

TP1 OS avancés et embarqués

Table des matières

| | |
|--|---|
| La hiérarchie des répertoires sous Linux..... | 2 |
| Les répertoires à la racine | 2 |
| Les répertoires dans /usr..... | 3 |
| Les répertoires dans /var..... | 4 |
| Quelques spécificités de Linux | 4 |
| Le système de fichiers virtuel « /proc » | 4 |
| Quelques exemples de fichiers..... | 5 |
| Les processus en cours d'exécution | 5 |
| Interaction avec le noyau | 6 |
| Quelques exercices..... | 6 |

La hiérarchie des répertoires sous Linux

Dans un système Linux, les fichiers sont organisés selon une arborescence bien précise. Elle suit effectivement le standard FHS (File Hierarchy Standard¹). Cette partie du TP est totalement inspiré du standard FHS (document à consulter absolument)

Nous pouvons grossièrement classer les répertoires d'après le fait que les fichiers soient partageables (ou pas) ou statiques (ou dynamique) :

- Les fichiers partageables sont ceux qui peuvent être stockés sur un seul hôte et partagés par tous.
- Les fichiers statiques incluent les binaires, les bibliothèques, les docs, et d'autres fichiers qui ne changent pas sans l'intervention de l'administrateur.

| | Partageable | Non partageable |
|----------------------------------|------------------------------|-----------------------|
| Statiques | /usr /opt | /etc /boot |
| Variables (ou dynamiques) | /var/mail /var/spool/news | /var/run /var/lock |

Les répertoires à la racine

«/» représente la racine du système de fichiers des OS unixiens. Ce répertoire est souvent appelé répertoire *racine*. C'est le point d'entrée de toute l'arborescence.


/bin contient les programmes (commandes) ou binaires qui peuvent être utilisés par l'administrateur ou les utilisateurs. Il ne doit y avoir aucun sous répertoire dans */bin*. On trouve un minimum de commandes dans ce répertoire : cat, chgrp, chmod, chown, cp, date, dd, df, dmesg, ...). Certains programmes doivent absolument se trouver dans ce répertoire s'ils sont installés : tar, gunzip, netstat, etc.

/boot contient tous les fichiers utiles pour l'exécution du chargeur (ou boot loader). Ce répertoire ne contient pas les fichiers de configuration qui ne sont pas nécessaires pour que le démarrage puisse se faire. Le noyau du système doit absolument se trouver dans ce répertoire ou dans le répertoire racine.

/dev contient des fichiers spéciaux ou les fichiers périphériques qui sont des points d'entrée vers des périphériques physiques. Exemple : les terminaux correspondent aux *fichiers /dev/ttyxxx*.

/etc contient les fichiers de configuration dont les divers programmes se servent pour avoir des informations sur la configuration du système. Ces fichiers doivent être statiques et non exécutables.

¹ <http://www.pathname.com/fhs/> reprise ensuite par la Linux foundation : <https://refspecs.linuxfoundation.org/fhs.shtml>

| | | |
|---|--|-----------------------------------|
|  | TP1 – OS Avancés et embarqués FHS | jalil.boukhobza@ensta-bretagne.fr |
|---|--|-----------------------------------|

Les répertoires (ou liens symboliques) suivants se trouvent dans ce répertoire : (opt, X11, xml, etc.). Les répertoires suivants peuvent aussi se trouver dans /etc si les fonctionnalités correspondantes sont installées : exports, fstab, gateways, hosts, etc.)

/etc/X11 *idem*, mais spécifique au système Xwindow

/home contient les données propres à chaque utilisateur. La structure interne du répertoire /home est laissée au bon vouloir de l'utilisateur/administrateur.

/lib contient les bibliothèques partagées nécessaires au démarrage et à l'exécution des commandes sur le système de fichiers racine (principalement des programmes placés dans/bin et /sbin).

/media contient des répertoires utilisés comme points de montage pour les périphériques tels qu'un CDROM, clé USB ou autre.

/mnt est le point de montage d'un système de fichiers temporaire.

/root le répertoire « home » du super utilisateur (peut être changé, mais cet emplacement est conseillé).

