M2 CCI

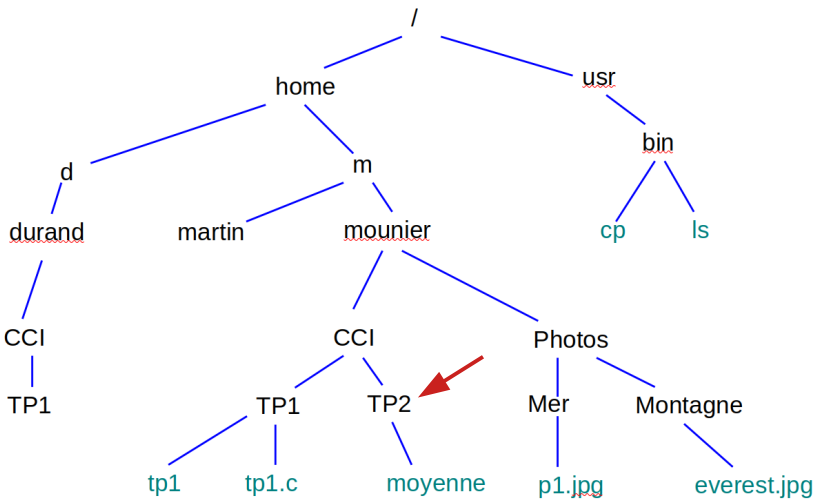
UE Programmation et Langages

Cours « Programmation »

Laurent Mounier

2023 - 2024

Retour sur les commandes Unix



pwd	print working directory
cd	change directory
ls	list
mkdir	make directory
rmdir	remove directory (s'il est vide)
cp	copy (de fichiers, de répertoires)
rm	remove (un fichier)
mv	move (déplacer/renommer un fichier ou répertoire)

Exercice : en supposant que *TP2* est le répertoire courant

1. supprimer le fichier *tp1.c*
2. copier le fichier *p1.jpg* dans *Montagne*
3. effacer l'ensemble du répertoire *durand*
4. copier le fichier *tp1* dans *TP2*

Un mot sur les droits d'accès (Linux)

Attributs associés à un fichier ou répertoire :
Taille, date de création, propriétaire, droits d'accès

Droits associés à un fichier ou répertoire : *read* (r), *write* (w), *execute* (x)

Catégorie d'utilisateur par rapport à un fichier ou répertoire donné :
user (u), *group* (g), *others* (o), *all* (a)

Visualiser les attributs d'un fichier : *ls -l*

---|---|---

user group others

```
im2ag-mandelbrot:~/CCI_PL1/Rentree1/PL1/Rentree1
[08:24:47]mounlaur$ ls -l
total 0
-rwxr-xr-- 1 mounlaur 21003 23 Aug 18  2015 hello.c
-rwxr-xr-- 1 mounlaur 21003 28 Aug 18  2015 ping
-rwxr-xr-- 1 mounlaur 21003 28 Aug 18  2015 pong
```

Modifier les droits d'accès : *chmod*

chmod u+x fichier1	ajout des droits d'exécution au propriétaire sur le fichier
chmod g-wx fichier2	enlève les droits ecriture/exécution au groupe sur le fichier
chmod a+r fichier3	ajout des droits de lecture à tous sur le fichier
chmod -R a+r repertoire1	ajout des droits de lecture à tous sur l'ensemble du répertoire

Exemple concret

- Le binôme *dupont* et *durand* travaillent ensemble sur le TP1

- Ils créent des fichiers et répertoires sur le compte de *dupont*, dans le répertoire *TP1*

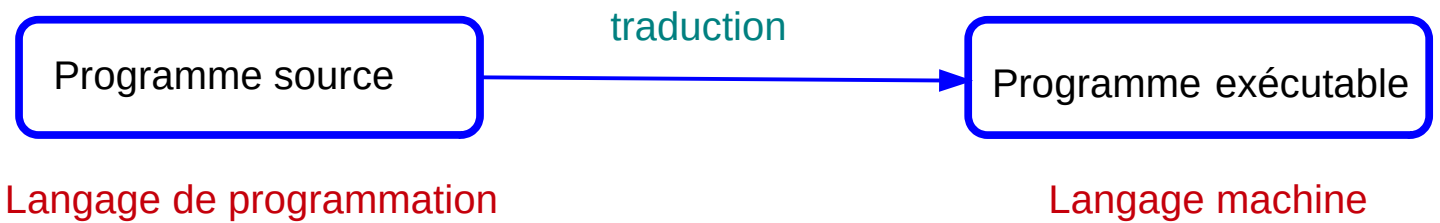
- En fin de séance, comment faire une copie, dans le répertoire de *durand,*
de tout ce qui a été fait ?

