TP n°4: Fonctions avancées d'Unix

D. Pellier, D. Janiszek, J. Mauclair

15 août 2018

Sommaire

http://www.ens.math-info.univ-paris5.fr/cbi/

Objectifs :
- Découvrir et comprendre les expressions régulières usuelles
- Rechercher et manipuler des textes dans un ensemble de fichiers
- Maîtriser la priorité d'un processus
- Se connecter à distance

Éléments techniques abordés :
- *Les commandes*
  - head, tail, cut, sort
  - grep, sed
  - top, nice, renice
  - hostname, ping, traceroute

1 Recherche et manipulation de textes

Les outils de base

Pour afficher l'entête d'un fichier vous pouvez utiliser la commande `head`. Par défaut, cette commande affiche les 10 premières lignes d'un fichier. Si vous souhaitez spécifier un nombre de ligne, il faut utiliser l'option `-n`
- Pour obtenir les 5 premières lignes du fichier :

```
$ head -n 5 fichier
```

De la même manière, pour obtenir la fin d'un fichier, vous pouvez utiliser la commande `tail`. Par défaut, cette commande affiche les 10 dernières lignes d'un fichier. Si vous souhaitez spécifier un nombre de ligne, il faut utiliser l'option `-n`

— Pour obtenir la dernière ligne du fichier :

```
$ tail -n 1 fichier
```

— Pour obtenir le fichier à partir de la deuxième ligne

```
$ tail -n +2 fichier
```

Si aucun nom de fichier n'est spécifié, ces commandes traitent l'entrée standard.

— Les 3 écritures suivantes sont équivalentes :

```
$ cat fichier | tail -n 1
$ tail -n 1 fichier
$ tail -n 1 < fichier
```

La commande `cut` permet l'extraction de segments de ligne d'un fichier en entrée en procédant ligne par ligne. Les segments de ligne peuvent être des octets (-b), des caractères (-c) ou des champs (-f) ils sont séparés par des délimiteurs (-d). Pour délimiter le segment retenu, on doit préciser un intervalle. Le délimiteur par défaut est la tabulation.

La commande `sort` est une commande qui permet de trier des fichiers ou leurs contenus. L'option -u enlève les doublons.

Exemples Pour obtenir la liste des PID des processus actifs sur une machine, on conserve les 5 premiers caractères de chaque ligne :

```
$ ps -e | cut -c1-5
```

Pour obtenir la liste des noms des processus actifs sur une machine, on garde tous caractères au delà du 25ème caractère de chaque ligne :

```
$ ps -e | cut -c25-  | sort -u
```

http://www.ens-math-info.univ-paris5.fr/~bi/

1.1 Exercice 4.1

1. Déterminez le nombre d'utilisateurs ayant un compte sur votre machine à partir du fichier `/etc/passwd`
2. Déterminez la liste des utilisateurs ayant un compte sur votre machine à partir du fichier `/etc/passwd`
3. Quel est le 15ème nom ?
4. Déterminez le nombre de shells différents dans le fichier `/etc/passwd`
5. A partir du fichier `/etc/passwd`, générez le fichier contenant uniquement le login et le home-directory de chaque utilisateur.

1.2 Les expressions régulières

Une expression régulière est une manière compacte de décrire un motif complexe dans un texte. On utilise une expression régulière pour rechercher ou modifier un motif. Les expressions régulières peuvent être combinées grâce à différents opérateurs. Un motif correspond à un ensemble de chaines de caractères possibles.

Les expressions régulières sont composées de caractères ordinaires et de caractères spéciaux. Les méta-caractères ou caractères spéciaux ont une signification particulière ; il peut s'agir de raccourcis ou d'opérateurs.

En anglais, une expression régulière est une *regular expression*. Aussi, dans de nombreux textes vous pourrez trouver les abréviations : **regex** ou **regexp**.

