UNIVERSITÉ PARIS 8
VINCENNES-SAINT-DENIS

Méthodologie de la programmation

Sylvia Chalençon

Licence Informatique 1re année

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Déroulement du cours

- Mon mail : sc@up8.edu
- Le support de cours et les sujets de TD/TP sont disponibles à l'adresse : https://sc.up8.site/methodo/2122.html
- Évaluations

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Chapitre 1 : Introduction

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

L'informatique ?

- Informatique : traitement automatique de l'information par un programme exécuté sur un ordinateur, un système embarqué, un robot, un automate…
- Exemples d'informations : nombres, textes, images, sons, vidéos.
- Les informations circulent sous forme d'une suite de bits (chiffres en base 2).

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Ordinateur ?

- Unité *d'entrée* : permet de faire entrer les informations dans le système.
- Unité de *stockage* : permet de conserver les informations.
- Unité de *traitement (processeur)* : traite les informations en suivant scrupuleusement les instructions d'un programme informatique.
- Unité de *sortie* : permet de faire sortir les résultats des traitements.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Programme informatique ?

- Séquence d'opérations destinées à être exécutées par l'ordinateur pour réaliser une tâche.

- Un programme est soit :
  - un code source écrit par un informaticien dans un langage de programmation. Il peut être compilé vers une forme binaire, ou directement interprété.
  - un code binaire : suite de bits interprétée par le processeur d'un ordinateur exécutant un programme informatique. C'est le langage natif du processeur (langage machine).

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Intérêt du langage de programmation ?

- L'ordinateur ne comprend que le langage machine :
  - Opérations élémentaires sur des données binaires.
  - Spécifique à chaque processeur.
  - Difficile et pénible à utiliser.
- Langage de programmation :
  - Facilite la programmation.
  - Interface homme-machine.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Langage de programmation

- Notation conventionnelle destinée à formuler des algorithmes et produire des programmes informatiques qui les appliquent.

- Composé d'un alphabet, d'un vocabulaire, de règles de grammaire et de significations.

- Permet de décrire les structures des données qui seront manipulées par l'appareil informatique, et d'indiquer comment sont effectuées les manipulations.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Vocabulaire

- **Instruction** : ordre donné à l'ordinateur.
- **Variable** : nom utilisé dans un programme pour faire référence à une donnée manipulée par le programme.
- **Constante** : nom utilisé pour faire référence à une valeur permanente.
- **Type** : chaque donnée a une classification qui influe sur la plage de valeurs possibles, les opérations qui peuvent être effectuées, et la représentation de la donnée sous forme de bits.
- **Procédures, fonctions, méthodes** : fragment de programme transformée en opération générale, paramétrable, susceptible d'être utilisée de façon répétée. Ces fragments sont appelés *procédures*, *fonctions* ou *méthodes*.
- **Déclaration** : phrase du programme qui sert à renseigner au traducteur (compilateur, interpréteur...) les noms et les caractéristiques des éléments du programme tels que des variables, des procédures, des types…

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Paradigmes de programmation 1/2

- **Impératif :**
  - Basé sur l'idée d'une exécution étape par étape.
  - Abstraction réalisée à l'aide de procédures auxquelles sont transmises des données.
  - Une procédure principale, la première à être exécutée, et qui peut faire appel à d'autres procédures.
  - *C*, *Pascal*, *Fortran* ou encore *COBOL* sont en paradigme impératif.
- **Fonctionnel :**
  - Basé sur l'idée d'évaluer une formule, et d'utiliser le résultat pour autre chose.
  - Tous les traitements sont faits en évaluant des expressions et en faisant appel à des fonctions.
  - Le résultat d'un calcul sert de matière première pour le calcul suivant, et ainsi de suite, jusqu'à ce que toutes les fonctions aient produit un résultat.
  - Introduit par le langage *Lisp* à la fin des années 1960.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Paradigmes de programmation 2/2

- **Logique :**
    - Basé sur l'idée de répondre à une question par des recherches sur un ensemble, en utilisant des axiomes, des demandes et des règles de déduction.
    - Cascade de recherches de données dans un ensemble, en faisant usage de règles de déduction. Les données obtenues, et associées à un autre ensemble de règles peuvent alors être utilisées dans le cadre d'une autre recherche.
    - L'exécution du programme se fait par évaluation : le système effectue une recherche de toutes les affirmations qui, par déduction, correspondent à au moins un élément de l'ensemble.
    - Introduit par le langage *Prolog* en 1970.

