Algorithmes, programmes, processus

Prof d'info

Lycée Thiers

Crédits

- paradigmes de programmation Wikipedia

Crédits

- paradigmes de programmation Wikipedia
- programme vs algorithme edu Java

Algorithme

- Un *algorithme* est une méthode, une façon de résoudre un problème. C'est un concept.

Algorithme

- Un *algorithme* est une méthode, une façon de résoudre un problème. C'est un concept.
- En général, il se présente sous la forme d'un ensemble d'instructions qui indiquent, étape par étape, ce qu'il faut faire pour la résolution.

Algorithme

- Un *algorithme* est une méthode, une façon de résoudre un problème. C'est un concept.
- En général, il se présente sous la forme d'un ensemble d'instructions qui indiquent, étape par étape, ce qu'il faut faire pour la résolution.
- C'est donc un concept que l'on peut décrire de plusieurs manières, par exemple en français, ou en utilisant un graphique, ou encore *pseudo-code*

Programme

Un *programme* est indissociable de l'existence d'un ordinateur capable de l'exécuter.

- C'est la transcription d'un algorithme dans un langage compréhensible par la machine (donc écrit directement en langage machine ou bien écrit d'abord en langage de programmation puis traduit)

Programme

Un *programme* est indissociable de l'existence d'un ordinateur capable de l'exécuter.

- C'est la transcription d'un algorithme dans un langage compréhensible par la machine (donc écrit directement en langage machine ou bien écrit d'abord en langage de programmation puis traduit)
- Le programme lit des données depuis une *entrée* (un fichier, une bases de données, un réseau ou encore un clavier) et il écrit d'autres données sur une *sortie* (un écran, un fichier, une base de données, un réseau, une feuille de papier via une imprimante) etc.

Programme vs algorithme

- Lorsque nous parlons de « programme », c'est toujours dans l'idée qu'il doit être exécuté par un ordinateur tandis qu'un algorithme peut être réalisé par une personne.

Programme vs algorithme

- Lorsque nous parlons de « programme », c'est toujours dans l'idée qu'il doit être exécuté par un ordinateur tandis qu'un algorithme peut être réalisé par une personne.
- Dans le travail du développeur, l'algorithme précède le programme :

Programme vs algorithme

- Lorsque nous parlons de « programme », c'est toujours dans l'idée qu'il doit être exécuté par un ordinateur tandis qu'un algorithme peut être réalisé par une personne.
- Dans le travail du développeur, l'algorithme précède le programme :
  1. concevoir un algorithme pour résoudre le problème (en général sur papier),

Programme vs algorithme

- Lorsque nous parlons de « programme », c'est toujours dans l'idée qu'il doit être exécuté par un ordinateur tandis qu'un algorithme peut être réalisé par une personne.
- Dans le travail du développeur, l'algorithme précède le programme :
  1. concevoir un algorithme pour résoudre le problème (en général sur papier),
  2. écrire cet algorithme dans un *langage de programmation* (JAVA, OCAML, C...) puis, selon les langages, le transmettre à un *interpréteur* ou un *compilateur* (on dit alors qu'on a *implémenté* l'algorithme).

Programme vs algorithme

- Lorsque nous parlons de « programme », c'est toujours dans l'idée qu'il doit être exécuté par un ordinateur tandis qu'un algorithme peut être réalisé par une personne.
- Dans le travail du développeur, l'algorithme précède le programme :
  1. concevoir un algorithme pour résoudre le problème (en général sur papier),
  2. écrire cet algorithme dans un *langage de programmation* (JAVA, OCAML, C...) puis, selon les langages, le transmettre à un *interpréteur* ou un *compilateur* (on dit alors qu'on a *implémenté* l'algorithme).
- La distinction entre programme et algorithme est de la même nature qu'entre un livre et une histoire.

Pseudo-code

- Les *pseudo-codes* sont des langages informels de description des algorithmes.

Pseudo-code

- Les *pseudo-codes* sont des langages informels de description des algorithmes.
- C'est une voie intermédiaire entre le *langage naturel* (français, anglais, serbo-croate...) et un langage de programmation.

