shell script)
- Terminal

Le shell



- Différentes familles de shell:
  - sh : Bourne Shell (premier, shell std unix)
  - ksh : Korn Shell
  - csh : C Shell
  - tcsh : extension de C Shell
  - bash : GNU: Bourne advanced Shell

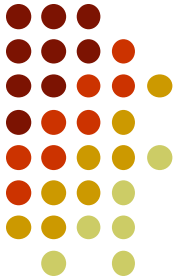
Le shell

- Syntaxe des commandes :
  - **nom_commande** *option argument*
  - *Options :* précédées par un – ou - - (Linux)
  - Argument : par exemple, nom d'un fichier
  - Possibilités d'avoir plusieurs arguments ou aucun
  - Attention : majuscules ≠ minuscules
  - *exemples*

Le shell : les aides

- man [option] <commande>
  - Formater et afficher les pages de manuel en ligne
- <commande> --help
  - Aide intégrée
- whatis <commande>
  - Rechercher des mots entiers dans la base de données whatis
- apropos <commande>
  - Rechercher une chaîne de caractère dans la base de données whatis

Le shell : le contenu des fichiers

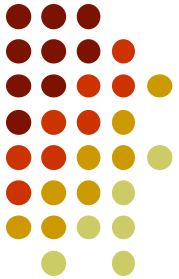


- **cat** afficher le contenu d'un fichier
- **more** afficher (par page) un fichier
- **less** idem à more mais pour les gros fichiers
- **head** afficher le début d'un fichier
- **tail** afficher la fin du fichier
- **ls** lister les attributs
- **ln** créer un lien symbolique
- **touch** créer un fichier vide
- **file** permet de lire l'entête du fichier
- **cmp/diff** comparer les fichiers

Le shell : manipuler les fichiers

- **rm** supprimer
- **cp** copie un fichier
- **mv** déplacer un fichier
- **wc** compter les lignes /mots /caractères
- **tar** gérer les archives

Le shell : les répertoires

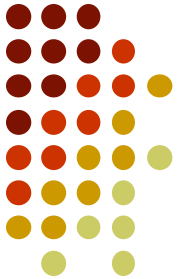


- **cd** changer de répertoire
- **pwd** le répertoire courant de travail
- **mkdir** créer un nouveau répertoire
- **rmdir** supprimer un répertoire
- **cp** recopier un répertoire
- **mv** déplacer un répertoire
- **ls** lister les attributs

Le shell : Les notations « . », « .. », « ~ »

- Permet de ne pas avoir à écrire à chaque fois les chemins en entier
- La notation « . »
  - Désigne le répertoire de travail
  - Utilisé pour exécuter un fichier dans le répertoire courant
    - ./exo1
  - Utilisé avec la copie :
    - cp /tmp/essai .
- La notation « .. »
  - Représente le répertoire père
  - Utilisé avec cd
    - cd .. Ou cd ../..
- La notation « ~ »
  - Représente le répertoire de login

Le shell : imprimer



- **pr** mise en forme
- **lpr** envoie à l'imprimante
- **lprm** suppression d'une tâche d'impression
- **lpq** interroge la file d'impression
- **echo** affiche les arguments

Commande grep

- **grep** recherche d'une expression dans un fichier, dans une arborescence
  - Recherche le mot bonjour dans le fichier exemple

    → grep bonjour exemple
  - Recherche le mot printf dans les fichiers .h
    - → grep printf /usr/include/*.h

Commande find

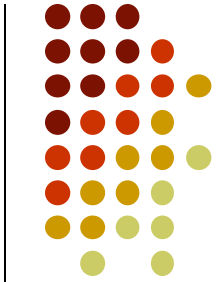
find : recherche d'informations dans l'arborescence

- Recherche de tous les fichiers *.h dans l’arborescence /usr/include
  - find /usr/include *.h
- Recherche les fichiers commençant par hosts sur le repertoire /etc
  - find /etc –name hosts*
- Recherche de fichiers avec une taille supérieure à 2M (racine)
  - find / -type f -size +2M
- Recherche des fichiers sur votre répertoire courant qui ont été modifiés depuis moins de 10 minutes
  - find $HOME –mmin -10

