iNFO

IUT GRAND OUEST NORMANDIE

R 1.04

2023 - 2024

Introduction aux systèmes d'exploitation et à leur fonctionnement

TD N°5

« Gestion de processus : première approche »

ANNE Jean-François

D'après le TD de F. BOURDON

Le but de ce TD est de se familiariser avec les systèmes d'exploitation et avec leur fonctionnement.

« Gestion de processus : première approche »

Notions vues dans ce TD :

Commandes " echo ", " exec ", " . ", création de processus

Nombre de séance de 2h prévu pour faire ce TD : 1.

PS : Les parties correspondant à du travail à faire sont toutes en italiques ; le restant étant du complément au cours.

1. Gestion des processus

En utilisant le langage de commande shell (« bash ») et les commandes « echo », « exec » et « . » nous allons voir la gestion des processus, la création de processus fils et l'exécution d'une commande dans un processus.

✍ *Construire avec l'éditeur de texte « vi » les trois fichiers de commandes « com », « com2 » et « com3 » tels qu'ils sont décrits ci-dessous :*

```
prompt> cat com1
        echo je suis dans com1
        ps fl
        com2
        echo je suis de retour dans com1
prompt> cat com2
        echo je suis dans com2
        ps fl
        com3
        echo je suis de retour dans com2
prompt> cat com3
        echo je suis dans com3
        ps fl
prompt>
```

✍ *Essayer de lancer la commande « com1 ». Que se passe-t-il ? Que devez-vous faire pour que l'exécution puisse être lancée ?*

✍ *Une fois ce problème résolu, lancer la commande « com1 » en redirigeant les traces dans le fichier « trace1 ».*

2. Utilisation de la commande « . »

✍ *Reprendre les fichiers correspondant aux commandes « com1 », « com2 » et « com3 » et remplacer les appels aux commandes « com? » par « . com? ». Vous pouvez créer de nouveaux fichiers « com1p », « com2p » et « com3p » ainsi modifiés.*

✍ *Lancer la nouvelle commande « com1p » en redirigeant les traces dans le fichier « trace2 ».*

---

3. Utilisation de la commande « exec »

✍ *Reprendre les fichiers correspondant aux commandes « com1 », « com2 » et « com3 » et remplacer les appels aux commandes « com? » par « exec com? ». Vous pouvez créer de nouveaux fichiers « com1e », « com2e » et « com3e » ainsi modifiés.*

✍ *Lancer la nouvelle commande « com1e » en redirigeant les traces dans le fichier « trace3 ».*

---

4. Conclusions

✍ *A partir de l'analyse des trois fichiers de trace obtenus (« trace1 », « trace2 » et « trace3 »), expliquer les différences entre l'appel d'une commande « com » (prompt> com), l'utilisation de la commande « . » (prompt> . com) et l'utilisation de la commande « exec » (prompt> exec com).*

---

5. Exécutions séquentielles de processus

Dans la suite des exercices lancez les commandes proposées et concluez sur les résultats obtenus.

La commande « ps » affiche la liste des processus lancés depuis le shell courant.

```
prompt> ps
...
prompt> echo $$ (donne le PID du processus correspondant
au shell courant)
...
prompt> tty (donne le terminal rattaché au shell courant)
...
```

La commande « ps aux » liste tous les processus (processus système et processus utilisateur) en mentionnant le nom du propriétaire.

```
prompt> ps -aux
...
```

Lorsque TTY = ? cela signifie que le processus n'a pas été initialisé depuis un terminal. C'est le cas notamment des processus système.

USER : propriétaire
PID : identifiant du process
%CPU : temps CPU utilisé par le processus
%MEM : RAM occupée par le processus
TTY : terminal de rattachement du processus
START : date à laquelle le process a été lancé
TIME : durée totale d'activité du process.
CMD : programme à l'origine du process.

✍ *Proposez une commande qui liste uniquement les process lancés par « root » ?*

Un process est créé lorsque l'on exécute un programme.

```
prompt> echo $$
...
prompt> bash (lancement d'un nouveau shell = création d'un nouveau process)
prompt> ps
...
prompt> echo $$
...
prompt>
```

✍ *Précisez quel processus est parent de quel processus !*

Les options "fl" de la commande "ps" affiche l'arborescence des processus.

