TP commandes unix

É. Carry, J.-M Friedt

15 novembre 2020

1 Arborescence

L'arborescence de GNU/Linux et comment s'y retrouver, dans l'ordre de priorité :

- `/` est la racine de l'arborescence. Le séparateur entre répertoire sera toujours le `/`.
- `/home` contient les répertoires des utilisateurs. Unix, et Linux en particulier, impose un confinement strict des fichiers de chaque utilisateur dans son répertoire. Seul l'administrateur (`root`) peut accéder aux fichiers du système en écriture : personne ne travaille comme administrateur.
- `/usr` contient les programmes locaux à un ordinateur : applications, entêtes de compilation (`/usr/include`), sources du système (`/usr/src`)
- `/lib` contient les bibliothèques de fonctions du système d'exploitation
- `/proc` et `/sys` contient des pseudo fichiers fournissant des informations sur la configuration et l'état du matériel et du système d'exploitation. Par exemple, `cat /proc/cpuinfo`.
- `/etc` contient les fichiers de configuration du système ou la configuration par défaut des logiciels utilisés par les utilisateurs en l'absence de fichier de configuration locale (par exemple `cat /etc/profile`).
- `/bin` contient les outils du système accessibles aux utilisateurs et `/sbin` contient les outils du systèmes qu'*a priori* seul l'administrateur a à utiliser.
- . (un point) est le répertoire courant
- .. (deux points) est le répertoire au-dessus du répertoire courant et est utilisé pour remonter dans l'arborescence

Les commandes de base :

- `ls` : list, afficher le contenu d'un répertoire
- `mv` : move, déplacer ou renommer un fichier. (`mv source dest`)
- `cd` : change directory, se déplacer dans l'arborescence. Pour remonter dans l'aroborescence : (`cd ..`)
- `rm` : remove, effacer un fichier
- `cp` : copy, copier un fichier. (`cp source dest`) Afin de copier un répertoire et son contenu, (`cp -r repertoire destination`).
- `cat` : afficher le contenu d'un fichier. (`cat fichier`)
- `mkdir` : créer un répertoire (`mkdir toto`)
- `man`, manuel, est la commnde la plus importante sous Unix, qui donne la liste des options et le mode d'emploi de chaque commande.

Créez un répertoire nommé `TP1` et déplacez-vous dedans.

Afin de remonter dans l'historique, utilisez la flèche vers le haut. Toute commande est lancée en tâche de fond lorsqu'elle est terminée par `&`.

2 Commandes simples

2.1 man

La commande la plus importante, et au pire la seule à connaître, est `man` pour accéder aux pages de documentations. `man commande` informe de l'utilisation d'une commande, ses arguments et ses options. Les pages de manuel sont organisées par sections, avec l'organisation décrite dans ... `man man`, et reproduite ci-dessous

```
1   Executable programs or shell commands
2   System calls (functions provided by the kernel)
3   Library calls (functions within program libraries)
4   Special files (usually found in /dev)
5   File formats and conventions eg /etc/passwd
6   Games
7   Miscellaneous  (including  macro  packages and conven-
tions), e.g. man(7), groff(7)
8   System administration commands (usually only for root)
```

Observer la différence entre `man 2 open` **et** `man 1 open`**. Commenter**

Si une commande n'est pas connue, `man -k mot_cle` recherche l'occurence du mot clé dans la desription des pages de manuel et facilite l'identification de la commande associée.

Imaginons que nous voulions trouver la commande en C qui nous permet de changer l'ordre des octets dans un mot de plus de 8 bits (et régler l'endianness : tapez la commande `man -k order` **et identifiez la commande appropriée.**

2.2 grep

`grep` recherche l'occurence d'un motif dans une chaîne de caractères. Elle est très utile pour *filtrer* la masse d'informations issues des autres commandes que nous verrons plus bas, et s'utilise donc quotidiennement pour rendre le résultat d'une recherche digeste.

Nous désirons connaître les utilisateurs ayant accès aux ports série du PC, et donc appartenant au group *dialout*.
`grep dialout /etc/group`

`grep` recherche la présence d'une chaîne de caractères dans l'argument qui lui est fourni, par exemple un fichier. Il sait aussi recherche récursivement dans une arborescence avec l'option `-r`

Nous avons vu plus haut que la commande `htons` **inverse les octets d'un mot de 16 bits. Nous désirons connaître l'implémentation de cette commande : trouver l'occurence de cette commande dans les sous-répertoires de le répertoire des fichiers d'entête et ses sous-répertoires** `/usr/include/linux`

`grep` propose une multitude d'options qui s'apprennent avec le temps : `-i` pour ne pas faire dépendre la recherche de majuscule/minuscule, `-v` pour rejeter au lieu d'accepter la correspondance du motif, `-A` et `-B` pour afficher les lignes avant et après celle correspondant au motif.

