Ordonnancement - implémentation

Christophe Viroulaud

Terminale - NSI

Archi 05

Le processeur peut adopter plusieurs stratégies pour exécuter l'enchaînement des processus. Selon l'algorithme utilisé la structure adoptée pour stocker la liste des tâches a une importance fondamentale.

Quelles structures de données adopter pour implémenter les algorithmes d'ordonnancement ?

Sommaire

Pile

À retenir

Les piles (*stack*) sont fondées sur le principe du *dernier arrivé premier sorti* : **L**ast **I**n **F**irst **O**ut.

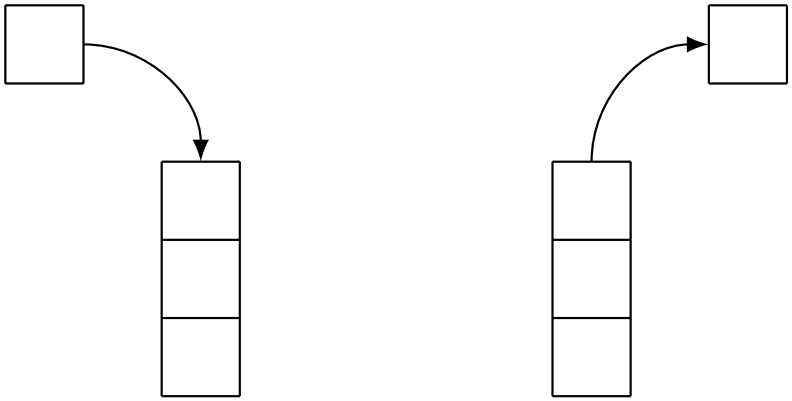

FIGURE 1 – Empiler - dépiler

Interface d'une pile

Une pile stocke des éléments de type `T` quelconque.

- `creer_pile()` $\rightarrow$ `Pile()` : crée une pile vide
- `est_vide(p: Pile)` $\rightarrow$ `bool` : renvoie `True` si la pile est vide, `False` sinon.
- `empiler(p: Pile, e: T)` $\rightarrow$ `None` : ajoute un élément e au sommet de la pile.
- `depiler(p: Pile)` $\rightarrow$ `T` : retire et renvoie l'élément du sommet de la pile.

Implémentation

- `creer_pile()` → `Pile()`
- `est_vide(p: Pile)` → `bool`
- `empiler(p: Pile, e: T)` → `None`
- `depiler(p: Pile)` → `T`

Activité 1 :

1. Créer une classe `Noeud`. Son constructeur initialisera deux attributs :
   - `donnees: int`
   - `successeur: Noeud`
2. Écrire les fonctions de l'interface d'une pile. On considérera qu'elle stocke des entiers.