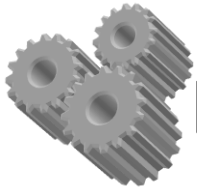
UNIX
Utilisateur

Shell, etc..

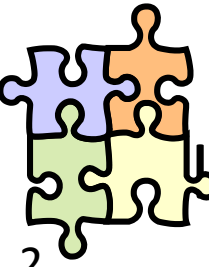
Plan

Place d'Unix dans les OS

Organisation d'Unix



Les commandes d'Unix



La programmation de scripts

Place d’Unix dans les OS

- Le rôle d’un système d’exploitation
- Les types de systèmes d’exploitations
- Les différents versions d’Unix
- Le cas Linux

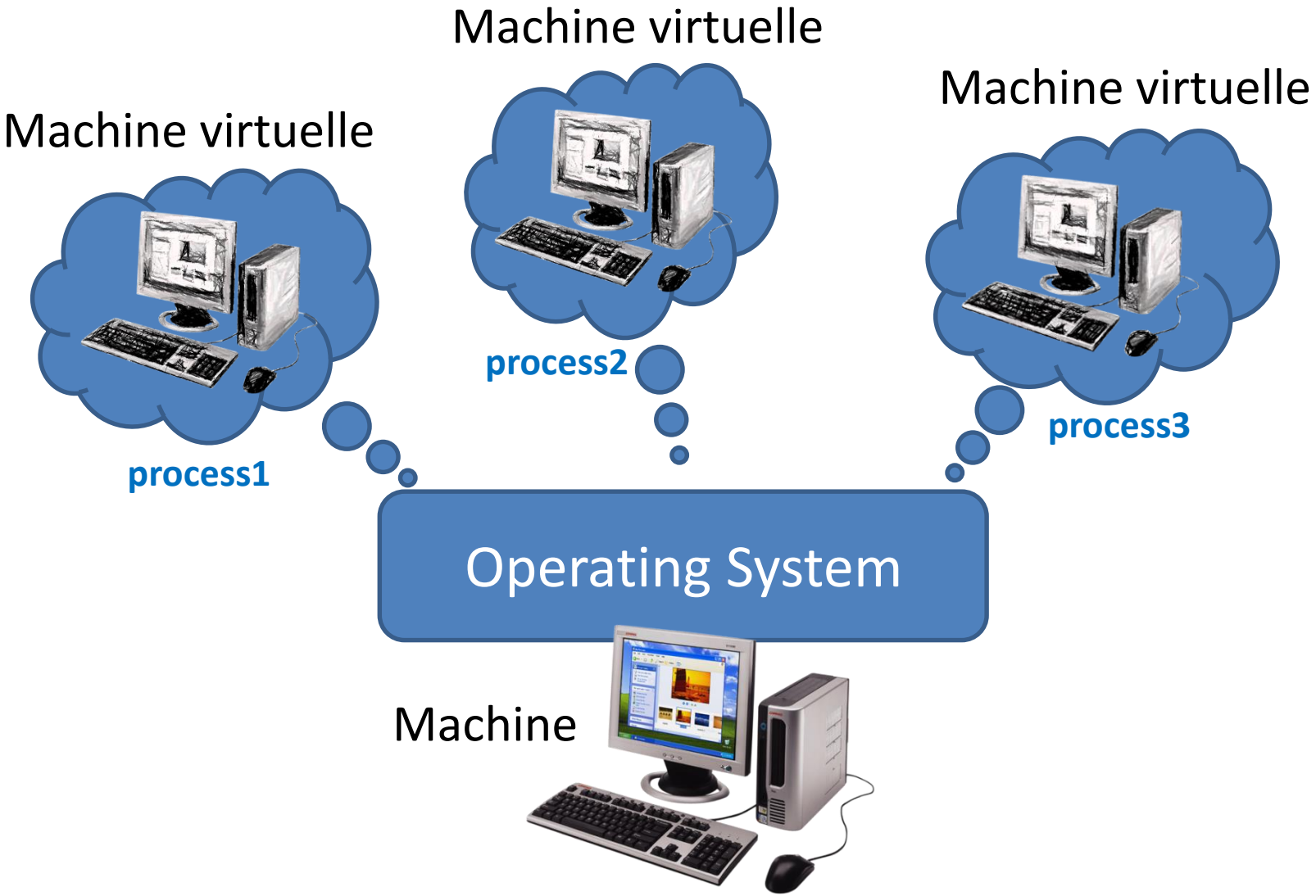
Pourquoi un OS ?

Ouverture (SDK)
Portabilité

Firmware

OS

Fonctionnement de l'OS



Les Services de l'OS

Environnement pour l'Utilisateur

- TTY shell
- GUI shell

Environnement pour les Développeurs

- API système en C (LIBC, POSIX, Win32 …)
- Framework objet (JAVA, .NET …)

Services

Virtualisation des I/O

- Persistance → File System, distribution (NFS …
- IPC → pipes, shm, sockets …
- Périphériques → Série, Parallèle, USB …

Pilotes

Mémoire

- MMU → processus → protection

Exécution

- Ordonnanceur → thread → objets de synchronisation

Noyau

Le rôle d'un OS

- Ordonnancement des tâches
- Gestion de la mémoire
- Gestion du stockage
- Gestion des entrées/sorties

L’ordonnanceur

- Dans les systèmes multi-tâches, l’utilisation de la CPU (1 ou plusieurs processeurs) est sous le contrôle de l’ordonnanceur (« scheduler »)
- Les tâches sont placées dans une file d’attente
- Il peut exister plusieurs files d’attentes en fonction de la priorité des tâches

Recyclage des tâches

- Lorsqu'une tâche utilise le processeur, elle libérera le processeur :
  - si elle a terminé son travail (le programme est terminé)
  - si elle est en attente d'une autre ressource (donnée sur le disque, message réseau, …) dans ce cas, l'ordonnanceur la replace en fin de la file d'attente (recyclage)
  - si elle a dépassé son quota de temps d'utilisation du processeur (« time slice »)
  - si une tâche plus prioritaire apparaît (préemption)
- Tous les OS n'implantent pas forcément les mécanismes de quotas de temps ou de préemption

Gestion de la mémoire

- Lorsqu'une application s'exécute, elle a besoin de mémoire RAM pour stocker son code et les données qu'elle traite (traitement de texte, jeu 3D …)
- Le gestionnaire de mémoire est responsable de l'attribution des blocs de RAM (pages) aux applications. Il garantit qu'il n'y a pas de conflit d'accès (protection mémoire)
- Lorsque les applications demandent plus de mémoire qu'il n'y a de RAM disponible, le gestionnaire utilise une zone du Disque Dur comme zone tampon (swap), on parle de « mémoire virtuelle ».
- Lorsque qu'un programme se termine, le système récupère sa mémoire pour la « recycler ».

Gestion du stockage

- Le gestionnaire de disques organise le stockage des données en systèmes de fichiers
- Un système de fichiers est en première approximation, une arborescence de répertoires pouvant contenir des fichiers ou des répertoires (sous-répertoires)
- Un système de fichiers peut correspondre à une unité physique (un disque dur) ou à une partie d'un disque (partition) ou de plus en plus s'abstraire des limites physiques (volumes logiques)

→ Les différents processus se partagent les disques (différence UNIX / VmWare)

Gestion des entrées/sorties

- Clavier, souris,
- Écran
- Carte son
- Ports séries, parallèles, USB, Firewire, …
- Réseau
- …

Les types d'OS

- Universels
  - Unix
  - Windows
- Dédiés aux serveurs (Netware)
- Embarqués
  - Symbian
  - Palm OS
- Temps réels
  - Windows CE
  - VxWorks
  - QNX
  - FreeRTOS

Caractéristiques d'Unix

- Simple
- Modulaire
- Fonctionne sur la plupart des architectures matérielles
  - Intel
  - Motorola/IBM PowerPC
  - Sun Sparc
  - ARM (XScale, MX …)
  - Etc.
- Source disponible
- Économique

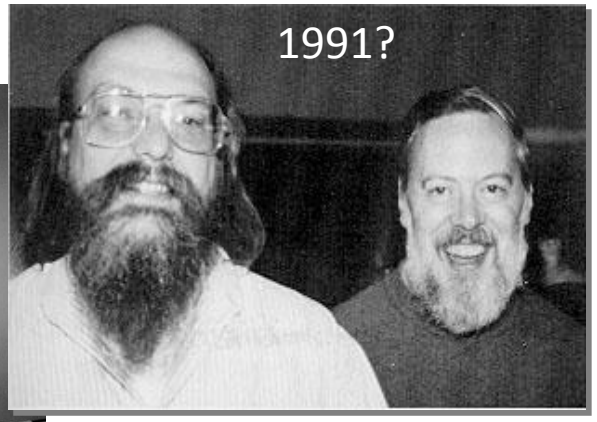
❑ Vrai

Une inspiration en 1969

…dès 2009

Cloud

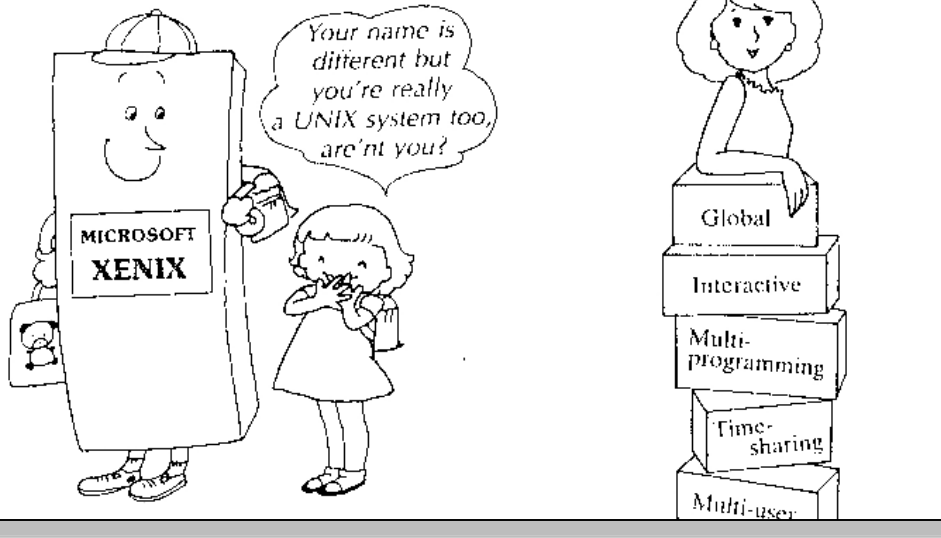
Bell Labs - 1972



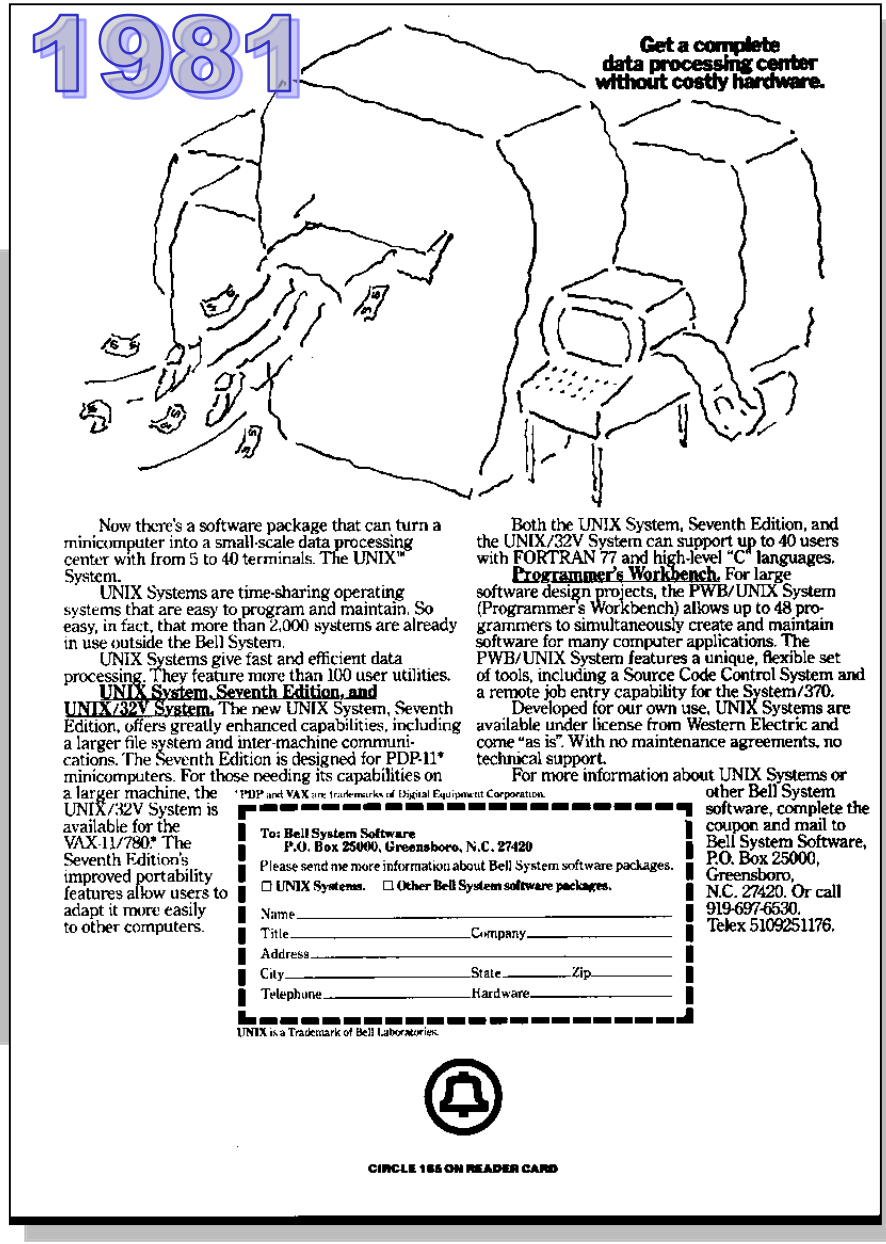
Finlande 1991

Les « UNIX »