Le shell : autres commandes

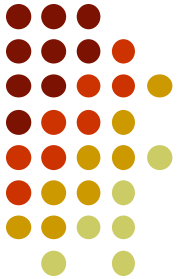
- **cut** sélectionner des colonnes
- **join** effectuer une jointure
- **uniq** suppression des doublons
- **paste** regroupe les lignes d'un fichier
- **sort** trie
- **awk** examine et traite des motifs
- **sed** traitement particulier des textes
- **tr** convertir ou éliminer des caractères

Le shell : autres commandes



- **date** donne la date (selon format…)
- **hostname** renvoie le nom de la machine
- **df** informations sur la capacité totale
- **du** taille occupée
- **alias** permet de remplacer une ou des commandes
- **whereis** localise l'exécutable
- **mount/umount** monte/démonte un système de fichiers

Le shell : le réseau



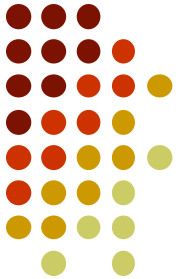
- **ftp** transfert de fichiers
- **ping** teste d'une interface
- **ifconfig** configuration d'une interface réseau
- **traceroute** visualiser le trajet d’un packet IP
- **netstat** statistique sur l'interface réseau
- **host** interrogation du DNS
- **ssh** connexion cryptée à distance

Quelques notions sur les commandes

- Valeur de code retour
  - Chaque commande transmet au programme appelant un code, appelée valeur de retour qui stipule la manière dont son exécution s'est déroulée.
  - La valeur de retour est toujours 0 si la commande s’est déroulée correctement
  - Variable système $? contient la valeur de retour de la précédente commande
- Processus lié à une commande
  - Pour chaque commande effectuée, un processus est créé

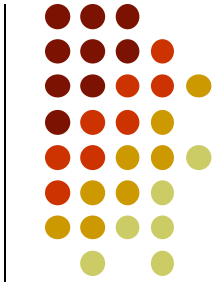
L'historique des commandes (bash shell)

- la variable HISTSIZE définit le nombre de commande à mémoriser (dans ~/.bash_history)
- les flèches haut,bas permettent le déplacement dans l'historique des commandes
- les flèches ← et → le déplacement sur la ligne de commande pour la modifier
- !! rappelle la dernière commande
- !-n rappelle la –n ième commande
- ^A ^E envoie au début/fin de la ligne
- ESC-D efface depuis le curseur jusqu'à la fin du mot



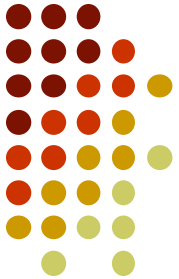
LE SHELL UNIX
Commandes composées

Commandes composées



- Séparateur : le point-virgule « ; »
  - Le shell exécute la seconde commande une fois la première terminée
- L'opérateur &
  - Le shell exécute la commande en arrière-plan (background)
- L'opérateur && (ET logique)
  - exécute la deuxième commande ssi la première renvoie un code = 0
- L'opérateur || (OU logique)
  - exécute la deuxième commande ssi la première renvoie un code <> 0
- Parenthèse : (list)
  - List est exécuté dans un sous-shell
  - Code retour est celui de list

