Introduction au *shell* Unix

Jérôme Champavère

20 janvier 1010

Où retrouver ce document ?

Il est disponible en ligne à partir de l'adresse :

`http://www.grappa.univ-lille3.fr/~champavere/?page=Enseignement`

Pour commencer. . .

Sondage

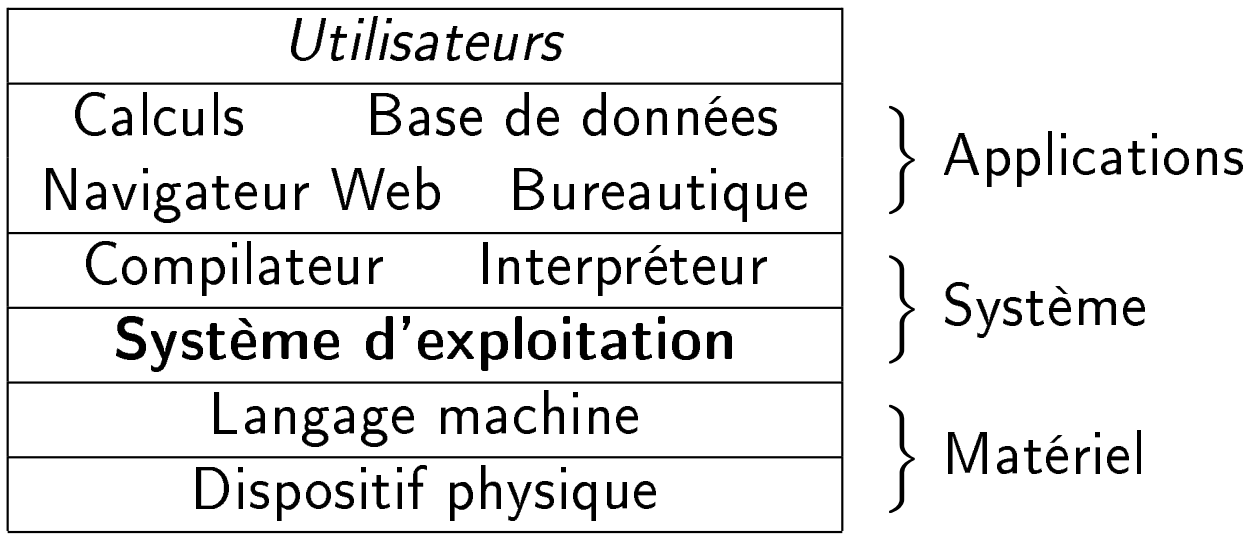
- Qui possède un ordinateur personnel ?
- Qui possède un ordinateur personnel sous Linux ou *BSD ?

Système informatique

Crédits : Everaldo Coelho and YellowIcon, GNOME icon artists, David Vignoni / Wikimedia Commons.

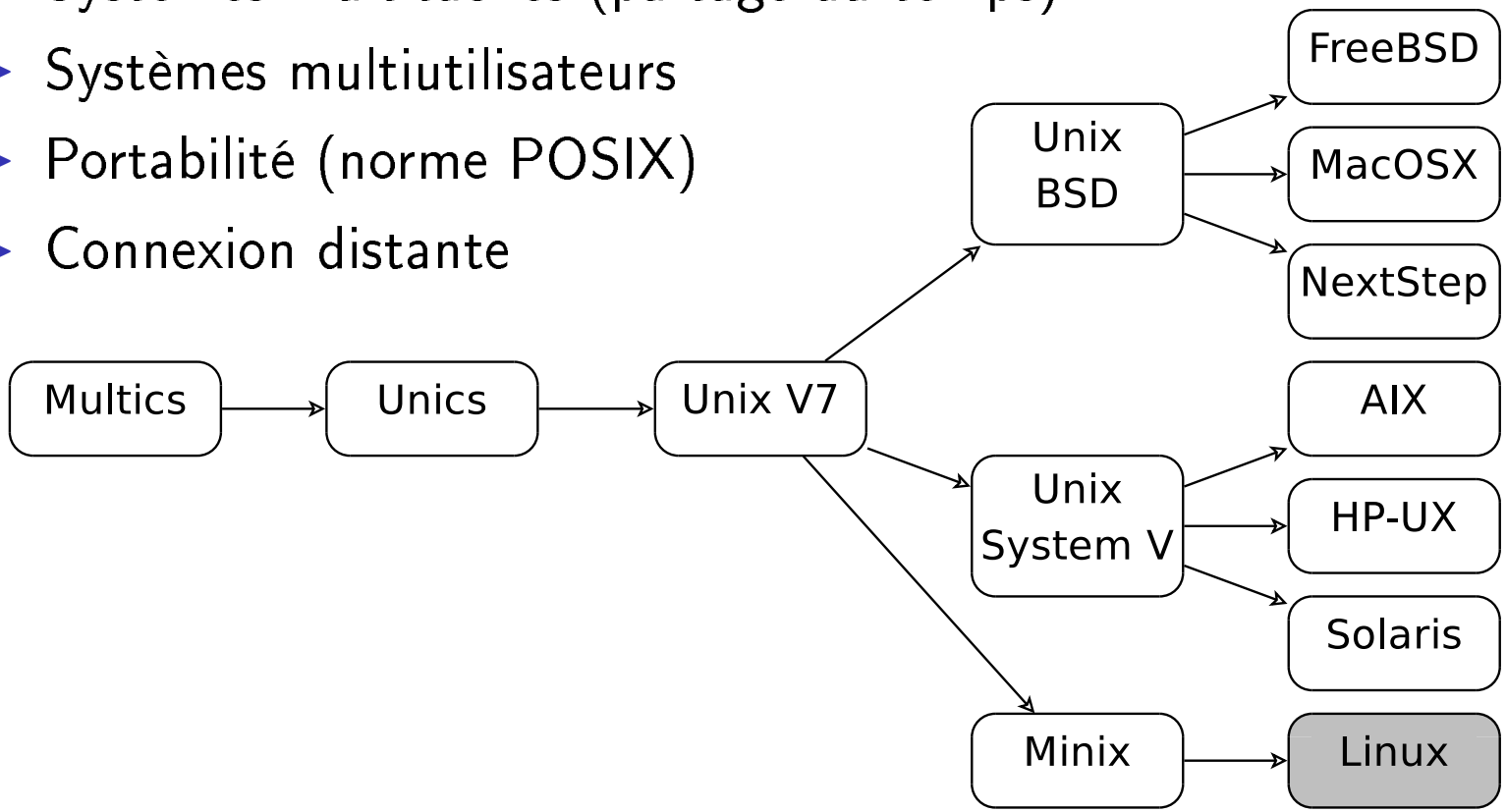
Système d'exploitation

Le *système d'exploitation* est une couche logicielle dont le rôle est de gérer tous les périphériques et de fournir aux programmes utilisateur une interface simplifiée avec le matériel.

