Introcution aux threads

PCO

1 - Introduction

Résumé du document

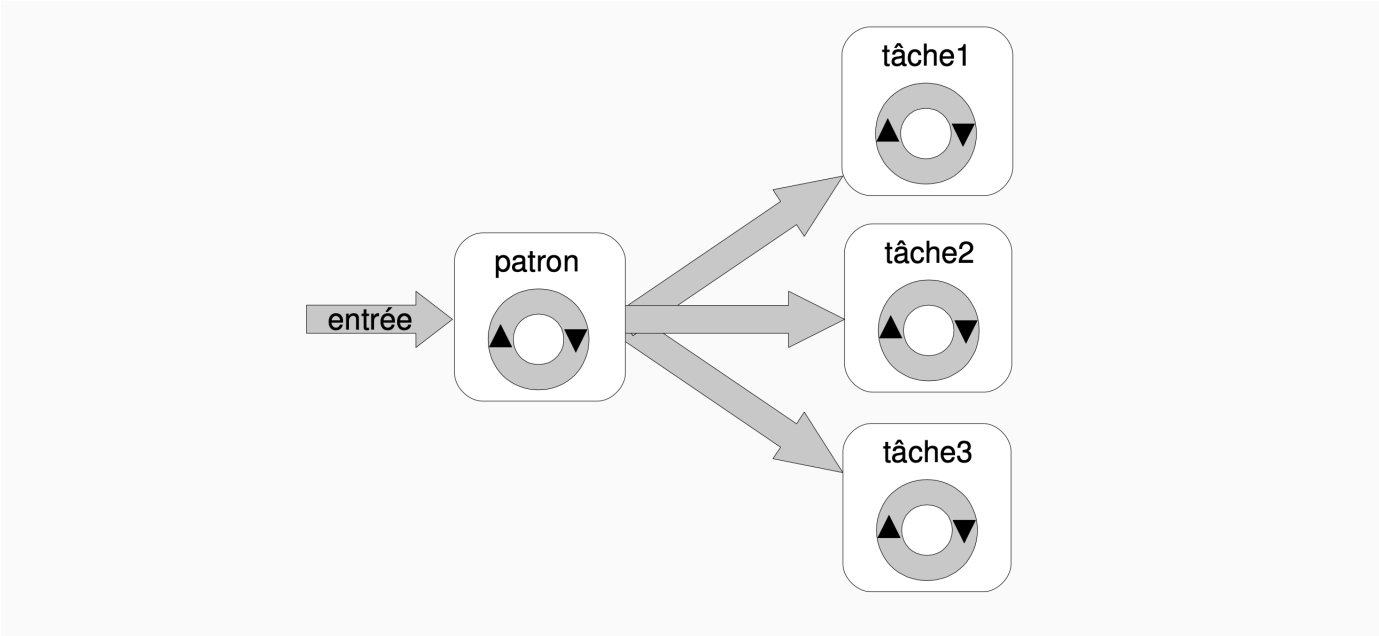
Table des matières

1. Modèles de délégation

1.1. Modèle de délégation (boos-worker model / delegation model)

Le modèle de délégation est un modèle de conception qui permet à un objet de déléguer une partie de son comportement à un autre objet. Ce modèle est utilisé pour déléguer des tâches à un autre objet. Il est utilisé pour réutiliser le code. Il est un moyen de réutiliser le code en utilisant la composition.