Il existe 2 types d'expressions régulières :

— Les expressions régulières basiques (ERB)

— Les expressions régulières étendues (ERE)

Le tableau suivant dresse la liste des commandes utilisant des expressions régulières en fonction de leur type (basique ou étendue)

ER Basiques	ER Etendues
grep	grep -E
sed	sed -r
vi	awk
expr	

Les expressions régulières constituent un outil puissant et très expressif. C'est la raison pour laquelle l'écriture d'une expression régulière est une tâche considérée comme aussi complexe que la programmation. Il est toujours préférable de résoudre un problème simple d'une manière simple ; lorsque ce n'est pas possible, pensez aux expressions régulières ;-)

Les caractères

Le motif le plus simple correspondant à une expression régulière est un caractère ou une séquence de caractères. Lorsqu'on effectue une recherche, tous les caractères du texte cible se trouvant dans l'ordre exact à celui indiqué dans le motif correspondront. Un caractères est considéré comme une expression régulière concordant avec elle-même (sauf s'il s'agit d'un méta-caractères) Ainsi, une lettre minuscule n'est pas identique à sa version en majuscule, et inversement.

> Dans une expression régulière, l'espace correspond à un espace littéral dans la cible tandis que dans la plupart des langages de programmation ou interpréteurs en ligne de commande, c'est différent : l'espace y sert à séparer les mots clés.

> Dans la suite, les parties soulignées servent à expliciter les caractères qui ont permis la correspondance. Lorsque vous utiliserez une commande utilisant une expression régulière, ces caractères apparaîtront normalement.

https://www.ens.math.ufo-univ-paris5.fr/cbi/

Exemples

— Si on recherche l'expression régulière : /a/ dans le texte suivant, on trouve les éléments soulignés :

Il était un petit navire
Qui n'avait jamais navigué
Il partit pour un long voyage
Sur la mer Méditerrannée

— Si on recherche l'expression régulière : /un/ dans le texte suivant, on trouve les éléments soulignés :

Il était un petit navire
Qui n'avait jamais navigué
Il partit pour un long voyage
Sur la mer Méditerrannée

Les classes de caractères

Les expressions régulières permettent de mettre en correspondance un caractère appartenant à un ensemble. Cet ensemble est défini comme une classe de caractères. Une classe de caractères peut être donnée comme une simple liste de caractères entre crochets ([]). On peut définir un intervalle en plaçant un tiret (-) entre le premier

élément et le dernier élément de l'intervalle. Lorsqu'il s'agit de ne définir qu'un intervalle, on peut omettre les crochets mais **cette pratique est fortement déconseillée**.

— Exemples

Classe	Rôle
[aeiou]	Toute voyelle en minuscule
[a-m]	Intervalle : toute lettre comprise entre a et m (inclus)
[0123456789]	Tout chiffre
[0-9]	Intervalle : tout chiffre compris entre 0 et 9

Inversion Un accent circonflexe (^) placé au début de la liste permet d'inverser la correspondance. On peut alors le lire comme "une liste privé de ...". Ainsi, une liste dont le premier élément est un accent circonflexe correspond à n'importe quel caractère absent de la liste.

— Exemple

Classe	Rôle
[^0-9]	Tout caractère non numérique
[^a-zA-Z]	Tout caractère non alphabétique

Paramètres régionaux Les intervalles de caractères dépendent des paramètres régionaux de l'ordinateur Ainsi par défaut [a-d] est équivalent à [abcd]. Pourtant avec d'autres paramètres régionaux, les caractères peuvent être triés selon l'ordre lexicographique. Dans ce cas [a-d] est alors équivalent à [aBbCcDd].

Au cas où les paramètres régionaux auraient été modifiés sur votre machine, vous pouvez revenir au comportement par défaut en modifiant la variable d'environnement `LC_ALL` afin qu'elle prenne la valeur C.

```
$ export LC_ALL=C
```

https://www.ens.math-info.univ-paris5.fr/cbi/

Classes de caractères prédéfinies Il existe des classes de caractères prédéfinies ; contrairement à une liste définie par l'utilisateur elles présentent l'avantage de ne pas dépendre des paramètres régionaux ni du codage de caractères ASCII [1] (cf table ASCII). Leurs noms sont assez explicites.

