Couplage mémoire

Thème

- Couplage mémoire

Ressources

Pour ce TP, vous pourrez vous appuyer sur

- Le polycopié du TD sur la mémoire virtuelle, qui fournit une référence a priori suffisante sur la sémantique et la syntaxe d'appel des différentes primitives utilisées ici.
- Les pages du manuel en ligne (commande `man`), et plus particulièrement les sections 2 et 3.
- Les squelettes de code fournis pour les exercices 1, 2 et 4.

Déroulement de la séance :
Dans l'ensemble, les exercices proposés demandent très peu de programmation, mais font appel à des primitives assez complexes, dont les paramètres d'appel doivent être bien compris.
Les exercices 1, 3, 4, 5 correspondent aux exercices présentés en TD. L'exercice 2 est une variante complémentaire de l'exercice 1.

- Les exercices 1, 2 et 3 doivent être traités par tous. La durée de la séance de TP devrait permettre de traiter ces exercices. Un squelette de code est fourni afin de faciliter la réalisation des exercices 1 et 2.
- L'exercice 4 peut être traité simplement, après les exercices 1 et 2. Le schéma utilisé se retrouvera au cœur de la réalisation de la version tableau blanc du minichat. Cet exercice devrait pouvoir être traité lors de la séance de TP par la plupart.
- L'exercice 5 est un exercice complémentaire, non essentiel.

Opérations essentielles

`mmap(-)` définit un segment de mémoire virtuelle, avec des droits d'accès donnés, partagé ou privé, couplé à un fichier ou non ; retourne l'adresse de base de ce segment.

`munmap(-)` découple tout ou partie d'un segment préalablement couplé.

`mprotect(-)` (re)définit les droits d'accès à un segment préalablement couplé.

Exercices

1. *(Fonctionnement d'un segment partagé, couplage segment/fichier)*
   Ecrire un programme qui exécute l'algorithme suivant :
   - Créer un fichier contenant deux pages de caractères 'a' ;
   - Ouvrir de ce fichier en lecture/écriture ;
   - Coupler un segment de taille 2 pages à ce fichier en mode partageable et lecture/écriture ;
   - Créer un processus fils ;
   - Pour le processus fils, attendre 2 secondes et lister les 10 premiers octets de chaque page du segment, puis remplir la première page de caractères 'c' et terminer.
   - Pour le processus père, remplir la deuxième page du segment de caractères 'b' et attendre la fin du fils. Lister alors les 10 premiers octets de la première page et terminer.

   (a) Quel va (vraisemblablement) être l'affichage observé ?

   (b) Quel va être le contenu du fichier ?

2. *(Segment partagé, puis couplé en mode privé : les écritures en mode partagé ne sont dès lors plus vues)* Ecrire un programme qui exécute l'algorithme suivant :
   - Créer un fichier contenant trois pages de caractères 'a' ;
   - Ouvrir ce fichier en lecture/écriture ;
   - Coupler un segment de taille 3 pages à ce fichier, en mode partageable et lecture/écriture ;
   - Créer un processus fils ;
   - Pour le processus fils, (re)coupler en mode privé, lister les 10 premiers caractères de la page 1, attendre 4 secondes ; lister les 10 premiers caractères de chaque page du segment puis remplir la deuxième page de caractères 'd' ; attendre 8 secondes, lister les 10 premiers octets de chaque page et terminer.
   - Pour le processus père, attendre 1 seconde, remplir les pages 1 et 2 du segment de caractères 'b', attendre 6 secondes, remplir la page 2 de caractères 'c', et terminer.

   Quel va (vraisemblablement) être l'affichage observé ?
3. Ecrire un programme qui couple, en mode anonyme, une zone d'adressage de la taille d'une page en interdisant tout accès.
   - Provoquer l'exception de violation mémoire en tentant un accès en écriture dans cette zone.
   - Par couplage d'un traitant au signal SIGSEGV, modifier la protection en utilisant la primitive mprotect de façon à permettre l'écriture.
4. Ecrire une implémentation de la commande `cp f1 f2` basée sur le couplage mémoire.
   Compléments :
   - la fonction `memcpy` de la librairie C standard effectue la copie de zones mémoire.
   - la fonction `fstat` permet d'obtenir la taille d'un fichier donné.
5. On considère le programme suivant :

```
#include <stdio.h>
#include <setjmp.h>
#include <sys/mman.h>
#include <sys/signal.h>

char    *base;      // adresse de base de page
long    pagesize;   // taille d'une page
jmp_buf env;        // zone de sauvegarde de point de reprise

void hndler (int quelsignal) { ... }   // traitant du signal SIGSEGV

int main ( int argc, char *argv[]) {   // programme principal
  mon_action.sa_handler = traitant;
  sigemptyset(&mon_action.sa_mask);
  mon_action.sa_flags = 0;

  pagesize = sysconf(_SC_PAGESIZE);
  sigaction(SIGSEGV, &mon_action, NULL);
  setjmp(env);
  printf("base[0] = %c\n",base[0]);
}
```

   Que devrait faire le traitant du signal pour que finalement l'instruction printf soit exécutable ?