2.3 wget

Internet est devenu incontournable, mais l'accès aux informations qui y sont stockées est freiné par les interfaces graphiques aussi lentes qu'inconfortables pour rendre la recherche des données systématiques. Parmi les multitudes de protocoles permettant d'accéder aux données accessible par internet, HTTP[1] décrit l'accès au web. HTTP est implémenté dans la commande `wget` qui permet depuis la ligne de commande d'accéder à un site.

1. **Accéder à la page web** `http://jmfriedt.free.fr`**. Sachant que la sortie de la commande peut être la sortie standard si nous le demandons par** `-O -`**, filtrer uniquement les énoncés de documents relatifs au master d'électronique.**
2. **Récupérer toutes les images contenues dans le répertoire accessible sur le web** `http://jmfriedt.sequanux.org/130929_kvad/` **: étudier les options de** `wget` **nécessaires à atteindre ce résultat.**

2.4 find

`find` est une commande à la syntaxe un peu complexe mais fort utile pour trouver un ou des fichier(s) dans une arborescence.

Recherche d'un fichier : `find` (locate) qui prend comme argument le répertoire de départ de la recherche, la nature de la recherche (nom du fichier, date de création, permission ...), le motif à rechercher, et l'opération à effectuer.

Exemple : on trouve l'emplacement de `stdio.h` par `find /usr/include/ -name stdio.h -print` L'action `-print` est celle effectuée par défaut si aucune action n'est précisée.

Rechercher tous les fichiers liés au microprocesseur 32u4 dans le répertoire des fichiers d'entête de `avr-gcc`**, à savoir** `/usr/lib/avr`

Une option très puissance de `find` est l'exécution d'une commande sur le résultat de la recherche, par l'action `-exec`. Dans ce cas, l'argument est passé sous forme de la chaîne `{}` et la commande se termine par `\;`

Exemple : faire défiler toutes les photos d'un répertoire au moyen de l'affichage par la commande `display` de ImageMagick

```
find . -name '*.jpg' -exec display -delay 100 {} \;
```

1. RFC7230 à `https://tools.ietf.org/html/rfc7230`

2.5 cut

Au sein d'une ligne, nous voulons sélectionner (couper) certaines colonnes selon un motif – colonnes d'abscisses connu, ou de numéro de champ connu selon un séparateur connu par exemple.

Couper une/plusieurs colonnes d'un fichier selon sa position en nombre de caractères : `cut -c debut-fin`

Couper une/plusieurs colonnes d'un fichier selon sa position en indice du champ (`-f`) en choisissant un certain délimiteur (`-d`) : `cut -d\  -f 1-3`

Application : soit le fichier de trames GPS suivant `130810_heraklion_paris.nmea` (un extrait ci-dessous)

```
$GPGSA,A,3,02,24,12,10,,,,,,,,3.2,3.0,1.0*3
$GPRMC,182653.000,A,4714.9166,N,00601.2024,E,1.16,84.77,090709,,*38
$GPGGA,182654.000,4714.9176,N,00601.2037,E,1,07,2.1,240.2,M,48.0,M,,0000*52
$GPGSA,A,3,02,29,24,12,31,10,30,,,,,3.2,2.1,2.4*3A
$GPRMC,182654.000,A,4714.9176,N,00601.2037,E,1.56,62.01,090709,,*31
$GPGGA,182655.000,4714.9182,N,00601.2047,E,1,07,2.1,242.0,M,48.0,M,,0000*5F
$GPGSA,A,3,02,29,24,12,31,10,30,,,,,3.2,2.1,2.4*3A
$GPRMC,182655.000,A,4714.9182,N,00601.2047,E,1.85,58.42,090709,,*3C
$GPGGA,182656.000,471.9189,N,00601.2055,E,17,2.1,244.8,M,48.0,M,,0000*5A
$GPGSA,A,3,02,29,24,12,31,10,30,,,,,3.2,2.1,2.4*3A
$GPGSV,3,1,11,29,85,043,20,31,58,275,20,30,51,103,25,24,48,127,25*7E
$GPGSV,3,2,11,21,28,172,,12,21,105,24,02,13,039,30,16,12,291,*70
$GPGSV,3,3,11,23,05,334,,10,05,078,28,14,02,219,*45
$GPRMC,182656.000,A,4714.9189,N,00601.2055,E,1.28,39.53,090709,,*37
$GPGGA,182657.000,4714.9195,N,00601.2057,E,1,07,2.1,245.3,M,48.0,M,,0000*5E
$GPGSA,A,3,02,29,24,12,31,10,30,,,,,3.2,2.1,2.4*3A
$GPRMC,182657.000,A,4714.9195,N,00601.20,E,1.13,18.42,090709,,*32
$GPGGA,182658.000,4714.9202,N,00601.2061,E,1,07,2.1,246.8,M,48.0,M,,0000*51
$GPGSA,A,3,02,29,24,12,31,10,30,,,,,3.2,2.1,2.4*3A
$GPRMC,182658.000,A,4714.9202,N,00601.2061,E,1.58,41.94,090709,,*3D
$GPGGA,182659.000,4714.9206,N,00601.2065,E,1,07,2.1,247.5,M,48.0,M,,0000*5C
$GPGSA,A,3,02,29,24,12,31,10,30,,,,,3.2,2.1,2.4*3A
$GPRMC,182659.000,A,4714.9206,N,00601.2065,E,1.36,44.79,090709,,*32
$GPGGA,182700.000,4714.9216,N,00601.2072,E,1,07,2.1,250.0,M,48.0,M,,0000*55
```

Proposer la commande qui ne garde que les lignes contenant GPGGA. Étendre la commande pour ne garder que les colonnes contenant l'horaire, la latitude et la longitude.

