

Systeme d'exploitation Unix

aspects utilisateur

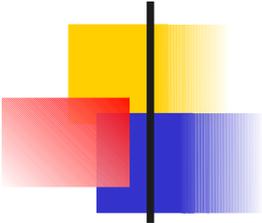
http://www.lisyc.univ-brest.fr/pages_perso/lemarch/Cours/

Laurent Lemarchand

LISyC/UBO

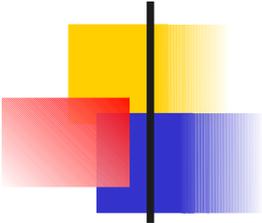
Laurent.Lemarchand@univ-brest.fr

D'après Ph. LeParc, S. Rubini, A. Lewandowski, Wikipedia, ...



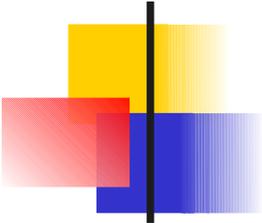
Organisation du cours

- Plan général:
 - Introduction sur les systèmes d'exploitation
 - Le système de fichiers
 - Les processus
 - Interface graphique et connexion
 - Initiation au shell (csh) et à la gestion de l'environnement de travail
 - Les expressions régulières
 - Awk



Bibliographie

- J.M Rifflet, *La programmation sous Unix*, 3ème édition, chez Ediscience
- *Learning the Unix Operating System*, chez O'Reilly
- *Learning the `vi` editor*, chez O'Reilly
- Consultable en ligne:
 - <http://www.root66.net/linux/Linux-france.org/article/ohoarau/>
 - ...

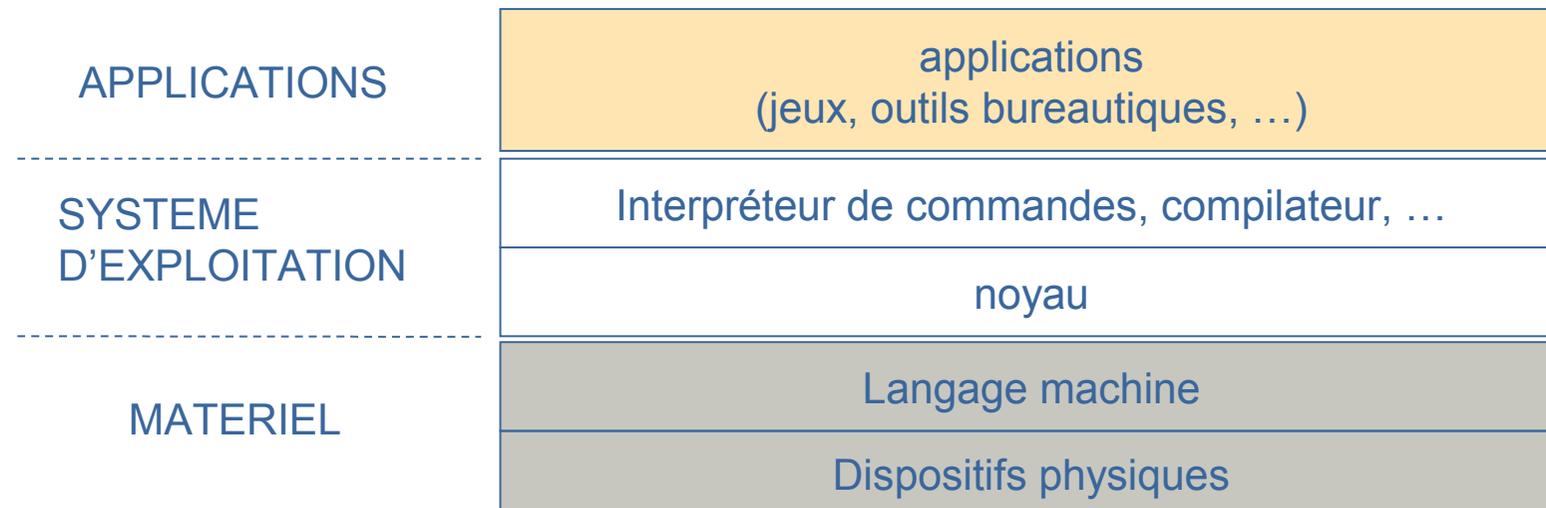
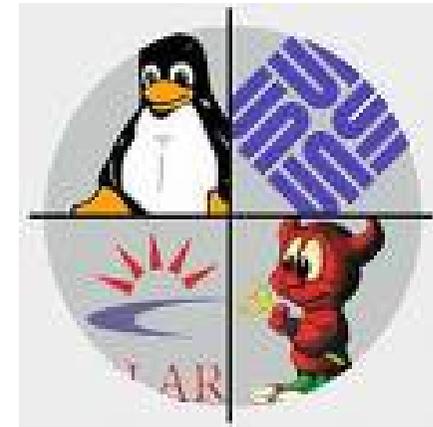


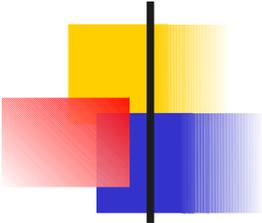
Les systèmes d'exploitation

- C'est l'interface entre l'utilisateur et le matériel
- Ses fonctions principales sont :
 - Contrôle des ressources (allocation et gestion du CPU et de la mémoire)
 - Contrôle des processus
 - Contrôle des périphériques
 - ...
- Il contient des outils de gestion utilisables par les applications, tels que la manipulation de fichiers, gestion d'impressions, date...

Les systèmes d'exploitation

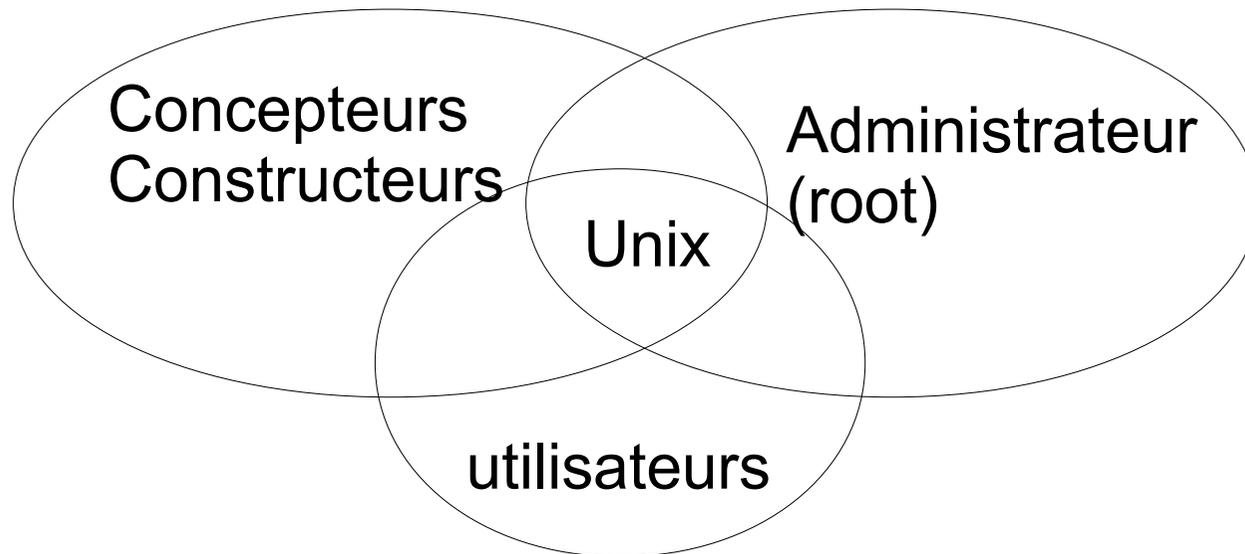
- Exemples:
 - Unix, DOS, Windows, Mac OS, Linux, OS/2, BSD, android ...
- Architecture-type:





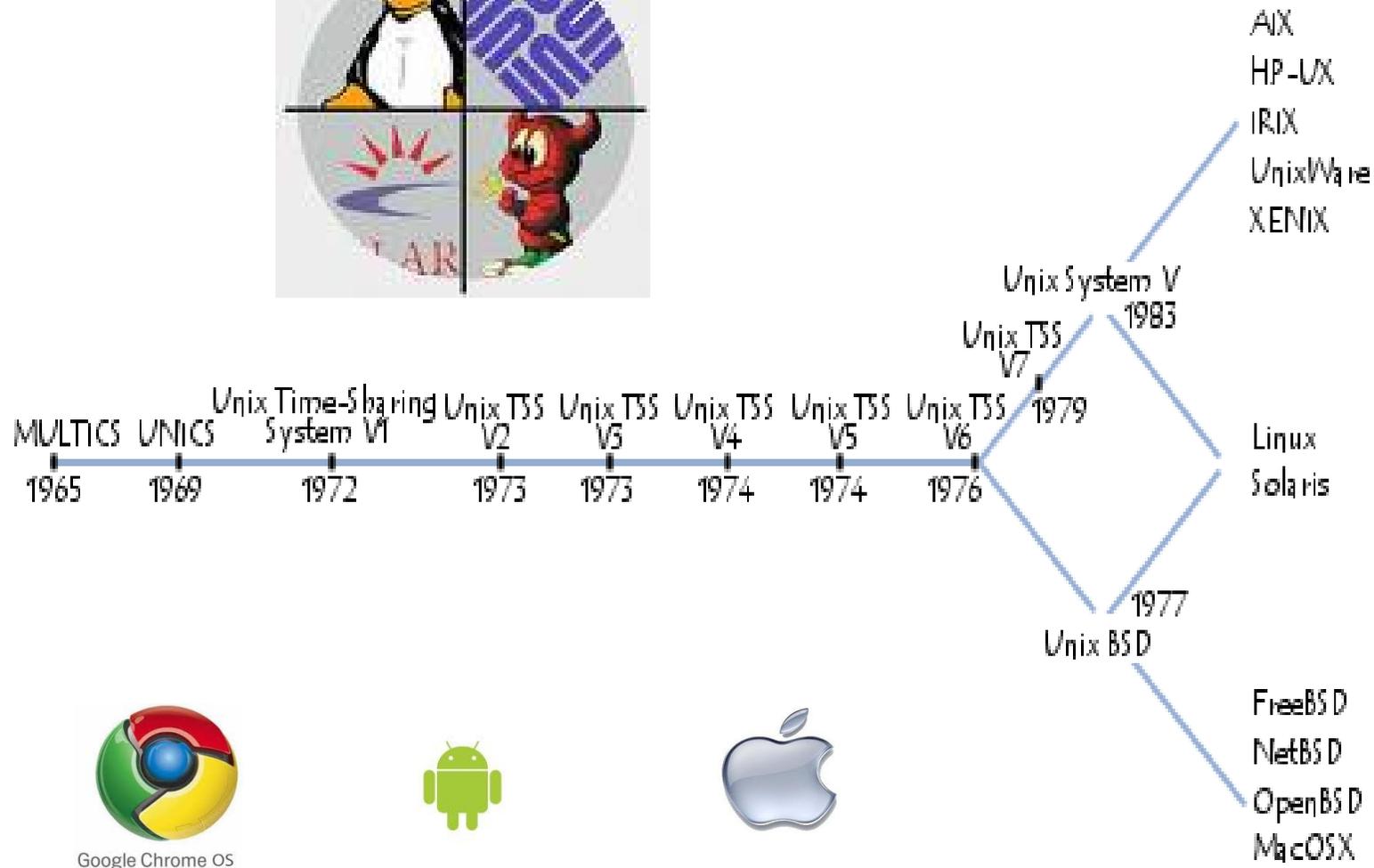
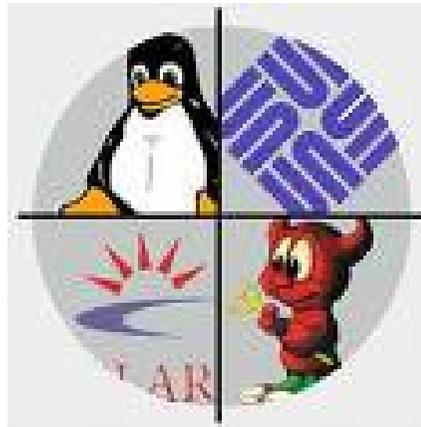
Evolution du système

- Intervenants sur le système



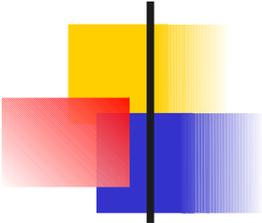
- **Constructeurs** : nouvelles primitives, pilotes pour nouveaux matériels, normalisation
- **Administrateur** root : maitre de la machine, gère les utilisateurs, les matériels, les logiciels ajoutés
- **Utilisateurs** : utilisation, personnalisation possible de l'environnement

Unix/Linux - Historique



Google Chrome OS





Unix/Linux - Historique

1969: Ken Thompson (Bell labs / MIT) : volonté d'avoir un système multitâches et multiutilisateurs, fiable pour être utilisé par des laboratoires de recherche.

1973: écriture en C -> système portable

1974-1977: distribution aux universités

1979: AT&T commercialise UNIX

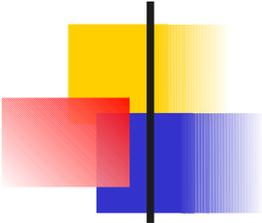
Apparitions des UNIX propriétaires

ULTRIX (DEC), AIX (IBM), HP-UX

1987 : A.S. Tanenbaum : Minix en cours d'informatique

1990 : Larry Wall : Perl pour la gestion système

1991 : Linus Torvalds (étudiant finlandais de 21 ans) : LINUX basé sur Minix



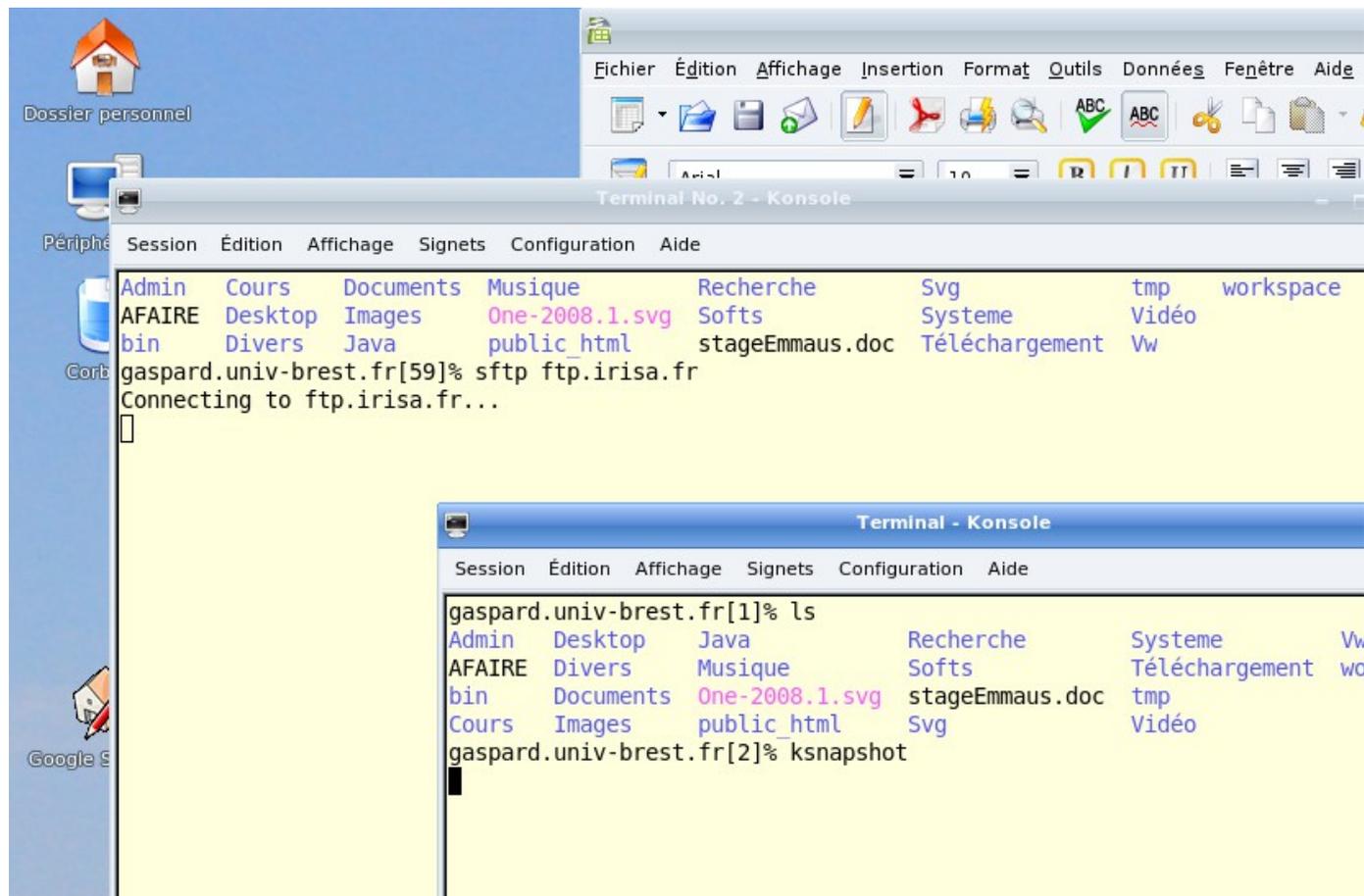
Unix/Linux - Caractéristiques

- Système multi-utilisateurs
 - Les ressources du système sont partagées entre plusieurs utilisateurs
- Système multi-tâches
 - plusieurs travaux (processus) sont réalisés « en même temps ».
- Système portable : essentiellement écrit en C
- Parfois gratuit (et libre)

