INSTITUT Mines-Télécom

TELECOM SudParis

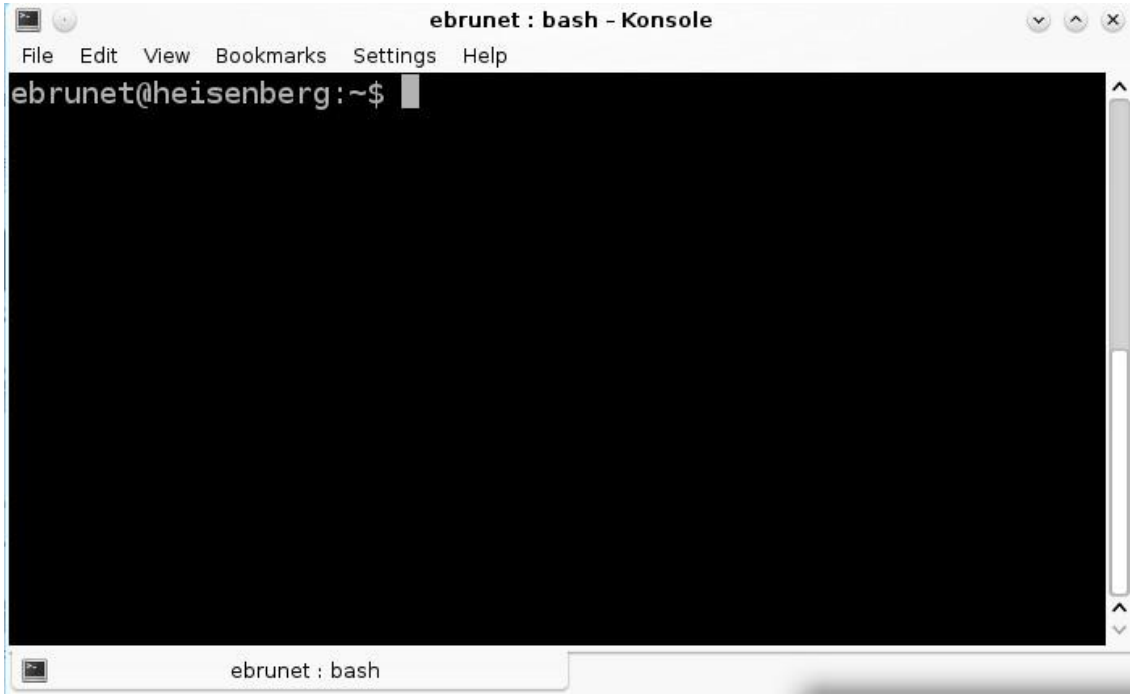
CI4-Shell

CSC3102 – Introduction aux systèmes d'exploitation

Elisabeth Brunet

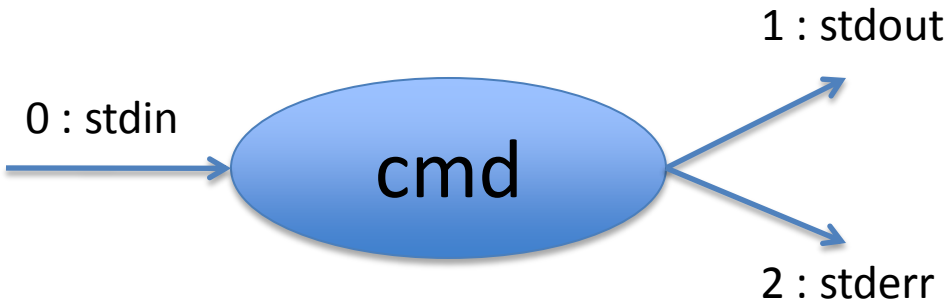
08/09/2015

Terminal



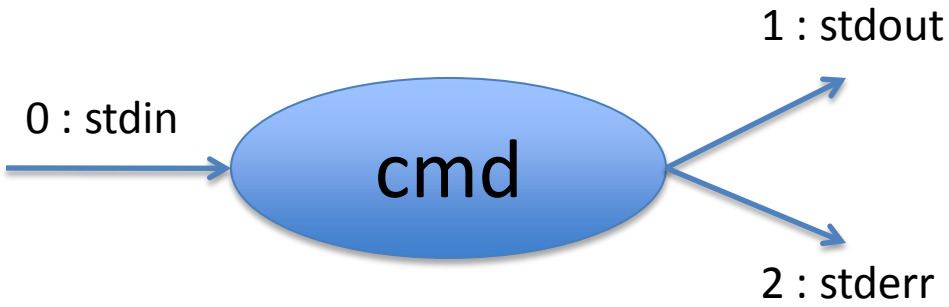
- Shell = Interpréteur de commande
  - Lit les commandes saisies par l'utilisateur dans le terminal,
    - Commande + options + arguments
  - Les interprète,
    - Identification de la commande, des options, analyse des caractères spéciaux
  - Les exécute,
  - Et transmet les résultats à l'utilisateur.

