Processus, Droits, et variables d'environnement

Objectifs

- notion de processus, commandes `ps`, `kill`,
- comprendre et manipuler la notion de droits sous Unix,
- connaître les variables globales d'environnement les plus importantes.

1 Processus Unix

Sous Unix, l'activité d'exécution d'un programme par le processeur est représentée par un processus. Le contrôle de l'exécution de ces processus est possible au moyen :

- de commandes Unix : `ps` (pour lister les processus existants), `kill` (pour transmettre des signaux à un processus, en particulier tuer un processus),
- de commandes spécifiques au Shell : `jobs`, `fg`, `bg` ...

1.1 La commande `ps`

La commande `ps` permet de donner des informations à propos des processus existants, notamment :
- `PID` : identifiant (unique) du processus, permettant de le désigner,
- état : actif, suspendu ...,
- commande exécutée par le processus.

Sans option, cela concerne les processus qui ont été lancés à partir du terminal d'où `ps` est exécuté.

Travail à effectuer :
- dans un terminal, tapez `ps`,
- dans ce même terminal, tapez `evince &`,
- retapez `ps`.

Cette commande `ps` dispose de nombreuses options parmi lesquelles on trouve `ps -e` et `ps -f` (ou combinées `ps -ef`).

- `ps -e` permet de lister l'ensemble des processus actifs de l'ordinateur,
- `ps -f` affiche les processus sous un format long.

Dans ce format long, quelques informations peuvent être utiles :

- le propriétaire du processus (a priori si vous êtes seul sur la machine, il n'y a que le super-utilisateur `root` et vous),
- `PID` du processus père, c'est-à-dire du processus ayant lancé le processus considéré, et lui ayant ainsi transmis la plupart des éléments de son environnement d'exécution (variables, entrées/sorties...). Tout processus a un père, sauf le processus initial, appelé `init`, de `PID` 1.

Travail à effectuer :
- expérimentez les différentes options,
- comment lister l'ensemble des processus dont vous êtes propriétaire ? (deux solutions : utilisation de `grep` ou option supplémentaire)

La commande `top`, disponible sur la plupart des Unix, fournit une liste des processus analogue à celle donnée par la commande `ps`, où les informations relatives à la consommation des ressources (mémoire vive, temps processeur...) sont rafraîchies et classées à intervalles réguliers.

1.2 la commande `kill`

Quelques fois, une application (par exemple graphique) refuse de répondre et en particulier de s'arrêter, même en tuant la fenêtre (icône de la fenêtre avec la croix).

Une façon de l'arrêter à coup sûr est de tuer le processus correspondant. Pour cela, il faut utiliser la commande `kill` avec comme paramètre l'identificateur du processus à tuer (le `PID`), d'où l'utilité de connaître le PID des processus avec la commande `ps`.

Travail à effectuer :
- dans un terminal, tapez `evince &`,
- retapez `ps -ef` et repérez le `PID` du processus `evince`,
- tapez `kill -9 PID`, cette commande enverra le signal `KILL` au processus, ce qui le forcera à s'arrêter.

1.3 Processus en avant-plan et en arrière-plan

Un processus peut être exécuté en arrière-plan. Dans ce cas, il s'exécute en parallèle avec le processus père, et n'interagit pas avec le terminal (souvenez-vous du TP1 !). Il est possible de lancer une commande en arrière plan à partir du Shell, en la faisant suivre d'un `&`. Ainsi :

```
> evince &
>
```

lance `evince`, et passe en attente de la ligne de commande suivante, sans attendre la fin de la commande `evince`.

Il est possible, sur la plupart des Shell, de changer le mode d'exécution d'un processus. Avec les `C-shell`, par exemple :

- la commande `jobs` permet de connaitre la liste des processus qui ne sont pas en avant-plan. Il faut noter que les identifiants de jobs de cette liste sont spécifiques au shell, et n'ont pas de rapport avec les `PID` fournis par la commande `ps`,
- la commande `fg %n` permet de poursuivre l'exécution du job `n` en avant-plan,
- la commande `bg %n` permet de poursuivre l'exécution du job `n` en arrière-plan.

Vous pouvez passer en tsch (en tapant la commande `tcsh` et suivre les étapes suivantes en vérifiant ce qui se passe :
- dans un terminal, tapez `evince &`,
- tapez `firefox &`

— tapez `jobs`
— tapez `fg 1`
— tapez `Ctrl-Z`, qui suspend le processus courant
— tapez `jobs`
— tapez `bg 1`
— tapez `jobs`

1.4 Suspension et reprise d'un processus

Un processus en avant-plan peut être suspendu, c'est-à-dire que son exécution est provisoirement arrêtée, mais pourra être reprise par la suite (à partir de l'état atteint au moment de l'arrêt).

— la frappe de `Ctrl-Z` permet de suspendre le processus en avant-plan,
— les commandes `fg` ou `bg` permettent de reprendre l'exécution d'un processus suspendu, en avant-plan ou en arrière-plan.

Vous verrez plus en détail les processus et les signaux au second semestre dans le cours de Systèmes Centralisés.