Les diverses commandes que nous étudions prennent pour argument un nom de fichier sur lequel appliquer le traitement. Un nom de fichier particulier est l'entrée standard – *stdin* – qui est le choix par défaut si aucun nom de fichier n'est fourni. L'entrée standard peut être connectée au clavier, et dans ce cas le traitement se fait sur les lettre tappées par l'utilisateur, une fonction peu utile. Plus intéressant, il est possible de connecter l'entrée standard à la sorte – *stdout* – de la commande précédente, par le tuyau (*pipe*) symbolisé par le caractère `|`. Ainsi, `dmesg | grep sd` propose toutes les opérations sur les supports de stockage au format compatible SCSI dans les messages du système

```
[ 1565.340777] sd 0:0:0:0: [sda]
[ 1565.340790] sd 0:0:0:0: [sda] CDB:
[ 1565.340813] end_request: I/O error, dev sda, sector 13813373
[ 1569.028648] sd 0:0:0:0: [sda] Unhandled sense code
[ 1569.028655] sd 0:0:0:0: [sda]
[ 1569.028664] sd 0:0:0:0: [sda]
[ 1569.028712] sd 0:0:0:0: [sda]
[ 1569.028725] sd 0:0:0:0: [sda] CDB:
[ 1569.028747] end_request: I/O error, dev sda, sector 13813373
```

Le *pipe* est une fonction puissance qui permet de cascader les opérations, un point clé dans le traitement de chaînes de caractère tel que nous le verrons ci-dessous.

2.6 head et tail

L'affichage de `dmesg` prend beaucoup de temps dans un terminal car tous les messages depuis le démarrage de l'ordinateur sont affichés – un message très long lorsque l'ordinateur n'a pas été éteint depuis plusieurs mois. Nous ne nous intéressons généralement qu'aux dernières informations les plus récentes, donc la find du fichier `/var/log/messages`.

Nous pouvons couper une/plusieurs lignes d'une fichier à partir du début ou de la fin avec `head` et `tail` respectivement.

Application : `dmesg | tail`

Afficher les 3 dernières lignes de la séquence NMEA proposée auparavant. Les trois premières.

2.7 regexp

Les arguments des commandes que nous avons vu auparavant, et en particulier `grep`, peuvent être de simples chaînes des caractères, ou des motifs décrivant des classes de chînes de caractères. Ainsi, `[a-z]` indique toutes les lettres minuscules tandis que `[a-zA-Z]` décrit tout l'alphabet. Certains caractères spéciaux indiquent une position dans la ligne : `^` indique le début de ligne, et `$` le fin de ligne. Le symbole `.` indique "n'importe quel caractère".

Étendre la commande de la section 2.5 pour ne garder que les informations valides et non-corrmpues, à savoir une partie entière de la latitude contenant 4 chiffres. Modifier la commande pour garantir que tous ces caractères sont bien des chiffres.

3 Programmation en langage C

3.1 Hello World

1. Écrivez dans l'éditeur « `geany` » un programme « `hello.c` » qui affiche « `hello` ».
2. Compilez le programme dans un terminal avec la commande « `gcc -Wall -o hello hello.c` ».
3. Corrigez les erreurs de syntaxe et recompilez.
4. Exécutez le programme avec la commande « `./hello` »

3.2 Hello World (printf/scanf)

Écrivez un programme permettant de demander l'âge de l'utilisateur et de l'afficher

3.3 Bibliothèque math.h

Écrire un programme qui va demander la partie réelle et la partie imaginaire d'un complexe de la forme $a + jb$. Vous écrirez une fonction qui permet de calculer le module de ce nombre complexe ainsi qu'une fonction qui écrira son argument et ainsi affichera sa représentation polaire de la forme $\rho e^{(j\theta)}$.

3.4 Tableau

— Écrire un programme qui demande la taille d'un tableau d'entiers (on aura déclaré au préalable un tableau d'entiers de 50 cases et s'assurera que la taille demandée est inférieure à cette limite de 50), puis remplir, case par case, et afficher le contenu du tableau ainsi que la somme de tous les éléments.

3.5 Lecture écriture dans un fichier

À l'aide d'un éditeur de textes, créer un fichier `nombres.txt` qui contient une liste de nombres entiers. Dans le fichier, chaque nombre doit être suivi par un retour à la ligne. Écrire un programme qui affiche les nombres du fichier, leur somme et leur moyenne.