Unix/Linux : propriétés

■ Multi-taches

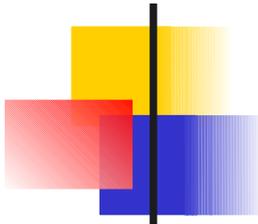
- <> DOS
- Plusieurs programmes ou commandes exécutées simultanément



Unix/Linux : machine virtuelle

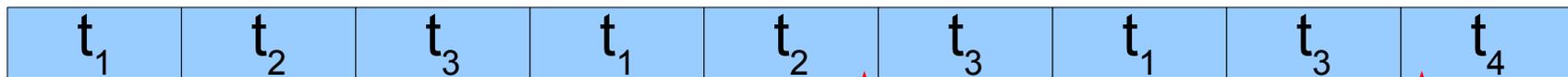
- **Un Linux dans votre Windows**
 - Host : l'hôte, votre système par exemple Windows
 - Guest : l'invité (machine virtuelle) par exemple Linux Ubuntu
- Pour faire tourner un OS dans un autre
 - L'invité est un simple fichier image (2-6 Go, quand même:-)
- Le logiciel
 - <http://www.virtualbox.org/wiki/Downloads>
- Des images
 - <http://virtualboxes.org/images/ubuntu/>





Multi-tâche

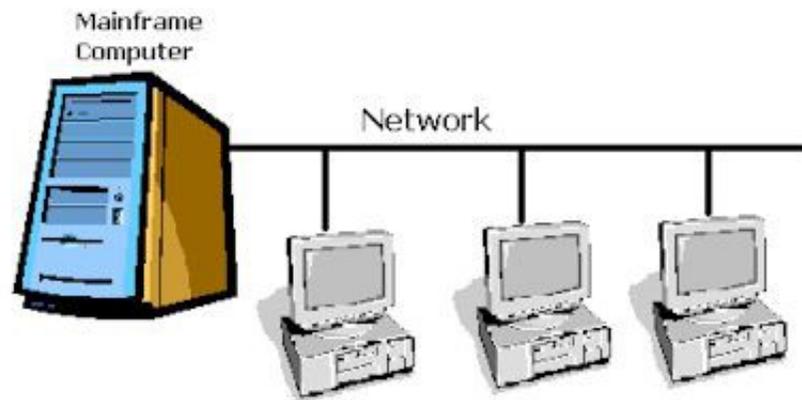
- L'ensemble des activités (tâches) se déroule simultanément
 - Chaque tâche a droit à un temps de calcul
 - Le temps est découpé en tranches
 - En apparence, les tâches s'exécutent en parallèle
 - Tire partie de la lenteur des tâches / possibilités de l'ordinateur



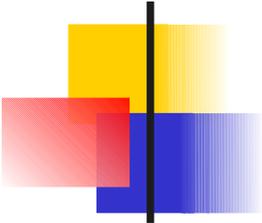
- L'ordonnanceur gère
 - l'attribution des slots de temps
 - les changements de contexte et de mémoire
 - Les priorités des tâches
- fin t_2 début t_4

Multi-utilisateurs

- Les tâches peuvent appartenir à des utilisateurs différents



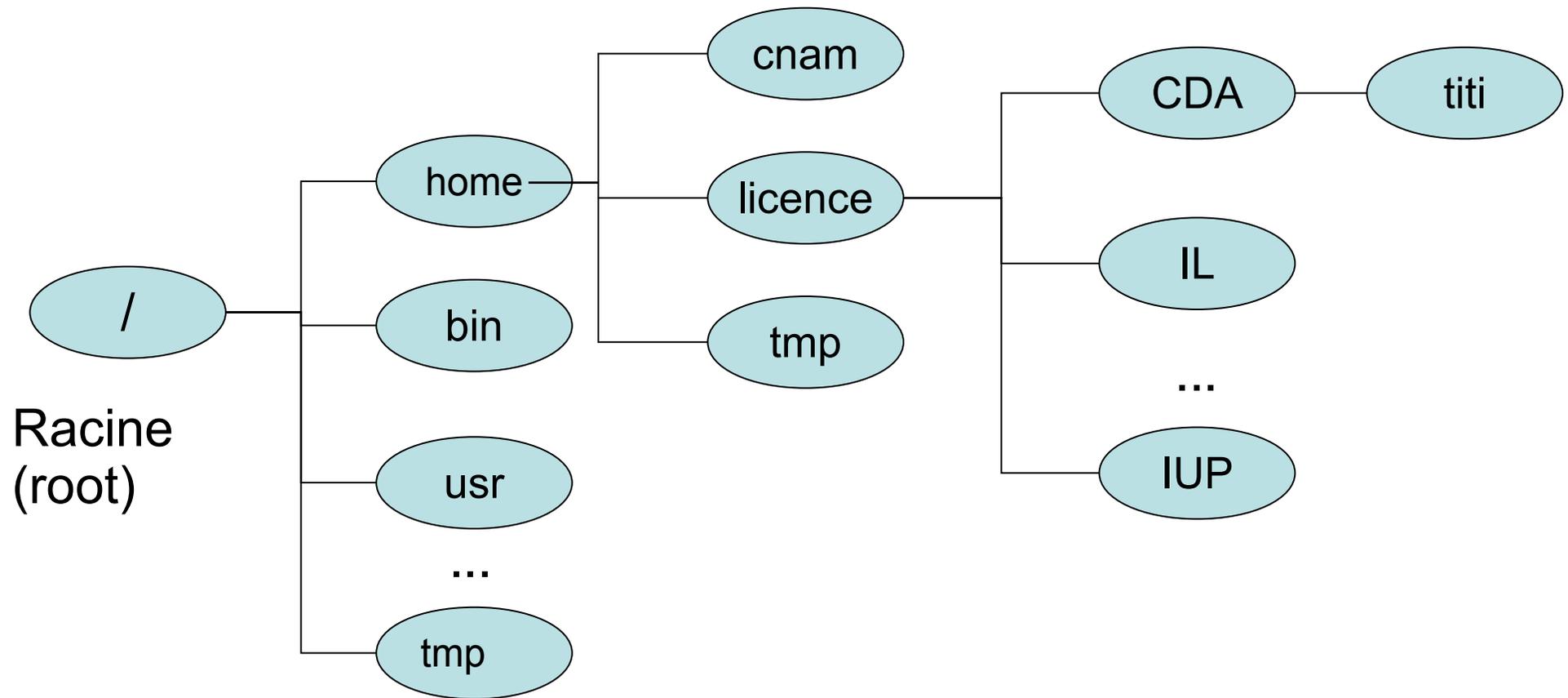
- Connexions à distance
- Taches système
- Séparation des espaces de travail
(notion de **propriétaire** pour les données et les pages mémoire)

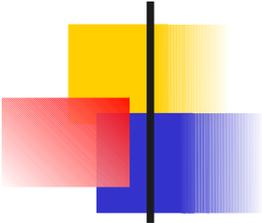


L'arborescence du système de fichiers

- Un fichier : flot d'octets (vue uniforme sur tous les périphériques d'I/O)
- Structure hiérarchique
 - Racine (root) notée / (slash)
 - Arborescence de répertoires
 - Répertoires utilisateurs (homedir ~)
- Répertoire courant d'exécution des processus
ex: pour un shell, commande `pwd`

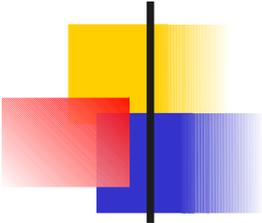
L'arborescence du système de fichiers





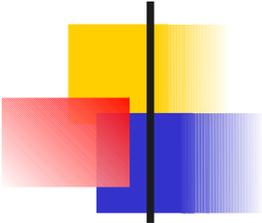
L'arborescence du système de fichiers

- `/` racine
- `/bin` utilitaires essentiels bas-niveau
- `/usr/bin` utils haut niveau et programs applicatifs
- `/sbin` utilitaires système (administration système)
- `/lib` librairies (collections d'appels systèmes liables à des programmes bas niveau par un compilateur)
- `/usr/lib` idem, mais pour programmes haut niveau
- `/tmp` stockage temporaire (pour tous)
- `/home` or `/homes` répertoires utilisateurs
- `/etc` fichiers de configuration UNIX
- `/dev` périphériques
- `/proc` pseudo-fichiers d'interface avec le noyau. Inclut un sous-répertoire pour chaque prog. actif (processus).



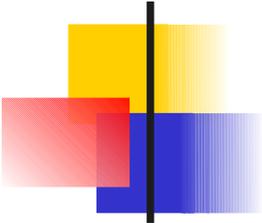
Les fichiers (1/3)

- Fichier : enregistrement sur le disque contenant des informations
- 3 types:
 - **Fichiers** standard **texte** (le codage du texte change d'une machine à une autre : transfert de fichiers) ou **code binaire** : Images (tiff, jpg, gif...)Code objet, ...
 - **Répertoires**
 - **Fichiers spéciaux** associés aux ressources du système (ex: `/dev/tty1`)



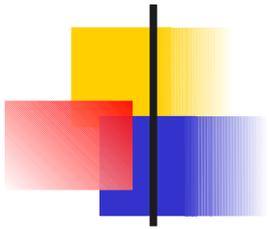
Les fichiers (2/3)

- Un fichier est identifié par son nom
 - Caractères autorisés: a-zA-Z0-9 - _
 - Caractères spéciaux :
 - Espace . , / : * & ; % ` " \ \$ ()[]{}+=<>
- exemples:
 - budget92, budget92.doc
 - Budget_2003.exc
 - notes_biophy_2003.txt



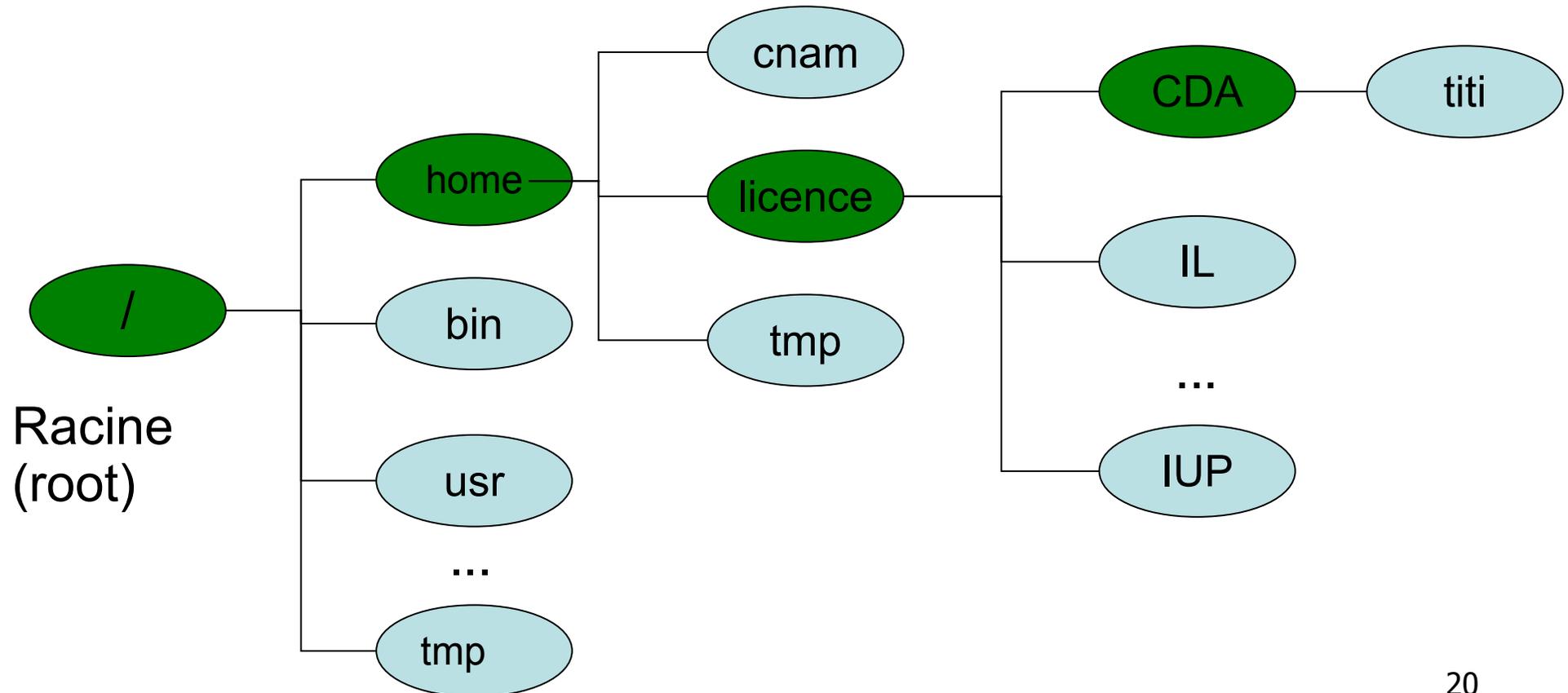
Les fichiers (3/3)

- l'extension (caractères après le point) informe sur l'application qui a créé le fichier
.c .o .ps .tar .gz .txt
- date de création du fichier
- taille en octets
- Propriétaire et droits d'utilisation



