

# Introduction à Unix

## Licence 2

2013-2014

*Sylvain Cherrier*

*cherrier@univ-mlv.fr*

# Bibliographie

- **Les clés de l'administration système sous Linux**  
de Tom Adelstein, Bill Lubanovic aux Editions Digit Books
- **Linux administration**  
de Jean-Francois Bouchaudy, Gilles Goubet dans la Collection  
Les guides de formation Tsoft
- **Essential System Administration**  
par Æleen Frisch Tools and Techniques for Linux and  
Unix Administration chez O'Reilly
- **Linux System Administration**  
Solve Real-life Linux Problems Quickly  
de Tom Adelstein, Bill Lubanovic Chez O'Reilly

—

# Références

- Guide Bash du débutant :  
[http://traduc.org/Guides\\_pratiques/Suivi/Bash-Beginners-Guide/Document](http://traduc.org/Guides_pratiques/Suivi/Bash-Beginners-Guide/Document)
- Guide avancé d'écriture des scripts Bash  
<http://abs.traduc.org/abs-5.3-fr/>

# Objectif et Contenu

- **Objectif**  
Être capable d'utiliser un système Unix de façon autonome, et d'en comprendre l'organisation
- **Contenu**  
connaître l'arborescence  
Utiliser le shell  
Connaître les principaux outils du shell  
Programmer le shell

# Contenu : Linux et le logiciel libre

- **Interopérabilité**
- **Réseau**
- **Les concepts Unix (Tout est fichier, Keep It Stable and Stupid (KISS))**
- **Logiciel libre, qu'est ce que c'est ?**
- **Linux = Unix ?**
- **Unix, quel utilisation ? (Réseau, Android, Box, MacOSX, embarqué, Top500...)**

# Contenu : Shell et ... vi !!!

- Shell
  - Seul interface éventuellement disponible sur un système minimal
  - Minimum de ressources utilisées
  - Totale compatibilité
  - Télé-dépannage
- Vi

Seul éditeur (qu'est ce qu'un éditeur ?) a être systématiquement installé, ne nécessite pas d'interface graphique, grande puissance pour la manipulation des textes

# Installer Unix ?

- **Choisir un Unix : BSD, Linux, MacOSX...**
- **Choisir une distribution**
  - **Debian ( ubuntu )**
  - **RedHat, Suse, Mandriva, Knoppix, etc**
  - **BSD**
- **Méthode d'installation**
  - **CD, CD netinstall**
  - **Clé Usb**
  - **Wubi**
  - **Machine virtuelle**

# Installation : Choix de l'image

Distribution **Debian** : Choix de la version  
(*stable, testing, unstable, sid*)

Choix de l'install (DVD, CD, NetInstall, Clé  
USB, autre...)

Téléchargement sur le site Debian : ftp, http,  
ou peer to peer (légal)



# Installation

- Pas si difficile (attention aux matériels récents cependant)
- Boot sur périphérique d'installation (attention windows 8)
- Avoir éventuellement prévu de la place sur DD
- Utilisation possible dans une machine virtuelle.

# Installation : A savoir :

- Installation de base : légère et rapide
- 6 écrans textes (plus 1 graphique si install du serveur X)
- Login : utilisateur root, et création d'autres comptes (TOUJOURS !! Bases de sécurité)
- Disque dur : 1 partition pour le swap, et le reste à voir (une seule, ou /home + le reste, ou /home, /tmp, et le reste, ou...)

# BASH : user

- Commandes utilisateur
  - **login** et **logout** : connexion et déconnexion
  - **who** : qui est là ?
  - **passwd** : changement de mot de passe
  - **adduser**, **useradd**, **deluser** et **userdel** (si on est root pour gérer les utilisateurs !)

