L'informatique au service de la physique

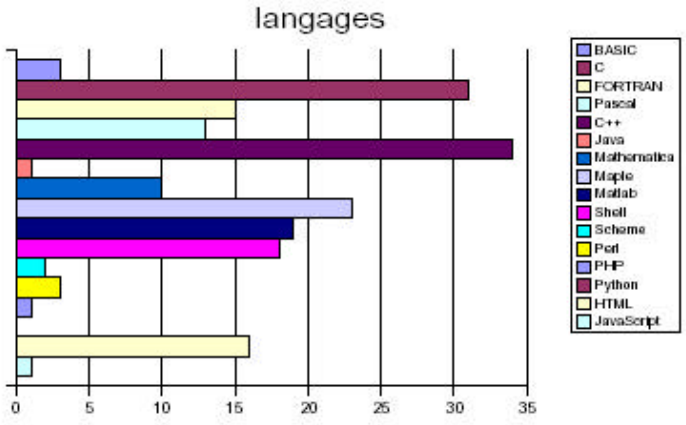
Introduction :

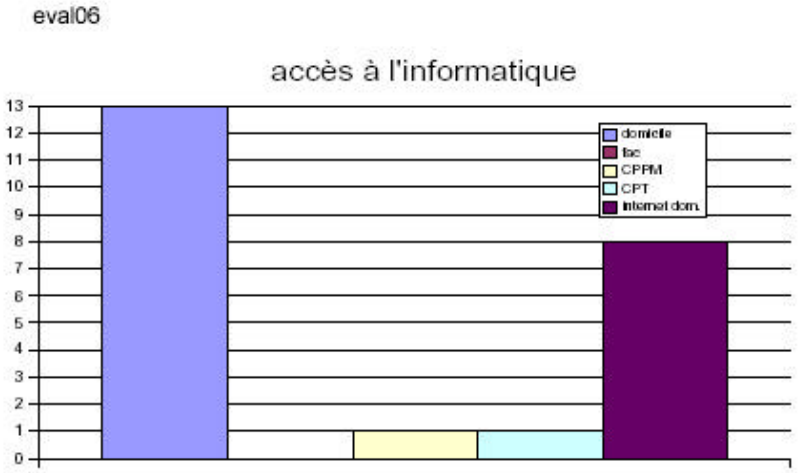
Utilisation d'un ordinateur sous Unix

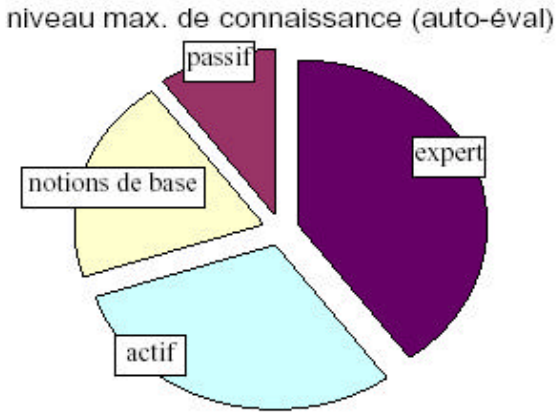
Plan du cours

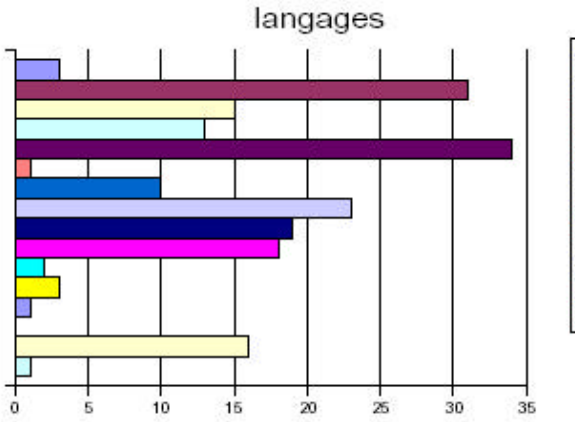
- **Partie Unix (cours 1-3)**
  - Session d'utilisateur et système de fichiers
  - Shell, commandes et applications ; editeurs
  - Internet ; WWW, mail / Organisation de programmes
- **Partie Programmation (cours 4-8)**
  - C / C++
  - Applications C++ : root, FeynDiag
  - Java + applications

Résultats du sondage

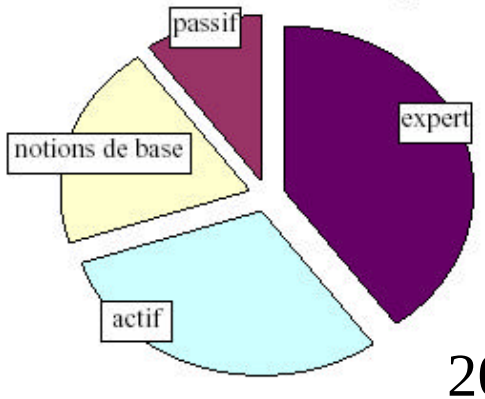




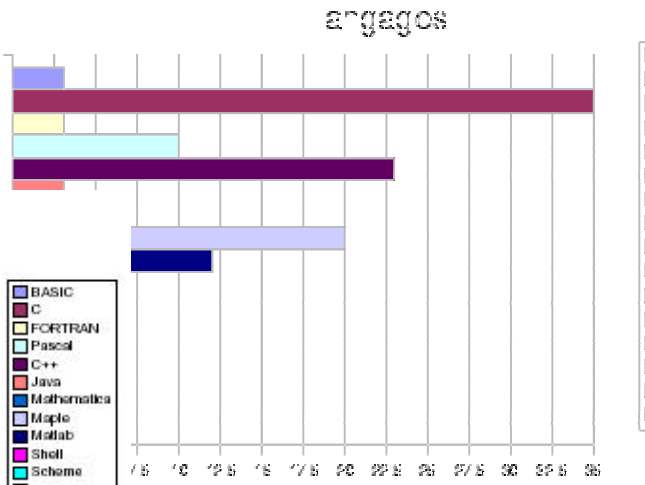


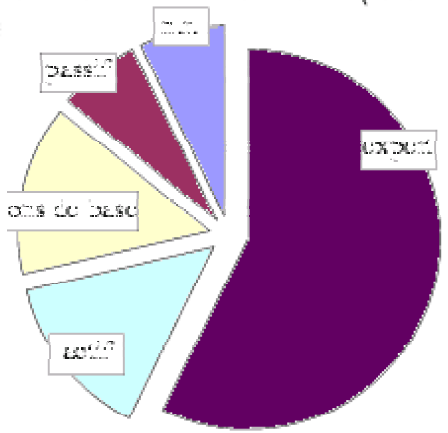


niveau max. de connaissance (auto-éval)