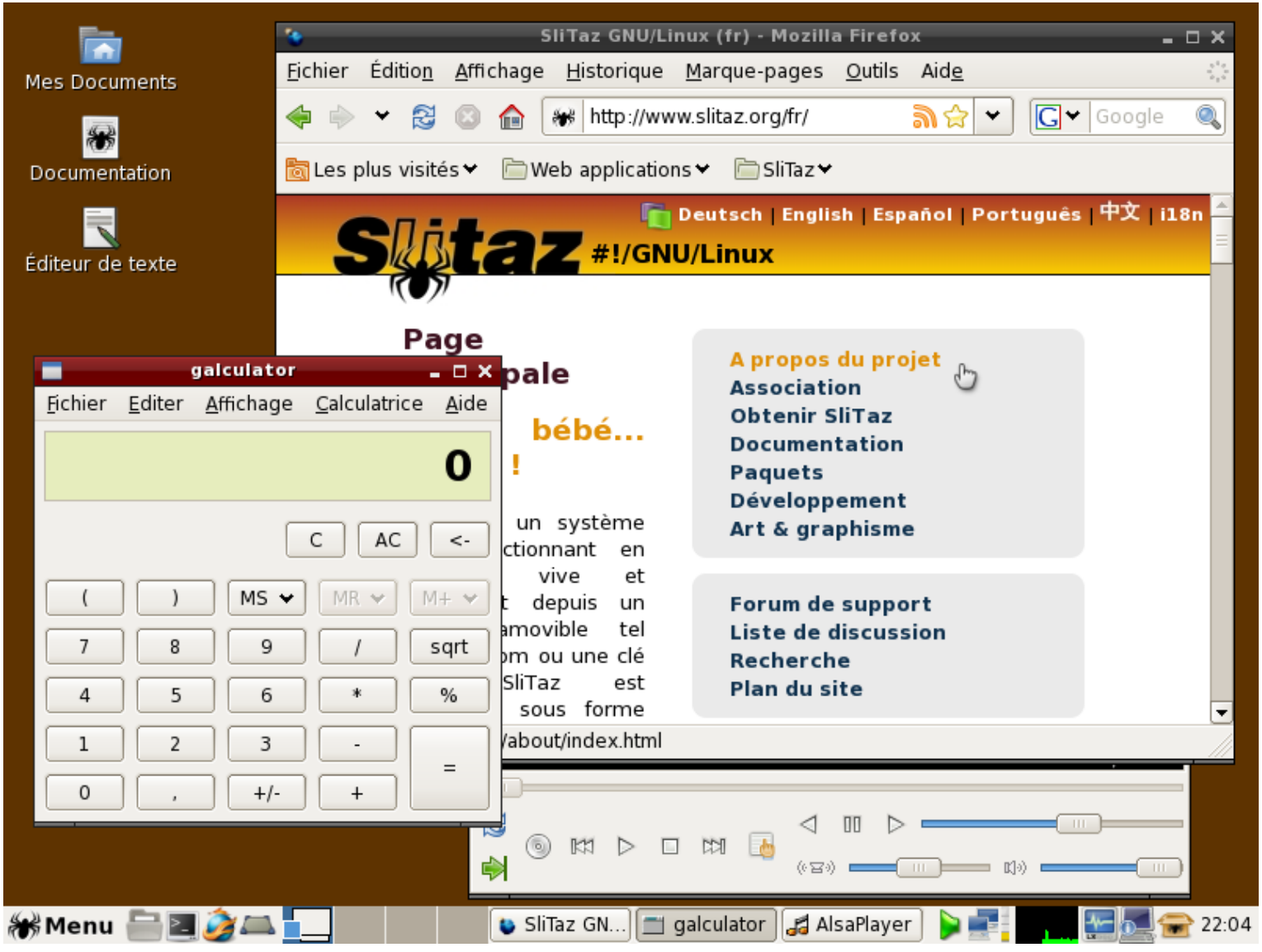


Unix et dérivés

- Systèmes multitâches (partage du temps)
- Systèmes multiutilisateurs
- Portabilité (norme POSIX)
- Connexion distante

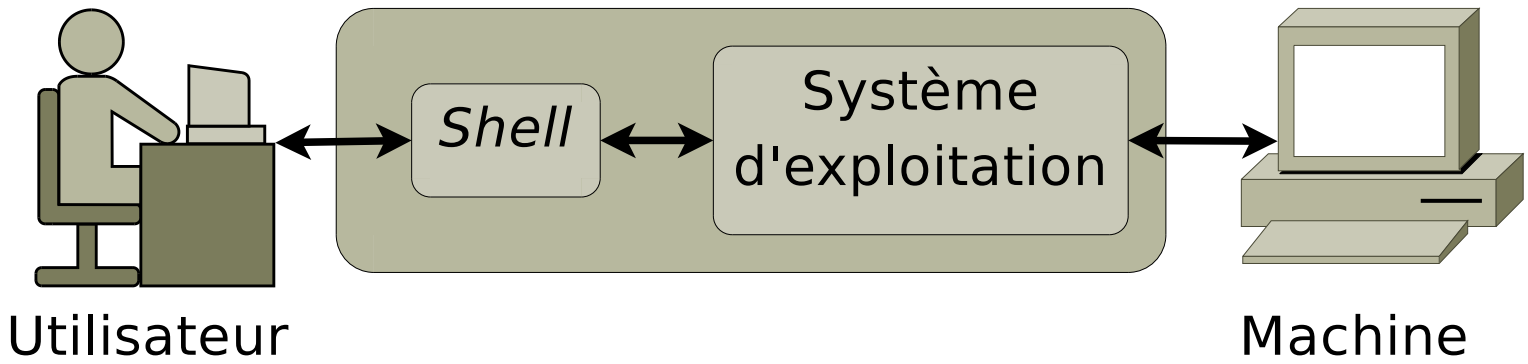


Interface graphique



Shell

Le *shell* (littéralement *coquille*) est un interpréteur en ligne de commande et un outil de scripts qui fournit une interface entre l'utilisateur et le système d'exploitation.



Interface graphique ou ligne de commande ?

Interface graphique

Utilisateur lambda. Les interfaces graphiques cachent pratiquement tous les détails du système d'exploitation.

Ligne de commande

Utilisateur avancé. Le *shell* permet à l'utilisateur de communiquer directement avec le système d'exploitation.

Avantages de la ligne de commande

- Précision et simplicité d'automatisation des tâches
- Contrôle à distance
- Uniformité
- Stabilité
- Faible consommation des ressources

Utilisateur

Dans Linux, un *utilisateur* est caractérisé par :

- son *login* (nom d'utilisateur) ;
- son mot de passe ;
- un numéro d'identification unique (uid) ;
- un numéro de groupe utilisateur (guid) ;
- un nom d'usage ;
- un répertoire (espace disque) ;
- un interpréteur *shell*.

