OCEANOMICS

Linux Initiation

Ateliers ABiMS 2017

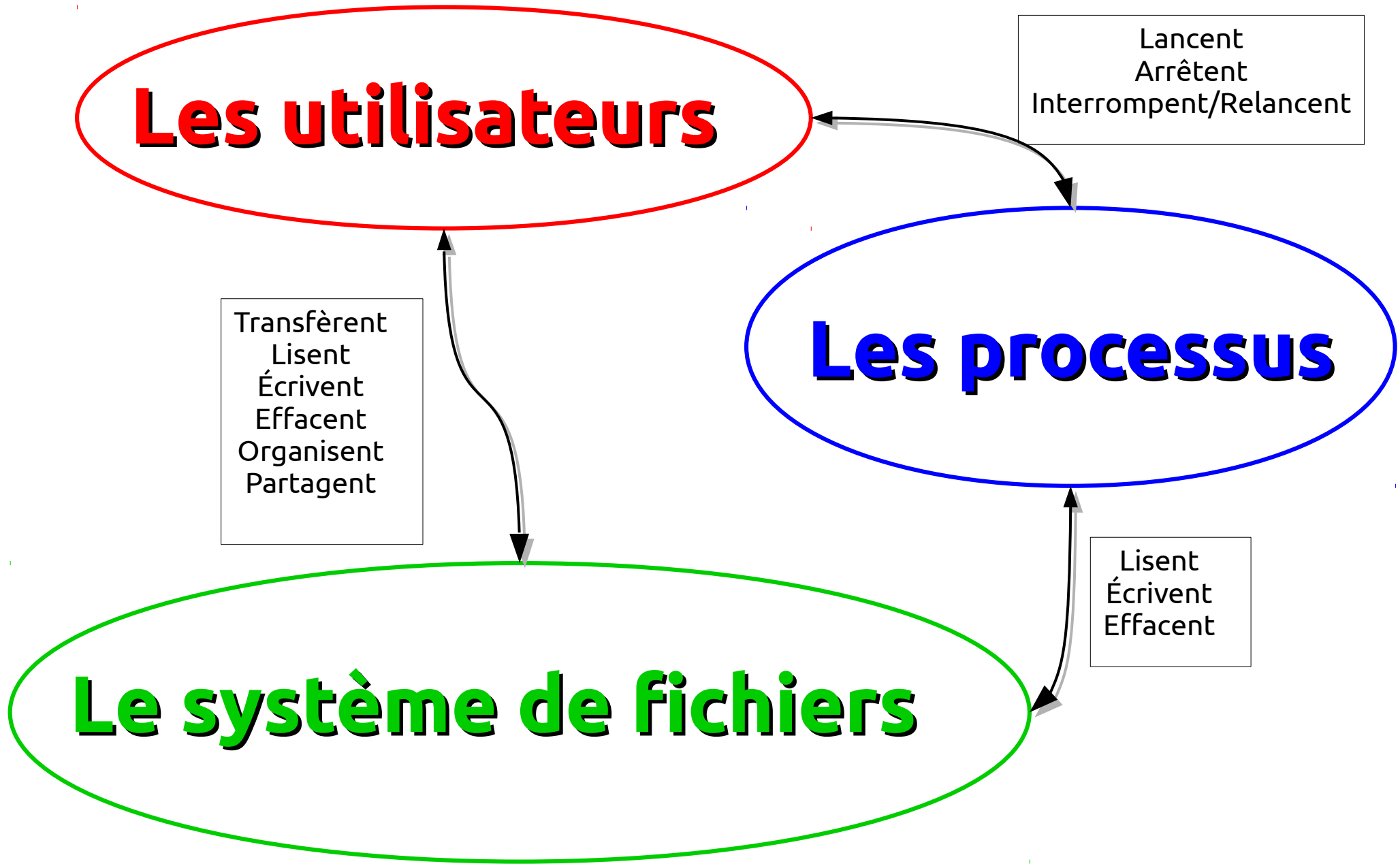
Mark Hoebeke
Philippe Bordron
Gildas Le Corguillé

Linux initiation | Objectifs de l'atelier

Mon {collègue|concurrent} m'a parlé d'un outil super-intéressant pour analyser des données qui ressemblent aux miennes. C'est un truc qui tourne sous *Linusque* et est installé sur un serveur quelque part.

Comment dois-je faire pour :

- **me** connecter au serveur ?
- transférer **mes fichiers** de données sur la machine ?
- lancer le **programme** ?
- lui dire où se trouvent les **fichiers** avec les données qu'il doit traiter ?
- lui dire où écrire les **fichiers** avec les résultats qu'il va générer ?
- organiser tous ces **fichiers** dans des **dossiers** pour m'y retrouver par la suite ?
- lancer «en parallèle» plusieurs **programmes** (ou plusieurs «exemplaires» du même programme) avec un paramètre ou des données différentes ?
- arrêter un exemplaire du **programme** s'il prend trop de temps ?
- *(ne pas)* partager **mes fichiers** de données ou de résultats avec **mes** {collègues|concurrents} ?

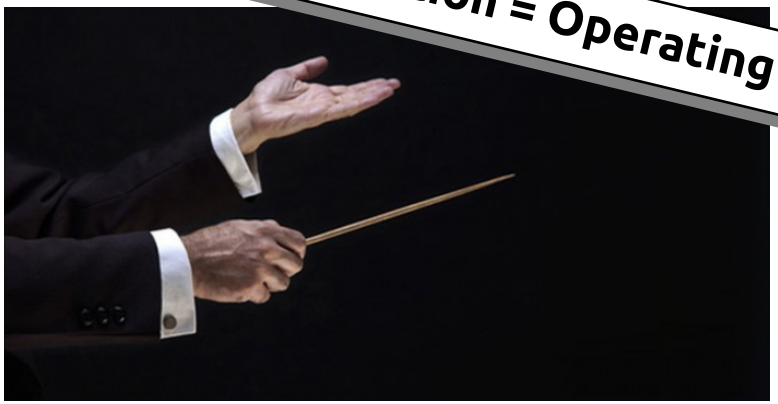

Les utilisateurs
Lancent
Arrêtent
Interrompent/Relancent
Les processus
Transfèrent
Lisent
Écrivent
Effacent
Organisent
Partagent
Lisent
Écrivent
Effacent
Le système de fichiers

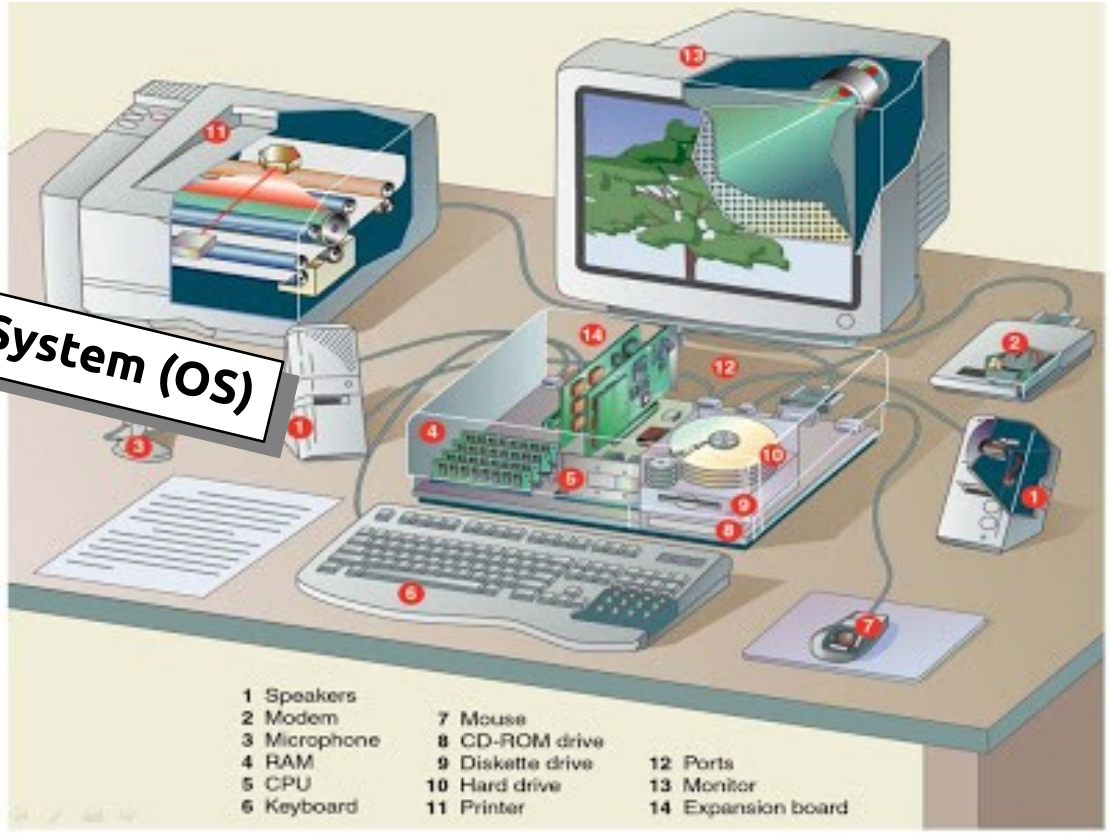
Linux initiation | Plan

Linux initiation | Plan

Rôle d'un système d'exploitation

Système d'exploitation = Operating System (OS)





Un OS est un programme «privilégié» chargé au démarrage de la machine qui :

- charge en mémoire les autres programmes (applications),
- leur alloue des ressources (mémoire, temps CPU, espace disque),
- gère leur communication (entrées/sorties) avec les périphériques (écran, clavier,souris,réseau,imprimante…
- met fin à leur exécution,
- récupère les ressources allouées.

Pourquoi Linux ?

- Multi-tâches **préemptif** & multi-utilisateurs
  - Robustesse & stabilité éprouvées depuis 1994
- *Open-source* **et gratuit**
  - Son code est librement copiable, modifiable, re-distribuable
- Proposant une logithèque fournie :
  - Bureautique : LibreOffice
  - Utilisation d'internet : navigateurs (Firefox, Chrome), messagerie (Thunderbird, Evolution)
  - Multimedia : lecteurs audio/video (VLC, Totem)
  - Graphisme : manipulation d'images (Gimp), modélisation 3D (Blender)
  - Développement logiciel : langages (Python, Java, C/C++…), environnements (Eclipse, IDLE, PyDev, DDD)
- Y compris dans le domaine scientifique :
  - Bioinformatique : blast, emboss, phylip, mafft, clustal, trimal...

Distributions Linux

Une distribution Linux comprend :

- Une variante du noyau Linux.
- Une palette de programmes pré-emballés sous forme de *packages*
- Des outils d'administration facilitant l'installation et le maintien à jour et la migration des *packages.*

Les principales différences entre distributions sont :

- Sur le plan technique :
  - Le format utilisé pour le *packaging* des programmes.
  - Les outils pour la gestion des *packages.*
- Au niveau du modèle économique :
  - La nature du support : communautaire vs. commercial.
  - La nature des licences des outils inclus dans la distribution.

Linux initiation | Plan

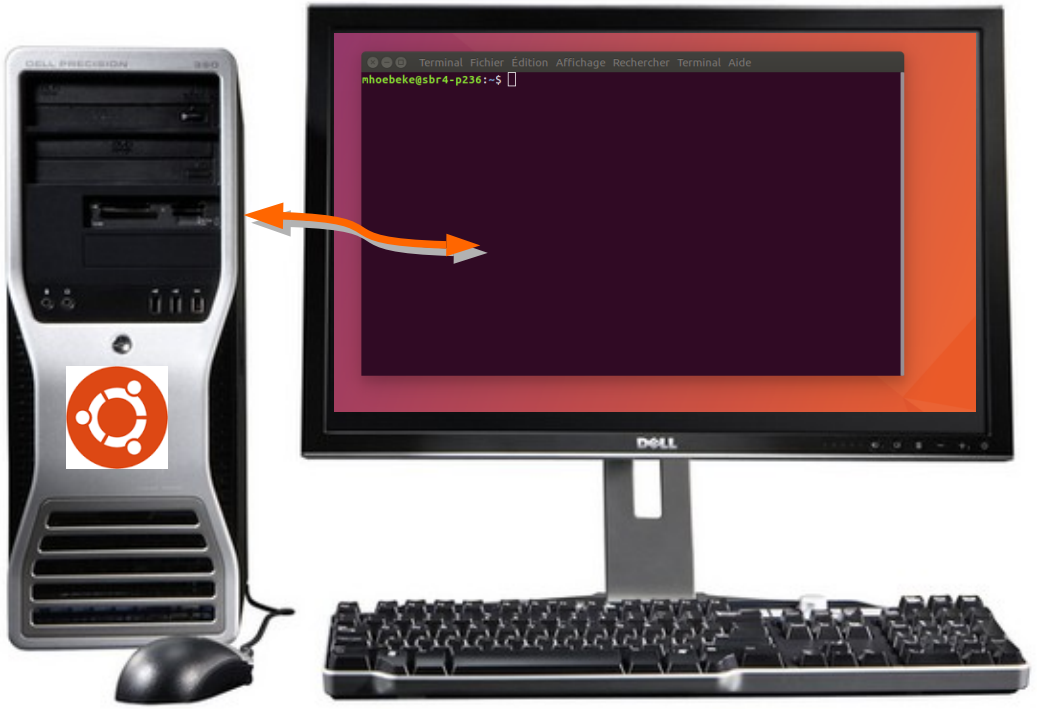
Connexion | Le terminal

Le terminal :

- Un moyen pour «dialoguer» avec **une machine** en utilisant une **ligne de commande** dans le contexte d'une **session**

Sessions locales :

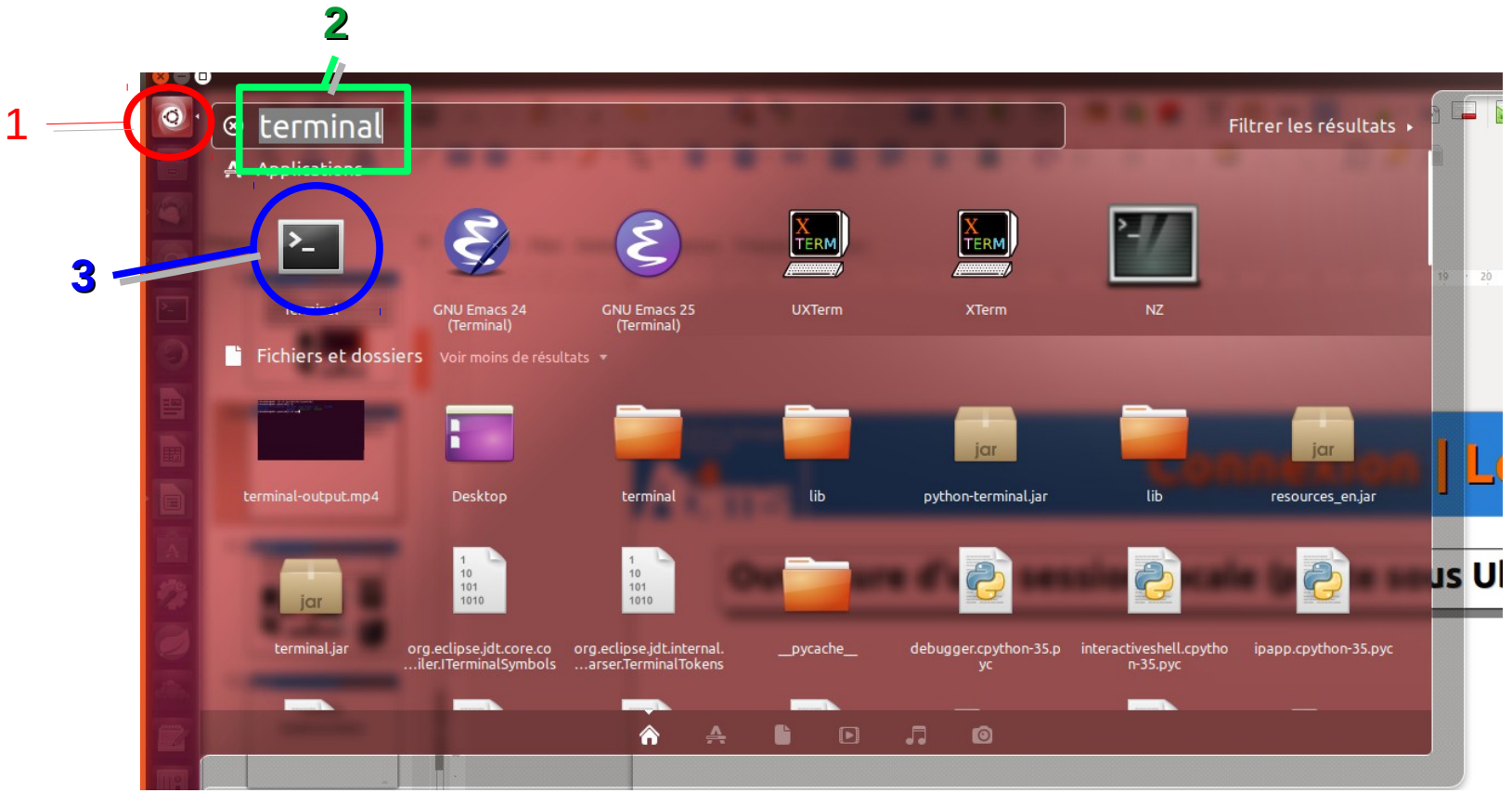
- Pour exécuter des commandes qui tournent sur son propre poste de travail.



Connexion | Le terminal

Ouverture d'une session locale (poste sous Ubuntu) :

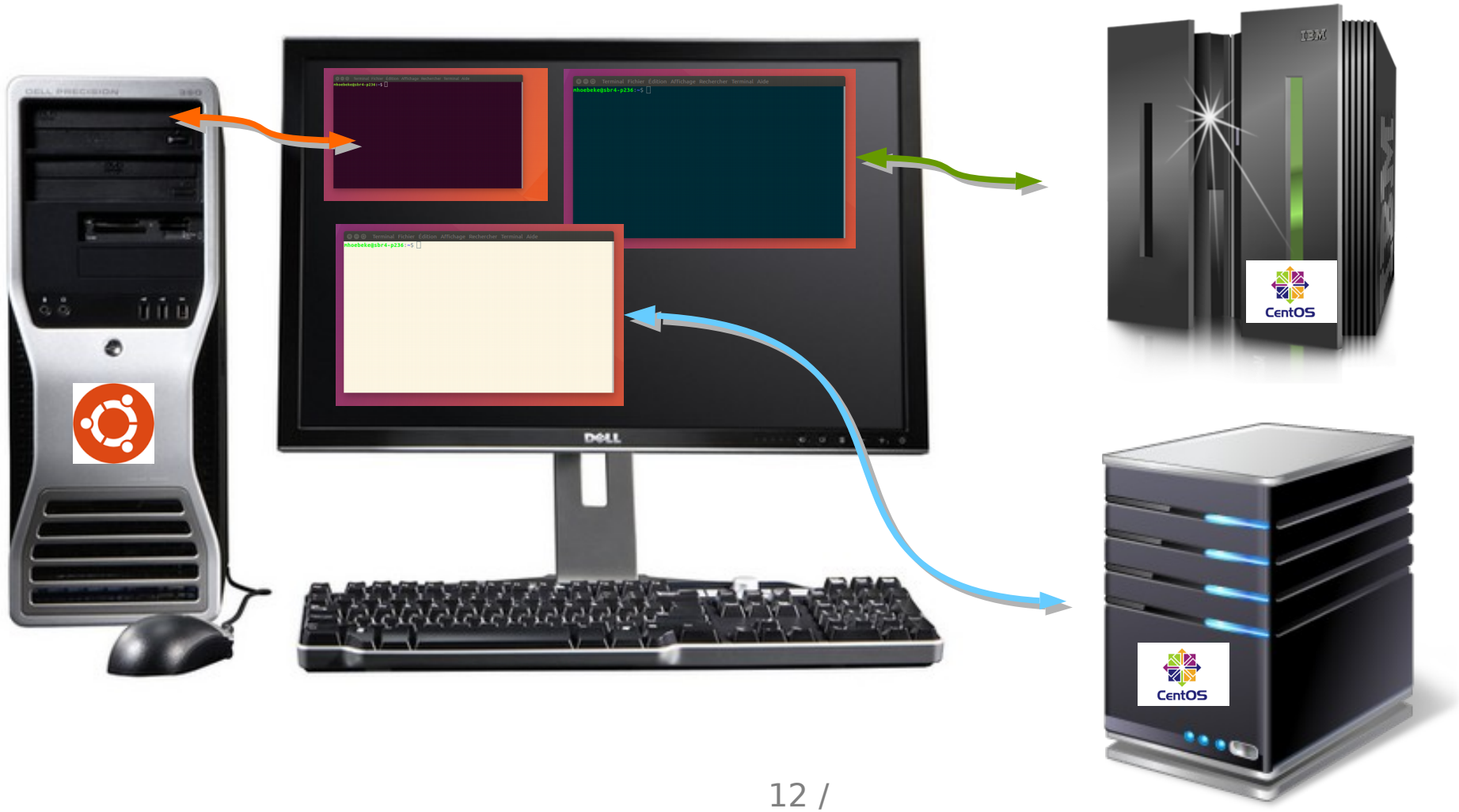
- Méthode « digitale » : combinaison **Ctrl+Alt+T**
- Méthode « murine » :