/sbin contient (avec les répertoires /usr/bin et /usr/sbin) les programmes systèmes nécessaires pour l'administration du système : fdisk, fsck, ifconfig, init, mkfs, reboot, swapon, etc.

/srv données relatives au serveur implémenté sur la machine au cas où il y en a un.

/tmp contient tous les fichiers temporaires générés par le système ou les applications en cours d'exécution.

Les répertoires dans /usr

/usr contient des données en lecture seule partageables. Ces données ne doivent pas être écrites (bin, include, librairies, sbin, etc.)


/usr/bin est le répertoire général pour les programmes utilisables sur le système : perl, python, wish, etc.

/usr/include contient les fichiers entêtes (.h) pour le compilateur C. Des sous-répertoires permettent de les classer par catégorie (net, sys, rpc, etc). Certains sous-répertoires sont des liens symboliques vers les fichiers entêtes du noyau (comme par exemple scsi, asm, linux, etc).

/usr/lib contient les bibliothèques utilisées par les programmes des utilisateurs ainsi que divers programmes qui ne sont pas faits pour être lancés directement par les utilisateurs.

/usr/local contient une arborescence utilisée pour stocker, les programmes (/usr/local/bin), les bibliothèques (/usr/local/lib), les documentations (/usr/local/doc), etc qui est spécifique à la machine ou au site.

usr/sbin contient des exécutables standards non essentiels utilisés en exclusivité par l'administrateur.

| | | |
|---|--|-----------------------------------|
|  | TP1 – OS Avancés et embarqués FHS | jalil.boukhobza@ensta-bretagne.fr |
|---|--|-----------------------------------|

/usr/share contient les fichiers à partager indépendants de l'architecture.

/usr/src le code source du noyau peut être placé dans cet endroit.

Les répertoires dans /var

/var contient les fichiers de données variables. Ceci inclut les répertoires de spool, fichiers temporaires, données de connexion, etc.

/var/cache cache de données pour l'application

/var/crash contient les fichiers *dumps* des crashes système.

/var/lib contient les informations d'état des applications

/var/lock contient les « fichiers de verrouillage ». Par convention, le nom des fichiers est toujours LCK.<périphérique> où <périphérique>.

/var/log est utilisé pour stocker les divers journaux du système. Ces fichiers journaux sont très utiles pour diagnostiquer un problème.

/var/spool contient les files d'attente pour les impressions et les fichiers.

/var/mail contient les boîtes à lettres (mailbox) et les messages (mails) des utilisateurs.

Quelques spécificités de Linux

/dev /null est la poubelle de Linux, tout ce qu'on y envoie est perdu et une lecture retourne un EOF.


/dev/zero est une source de zéros, une lecture donnera autant de zéros que le permet la taille de la donnée et une écriture est impossible.

/proc : le système de fichiers proc donne les informations sur l'état actuel du système.

Le système de fichiers virtuel « /proc »

Le noyau Linux fournit un mécanisme qui donne accès via une arborescence de fichiers aux informations internes du noyau et en permet la modification pendant son exécution grâce au système de fichiers /proc.

Le système de fichiers /proc est un mécanisme qui est utilisé par le noyau et ses modules pour communiquer des informations concernant processus. Il permet d'interagir avec les structures de données internes du noyau, d'acquérir des informations utiles sur les processus, et de changer certains paramètres du noyau à la volée. A la différence des autres systèmes de fichiers, /proc est stocké en mémoire plutôt que sur disque.

| | | |
|---|--|-----------------------------------|
|  | TP1 – OS Avancés et embarqués FHS | jalil.boukhobza@ensta-bretagne.fr |
|---|--|-----------------------------------|

Si l'on exécute une commande « ls -l » sur /proc, on voit que la plupart des fichiers ont une longueur nulle. Cependant, quand le fichier est visualisé, on aperçoit un certain nombre d'informations. Cela est possible car /proc s'inscrit dans la couche système de fichier virtuel (VFS). En revanche, lorsque « VFS » fait des appels d'inodes adressant des fichiers/répertoires au système de fichiers /proc, ce dernier en réalité crée ces fichiers/répertoires à partir des informations contenues dans le noyau.