— Les classes de caractères prédéfinies :

Liste	Rôle
[:alnum :]	Tout caractère alphanumérique (de a à z, de A à Z et de 0 à 9)
[:alpha :]	Tout caractère alphabétique (de a à z et de A à Z)
[:cntrl :]	Tout caractère de contrôle *
[:digit :]	Tout chiffre (de 0 à 9)
[:space :]	Tout espace (espace, tabulation, NL, FF, VT, CR)
[:graph :]	Sauf l'espace et le tabulation
[:print :]	Tout caractère affichable
[:punct :]	Tout caractère de ponctuation
[:upper :]	Tout caractère alphabétique en majuscule
[:lower :]	Tout caractère alphabétique en minuscule
[:xdigit :]	Tout chiffre hexadécimal (de 0 à 9 et de A à F)

Raccourcis De nombreux outils utilisant des expressions régulières fournissent également des raccourcis ou des synonymes pour les classes de caractères les plus couramment utilisés, tels que les caractères d'espacement et les

1. American Standard Code for Information Interchange

chiffres. Il est possible de définir ces classes de caractères avec des crochets, mais les raccourcis peuvent faire des expressions régulières plus compactes et plus lisibles.

Raccourci	Synonyme de	Rôle
{\}s	[:space :]	Tout caractère d'espacement
{\}d	[:digit :]	Tout chiffre
{\}v	[:alnum :]	Tout caractère alphanumérique
{\}W	[^[:alnum]]	Tout caractère non-alphanumérique

— Exemples
— Si on recherche l'expression régulière : /{\}s/ dans le texte suivant, on trouve tous les caractères d'espacement (en souligné) :

Il_était_un_petit_navire
Qui_n'avait_jamais_navigué
Il_partit_pour_un_long_voyage
Sur_la_mer_Méditerrannée

Les méta-caractères

Certains caractères, les méta-caractères, ont une signification spéciale.

Les méta-caractères de positionnement

Les limites de ligne Dans presque tous les outils utilisant des expressions régulières, il existe deux méta-caractères utilisés l'un, pour marquer le début et l'autre, la fin d'une ligne :

Méta-caractère	Nom usuel	Rôle
^	accent circonflexe	début de ligne
$	dollar	fin de ligne

> Les méta-caractères accent circonflexe (^) et dollar ($) perdent leur signification s'ils ne sont pas placés respectivement au début ou à la fin de l'expression régulière. Par ailleurs, on considère que ces méta-caractères correspondent à des motifs de longueur nulle. Autrement dit, la longueur de la chaîne trouvée par un accent circonflexe ou par un dollar est égale à zéro.

https://www.ons.mathinfo.univ-paris5.fr/cbi/

De nombreux outils d'expressions régulières acceptent d'autres motifs de longueur nulle : les limites de mot. En effet, les mots peuvent être séparés par des espaces, des tabulations, des retours à la ligne, etc. Le motif de limite de mot correspond à l'endroit même où un mot commence ou se termine, et ne prend pas en compte un caractère d'espacement particulier.

Les limites de mots

Expression régulière	Rôle
{\}<	chaîne vide en début de mot
{\}>	chaîne vide en fin de mot
{\}b	chaîne vide en limite de mot (début ou fin)
{\}B	chaine vide ne se trouvant pas à la limite d'un mot

— Exemples

Si on recherche l'expression régulière : /^il/ dans le texte suivant, on trouve les éléments soulignés : il était un petit navire qui n'avait jamais navigué, il partit pour un long voyage sur la mer Méditerrannée.