Ces informations (excepté le mot de passe) sont stockées dans le fichier `/etc/passwd`. Par exemple, voici la ligne concernant le superutilisateur :

```
root:x:0:0:Administrateur:/root:/bin/bash
```

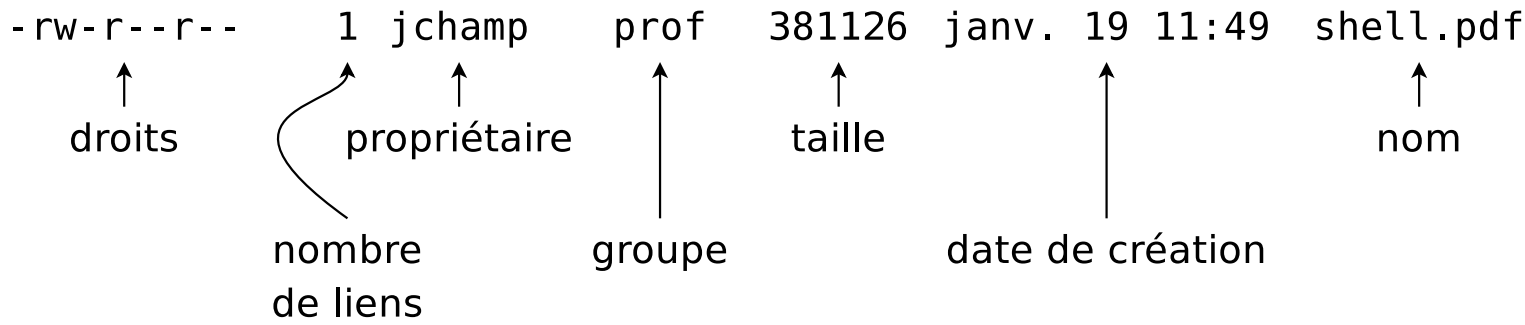
Connexion au système

- Chaque utilisateur du système dispose d'un compte protégé par un mot de passe.
- La procédure d'entrée dans le système se nomme *login* ; la procédure de sortie s'appelle *logout*. Pour entrer, l'utilisateur fournit son nom et son mot de passe. Après vérification de ce dernier, le système lance un *shell*. Le mot de passe peut être modifié avec la commande `passwd`.
- Un utilisateur dispose de ses propres fichiers et peut lancer l'exécution de processus.

Fichier

Un *fichier* est une suite d'octets caractérisée par :

- un nom, un type et une taille ;
- un propriétaire (utilisateur) et un groupe ;
- une date de création et une date de dernière modification ;
- des droits d'accès.



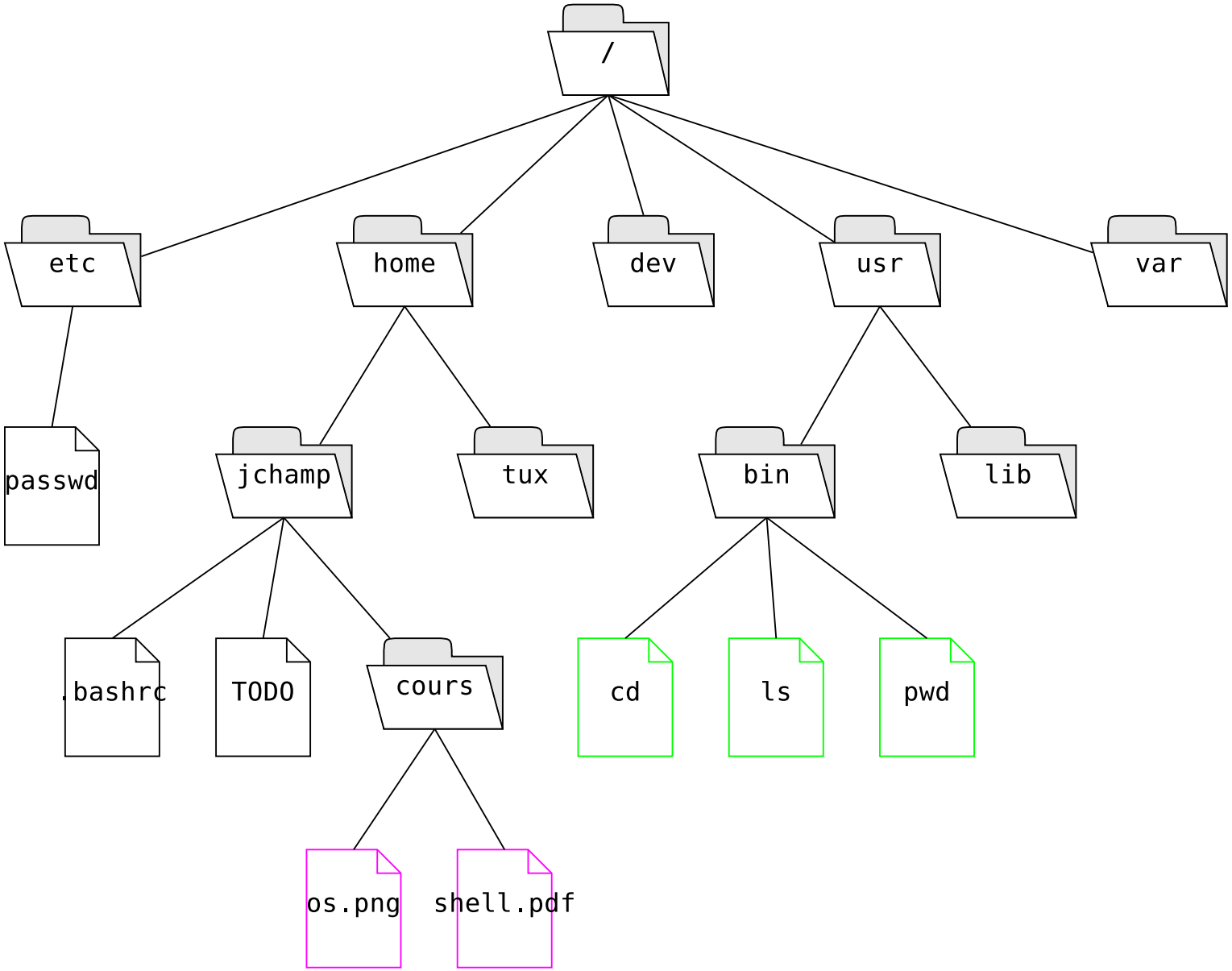
Ces informations ont été obtenues en tapant la commande :

```
$ ls -l shell.pdf
```

Système de fichiers

- Un *système de fichiers* est une structure de données permettant de stocker les informations et de les organiser dans des fichiers sur les supports physiques.
- Il offre à l'utilisateur une vue abstraite sur ses données et permet de les localiser à partir d'un chemin d'accès.
- Les fichiers sont structurés autour de la notion de *répertoire*. Les répertoires contiennent soit des fichiers, soit d'autres répertoires. Cette organisation conduit à une hiérarchie arborescente.
- Dans Linux, le répertoire *racine* du système de fichiers se dénote /.