Exemple :

```
$ file /proc/cpuinfo
```

L'exécution de cette commande vous montre que le fichier n'existe pas et pourtant... :

```
$ cat /proc/cpuinfo
```

Vous donne bien un ensemble d'informations relatives au CPU.

Quelques exemples de fichiers

Le système de fichier /proc permet de rassembler des informations utiles sur le système et le fonctionnement du noyau. Voici un certain nombre de fichiers importants :

- /proc/cpuinfo - informations sur le CPU (modèle, famille, taille du cache, etc.)
- /proc/meminfo - informations concernant la RAM physique, l'espace réservé pour le « Swap » etc.
- /proc/mounts - liste des systèmes de fichiers montés
- /proc/devices - liste des périphériques disponibles
- /proc/filesystems - les systèmes de fichiers supportés
- /proc/modules - liste des modules activés
- /proc/version - version du noyau
- /proc/cmdline - les paramètres passés au noyau lors de la mise en route

Il y a, bien évidemment, beaucoup plus de fichiers que ceux énumérés plus haut. Vous pouvez faire appel à la commande « more » pour scruter chaque fichier du répertoire /proc.

Les processus en cours d'exécution

Le système de fichiers /proc peut servir à consulter les informations relatives aux processus en cours d'exécution. Le répertoire /proc contient des sous-répertoires numérotés qui correspondent au numéro d'identification des processus (PID). Ainsi, pour chaque processus en cours d'exécution, il y a un sous-répertoire de /proc qui porte comme nom, le PID de chaque processus. Ces sous-répertoires contiennent des fichiers qui fournissent chacun des détails concernant l'état actuel et l'environnement d'un processus.

```
$ ps -aef |grep firefox
jboukh 5608 1 16 19:01 ? 00:20:21 /usr/lib/firefox/firefox-bin
```

La commande ci-dessus montre qu'il y a un processus en cours de Mozilla avec le PID 5608. Par correspondance, il devrait exister un sous-répertoire dans /proc portant le numéro 5608.

```
$ ls -l /proc/5608
total 0
-r--r--r-- 1 jboukh jalil 0 2017-10-07 19:59 cmdline
```

```
-r--r--r-- 1 jboukh jalil 0 2017-10-07 19:59cpuset
lrwxrwxrwx 1 jboukh jalil 0 2017-10-07 19:59cwd -> /home/jboukh
-r----- 1 jboukh jalil 0 2017-10-07 19:59environ
lrwxrwxrwx 1 jboukh jalil 0 2017-10-07 19:59exe -> /usr/lib/firefox/firefox-bin
dr-x----- 2 jboukh jalil 0 2017-10-07 19:59fd
-r--r--r-- 1 jboukh jalil 0 2017-10-07 19:59maps
-rw----- 1 jboukh jalil 0 2017-10-07 19:59mem
1 jboukh jalil 0 2017-10-07 19:59mounts
lrwxrwxrwx 1 jboukh jalil 0 2017-10-07 19:59root -> /
-r--r--r-- 1 jboukh jalil 0 2017-10-07 19:59stat
-r--r--r-- 1 jboukh jalil 0 2017-10-07 19:59stator
-r--r--r-- 1 jboukh jalil 0 2017-10-07 19:59status
..etc
```

Le fichier « cmdline » contient la commande invoquée pour démarrer le processus. Les variables d'environnement du processus sont incluses dans le fichier « environ ». Le fichier « status » contient des informations actualisées sur l'état du processus. Parmi lesquelles, le numéro utilisateur (UID) et le numéro du groupe d'utilisateurs (GID) de l'utilisateur qui a lancé le processus, le numéro d'identification du processus parent (PPID) qui a instancié le PID, et l'état courant du processus « Sleeping » (dormant) ou « Running » (en cours). Dans chaque répertoire dédié à un processus se trouvent plusieurs liens symboliques. Ainsi « cwd » est le lien au répertoire courant de travail du processus, le lien « exe » pointe sur le programme en cours d'exécution par le processus, « root » est le lien du processus à son répertoire racine (normalement « / »). Le répertoire « fd » contient des liens aux descripteurs de fichiers utilisés par le processus.

