# M1105 - Systèmes d'exploitation

## Interpréteur de commande

sebastien.kramm@univ-rouen.fr

IUT de Rouen, dépt. Réseaux & Télécoms

Version du 9 novembre 2016

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

# Sommaire



INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

# Introduction : besoin ?

Un OS doit permettre à un utilisateur humain de lancer des programmes :

- programmes applicatifs de type desktop (interactifs)
- programmes applicatifs de type serveur (non interactifs)
- Utilitaires de gestion
- etc.

Ceci se fait via une **Interface Homme - Machine** (IHM)
ou *User Interface* (UI) en anglais
Deux paradigmes de contrôle :

- Interface graphique ou *Graphical User Interface* : **GUI**
- Interface en ligne de commande ou *Command Line Interface* : **CLI**

Dans les deux cas, le programme permettant cette interaction peut être intégré à l'OS ou distinct.

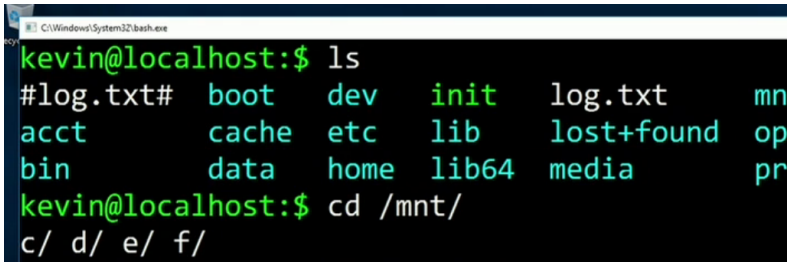
- Windows : GUI et CLI complètement intégré à l'OS
- Linux : GUI et CLI indépendant du noyau, mais lié au choix de la distribution

# Notions de "Shell"

- Definition : couche logicielle qui fournit l'interface utilisateur d'un système d'exploitation.
- En pratique : un shell est un **interpréteur de commande** qui convertit des chaînes de caractères dans un langage codifié en une action sur le système.
- Par extension, peut désigner aussi l'**émulateur de terminal** : console interactive permettant de saisir au clavier ces commandes et d'avoir à l'écran la **sortie** de la commande.

# Shell : interpréteur de commandes & console

- Linux & systèmes Unix (Mac OSX) :
  - les deux sont dissociés
  - plusieurs shells historique : `sh` , `csh` , `tcsh` , `ksh` , ...
  - shell dominant aujourd'hui : Bourne Shell, appelé **bash**
- Windows : les deux sont liés
  - Historiquement : `COMMAND.COM` sur MSDOS et Windows 3.1
  - Depuis XP/NT : programme `CMD.EXE`. Toujours existant, mais developpement arrêté.
  - Depuis Windows 7 : apparition de PowerShell : approche "objet" (non traité ici)
  - 2016 : annonce par MS du portage de Bash sur Windows.



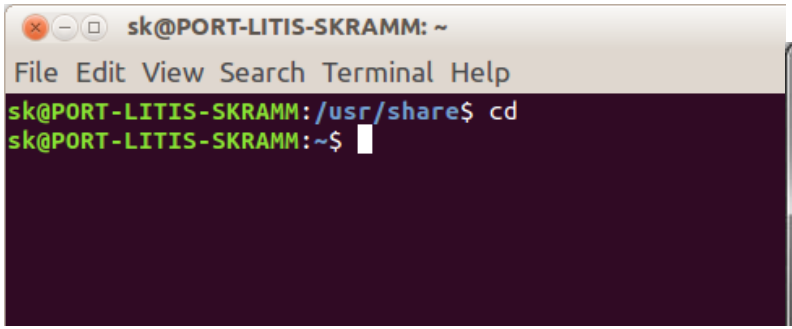

- Ce cours traite des deux en parallèle, les spécificités de chaque seront vues plus tard.

