Sujet de TP - Recherche d'informations

Remarques préliminaires

Les remarques constructives relatives à ce support sont à transmettre à `clot@univ-lyon1.fr`.

Sous la titre de «recherche d'informations», nous entendons désigner dans un premier temps la recherche à partir des fichiers d'aide installés sur le système et dans un second temps la recherche d'information relative à des fichiers, que ce soit sur leurs attributs ou sur leur contenu.

La première partie concernera donc les commandes incontournables que sont `man`, `info`, et quelques autres pointeurs (option `-h`, `--help`, fichiers rfc, howto et `/usr/share/doc`...). La seconde partie s'intéressera à `grep`, `sed` puis `find`.

Aide à la lecture des sujets

Les supports de cet enseignement mettent en évidence les commandes ou noms de commande par le recours à une police de type `courier`. Ceci permet de distinguer facilement les éléments à saisir au clavier. Cette police introduit cependant quelques caractères dont la ressemblance peut prêter à confusion. Certaines touches sont également symbolisées sous la forme `<touche>`. Ainsi `<space>`, `<Up>` ou encore `<Tab>` représentent respectivement la barre d'espace, la flèche pointant vers le haut et la touche de tabulation.

Certains symboles apparaissent parfois, résultant seulement du formatage du sujet par LaTeX et ne devant pas être interprétés comme un caractère faisant partie de la commande.

`O`	lettre o majuscule
`0`	chiffre 0
`1`	chiffre 1
`l`	lettre L minuscule
`I`	lettre i majuscule
`|`	caractère "pipeline"
¬	caractère ne faisant pas partie du texte et provoqué par une coupure de ligne lors du formatage du document

Le tableau ci-contre précise quelques caractères afin de lever d'éventuelles ambiguïtés dans ce qui suit.

À l'aide !

La principale source d'aide sur les commandes installées sur un système unix est un ensemble de pages organisées en sections, l'ensemble formant le manuel. La commande `man` permet de rechercher les pages en lien avec une information et d'accéder au contenu d'une page dont la référence est connue. Lorsqu'une page est consultée avec la commande `man`, l'affichage est implicitement délégué à un programme particulier : un visualiseur - *pager* en anglais - capable de gérer le format particulier d'une page du manuel pour permettre son affichage. L'utilisateur peut choisir, par le biais d'une variable d'environnement, le programme utilisé pour visualiser le manuel.

Visualiseurs de fichiers

La commande `man` utilise le *pager* indiqué par la variable d'environnement `MANPAGER` (resp. `PAGER` sur les vieux systèmes). Le *pager* le plus courant est `less`. Des pagers spécialisés existent pour les fichiers compressés. Cf. page 9.

- Ouvrez une session dans une console ou un `xterm` et exécutez la commande `less --help`. Sachant que les flèches permettent de se déplacer et que la touche `Q` permet de sortir de cette page d'aide, identifiez les touches permettant d'afficher la page suivante (resp. précédente), d'afficher la demi-page suivante (resp. précédente), d'avancer (resp. reculer) d'une ligne, de repérer rapidement une chaîne de caractères, et d'exécuter une commande extérieure.
- Sortez de l'aide précédente et exécutez la commande `man date`. Recherchez la chaîne `^EXAM` et testez l'un des exemples directement à partir de la page du manuel (en utilisant la possibilité d'exécuter une commande extérieure). Sortez de la page du manuel.
- Testez si la variable `MANPAGER` existe en affichant sa valeur à l'aide de la commande `echo $¬ MANPAGER`.
- Affectez cette variable de la valeur `cat` avec la commande `export MANPAGER=cat`, puis exécutez la commande `man date` pour observer les effets de ce nouveau paramétrage. Quels sont-ils ? Supprimez ce paramétrage avec la commande `unset MANPAGER`.
- `less` est lui-même une amélioration de `more`, commande devenue rare de nos jours. Lancez la commande `man less` afin d'afficher le manuel de la commande `less`. Quels aspects vous semblent importants pour distinguer `less` de `more`.