Connexion | Le terminal

Sessions distantes :

- Pour travailler sur une machine accessible en réseau.
- Si possible en ayant la possibilité d'ouvrir des fenêtres supplémentaires à partir de la machine distante

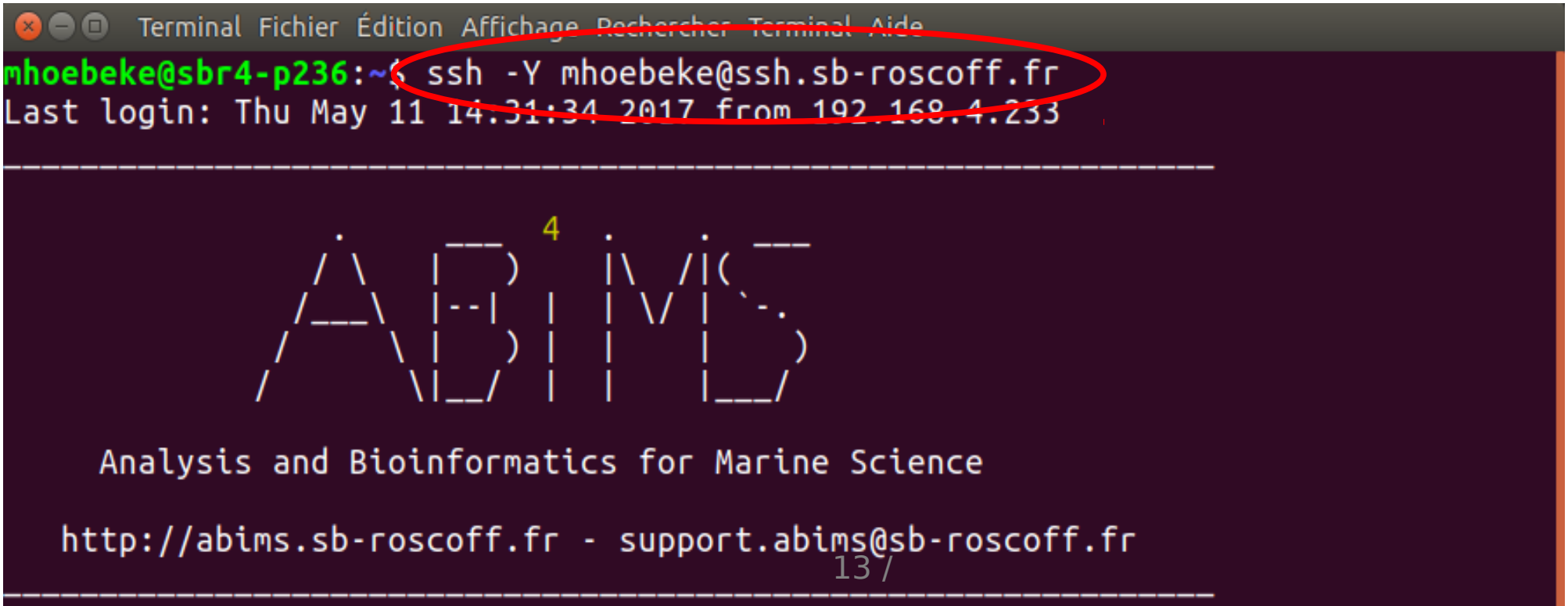


Connexion | Session distante & SSH

Utilisation du protocole Secure Shell *(SSH)*

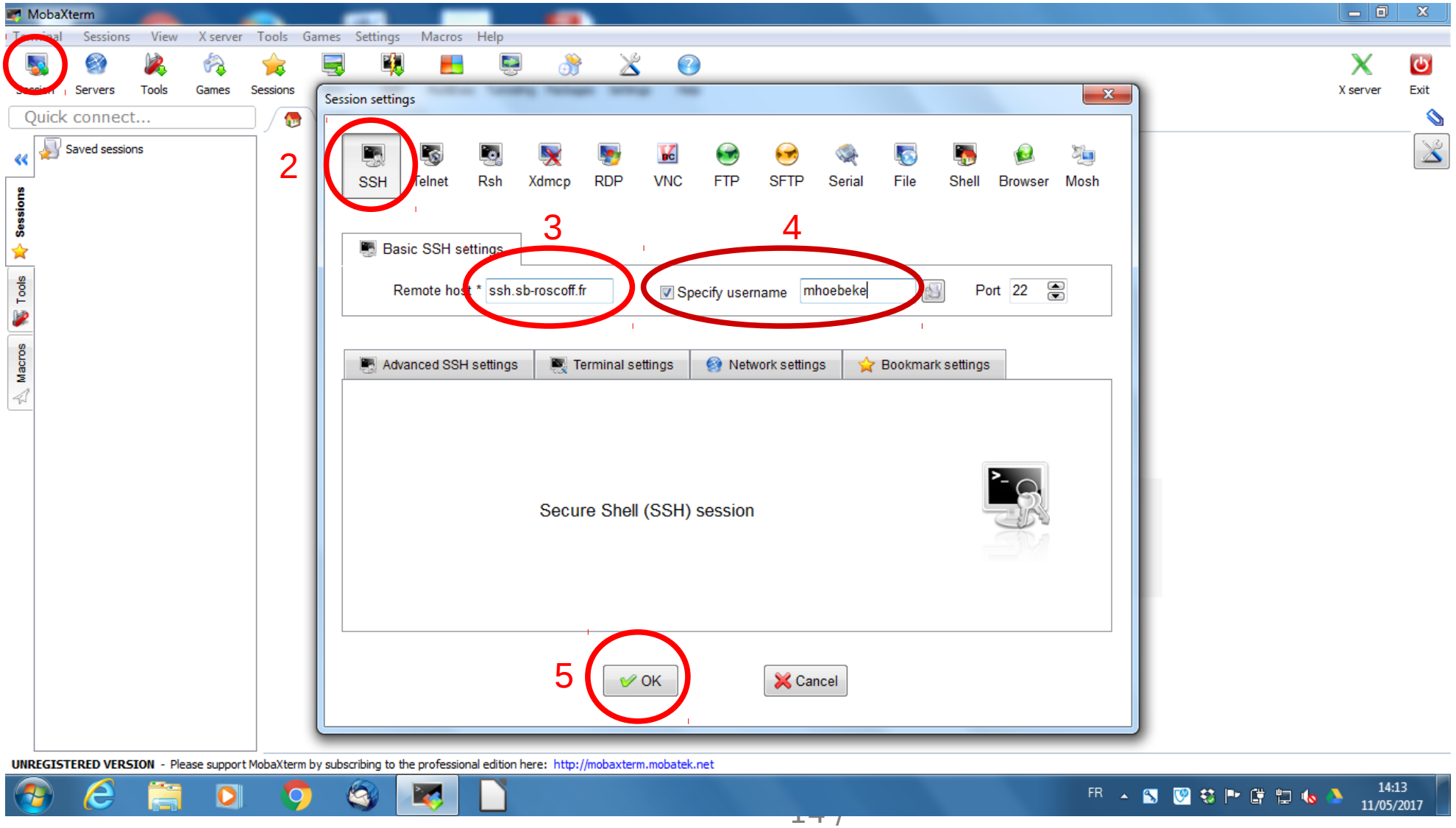
- Protocole de communication chiffré (sécurisé) établissant un canal pour l'échange de données entre deux machines.
- Son utilisation nécessite de connaître :
  - Le nom (ou l'adresse) de la machine cible,
  - Le nom d'utilisateur (login) sur la machine cible.

Utilisation de SSH à partir d'une session active (Linux)



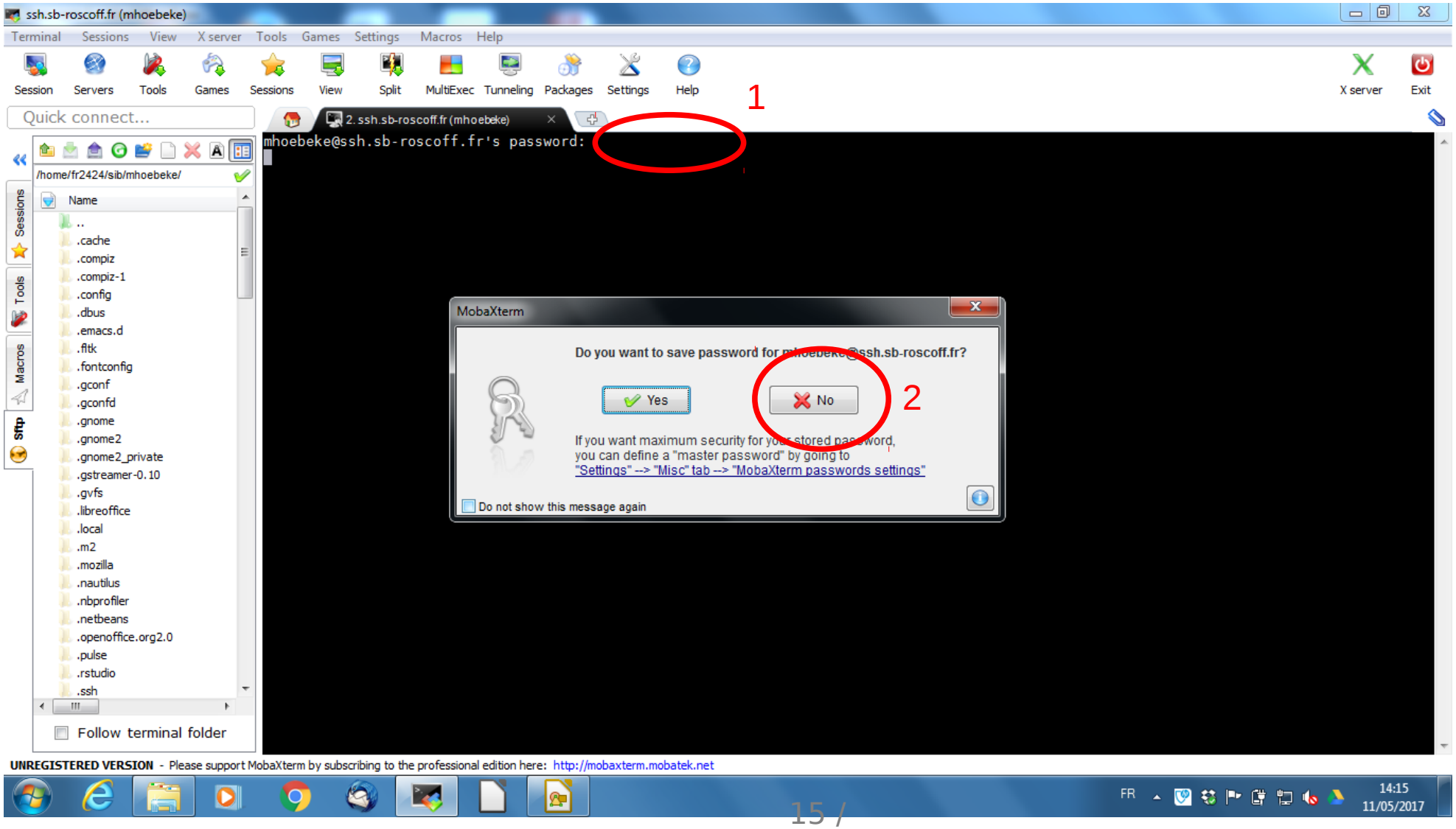
Connexion | Session distante & SSH

Utilisation de SSH à partir d'un poste sous Windows à l'aide de MobaXTerm



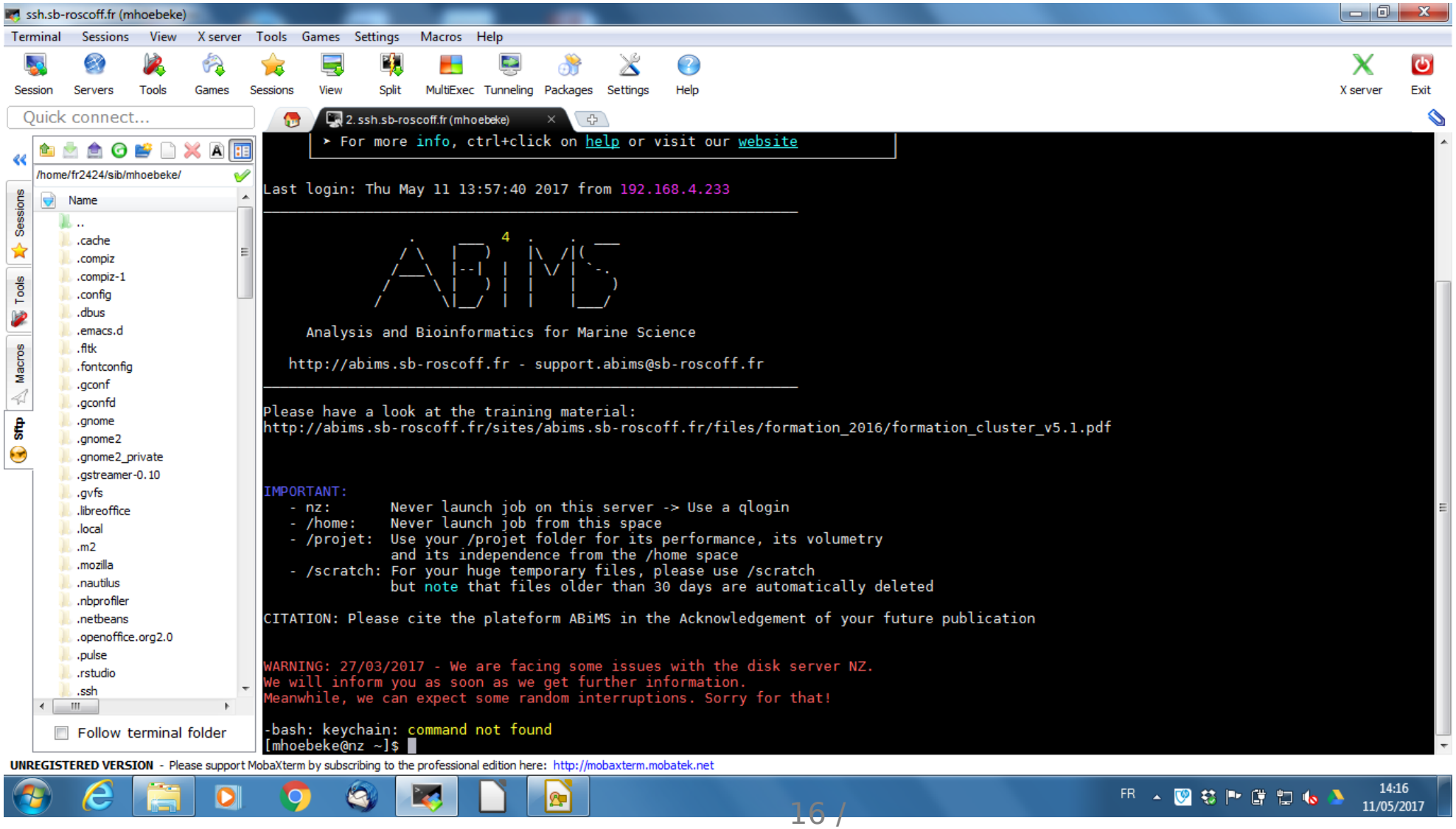
Connexion | Session distante & SSH

Utilisation de SSH à partir d'un poste sous Windows à l'aide de MobaXTerm



Connexion | Session distante & SSH

Utilisation de SSH à partir d'un poste sous Windows à l'aide de MobaXTerm



Interlude

- Copie des fichiers pour les TP
- Visualisation du support de cours

```
1  [stage01@nz~]$ cdprojet
2  [stage01@nz~]$ cp -r /tmp/Linux-Initiation .
3  [stage01@nz~]$ cd Linux-Initiation
4  [stage01@nz~]$ ls
     acteur.csv  cours        insulin.fas    insulin_vs_nt.blast
     tmp
5  [stage01@nz~]$ evince cours/Linux-Initiation-slides.pdf &
```

1 *Teleport :* **c**hange **d**irectory (to current user's project directory)

2 *Clone :* **co**py courseware (from `/tmp/Linux-Initiation` directory to current directory)

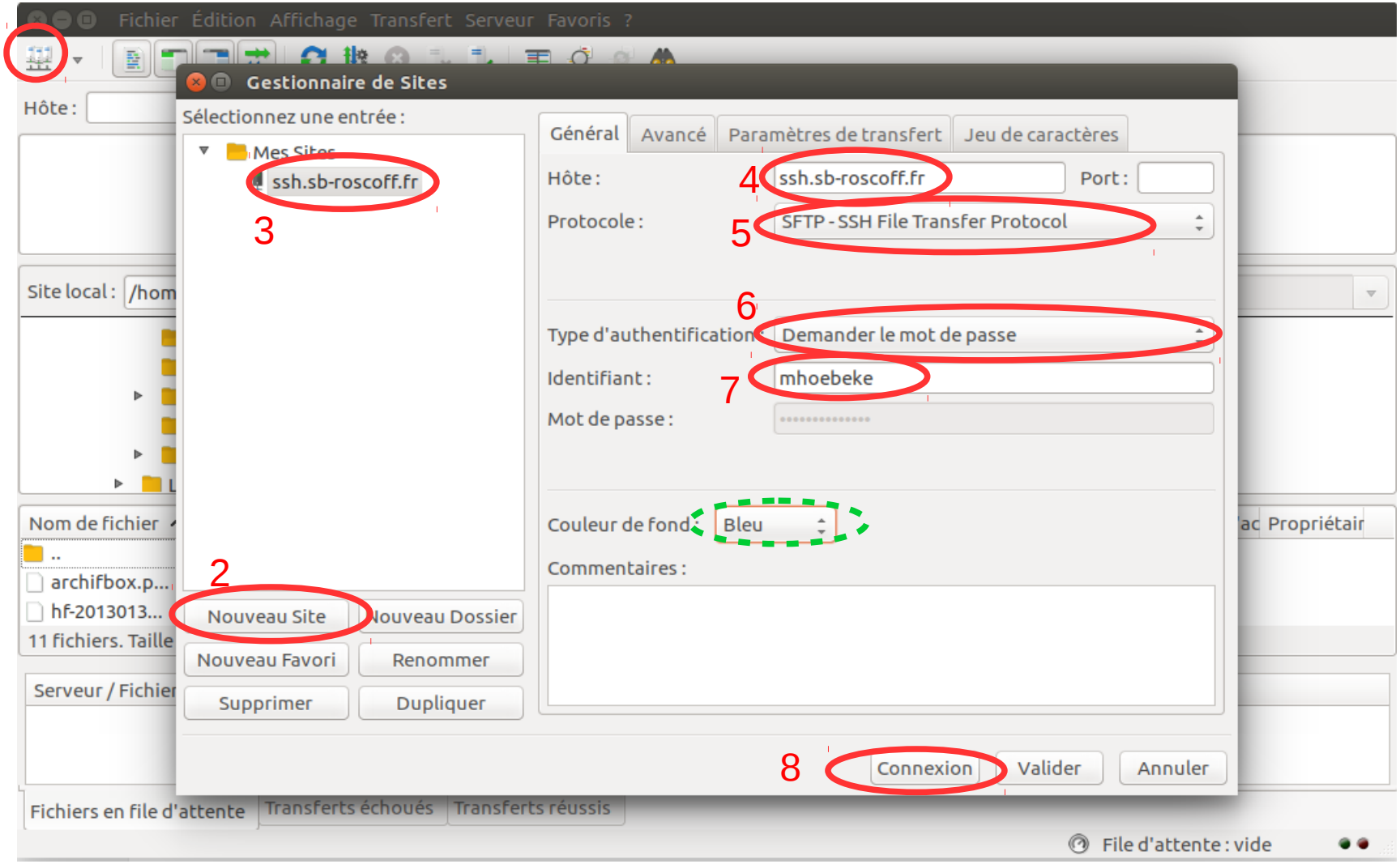
3 *Teleport again : **c**hange **d**irectory* (to freshly copied directory)

4 *Look around :* **li**st (contents of current directory – sanity check)

