

# Les langages de script, origines et tendances

Toulouse, 25/11/2010

[Pascal.dayre@enseeiht.fr](mailto:Pascal.dayre@enseeiht.fr)

# Acception commune & co

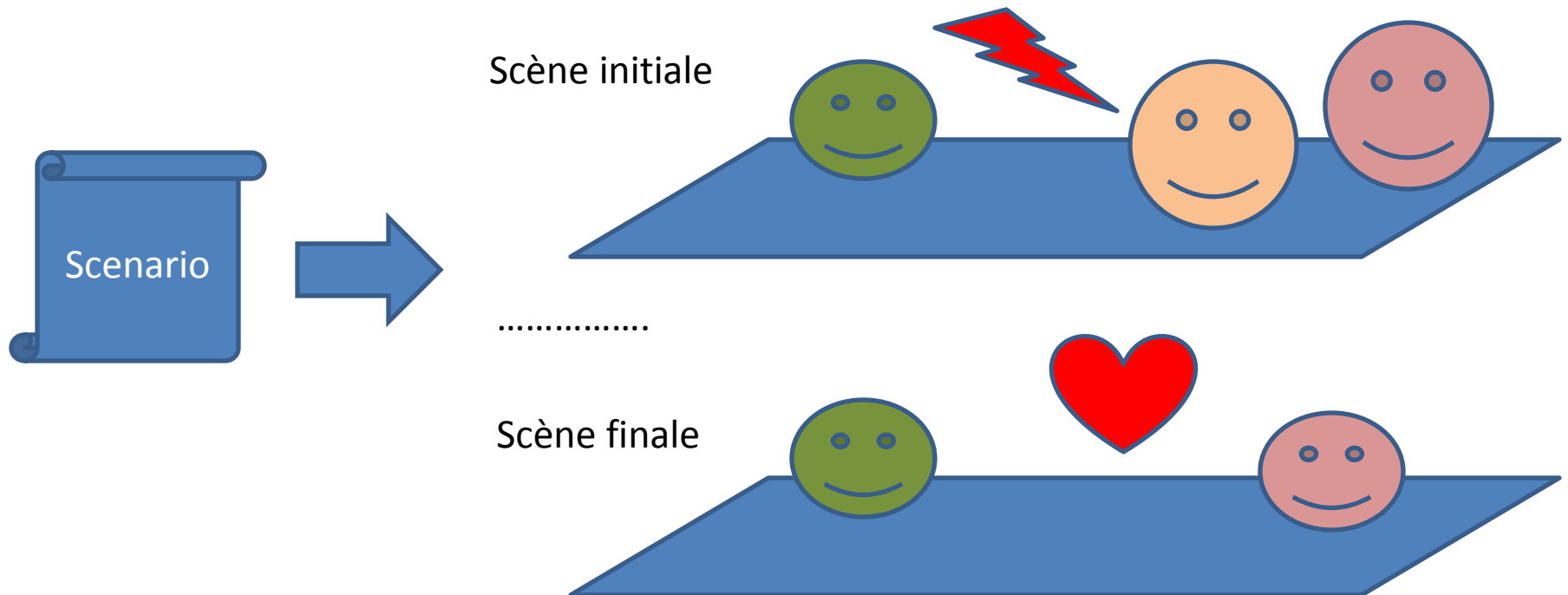
« *Les langages de scripts, ça compile pô et les langages de programmation, ça compile* »  
comme dirait Titeuf

Mais par quoi est défini un langage?

- Par sa syntaxe
- Par sa sémantique
- Et non par son implémentation!

# Définitions

- **script:** « Transposition d'un scénario en un ensemble de scènes »



- **Langage de script:** un abus de langage?