Si on recherche l'expression régulière : /navire$/ dans le texte suivant, on trouve les éléments soulignés : Il était un petit navire
Qui n'avait jamais navigué

Ce navire partit pour un long voyage
Sur la mer Méditerrannée

Le joker Dans les expressions régulières, un point (.) peut être mis en correspondance avec n'importe quel caractère ; par analogie aux jeux de cartes, l'appelle le joker. Normalement, le caractère de nouvelle ligne n'est pas inclus, mais la plupart des outils ont un commutateur facultatif qui force l'inclusion du caractère de nouvelle ligne. Utiliser un point dans un motif est le moyen d'exiger que «quelque chose» se produise à cet endroit, sans avoir à préciser quoi.

— Exemple

Si on recherche l'expression régulière : /.a/ dans le texte suivant, on trouve les éléments soulignés : Il était un petit navire
Qui n'avait jamais navigué
Il partit pour un long voyage
Sur la mer Méditerrannée

La protection

Les méta-caractères peuvent être mis en correspondance ; pour cela, il faut empêcher leur évaluation en les "protégeant". On protège un méta-caractère en le faisant précéder d'un anti-slash ({\}). Cela inclut le caractère anti-slash : pour mettre en correspondance un anti-slash dans la cible, l'expression régulière est la suivante : "{\}{\}" .

Exemples

— Si on recherche l'expression régulière : /.*/ dans le texte suivant, on trouve les éléments soulignés :
Il était un petit navire.*

— Si on recherche l'expression régulière : /{\}.{\}*/ dans le texte suivant, on trouve les éléments soulignés :
Il était un petit navire.*

> La plupart des méta-caractères perdent leur signification spéciale au sein des listes. Ainsi :
> — Pour inclure un caractère], il faut le mettre en premier dans la liste.
> — Pour inclure un caractère ^, il faut le placer n'importe où sauf au début de la liste.
> — Pour inclure un -, il faut le placer en dernier.

Le regroupement d'expressions régulières

Dans une expression régulière, chaque caractère, chaque motif de position, est considéré comme un atome. On peut regrouper plusieurs atomes pour former une petite expression régulière, qui elle même peut contribuer à la formation d'une expression régulière plus grande. On pourrait être enclin à appeler un tel regroupement une "molécule", mais en réalité, on l'appelle aussi un atome. Lorsqu'on observe une expression régulière constituée d'atomes, chaque atome peut être considéré comme une sous-expression. Pour faciliter la lecture d'une expression régulière complexe, on peut utiliser des parenthèses pour distinguer ses différentes sous-expressions.

Exemple

— Si on recherche l'expression régulière : /(petit)( )(navire)/ (qui en réalité équivaut à : /petit navire/) dans le texte suivant, on trouve les éléments soulignés :

Il était un petit navire
Qui n'avait jamais navigué
Ce petit navire partit pour un long voyage

https://www.ens.math-info.univ-paris5.fr/~doi/

Sur la mer Méditerrannée

Motifs alternatifs

Deux expressions régulières peuvent être reliées par l'opérateur infixe — ; l'expression résultante correspondra à toute chaîne concordant avec l'une ou l'autre des deux expressions.

L'utilisation de classes de caractères permet d'indiquer qu'un caractère ou un autre peut correspondre en un point particulier du texte cible. On peut étendre ce mécanisme aux sous-expressions. Ainsi, pour préciser que l'une des deux sous-expressions peut être mise en correspondance dans une expression régulière, on utilise l'opérateur d'alternance : la barre verticale ("—"). On le lit comme un "ou" (ou bien). C'est le symbole qui est également utilisé pour indiquer un pipe dans les shells Unix.

Pour bien maitriser la portée de l'alternative, il est recommandé de la mettre entre parenthèses.