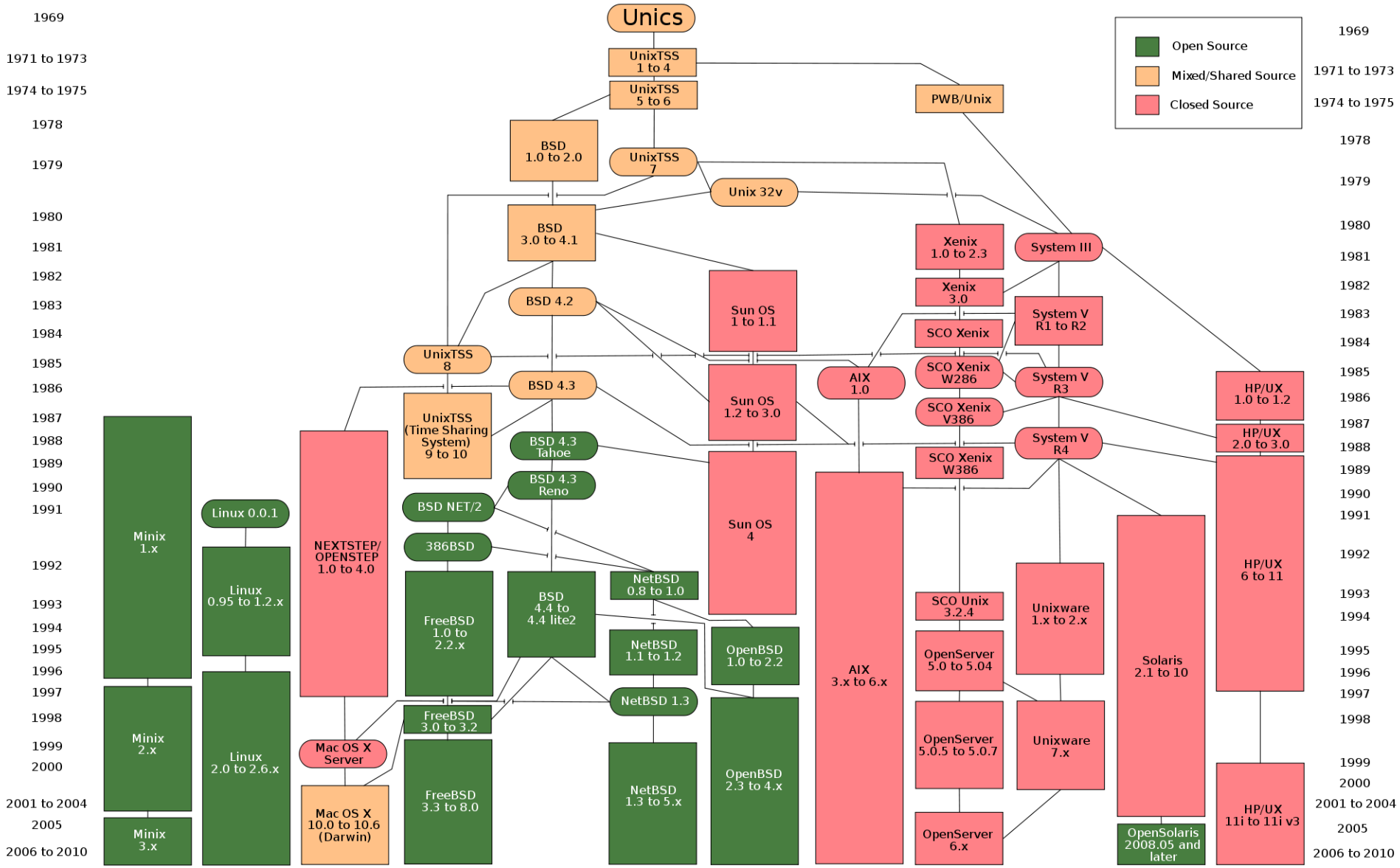
1980



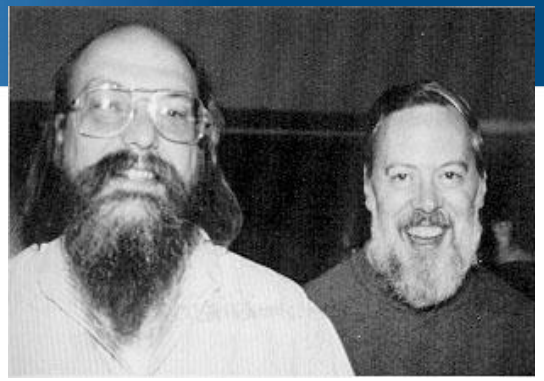
1981



Vous avez dit `fork` ?



Petit historique d'Unix



- 1969 : Ken Thompson & Dennis Ritchie, UNICS sur DEC PDP-7 (4ko RAM), Langage B
- 1972 : V4, Langage C (Kernighan & Richie)
- 1974 : UNIX V6, diffusion dans les universités
- 1978 : 1BSD (Bill Joy) réalisé à Berkeley
- 1979 : V7 : UUCP, Bourne Shell, Compilateur C…
- 1980 : Xenix (Microsoft), Apollo, Onyx…
- 1981 : Silicon Graphics, Unix System III (AT&T/USG)
- 1982 : Sun Microsystems => Sun 1/SunOS 1.0 Ethernet, TCP/IP

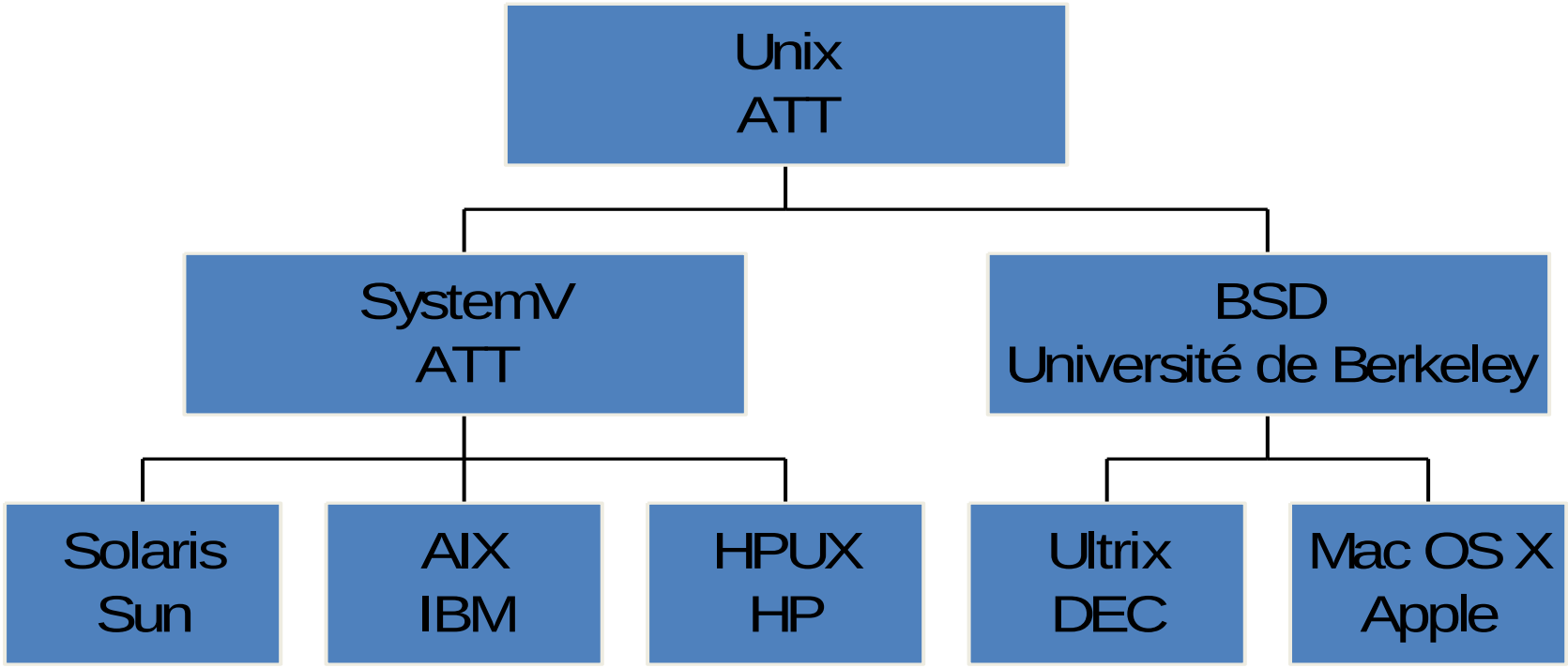
Petit historique d'Unix (2)

- Janvier 1983 : System V (AT&T/USD)
- 1983 : SGI IRIS 1000 (68000/8MHz/4Mo), GL
- 1984 : Débuts de X Window au MIT
- 1985 : CRAY 2 sur System V (UNICOS)
- 1986/7 : X Window v11, System V R3
- 1988 : POSIX.1, 4BSD, OSF, Unix International
- 1989 : UNIX System V R4 (unif. SV, BSD, Xenix)
- 1991 : Debut du developpement de Linux (Linus Torvalds), inspiré de Minix (Tannenbaum)

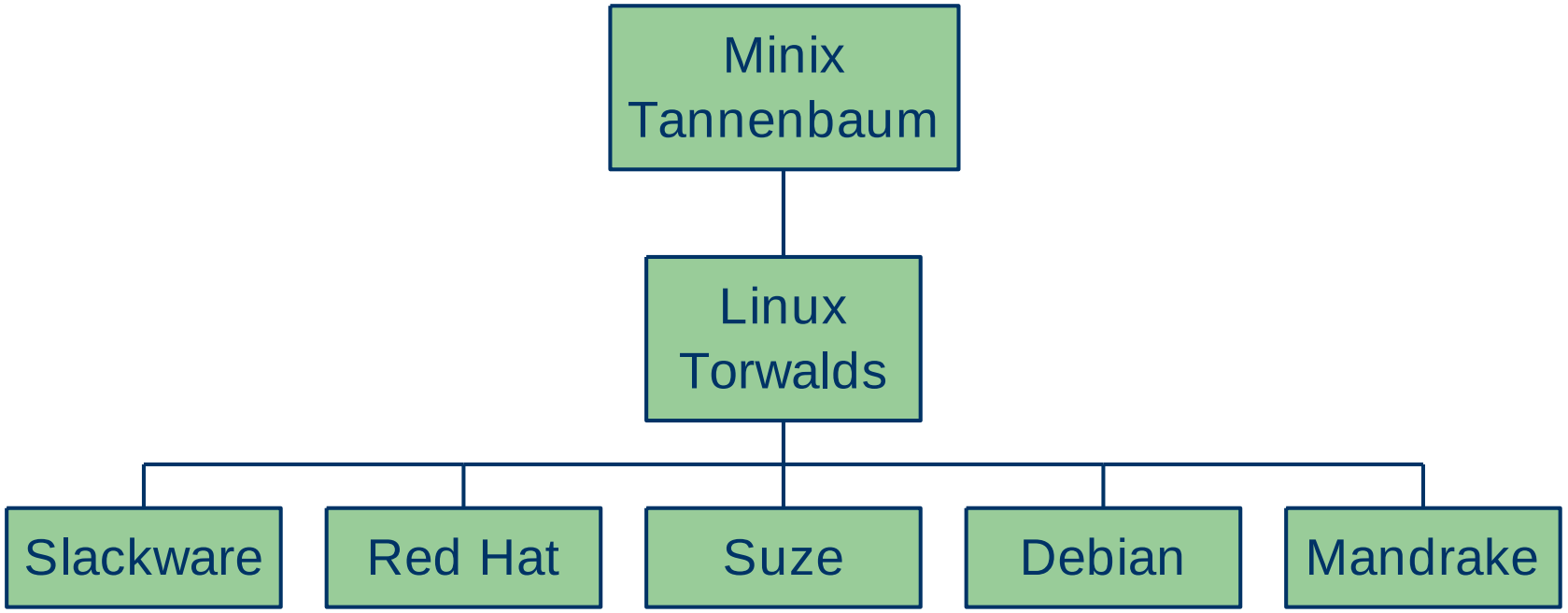
Petit historique d'Unix (3)