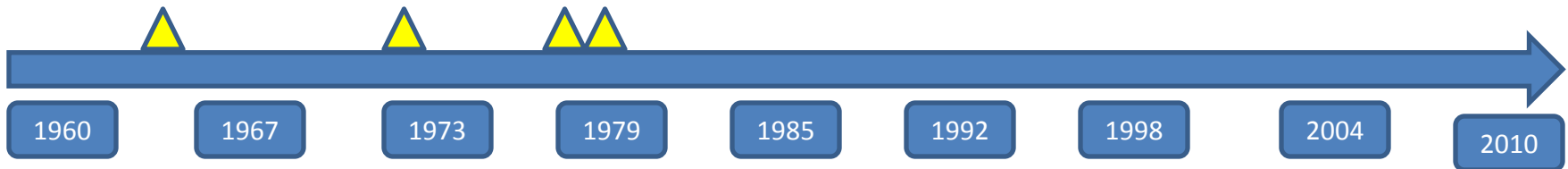
# Quelles tentatives de définition

- « Langage qui permet d'écrire rapidement de petits programmes interprétés. »
- « Langage dont l'utilisation principale est l'automatisation de certaines petites tâches et dont les fonctionnalités sont peu nombreuses. »
- « Les langages de script se concentrent sur la mise en forme de texte, ou l'appel et l'utilisation de composants compilés écrits dans un langage de programmation »
- « l'automatisation de tâches répétitives : traitement de logs, création de comptes utilisateurs, récupération périodique de données sur un site web, administration réseau, installation de logiciels »

# Historique et usages

Langages  
De commande

Langages  
Dynamiques,  
autres

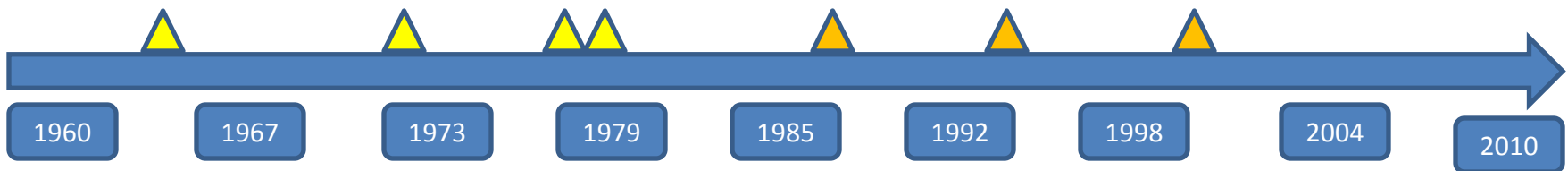


- 1964 (OS/360): JCL (Job Control Language). JOB={STEP} et un STEP=cmd|STEP
- 1971 (UNIX v1): Thompson shell (Interpréteur de ligne de cmd mais pas un LS)
- 1973(sed): cmd de transformation de texte sur les expressions régulières
- 1977(awk): traitement des textes (données, cmd) fichiers ou flux
- 1979: bourne shell (UNIX v7) et cshell (BSD) -> script (batch) + variables, structures de contrôle –conditionnelles, boucles-

# Historique et usages

Langages  
De commande

Langages  
Dynamiques,  
autres



- **1987**: PERL=f(C, sh, awk, sed); créé au départ pour le traitement des logs système, de fichiers texte de taille qq

Puis IHM, admin système, programmation réseau, accès BD, script C  
GI (serveur web), ... le couteau suisse du programmeur

