esprit
Ecole Supérieure Privée
d'Ingénierie et de Technologies

## Système d'exploitation 2 (Unix User)

# Chapitre 3 : Le Shell et les commandes de base

**Par : Yness Boukhris**
**2013-2014**

esprit
Ecole Supérieure Privée
d'Ingénierie et de Technologies

# Licence



# Objectifs

- Manipulation des fichiers et des répertoires
- Gérer les liens symboliques et physiques sur un fichier
- Appliquer à travers les commandes les caractères de joker (*,?,[..] ).
- Employer les commandes de recherche des fichiers selon les critères

# Première connexion

# Environnement graphique (1/5)

À l'origine, les systèmes Unix n'étaient pas dotés d'une interface graphique, mais seulement d'une console texte, L'interface graphique est rajoutée sur un système Unix par le biais d'un serveur. Sous Linux, ce serveur s'appelle Xfree, aussi appelé serveur X.

# Environnement graphique (2/5)

- Un serveur X tourne sur l'ordinateur.
- Un utilisateur se connecte grâce à un gestionnaire de connexion (xdm, gdm, kdm, mdkkdm).
- Le gestionnaire de fenêtre (window manager) gère la manipulation et l'apparence des fenêtres.
- Un environnement de bureau (KDE, Gnome) peut être utilisé en plus, pour pouvoir utiliser les barres des tâches et les icônes sur le bureau.

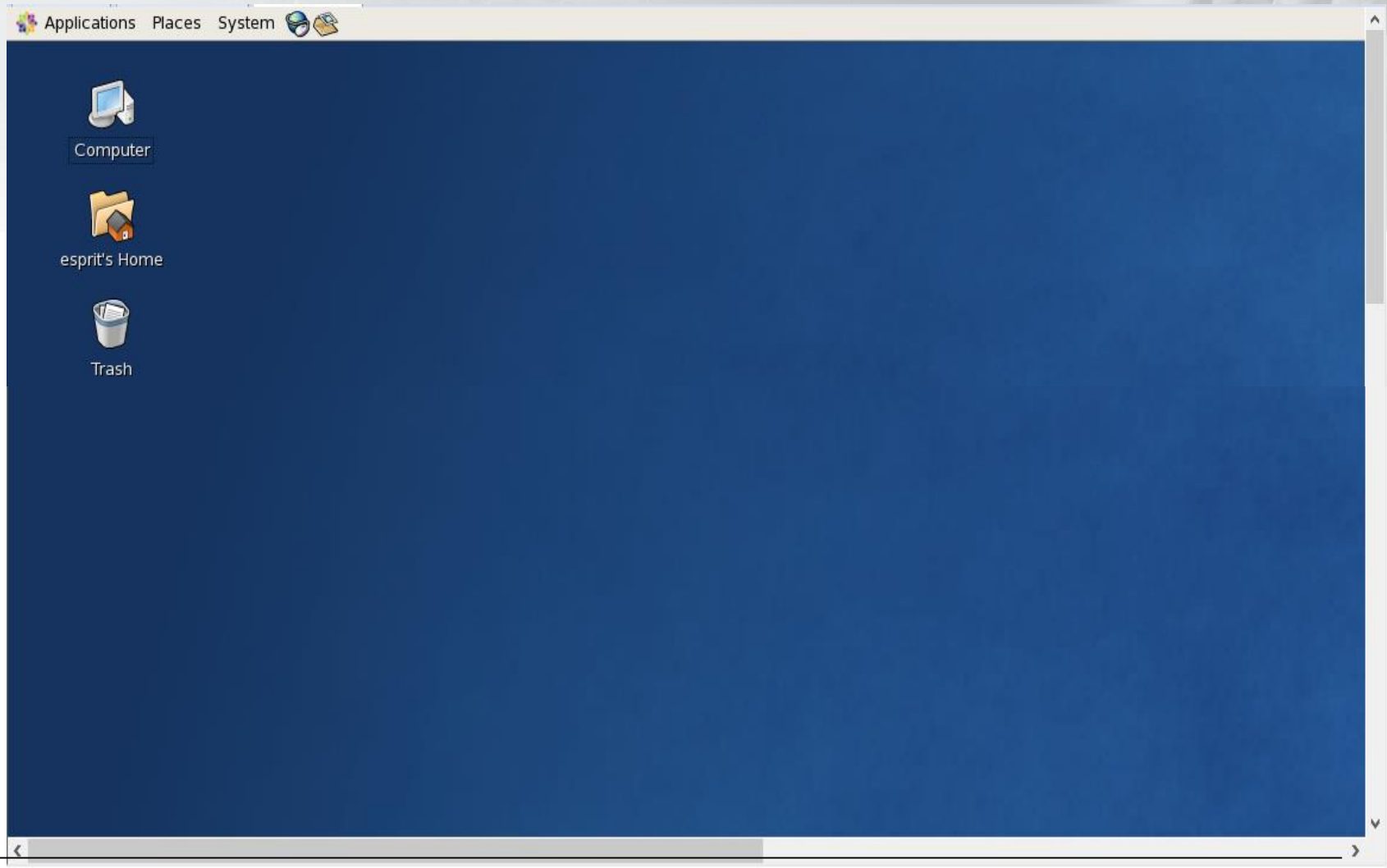
# Environnement graphique (3/5)