- 1993 : 4.4BSD, Novell rachète Unix System Laboratories (AT&T/USL) => SCO & X/Open
- 1994 : X11R6
- 1995 : UNIX 95 Specifications (X/Open)
- 1998 : UNIX 98 (Base, Workstation, Server)
- 1999 : Linux kernel 2.2
- 2001 : Linux kernel 2.4
- 2002 : Single UNIX Spec. v3 standard international ISO/IEC 9945
- UNIX 03, http://www.opengroup.org

Les différentes versions d'UNIX



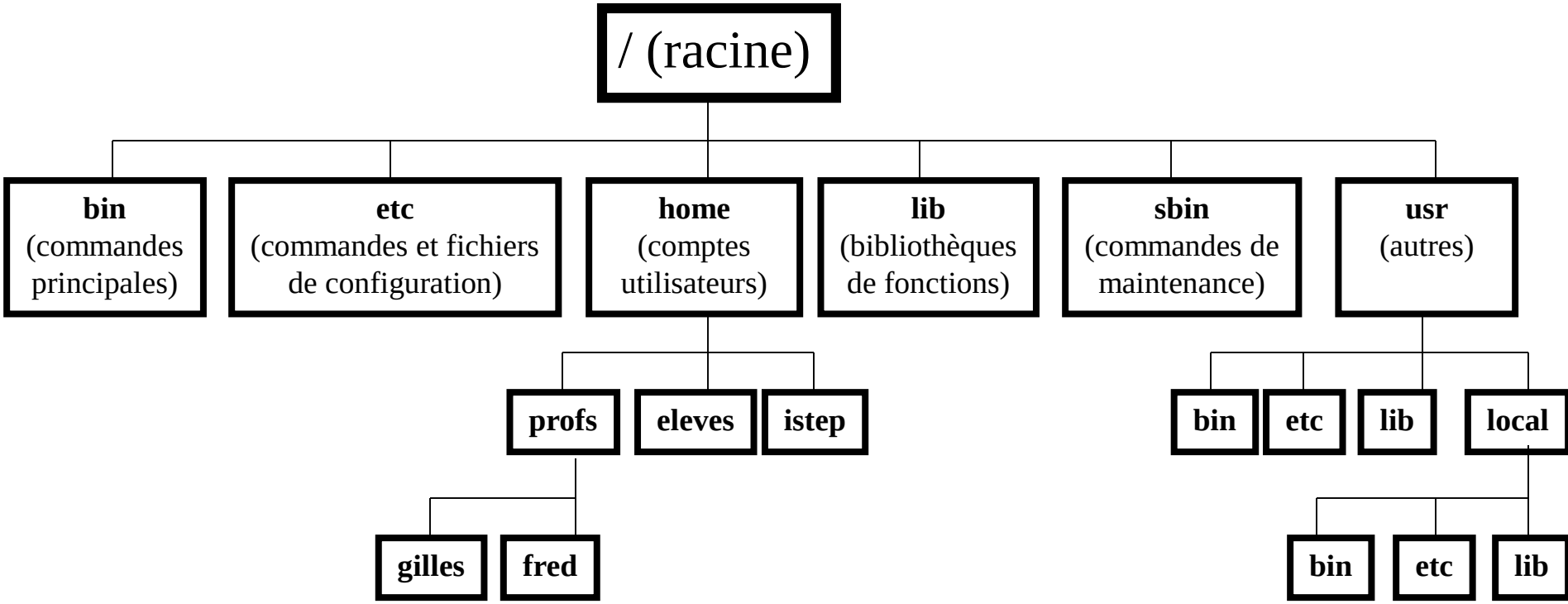
Le cas Linux


Minix
Tannenbaum
Linux
Torwalds
Slackware
Red Hat
Suze
Debian
Mandrake

Unix vu de l'utilisateur

- Le système de fichiers
- Les interpréteurs de commandes
- L'interface graphique

Le système de fichiers



Le chemin

- Le signe « / » sert à la fois à désigner la racine, et de séparateur entre les sous-répertoires
- Le chemin d'accès à un fichier ou répertoire peut être effectué
  - soit de manière absolue en partant de la racine
    - Exemple : /home/profs/gilles
  - soit de manière relative en partant de la position actuelle dans l'arborescence
    - **« . »** désigne le répertoire courant
    - **« .. »** désigne le répertoire père du répertoire courant
    - Exemple : supposons que le répertoire courant soit /home/profs/gilles et que je veuille faire référence au répertoire d'un autre prof : ../christian

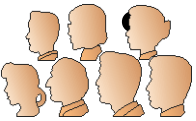
Les interpréteurs de commandes

- Un interpréteur de commandes (« shell ») fournit une interface utilisateur en mode texte
- Le shell attend que l'utilisateur tape une commande et l'exécute dès que l'utilisateur a frappé sur la touche « Entrée »
- Il existe plusieurs shells sous Unix :
  - Le Bourne shell (« sh »)
  - Le C shell (« csh ») dont la syntaxe est proche du langage C
  - Le Korn shell (« ksh »)
  - « bash » …

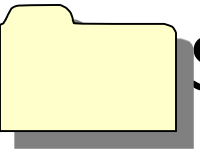
L’interface graphique

- L’interface graphique standard sous Unix est X/Window 11 (MIT)
- Sa particularité est d’être client/serveur :
  - Prise en compte de terminaux graphiques
  - Communication réseau
- Ne gère que le contenu des fenêtres
- Les fenêtres sont gérées par des clients X11 appelés gestionnaires de fenêtres :
  - CDE/Motif
  - Gnome, KDE, …

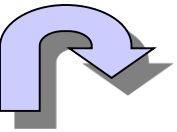
Les commandes d'Unix



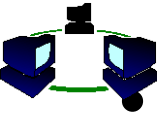
Les utilisateurs



Système de fichiers

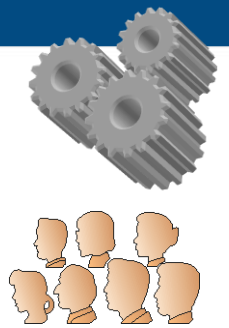


Les processus



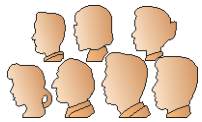
Le réseau

Le login



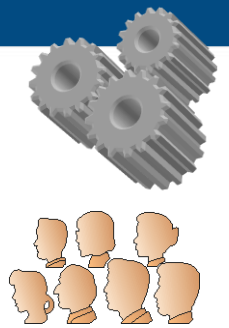
- Unix est un système multi-utilisateurs
- Pour utiliser Unix, il faut d'abord se loger sur son compte
- Le système demande un « login name » et un mot de passe
- Le « login name » doit être en minuscules
- Suivant le type d'Unix, le mot de passe doit contenir un certain nombre de caractères dont des caractères numériques ou spéciaux

L’identification des utilisateurs



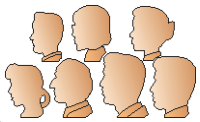
- Un fichier, un répertoire appartient à un utilisateur
- Chaque utilisateur est identifié par un numéro, le User Identifier (UID)
- Le fichier **/etc/passwd** établit la correspondance entre entre l’UID, le nom de login, le répertoire d’accueil et le shell d’un utilisateur
- Les utilisateurs appartiennent à un ou plusieurs groupes

Exemple de fichier passwd



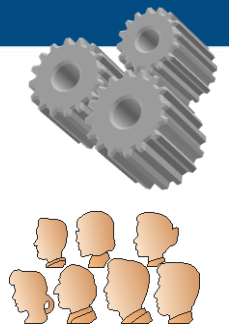
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:100:sync:/bin:/bin/sync
games:x:5:100:games:/usr/games:/bin/sh
man:x:6:100:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh

Le login



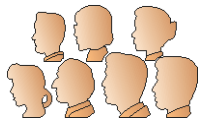
- Une fois l'utilisateur authentifié, en fonction du shell défini pour cet utilisateur dans le fichier **/etc/passwd**, le système exécute un script de démarrage par défaut (celui stocké dans **/etc**), puis celui spécifique à l'utilisateur (celui stocké dans le répertoire d'accueil de l'utilisateur)
- L'utilisateur a ensuite accès à son interpréteur de commandes et son répertoire courant est son répertoire d'accueil tel qui est indiqué dans le fichier **/etc/passwd**

Exit et logout



- La commande **logout** permet de se déconnecter du système Unix si l'utilisateur se trouve dans l'interpréteur de commandes initial (celui obtenu après le login)
- **exit** permet de sortir d'un interpréteur de commandes. Si celui-ci est le login-shell, alors **exit** aura le même effet que **logout**

Les commandes who et w



- La commande **who** permet de connaître la liste des utilisateurs connectés sur le même système et éventuellement depuis quels systèmes via le réseau
- Exemple : $ who

gilles pts/0 Aug 25 14:53 (nemesis.isep.fr)

- La commande **w** permet en plus de connaître leur activité
- Exemple :$ w

```
 14:55:50 up 59 min,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
gilles   pts/0    nemesis.isep.fr  14:53    0.00s  0.03s  0.02s  w
```

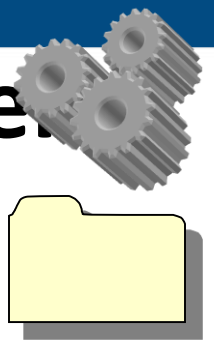
L'historique des commandes

- Le C-shell gère un historique des n dernières commandes lancées par l'utilisateur
- La commande **history** permet d'afficher cet historique
- Pour relancer une commande présente dans l'historique, l'utilisateur peut taper :
  - ! suivi du numéro de la commande
  - ! suivi du début de la commande
  - !! pour relancer la dernière commande
- Il est également possible d'utiliser les flèches haut et bas du clavier pour naviguer dans l'historique

Le manuel

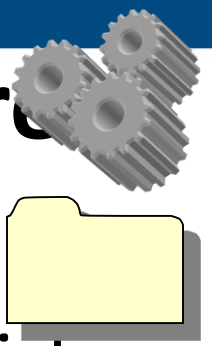
- Unix fournit une aide en ligne appelé le manuel
- Le manuel contient la documentation :
  - Des commandes
  - Des fonctions et appels systèmes de la bibliothèque du langage C (et d'autres langages)
  - Les formats de fichiers
- Le manuel est organisé en sections
- Exemple : **man who**

Navigation dans le système de fichier



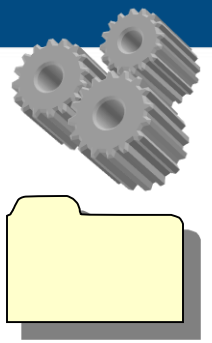
- La commande **pwd** (Print Working Directory) affiche le répertoire courant
- La commande **cd** (Change Directory) permet de changer de répertoire
  - **cd** sans paramètre permet de revenir au répertoire d'accueil
  - **cd** suivi d'un nom de répertoire spécifié par son chemin absolu ou relatif permet de se placer dans ce répertoire
  - Exemples : **cd** */home/profs/gilles*
    cd *../../eleves*
- Le shell complète les noms de fichiers dans une commande si on tape la touche TABULATION

Affichage du contenu d'un répertoire



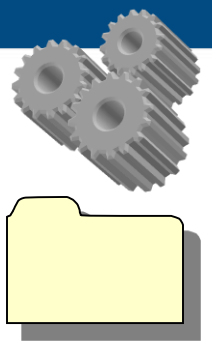
- La commande **ls** (LiSt) permet d'afficher la liste des éléments contenus dans un répertoire (les fichiers, sous répertoires, fichiers spéciaux)
- **ls [-arguments]** [*chemin d'accès*]
- Si le répertoire n'est pas passé en paramètre, c'est le contenu du répertoire courant qui est affiché