- **Objet :**
    - Destiné à faciliter le découpage d'un grand programme en plusieurs modules isolés les uns des autres.
    - Un objet représente un concept, une idée ou toute entité du monde physique.
    - Les objets ont une structure interne et un comportement.
    - Introduit les notions d'objet et *d'héritage* : un objet contient implicitement les variables et les fonctions de ses ancêtres, et cet héritage aide à réutiliser du code.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Chapitre 2 : Linux & Shell

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Présentation

- Linux est un système d'exploitation permettant de contrôler un PC et ses différents périphériques.
- **Multi-utilisateurs** = utilisable par plusieurs personnes simultanément
- **Multi-tâches** = peut exécuter plusieurs programmes en même temps
- Repose sur un **noyau** (*kernel*) utilisant 4 concepts principaux : fichiers, droits d'accès, processus et communication interprocessus (IPC).

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Histoire

- 1965 : Multics (laboratoires Bell - AT&T, MIT, General Electric)
- 1969 : Unics (Kenneth Thompson, laboratoires Bell, développé en langage d'assemblage)
- 1971 : réécriture de Unix en langage C (Dennis Ritchie, Brian Kernighan)
- 1973 : Premier article sur Unix au *Symposium on Operating System* à l'université de Purdue
- fin des années 70: reprise par le monde académique (Université de Californie, Berkeley)

Histoire

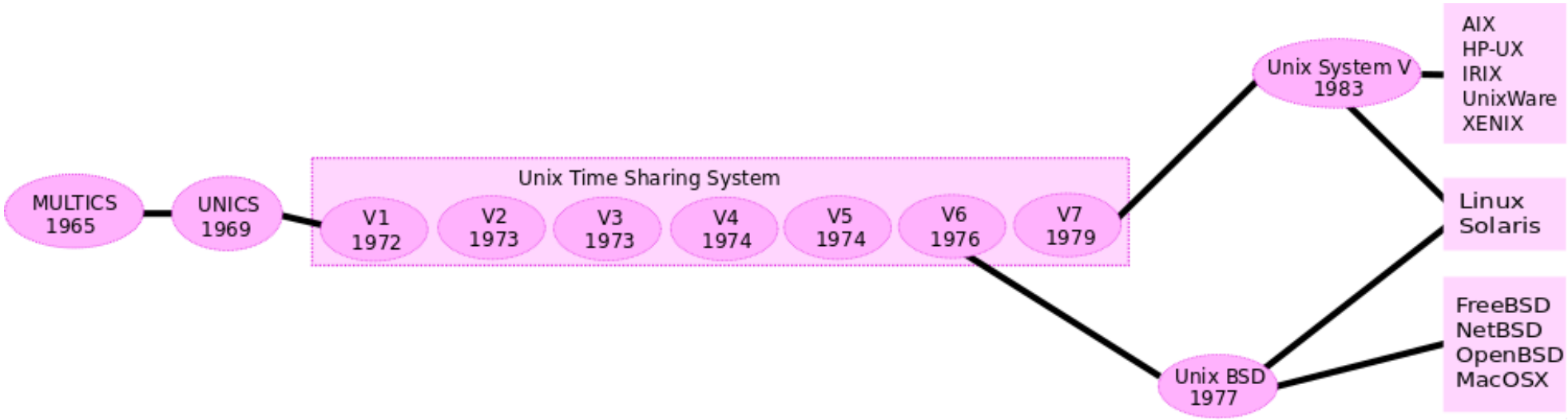
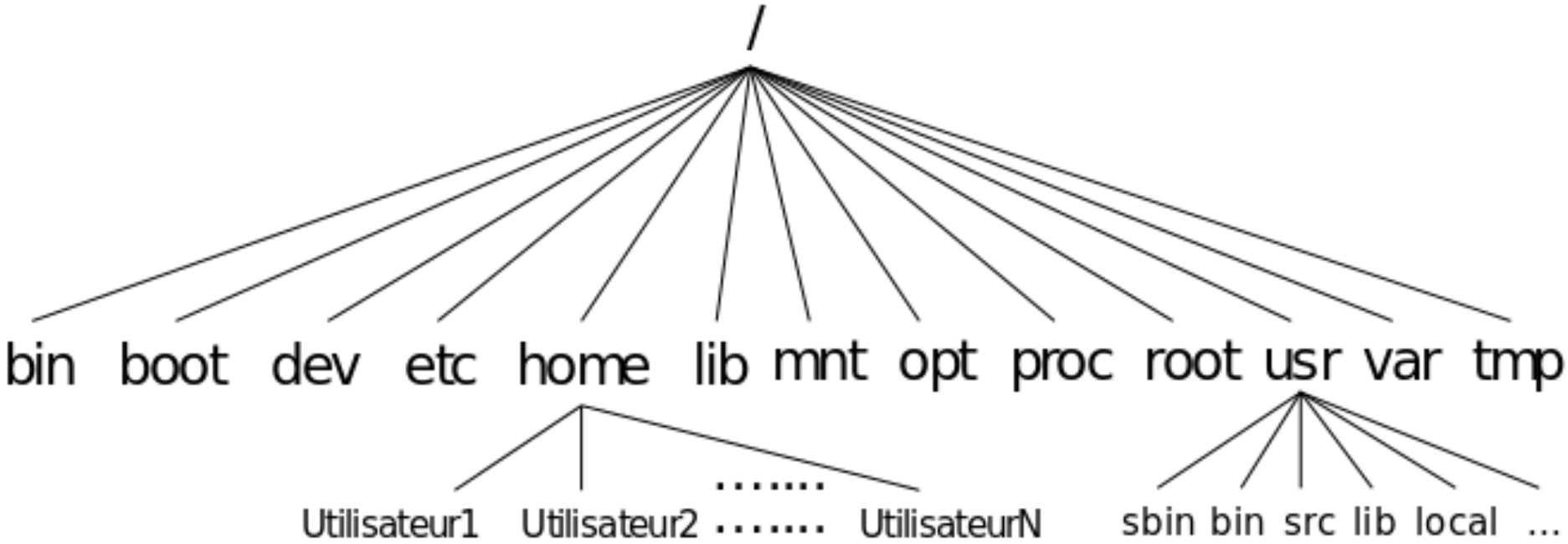