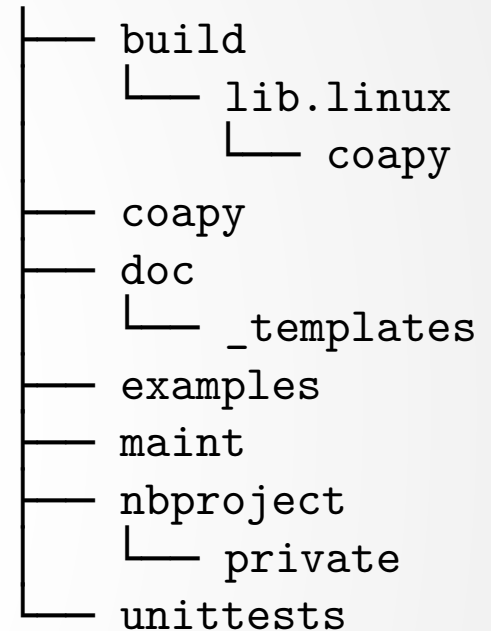
# BASH : directories

- Les commandes de répertoires ou dossier
- Un dossier contient des fichiers et des dossiers : une sorte de « fichier de fichiers »
  - **pwd** : ou suis-je ?
  - **cd *rep*** : changement de répertoire vers *rep*
  - ABSOLU / RELATIF
  - Well-known directory ( */* . *..* *~* )
  - **mkdir *rep*** et **rmdir *rep***

# Se repérer

Sur l'arbre suivant, comment se déplacer de :

- *doc* à *private*
- *templates* à *maint*
- En absolu, et en relatif
- Qu'affiche **pwd** lorsque je suis dans *private* ?



Arborescence  
contenue dans  
*/home/sylv/tp/*

# Système fichier : Arborescence

```
/
├── bin      // les logiciels nécessaires au démarrage
├── boot     // les éléments de démarrage
├── dev      // les périphériques
├── etc      // les fichiers de configuration
├── home     // les utilisateurs
├── lib      // les bibliothèques
├── lost+found // récupérateur (racine d'une partition)
├── mnt      // point de montage
├── proc     // les informations système
├── root     // le répertoire de root
├── sbin     // les logiciels système nécessaires au démarrage
├── sys      // système
├── tmp      // répertoire temporaire (droits spéciaux)
├── usr      // les logiciels utilisateurs
└── var      // espace variable (non prévisible)
```

# BASH : directories 2

- Les commandes de répertoires
  - **rm** *file* : supprime le fichier de ce répertoire
  - **cp** *source cible* : copie le fichier source vers cible
  - **mv** *source cible* : déplace la source vers cible
  - **ln** *source cible* : crée un lien de cible vers source

# Exercices Arborescence

`pwd` indique que nous sommes dans **~/nbproject/**

Effacer en relatif *Makefile*

```
rm ../doc/Makefile
```

Copier *layout.html* dans *private*

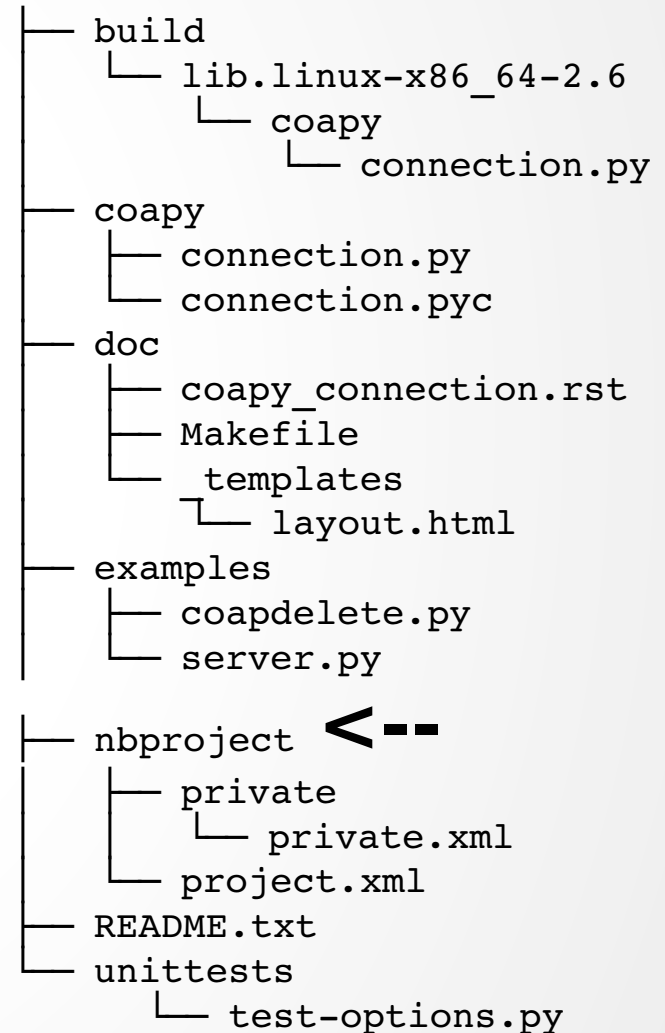
```
cp ../doc/templates/layout.html private
```