Arguments de la commande ls



- **-a** : affiche tous les éléments, y compris les éléments cachés (ceux dont le nom commence par un point)
- **-l** : affiche un listing plus détaillé
- **-g** : affiche le groupe pour lequel les droits de groupe s'appliquent
- Les arguments peuvent être combinés : **ls –alg**
- L'ordre des arguments n'a pas d'importance

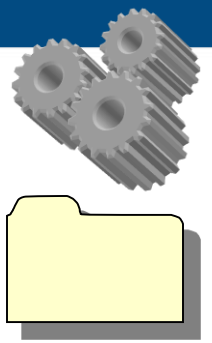
Exemple : ls -alg



```
kounak# ls -alg
total 3
drwxrwxr-x  3 purna    wheel          512 Mar 14  1998 .
drwxrwxrwx 23 gilles   wheel         1024 Mar 13 20:50 ..
-rwxr-xr-x  1 purna    wheel          421 Mar 13 20:57 .cshrc
-rwxr-xr-x  1 purna    wheel          263 Mar 13 20:58 .login
drwxrwxr-x  2 purna    wheel          512 Mar 14  1998 ada
```

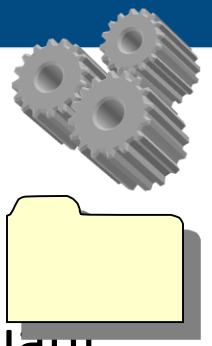
- On peut voir :
  - le . => le répertoire courant
  - le .. => le répertoire père
  - 2 fichiers cachés =>
    - .cshrc de taille 421 octets
    - .login de taille 263 octets
  - 1 répertoire => ada

Types de fichiers



- Le premier caractère de chaque ligne du résultat de la commande **ls –alg** indique le type de fichier :
- **-** => fichier normal
- **d** => répertoire
- **l** => lien symbolique
- **c** => périphérique en mode caractère (ligne série par exemple)
- **b** => périphérique en mode block (disque)
- **p** => tube nommé
- ….

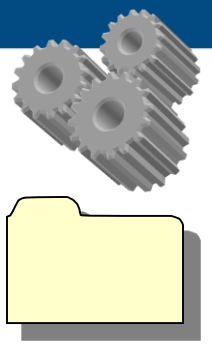
file



- La commande file permet d'obtenir plus d'informations quant au contenu d'un fichier. Exemple :

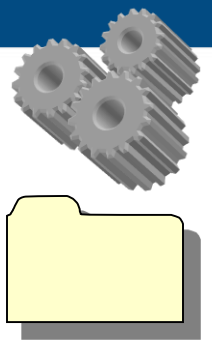
```
base.sql:                ISO-8859 text
bootimg:                 Netboot image, mode 2
carre:                   ELF 32-bit MSB executable, SPARC, version 1
   (SYSV), dynamically linked (uses shared libs), not stripped
dead.letter:             ASCII mail text
Desktop:                 directory
GEF.log:                 empty
Mail:                    directory
MscSI:                   ISO-8859 English text
odp_j2ee.tgz:            gzip compressed data, deflated, last modified: Mon
   Jan 28 03:09:42 2002, os: Unix
recup_passwd_cisco.html: ASCII HTML document text, with CRLF, LF line
   terminators
ssh:                     ELF 32-bit MSB executable, SPARC, version 1
   (SYSV), dynamically linked (uses shared libs), not stripped
system:                  ASCII C program text
```

Les droits



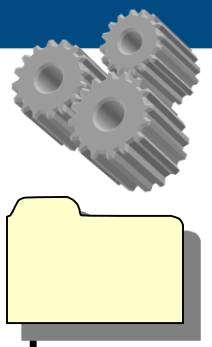
- 3 droits sont gérés par Unix :
  - **r** pour lecture
  - **w** pour écriture : création, modification, suppression
  - **x** pour exécution :
    - dans le cas d'un fichier, **x** indique effectivement que ce fichier est un programme et qu'il peut être exécuté
    - dans le cas d'un répertoire, **x** indique que le répertoire peut être parcouru
- Ces droits sont affectés :
  - au propriétaire du fichier
  - à un groupe
  - aux autres utilisateurs

Propriétaire et groupe



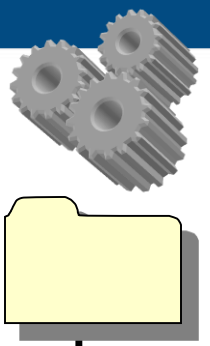
- La troisième colonne de l'affichage du **ls** indique l'UID ou le nom de l'utilisateur propriétaire du fichier ou répertoire.
- La quatrième colonne indique le numéro ou le nom du groupe pour lequel le propriétaire du fichier a accordé des droits

Exemple



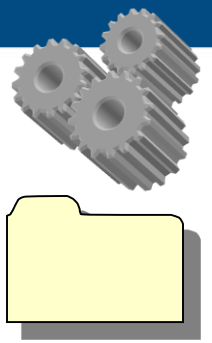
- Le répertoire « ada » de l'exemple précédent appartient à l'utilisateur « purna »
- Les droits du propriétaire sont « **rwx** »
- Les droits du groupe « wheel » sont « **rwx** »
- Les droits des autres utilisateurs sont « **r-x** », ils n'ont donc pas le droit en écriture

Comment changer les droits ?



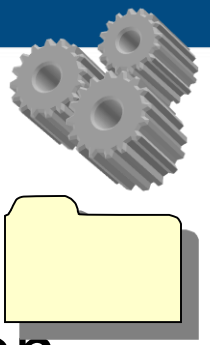
- La commande **chmod** (Change MODe) permet de changer les droits d'accès à un ou plusieurs fichiers
- Les droits d'accès (mode) sont codés symboliquement ou numériquement

Représentation symbolique



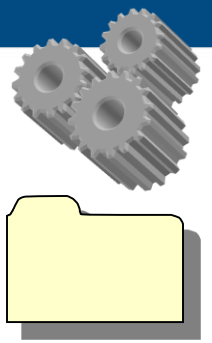
- **chmod [ugoa] [[+-=] [rwx]]** *fichiers*
- La combinaison « **ugoa** » indique la catégorie d'utilisateurs pour laquelle on modifie les droits d'accès
  - **u** : propriétaire
  - **g** : groupe
  - **o** : les autres
  - **a** : tous (valeur par défaut)
- L'opérateur
  - **+** ajoute un accès
  - **-** enlève un accès
  - **=** affecte un accès
- **rwx** correspondent aux droits en lecture, écriture, exécution

Représentation numérique



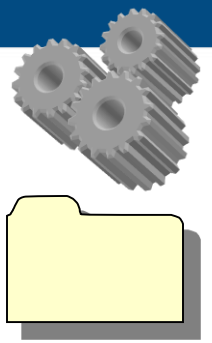
- Cette représentation utilise une combinaison de 3 chiffres en octal.
- Le premier chiffre indique les droits du propriétaire, le second ceux du groupe et le troisième ceux des autres utilisateurs.
- Pour chaque catégorie, on somme les droits correspondant suivant le codage suivant :
  - 4 lecture
  - 2 écriture
  - 1 exécution

Exemples



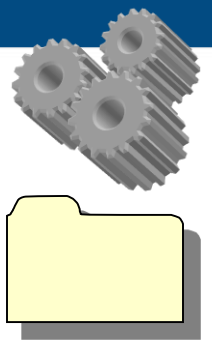
- **chmod o+x** *toto*
  - Ajoute le droit **x** aux autres utilisateurs que le propriétaire et les membres du groupe pour accéder au fichier *toto*
- **chmod o-w** *toto*
  - Enlève le droit **w** aux autres utilisateurs
- **chmod 750** *toto*
  - Donne tous les droits au propriétaire (4+2+1)
  - Les droits en lecture et exécution (4+1) au groupe
  - Aucun droits aux autres (0)

répertoires



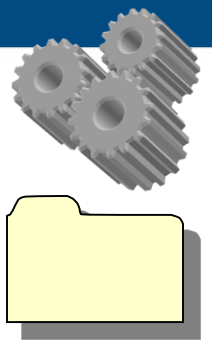
- Création de répertoire => **mkdir** (Make DIRectory)
  - Exemple **mkdir** *profs eleves istep*
- Suppression d'un répertoire vide => **rmdir** (ReMove DIRectory)

fichiers



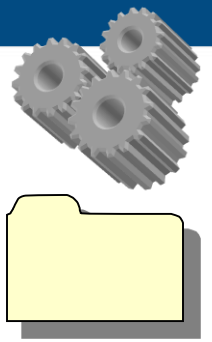
- **cp** pour copier des fichiers
- **rm** pour supprimer des fichiers
- **mv** pour déplacer ou renommer des fichiers

Les caractères de remplacement



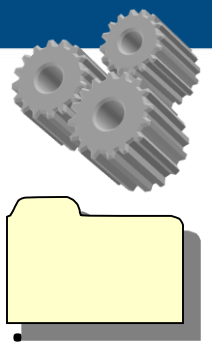
- La désignation de fichiers peut s'effectuer en plaçant des caractères « joker » dans le nom des fichiers
- Le caractère « * » remplace n'importe quelle succession de caractères
  - **cp** **.c prog* => copie tous les fichiers qui se terminent par « .c » dans le répertoire *prog*
  - **rm** *messages** => supprime tous les fichiers dont le nom commence par « messages »
- Le caractère « ? » remplace un caractère
  - **rm** *t?t?* => supprime tous les fichiers dont la longueur du nom est de 4 caractères et dont la première et troisième lettre est «t»

La récursivité



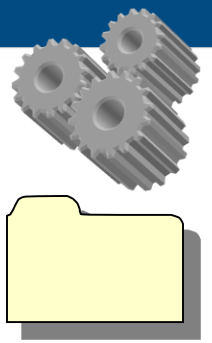
- La plupart des commandes acceptent l'argument **–r** (ou **–R** pour **chmod**) permettant d'indiquer que l'application s'effectuera de manière récursive
- Exemples :
  - **rm –r** *prog* => supprimera le répertoire *prog* et tous ses sous-répertoires
  - **chmod –R 750** * => changera les droits de tous les fichiers du répertoire courant et de tous ses sous-répertoires

Les liens



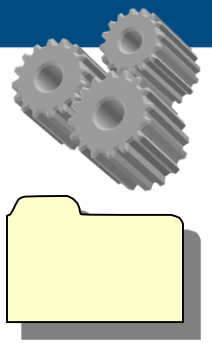
- Plutôt que de copier un fichier d'un répertoire dans un autre répertoire, il est possible de créer un lien sur ce fichier
- Il existe 2 types de liens
  - Les liens matériels
  - Les liens symboliques

Les liens matériels



- Un fichier dans le système de fichiers est identifié par un numéro unique appelé i-node. En fait un fichier est un lien sur cet i-node
- On peut créer autant de liens que l'on veut sur un même i-node à l'aide de la commande **ln** (LiNk)
- Les liens matériels ne peuvent être créés que sur des fichiers (pas des répertoires)
- La seconde colonne affichée par la commande **ls –l** montre le nombre de liens vers un fichier
- Effacer un fichier consiste à décrémenter le compteur de liens. Les données ne sont effacées que lorsque le compteur arrive à 0.