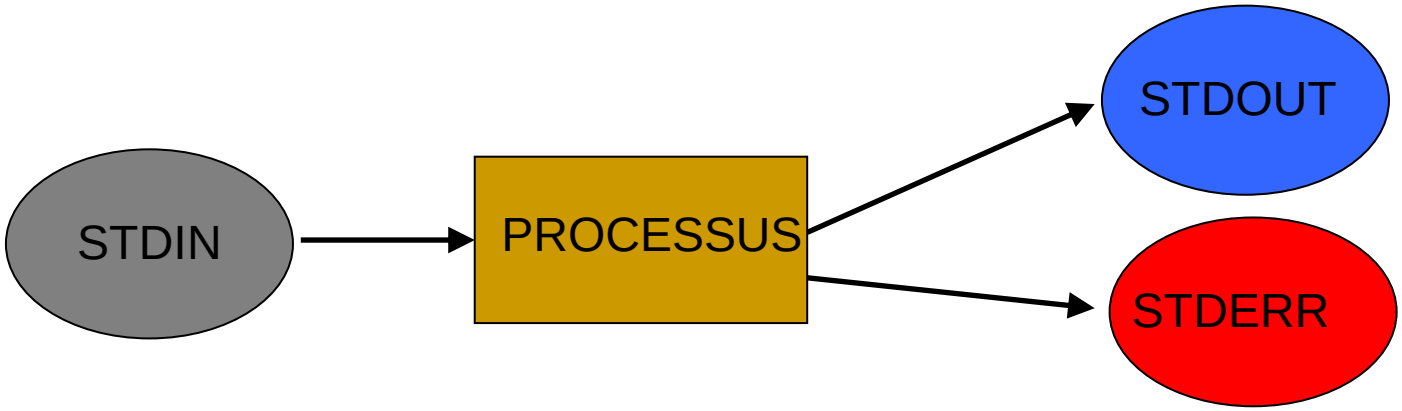
Substitution de commandes



- La substitution de commande permet de récupérer l'exécution d'une commande.
- Deux formes sont possibles :
  - $(command)
  - `command` (antiquote)
- Exemple
  - echo "`whoami`, nous sommes le `date`"
  - echo "$(whoami), nous sommes le $(date)"

Redirections d'entrées/sorties

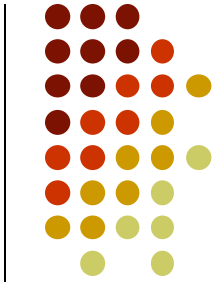
- Quand une commande s'exécute, un processus est créé
- Avant de lancer le processus, le shell lui attribue trois fichiers:
  - Une entrée standard (par défaut le clavier) appelée STDIN <
  - Une sortie standard (par défaut l'écran) appelée STDOUT >
  - Une sortie d'erreur standard (par défaut l'écran) appelée STDERR 2



Redirections d’entrées/sorties

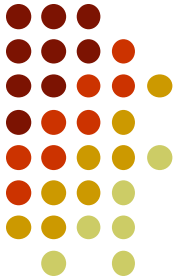
- Ces trois directions peuvent être liées à d’autres fichiers que ceux représentant l’écran ou le clavier grâce aux commandes de redirection
  - <fich1 l'entrée est le fichier fich1
  - >fich2 la sortie standard est le fichier fich2
  - >>fich3 la sortie standard est concaténée au fichier fich3
  - 2>fich5 La sortie d'erreur est dirigée vers le fichier fich5
  - 2>>fich7 La sortie d'erreur est concaténée au fichier fich7

Exemple de redirections



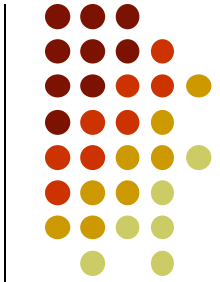
- Exemple
  - $ who > tmp/liste
  - La sortie de la commande who est redirigée vers le fichier /tmp/liste
  - $ ls > fich1 2>fich2
  - $ ls /toto >fich1 2>fich2

Tubes ou pipes : |



- Les **tubes** (en anglais « *pipes* », littéralement *tuyaux*) symbolisé par | redirige la sortie de la première commande sur l'entrée de la seconde
- Le code retour d'un pipe est toujours celui de la dernière commande
- Le shell attend toujours que toutes les commandes du pipeline soient terminées avant de renvoyer le code de retour
- Chaque commande du pipeline est un processus distinct (exécutée dans un sous-shell)

Tubes ou pipes : |



- Exemple :
  - ls | grep toto
    - récupére tous les fichiers toto
  - ls –l | more
    - affiche page par page

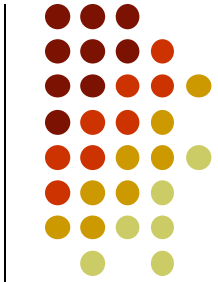
Filtres

Les commandes Unix qui peuvent utiliser leur entrée standard pour lire les données et produire un résultat sur la sortie standard sont appelées des filtres

Commande	**Rôle**
sort	tri des lignes d'un fichier
wc	compte les lignes, mots, signes
grep	recherche d'expressions régulières
cat	copie l'entrée standard sur la sortie standard
tail,head	copie de parties d'un fichier
tr	remplacement de caractères
uniq	élimination des doublons dans un fichier trié
sed	éditeur de mot de données
awk	langage de génération de rapports