- **Environnement de bureau**


**Gnome** (GNU Network Object Model Environment) est un environnement de bureau libre dont l'objectif est de maximiser l'utilisation du système d'exploitation GNU; cette interface est actuellement populaire sur les systèmes entièrement libres,

**KDE** est un environnement de bureau libre. Ce projet a évolué et est devenu multiplate-forme. Cette environnement peut être utilisé sur Linux, BSD également disponible sous Mac OS X, quelques autres UNIX (Solaris notamment) ainsi que Windows

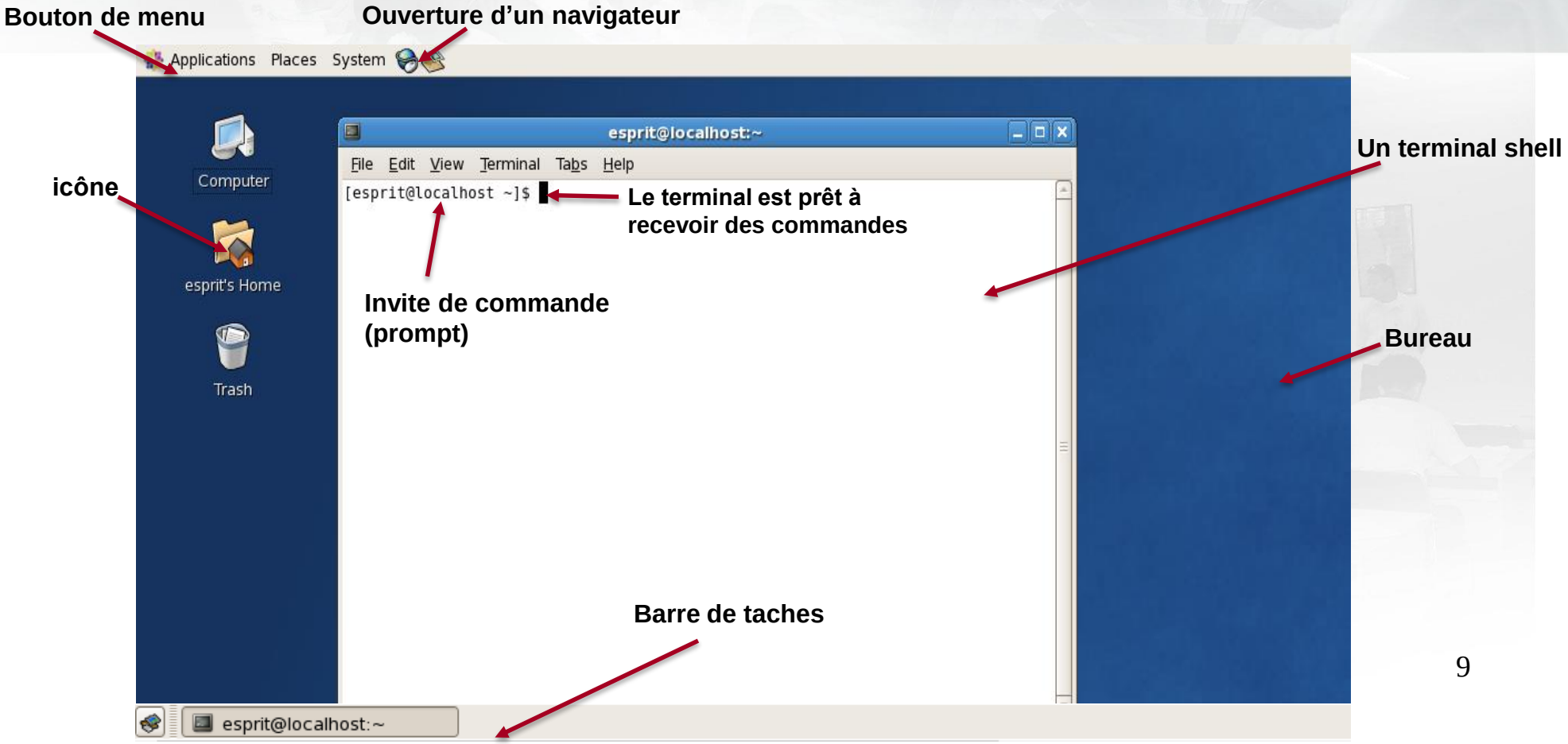
# Environnement graphique (4/5)



# Environnement graphique (5/5)

- **Ouverture d'un terminal (interpreteur de commande) :**

Accédez au menu : applications >accessoires> terminal



# Le shell

# Environnement graphique VS console

## Inconvénients de la ligne de commande

- Difficulté d'usage
- Inefficacité pour les débutants

## Avantages de la ligne de commande

- Automatisation,
- Rapidité d'usage par rapport à l'environnement graphique,
- Consommation de ressources négligeables (CPU, réseaux),
- Modularité et extensibilité,
- Administration et Contrôle d'accès au système.