Pseudo-code

- Les *pseudo-codes* sont des langages informels de description des algorithmes.
- C'est une voie intermédiaire entre le *langage naturel* (français, anglais, serbo-croate...) et un langage de programmation.
- Il y a autant de façon d'écrire du pseudo-code que d'auteurs. Un code écrit en pseudo-code a vocation à être lu par un humain. Le pseudo-code se doit d'être compréhensible immédiatement

Pseudo-code

- Les *pseudo-codes* sont des langages informels de description des algorithmes.
- C'est une voie intermédiaire entre le *langage naturel* (français, anglais, serbo-croate...) et un langage de programmation.
- Il y a autant de façon d'écrire du pseudo-code que d'auteurs. Un code écrit en pseudo-code a vocation à être lu par un humain. Le pseudo-code se doit d'être compréhensible immédiatement
  - sans qu'il soit nécessaire de perdre du temps à assimiler une syntaxe particulière.

Pseudo-code

- Les *pseudo-codes* sont des langages informels de description des algorithmes.
- C'est une voie intermédiaire entre le *langage naturel* (français, anglais, serbo-croate...) et un langage de programmation.
- Il y a autant de façon d'écrire du pseudo-code que d'auteurs. Un code écrit en pseudo-code a vocation à être lu par un humain. Le pseudo-code se doit d'être compréhensible immédiatement
  - sans qu'il soit nécessaire de perdre du temps à assimiler une syntaxe particulière.
  - ni de décrire finement comment accéder aux données ou au matériel (mémoire, écran, clavier etc.).

Pseudo-code

Le pseudo-code est un des moyens de décrire un algorithme.

- Utilisation de *variables* et de leurs *assignations* (exemple : $\mathrm{a} \leftarrow 5$ signifie qu'on assigne la constante 5 à la variable a).

Pseudo-code

Le pseudo-code est un des moyens de décrire un algorithme.

- Utilisation de *variables* et de leurs *assignations* (exemple : $\mathrm{a} \leftarrow 5$ signifie qu'on assigne la constante 5 à la variable a).
- Indication des données d'entrée et de sortie.

Pseudo-code

Le pseudo-code est un des moyens de décrire un algorithme.

- Utilisation de *variables* et de leurs *assignations* (exemple : $\mathrm{a} \leftarrow 5$ signifie qu'on assigne la constante 5 à la variable a).
- Indication des données d'entrée et de sortie.
- Utilisation d'*instructions conditionnelles*. Par exemple :
  si $<$ condition $>$ **alors** $<$ instruction $>$ **sinon** $<$ instruction $>$

Pseudo-code

Le pseudo-code est un des moyens de décrire un algorithme.

- Utilisation de *variables* et de leurs *assignations* (exemple : $\mathrm{a} \leftarrow 5$ signifie qu'on assigne la constante 5 à la variable a).
- Indication des données d'entrée et de sortie.
- Utilisation d'*instructions conditionnelles*. Par exemple :
  si $<$ condition $>$ **alors** $<$ instruction $>$ **sinon** $<$ instruction $>$
- Utilisations d'instructions de boucles. Par exemple :
  répéter $<$ instruction $>$ **jusqu'à** $<$ condition $>$

Pseudo-code

Le pseudo-code est un des moyens de décrire un algorithme.

- Utilisation de *variables* et de leurs *assignations* (exemple : $\mathrm{a} \leftarrow 5$ signifie qu'on assigne la constante 5 à la variable a).
- Indication des données d'entrée et de sortie.
- Utilisation d'*instructions conditionnelles*. Par exemple :
  si $<$ condition $>$ **alors** $<$ instruction $>$ **sinon** $<$ instruction $>$
- Utilisations d'instructions de boucles. Par exemple :
  répéter $<$ instruction $>$ **jusqu'à** $<$ condition $>$
- mots différents identifiants *fonctions* et *procédures*

Pseudo-code

Exemple d'un algorithme écrit en pseudo-code

```
1  fonction triangle
2          /* Détermine la nature d'un triangle */
3          entree : side1, side2, side3 /*lg des 3 côtés*/
4          sortie : nature du triangle (quelconque, isocèle, équilatér
5
6          si side1 = side2 ET side1 = side3
7          alors
8                  nature ← equilateral
9          sinon
10                 debut
11                 si  (side1 = side2) OU (side1 = side3) OU (side2 =
12                   alors
13                         nature ← isocèle
14                   sinon
15                         nature ← quelconque
16                 fin
17         renvoyer nature
```

On s'attache à l'idée. On ne sait pas trop ce que sont `quelconque, isocèle, équilatéral`.