* Exemple

Si on recherche l'expression régulière : /(un—navire—voyage)/ dans le texte suivant, on trouve les éléments soulignés : Il était un petit navire
Qui n'avait jamais navigué
Il partit pour un long voyage
Sur la mer Méditerrannée

http://www.ens.mathinfo.univ-paris5.fr/cbi/

Les opérateurs de répétition

L'une des fonctionnalités les plus puissantes et des plus courantes proposée par les expressions régulières est la possibilité de spécifier le nombre d'occurrence d'un atome dans une expression régulière complète. En effet, il est parfois souhaitable de pouvoir spécifier le nombre de correspondances d'un caractère, d'une classe de caractères ou d'une sous-expression. Pour utiliser un opérateur de répétition, on le place à la suite de l'atome concerné.

Par exemple, l'opérateur de répétition le plus courant est l'astérisque (*) ; on peut le lire : "aucune fois ou plusieurs" . Ainsi, lorsqu'on souhaite préciser qu'un atome peut correspondre aucune fois ou plusieurs fois dans une expression régulière, alors on fait suivre l'atome par une astérisque.

— Exemple

Si on recherche l'expression régulière : /[an]*/ dans le texte suivant, on trouve les éléments soulignés : Il était un petit navire
Qui n'avait jamais navigué
Il partit pour un long voyage
Sur la mer Méditerrannée

Liste des opérateurs de répétition

Opérateur	Rôle
?	L'élément précédent est facultatif et peut être rencontré au plus une fo
*	L'élément précédent peut être rencontré zéro ou plusieurs fois
+	L'élément précédent peut être rencontré une ou plusieurs fois
{n}	L'élément précédent doit être cherché exactement n fois
{n,}	L'élément précédent doit être cherché n fois ou plus
{n,m}	L'élément précédent doit être cherché au moins n fois, mais au plus m
{,m}	L'élément précédent doit être cherché au plus m fois

Deux expressions régulières peuvent être juxtaposées ; l'expression résultante correspondra à toute chaîne formée par la juxtaposition de deux sous-chaînes correspondant respectivement aux deux expressions. Les répétitions ont priorité sur les juxtapositions, qui à leur tour ont priorité sur les alternatives. Une sous-expression peut être entourée par des parenthèses pour modifier ces règles de priorité.

Les expressions de regroupement présenteraient peu d'intérêt sans opérateur de répétition. Pouvoir quantifier une sous-expression, permet de définir des contraintes d'apparition dans une expression.

— Exemple

Si on recherche l'expression régulière : /a+b*c ?d/ dans les textes suivants, on met en correspondance les éléments soulignés :

aaad
abbbbcd
bbbcd
abccd
aaabbbc

Si on recherche l'expression régulière : /a+b{2,}c ?/ dans les textes suivants, on met en correspondance les éléments soulignés :

aaaaabbbbbccccc
aaabbbccc
aaaaabbbbbbbbbbbbbccccc

Si on recherche l'expression régulière : / [:alpha :]{4} / (les mots de 4 lettres) dans le texte suivant, on trouve les éléments soulignés :

Il était un petit navire
Qui n'avait jamais navigué
Il partit pour un long voyage
Sur la mer Méditerrannée

https://www.ons.info-info.univ-paris5.fr/cbi/

1.3 La commande : grep

La commande `grep` est une commande qui affiche les lignes correspondant à un motif donné. La recherche peut être effectuée sur l'entrée standard, sur un fichier ou sur un ensemble de fichiers.

Le motif de recherche est défini par une expression régulière

Par défaut, la commande `grep` accepte les expressions régulières simples. Dans celles-ci, les méta-caractères ?, +, {, —, (, et) perdent leur signification spéciale, il faut utiliser à la place leurs versions protégées (avec l'anti-slash) {\} ?, {\}+, {\}{, {\}—, {\}(, et {\}). Pour pouvoir utiliser les expressions régulières étendues, il faut utiliser l'option -E