Les liens matériels



- **ln** *cible nom_du lien*
- Exemple : **ln** *toto tata*

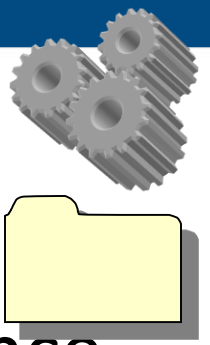
toto i-node =1, compteur = 2
titi i-node = 2, compteur =1
tata i-node =1, compteur = 2
tutu i-node = 3, compteur =1

Fichier (i-node = 1)

Fichier (i-node = 2)

Fichier (i-node = 3)

Les liens symboliques

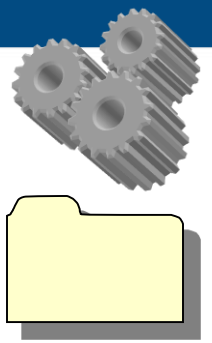


- Les liens symboliques sont créés par référence au nom de la cible (son chemin) au lieu de l'i-node.
- C'est la commande **ln** avec l'argument **–s** qui permet de créer un lien symbolique
- Il est possible de créer un lien symbolique aussi bien vers un répertoire que vers un fichier
- Le fichier peut être situé sur un autre système de fichiers

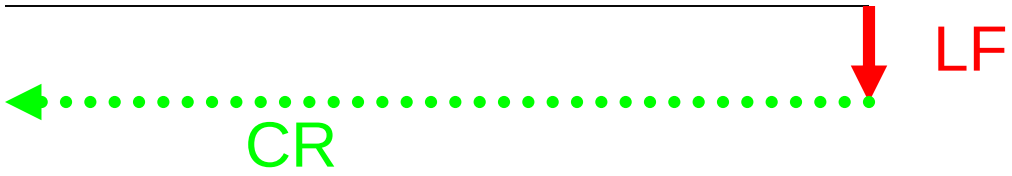
Les liens symboliques

- **ln –s** *cible nom_du lien*
- Exemple : **ln –s** *../purna/ada Ada*

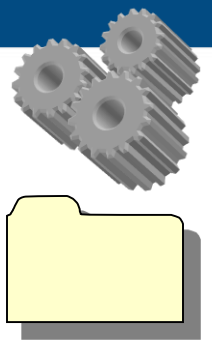
Les fichiers textes



- Les fichiers textes (ASCII pur) contiennent le caractère saut de ligne (« LF = Line Feed ») à chaque fin de ligne
- Lorsque le fichier est affiché par une commande de l'interpréteur, le terminal Unix affiche automatiquement un retour chariot (« CR = Carriage Return») en plus du saut de ligne

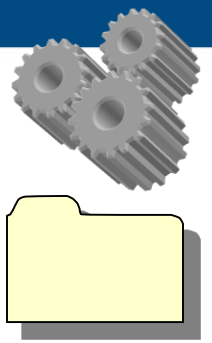


Incompatibilité des formats



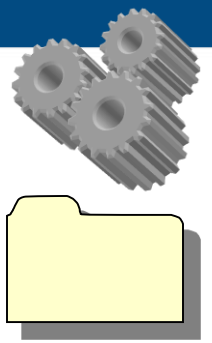
- Or, sous DOS/Windows, le retour chariot se trouve dans le fichier texte avant le saut de ligne
- Conséquences :
  - si un fichier texte Unix est transféré sans conversion vers Windows, il manquera le retour chariot, le texte s'affichera ou s'imprimera en escalier
  - si un fichier texte Windows est transféré sans conversion vers Unix, il contiendra le caractère CR en trop, qui peut poser quelques problèmes à certains programmes (éditeurs, compilateurs, …)

Conversion de formats



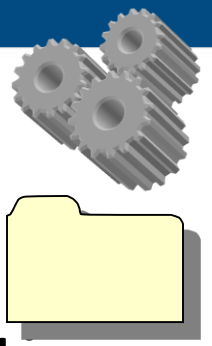
- Les commandes **unix2dos** et **dos2unix** permettent sous Unix de convertir les formats de fichiers
- La commande interne (**ascii**) du programme de transfert de fichiers **ftp** permet d'effectuer la conversion au moment du transfert

impression



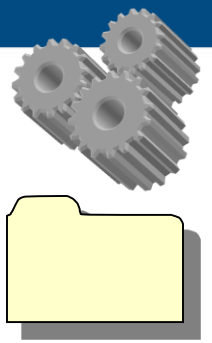
- La commande **lpr** (Line Printer) permet d'imprimer des fichiers imprimables
  - Textes
  - Postscript
  - HPGL
- L'argument **–P** suivi du nom de l'imprimante permet de choisir l'imprimante
- Exemple : **lpr –P** *orsay cours.ps*
- **lpq** permet de connaître la liste des travaux d'impression en cours pour une imprimante
- **lprm** *[id] [utilisateur]* permet de supprimer le travail d'impression spécifié par l'id, ou tous les travaux d'impression d'un utilisateur

Redirection vers un fichier



- Il est possible de rediriger le résultat (la sortie standard) d'une commande vers un fichier au lieu de l'écran à l'aide de l'opérateur « **>** » ou de l'opérateur « **>>** » pour ajouter à la fin d'un fichier existant
- Exemples :
  - **ls –l >** *listing*
  - **unix2dos** *programme.c* **>** *programme_dos.c*

Redirection depuis un fichier

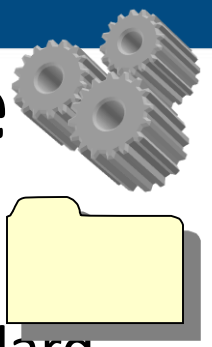


- Pour automatiser l'exécution de certains programmes interactifs, il est possible de stocker dans un fichier les réponses que l'utilisateur aurait à entrer au clavier (entrée standard) et ensuite de lancer le programme en redirigeant son entrée standard sur ce fichier à l'aide de l'opérateur **<**
- Exemple :

Le fichier reponse contient le nom de la personne

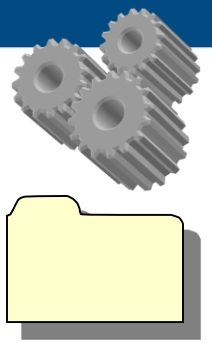
saisie < reponses

Redirection vers un autre programme



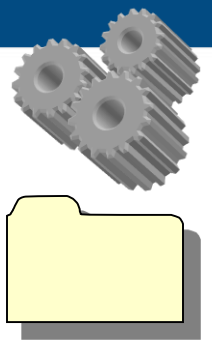
- L'opérateur « **|** » (tube) permet de rediriger la sortie standard d'un programme vers l'entrée standard d'un autre programme
- Cela permet d'enchaîner les traitements sur un flux de données
- L'exécution des programmes reliés par un tube est parallèlisée
- Exemple : **ls –l | unix2dos | lpr –P** *orsay*
  imprime sur une imprimante windows (« orsay ») le résultat de la commande **ls –l** après avoir converti le format

Concaténation de fichiers



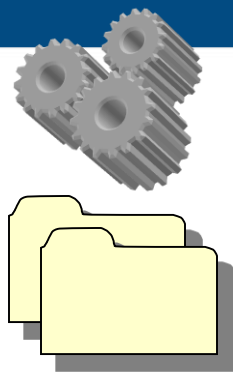
- La commande **cat** (CATenate) permet de concaténer des fichiers textes
- **cat** *toto tutu*
  - affichera à l'écran le contenu du fichier *toto*, puis le contenu du fichier *tutu*
- Il est bien sûr possible de rediriger la sortie standard de **cat** vers un fichier
- **cat** *toto tutu* **>** *titi*
  - *titi* contiendra le texte contenu dans *tutu* suivi du texte contenu dans *titi*

Affichage d'un fichier



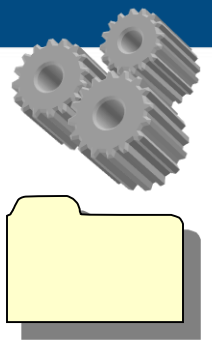
- La commande **more** permet d'afficher un fichier page par page
- À la fin de chaque page, more attend que l'utilisateur frappe
  - la barre d'espacement pour passer à la page suivante,
  - ou entrée pour passer à la ligne suivante
  - Ctrl-B pour revenir à la page précédente
  - q pour quitter **more**

Dump d'un fichier



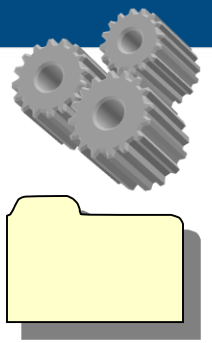
- Quelquefois, il est nécessaire d'afficher le contenu d'un fichier sous forme numérique :
  - Le fichier n'est pas un fichier texte
  - Le fichier contient du texte et des séquences de formatage ou des caractères non affichables
- La commande **od** (Octal Dump) affiche le contenu d'un fichier en octal par défaut,
  - -h en hexadécimal
  - -d en décimal
  - -c sous forme de codes ASCII

exemple



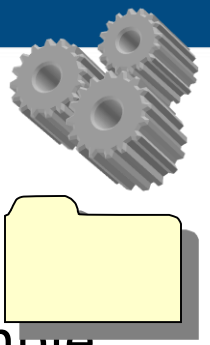
```
gilles@srv-linux:~$ od -hc /etc/hosts
0000000 3231 2e37 2e30 2e30 0931 6f6c 6163 686c
          1   2   7   .   0   .   0   .   1 \t   l   o   c   a   l   h
0000020 736f 0a74 3031 322e 3230 312e 2e30 2031
          o   s   t \n   1   0   .   2   0   2   .   1   0   .   1
0000040 7309 7672 6c2d 6e69 7875 692e 6573 2e70
         \t   s   r   v   -   l   i   n   u   x   .   i   s   e   p   .
0000060 7266 7309 7672 6c2d 6e69 7875 6c20 676f
          f   r \t   s   r   v   -   l   i   n   u   x       l   o   g
0000100 6f68 7473 310a 2e30 3231 2e32 2e33 2032
          h   o   s   t \n   1   0   .   1   2   2   .   3   .   2
0000120 2020 6c09 7561 6164 756e 206d 310a 2e30
                \t   l   a   u   d   a   n   u   m       \n   1   0   .
0000140 3231 2e32 2e33 2034 2020 7209 6d6f 2061
          1   2   2   .   3   .   4               \t   r   o   m   a
0000160 000a
         \n  \0
```

Édition d'un fichier texte



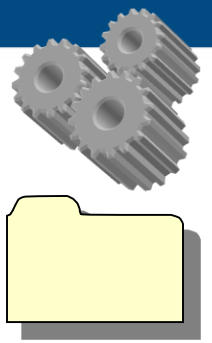
- L'interface graphique propose un grand nombre d'éditeurs de textes
- Pour éditer les sources de vos programmes, je vous conseille **nedit** ou Jext
- D'autre part les environnements intégrés de développement (IDE) fournissent tous un éditeur de sources
- Si vous ne disposez pas de l'environnement graphique, il vous faudra utiliser l'éditeur en mode texte **vi, ou emacs.**

vi (très simplifié)



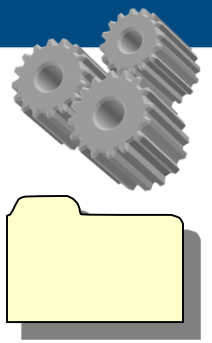
- **vi** ne permet pas d'effectuer des opérations (même un simple déplacement du curseur) et de la saisie de texte en même temps.
- Il existe donc 2 modes : le mode saisie et le mode commande
- On passe en mode saisie :
  - soit en frappant la commande « i » pour insérer du texte à la position du curseur
  - soit la commande « A » pour ajouter du texte à la fin de la ligne
- On revient en mode commande avec la touche d'échappement (Escape)

vi (quelques commandes)



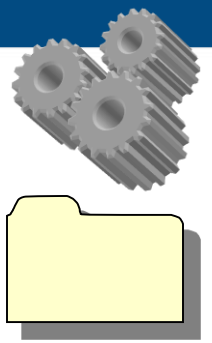
- **Control-F** passe à la page suivante
- **Control-B** passe à la page précédente
- **x** efface le caractère sous le curseur
- **dd** efface la ligne du curseur
- **:w** [nom du fichier] sauvegarde le fichier
- **:q** pour quitter vi
- **:q!** pour quitter sans sauvegarder

nedit



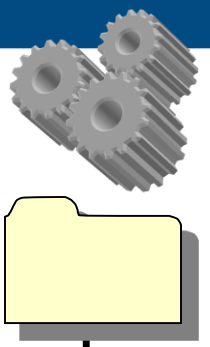
- Nedit est un éditeur graphique avec les avantages suivants :
  - Colorisation syntaxique en fonction du type de contenu (C, Java, SQL, HTML, …)
  - Affichage du numéro de ligne
  - Fenêtres multiples
- Vous utiliserez nedit pour écrire vos programmes en C-shell, C et Java

Recherche d'un texte (grep)



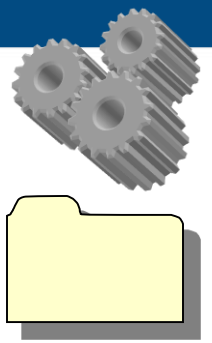
- **grep** (General Regular Expression Parser) effectue la recherche d'une expression dans un ou plusieurs fichiers
- Par défaut la commande affiche les lignes qui contiennent l'expression
- L'option **–n** affiche en plus, le numéro de ligne
- Exemple :
  - **grep –n** *hosts /etc/init.d/**
  - Recherche toutes les occurrences du mot « *hosts* » dans tous les fichiers du répertoire */etc/init.d*