LES VARIABLES D'ENVIRONNEMENT

Variables d’environnement



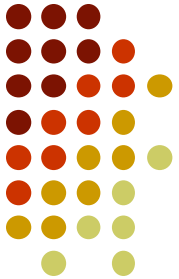
- Une liste de variables qui sont dites d'environnement est gérée par l'interpréteur de commandes
- Ces variables peuvent être lues par d'autres programmes exécutées par l'usager ou utilisées par le « shell » lui-même

Variables d’environnement

- PATH : donne la liste des différents noms de répertoires dans lesquels se trouvent les commandes ou logiciels.
- SHELL : son contenu indique l'interpréteur de commande utilisé.
- HOME : nom du répertoire privé de l'usager.
- MAIL : chemin et fichier dans lequel le courrier de l'usager est placé.
- TERM : type du terminal
- OSTYPE : nom du système d'exploitation
- USER : nom de l’utilisateur de la session
- HOST : nom de l'ordinateur
- PWD : chemin du répertoire de travail courant réel
- PS1 : prompt (par défaut $)
- etc ….

Variables d’environnement

- set : permet de lister la totalité des variables d’environnement
- env :Permet de lancer un programme dans un environnement particulier
- On peut aussi affecter une valeur à ces variables
  - variable=valeur
- La valeur est récupérée en utilisant le nom de la variable précédé par un '$'
  - echo $variable
- export toto="essai "



LE SHELL UNIX
Les scripts

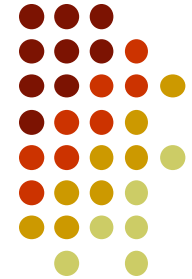
Les scripts

- Fichiers exécutables permettant de lancer successivement plusieurs commandes et de faire de la programmation (automatisation, batch)
- Les scripts shell peuvent comporter:
  - des commandes Unix
  - des programmes exécutables (ou des scripts) avec passage ou non de paramètres
  - des instructions d'assignation de variables
  - des instructions d'E/S
- Commence toujours par #!/bin/bash ou #!/bin/sh

Les scripts : variables prédéfinies

- **$?** C'est la valeur de sortie de la dernière commande. Elle vaut 0 si la commande s'est déroulée sans pb.
- **$0** Cette variable contient le nom du script
- **$1 à $9** Les premiers arguments passés à l'appel du script
- **$#** Le nombre d'arguments passés au script
- **$*** La liste des arguments à partir de $1
- **$$** le n° PID du processus courant
- **$!** le n° PID du processus fils

Instructions de lecture/écriture



- **read** variable1 variable2 variable3 ….
  - Lit une ligne de texte à partir du clavier. S'il y a plusieurs variables, découpe la ligne en mots et affecte à chaque variable.

- **echo** permet l'écriture de chaine de caractères

Travail avec les variables

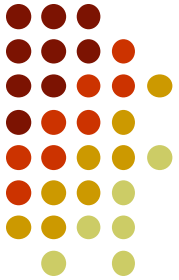
- Variable définie comme des chaines de caractères
- Attention au numérique !
- Les expressions arithmétiques
  - var=`expr $1 + 2`
  - Var=$((var+2))
  - Attention var=a+1 => caractère

Structures de contrôle : if et conditions

- **If** *cond1* **then** *list* [**elif** *cond2* **then** *list*] … [**else** *list*] **fi**
  - *cond1* est exécuté. Si le code de retour = 0, la liste du then est exécutée
  - Sinon, on continue les tests
  - Le code de retour est celui de la dernière commande exécutée
  - Ou 0 si aucune condition n'était vraie
- Deux possibilités :
  - La commande test
  - Les crochets [ ]

  renvoie un code = 0 si le test est vrai ou 1 autrement

Les structures de contrôle : les conditions



Conditions	Explication
-d file	si le fichier est un répertoire
-f file	si le fichier est un fichier ordinaire
-e file	si le fichier existe
-r|w|x file	si le fichier est en lecture|écriture|exécution
-s	si le fichier existe et sa taille > 0
-z string	si longueur de string=0
-n string	si longueur de string > 0
str1 = str2	si str1 égal str2
str1 != str2	si str1 différent de str2
! expr	inverse le résultat de expr (NOT)
expr1 –a expr2	AND (même chose avec OR : option –o)
arg1 OP arg2	OP valant **–eq -ne -gt -ge -lt –le :** test numérique