RTFM [1] man

Beaucoup d'informations relatives aux commandes, aux appels systèmes, aux fichiers spéciaux etc, sont rassemblées dans les pages man. Les pages du manuel sont classées en sections (indiquées ci-dessous) et sont structurées pour permettre d'identifier rapidement divers types d'information en fonction de la nature de la «cible» : pour une commande, après un bref descriptif, les options et les bugs, des pointeurs vers d'autres pages sont indiqués. Pour des appels systèmes, les arguments, les codes d'erreurs et de retour sont détaillés...

```
The table below shows the section numbers of the manual followed by the types of pages
they contain.

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 conventions), e.g. man(7), groff(7)
8  System administration commands (usually only for root)
9  Kernel routines [Non standard]
```

Supposons que des précisions soient nécessaires pour la fonction `read` du langage `C`. La commande `man read` renvoie bien une page du manuel, mais sans les informations attendues car `read`, par ailleurs, est une commande interne du bash.

Sans indication de section, `man` renvoie la première page rencontrée dans l'ordre des sections. Ainsi, `man read` renvoie la page du bash et non celle de la section 2 (qui est celle contenant les informations attendues). Il est possible de préciser la section voulue à la commande `man` (e.g. `man 2 read`).

L'option `-k` de la commande `man` permet de chercher dans l'ensemble des sections les pages man associées à un mot clef, ce mot clé pouvant soit être une (partie d'une) entrée dans la base des pages, soit apparaître dans la courte description d'une page. Ainsi `man -k read` produit une sortie de plus de 300 lignes :

```
pthread_cleanup_push (3thr) - Installer et enlever des gestionnaires de nettoyage.
pthread_condattr_init (3thr) - attributs de création de conditions
...
gconfigger (1)       - Tool to change/read GNOME configuration entry.
```

1. read the ☻ manual !

Chaque ligne comporte la séquence read. Quelques aménagements permettent de rendre les choses plus lisibles. Nous allons transmettre ces lignes à la commande `sort` qui les triera et les lignes produites seront envoyées à `less` afin de les visualiser :
`> man -k read | sort -u | less`
Sélectionner les seules lignes commençant par read permettra de réellement réduire les lignes aux meilleures candidates :
`> man -k read | sort -u | grep ^read | less`
ou encore mieux :
`> man -k read | sort -u | grep ^read\␣ | less`
Sur le serveur `minisfa`, ces deux dernières commandes sont infructueuses car aucune ligne ne comporte `read` comme premier mot. Il faut plus de travail :
`> man -k read | sed -e 's/-.*$//' | gsed -e 's/ /\n/g'` Notez bien que la commande `gsed` n'est pas version native de `sed` sur les systèmes BSD mais une version gnu utilisant la même syntaxe que sur les systèmes linux.

Tout ceci pour finalement réussir à identifier la page de la section 2 associée à read : `man 2 read`

- Utilisez la commande `man` pour identifier les pages man relatives au processus `init`.
- Identifiez des commandes fortement liées à `man`.
- Quelle structure de présentation suivent les pages du manuel de la section 2 ? Illustrez votre réponse avec deux entrées de cette section. Faites de même pour la section 1 ? Sections 5 et 8 ?

Un autre système de documentation très développé regroupe des manuels en ligne pour de nombreuses commandes. Ce système est basé sur un format de fichier d'aide completement différent des pages man, organisant l'accès à l'information à la façon d'une documentation HTML comportant des liens pointant vers d'autres informations.

Commande `info`

Comme indiqué précédemment, l'ensemble des fichiers infos est organisé en une arborescence qu'il est possible de parcourir à partir de la commande `info`, visualiseur dédié à l'affichage de pages respectant ce format.

- Exécutez la commande `info info` afin d'afficher la page info sur la commande `info`. Cette entrée dans les pages infos décrit les principes d'interaction avec ce visualiseur. A partir des informations données, donnez un sens aux commandes suivantes :

`n`	`<Pg-Down>`	`m`	`<tab>`	`t`
`p`	`<Ctrl-L>`	`m+<tab>`	`f`	`q`
`<space>`	`b`	`m+?`	`l`	
`<del>`	`?`	`<Ctrl-g>`	`f+?`	
`<Pg-Up>`	`<Ctrl-x>+0`	`u`	`d`	

- Sur quels concepts repose la struture d'un ensemble de pages relatives à une commande ?
- Quelle est la structure des pages info de la commande `tar` ?
- Quelle est la structure des pages info de la commande `ifconfig` ?

Flags `-h` et `--help`

Que faire dans le cas d'une commande pour laquelle ni page man ni page info existe ? Certaines commandes disposent d'une option `-h` ou `--help` permettant d'obtenir un peu d'aide.