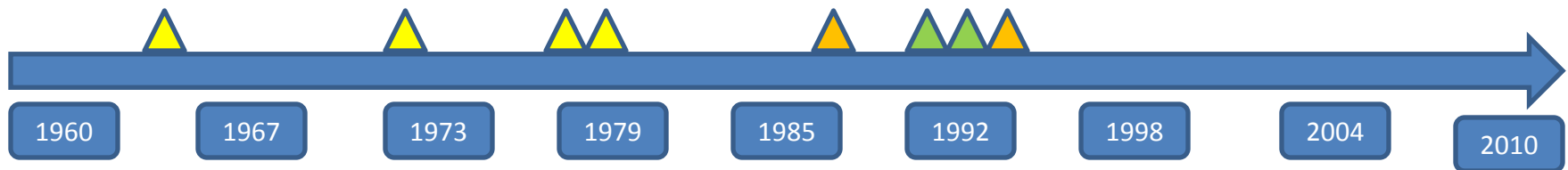
- **1994**: PERL 5
- **2000**: PERL 5.6

<http://www.projet-plume.org/fr/fiche/perl>

# Historique et usages

Langages  
De commande

Langages  
Dynamiques,  
autres



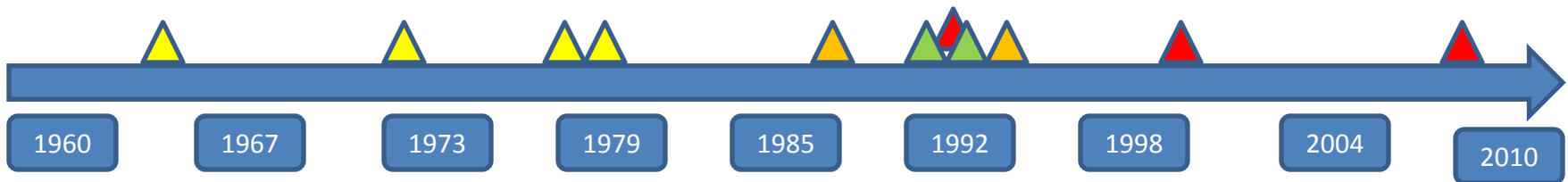
**1988:** TCL (Tool Command Language) conçu comme un langage de commande commun aux outils de conception de circuits imprimés

**1991:** TK composants graphiques pour tester les possibilités de TCL d'intégrateur de composants

# Historique et usages

Langages  
De commande

Langages  
Dynamiques,  
autres



1989: PYTHON v1

2000: PYTHON v2

2008: PYTHON v3

Initialement conçu comme un Langage de script extensible pour programmer un nouveau système d'exploitation en remplacement des primitives systèmes C et du Bourne shell d'UNIX

Utilisé dans les serveurs WEB via WSGI –Web Server Gateway Interface), et comme langage de script pour les applications (cf. Blender, GIMP, ArcGIS, GoogleDocs, ...)

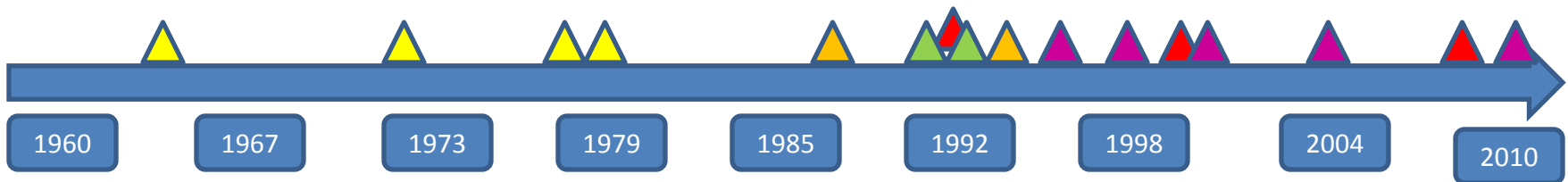
Implémentation CPYTHON, JPYTHON => PYTHON peut scripter des applications Java, des composants ActiveX



