Jeudi 17 Février 2022
Professeur : Khalid MEKOUAR

Projet d'Algorithmique (A rendre le Jeudi 21 Mai 2022)
2ème Année ESISA 2021-2022

Plus généralement pour, représenter une information, on a besoin de blocs de mémoire, fournis d'un mot ou de plusieurs mots consécutifs ; d'autres part, certaines modifications apportées à une information (adjonction d'un élément à une liste par exemple) demande l'attribution de nouveaux blocs en cours de traitement et il faut donc trouver des blocs libres. C'est le problème de l'allocation de mémoire.

Espace libre

Pour réaliser l'allocation, il est nécessaire de connaître l'ensemble des blocs qui n'entrent pas, ou n'entrent plus, à une instante donnée, dans la représentation d'informations en mémoire. De tels blocs sont dits libres. L'ensemble des blocs libres est appelé Espace libre.

Lors d'une allocation, on doit avoir accès à l'un des blocs de l'espace libre ; ce bloc est alors supprimé de l'espace libre et on doit pouvoir accéder à un autre bloc pour une allocation ultérieure.

Ce bilan des accès nécessaires invite à faire de l'espace libre une liste.

Lors d'une allocation on y prélève le bloc de tête (suppression en tête de la liste) et, lors d'une restitution, on y ajoute un bloc en tête (adjonction en tête de liste). De manière à faciliter ces modifications, on accède en tête de l'espace libre indirectement.

Cas d'un espace libre contiguë

L'ordre des allocations-restitutions est celui des entrées-sorties d'une pile. L'ensemble des blocs alloués et l'espace libre ont tous deux une structure de pile ; les adjonctions et suppressions se font uniquement en tête (fig. 1).

L est l'adresse d'un mot qui contient celle du bloc de tête de l'espace libre
Initialisation : l=début de l'adresse du début de la zone utilisée.
L'allocation se traduit par l = l + k, k étant la taille d'un bloc, après avoir éventuellement contrôlé que l+k ne dépasse pas la dernière adresse de l'espace libre.
La restitution d'un bloc est effectuée par l = l-k.

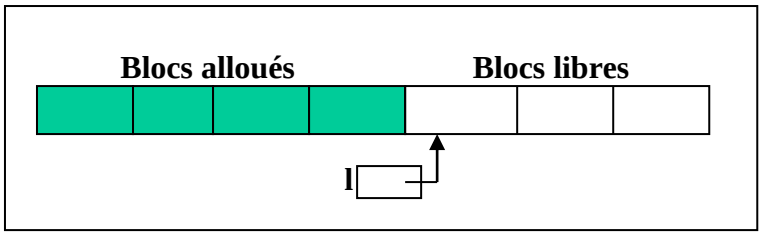

Fig. 1 Représentation contiguë

Cas d'un espace libre chaîné

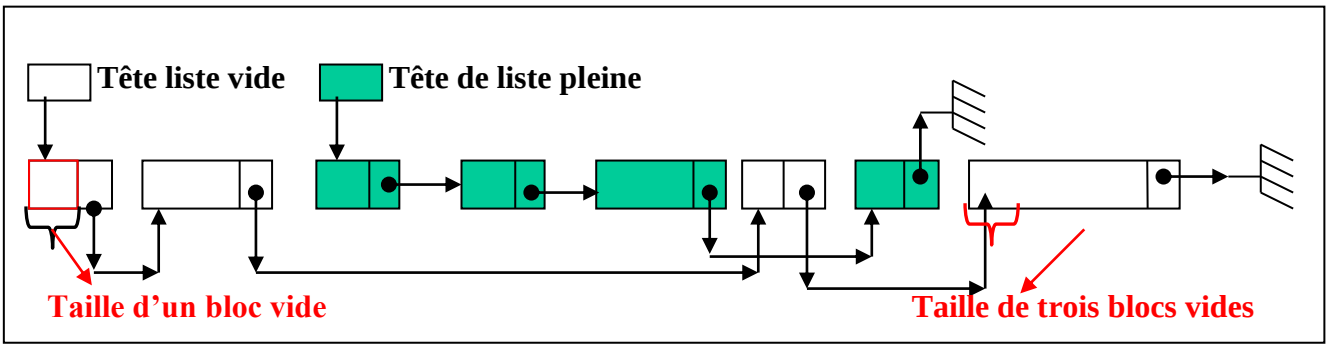

Fig. 2. Représentation chaînée

- **si on doit ajouter un élément en tête de la liste, on supprime une case (la taille d'un bloc mémoire) en tête de l'espace libre. (voir fig2)**
- **Si on supprime un élément en tête de la liste de donnée, on l'ajoute en tête de l'espace libre.**