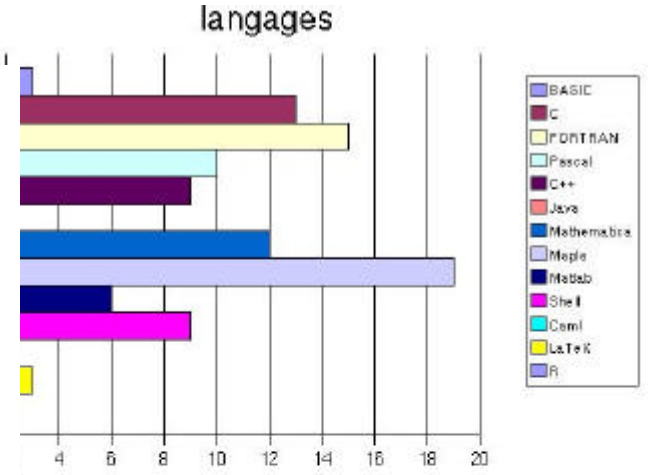


2006

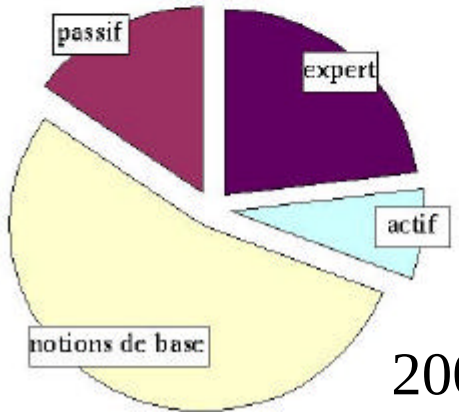




2005

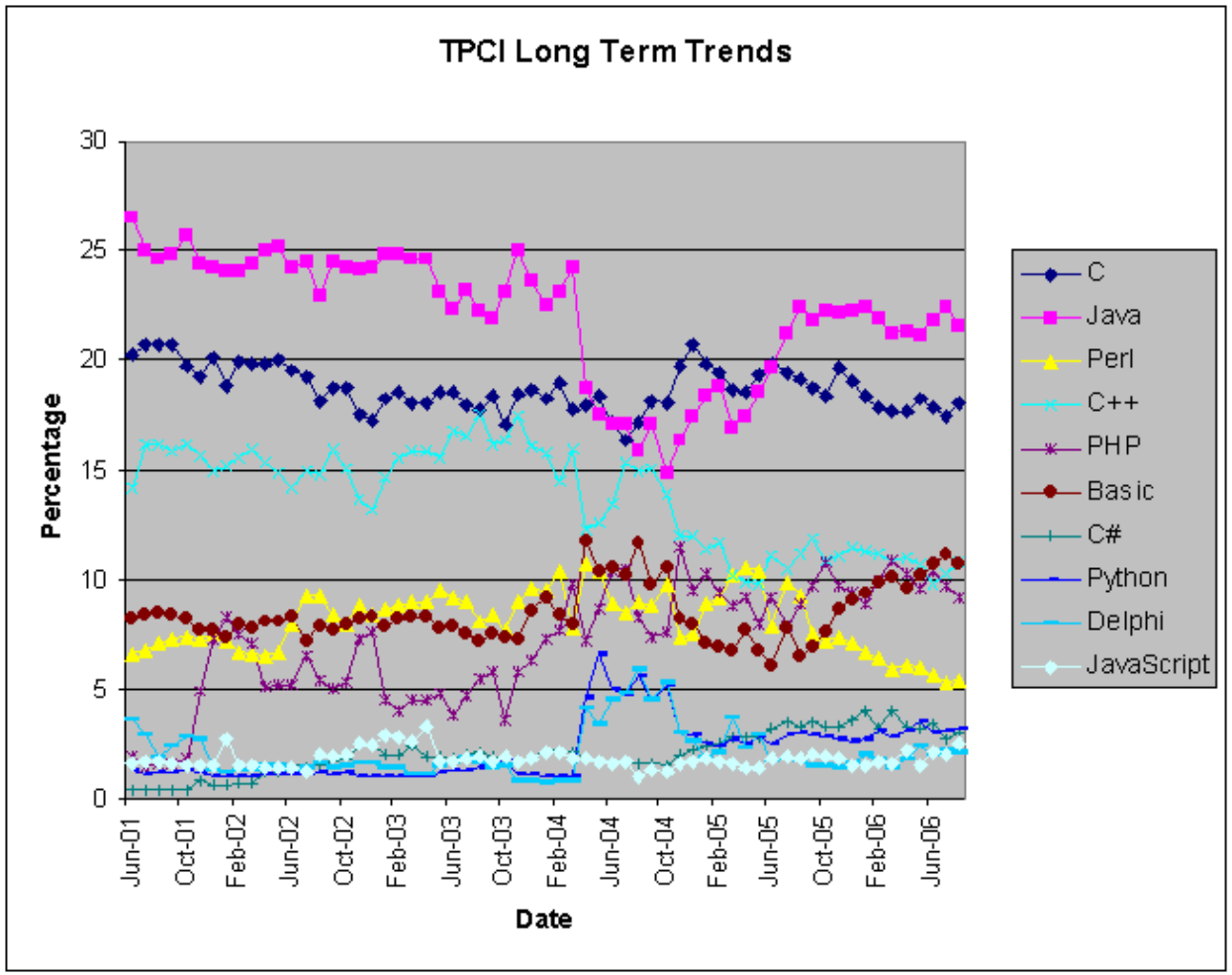


u max. de connaissance (auto- éval)



