Opsys2 – TP 2 : IPC / Inter-Process Communication

Objectif

L'objectif de ce TP est de développer un système avec plusieurs processus qui travaillent en parallèle et collaborent au moyen des outils d'IPC vus en cours. Pour ceci vous allez développer une simulation de système de pagination de la mémoire.

Description

Résumé

Plusieurs processus de tris (**Sorter**) vont trier des tableaux d'entiers. Pour stocker ces tableaux, ils auront besoin de mémoire. L'accès à la mémoire « vive » est effectué par des requêtes au processus **MainMemory**, qui gère les pages en mémoires. La mémoire virtuelle est gérée par le processus **VirtualMemory**, qui peut contenir un grand nombre de pages. Le processus **MemorySimulation** initial démarre tous les autres processus.

Spécification des programmes / processus :

1) **MemorySimulation :** processus initial qui démarre tous les autres processus. Peut avoir les paramètres suivants :
   - Nombre de **Sorter**
   - -p X : où X représente la taille d'une page
   - –m X: où X représente le nombre de pages en mémoire vive
   - –s X: où X représente le nombre de caractères dans les tableaux des **Sorters**
2) **Sorter :** crée un tableau d'entiers de façon aléatoire, puis le trie au moyen de l'algorithme quicksort. L'accès des **Sorter** à la mémoire « vive » se fait au moyen de requêtes au processus **MainMemory.** Les **Sorter** ne gèrent pas d'allocation de mémoire, car celle-ci est déjà gérée par défaut pas les autres processus.
3) **MainMemory :** donne accès à la mémoire **« vive »** pour les **Sorter**. Le processus **MainMemory** partage un segment de mémoire partagée avec le processus **VirtualMemory** dans lequel se trouve la mémoire « vive ». Cette mémoire est implémentée au moyen d'un grand tableau d'entier (int). La taille de la mémoire et des pages est données en nombre d'entiers.
4) **VirtualMemory :** mémoire virtuelle qui gère la mémoire et la pagination. Tous les tableaux des **Sorters** doivent être alloués et instanciés au moyen du **VirtualMemory.** Lorsque **MainMemory** a besoin d'accéder à une page qui n'est pas dans le segment de mémoire partagée, il demande au **VirtualMemory** de mettre la page demandée dans la mémoire partagée, en utilisant l'algorithme de remplacements de page vu en cours : *Clock Hand Algorithm.*

Sources

Les fichiers MemorySimulation.c et Sorter.c contiennent un début d'implémentation des processus **Sorter** et **MemorySimulation**. Ces fichiers doivent être complétés, et les fichiers MainMemory.c et VirtualMemory.c doivent être implémentés comme spécifiés ci-dessus.

Informations supplémentaires:

1. Ce TP doit être effectué par groupes de 2 personnes.
2. Il peut y avoir des différences d'implémentation pour certains appels système d'une plateforme à l'autre. Donc en cas de doute, n'hésitez pas à relire les *man pages* relatives aux commandes utilisées !
3. Vous devez fournir un makefile pour compiler l'ensemble de vos sources.
4. Ce makefile contient une cible `clean` pour effacer tous les fichiers objets, et une `clean_all` qui, en plus d'effectuer la même chose que clean, efface également tous les exécutables.
5. Ce makefile contient une cible `run` qui lance un exemple d'application de vos processus.

1. Tous les fichiers nécessaires sont à rendre dans une archive zip, nommée avec votre nom : `memsym_LastName.zip`. Tous les fichiers de l'archive doivent s'extraire dans le même répertoire (pas de sous-répertoire). **C**e fichier doit être rendu sur le site Moodle du cours.
2. Assurez-vous que votre code compile et fonctionne sur la machine de référence suivante : c0007debian.tic.eia-fr.ch.
3. **Le délai pour rendre ce travail est le vendredi 23 janvier 2015, 23:00**.