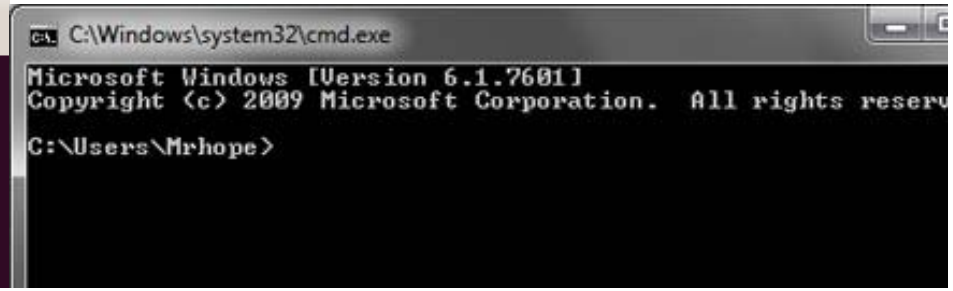
INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Utilisation de la console

De façon à peu près universelle, des facilités sont intégrées aux programmes "console" :

- Autocomplétion via la touche TAB
- Historique des commandes, via ⇑ (fleche haut curseur)
- "Glisser-deposer" depuis un explorateur de fichier GUI
- Personnalisation du "prompt"




INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

# Utilisation interactive ou "script"

- Le shell peut être utilisé de façon
  - interactive : répétition du cycle : commande → affichage
  - non-interactive : notion de "script".
- Un script est un fichier texte contenant une suite de commandes : forme de "programme exécutable"
- Peut ensuite être exécuté :
  - en saisissant son nom dans une console ;
  - via GUI (double clic sur le fichier).

  (sous réserve qu'il soit **accessible** depuis le dossier courant...)
- Certaines constructions grammaticales n'ont de sens que dans un contexte de script (boucles, ...)
- Windows : certaines syntaxes ne sont pas identiques en interactif et en script...

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Utilisation interactive ou "script"

- Le shell peut être utilisé de façon
  - interactive : répétition du cycle : commande → affichage
  - non-interactive : notion de "script".
- Un script est un fichier texte contenant une suite de commandes : forme de "programme exécutable"
- Peut ensuite être exécuté :
  - en saisissant son nom dans une console ;
  - via GUI (double clic sur le fichier).

  (sous réserve qu'il soit **accessible** depuis le dossier courant...)
- Certaines constructions grammaticales n'ont de sens que dans un contexte de script (boucles, ...)
- Windows : certaines syntaxes ne sont pas identiques en interactif et en script...

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Utilisation interactive ou "script"

- Le shell peut être utilisé de façon
  - interactive : répétition du cycle : commande → affichage
  - non-interactive : notion de "script".
- Un script est un fichier texte contenant une suite de commandes : forme de "programme exécutable"
- Peut ensuite être exécuté :
  - en saisissant son nom dans une console ;
  - via GUI (double clic sur le fichier).

  (sous réserve qu'il soit **accessible** depuis le dossier courant...)
- Certaines constructions grammaticales n'ont de sens que dans un contexte de script (boucles, ...)
- Windows : certaines syntaxes ne sont pas identiques en interactif et en script...

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Utilisation interactive ou "script"

- Le shell peut être utilisé de façon
  - interactive : répétition du cycle : commande → affichage
  - non-interactive : notion de "script".
- Un script est un fichier texte contenant une suite de commandes : forme de "programme exécutable"
- Peut ensuite être exécuté :
  - en saisissant son nom dans une console ;
  - via GUI (double clic sur le fichier).

  (sous réserve qu'il soit **accessible** depuis le dossier courant...)
- Certaines constructions grammaticales n'ont de sens que dans un contexte de script (boucles, ...)
- Windows : certaines syntaxes ne sont pas identiques en interactif et en script...

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Utilisation interactive ou "script"

- Le shell peut être utilisé de façon
  - interactive : répétition du cycle : commande → affichage
  - non-interactive : notion de "script".
- Un script est un fichier texte contenant une suite de commandes : forme de "programme exécutable"
- Peut ensuite être exécuté :
  - en saisissant son nom dans une console ;
  - via GUI (double clic sur le fichier).