5 *Cast Spell :* run PDF file viewer (**evince**) and keep control

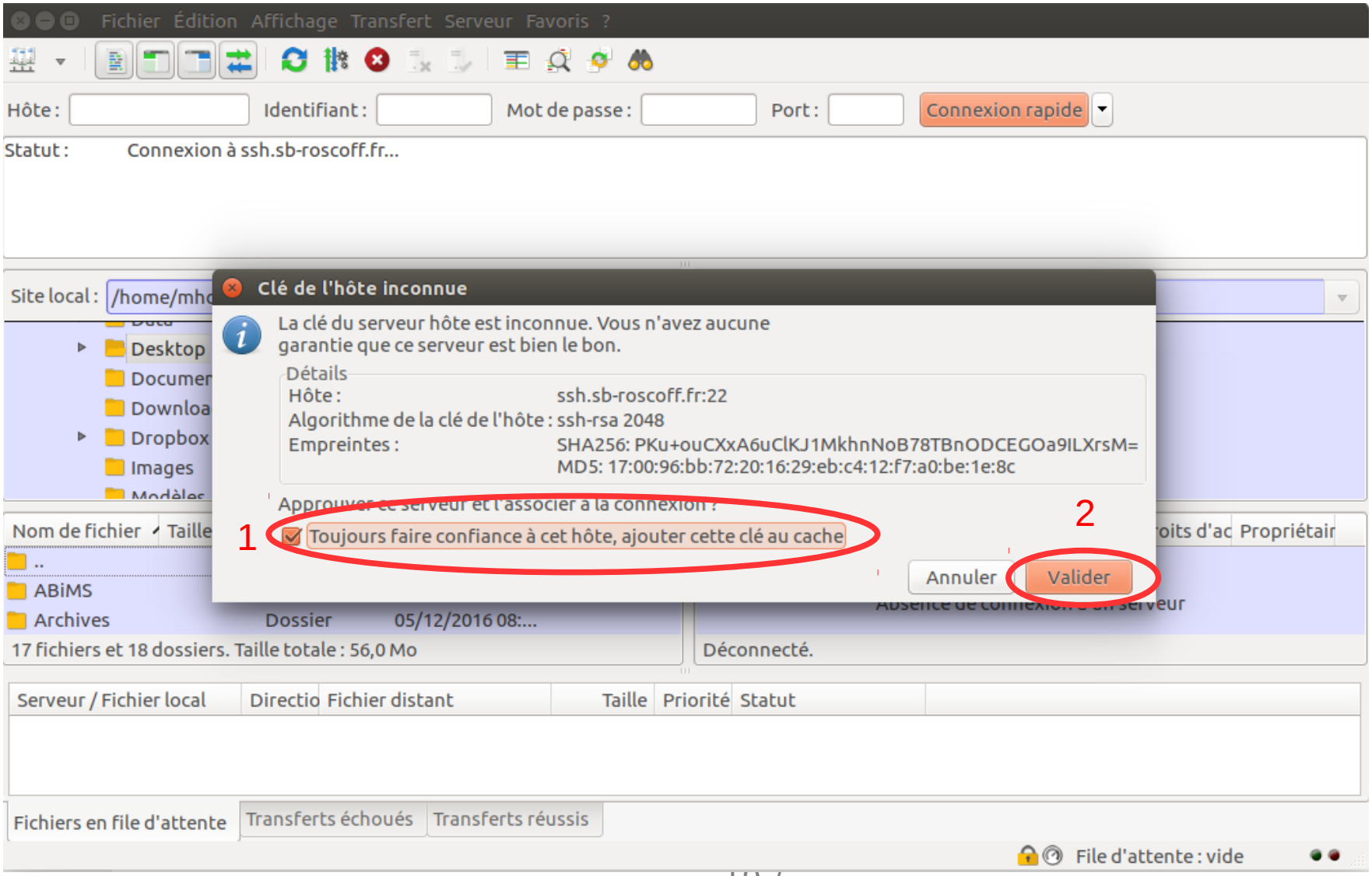
Transfert de fichiers avec FileZilla

Définition des paramètres de connexion
Établissement de la connexion



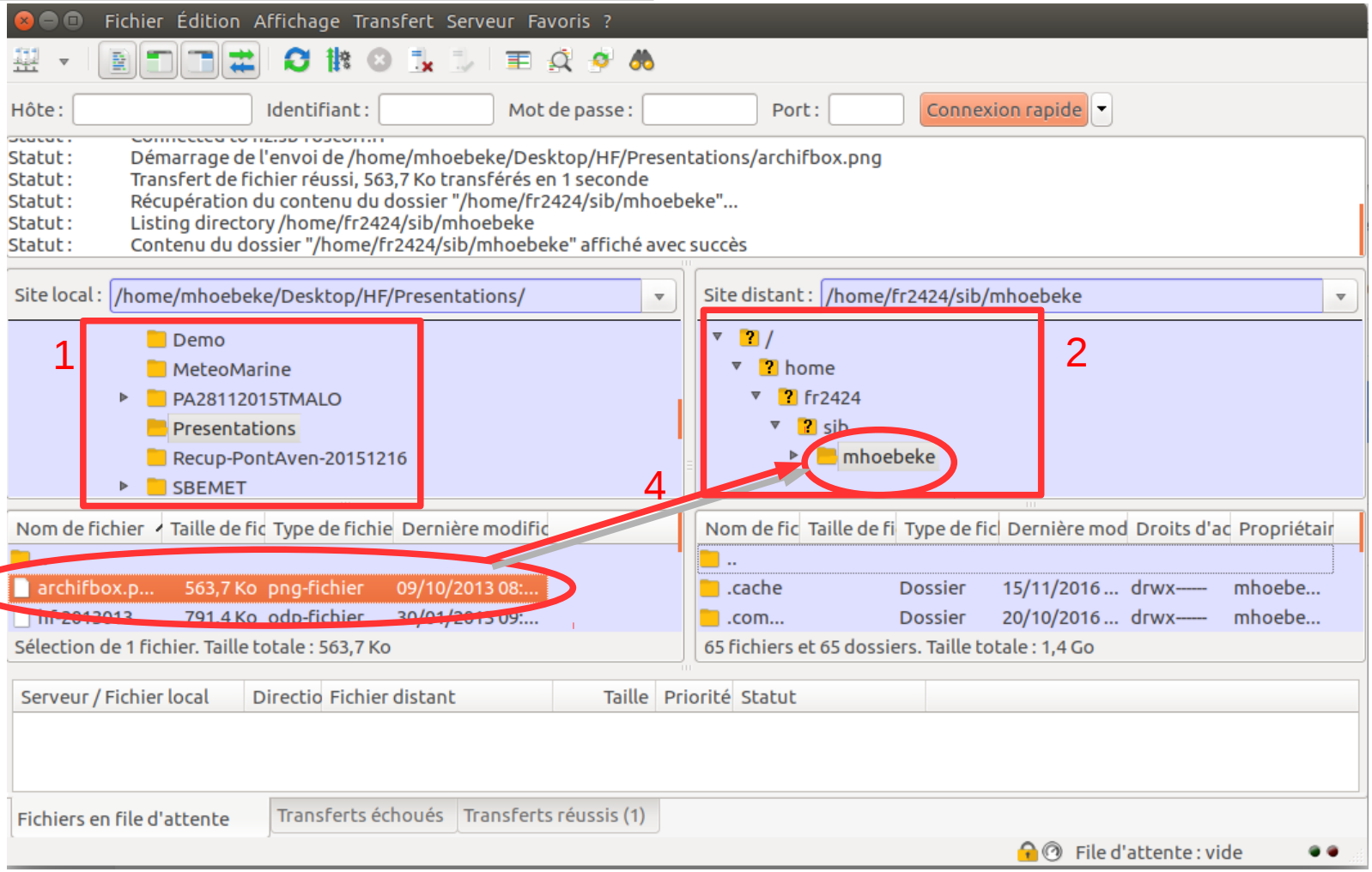
Transfert de fichiers avec FileZilla

Validation de la connexion



Transfert de fichiers avec FileZilla

Sélection du dossier source
Sélection du dossier destination
Sélection du fichier à transférer
Transfert par «glisser/déposer»



Linux initiation | Plan

Anatomie de la ligne de commande

```
[stage01@nz ~]$ head -n 20 insulin.fas  #print the first 20 lines
```

[stage01@nz ~] $ L'invite *(prompt) :* affiche le nom d'utilisateur courant **(stage01)**, le nom de la machine **(nz)**, le répertoire (a.k.a dossier, directory) courant **(~)**

head Le nom de la commande à exécuter (premier mot après l'invite)

-n 20 Une option de la commande **(-n)** éventuellement avec une valeur **(20).**

insulin.fas Un argument pour la commande

 # print (...) Des commentaires (ignorés par la commande)

- L'espace sert de séparateur entre les différents champs de la ligne de commande
- **La casse des caractères est importante** (**head** n'est pas la même chose que **HEAD**)
- Chaque commande dispose de son propre jeu d'options et d'arguments
- ***La touche Entrée (ou ↵) provoque l'exécution de la ligne de commande***

Ligne de commande | Aide en ligne

Chaque commande (qui se respecte) est documentée

Pour obtenir la documentation succincte sur la manière d'utiliser une commande, il est possible de lui ajouter l'option –**help** ou **-h**

```
[stage01@nz~]$ ls --help
Utilisation : ls [OPTION]... [FICHIER]...
Afficher des renseignements sur les FICHIERs (du répertoire actuel par
défaut).
```

Pour obtenir la documentation complète d'une commande, on peut utiliser la commande **man** *(*i.e. ***man****ual)* en lui spécifiant en argument le nom de la commande dont on veut obtenir la documentation.

```
[stage01@nz~]$ man ls
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information about the FILEs (the current directory by default).  Sort
       entries alphabetically if none of -cftuvSUX nor --sort is specified.
```

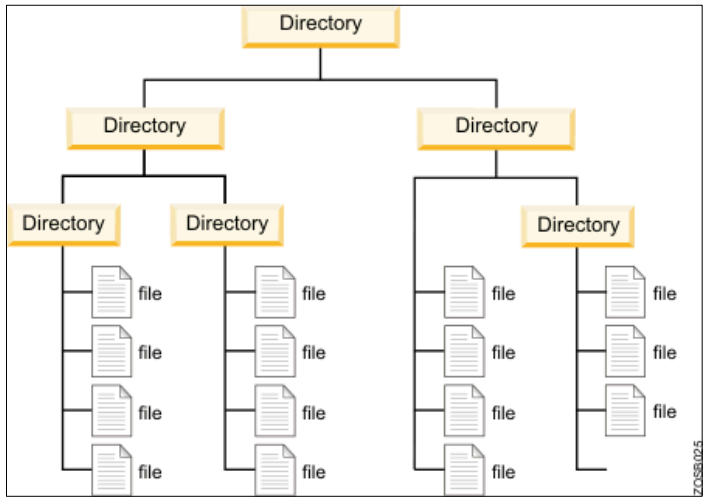
Linux initiation | Plan

Le système de fichiers | Utilité & concepts de base

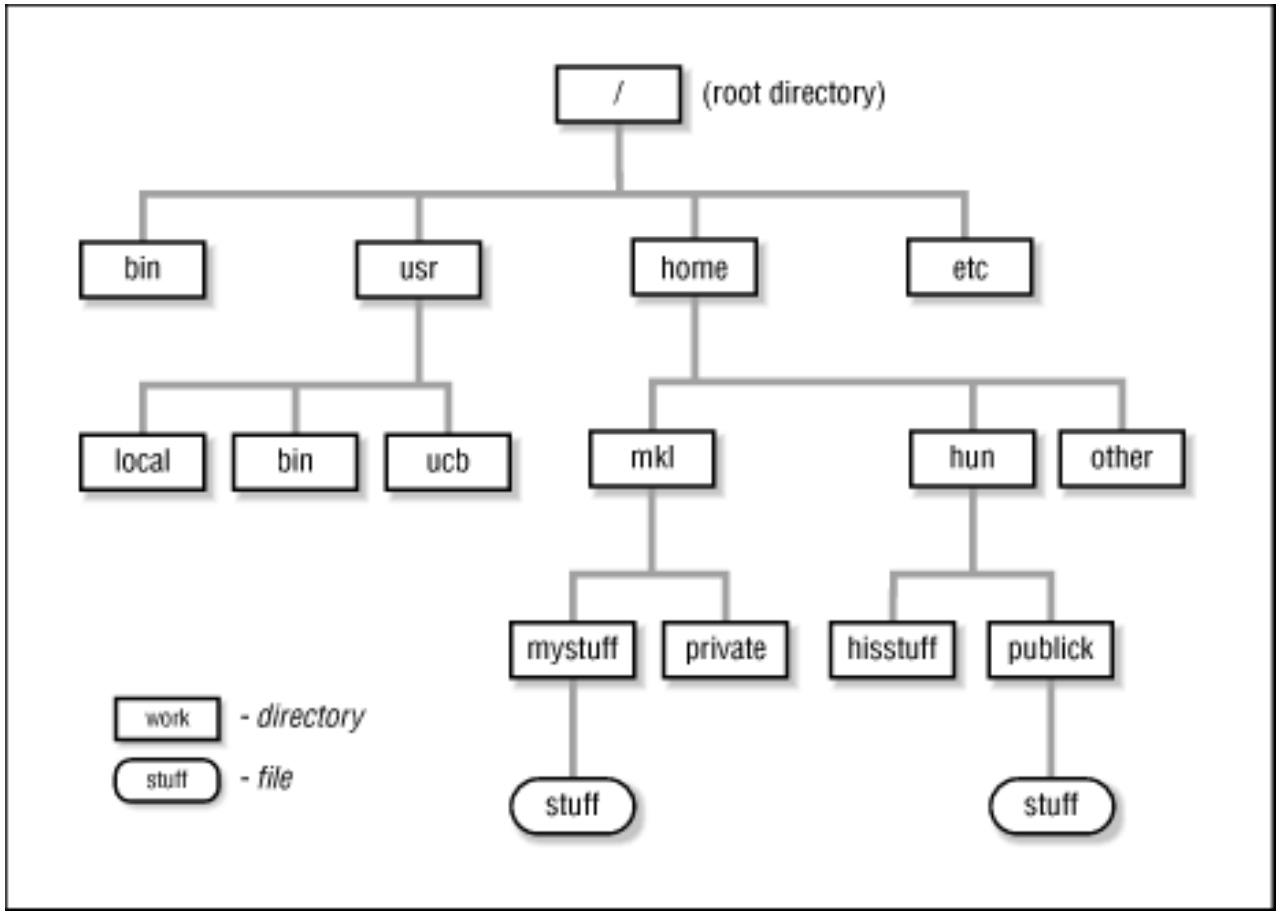
Dans la plupart des systèmes d'exploitation, les données sont stockées dans des **fichiers** (textes, images, tableaux, séquences, séries de mesures….). Très rapidement, le nombre de fichiers croît et il devient nécessaire de les organiser pour s'y retrouver, en les regroupant dans des **dossiers (répertoires, *directories*)**. Les dossiers peuvent être rangés dans d'autres dossiers, qui à leur tour peuvent être rangés dans d'autres dossiers, qui peuvent également être rangés...

Le système de fichiers | Arborescence

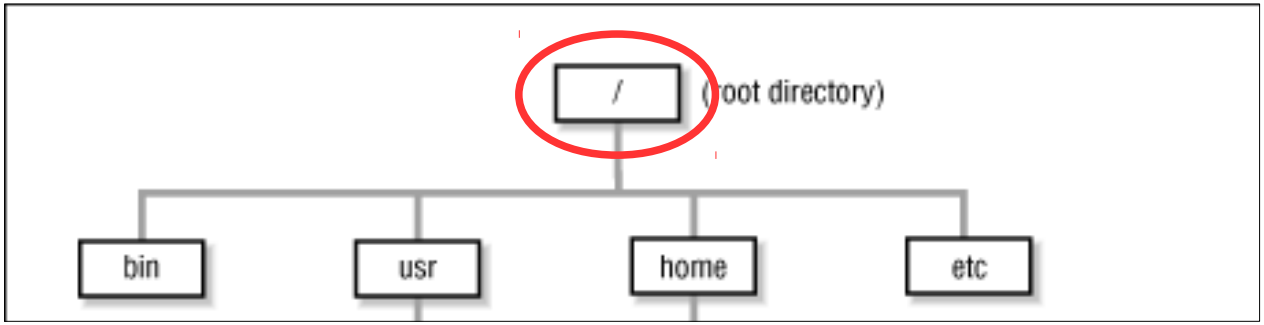
Le concept



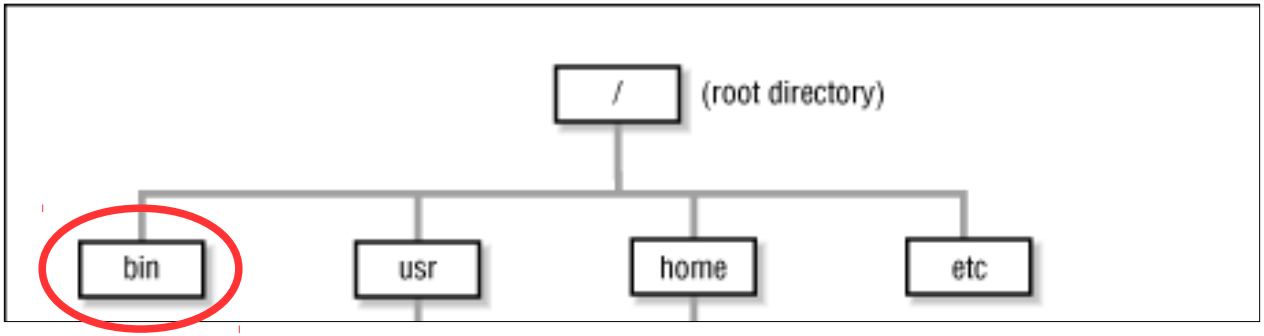
Sa concrétisation sous Linux (UNIX)



Le système de fichiers | Répertoires importants

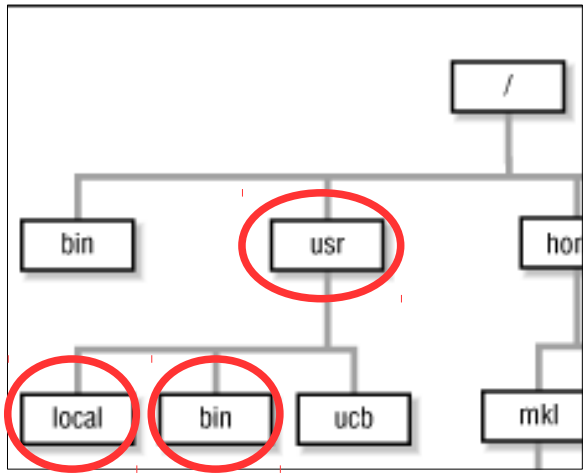


Slash* - répertoire racine - *root directory : c'est le point d'accroche **unique** de l'ensemble du système de fichiers. Le «chemin» vers n'importe quel répertoire ou fichier dans le système de fichier peut être construit à partir de cette racine.



/bin : répertoire contenant l'essentiel des commandes du système.

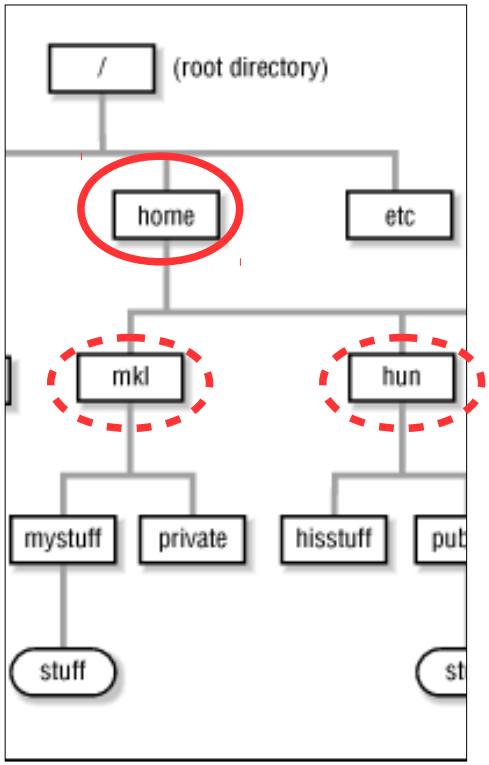
Le système de fichiers | Répertoires importants



/usr : répertoire contenant des sous-répertoires avec des outils plutôt destinés aux utilisateurs.

/usr/bin : répertoire avec des commandes (moins essentielles au fonctionnement du système que dans /bin).