— Quelques options indispensables

Option	**Rôle**
-i	Ignorer les différences majuscules/minuscules aussi bien dans le *motif* que dans les fichiers
-l	Indiquer le nom des fichiers pour lesquels des résultats auraient été affichés
-n	Ajouter à chaque ligne de sortie un préfixe contenant son numéro dans le fichier
-r	Lire tous les fichiers à l'intérieur de chaque répertoire de manière récursive
-v	Inverse la sélection

Exemples

— Si on lance la commande `grep 'un'` sur un fichier contenant le texte suivant :

Il était un petit navire
Qui n'avait jamais navigué
Il partit pour un long voyage
Sur la mer Méditerrannée

— La sortie obtenue est :

Il était un petit navire
Il partit pour un long voyage

— Si on lance la commande `grep -v 'pour'` sur un fichier contenant le texte suivant :

Il était un petit navire
Qui n'avait jamais navigué
Il partit pour un long voyage

Sur la mer Méditerranée

— La sortie obtenue est :
Il était un petit navire
Qui n'avait jamais navigué
Sur la mer Méditerranée

> Avec la commande grep, on utilise l'option -e pour utiliser des expressions régulières et l'option -E pour les expressions régulières étendues

Les guillemets

Il est recommandé d'utiliser les simples guillemets car ils sont plus sûrs ; ils protègent l'expression régulière de toute intervention du shell. Il vaut mieux réserver l'utilisation des guillemets doubles au cas où il est nécessaire d'utiliser des variables d'environnement.

Différence

— La commande `grep "$HOME" fichier` cherche la chaîne de caractères correspondant au répertoire personnel de l'utilisateur (par exemple : /home/utilisateur) dans le *fichier*
— La commande `grep '$HOME' fichier` cherche la chaîne de caractères $HOME dans le *fichier*

1.4 Exercice 4.2

1. Comptez le nombre d'utilisateurs qui n'ont pas de shell permettant une connexion
2. A quelle ligne se trouve l'utilisateur "pulse" ?
3. Combien y a t-il de répertoires ou de fichiers commençant par un u à la racine ?
4. Combien y a t-il de répertoires ou de fichiers cachés dans votre répertoire personnel ?
5. Filtrez le fichier `/proc/cpuinfo` pour obtenir le nom du modèle du premier processeur
6. Filtrez le fichier `/proc/cpuinfo` pour obtenir la puissance de calcul du deuxième processeur (en bogomips)

1.5 Exercice 4.3

1. Téléchargez la version texte du tome I des Misérables (more files, puis 17489-8.txt) dans votre répertoire personnel en utilisant la commande suivante :
   wget http://www.gutenberg.org/files/17489/17489-8.txt
2. Combien y a t-il de mots dans ce livre ?
3. Combien y a t-il de lignes commençant par "Il" ?
4. Combien de fois le nom Magloire apparaît-il ?
5. Combien y a t-il de mots de 18 lettres ?
6. A quelles lignes se trouvent les mots de 16 lettres ?
7. Combien y a t-il de lignes comportant un point d'interrogation ?

1.6 La commande : sed

La commande `sed` est un des outils de manipulation de chaînes de caractères indispensables avec `grep`, `cut`, `sort`, etc. Elle permet de modifier/supprimer des occurrences dans une chaîne.

Pour utiliser `sed`, il faut lui fournir une chaîne de caractères à traiter. Elle peut provenir d'un fichier ou d'une variable.

En règle générale la syntaxe est de la forme :

https://www.ens-math-info.univ-paris5.fr/cbi/

's/occurrence_cherchée/occurrence_de_substitution/comportement'

Où l'occurrence cherchée est une expression régulière (`sed` ne reconnait pas les expressions régulières étendues)

Exemples Pour remplacer tous les espaces par le caractère souligné (_) :

```
echo "Le petit poisson vert" | sed -e "s/ /_/g"
Le_petit_poisson_vert
```

Le `g` permet d'invoquer un comportement global ; toutes les occurrences trouvées sont substituées.
Pour remplacer Windows par GNU/Linux ;-)

```
$ echo "J'aime Windows (windows)" | sed "s/[Ww]indows/GNU\/Linux/g"
J'aime GNU/Linux (GNU/Linux)
```