- Testez ces flags avec les commandes `sh`, `mv`, `echo`, `ifconfig`, `lp` et `mail`. Quelles limites présente cette méthode ?

Les informations proposées par le manuel et la documentation `info` peuvent être en déphasage par rapport aux commandes installées. Cela peut arriver lorsqu'une version obsolète de commande au sein d'une distribution est identifiée et qu'une opération de mise à jour est entreprise dans la hâte : l'oubli de la mise à jour de la documentation accompagnant la nouvelle version est classique...
L'avantage procuré par ce moyen d'information est que l'utilisateur est certain d'obtenir une information directement en relation avec la commande utilisée.

Autres sources d'informations

— Il est classique de trouver de la documentation pour les programmes installés dans le répertoire `/usr/share/doc`, `/usr/share/doc/packages` ou dans `/usr/share/PROGNAME/doc` pour un programme nommé `PROGNAME`. Sur `minisfa`, consulter également `/opt/local/share/doc`.

Identifiez la localisation des fichiers d'aide pour openssl. Pour ghostscript. Pour Git.

— Il est classique de trouver sur un système un regroupement de HowTos relatifs à divers sujets. Il s'agit en général de tutoriaux sur des sujets précis, plutôt que des indications sur des commandes précises. Le panel des thèmes est large :

- LDP HOWTO-INDEX
- The Printing HOWTO
- Linux Cluster
- IP Sub-Networking Mini-Howto
- Wireless Howto
- DHCP mini-HOWTO
- Network Install HOWTO
- ...
- Deciding if Linux is Right for You
- Encourage Women in Linux (? !)

Les howtos ne sont pas installés sur `minisfa`, mais le lien `http://tldp.org/HOWTO/text` permettra aux curieux de les parcourir.

— Les *Requests for Comment* (RFC) constituent une source considérable d'information sur les standards officiels de l'internet. Les rfcs sont classées en catégories, et ces catégories sont définies dans la `rfc1000.txt`. Par ailleurs, `rfc-index.txt` donnent des indications pour une rfc sur les rfcs rendues obsolètes et rendant obsolètes par cette rfc.
Plus loin, nous considérerons des moyens simples de recherche dans les rfcs.

Expressions régulières - *regular expressions*

Divers outils permettent de chercher des chaînes de caractères dans un ensemble de lignes formant un ou plusieurs fichiers ou provenant de l'entrée standard. Les chaînes cherchées sont décrites par un modèle ou patron. Ce mécanisme est familier à tout utilisateur interagissant avec le shell pour la désignation de fichier : les caractères `*` et `?` représentent respectivement toute chaîne de caractères et tout caractère. Les expressions régulières permettent de définir des formes de chaînes avec une grande souplesse. Elles sont utilisées dans `(e)grep`, `sed` et `awk` qui seront présentés plus loin, mais aussi dans `ex` et `vi`. Leur construction, basée sur des méta-caractères, présente quelques variantes selon le programme employé.

Métacaractères

Ci-dessous, les différents métacaractères employés pour la construction des expressions régulières.

Symbol	ed	ex	vi	sed	awk	grep	egrep	Action
`.`	☺	☺	☺	☺	☺	☺	☺	Match any character.
`*`	☺	☺	☺	☺	☺	☺	☺	Match zero or more preceding.
`^`	☺	☺	☺	☺	☺	☺	☺	Match beginning of line/string.
`$`	☺	☺	☺	☺	☺	☺	☺	Match end of line/string.
`\`	☺	☺	☺	☺	☺	☺	☺	Escape following character.
`[ ]`	☺	☺	☺	☺	☺	☺	☺	Match one from a set.
`\( \)`	☺	☺	☺	☺		☺		Store pattern for later replay.[1]
`\n`	☺	☺	☺	☺		☺		Replay subpattern in match.
`{ }`					☺		☺	Match a range of instances.
`\{ \}`	☺			☺		☺		Match a range of instances.
`\< \>`	☺	☺	☺					Match word's beginning or end.
`+`					☺		☺	Match one or more preceding.
`?`					☺		☺	Match zero or one preceding.
`|`					☺		☺	Separate choices to match.
`( )`					☺		☺	Group expressions to match.