/usr/local : répertoire avec des sous-répertoires contenant des outils spécifiquement installés sur «cette» machine (en particulier ***/usr/local/bin***)



/home : début de la sous-arborescence avec les données des utilisateurs.

Son organisation dépend du nombre d'utilisateurs enregistrés sur la machine :
- peu d'utilisateurs (postes de travail) : le répertoire de chacun se trouve directement sous ***/home***.
- beaucoup d'utilisateurs : les répertoires des utilisateurs sont rangés dans des sous-(sous-(sous-))répertoires.

À la SBR : l'organisation reflète l'organigramme des unités/équipes : **/home/fr2424/sib/mhoebeke** et ces répertoires utilisateur sont accessibles sur toutes les machines (Linux).

/tmp : répertoire où tout le monde peut lire et écrire des fichiers (mais seul le dépositaire peut en principe détruire ses propres affaires). Pratique pour s'échanger des fichiers entre utilisateurs **sur une même machine** (en faisant attention au volume).

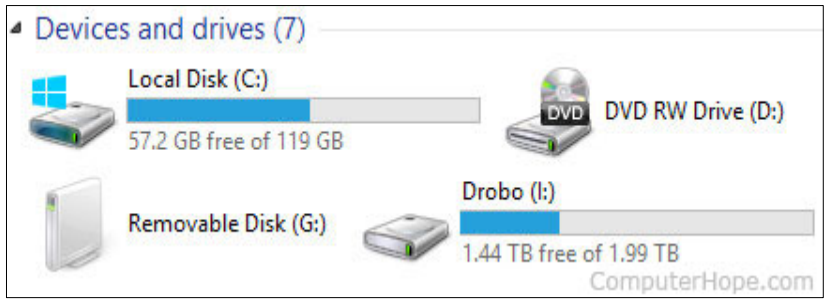
Spécificités de la SBR

/projet : répertoire contenant une arborescence calquée sur celle des ***/home*** *(unité/équipe/utilisateur)* destinée à accueillir les données des projets des utilisateurs devant être sauvegardés (jeu de donnés initiaux, résultats finals).

/scratch : répertoire contenant une arborescence calquée sur celle des ***/home*** *(unité/équipe/utilisateur)* destinée à accueillir les données de travail des calculs en cours ne nécessitant pas d'être sauvegardés. Les «vieux» fichiers sont automatiquement effacés passé un certain délai d'inactivité.

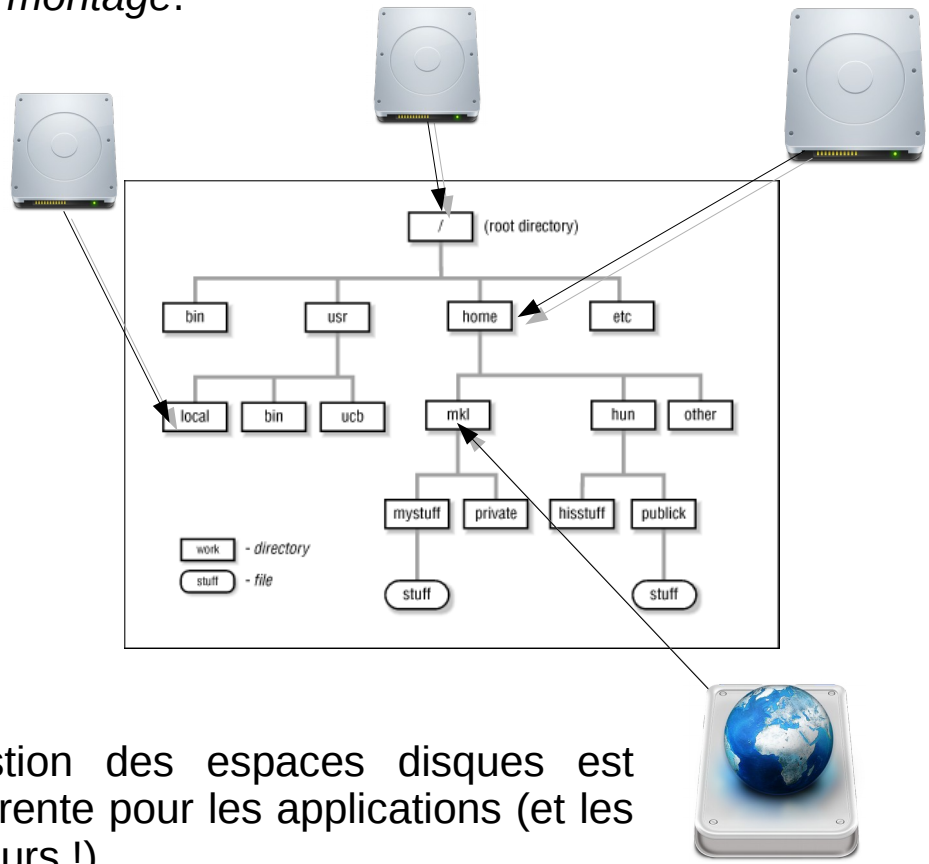
Le système de fichiers | Et les disques dans tout ça ?

Les systèmes de la famille Windows définissent une lettre par périphérique de stockage (disque, lecteur DVD, clef USB, lecteur réseau), conduisant à une **arborescence à racines multiples**.



Pour pouvoir utiliser un fichier, il faut déterminer sur quel lecteur il est physiquement stocké. L'extension de l'espace disponible par ajout de disques peut nécessiter des reparamétrages d'applications.

Les systèmes de la famille UNIX/Linux associent les périphériques (locaux ou distants) à des dossiers dans l'**arborescence à racine *unique,*** au travers de *points de montage*.



La gestion des espaces disques est transparente pour les applications (et les utilisateurs !).

Navigation dans le système de fichiers

L'exécution d'une commande se fait dans le contexte d'une **session** qui définit à chaque instant un **utilisateur courant** (l'utilisateur qui lance la commande) et un **répertoire courant** ou ***working directory*** (le répertoire à partir duquel la commande est lancée).

Lors de l'ouverture d'une session, le répertoire courant correspond au répertoire de connexion de l'utilisateur courant : son **home directory**

Qui suis-je (quel est l'utilisateur courant) ?

```
[stage01@nz ~]$ whoami
stage01
```

Où suis-je (quel est le répertoire courant) ?

```
[stage01@nz ~]$ pwd    # print working directory
/home/fr2424/stage/stage01
```

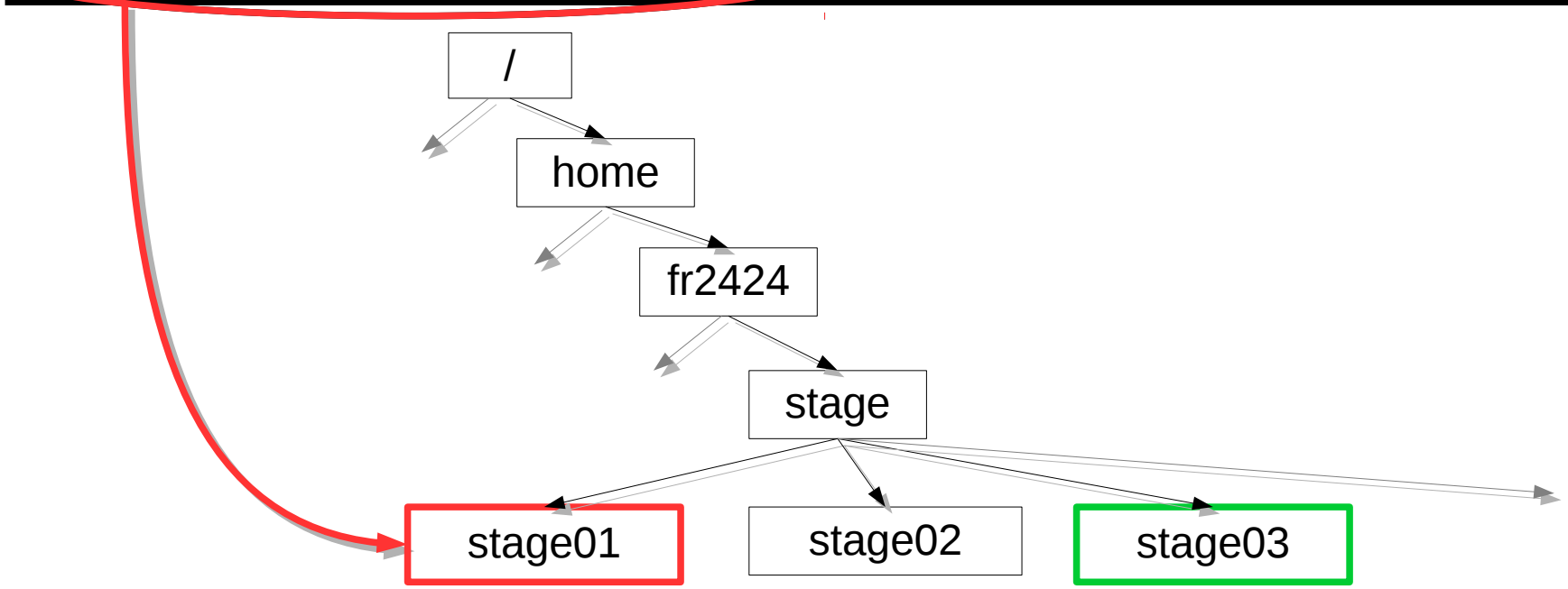
Qu'y a-t-il à l'endroit où je suis (quel est le contenu du répertoire courant) ?

```
[stage01@nz ~]$ ls    # list (contents of working directory)
Bureau  Documents  Images  Modèles  Musique  Public
Téléchargements  Vidéos
```

Navigation dans le système de fichiers

Changer de répertoire courant (a.k.a «se déplacer» dans l'arborescence)

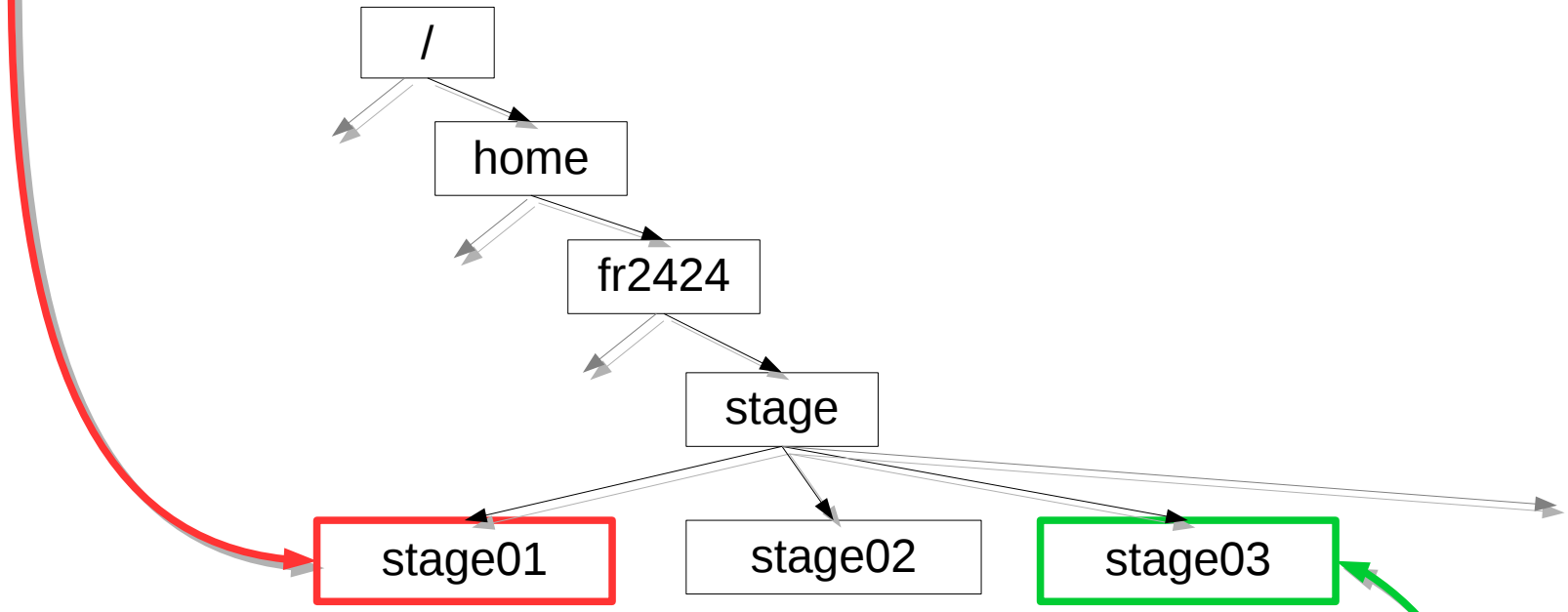
```
[stage01@nz ~]$ pwd    # print working directory
/home/fr2424/stage/stage01
```



Navigation dans le système de fichiers

Changer de répertoire courant (a.k.a «se déplacer» dans l'arborescence)

```
[stage01@nz ~]$ pwd    # print working directory
/home/fr2424/stage/stage01
```

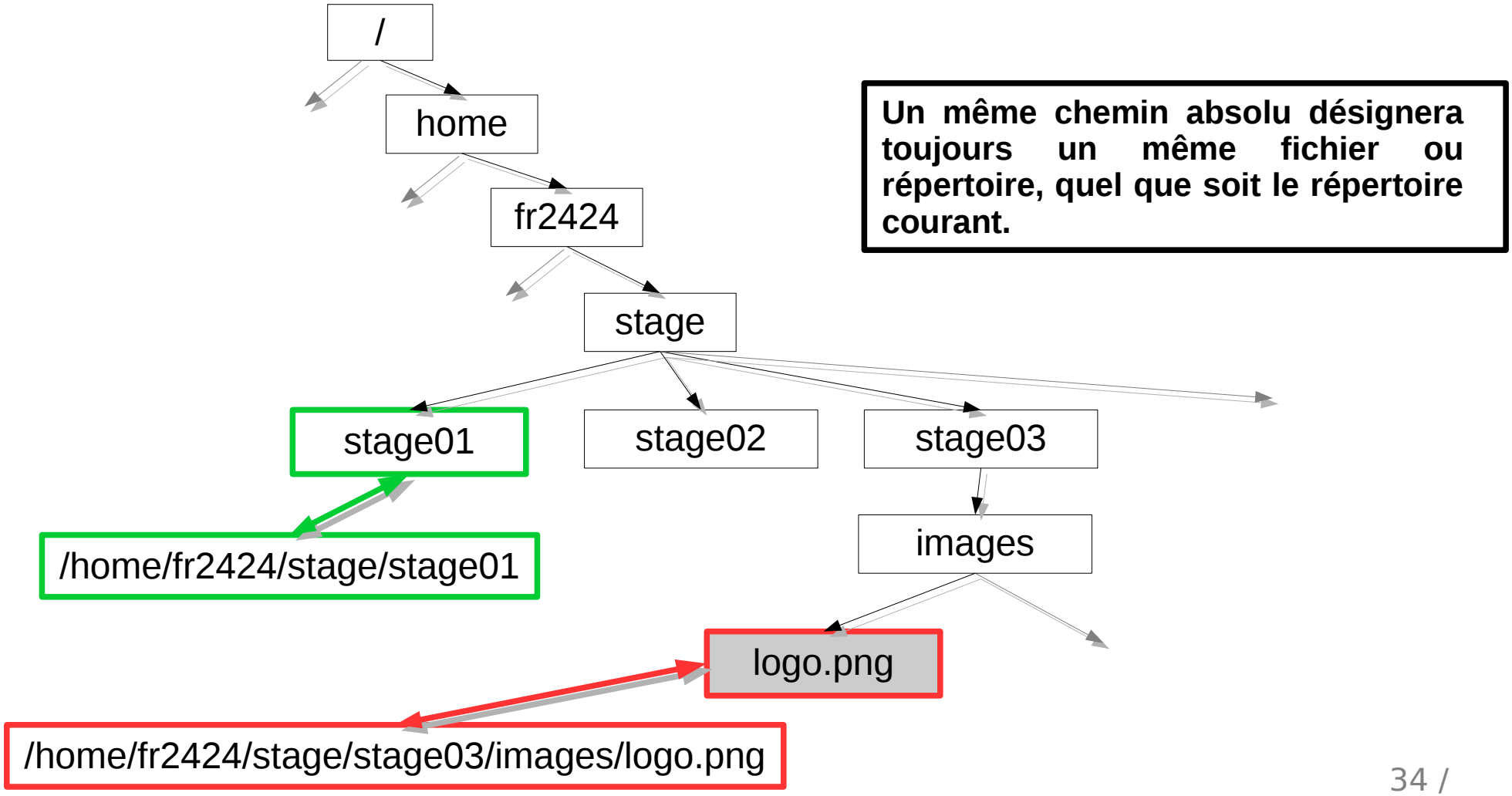



```
[stage01@nz ~]$ cd /home/fr2424/stage/stage03 # change dir
[stage01@nz ~]$ pwd
/home/fr2424/stage/stage03
```

Chemins absolus

La désignation des fichiers et des dossiers dans le système de fichiers se fait à l'aide de **chemins.**

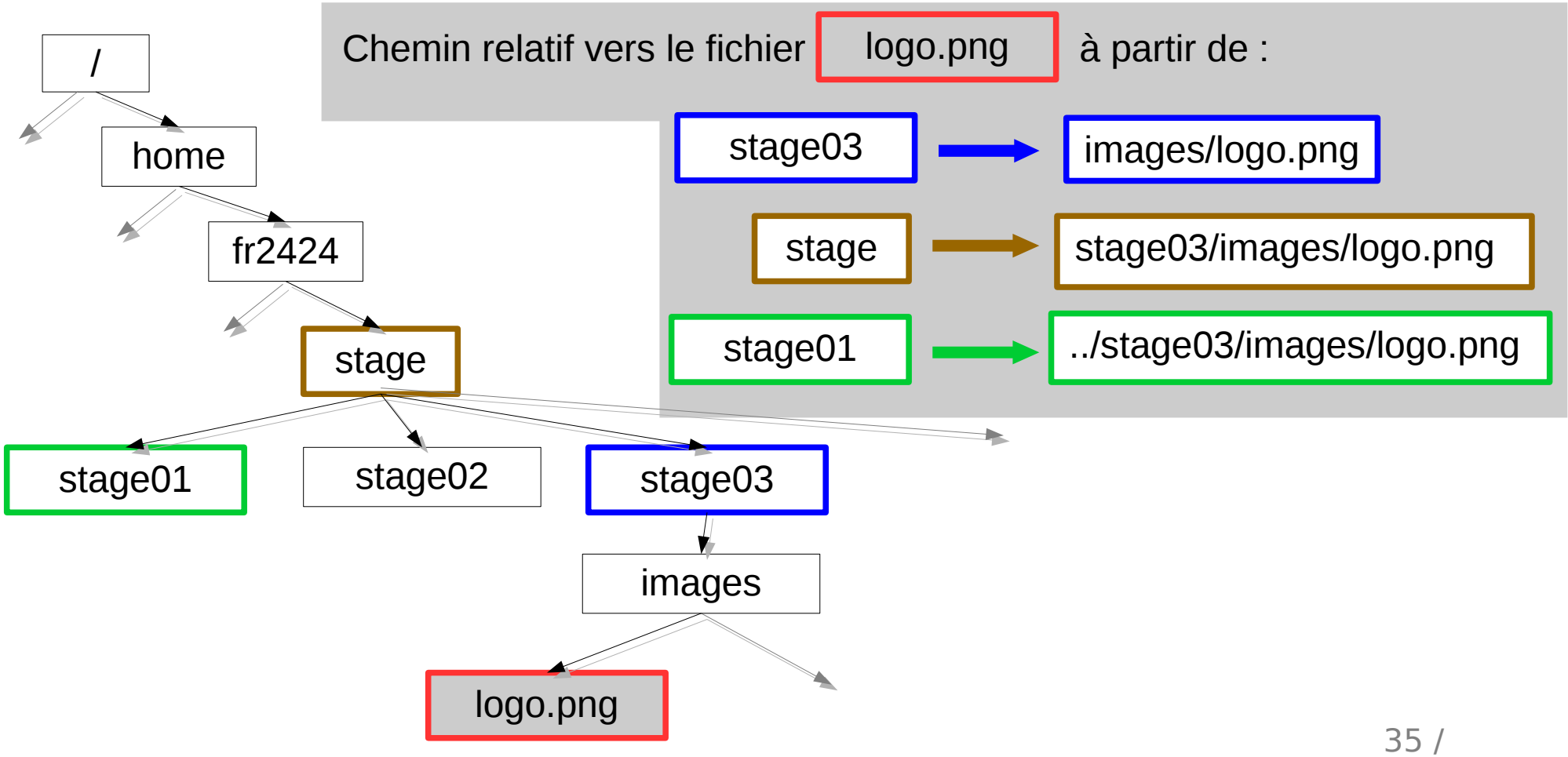
Les **chemins absolus** sont construits en partant de la racine de l'arborescence, et en ajoutant la succession des répertoires, séparés par des *slashes* (/), jusqu à atteindre le répertoire ou fichier souhaité.