# Qu'est ce qu'un shell ? (1/3)

➢ **Un langage de programmation**

Un langage de programmation interprété, incluant les notions de variable, d'opérateur arithmétique, de structure de contrôle,..

➢ **Un interpréteur de commande**

- Affiche une invite (prompt),
- Analyse syntaxiquement (découpage en mot).
- Interprète les caractères spéciaux.
- Exécute des commandes,
- Exécute des scripts.

# Qu'est ce qu'un shell ? (2/3)

- **Shell Entrée/Sortie**

Le shell fonctionne au sein d'un terminal. Un terminal est originellement une véritable machine ne disposant que du nécessaire pour saisir des instructions (le clavier) et visualiser les résultats (un écran, voire il y a très longtemps une simple imprimante à papier listing).. [1]

# Qu'est ce qu'un shell ? (3/3)

Il existe plusieurs shells, parmi les plus courants:

- **sh**: Bourne shell est le shell standard pour les scripts.
- **csh**: Berkeley shell
- **ksh**: shell Korn, syntaxe de type Bourne avec des fonctionnalités Berkeley.
- **bash**: "Bourne Again shell", logiciel libre à la manière du shell Korn. Il est conforme à la norme IEEE POSIX P1003.2/ISO 9945.2
- **tcsh:** Libre, le clone de Berkeley avec de nombreuses extensions.

# Qu'est ce qu'un prompt ?

Lors de la connexion au Shell, la première chose qui apparaît est le **prompt ou invite.**

Il est structuré comme suit :


1. Le login ou le nom de l'utilisateur connecté
2. Nom de l'hôte ou de la machine
3. La position actuelle dans le système de fichier
4. Types d'utilisateurs (**$** : utilisateur simple / # : administrateur)

# Qu’est ce qu’une commande ? (1/3)

Une commande est un programme appelé par l’interpréteur de commande.  Certaines d'entre elles sont fournies directement par le shell, alors que d'autres sont des exécutables situés dans l’arborecence.

Toutes les lignes de commande ont cette syntaxe:

**command [options] [arguments]**

# Qu'est ce qu'une commande ? (2/3)

- **Types de commande**

  ▸ **Les commandes externes**

  Ce sont des programmes binaires présents en tant que fichiers sur votre disque dur, Quand vous exécutez la commande, ce fichier est chargé en mémoire et lancé en tant que processus (cette notion sera expliquée au niveau du 6ème chapitre).

  ➢ **Les commandes internes**

  Ces commandes sont internes au shell et exécutées au sein de celui-ci. Ces commandes font parties du programme shell.

# Qu'est ce qu'une commande ? (3/3)

- **Types de commande (…)**

  - **Les commandes d'identification**

  - **which** : recherche les exécutables des commandes selon les chemins indiqués au niveau de la variable PATH. Elle retourne comme résultat le chemin absolu de la commande externe ou indique qu'il n'y a pas de chemin vers la commande interne,

  - **type** : cette commande permet de retourner soit le chemin absolu de la commande externe ou d'indiquer qu'il s'agit d'une commande interne (shell builtin),

  - **whereis** : même fonctionnement que la commande which mais elle affiche en plus le chemin vers les pages manuelles des commandes spécifiées en paramètres.

# Configuration du shell

# Configuration du shell (1/2)

Comme tout autre programme, le shell se base sur des fichiers de configuration. Pour le Bash, les deux types de fichiers de configuration utilisés pour la personnalisation de l'environnement sont :

- **Fichiers de configuration utilisateur**: **~/.bashrc** et **~/.bash_profile**. Chaque utilisateur dispose de ses propres fichiers dans son répertoire personnel.

- **Fichiers de configuration globaux**: **/etc/profile** et **/etc/bash.bashrc**. Le contenu est modifié que par l'administrateur et sera appliqué sur tous les utilisateurs du système. [1]

# Configuration du shell (2/2)

Dans les fichiers d'environnement, il est possible de:

- Positionner des chemins de recherche
- Modifier umask
- Positionner le type de terminal et l'initialiser
- Définir des variables d'environnement et des alias…

# Les variables d'environnement (1/4)

- Variables dynamiques stockant des données et des options de programmes, des informations sur l'ordinateur qui peuvent être utilisées par les programmes pour modifier le comportement de ces derniers selon la manière appropriée à l'environnement courant.