Représentation arborescente



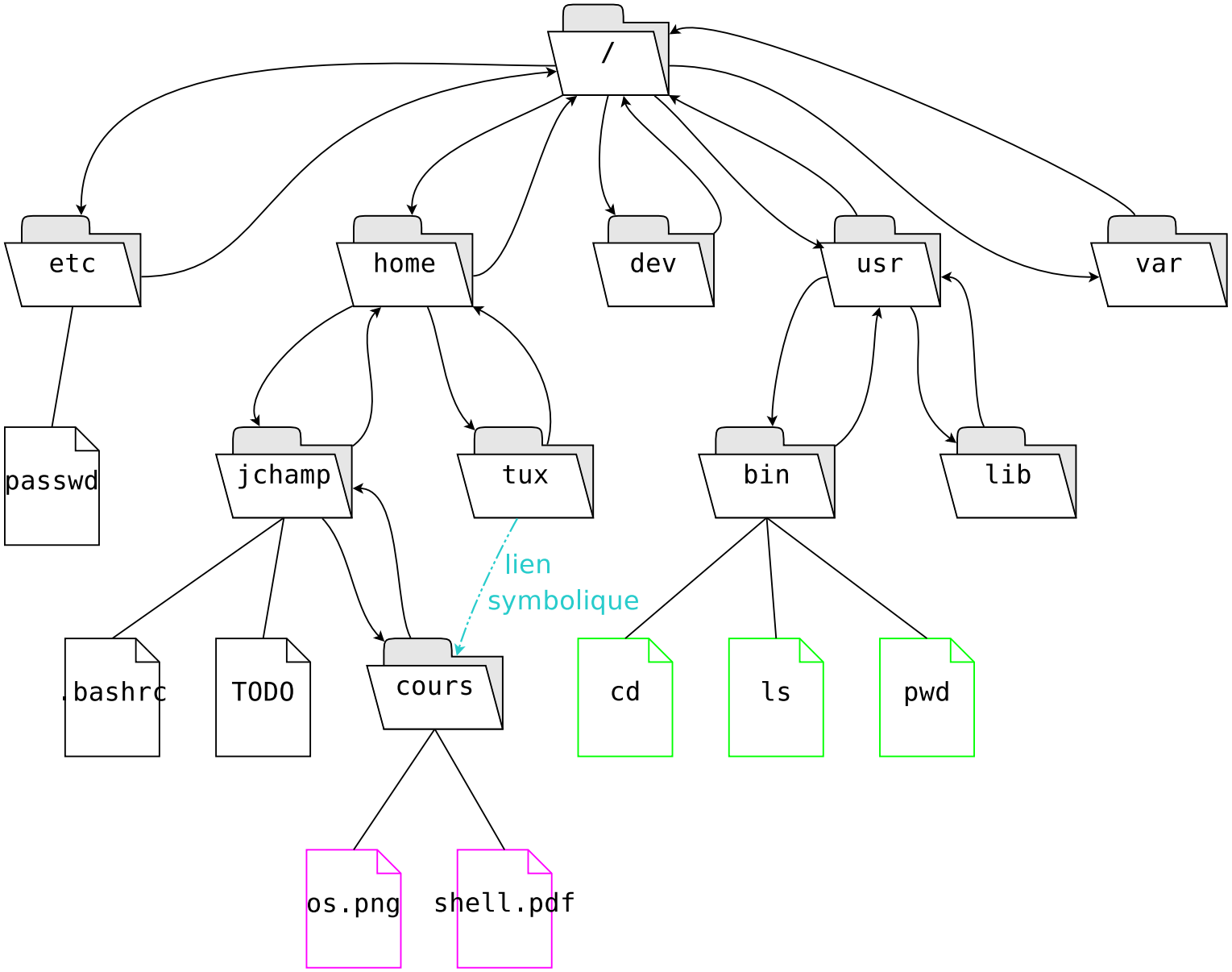
Localisation des fichiers

- On appelle *répertoire courant* celui dans lequel on se trouve à un instant donné durant la navigation dans le système de fichiers. Il est noté “ .” (point).
- Le *répertoire parent* est celui hiérarchiquement immédiatement supérieur à celui courant. Il est noté “ . .” (point-point).
- Chaque fichier de la hiérarchie est identifié par un *chemin absolu* depuis la racine. Ce chemin est composé des répertoires à traverser (séparés par un *slash* /) pour accéder au fichier. Par exemple, le chemin absolu vers le fichier `shell.pdf` est :

  `/home/jchamp/cours/shell.pdf`
- On peut également identifier un fichier à partir du répertoire courant par un *chemin relatif*. Par exemple, le chemin relatif vers `shell.pdf` à partir du répertoire `/home/jchamp` est :

  `cours/shell.pdf`
- Les *liens* permettent d'associer plusieurs noms à un seul et même fichier. Un lien *symbolique* est un simple pointeur vers un fichier.

Représentation par un graphe



Droits d'accès

Les droits définissent les permissions accordées aux utilisateurs pour accéder aux ressources du système. Les différentes actions possibles sur un fichier sont :

- la lecture (`r`) ;
- l'écriture (`w`) ;
- l'exécution (`x`) ;

Dans Linux, les droits d'accès aux fichiers sont codés sur 10 bits :

- le premier bit donne une information sur le type du fichier (répertoire (`d`), lien symbolique (`l`), simple fichier (`-`), etc.) ;
- les neuf autres bits se décomposent en trois fois trois bits indiquant les droits d'accès pour, respectivement, le propriétaire, son groupe, et les autres utilisateurs.

Par exemple, les droits pour le fichier `/bin/cp` sont :

```
-rwxr-xr-x
```

Modifier les droits d'accès d'un fichier

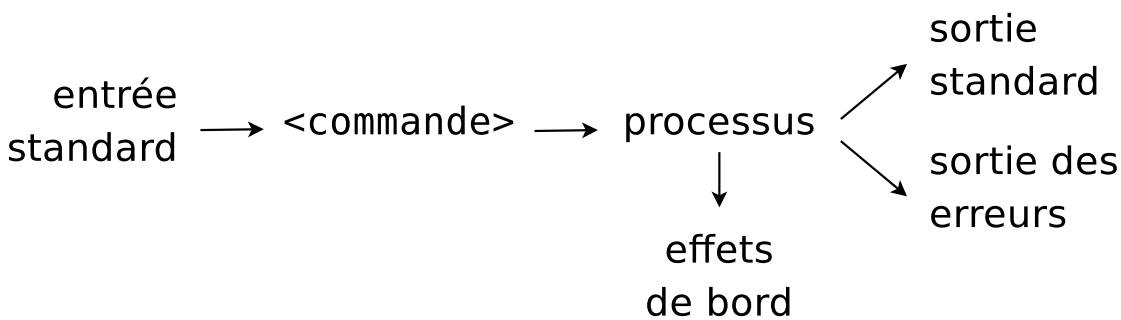
- Chaque utilisateur dispose d'un espace de stockage dans le répertoire `/home` du système. Ainsi, les fichiers de l'utilisateur `tux` sont stockés dans le sous-répertoire `/home/tux`.
- Pour changer les droits d'accès à un fichier, on utilise la commande `chmod`. Par exemple,

