Projet de TP: Manipulation de structures dynamiques en C

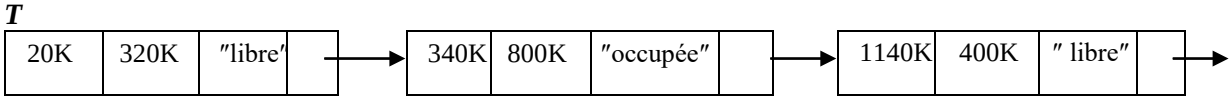
Politiques d'allocation de la mémoire à des processus

Objectifs du TP

- Manipulation de structures dynamiques en C, notamment les listes chaînées, files et piles
- Implémentation de politiques d'allocation de la mémoire à des programmes
- Ecriture et appel de fonctions
- Utilisation d'un menu à choix multiples
- Initiation aux fonctions d'affichage graphique en C
- Apprendre à rédiger un compte rendu de TP

Dans ce TP, on voudrait simuler l'organisation et le partage de la mémoire centrale (RAM) entre plusieurs programmes s'exécutant en parallèle sur un même ordinateur. Pour cela, on utilisera des structures de *listes chainées unidirectionnelles* et de *files*. Un programme admis à l'exécution est appelé *processus* et devra être chargé en mémoire RAM.
On suppose que la mémoire est partagée en ***partitions*** de tailles variables. Une partition est un espace mémoire contigu, ayant une adresse de début (où commence la partition), une taille en nombre d'octets et un état (libre ou occupée).
On considère que la description des partitions de la mémoire se trouve dans une ***liste chainée simple*** de point d'entrée ***T***, dans l'ordre croissant des adresses de partitions.

Exemple d'une liste de partitions



Partie I

Les processus qui sont admis dans le système pour exécution sont rangés selon l'ordre de leur arrivée (FIFO) dans une structure de ***File*** de points d'entrée ***Tete et Queue***. Un processus est décrit par un id, un temps (instant) d'arrivée et une durée d'exécution. Un processus qui arrive dans le système est ajouté en queue de file et un processus qui est admis à l'exécution est supprimé de la tête de la file.
Un processus qui est admis à l'exécution doit être chargé en mémoire dans une partition libre qui peut le contenir (càd taille de la partition >= taille du processus).
Lorsqu'un processus est chargé dans la partition convenable, on peut avoir création d'un résidu mémoire (c'est-à-dire une partition libre plus petite correspondant à ce qui reste de la partition, une fois le programme chargé).

Exemple : Si un processus de taille 520K est chargé dans une partition de taille 750K, le résidu sera une nouvelle partition de taille 230K, à partir d'une nouvelle adresse.

Pour choisir la partition à allouer (affecter) à un processus, il existe différentes stratégies ou politiques d'allocation dont les trois décrites ci-dessous.

Politiques d'allocation

Politique « *First Fit* »: la *première* partition libre qui convient est allouée au processus, c'est-à-dire la première partition dont la taille est supérieure ou égale à la taille du processus.
Politique « *Best Fit* »: la partition dont le résidu mémoire est *minimal* est allouée au processus
Politique « *Worst Fit* »: la partition dont le résidu mémoire est *maximal* est allouée au processus

Exemple

On suppose les données suivantes :

5 partitions **libres** dans cet ordre avec les tailles : 750K, 480K, 1020K, 250K, 320K