Langage de programmation

- Un *langage de programmation* (PYTHON, C, OCAML, HTML...) est un langage contraint par des règles syntaxiques précises comme d'ailleurs le français, l'anglais ou le mandarin.

Langage de programmation

- Un *langage de programmation* (PYTHON, C, OCAML, HTML...) est un langage contraint par des règles syntaxiques précises comme d'ailleurs le français, l'anglais ou le mandarin.
- Cependant, contrairement aux langages naturels, un texte écrit dans un langage de programmation a une et une seule signification (on dit aussi *sémantique*). Les phrases suivantes sont par exemple écrites dans un français correct :

Langage de programmation

- Un *langage de programmation* (PYTHON, C, OCAML, HTML...) est un langage contraint par des règles syntaxiques précises comme d'ailleurs le français, l'anglais ou le mandarin.
- Cependant, contrairement aux langages naturels, un texte écrit dans un langage de programmation a une et une seule signification (on dit aussi *sémantique*). Les phrases suivantes sont par exemple écrites dans un français correct :

  Existence « La pilosité musicale suppose rouge ». Il est clair que cette phrase n'a pas de sens sauf pour un poète surréaliste.

Langage de programmation

- Un *langage de programmation* (PYTHON, C, OCAML, HTML...) est un langage contraint par des règles syntaxiques précises comme d'ailleurs le français, l'anglais ou le mandarin.
- Cependant, contrairement aux langages naturels, un texte écrit dans un langage de programmation a une et une seule signification (on dit aussi *sémantique*). Les phrases suivantes sont par exemple écrites dans un français correct :

  Existence « La pilosité musicale suppose rouge ». Il est clair que cette phrase n'a pas de sens sauf pour un poète surréaliste.

  Unicité « Voici un gâteau et un vin que j'aime beaucoup ». Malheureusement, on ne sait pas si la personne aime le vin uniquement ou bien le gateau ET le vin.

Langage de programmation

- Ces deux situations (absence de sens et double signification) ne doivent pas se produire avec un langage de programmation (et ne se produisent pas).

Langage de programmation

- Ces deux situations (absence de sens et double signification) ne doivent pas se produire avec un langage de programmation (et ne se produisent pas).
- C'est cette correspondance univoque (une phrase correcte a une et une seule signification) qui permet à un programme particulier (appelé *compilateur* ou *interpréteur*) de transformer un script écrit en langage C (ou JAVA, OCAML, PYTHON etc...) en langage machine compréhensible par l'ordinateur.

Langage de programme vs algorithme

- Comme le pseudo-code, un langage de programmation sert donc à décrire des algorithmes et est compréhensible par un humain (avec plus ou moins d'entraînement).

Langage de programme vs algorithme

- Comme le pseudo-code, un langage de programmation sert donc à décrire des algorithmes et est compréhensible par un humain (avec plus ou moins d'entraînement).
- Mais il a vocation à être ensuite transformé par un compilateur ou un interpréteur en une suite d'instructions exécutables par un ordinateur.

Langage de programme vs algorithme

- Comme le pseudo-code, un langage de programmation sert donc à décrire des algorithmes et est compréhensible par un humain (avec plus ou moins d'entraînement).
- Mais il a vocation à être ensuite transformé par un compilateur ou un interpréteur en une suite d'instructions exécutables par un ordinateur.
- Un compilateur ou un interpréteur joue donc le rôle de traducteur d'un langage de programmation en *langage machine*.

Langage de programme vs algorithme

- Comme le pseudo-code, un langage de programmation sert donc à décrire des algorithmes et est compréhensible par un humain (avec plus ou moins d'entraînement).
- Mais il a vocation à être ensuite transformé par un compilateur ou un interpréteur en une suite d'instructions exécutables par un ordinateur.
- Un compilateur ou un interpréteur joue donc le rôle de traducteur d'un langage de programmation en *langage machine*.
- Il y a au moins autant de compilateurs ou interpréteurs que de langages de programmation (en fait il y en a plus : par exemple GCC et CLANG sont deux compilateurs du langage C).