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

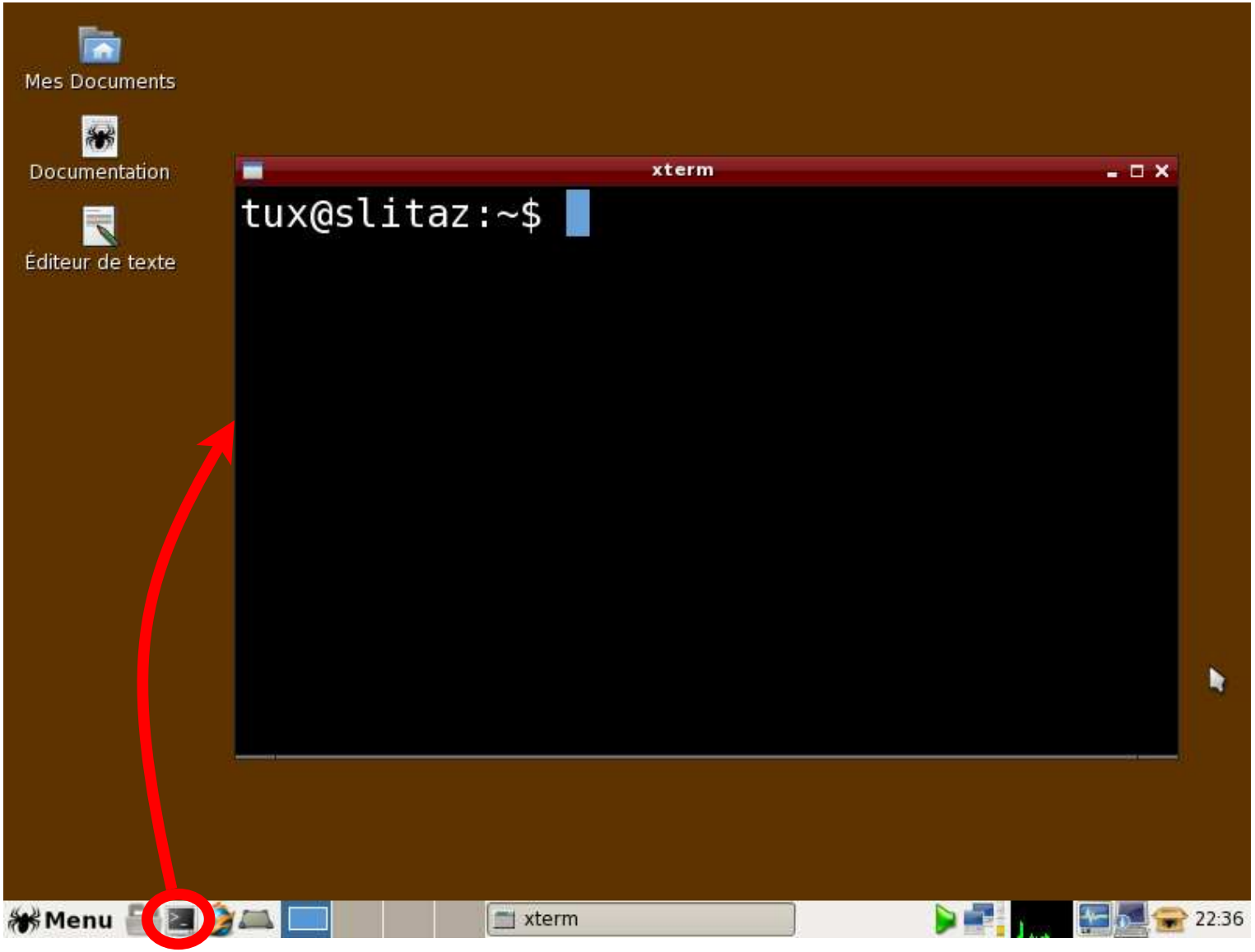
  permet de rendre le fichier `script.sh` exécutable par tous les utilisateurs.
- Notez que l'utilisateur *root* (administrateur) a tous les droits quel que soit le fichier.

Processus *shell*

- Très schématiquement, un processus est un programme en cours d'exécution.
- Le *shell* est un processus particulier qui permet de créer d'autres processus par le biais de commandes.
- Un processus est créé par une commande entrée au clavier par l'utilisateur, il produit éventuellement un résultat sur la sortie standard (l'écran), affiche possiblement des erreurs, et peut avoir des effets de bord sur le système de fichiers.



Terminal graphique



Invite de commande

- Une fois lancé, le *shell* affiche une *invite de commande* ou *prompt*, généralement $ pour un utilisateur standard et # pour l'administrateur.
- Des informations peuvent précéder le *prompt* :

  ```
  tux@slitaz:~$
  ```

  Ici, `tux` est le nom de l'utilisateur, `slitaz` le nom de la machine et `~` le répertoire courant (`~` est un raccourci pour `/home/tux`).
- On distingue deux types de commandes *shell* : les *commandes internes* et les *commandes externes*. La commande `type` donne cette information. Notez que les commandes internes ne créent pas de nouveau processus.

Lancer une commande

- Une commande *shell* est constituée du nom de la commande, possiblement suivi d'options et d'arguments. La syntaxe générale est la suivante :

  ```
  $ <commande> [options] [arguments]
  ```

- La commande s'exécute dès lors que l'utilisateur presse la touche entrée.
- On peut écrire une commande sur plusieurs lignes en utilisant le caractère \ (*backslash*) en fin de ligne :

  ```
  $ debut de la commande \
  > fin de la commande
  ```

  Le caractère > indique le début d'une nouvelle ligne.

Où trouver de l'aide ?

- La première commande à connaître est la commande `man` (manuel). Celle-ci permet d'obtenir la documentation d'une commande. Par exemple :

  ```
  $ man man
  ```

  documente la commande `man`. Pour quitter une page de manuel, on tape sur la touche `Q` du clavier (*quit*).
- L'option `-k` de la commande `man` permet d'effectuer une recherche par mot-clé dans l'ensemble des pages de manuel. Par exemple :

  ```
  $ man -k mount
  ```

  liste l'ensemble des pages d'aides contenant l'expression `mount`. La commande `apropos` est similaire.
- La commande `info` donne des informations plus détaillées.

Structure d'une page de manuel

Toutes les commandes de base sont documentées avec `man`. Les pages d'aide sont plus ou moins structurées suivant les sections :

- **NOM** nom de la commande et description sommaire de son action ;
- **SYNOPSIS** résumé de la syntaxe de la commande avec les options et les arguments ;
- **DESCRIPTION** description complète de l'action de la commande ;
- **OPTIONS** suite des options disponibles et façon dont elles modifient l'action de la commande ;
- **EXEMPLES** un ou plusieurs exemples d'utilisation ;
- **VOIR AUSSI** pages d'aides ayant rapport avec la commande.

Navigation dans le système de fichiers

- La commande interne `pwd` (*print current working directory*) affiche sur la sortie standard le chemin absolu vers le répertoire de travail courant :

```
$ pwd
/home/jchamp
```

