IPC

Questions principales

Expliquez le mécanisme du producteur-consommateur. Détaillez-en le principe, le code, les appels système liés.

Expliquez la réalisation d'une section critique via "variable partagée", "blocage des interruptions" et via "sémaphores de Dijkstra". Détaillez les appels système Down et Up. Comparrez ces trois approches.

Expliquez la réalisation d'une section critique via "BTS", "alternance" et via "sémaphores de Deijkstra". Détaillez les appels système Down et up. Comparez ces trois approches.

semget(), semctl(), semop() : Quelle est l'utilité ? Quels sont les arguments ? Quelle est la valeur de retour ? Etablissez le lien entre ces appels système et ceux vus en théorie (up() et down())

ls | wc -l : Comment l'OS parvient à exécuter cette ligne de commande ? Expliquez en détail le mécanisme sous-jacent. De façon très détaillée, expliquez comment il parvient à synchroniser wc et ls afin que wc ait toujours des données à lire.

Lors de l'éxecution de la commande, le shell va trouver le token "|" signifiant un "pipe" entre deux commandes.

Au départ :

- ls écrit son résultat dans la sortie standard (1, stdout).
- wc lit dans l'entrée standard (0, stdin).

Il faut donc créé un pipe à l'aide de la fonction du même nom qui prend un tableau à deux entrée d'entier : `C int p[2]; pipe(p);` Lorsque le processus père va créer ses deux fils par le biais du fork(), chacun va hériter de ce tableau de handle. Le fork s'occupant du ls modifier sa sortie standard pour qu'elle écrive dans le pipe

```
/* ls */
close(p[0]); // Fermeture de la lecture
dup2(p[1], 1);
```

```
/* wc -l */
close(p[1]); // Fermeture de l'écriture
dup2(p[0], 0); // Entrée standard = la lecture dans le pipe
```

Après cela, par le biais du principe de "producteur consommateur", le 2ème fils peut lire dans le pipe jusqu'à obtention du E.O.F. que le ls aura envoyé quand il n'aura plus rien à écrire.

Ainsi, executer les logiciels via la commande exec ne remplacera pas la table des handles mais seulement text data et stack.

```
execl("/usr/bin/ls", "ls", NULL);
```

```
execl("/usr/bin/wc", "wc", "-l", NULL);
```

Quels appels système permettent de gérer les signaux ? Détaillez-en les paramètres et le fonctionnement. Quelles sont les limites et les défauts de ces signaux ? Quel est le rôle de la table des interruptions, de la table des processus, de l'ordonnanceur dans ces cas ?

socket(), bind(), listen(), accept(), connect() : Quelle est l'utilité ? Quels sont les arguments ? Quelle est la valeur de retour ?

Utilisation de l'appel système pipe et situations d'interblocages : expliquez en vous basant sur des exemples de code comment une telle situation peut être obtenue. Détaillez et faites le lien avec les appels système Up et Down.

Questions secondaires

(socket) Quelle est l'utilité de la fonction htons() ? Comment faire parvenir un message à un processus qui s'exécute sur un autre ordinateur ?

(socket) Comment écrire une application client / serveur où plusieurs clients peuvent être connectés au serveur en même temps ?