Entrée/Sorties standards



- Trois flux standards d'entrée/sorties
  - `stdin` : *standard input*
    - passage des arguments de la commande
  - `stdout` : *standard output*
    - Transmission des résultats de la commande à l'utilisateur
  - `stderr` : *standard error*
    - Messages d'erreurs générés par l'exécution de la commande

Entrée/Sorties standards



- Trois flux standards d'entrée/sorties
  - `stdin` : *standard input* → par défaut, clavier
    - passage des arguments de la commande
  - `stdout` : *standard output* → par défaut, l'écran
    - Transmission des résultats de la commande à l'utilisateur
  - `stderr` : *standard error* → par défaut, l'écran
    - Messages d'erreurs générés par l'exécution de la commande

Périphériques et pseudo-périphériques : des fichiers spéciaux

- Périphériques : Matériel physique connecté à l'unité centrale
  - Écran, clavier, souris, réseau, usb, etc.
- Pseudo-périphériques :
  - Virtuels : écrans virtuels (ou bureaux), partitions
  - + Pseudo-périphériques standards dont
    - `null` : qui sert de poubelle irrémédiable (!! Différent de la poubelle placée en icône sur le bureau)
    - `tty` : qui permet de repérer les terminaux

Périphériques et pseudo-périphériques : des fichiers spéciaux

- Périphériques : Matériel physique connecté à l'unité centrale
  - Écran, clavier, souris, réseau, usb, etc.
- Pseudo-périphériques :
  - Virtuels : écrans virtuels (ou bureaux), partitions
  - + Pseudo-périphériques standards dont
    - `null` : qui sert de poubelle irrémédiable (!! Différent de la poubelle placée en icône sur le bureau)
    - `tty` : qui permet de repérer les terminaux

- Accès aux (pseudo-)périphériques via le système de fichiers
  - Opérations classiques d'ouverture, fermeture, lecture, écriture uniformes à tout accès à une entrée du SF
  - Opérations fournies par le pilote du périphérique
  - Accès depuis le répertoire `/dev`
    - `/dev/null`, `/dev/lp` pour accéder à l'iimprimante, etc.

Redirection de la sortie standard

- Détournement de `stdout` vers un autre fichier que l'écran
  - Valable uniquement pour la commande concernée
- Caractère spécial « > »
  - `cmd > fichier`

```
$
```

Redirection de la sortie standard

- Détournement de `stdout` vers un autre fichier que l'écran
  - Valable uniquement pour la commande concernée
- Caractère spécial « > »
  - `cmd > fichier`

```
$ echo titi
titi
$ echo titi > titi.txt
$ more titi.txt
titi
$
```

Redirection de la sortie standard

- Détournement de `stdout` vers un autre fichier que l'écran
  - Valable uniquement pour la commande concernée
- Caractère spécial « > »
  - `cmd > fichier`
  - Écrit au début du fichier
    - !! Écrase le contenu du fichier

```
$ echo titi
titi
$ echo titi > titi.txt
$ more titi.txt
titi
$ echo tutu > titi.txt
$ more titi.txt
tutu
$
```

Redirection de la sortie standard

- Détournement de `stdout` vers un autre fichier que l'écran
  - Valable uniquement pour la commande concernée
- Caractère spécial « > »
  - `cmd > fichier`
  - Écrit au début du fichier
    - !! Écrase le contenu du fichier
- Double caractère « >> »
  - Écrit à la fin du fichier

```
$ echo titi
titi
$ echo titi > titi.txt
$ more titi.txt
titi
$ echo tutu > titi.txt
$ more titi.txt
tutu
$
```

Redirection de la sortie standard

- Détournement de `stdout` vers un autre fichier que l'écran
  - Valable uniquement pour la commande concernée

- Caractère spécial « > »
  - `cmd > fichier`
  - Écrit au début du fichier
    - !! Écrase le contenu du fichier

- Double caractère « >> »
  - Écrit à la fin du fichier

```
$ echo titi
titi
$ echo titi > titi.txt
$ more titi.txt
titi
$ echo tutu > titi.txt
$ more titi.txt
tutu
$ echo tata >> titi.txt
$ more titi.txt
tutu
tata
$
```