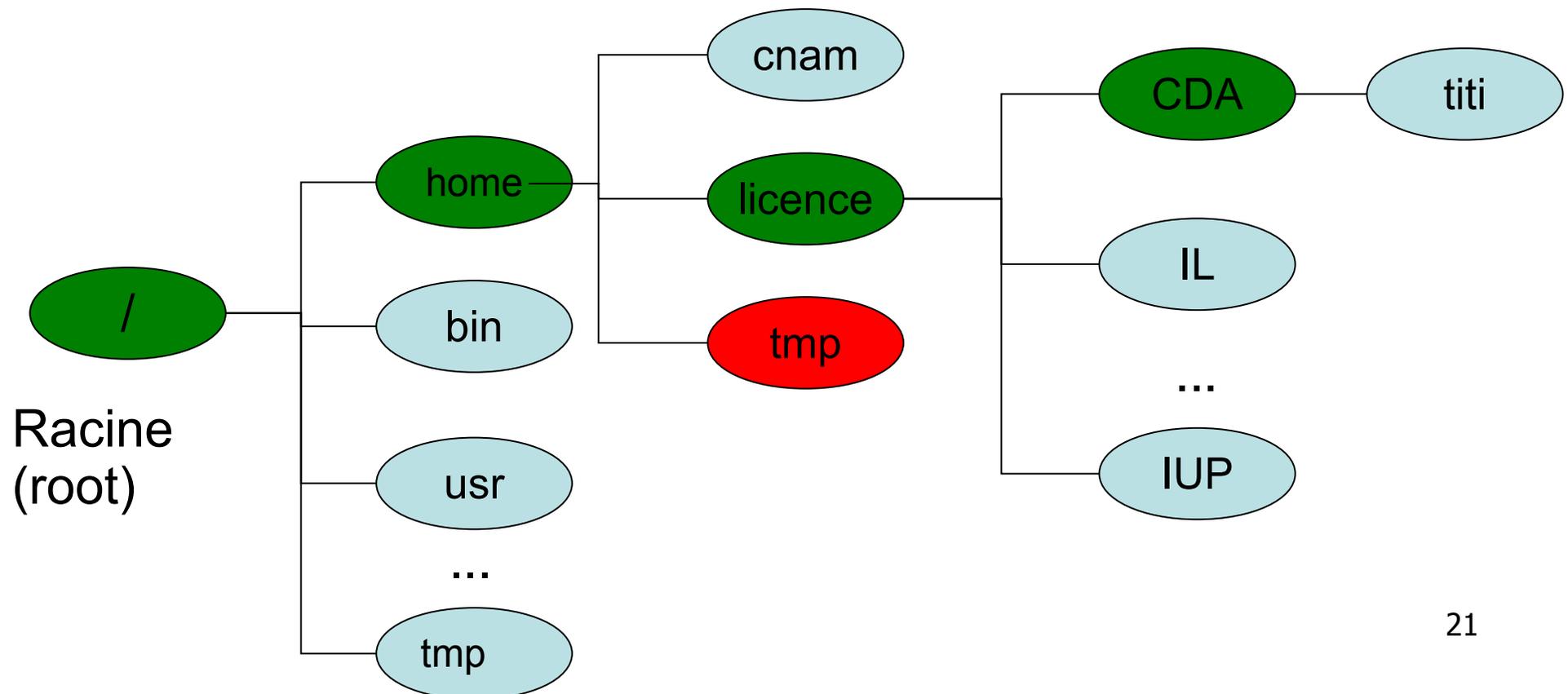
Chemin d'accès (1/2) absolu

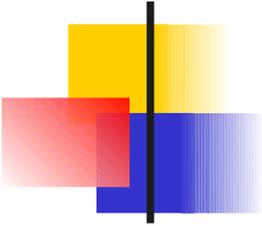
- À partir de la racine
`/home/licence/CDA`
`//tmp/bidule\ truc/machin`



Chemin d'accès (2/2) relatif

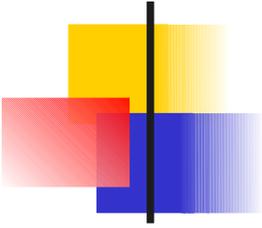
- À partir du répertoire **courant**
 - ../licence/CDA
 - ../tmp/../../../../home/licence/CDA
 - ~titi/..





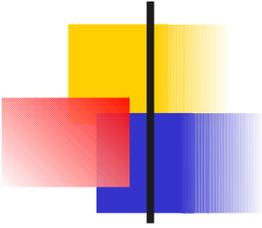
Commandes de gestion de fichiers (1/3)

- `pwd` (Print Working Directory) donne le chemin du répertoire courant
- `ls` (LiSt) (options `-l` et `-a`) liste des fichiers et répertoires dans le répertoire courant ou le répertoire donné en argument
- `cd` (Change Directory) change de répertoire
 - `cd Tests` pour aller dans le répertoire Tests
 - `cd` pour aller dans le répertoire par défaut de l'utilisateur (`cd ~user`)
 - `cd ..` pour remonter l'arborescence



Commandes de gestion de fichiers (2/3)

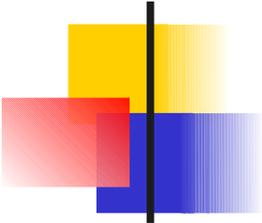
- `touch` crée un fichier vide
- `rm` efface des fichiers. `rmdir` efface des répertoires vides
- `mv` renomme fichiers et répertoires
- `cp` copie des fichiers remonter l'arborescence
- `ln` crée un nouveau lien vers un fichier



Commandes de gestion de fichiers (3/3)

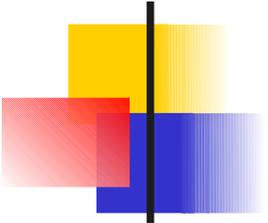
- `find` pour la recherche dans l'arborescence
- `diff` pour comparer des fichiers
- `cat` pour les afficher sur la sortie standard
- `df` pour lister les partitions

 `man` pour le manuel des commandes en ligne



Exercices

- **Q1** Quel est le répertoire courant ?
- **Q2** Vous êtes dans `/home/truc/bidule`.
Comment aller dans `/` ? dans `/home`, dans `/usr/local/bin`, dans `/home/truc`, dans votre répertoire personnel ?
- **Q3** Comment renommer le fichier `toto.c` en `titi.c` ? , le déplacer dans le répertoire `SVG` en le renommant `titi.c` ? en lui laissant son nom ?
- **Q4** Comment copier `titi.c` et `toto.c` dans `SVG` ?
- **Q5** Comment créer un nouveau fichier `truc.txt` ?
- **Q6** Comment avoir la taille de tous ces fichiers ?
- **Q7** Y a t il un fichier `.cshrc` dans le répertoire `TRUC` ?
- **Q8** Comment effacer le fichier `titi.c` et le répertoire `SVG` ?
- **Q9** Quelle est l'option de `ls` qui permet d'avoir les caractéristiques d'un répertoire et non son contenu ?



Montage de partitions (1/4)

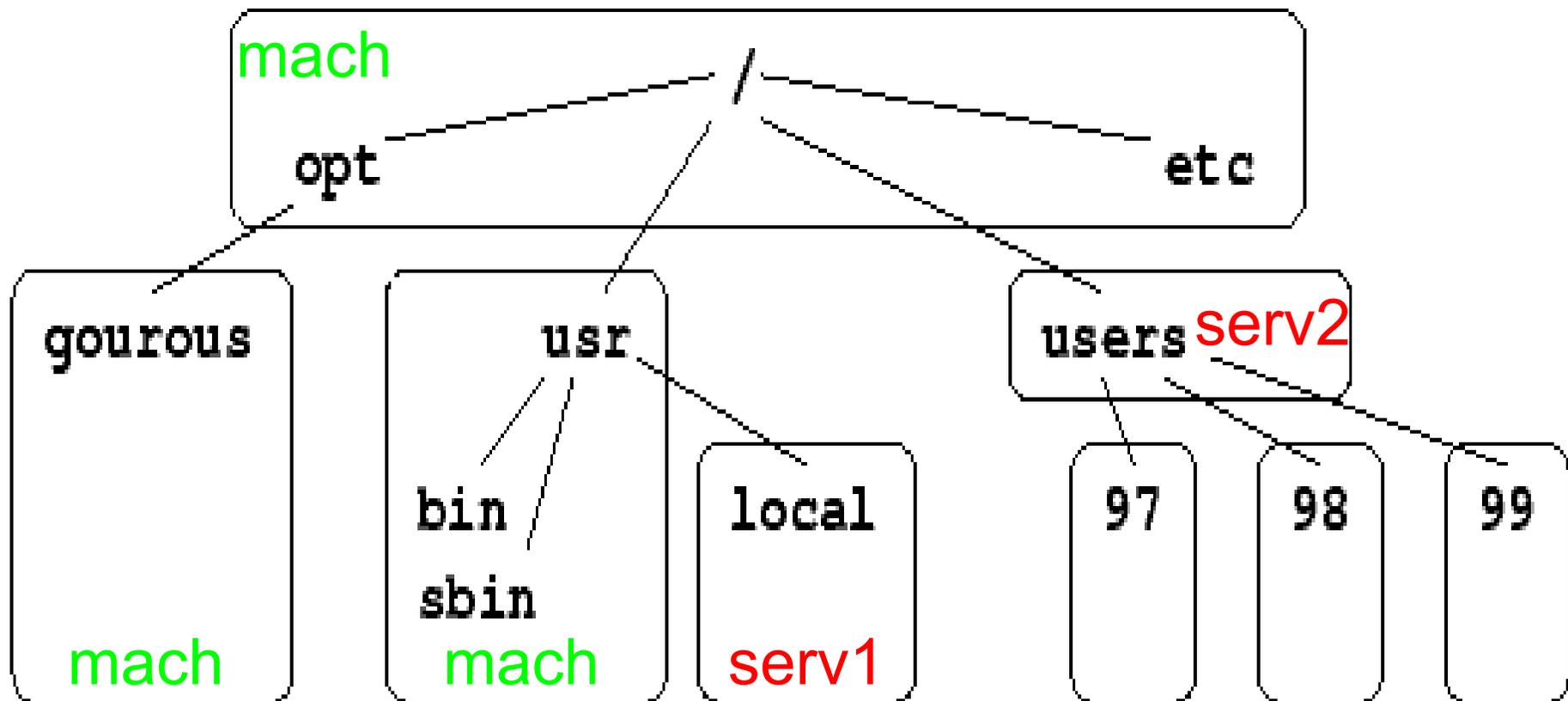
- Un disque dur est partitionné en un ensemble d'unités logiques appelées *partitions*
 - Un système de fichiers par partition
 - *Montage* de partitions **locales** ou **distantes** à l'ordinateur

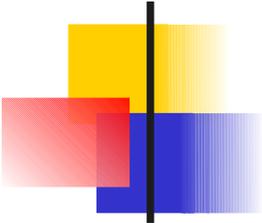
% df -h

Sys. de fich.	Tail.	Occ.	Disp.	%Occ.	Monté sur
/dev/sda1	7,7G	1,6G	5,7G	22%	/
/dev/sda7	49G	1,6G	45G	4%	/opt/gourous
/dev/sda6	16G	7,0G	8,1G	47%	/usr
serv1:/usr/local	27G	23G	3,9G	86%	/usr/local
serv2:/home2	202G	87G	106G	45%	/users

Montage de partitions (2/4)

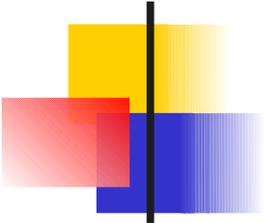
- Points de montage pour chaque partition
 - Vue de l'arborescence de la machine **mach**





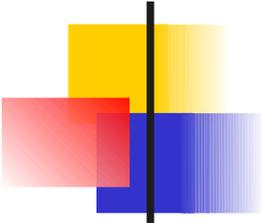
Montage de partitions (3/4)

- tous les disques amovibles (disquette, cdrom, clé usb) dans : `/mnt`
- ex pour utiliser une disquette:
 - Montage:
`mount /mnt/floppy`
 - lire/écrire dans `/mnt/floppy`
 - Démontage:
`umount /mnt/floppy`
- idem pour clés usb



Montage de partitions (4/4)

- Montages des comptes étudiants
 - /home/filieres/lic12/<login> H:
 - /home/filieres/lic12/commun I:
 - /home/filieres/commun J:



Droits d'accès (1/4)

- Accès aux fichiers réglementé (sauf: tous les droits pour **root**)
- 3 types d'utilisateurs:
 - propriétaire (**user**)
 - personnes du mm groupe (**group**)
 - les autres (**others**)

- 3 types de permissions

- lecture (**r**)
- écriture (**w**)
- exécution (**x**)

	afficher le contenu	afficher le contenu
	modifier	créer/supp fichiers
	exécuter	traverser
	fichier	répertoire

Droits d'accès (2/4)

- Affichage des caractéristiques: **ls -l**

groupe

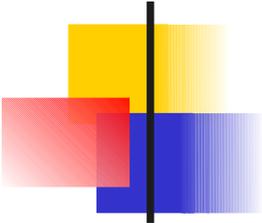
-rw-r--r-- 1 lewandowski staff 58K 16 Jul 09:19 tp1.tex

nb liens propriétaire taille date nom

type group

-rw-r--r--

user others



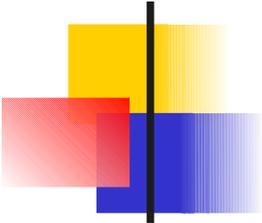
Droits d'accès (3/4)

- `ls -l` pour visualiser les droits
 - En lecture (r)
 - En écriture (w)
 - En exécution (x)

```
drwxr-xr--  1 lemarch  lib          0 Jun  6 14:25 Stmp
-rwxr--r--  1 lemarch  lib    859053 Feb  5 1999 util
-rw-r--r--  1 lemarch  lib    99040 Mar  2 1999 poly.tar.gz
```

- `chmod droits fichiers` pour gérer les droits
 - Du propriétaire (u)
 - Du groupe (g)
 - Du reste du monde (o)

```
chmod go+x util
```



Droits d'accès (4/4)

- Changer les permissions: **chmod**

chmod <classe op perm, ...>|nnn <fic>

- classe:
 - u : user
 - g : group
 - o : others
 - a : all
- op:
 - = : affectation
 - : suppr.
 - + : ajout
- perm:
 - r : lecture
 - w : écriture
 - x : exécution

- chaque perm = 1 valeur:

r	4
w	2
x	1
rien	0

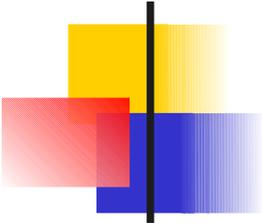
- déf. des permissions (par addition) pour chaque classe

exemples:

```
chmod u=rwx,g=rx,o=r tp1.tex
```

```
chmod a+x script.sh
```

```
chmod 755 script.sh
```