- L'utilisation de ces variables est définie par chaque programme, certaines variables définissent des fonctions précises:

# Les variables d'environnement (2/4)

- **PATH** : indique la liste des chemins où trouver les programmes à lancer.
- **SHELL** : indique l'interpréteur shell utilisé par défaut.
- **HOME** : contient le chemin absolu vers le répertoire personnel de l'utilisateur connecté.
- **LOGNAME** : pour le nom de l'utilisateur
- **HISTFILE** : pour le fichier historique.
- **HISTSIZE** : pour la limite de commandes historiques accessibles.
- **EDITOR** pour l'éditeur de ligne de commandes.

# Les variables d'environnement (3/4)

- Pour afficher la valeur d'une variable**:**

**$ echo $VAR_ENV**

- Pour déclarer des variables d'environnement

**$ VARNAME=valeur**
**$ export VARNAME**

export rend la variable visible pour tous les shells et les programmes

- Pour remplacer les deux commandes précédentes:

**$ export VARNAME=valeur**

- Pour effacer le contenu d'une variable

**$ unset VARNAME**

- Afficher toutes les variables d'environnements

**$ env**

# prompt ? (4/4)

La plupart des shells vous permettent de personnaliser votre prompte et votre environnement.

- Pour le Bash, la variable qui définit le prompte est **PS1**.

```
[esprit@localhost ~]$ echo $PS1
[\u@\h \W]\$
```

# Les alias

- Pour afficher la valeur d'une variable**:**

**$ echo $VAR_ENV**

- Pour déclarer des variables d'environnement

**$ VARNAME=valeur**

**$ export VARNAME**

export rend la variable visible pour tous les shells et les programmes

- Pour remplacer les deux commandes précédentes:

**$ export VARNAME=valeur**

- Pour effacer le contenu d'une variable

**$ unset VARNAME**

- Afficher toutes les variables d'environnements

**$ env**

# Les Commandes de base

## 1- Utilitaires d'aide

# Les utilitaires d'aide (1/7)

- Il existe une multitude de commandes sous linux. Il est donc impossible pour un être humain de se rappeler de l'intégralité de ces dernières.

- C'est ainsi qu'un ensemble d'utilitaires d'aide ont été mis à la disposition des utilisateurs pour faciliter la manipulation de cet environnement.

# Les utilitaires d'aide (2/7)

## ➢ Les pages manuelles (Man)

Cette commande permet de fournir une page manuelle d'une commande envoyé en argument. Dans cet exemple, la commande man retourne un descriptif de la commande ls ainsi qu'une liste d'options qu'on peut associer à cette commande,

```
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.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.

       -a, --all
              do not ignore entries starting with .
```

# Les utilitaires d'aide (3/7)

## ➢ Les pages infos (info)

Cette commande permet de fournir une page info d'une commande envoyé en argument :

```
File: coreutils.info,  Node: cp invocation,  Next: dd invocation,  Up: Basic op\
erations

11.1 `cp': Copy files and directories
=====================================

`cp' copies files (or, optionally, directories).  The copy is
completely independent of the original.  You can either copy one file to
another, or copy arbitrarily many files to a destination directory.
Synopses:

     cp [OPTION]... [-T] SOURCE DEST
     cp [OPTION]... SOURCE... DIRECTORY
     cp [OPTION]... -t DIRECTORY SOURCE...

   * If two file names are given, `cp' copies the first file to the
     second.
```

# Les utilitaires d'aide (4/7)

## ➢ La commande apropos

apropos est une commande POSIX qui permet de lister les manuels dont la description comprend les mots passés en arguments.

- Chaque page de manuel comporte une courte description;
- apropos recherche et affiche cette description pour chaque page correspondant au mot-clé qu'on lui fournie.

```
[esprit@localhost ~]$ apropos copy
bcopy                (3)  - copy byte sequence
bcopy                (3p)  - memory operations (LEGACY)
bcopy [bstring]      (3)  - byte string operations
copysign             (3)  - copy sign of a number
copysign             (3p)  - number manipulation function
copysignf [copysign] (3)  - copy sign of a number
copysignf [copysign] (3p)  - number manipulation function
copysignl [copysign] (3)  - copy sign of a number
copysignl [copysign] (3p)  - number manipulation function
cp                   (1)  - copy files and directories
cp                   (1p)  - copy files
```

# Les utilitaires d'aide (5/7)

## La commande history

La commande **history** vous permet de visualiser les commandes saisies au niveau de votre terminal. Les commandes sont sauvegardées dans un fichier de configurations du shell, nommé **.bash_history** qui se trouve au niveau de votre répertoire personnel.
La commande associée à l'option « -c » nous permet d'effacer l'historique de la session courante.

```
[esprit@localhost ~]$ history
    1  ls
    2  rm file2
    3  cd rep1
    4  cat file1
    5  mv rep2 /tmp
    6  clear
    7  history
```

# Les utilitaires d'aide (6/7)

## La touche tabulation

Tapez le début de votre commande et en appuyant sur la touche TAB ↹, Linux vous la complète ou vous propose les différentes possibilités pour la compléter, à défaut vous aurez droit à un petit bip, s'il existe un trop grand nombre de propositions.