Recherche de fichiers (find)



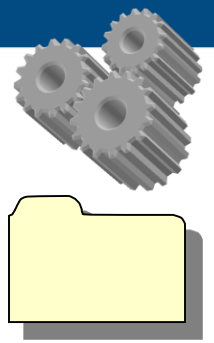
- **find** effectue une recherche de fichiers suivant les conditions spécifiées en paramètres
- **find** *chemins conditions*
- *« chemins »* indique les répertoires dans lesquels effectuer la recherche
- « *conditions* » indique le critère de recherche et les actions à effectuer en cas de découverte

Expressions de recherche



- Le critère de recherche
  - -**name** *nom* : sur le nom de fichier
  - -**user** : utilisateur : sur l'appartenance à un utilisateur
  - -**type** *type* de fichier : f (fichier), d (répertoire), l (lien symbolique) …
  - -**mtime** *n* : sur la date de modification
  - -**size** *n* : sur la taille
- La commande à exécuter en cas d'occurrence
  - -**print** : affiche le chemin absolu du fichier trouvé
  - -**exec** : commande à exécuter

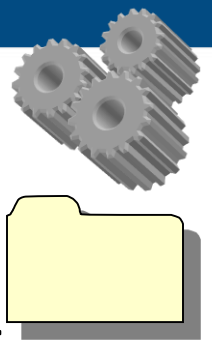
Gestion de l'espace disque



- La commande **df** (Disk Free) affiche l'espace disque disponible sur tous les systèmes de fichiers (-k en Koctest –m en Moctets)
- Exemple : df -m

```
Système de fichiers        1M-blocs Utilisé disponible %  Monté sur
/dev/hda1                  35671    1058     32801    4% /
s_ingres:/home/eleves/a01  15830    9235      5012   65% /home/eleves/a01
s_ingres:/home/eleves/2005 8180     3700      3662   51% /home/eleves/2005
s_ingres:/home/profs       8468     3766      3856   50% /home/profs
s_ingres:/home/stages      8468     1050      6571   14% /home/stages
s_ingres:/home/eleves/commun 7593   1041      5793   16% /home/eleves/commun
```

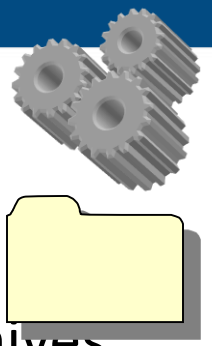
Gestion de l'espace disque



- La commande ***du*** (Disk Usage) affiche l'espace utilisé par chaque répertoire spécifié (sous-répertoires inclus). Le répertoire par défaut est le répertoire courant.
- L'option **–s** (summary) n'affiche que le total par répertoire
- Exemple :

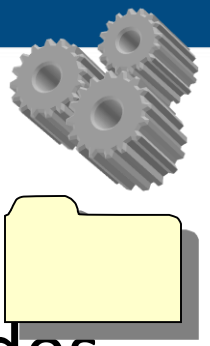
```
kounak% du
6943    ./MATT/webdocs/api
48      ./MATT/webdocs/tools
7025    ./MATT/webdocs
8129    ./MATT
9886    .
```

Archivage (tar)



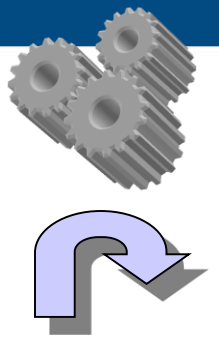
- La commande **tar** (Tape Archive) permet de créer des archives ou d'extraire des fichiers d'archives. Les fichiers d'archives portent l'extension « .tar »
- **tar [options]** *[fichier .tar] [fichiers]*
- Options :
  - **c** (create) pour créer une archive
  - **t** (table) pour afficher la liste des fichiers contenus dans l'archive
  - **x** (eXtract) pour extraire des fichiers
  - **f** (file) doit précéder le nom de l'archive
- Exemple :
  - **tar cvf** *programmes.tar *.c*
  - Crée un archive « *programmes.tar* » contenant tous les fichiers dont le nom se termine par « *.c* »

compression



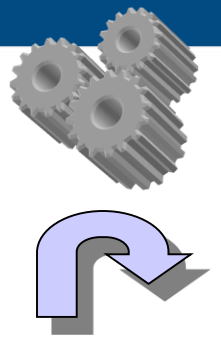
- **compress** et **uncompress** sont les commandes d'Unix pour la compression/décompression de fichiers
- Les fichiers compressés par **compress** portent l'extension « .Z »
- Mais vous pouvez généralement utiliser **gzip**/**gunzip** pour être compatibles avec les utilitaires sous DOS/Windows
- Les fichiers compressés par **gzip** portent l'extension « .gz »

Les processus



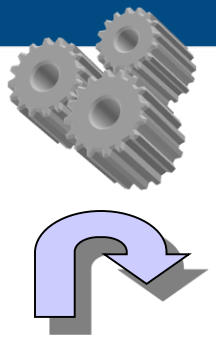
- Un programme est un fichier stocké dans le système de fichiers, qui contient du code exécutable par le système
- Un processus est une instance (une copie en RAM) d'un programme en cours d'exécution
- Il peut y avoir plusieurs processus correspondant à un même programme, par exemple plusieurs shells en cours d'exécution
- Un processus peut lui même lancer d'autres processus

L’identification des processus



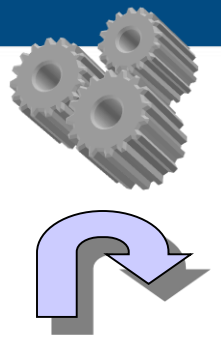
- Chaque processus se voit attribuer un identifiant unique (Process Identifier ou PID) par le système
- Chaque processus est rattaché au processus qui l’a lancé (le processus père)
- Le PID du processus père est appelé le PPID (Parent Process Identifier)
- L’ensemble des processus constitue une arborescence dont la racine est le premier processus lancé par le système

Affichage de la liste des processus



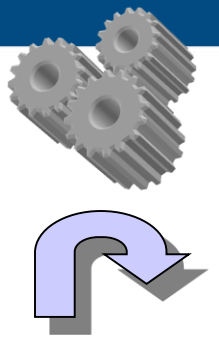
- La commande **ps** (Print process status Statistics) affiche des informations sur les différents processus en cours
- Il existe beaucoup d'options pour la commande **ps** permettant de sélectionner les processus et de choisir les informations à afficher
- Vous lirez la page du manuel (**man**) de **ps** pour plus d'informations

Exemple



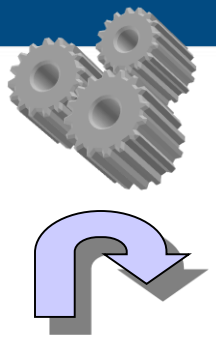
```
USER      PID %CPU %MEM    SZ   RSS TT STAT TIME COMMAND
root       76  0.0  0.0     0     0 ?  IW   0:00  (biod)
root       75  0.0  0.0     0     0 ?  IW   0:00  (biod)
root       81  0.0  0.0     0     0 ?  IW   0:00  (nfsd)
root       74  0.0  0.0     0     0 ?  IW   0:00  (biod)
root       61  0.0  2.3   264   244 ?  I    0:00 /usr/lib/sendmail -bd -q30m
gilles    240  0.0  0.1     0     0 p0 D    0:00  (csh)
root      186  0.0  0.2    16    12 ?  S    0:00 /etc/update
root       45  0.0  1.2   144   124 ?  I    0:02 /etc/syslogd
root       78  0.0  0.0     0     0 ?  IW   0:00  (biod)
root       25  0.0  1.1   128   112 ?  I    0:00 portmap
root      224  0.0  6.1   776   660 ?  I    0:00 /usr/bin/X11/sxdm
root      223  0.0  0.6    68    56 co I    0:00 - std.9600 console (getty)
root        3  0.0  0.1     0     0 ?  D    0:00  (syncd)
root        2  0.0  0.4 32768     0 ?  D    0:00 pagedaemon
root        1  0.0  0.5    64    48 ?  I    0:00 init -
root        0  0.0  0.1     0     0 ?  D    0:13 swapper
root      214  0.0  0.0           0 co IW   0:00  (consd)
```

Avant-plan / arrière-plan



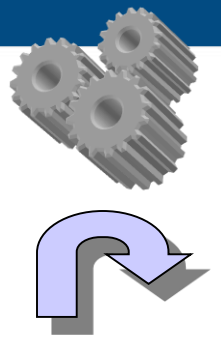
- Un processus lancé en avant-plan bloque l'utilisation de l'interpréteur de commande jusqu'à la fin de son exécution
- Pour lancer un processus en arrière-plan, il suffit d'ajouter le signe « **&** » à la fin de la commande
- Le processus lancé en arrière-plan se lance, affiche son PID, et l'utilisateur peut continuer à utiliser son shell

Signalisation des processus



- **Control-C** permet d' arrêter un processus lancé en avant-plan
- **Control-Z** permet de suspendre l'exécution d'un processus lancé en avant-plan
- la commande **fg** (ForeGround) permet de reprendre en avant plan l'exécution d'un programme suspendu
- la commande **bg** (BackGround) permet de reprendre en arrière-plan l'exécution d'un processus suspendu

Gestion des tâches (jobs)



- La commande jobs affiche l'état des processus lancés depuis un interpréteur de commandes

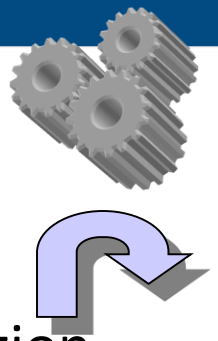
nemesis> jobs

[1] - Running xterm

[2] + Suspended xclock

- Le numéro entre crochet peut être utilisé pour signaler le processus :
- bg %2

La commande kill



- La commande **kill** suivie d'un PID permet d'arrêter l'exécution d'un processus en arrière-plan
- La commande **kill** accepte des arguments qui permettent d'envoyer au processus, différents types de signaux
- Par défaut, le signal envoyé est TERMINATE
- Exemples :

  kill 1238 (arrête le processus dont le PID est 1238)

  kill %2 (arrête le processus numéro 2 dans la liste affichée par la commande **jobs**)

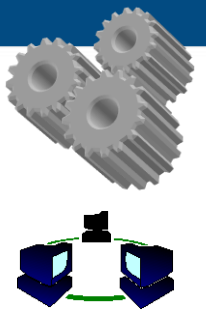
Quelques commandes réseau

- **hostname** => nom du système
- **telnet** => terminal distant
- **ftp** (File Transfer Program) => transfert de fichiers
- L'utilisation de X11

hostname

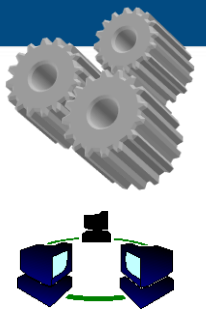
- La commande **hostname** retourne le nom du système sur lequel l'utilisateur est connecté
- Le nom d'un système doit toujours commencer par une minuscule
- Ce nom doit être unique

Terminal distant



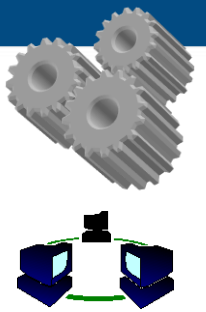
- **telnet** permet de se connecter à un autre système (généralement sous Unix) et d'obtenir un shell
- **telnet** *hostname* ou *@IP du système distant*
- **Control-]** permet d'accéder à un petit menu dans **telnet** et notamment de sortir de **telnet** quand le shell distant est bloqué

Transfert de fichiers



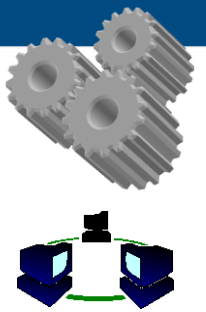
- fonction : transfert de fichiers entre machines
- interface : interactif, aide en ligne
- daemon : **ftpd**, peut être lancé par **inetd**
- fichiers utilisés :
  - **/etc/passwd** : le nom d'utilisateur et le mot de passe fourni doivent exister dans le fichier **passwd** distant
  - **/etc/shells** : l'utilisateur doit utiliser un shell appartenant au fichier **/etc/shells** distant
  - **/etc/ftpusers** : liste des utilisateurs n'ayant pas le droit d'utiliser **ftp** (typiquement les utilisateurs anonymes comme root, uucp, ingres, ...)
  - $HOME/.netrc : permet à ftp de générer automatiquement le login. !!! TRES DANGEREUX !!!