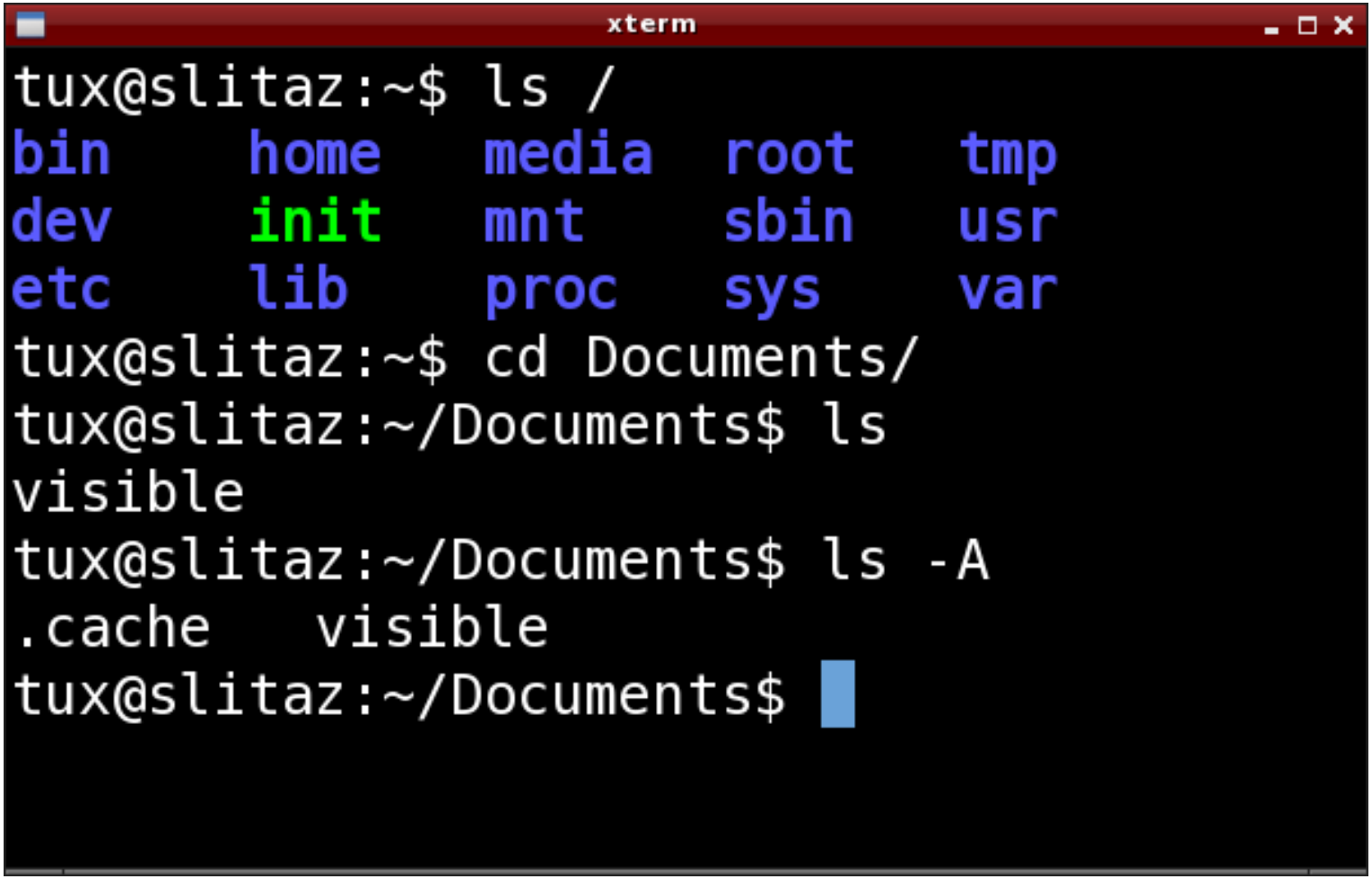
- La commande interne `cd` (*change directory*) permet de changer de répertoire courant. On peut spécifier un chemin absolu ou un chemin relatif. Par exemple :

```
$ cd cours
```

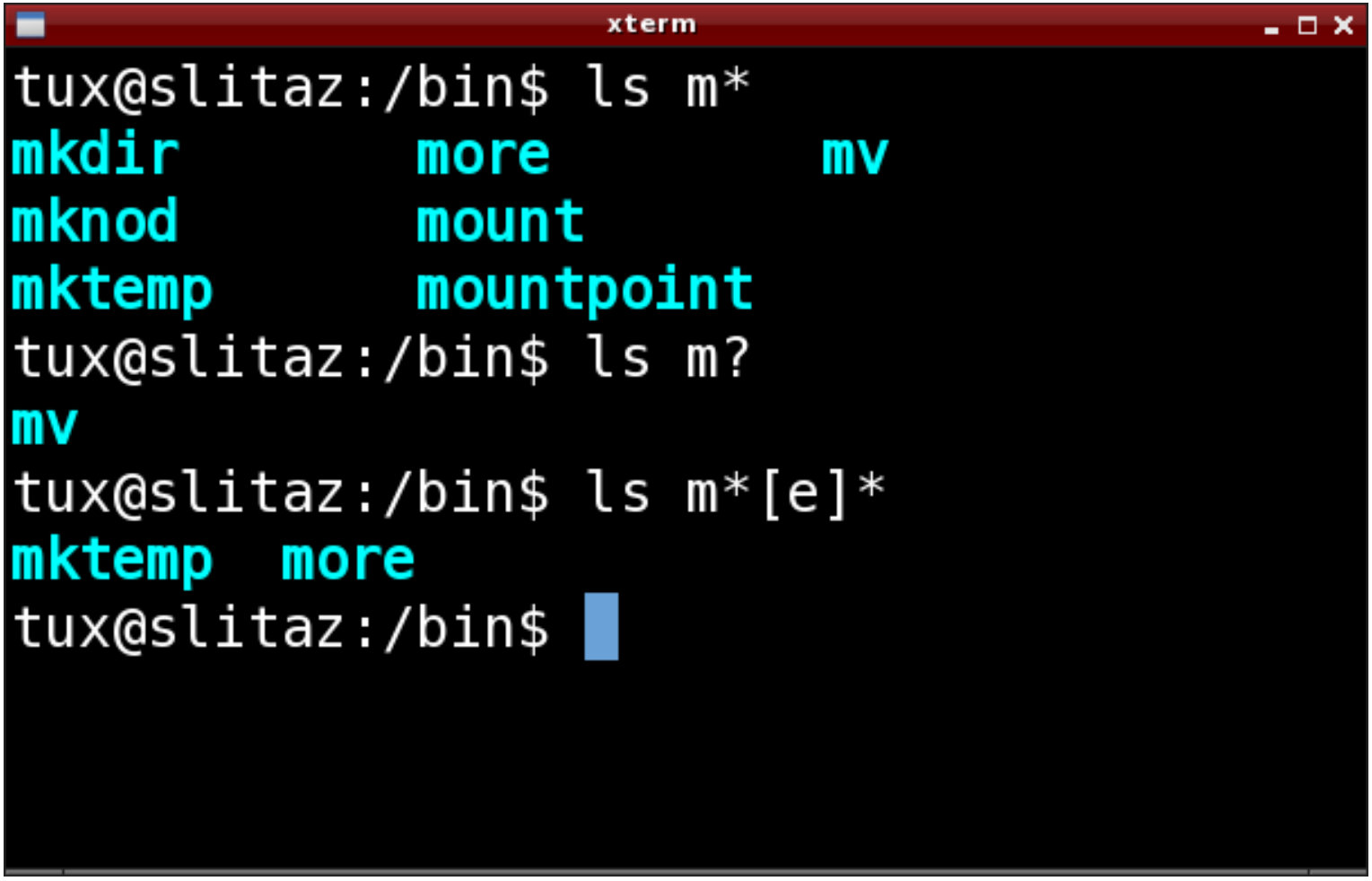
permet d'atteindre le répertoire `cours` à partir du répertoire courant (s'il existe). Ce qui équivaut à :

```
$ cd /home/jchamp/cours
```

Lister le contenu d'un répertoire



Métacaractères



Créer, supprimer des répertoires

- La commande `mkdir` (*make directory*) permet de créer un répertoire :

  ```
  $ mkdir travail
  ```

  crée le sous-répertoire `travail` dans le répertoire courant. La création d'un sous-répertoire nécessite d'avoir les droits d'écriture dans son répertoire parent.
- La commande `rmdir` (*remove directory*) permet de supprimer un répertoire à condition qu'il soit vide :

  ```
  $ rmdir Documents/
  rmdir: 'Documents/': Le répertoire n'est pas
  vide.
  ```

Copier-coller ou couper-coller en une seule commande

- La commande cp (*copy*) permet de copier des fichiers et/ou des répertoires :

  ```
  $ cp /home/jchamp/cours/shell.pdf .
  ```

  copie le fichier `shell.pdf` dans le répertoire courant.
- La commande `mv` (*move*) permet de déplacer ou de renommer des fichiers :

  ```
  $ mv shell.pdf travail
  ```

  déplace le fichier `shell.pdf` dans le répertoire `travail`.
- Dans les deux cas, si le ou les fichiers de destination existent déjà dans le répertoire de destination, ils sont purement et simplement écrasés. L'option `-i` permet de contrôler ce comportement :

  ```
  $ cp -i /home/jchamp/cours/shell.pdf travail
  cp: écraser ‘shell.pdf’?
  ```

  Pour confirmer, on tape `y` puis entrée, sinon `n`.