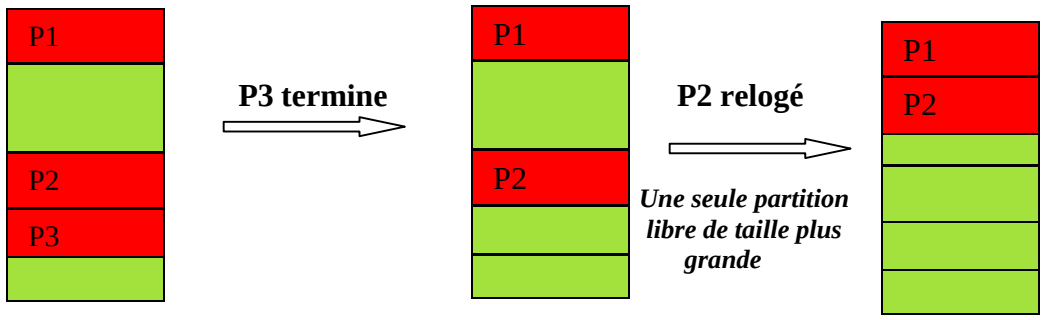
Et un processus P (programme) de taille **300K** en tête de file (la file des processus en attente).

Avec la politique « First Fit », on alloue à P, la partition de 750K (le résidu sera égal à 450K)
Avec la politique « Best Fit », on alloue à P, la partition de 320K (le résidu sera égal à 20K)
Avec la politique « Worst Fit », on alloue à P, la partition de 1020K (le résidu sera égal à 720K)

Remarques

- Si un processus *P* bloque l'allocation de la mémoire (aucune partition ne lui convient) et les processus suivants dans la file peuvent être logés dans des partitions libres convenables, on met le processus P en queue de file pour pouvoir charger un processus de plus petite taille.
- Lorsqu'un processus termine son exécution, il libère la partition qui lui était allouée, le gestionnaire de la mémoire pourrait réorganiser (reloger) les processus en cours de manière à créer d'autres partitions libres de plus grande taille.

Par exemple



Questions

Questions

Le programme doit permettre de répondre aux questions suivantes (à l'aide d'un menu)

- Créer l'état initial de la mémoire (ensemble de partitions de tailles variables pouvant être libres ou occupées)
- Afficher l'état de la mémoire (c-à-d la description des partitions) de manière textuelle, sous forme d'un tableau descriptif.
- Afficher l'état de la mémoire de manière graphique (les partitions de couleur rouge sont occupées et les partitions de couleur verte sont libres, voir schéma ci-dessus)
- Créer un ensemble de processus en attente dans la file (voir description d'un processus plus haut)
- Afficher l'état initial de la file des processus
- Choisir une politique d'allocation (parmi les 3 su-décrites)
- Charger un processus en mémoire (càd affecter une partition selon la politique choisie)
- Charger tous les processus se trouvant dans la file (des processus peuvent rester en attente s'il n'ya pas de partition convenable qui puisse les contenir)
- Supprimer un processus « terminé » de la mémoire (et charger aussitôt un processus enattente).
- Réorganiser (reloger) les processus en mémoire (pour créer des partitions libres de plus grande taille)

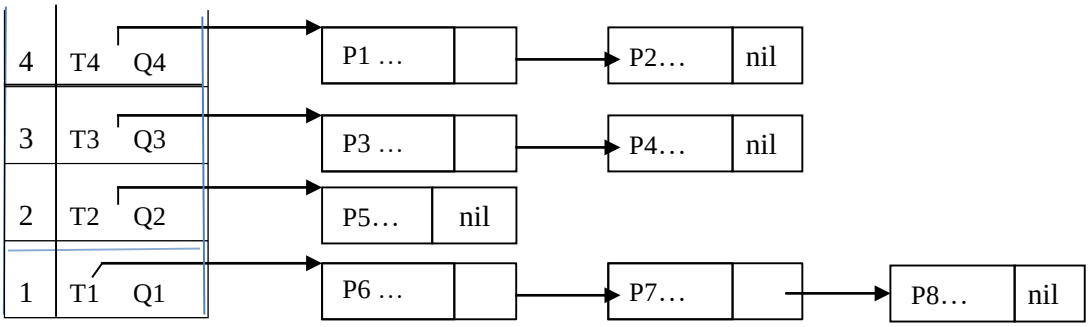
Après chaque opération, il faudra afficher :