Archivage

- `tar` pour gérer une archive

`tar <commandes> <archive> <fichiers>`

- Création : `tar cvf archi.tar Rep f1 f2`

- Listing : `tar tvf archi.tar`

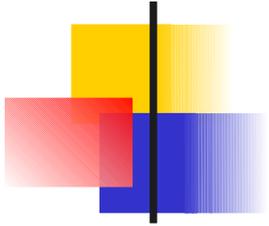
- Extraction : `tar xvf archi.tar`

- `gzip bzip compress` pour compresser une archive

`gzip archi.tar`

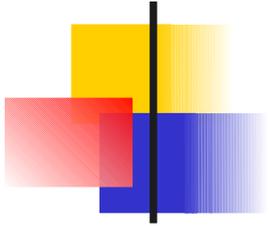


`archi.tar.gz`



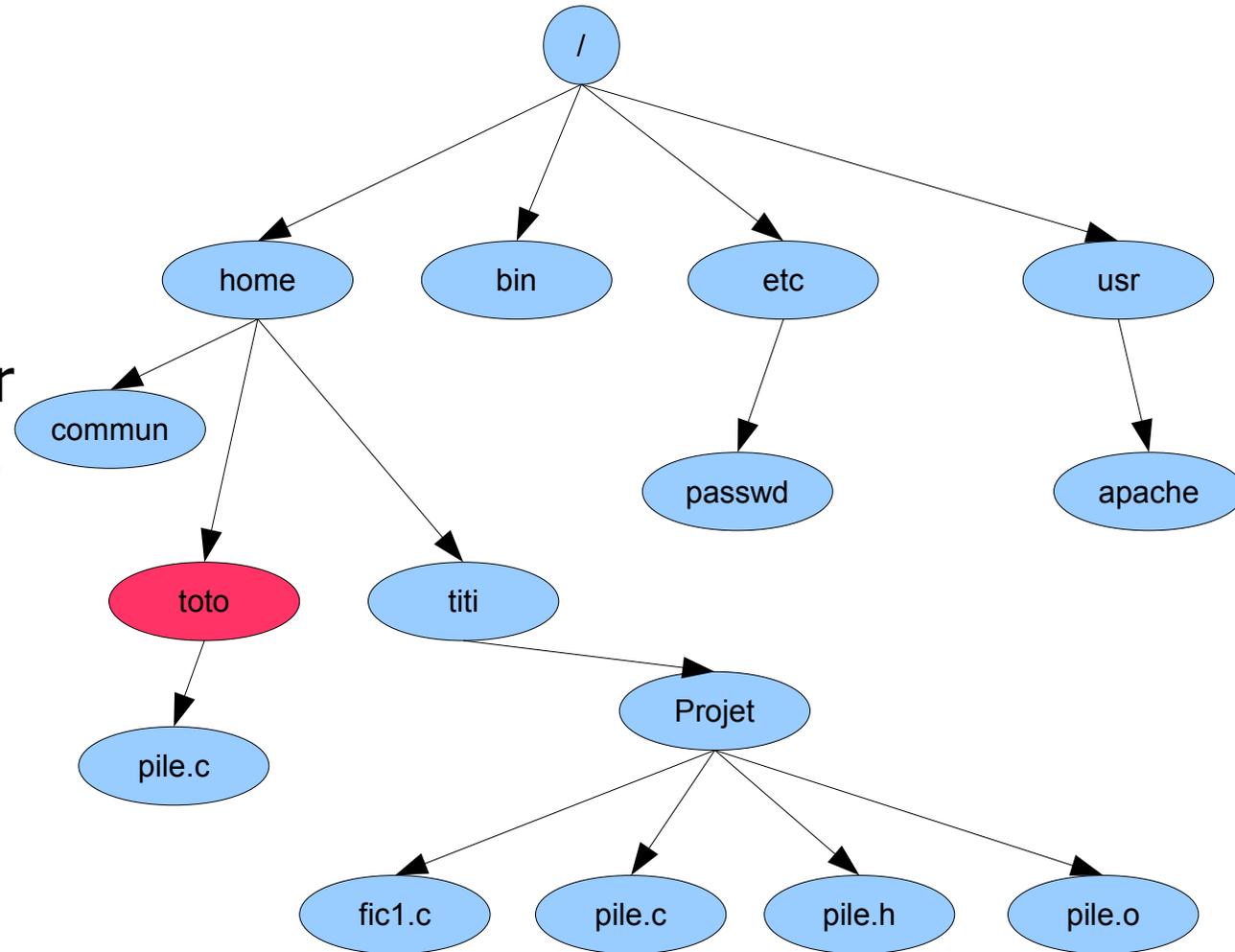
Quelques commandes

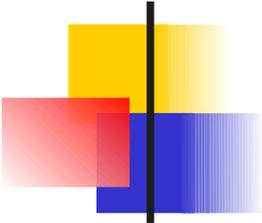
Cal	donne le calendrier d'une année
date	donne la date
sort	tri des données
head -n	affiche les n premières lignes reçues
tail -n	affiche les n dernières lignes reçues
grep	affiche les lignes contenant une expression régulière indiquée en argument
cut	sélectionne des colonnes dans une ligne
wc	compte le nb de caractères, mots, lignes
time programme	durée d'exécution de programme
which programme	localisation d'un programme



Exercices

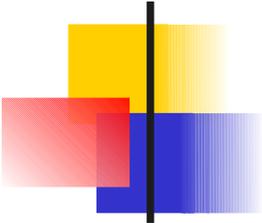
- Vous êtes dans **Toto**
- Le répertoire de chaque utilisateur n'est accessible que par lui. Le répertoire *commun* est accessible par tous en lecture et écriture. Vous travaillez avec *Titi* sur un projet *Projet* en C. Vous voulez récupérer avec *Titi* les fichiers sources qui vous manquent.





Exercices

- **Q1** Comment recopier le répertoire du projet en passant par le répertoire commun ?
- **Q2** *Titi* peut il vous donner les droits pour réaliser la copie directement ? comment faire ?
- **Q3** Comment ne copier que les fichiers sources, et pas les objets et les exécutables ?
- **Q4** Que faire pour ne pas écraser *pile.c* que vous avez modifié ? Quelle est la commande pour voir les différences entre vos 2 versions du fichier ?
- **Q5** Comment archiver votre projet ?
- **Q6** Comment connaître la partition sur laquelle vous êtes ?



Les processus (1/6)

- Processus = objet dynamique qui représente un programme en cours d'exécution et son contexte
- Caractéristiques:
 - identification (pid)
 - identification du proc. parent (ppid)
 - propriétaire
 - priorité
 - ...
- Pour voir les processus en cours: `ps`

Les processus (2/6)

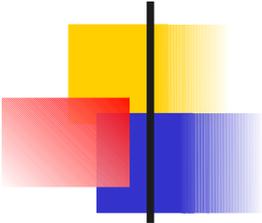
■ Infos retournées par **ps**:

	PID	TT	STAT	TIME	COMMAND	
numéro de processus	3899	p1	S	0:00.08	-zsh	temps CPU utilisé
	4743	p1	S+	0:00.14	emacs	commande exécutée
terminal associé	4180	std	S	0:00.04	-zsh	état du processus:

■ Options de **ps**:

- a liste tous les processus actifs
- u format d'affichage long
- x inclut les processus sans terminal

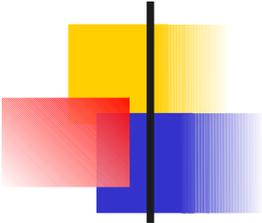
R	actif
T	bloqué
P	en attente de page
D	en attente de disque
S	endormi
IW	swappé
Z	tué



Les processus (3/6)

```
top - 13:55:46 up 24 days,  2:28,  4 users,  load average: 0.42, 0.28, 0.15
Tasks: 147 total,   2 running, 145 sleeping,   0 stopped,   0 zombie
Cpu(s):  2.3%us,  1.0%sy,  0.0%ni, 96.7%id,  0.0%wa,  0.0%hi,  0.0%si,0.0%st
Mem:   2073800k total,  1951804k used,   121996k free,   66148k buffers
Swap:  4088500k total,  448124k used,  3640376k free,   863668k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
20461	lemarch	20	0	269m	98m	41m	S	3	4.9	0:53.52	soffice.bin
3775	root	20	0	414m	110m	2924	S	2	5.5	3779:15	X
6238	lemarch	20	0	78020	12m	6532	R	1	0.6	1:14.68	konsole
4845	root	20	0	15452	13m	284	S	0	0.7	464:52.59	mandi
6282	lemarch	20	0	46132	9744	3928	S	0	0.5	57:21.46	mdkapplet
8524	lemarch	20	0	2264	1068	812	R	0	0.1	0:00.16	top
1	root	20	0	1732	296	272	S	0	0.0	0:09.96	init
2	root	15	-5	0	0	0	S	0	0.0	0:00.01	kthreadd
3	root	RT	-5	0	0	0	S	0	0.0	0:01.69	migration/0
4	root	15	-5	0	0	0	S	0	0.0	0:00.55	ksoftirqd/0
5	root	RT	-5	0	0	0	S	0	0.0	0:06.94	migration/1



Les processus (4/6)

- Un processus interactif utilise la sortie et l'entrée standard du terminal.
- On peut lancer une commande en tâche de fond grâce au caractère **&** (le terminal n'est pas bloqué) :

```
% textedit monfichier.c &
```

- Le processus peut être amené au premier plan avec la commande **fg** (en arrière plan avec **bg**)

```
% okular image.jpeg
```

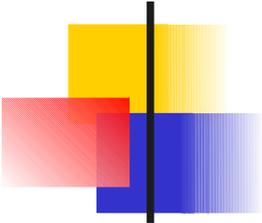
```
^Z
```

```
Suspended
```

```
% bg
```

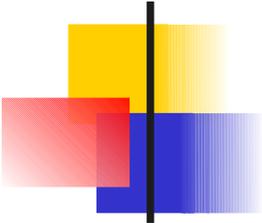
```
[1]      okular images.jpeg &
```

```
%
```



Les processus (5/6)

- Arrêt d'un processus : envoi d'un **signal**
`kill -9 PID`
- `ctrl-C` pour un processus interactif
(kill -15)
- Rappel : Interruption momentanée d'un processus
`ctrl-Z`, typiquement suivi de `bg`
- Par nom : `killall textedit`



Les processus (6/6)

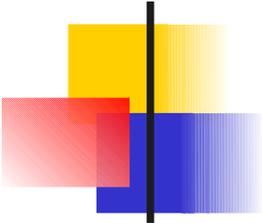
- Exécution périodique automatisée : **cron**
crontab -e

- Format

mm hh jj MMM JJJ tâche > log

- mm représente les minutes (de 0 à 59)
- hh représente l'heure (de 0 à 23)
- jj représente le numéro du jour du mois (de 1 à 31)
- MMM représente le numéro du mois (de 1 à 12)
- JJJ représente l'abréviation du nom du jour ou le chiffre correspondant au jour de la semaine (0, 7 dim, 1 lun, ...)

Caractère *
pour
indifférent



Exemples de crontab

```
# min heu jmois mois jsem tache
```

```
# Lance /bin/false à chaque minute,
```

```
# toute l'année
```

```
* * * * * /bin/false
```

```
# Lance /bin/false du lundi au mercredi et le 4 de chaque  
mois, à 1h35
```

```
35 1 4 * mon-wed /bin/false
```

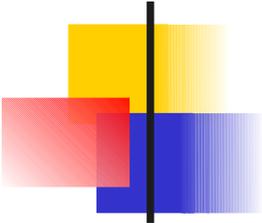
```
# Lance /bin/true le 2 mars à 22h25
```

```
25 22 2 3 * /bin/true
```

```
# Lance /bin/false chaque lundi, mercredi et vendredi, à 2h
```

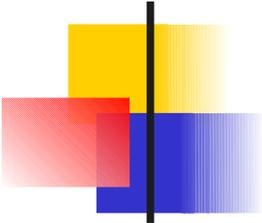
```
0 2 * * 1-5/2 /bin/false
```

Caractère *
pour
indifférent



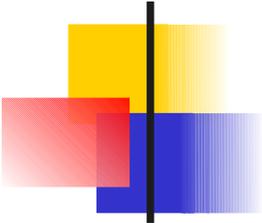
Exercices

- **Q1** Quel est ce processus qui ralentit actuellement votre machine ?
- **Q2** Comment le terminer ?
- **Q3** Vous avait lancé un rapatriement de fichier avec ftp (ou sftp) et il dure plus longtemps que prévu. Comment le mettre en tâche de fond, sans perdre les données déjà rapatriées ?
- **Q4** Votre éditeur de texte ne répond plus (mais vous avez toujours la main dans votre shell). Que faire ?
- **Q5** Votre interface graphique est actuellement complètement gelée. Comment la redémarrer (sans éteindre l'ordinateur) ?
- **Q6** Comment archiver toutes les nuits votre compte ?



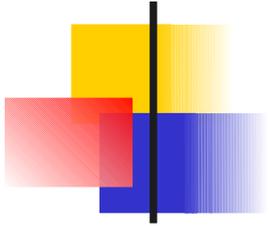
Environnement de travail

- Soit en mode texte (shells) soit en mode graphique
 - Sous Linux, ctrl-alt-F1 – F6 pour une console
 - Ctrl-alt F7 pour le serveur X11/xorg
fenêtres, souris
- Savoir travailler à partir d'une console !
 - Outils graphiques d'administration sans toutes les options, mettent à jour des fichiers texte.
 - Rapidité (ex: *déplacer toutes photos dans le sous-répertoire Toto*)



Connexion sur une machine

- Ouverture de la session de travail
 - Authentification de l'utilisateur:
 - Login (username)
 - Password (mot de passe)
 - Nouvelle connexion :
su lemach
par exemple pour changer d'utilisateur dans un terminal



Connexion à distance sur une machine

- ssh

`ssh machine`

- Login (username)
- Password (mot de passe)

Option `-X` pour rediriger l'affichage

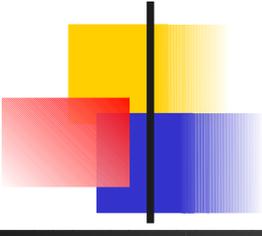
- Authentification directe :

`ssh toto@machine`

- Service ftp associé

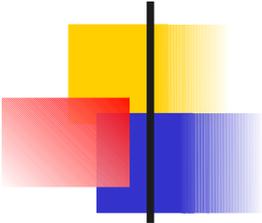
`sftp toto@machine`

Remplace `telnet` (mdp en clair)



Connexion sur une machine distante sans mot de passe

- Générer une paire de clé sur la machine locale
`ssh-keygen -t dsa`
- Copier la clé publique sur la machine distante
`ssh-copy-id -i ~/.ssh/id_dsa.pub user@dist`
 - Si pas de *phrase de passe* : ok mais pas de sécurité locale
 - Si *phrase de passe*
 - Authentification locale avant chaque connexion distante
 - Authentification locale automatique
 - Lancer le mémoriseur de mdp : `eval $(ssh-agent)`
 - Lui confier la *phrase de passe* : `ssh-add`
 - *Phrase de passe simplement une fois par session*

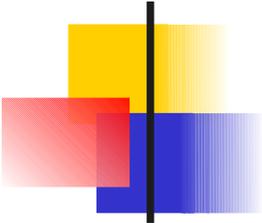


Les mots de passe (1/2)

- Les mauvais mots de passe (dictionnaire)
 - titi
 - albatros
 - rex

- Un bon mot de passe
 - ja!m34r)%

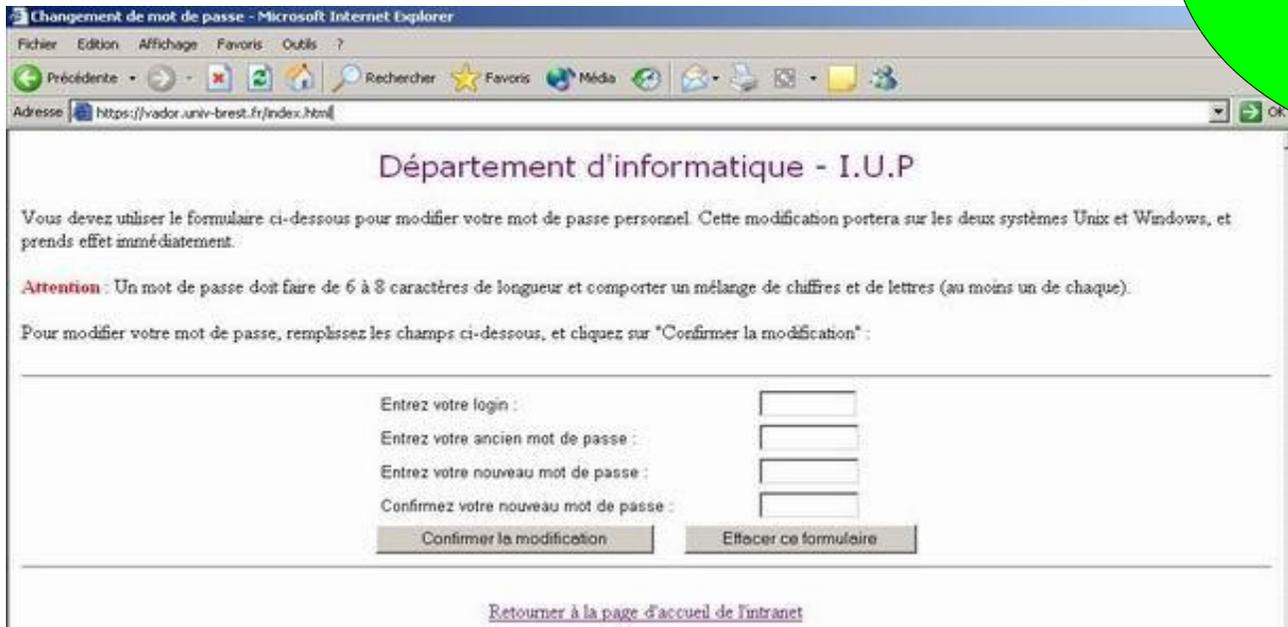
- Changer régulièrement sur votre système commande `passwd`



Les mots de passe (2/2)

- Connexion sur <https://vador.univ-brest.fr/index.html>
- Mot de passe Unix et Windows

F.A.Q
~/liens_util.htm



Changement de mot de passe - Microsoft Internet Explorer

Fichier Edition Affichage Favoris Outils ?

Précédente Recherche Favoris Média

Adresse <https://vador.univ-brest.fr/index.html>

Département d'informatique - I.U.P.

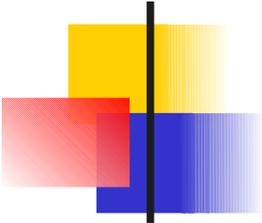
Vous devez utiliser le formulaire ci-dessous pour modifier votre mot de passe personnel. Cette modification portera sur les deux systèmes Unix et Windows, et prends effet immédiatement.

Attention : Un mot de passe doit faire de 6 à 8 caractères de longueur et comporter un mélange de chiffres et de lettres (au moins un de chaque).

Pour modifier votre mot de passe, remplissez les champs ci-dessous, et cliquez sur "Confirmer la modification" :

Entrez votre login :	<input type="text"/>
Entrez votre ancien mot de passe :	<input type="password"/>
Entrez votre nouveau mot de passe :	<input type="password"/>
Confirmez votre nouveau mot de passe :	<input type="password"/>

[Retourner à la page d'accueil de l'intranet](#)



Lancement d'un terminal

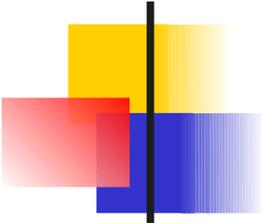
- Pour utiliser une machine, il faut se mettre en relation via un TERMINAL avec le système
 - Terminal physique (ex. vt100)
 - Terminal virtuel (ssh, telnet)
- Tâches au démarrage du terminal :
 - type de terminal
 - lance un interpréteur de commande (shell)
 - définit le clavier comme entrée standard
 - définit l'écran comme sortie standard
 - fichiers .cshrc pour définir des variables d'environnement : PATH, GROUP, TERM ...

Une fenêtre shell

