Communications Interprocessus par mémoire partagée

ENSIMAG 2A

1 Introduction

Le but de ce TP est d'implanter un espace d'échange entre plusieurs processus. Il s'agit d'un « tableau noir » sur lequel tous les processus vont dessiner simultanément. Dans un premier temps, chacun des processus va afficher la surface, ainsi s'il y a n processus, il y aura 1 surface répliqué n fois, donc le même dessin sur toutes les surfaces.

La technique utilisée sera la mise en place d'un *espace de mémoire partagée* commun entre les processus en utilisant les fonctions POSIX de partage d'une zone mémoire.

2 Shmem

Il existe de très nombreuses possibilités pour faire communiquer plusieurs processus, plus ou moins difficiles. Pour ce TP nous allons utiliser les fonctions suivantes : `shm_open`, `ftruncate`, `mmap`, `munmap`, `close`, `shm_unlink`. Pour commencer, `man shm_overview`.

2.1 Utilisation de la mémoire partagée

Ce qui doit changer par rapport au code d'exemple fournit est la façon de créer le tampon mémoire (le seul `malloc`).

```
...
/* Le tampon a changer */
void *tampon = malloc(TAILLEX*TAILLEY*4);
...
```

Il faudra réaliser les points suivants :

1. Créer un pseudo-fichier qui sera partagé entre les processus (`shm_open`). Ce fichier est au départ de taille 0 (RTFM).
2. Donner une taille appropriée à ce pseudo-fichier (`ftruncate`)

3. Coupler le contenu de ce pseudo-fichier à la mémoire (`mmap`)

Le reste du code devrait rester plus ou moins inchangé sauf la libération et la terminaison qui vont devoir libérer et découpler ce qui a été mis en place.

2.2 Intéraction avec le shell

Il existe deux commandes shell particulièrement utiles pour voir l'état des segments de mémoire partagées et effacer les segments qui n'auraient pas été libérés convenablement. Il s'agit de `ipcs` et `icprm`.

Petit exercice de shell : que fait la commande suivante ?

```
for shm in $(ipcs -m | grep 0x | awk '{print␣$2}') ; do
    ipcrm -m $shm
done
```

Astuce : elle pourrait vous être utile en cours de mise au point de votre programme, ...

2.3 Pour aller plus loin : surface unique et/ou System V

Deux extensions possibles :

— Proposez une solution pour que seul le premier processus lancé crée la fenêtre et que les autres utilisent cette fenêtre au lieu d'en créer une eux même. On pourra, pour distinguer le processus qui crée la fenêtre des autres, passer par exemple un argument queconque sur la ligne de commande ;

— Vous pouvez adapter le programme pour utiliser l'interface System V qui est encore très utilisée, même si non POSIX. Il faudra alors utiliser les fonctions `shmget`, `shmat`, `shmdt`.