# Historique et usages

Langages  
De commande

Langages  
Dynamiques,  
autres



1995: Ruby v1

1998: Ruby v1.2

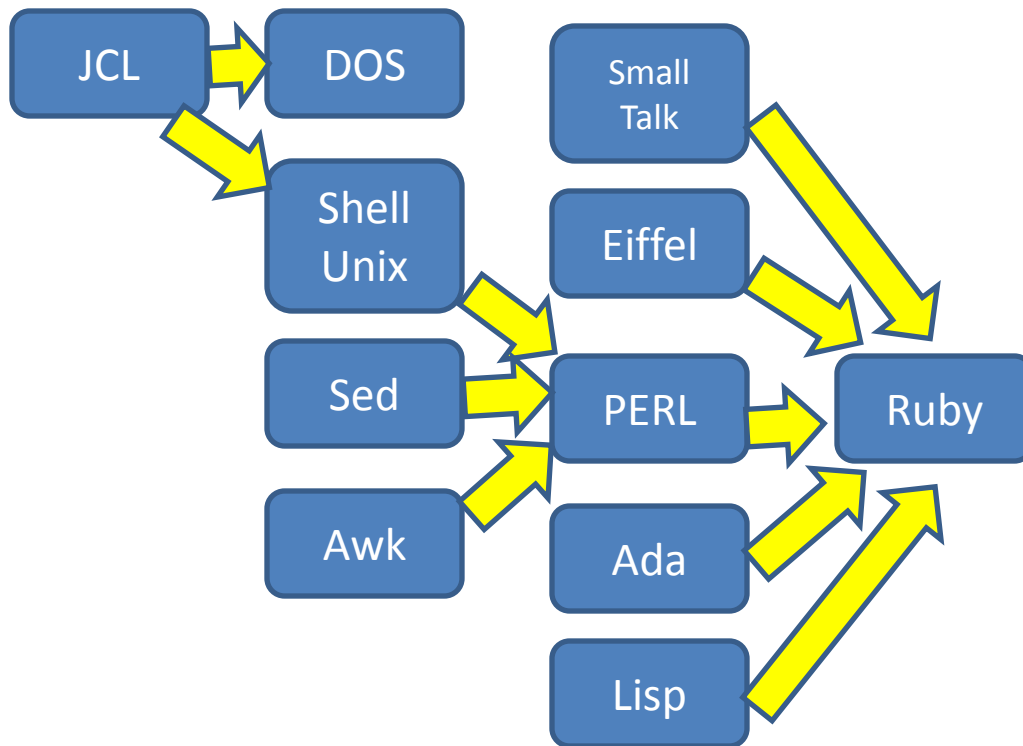
2000: Ruby v1.6

2004: Ruby On Rails (Rails ou RoR): cadre applicatif web (modèle MVC). Influence framework PHP

