isep
École d'ingénieurs du numérique

isep
Formation continue

ICP
ISP - FACULTÉ D'ÉDUCATION
L'esprit grand ouvert sur le monde

DU FENSIL

Langages et programmation

Gilles Carpentier

Correspondance avec le BO et l'UCO

1)Notion de programme en tant que donnée (page 7)

2)Calculabilité, décidabilité (page 7) (2.2.3 page 20)

3)Récursivité (page 7)

4)Modularité (page 7)

5)Paradigmes de programmation (page 8) (4.2.1 page 17)

6)Mise au point des programmes, gestion des bugs (page 8)

1) Notion de programme en tant que donnée

- 2 cas un peu différents :
  - sans essayer de comprendre le code du programme :
    - téléchargement d'un logiciel
  - en essayant de comprendre le code du programme :
    - compilateur, interpréteur
- activité pédagogique
  - utilisation des fonctions **eval** ou **exec** en python

2) Calculabilité, décidabilité

- une fonction est calculable, un problème est décidable, s'il existe un algorithme qui permet de trouver un résultat, une solution, en un nombre fini d'étapes.
- il s'agit bien d'algorithmique, peu importe dans quel langage est implanté l'algorithme.
- activités pédagogiques :
  - une fonction qui teste si un entier appartient à un tableau est calculable
  - le castor affairé est une fonction non calculable si son paramètre est suffisamment grand

3) Récursivité

- il faut distinguer la récursivité dans la définition des structures de données et la récursivité algorithmique
- une structure de données peut posséder un champ dont le type **est** la structure de données :
  - une personne peut contenir un champ père qui est de type personne
- la récursivité opérationnelle correspond aux fonctions qui s'appellent elles-mêmes :
  - exemple de la fonction factorielle

3) Récursivité

- avantage de la récursivité :
  - certains problèmes ne peuvent facilement être résolus que par des algorithmes récursifs (moins de lignes de code à écrire)
- inconvénients :
  - de la récursivité pour les structures de données :
    - impossibilité de sauvegarder la donnée sur disque ou de la transmettre sur le réseau
  - de la récursivité opérationnelle :
    - totalement inefficace en utilisation mémoire et processeur

3) Récursivité

- activités pédagogiques :
  - faire écrire une fonction récursive
  - l'exécuter avec un débogueur en mode pas à pas
  - mesurer le temps d'exécution avec **time.process_time()**
  - réécrire la même fonction en itératif, mesurer le temps d'exécution et comparer avec la version itérative
  - mieux, utiliser **Jeliot** pour visualiser comment l'interpréteur exécute le programme, mais ce n'est que pour Java

4) Modularité

- Dans le BO, la modularité est liée à la conception et utilisation d'API. C'est un peu restrictif. Je reviendrai sur les fondements de la modularité dans la partie paradigmes de programmation.
- activités :
  - fournir aux élèves un script qui contient des fonctions et une documentation et leur faire construire un programme utilisant cette API
  - ou utiliser une API existante par exemple **turtle**
  - leur faire construire ou modifier une API avec une documentation avec l'outil **pydoc**

