IS 3 - T.P. Système
Gestion de processus

Comme précédemment on va utiliser les terminaux en mode texte, via la combinaison de touches Ctrl + Alt + Fn , où $1 <= n < 10$. Notez que vous pouvez vous connecter simultanément dans plusieurs terminaux et basculer de l'un à l'autre avec cette même combinaison de touches.

1 Commandes de gestion de processus

Question préliminaire : écrire un programme C, nommé `affiche.c`, constitué d'une boucle infinie qui affiche un texte à l'écran, puis fait une pause (pour la pause, voir `man 3 sleep`).

1. Lancez le programme `affiche` dans un premier terminal.
2. Connectez vous dans un autre terminal et entrez une commande pour afficher tous les processus qui vous appartiennent.
3. Terminez `affiche` à l'aide de la commande `kill`. Affichez de nouveau la liste de vos processus pour vérifier que le processus est bien terminé.
4. Relancez le programme `affiche` dans votre premier terminal.
5. Suspendez, sans le terminer, le processus ainsi créé tout en restant dans ce terminal.
6. Reprenez l'exécution du processus ainsi suspendu, mais en arrière-plan.
7. Lancez le programme `affiche` en arrière-plan dans le premier terminal.
8. Essayez de tuer ou de suspendre ce processus à l'aide de Ctrl + C .
9. Terminez le processus à l'aide de la commande `kill`.

2 Ordonnancement

2.1 Taux d'utilisation du processeur

1. Écrivez un programme `calcule.c` qui boucle indéfiniment sur l'incrémentation d'une variable (sans faire d'affichage). Compilez, exécutez et observez l'occupation du processeur (`top`).
2. Écrivez un programme `calcule-affiche.c` qui boucle indéfiniment et alterne entre afficher un caractère (pas de retour à la ligne) et incrémenter une variable. Compilez, exécutez et observez l'occupation du processeur.
3. Écrivez un programme `calcule-lit.c` qui boucle indéfiniment et alterne entre lire un entier et incrémenter une variable. Compilez, exécutez et observez l'occupation du processeur.
4. Expliquez les différences observées entre l'exécution des trois programmes.

2.2 Exécution concurrente

1. Lancez deux exécutions concurrentes de `calcule.c` et observez l'occupation. Rajoutez plusieurs exécutions, jusqu'à observer le partage d'un cœur entre 2 processus. En conclure de le nombre de cœurs de votre ordinateur.
2. Ajoutez alors une nouvelle exécution à l'aide de la commande `nice` et observez l'occupation. Quelle est la priorité la plus forte : 0 ou 10 (attention à ne pas confondre PRiorité et NIceness) ?
3. Tentez de lancer votre programme avec une priorité plus forte. Expliquez ce résultat.

2.3 Répertoire `/proc`

A chaque processus correspond un répertoire dans `/proc` ayant pour nom le PID du processus et contenant divers fichiers stockant les informations utilisées par le système afin de gérer les processus. Reportez vous au `man` de `proc` pour plus d'informations.

1. Lancez le programme `calcule` réalisé précédemment.
2. Retrouvez dans le répertoire `/proc` le statut, le nombre de changements de contexte de ce processus (*ctxt switches*, fichier `status`).
3. Lancez, sans arrêter le programme `calcule`, un éditeur de texte.
4. Comparez l'évolution au fil du temps du nombre de changements de contexte de `calcule` et de votre éditeur de texte. Observez en particulier l'impact de vos interactions avec l'éditeur sur les changements de contexte, et expliquez la différence entre `voluntary` et `nonvoluntary`. Pour rappel, les changements de contexte sont liés aux changements d'état des processus (principalement la préemption, et la suspension en attente de ressources).

2.4 Ordonnanceur Linux

En vous basant sur `man 7 sched`, déterminez à quels algorithmes d'ordonnancement vus en cours correspondent les ordonnanceurs Linux `SCHED_FIFO`, `SCHED_RR` et `SCHED_OTHER`.

3 Bonus

3.1 Création de processus en C

1. Écrivez un programme C qui, lorsqu'il s'exécute :
   — Crée un processus fils (voir `man fork`) ;
   — Affiche le numéro de processus (pid) et le numéro de parent (ppid) des deux processus, c'est-à-dire du programme qui s'exécute et de son fils (voir `getpid, getppid`) ;
2. Jouez à multiplier les appels à `fork` (mais pas dans une boucle !) et observez le résultat.

3.2 Redirection de terminaux

Les différents terminaux que vous manipulez avec les raccourcis Ctrl + Alt + Fn sont associés à des fichiers de périphériques du répertoire `/dev`. Ainsi le terminal 1 correspond à `/dev/tty1`, le terminal 2 à `/dev/tty2`, etc. Il est possible d'écrire dans ces fichiers comme on écrirait dans n'importe quel fichier texte.

1. Lancez le programme `affiche` en arrière-plan dans le terminal `tty1` tout en redirigeant sa sortie standard vers le terminal `tty2`.
2. Créez un second programme `affiche2.c` similaire à `affiche.c` mais avec un affichage différent. Lancez ce programme, en arrière-plan, dans le terminal `tty2` en redirigeant la sortie vers le terminal `tty1`.
3. Affichez la liste des *travaux* (processus en arrière plan) au moyen de la commande `jobs` et suspendez l'exécution du processus `affiche`. Affichez l'état des travaux.
4. Reprenez l'exécution de `affiche` en arrière plan. Suspendez l'exécution du processus `affiche2`. Continuez l'exécution de ce processus en avant-plan.
5. Tuez les 2 processus avec la commande `kill` appliquée au numéro de travail de chacun de ces processus.