Les items suivants définissent des classes de caractères pouvant apparaitre entre `[` et `]` :

`[:blank:]` espace ou tabulation
`[:digit:]` un chiffre
`[:alpha:]` une lettre
`[:alnum:]` un caractère alphanumérique
`[:punct:]` un caractère de ponctuation
`[:cntrl:]` un caractère de contrôle
`[:graph:]` un caractère imprimable et visible
`[:lower:]` un caractère minuscule
`[:upper:]` un caractère majuscule
`[:space:]` un caractère d'espacement (space, tab, formfeed...)
`[:print:]` un caractère imprimable
`[:xdigit:]` un caractère héxadécimal

Ci-dessous, quelques exemples :

`^$` ligne vide
`^..*$` ligne non vide
`\$` $ himself
`\.` . himself
`[abc]` a ou b ou c
`[b-v]` une lettre comprise entre b et v
`[A-F]` ... entre A et F
`[b-v0-3]` un caractère entre b et v ou entre 0 et 3
`\n` end of line
`\t` <Tab>
`\e` <Echap>
`\b` <BS>
`\r` <CR>
`[-+]` + ou −
`[^b-v0-3]` le complémentaire de ci-dessus
`\>` fin de mot
`\([a-z][a-z]*\) \1` une chaîne de caractères minuscules répétée et séparée de sa répétition par un espace
`[[:alpha:]]` une lettre
`[[:upper:]12][[:lower:]]` une majuscule ou 1 ou 2, puis une minuscule

Comment interprétez-vous les expressions suivantes (Attention, ce ne sont pas des commandes à saisir pour l'instant ! Elles serviront plutard comme motifs donnés en arguments de `sed`, `grep`.) sachant que l'interprétation dépend du programme utilisé :

1. `^[[:space:]]*$`
2. `^\t\t*`
3. `^\t+`
4. `[^[:alnum:]][Nn]0[0-9]`
5. `\([X-Y]\)\([0-4]\)\$\2\1`
6. `[a-zA-Z][[:alnum:]]\{1,7\}\.[Tt][Xx¬][Tt]`
7. `^[^#]`
8. `\[SiMpLe\$\\\]`

Comment interpréteront `awk` et `egrep` les expressions suivantes

1. `^#|#$`
2. `\\$\n[^#]`
3. `\\$\n[^[:blank:]]+`
4. `^[^=]*$|^(={2,}|[^=])*={2,}(={2,}|[^=])*$`

Chercher. . .ou filtrer des lignes

Nous introduisons ici `grep`, `e(xtended)grep` et `sed`. Les deux premières permettent de filtrer les lignes fournies en entrée qui correpondent à un patron alors que la suivante permet d'appliquer un traitement sur les lignes lues.

La famille des `grep`

`grep` : ce drôle de nom proviendrait d'une commande de l'éditeur `ed` : `g/re/p` où `re` symbolise l'expression régulière. Cette commande permet de sélectionner (`g`) toutes les lignes correspondant avec `re` et les imprime (`p`).
`egrep` : extended grep, utilise un modèle d'expression régulière étendu
`fgrep` : fast grep, le bien mal nommé. . .

Les commandes `grep` permettent d'afficher une information par rapport aux lignes correspondant à un patron donné. La syntaxe basique est la suivante : `grep [option] patron [fichier...]` Sans option particulière, les lignes correspondant au patron sont affichées. Diverses options permettent d'engendrer un output particulier :

`-c` permet d'afficher le nombre de ligne concordant
`-i` permet d'inhiber la sensibilité à la casse
`-n` permet d'afficher la ligne précédée de son numéro
`-v` permet d'afficher les lignes ne concordant pas avec le patron
`-m #` permet d'afficher les # premières concordances
`-l` affiche les noms des fichiers au lieu des lignes
`-L` affiche les noms des fichiers n'ayant aucune concordance
`-r` ...
La page du manuel détaille l'ensemble des options.

Dernière remarque : une expression régulière peut contenir des caractères sujet à interprétation par le shell (`*`, `|`...). Il est donc important de protéger un patron des mécanismes du shell afin qu'il soit transmis sans transformation à `grep` ! Vous réalisez cette protection en encadrant chaque patron par une paire de simples quotes (`'`).