  (sous réserve qu'il soit **accessible** depuis le dossier courant...)
- Certaines constructions grammaticales n'ont de sens que dans un contexte de script (boucles, ...)
- Windows : certaines syntaxes ne sont pas identiques en interactif et en script...

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Utilisation interactive ou "script"

- Le shell peut être utilisé de façon
  - interactive : répétition du cycle : commande $\rightarrow$ affichage
  - non-interactive : notion de "script".
- Un script est un fichier texte contenant une suite de commandes : forme de "programme exécutable"
- Peut ensuite être exécuté :
  - en saisissant son nom dans une console ;
  - via GUI (double clic sur le fichier).

  (sous réserve qu'il soit **accessible** depuis le dossier courant...)
- Certaines constructions grammaticales n'ont de sens que dans un contexte de script (boucles, ...)
- Windows : certaines syntaxes ne sont pas identiques en interactif et en script...

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Utilisation interactive ou "script"

- Le shell peut être utilisé de façon
  - interactive : répétition du cycle : commande $\rightarrow$ affichage
  - non-interactive : notion de "script".
- Un script est un fichier texte contenant une suite de commandes : forme de "programme exécutable"
- Peut ensuite être exécuté :
  - en saisissant son nom dans une console ;
  - via GUI (double clic sur le fichier).

  (sous réserve qu'il soit **accessible** depuis le dossier courant...)
- Certaines constructions grammaticales n'ont de sens que dans un contexte de script (boucles, ...)
- Windows : certaines syntaxes ne sont pas identiques en interactif et en script...

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Utilisation interactive ou "script"

- Le shell peut être utilisé de façon
  - interactive : répétition du cycle : commande $\rightarrow$ affichage
  - non-interactive : notion de "script".
- Un script est un fichier texte contenant une suite de commandes : forme de "programme exécutable"
- Peut ensuite être exécuté :
  - en saisissant son nom dans une console ;
  - via GUI (double clic sur le fichier).

  (sous réserve qu'il soit **accessible** depuis le dossier courant...)
- Certaines constructions grammaticales n'ont de sens que dans un contexte de script (boucles, ...)
- Windows : certaines syntaxes ne sont pas identiques en interactif et en script...

# Utilisation interactive ou "script"

- Le shell peut être utilisé de façon
  - interactive : répétition du cycle : commande $\rightarrow$ affichage
  - non-interactive : notion de "script".
- Un script est un fichier texte contenant une suite de commandes : forme de "programme exécutable"
- Peut ensuite être exécuté :
  - en saisissant son nom dans une console ;
  - via GUI (double clic sur le fichier).

  (sous réserve qu'il soit **accessible** depuis le dossier courant...)
- Certaines constructions grammaticales n'ont de sens que dans un contexte de script (boucles, ...)
- Windows : certaines syntaxes ne sont pas identiques en interactif et en script...

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Différences Linux / Windows

## Windows

- pensé dès l'origine pour une utilisation en GUI : le pilotage du système via une CLI n'a pas été prioritaire ;
- encore beaucoup de manipulations doivent se faire en GUI.

## Linux

- inspiration : systèmes UNIX ;
- pensé dès le départ comme pilotable via CLI ;
- GUI complètement dissociée du noyau ;
- beaucoup de manipulations ne peuvent se faire que via CLI.

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Différences Linux / Windows

## Windows

- pensé dès l'origine pour une utilisation en GUI : le pilotage du système via une CLI n'a pas été prioritaire ;
- encore beaucoup de manipulations doivent se faire en GUI.

## Linux

- inspiration : systèmes UNIX ;
- pensé dès le départ comme pilotable via CLI ;
- GUI complètement dissociée du noyau ;
- beaucoup de manipulations ne peuvent se faire que via CLI.