2 Droits d'accès et modification

2.1 Description des droits d'accès

Nous avons vu que la commande `ls` permet de lister les fichiers d'un répertoire. Cependant, cette commande possède plusieurs modes (`man ls`), notamment un mode « long » qui permet d'afficher de nombreux détails sur les fichiers : leurs droits d'accès, leur taille en octets, l'heure de dernière modification et le nom. Il s'agit de l'option `-l` (*long*) :

```
> ls -l
total 4
drwxr-xr--   2 camichel    in           512 Jul 29 09:54 CaML/
-rw-r-----   1 camichel    in           437 Jul 29 10:00 tolkien.txt
```

Après la ligne indiquant "total", chaque ligne donne des informations sur un fichier ou un dossier :
— la première colonne indique les droits (voir plus loin)
— la troisième indique le nom du propriétaire
— la quatrième indique le groupe propriétaire
— la cinquième indique la taille de l'objet
— les suivantes indiquent les dates et heures de dernière modification
— la dernière indique le nom de l'objet.

Intéressons-nous à la suite de lettres et de tirets de la première colonne : elle donne le type de fichier ainsi que ses droits d'accès. Cette séquence de dix caractères est classiquement séparée en quatre groupes :

`-`	`rw-`	`r--`	`r--`
Type de fichier	Droits		
	`u`	`g`	`o`

Type de fichier

Le premier caractère renseigne le type de fichier :
- `-` (tiret) : fichier normal (fichier texte, ...) ;
- `d` (**d***irectory*) : répertoire ;
- `l` (**l***ink*) : lien symbolique (`man ln`) ;
- etc... (`man ls` pour en savoir plus).

Les droits

Les neuf caractères restants sont séparés en trois groupes de trois caractères :
- `u` : le premier groupe précise les droits du propriétaire (**u***ser*) du fichier ;
- `g` : le deuxième groupe précise les droits des autres utilisateurs du groupe [1] (**g***roup*) du propriétaire ;
- `o` : le troisième groupe précise les droits des autres utilisateurs extérieurs (**o***thers*) au groupe.

Remarque : Il existe un super-utilisateur (`root`), administrateur de l'ordinateur, qui possède tous les droits et n'est donc pas concerné par ces restrictions.

Chaque groupe de trois caractères est construit de façon identique :
- le premier caractère précise s'il y a droit de lecture (**r***ead*) ou non, respectivement `r` ou `-` ;
- le deuxième caractère précise s'il y a droit d'écriture (**w***rite*) ou non, respectivement `w` ou `-` ;
- le troisième caractère précise s'il y a droit d'exécution [2] (*e***x***ecute*) ou non, respectivement `x` ou `-`.

Par exemple : la ligne 1 indique un dossier qui porte le nom CaML, et qui est accessible en :
- lecture, ecriture et exécution pour son propriétaire
- lecture et exécution pour les membes du groupe, mais pas en écriture
- lecture seulement pour les autres (interdiction d'accès avec cd)

2.2 Modifications des droits d'accès : `chmod`

Pour modifier les droits des fichiers **dont vous êtes le propriétaire** [3], il faut utiliser la commande `chmod` dont la syntaxe est :

`chmod` [`-R`] <*Droits*> <*Fichiers*>

L'argument *Droits* a deux écritures possibles :
- la notation octale : chaque droit est représenté par un bit qui est à 1 si le droit est accordé et à 0 sinon. Chaque groupe est alors représenté
  - soit par un nombre binaire sur trois bits, le droit en lecture étant le bit de poids fort,
  - soit par la représentation de ce nombre en décimal (compris, donc, entre 0 et 7).

  Exemple :

r	w	-		r	-	-		r	-	-
1	1	0		1	0	0		1	0	0
	6				4				4	

1. En unix, tout utilisateur fait partie d'au moins un groupe d'utilisateurs.
2. important : Ce droit est indispensable pour pouvoir entrer dans un répertoire avec la commande `cd`.
3. On rappelle que le propriétaire d'un fichier, ainsi que son groupe, est visible dans l'affichage long de `ls` (`ls -l`).

— la notation « additive » : au lieu de remplacer complètement les droits comme avec la notation octale, il est possible d'ajouter (ou d'enlever) certains droits localement. La notation utilisée est la suivante (principaux choix, faire `man chmod` pour plus de détails) :

[u|g|o|a](+|-—=)(r|w|x)

— le premier caractère (optionnel) précise pour quel groupe d'utilisateurs la modification sera faite : s'il n'est pas précisé, la modification se fera pour tous (groupe `a`, **a***ll*) ; le caractère `a` indique que la modification est effectuée pour les trois groupes (`u`, `g` et `o`)

— le deuxième caractère indique s'il faut ajouter (`+`), enlever (`-`) ou affecter (`=`) le droit ;

— le troisième caractère précise quel type de droit modifier : lecture (`r`), écriture (`w`) ou exécution (`x`). Le caractère `X` existe également et peut être très utile avec l'option `-R` de `chmod` (`man chmod` pour plus de détails).