Supprimer des fichiers

- La commande `rm` (*remove*) permet de supprimer un ou plusieurs fichiers :

  ```
  $ rm shell.pdf travail/shell.pdf
  ```

  supprime les fichiers `shell.pdf` et `travail/shell.pdf`.
- Attention, il n'y a pas de corbeille ! Pour éviter les mauvaises surprises, on pourra également utiliser l'option `-i` :

  ```
  $ rm -i shell.pdf
  rm: détruire fichier régulier ‘shell.pdf’?
  ```
- On peut également utiliser `rm` pour supprimer des répertoires ainsi que leur contenu en combinant les options `-r` et `-f` :

  ```
  $ rm -rf travail
  ```
- Quoiqu'il arrive, la commande `rm` doit être manipulée avec précaution.

Créer, supprimer un lien symbolique

- La commande `ln` (*link*) permet de créer un lien entre deux fichiers. L'option -s spécifie un lien symbolique, autrement dit un pointeur :

  ```
  $ ln -s /home/jchamp/cours/shell.pdf
  ```

  crée un lien symbolique vers le fichier `shell.pdf` dans le répertoire courant.

  ```
  $ ls -l shell.pdf
  lrwxrwxrwx 1 jchamp jchamp 48 janv. 20 02:48
  shell.pdf -> /home/jchamp/cours/shell.pdf
  ```

- Le lien symbolique étant par définition un simple pointeur, sa suppression n'entraîne pas la suppression du fichier pointé par le lien. Inversement, la suppression du fichier cible laissera le lien orphelin.
- Les liens "physiques" sont au contraire bidirectionnels. Si l'un des deux fichiers est supprimé, l'autre conservera les données.

Visualiser des fichiers texte

- Les fichiers “ordinaires” peuvent contenir du texte, des données, ou encore du code machine. Seuls les fichiers texte peuvent être raisonnablement visualisés.
- La première solution pour consulter un fichier texte consiste à utiliser la commande `cat` (*catenate*) :

  ```
  $ cat /etc/passwd
  ```

  affiche le contenu du fichier `/etc/passwd` sur la sortie standard et rend la main à l'utilisateur.
- La commande `cat` n'est pas adaptée à la visualisation de longs fichiers. On lui préréfera dans ce cas la commande `less`, qui permet de naviguer dans le fichier et dispose entre autres d'une fonction de recherche. Pour quitter le programme `less`, on tape sur la touche `Q` du clavier.
- Une autre solution consiste à utiliser un éditeur de texte comme `vi`, `emacs` ou autre. Comme n'importe quel programme utilisateur, ceux-ci peuvent être lancés depuis la ligne de commande.

Visualiser le début ou la fin d'un fichier

- La commande `head` permet d'afficher les premières lignes d'un fichier donné en argument. Par défaut, `head` produit les 10 premières lignes. On peut spécifier le nombre de lignes désirées avec l'option `-<n>` :

  ```
  $ head -3 /etc/passwd
  ```

  affichera les trois premières lignes du fichier `/etc/passwd` sur la sortie standard.

- La commande complémentaire à `head` est la commande `tail`. Elle s'utilise de façon analogue :

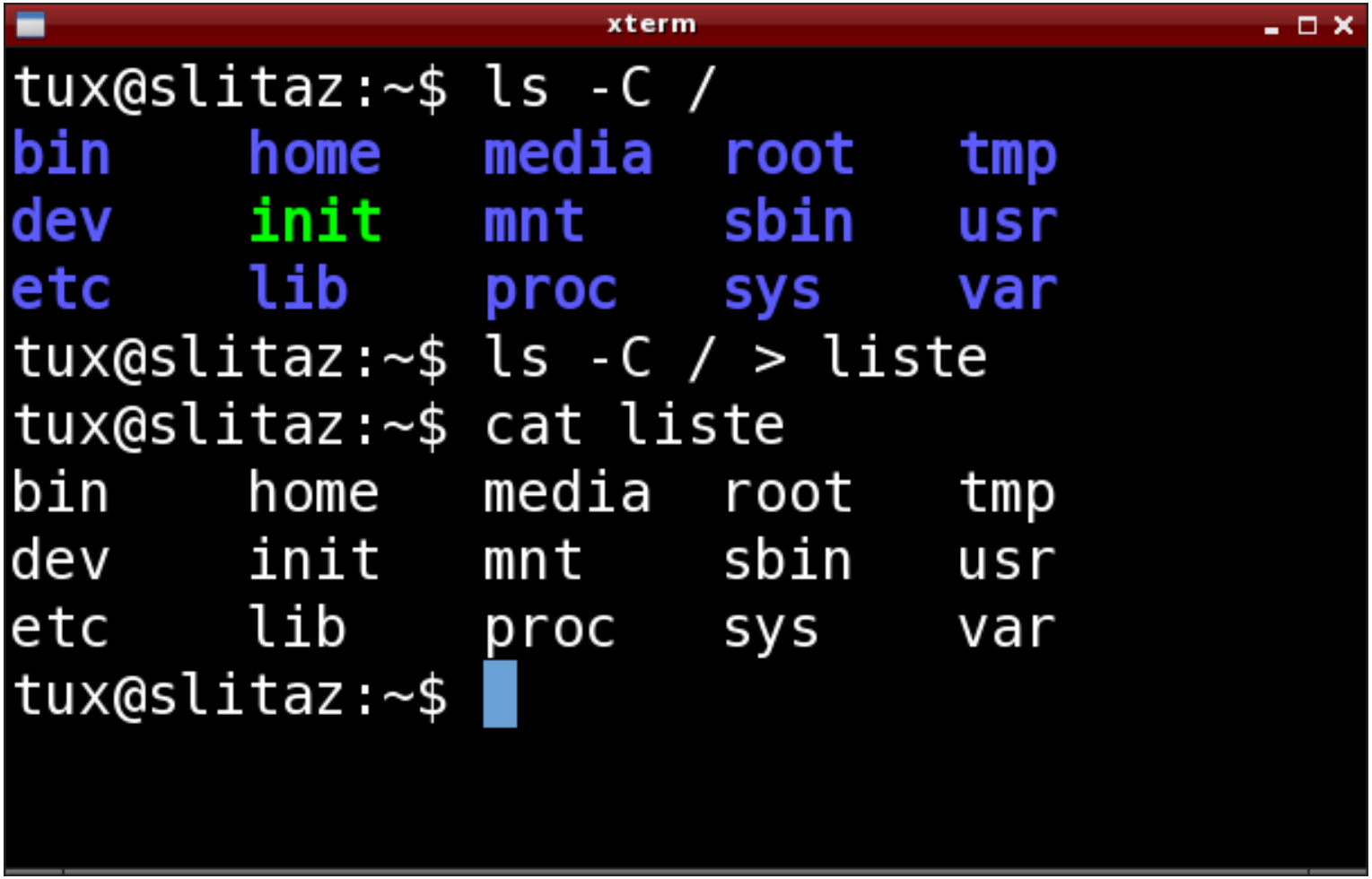
  ```
  $ tail -3 /etc/passwd
  ```

  affichera les trois dernières lignes du fichier `/etc/passwd` sur la sortie standard.