```
Fichier  Édition  Affichage  Historique  Signets  Configuration  Aide
mach1[10]%
Admin      Desktop    GTK        Musique    Systeme
bin        Divers     Images     One-2008.1.svg  tmp
Bureau     Documents  Java       public_html  Vidéos
cmvan.pdf  Downloads  JY         Recherche    Vw
Cours      FREE      Mail       Softs
coursCetAlgoParis.ppt  Germaine  Modèles    Svg
mach1[11]%
mach1[11]% ksnapshot &
[2] 997
mach1[12]% ssh mach2
lemarch@mach2's password:
Last login: Sat Jan 24 10:24:05 2009 from mach1.dyndns.org
mach2.univ-brest.fr[1]% ls
Admin      Images     public_html  Vidéo
AFAIRE     Java       Recherche    Vw
bin        llunix.tar.gz  Softs        VWInstallerLinux86.run
Cours      Mail       stageEmmaus.doc  VWInstallerLinuxX86
Desktop    Modèles    Svg          workspace
Divers     Musique    Systeme
Documents  One-2008.1.svg  Téléchargement
GTK        ordredémissionPloemur.doc  tmp
mach2.univ-brest.fr[2]%
```

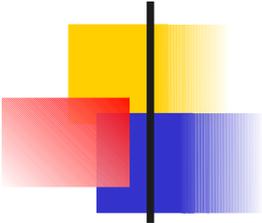
23, 25 Haut

gaspard :



Fichier .cshrc

- Environnement de travail pour csh
- Exécuté au lancement d'un terminal
 - Variables d'environnement
ex: `CLASSPATH`
 - Chemins d'accès
`PATH`
 - Alias
`alias rm 'rm -i'`



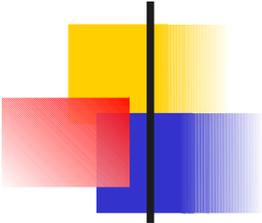
Variables d'environnement

- Exemples

- PATH : chemins vers les exécutable
- TERM : type de terminal (clavier)
- DISPLAY : écran d'affichage

- Gestion avec csh

- `env`
- `setenv PATH $PATH: "monrep"`
- `echo $USER`



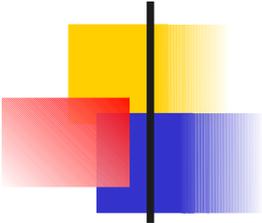
Fichier .cshrc au département

- http://intranet-depiup.univ-brest.fr/faq/csh_login.htm

```
# .cshrc fichier execute' lorsqu'un shell csh ou tcsh est lance' et avant le .login
setenv MANPATH /usr/man:/usr/local/man:/usr/share/man:/usr/dt/man
setenv PATH "/usr/dt/bin:/usr/local/bin:/usr/openwin/bin:/opt/bin:/opt/sfw/bin"
setenv PATH "/usr/local/sbin:/usr/sbin:/sbin:${PATH}:${HOME}/bin:/opt/prolog/kit45:."
setenv PATH "/usr/local/j2sdk1.4.1/jre/bin":${PATH}
# Environnement Eclipse
setenv CLASSPATH
setenv CLASSPATH ${CLASSPATH}:"/usr/local/eclipse/startup.jar"
# Définition des paramètres d'environnement
set host=`hostname`
set prompt= ( `echo $host` "[!]% " )
set history = 25
set ignoreeof noclobber notify nonomatch listpathnum
set filec
set correct=all
```

Définition des alias

```
alias rm 'rm -i'
alias cp 'cp -i'
alias mv 'mv -i'
alias h history
alias ll /bin/lis -lg
```

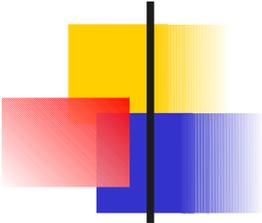


Fichier .login au département

- http://intranet-depiup.univ-brest.fr/faq/csh_login.htm

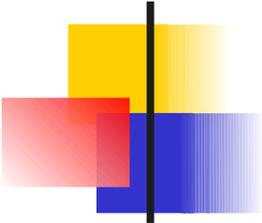
```
# -----  
# Si vous voulez ajouter des instructions particulieres a ce script il vous est vivement conseille de ne PAS MODIFIER le  
# fichier  
# directement mais plutot de travailler sur le script .monlogin qui est à considérer comme votre script personnel, et qui  
# est  
# lancé à la fin de celui-ci, et dans lequel vous pouvez faire ce que vous voulez...  
# Ainsi en cas de probleme, il vous suffira de commenter la dernière ligne pour revenir a l'etat initial.  
#  
# Systeme d'exploitation  
set TYPE_SYSTEME = ( `uname -rs` )  
# Texte de bienvenue  
echo " "  
echo "Bonjour $USER, bienvenue au departement d'informatique - I.U.P"  
date '+Nous sommes le %A %d %B %Y et il est %kH%M.'  
echo "\
```

```
Votre systeme est :  
    $TYPE_SYSTEME\  
Votre terminal est un : $term \  
Votre home-directory est : $cwd\  
"  
cat /home2/applis/motd  
  
source .monlogin
```



Invites de commandes (1/2)

- Il existe différents types de shell (invite) :
 - Bourne shell sh ~ 1975
 - C shell csh
 - Korn shell ksh
 - Bourne again shell bash
- Lancé dans un terminal
- Interaction en mode commande avec le système
ATTENTION SENSIBLE A LA CASSE



Invites de commandes (2/2)

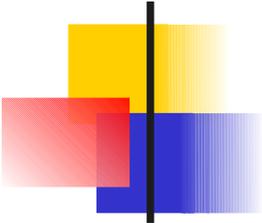
- Le shell présente un prompt et attend des commandes
- Il reprend la main après l'exécution du processus associé à l'exécution de la commande demandée
- Les commandes ont un nom, des arguments et des options

```
ls
```

```
ls -l
```

```
ls -l MonDirectory
```

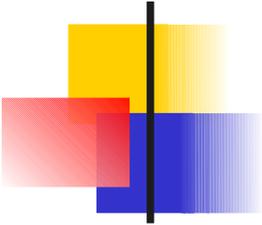
```
ls MonDirectory -la
```



Raccourcis du shell (1/3)

Génération de noms de fichiers

- Avant toute exécution de commande, le shell effectue la substitution des caractères spéciaux suivant par les noms de fichiers qui y correspondent
 - `% ls`
fic.c fic.o fiche a.out
 - `% echo *`
fic.c fic.o fiche a.out
 - `% echo *o`
fic.o
 - `% echo fic.?`
fic.c fic.o
 - `% echo fic.[co]`
fic.c fic.o
 - `% echo /etc/l.d.*`
ld.so.cache ld.so.conf
- `*` suite quelconque (peut être vide)
- `?` un caractère
- `[...]` un des caractères entre crochets



Raccourcis du shell (2/3)

répétition de commandes

- La lettre **!** Permet de répéter des commandes déjà utilisées.

- **history** pour lister les dernières commandes

- **!!** la dernière commande

- **!-2** l'avant-dernière

- **!175** la commande numéro 175 de l'historique

- **!xd** la dernière commande commençant par xd

- **!\$** le dernier argument de la dernière commande

% **history**

165 16:08 vi td1.tex

166 16:09 latex td1

167 16:09 xdvi td1

173 16:36 cw

174 16:38 ls

175 16:38 cd

176 16:38 cd Cours/L1-Unix/

177 16:38 ls

178 16:38 vi plan

179 16:59 ls

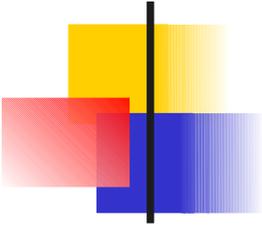
180 17:00 vi plan

181 17:00 cd TD/

182 17:00 ls

183 17:00 vi td1.tex

184 17:12 history



Raccourcis du shell (3/3)

Complétion de nom

- Sur chaque terme de la ligne de commande
 - 1^{er} mot : commande
 - Mots suivants : noms de fichiers
- En appuyant sur <tab> ou ^D

% ls<tab>

ls	lsbininstall	ls-F	lskat	lsnetdrake
lspcidrake	lsattr	lsb_release	lshal	lskatproc
lspci	lspgpot			

% ls to<tab>

total.c torrent.in

% ls tot<tab>

% ls total.c

Chaines de caractères

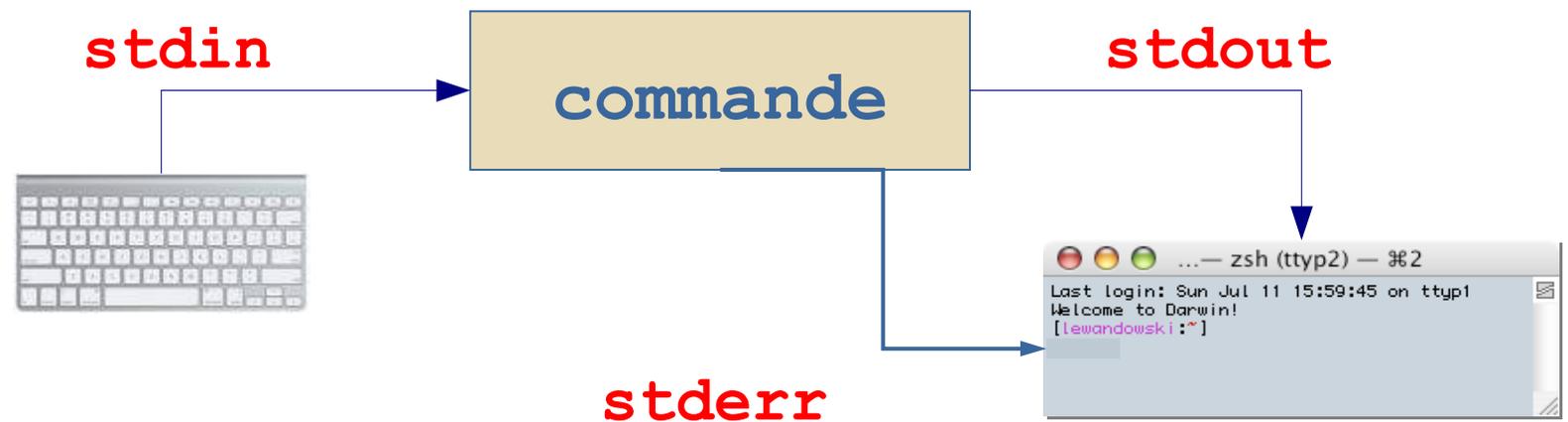
- Interprétation des chaînes de caractères
 - Texte entre `' '` (simples quotes): le texte n'est pas interprété mais considéré comme un mot
 - Texte entre `" "` (doubles quotes): seuls sont interprétés les métacaractères `$`, `\` et ```
 - Texte entre `` `` (anti quotes): considéré comme une commande à interpréter, et c'est le résultat qui sera utilisé.



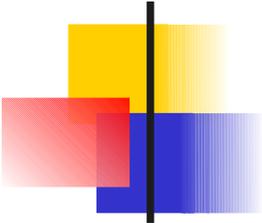
```
...inal — zsh (tty2) — 362
[Lewandow:~] echo 'date'
date
[Lewandow:~]
[Lewandow:~]
[Lewandow:~] echo `date`
Mon Jul 19 17:59:10 CEST 2004
[Lewandow:~]
[Lewandow:~]
[Lewandow:~] echo "date = `date`"
date = Mon Jul 19 17:59:19 CEST 2004
[Lewandow:~]
[Lewandow:~]
[Lewandow:~]
```

Les redirections

- Une commande ouvre 3 descripteurs de fichiers; par défaut:



- Redirections= remplacer les canaux par défaut, rediriger vers une autre commande ou un fichier

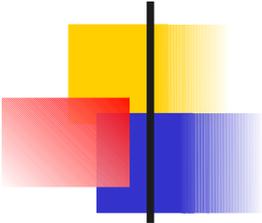


Les redirections (1/2)

<	redirige l'entrée standard
>	redirige la sortie standard
>>	concatène la sortie standard
2>	redirige la sortie d'erreur (sh, bash)
>&	redirige la sortie standard et la sortie d'erreur (csh)

exemples:

<code>% ls . > liste</code>	crée/écrase le fichier liste et y dirige la sortie de `ls`
<code>% date >> liste</code>	ajoute à la fin du fichier liste la sortie de `date`
<code>% wc -l < liste</code>	envoie comme entrée à la commande `wc` le fichier liste



Les redirections (2/2)

Exemples avec sh (suite):

```
% sort < mon_fichier > fichier_trie
```

```
% cat << fin > mon_fichier
```

je tape du texte qui sera sauvegardé dans
mon-fichier

pour terminer, je tape fin

```
%
```

```
% commande 2> fichier_erreur
```

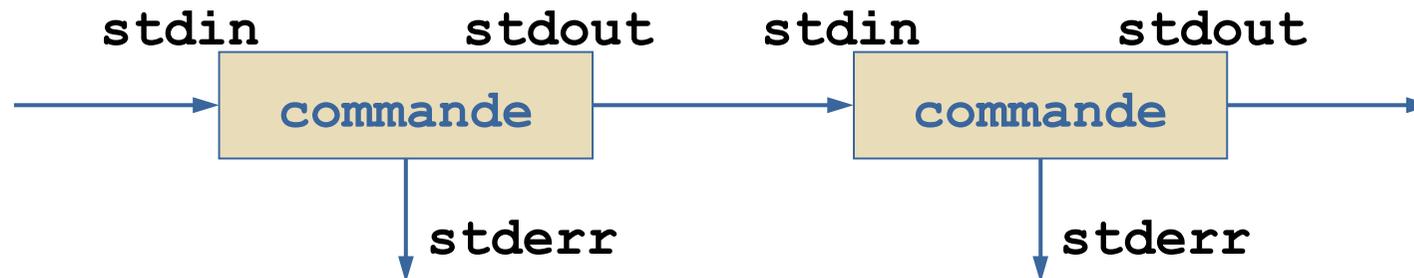
Les erreurs vont dans fichier_erreur