✍ *Lancez la commande "ps" avec ces options.*

6. Enchaînement des processus

```
prompt> <commande1> ; <commande2>
```

Les commandes sont exécutées séquentiellement.

✍ *Que fait la ligne de commandes suivante ? Expliquez le résultat obtenu. Vous regarderez dans le manuel le rôle de la commande « sleep ».*

```
prompt> echo début ; sleep 5 ; echo milieu ; sleep 5 ; echo fin
…
```

✍ *En utilisant les commandes « true » et « false », qui renvoient respectivement un code de retour qui indique une réussite et un échec, montrer que lors de l'exécution successive (en utilisant l'opérateur « ; ») de deux commandes le résultat de la première n'influence pas celui de la deuxième.*

Pour cela, lancez les commandes suivantes :

```
prompt> echo debut ; false ; echo fin
…
prompt> echo debut ; true ; echo fin
…
prompt>
```

Rappels :

Ne pas confondre avec :

Le pipe <com1> | <com2> où l'on redirige la sortie de "com1" vers l'entrée de "com2"

La redirection <com1> > <fichier.data> où l'on redirige la sortie de "com1" vers le fichier "fichier.data".

7. Lancement en mode détaché (background ou en tâche de fond)

```
prompt> date
…
prompt>
```

Le Shell nous redonne la main lorsque l'exécution de la commande est terminée

```
prompt> xterm
```

Le Shell ne nous redonne pas la main !

La raison : le process (xterm) que l'on vient de lancer est toujours en cours d'exécution (dans le cas de la commande date, c'est pratiquement instantané).

✍ *Vous pouvez vérifier quels sont les processus créés en appelant la commande "ps" avec l'option "lf" dans le "xterm" créé.*

En fermant le "xterm" (commande "exit" dans la fenêtre créée), on s'aperçoit que le Shell nous redonne la main.

Pour que le Shell nous redonne aussitôt la main, il faut lancer le "xterm" en tâche de fond en utilisant l'opérateur "&".

```
prompt> xterm &
```

Le Shell affiche l'identifiant du processus.

✍ *Vérifier le numéro du process, ainsi que l'héritage des processus créés en lançant "ps lf" dans le Shell d'où a été lancé le "xterm", mais aussi dans le "xterm" lui-même.*

```
prompt> ps lf
…
```

✍ *Comparez le résultat de chacune des deux lignes de commandes suivantes. Expliquez ce qui se passe dans le second cas, lorsque l'on ferme le "xterm" ?*

```
prompt> xterm & ps lf
```

avec

```
prompt> xterm ; ps lf
```

8. Interruption d'un processus

Il est possible d'envoyer des signaux aux processus. En particulier des signaux permettant d'interrompre ces processus. Les process peuvent réagir différemment à ces signaux.

"Localement"

On envoie les signaux par des séquences de touches (ex. Ctrl C, Ctrl D).

```
prompt> bash
prompt> echo $$
...
prompt> (Ctrl D)
prompt> echo $$
…
prompt>

prompt> cat > trace
Du texte (Ctrl D) envoie un signal d'interruption au process
prompt> cat trace
Du texteprompt>

prompt> cat > trace
Du texte (Ctrl C) tue le processus
prompt> cat trace
prompt>
```

✍ *Lancez les signaux correspondant dans les trois situations suivantes :*

```
prompt> xterm
(CTRL C)
prompt> xterm &
(CTRL D)
prompt> xterm &
(Ctrl D dans le xterm)
```

"A distance"

La commande "kill"

```
prompt> kill PID
```

```
prompt> xterm &
[1] 9623
prompt> kill 9623
```

9. Déterminer le type d'une commande

Commande = programme ou directive Shell

Programme = fichier exécutable = binaire ou script

Les binaires créent de nouveaux processus

Les directives du Shell n'en créent pas

Pour les scripts, cela dépend de la manière dont on les exécute

```
prompt> type cd
cd is a shell builtin --> directive du Shell

prompt> type ls
/bin/ls is /bin/ls --> programme
```

Si c'est un programme …

```
prompt> file /bin/ls
/bin/ls: ELF 32-bit ... executable ... --> binaire
```

Remarque :

```
prompt> file .
directory
```

II. Webographie :

- https://bourdon.users.info.unicaen.fr/cours/IUT-1A/index.html
-