## La combinaison ctrl+R

Apparaît alors une invite permettant la recherche sous la forme suivante :

```
(reverse-i-search)``:
```

La recherche se fait de manière incrémentale. C'est à dire qu'au fur et à mesure que l'on tape des caractères, la plus récente ligne de l'historique trouvée sera affichée. La touche Entrée permet d'exécuter la commande affichée.

# Les utilitaires d'aide (7/7)

## ➢ Utilitaires divers

- **date :** affiche la date,
- **cal :** affiche le calendrier,
- **who :** affiche les utilisateurs connectés,
- **whoami :** affiche votre nom d'utilisateur,
- **uname -a :** affiche toutes les informations sur le système,
- **whatis :** affiche une description de la commande,
- **hostname:** afficher le nom de la machine
- **uptime:** afficher depuis combien de temps le système n'a pas rebooté.
- **halt** ou **shutdown -h now** ou **init 0**: éteindre l'ordinateur.
- **reboot** ou **shutdown -r now** ou **init 6**: rebooter l'ordinateur.
- **lsb_release –d :** afficher le nom de la distribution

# Les Commandes de base

## 2- Manipulation des fichiers et des répertoires

# Manipulation des fichiers et des répertoires (1/7)

## ➢ La commande cd

Cette commande nous permet de nous déplacer dans l'arborescence. Sans paramètre, elle nous renvoie vers le répertoire personnel de l'utilisateur actuel, sinon au chemin spécifié en argument,

Avec les arguments listés ci-dessous, elle permet de :

. : se déplacer dans le répertoire courant

.. : se déplacer vers le répertoire parent

- : se déplacer vers le répertoire précédent

/ : se déplacer vers le répertoire précédent

~ : se déplacer vers le répertoire personnelle de l'utilisateur courant,

# Manipulation des fichiers et des répertoires (2/7)

## La commande pwd

Cette commande permet d'afficher le chemin absolu du répertoire courant.

```
[esprit@localhost ~]$ pwd
/home/esprit
```

## La commande ls

Cette commande nous permet principalement de lister le contenu d'un répertoire. Nous pouvons lui associer plusieurs options dont les plus évidentes sont :

| Option | Rôle |
|---|---|
| **-l** | Lister les métadonnées relatives à un fichier ou un répertoire, |
| **-a** | Afficher tous les fichiers, y compris les fichiers cachés |
| **-R** | Pour assurer un affichage récursif (le contenu du répertoire et de ses sous répertoires) |
| **-i** | Afficher le numéro d'i-node des fichiers et des répertoires |
| **-d** | Afficher les informations d'un dossier |

# Manipulation des fichiers et des répertoires (3/7)

## La commande touch

Cette commande permet de créer un ou plusieurs fichiers vides. Dans cet exemple la commande touch crée les fichiers file1 ainsi que file2, spécifiés en paramètres

```
[esprit@localhost ~]$ touch file1 file2
```

## La commande mkdir

Cette commande permet la création d'un répertoire sous le chemin spécifié en paramètres

```
[esprit@localhost ~]$ mkdir /home/esprit/rep1
```

Pour la création d'une arborescence, il faut ajouter l'option « -p »

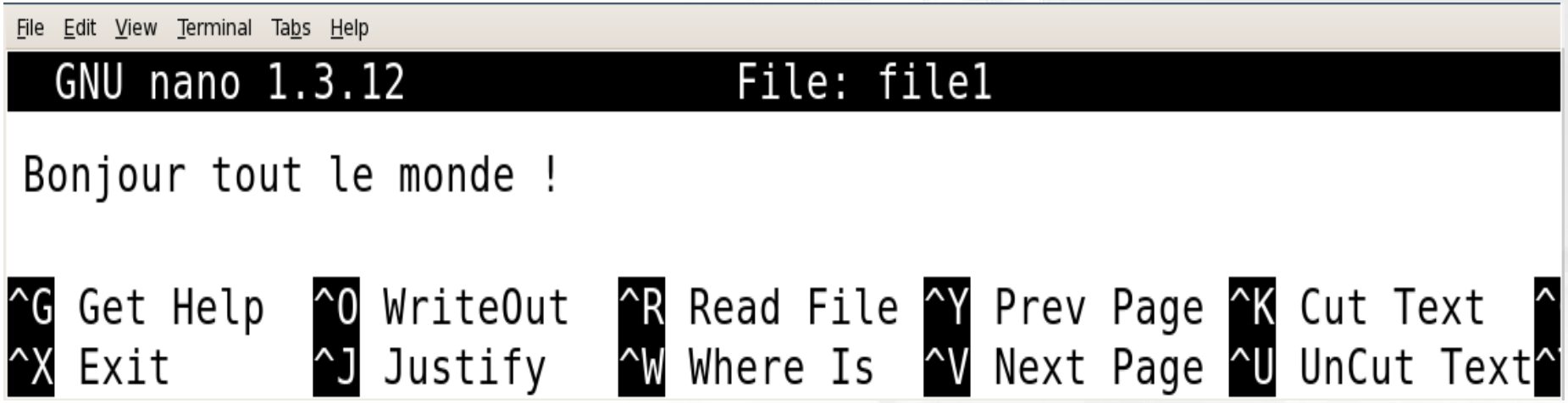
```
[esprit@localhost ~]$ mkdir -p rep2/cours/SE2
```