```
% sort mon_fichier > fichier_trie 2>&1
```

Les erreurs vont aussi dans fichier_trie

Les tubes (pipes) (1/2)

- Tube: |
- pour "connecter 2 commandes"



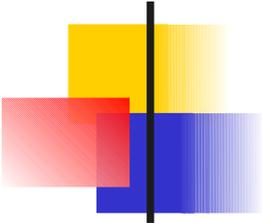
ex: combien de fichiers dans le rep. courant ?

sans pipe:

```
ls > temp ; wc -l < temp ; rm temp
```

avec un pipe:

```
ls | wc -l
```



Les tubes (pipes) (2/2)

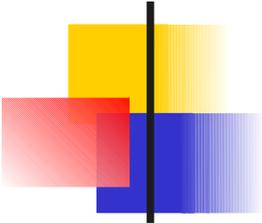
- Simple

```
cat fic | wc -l
```

- Un peu plus dur :

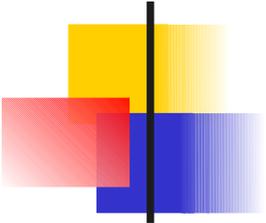
```
uuencode im.jpg | mail lemarch -s "trombi"
```

```
grep IUP listing.txt | grep licence | \  
cut -d: -f1 | sort
```



Exercices

- **Q1** Compter le nombre de sources C dans le répertoire courant
- **Q2** Concatener tous les fichiers commençant par t ou h dans un seul fichier all.txt
- **Q3** Stocker la liste des processus tournant sur la machine dans un fichier
- **Q4** Trier les processus par PID croissant
- **Q5** Que fait l'enchaînement de commandes suivant :
echo Pages MAN > pourTD
man sort >> !\$
^sort^cut^
echo exemple de fichier passwd >> !\$
head -15 /etc/passwd >> !\$
a2ps !\$ | lpr



Les filtres

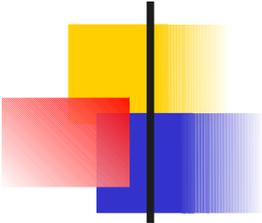
■ Filtres simples :

cat	<ul style="list-style-type: none">– affiche le contenu des fichiers passés en paramètres (par défaut, stdin)– options -b, -n, -v
more	<ul style="list-style-type: none">– affiche page par page les fichiers passés en paramètres (par défaut, stdin)h pour avoir le détail des commandes
tee	<ul style="list-style-type: none">– recopie l'entrée std sur la sortie standard et dans le fichier passé en paramètre– option -a

exemples:

```
cat fic1 fic2
ls | tee liste.fic
```

```
more enormous_file
cat -n toto | more
```



Les filtres

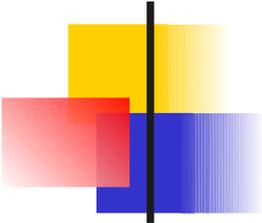
Plus de filtres...

sort

- trie l'entrée ligne par ligne
- options: **-r** (inverse l'ordre de tri)
 +n (ignore les n 1^{ers} champs)
- ex: **ls | sort**
 ls -l | sort +4

comm

- sélectionne les lignes entre deux fichiers
- syntaxe: **comm [-123] fic1 fic2**
 - 1 = supprime les lignes spécifiques à fic1
 - 2 = supprime les lignes spécifiques à fic2
 - 3 = supprime les lignes communes



Les filtres

uniq

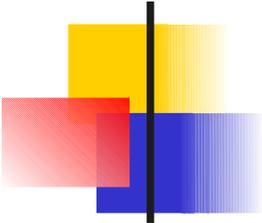
- détruit les lignes consécutives dupliquées
- options: **-u** (affiche les lignes "uniques"),
-d (affiche les lignes "dupliquées")
- ex:

```
uniq -u fic  
uniq -d fic
```

diff

- compare deux fichiers
- options: **-b** (ignorer les lignes vides)
- ex:

```
diff fic1 fic2
```



Les filtres

cut

- sélectionne uniquement certaines colonnes du fichier passé en paramètre
- options:
 - f<liste> : liste des champs à garder
 - c<liste> : liste des colonnes à garder
 - d<char> : séparateur de champs

— ex:

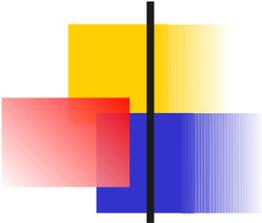
- `cut -c-10 rep.txt`

```
1 tonton 0
2 tux 0077
3 vuja 013
```
- `cut -f1,2 -d" " rep.txt`

```
1 tonton
2 tux
3 vuja
```

rep.txt

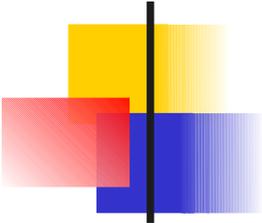
```
1 tonton 0311333300
2 tux 0077885566
3 vuja 0133220011
```



Les filtres

tr

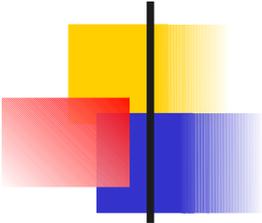
- recopie `stdin` sur `stdout` en substituant des caractères
- syntaxe: `tr [-c ds] [s1 [s2]]`
- options:
 - c (complément de `s1`)
 - d efface les car. de `s1`
 - s tte séquence dans `s1` est substituée par un car. unique dans `s2`
- ex:
 - `tr A-Z a-z < essai`
remplace les majuscules par des minuscules
 - `tr A-Z a-z < essai | tr -sc a-z '\012'`
remplace les majuscules par des minuscules, puis remplace tout ce qui n'est pas une lettre minuscule par un retour chariot (`\012`)



Les filtres

grep

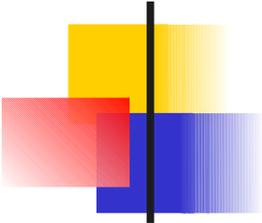
- recherche, dans le fichier passé en paramètre, les lignes vérifiant une expression régulière donnée
- syntaxe : `grep expr_reg [fichier]`
- ex:
 - `grep 'toto' essai`
cherche dans `essai` toutes les lignes qui contiennent le mot `toto`
 - `grep '^[A-Z]' essai`
cherche dans `essai` toutes les lignes qui commencent par une majuscule
- (voir TP sur `grep` et les expressions régulières)



Les filtres

sed

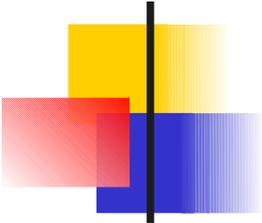
- Stream EDitor
- syntaxe pour effectuer de la substitution en ligne : `sed -e 's/expr/repl/g'`
- ex:
 - `cat fichier | sed -e 's/oo/i/'`
 - `cat fichier | sed -e 's/oo/i/g'`



Les filtres

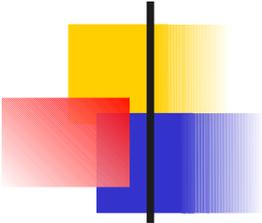
- Et encore plein d'autres...
sed, awk, cmp, ...

- Beaucoup de filtres et commandes...
- Savoir qu'elles existent
- Savoir ce qu'on peut en attendre
- Pour le reste, => **man !!**



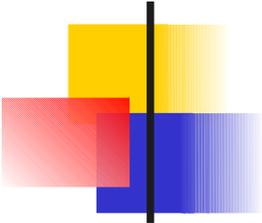
Visual Editor – vi (vi aïe)

- Un éditeur en mode texte, sans formattage wysiwyg
 - Utilisable, même sans environnement graphique
 - Présent sur tous les unix
 - Administration système
- Très riche et très puissant
- Nécessite un apprentissage qui peut rebuter ...
- 2 modes de fonctionnement
 - **Commande** (ce qu'il y a dans les menus d'un éditeur wysiwyg et +)
 - **Insertion** de texte



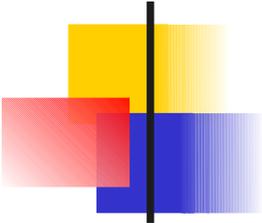
Vi – les indispensables

- Initialement en mode commande
 - **i** pour insérer du texte
 - **Esc** pour repasser en mode commande
- Suppression
 - **x** pour supprimer un caractère
 - **dd** pour supprimer une ligne
- Sauvegarde et sortie
 - **:w** pour sauvegarder
 - **:q** pour sortir (**:q!** Pour sortir sans sauvegarder)
- **u** pour undo



Les expressions régulières (1/2)

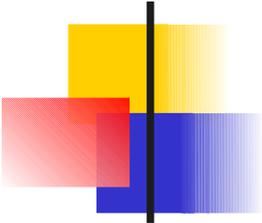
- Définir des motifs
 - ensembles de mots
- Utilisation
 - En analyse lexicale (lex)
 - Filtrage de fichiers (awk)
 - Recherche de motifs (grep, vi)
- Attention, vraies-fausses expressions régulières avec le shell :
 - * : *tous les fichiers* **sauf ceux commençant par** .



Les expressions régulières (2/2)

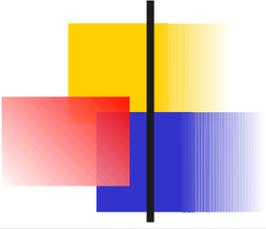
- Composées de
 - Opérandes atomiques
 - Opérateurs

- Chaque expression régulière E définit un motif qui représente un ensemble de chaînes de caractères appelé **langage de E** et noté **$L(E)$**



Opérandes

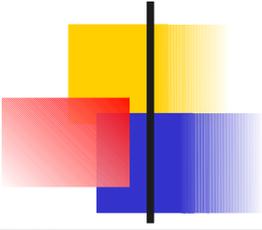
- Un caractère a
 - $L(a) = \{a\}$
- La chaîne vide ϵ
 - $L(\epsilon) = \{\epsilon\}$
- Une variable désignant une expression régulière définie auparavant



Opérateurs (1/3)

Union |

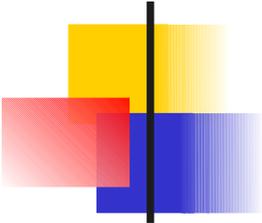
- Union : Si R et S sont 2 E.R, alors R|S représente l'ensemble des langages décrit par R et S : $L(R|S) = L(R) \cup L(S)$
- Exemples
 - $a|b : L(a|b) = \{a, b\}$
 - $(a|b)|c : L((a|b)|c) = \{a, b, c\}$



Opérateurs (2/3)

Concaténation . ou

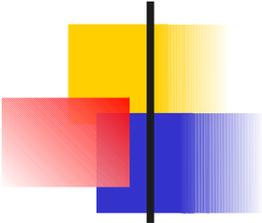
- Union : Si R et S sont 2 E.R, alors RS représente la concaténation des langages décrit par R et S :
 $L(R|S) = L(R)L(S)$
- Exemples
 - $a b : L(ab) = \{ab\}$
 - $E=(a|ab)(c|bc) : L(E) = \{ac, abc, abbc\}$



Opérateurs (3/3)

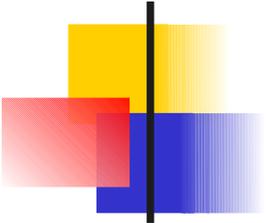
La fermeture (de Kleene) *

- S'applique sur l'opérande E qu'il suit
 - **E*** signifie *0 ou plus occurrences de E*
- Exemples
 - $a^* : L(a^*) = \{\epsilon, a, aa, aaa, \dots\}$
 - $E=(a|b) : L(E^*) = \{\epsilon, a, b, ab, aa, bb, aab, \dots\}$
toutes les combinaisons de a et b



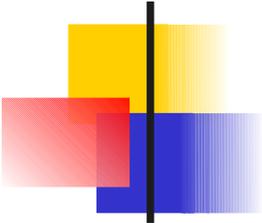
Priorité entre opérateurs

- Du plus prioritaire au moins prioritaire
 - $()$
 - Fermeture a^*
 - Contaténation ab
 - Union $a|b$
- Exemple : $E = a | bc^*d \quad (a | (b(c^*)d))$
 $L(E) = \{a, bcd, bccd, bcccd, bc\dots cd\}$



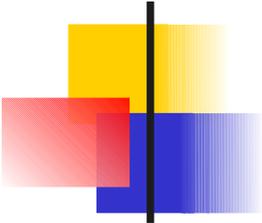
Un exemple : les nombres

- Les entiers naturels
 - entier = $(0|1|\dots|9)(0|1|\dots|9)^*$
- Les entiers relatifs
 - relatif = $(+|-)entier$
 - relatif = $((\sqrt{\quad}|+)|-)entier$
- Les réels ?
- Les identifiants en Pascal ?



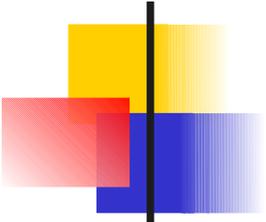
Un exemple : les nombres

- Les entiers naturels
 - chiffre = (0|1|...|9)
 - entier = (chiffre)(chiffre)*
- Les entiers relatifs
 - relatif = (+|-)entier
 - relatif = (($\sqrt{\quad}$ |+)|-)entier
- Les réels
 - relatif($\sqrt{\quad}$ |.entier)
- Les identifiants en Pascal
 - lettre = (a|b|...|z|A|...|Z)
 - ident = (lettre)(lettre|chiffre)*



Exercices

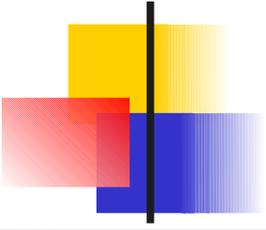
- **Q1** Quelles sont les E.R suivantes
 - les chaînes de 0 et de 1 finissant par un 0
 - les chaînes de 0 et de 1 contenant au moins un 1
 - les chaînes de 0 et de 1 contenant au plus un 1
 - les chaînes de 0 et de 1 telles que la troisième position en partant de la droite soit occupée par un 1
- **Q2** Décrire les langages définis par les E.R suivantes
 - $(a|b)^*$
 - $(a^*b^*)^*$
 - $(a^*ba^*b)^*a^*$



Les Expressions Régulières sous Unix

(1/3)

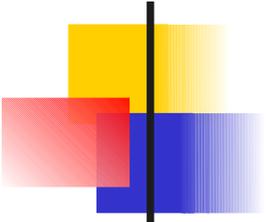
- Les extensions pour `grep`, `egrep`, `vi`, `lex`
- Ensembles de caractères **[]**
 - Pour spécifier un ensemble de caractères
 - **[abcd]** équivaut à `(a|b|c|d)`
 - **[a-zA-Z]** pour des intervalles en fonction du code ASCII des caractères
 - **[^abcd]** pour le complémentaire d'un ensemble



Les Expressions Régulières sous Unix

(2/3)

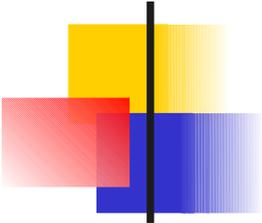
- Début (^) et fin de ligne (\$)
 - **grep '^#include' monFic.c**
 - **grep '^[\t]*\$' | wc -l**
- Le joker (.) (point) remplace tous les caractères possibles sauf le retour à la ligne
 - **.*** toutes les chaînes possibles
- Les caractères spéciaux **[] . * ^ \$** retrouvent leur valeur si ils sont précédés de \
 - **\.[^ \t]*** tous les noms commençant par .



Les Expressions Régulières sous Unix

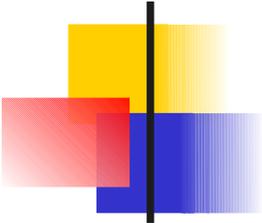
(3/3)

- Classes de caractères prédéfinies **[:classe:]**
 - **[:digit:] [:alpha:] [:alnum:] [:space:]**
 - **[:upper:] [:lower:] ...**
- Le repérage **\(motif \)**
 - Un motif **motif** ainsi encadré est référençable dans le reste de l'expression
 - **\(.\).*\1.*\1** une chaîne contenant 3 fois au moins la même lettre
 - **\(.\) \(.\) .\2\1** ?



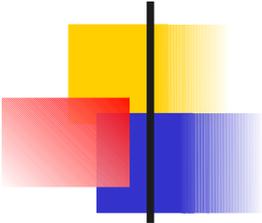
grep/egrep

- Utilise les expressions étendues pour filtrer/rechercher des lignes dans un fichier
 - Attention : les parenthèses n'ont pas de signification particulière
- egrep ajoute **+** (*au moins 1*) et **?** (*0 ou 1*)
grep '[pP]?rintf' *.c
- Quelques options
 - **-c** : nb lignes correspondantes au lieu des lignes elles-mêmes
 - **-i** : pas de différence majuscules/minuscules
 - **-n** : numéros de lignes
 - **-v** : lignes ne correspondant pas
 - **-h** : pas de noms de fichiers



Exercices

- Usages courants de grep
 - **Q1** Compter le nombre de lignes vides dans un fichier
 - **Q2** Comment trouver le PID de l'application mozilla vous appartenant qui ne répond plus dans votre environnement graphique et que vous voulez du coup faire terminer. Donner l'enchaînement de commandes à utiliser.
 - **Q3** Lister tous les fichiers *dangereux* de votre arborescence personnelle, avec les droits en écriture pour le reste du monde
 - **Q4** Quelle est la valeur de la constante symbolique `UINT_MAX` définie dans un des fichiers d'entête système du langage C ?
 - **Q5** Comment extraire les lignes concernant le préprocesseur ?



Exercices

- Usages courants de grep
 - **Q1** Comment extraire l'IUD de l'utilisateur lemarch du fichier /etc/passwd ? Le nom de l'utilisateur 2345 ?