2004

Tendances générales



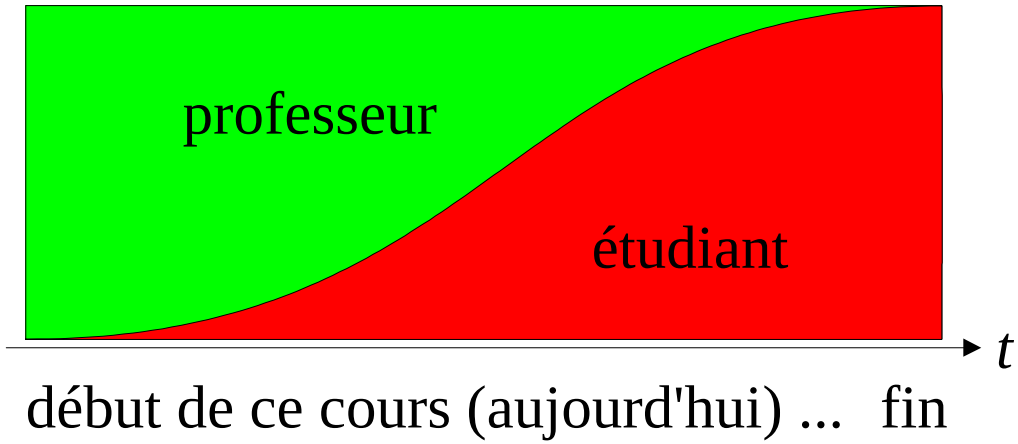
http://www.tiobe.com/

Plan du cours

- **Partie Unix (cours 1-3)**
  - Session d'utilisateur et système de fichiers
  - Shell, commandes et applications ; editeurs
  - Internet ; WWW, mail
- **Partie Programmation (cours 4-8)**
  - C++
  - Java
  - Applications et exemples

Mode d'emploi

- **Ce cours vous donne des concepts et des idées,**
- **... complétées par les travaux pratiques.**
- **Le travail <u>individuel</u> et la <u>propre initiative</u> d'approfondissement sont <u>indispensables</u> !**
- **Courbe de contribution professeur – élèves**



Lecture / Références

- **Unix / Linux / SunOS sont des systèmes «auto-documentés».**
  Le code source est (en principe) disponible pour la plus grande partie des programmes.
- **L'utilisateur dispose de deux outils très puissants pour en tirer le meilleur avantage :**
  - **`man`** (manuel en ligne d'Unix)
  - WWW ! (depuis 1991, puis à travers Google, „tutorials“)
- **Ou, peut-être, un livre de poche :**
  - L'indispensable pour Unix, J.-P. Mesters, Marabout (1992)

Historique

- **Première version Unix vers 1970 (Bell Labs)**
- **Dennis Ritchie, Ken Thompson**
- **multi-utilisateur, multi-procès**
- **Réécrit en C 1972-1973, portabilité**
- **Développement dans trois branches principales**
  - BSD (Berkeley), V1-V8 (Bell), System V (AT&T)
- **Première version de Linux en 1991**
  - freeware, support IBM-PC (MS-DOS)

Programme d'aujourd'hui

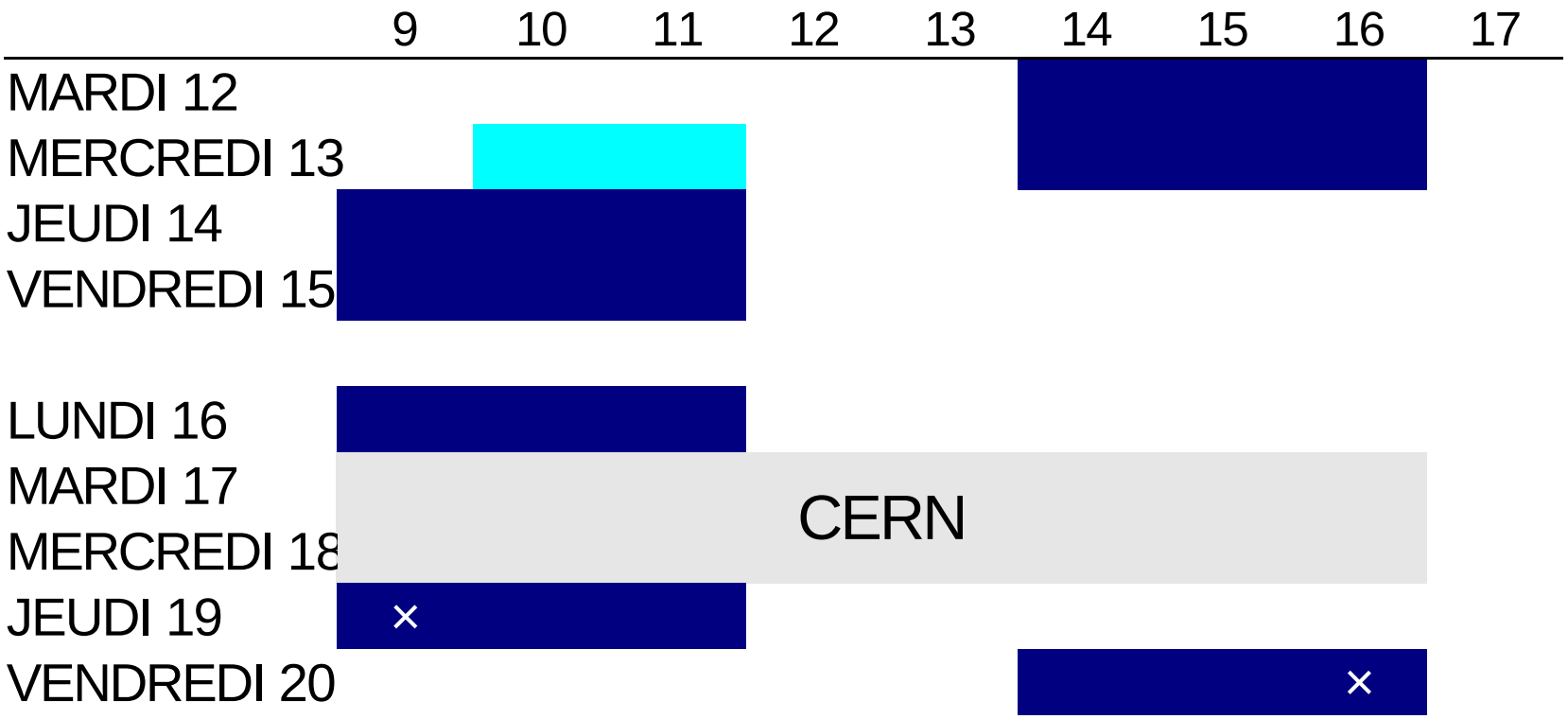
- **entrée dans une séance individuelle (login)**
- **manipuler des fichiers (copier, déplacer, effacer)**
- **modifier leur contenu à l'aide d'un éditeur**
- **envoyer / lire des message électroniques (e-mail)**
- **récupérer des documents de la toile**
- **visualiser et imprimer fichiers et documents**
- **fin de séance (logout)**

Comptes de la fac

- **b223009 – Buzzi**
- **h604480 - Halladjian**

- ***les autres :***
  - comptes génériques (sr1, sr2, … *luminy*)
  - Demandez votre UID à la scolarité !