# Commande internes et externes

- Un shell peut utiliser des commandes **internes** ou **externes**.
  - commande interne : exécutée directement par le shell lui-même
  - commande externe : programme distinct, lancé par le shell lorsque l'utilisateur le demande.
    Doit correspondre à un fichier contenant un programme exécutable, et qui soit **accessible** depuis le dossier courant.
- Pour l'OS, pas de différence entre une commande externe et un programme : les deux sont vus comme des **fichiers exécutables**.
- Linux :
  - Liste des commandes internes : `help`
  - Commande interne / externe ? : `type commande`
- Windows : les commandes externes sont des programmes stockés dans `c:\Windows\system32`

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Comparaison Bash/CMD

- Programme "terminal" (console)
  - Windows : rudimentaire et peu évolué. Versions alternatives disponibles.
  - Linux : grand choix de programmes.
- Scripts Windows : beaucoup plus rudimentaire
  - moins de commandes internes
  - pas de structure de controle évoluées (fonctions, boucle "while")
  - support et documentation de MS aléatoire et variant suivant les versions de Windows, documentation éclatée.
- bash
  - Syntaxe parfois obscure
- Windows : insensible à la casse

# Sommaire



INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

# Syntaxe générale

- Syntaxe générale : `commande [option] [argument]` (sur **une** ligne)
  - Windows : les options sont données avec `/`
    Exemple : `dir /w`
  - Bash : les options sont données avec `-` ("option courte") ou `--` ("option longue")
    Exemple : `ls -a` ou `ls --all`
- Le caractère SPC (ASCII :32, $20) est utilisé comme séparateur
  ⇒ Si nom de fichier avec des espaces, il sera nécessaire de les encadrer par des "
  Exemples :
  - `commande mon super fichier.txt` : illégal
  - `commande "mon super fichier.txt"` : ok
- On peut interrompre une commande en cours (ou annuler l'édition en cours) avec **CTRL-C**.

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Syntaxe générale

- Syntaxe générale : `commande [option] [argument]` (sur **une** ligne)
  - Windows : les options sont données avec `/`
    Exemple : `dir /w`
  - Bash : les options sont données avec `-` ("option courte") ou `--` ("option longue")
    Exemple : `ls -a` ou `ls --all`
- Le caractère SPC (ASCII :32, $20) est utilisé comme séparateur
  $\Rightarrow$ Si nom de fichier avec des espaces, il sera nécessaire de les encadrer par des "
  Exemples :
  - `commande mon super fichier.txt` : illégal
  - `commande "mon super fichier.txt"` : ok
- On peut interrompre une commande en cours (ou annuler l'édition en cours) avec **CTRL-C**.

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Syntaxe générale

- Syntaxe générale : `commande [option] [argument]` (sur **une** ligne)
  - Windows : les options sont données avec `/`
    Exemple : `dir /w`
  - Bash : les options sont données avec `-` ("option courte") ou `--` ("option longue")
    Exemple : `ls -a` ou `ls --all`
- Le caractère SPC (ASCII :32, $20) est utilisé comme séparateur
  $\Rightarrow$ Si nom de fichier avec des espaces, il sera nécessaire de les encadrer par des "
  Exemples :
  - `commande mon super fichier.txt` : illégal
  - `commande "mon super fichier.txt"` : ok
- On peut interrompre une commande en cours (ou annuler l'édition en cours) avec **CTRL-C**.

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Caractères génériques

- Beaucoup de commandes acceptent des caractères génériques dans une spécification de nom de fichier :
  - `*` : un ou plusieurs caractère(s)
  - `?` : un seul caractère
- Exemples
  - `DIR a*.txt` : affiche la liste de tous les fichiers commençant par a et ayant l'extension `txt`
  - `DIR ???.mp3` : tous les fichiers mp3 de 3 lettres
  - `DIR IUT*.pdf` : tous les fichiers pdf avec IUT comme 1$^{res}$ lettres