```
lemarch:x:1234:3000:Laurent Lemarchand:/home/lemarch:/bin/tcsh
```

Exercices

- Exploiter des informations Internet. Récupérer le vent moyen le 27 Décembre 1999.

Résumé de la journée du 27 Décembre 1999 à Brest - Guipavas

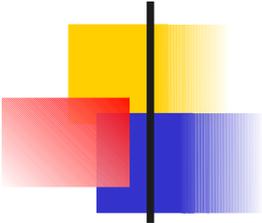
[<< Jour précédent](#) [Jour suivant >>](#)

Temp. min		Temp. max		Temp. moy		Pluie	Vent moyen	Rafale max	Ensoleil.	
1.6°C	-3.2	7.2°C	-2.7	4.4°C	-3	26 mm	→	26.8 km/h	107.3 km/h	0 h

Normales calculées entre 1981 et 2010

wget 'http://meteo-bretagne.fr/archive-observation.php?icao=07110&d=27&m=12&y=1999&metar='

```
Source de : http://meteo-bretagne.fr/archive-observation.php?icao=07110&d=27&m=12&y=1999&metar= - SeaMonkey
Fichier Édition Affichage Outils Fenêtre ?
<a href="javascript:void(0)" class="tooltip"><a href="javascript:void(0)" class="tooltip"><a href="javascript:void(0)" class="tooltip"><a href="javascript:void(0)" class="tooltip">Pluie<
 <a href="javascript:void(0)" class="tooltip"><a href="javascript:void(0)" class="tooltip">Rafale  <a href="javascript:void(0)" class="tooltip">Ensoleil. </tr> | | | |
```



Controle continu

- Exploiter des informations Internet. Récupérer les rafales sur une période de jours : exemple : 5 jours à partir du 27 décembre 1999. Produire un graphique avec gnuplot (Linux) ou Exel (Windows)
- Renvoyer votre script par mail à laurent.lemarchand@univ-brest.fr pour le 11 mars 2013 avec 'CC Unix' en sujet et dans le message votre nom et numéro étudiant (binome possible), le script et une trace d'exécution (données et graphique)

```
wget 'http://meteo-bretagne.fr/archive-observation.php?icao=07110&d=27&m=12&y=1999&metar='
```

Résumé de la journée du 27 Décembre 1999 à Brest - Guipavas

<< Jour précédent

Jour suivant >>

Temp. min		Temp. max		Temp. moy		Pluie	Vent moyen	Rafale max	Ensoleil.
1.6°C	-3.2	7.2°C	-2.7	4.4°C	-3	26 mm	→ 26.8 km/h	107.3 km/h	0 h

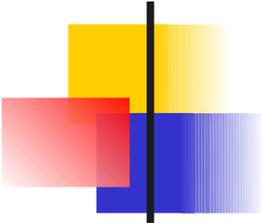
Normales calculées entre 1981 et 2010

Exercices

- Trouver et filtrer la ligne intéressante
- En extraire la donnée

```
Source de : http://meteo-bretagne.fr/archive-observation.php?icao=07110&d=27&m=12&y=1999&metar= - SeaMonkey
Fichier Édition Affichage Outils Fenêtre ?
tr>
td class="forumline" style="padding:0;font-weight:bold;height:40px;" colspan="2"><a href="javascript:void(0)" class="toolt
td class="forumline" style="padding:0;font-weight:bold;height:40px;" colspan="2"><a href="javascript:void(0)" class="toolt
td class="forumline" style="padding:0;font-weight:bold;height:40px;" colspan="2"><a href="javascript:void(0)" class="toolt
td class="forumline" style="padding:0;font-weight:bold;height:40px;"><a href="javascript:void(0)" class="tooltipT">Pluie<
td class="forumline" style="padding:0;font-weight:bold;height:40px;" colspan="2"><a href="javascript:void(0)" class="toolt
td class="forumline" style="padding:0;font-weight:bold;height:40px;"><a href="javascript:void(0)" class="tooltipT">Rafale
td class="forumline" style="padding:0;font-weight:bold;height:40px;"><a href="javascript:void(0)" class="tooltipT">Ensolei
/tr>
tr>
td class="forumline_metar" width="13%">1.6°C</td><td class="forumline_metar" style="background:RGB(0,156,255)">-3.2</td>
td class="forumline_metar" width="13%">7.2°C</td><td class="forumline_metar" style="background:RGB(0,156,255)">-2.7</td>
td class="forumline_metar" width="13%">4.4°C</td><td class="forumline_metar" style="background:RGB(0,156,255)">-3</td>
td class="forumline_metar" width="13%">26 mm</td>
td class="forumline_metar"></td>
td class="forumline_metar" width="13%">26.8 km/h</td>
td class="forumline_metar" width="13%">107.3 km/h</td>
td class="forumline_metar" width="13%">0 h</td>
/tr> </table>
div style="padding:0;margin:0;font-size:10px;">Normales calculées entre 1981 et 2010<br/>
elon les normes officielles de l'OMM :<br/>
```

- Généraliser et en faire un script



Exercices

- Déroulement

- `grep 'km/h' meteo.data`

```
<td class="forumline_metar" width="13%">26.8 km/h</td>
```

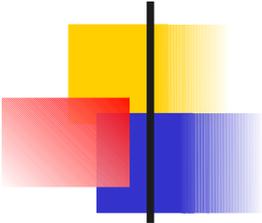
```
<td class="forumline_metar" width="13%">107.3 km/h</td>
```

```
<td class="forumline_metar" style="background:RGB(94,168,247)">14.8 km/h </td>
```

```
<td class="forumline_metar" style="background:RGB(90,128,255)">3.7 km/h </td>
```

```
<td class="forumline_metar" style="background:RGB(90,128,255)">7.4 km/h </td>
```

```
...
```



Exercices

- Déroulement

- `grep 'km/h' meteo.data`

```
<td class="forumline_metar" width="13%">26.8 km/h</td>
```

```
<td class="forumline_metar" width="13%">107.3 km/h</td>
```

```
<td class="forumline_metar" style="background:RGB(94,168,247)">14.8 km/h </td>
```

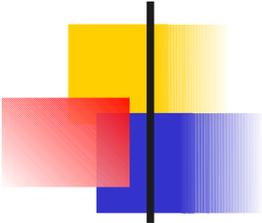
```
<td class="forumline_metar" style="background:RGB(90,128,255)">3.7 km/h </td>
```

```
<td class="forumline_metar" style="background:RGB(90,128,255)">7.4 km/h </td>
```

...

- `grep 'km/h' meteo.data | head -1`

```
<td class="forumline_metar" width="13%">26.8 km/h</td>
```



Exercices

■ Déroulement

- `grep 'km/h' meteo.data`

```
<td class="forumline_metar" width="13%">26.8 km/h</td>
```

```
<td class="forumline_metar" width="13%">107.3 km/h</td>
```

```
<td class="forumline_metar" style="background:RGB(94,168,247)">14.8 km/h </td>
```

```
<td class="forumline_metar" style="background:RGB(90,128,255)">3.7 km/h </td>
```

```
<td class="forumline_metar" style="background:RGB(90,128,255)">7.4 km/h </td>
```

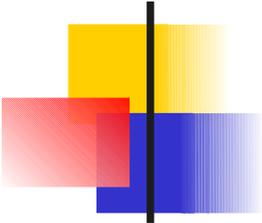
...

- `grep 'km/h' meteo.data | head -1`

```
<td class="forumline_metar" width="13%">26.8 km/h</td>
```

- `grep 'km/h' meteo.data | head -1 | cut -d' ' -f3`

```
width="13%">26.8
```



Exercices

■ Déroulement

- `grep 'km/h' meteo.data`

```
<td class="forumline_metar" width="13%">26.8 km/h</td>
```

```
<td class="forumline_metar" width="13%">107.3 km/h</td>
```

```
<td class="forumline_metar" style="background:RGB(94,168,247)">14.8 km/h </td>
```

```
<td class="forumline_metar" style="background:RGB(90,128,255)">3.7 km/h </td>
```

```
<td class="forumline_metar" style="background:RGB(90,128,255)">7.4 km/h </td>
```

...

- `grep 'km/h' meteo.data | head -1`

```
<td class="forumline_metar" width="13%">26.8 km/h</td>
```

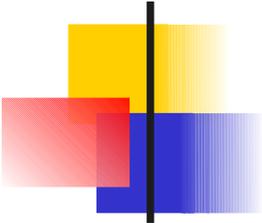
- `grep 'km/h' meteo.data | head -1 | cut -d' ' -f3`

```
width="13%">26.8
```

- `grep 'km/h' meteo.data | head -1 | cut -d' ' -f3 | cut -d'>' -f2`

```
26.8
```

■ Comment avoir la rafale maximale ?

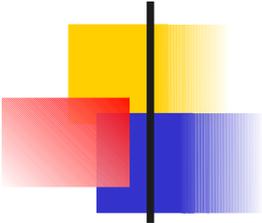


Exercices

- Une solution complète

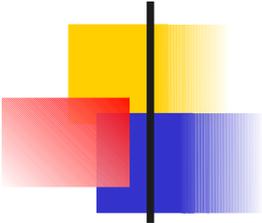
```
[lemarch]% ./getVentMoyen.csh 27 12 1999  
26.8
```

```
#!/bin/csh  
# getVentMoyen jour mois annee  
  
mkdir TEMPMETEO  
cd TEMPMETEO  
  
set url = 'http:// ... &d='$1'&m='$2'&y='$3'&metar=#dataTab'  
# echo recuperation des donnees a: "$url"  
wget "$url" -o m.trace  
mv archive* meteo.data  
grep 'km/h' meteo.data | head -1 | cut -d' ' -f3 | cut -d'>' -f2  
  
cd ..  
rm -r TEMPMETEO
```



Le langage C-shell

- Langage de programmation
 - Des instructions : les commandes (ls, mv, cd, echo, wc, man, ...)
 - Des variables : déclaration, initialisation, modification ...
 - Des structures de contrôle pour le déroulement du programme : **;** **|** **if repeat foreach while switch**



Utilisation de variables (1/2)

- Déclaration locale au shell

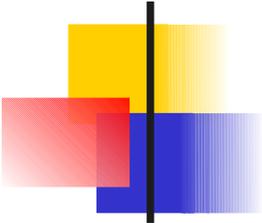
set a = titi

- Visible dans l'environnement des shells ou commandes exécutées ensuite

setenv b titi

```
# ex.sh  
  
echo $v1  
echo $v2
```

```
unix[45]% setenv v1 TRUC  
unix[46]% set v2 = TROC  
unix[47]% csH ex.sh  
TRUC  
v2 undefined variable  
unix[48]%
```



Utilisation de variables (2/2)

- **Calcul** sur des variables numériques

set nb = 5

déclaration

@ nb = \$nb + 5

utilisation

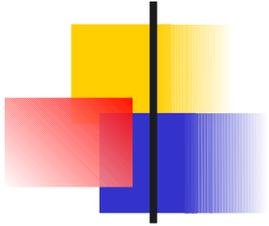
Attention
aux blancs !

- Tous les opérateurs classiques sont possibles

+ - * / ++ - *=

set sum = 0

@ sum = \$sum + \$nb - 8 * 3



Tableaux

```
unix[34]% set tab = ( toto 25 titi )
```

```
unix[35]% echo $tab[2]
```

```
25
```

```
unix[36]%
```

```
unix[37]% set ladata = ( `date` )
```

```
unix[38]% echo $ladata
```

```
Tue Mar 14 10:33:00 MET 2009
```

```
unix[39]% @ ladata[3]++
```

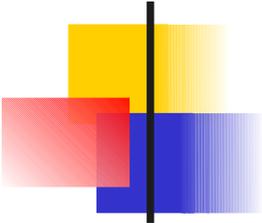
```
unix[40]% !e
```

```
Tue Mar 15 10:33:00 MET 2009
```

```
unix[41]% echo $#ladata
```

```
6
```

Taille d'un tableau



Variables spécifiques

- Pour la saisie d'un mot : **\$<**

```
unix[10]% set lu = $<
```

```
6
```

```
unix[11]% echo $lu + 1
```

```
7
```

- Arguments de la ligne de commande : **\$0 \$1 \$2 ...**

```
unix[11]% somme.sh 11 20
```

```
31
```

- Test de l'existence d'une variable : **\$?var**

```
unix[13]% set var = toto
```

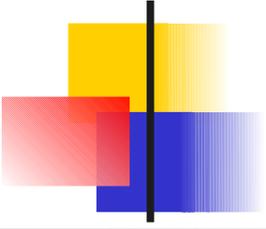
```
unix[14]% echo $?var
```

```
1
```

```
#!/bin/csh  
# somme.sh
```

```
set v = 0  
@ v = $1 + $2  
echo $v
```

```
$0 $1 $2 ...  
$argv[]
```



Structure de contrôle conditionnelle (1/2)

If (condition) then
instructions

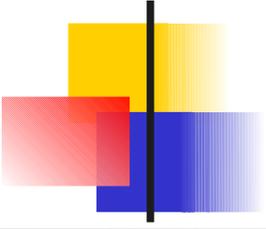
else *optionnel*
instructions

endif

- Opérateurs de condition
 - ! && || < > == != ...
 - tests sur fichier
- Maximum des 2 paramètres d'une fonction

```
#!/bin/csh
# max.sh

If ( $1 > $2 ) then
    set m = $1
else
    set m = $2
endif
echo maxi = $m
```



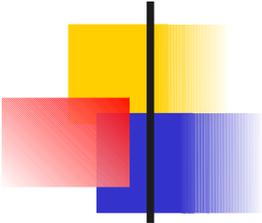
Structure de contrôle conditionnelle (2/2)

- Tests sur les fichiers
 - **-d** répertoire
 - **-e** existe
 - **-f** ordinaire
 - **-r, -w, -x** droits
 - **-z** taille nulle

if (-spec fichier) then

```
#!/bin/csh
# type.sh

If ( -e $1 ) then
    If ( -d $1 ) then
        echo repertoire
    else
        echo fichier
    endif
else
    echo inexistant
endif
```

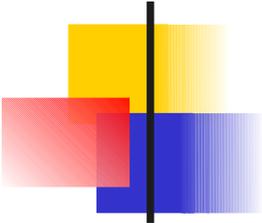


Boucle foreach (1/2)

```
foreach var ( liste de mots )  
    instructions  
end
```

- **\$var** vaut successivement chacun des mots de la liste