Un même chemin absolu désignera toujours un même fichier ou répertoire, quel que soit le répertoire courant.

Chemins relatifs

Les **chemins relatifs** sont construits **en partant du répertoire courant** et en parcourant l'arborescence en montant et/ou en descendant, jusqu à atteindre le répertoire ou fichier souhaité. Les différents composants du chemin sont séparés par le caractère *slash (/)* :
- À chaque fois que l'**on remonte** dans l'arborescence on ajoute deux points **(..)** au chemin.
- À chaque fois que l'**on descend** dans l'arborescence on ajoute le nom du dossier dans lequel on descend.



Pour conclure (provisoirement)

Comment désigner le répertoire courant ?

- Le point (caractère .) désigne le répertoire courant

```
[stage01@nz ~]$ pwd
/home/fr2424/stage/stage01
[stage01@nz ~]$ cd . # ???
[stage01@nz ~]$ pwd
/home/fr2424/stage/stage01
```

Cas d'utilisation : exécuter une commande qui correspond à un fichier dans le répertoire courant : `./macommande`

Comment désigner son répertoire de connexion (home directory) ?

- La tilde (caractère ~) désigne le répertoire de connexion de l'utilisateur courant

```
[stage01@nz ~]$ pwd
/home/fr2424/stage/stage03
[stage01@nz ~]$ cd ~ # change to home dir
[stage01@nz ~]$ pwd
/home/fr2424/stage/stage01
```

Commandes pour la manipulation du système de fichiers

Examen du contenu des répertoires : `ls`

```
[stage11@nz ~]$ ls                    # no arguments : current dir
Bureau  Documents  Images  Modèles  Musique  Public  Téléchargements
Vidéos
[stage11@nz ~]$ ls /tmp/Linux-Initiation # absolute dir path
acteur.csv  cours  insulin.fas  insulin_vs_nt.blast  tmp
[stage11@nz ~]$ ls .. # relative dir path (parent dir)
common                stage02  stage1   stage17  stage24  stage31  stage6
common.linux-avance   stage03  stage10  stage18  stage25  stage32  stage7
(...)
```

Les fichiers «cachés»

Par défaut, **`ls`** n'affiche pas les fichiers dont les noms commencent par un point. Il faut ajouter l'option **`-a`** pour les inclure dans l'affichage (ou bien utiliser le raccourci **`la`** au lieu de **`ls`**).

```
[stage11@nz ~]$ ls -a                 # also show hidden files
.              Bureau   Documents      .kde      Public              .zshrc
..             .cache   .emacs         .local    Téléchargements
.bash_logout   .compiz  .gconf         Modèles   Vidéos
(...)
```

Afficher les fichiers correspondant à un motif

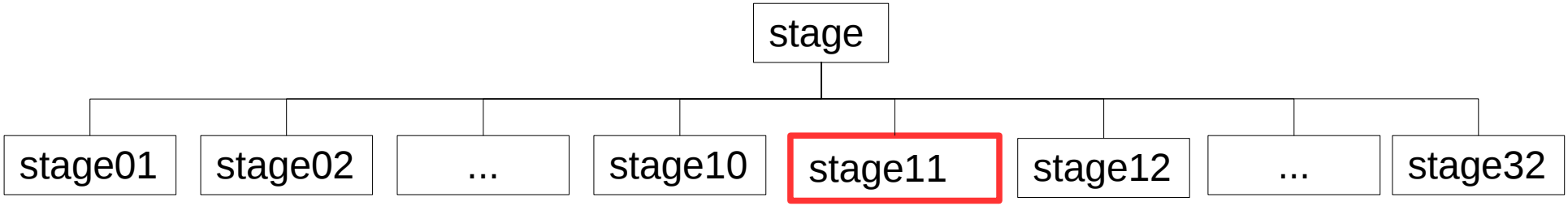
L'utilisation du caractère ***** dans l'argument de **`ls`** restreint l'affichage aux fichiers/répertoires dont le nom correspond au motif formé par l'argument :

- **`image*`** : tous les fichiers dont le nom commence par image (**`image-001, images-des-vacances, imagettes`**)
- **`*seq*`** : tous les fichiers dont le nom comporte les lettres **`seq (sequences, mes-sequences, maiseqoidon)`**
- **`*`** : tous les fichiers (pas de restriction)

```
[stage11@nz ~]$ ls Linux-Initiation/*ins*
Linux-Initiation/insulin.fas  Linux-Initiation/insulin_vs_nt.blast
```

Utiliser l'autocomplétion

Pour éviter d'avoir à saisir des noms de fichier en entier, il est possible d'utiliser la touche tabulation **[TAB]**
Un appui sur **[TAB]** va provoquer la recherche de noms de fichiers (ou de répertoires) dont le début correspond à ce qui a déjà été saisi.
- Si une seule correspondance est trouvée, elle sera affichée.
- Si plusieurs correspondances sont trouvées, rien ne sera affiché et il faudra un deuxième appui sur **[TAB]** pour les lister toutes.




```
[stage11@nz ~]$ ls ../sta[TAB]
[stage11@nz ~]$ ls ../stage
[stage11@nz ~]$ ls ../stage[TAB]
stage01/ stage07/ stage12/ stage18/ stage23/ stage29/ stage34/ stage7/
stage02/ stage08/ stage13/ stage19/ stage24/ stage3/  stage35/ stage8/
stage03/ stage09/ stage14/ stage2/  stage25/ stage30/ stage36/ stage9/
(...)
```

Afficher (une partie de) l'arborescence avec une seule commande

À l'aide de la commande **`ls`** en utilisant l'option **`-R`** (comme récursif) :

```
[stage11@nz ~]$ ls -R Linux-Initiation
Linux-Initiation/:
acteur.csv  cours  insulin.fas  insulin_vs_nt.blast  tmp
Linux-Initiation/cours:
Linux-Initiation/tmp:
```

À l'aide de la commande **`tree`** :

```
[stage11@nz ~]$ tree Linux-Initiation
Linux-Initiation
├── acteur.csv
├── cours
├── insulin.fas
├── insulin_vs_nt.blast
└── tmp

2 directories, 3 files
```

Organiser ses données en créant des (sous-)répertoires

À l'aide de la commande **`mkdir`** (*make directory*), qui ne crée par défaut que le dernier répertoire du chemin donné en argument :

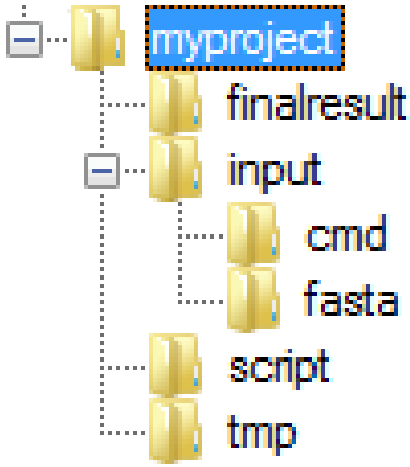
```
[stage11@nz ~]$ mkdir Linux-Initiation/tmp/essais
[stage11@nz ~]$ ls -R Linux-Initiation
Linux-Initiation/:
acteur.csv  cours  insulin.fas  insulin_vs_nt.blast  tmp
Linux-Initiation/cours:
Linux-Initiation/tmp:
Linux-Initiation/tmp/essais:
```

Son option **`-p`** permet de créer une sous-arborescence de répertoires en une seule fois :

```
[stage11@nz ~]$ mkdir -p Linux-Initiation/exercices/ex1/data
[stage11@nz ~]$ ls -R Linux-Initiation/
(...)
Linux-Initiation/exercices:
ex1
Linux-Initiation/exercices/ex1:
data
Linux-Initiation/exercices/ex1/data:
```

Application : exercices

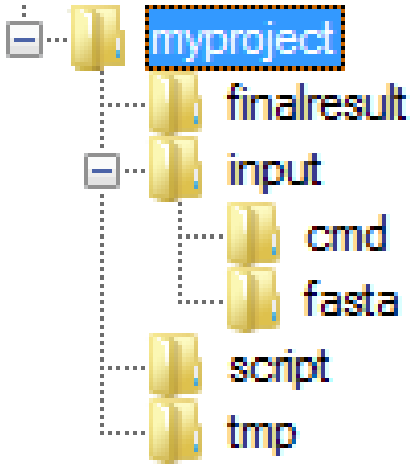
- Créez l'aborescence ci-dessous dans votre *home dir* :



- Vérifiez qu'elle a bien été créée en l'affichant :

Application : exercices

- Créez l'aborescence ci-dessous dans votre *home dir* :



Solution

```
$ cd
$ mkdir myproject
$ cd myproject
$ mkdir finalresult input script tmp
$ cd input
$ mkdir cmd fasta
```

- Vérifiez qu'elle a bien été créée en l'affichant :

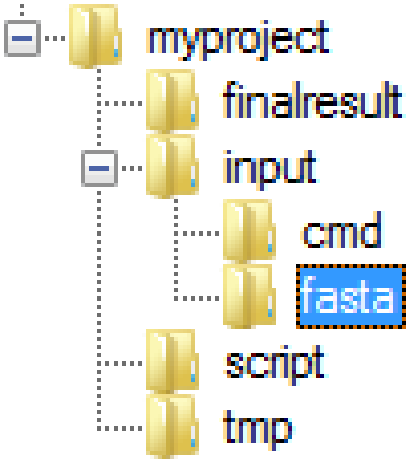
Solution

```
$ tree
$ tree -L 1
$ tree -L 2
```

Application : exercices

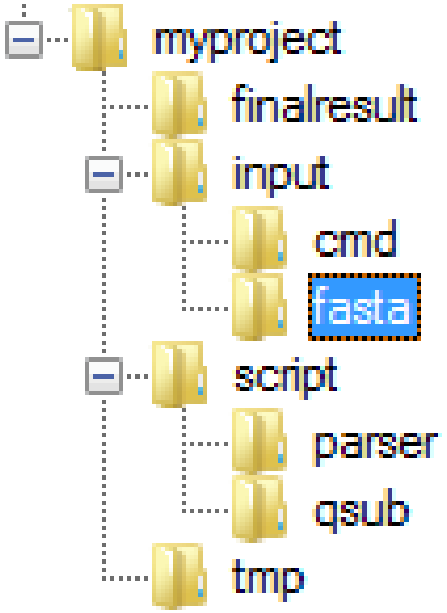
Avec une seule ligne de commande pour chaque point :

- Revenez dans votre *home dir*



- Déplacez-vous dans le dossier **`fasta`**

- Créez un dossier **`parser`** dans le dossier **`script`**



Application : exercices

Avec une seule ligne de commande pour chaque point :

- Revenez dans votre *home dir*

Solution

```
$ cd
```

- Déplacez-vous dans le dossier **`fasta`**

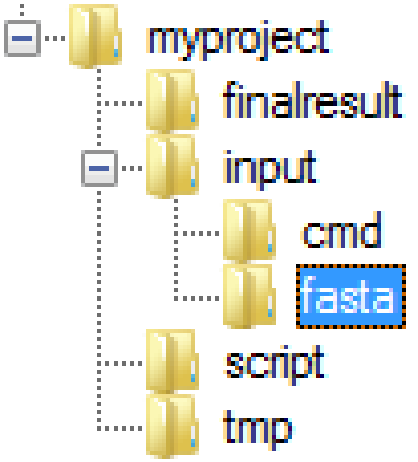
Solution

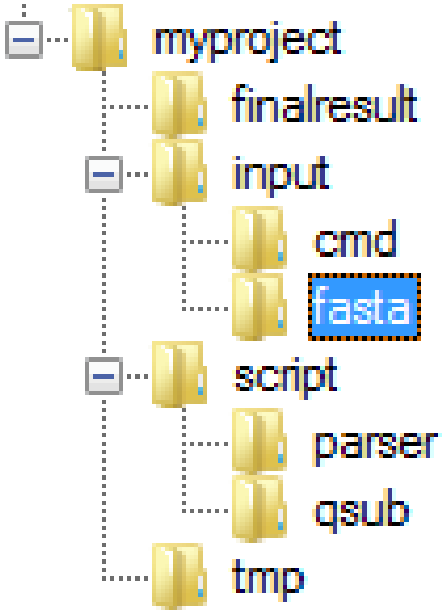
```
$ cd myproject/input/fasta
```

- Créez un dossier **`parser`** dans le dossier **`script`**

Solution

```
$ mkdir ../../script/parser
```





Copier des données

La commande **cp** *(copy)* effectue la copie d'un ou plusieurs fichiers, voire d'arborescences entières. Elle se construit comme suit : **cp *SRC DEST*** ou **SRC** désigne le chemin vers les données à copier (la source) et **DEST** le chemin où l'on veut les copier.

Ex. 1 : copie d'un seul fichier

```
[stage11@nz ~]$ cp acteur.csv acteur_bak.csv
```

Ex. 2 : copie d'un seul fichier vers un autre répertoire

```
[stage11@nz ~]$ cp acteur.csv tmp # keep same filename in DEST
[stage11@nz ~]$ cp acteur.csv tmp/stars.csv # change filename
```

Ex. 3 : copie d'un ensemble de fichiers dans un autre répertoire à l'aide d'un motif

```
[stage11@nz ~]$ cp insulin* tmp
[stage11@nz ~]$ ls tmp
insulin.fas  insulin_vs_nt.blast
```

Ex. 4 : copie d'une arborescence à l'aide de l'option **-r** *(recursive)*

```
[stage11@nz ~]$ cp -r ../stage10/exercices/solutions  .
```

Déplacer ou renommer des données

La commande **mv** *(move),* en fonction des arguments qui lui sont spécifiés, effectue le renommage ou le déplacement d'un ou plusieurs fichiers, voire d'arborescences entières.
Elle se construit comme suit : **mv *SRC DEST*** ou **SRC** désigne le chemin vers les données existantes à déplacer ou renommer (la source) et **DEST** leur nouveau nom ou leur nouvel emplacement.

Ex. 1 : renommage d'un seul fichier

```
[stage11@nz ~]$ mv acteur.csv liste_acteurs.csv
```

Ex. 2 : déplacement d'un seul fichier vers un autre répertoire **(qui doit déjà exister)**

```
[stage11@nz ~]$ mv acteur.csv tmp # keep same filename in DEST
[stage11@nz ~]$ mv acteur.csv tmp/stars.csv # change filename
```

Ex. 3 : déplacement d'un ensemble de fichiers dans un autre répertoire **(existant)** à l'aide d'un motif

```
[stage11@nz ~]$ mv insulin* tmp
```

Ex. 4 : déplacement d'une arborescence

```
[stage11@nz ~]$ mv tmp/work/last_stage/output ./finalresults
```

- Si **./finalresults** existe déjà, le répértoire **output** y sera déplacé avec tout son contenu.
- Si **./finalresults** n'existe pas, il sera crée et contiendra ce qu'il y avait dans **output**

Effacer des données

La commande **`rm`** *(remove)* efface le(s) fichier(s) dont le(s) chemin(s) est (sont) donné(s) en argument.

Ce qui est effacé avec `rm` ne peut pas être récupéré.

(jamais, never, jamas, nie, nooit, gwech ebet, någonsin, никогда, 曾經)

Ex. 1 : effacement d'un seul fichier

```
[stage11@nz ~]$ rm acteur_bak.csv
```

Ex. 2 : effacement de tous les fichiers correspondant à un motif

```
[stage11@nz ~]$ rm insulin*
```

Ex. 3 : effacement de toute une arborescence à l'aide de l'option **`-r`** *(recursive)*

```
[stage11@nz ~]$ rm -r Linux-Initiation/tmp
```

Ex. 4 : **Armageddon** : effacement **forcé** de toute une arborescence (option **`-f`**)

```
[stage11@nz ~]$ rm -rf ~/tmp/worthless_files
```

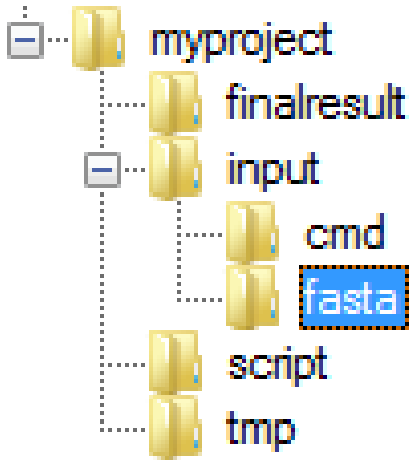
Cas particulier : effacement d'un répertoire vide à l'aide de la commande **`rmdir`**

```
[stage11@nz ~]$ rmdir ~/tmp/empty_directory
```

Application : exercices

Copiez le fichier **`insulin.fas`** dans le répertoire **`fasta`**

- Allez dans votre répertoire projet



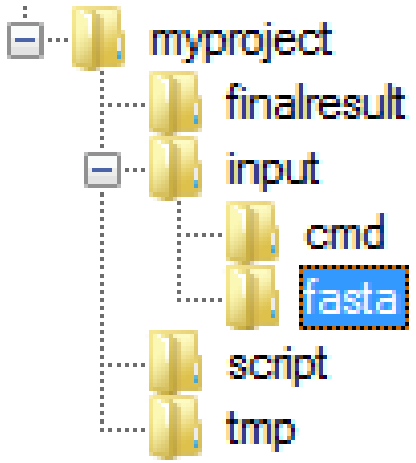
- Copiez le fichier dans le répetoire destination

Copiez le fichier **`insulin.fas`** dans le répertoire **`fasta`**

- Allez dans votre répertoire projet

Solution

```
$ cdprojet
```



- Copiez le fichier dans le répetoire destination

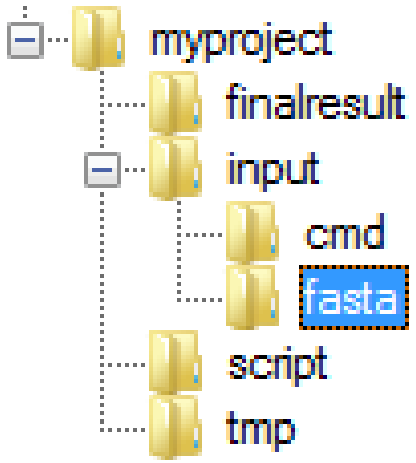
Solution

```
$ cp Linux-initiation/insulin.fas ~/myproject/input/fasta
```

Application : exercices

En vous plaçant dans le répertoire **`myproject/finalresult`**

- Déplacez le fichier **`insulin.fas`** à partir du répertoire **`input/fasta`** vers le répertoire **`tmp`**

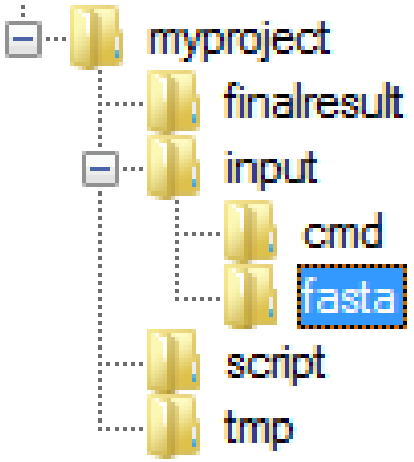


- Effacez le répertoire tmp et son contenu

Application : exercices

En vous plaçant dans le répertoire **myproject/finalresult**

- Déplacez le fichier **insulin.fas** à partir du répertoire **input/fasta** vers le répertoire **tmp**



Solution

```
$ cd ~/myproject/finalresult
$ mv ../input/fasta/insulin.fas ../tmp
```

- Effacez le répertoire tmp et son contenu

Solution

```
$ rm -r ~/myproject/tmp
```

Linux initiation | Plan

Quelques mots sur les noms de fichier (1)

Linux est **très** tolérant sur les caractères autorisés pour les noms des fichiers (espacements, caractères accentués…). Il vaut mieux éviter d'en abuser. Quelques préconisations :

Lettres minuscules & majuscules ; chiffres ; trait d'union ; sous-ligné («tiret bas») ; point.

Caractères accentués ou comportant d'autres signes diacritiques.