Horaires


9
10
11
12
13
14
15
16
17
MARDI 12
MERCREDI 13
JEUDI 14
VENDREDI 15
LUNDI 16
MARDI 17
MERCREDI 18
JEUDI 19
VENDREDI 20
CERN

Modalités « exam » :
Projet Informatique

- **vérification des connaissances acquises/existantes**
- **en rapport avec le séminaire *(recommandation)***
  illustration ou mise en application du sujet
- **sujets disponibles après le 26 septembre**
- **volume de travail : 20h / 1 mois (2 oct – 3 nov)**
- **rapport : max. 5 pages (hors listing)**
  permettant de juger les compétences et
  la contribution originale de l'étudiant
- **soutenance : 10' + 5' questions (6–9 nov)**

» Login «

- **Interface graphique (XDM) spécifique à chaque installation (moyen plus simple : `ssh`)**
- **Premier écran : choisir le serveur Options/Connexion à distance/Selectionner**
  - **`apps`** ou **`data`**
- **Second écran : entre UID / mot de passe**
- **Ouvrir (au moins) un terminal**

```
user@apps$ ls
```

```
user@apps$ date
```

A vos claviers!

Le mot de passe

- **Protège le compte contre l'accès par autrui**
- **Contient plus de 6 caractères, possiblement des caractères »spéciaux« (0-9,.!%&)**
  - mnémotechnique : ie1pnqnajn
  - Que j'aime à faire apprendre un nombre utile aux sages…
    3,14 15 9 2 6 5 3 5
- **Pour le changer, on se sert de la commande `passwd`**

Orientation dans le système de fichiers

```
$ pwd
$ ls
$ cd ..
$ pwd
$ ls
$ cd ..
$ ls
$ cd
$ pwd
```

A vos claviers!

Créer un fichier

```
$ touch f1
$ ls
$ cat f1
$ cat > f2
  contenu
  ^D (Ctrl-D!)
$ ls
$ cat f2
```

A vos claviers!

Exploration (suite)

- $ Repérage absolu
- $ ls ~hoffmann
- $ ls ~hoffmann/1
- $ cat ~hoffmann/1/Proust

- $ Repérage relatif
- $ cd ~hoffmann
- $ ls
- $ cd 1
- $ pwd
- $ ls
- $ cat Proust

- $ cp ~hoffmann/1/Proust monProust
- $ cat monProust

A vos claviers!

Examination du fichier

- **head Proust**
  - début du fichier
- **tail Proust**
  - fin du fichier
- **wc Proust**
  - compte caractères, mots, lignes

A vos claviers!

Ligne de commande, grammaire

- **Appel d'une commande**
  - ***cmmd***
- **avec arguments arg1 arg2**
  - ***cmmd*** `arg1 arg2`
- **arguments d'option**
  - ***cmmd*** `-a -b -c`
- **redirection de la sortie dans un fichier**
  - ***cmmd*** `> output` **(écrase le fichier)**
  - ***cmmd*** `>> output` **(ajoute à la fin du fichier)**

Le système de fichiers

- **Arborescence**
  - **/** (répertoire racine)
    - **/etc/**
      - **/etc/passwd**
      - **...**
    - **/home1/**
      - **/home1/hoffmann/**
      - **...**
    - **/home2/**
      - **/home2/n374893/**
      - **/home2/t347854/**
      - **...**

Manipulation du s.f.

- **`cd` – change directory**
- **`cp` – copie de fichier**
- **`rm` – destruction de fichier**
- **`mv` – déplacement de fichier**
- **`mkdir` – créer sous-répertoire**
- **`rmdir` – détruire sous-répertoire**
- **`pwd` – répertoire «actuel», «de travail»**

Exercice

- **créer un sous-répertoire pour travailler**
- ***transférer* tous les fichiers créés jusqu'à présent dans ce sous-répertoire**
- **créer un deuxième sous-répertoire au même niveau que le premier**
- ***copier* les fichiers du premier dans le deuxième**

```
user@apps$ mkdir travail
user@apps$ cp f1 travail
user@apps$ cp Proust travail
...
user@apps$ ls -l travail
```

```
user@apps$ mkdir copie
user@apps$ cp travail/* copie
user@apps$ ls -l copie
```

Solutions

- **`man` – manuel sur *toutes* les commandes Unix**
- **options (`cp -r`)**
- **jokers :**
  - Créez un troisième répertoire ...
  - Copiez tous les fichiers d'un coup :
    `cp ssrep1/* ssrep2`
  - Vérifiers (`ls`)
  - Essayer de sur-écrire par la même commande **`cp`**
  - De même avec **`cp -i`**
  - **`rm -i *`**

A vos claviers!

Examiner un fichier

- **Revenez à «Proust» (ou une nouvelle copie)**
- **`wc` – compter**
- **`grep Madeleine Proust`**
- **options ? (`grep -i`)**

A vos claviers!

Pour finir

- **Editeurs :**
  - vi – proche de la machine, le plus puissant, difficile d'apprentissage
  - emacs – le plus complet, contient une interface de programmation interne (LISP), difficile à apprendre
  - nedit
  - xedit
  - pico
- **Essayez-les : `<editcmd> Proust`**

Résumé du 1er cours

- **Qu'est-ce que nous avons appris ?**
  - Entrer dans une session Unix (et sortir)
  - se repérer, créer des fichiers, les manipuler
  - editeur(s)
- **Qu'est-ce qui reste pour demain ?**
  - Commandes avancées (sort, compression)
  - scripting (exécution automatique)

L'informatique au service de la physique

Programmation shell, éditeurs et autres outils

Dirk Hoffmann, Cours d'Informatique, 2nde année Master Phys. Théo., Phys. Math., Phys. Part.