2010: Ruby v1.9.2

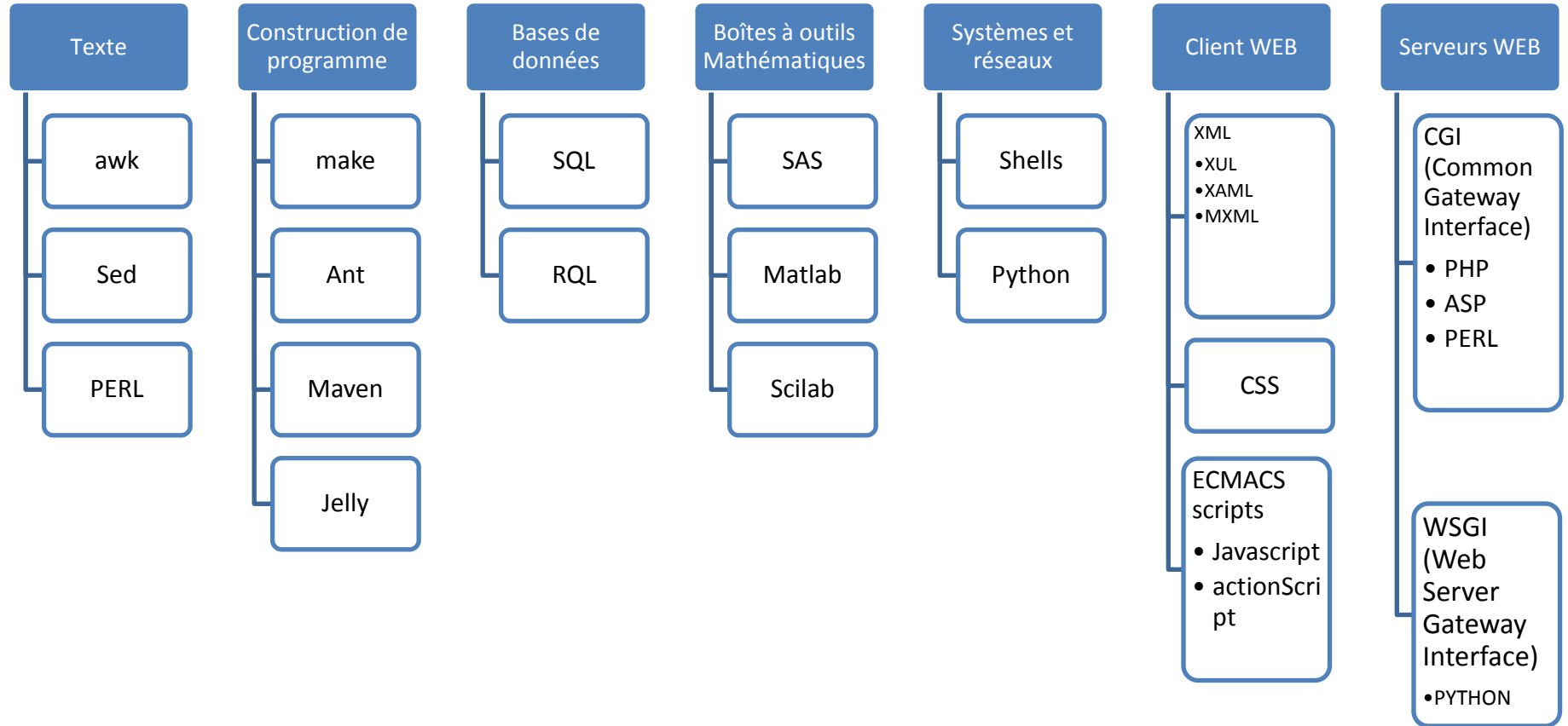
# Héritages et emprunts entre LS

- Exemple, la genèse de Ruby

