Programmation Système

Travaux Pratiques (6), Licence 2 Informatique

Les IPC SYSTEM V - Les segments de mémoire partagée

Les exercices suivants ont pour but de vous familiariser avec les IPC (Inter-Process Communication) SYSTEM V. Cette partie se consacre aux segments de mémoire partagée.
Il vous est recommandé de consulter les pages man des primitives relatives aux IPC pour de plus amples informations sur leur syntaxe, leur sémantique et les éventuelles options qu'elles offrent.

Les instructions des exercices se repèrent par des icônes, qui sont les suivantes :

	Information	Information concernant l'usage ou le rôle d'une commande, par exemple. Dans certains cas, il s'agit d'une information sur ce que vous êtes en train de faire ou sur ce qui se passe.
	Exemple	Exemple d'utilisation.
	Contrôle	Vérifier le résultat d'une (ou plusieurs) action(s).
	Action	Effectuer la ou les action(s) décrite(s).
	Question	Questions auxquelles vous devez répondre.

De plus, un `texte en police courier` correspond soit à une sortie écran soit à des noms spécifiques (menus, fenêtre, icône, processus, commandes…).
Un **texte en police times gras** correspond à ce que l'utilisateur doit introduire comme valeur de paramètre, ou encore, est utilisé pour attirer l'attention de l'utilisateur.

Création d'un segment de mémoire partagée

La mémoire partagée constitue le moyen le plus rapide de communication entre utilisateurs. Les données à transmettre sont rangées dans une zone de mémoire où deux (ou plus) processus ont accès. Cette zone de mémoire partagée est appelée *segment*.

Écrire un programme qui crée un segment de mémoire partagée.

Avant la terminaison du programme, vérifier que le segment a bien été créé. Utiliser pour cela la primitive `ipcs` alors que le programme est suspendu (`sleep`).

Le segment créé existe-t-il encore après la terminaison du programme ? Qu'en déduisez-vous ?

Mohamed Taghelit , taghelit@univ-tours.fr

Recherche de l'identifiant d'un segment de mémoire partagée et écriture

Modifier le programme précédent de façon qu'il recherche l'identifiant du segment précédemment créé et qu'il y écrive une donnée.

S'assurer qu'un nouveau segment n'a pas été créé.

Recherche de l'identifiant d'un segment de mémoire partagée et lecture

Écrire un nouveau programme qui recherche l'identifiant du segment précédemment créé et qui y lise la donnée contenue.

Est-ce la même donnée que celle écrite par le précédent programme ? Qu'en déduisez-vous ?

Opérations de contrôle sur les segments de mémoire partagée

Extraction et modification des caractéristiques d'un segment de mémoire partagée

La primitive `int shmctl(int shm_id, int cmd, shmid_ds *buf);` permet l'extraction des caractéristiques d'un segment de mémoire partagée.

Écrire un programme qui affiches certaines caractéristiques (créateur, propriétaire, …) d'un segment existant.

La primitive précédente, permet également de modifier certaines caractéristiques d'un segment existant.

Modifier le programme précédent de façon à ce qu'il modifie le propriétaire d'un segment déjà existant.

Suppression d'un segment de mémoire partagée

La primitive `int shmctl(int shm_id, int cmd, shmid_ds *buf);` permet enfin la suppression d'un segment de mémoire partagée existant.

Modifier le programme précédent de façon qu'il supprime le segment de mémoire partagée une fois terminé.

Communication et synchronisation entre processus

Programmer l'exemple de communication entre deux processus, un écrivain et un lecteur, suivant :

L'écrivain a le comportement suivant:

- il appelle la primitive `ftok`, pour composer une clé unique, en spécifiant un `path`="**mempart**" (fichier existant et accessible) et un identificateur `code`='**M**',
- il crée un segment de mémoire partagée avec la clé rendue par `ftok`,
- il attache ce segment à son espace de travail,
- il écrit une suite de nombres entiers dans ce segment. Pour la synchronisation, entre l'écrivain et le lecteur, on imposera que le lecteur ne pourra accéder au segment que lorsque le segment contiendra, en début, un entier non nul.
- il met un entier non nul en début du segment, et détache ce dernier de son espace de travail,
- il se met en attente d'arrivée du signal `SIGUSR1`, qui traduira que le lecteur a fini de lire les données contenues dans le segment,
- sur réception du signal `SIGUSR1`, il supprime le segment.

Le lecteur a le comportement suivant:

- il appelle la primitive `ftok`, pour composer une clé unique, en spécifiant un `path`="**mempart**" (fichier existant et accessible) et un identificateur `code`='**M**',
- il recherche l'identifiant du segment de mémoire partagée associé à la clé rendue par `ftok`,
- il attache ce segment à son espace de travail,
- il attend que le début du segment contienne un entier non nul,
- il lit (imprime) les données contenues dans le segment,
- il recherche la structure associée au segment, et détache ce dernier de son espace de travail,
- il émet le signal `SIGUSR1` à l'écrivain (créateur du segment).

Commandes `shell` relatives aux IPC SYSTEM V

La commande `ipcs` permet d'afficher les ressources IPC existantes.

Utiliser la commande `ipcs` pour prendre connaissances des différents segments de mémoire partagée existants. Essayer quelques options permises.

La commande `ipcrm` permet de supprimer les ressources IPC existantes.

Utiliser la commande `ipcrm` pour supprimer des segments de mémoire partagée existants. Essayer quelques options permises.