Plan du cours

- suite de l'exercice d'hier (manipulation de fichiers)
- **Commandes, options et grammaire shell**
- **Editeur**
- **Scripts (shell)**
- **Outils d'affichage, applications, paquets**

- Utilisation du WWW

Exercice

- créer un sous-répertoire pour travailler
- *transférer* tous les fichiers créés jusqu'à présent dans ce sous-répertoire
- créer un deuxième sous-répertoire au même niveau que le premier
- *copier* les fichiers du premier dans le deuxième

```
user@apps$ mkdir travail
user@apps$ cp f1 travail
user@apps$ cp Proust travail
...
user@apps$ ls -l travail
```

```
user@apps$ mkdir copie
user@apps$ cp travail/* copie
user@apps$ ls -l copie
```

Système de f

Séparateur de niveaux : **/**

Noms de répertoire spéciaux :

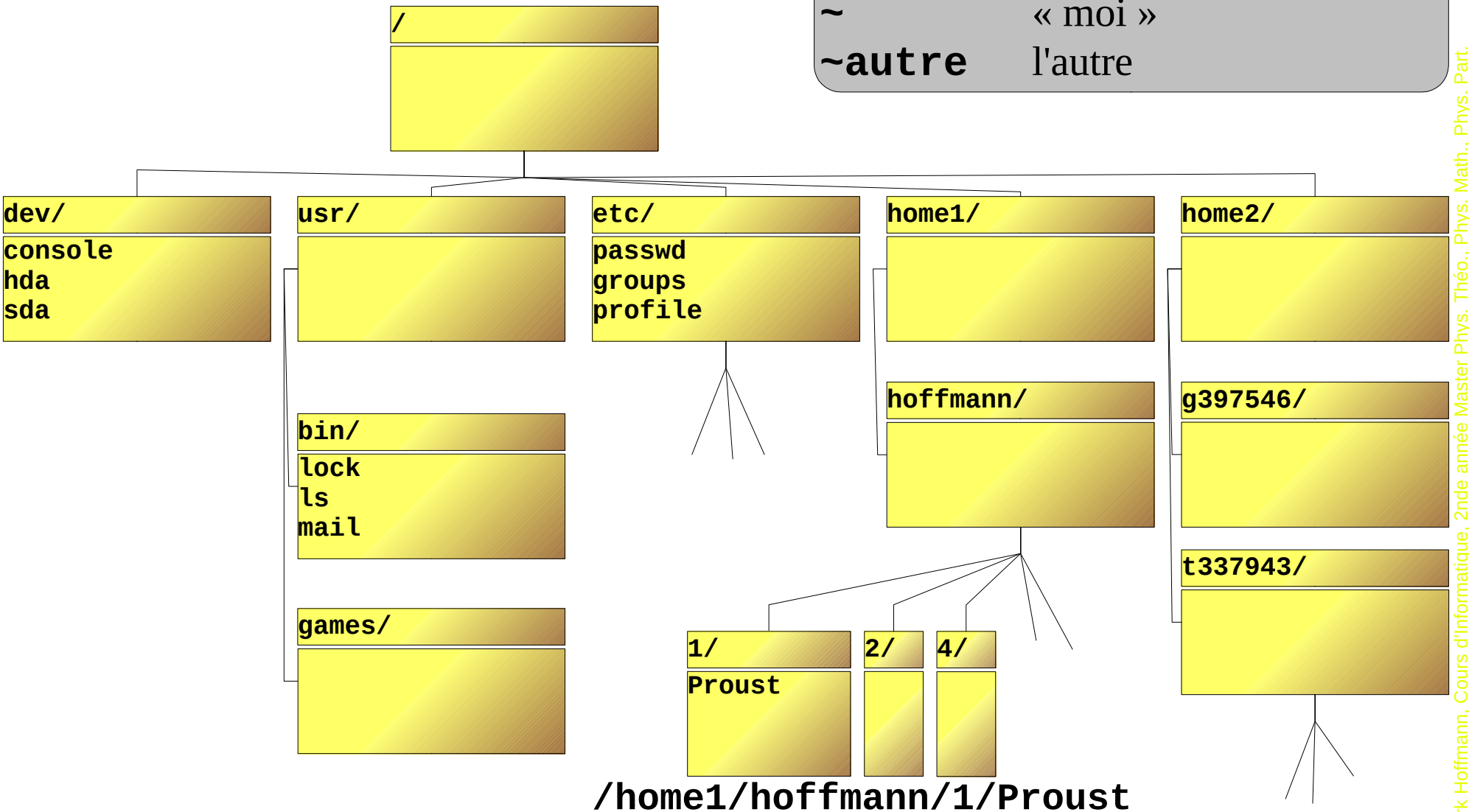
.. répertoire supérieur

. répertoire courant

Répertoires principaux (HOME) :

~ « moi »

~autre l'autre



/home1/hoffmann/1/Proust

ls

- `$ ls` – agit sur le répertoire actuel
- `$ ls .` – équivalent !
- `$ ls -l` – «long», détails des fichiers
- `$ ls -l ~hoffmann` – voir un autre répertoire
- `$ ls ..` – répertoire au-dessus
- `$ ls /` – répertoire «racine»
- `$ ls -F` – avec indication du type
- `$ ls -R` – récursivement
- `$ ls -l f1 f2` – plusieurs arguments
- `$ ls -l f*` – «joker», caractères génériques !

Noms des fichiers

- **limités à 256 caractères, choisi parmi les 52 lettres, 10 chiffres et les signes, …**
- **SAUF : `/ \ < | > * ? ! ^ & ;`**
  (et quelques autres, selon saveur du shell)
- **Noms interdits : `.` `..`** (répertoires act. et sup.)
- **Fichiers commençant par point (`.secret`) sont cachés dans `ls` normal : `ls -a` («all»)**

Créez un fichier secret chez vous! Vérifiez!

Jokers, wild patterns

- *** se substitue à n'importe quelle chaîne de caractères**
  - `*06` = `oct06 nov06 dec06 total06`
- **? se substitue à exactement un caractère**
  - `oct?6` = `oct96 oct06 oct16`
- **`[axy]` se substitue à *un* des caractères a, x, y**
  - `ju[ln]0[4-6]` = `jun04 jul05 jun06 jul06`
- **L'interprétation est effectuée avant l'exécution de la commande, en fonction des fichiers existants!**

A vos claviers !