- Sélectionnez dans `/var/log/boot.msg` les lignes contenant `/dev`. Si ce fichier n'existe pas, filtrez les lignes produites par la commande `dmesg` (ou la commande `syslog` sur `minisfa`).
- Proposez une commande permettant de repérer les occurences de la commande `ifconfig` dans les fichiers présents dans `/etc/`. Positionnez l'option nécessaire pour la suppression des messages d'erreur.
- Proposez à présent une commande permettant de repérer les occurences de l'adresse ip d'un serveur DNS du campus (134.214.100.9 ou 134.214.100.245) dans les fichiers présents dans `/etc`.
- Proposez à présent une commande permettant de repérer les occurences de toute adresse ip dans les fichiers présents dans `/etc/`.
- Proposez une commande permettant d'afficher toutes les entrées du manuel en relation avec le mot clef `time` et filtrez celles-ci afin de ne retenir que celles pour lesquelle le nom de la commande contient la chaîne `time`.
- Proposez un filtre pour la commande `ls -l /tmp` (ou `ls -lH /tmp` sur `minisfa`) permettant de retenir les lignes contenant la date du jour ou votre login.

 Les lignes selectionnées ci-dessus contiennent potentiellement plus que les fichiers dont vous êtes propriétaire ou modifiés le jour même. Proposez un nouveau filtre permettant de retenir strictement ces derniers.

 Nous allons nous interesser à la rfc n°1000. Un fichier nommé `rfc1000.txt.gz` devrait exister dans le répertoire `/usr/share/doc/rfc` sur l'hôte 195.220.111.226. Observez ce fichier à l'aide de la commande `zless` et repérez les lignes détaillant les catégories et sous-catégories. Combinez les commandes `gzcat` et `grep` en un filtre pour retenir ces lignes contenant les intitulés des catégories et sous-catégories. Vous pourrez aussi essayer la commande `zgrep` au lieu de la combinaison précédente.

 Proposez un filtre qui permet d'identifier les rfcs rendant obsolète une rfc d'un numéro donné (e.g. 84 ou 1084). Notez que pour se simplifier la tâche, nous supposerons que les numéros des rfcs indiqués se trouvent regroupés sur une seule ligne.

`sed`, the stream editor

`sed` est un éditeur de ligne. Les lignes qu'il reçoit en entrée sont traitées selon les commandes passées. Il est important de souligner deux points :

— `sed` ne change pas le fichier dont les lignes sont lues[2]. Il lit des lignes en entrée, les transforme et les envoie vers sa sortie standard.

— `sed` applique les commandes à toutes les lignes lues, sauf si une sélection des lignes a lieu.

Les syntaxes possibles d'invocation de `sed` sont les suivantes :

— `sed [option] [-e 'command'] [file(s)]`

— `sed [option] -f scriptfile [file(s)]`

Nous nous restreindrons à une présentation de la première syntaxe. La syntaxe pour la partie `command` est la suivante : `[address[,address]][!]command [arguments]`. La partie `[address[,address]]` permet de définir une sélection des lignes à traiter. Le caractère `!` permet d'inverser cette sélection. Par exemple :

`1,10` lignes 1 à 10	`/ *if /,/ *fi/` tous les blocs encadrés par `if` et `fi`
`$` dernière ligne	`/^$/!` toutes les lignes non vides
`truc` toutes les lignes contenant `truc`	

La partie `command` peut être simple ou composée. Un extrait des commandes simples est proposé ci-dessous :

`d` efface	`s/p1/p2/4` substitue la 4ème occurence de `p1`
`cTEXT` remplace la ligne par `TEXT`	`s/p1/p2/g` substitue toutes les occurences de `p1`
`iTEXT` insert `TEXT` avant la ligne	`q` quitte
`s/p1/p2/` substitue la 1ère occurence de `p1` par `p2`	

Par exemple :
`/^./!d` efface les lignes vides
`/^$/d` efface les lignes vides
`XXq` quitte après la première ligne contenant `XX`
`s/^\([^ ][^ ]*\) \([^ ][^ ]*\)/\2 \1/` intervertit les deux premiers mots
`s/;/;\n/g` substitue toutes les occurences de `;` par `;\n`

Plusieurs commandes peuvent être appliquées aux lignes correspondant à une sélection comme suit :
`sed -e '/^[[:blank:]]/s/^[[:blank:]]*//;s/$/ # ligne raccourcie/'` ou
`sed -e '/^[[:blank:]]/{s/^[[:blank:]]*//}`
`s/$/ # ligne raccourcie/'`
Par ailleurs, plusieurs commandes peuvent être enchaînées comme suit :

2. Aucun changement n'est fait sur le fichier d'origine implicitement, mais cela peut-être modifié avec l'option `-i`, et ceci permet des traitements très intéressants !