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

# Navigation et manipulation fichiers

| | Linux | Windows |
|---|---|---|
| Copie de fichier | `cp source dest` | `copy source dest` |
| Copie de dossiers | `cp source dest` | `xcopy source dest` |
| Renommage de fichier | `mv ancien nouveau` ("*move*") | `ren ancien nouveau` ("*rename*") |
| Déplacement de fichier | `mv ancien nouveau` | `move ancien nouveau` |
| Effacement | `rm fich` | `del fich` ("*delete*") |
| Afficher le contenu d'un fichier | `cat fich` | `type fich` |

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Répertoires

| | Linux | Windows |
|---|---|---|
| Changement répertoire actif | `cd un/chemin` | `cd un\chemin` |
| déplacement rép. actif vers le rep. parent | `cd ..` | `cd ..` |
| Création répertoire | `mkdir rep/sousrep` | `md rep\sousrep` |
| Effacement répertoire | `rmdir rep` | `rmdir rep` |
| Affichage du rép. courant | `pwd` [1] | `cd` |

1. pwd : *Print Working Directory*

# Autre commandes

| | Linux | Windows |
|---|---|---|
| Aide sur commande | `man commande` | `help commande` ou `commande /?` |
| terminer une session/un script | `exit` | `exit` |
| Affichage de texte | `echo` | `echo` |

- Commentaires dans un script :
  - Linux : toute ligne commencant par `#`
  - Windows : toute ligne commencant par `::` ou `REM`

# Sommaire



INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

# E/S standard

- Par défaut, chaque processus se voit attribué des "flots" d'E/S
  - un flot d'entrée, associée aux entrées clavier : `stdin`
  - un flot de sortie standard (1), associé à la console : `stdout`
  - un flot de sortie d'erreur (2), associé à la console : `stderr`
- Ces différents flots peuvent être **redirigés** ponctuellement vers un autre périphérique ou dans un fichier
- Exemple 1 : redirection vers l'imprimante de `stdout` (1)
  - Linux : `commande > /dev/lp` ou `commande 1> /dev/lp`
  - Windows : `commande > LPT1` ou `commande 1> LPT1`
- Exemple 2 : redirection de la sortie d'erreur vers le néant
  - Windows : `commande 2> NUL` (NUL : périphérique "spécial")
  - Linux : périphérique virtuel dédié : `commande 2> /dev/null`

  Intérêt : suppression de certains affichages

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# E/S standard

- Par défaut, chaque processus se voit attribué des "flots" d'E/S
  - un flot d'entrée, associée aux entrées clavier : `stdin`
  - un flot de sortie standard (1), associé à la console : `stdout`
  - un flot de sortie d'erreur (2), associé à la console : `stderr`
- Ces différents flots peuvent être **redirigés** ponctuellement vers un autre périphérique ou dans un fichier
- Exemple 1 : redirection vers l'imprimante de `stdout` (1)
  - Linux : `commande > /dev/lp` ou `commande 1> /dev/lp`
  - Windows : `commande > LPT1` ou `commande 1> LPT1`
- Exemple 2 : redirection de la sortie d'erreur vers le néant
  - Windows : `commande 2> NUL` (NUL : périphérique "spécial")
  - Linux : périphérique virtuel dédié : `commande 2> /dev/null`

  Intérêt : suppression de certains affichages

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# E/S standard

- Par défaut, chaque processus se voit attribué des "flots" d'E/S
  - un flot d'entrée, associée aux entrées clavier : `stdin`
  - un flot de sortie standard (1), associé à la console : `stdout`
  - un flot de sortie d'erreur (2), associé à la console : `stderr`
- Ces différents flots peuvent être **redirigés** ponctuellement vers un autre périphérique ou dans un fichier
- Exemple 1 : redirection vers l'imprimante de `stdout` (1)
  - Linux : `commande > /dev/lp` ou `commande 1> /dev/lp`
  - Windows : `commande > LPT1` ou `commande 1> LPT1`
- Exemple 2 : redirection de la sortie d'erreur vers le néant
  - Windows : `commande 2> NUL` (NUL : périphérique "spécial")
  - Linux : périphérique virtuel dédié : `commande 2> /dev/null`