Déplacer *connection.pyc* vers `~`

```
mv ../coapy/connection.pyc ~
```

Lier *server.py* à *link.py* dans *doc*

```
ln ../examples/server.py ../doc/link.py
```





# BASH : files

- Les fichiers
  - **cat** *file* , **more** *file* , **less** *file* : voir les contenus
  - **vi** *file* , **nano** *file* , **emacs** *file* : éditer les contenus
  - **file** *file* : connaître le type du fichier
  - **stat** *file* : avoir des informations sur le fichier

# BASH : Feel the power

- **man** commande : aide sur la commande
- **locate** fichier : mais ou est ce fichier ?
- **which** fichier : lequel est choisi ?
- TAB TAB !! : la complétion
- **history** : historique de toutes vos commandes
- **!num** **CTRL-R** **fleche en haut** : utilisation de l'historique

# bash

- le système de fichier : **cd – pwd – ls – mkdir – rmdir – tree**
- Contenu des fichiers : **cat – more (or less)**
- les droits : **chmod – chown – chgrp**
- Les filtres : **grep – sed – cut – tr**
- les processus : **ps – kill – top**
- Autres outils : **netstat – sniffit – lsof**

prompt

# Le système de fichiers

Propriétaire et groupe

Date et heure

Nom du fichier

```
sylvvain@ilium:/var/tmp>ls -l
```

```
total 8
```

```
drwxr-sr-x 2  fs home 4096 avr 24 10:46 dir1
drwxr-xr-x 2  fs home 4096 avr 24 10:46 dir2
-rw-r--r-- 2  fs home    0 avr 24 10:44 fichier1
-rwxrwxr-x 1  fs home    0 avr 24 10:46 fichier2
-rw-r--r-- 2  fs home    0 avr 24 10:44 fichier3
drwxrwxrwt 15 fs home 8,0K jun 02 12:58 tmp/
lrwxrwxrwx 1  fs home    4 avr 24 10:47 tempo -> /tmp
```

Le type (-dlcb)

4 nombres en octal,

- droits spéciaux
- droit de lecture
- droit d'écriture
- droit d'exécution

Nb liens

Droits

Lien symbolique

# Droits (ou modes) du File System

- **Droits** exprimés pour le **propriétaire** indiqué, le **groupe** ou enfin les **autres**.
- 3 droits : **R**ead, **W**rite et **eX**ecute
- Un utilisateur récupère sur l'objet le droit indiqué par sa catégorie
- Objet du FileSystem : Fichier **régulier**, **répertoire**, **lien**, fichier **spécial**, **périphérique**
- **X** : fichier exécutable (programme) ou répertoire « traversable »
- Exprimés en littéral (rwx) ou octal (4+2+1)

# Droits (ou modes)

- En **littéral**, on peut utiliser les lettres u g o et a, les signes =, + ou – et les droits r w et x.
- En **octal**, on additionne les valeurs 4r 2w et 1x
- Rwx-w-r-x donne u=rwx,g=w,o=rx ou 725
- L'octal impose, le littéral permet le respect des différences existantes (+ et -)
- Droits « farfelus » possibles (007)
- Droits intéressants (--x--x--x sur un répertoire?)
- Droits « secrets » (*SETUID, SETGID et sticky*)

# Possibilités....

- Pour un objet du file sytem: droit **r w x** pour trois types d'utilisateurs : le **propriétaire**, un **groupe**, et les **autres**
- Un utilisateur a un identifiant unique (**ID**)
- Il appartient à un groupe (dit **principal**)
- Il peut appartenir à *plusieurs autres groupes*
- Cet **ensemble** (*droits sur les objets, et identification+appartenance à des groupes d'un utilisateur*) offre beaucoup de possibilités de gestion

# ...les atouts du File System...

Exemple : 3 étudiants en cinéma (groupe student). Leur home est protégé. Les profs accèdent. Les autres étudiants n'accèdent pas.