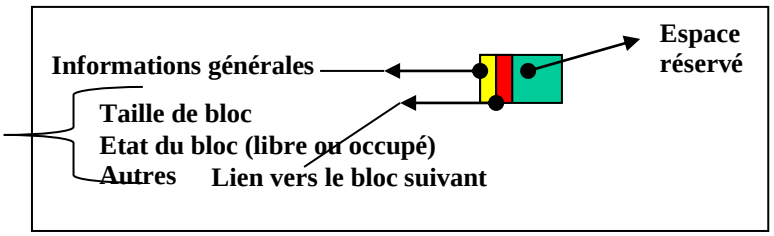

Fig 3 Constitution d'un bloc de données

On peut imaginer le scénario suivant :
Soit S une source génératrice de tâches, elle crée à chaque instant « t » une tâche ; ces tâches sont dirigées dans une file d'attente F.

La file d'attente délivre la première tâche dans un système à moulinette qu'on appelle système de Tourniquet.

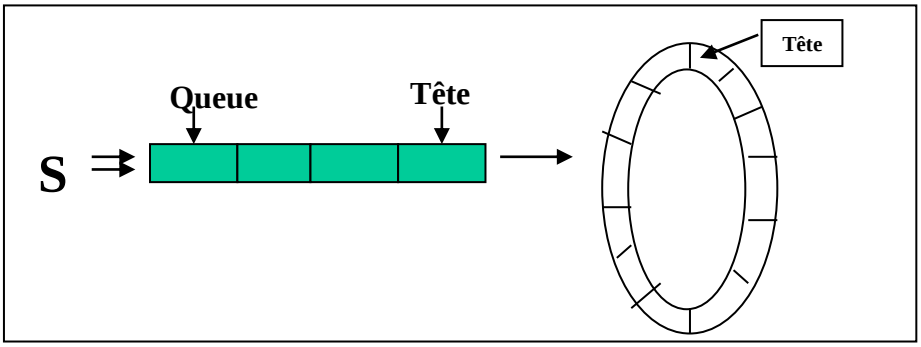

Fig. 4 Alimentation de la liste circulaire

Ce système de tourniquet ce n'est qu'une liste circulaire dont chaque cellule est représentée par la tâche correspondante. Le système exécute une partie de cette tâche et donne la main à la suivante jusqu'à épuisement de celle -ci

Une telle tâche est caractérisée par :

- **Son libellé « Lib_Task »**
- **Tâche à exécuter, qu'on l'a limitée à un affichage d'un contenu (chaîne de caractères) un caractère par abs de temps**
- **Son temps d'exécution « Time_Task » qu'on suppose connu à l'avance un multiple de n secondes**
- **Son espace mémoire « Mem_Task » requis supposé figé pendant son exécution (un multiple de 512 ko, un bloc = 512k). La réservation et la restitution se feront en nombre de blocs.**
- **Son degré de priorité « Level_Task » que pour l'instant fixe (toutes les tâches ont le même degré de priorité)**
- **Etat de la tâche « State_Task » (active ou dormante), (une tache dormante, une fois arrivé son tour d'exécution, elle donne la main à la suivante.**

On peut imaginer un espace d'exécution :