Exemple

Voici l'algorithme des triangles écrit en OCAML :

```
type triangle = Isocele | Equilateral | Quelconque;;

let nature_triangle s1 s2 s3 =
  if s1 = s2 && s2 = s3
  then Equilateral
  else
    if s1 = s2 || s2 = s3 || s3 = s1
    then Isocele
    else Quelconque;;
```

Exemple

Le même en langage C :

```c
#include <stdio.h>

enum triangle { Isocele , Equilateral , Quelconque };


enum triangle nature_triangle (int s1, int s2, int s3)
{
  if (s1 == s2 && s2 == s3) {return Equilateral;}
     else
        {
   if (s1 == s2 || s2 == s3 || s3 == s1) {return Isocele;}
   else {return Quelconque;}
        };
}

```

Notion de processus

- Un *processus* est un programme en cours d'exécution par un ordinateur.

Notion de processus

- Un *processus* est un programme en cours d'exécution par un ordinateur.
- Un programme est une suite d'instructions écrites en langage machine (par exemple sur le disque). Il est statique.

Notion de processus

- Un *processus* est un programme en cours d'exécution par un ordinateur.
- Un programme est une suite d'instructions écrites en langage machine (par exemple sur le disque). Il est statique.
- Le processus est un concept dynamique. Il a un début, dure un certain temps et (souvent) se termine. Il est composé de

Notion de processus

- Un *processus* est un programme en cours d'exécution par un ordinateur.
- Un programme est une suite d'instructions écrites en langage machine (par exemple sur le disque). Il est statique.
- Le processus est un concept dynamique. Il a un début, dure un certain temps et (souvent) se termine. Il est composé de
  - un ensemble d'instructions (le programme) souvent chargé depuis la mémoire de masse vers la mémoire vive.

Notion de processus

- Un *processus* est un programme en cours d'exécution par un ordinateur.
- Un programme est une suite d'instructions écrites en langage machine (par exemple sur le disque). Il est statique.
- Le processus est un concept dynamique. Il a un début, dure un certain temps et (souvent) se termine. Il est composé de
  - un ensemble d'instructions (le programme) souvent chargé depuis la mémoire de masse vers la mémoire vive.
  - un espace d'adressage en mémoire vive pour stocker la *pile*, les données de travail etc.

Notion de processus

- Un *processus* est un programme en cours d'exécution par un ordinateur.
- Un programme est une suite d'instructions écrites en langage machine (par exemple sur le disque). Il est statique.
- Le processus est un concept dynamique. Il a un début, dure un certain temps et (souvent) se termine. Il est composé de
  - un ensemble d'instructions (le programme) souvent chargé depuis la mémoire de masse vers la mémoire vive.
  - un espace d'adressage en mémoire vive pour stocker la *pile*, les données de travail etc.
  - des ressources permettant des entrées sorties de données comme des ports-réseaux

Notion de processus

- Un *processus* est un programme en cours d'exécution par un ordinateur.
- Un programme est une suite d'instructions écrites en langage machine (par exemple sur le disque). Il est statique.
- Le processus est un concept dynamique. Il a un début, dure un certain temps et (souvent) se termine. Il est composé de
  - un ensemble d'instructions (le programme) souvent chargé depuis la mémoire de masse vers la mémoire vive.
  - un espace d'adressage en mémoire vive pour stocker la *pile*, les données de travail etc.
  - des ressources permettant des entrées sorties de données comme des ports-réseaux
- Un processus peut être démarré par un utilisateur, un périphérique ou un autre processus.

Notion de processus

- Un *processus* est un programme en cours d'exécution par un ordinateur.
- Un programme est une suite d'instructions écrites en langage machine (par exemple sur le disque). Il est statique.
- Le processus est un concept dynamique. Il a un début, dure un certain temps et (souvent) se termine. Il est composé de
  - un ensemble d'instructions (le programme) souvent chargé depuis la mémoire de masse vers la mémoire vive.
  - un espace d'adressage en mémoire vive pour stocker la *pile*, les données de travail etc.
  - des ressources permettant des entrées sorties de données comme des ports-réseaux
- Un processus peut être démarré par un utilisateur, un périphérique ou un autre processus.
- Un programme décrit une façon de procéder. Il n'*agit* pas. c'est un processus dont le code est ce programme qui *agit*.

