Indications de TP 1 **S**R - Programmation système L3 Info / UFR ST/UBFC

Mémoire partagée (1)

V. Felea & E. Merlet

Exo1 : Communication fils → parent

Pseudo-code algorithmique : tr.24 Cours Programmation système (progSyst.pdf)

Définition du segment

- contenu : une chaîne de caractère (donc typage `char*`)
- taille : autant de caractères que la chaîne en contiendra (+1 pour '\0' - le marqueur de fin). La taille est à surdimensionner (utiliser une constante macrodéfinie).
  Remarque. Le processus parent définit la taille du segment partagé lors de la création, et le processus fils y écrit. Si le processus fils écrit plus de caractères que prévu dans la mémoire, il peut y avoir SEGFAULT. Cela peut arriver seulement si la chaîne écrite est bien supérieure en taille à la taille de création, car la bibliothèque IPC crée un segment surdimensionné (voir doc. fct. `shmget` - Shared memory limits). Confronter cette remarque avec la sortie de `ipcs -m / colonne octets` pendant que le segment est toujours actif.
  Attention à la taille. sizeof(char*) est la taille d'un pointeur, c'est-à-dire d'une adresse (toujours 8 octets sur une machine 64 bits, peu importe le type pointé).
  sizeof(char*) n'est donc pas égal à la taille de l'espace référencé par l'adresse.

Partage d'un segment entre parent et fils Le processus parent est censé créer un segment avant la création du processus fils. Pour le reste des opérations, il y a deux solutions :

- soit le processus parent crée le processus fils avant l'attachement - dans ce cas les deux processus parent et fils doivent attacher, respectivement détacher le segment. Pour l'attacher, l'id du segment sera utilisé, disponible dans le fils par la copie de l'espace mémoire.
- soit le processus parent crée le processus fils après l'attachement - dans ce cas le processus fils n'a plus besoin d'attacher le segment, car il peut directement utiliser la variable de type adresse (obtenue par `shmat`) qui pointe vers le segment partagé. (man shmat : After a fork(2), the child inherits the attached shared memory segments.)
  Attention : dans ce cas, les 2 processus doivent détacher le segment.

Le processus parent - créateur du segment - doit finir par le marquer pour la suppression.

Q1 Il faut "synchroniser" l'écriture avec la lecture. En absence d'outils explicites de synchronisation, s'appuyer sur les fonctions de base de coordination entre les processus parent et fils, notamment l'attente de terminaison du processus fils par le processus parent.

Q2 Identifier le paramétrage de la commande `ipcs` permettant d'inspecter, entre autres, les segments de mémoire partagée et ajuster, le cas échéant le programme (pour afficher, par exemple, l'id du segment).

Q3/Q4 Comprendre la différence entre détacher, marquer pour suppression et supprimer un segment. Il y a des fonctions pour les deux premières actions. La dernière est à la charge du système d'exploitation - voir doc. fct. `shmdt`.

La commande système générique pour visualiser les ressources IPC est `ipcs`. Si le programme informe de l'id du segment, alors la commande peut filtrer, par options spécifiques, le segment concerné par l'inspection - voir doc. cmd. `ipcs`. Parfois une expression/des instructions plus complexes (type programmation shell) peuvent être nécessaires (voir un petit rappel de la programmation shell sous Moodle).

Important. La commande `ipcs` est très utile dans la phase du développement du code. Selon les paramètres de création d'un segment, un nouveau segment peut être créé ou un existant peut être retourné. Effacer un segment par la commande système `ipcrm` peut aider à repartir d'un environnement vierge quant à l'applicatif courant.

Lecture/écriture dans le segment Manipulant ici une chaîne de caractère dans le segment de mémoire partagée, l'affectation ne peut pas convenir pour l'écriture.

`*p = ''Bonjour''`
où `p` est le pointeur vers l'espace mémoire partagée,
n'est pas à faire - pourquoi ?

La fonction `strcpy` est adéquate.

Exo2 : Nombre aléatoire

Le principe de communication reste identique à l'exercice précédent.

Le typage de la mémoire partagée doit être adapté, ce qui impacte la création du segment et les lectures/écritures le concernant. Ici la mémoire partagée contient un entier et non plus une chaîne de caractères.

Remarques générales

- éviter le codage en dur (ex. sizeof(char)*200 : à remplacer par sizeof(char)*MSG_SIZE où MSG_SIZE est une constante - #define).
- afficher une chaîne : printf avec format %s, et pas de boucle pour afficher caractère par caractère.
  Attention, avec la spécification de format %s, il faut transmettre à printf() l'adresse du 1er caractère d'une chaîne de caractères, donc terminée par un '\0'
- initialiser une chaîne : strcpy/strcat/snprintf, et pas mess[i]='c'
- vérifier les retours d'appels de fonctions
- éviter d'imbriquer les appels de fonctions dont le retour doit être vérifié car le décodage d'erreur est plus difficile - ex. shmat(atoi(argv[0]),...)

Conventions codage :

- ne pas oublier les commentaires
- à marquer également, en début du fichier, l'objectif du programme et l'auteur