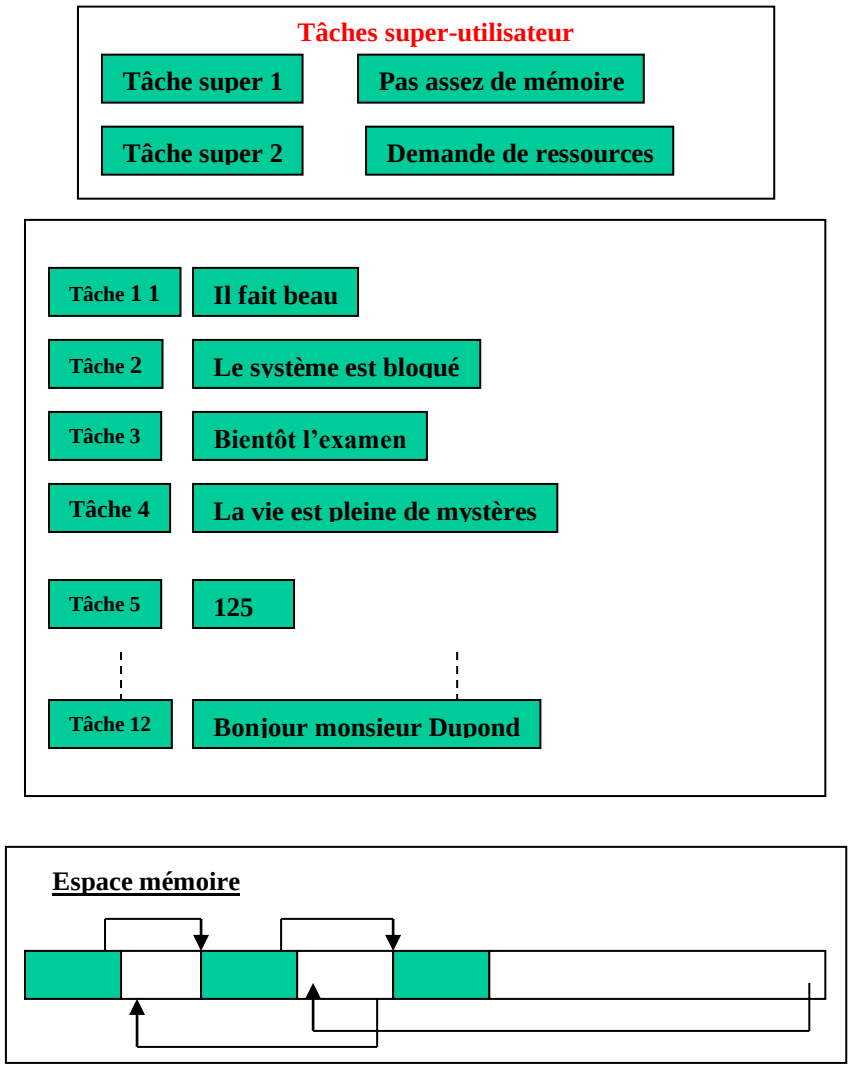

Fig. 5 Interface utilisateur

Ramasse miette ou garbage collector (cas général)

Lors de la suppression d'un élément d'une liste, on ne cherche pas à restituer le bloc qu'il occupait ; les restitutions sont différées jusqu'à ce qu'une demande d'allocation ne puisse pas être satisfaites, l'espace vide étant vide. On exécute alors un programme, appelé ramasse-miettes qui détermine les blocs occupés et restitue les autres à l'espace libre. Le ramasse-miettes contient deux parties :

1. **Détermination des blocs occupés :**
Tous les blocs ayant leurs bits d'occupation initialisé à 0, on parcourt successivement toutes les listes à partir des têtes t0, t1, t2, t3, …… tn en positionnant à 1 les bits d'occupation des blocs rencontrés, jusqu'à trouver, soit une fin de liste, soit un bit d'occupation déjà positionné.

2. **Restitution des blocs non occupés :**
On parcourt la mémoire (par exemple dans l'ordre des adresses croissantes) en restituant à l'espace libre les blocs dont les bits d'occupation sont à 0. On remet d'autre part à 0 les bits d'occupation positionnés à 1 dans la première étape, afin de préparer une exécution ultérieure du ramasse-miette.

Un bloc mémoire sera représenté par :

- **une information sur le bloc**
  - **Taille mémoire occupée (en nombre de blocs)**
  - **Etat du bloc (libre ou occupé)**
  - **Pointeur vers la zone suivante**
- **la zone en elle même**

Remarque :

L'interface proposée ce n'est qu'une idée très simpliste ; toutes suggestions sera bien acceptée et souhaitée.

Le langage de programmation est le langage « C » ou « C++ ».