Paramètres de ftp



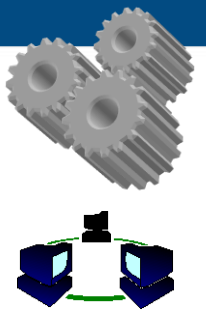
- -**v** : verbose, affiche les messages du serveur distant (**ftpd**) et le nombre de blocks transférés
- -**n** : nologin, supprime le login automatique à la connexion
- -**i** : no inquire, supprime le "prompt" dans les transferts multiples
- -**g** : no global, supprime le traitement des caractères de remplacement (wildcards : *, ?)
- -**d** : debug
- *hostname* : nom du système distant

Commandes de ftp



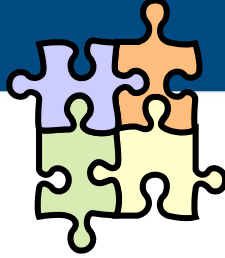
- **! [command]** : exécution en local de la commande passée en paramètre ou du shell si pas de paramètre
- **append** *fl [fd]* : concatène le fichier local (*fl*) au fichier distant (*fd*)
- **ascii** : transfert de fichiers de type texte
- **bell** : sonnerie à la fin de chaque transfert de fichier
- **binary** : transfert de fichiers binaires
- **bye, quit** : fin de ftp
- **cd** *répertoire* : changement de répertoire sur le système distant
- **close** : termine la connexion avec le système distant
- **delete** *fd* : efface un fichier distant (fd)
- **ls [-l]** [*répertoire*] [*fl*] : liste un répertoire distant avec redirection éventuelle dans un fichier local (*fl*) (**dir** = **ls -l**)
- **get** *fd [fl]* : réception du fichier distant (*fd*); **recv**=**get**
- **glob** : identique à l'option **-g**

Commandes de ftp



- **hash** : supprime l'affichage de # par bloc transféré
- **help** [*commande*] : liste des commandes ou aide sur la commande passée en paramètre. (? = help)
- **lcd** [*répertoire*] : changement du répertoire local.
  répertoire initial ($HOME) par défaut.
- **mdelete** *fd1 .. fdn* : effacement de plusieurs fichiers distants
- **mls** : identique à **ls** pour une liste avec caractères de remplacement (mode global activé)
- **mget** *fd1 .. fdn* : réception de plusieurs fichiers distants
- **mkdir** *répertoire* : création d'un répertoire distant
- **mput** *fl1 .. fln* : émission de plusieurs fichiers
- **open** *hostname* : connexion avec le système distant
- **prompt** : demande confirmation à chaque fichier pour les commandes en m (**mget**, **mput**, **mdelete**).
- **put** *fl [fd]* : émission du fichier local (*fl*); **send**=**put**
- **pwd** : affiche le nom du répertoire distant courant

Programmation de scripts

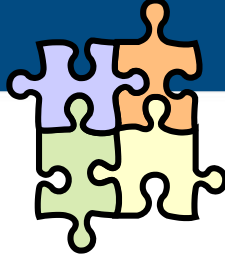


- Les commandes de l'interpréteur C-shell d'Unix et les programmes peuvent être combinés dans des scripts. La syntaxe du C-shell est proche de celle du langage C, d'où son nom.
- Un script est un fichier texte qui regroupe des commandes.
- Le script commence par l'indication du shell capable de l'interpréter. En l'occurrence pour le C-shell :
- **# !** /bin/csh
  si csh se trouve dans le répertoire /bin.

- Le fichier est rendu exécutable en ajoutant les droits d'exécution : **chmod** +**x** *nom_du_fichier*

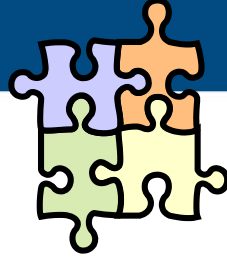
- Lorsque le script est lancé, l'interpréteur de commandes (le Shell) exécute successivement chaque ligne.

Le mode mise au point



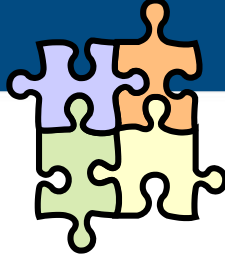
- Lorsqu'une erreur survient lors de l'exécution d'un script, il n'est pas toujours facile de déterminer à quelle ligne l'erreur s'est produite.
- Il est possible de lancer le csh en mode debug pour qu'il affiche les lignes de code au cours de l'exécution :
- csh –x programme

Les variables



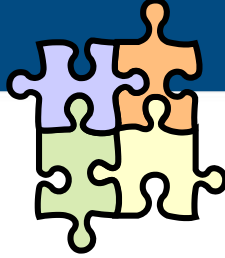
- Les noms des variables ne doivent pas contenir de caractères spéciaux comme $, *, ?, … Mais le caractère _ « souligné » est possible.
- La longueur maximale du nom de la variable est de 20 caractères.
- La valeur d'une variable peut être une chaîne de caractères ou une liste de chaînes de caractères.
- Certaines variables sont prédéfinies :
  La variable **argv** contient la liste des paramètres passés à votre script.
- Ces variables sont locales au shell, elles ne sont pas visibles de l'extérieur du script et ne survivent pas à la terminaison du script

Affectation d'une valeur



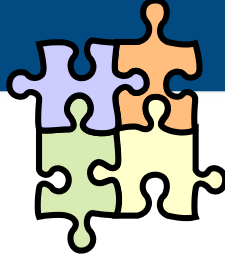
- Une valeur est affectée à une variable par la commande **set**.
- **set** *nom_de_variable*
  affecte une chaîne de caractères vide à la variable.
- **set** *nom_de_variable* **=** *valeur*
  affecte une valeur chaîne de caractères à la variable.
  - Exemple : **set** *var* **=** toto

Variables de type liste



- **set** *nom_de_variable* **=** (liste de chaînes de caractères)
  affecte une liste à une variable.
  - Exemple : **set** *var* **=** **(***toto titi tutu***)**
- **set** *nom_de_variable[indice]* **=** valeur
  affecte une valeur chaîne de caractères à la position indiquée par l'indice dans la liste contenue dans la variable. La variable doit préalablement exister.
  - Exemple : **set** *var*[*1*] **=** *tata*

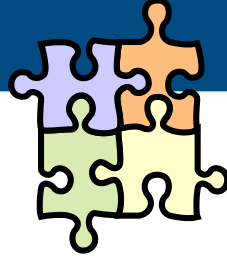
Destruction de variables



- **unset** *nom_de_variable*
  détruit la variable.

- **unset** *pattern*
  Il est possible d'utiliser des caractères « joker » à la place du nom de variable :
  - **unset** *v**
    détruit toutes les variables dont le nom commence par v.
  - **unset** *
    détruit toutes les variables.

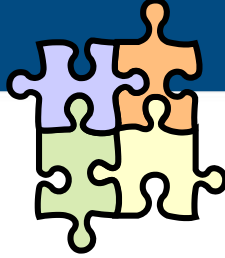
calculs



- Bien que les variables soient de type chaînes de caractères, il est possible d'effectuer des calculs.
- **@** *variable* **=** expression mathématique
- affecte le résultat de l'expression mathématique dans la variable.
- **@** *variable***[**indice**] =** expression mathématique.
  affecte le résultat de l'expression mathématique à la position indiquée par indice dans la liste contenue par la variable.
- Exemples **:**

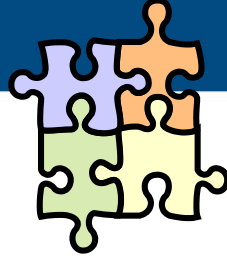
@ *var* **= 2 + 3**	var contiendra 5
@ *var***++**	var est incrémentée de 1

La variable status



- La variable interne status contient la valeur du code retour de la dernière commande exécutée
- 0 lorsque la commande c'est bien exécutée
- > 0 en cas d'erreur

Évaluation d'une expression

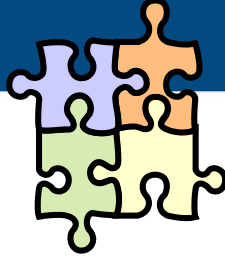


- La commande expr permet d'évaluer une expression
- C'est un moyen commode pour savoir si une valeur est un entier

Exemple :

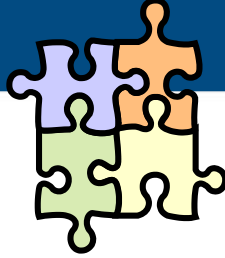
```
expr $argv[1] + 1 >& /dev/null
if ($status != 2) then
        echo le parametre est bien un entier
endif
```

Les variables d'environnement



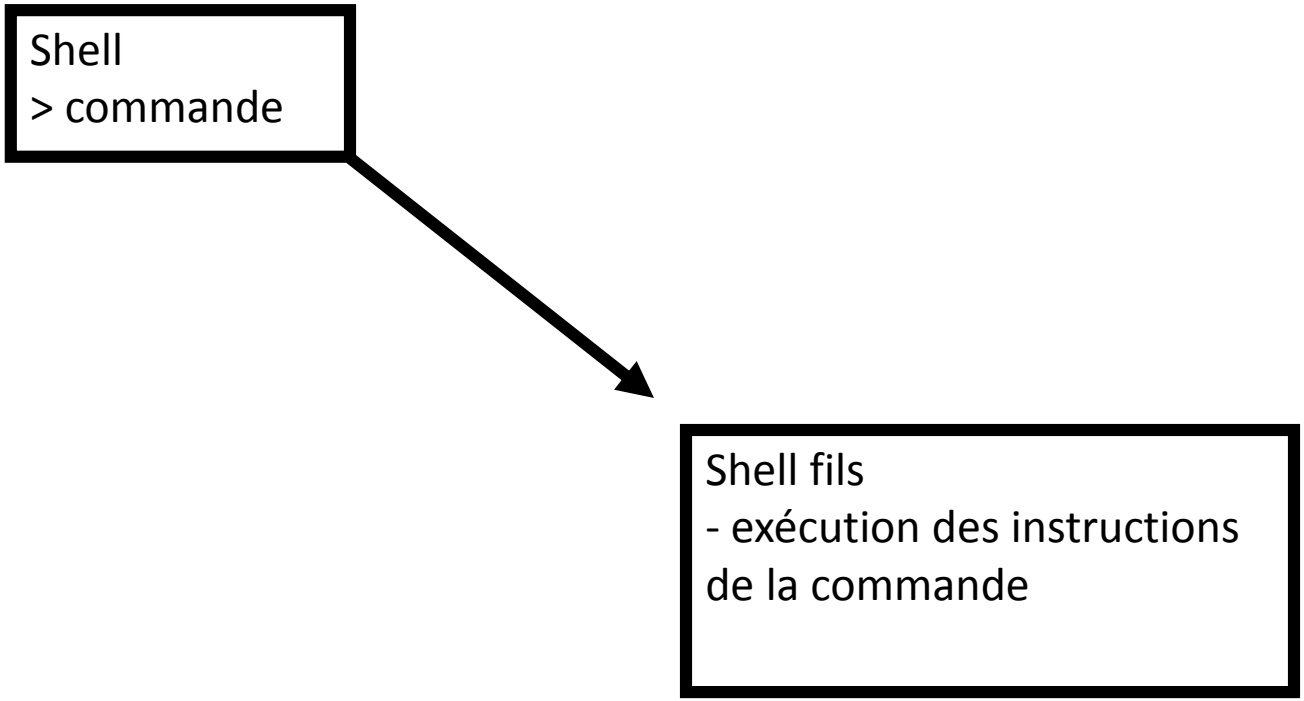
- Les variables d'environnement sont globales : elles sont accessibles en lecture depuis les scripts
- Elles sont généralement notées en majuscules
- Exemples :
  - DISPLAY
  - USER
  - PATH

Exécution d'un script dans l'environnement courant

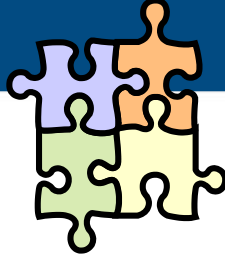


- Les variables d'environnement positionnées dans un Shell sont visibles de tous les processus fils de ce Shell, mais n'ont pas d'effet sur les valeurs des variables d'environnement du processus père sauf en demandant l'exécution du script dans le même processus que l'interpréteur de commandes (et non pas dans un de ses fils).
- **source** *nom_du_programme*

Exécution normale

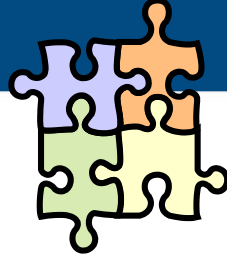


Exécution avec source



shell
> source commande
- exécution des instructions de la commande

Positionnement des variables d'environnement

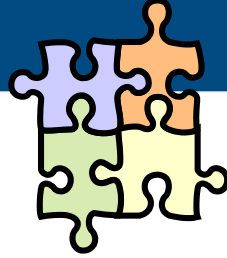


- **setenv**
  sans paramètres affiche la liste des variables d'environnement.