Image issue de wikipedia (https://fr.wikipedia.org/wiki/Unix)

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Le système de fichiers 1/3

Arborescence que l'on parcourt de la racine (*root)*, notée **/**, vers les feuilles.



UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Le système de fichier 2/3

- **/bin :** exécutables essentiels au système, employés par tous les utilisateurs.
- **/boot :** fichiers permettant à Linux de démarrer.
- **/etc :** fichiers de configuration nécessaires à l'administration du système (fichiers *passwd, group, inittab, ld.so.conf, lilo.conf, …*).
- **/dev :** points d'entrée des périphériques.
- **/home :** répertoires personnels des utilisateurs.
- **/mnt :** points de montage des partitions temporaires (cd-rom, usb, …).
- **/opt :** contient des packages d'applications supplémentaires
- **/proc** : fichiers permettant d'accéder aux informations sur le matériel, la configuration du noyau et sur les processus en cours d'exécution.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Le système de fichier 3/3

- /**root :** répertoire personnel de l'administrateur root.
- **/sbin :** exécutables système essentiels.
- **/usr/bin :** majorité des fichiers binaires et commandes utilisateur.
- /**usr/lib :** la plupart des bibliothèques partagées du système.
- **/usr/local :** données relatives aux programmes installés sur la machine locale par le root.
- **/usr/sbin :** fichiers binaires non essentiels au système réservés à l'administrateur système.
- **/usr/src :** sources des diverses applications installées sur le système.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Shell 1/2

- Interpréteur de commandes qui permet d'accéder aux fonctionnalités internes du système d'exploitation
- Ne fait pas partie du noyau mais peut être vu comme une enveloppe externe (d'où *shell)* permettant d'interagir avec.
- Interface en ligne de commande accessible depuis un terminal

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Shell 2/2

Voici les noms de quelques-uns des principaux *shells* qui existent :

- **sh** : *Bourne Shell*. L'ancêtre de tous les *shells*.
- **bash** : *Bourne Again Shell.* Une amélioration du *Bourne Shell*, disponible par défaut sous *Linux*.
- **ksh** : Korn Shell. Un shell puissant assez présent sur les Unix propriétaires, mais aussi disponible en version libre, compatible avec bash.
- **csh** : *C Shell*. Un *shell* utilisant une syntaxe proche du langage C.
- **tcsh** : *Tenex C Shell*. Amélioration du *C Shell*.
- **zsh** : *Z Shell*. *Shell* assez récent reprenant les meilleures idées de *bash*, *ksh* et *tcsh*.