Diagramme d'état simplifié

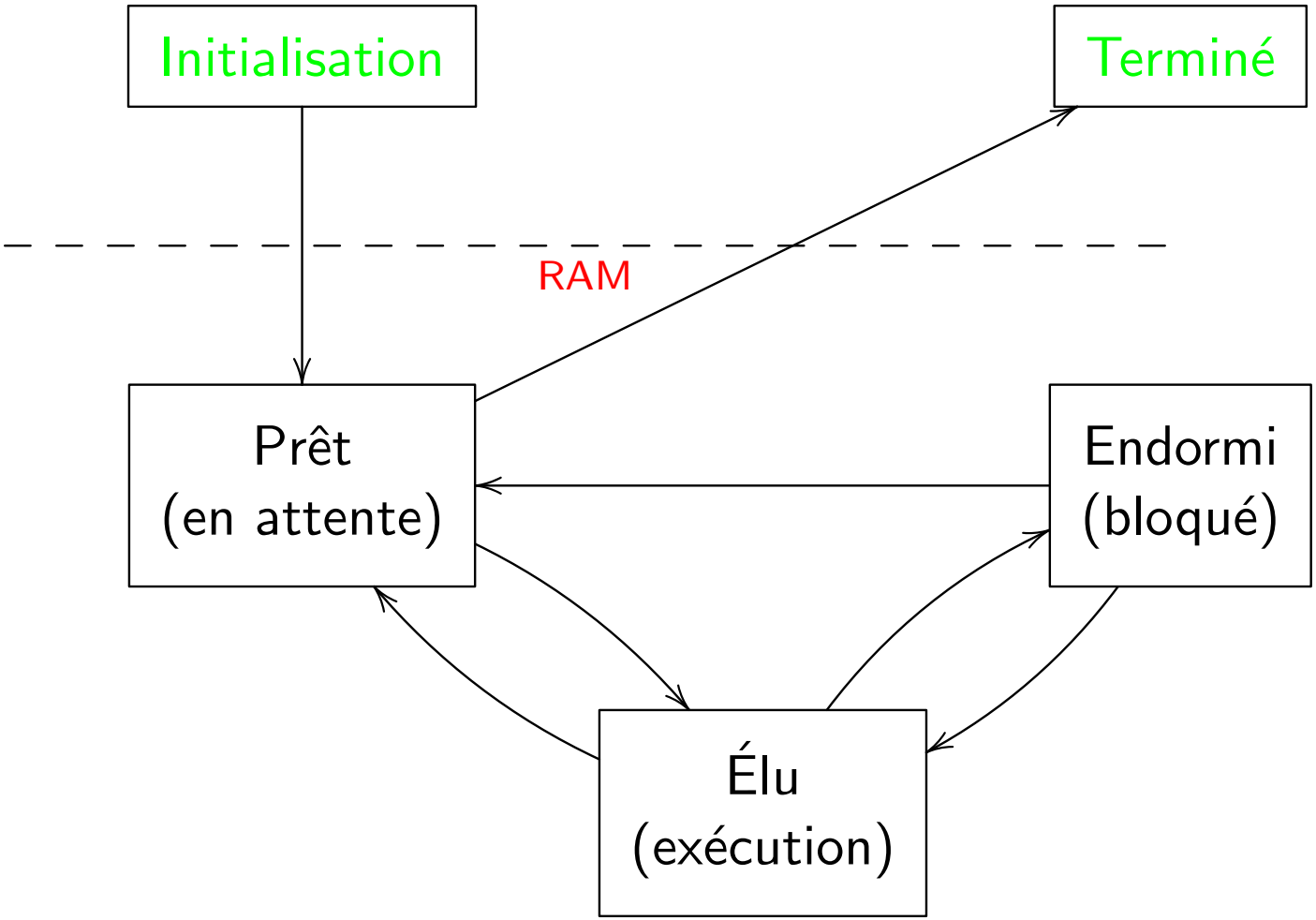

Diagramme d'état simplifié (Wikipedia)

- L'*Initialisation* est le 1er état d'un processus. Il attend que l'ordonnanceur le place dans l'état *prêt*.