Pour encadrer le premier nombre de la ligne avec des étoiles :

```
$ echo "Il est 08:00" | sed -e "s/\([0-9][0-9]*\)/**\1**/"
Il est **08**:00
```

Autre possibilité :

```
echo "Il est 08:00" | sed -e "s/[0-9][0-9]*/**&**/"
Il est **08**:00
```

Dans l'occurrence de substitution, l'utilisation du métacaractère & (ou de la référence inverse {\}1), est une référence à la première concordance dans l'occurrence recherchée.

Il est également possible d'utiliser la commande sed pour passer les premières lignes d'un fichier. Par exemple, la commande suivante efface (d) les 5 premières lignes d'un fichier :

```
$ sed '1,5d' fichier
```

https://www.cas.math-info.u...paris5.fr/c...

La référence inverse

La référence inverse correspond à la sous-chaîne déjà mise en correspondance avec la n-ième sous-expression régulière entre parenthèses ; on la note {\}n , où n est un chiffre unique.

Par exemple, dans l'expression régulière : /(un)(deux)(trois)/ les références inverses sont {\}1 pour un, {\}2 pour deux et {\}3 pour trois.

La référence inverse ne concerne que la sous-expression correspondante, et ce même si la chaîne de caractères correspondante est présente plus loin dans le texte cible.

Exemples

— Si on recherche l'expression régulière : /(abc—def) {\}1/ dans les textes suivants, on met en correspondance les éléments soulignés :

abc abc ijk
abc def ijk
def abc ijk
def def ijk

elle est équivalente à l'expression : /(abc abc—def def)/

— On peut aussi s'en servir pour réaliser des commandes complexes telles que l'inversion de 2 colonnes en utilisant un `sed` :

Soit un fichier fruits contenant :

```
abricot 2
banane 3
poire 4
orange 5
```

p

En utilisant la commande `sed` (et en protégeant les parenthèses), on obtient :

```
$ cat fruits | sed -e 's/\([a-z]*\) \([0-9]*\)/\2\t\1/'
1    pomme
2    abricot
3    banane
4    poire
5    orange
```

La référence inverse {\}1 fait référence à la première expression parenthésée contenant [a-z]
La référence inverse {\}2 fait référence à la deuxième expression parenthésée contenant [0-9]
Le méta-caractère {\}t permet d'ajouter une tabulation

1.7 Exercice 4.4

Reprenez le tome 1 des Misérables de Victor Hugo

1. Supprimez les ponctuations
2. Séparez les ponctuations des mots (ajoutez un espace avant les points, les virgules)
3. Supprimez les doubles espaces littéraux (deux espaces qui se suivent)
4. Séparez les ponctuations des mots et supprimez les doubles espaces littéraux

https://www.ens.math-info.univ-paris5.fr/cbi/

2 Les scripts shell

Un *script shell* ou *script* est un fichier texte qui regroupe des commandes qui seront exécutées au lancement du script.

Premier script

Dans ce premier script, on va simuler la commande *id* qui retourne l'UID et le GID d'un utilisateur.
Exemple d'utilisation de la commande *id* :

```
$ id root
```

Donne le résultat suivant :

```
uid=0(root) gid=0(root) groupes=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
```

Pour produire, un résultat similaire (UID seulement), on pourrait écrire le script suivant :

```
#!/bin/bash
# id.sh : retourne l'id d'un utilisateur en fonction du login
# sept 2013 DJ
# Utilisation id.sh <nom_login>
# Recupere le contenu du fichier /etc/passwd, garde le login et l'uid
# Puis on cherche le nom login et on retourne l'uid
echo "L'uid de $1 est: "
cat /etc/passwd | cut -d":" -f1,3 | grep $1 | cut -d":" -f2
```