`sed -e '/^[[:blank:]]$/d' -e '/^[[:blank:]]*#/d'`
Ces constructions sont des exemples de commandes composées évoquées plus haut.

- Reprenez le dernier problème considéré relatif aux rfcs sans supposer que les numeros des rfcs obsolètes sont regroupés sur une seule ligne. Vous pourrez observer les fichiers `rfc1020.txt.gz` et `rfc1010.txt.gz` afin d'observer une illustration de ce cas.
- Proposez un script permettant de sélectionner les lignes comportant un `=` isolé dans un bloc `if¬ ,then...,fi`. Vous pourrez appliquer ce filtre à la commande suivante

```
 echo -e "if == then = fi \navant == \navant = \nif ...\nthen...\nfi\nla suite = \n...if
=...\nthen.==..\nfi"
```

- Certaines lignes du fichier http://isfaserveur.univ-lyon1.fr/~denis.clot/IRM2.html comporte le tag `href`. Proposez un filtre permettant de sélectionner ces lignes. La commande `wget` sera utilisée pour télécharger cette url et envoyer son contenu vers la console (consultez la manuel !).
- Nous souhaitons améliorer ce filtre afin qu'il isole le tag `href` et le lien indiqué. Ecrire un filtre supprimmant tous les caractères précédant `href` et supprimant tous les caractères après la référence indiquée, sachant que celle-ci est encadrée par des `"`.
- Ayant remarqué que cette solution engendre des pertes dans le cas où plusieurs `href` se trouvent sur la même ligne, proposez un autre filtre qui ne perd aucun `href`.
- Quelle information donne la commande `last` ? Consultez le manuel...Proposez une commande permettant de transformer les lignes renvoyées par la commande `last` en ne conservant de ces lignes que le login et l'hôte par lequel la demande de connexion est parvenue.
- Proposez un filtre permettant de sélectionner parmi les lignes renvoyées par `netstat -tn` sur linux (`netstat -np tcp` sur `minisfa`) les lignes contenant la ou les IPs des interfaces réseaux et de retenir de ces lignes les seuls champs des adresses locales et distantes.

Filtrer des fichiers...

Comment chercher un fichier à partir d'un point de l'arborescence vérifiant certains critères ? Maintenant que `grep` vous est familier, il est toujours possible d'imaginer un filtre pour les lignes produites par la commande `ls -Rl /a/partir/d/ici` ou quelque chose d'apparenté. Il existe toutefois une commande simple et bien plus puissante pour ce genre de recherche : `find`.

`find` permet de rechercher tous les fichiers satisfaisant un ensemble de critères à partir d'un ou plusieurs points de l'arborescence et d'effectuer des opérations sur les fichiers identifiés.

La syntaxe de `find` diffère entre les systèmes Linux et BSD. Sur `Linux`, tout est optionnel (`find [chemin...] [option] [test] [action]`) alors que la version BSD (`find [option] chemin [test] [¬ action]`) impose la spécification d'un chemin. Par ailleurs Les arguments sont les suivants :

- `chemin` est une liste de chemins, chacun étant séparé du suivant par une espace. Pour la version de `find` utilisée sur `minisfa`, `chemin` est nécessaire, contrairement à la version `Linux` pour laquelle le chemin implicite est le réperoitre courant.
- `option` est une liste d'options à consulter par les curieux dans le manuel.
- `test` est un test ou une chaîne de tests. Les tests peuvent être composés à l'aide des opérateurs `( )`, `!`, `-a`, `-o`. L'opérateur appliqué implicitement à deux tests successifs est le ET logique. Les tests simples sont les suivants :
  - `-name PATRON` recherche les fichiers dont le nom correspond au patron, ce patron pouvant être construit avec les caractères `*`, `?`, `[]`
  - `-iname PATRON` idem, mais insensible à la casse
  - `-user USERNAME` recherche les fichiers dont le propriétaire est `USERNAME`
  - `-group GROUPNAME` recherche les fichiers dont le groupe est `GROUPNAME`
  - `-type X` recherche les fichiers dont le type est `X` (`f` pour fichier, `d` pour répertoire, `l` pour lien, etc)

`-perm PERM` recherche les fichiers dont le mode correspond exactement à `PERM`
`-perm -PERM` recherche les fichiers dont le mode est au moins `PERM`
`-amin` (resp. `atime`) `n` (resp. `+n` ou `-n`) recherche les fichiers ayant été accédés il y a `n` (resp. plus de ou moins) minutes (resp. `n`*24 heures)
`-cmin` (resp. `ctime`) `n` les temps considérés sont ceux de dernière modification de status
`-mmin` (resp. `mtime`) `n` les temps considérés sont ceux de dernière modification du contenu
...D'autres tests existent. Cf le manuel.

