Job Control

Fonctionnement, Trucs et Astuces

Olivier DALLE

Command, process, programs, jobs

- **Process**us: un **program**me en cours d'exécution
- **Command**e: une action demandée au shell
  - En général, saisie clavier jusqu'à un appui <return>
  - Un commande = un ou plusieurs processus
    - Pipeline : enchainement de pipes
    - Un processus peut créer des (nombreux) fils
- **Job:** ensemble de **TOUS** les processus **liés à une commande**
  - On parle aussi de **Process Group**

Job (groupe) pour quoi faire?

- Pour pouvoir les désigner **tous ensemble, collectivement**
  - Envoi d'un signal (exemple Ctrl-C : SIGINT à tous)
- PRINCIPE: TOUS les processus appartiennent à un groupe
  - Pas d'exception!
  - Quand **un processus est créé**: même groupe que le **parent**
- Un processus peut **changer de groupe:**
  - Avec os.setpgid()
  - Ssi groupe destination dans la **même session**

Groupe / Session?

- Session: Un ensemble de processus **plus large** que le groupe
  - Les processus descendant d'un shell de login
- **Déplacer** un processus dans **une autre session?**
  - **Impossible**… Sauf **en créant une nouvelle session**
  - Le processus qui change de session doit devenir **leader de session**
    - Primitive os.setsid()
    - Deplace aussi le processus dans **un nouveau groupe**

Shell avec Job Control?

- Le shell organise **qui accède** au terminal
  - **Qui** = quel groupe (cad quel job)
  - **Accède** = qui reçoit les **entrées**
    - **Accès total:** avant plan (foreground)
    - **Accès limité:** arrière plan (background)
- **Comment ?**
  - **PROTOCOLE** entre shell et pilote du terminal

Protocole de Job Control

- Tentative de lecture en arrière plan?
  - STOPé par le terminal
- Tentative d'écriture en arrière plan?
  - STOPé dans certains cas (TOSTOP mode)
- Caractères de contrôle:
  - SUSP (Ctrl-Z) : envoie SIGTSTP à tous les processus du job en avant plan
  - QUIT (Ctrl-\): envoie SIGQUIT à tous les processus du job en AvP.
  - INTR (Ctrl-C): envoie SIGINT à tous les processus du job en AvP.

Tentatives d'E/S en Arrière Plan

- Tentative de lecture
  - SIGTTIN envoyé à tout le groupe
    - Default = STOP les processus
    - Si default ignoré/redirigé: Erreur EIO en lecture
- Tentatives d'écriture
  - SIGTTOU envoyé à tout le groupe
    - Ssi mode TOSTOP (non supporté par python??)
  - Default = STOP les processus
  - Si default ignoré/redirigé : écriture **permise**

Terminal de Contrôle

- Terminal affecté au niveau de la session
  - Le shell n'a pas à s'occuper de l'affectation initiale
- Les processus créés par fork() héritent du TC de leur père
  - Déconnection par setsid()
- Un job possède le contrôle du TC
  - Possibilité de donner le contrôle du TC à un job avec tcsetpgrp()

Travail du shell lors de la création en avant-plan

- Etat initial: le shell possède le contrôle du TC
  - Créer processus fils
  - Les placer dans un groupe : setpgid()
  - Donner le contrôle du terminal : tcsetpgrp()
  - Attendre la terminaison / arrêt (stop)
  - Récupérer le contrôle du terminal : tcsetpgrp()

Trucs et conseils

- Penser à bloquer systématiquement la réception de SIGTTIN/ SIGTTOU par le **processus shell**
  - signal.signal(signal.SIGTTIN, signal.SIG_IGN)
- Comment être informé de l'arrêt d'un groupe en Arrrière plan?
  - Attendre avec os.waitpid(pid,os.WUNTRACED)
  - Tester avec if WIFSTOPPED(status)
    - Compatible avec tests habituels WIFEXITED ou WIFSIGNALED

Trucs et conseils (2)

- Pb de *race condition* avec setpgid()…
  - Le pere ET les fils appellent tous setpgid
    - Le fils leader: setpgid(0,0)
    - (opt?) Les autres fils: setpgid(0,fils_leader)
    - Le pere: setpgid(fils, fils_leader)