  Intérêt : suppression de certains affichages

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Redirection vers un fichier

- Très souvent, on redirige vers un fichier
- Redirection en sortie :
  - `commande > fich` : le fichier `fich` est crée (effacé s'il existe déjà)
  - `commande >> fich` : la sortie de `commande` est ajoutée à la fin de `fich`

# Enchainement de commandes

- Exécuter deux commandes à la suite :
  - Linux : `prog1; prog2`
  - Windows : `prog1 & prog2`
- Exécuter deux commandes en parallèle (Linux seulement) :
  `prog1 & prog2`
- Exécuter une 2e commande uniquement si la 1re n'échoue pas (code de retour = 0) :
  `prog1 && prog2`
- Exécuter une 2e commande uniquement si la 1re échoue (code de retour != 0) :
  `prog1 || prog2`
- Utiliser comme entrée d'une 2e commande la sortie d'une 1re :
  `prog1 | prog2`
  (ceci s'appelle un "*pipe*")

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Sommaire



INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

# Variables

- Comme n'importe quel langage de programmation, on peut définir des **variables**
- Quel que soit le shell, les variables sont toujours de type "texte" (chaînes de caractères)
- Mais possibilité de faire de l'arithmétique élémentaire.
- On distingue :
  - variables locales : connues uniquement dans le script.
  - variables d'environnement : prédéfinies par l'OS.

**Attention**

Les variables crées dans un shell ne sont connues **que** dans celui-ci.

- En cas d'appel d'un sous-shell, les variables ne sont pas transmises.
- la création de variables d'environnement passe par des commandes spécifiques.

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Variables

- Comme n'importe quel langage de programmation, on peut définir des **variables**
- Quel que soit le shell, les variables sont toujours de type "texte" (chaînes de caractères)
- Mais possibilité de faire de l'arithmétique élémentaire.
- On distingue :
  - variables locales : connues uniquement dans le script.
  - variables d'environnement : prédéfinies par l'OS.

## Attention

Les variables crées dans un shell ne sont connues **que** dans celui-ci.

- En cas d'appel d'un sous-shell, les variables ne sont pas transmises.
- la création de variables d'environnement passe par des commandes spécifiques.

# Manipulation de variables

| | Linux | Windows |
|---|---|---|
| Donner une valeur à une variable | `mavar=un` | `SET mavar=un` |
| Afficher une variable | `echo "mavar=$mavar"` | `echo mavar=%mavar%` |
| Saisie au clavier | `read var` | `SET /P var=texte_invite` |

- Test de variable
  - Windows : `IF [/I] [NOT] item1==item2 command`

    Exemple : `if %mavar% == coucou echo ok`

    Si espaces ; alors il faut "quoter" :

    `if "%mavar%" == "Ah Que Coucou" echo ok`
  - Linux/Bash :

    Action si égalité : `if [ item1 = item2 ] command`

    Action si différence : `if [ item1 != item2 ] command`

# Variables d'environnement

- Chaque OS définit ses variables d'environnement (VE).
- Permet de connaitre depuis un script des informations sur le système.
- Liste complète :
  Linux : `env` Windows : `set`
- Facilités :
  - Windows : `set ABC` : renvoie toutes les VE commencant par "ABC"
  - Linux : `env | grep ABC` : renvoie toutes les VE contenant la chaîne "ABC"
- Exemples :

| | Linux | Windows |
|---|---|---|
| Nom de l'utilisateur courant | `USER` | `USERNAME` |
| Localisation des applications | `PATH` | `PATH` |
| Dossier pour fichier temporaires | (néant) [2] | `TEMP` |
| Date | (néant) | `DATE` |
| "Home" de l'utilisateur courant | `HOME` | `HOMEPATH` |

- Windows : plus de VE (Linux a des programmes dédiés à la place)