Redirection de la sortie standard

- Détournement de `stdout` vers un autre fichier que l'écran
  - Valable uniquement pour la commande concernée
- Caractère spécial « > »
  - `cmd > fichier`
  - Écrit au début du fichier
    - !! Écrase le contenu du fichier
- Double caractère « >> »
  - Écrit à la fin du fichier
- IIEMC « Redirection stdout »

```
$ echo titi
titi
$ echo titi > titi.txt
$ more titi.txt
titi
$ echo tutu > titi.txt
$ more titi.txt
tutu
$ echo tata >> titi.txt
$ more titi.txt
tutu
tata
$
```

Redirection de la sortie standard d'erreur

- Détournement de `stderr` vers un autre fichier que l'écran
  - Très utile pour isoler et traiter les messages d'erreur
  - Valable uniquement pour la commande concernée

- Chaîne « `2>` »
  - `cmd 2> fichier_erreur`
  - `stderr` est numérotée 2

```
$
```

Redirection de la sortie standard d'erreur

- Détournement de `stderr` vers un autre fichier que l'écran
  - Très utile pour isoler et traiter les messages d'erreur
  - Valable uniquement pour la commande concernée

- Chaîne « `2>` »
  - `cmd 2> fichier_erreur`
  - `stderr` est numérotée 2

```
$ rm fichier_inexistant
rm : impossible de supprimer
<<fichier_inexistant>> : Aucun
fichier ou dossier de ce type
$
```

Redirection de la sortie standard d'erreur

- Détournement de `stderr` vers un autre fichier que l'écran
  - Très utile pour isoler et traiter les messages d'erreur
  - Valable uniquement pour la commande concernée
- Chaîne « `2>` »
  - `cmd 2> fichier_erreur`
  - `stderr` est numérotée 2
- IIEMC « Redirection stderr »

```
$ rm fichier_inexistant
rm : impossible de supprimer
<<fichier_inexistant>> : Aucun
fichier ou dossier de ce type
$ rm fichier_inexistant 2>
fic_err.txt
$ more fic_err.txt
rm : impossible de supprimer
<<fichier_inexistant>> : Aucun
fichier ou dossier de ce type
$
```

Redirection de l'entrée standard

- Acquisition de `stdin` depuis un autre fichier que le clavier
  - Utilisation de données d'entrée stockées dans un fichier plutôt que d'exiger leur saisie au clavier par l'utilisateur
  - Valable uniquement pour la commande concernée

- Caractère spécial « < »
  - `cmd < fic`

```
$
```

Redirection de l'entrée standard

- Acquisition de `stdin` depuis un autre fichier que le clavier
  - Utilisation de données d'entrée stockées dans un fichier plutôt que d'exiger leur saisie au clavier par l'utilisateur
  - Valable uniquement pour la commande concernée

- Caractère spécial « < »
  - `cmd < fic`

```
$ read jour heure degre
```

Redirection de l'entrée standard

- Acquisition de `stdin` depuis un autre fichier que le clavier
  - Utilisation de données d'entrée stockées dans un fichier plutôt que d'exiger leur saisie au clavier par l'utilisateur
  - Valable uniquement pour la commande concernée

- Caractère spécial « < »
  - `cmd < fic`

```
$ read jour heure degre
01/01/2040 00:00:00 0
$
```

Redirection de l'entrée standard

- Acquisition de `stdin` depuis un autre fichier que le clavier
  - Utilisation de données d'entrée stockées dans un fichier plutôt que d'exiger leur saisie au clavier par l'utilisateur
  - Valable uniquement pour la commande concernée

- Caractère spécial « < »
  - `cmd < fic`

```
$ read jour heure degre
01/01/2040 00:00:00 0
$ echo « Le $jour à $heure, il fera $degre
°. »
Le 01/01/2040 à 00:00:00, il fera 0 °.
$
```

Redirection de l'entrée standard

- Acquisition de `stdin` depuis un autre fichier que le clavier
  - Utilisation de données d'entrée stockées dans un fichier plutôt que d'exiger leur saisie au clavier par l'utilisateur
  - Valable uniquement pour la commande concernée

- Caractère spécial « < »
  - `cmd < fic`

```
$ read jour heure degre
01/01/2040 00:00:00 0
$ echo « Le $jour à $heure, il fera $degre
°. »
Le 01/01/2040 à 00:00:00, il fera 0 °.
$ more data.txt
 25/06/2017 12:00:00 37
$ read jour heure degre < data.txt
$
```