L'option `-R` (pour **R***ecursive*) permet d'appliquer la modification de droits pour tous les fichiers (fichiers classiques, répertoires, liens symboliques...) de tous les sous-répertoires des répertoires présents dans *Fichiers*.

Travail : Effectuez la séquence suivante et vérifiez ce qui se passe :

```
> cd EnvironnementInformatique
> mkdir TEST
> chmod 600 TEST
> ls -l
> cd TEST
> chmod +x TEST
> ls -l
> cd TEST
> cd ..
> chmod a+x TEST
> ls -l
> chmod go-x TEST
> ls -l
```

3 Les variables d'environnement : suite

Votre environnement de travail est en partie défini par un certain nombre de variables utilisées par le Shell ; ce sont les variables d'environnement. Nous allons en étudier quelques-unes.

3.1 Quelques variables d'environnement

Quelques petits points :

— les variables d'environnement sont conventionnellement en `MAJUSCULES` ;

— le `$` qui précède la variable signifie que l'on en prend la valeur.

Voici quelques variables parmi les plus importantes :

— `HOME` : cette variable contient le chemin absolu de votre répertoire de connexion et ne doit pas être modifiée ;

— `PATH` : cette variable contient une liste de répertoires dans lesquels chercher les commandes désignées par des chemins relatifs (dont les commandes vues jusqu'à présent) ;

Ainsi, `cd` (appel sans paramètre) est équivalent à `cd $HOME`.

3.2 Manipuler les variables d'environnement

Vous avez à votre disposition

- `echo` qui permet afficher une chaîne et en particulier les variables d'environnement :
  ```
  > echo $PATH
  > echo $HOME
  ```
  Noter la présence du `$`. (Essayez `echo PATH` pour voir la différence)
- l'affectation `=` qui permet de modifier une variable :
  ```
  > PATH=${PATH}:/usr/local/bin/
  ```
- `export` qui permet de rendre visibles les valeurs des variables d'environnement à des shells ou commandes consécutifs au shell courant :
  ```
  > export PATH
  ```
- souvent l'affectation et l'exportation se font en une seule commande :
  ```
  > export PATH=${PATH}:/usr/local/bin/
  ```

3.3 Retour sur `PATH`

3.3.1 La commande `which`

Une commande intéressante est la commande `which` qui permet de retrouver l'endroit où se situe un fichier **à condition que le chemin cherché fasse partie des chemins spécifiés dans** `PATH` (c'est-à-dire si vous pouvez le référencer sans donner le chemin complet ou l'exécuter s'il s'agit d'une commande).

3.3.2 `source`

`source` *Fichier* exécute toutes les commandes Shell incluses dans le fichier *Fichier*.

3.3.3 Vos exécutables

Vous allez être amenés à écrire des programmes dans différents langages et à générer des exécutables. Le lancement de ces éxécutables se fait en tapant :

```
./nom_exécutable
```

depuis le dossier où il se trouve, ou en tapant son chemin absolu. Mais vous pouvez, en initialisant correctement la variable PATH, éviter de taper `./` ou même lancer vos programmes depuis n'importe quel dossier si le chemin de ces exécutables est enregistré dans la variable PATH.

Travail à effectuer :

- rechercher le chemin de `chmod`, `matlab...` et `which`,
- enregistrez le fichier bonjour.c depuis Moodle dans votre répertoire de travail
- ouvrez le avec gvim et essayez de comprendre ce qu'il fait (notez les facilités offertes par gvim : couleurs, indentation, encadrement des sections de code, ...)
- compiler ce programe avec la commande `gcc -o bonjour bonjour.c` qui produira si tout va bien un exécutable nommé bonjour ; Et effectuez la séquence suivante :

```
> bonjour
> ./bonjour
> export PATH=$PATH:.
> echo $PATH
> which bonjour
> bonjour
```

Que constatez-vous ? Comment l'expliquer ?
Même si l'ajout du '.' dans PATH peut sembler pratique (exécution de votre programme directement avec son nom sans nécessité de le faire précéder par './'), cela peut s'avérer dangereux et est déconseillé.

Pensez-donc à compléter votre .bashrc avec :

```
alias ll="ls -laF"
alias la="ls -a"
alias del="rm -i"
... et tout autre alias utile
```

Dernières remarques :

- la commande `mkdir TEST` peut indiquer que le répertoire `TEST` existe déjà si vous avez suivi correctement les instructions de l'exercice de la section 2.2,
- le raccourci `~` est équivalent à votre répertoire de connexion ; du coup, `~` permet de donner facilement un chemin relatif à un répertoire de connexion.

4 CQFAR (Ce Qu'il Faut Avoir Retenu)

- savoir identifier les processus en cours d'exécution,
- savoir tuer le processus associé à une application qui ne répond plus,
- savoir déchiffrer les droits d'un fichier Unix,
- savoir modifier les droits d'un fichier Unix,
- connaître les variables d'environnement `HOME` et `PATH` ; savoir modifier cette dernière.