Protections (1)

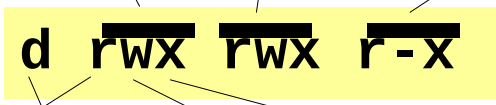
- **Résultat de `ls -l`**
  - type de fichier et protections
  - nombre de liens (fichier)
    nombre de fichiers (répertoire)
  - propriétaire (owner), groupe
  - taille
  - date de dernière modification

Essayez/vérifier chez vous!

```
drwxrwxr-x   2 hoffmann profs        512 sept 12 17:28 1
drwxr-xr-x   2 hoffmann profs        512 sept 13 12:16 2
drwxr-xr-x   2 hoffmann profs        512 sept 11 11:15 bin
drwx------   2 hoffmann profs        512 sept 15 2004  Mail
drwxr-x---   5 hoffmann profs        512 sept 12 10:33 old2005
drwxr-xr-x   2 hoffmann profs        512 sept 13 11:57 public_html
drwxr-x---   3 hoffmann profs        512 sept 11 10:31 so7
```

Protections (2)

- **Signification**
  - propriétaire (owner), groupe, tous (world)



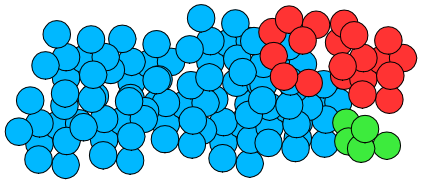
  - lecture (read), écriture (write), exécution (activation d'un répertoire)
  - «fichier» spécial (**d**, **c**, …)

Protections (3) : changer

- Commande **chmod** (man chmod!)
  - **chmod <mode> fichier1 fichier2 fichier 3**
- **changement différentiel :**
  - **u=user, g=group, o=others, a=all**
  - **r=read, w=write, x=execute**
    - **ajouter accès en lecture pour tous** — **chmod a+r f1**
    - **enlever droit d'écriture/modif aux autres** — **chmod o-r f1**
    - **attribuer droits écrit./lect. à moi** — **chmod u+rw f1**
- **changement absolu**
  - **chmod 660 f1**

Essayez !

$

$ $$125 = 5 \cdot 10^0 + 2 \cdot 10^1 + 1 \cdot 10^2 =$$ 

$

$

$

$

$

$

$

cp

- `$ mkdir mardi; cd mardi`
- `$ ls ~hoffmann/2`
- `$ cp ~hoffmann/2/Nobel* .`
- `$ ls -l` *OK ?*
- `$ mkdir copie`
- `$ cp Nobel* copie` `cp Nobel* copie/.`
- `$` De même avec `cp -i`

- `$` Finalement, retenez : `cp -r` (récursif)

Espace créativité

- `$ cat Nobel1 Nobel2 > Nobel1+2`
- `$ head Nobel1`
- `$ head Nobel*`
- `$ tail, wc, …`
- **$ Combien de prix Nobel en phys/chem/… ?**
- **$ Depuis quand le prix d'économie ?**
- **$ …**

grep : Chercher dans un fichier

- **Revenez à «Proust» (ou une nouvelle copie)**
- **`wc` – compter**
- **`grep Madeleine Proust`**
- **options ? (`grep -i`)**
- **Combien de „madeleine“ dans Proust ?**
  - **`grep -i madeleine Proust | wc`** – pipe, tuyeau
- **Combien de „marie“ inscrits (utilisateurs)**
  - **`grep -i marie /etc/passwd`**

A vos claviers!

Prix Nobel en 1999

- `grep 1999 Nobel.*`
- `awk '/1999/' Nobel.*`
- `awk '/1999/ {print $1, $2}' Nobel.*`
- `awk -F- '/1999/ {print FILENAME,":",$2}' \ Nobel.*`
- **awk = processeur de texte**
  - séparateur (par défaut) : espace
  - expression régulière (essayer **`/19*/`**, **`/19?4/`** !)

2ème espace créativité

- `$ cp *`
- `$ mv Nobel.chem chem.Nobel`
- `$ générer Nobel1999`
- `$ …`

Pour finir

- **Editeurs :**
  - vi – proche de la machine, le plus puissant, difficile d'apprentissage
  - emacs – le plus complet, contient une interface de programmation interne (LISP), difficile à apprendre
  - nedit
  - pico
- **Essayez-les : `<editcmd> Proust`**

vi

- **mode déplacement**
  - „une touche“
    - 4 directions : ijkl
    - par mot (word): wb
    - début/fin de ligne : 0$
  - multiplication d'opérations :
    - 5 mots : 5w
  - …
- **mode commande :**
  - :q – sortir
- **mode édition**
  - i – insert
  - a – append
  - 4s – subsitute (4 caract.)
  - fin avec ESC !

Autres éditeurs

- **nedit, pico : utilisation intuitive,**
  optimale (?) pour le débutant
- **emacs : beaucoup de facilités d'utilisation**
  beaucoup d'options
  entièrement programmable
- ***Une grande question de goût (et de polémique)!***

Informations sur le Web

- **It's surftime ...**
  - Google
  - hep-ph
    - Combien de publications „Talby“ ? „Hoffmann“ ?
  - Wikipedia
    - dactylographie („touch typing“)

Visualisation

- **PostScript *.ps**
  - `gv article.ps`
- **PDF *.pdf**
  - `acroread livre.pdf`
- **graphiques *.gif *.jpg *.png *.tiff ...**
  - `gimp belle_image.gif`
- **binaire**
  - `od -x -a | less`

Résumé du 2e cours

- **Qu'est-ce que nous avons appris ?**
  - les commandes principales sous Unix
  - manipulation et édition de fichiers
  - outils
  - navigation sur la toile
- **Qu'est-ce qui reste pour demain ?**
  - scripts
  - organisation de programmes avec **`make`**
  - compilation

L'informatique au service de la physique

Programmation shell, gestion de fichiers et compilations

Dirk Hoffmann, Cours d'Informatique, 2nde année Master Phys. Théo., Phys. Math., Phys. Part.

Plan du cours

- **Environnement, initialisation du shell `bash`**
- **Compilateur de texte TeX**
- **Scripts (shell)**
- **L'outil «make»**

Entrée en scène

- **Quelques commandes pour se répérer sur l'ordinateur**
  - **`hostname`** – affiche le nom de l'ordinateur
  - **`who`** – qui d'autre est là ?
  - **`who am i`** – La réponse risque de décevoir…
  - **`w`** – qu'est-ce qu'ils font ?
  - **`ps`** – table des processus
    - options : -a (all), -f (full)

A vos claviers!