Exemple de if

```
if  [  –d   toto ]  ;  then
    echo "Répertoire toto existe"
else
    mkdir toto
fi


if [ $#  -ne 2 ]
then
       echo "Vous devez rentrer 2 paramètres "
fi
```

Structures de contrôle : case

- **case** *word* **in** [pattern [| pattern]…) *list* ;;] … **esac**
  - Effectue une sélection si *word* correspond à pattern
  - La liste *list* correspondante est exécutée
  - Le code de retour = 0 si aucun *word* n'a de correspondance
  - Sinon le code de retour est celui de *list*

Exemple :

```
read rep
case  $rep  in
      oui | o | O | y | Y)  echo "La réponse est oui";;
      non | n | N)  break;;
      *) echo "Usage: $0  o[ui] | n[on]"  ;;
esac
```

Structures de contrôle : while

- **while** *list1* **do** *list* **done**
- **until** *list1* **do** *list* **done**
  - **while** est une boucle **O-N**
  - **until** est une boucle **1-N**
  - *List* est exécuté aussi souvent que *list1* renvoie un code=0

Exemple :

```
i=1 ; while  [ $# -ge $i ]  ;  do
    echo "Le paramètre $i vaut ${!i}"
    i = $(( i  +  1 ))
done
```

Structures de contrôle : for

- **for** *name* [**in** *word*;] **do** *list* ; **done**
  - La liste de mots suivant **in** est établie
  - Cela génère une liste d'article (ex: ls *)
  - La variable *name* est positionné pour chaque item de la liste produite
  - *list* est exécuté pour chaque occurrence de *name*
  - Si [**in** *word*] est omis, la commande exécute *list* une fois pour chaque valeur de *name*

Exemple :

for** i **in** t e s t ; **do echo** "i=$i" ; **done

for** i in $* ; do echo $i ; **done

for fichier in *; do

echo « Les fichiers du répertoire sont : $fichier"

done

Structures de contrôle :select

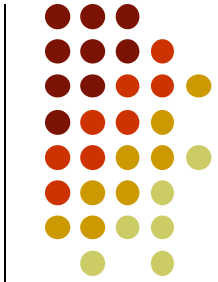
- **select** *name* [**in** *word* ;] **do** *list* ; **done**
  - Permet de choisir un item parmi une liste de choix possibles

Exemple :

```
select  choix  in  "Créer un utilisateur" "Remplacer un utilisateur"
    "Supprimer un utilisateur"  "Quitter"
do
    case  $REPLY  in
      1) add_user;;
      2) mod_user;;
      3) supp_user;;
      4) break;;
      *) echo " Mauvais choix  ";;
    esac
done
```

LES PROCESSUS UNIX

Les processus Unix



- Pour identifier les processus, le système d'exploitation leur attribue un numéro appelé PID (Process IDentification).
- Chaque processus (à l'exception du tout premier) est créé par un autre processus; le processus père est repéré par le PPID (Parent Process IDentification).
- Le tout premier processus porte le numéro 1.

Visualisation des processus

- Par la commande ps : *ps -ef*

```
$ ps -ef
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 Jan25 ?        00:00:05 init
root           2     1  0 Jan25 ?        00:00:00 [keventd]
root           3     1  0 Jan25 ?        00:00:00 [kapmd]
…
root       19342 19340  0 09:13 pts/0    00:00:00 -bash
root       19385 19342  0 09:14 pts/0    00:00:00 vim ex1.awk
rc
rc                                                        [priv]
dp
dp
dp
```

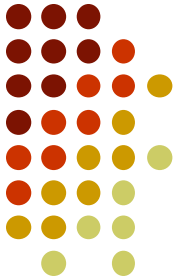
UID	Nom de l'utilisateur qui a lancé le processus.
PID	Numéro du processus
PPID	Numéro du processus père
C	Facteur de priorité; plus la valeur est grande, plus le processus est prioritaire.
STIME	Heure de lancement du processus
TTY	Nom du terminal associé au processus
TIME	Durée du traitement du processus
CMD	Nom du processus