Drwxrwx-	- -	1	rir	teachers	4096	rir
Drwxrwx-	- -	1	fifi	teachers	4096	fifi
Drwxrwx-	- -	1	loulou	teachers	4096	loulou

Ils peuvent déposer leur films dans un rep commun.. les autres ne peuvent pas effacer

drwxrwxr	- t	1	prof	students	4096	commun
----------	-----	---	------	----------	------	--------



# Et les limites....

- Malgré la multi appartenance des utilisateurs à des groupes, certains problèmes sont difficilement résolus. *Par exemple, comment vous êtes-vous organisés pour vos projets en binome ?*
- Pour résoudre ces cas qui nécessiteraient une explosion du nombre de groupes, on utilise les **Access Control Lists (ACLs)**
- Pour chaque objet, on a une **liste** d'utilisateur et leur droits, une **liste** de groupe et leurs droits
- Il faut tout **recouper** pour trouver la règle à appliquer
- Ce système a aussi des défauts (**complexité**)

# Les droits...

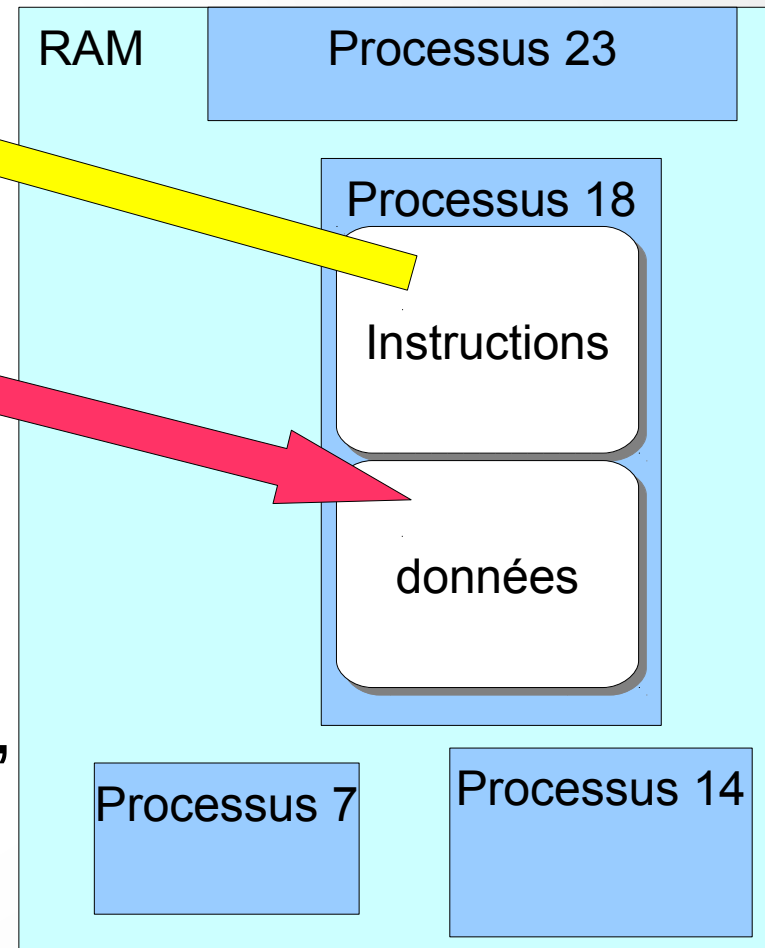
- S'appliquent aussi à d'autres objets que ceux du FileSystem
- Par exemple, pour les processus (programmes présents en mémoire), ou les ressources, vous pouvez éventuellement interagir avec eux selon ces droits...
- Par exemple **ifconfig** (utilisable, mais modification interdite), **kill** (qui permet l'échange avec un processus seulement si il vous appartient), le **ssh** dans les salles de tp, etc...