/proc/self est un sous-répertoire intéressant, il facilite l'utilisation de /proc par un programme pour trouver des informations sur ses propres processus. La commande /proc/self est un lien symbolique au répertoire /proc qui correspond au processus accédant au répertoire /proc.

Interaction avec le noyau

La plupart des fichiers /proc précédemment présentés sont accessibles **uniquement en lecture**. Toutefois, à l'intérieur du système de fichiers /proc il est prévu d'interagir avec le noyau à partir de fichiers en lecture/écriture.

L'écriture dans ces fichiers peut changer l'état du noyau, il faut les modifier avec beaucoup de précaution. Le répertoire /proc/sys est celui qui contient tous les fichiers en lecture/écriture ce qui permet de modifier le fonctionnement du noyau.

Pour plus de détails concernant le système de fichiers /proc :
/usr/src/linux/Documentation/filesystems/proc.txt (une fois le code source installé)
Explorer le contenu du répertoire /proc en vous aidant de la documentation.

Quelques exercices

- 1) Développez un programme permettant d'afficher les caractéristiques de votre machine et système en vous appuyant sur le répertoire virtuel /proc.

Votre programme devra afficher les informations suivantes :

- Concernant le CPU (fichier *cpuinfo*) :
 - o Modèle de CPU
 - o Fréquence

| | | |
|---|--|-----------------------------------|
|  | TP1 – OS Avancés et embarqués FHS | jalil.boukhobza@ensta-bretagne.fr |
|---|--|-----------------------------------|

- Taille du cache
 - Taille de l'espace d'adressage
 - Concernant la mémoire (fichier *meminfo*):
 - Taille de la mémoire totale
 - Taille de la mémoire libre
 - Concernant les partitions (fichier *partitions*)
 - Les partitions et leur taille en MO.
 - Concernant le système d'exploitation (fichier *version*)
 - La version du noyau installé
- 2) Ecrivez un programme qui prend en paramètre un numéro de pid et qui affiche pour ce processus :
- Son pid
 - Le nom de la commande exécutée
 - Son tgid
 - Le pid du père et le nom de la commande du père
 - Le nombre de threads fils de cette tâche
 - L'Uid et le Gid de la tâche.

Pour aller plus loin :

- 3) Regrouper le programme de la question 1) avec celui de la question 2) en utilisant les options de ligne de commande. Par exemple, si votre programme s'appelle « *my_procinfo* », pour déterminer les informations du CPU, vous pouvez utiliser l'option « -c », pour la mémoire « -m » et pour les processus, « -p <pid> ». Pour ce faire, vous utiliserez la fonction **getopt()**, voir exemple sur https://www.gnu.org/software/libc/manual/html_node/Example-of-Getopt.html.
- 4) Il s'agit dans cette partie d'explorer l'utilisation de la primitive **ptrace()** permettant de tracer un processus fils par son père et de connaître l'ensemble des appels systèmes réalisés par ce dernier. Pour ce faire, vous allez suivre un tutoriel (bien fait, même si daté) sur Linux Journal et qui se trouve ici <https://www.linuxjournal.com/article/6100> ²
- Pour ce tutoriel, il est demandé de faire le premier (interception de l'exec) et le second exemple (celui de l'interception de l'appel à write). Pour le second exemple, n'utilisez pas « /bin/lis » comme commande pour l'**execl**. A la place, vous pouvez créer votre programme qui écrit dans un fichier, le compiler séparément et y faire appel avec un **execl** (à la place du « /bin/lis »).

² Attention, il est possible que vous ayez besoin de changer le nom des registres pour choisir ceux de 64bits et inclure `/sys/reg.h` pour les reconnaître. Aussi, les noms des registres utilisés pour les processeurs 32bits et ceux 64bits n'est pas le même. Vous pouvez faire un tour sur `/usr/include/sys/reg.h` pour trouver les noms (attention le répertoire peut changer selon la distribution). Cf aussi <https://medium.com/@bjammal/process-tracing-system-call-analysis-with-pttrace-685a8844dbfa>