- **setenv** *nom_de_variable* valeur
  affecte une valeur dans la variable

- **unsetenv** *pattern*
  destruction d'une ou de plusieurs variables d'environnement

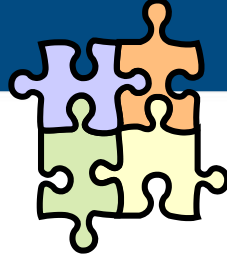
printenv



- La commande **printenv** affiche la liste des variables d'environnement avec leurs valeurs
- Exemple :

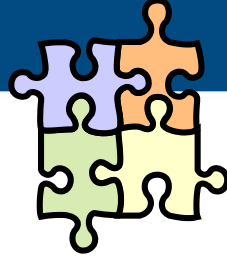
```
kounak% printenv
TERM=vt100
HOME=/home/gilles
SHELL=/bin/csh
USER=gilles
PATH=/local_user/unix/ioffice/bin:/usr/local/bin:/etc:/usr/etc:/u
   sr/ucb:/bin:/usr/bin:/usr/bin/X11:/usr/sony/bin:/usr/sony/bin/
   X11:/usr/new:/usr/new/mh:/usr/pds/bin:/usr/hosts:/usr/games:.
LOGNAME=gilles
MANPATH=/usr/local/man:/usr/man
IOFFICE=/local_user/unix/ioffice
```

Variables prédéfinies



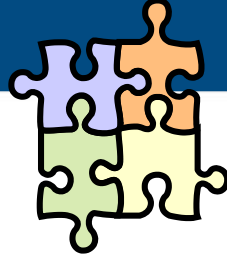
- Certaines variables (systèmes) sont prédéfinies :
  - path
  - home
  - term
  - user
  - shell
  - status
- Elles sont en minuscules en Cshell, mais s'affichent en majuscules avec **printenv**

La variable path



- La variable d'environnement **path** contient la liste des répertoires dans les quels l'interpréteur de commandes cherchera le code correspondant à la commande tapée
- Lorsque le shell répond « command not found », la commande ne se trouve pas dans le chemin de recherche, mais elle peut exister ailleurs sur le disque
- Si une commande existe dans plusieurs répertoires indiqués dans le **path** (par exemple **ps**), c'est la première dans l'ordre du **path** qui sera exécutée

which

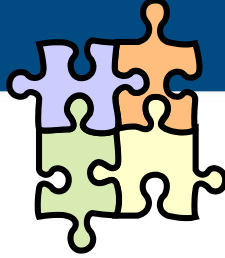


- La commande **which** permet de savoir dans quel répertoire indiqué dans le PATH, une commande se trouve
- Exemple :

which ls

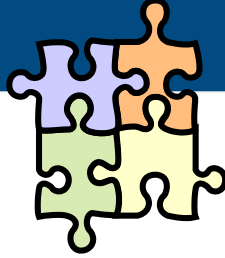
/bin/ls

Substitution de variables



- L'accès au contenu d'une variable s'effectue par substitution.
- L'interpréteur cherche le caractère **$** sur une ligne de commande.
- Lorsqu'il le trouve, il vérifie que le caractère **$** précède le nom d'une variable.
- Si ce n'est pas le cas, une erreur est générée.
- Sinon, l'interpréteur remplace **$** et le nom de la variable par la valeur de la variable.

Affichage dans un script



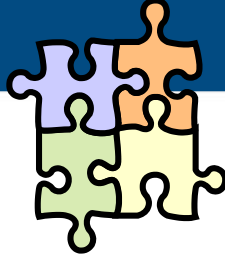
- **echo** *liste de mots*
- affiche ses arguments sur la sortie-standard (généralement l'écran) suivi d'un saut de ligne sauf si **–n** est précisé avant la liste de mots.

- Exemple :

set var **=** *toto*

echo $var *essai* affiche toto essai

Les accolades



- Les accolades **{ }** permettent d'isoler le nom de la variable lorsque celui-ci est collé avec d'autres caractères (dans une liste par exemple) afin d'éviter que le shell n'interprète les caractères suivants comme faisant partie du nom de la variable.
- Exemple :

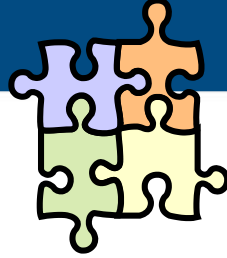
```
setenv CLASSPATH $CLASSPATH:../classes:.
```

=>provoquerait une erreur

pour l'éviter =>

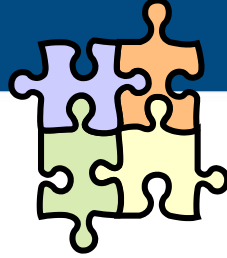
```
setenv CLASSPATH ${CLASSPATH}:../classes:.
```

Extraction de sous-listes



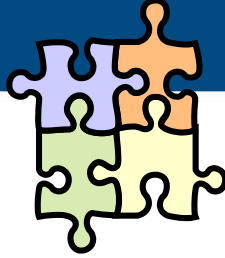
- Dans le cas d'une variable qui contient une liste, il est possible d'extraire un élément de la liste, ou une sous-liste :
- **$**variable**[***indice***]** extrait l'élément à la position indiquée par *l'indice*. Les indices commencent à 1.
- **$**variable**[***début* **–** *fin***]** extrait la sous-liste commençant à la position *début* et se terminant à la position *fin*.
  - Si *début* n'est pas spécifié, l'interpréteur assume que c'est 1.
  - Si *fin* n'est pas spécifié, l'interpréteur assume que c'est le dernier élément de la liste.
  - *début* et *fin* peuvent être remplacés par *, dans ce cas tous les éléments seront extraits de la liste.
- **$#**variable retourne le nombre d'éléments de la liste contenue dans la variable.

Récupération des paramètres du script



- **$***nombre* est équivalent à **$***argv***[***nombre***]**
- retourne la valeur de l'argument à la position indiquée par *nombre* de la commande qui à lancé le script.
- **$**0 ou **$**argv**[**0**]** renvoie le nom de la commande (le nom du fichier qui contient le script).
- **$*** est équivalent à **$**argv**[*]** et renvoie la ligne de commande qui a lancé le script.

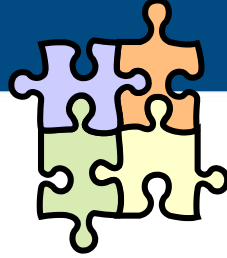
Entrée au clavier



- **$<** substitue une ligne lue sur l'entrée standard (le clavier) et permet donc de saisir des chaînes de caractères dans les scripts.
- Exemple :

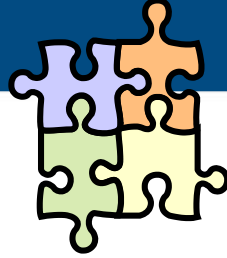
set ligne **= $<**

Les instructions de contrôle



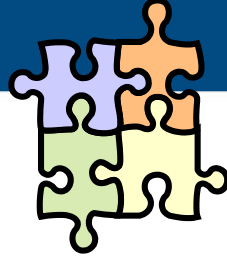
- Les instructions de contrôle (**if**, **foreach**, **while**, …) évaluent les expressions booléennes pour décider de la séquence des instructions.
- **&&** : et
- **||** : ou
- **!** : non
- **==** : est égal
- **!=** : est différent de
- **-d** *nom* est vrai si *nom* est un répertoire
- **-f** *nom* est vrai si *nom* est un fichier
- **-e** *nom* est vrai si *nom* existe
- **-r –w –x –o** pour tester les droits en lecture, écriture, exécution et propriété.

Les opérateurs



- Ne confondez pas l'affectation d'une valeur dans une variable (=) et la comparaison de deux valeurs (==)
- Placez des espaces autour des opérateurs logiques et arithmétiques

Condition : if then else



- **if (***expression***)** commande
- si l'expression est vraie, alors la commande est exécutée.
- **If (***expression1***) then**

  bloc de commande 1

else if (*expression2***) then**

  bloc de commande 2

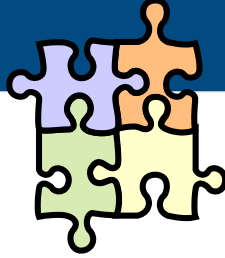
else

  bloc de commande 3

endif

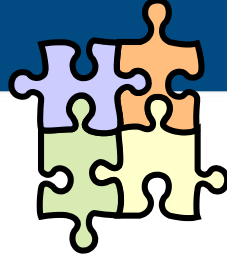
- si l'expression1 est vraie alors le bloc de commande 1 est exécuté, sinon si l'expression2 est vraie alors le bloc de commande 2 est exécuté sinon le bloc de commande 3 est exécuté.

Exemple



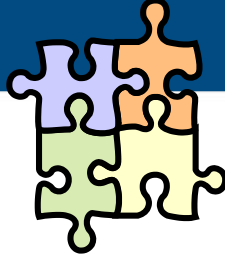
```
if ($#argv != 1) then
echo nombre de parametres
  incorrect
  exit 2
endif
```

Branchements conditionnels



- **switch (**chaine**)**
  case chaine1 **:**
  bloc1
  breaksw
  case chaine2 **:**
  bloc2
  breaksw
  default:
  bloc3
  breaksw
  endsw
- chaque chaîne de **case** est successivement comparée à la chaîne du **switch** et en cas d'égalité, le bloc d'instruction correspondant est exécuté. Si aucune chaîne ne correspond, le bloc d'instruction par défaut est exécuté.

La boucle while



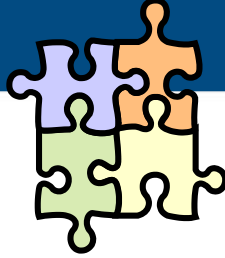
- **while (***expression***)**

  bloc de commande

end

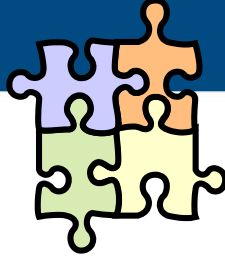
- le bloc de commande est exécuté tant que l'expression n'est pas nulle.

La boucle repeat



- **repeat** *nombre commande*
- la commande est exécutée autant de fois que l'indique le nombre.

La boucle foreach



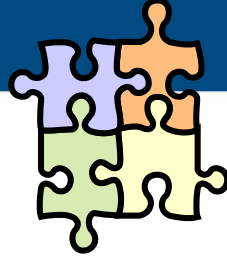
- **foreach** variable **(**liste de valeurs**)**

  bloc d'instructions

end

- la boucle est exécutée autant de fois qu'il y a de valeurs dans la liste. Chaque valeur de la liste est affectée successivement à la variable.

Exemple de foreach



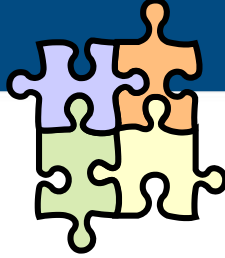
- Exemple :

```
foreach var (toto titi tutu)
    echo $var
end
```

- Affichera

```
toto
titi
tutu
```

Sortie inconditionnelle de boucle



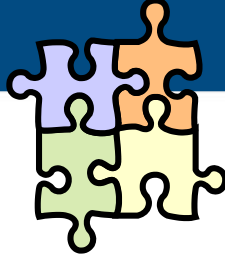
- **break**

  sortie d'une boucle.

- **continue**

  passage à l'itération suivante de la boucle.

Substitution de commandes

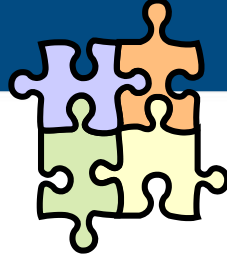


- Il est possible de récupérer le résultat de l'exécution d'une instruction en l'entourant par les caractères `` (backquote)
- Exemple :

```
set nblignes = `wc -l mon_fichier`
```

affecte dans la variable nblignes le nombre de lignes du fichier *mon_fichier*

alias



- Il est possible de créer des alias (des raccourcis) pour éviter de taper de longues commandes avec beaucoup de paramètres
- Exemple :
  alias ll ‘ls –alg’
  la commande **ll** se comportera comme si l’utilisateur avait tapé **ls –alg**
- **alias** sans paramètre affiche la liste des définitions d’alias