Les processus

- **<ctrl><C>** termine l'exécution du processus en cours
- **<ctrl><Z>** suspend le processus en cours
- **bg** demande l'exécution d'un processus en arrière-plan
- **fg** reprend l'exécution en premier-plan
- **jobs** liste des processus en tâche de fond
- **kill/killall** envoie un signal de terminaison à un processus
- **pidof** récupérer le PID d'une commande en exécution
- **ps/pstree** afficher les processus en cours
- **top** afficher les processus
- **nice** changement de priorité
- **free** afficher les données sur la mémoire

Automatisation des tâches :

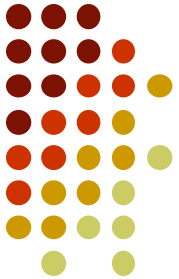
- crontab : pouvoir exécuter un shell à une date et heure données de façon répétitive (sauvegarde, nettoyage, etc)
  - crontab –l : voir les shells en cours
  - crontab –e : ajouter/modifier les shells
- at : identique mais seulement pour une fois



PARTIE 3
Le système WINDOWS

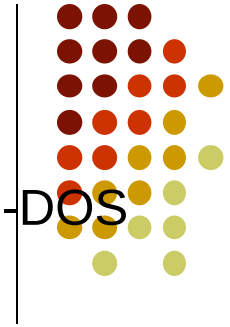
Système Windows : à revoir complètement

- Historique de Windows
- Organisation des fichiers et utilisateurs
- Langage de commande et script
- Les processus



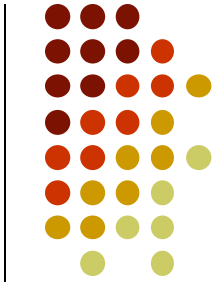
Historique de Windows

Historique Windows



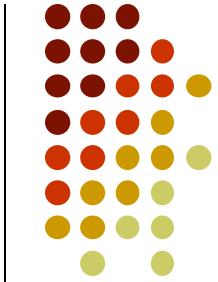
- 1975 : création de Microsoft par Bill Gates et Paul Allen 1980 : MS-DOS 1.0 (Microsoft Disk Operating System)
  - 16 bits
- 1985 : Windows 1.0
- 1987 : Windows 2.0
  - Fat-12
- 1990 : Windows 3.0
  - Fat-16
- 1992 : Windows 3.1 (avec IBM)
- 1993 : Windows 3.11 (fonctionnalités réseau)
- 1995 : Windows 95 (alias Windows 4 et MS-DOS 7)
  - Menu Démarrer
  - 32 bits et Fat-32
- 1998 : Windows 98, 98 SE
- 2000 : Windows Millenium (Multimédia)
- 2001 : Windows XP familiale
- 2006 : Windows Vista familiale
- 2008-2009 : Windows Seven
- 2012 : Windows 8
- 2015 : windows 10

Historique Windows - serveur



- 1993 : Windows NT 3.1
  - Usage professionnel : stabilité;multi-utilisateur
  - NT : New Technology
  - NTFS
- 1994 : Windows NT 3.5
- 1995 : Windows NT 3.51
- 1996 : Windows NT 4.0
  - Stable
  - Réseaux cohérents
  - Editions Workstation, Server, Entreprise, Terminal Server
- 2000 : Windows 2000
  - Editions Professional, Server, Advanced Server, Datacenter
  - NTFS 5
  - Annuaire Active directory
- 2001 : Windows XP PRO
  - Système 64 bits
  - Version professionnelle
- 2003 : Windows server 2003
  - Version Standard Edition, Enterprise Edition, Datacenter Edition
  - Gestion de la sécurité
- 2008 : Windows server 2008
  - Améliorations diverses : AD, Sécurité, stockage, etc
- 2012 : Windows server 2012
  - Serveur plus évolutif, virtualisables, clouds, datacenters

Systèmes d'exploitation Windows



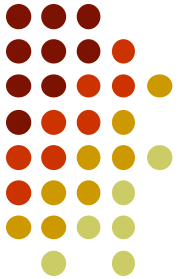
- Windows 3.11
  - multi-tâche préemptif , pas de multi-utilisateurs
- Windows 95
  - **multi-tâche**
  - premier système 32 bits
- Windows 98
  - Internet intégré dans le GUI
  - Plug & Play
- parallèlement Windows NT
  - système d'exploitation réseaux **multi-utilisateur**
- Windows 2000
  - première version unifiée poste de travail et serveur
- Windows 2003 Server et Windows XP
  - premiers systèmes 64 bits