# Les processus

- Un **processus** est l'exécution d'un programme sur un ordinateur
- C'est une suite d'instructions rangées en mémoire centrale, exécutées par le processeur, et manipulant des données elles-aussi stockées en mémoire
- Des entrées sorties permettent les échanges avec l'extérieur (utilisateur, support de masse, réseau, périphériques)

# processus

Les instructions (écrites par le programmeur, et compilées dans le cas du C) donne une image binaire du programme. Ce sont des instructions pour le processeur. L'exécution de ce programme crée, manipule, génère puis efface des données



# Processus...

- Entrées sorties : Un processus a des données qui entrent, il les traite, et elles ressortent...
- Sur un système multitache, plusieurs processus sont exécutés en même temps
- en général, le système en exécute des bribes, intercalées, afin de donner l'impression du multitache
- Sur les systèmes bien conçus, chaque processus est sécurisé, protégé, isolé tout en permettant de façon fluide les interactions ...
- Protection en action : Segmentation fault !
- Echec de la protection : Blue screen of death

# processus

- Un processus, c'est un espace mémoire contenant des instructions et des données, appartenant à un utilisateur, manipulant des fichiers (mais tout est fichier), ayant un historique (*où en est-on, depuis combien de temps fonctionnons-nous ?*)
- Un processus est identifié par une numéro unique à un instant t (le PID)
- Un processus a un père, et peut avoir plusieurs fils

# Manipulation des processus

- Une processus peut être **bloquant** : Il attend un action
- On peut lancer des processus en **tache de fond** (&)
- On peut ensuite **basculer** de processus en processus
- Attention aux conflits sur stdin stdout (si # processus parlent en même temps, ou attendent une saisie en même temps)
- On peut dialoguer avec les processus de façon très succincte avec **kill**

# Manipulation de processus

- **CTRL C**    **CTRL Z**
- **Jobs** liste les processus en arrière-plan
- **Fg** fait passer un processus d'arrière-plan en avant-plan
- **Top** permet de voir les processus les plus actifs de la dernière seconde écoulée
- **Ps** (et toutes ses options) liste les processus
- **Kill** permet d'envoyer un signal à un PID



# Notions de multitache

- Plusieurs façons d'organiser la collaboration entre processus
- Un ordinateur = ressources limitées (un seul processeur) : il y a donc une lutte pour l'exécution
- Il faut permettre un avancement pseudo parallèle des travaux en donnant l'impression de fluidité