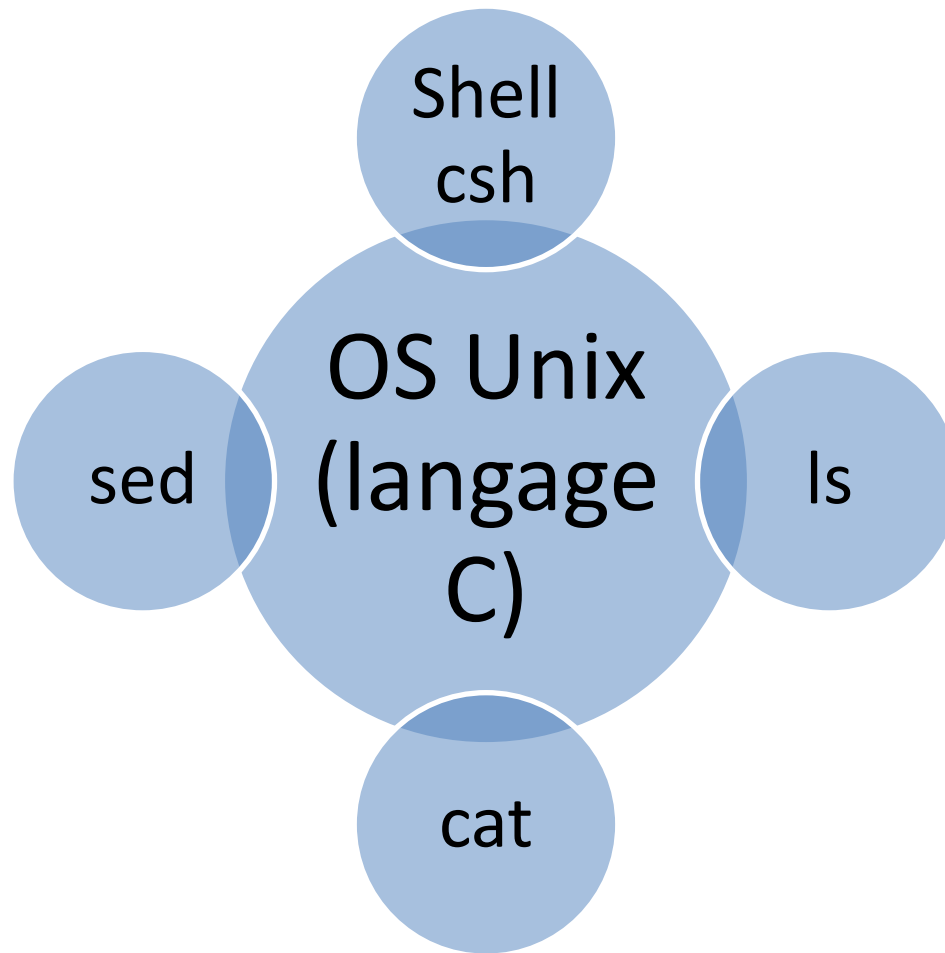


## Les langages de script, origines et tendances

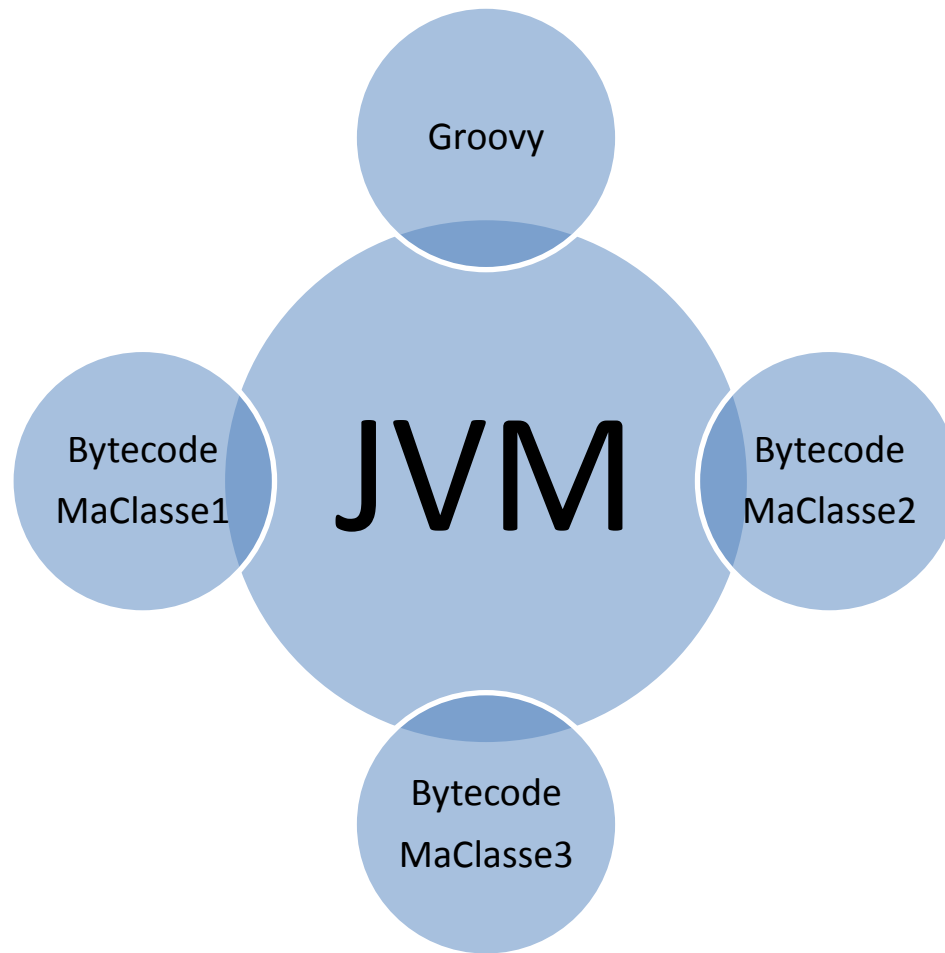
# Mais et les autres?