— `action` permet d'agir sur un fichier ayant satisfait les tests définis avant dans la commande. Une action est également un test, car `find` utilise le code retour de l'action et peut s'en servir pour conditionner une action suivante...
`-exec Truc -argTruc {} \;` execute la commande `Truc` avec les arguments `-argTruc` et comme dernier argument qui représente le nom du fichier satisfaisant les tests
`-ok Truc -argTruc {} \;` idem, mais demande la confirmation de l'utilisateur avant l'exécution de la commande `Truc`
`-print` imprime le nom du fichier s'il satisfait les tests ; c'est l'action par défaut
`-printf FORMAT` imprime une chaîne selon `FORMAT` dérivant du langage C (`man 3 printf`)
...D'autres actions existent. Consultez le manuel du système utilisé.

Ainsi, la commande `find . -type f -user toto -exec egrep -q '@gmail.com' {} ';' -exec file {} ';'` permet d'identifier les fichiers dont le propriétaire est toto, contenant la chaîne `@gmail.com` en indiquant la nature de ces fichier.

La syntaxe présentée est abusive dans le sens où l'ordre `[test][action]` n'existe pas. Les actions ne sont que des tests un peu particuliers et tous ces tests peuvent se mélanger.

Pour finir, notez que la commande `gfind` est installée sur `minisfa`, version gnu de la commande `find` qui permet d'utiliser la syntaxe des systèmes Linux.

 Construire une commande permettant d'identifier, à partir de votre répertoire personnel, tous les répertoires contenant des fichiers d'extension `.tex`. Consultez le manuel pour connaitre les détails de l'action `-printf`.

 Comment déterminer les fichiers de votre répertoire personnel qui sont exécutables par vous, dont le propriétaire est vous-même ou `root` et qui ont été modifiés il y a au plus trois jours ? (à reprendre après le TP portant sur le partage !)

 Comment lister les fichiers `tar` qui contiennent des fichiers `.tex` ? Soulignons que l'option `-t` de la commande `tar` permet de lister le contenu de l'archive passée en argument.

 Comment lister les fichiers exécutables pointés par les liens d'un répertoire donné ?

 Comment supprimer (avec confirmation) les fichiers du répertoire `/tmp` dont vous êtes propriétaire et qui n'ont été ni modifiés ni accédés depuis plus d'un mois ?

 Proposez une commande qui recherche toutes les entrées à partir de votre home qui sont des fichiers exécutables (mais pas des liens), et qui produit une sortie formatée donnant pour chaque entrée le nom du fichier, sa date de modification de contenu et le répertoire où il a été trouvé.

 Proposez une commande qui liste les fichiers correspondant à des images (test avec la commande `file`) suivant le format suivant : "nom du fichier - repertoire"

Commandes annexes

 Prenez connaissance de chacune des commandes ci-dessous

1. `cat`
2. `tac`
3. `zcat`
4. `bzcat`
5. `zgrep`
6. `cut`
7. `paste`
8. `head`
9. `tail`


Proposez une commande permettant de lister les fichiers du répertoire `/usr/share/doc/rfc` contenant la chaîne `wifi` ou `wlan`, sans tenir compte de la casse des caractères.


A partir des commandes ci-dessus, proposez une composition permettant d'identifier les rfcs contenant un mot clef et affichant pour chacune son numéro et son titre. Rappellons que le fichier `rfc-index.txt.gz` contient ces deux dernières informations.


A partir des commandes ci-dessus, proposez une composition permettant d'identifier les archives tar gzipées contenant des images (en utilisant la commande `file` pour reconnaitre les images et les archives gzipées (`man file`)) dans tous les répertoires de `/home`

Bilan

A l'issue de ce sujet, vous devriez :
— savoir utiliser `less` pour parcourir le contenu d'un fichier et y chercher une information ;
— savoir consulter le manuel ;
— connaître les expressions régulières simples et étendues ;
— maitrîser `grep` et `egrep` sur des fichiers et des répertoires ;
— savoir réaliser des traitements complexes avec `sed` ;
— savoir rechercher des fichiers satisfaisant des critères simples et éventuellement leur appliquer des traitements.