Université de Rouen
Licence informatique
Module : Système II

T.P. 8 : Gestion de la mémoire (Partie 2) - Swapping

- Les processus clients devront comporter les appels décrits dans le T.P. précedent (debut(),mmalloc(),mrealloc(), ..., fin()).

  prg.c

  ```
  debut(prg.c)
  ad1=mmalloc(231)
  attend(2)
  ad2=mmalloc(433)
  putmem(ad2,struct toto,sizeof(struct toto))
  attend(3)
  realloc(ad2,547)
  fin()
  ```

  La demande de reservation de mémoire d'un processus client ne devra plus entrainer d'erreur si le gestionnaire ne trouve plus d'espace libre disponible. Dans ce cas, le gestionnaire devra choisir un processus à éliminer de la mémoire (en le recopiant sur le disque). Dans un premier temps , les processus seront intégralement swappés vers le disque ou remontés du disque. On propose comme premier algorithme de swap d'éliminer le premier processus libérant suffisamment de mémoire pour satisfaire la nouvelle demande.
- Le séquencement et la priorité de chaque processus seront assurés par l'ordonnanceur Unix qui répartira les demandes dans la file de messages. On pourra ainsi pour certains algorithmes définir une priorité à chaque processus.
- Pour gérer le swap, vous devrez créer un fichier qui sera consideré comme périphérique de swap. On créera une liste spécifique des processus swapper contenant les informations suivantes :
  - pid
  - numéro de bloc (sur le disque)
  - nombre de blocs

  On devra mettre un indicateur dans la liste originale pour dire que le processus est en zone de swap (numéro de bloc à -1). On pourra prendre une taille de bloc disque égale à la taille de bloc mémoire pour simplifier le travail.
  L'interface devra implanter les fonctions suivantes :

  swapout : recopie les données du processus vers le disque et libération des blocs.

  swapin : allocation des blocs nécessaires à la copie du disque vers la mémoire.

  L'interface devra gérer le périphérique de swap avec un algorithme de première zone libre. Le choix du processus à sortir de la mémoire se fera grâce à l'algorithme de l'horloge. On conservera une liste circulaire des processus en mémoire. Chaque processus aura un bit d'utilisation. Le Processus pointé pourra être swappé si son bit d'utilisation est à 0 sinon on passe ce bit d'utilisation à 0 et on avance sur la liste circulaire. A chaque utilisation d'un processus, son bit est passé à 1. Il faudra bien sur prendre en compte la taille de la mémoire demandé (swappé le moins de processus possible por répondre à une requête).