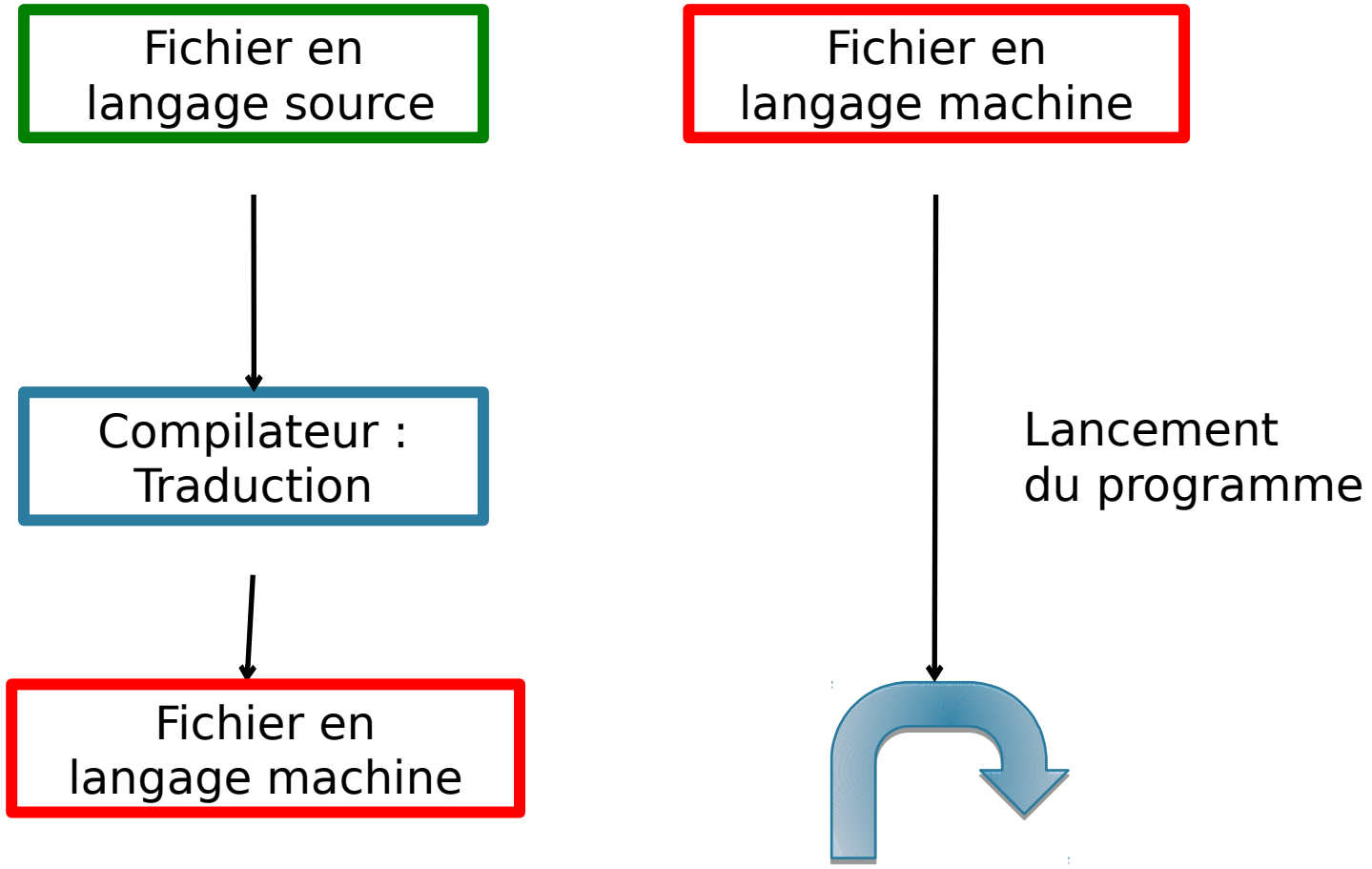
5) Un peu de vocabulaire

- programme // processus
- tâche
- application
- application autonome (stand-alone)
- client-serveur, service
- distribué

5) Programme et processus

- Un programme est un fichier qui contient du code exécutable :
  - soit directement par le système d'exploitation (programme compilé)
  - soit c’est l’interpréteur (par exemple python) qui exécutera un script
- Le programme, une fois lancé devient un (ou plusieurs) processus => on peut donc lancer plusieurs fois le même programme, chaque processus issu de ce programme, est indépendant des autres.
- Un processus est une instance d'un programme.