Programme exécutable

- Il existe une grande variété de **plateformes d'exécution** … et de **processeurs**
- Chaque processeur ne peut exécuter que des programmes :
  - codés en **binaire** (~ séquence de « 0 » et de « 1 »)
  - comportant uniquement des instructions du ***langage machine*** de ce processeur
  - ces instructions dépendent de l'architecture du processeur

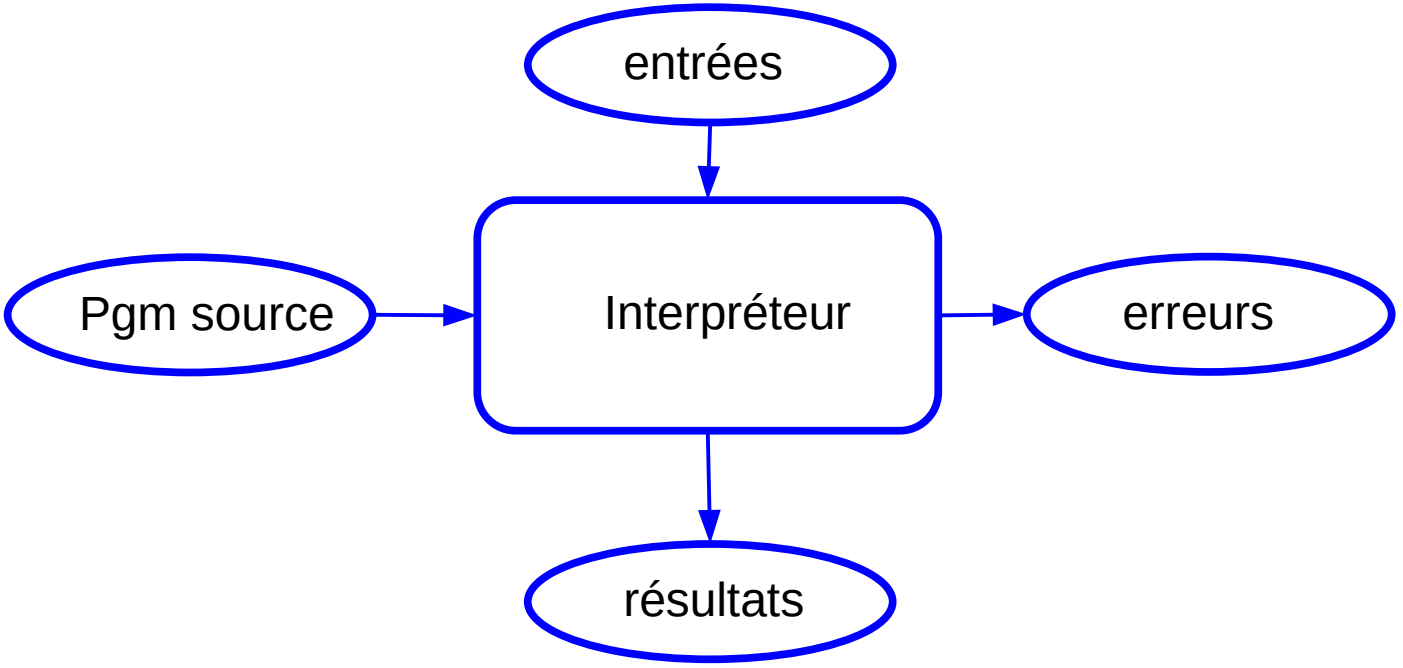
→ Nécessaire de :

- Disposer de langages de programmation de plus **haut-niveau**
- Disposer de logiciels pour **traduire** ces programmes en langage machine