# Manipulation des fichiers et des répertoires (4/7)

## La commande nano

C'est une commande d'édition de fichier. Un **éditeur de texte** est un programme qui permet de modifier des fichiers de texte brut, sans mise en forme (gras, italique, souligné…)

En bas de votre écran, vous pouvez voir un espace d'aide, il s'agit d'un aide-mémoire pour vous rappeler à tout moment les commandes principales que vous pouvez lancer sous Nano.



# Manipulation des fichiers et des répertoires (5/7)

Le symbole ^ signifie Ctrl (la touche Contrôle de votre clavier). Ainsi, pour quitter Nano, il suffit de taper Ctrl + X. Voici les raccourcis les plus importants :

- **Ctrl + K** : couper la ligne de texte (et la mettre dans le presse-papier) ;
- **Ctrl + U** : coller la ligne de texte que vous venez de couper ;
- **Ctrl + C** : afficher à quel endroit du fichier votre curseur est positionné (numéro de ligne…) ;
- **Ctrl + W** : rechercher dans le fichier ;
- **Ctrl + X** : quitter Nano.

# Manipulation des fichiers et des répertoires (6/7)

## ➢ La commande cp

Cette commande permet de copier une entité (fichier/répertoire) d'un emplacement à un autre,

```
[esprit@localhost ~]$ cp file1 rep1/
```

Dans cet exemple, nous allons copié le fichier « file1 » dans le répertoire « rep1 », Si nous voulons copier un répertoire, il faut associer à cette commande l'option « -R »,

```
[esprit@localhost ~]$ cp -R rep2 rep1/
```

## ➢ La commande mv

Cette commande permet de déplacer un fichier ou un répertoire, d'un répertoire à un autre.

```
[esprit@localhost ~]$ mv rep1 /tmp/
```

# Manipulation des fichiers et des répertoires (7/7)

## La commande rmdir

Cette commande permet de supprimer un répertoire vide.

```
[esprit@localhost ~]$ ls -l  rep2/
total 0
[esprit@localhost ~]$ rmdir rep2/
```

## La commande rm

Cette commande permet de supprimer une entité (fichier/répertoire)

```
[esprit@localhost ~]$ rm file1
```

Dans cet exemple, nous allons supprimer le fichier « file1 », si nous voulons copier un répertoire, il faut associer à cette commande l'option « -R »,

```
[esprit@localhost ~]$ rm -R rep1
```