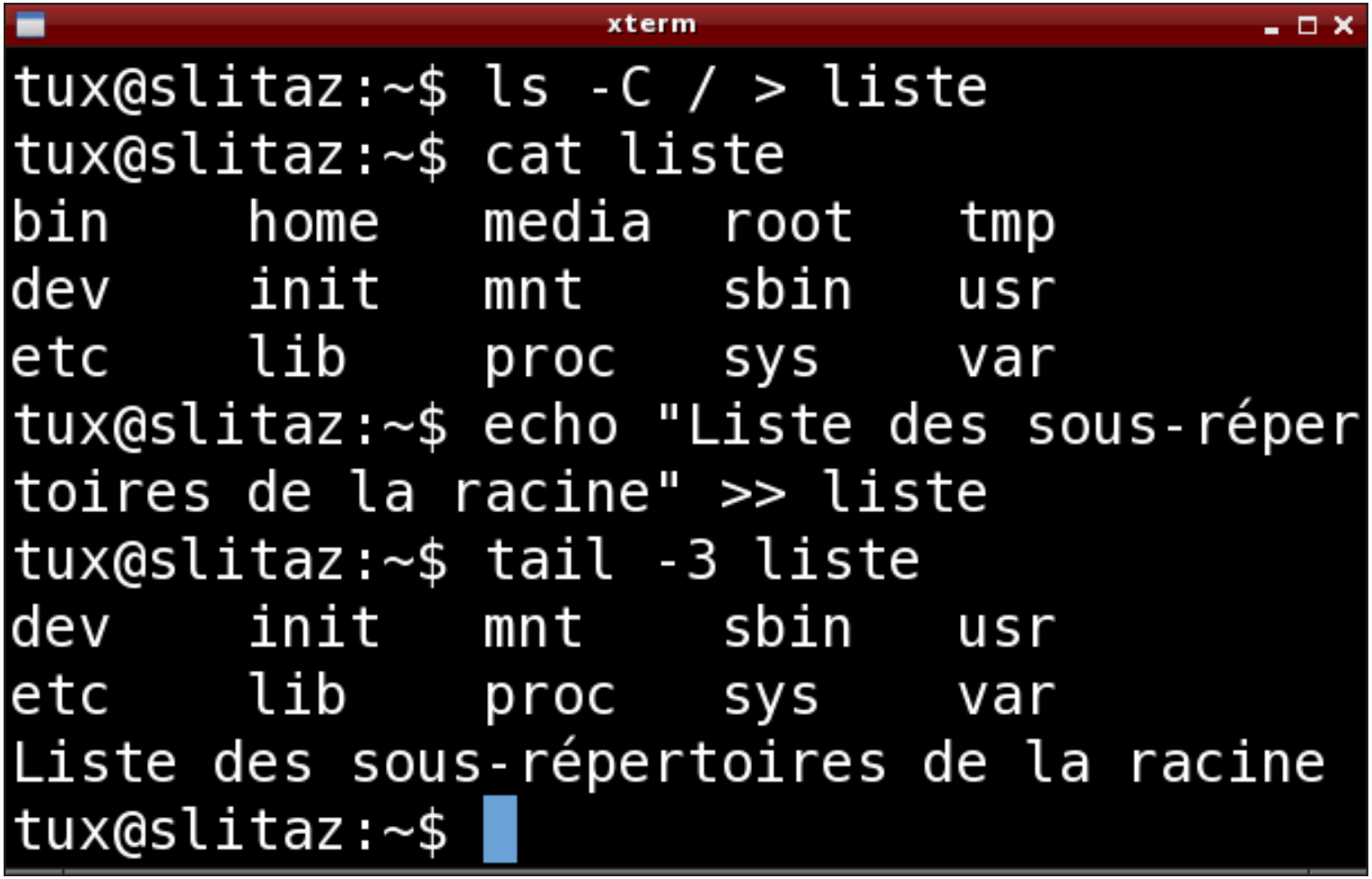
Quelques opérateurs du *shell*

`ESP, TAB`	séparateurs
`ENTER`	envoi d'une expression
`\`	caractère d'échappement
`&`	lance un processus en tâche de fond
`;`	séparateur entre deux expressions
`<`	redirection d'entrée
`>`	redirection de sortie
`|`	concaténation d'entrées-sorties dans un tube
`||`	ou logique
`&&`	et logique

Rediriger la sortie standard vers un fichier



Ajouter la sortie standard à la fin d'un fichier



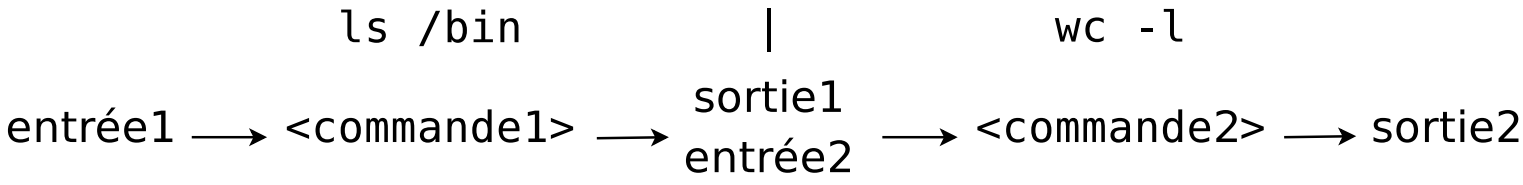
Connecter des commandes avec un tube (*pipe*)

- On veut afficher le nombre de fichiers du répertoire `/bin` grâce aux commandes `ls` et `wc` (*word count*). Première solution :

```
$ ls /bin > temp; wc -l < temp; rm -f temp
15
```

  L'inconvénient est que l'on écrit la sortie de la commande `ls` dans un fichier temporaire avant de traiter celui-ci avec `wc` puis de le supprimer.

- La plupart des commandes *shell* sont des filtres, c'est-à-dire des programmes qui lisent un flux sur l'entrée standard et écrivent sur la sortie standard. Le principe du tube est de combiner la sortie d'un filtre (ici `ls /bin`) avec l'entrée d'un autre (`wc -l`) :



Quelques raccourcis clavier

Ctrl+C	destruction d'un processus
Ctrl+Z puis bg	interruption puis mise en tâche de fond d'un processus
Ctrl+L	efface le contenu de la console (équivalent à `clear`)
Ctrl+D	sortie du (sous-)*shell* (équivalent à `exit`)
flèches haut/bas	navigation dans l'historique des commandes
`TAB`	complétion automatique

Quelques commandes utiles

`find`, `locate`	retrouver des fichiers
`grep`	chercher des motifs dans un fichier
`tar`	utilitaire d'archivage
`gzip`, `bzip2`	utilitaires de compression

Variables d'environnement du *shell*

- Les variables d'environnement sont des variables dynamiques utilisées par les différents processus du système d'exploitation.
- Les variables d'environnement du *shell* permettent d'obtenir des informations importantes telles que le *login* de l'utilisateur (stocké dans la variable $USER) ainsi que son répertoire de connexion ($HOME), la liste des répertoires dans lesquels aller chercher les exécutables des commandes externes ($PATH), etc.
- La commande `env` affiche la liste de toutes les variables d'environnement du *shell* avec leurs valeurs.
- On peut facilement déclarer une nouvelle variable :

```
$ var=valeur
$ echo $val
valeur
```