Processus

- **Lancement du navigateur (ou autre)**
  - **`mozilla`**
  - **`mozilla &`** – A quoi sert le signe «et» ?
- **Ou est-il ?**
  - **`ps`**
  - **`ps -f | grep mozilla`**
- **Sniper**
  - kill 84

```
user@apps# mozilla &
[2] mozilla
user@apps# kill %2
```

```
PID  TTY    TIME COMMAND
 41   09    0:10 bash
 84   09    0:01 mozilla.bin
 85   09    0:00 ps
```

A vos claviers!

Foreground / background

- **Lancement au premier plan**
  - `mozilla`
- **Arrêt (kill) via clavier : ^C (`kill -STOP`)**
- **Suspension : ^Z (`kill -SUSP`)**
- **rappel au premier plan (foreground) : `fg`**
- **envoi à l'arrière-plan (background) : `bg`**

A essayer «à la maison»

Variables

- `$ echo Hello world`
- `$ echo contenu > f1; cat f1`
- `$ echo $HOME`
  `echo $PS1`
- `$ set` – pour voir toutes les variables
- `$ MYVAR=a`
  `myvar=b`
  `EXAMPLES=~hoffmann/3/`
- `$ echo $MYVAR/$myvar`
  `ls $EXAMPLES`

A vos claviers!

Le prompt

- `$ PS1='Quoi faire ? '`
- `$ PS1=\u@\h{\w} '`
- `$ bash`
- `$ ps`
- `$ exit` (ou Ctrl-D)
- `$ export PS1`
- `$ bash`
- `$ exit` (pour ne pas laisser trainer de shell !)

A vos claviers!

alias

- **Création d'un alias pour listing „long“**

```
alias ll='ls -l'
ll
which ll
```

- **„ls“ plus explicite**

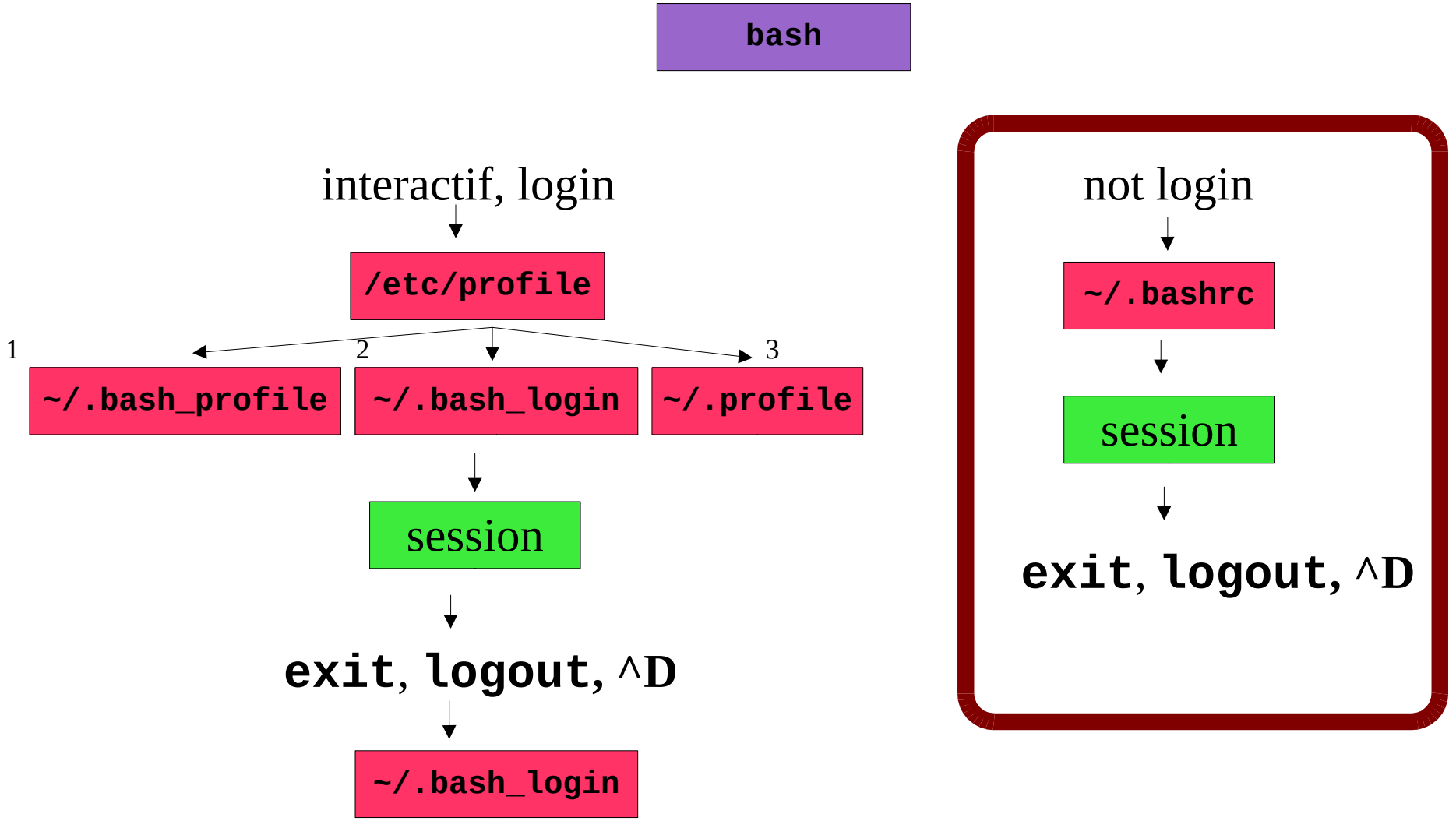
```
alias ls='ls -F'
ls
\ls
```

- **liste des alias**

```
alias
unalias ll
```

A vos claviers!

Environnement de bash



Modification de l'environnement

- **Créez un (votre!) fichier `.bashrc`, dans le répertoire principal (`$HOME`)**
- **Ajoutez (par example)**

```
alias la='ls -a'
alias ll='ls -l'
alias del='rm -i'
alias cp='cp -i'
export PS1=\u@\h{\w} '
```

- **Sortez de la session, rentrez, vérifier**

A vos claviers!