```
#!/bin/csh  
# bcl.sh  
  
foreach fic ( toto titi tata )  
    if ( -e $fic ) then  
        wc -l $fic  
    endif  
end
```



Boucle foreach (2/2)

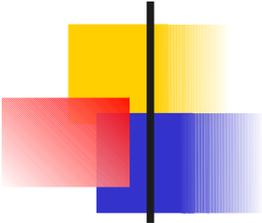
- Motifs pour la liste de mots
 - **\$*** les arguments du script (\$1 \$2 \$3 ...)
 - ***** ou une autre expression régulière du shell
la liste des fichiers correspondant

```
#!/bin/csh
# bcl2.sh

foreach fic ( $* )
  if ( -e $fic ) then
    wc -l $fic
  endif
end
```

```
#!/bin/csh
# bcl3.sh

set lignes = 0
foreach fic ( *.ch )
  @ lignes += `cat $fic | wc -l`
end
echo $lignes
```



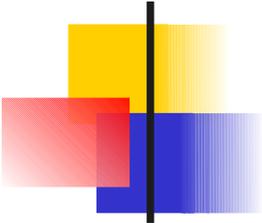
Boucle while

```
while ( condition )  
    instructions  
end
```

- Sert également pour les boucles **for** (<> foreach)

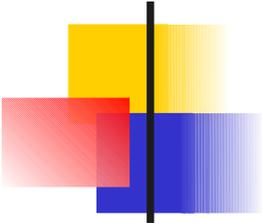
```
#!/bin/csh  
# while.sh  
  
while ( $< != 10 )  
    echo entrez un autre nombre  
end
```

```
#!/bin/csh  
# while4for.sh  
  
set total = 0  
set i = 0  
while ( $i < 10 )  
    @ total += $i  
    @ i++  
end  
echo $total
```



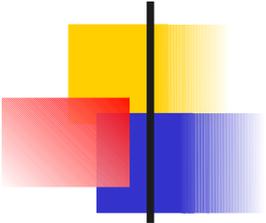
Exercices

- **Q1** Que fait la commande `head -10 f1 | tail -1`
(f1 est un fichier texte)
 - Construire un programme C-shell permettant d'afficher à l'écran la i^{eme} ligne d'un fichier f passé en paramètre (ligne i f).
- **Q2** Calcul de la taille globale d'une liste de fichiers passés en argument au script
 - penser à utiliser un tableau pour décomposer le résultat du `ls -l`
- **Q3** Modifier **Q2** pour afficher tous les fichiers dont la taille dépasse un seuil donné en argument et en Mo



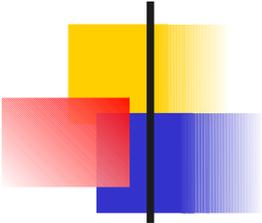
Exercices

- Calculer l'espace disque utilisé par chaque utilisateur (les homedir sont dans le répertoire courant)
 - **Q1** Quel est l'espace total utilisé ?
 - **Q2** Quel est l'utilisateur consommant le plus d'espace ?
- **Q3** Attribuer un numéro à chaque étudiant d'une liste
 - utiliser cat dans le foreach
- **Q4** Lire une suite de nombres tant que leur somme ne dépasse pas 100
 - Utiliser $\$<$ pour la lecture et une structure



Exercices

- **Q1** Calculer le nombre de fichiers et répertoires d'une arborescence
 - récursivité

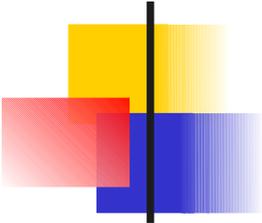


Exercices

- **Q1** Calculer le nombre de fichiers et répertoires d'une arborescence
 - récursivité

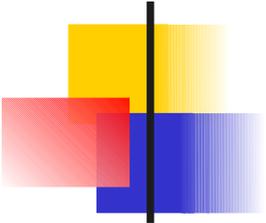
```
#!/bin/csh
# ~/compteFic.sh

set nb = 0
foreach fic ( * )
    if ( -d $fic ) then
        cd $fic
        @ nb += `~/compteFic.sh`
        cd ..
    else
        @ nb++
    endif
end
echo $nb
```



Awk

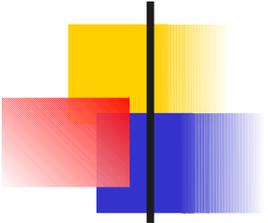
- Aho, Weinberger, Kernighan
- Ecriture de **moulinettes** : transformation de données en quelques lignes de code
 - grep intelligent, avec
 - des possibilités logiques et numériques
 - Manipulations tabulaires (lignes et colonnes)
- Syntaxe C
 - Le K de Awk est le K de K&R



Lancement d'awk/gawk

- 2 manières simples
 - En ligne de commande
 - **cat file | gawk '(pattern){action}'**
 - **cat file | gawk -f program.awk**
 - Dans un script awk

```
#!/bin/awk -f  
# commentaire  
  
(pattern) {action}  
...
```

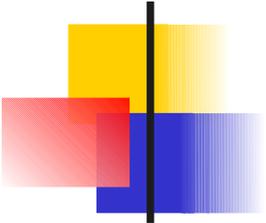


Programmes awk

- Un programme est une suite de **règles**
- Les règles sont appliquées séquentiellement à chaque **enregistrement** du flux ou fichier d'entrée
 - Par défaut chaque ligne est un enregistrement
- Les règles sont en 2 partie : un **motif** et une **action**
- Si l'entrée respecte le motif, alors l'action est exécutée

```
(pattern1) { action }  
(pattern2) { action }
```

...



Programmes awk

tag.awk

```
#!/bin/awk -f
/soleil/ { print "$0 tag soleil"; }
/vin/    { print "$0 tag vin"; }
```

mots.txt

```
Bordeaux: Garonne vin ville
Hugo: Victor écrivain
Tomate: rouge soleil fruit mure
Provence: soleil vin Pagnol
Disque: vinyle chanteur CD
....
```

- Exemple

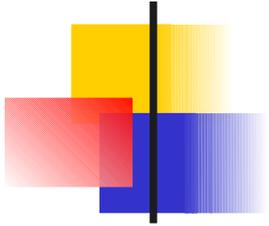
```
% cat mots.txt | tag.awk
```

```
Tomate: tag soleil
```

```
Bordeaux: tag vin
```

```
Provence: tag soleil
```

```
Provence: tag vin
```



Exemple

entrée

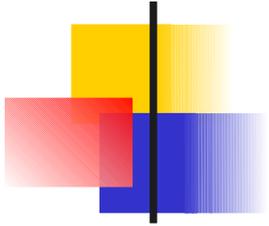
```
PING dt033n32.san.rr.com (24.30.138.50): 56 data bytes
64 bytes from 24.30.138.50: icmp_seq=0 ttl=48 time=49 ms
64 bytes from 24.30.138.50: icmp_seq=1 ttl=48 time=94 ms
64 bytes from 24.30.138.50: icmp_seq=2 ttl=48 time=50 ms
64 bytes from 24.30.138.50: icmp_seq=3 ttl=48 time=41 ms
...
----dt033n32.san.rr.com PING Statistics----
1281 packets transmitted, 1270 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 37/73/495 ms
```

Programme

```
(/icmp_seq/) {print $0}
```

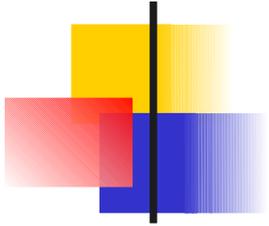
sortie

```
64 bytes from 24.30.138.50: icmp_seq=0 ttl=48 time=49 ms
64 bytes from 24.30.138.50: icmp_seq=1 ttl=48 time=94 ms
64 bytes from 24.30.138.50: icmp_seq=2 ttl=48 time=50 ms
64 bytes from 24.30.138.50: icmp_seq=3 ttl=48 time=41 ms
```



Champs

- Awk divise le fichiers en **enregistrements** et en **champs**
 - Par défaut, chaque ligne est un enregistrement
 - Les champs sont délimités par un caractère spécifique (Blanc ' ' par défaut)
- Le séparateur de champs peut être modifié
 - en ligne de commande avec l'option **-F**
awk -F: -f sc.awk < fichier
 - dans un script en modifiant la variable **FS**
{ FS = ":" }



Champs (2/2)

- L'accès aux champs se fait avec **\$**
 - **\$1** est le premier champ, **\$2** le second ...
 - **\$0** est la ligne entière
 - **NF** est une variable contenant le nombre de champs de l'enregistrement courant (**NR** : enregistrement courant)

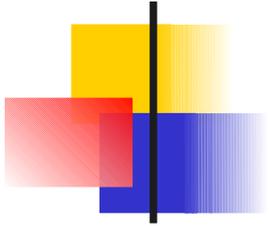
entrée

```
PING dt033n32.san.rr.com (24.30.138.50): 56 data bytes
64 bytes from 24.30.138.50: icmp_seq=0 ttl=48 time=49 ms
64 bytes from 24.30.138.50: icmp_seq=1 ttl=48 time=94 ms
```

Programme `(/icmp_seq/) {print $7}`

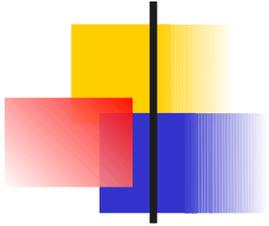
sortie

```
time=49 ms
time=94 ms
```



Variables

- Simples à utiliser
 - Pas besoin de les déclarer
 - Initialisation à **0** ET à *chaine vide*
- Un seul type de variable en awk
 - À la fois nombre flottant et chaine de caractères
 - Conversion automatique suivant les besoins
- Peut importe ce que contient x, on peut toujours faire
 - **$x = x + 1$**
 - **`length(x)`**



Exemple

entrée

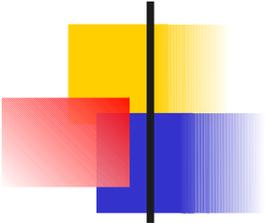
```
PING dt033n32.san.rr.com (24.30.138.50): 56 data bytes
64 bytes from 24.30.138.50: icmp_seq=0 ttl=48 time=49 ms
64 bytes from 24.30.138.50: icmp_seq=1 ttl=48 time=94 ms
64 bytes from 24.30.138.50: icmp_seq=2 ttl=48 time=50 ms
64 bytes from 24.30.138.50: icmp_seq=3 ttl=48 time=41 ms
...
```

programme

```
(/icmp_seq/) {
    n = substr($7,6);
    printf( "%s\n", n/10 );    #conversion
}
```

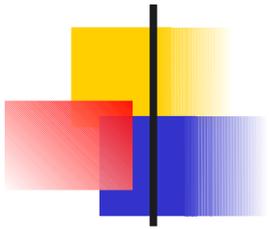
sortie

```
4.9
9.4
5.0
4.1
...
```



Motifs (*Patterns*)

- Le motif associé à une règle peut être :
 - Vide: l'action est exécutée pour toute ligne
{print \$0} imprime chaque ligne
 - Une expression régulière
(/expression/) ou **(\$4~/expression/)**
 - Une expression booléenne
(\$2=="truc" && \$7=="machin")
 - Un ensemble
(\$2=="on" , \$3=="off")
 - 2 motifs spéciaux: **BEGIN** et **END**



Exercices - Que font ces programmes ?

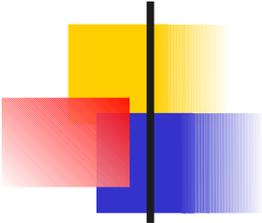
```
NF = 2
```

```
{print $2}
```

```
length($0) > 80
```

```
END { print NR, "lignes"; }
```

```
/mkr/ { n++; }  
END { print n, "lignes"; }
```



Que font ces programmes ?

```
NF = 2
```

Affiche toutes les lignes avec 2 champs

```
{print $2}
```

Affiche le 2ème champ de chaque ligne (cut -f 2)

```
length($0) > 80
```

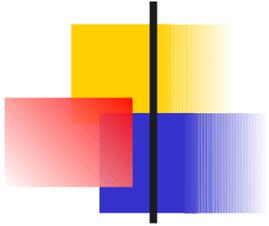
Lignes de plus de 80 caractères

```
END { print NR, "lignes"; }
```

Nombre de lignes du fichier

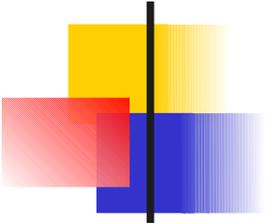
```
/mkr/ { n++; }  
END { print n, "lignes"; }
```

Nombre de lignes du fichier contenant mkr



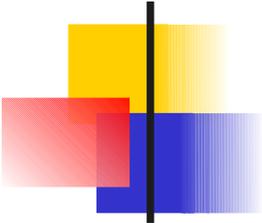
Tableaux

- Tous les tableaux awk sont associatifs
 - **A[1] = "truc";**
 - **B["repas lundi"] = "pizza";**
- Pour vérifier si un élément est dans un tableau
 - mot-clé **in** : **If ("repas lundi" in B) ...**
- Les tableaux sont alloués et initialisés automatiquement. Leur taille est aussi auto-ajustée
- **in** sert également pour itérer sur les éléments :
 - **for (x in myarray) {**



Tableaux associatifs

- Les tableaux awk peuvent servir à implanter toutes les structures de données courantes
 - Ensembles :
 - **myset["a"]=1; myset["b"]=1;**
 - **If ("b" in myset)**
 - Tableaux multi-dimensionnels:
 - **tab2D[1,3] = 2; tab2D[1,"happy"] = 3;**
 - Listes :
 - **maliste[1,"data"] = 2; maliste[1,"next"] = 3;**



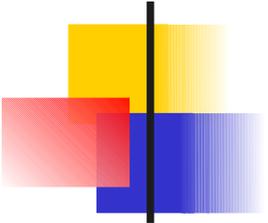
Un exemple utilisant les tableaux associatifs : fréquence de mots

```
#!/usr/bin/awk -f
```

```
BEGIN { FS="[ ^A-Za-z0-9_]*" }           # séparateur de champs  
                                             # retirant la ponctuation
```

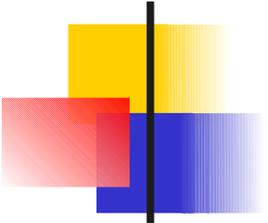
```
{ $0 = tolower($0);                       # tout en minuscules  
  for (i = 1; i <= NF; i++)                # integre les champs sauf  
    if ($i != "")                          # s'ils sont vides  
      tab_freq[$i]++                       # dans le tableau associatif tab_freq  
}
```

```
END {                                     # parcours le tableautab_freq  
  for (mot in tab_freq)  
    printf "%s\t%d\n", mot, tab_freq[mot]  
}
```



Exercices

- **Q1** Afficher les lignes contenant `srb`
- **Q2** Afficher le 3eme champ de chaque ligne
- **Q3** Dans le fichier `/etc/passwd`, extraire les login des utilisateurs dont l'`IUD` est < 100
- **Q4** Nombre de lignes d'un fichier
- **Q5** sommes des nombres stockés dans un fichier et séparés par des blancs ou des virgules

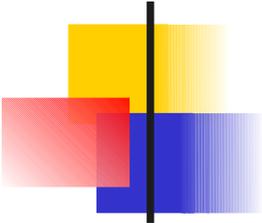


Exercices

- A partir des informations de `/etc/passwd`
 - **Q1** Nombre d'utilisateurs de `bash` et `csh`
 - **Q2** Pourcentage d'utilisateurs des différents shells (pensez aux tableaux associatifs)

Les comptes système ont normalement un UID inférieur à 1000 et ne permettent pas de se loguer (`/bin/false` ou `/bin/nologin`).

- **Q3** Afficher le login et le numéro de ligne des comptes en anomalie.
- **Q4** En donner également le nombre.



En plus ...

- Liens physiques

`ln <nom_fic> <nouveau_nom_fic>`

- permet de donner plusieurs noms à un fichier
- pas pour les répertoires
- ne traverse pas les partitions
- un fic est détruit quand TOUS ses liens physiques sont supprimés (≠ raccourcis)

- Liens symboliques

`ln -s <nom_fic> <nouveau_nom_fic>`

- crée un **raccourci**
- traverse les partitions
- fonctionne aussi pour les répertoires

- Lister les liens d'un fichier: `ls -l <nom_fic>`