# Csh pour scripter unix



# Groovy pour scripter Java

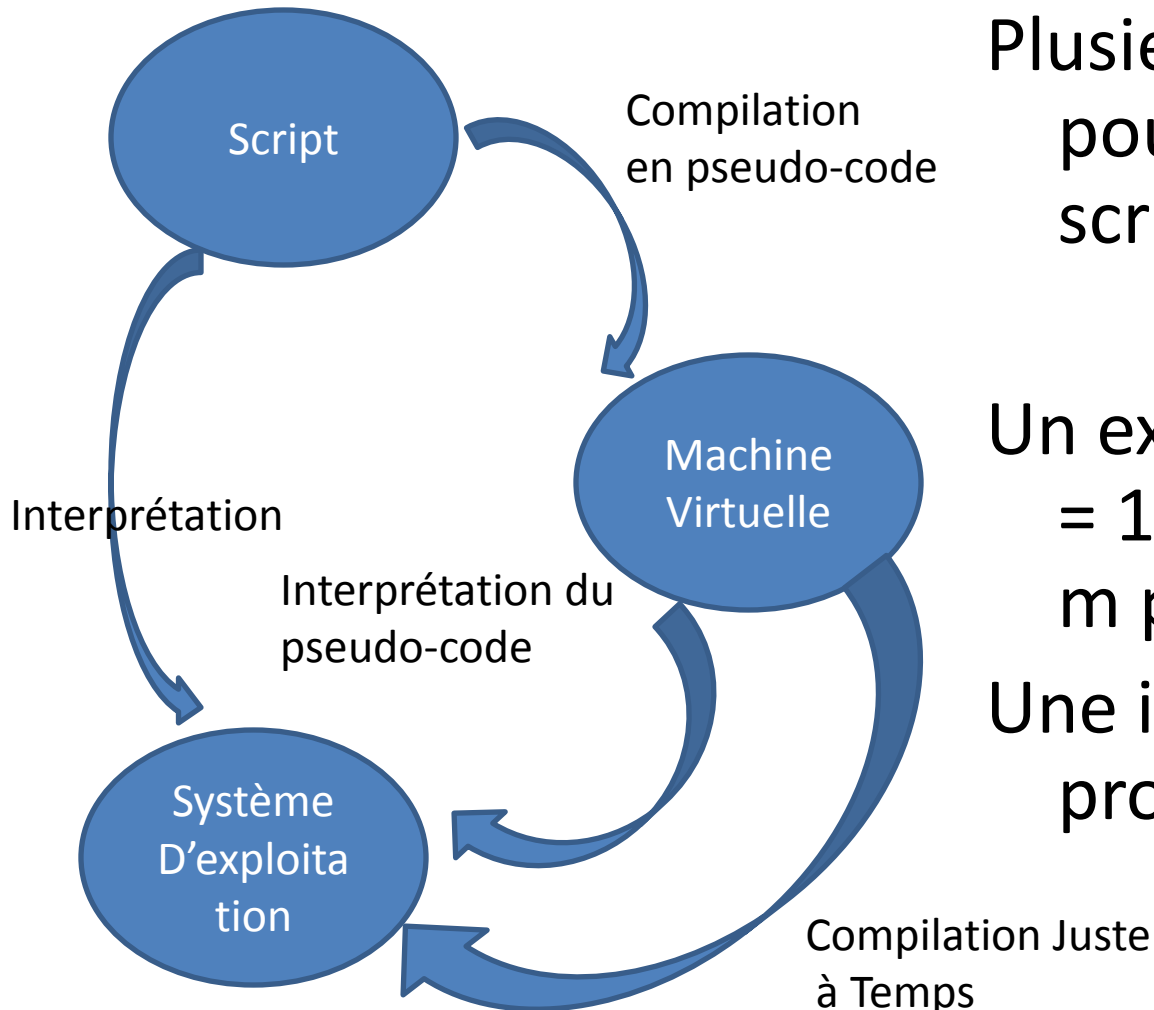


# Optimisation et implémentations

Plusieurs possibilités  
pour optimiser un  
script.

Un exécutable compilé  
= 1 processus lourd,  
m processus légers

Une interprétation = n  
processus lourds...



# Interpréter versus Compiler

- Compiler (traduction d'un langage en un autre):  
analyse syntaxique → analyse sémantique →  
code objet puis exécution → résultat
- Interpréter : toutes les étapes se font à  
l'exécution (contrôle | adaptation du typage,  
adaptation de code au contexte, aux données)

# Définition et caractéristiques

**« Initialement conçus pour le lancement d'une séquence de commandes regroupées dans un fichier, les langages de script sont des langages de programmation qui servent de « glue logicielle » pour scénariser et étendre un environnement par la réutilisation de ses composants compilés de manière souple et flexible (système d'exploitation, machine virtuelle et même les applications). »**

Les plus:

- Concision du code, simplification => puissance, souplesse
- Augmentation de la productivité logicielle (Syntaxe riche sémantiquement, langages de haut niveau => moins de lignes de code)
- Fichier de commande ou interactif (mode console)
- Langages dynamiques (adaptation au runtime)
- Typage faible | explicite | implicite | dynamique (abandon du typage fort à l'Ada)

Les moins:

- PB de performances
  - langage de haut niveau => + d'instructions machine
  - Interprétation => processus lourd contre processus légers
- Non fortement typé => risque sémantique



# Comment créer son propre langage?

- Théorie des langages
- Lex-Yacc
- Définissez votre grammaire
- Générez l'interpréteur lexicale et syntaxique avec lex
- Générez l'interpréteur sémantique avec yacc
- Ecrivez vos premiers scripts
- Interprétez-les avec votre interpréteur sémantique!

# Questions?

Pascal.dayre@enseeiht.fr