Redirection de l'entrée standard

- Acquisition de `stdin` depuis un autre fichier que le clavier
  - Utilisation de données d'entrée stockées dans un fichier plutôt que d'exiger leur saisie au clavier par l'utilisateur
  - Valable uniquement pour la commande concernée

- Caractère spécial « < »
  - `cmd < fic`

- IIEMC « Redirection de stdin »

```
$ read jour heure degre
01/01/2040 00:00:00 0
$ echo « Le $jour à $heure, il fera $degre
°. »
Le 01/01/2040 à 00:00:00, il fera 0 °.
$ more data.txt
 25/06/2017 12:00:00 37
$ read jour heure degre < data.txt
$ echo « Le $jour à $heure, il fera $degre
° »
Le 25/06/2017 à 12:00:00, il fera 37 °.
$
```

Variables d'environnement du shell

- Variables standards aux noms réservés modifiables
- Consultation grâce à la commande `env`
- Variables notables :
  - `HOME` : chemin absolu du répertoire de connexion
    - `cd` , `cd ~` et `cd $HOME` sont des commandes équivalentes
  - `PS1` : prompt (par défaut `$`)
  - `PATH` : liste des répertoires de recherche des commandes
    - Entre chaque chemin, séparateur « : »

Exécution d'une commande shell

- Recherche de son fichier exécutable
  - Si c'est une commande interne : exécute la commande
  - Sinon si l'utilisateur donne le chemin de la commande
    - Exécute la commande désignée
  - Sinon
    - Parcours de la liste de répertoires de la variable d'env. `PATH`
      - ➢ Ordre significatif
      - ➢ Permet le choix de la version d'une commande
      - Si trouvée : exécution
      - Sinon, message d'erreur « `Command not found` »

- `which` : affiche le chemin absolu de l'exécutable sélectionné
  - `which ls` → chemin absolu vers le fichier exécutable de la commande `ls`

Alias de commande

- Sert à (re)définir le nom d'une commande
  - Pour créer des noms abrégés ou des mnémoniques

- Création :

  `cmd alias`

- Suppression :

  `cmd unalias`

```
$
```

Alias de commande

- Sert à (re)définir le nom d'une commande
  - Pour créer des noms abrégés ou des mnémoniques

- Création :

  `cmd alias`

- Suppression :

  `cmd unalias`

```
$ ls
Fic1   repA   test.txt
$
```

Alias de commande

- Sert à (re)définir le nom d'une commande
  - Pour créer des noms abrégés ou des mnémoniques

- Création :

  `cmd alias`

- Suppression :

  `cmd unalias`

```
$ ls
Fic1  repA  test.txt
$ alias ll=''ls -l''
$
```

Alias de commande

- Sert à (re)définir le nom d'une commande
  - Pour créer des noms abrégés ou des mnémoniques

- Création :

  `cmd alias`

- Suppression :

  `cmd unalias`

- IIEMC « Alias »

```
$ ls
Fic1  repA  test.txt
$ alias ll=''ls -l''
$ ll
-rw-r—r- brunet_e 171 Sep 01 08:00
Fic1
drw-r—r- brunet_e 93 Sep 01 08:00
repA
-rw-r—r- brunet_e 659 Sep 01 08:00
test.txt
$
```

Fichiers de configuration du shell

- Exécutés automatiquement au démarrage du shell
  - La prise en compte d'une modification de configuration impose le redémarrage du terminal (ou l'utilisation de la cmd `source`)
- Configuration
  - Globale du système d'exploitation par l'administrateur
    - Fichier `/etc/profile`
  - Pour son compte par l'utilisateur
    - Fichier `~/.bashrc` (+ d'autres fichiers non utilisés dans ce cours)
- Opérations pérennes typiquement réalisées :
  - Affectation de variables d'environnement : `PATH`, `PS1`, etc.
  - Déclaration de variables liées à des logiciels installés en sus
  - Création d'alias
  - Positionnement du masque des droit d'accès
  - Etc.

Conclusion

- Concepts clés :
  - Entrée/Sorties de commande
  - Redirections des E/S : `>`, `>>`, `2>`, `<`
  - Variables d'environnement : `HOME`, `PS1`, `PATH`
  - Recherche de l'exécutable d'une commande
  - Alias
  - Fichiers de configuration : `.bashrc`
- Commandes clés :
  - `env`
  - `alias`, `unalias`
  - `which`

En route pour le TP!