Organisation et utilisateur

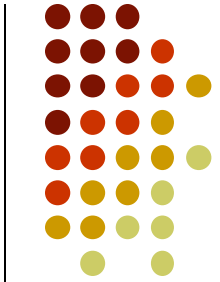
Organisation des fichiers et des répertoires

- Arborescence
- Nom de fichier
  - plus de problème de longueur (Win95, NT)
  - espace possible (utilisation de ˝)
- Toujours système d’extension

Les utilisateurs



- Un compte qui peut tout faire : administrateur
- Notion de groupe
- Droits sur les fichiers et répertoires
  - Pour un utilisateur
  - Pour un groupe
  - Par héritage
- UUID (Unique User Identification)
- SID (Security Identification)



Le langage de commande DOS

DOS

- DOS : Disk Operating System
- Plus de DOS depuis Windows NT
- Langage de commandes
  - Powershell aujourd'hui
- Lancé par « cmd »
- Aide :
  - commande /?
  - help commande
  - help

DOS : répertoire

- CD/chdir : change de répertoire
- MD/mkdir : crée un répertoire
- RD/rmdir : détruit un répertoire
- DEL /s : détruit un répertoire et son contenu
- MOVE : renomme un répertoire

DOS : fichier

- COPY : copie un fichier
- MOVE : déplace un fichier
- REN : renomme un fichier
- DEL/erase: détruit un fichier
- TYPE : affiche le contenu d'un fichier
- MORE : affiche le contenu d'un fichier page/page
- SORT : tri d'un fichier
- FINDSTR : cherche une chaine dans un fichier

DOS : autres commandes

- DATE /t : modifie / affiche la date du système
- TIME /t : modifie / affiche l'heure du système
- DIR : affiche les informations sur les fichiers et les répertoires
- XCOPY : copie des fichiers et des répertoires ainsi que leurs sous-répertoires
- CLS : efface l 'écran
- EXIT : quitte l'écran
- FORMAT: formatage de disques

DOS : commande script

- REM : commentaire
- SET : affectation des variables
  - SET MA_VAR=valeur
  - SET /p MA_VAR=« texte » : entrée à l'utilisateur
- ECHO : affichage en utilisant le %
  - ECHO %MA_VAR%
- @ECHO OFF : permet de n'afficher que le résultat

DOS : structure de contrôle

- IF condition GOTO suite

  ECHO condition fausse

  GOTO fin

  :suite

  ECHO condition vraie

  GOTO fin

  :fin
- NOT : contraire
- EXIST : si fichier existe
- ″Chaine1″ == ″chaine2″

DOS : structure de contrôle

- for %%var in (ensemble) do commande
  - FOR %%f IN (*.txt) DO notepad %%f
- Break ON/OFF : autorise ou non l'utilisateur d'interrompre un fichier de commande avec un Ctrl-C
  - (ensemble) : ensemble de fichiers séparés par des espaces

DOS : paramètre et autres

- %1, %2, etc :liste des paramètres
- Erreur gérables par la variable ERRORLEVEL
- CALL : permet d'appeler d'autres programmes, et à la fin de ce programme, retour au programme appelant
- PAUSE : affiche un message à l'écran, le programme s'arrête jusqu'à l'utilisateur appuie sur une touche
- PATH : chemin de recherche

DOS : information fichier

- Il est possible d'obtenir une multitude d'informations sur un fichier, pour cela, on utilise **%~xp** où x est différent suivant l'information désiré et p le numéro de paramètre
  - f : Affichage du chemin complet (folder)
  - d : Affichage du lecteur (drive)
  - p : Affichage du chemin sans le lecteur et le nom du fichier (path)
  - n : Affichage du nom (name)
  - x : Affichage de l'extension (xtension)
  - s : Affichage du chemin complet (identique à f)
  - a : Affichage des attributs (attributes)
  - t : Affichage de la date et de l'heure de création (time)
  - z : Affichage de la taille du fichier (size)

DOS : redirection

- Redirection d’entrée-sortie : > et >>
- Utilisation aussi de | : pour rediriger sortie sur entrée

Processus et services :

Processus et services : gestionnaire de taches

- Architecture en services qui lance des processus