Il y en a quelques autres, mais vous avez là les principaux.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Le terminal et l'invite de commande shell

- Le terminal correspond à une fenêtre présentant un prompt, encore appelé invite de commande. Celle-ci est paramétrable et par défaut en bash se compose comme suit : *login@machine$*
- suffixe $ pour l'utilisateur normal, suffixe # pour le super-utilisateur encore appelé *root* ou administrateur
- On saisit les commandes à la suite du prompt
- Pour stopper la commande en cours : *Ctrl-C*
- Pour mettre en attente la commande en cours : *Ctrl-Z*
- Pour terminer l'entrée standard (les éventuelles paramètres saisis par l'utilisateur via le clavier) : *Ctrl-D*

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Entrée et sortie standards

- ***stdin*** : l'entrée standard, par défaut le clavier, identifiée par l'entier 0.
- ***stdout*** : la sortie standard, par défaut l'écran, identifiée par l'entier 1
- ***stderr*** : la sortie d'erreur standard, par défaut l'écran, identifiée par l'entier 2

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Redirection des flux d'entrée-sortie

- **>** : redirection de la sortie standard
- **<** : redirection de l'entrée standard
- **>>** : redirection de la sortie standard avec concaténation
- **>&** : redirection des sorties standard et d'erreur
- **|** : redirection de la sortie standard sur l'entrée standard

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Quelques commandes

- ***man*** comme manuel ;-)
- *man [n] commande : v*isualisation à l'écran des informations concernant la commande spécifiée.
- Affichage réalisé par la commande *more* donc :
  - la touche '*entrée*' pour afficher la ligne suivante,
  - la touche '*espace*' pour afficher la page suivante,
  - la touche '*q*' pour quitter

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Quelques commandes

- *pwd* : affiche le chemin absolu du répertoire courant.
- *cd [chemin]* : change le répertoire courant pour celui spécifié par le chemin.
  - cd - : change le répertoire courant pour le répertoire précédent
  - cd : change le répertoire courant pour le home directory.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Absolu *vs* relatif

- Alias :
  - . désigne le répertoire courant
  - .. désigne le répertoire parent
- Si *pwd* nous retourne */home/login/methodo* alors *cd ..* nous amène dans le répertoire */home/login/*
- *cd methodo/shell* nous amène dans le répertoire /home/login/methodo/shell
- *cd /home/login/prog_fonctionnelle* nous amène dans le répertoire */home/login/prog_fonctionnelle*

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Visualiser le contenu d'un répertoire

La commande *ls [-options]* liste le contenu d'un répertoire. Voici quelques options :

- *-F* : positionne à la fin des noms un / pour les répertoires et un * pour les fichiers exécutables
- *-a* : affiche tous les fichiers, y compris les fichiers cachés (ceux qui commencent par .)
- *-l* : description complète du contenu d'un répertoire (une ligne par fichier). Le premier caractère de la ligne indique le type du fichier :
    - - : standard
    - *d* : répertoire
- *-d* : évite de lister le contenu d'un répertoire : si *rep* est un repertoire, *ls -l rep* listera le contenu du répertoire *rep*, alors que *ls -ld rep* listera la description du répertoire

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Permission sur les fichiers

- Pour chaque fichier, il y a trois classes d'utilisateurs
  - utilisateur (*u*) : le propriétaire du fichier
  - groupe (*g*) : le groupe d'utilisateurs auquel appartient le fichier
  - autre (*a*) : tous les autres
- Les permissions accordées à ces trois classes sont :
  - *r* : lecture
  - *w* : écriture
  - *x* : exécution (pour un fichier, peut être exécuté, pour un répertoire, peut être parcouru)
- *chmod [options] droits fichier1, fichier2, …* : change les droits d'un ou plusieurs fichiers
  - *chmod a+r toto.txt :* ajoute l'accès en lecture du fichier toto.txt à tout le monde
  - *chmod 444 toto.txt* : donne les droits en lecture (et uniquement en lecture) à tous les utilisateurs