Espaces ou autres caractères de ponctuation

Cas particulier des espaces : déspécialisation avec *backslash* (\) ou les guillemets (")

```
[n00b@nz ~]$ mkdir nouveau dossier # creates 2 dirs, :(
[n00b@nz ~]$ ls .
./:
nouveau
dossier
```

```
[tux@nz ~]$ mkdir nouveau\ dossier # creates 1 dir, gg
[tux@nz ~]$ mkdir "nouveau dossier 2" # id.
[tux@nz ~]$ ls .
./:
nouveau dossier
Nouveau dossier 2
```

Quelques mots sur les noms de fichier (2)

Linux n'impose aucune règle sur les extensions des noms de fichier (**`.txt`**, **`.csv`**, **`.pdf`**, **`.html`**, etc.). N'importe quelle extension peut être donnée à n'importe quel fichier.

IL EST VIVEMENT RECOMMANDÉ DE RESTER COHÉRENT

Linux utilise d'autres méthodes pour déterminer la nature d'un fichier (cf. la commande **`file`** plus loin).

Déterminer la nature d'un fichier (1)

La commande **`file`** affiche une hypothèse sur la nature d'un fichier. Elle en examine le début et compare cette empreinte à une «base de données» d'empreintes.

Ex. 1 & 2 : fichiers spécifiques à certaines applications.

```
[stage11@nz ~]$ file Linux-Initiation-2017.pdf
Linux-Initiation-2017.pdf: PDF document, version 1.4
```

```
[stage11@nz ~]$ file Linux-Initiation-2017.pptx
Linux-Initiation-2017.pptx: Microsoft PowerPoint 2007+
```

Ex. 3 & 4 : fichiers avec des archives compressées

```
[stage11@nz ~]$ file Linux-Initiation-supports.zip
Linux-Initiation-2017-supports.zip: Zip archive data, at
least v2.0 to extract
```

```
[stage11@nz ~]$ file Linux-Initiation-supports.tar.gz
Linux-Initiation-2017-supports.tar.gz: gzip compressed data,
last modified: Sat May 7 23:56:36 2017, from Unix
```

Déterminer la nature d'un fichier (2)

Ex. 5 & 6 : fichiers exécutables (commandes ou suites de commandes dans un fichier texte)

```
[stage11@nz ~]$ file /usr/bin/file
/usr/bin/file: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux
2.6.32, BuildID[sha1]=a4f09f32eb214a3f9435484fa01d54c939bcf30c, stripped
```

```
[stage11@nz ~]$ file monscript.sh # textfile with commands
monscript.sh: POSIX shell script, ASCII text executable
```

Ex. 7 & 8 : fichiers contenant du texte

```
[stage11@nz ~]$ file insulin.fas
acteur.csv: ASCII text, with CRLF line terminators
```

```
[stage11@nz ~]$ file acteur.csv
acteur.csv: ASCII text, with CRLF line terminators
```

Ex. 9 : fichier de données (dans un format inconnu de la commande **`file`**)

```
[stage11@nz ~]$ file random.dat
random.dat: data
```

Examiner le contenu d'un fichier (texte)

La commande **cat** affiche l'intégralité du contenu d'un fichier.

```
[stage11@nz ~]$ cat acteur.csv
First Name;Last Name;Age
Chuck;Norris;70
Sylvester;Stallone;64
Steven;Seagal;59
```

La commande **head** affiche les (10) premières lignes d'un fichier. **head -*n*** en affiche les ***n*** premières.

```
[stage11@nz ~]$ head -2 acteur.csv
First Name;Last Name;Age
Chuck;Norris;70
```

La commande **tail** affiche les (10) dernières lignes d'un fichier. **tail -*n*** en affiche les ***n*** dernières.

```
[stage11@nz ~]$ tail -2 acteur.csv
Sylvester;Stallone;64
Steven;Seagal;59
```

Examiner *interactivement* le contenu d'un fichier (texte)

La commande **more** affiche le contenu d'un fichier «page par page». La barre d'espace sert à afficher la page suivante ; la touche « q » à quitter.

```
[stage11@nz ~]$ more insulin.fas
>gi|163659904|ref|NM_000618.3| Homo sapiens insulin-like growth factor
1 (somatomedin C) (IGF1), transcript variant 4, mRNA
TTTTGTAGATAAATGTGAGGATTTTCTCTAAATCCCTCTTCTGTTTGCTAAATCTCACTGTCACTGCTAA
(…)
```

La commande **less** affiche le contenu d'un fichier «page par page». La barre d'espace sert à afficher la page suivante ; la touche « q » à quitter ; et les flèches ⬆ et ⬇ permettent de naviguer dans le fichier.

```
[stage11@nz ~]$ less insulin.fas
>gi|163659904|ref|NM_000618.3| Homo sapiens insulin-like growth factor
1 (somatomedin C) (IGF1), transcript variant 4, mRNA
TTTTGTAGATAAATGTGAGGATTTTCTCTAAATCCCTCTTCTGTTTGCTAAATCTCACTGTCACTGCTAA
(…)
```

Aussi bien **more** que **less** permettent de rechercher un morceau de texte dans le fichier en saisissant un slash (**/**) suivi du texte à rechercher et de la touche **Entrée.** Ex : **/variant**↵

Rechercher de l'information dans des fichiers (1)

La commande **grep** prend en argument un ***motif*** et un ***nom de fichier*** ; et affiche les lignes du fichier qui contiennent ce motif.

```
[stage11@nz ~]$ grep transcript insulin.fas
>gi|163659904|ref|NM_000618.3| Homo sapiens insulin-like growth factor 1 (somatomedin C) (IGF1),
transcript variant 4, mRNA
>gi|163659900|ref|NM_001111284.1| Homo sapiens insulin-like growth factor 1 (somatomedin C) (IGF1),
transcript variant 2, mRNA
>gi|163659895|ref|NM_001111276.1| Mus musculus insulin-like growth factor 1 (Igf1), transcript
variant 5, mRNA
>gi|163659893|ref|NM_001111275.1| Mus musculus insulin-like growth factor 1 (Igf1), transcript
variant 4, mRNA
>gi|163659891|ref|NM_010512.4| Mus musculus insulin-like growth factor 1 (Igf1), transcript variant
1, mRNA
```

Par défaut, **grep** est sensible à la casse. Pour s'en affranchir, on peut utiliser l'option **-i** (ignorecase)

```
[stage11@nz ~]$ grep TRANSCRIPT insulin.fas # returns nothing
[stage11@nz ~]$ grep -i TRANSCRIPT insulin.fas
>gi|163659904|ref|NM_000618.3| Homo sapiens insulin-like growth factor 1 (somatomedin C) (IGF1),
transcript variant 4, mRNA
>gi|163659900|ref|NM_001111284.1| Homo sapiens insulin-like growth factor 1 (somatomedin C) (IGF1),
transcript variant 2, mRNA
(...)
```

Rechercher de l'information dans des fichiers (2)

grep permet également de compter les lignes qui correspondent au motif à l'aide de l'option **-c** (count)

```
[stage11@nz ~]$ grep -c transcript insulin.fas
5
```

Comme pour la plupart des commandes, les options de **grep** peuvent être combinées.

```
[stage11@nz ~]$ grep -c -i TRANSCRIPT insulin.fas
5
```

Par défaut, **grep** peut également afficher toutes les lignes ***qui ne contiennent pas*** le motif donné, à l'aide de l'option **-v** (in**v**ert)

```
[stage11@nz ~]$ grep -v -c -i TRANSCRIPT insulin.fas
511
```

grep peut servir à rechercher un motif dans tous les fichiers d'une arborescence, à l'aide de l'option **-r** (recursive). Dans ce cas, le deuxième argument doit être le nom d'un répertoire. Les lignes avec les informations sur les motifs sont alors préfixées par le nom du fichier auquel ils appartiennent.

```
[stage11@nz ~]$ grep -r -c -i TRANSCRIPT .
./insulin_vs_nt.blast:144
./acteur.csv:0
./insulin.fas:5
```

Application : exercices

Trouvez deux façons d'afficher la première ligne du fichier **`acteur.csv`** (en utilisant deux commandes différentes)

Trouvez deux façons d'afficher les trois dernières lignes du fichier **`acteur.csv`** (en utilisant deux commandes différentes)

Application : exercices

Trouvez deux façons d'afficher la première ligne du fichier **`acteur.csv`** (en utilisant deux commandes différentes)

Solution

```
$ head -1 acteur.csv
$ grep First  acteur.csv
```

Trouvez deux façons d'afficher les trois dernières lignes du fichier **`acteur.csv`** (en utilisant deux commandes différentes)

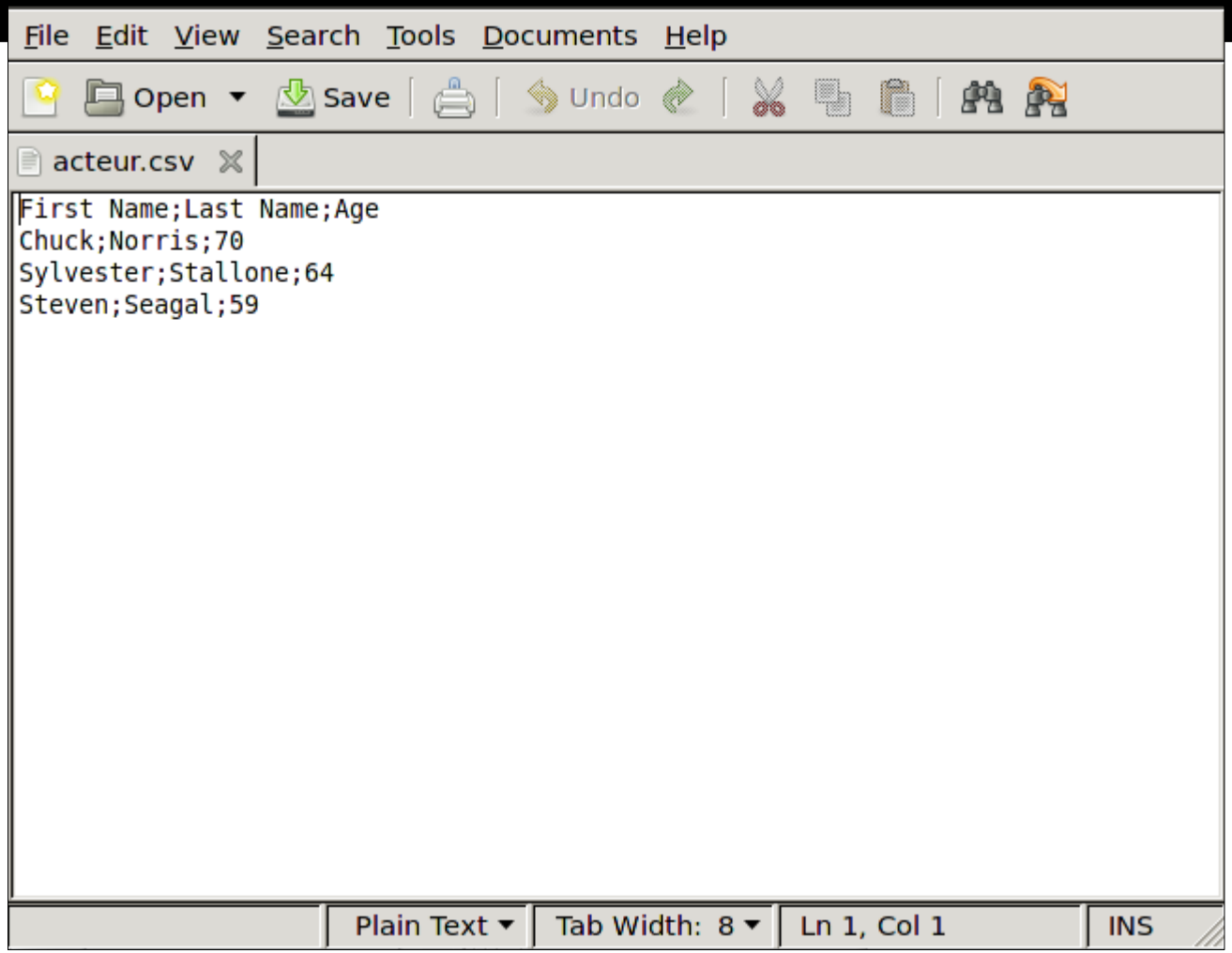
Solution

```
$ tail -3 acteur.csv
$ grep -v First  acteur.csv
```

Modifier le contenu d'un fichier (texte)

La commande **gedit** ouvre une fenêtre avec un éditeur de texte.

```
[stage11@nz ~]$ gedit acteur.csv
```



Modifier le contenu d'un fichier (texte)

La commande **nano** ouvre une éditeur dans la fenêtre avec la session active.

```
[stage11@nz ~]$ nano acteur.csv
```

```
  GNU nano 2.7.4                Fichier : acteur.csv

First Name;Last Name;Age
Chuck;Norris;70
Sylvester;Stallone;64
Steven;Seagal;59

                [ Lecture de 5 lignes (converties du format DOS) ]
^G Aide       ^O Écrire     ^W Chercher   ^K Couper     ^J Justifier  ^C Pos. cur.
^X Quitter    ^R Lire fich. ^\ Remplacer  ^U Coller     ^T Orthograp. ^_ Aller lig.
```

Gestion de fichiers & archivage

Il est possible de connaître la taille d'un fichier à l'aide des options **`-l (-h)`** de la commande **`ls.`**
La taille est affichée dans le cinquième champ.

```
[stage08@nz ~]$ ls -l insulin_vs_nt.blast
-rw-r--r-- 1 stage08 stage 30025889 May 03 22:42 insulin_vs_nt.blast
[stage08@nz ~]$ ls -l -h  insulin_vs_nt.blast
-rw-r--r-- 1 stage08 stage 29M May 03 22:42 insulin_vs_nt.blast
```

La commande **`wc`** (word count) donne également des informations sur la taille d'un fichier (en lignes, mots et caractères). L'utilisation de l'option **`-l`** n'affiche que le nombre de lignes.

```
[stage08@nz ~]$ wc insulin_vs_nt.blast
622756  2377511 30025889 insulin_vs_nt.blast
[stage08@nz ~]$ wc -l insulin_vs_nt.blast
622756  insulin_vs_nt.blast
```

Gestion de fichiers & archivage

Pour déterminer la taille d'un dossier (et de tout son contenu), on utilise la commande **du** *(disk usage)*, de préférence avec l'option **-h.**

```
[stage08@nz ~]$ du -h .
64.0K   ./tmp
4.0K    ./cours
29M     .
```

L'ajout de l'option **-s** (summary) renvoie le volume total occupé par le dossier (et tout son contenu).

```
[stage08@nz ~]$ du -s -h .
29M     .
```

Gestion de fichiers & archivage

Pour déterminer la place disponible sur un point de montage (un disque), on utilise la commande **df** qui dispose également d'une option **-h.**

```
[stage08@nz ~]$ df -h
Filesystem                      Size  Used Avail Use% Mounted on
/dev/sda7                      1008M  750M  208M  79% /
(…)
/dev/sda1                       504M  152M  327M  32% /boot
(…)
/dev/sda2                        32G  541M   30G   2% /tmp
/dev/sda5                        16G  9.5G  5.5G  64% /usr
/dev/mapper/VolGroup00-LogVol00 4.0G  3.0G  795M  80% /usr/local
(…)
brazil:/home/umr7139/mma        247G  210G   25G  90% /home/umr7139/mmaucture
brazil:/home/umr7139/tccd       591G  538G   24G  96% /home/umr7139/tccd
brazil:/home/umr7144/abice      1.2T  440G  636G  41% /home/umr7144/abice
(…)
```

En spécifiant un répertoire en argument, **df** affiche les informations qui correspondent au point de montage qui contient ce répertoire.

```
[stage08@nz ~]$ df -h /home/fr2424/sib/mhoebeke
Filesystem                Size  Used Avail Use% Mounted on
brazil:/home/fr2424/sib   2.0T  1.6T  242G  88% /home/fr2424/sib
(…)
```

Gestion de fichiers & archivage

Linux propose différentes commandes de compression de fichiers, comme **gzip** (plus ancien) ou **bzip2** (meilleure compression, mais moins portable entre systèmes). Par défaut **gzip** remplace le fichier dont le nom est donné en argument par sa version compressée et ajoute **.gz** au nom du fichier (**.bz2** pour **bzip2**).

```
[stage08@nz ~]$ gzip insulin_vs_nt.blast
[stage08@nz ~]$ ls -l -h insulin_vs_nt.blast.gz
-rw-r--r-- 1 stage08 stage 5.0M May 13 20:42 insulin_vs_nt.blast.gz
```

```
[stage08@nz ~]$ bzip2 insulin_vs_nt.blast
[stage08@nz ~]$ ls -l -h insulin_vs_nt.blast.gz
-rw-r--r-- 1 stage08 stage 3.2M May 13 20:42 insulin_vs_nt.blast.bz2
```

La décompression se fait à l'aide des commandes **gunzip** (pour les fichiers **.gz**) ou **bzip2** (pour les fichiers **.bz2**).

```
[stage08@nz ~]$ gunzip insulin_vs_nt.blast.gz
```

```
[stage08@nz ~]$ bunzip2 insulin_vs_nt.blast.bz2
```

Le taux de compression dépend de la nature des fichiers : meilleur pour les fichiers texte, très faible pour les données déjà compressées (images, sons, vidéos).

Gestion de fichiers & archivage

La création d'une archive regroupant plusieurs fichiers se fait à l'aide de la commande **tar.**

Ex 1 : Création d'une archive avec tout le contenu du répertoire **Linux-Initiation** : l'option **-c** indique que l'on va créer une archive, l'option **-f** indique que le nom du fichier archive à créer va suivre. L'argument final est le nom du répertoire à archiver.

```
[stage08@nz ~]$ tar -cf Linux-Initiation.tar Linux-Initiation
```

tar laisse en l'état les données à archiver.

Ex 2 : Extraction de tous fichiers d'une archive précédemment créée : l'option **-x** indique que l'on va extraire les données d'une archive, la signification de l'option **-f** est la même que ci-dessus. L'option **-v** active le mode « verbeux » qui affiche les fichiers à mesure qu'ils sont extraits.

```
[stage08@nz ~]$ tar -xvf Linux-Initiation.tar
Linux-Initiation/
Linux-Initiation/tmp/
Linux-Initiation/insulin_vs_nt.blast
(...)
```

Ex 3 : Extraction d'un seul fichier : on ajoute le nom du fichier à extraire en argument.

```
[stage08@nz ~]$ tar -xvf Linux-Initiation.tar Linux-Initiation/insulin.fas
Linux-Initiation/insulin.fas
```

Gestion de fichiers & archivage

Il est possible de demander à **tar** de réaliser la compression «à la volée». Pour une compression-décompression à l'aide de **gzip,** on ajoute l'option **-z.** Pour une compression-décompression à l'aide **bzip2,** on ajoute l'option **-j.**

Ex1. : Compression/décompression à la volée avec **gzip.**

```
[stage08@nz ~]$ tar -czf Linux-Initiation.tar.gz Linux-Initiation
[stage08@nz ~]$ file Linux-Initiation.tar.gz
Linux-Initiation.tar.gz: gzip compressed data (…)
[stage08@nz ~]$ tar -xzf Linux-Initiation.tar.gz
```

Ex1. : Compression/décompression à la volée avec **bzip2.**

```
[stage08@nz ~]$ tar -cjf Linux-Initiation.tar.bz2 Linux-Initiation
[stage08@nz ~]$ file Linux-Initiation.tar.bz2
Linux-Initiation.tar.bz2: bzip2 compressed data (…)
[stage08@nz ~]$ tar -xjf Linux-Initiation.tar.bz2
```

Utiliser les raccourcis : les liens symboliques

Il est souvent pratique de pouvoir accéder à un ensemble de fichiers dans un même répertoire, même si ces fichiers sont à l'origine répartis dans plusieurs répertoires ; **et ceci sans en faire des copies**.

