Programmation système

Steve Alabi - Taha Bendjeddou - Younes Benyamna - Malek Zemni

Feuille d'exercices

22/11/2018

Exercice 1

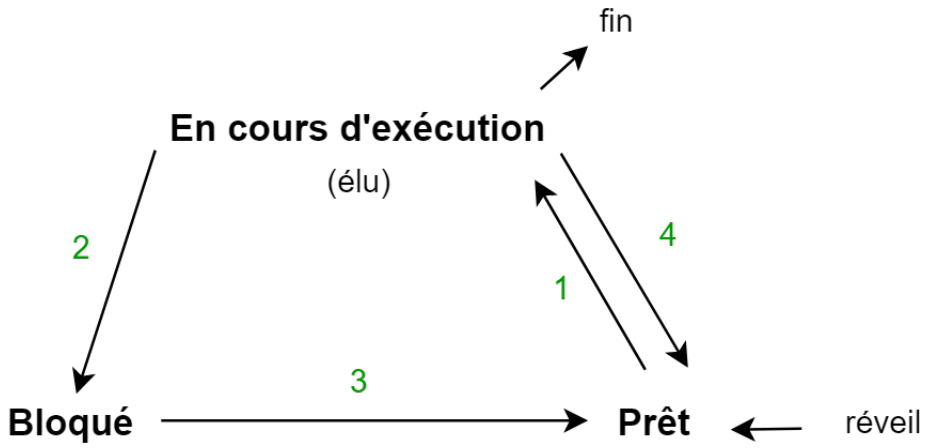
(définitions générales, gestion de flux et entrées / sorties)

1/ Définir précisément un appel-système. Comment s'exécutent-ils ?
2/ En quoi la redirection de flux peut-elle servir dans la gestion d'erreurs ?
3/ Écrire une fonction `myGetChar` analogue à la fonction `fgetc` (lecture d'un seul caractère à partir d'un fichier) en utilisant uniquement des appels-systèmes.
4.1/ Définir un buffer.
4.2/ Expliquer comment construire une version bufferisée de la fonction `myGetChar`.
4.3/ Écrire la fonction `myGetChar` bufferisée.

Exercice 2

(processus)

1/ Quelle est la particularité des processus sous les systèmes UNIX ?
2/ Donner une légende correspondant aux numéros (1, 2, 3 et 4) de la figure suivante :



3/ Définir un processus zombie. Quelle est l'utilité de la fonction `wait` ?

4/ Lors de la création d'un nouveau processus, quelles ressources sont dupliquées et lesquelles sont partagées ?

5/ Écrire un programme qui crée un processus et qui affiche un message différents pour chaque processus (en incluant la gestion d'erreurs).

Exercice 3

(threads)

1/ Quelle est la différence entre un processus et un thread ? Quel est l'avantage des threads ?

2/ Écrire un programme ayant le comportement suivant :

- des threads sont créés (leur nombre étant passé en paramètre lors du lancement du programme)
- chaque thread affiche un message (par exemple "hello world !") et son PID
- le thread principal attend la terminaison des différents threads créés

3/ Définir un mutex et donner son rôle.

4/ Écrire un programme qui crée deux threads : un qui incrémente une variable compteur par un nombre tiré au hasard entre 0 et 10, et l'autre qui affiche un message lorsque la variable compteur dépasse 20 (utiliser mutex et conditions).