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Manipuler les fichiers

- *cat [option] [fichier1, fichier 2, etc] :* affiche le contenu d'un ou plusieurs fichiers.
- *more [fichier]* pour visualiser le contenu d'un fichier page par page.
- find [options] : effectue une recherche à partir des informations données en option
  - *find rep -name fichier* : cherche, dans *rep* et ses sous-répertoires, un fichier nommé *fichier*
  - *find rep -type d* : cherche tous les sous-répertoires de *rep*

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Manipuler les fichiers

- *cp [option] [fichier_source] [fichier_destination]* : copie le fichier source dans le fichier destination.
    - Si le fichier destination n'existe pas, il est créé.
    - Sinon son contenu est écrasé sans avertissement.
    - Si la destination est un répertoire, alors la source peut être une liste de fichiers.
- *mv [option] [fichier_source] [fichier_destination] :* déplace un fichier source en le renommant si le chemin du fichier destination contient un nom de fichier.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Manipuler les fichiers

- *rm [option] [fichier]* : supprime un fichier.
  - *-i* : demande confirmation à l'utilisateur
  - *-f* : supprime sans demander confirmation à l'utilisateur
  - *-r* : efface récursivement le contenu du répertoire
- *wc [option] [fichier] :* obtenir des statistiques sur un fichier. Exemples :
  - *wc -l toto* affiche le nombre de lignes du fichier toto
  - *ls* | *wc -l* affiche le nombre de fichiers dans le répertoire courant
  - wc -c toto affiche le nombre de caractères du fichier toto

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Répertoires

- *mkdir [répertoire]* : crée un répertoire
- *rmdir [répertoire]* : supprime un répertoire vide
- *du [option] chemin* : donne la taille en octets d'un fichier ou d'un répertoire

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Les variables d'environnement

- Variables permettant de paramétrer le fonctionnement du système (langue utilisée, chemins vers les fichiers exécutables, chemin vers les bibliothèques, etc)

- HOME : Chemin d'accès de votre répertoire d'accueil.

- PATH : liste des répertoires a parcourir pour trouver un programme. Par exemple, *PATH=.:/usr/bin:/usr/sbin:/bin:/sbin* Les différents répertoires sont séparés par :.

- SHELL : chemin d'accès du programme shell (souvent */bin/bash*).

- USER : identifiant de l'utilisateur connecté.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Les variables d'environnement

- Lire une variable d'environnement : *echo $HOME*
- Redéfinir une variable d'environnement :
  export PATH=$PATH:/home/login/monbin
  Mais attention, la modification de la valeur de *PATH* ne vaut que dans ce terminal
- Pour définir une variable d'environnement de manière à ce qu'elle affecte l'ensemble de la session il suffit de placer une commande la définissant dans un des fichiers cachés dans le répertoire personnel par exemple *$HOME/.bash_profile* (bash).

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Les variables

- Pour affecter une valeur à une variable : *nom_variable=valeur*
- Pour accéder au contenu de la variable : *$nom_variable*
- Il existe des variables un peu spéciales :
  - *$** : contient tous les arguments passés à la fonction
  - *$#* : contient le nombre d'arguments
  - *$@* : contient la liste des arguments du script
  - *$?* : contient le code de retour de la dernière opération
  - *$0* : contient le nom du script
  - *$n* : contient l'argument *n*, *n* étant un nombre
  - *$!* : contient le PID de la dernière commande lancée

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Test

- Opération dont le but est d'évaluer la valeur d'une expression
- 2 façons équivalentes de réaliser un test :
  - *test expression*
  - *[expression]*
- Renvoie un *code de retour :* un nombre qui correspond à une réponse de type « *vrai* » ou « *faux* ». 0 pour vrai, n'importe quel autre pour faux.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Instruction *if*

- Syntaxe 1 :

```
if [ condition ]
then
 action1
fi
```

- Syntaxe 2 :

```
if [ condition ]
then
 action1
else
 action2
fi
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Opérateurs arithmétiques

- *-eq* (*equal*) : « égal à » (=)
- *-ne* (*not equal*) : « différent de » (≠)
- *-gt* (*greater than*) : « strictement supérieur à » (>)
- *-lt* (*lesser than*) : « strictement inférieur à » (<)
- *-ge* (*greater or equal*) : « supérieur ou égal à » (≥)
- *-le* (*lesser or equal*) : « inférieur ou égal à » (≤) ;

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Exemple