- L'état de la mémoire (comportant de nouvelles partitions libres et occupées)
- L'état de la file des processus (en attente)
- L'état d'affectation de la mémoire aux différents processus

Important : *Pour des besoins de test, afin d'éviter la saisie fastidieuse des données (partitions et processus) au clavier, il faudra écrire une fonction de création pour chaque type de données (partitions et processus) qui permettra de remplir automatiquement des données définies dans la fonction ou alors récupérer les données à partir de fichiers de données déjà remplis.*

Partie II

Dans cette partie, on reprend le même contexte mais on suppose que **les processus sont dotés d'une priorité** (un attribut de type entier), les processus sont alors rangés dans plusieurs files (une file correspond à **une priorité**). Les « tête » et « queue » de files sont rangées dans une pile (voir figure). La file de plus haute priorité se trouve au sommet de la pile (sur la figure, on a 4 files avec priorités 1, 2, 3, 4 ; les processus P1 et P2 sont les plus prioritaires).



Pile

- Un processus de priorité ***i*** ne peut être chargé en mémoire que si tous les processus de priorité supérieure(i+1, i+2, …) sont déjà chargés
- Si un processus bloque l'allocation de la mémoire, on le déplace vers une file (en queue de file) de priorité plus faible (inférieure).

Questions

Reprendre les questions de la Partie I, en considérant trois niveaux de files (priorités 1, 2 et 3).

Indications

- L'interface principale du programme doit se présenter sous forme d'un menu à choix multiples pour répondre à l'une ou l'autre des requêtes (questions) de l'utilisateur.
- Considérer au préalable, un menu principal avec deux choix : *« Gestion sans Priorité »* et *« Gestion avec Priorité »*
- Le programme doit être écrit sous forme de fonctions.
- Les affichages des résultats à l'écran doivent être lisibles et bien faits.

Remarques importantes

Le code source doit être :

- **Bien commenté**, une ligne de commentaire est obligatoire avant chaque fonction, indiquant ce que fait la fonction plus d'autres commentaires jugés nécessaires.
- Les noms des étudiants doivent être indiqués au début du programme.

Rapport à remettre

Rédiger un rapport (environ 10 pages) comportant une description des différentes structuresde données utilisées et les différents algorithmes implémentés pour chacune des parties, plus un exemple de test de l'application (captures d'écran).

Pour la mise en forme

Page de garde

Une page de garde est nécessaire et doit être structurée comme suit :
- En haut, au centre : le nom de l'université, la faculté et le département (police Times NewRoman, 11) + le **Logo** de l'USTHB.
- En dessous (avec espacement)
  La mention « **Rapport de Mini projet de TP Algorithmique et Structures de données –Programmation C** », (Times New Roman, 14)
- Au milieu de la page, le titre du projet « **Manipulation de structures dynamiques,…** », Arial 16, gras
- En dessous à gauche : Réalisé par : Noms et prénoms des étudiants, Times New Roman 12
- A droite : Nom et prénom des enseignants : Responsable du module et enseignant de TP
- En bas de la page, au centre, Année : 2023/2024, Times New Roman, 11.

Les autres pages :
Il faut une introduction, Objectifs du TP,…etc.
Il faut une conclusion à la fin
Il faut des titres de sections pour séparer les différentes parties
Marges : 2,5cm de tous les côtés (gauche, droite, haut et bas)
Police de caractères (11 ou 12 Times New Roman)
Interligne (multiple de 1,3)
Titres de sections : Times New Roman (13 ; gras)

NB

- Le travail doit être fait par binôme et remis le **07 janvier 2024.**
- Le travail (code source wordpad et rapport en pdf) doit être envoyé à l'adresse recupspace@gmail.com,en précisant les **noms des étudiants** comme nom de fichier et dans l'objet du mail également.
- Préciser aussi le **groupe de TP du binôme** dans l'objet du mail
- Aucun retard ne sera toléré
- Les cas de copiage seront sévèrement sanctionnés