Scripts

- **Créez un fichier `makeBackup`, par éditeur (voir par `cat`) :**

`~hoffmann/3/$EXAMPLES`

```
#!/bin/sh
echo Give backup directory name:
read BAKDIR
mkdir $HOME/$BAKDIR
echo Give original directory name:
read ORIDIR
cp -r $ORIDIR/* $HOME/$BAKDIR
exit 0
```

- **Authorisez l'exécution**

```
user@apps{~} chmod a+x makeBackup
```

- **Essayez**

```
user@apps{~} makeBackup
Sauveg
rep1
```

A vos claviers!

PATH

- **Contient les répertoires qui sont parcourus pour trouver une commande exécutable. (Dans l'ordre ; première coïncidence gagne!)**
- `echo $PATH`
- `PATH=$PATH:$HOME`
- **Plus propre (convention) :**
- `mkdir $HOME/bin`
  `PATH=$PATH:$HOME`
- **… et toutes vos commandes vont dans `~/bin`**

A vos claviers!

inverted high commas

- **Commande date avec arguments (formattage)**

```
user@apps{~} date
Monday July 24th, 2006 10:35
user@apps{~} date +%y%m%d-%H%M
060724-1035
```

- **résultat dans variable :**

```
date +%y%m%d-%H%M | read DATE
```

```
DATE=`date +%y%m%d-%H%M`
```

- **Créez un BAKDIR systématique dans makeBackup**

Essayez, puis comparez : `~hoffmann/3/makeBackup2`

shell : boucles

- **Créez, (dans $HOME/bin, …) la commande `makeMonthDirs` :**

```
#!/bin/sh

for i in jan feb mar apr may jun jul aug sep oct nov dec
     do
     mkdir $i
     done
ls -l
```

- **Essayez !**
  `chmod` ?
  `mkdir test; cd test` ?

A vos claviers!

shell : if

$ 2e amélioration de makeBackup

```
#!/bin/sh

# Make backup directory name:
BAKDIR=$HOME/Backup-`date +%y%m%d`
if [ -x $BAKDIR ]; then
     echo Backup directory for today already exists: $BAKDIR
     exit 1
else
     mkdir $BAKDIR
fi

echo Give original directory name:
read ORIDIR

cp -r $ORIDIR/* $HOME/$BAKDIR
exit 0
```

A vos claviers!

récupérer les exemples

- **Ecrivez un script pour récupérer les exemples du jour**

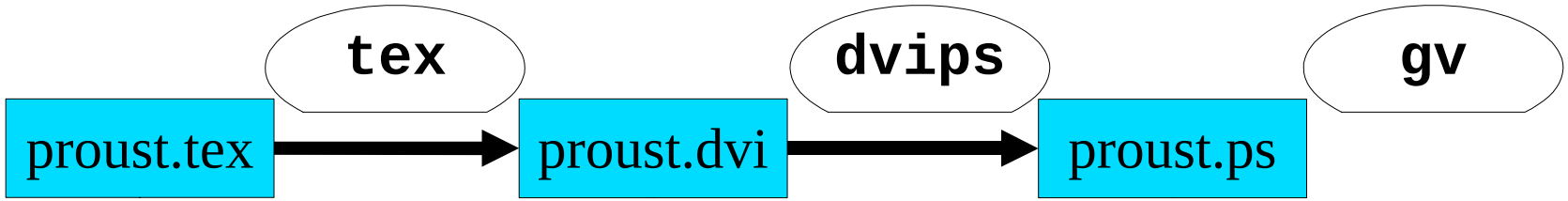
```
#!/bin/sh
# This is getEx, DH -CPPM-, Sep '06

if [ -n "$EXAMPLES" ]; then
     echo No EXAMPLE directory defined today!
     exit 1
fi

if [ $# = 0 ]; then
     ls $EXAMPLES
     fi

cp $EXAMPLES/$1 .
```

A vos claviers!

Traitement de texte

- **Deux possibilités sont à mentionner, dans le contexte Unix/Solaris (Sun) :**

1. Open/StarOffice : **`openoffice`**
   installé en v.7 sur apps/data
2. TeX/LaTeX :
   - **`tex`**
   - **`latex`**



mini-exemple TeX

- **Créez un fichier littéraire, par exemple**

- **Editez `proust.tex`**
- **Ajoutez à la fin, une ligne :**

```
\bye
```

- **Compilez-le (`tex`, vérifiez, `dvips`, vérifiez)**
- **Regardez le résultat (`gv`)**

```
tex proust.tex
ls pro*
dvips proust.dvi
ls pro*
gv proust.ps
```

A v[illegible]rs!

make !

- **L'outil make permet de gérer un grand nombre de fichiers sources et d'en générer automatiquement les résultats (destinations, targets), en tenant compte des dépendances.**
- **Il utilise deux sortes de règles :**
  - règles implicites
  - règles définies dans **`Makefile`** (ou **`makefile`**)

Test the `make`

- `$` **`getEx Makefile` – Etes-vous préparé ?**
- `$` **`less Makefile` – oui !**
- `$` **`make`**
- `$` **`ls -l`**
- `$` **Modifiez (corrigez) `proust.tex` :**
  Madeleines et madeleines (une seule variante)
- `$` **`ls -l`**
- `$` **`make`**
- `$` **`ls -l`**

A vos claviers!

Contenu / grammaire Makefile

```
# Makefile pour TeX
# version explicite

.SUFFIXES
.SUFFIXES .tex .dvi .ps

all: proust.ps

proust.ps: proust.dvi
	dvips proust.dvi

proust.dvi: proust.tex
	tex proust
```

```
# Makefile pour TeX
# Dirk Hoffmann -CPPM-, 1996-2006

.SUFFIXES
.SUFFIXES .tex .dvi .ps

all: proust.ps marcel.ps

.dvi.ps:
	dvips $<

.tex.dvi:
	tex $<
```

Résumé du 3e cours et de la partie Unix

- **Qu'est-ce que vous savez maintenant sur Unix ?**
  - Entrée/sortie de séance
  - manipulation du système de fichiers
  - outils de visualisation et de traitement de texte, compilation de texte (TeX)
  - scripts shell
  - organisation de programmes avec **`make`**
- Quelques points de chûte sur WWW