# Les liens (1/7)

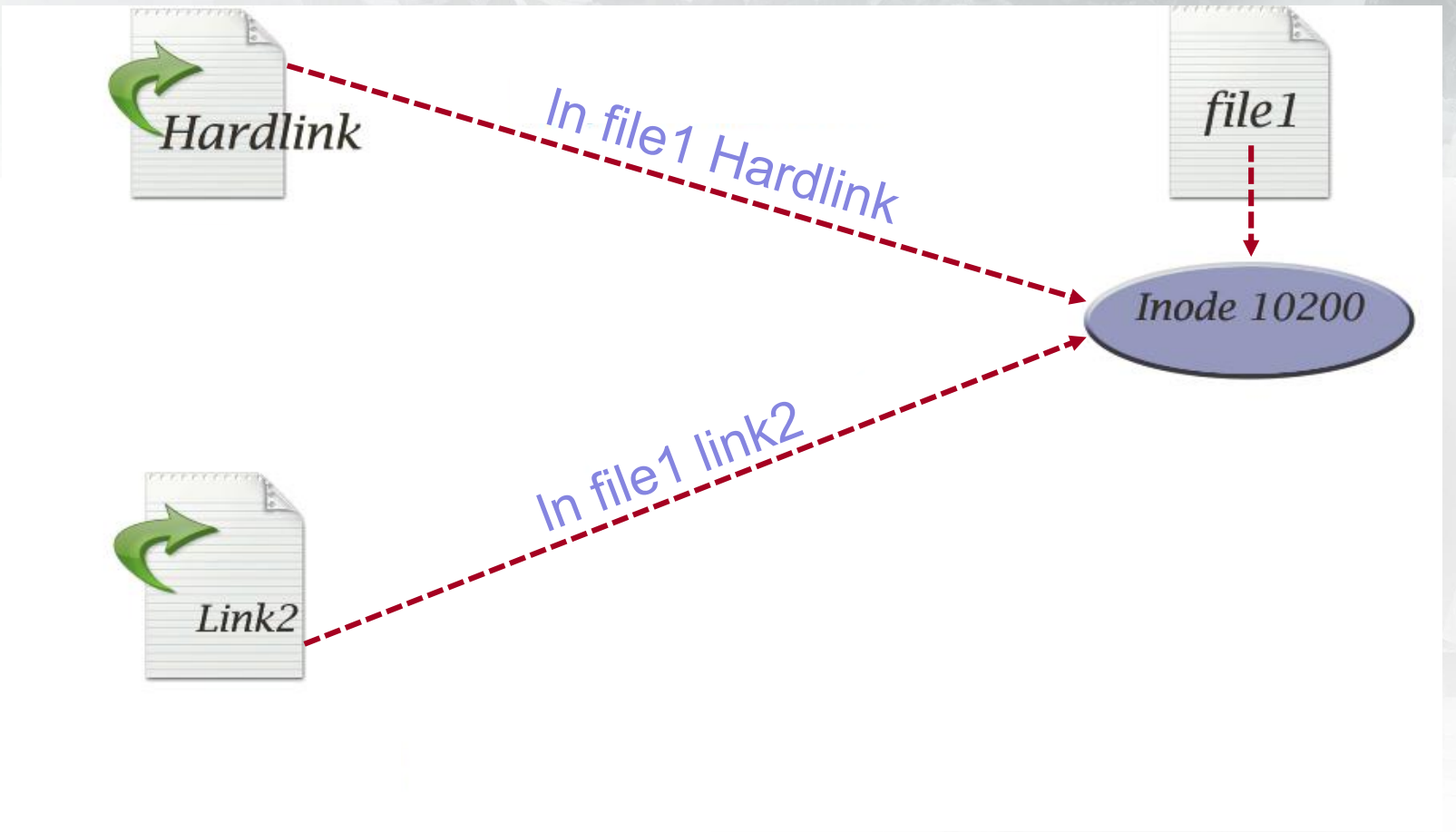
Un lien est un fichier spécial qui permet de faire plusieurs références à un même fichier sur le disque dur. L'intérêt est de pouvoir accéder à un fichier de plusieurs endroits de l'arborescence. Il existe deux types de liens : les liens physiques et les liens symboliques.

## ➢ Les liens physiques

Le système de fichier de Linux enregistre les fichiers sur la base d'un numéro et pas sur la base d'un nom. Grâce à un lien physique, un fichier peut avoir plusieurs noms/chemin d'accès, à un même fichier en pointant sur un numéro unique d'**inode** qui identifie le fichier dans le système de fichiers comme l'indique le schéma suivant :

# Les liens (2/7)

## ➢ Les liens physiques (…)







2014 -Yness Boukhris

# Les liens (3/7)

## Les liens physiques (…)

La commande utilisée pour créer un hard link est la commande **ln**.

```
[esprit@localhost ~]$ ln file1 hardlink
[esprit@localhost ~]$ ln file1 link2
[esprit@localhost ~]$ ls -li file1 hardlink link2
9910 -rw-rw-r--. 3 esprit esprit 1 Sep  4 09:50 file1
9910 -rw-rw-r--. 3 esprit esprit 1 Sep  4 09:50 hardlink
9910 -rw-rw-r--. 3 esprit esprit 1 Sep  4 09:50 link2
```

Le numéro affiché en utilisant la commande ls –i est le numéro d'inode.