# Notions de multitache

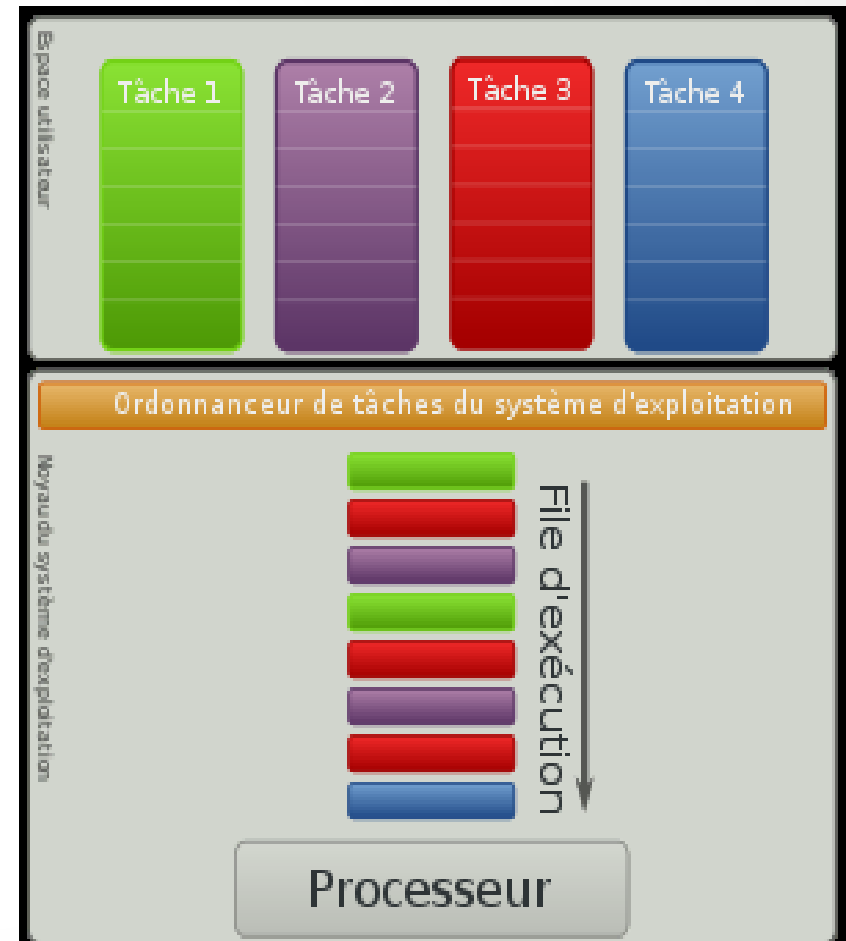
- Les processus peuvent s'exécuter complètement avant de « rendre la main » au suivant : **inefficace**
- Les processus peuvent s'interrompre à certains moments pour laisser passer les autres : **inefficace** (égoïsme, maladresse...) (multitache partagé)
- Un mécanisme contrôle les processus, les interrompt, et organise la succession des interactions (multitâche **préempté** grâce à un ordonnanceur)
- Enfin, il existe différents ordonnanceurs (best effort, temps réel, temps réel dur ou mou)

# Le scheduler en action

Source : Wikipedia

## Scheduler Unix

- 4 tâches s'exécutent
- Elles sont préemptées régulièrement par l'ordonnanceur.
- Selon leurs besoins, leur priorité, la disponibilité du processeur, le scheduler choisit le prochain.



# shell ou shells

## Un interpréteur de commande

- **sh**
- **csh**
- **ksh**
- **bash**
- **dash**
- **zsh**

# Interpréteur de commande

- **des commandes systèmes**
- **des commandes intégrées**
- **Interaction avec le Système**
- **Lancement de commandes**
- **Gestion de la machine**
- **Possibilité de Scripter**

# commandes systèmes

## Commandes extérieures utilisables dans le shell.

- pwd – ls – mkdir – rmdir
- cat – more (or less)
- chmod – chown – chgrp
- grep – sed
- ps – top
- Etc..

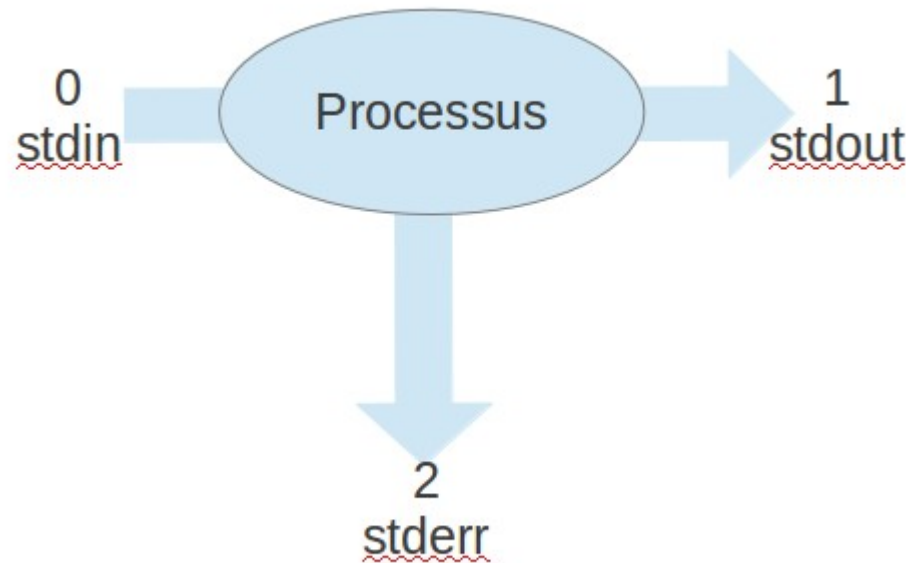
*situées dans l'un des quatre répertoires dédiés :  
/bin, /sbin, /usr/bin, /usr/sbin*