Diagramme d'état simplifié (Wikipedia)

- L'*Initialisation* est le 1er état d'un processus. Il attend que l'ordonnanceur le place dans l'état *prêt*.
- Dans l'état *prêt*, le processus a été chargé en mémoire vive et attend l'accès au processeur.

Diagramme d'état simplifié (Wikipedia)

- L'*Initialisation* est le 1er état d'un processus. Il attend que l'ordonnanceur le place dans l'état *prêt*.
- Dans l'état *prêt*, le processus a été chargé en mémoire vive et attend l'accès au processeur.
- Dans l'état *Élu*, le processus est en cours d'exécution par le processeur. Souvent, il ne peut pas s'exécuter en entier dans le court laps de temps où il dispose du CPU. Il s'exécute par *bouts* et alterne les états Élu/Endormi/Prêt.

Diagramme d'état simplifié (Wikipedia)

- L'*Initialisation* est le 1er état d'un processus. Il attend que l'ordonnanceur le place dans l'état *prêt*.
- Dans l'état *prêt*, le processus a été chargé en mémoire vive et attend l'accès au processeur.
- Dans l'état *Élu*, le processus est en cours d'exécution par le processeur. Souvent, il ne peut pas s'exécuter en entier dans le court laps de temps où il dispose du CPU. Il s'exécute par *bouts* et alterne les états Élu/Endormi/Prêt.
- Dans l'état *Endormi*, le processus a été interrompu ou attend un événement (la fin d'une opération d'entrée/sortie, un signal, ...).

Diagramme d'état simplifié (Wikipedia)

- L'*Initialisation* est le 1er état d'un processus. Il attend que l'ordonnanceur le place dans l'état *prêt*.
- Dans l'état *prêt*, le processus a été chargé en mémoire vive et attend l'accès au processeur.
- Dans l'état *Élu*, le processus est en cours d'exécution par le processeur. Souvent, il ne peut pas s'exécuter en entier dans le court laps de temps où il dispose du CPU. Il s'exécute par *bouts* et alterne les états Élu/Endormi/Prêt.
- Dans l'état *Endormi*, le processus a été interrompu ou attend un événement (la fin d'une opération d'entrée/sortie, un signal, ...).
- Dans l'état *Terminé*, le processus est... terminé ! Soit parce qu'il est arrivé au bout de ce qu'il devait faire, soit parce qu'il a été forcé de s'arrêter.

États particuliers

D'autres états sont possibles (cela dépend des systèmes d'exploitation) :

- *Zombie* : Si un processus terminé ne peut pas être déchargé de la mémoire, par exemple parce que son processus parent n'a pas récupéré son signal de terminaison, il passe dans cet état.

États particuliers

D'autres états sont possibles (cela dépend des systèmes d'exploitation) :

- *Zombie* : Si un processus terminé ne peut pas être déchargé de la mémoire, par exemple parce que son processus parent n'a pas récupéré son signal de terminaison, il passe dans cet état.
- *Swappé* : Lorsqu'un processus est transféré de la mémoire centrale dans la mémoire virtuelle, il est dit « swappé ». Un processus swappé peut être dans un état endormi ou prêt.

États particuliers

D'autres états sont possibles (cela dépend des systèmes d'exploitation) :

- *Zombie* : Si un processus terminé ne peut pas être déchargé de la mémoire, par exemple parce que son processus parent n'a pas récupéré son signal de terminaison, il passe dans cet état.
- *Swappé* : Lorsqu'un processus est transféré de la mémoire centrale dans la mémoire virtuelle, il est dit « swappé ». Un processus swappé peut être dans un état endormi ou prêt.
- *Préempté*. L'ordonnanceur a décidé de suspendre l'activité d'un processus.
  Par exemple, un processus qui consomme trop de temps CPU finira par être préempté. Un ordonnanceur préemptif utilise aussi l'indice de priorité (nice) pour décider le processus qui sera préempté.