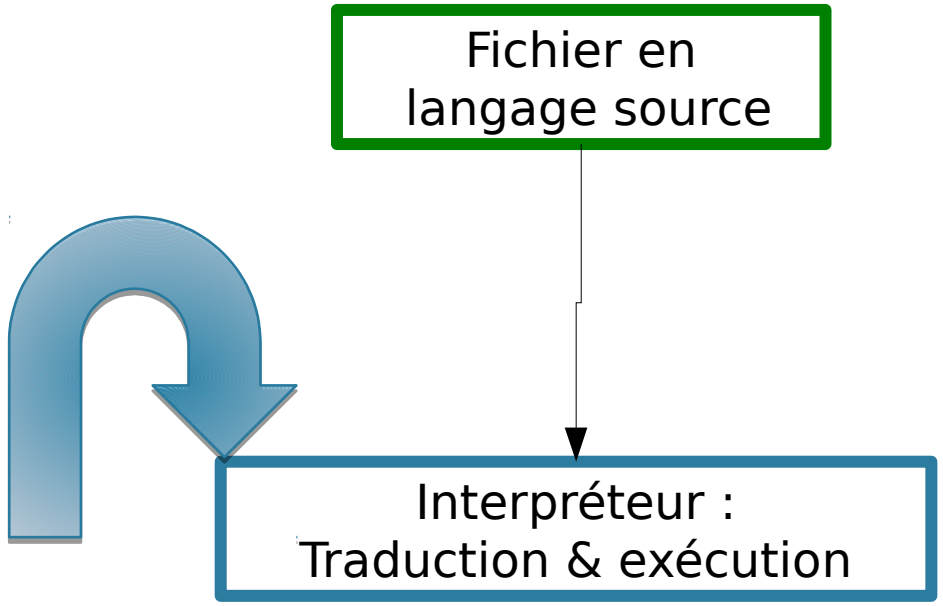
5) Programme compilé



Processus en cours d'exécution

5) Programme interprété

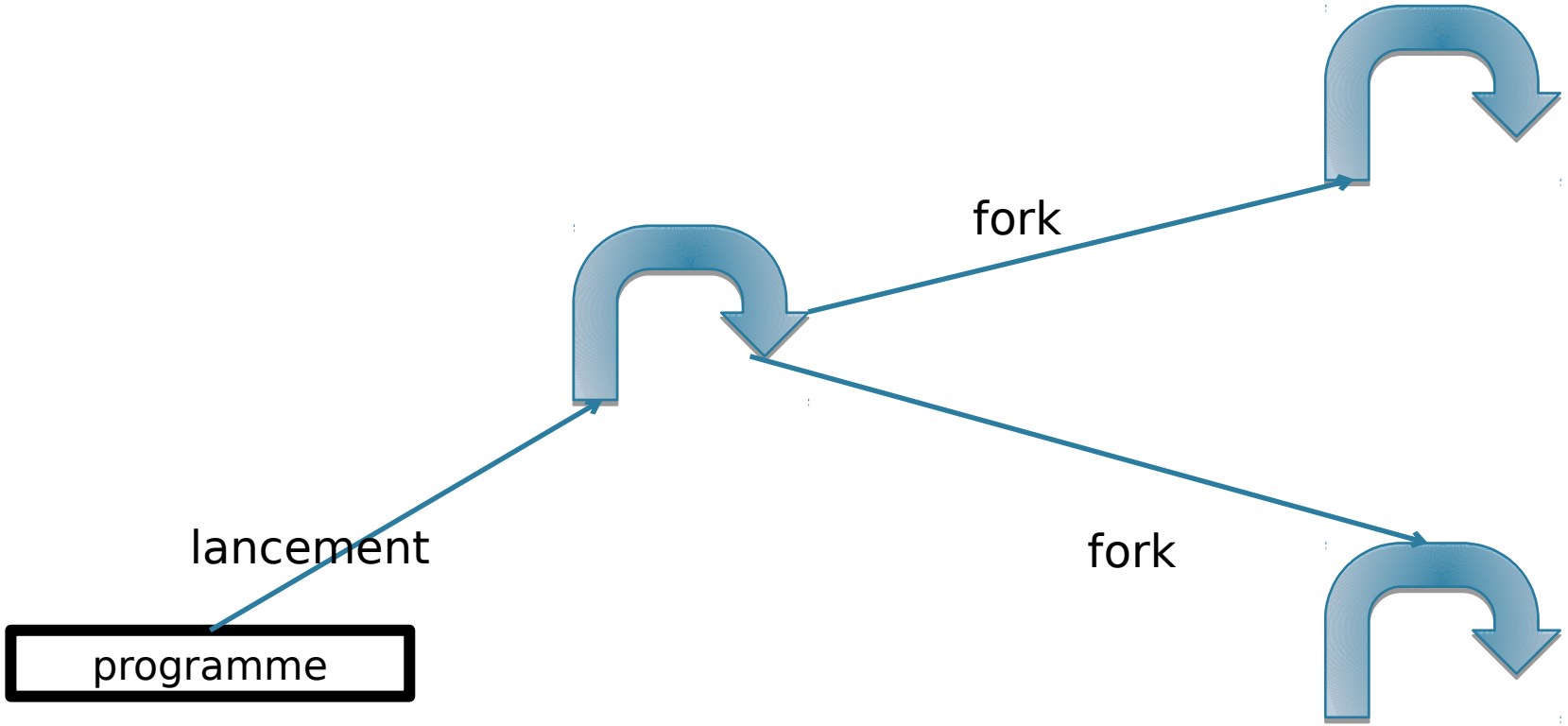
Processus en cours d'exécution



5) Langages compilés // interprétés

- Compilés
  - FORTRAN, COBOL, PASCAL, ADA
  - C, C++, C#
- Interprétés
  - python, ruby, TCL/Tk
  - PERL
  - php, javascript
  - tous les shells d'Unix : sh, csh, bash
- Compilé puis interprété : JAVA

5) Création de processus



5) Tâches

- un processus peut lui-même effectuer plusieurs tâches (threads) en même temps (programmation parallèle).
- Si le système ou le langage le permet.
- C'est possible en python => multi-threading

5) Paradigmes de programmation

- impérative // déclarative
- séquentielle
- procédurale
- modulaire
- parallèle
- événementielle
- fonctionnelle
- logique
- orientée objet

5) Programmation impérative // déclarative

- impérative : l'instruction effectue une action
  - en python → x = 0
- déclarative : l'instruction décrit :
  - un format : en HTML → <b>texte en gras>/b>
  - une structuration : en HTML → <header>...</header><body>...
  - une règle, un fait : en PROLOG →

```
mortal(X) :-
    human(X).

human(socrates).
```