# commandes intégrées

- **cd**
- **fg/bg**
- **getopts**
- **echo**
- **read**
- **set/unset**
- **kill**
- **trap**
- **alias**
- **export**
- **shift**
- **return**

Comment avoir de l'aide  
sur ces commandes ?

# Des redirections pour stocker...



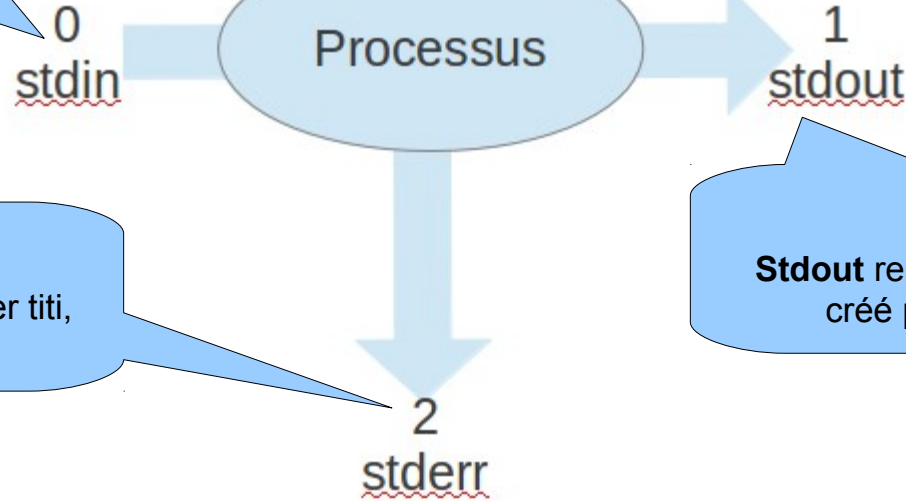
- Trois fichiers de base (stdin, stdout, stderr)
- **stdin**=*clavier*, **stdout**=*écran* , **stderr**=*écran*



# Redirections...

<binou

Ce n'est plus le clavier qui est lu,  
**Stdin** est alimenté par binou



2>>titi

**Stderr** est ajouté au fichier titi,  
qui sera créé sinon

>toto

**Stdout** remplit le fichier toto,  
créé pour l'occasion

- Les redirections changent les affectations

>   >>   2>   2>>   <   <<

- Possibilité d'utiliser des pseudo fichiers /dev/null, /dev/zero, /dev/tcp/adresseip/port/