Cas d'utilisation :
-Je dispose d'un répertoire **`allsequences`** qui contient une multitude de fichiers de séquences qui se rapportent à divers organismes.
-Les noms des fichiers renseignent sur l'organisme des séquences qu'ils contiennent (**`human_seq*.fasta`, `mouse_seq*.fasta`, `ecto_seq*.fasta`,** etc.)
-Je souhaite faire des traitements différenciés en fonction des organismes et regrouper données et résultats de chaque organisme dans un répertoire séparé **`process_human/`, `process_mouse/`, `process_ecto/`,** etc.)
- Mais je ne veux pas copier les fichiers de séquence de chaque organisme dans les répertoires **`process_*/`**

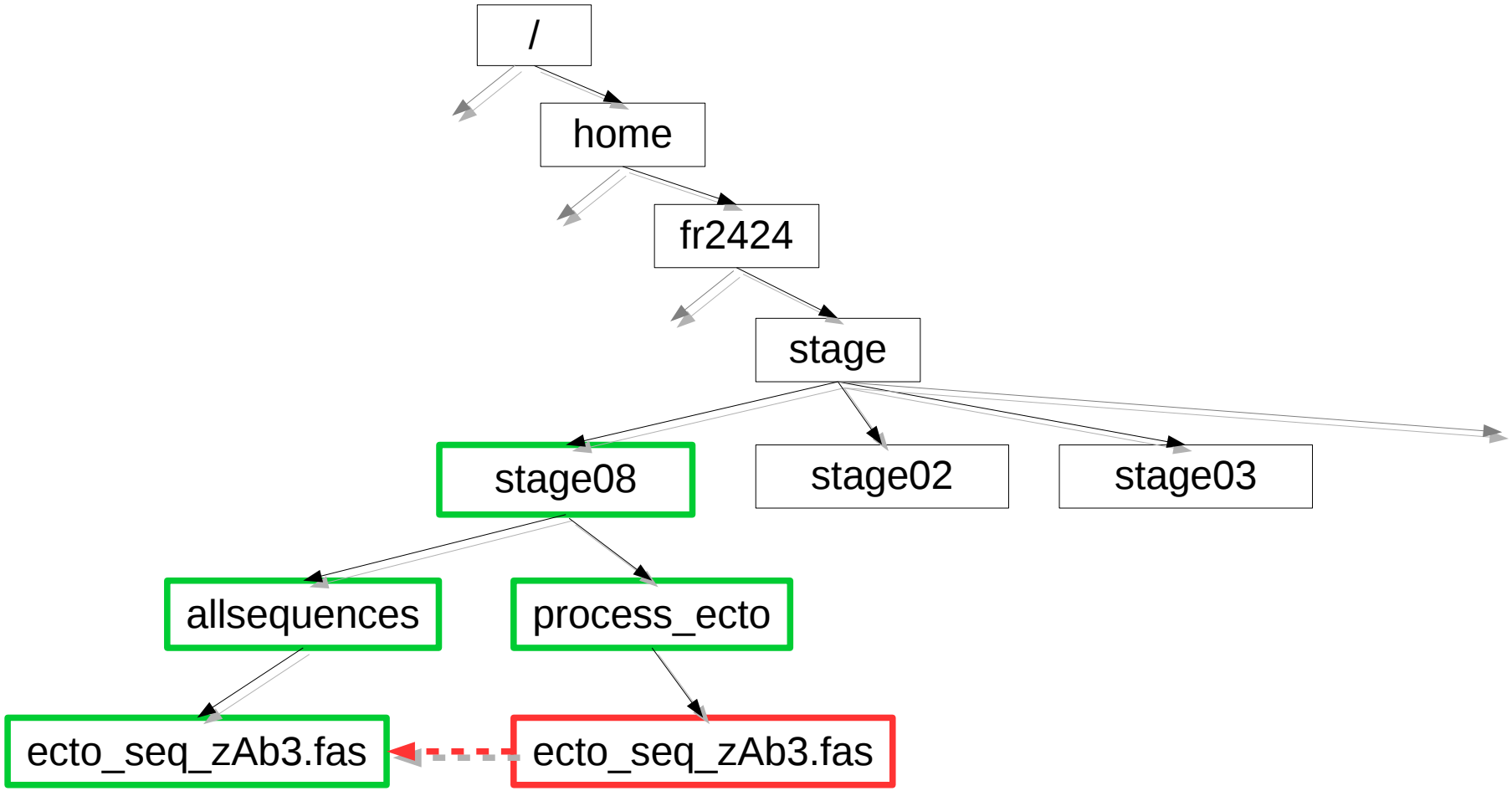
Une solution consiste à créer, dans les répertoires **`process_*/`** des raccourcis vers les fichiers de séquence de chaque organisme à l'aide de la commande **`ln -s`** *(link, symbolic)*.

Ex. 1 : création d'un lien symbolique pour un fichier.

```
[stage11@nz ~]$ ln -s allsequences/ecto_seq_zAb3.fas process_ecto/
[stage11@nz ~]$ ls -l process_ecto/
lrwxrwxrwx 1 stage08 stage 30 May 05 08:44 ecto_seq_zAb3.fas ->
allsequences/ecto_seq_zAb3.fas
```

Utiliser les raccourcis : les liens symboliques

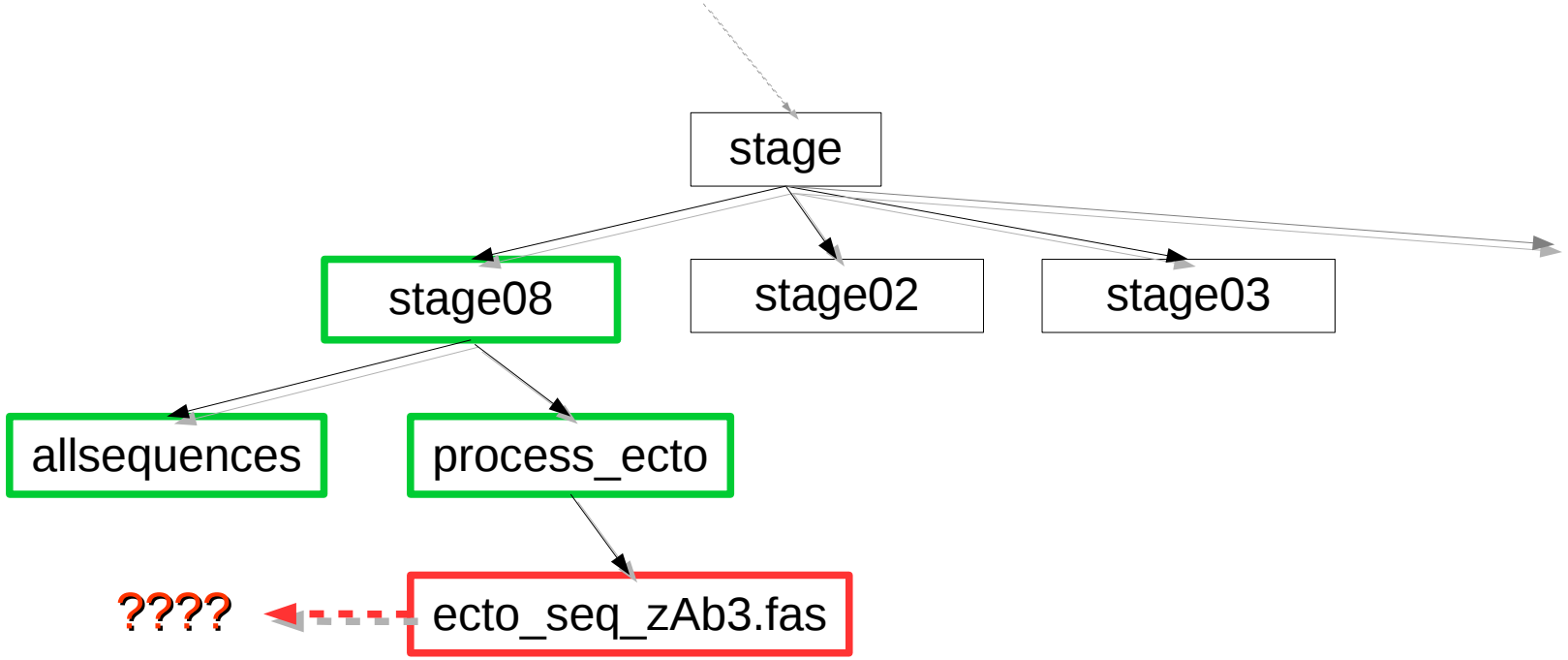
Le lien symbolique crée une nouvelle entrée dans le système de fichiers qui «pointe» sur une entrée existante.



Utiliser les raccourcis : les liens symboliques

L'effacement de la source du lien symbolique rend ce dernier inutilisable !

```
[stage11@nz ~]$ rm -f allsequences/ecto_seq_zAb3.fas
[stage11@nz ~]$ cat process_ecto/ecto_seq_zAb3.fas
cat: process_ecto/ecto_seq_zAb3.fas: No such file or directory
```

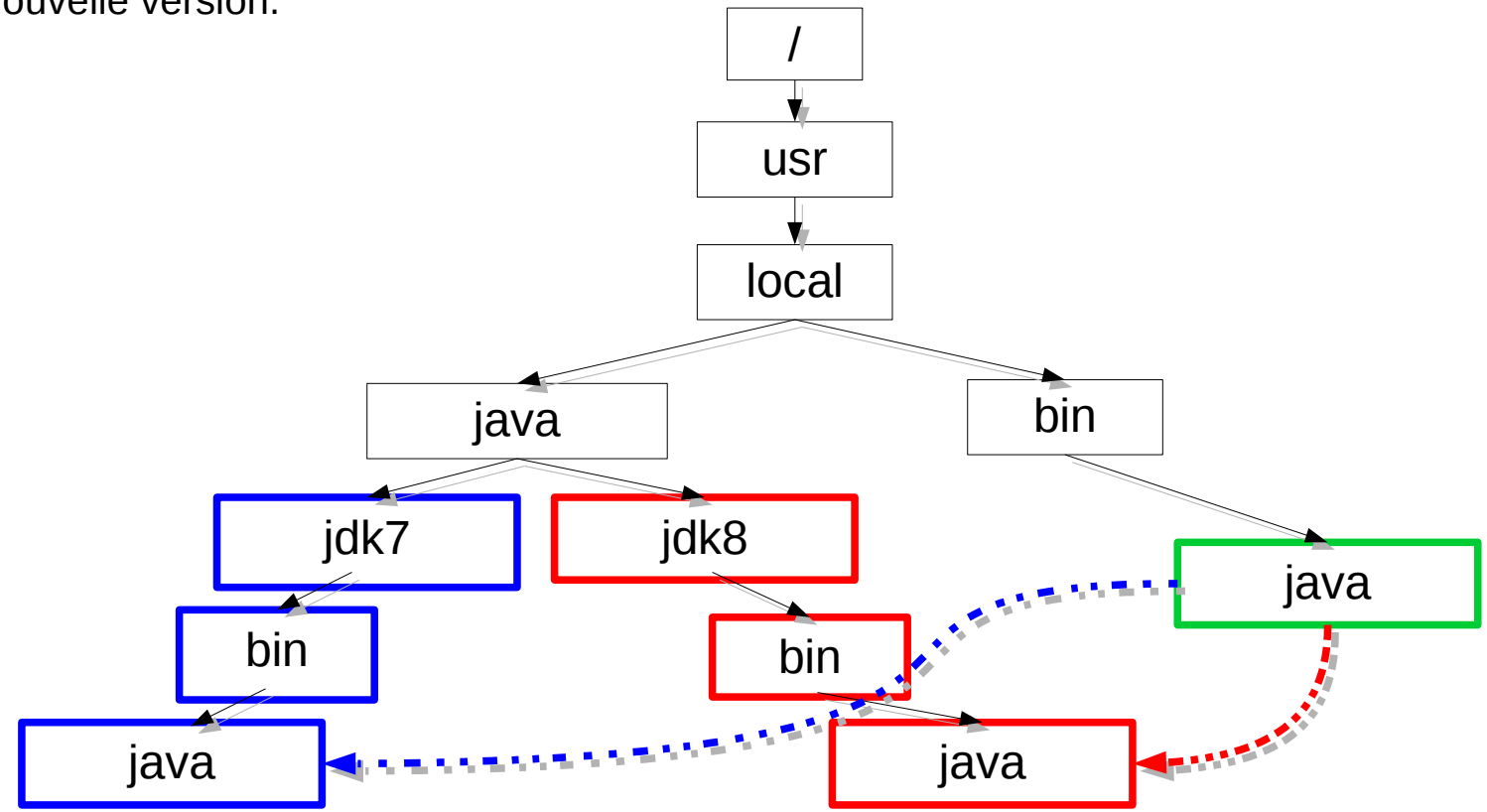


Utiliser les raccourcis : les liens symboliques

Il est facile de créer des liens symboliques «en masse» :

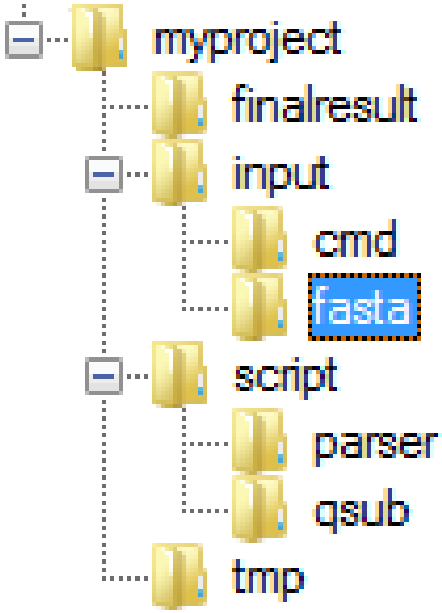
```
[stage11@nz ~]$ ln -s allsequences/ecto_seq*.fas process_ecto/
[stage11@nz ~]$ ln -s allsequences/mouse_seq*.fas process_mouse/
```

Les liens symboliques permettent également de gérer les mises à jour des commandes de manière transparente : une commande peut «pointer» sur une version précise d'un outil. Lors de la mise à jour de l'outil, on installe sa nouvelle version (sans remplacer l'ancienne !), et on fait pointer la commande sur la nouvelle version.



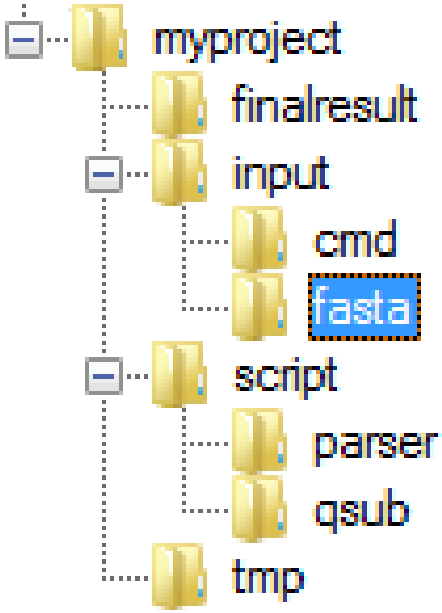
Application : exercices

- Créez un lien symbolique du répertoire script dans votre *home directory*
- Créez un lien du fichier **`test-TP.txt`** situé dans **`finalresult`** dans votre *home directory*
- Affichez le fichier **test-TP.txt** présent dans votre *home directory*
- Supprimez le fichier **finalresult/test-TP.txt**
- Constat ?



Application : exercices

- Créez un lien symbolique du répertoire script dans votre *home directory*
- Créez un lien du fichier **`test-TP.txt`** situé dans **`finalresult`** dans votre *home directory*
- Affichez le fichier **test-TP.txt** présent dans votre *home directory*
- Supprimez le fichier **finalresult/test-TP.txt**
- Constat ?




```
[stage11@nz ~]$ ln -s myproject/script ~
[stage11@nz ~]$ ln -s myproject/finalresult/test-TP.txt ~
[stage11@nz ~]$ cat test-TP.txt
(…)
[stage11@nz ~]$ rm -f myproject/finalresult/test-TP.txt
[stage11@nz ~]$ cat test-TP.txt
test-TP.txt : No such file or directory
```

Copie de fichiers de/vers une machine distante

La commande **scp** permet de copier des fichiers de/vers une autre machine Linux (UNIX). Elle nécessite de disposer des identifiants de connexion sur la machine distante. Elle s'utilise comme la commande **cp** mais l'un de ses arguments (source ou destination) intègre l'information sur la machine distante (nom d'utilisateur & nom ou adresse de la machine) sous la forme :

nom_utilisateur@nom_machine:

- Le mot de passe de l'utilisateur sur la machine distante sera demandé.
- Les transferts sont chiffrés : **scp** utilise le protocole de connexion SSH.

Ex1. : Copie d'un fichier vers une machine distante : les informations sur la machine distante sont dans **l'argument destination** de la copie

```
[stage08@nz ~]$ scp Linux-Initiation.tar.gz stage08@sbr2:cours/
```

Ex2. : Copie d'une arborescence à partir d'une machine distante : les informations sur la machine distante sont dans **l'argument source** de la copie

```
[stage08@nz ~]$ scp -r stage08@sbr2:cours/  .
```

Copie de fichiers accessibles via une adresse Web (URL)

La commande **wget** permet de récupérer une copie locale d'un (ensemble de) fichier(s) dont on connaît l'URL.

Ex1. : Récupération d'une entrée de l'ENA au format FASTA à partir de son numéro d'accession

```
[stage08@nz ~]$ wget "http://www.ebi.ac.uk/ena/data/view/BN000065&display=fasta"
--2017-04-14 12:42:09--  http://www.ebi.ac.uk/ena/data/view/BN000065&display=fasta
Resolving www.ebi.ac.uk... 193.62.193.80
Connecting to www.ebi.ac.uk|193.62.193.80|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: “BN000065&display=fasta”

    [ <=>                                                ] 320,575     --.-K/s   in 0.1s

2017-04-14 12:42:09 (2.07 MB/s) - “BN000065&display=fasta” saved [320575]
```

Ex2. : Récupération **récursive** (option **-r**) à partir d'une page HTML **(handle with care !)**

```
[stage08@nz ~]$ wget -r "http://abims.sb-roscoff.fr/training"
```

L'arborescence distante est recréée dans le répertoire d'où est lancé **wget**

Application : exercices

En une seule ligne de commande, copiez dans votre répertoire **`myproject/tmp`**, le fichier **`test-TP.txt`** localisé dans le répertoire **`/tmp`** de la machine **`sbr2`**

Application : exercices

En une seule ligne de commande, copiez dans votre répertoire **myproject/tmp,** le fichier **test-TP.txt** localisé dans le répertoire **/tmp** de la machine **sbr2**

Solution

```
$ scp sbr2:/tmp/test-TP.txt ~/myproject/tmp
```

Linux initiation | Plan

Notion d'utilisateur & groupe

Dans un système Linux (UNIX) chaque ressource (fichier, répertoire, programme en train de tourner…) est la propriété d'un utilisateur enregistré sur la machine. Cet utilisateur est le propriétaire ou **owner** de cette ressource.
Chaque utilisateur fait partie d'au moins un groupe. Le nombre de groupes auxquels un utilisateur peut appartenir n'est pas limité.
À chaque instant, il n'y a qu'un seul groupe actif pour chaque utilisateur. C'est ce groupe qui sera associé aux ressources auxquelles l'utilisateur tentera d'accéder ou tentera de créer.
Les utilisateurs (resp. les groupes) disposent d'identifiants numériques, les **`uid`** (resp. les **`gid`**).

La commande **`id`** affiche les informations sur l'identité de l'utilisateur qui lance la commande.

```
[stage08@nz ~]$ id
uid=7069(stage08) gid=1013(stage) groups=1013(stage)
```

La commande **`ls -l`** affiche les informations sur la propriété des fichiers et/ou des répertoires.

```
[stage08@nz ~]$ ls -l acteur.csv
-rw-r--r-- 1 stage08 stage 84 May 13 21:19 acteur.csv
```

Propriétaire *(owner)*

Groupe

Notion d'utilisateur & groupe

Les opérations qu'un utilisateur peut effectuer sur une ressource sont définies par les droits dont il dispose en tant qu'utilisateur et en tant que membre des groupes auxquels il appartient.

➡ Tout utilisateur est limité dans ses droits d'accès au système : lecture/écriture de fichiers ; lancement de programmes ; occupation d'espace disque et/ou mémoire.

➡ Un seul utilisateur dispose de tous les droits : l'administrateur système ou **`root`**

Gestion des droits sur les fichiers

La commande **`ls -l`** affiche les informations sur la propriété des fichiers et/ou des répertoires. Les droits sont groupés par paquets de trois caractères (lettres **`r,w,x`** ou **`-`**)

```
[stage08@nz ~]$ ls -l acteur.csv
-rw-r--r-- 1 stage08 stage 84 May 13 21:19 acteur.csv
```

```
[stage08@nz ~]$ ls -l acteur.csv
-rw-r--r-- 1 stage08 stage 84 May 13 21:19 acteur.csv
```

Droits du propriétaire (**`u : user`**)

```
[stage08@nz ~]$ ls -l acteur.csv
-rw-r--r-- 1 stage08 stage 84 May 13 21:19 acteur.csv
```

Droits du groupe (**`g : group`**)

```
[stage08@nz ~]$ ls -l acteur.csv
-rw-r--r-- 1 stage08 stage 84 May 13 21:19 acteur.csv
```

Droits des « autres » (utilisateurs n'appartenant pas au groupe, **`o : others`**)

Gestion des droits sur les fichiers

Position dans le paquet	Lettre	Droit associé	Caractère	Droit associé
1	r	Lecture autorisée	-	Lecture interdite
2	w	Écriture autorisée	-	Écriture interdite
3	x	Exécution autorisée	-	Exécution interdite

Exemple `-rw-r--r--`

Propriétaire `rw-` Lecture, écriture autorisées, exécution interdite

Groupe `r--` Lecture autorisée, écriture et exécution interdites

Autres `r--` Idem que pour le groupe.