```
#!/bin/bash
if test 2 -lt 3
then echo "C'est normal."
fi

if test 2 -gt 3
then echo "C'est absurde."
fi

petit=2
grand=3

if test $petit -ne 3
then echo "C'est normal."
fi

if test 2 -eq $grand
then echo "C'est absurde."
fi
```

L’exécution de ce script donne l’affichage suivant :

```
C'est normal.
C'est normal.
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Opérateurs sur les fichiers

- nature du fichier
    - *-e* (*exists*) : vérifie l'existence d'un fichier
    - *-f* (*file*) : vérifie l'existence d'un fichier, et le fait qu'il s'agisse bien d'un fichier au sens strict
    - *-d* (*directory*) : vérifie l'existence d'un répertoire
    - *-L* (*link*) : vérifie si le fichier est un lien symbolique ;
- attributs du fichier
    - *-s* (*size*) : vérifie qu'un fichier n'est pas vide ;
- droits sur le fichier
    - *- r* (*readable*) : vérifie si un fichier peut être lu
    - *-w* (*writable*) : vérifie si un fichier peut être écrit ou modifié
    - *-x* : vérifie si un fichier peut être exécuté
- comparaison de fichiers
    - *-nt* (*newer than*) : vérifie si un fichier est plus récent qu'un autre
    - *-ot* (*older than*) : vérifie si un fichier est plus ancien qu'un autre

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Exemple

```
#!/bin/bash
if test -e ~/premier
  then echo "~/premier existe."
  else echo "~/premier n'existe pas."
fi

if test -d ~/td1
  then echo "~/td1 est un répertoire."
  else echo "~/td1 n'est pas un répertoire."
fi

if test -f ~/premier
  then echo "~/premier est un fichier."
  else echo "~/premier n'est pas un fichier."
fi

if test ~/premier -nt ~/copie
  then "~/premier est plus récent que ~/copie."
fi
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Expression arithmétique

- Le shell peut évaluer des expressions arithmétiques délimitées par *$(( ))*

```
#!/bin/bash
n=2
echo $(( $n * 2 + 3 ))
p=$(( $n + 1 ))
echo $p
```

- Affiche

```
7
3
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Boucle *for*

- Syntaxe 1 :

```
for variable in liste_valeurs
    do instruction(s)
done
```

- Exemple :

```
#!/bin/bash
for i in "$@"
do
 echo "$i"
done
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Boucle *for*

- Syntaxe 2 :

```
for ((e1;e2;e3))
    do instruction(s)
done
```

- **`e1`**, **`e2`** et **`e3`** sont des expressions arithmétiques.

- Commence par exécuter l'expression **`e1`**, puis tant que l'expression **`e2`** est différente de zéro le bloc d'instructions est exécuté et l'expression **`e3`** évaluée.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Exemple

```
#!/bin/bash
for ((i=0 ; 10 - $i ; i=$(( i + 1 )) ))
    do echo $i
done
```

Affiche :

```
0
1
2
3
4
5
6
7
8
9
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Boucle *while*

- Syntaxe :

```
while condition
    do instruction(s)
done
```

- Exécute un bloc d'instructions tant qu'une certaine condition est satisfaite.
- Lorsque cette condition devient fausse la boucle se termine.
- Cette boucle permet donc de faire un nombre indéterminé de tours de boucle, voire infini si la condition ne devient jamais fausse.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Exemple

```
#!/bin/bash
while [ $j -lt 10 ]
do
   echo $j
   j=$(( j + 1 ))
done
```

Affiche la même chose que le script précédent.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Fonctions

- Intérêt :
    - éviter les répétitions de code;
    - diminuer les risques de bogues ;
    - augmenter la lisibilité du script.
- Définition d'une fonction — Syntaxe :

```
nom_fonction () {
 instruction1
 instruction2
 ...
}
```

- Appel de la fonction — Syntaxe :

```
nom_fonction argument1 argument2 …
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Fonctions - Exemple

```
#!/bin/bash
max (){
  if [ $1 -gt $2 ] then
    return $1
  else
    return $2
  fi
}
max $1 $2
echo $?
```

```
login@machine$./max.sh 23 56
56
login@machine$
```