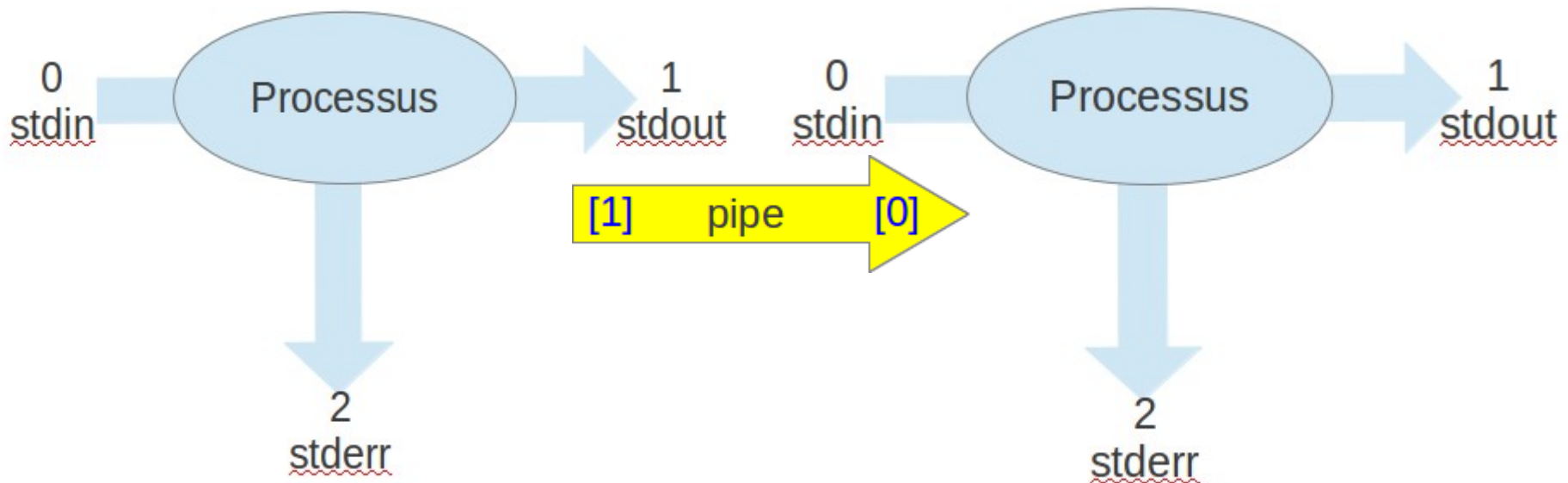
# Exemple de redirections

- `gcc -Wall -lm truc.c 2>/tmp/affichage`
- `gcc projet.c libs.o graphic.o 2>/dev/null`
- `mysqldump -p biblio > sav`
- `mysql -p biblio < sav`
- `ps ax >> ListeProcessus`
- `echo « Nom : $reponse » >> informations.txt`
- `xmllint -htmlout <clients.xml >clients.html`

# Un peu plus loin avec les redirections

- `ls -R / >/dev/null`
- `cp /dev/cdrom /dev/null 2>resultats`
- `cat > essai`
- `cat`
- `sort > infos`
- `sort`

# Des tubes pour communiquer



Le *stdout* de **cmd1** alimente le *stdin* de **cmd2**

# Des tubes pour communiquer

`cmd1 | cmd2`

Le *stdout* de **cmd1** alimente le *stdin* de **cmd2**

1

2

» *tail -f /var/log/syslog | grep -i cron*

» *echo \$var | sed -e 's/a/A'*

1

2

# Et des filtres

Très utilisés sur les tubes

- wc
- less more
- cut
- grep
- sort
- uniq
- tee
- tr
- Etc etc...

# Principes Unix

- Tout est fichier
- L'intelligence, c'est l'utilisateur
- Les commandes doivent être stables et stupides

Partant de ces principes, les tubes sont un cas typique d'application.

**WC** : compte les lignes, mots et caractères de l'entrée standard

**LESS MORE** : contrôle le défilement de l'entrée vers la sortie (limitation en taille)

**CUT** : Découpe les éléments de chaque ligne de l'entrée

**GREP** : sélectionne certaines lignes de l'entrée

**SORT** : trie toutes les lignes de l'entrée

**UNIQ** : élimine les lignes doublons qui se suivent

**TREE** : Organise une déviation : stdin est copié vers stdout, mais aussi vers un fichier

**TR** : transforme des caractères de l'entrée... Modifications à la volée...

# Quelques astuces sur les filtres

- **cut** découpe la ligne par caractères ou par champ (selon un délimiteur à définir) -c3 -c15 -c5,17 -f2 -f2,10 -d : -d' '
- **tr** permet de remplacer des caractères, d'en supprimer (-d) ou d'en tasser (-s, très utile)
- **grep** est très très puissant (man!!) pour rechercher une chaîne dans des lignes. On peut utiliser les jokers, les expressions rationnelles, et de multiples options.



# Des one-liners

- Une instruction composée de plusieurs commandes en tube s'appelle un one-liner
- On peut le construire petit bouts par petit bouts
- L'idée est d'inventer la commande dont on a besoin

```
who | cut -f1 -d' ' | grep root | wc -l
```

```
ps ax | grep firefox | cut -f1 -d' '
```

```
Cat access.log | grep « 404 » | cut -f1 -d' ' | sort | uniq | wc -l
```