- Un thread principal
- Des threads travailleurs

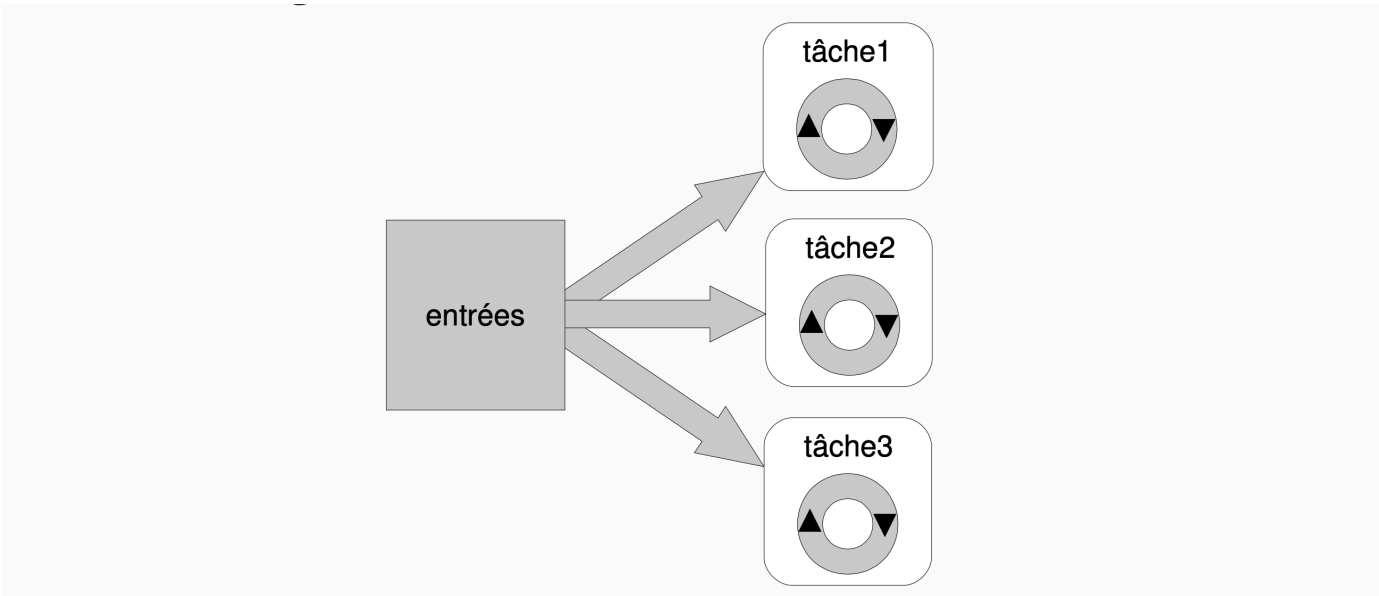



```
void tachePatron(void *ptr) {
  boucle infinie {
    attend une requête
      switch (requete) {
        case requeteX: startThread( ... tacheX); break;
        case requeteY: startThread( ... tacheY); break;
        ...
      }
    }
  }
void tacheX(void *ptr) {
  exécuter le travail demandé, puis se terminer
}
void tacheY(void *ptr) {
  exécuter le travail demandé, puis se terminer
}
```

1.2. Modèle pair (peer model)

Le modèle pair est un modèle de conception qui permet à un objet de partager une partie de son comportement avec un autre objet. Ce modèle est utilisé pour partager des tâches avec un autre objet. Il est utilisé pour partager des tâches entre deux objets. Il est un moyen de partager des tâches entre deux objets.

- Pas de thread principal
- Tous égaux
- Chacun s'arrange avec ses entrées/sorties




```
int main() {
  startThread( ... tache1);
  startThread( ... tache2);
  ...
  signale aux threads qu'ils peuvent commencer à travailler
}
void tache1() {
  attend le signal de commencement
  effectue le traitement, et synchronise avec les autres threads
  si nécessaire
}
void tache2() {
  attend le signal de commencement
  effectue le traitement, et synchronise avec les autres threads
  si nécessaire
}
```

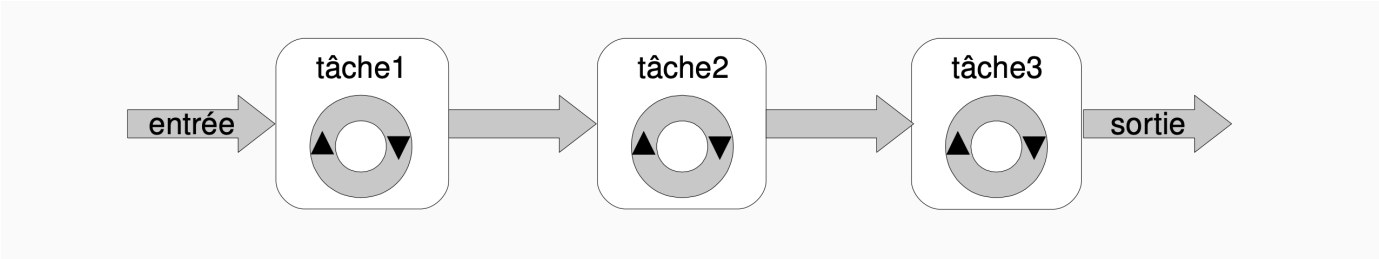
1.3. Modèle pipeline (pipeline model)

Le modèle pipeline est un modèle de conception qui permet à un objet de transmettre une partie de son comportement à un autre objet. Ce modèle est utilisé pour transmettre des tâches à un autre objet.

- Appliqué lorsque:
- L'application traite une longue chaîne d'entrée;
- Le traitement à effectuer sur ces entrées peut être décomposé en sous-tâches

(étages de pipeline) au travers desquelles chaque donnée d'entrée doit passer;

- Chaque étage peut traiter une donnée différente à chaque instant.
- Un thread attend les données du précédent
- Et les transmet ensuite au suivant




```
void etage1() {
  boucle infinie {
    récupérer une entrée du programme
    traiter cette donnée
    passer le résultat à l'étage suivant
  }
}
void etage2() {
  boucle infinie {
    récupérer une donnée de l'étage précédent
    traiter cette donnée
    passer le résultat à l'étage suivant
  }
}
void etageN() {
  boucle infinie {
    récupérer une donnée de l'étage précédent
    traiter cette donnée
    passer le résultat en sortie du programme
  }
}
```