→On remarque que les liens physiques créés pointent sur le même inode

# Les liens (4/7)

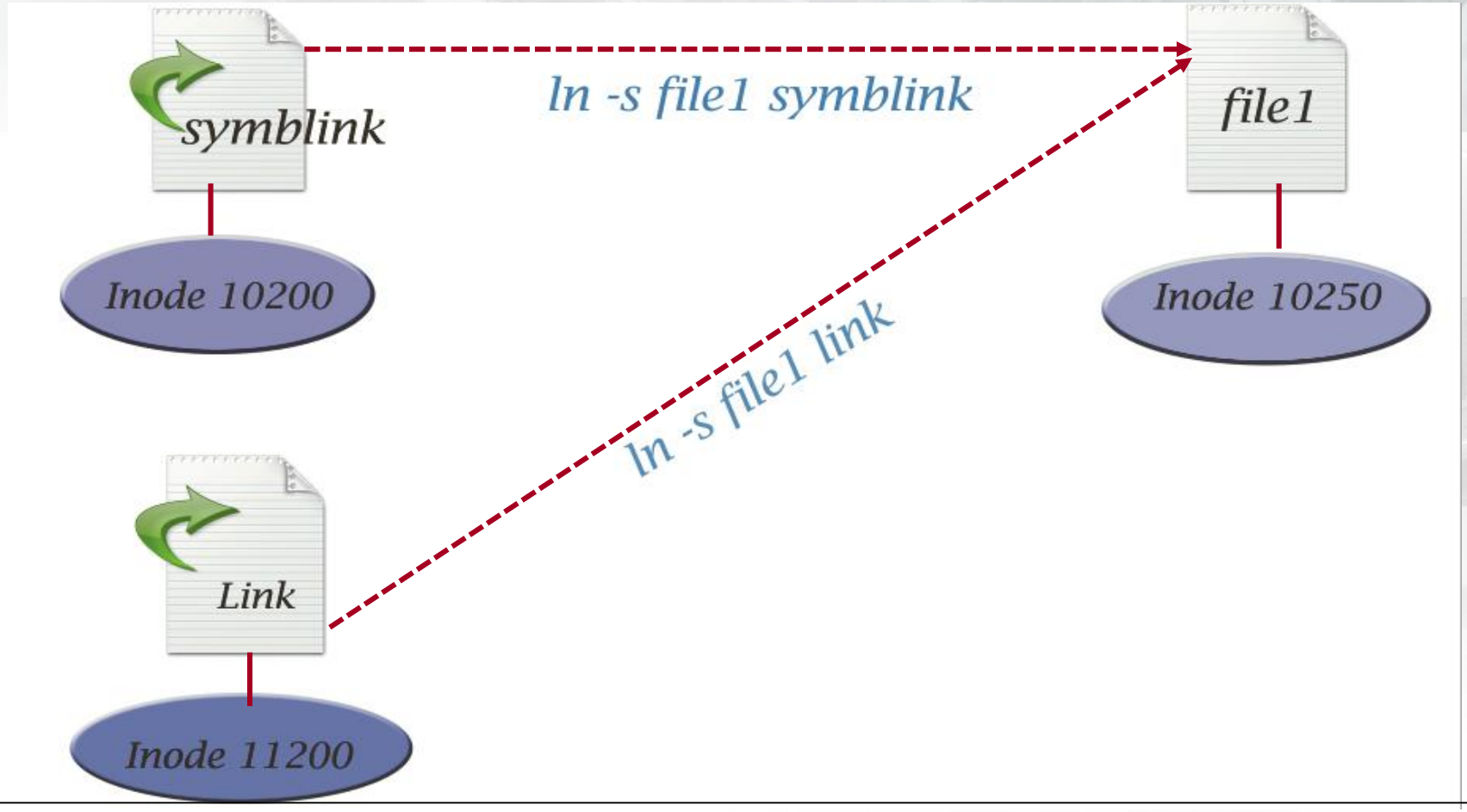
## Les liens symboliques (…)

Les liens symboliques ressemblent plus aux « raccourcis » sous Windows. La plupart du temps, on crée des liens symboliques sous Linux pour faire un raccourci, et non des liens physiques.

Le principe du lien symbolique est que l'on crée un lien vers un autre nom de fichier. Cette fois, on pointe vers le nom de fichier et non vers l'inode directement (figure suivante).

# Les liens (5/7)

## Les liens symboliques (…)



# Les liens (6/7)

## Les liens symboliques (…)

La commande utilisée pour créer un lien symbolique est la commande **ln avec l'option -s**.

```
[esprit@localhost ~]$ ln -s file1 symblink
[esprit@localhost ~]$ ln -s file1 link
[esprit@localhost ~]$ ls -li file1 symblink link
 9910 -rw-rw-r--. 3 esprit esprit 1 Sep  4 09:50 file1
10560 lrwxrwxrwx. 1 esprit esprit 5 Sep  5 04:22 link -> file1
10537 lrwxrwxrwx. 1 esprit esprit 5 Sep  5 04:22 symblink -> file1
```

**Le numéro affiché en utilisant la commande ls –i est le numéro d'inode.**

**→On remarque que les liens symboliques créés pointent sur des inodes différents.**

# Les liens (7/7)

## ➢ Différences entre liens physiques et syboliques

- Un lien physique utilisant le même numéro d'inode, il se limite à la même partition.
- Pour traverser les limites des partitions, Unix introduit des liens dits symboliques.
- La création d'un lien symbolique s'effectue avec l'option −s de la commande ln.
- Un lien physique pointe sur le même inode que le fichier contrairement au lien symbolique
- Un lien physique ne peux être créé que sur des fichier
- Un lien symbolique peut être créé sur un fichier comme sur un répertoire

# Références

# Webographie

**[1] : P**réparation à la certification LPIC1 : Linux **- Sébastien Rohaut**
**[2] :** http://doc.ubuntu-fr.org/