5) Programmation séquentielle

- Une application est constituée :
  - d'un seul fichier source
  - une seule procédure, une seule fonction, la fonction principale (main)
- Pénible à écrire, tester et maintenir
- Exemple : Cobol à ses débuts (1958)

```
000340 01   CAT-TYPE     PIC X(15) VALUE 'CALICO'.
000360 01   CAT-COUNT    PIC 99 VALUE 15.

004310
004320      PERFORM UNTIL CAT-TYPE(CAT-COUNT:1) NOT = SPACE
004325          OR CAT-COUNT < 1
004330          SUBTRACT 1 FROM CAT-COUNT
004340      END-PERFORM.
004440
```

5) Programmation procédurale

- Dans un même programme, il est fréquent que la même séquence d'instructions soit utilisée plusieurs fois, à des endroits différents du programme.
- Pour éviter de réécrire (copier/coller) ces séquences d'instructions chaque fois que vous en avez besoin, vous pouvez les appeler grâce à des procédures.
- Des procédures peuvent accepter des paramètres
- Des fonctions peuvent retourner une valeur (code retour)
- Exemple : PASCAL

5) Programmation modulaire

- Le code source de l'application peut être réparti entre plusieurs fichiers =>
  - ce qui permet de manipuler des fichiers plus petits
  - de travailler à plusieurs sur une même application
  - compilation séparée (on ne vérifie la syntaxe et on ne génère le code que d'un seul petit fichier au lieu de tout le programme)
  - construction et utilisation de **bibliothèques** de fonctions communes à toutes les applications => API
- Exemple : le langage C

5) Programmation concurrente

- La programmation concurrente (parallèle) consiste à effectuer des tâches en parallèle dans un même programme.
- Cette possibilité peut être mise en œuvre soit :
  - par le système d'exploitation (Threads de Windows et Linux)
  - par la machine virtuelle (Java) ou l'interpréteur (Javascript)
  - ou directement par le langage (OCCAM)
- C'est encore plus efficace aujourd'hui grâce à la disponibilité de microprocesseurs multi-cœurs.

5) Programmation événementielle

- Les instructions ne sont plus exécutées dans l'ordre du fichier source, mais lorsque certains événements se produisent.

- Une instruction comprend 3 parties :
  1. un événement :
     - un bouton a été appuyé
     - un message est arrivé sur une interface réseau
     - la fin d'un fichier a été atteinte ...
  2. un contexte :
     - ce bouton
     - cette socket (port) réseau
     - ce fichier ...
  3. une action : les instructions à exécuter ou une fonction à appeler

- Exemple : TCL/Tk, les bibliothèques d'interfaces graphiques (gTk, awt, swing, ...)

5) Programmation logique

- Utilisé en intelligence artificielle, la programmation logique consiste à déclarer
  - une liste de règles
  - une liste de faits
- puis de poser une question qui va être confronté aux faits et aux règles
- Exemple : PROLOG
  ?- mortal(socrates).

5) Programmation orientée objet

- Un objet encapsule (contient) des données (variables, propriétés, attributs) et des méthodes (code)
- Les méthodes s'appliquent aux données encapsulées dans l'objet
- Les objets logiciels doivent être conçus pour ressembler (d'une manière simplifiée) aux objets réels (facture, voiture, robot, ...) qu'ils représentent (modélisation)
- L'état d'un objet réel est stocké dans les attributs de l'objet logiciel
- Le comportement d'un objet réel est implanté dans les méthodes de l'objet logiciel

5) Langages orientés objet

- Objet :
  - JAVA
  - C#
  - Smalltalk
- Objet, mais pas que :
  - C++
  - python
  - javascript

5) Programmation fonctionnelle

- une fonction est une variable qui contient le résultat de la fonction pour le paramètre passé à la fonction
- la fonction n'est pas recalculée si le paramètre est le même => le résultat doit être le même
- Exemples :
  - CAML, F#
  - SCALA

6) Débogage

- l'activité de débogage consiste à identifier et corriger les erreurs de programmation.
- il y plusieurs types d'erreurs :
  - erreurs de syntaxe : s'il y a dse erruers de syntaxe, l'ordniateur n'est psa caapble de comprendre les insturctions !
  - erreurs de référencement : par exemple l'appel d'une fonction non définie
  - erreurs sémantiques : le programme ne fonctionne pas comme prévu

Le debogueur

- Outil qui permet de voir, instruction après instruction, l'exécution du script ou du programme
- Permet de voir les variables existantes, leurs valeurs
  - S'assurer que le fonctionnement est le bon
  - Repérer les dysfonctionnements
- utilisation du débogueur de PyCharm