Gestion des droits sur les fichiers

La commande **chmod** permet de modifier les droits d'accès aux fichiers. Son premier argument définit les modifications à apporter, et son deuxième argument le fichier (ou répertoire) sur lequel les modifications seront appliquées.

Ex. 1 : rendre un fichier «privé», a.k.a enlever (signe -) les droits de lecture (r), écriture (w) et exécution (x) au groupe (g) et aux autres (o).

```
[stage08@nz ~]$ chmod  go-rwx acteur.csv
-rw------- 1 stage08 stage 84 May 13 21:19 acteur.csv
```

Ex. 2 : autoriser (signe +) le groupe (g) a écrire dans le fichier.

```
[stage08@nz ~]$ chmod  g+w acteur.csv
-rw-rw-r-- 1 stage08 stage 84 May 13 21:19 acteur.csv
```

Ex. 4 : se prémunir contre la modification de fichiers, enlever (signe -) les droits d'écriture (w) pour tous.

```
[stage08@nz ~]$ chmod  -w acteur.csv
-r--r----- 1 stage08 stage 84 May 13 21:19 acteur.csv
```

Ex. 4 : rendre un fichier exécutable : autoriser (signe +) son exécution (x).

```
[stage08@nz ~]$ chmod  +x monprogramme
-rwxr-xr-x 1 stage08 stage 84 May 13 21:19 monprogramme
```

Gestion des droits sur les fichiers – cas des répertoires

Dans l'affichage long de la commande **ls**, les répertoires sont signalés par la lettre **d** avant celles définissant les droits.

```
[stage08@nz ~]$ ls  -l
drwxr-xr-x 4 stage08 stage    4096 May 01 11:41 Linux-Initiation
```

Position dans le paquet	Lettre	Droit associé	Car.	Droit associé
1	**r**	**Lecture *de la liste des fichiers* autorisée**	**–**	**Lecture *de la liste des fichiers* interdite**
2	**w**	**Création, suppression, renommage de fichiers autorisés**	**–**	**Création, suppression, renommage interdits**
3	**x**	**Autorisation d'entrer dans le répertoire (cd) et d'accéder à son contenu**	**–**	**Interdiction d'entrer dans le répertoire et d'accéder à son contenu**

La commande **chmod**, dispose d'une option **-R** (recursive) qui applique les droits à l'ensemble des fichiers sous-répertoires de son argument destination.

Gestion des droits sur les fichiers – changement de groupe

La commande **chgrp**, permet de définir un nouveau groupe pour un fichier ou un répertoire. **Il est nécessaire que l'utilisateur qui exécute la commande appartienne au nouveau groupe.**

```
[stage08@nz ~]$ chgrp  autregroupe acteur.csv
[stage08@nz ~]$ ls -l  acteur.csv
-r--r----- 1 stage08 autregroupe 84 May 13 21:19 acteur.csv
```

Il existe une commande pour changer le propriétaire d'un fichier **(chown)** mais son usage est réservé à l'administrateur système...

Application : exercices

Autorisez l'ensemble du groupe **`stage`** à écrire dans le répertoire **`Linux-Initiation`** et ses sous-répertoires. Vérifiez qu'un(e) voisin(e) peut y déposer/effacer des fichiers. Mais interdisez la modification du fichier **`acteur.csv`**

Application : exercices

Autorisez l'ensemble du groupe **stage** à écrire dans le répertoire **Linux-Initiation** et ses sous-répertoires. Vérifiez qu'un(e) voisin(e) peut y déposer/effacer des fichiers. Mais interdisez la modification du fichier **acteur.csv**

Solution

```
$ chmod -R g+rwx Linux-Initiation
$ chmod g-rwx Linux-Initiation/acteur.csv
```

Linux initiation | Plan

Quelques définitions

Un processus est un programme en cours d'exécution. À chaque fois qu'un utilisateur exécute une commande (lance un programme), le système d'exploitation le charge en mémoire et commence à l'exécuter.

Pour fonctionner correctement, un processus a besoin de **mémoire** et de **temps processeur** (CPU). C'est le rôle du système d'exploitation de réaliser l'arbitrage de ces ressources entre tous les processus qui tournent «en même temps» sur une machine. La **charge** d'une machine est le reflet de l'activité de l'ensemble des processus actifs à un instant donné.

Comme pour les fichiers, un processus possède un **propriétaire** (utilisateur) et un **groupe** ; et des **droits** lui sont associés.

L'utilisateur peut contrôler -dans une certaine mesure- l'exécution de ses processus : l'**arrêter** de force, l'**interrompre** pour le **reprendre** plus tard, modifier sa **priorité**.

Lancement / arrêt / interruption des processus

À chaque fois que l'on lance une commande dans la session courante, un processus est créé, et on ne récupère le contrôle qu'après que la commande se soit terminée ou ait été arrêtée ou interrompue.

```
[stage08@nz ~]$ gedit  acteur.csv
[stage08@nz ~]$
```

On récupère la main après avoir quitté gedit.

Un processus lancé dans la session courante peut être arrêté (brutalement) à l'aide de la combinaison de touches **Ctrl-C**

```
[stage08@nz ~]$ gedit  acteur.csv
^C
[stage08@nz ~]$
```

On récupère la main après avoir fait Ctrl-C. Le processus a été «tué».

Un processus «tué» par **Ctrl-C** a rendu toutes les ressources (mémoire, fichiers ouverts) dont il disposait.

Lancement / arrêt / interruption des processus

Un processus lancé dans la session courante peut être **interrompu** à l'aide de la combinaison de touches **`Ctrl-Z`**

```
[stage08@nz ~]$ gedit  acteur.csv

[1]+  Stopped                 gedit acteur.csv
[stage08@nz ~]$
```

On récupère la main après avoir fait **`Ctrl-Z`**. Le processus a été **interrompu**.

Un processus interrompu est «gelé», il ne libère pas ses ressources (à l'exception du processeur). Il lui est attribué un identifiant de ***job*** *(à ne pas confondre avec l'identifiant de processus, cf. plus loin).*

La commande **`jobs`** permet d'obtenir la liste des processus interrompus dans la session courante.

```
[stage08@nz ~]$ gedit  insulin.fas

[2]+  Stopped                 gedit insulin.fas
[stage08@nz ~]$ jobs
[1]-  Stopped                 gedit acteur.csv
[2]+  Stopped                 gedit insulin.fas
```

Réactivation d'un processus interrompu

La commande **fg** *(foreground)* permet la reprise de l'exécution d'un processus interrompu au point où il a été interrompu. Sans argument, c'est le dernier processus interrompu dont l'exécution reprend. Pour relancer un processus particulier, on peut utiliser l'argument `%` suivi de l'identifiant du job.

```
[stage08@nz ~]$ jobs
[1]-  Stopped                 gedit acteur.csv
[2]+  Stopped                 gedit insulin.fas
[stage08@nz ~]$ fg %1
gedit acteur.csv
```

La commande **bg** *(background)* fonctionne comme **fg** mais **rend immédiatement la main dans la session courante.** Le processus continue de s'exécuter **en arrière-plan.**

```
[stage08@nz ~]$ jobs
[1]-  Stopped                 gedit acteur.csv
[2]+  Stopped                 gedit insulin.fas
[stage08@nz ~]$ bg %2
[2]+ gedit insulin.fas &
[stage08@nz ~]$
```

Lancement d'un processus en arrière-plan

Un processus peut directement être lancé en arrière-plan en ajoutant l'esperluette (**&**) à la ligne de commande.

```
[stage08@nz ~]$ gedit acteur.csv &
[3] 26357
[stage08@nz ~]$
```

On obtient un numéro de job, mais également un identifiant de processus **(PID)**.

Visualisation des processus : ps

La commande **ps** permet d'afficher des informations sur les processus.

Ex. 1 : liste des processus de la session courante.

```
[stage08@nz ~]$ ps
  PID TTY          TIME CMD
16175 pts/14   00:00:00 bash
20693 pts/14   00:00:00 dbus-launch
26357 pts/14   00:00:00 gedit
27505 pts/14   00:00:00 ps
```

L'affichage comprend principalement le PID et le nom de la commande.

Ex. 2 : liste **détaillée** des processus de la session courante : option **-f** *(full)*.

```
[stage08@nz ~]$ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
stage08  16175 16174  0 17:21 pts/14   00:00:00 -bash
stage08  20693     1  0 17:37 pts/14   00:00:00 dbus-launch --autolaunch 9b7328b
stage08  26357 16175  0 17:58 pts/14   00:00:00 gedit insulin.fas
stage08  28638 16175  4 18:06 pts/14   00:00:00 ps -f
```

L'affichage se complète avec l'identifiant de l'utilisateur (UID), le PID du processus père (PPID), l'heure de lancement et les arguments de la commande.

Visualisation des processus : ps

Ex. 3 : liste détaillée des processus d'un utilisateur donné : option **-u** *(user)*.

```
[stage08@nz ~]$ ps -fu stage08
UID        PID  PPID  C STIME TTY          TIME CMD
stage08  16174 16172  0 17:21 ?        00:00:00 sshd: stage08@pts/14
stage08  16175 16174  0 17:21 pts/14   00:00:00 -bash
(…)
stage08  20696     1  0 17:37 ?        00:00:00 /usr/libexec/gconfd-2
stage08  26357 16175  0 17:58 pts/14   00:00:00 gedit insulin.fas
stage08  32327 16175  0 18:20 pts/14   00:00:00 ps -fu stage08
```

Ex. 4 : liste détaillée de tous les processus de la machine. courante : options **-elf** *(extended, long, full)*.

```
[stage08@nz ~]$ ps -elf
0 S uguyet   29185 29184  0  80   0 - 28353 n_tty_ Apr19 pts/35   00:00:00 /bin/bash
0 S lgueguen 30113 10845  0  80   0 - 26364 n_tty_ May12 pts/32   00:00:00 less macros.xml
4 S root     30980  5864  0  80   0 - 28926 unix_s May12 ?        00:00:00 sshd: nhenry
[priv]
5 S nhenry   31153 30980  0  80   0 - 28961 poll_s May12 ?        00:00:01 sshd: nhenry@notty
0 S nhenry   31154 31153  0  80   0 - 14977 poll_s May12 ?        00:00:01
/usr/libexec/openssh/sft
0 S pmandon  31370  2698  0  80   0 - 27641 n_tty_ May12 pts/64   00:00:00 /bin/bash
0 S lberdjeb 31598     1  0  80   0 - 26526 wait   12:22 ?        00:00:00 /bin/sh
/opt/sge/qlogin.
```

Visualisation interactive des processus : top

La commande top affiche, et rafraîchit en continu, une liste des processus tournant sur la machine. Par défaut, cette liste est triée en fonction de la charge (temps passé sur le processeur). Elle renseigne également sur l'état de la machine (charge, mémoire disponible et occupée…).

```
top - 18:31:24 up 171 days,  5:52,  4 users,  load average: 1.19, 1.16, 1.1
Tasks: 650 total,   2 running, 648 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  3.1%ni, 96.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0
Mem:  131915052k total, 130711176k used,  1203876k free,   203212k buffers
Swap:  1048572k total,   220144k used,   828428k free, 128719136k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
20471 roze      39  19  233m 223m 1300 R 100.0  0.2  11778:50 multi
    1 root      20   0 23500 1252 1044 S  0.0  0.0   0:02.36 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:20.70 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   2:38.13 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:06.98 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:12.25 watchdog/0
    7 root      RT   0     0    0    0 S  0.0  0.0   1:53.64 migration/1
    8 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/1
    9 root      20   0     0    0    0 S  0.0  0.0   0:09.10 ksoftirqd/1
   10 root      RT   0     0    0    0 S  0.0  0.0   0:12.57 watchdog/1
```

Charge globale de la machine sur 3 intervalles de temps : 1, 5 et 15 min.

Instantané de l'occupation du processeur.

Instantané de l'état de la mémoire.

Élimination des processus : `kill`

La commande **`kill`** envoie un signal à un processus dont on connaît le PID. En utilisant des options spécifiques (**`-HUP, -TERM, -KILL`**), on lui signale de s'arrêter plus ou moins gentiment.

Ex. 1 : Arrêt brutal d'un processus à l'aide de **`kill`** avec l'option **`-KILL.`**

```
[stage08@nz ~]$ gedit acteur.csv &
[1] 27908
[stage08@nz ~]$ kill -KILL 27908
[stage08@nz ~]$
```

Un utilisateur ne peut utiliser **`kill`** que sur ses propres processus.

Processus et héritage

Chaque processus est lancé à partir d'un processus père. Ex. : les commandes lancées dans la session courante ont comme processus père le processus d'établissement de la connexion.

Le processus de PID 1 est le processus qui, au démarrage de la machine, a engendré tout l'arbre des processus.

Arrêter un processus entraîne l'arrêt de tous ses processus fils (à méditer avant d'utiliser **`kill`**).

Lorsqu'un processus prend fin, le processus père en est informé. Tant que le père ne traite pas l'information sur la suppression de son fils, ce dernier reste à l'état de *zombie.* Les processus *zombie* ne représentent aucune charge pour la machine, sauf s'ils prolifèrent de manière incontrôlée. Un *zombie* ne peut être éliminé qu'en tuant son père. Il est alors rattaché au processus de PID 1 qui se charge du nettoyage des *zombies.*

Application : exercices

Ouvrez une connexion. Lancez la commande **`gedit`** en arrière-plan puis fermez la connexion. Que se passe-t-il ?

Application : exercices

Ouvrez une connexion. Lancez la commande **`gedit`** en arrière-plan puis fermez la connexion. Que se passe-t-il ?

Solution

Le processus correspondant à gedit est tué lors de la fermeture de la connexion : il était un processus fils du processus de connexion.

Processus détachés de la session

Il est fréquent de vouloir lancer des processus dont la durée de vie excède de loin la durée de la session. Il faut donc éviter que ces processus soient tués lors de la fermeture de la session.
La commande **nohup** (*no hang-up)* permet de détacher un processus de la session.

```
[stage08@nz ~]$ nohup mon_long_programme_de_bioinformatique.pl
nohup: ignoring input and appending output to `nohup.out'
```

Les éventuels affichages du programme vont être ajoutés à un fichier nommé **nohup.out**

Pour un programme déjà lancé (en arrière-plan) et dont on connaît le PID, la commande **disown** permet de le «détacher» de la session.

```
[stage08@nz ~]$ mon_long_programme_de_bioinformatique.pl &
[1] 29087
[stage08@nz ~]$ disown 29087
```

Bonus : personnaliser son environnement

Environnement d'une session - SHELL

Chaque commande s'exécute dans un **répertoire courant** et sous l'identité de **l'utilisateur courant.**
Mais une session permet de définir et de personnaliser bien d'autres paramètres.

Lorsque l'on se connecte à une machine, le processus qui gère le connexions lance un interpréteur de commandes, le **`shell.`** C'est le processus qui va afficher l'invite (prompt), attendre que l'utilisateur entre une ligne de commande, l'analyser, et lancer les processus demandés par l'utilisateur.

La personnalisation d'une session se fera au travers de la configuration du **`shell.`**

Le PATH ou chemin de recherche des commandes

Lorsque l'utilisateur entre le nom d'une commande, celle-ci est recherchée dans une suite bien définie de répertoires. La définition de cette suite se fait au travers d'une

variable d'environnement

Cette variable s'appelle **PATH**, et peut être affichée comme suit :

```
[stage08@nz ~]$ echo $PATH
/opt/sge/bin/lx24-amd64:/opt/python/bin:/usr/local/java/bin:/usr/lib64/qt-
3.3/bin:/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sb
in:/usr/sbin:/sbin:/opt/dell/srvadmin/bin:/usr/local/public/bin:/usr/local/genome2/bin:
/usr/local/genome/bin:/usr/local/adm/bin:/usr/local/admin/script:/usr/local/adm/script:
/usr/local/genome/script:/usr/local/genome2/h::/usr/local/genome2/seqclean:/usr/local/g
enome2/seqclean/bin:/usr/local/genome2/tgicl_linux:/usr/local/genome2/tgicl_linux/bin:/
usr/local/genome/emboss/bin:/usr/local/genome/phylip/bin:/usr/local/genome/mgadist:/usr
/local/genome/MUMmer:/usr/local/genome/TMHMM/bin:/usr/local/genome/hmmer/bin:/usr/local
/genome/fasta/bin:/usr/local/genome/mcl64/mcl-05-
321/bin:/usr/local/genome/WoLFPSORT_package_v0.2/bin:/usr/local/genome2/abyss/bin:/usr/
local/cristallo/bin:/home/fr2424/stage/stage08/bin:/opt/openmpi/bin:/opt/6.x/matlab/r20
13b/bin:/opt/6.x/matlab/r2013b/toolbox/abims/ffca:/opt/6.x/matlab/r2013b/toolbox/abims/
mexcdf:/usr/local/cristallo/shelx97:/home/fr2424/stage/stage08/bin
```

Le PATH ou chemin de recherche des commandes

Pour déterminer dans lequel de ces répertoires se trouve une commande que l'on veut lancer, on utilise la commande **which** :

```
[stage08@nz ~]$ which grep
/bin/grep
[stage08@nz ~]$ which java
/usr/local/java/bin/java
```

Pour exécuter une commande ne se trouvant pas dans un des répertoires de **PATH**, il faut spécifier son chemin (absolu ou relatif) :

```
[stage08@nz ~]$ ls -l myproject/script
-rwxr-xr-x 1 stage08 stage 804 May 4 20:58 myproject/script/supercalcul.sh
[stage08@nz ~]$ supercalcul.sh
-bash: supercalcul.sh: command not found
[stage08@nz ~]$ myproject/script/supercalcul.sh
```

Il est également possible d'ajouter au **PATH** de nouveaux répertoires pour que le shell aille y rechercher des commandes.

```
[stage08@nz ~]$ export PATH=~/myproject/script:${PATH}
[stage08@nz ~]$ supercalcul.sh
```

export La nouvelle valeur du **PATH** sera connue de tous les processus de la session

PATH= Une nouvelle valeur va être affectée à **PATH**

~/myproject/script La nouvelle valeur placée au début de **PATH**

:${PATH} L'ancienne valeur de **PATH** est mise au bout du nouveau répertoire

Utilisation des alias

Il peut être utile de paramétrer une commande pour qu'elle en exécute une autre avec un jeu d'options par défaut. On crée alors une nouvelle commande, un alias, qui définit avec quelles options exécuter la première commande.

Ex. 1 : création d'un alias pour que **grep** affiche les correspondances en couleur

```
[stage08@nz ~]$ alias grep='grep --color'
[stage08@nz ~]$ grep Homo insulin.fas
>gi|163659904|ref|NM_000618.3| Homo sapiens insulin-like growth factor 1 (somatomedin
C) (IGF1), transcript variant 4, mRNA
>gi|163659900|ref|NM_001111284.1| Homo sapiens insulin-like growth factor 1
(somatomedin C) (IGF1), transcript variant 2, mRNA
```

Ex. 2 : création d'un alias pour compter les séquences « humaines » dans un fichier multi-FASTA.

```
[stage08@nz ~]$ alias hsc='grep -c -i human'
[stage08@nz ~]$ hsc insulin_vs_nt.blast
373
```

Pérennisation des personnalisations

Les alias définis dans une session ne sont connus que dans cette session. Il en va de même pour les modifications des variables, comme le **`PATH`.**

Il existe un fichier **`~/.bashrc`** qui est interprété à chaque début de connexion.

L'ajout à ce fichier des instructions de modification du **`PATH`**, et des définitions des alias va les rendre actifs pour chaque session.

Si l'on modifie ce fichier en cours de session, il faut exécuter la commande ci-dessous pour que les modifications soient prises en compte dans la session courante

```
[stage08@nz ~]$ source ~/.bashrc
```

Affichage des paramètres d'environnement

La commande **env** affiche l'ensemble des variables d'environnement définies avec leurs valeurs.

```
[stage08@nz ~]$ env
(…)
LANG=fr_FR.UTF-8
GDM_LANG=fr_FR
MANAGERPID=25591
DISPLAY=:0
INVOCATION_ID=964f88f33972481e93c31234b2e4483f
COMPIZ_CONFIG_PROFILE=ubuntu
(…)
```

La commande **alias** affiche l'ensemble des alias actifs.

```
[stage08@nz ~]$ alias
(…)
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
(…)
```

Application : exercices

Ajoutez la personnalisation de grep à votre environnement, et rechargez celui-ci dans la session courante.

Une *cheat sheet* avec l'essentiel des commandes Linux vous sera remise en échange d'un formulaire d'évaluation rempli.