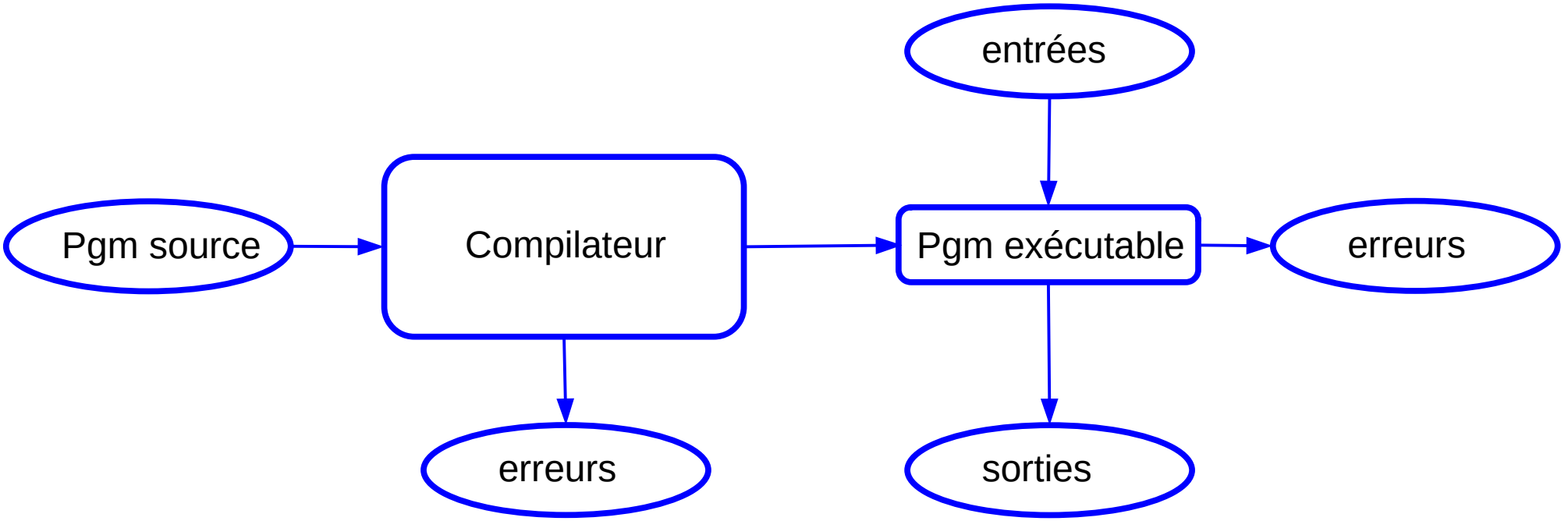
Démo : visualisation du contenu de/bin/cp

Exécuter un progrmme : Interprétation



Le programme est **simultanément** *traduit* **et** *exécuté*
L'utilisateur doit disposer d'un interpréteur sur sa machine
Avantage : portabilité (possibilité d'exécuter du code « distant »)
Inconvénient : temps d'exécution

Exécuter un programme : compilation



Le programme est *traduit* **puis** *exécuté* (plus tard, sur une autre machine, plusieurs fois ...)
L'utilisateur doit disposer d'un processeur **compatible** avec le pgm exécutable généré
Inconvénient : non portabilité
Avantage : temps d'exécution

Langages de programmation (1)

(très) grande variété de langage, mais des caractéristiques communes :

- **syntaxe :**

  règles qui définissent ce qu'est un programme « bien formé »

- **sémantique :**

  règles qui définissent le comportement de ce programme à l'exécution

But recherché :

- permettre une traduction automatique (pas d'ambiguïté)
- détecter les programmes (manifestement) incorrects
- offrir un niveau d'expressivité suffisant au programmeur

→ un ensemble de compromis …

(très!) bref historique des langages de programmation

- années 50-60 : Fortran (54), Cobol (59), Basic (64) Algol (68)
- années 70 : PL1 (70), Pascal (71), C (72)
- années 80 : C++ (83), HTML (89)
- années 90 : Python (91), Java (95), PHP (95), JavaScript (95)
- années 00 : C# (00)
- années 10 : go (09), rust (10)

Courte lecture possible sur l'histoire des langages de programmation :

http://projet.eu.org/pedago/sin/ICN/1ere/4-langages.pdf

Choix d’un langage de programmation ?

Une décision **multi-critère** :

- Souvent imposé par le contexte de développement …
  - insertion dans un programme existant
  - compétences et « culture » d’une entrepris, du développeur ...
- Des langages (plus) spécifiques à certaines classes d’applications :
  - Systèmes embarqués, programmation « bas niveau » : C, Rust, voire assembleur …
  - Programmation « Web » : JavaScript, PhP, .NET, …
  - Calcul scientifique : C, Fortran
  - Intelligence Artificielle : Python
- Des langages (plus) « généraux » : Java, C++, C#
- Existence de fonctions/librairies prédéfinies : Java, Python

https://spectrum.ieee.org/top-programming-languages-2022

Un mot sur le langage C

- développé en 1972, Thomson et Ritchie, puis Kernighan
  → gros succes car comble un vide (Fortran, Algol, Cobol)
  normalisation de différentes version (83, 89, 99 et 2011)

- objectif = écrire un OS (Unix !)
  - plutôt langage de "bas-niveau", proche des instructions processeur
  - occupation memoire "previsible"
  - concepts simple, compilation efficace

- des défauts :
  - concepts anciens (pas d'objets), pas fait pour la programmation "in the large" !
  - peu de vérifications a la compilation → risque élevé d'erreurs a l'exécution, problèmes de sécurité !

- Très largement utilisé :
  Linux et Windows, applications connus (Firefox, Acrobat Reader, Word, etc.)
  Systèmes embarqués ...

- Des successeurs : C++, un peu Java