Consolidation de Janvier 2017 (1h)

RS : Réseaux et Systèmes

Deuxième année

★ **Questions de cours.**(6pts)

▷ **Question 1:** Expliquez précisément ce que fait l'appel système `fork`.

▷ **Question 2:** Expliquez précisément ce que fait l'appel système `exec`.

▷ **Question 3:** Donnez un problème fondamental qui rend les signaux peu pratiques pour échanger des informations entre processus.

▷ **Question 4:** Citez quatre mécanismes permettant d'échanger des informations entre processus.

▷ **Question 5:** Qu'est-ce qu'un interblocage ?

▷ **Question 6:** Comment prévenir les interblocages ? Donnez et expliquez trois méthodes.

★ **Exercice 1: Utilisation de `fork`, `exec`, `wait`, `waitpid`, `pipe`, `dup`, `dup2` (6pts)**

Donner le code C (sans utiliser la fonction `system`) correspondant à ce que fait un *shell* lorsqu'on tape les lignes de commandes suivantes. Afin d'obtenir un code lisible et court, vous êtes dispensés des tests d'erreur des appels systèmes.

▷ **Question 1:** `prog1 && prog2`

(Il faut donc : lancer un programme *prog1*, attendre sa fin, récupérer son code de retour, et si ce code de retour est égal à 0, lancer un programme *prog2* et attendre sa fin)

▷ **Question 2:** `prog1 > fichier`

(Rappel : ">" redirige la sortie du processus vers un fichier)

Quelques fonctions utiles

```
pid_t fork(void);
int execlp(const char *file, const char *arg, ...);
pid_t wait(int *status);
pid_t waitpid(pid_t pid, int *status, int options);
WIFEXITED(status) -- returns true if the child terminated normally
WEXITSTATUS(status) -- returns the exit status of the child
int pipe(int pipefd[2]);
int dup(int oldfd);
int dup2(int oldfd, int newfd);
```

★ **Exercice 2: Clonage de processus (3pts)**

On considère le programme suivant :

```
int main() {
    int status;
    pid_t pid;

    fprintf(stderr,"Hello\n");
    pid = fork();
    fprintf(stderr,"%d\n", !pid);
    if (pid != 0) {
        if (waitpid(-1, &status, 0) > 0) {
            if (WIFEXITED(status) != 0)
                fprintf(stderr,"%d\n", WEXITSTATUS(status));
        }
    }
    fprintf(stderr,"Bye\n");
    exit(2);
}
```

Combien de lignes ce programme imprime-t-il ? Discuter les ordres possibles dans lesquels ces lignes sont imprimées.

★ Exercice 3: Savoir utiliser les sémaphores et reconnaître les schémas de synchronisation classiques (5pts)

Suite à une expérimentation malheureuse, un début d'incendie s'est déclaré au club robotique. Immédiatement, tous les étudiants présents sur place et au foyer se précipitent à la cafet pour remplir des carafes d'eau avant de les jeter sur les flammes. Un seul robinet, des couloirs exigus et des dizaines d'étudiants courant en tout sens pour porter de l'eau vers les flammes : le chaos est indescriptible.

▷ **Question 1:** Proposez une solution à base de sémaphore(s) pour d'éviter que les étudiants se bousculent devant le point d'eau : un seul étudiant à la fois doit accéder au robinet. Décrivez également la solution en pseudo-code, comme fait en TD.

▷ **Question 2:** De quel schéma de synchronisation classique la solution proposée à la question précédente se rapproche-t-elle ? Pourquoi ?

▷ **Question 3:** Les porteurs d'eau perdent un temps précieux à contourner le bar et attendre leur tour au robinet. Proposez une solution à base de sémaphore(s) où l'un d'entre eux seulement est en charge de remplir les carafes pour les autres, qui courent ensuite en jeter le contenu dans les flammes quand elles sont pleines. De quel schéma de synchronisation classique cela se rapproche-t-il ? Pourquoi ? Décrivez également la solution en pseudo-code, comme fait en TD.