2. Inutile, c'est toujours `/tmp`...

# Variable `PATH`

- Contient la liste des chemins absolus où l'OS va chercher la commande demandée.
- Avec une saisie de type `aaa bbb ccc`, l'OS va rechercher la commande `aaa` dans l'ordre suivant :
  1. est-ce une commande interne ?
  2. est-ce un fichier exécutable dans le dossier courant ? (Windows seulement)
  3. est-ce un programme qu'on trouve dans les chemins donnés dans la variable `PATH`, en faisant la recherche dans l'ordre qui est donné ?
- A défaut, il faudra donner le chemin (absolu ou relatif), par exemple :
  - Windows : `"c:\program files\superappli\aaa" bbb ccc`
  - Linux :
    `/home/mon_id/dev/mes_applis/aaa bbb ccc`
    `../../ici/labas/aaa bbb ccc`
- Linux : si le programme à exécuter est dans le dossier courant, alors il **faut** le spécifier explicitement en tapant `./commande`

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Recherche de commande : différences Windows/Linux

## Windows

- Un fichier exécutable est identifié comme tel par son extension.
- Les principales : `BAT`, `CMD` ou `EXE`, mais aussi beaucoup d'autres (voir la VE `PATHEXT`)
- On peut l'appeler uniquement par son nom, l'OS va chercher s'il trouve une correspondance, dans **cet** ordre.
  Attention : un dossier contient deux fichiers, `bidule.bat` et `bidule.exe`
  Si on tape `bidule`, lequel sera exécuté ?

## Linux

- Il faut donner le nom entier du programme, avec l'extension éventuelle.
- Exemple : soit un script qui s'appelle `mon_script.sh`
  Exécution :
  `./mon_script` : erreur `./mon_script.sh` : OK

# Recherche de commande : différences Windows/Linux

## Windows

- Un fichier exécutable est identifié comme tel par son extension.
- Les principales : `BAT`, `CMD` ou `EXE`, mais aussi beaucoup d'autres (voir la VE `PATHEXT`)
- On peut l'appeler uniquement par son nom, l'OS va chercher s'il trouve une correspondance, dans **cet** ordre.
  Attention : un dossier contient deux fichiers, `bidule.bat` et `bidule.exe`
  Si on tape `bidule`, lequel sera exécuté ?

## Linux

- Il faut donner le nom entier du programme, avec l'extension éventuelle.
- Exemple : soit un script qui s'appelle `mon_script.sh`
  Exécution :
  `./mon_script` : erreur
  `./mon_script.sh` : OK

# Variables spéciales

- Un certain nombre de variables spéciales sont automatiquement déclarées dans un script.

Avec par exemple la commande `ma_cde aaa bbb`

| | Linux | Windows | Exemple |
|---|---|---|---|
| Nombre d'arguments | `$#` | (néant) | 2 |
| Tous les arguments | `$*` | `%*` | `aaa bbb` |
| Chemin et nom du script | `$0` | `%0` | `ma_cde` |
| Arguments positionnels | `$1` | `%1` | `aaa` |
| | `$2` | `%2` | `bbb` |
| | `$3` | `%3` | |
| | ... | ... | |
| Code de sortie de la dernière commande | `$?` | `%ERRORLEVEL%` | |

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN

# Arithmétique entière

- Les Shell permettent de faire de l'arithmétique entière
- Windows :
  - option /A à la commande SET
  - De multiples opérateurs, arithmétiques et logiques (ET, OU, décalages, ...)
  - Exemple : `SET /A compt=%compt%+1`
  - Référence : ss64.com, ou `SET /?`
- Linux/Bash :
  - La notation `$(($nom_variable))` permet de traiter une variable comme un entier
  - Exemple :

```
var1=4
var2=$(($var1*2))
echo $var2 /* affiche 8 */
```

- Tests :
  - pour tester l'égalité : identique à de la comparaison de chaîne.
  - pour < et >, il existe des syntaxes spécifiques à chaque Shell.

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE iut ROUEN