Lorsque vous écrivez un script (et quand vous programmez), vous devez toujours préciser dans l'entête :
- l'interpréteur de commande qui doit être utilisé ou la version du compilateur souhaitée
- le nom du script et sa fonction
- la date et l'auteur
- comment utiliser le script
- l'algorithme utilisé
- la liste des révisions

Idéalement, il faudrait également gérer les erreurs possibles.
Pour créer ce script :

```
nano id.sh
```

Ensuite, il faut le rendre exécutable

```
chmod u+x id.sh
```

On peut maintenant le lancer

```
./id.sh root
```

Vous devriez obtenir :

```
L'uid de root est:
0
```

https://www.ens.ma...info.univ-paris5.fr/cbi/

3 Compléments sur les processus

La commande `uptime` indique depuis quand le système fonctionne

```
$ uptime
15:12:36 up 227 days, 11:25,  1 user,  load average: 0.00, 0.00, 0.00
```

La commande `pstree` permet de déterminer l'arborescence des processus lancés sur un système.

```
$ pstree
init-+-acpid
     |-atd
     |-auditd-+-audispd-+-sedispatch
     |        |         `-{audispd}
     |        `-{auditd}
     |-avahi-daemon---avahi-daemon
     |-console-kit-dae---61*[{console-kit-dae}]
     |-crond
     |-cupsd
     |-4*[dbus-daemon]
     |-gam_server
     |-gpm
     |-hald---hald-runner-+-hald-addon-acpi
     |                    |-hald-addon-cpuf
     |                    |-hald-addon-inpu
     |                    `-2*[hald-addon-stor]
     |-irqbalance
```

```
      |-6*[mingetty]
      |-nscd---10*[{nscd}]
      |-ntpd
      |-rpc.idmapd
      |-rpc.statd
      |-rpcbind
      |-rsyslogd---2*[{rsyslogd}]
      |-sshd---sshd---sshd---bash---pstree
      |-udevd
      `-yum-updatesd
```

```
$ pstree `whoami`
sshd---bash---pstree
```

Tous les processus se voient allouer une durée d'exécution par le noyau. Si tous les processus avait la même durée allouée, les performances se dégraderaient à mesure que le nombre de processus augmenterait. Le noyau Linux utilise donc un jeu de règles heuristiques pour estimer la durée à allouer à un processus. Il essaie d'être équitable afin que deux utilisateurs requérant le même usage du processeur puisse obtenir la même priorité, et donc la même durée.

A un instant donné, la plupart des processus sont en attente d'une frappe de touche, d'un envoi de données via réseau ou via un périphérique, etc. Pendant leur attente, les processus ne nécessitent pas de ressource processeur.

En revanche, quand plusieurs processus travaillent normalement, le noyau doit décider s'il faut donner une priorité plus grande à tel ou tel autre processus. En effet, il arrive qu'au cours de son exécution un processus nécessite plus de ressources processeur qu'un autre. Pour gérer cela, les systèmes Unix sont dotés d'un mécanisme de priorité : `nice`

La commande `nice` permet d'exécuter un programme avec une priorité d'ordonnancement modifiée. Sur une machine multi-utilisateur, il est courtois (nice en anglais) de lancer les processus que l'on s'attend à voir travailler longtemps sur une machine avec une priorité moindre. Cela évite de bloquer les autres utilisateurs du système qui auraient des processus courts à exécuter Si aucun argument n'est fourni, `nice` affiche la priorité d'ordonnancement en cours, dont cette commande a hérité au lancement. Sinon, `nice` exécute la commande désirée en ajustant la priorité d'ordonnancement. Si aucun ajustement n'est précisé, la valeur de priorité de la commande est augmentée de 10. La priorité peut être ajustée avec nice dans l'intervalle -20 (le plus prioritaire) à 19 (le moins prioritaire), mais seul le super-utilisateur (root) peut indiquer un ajustement négatif.

```
$ nice -n 15 procesus
```

http://www.lsv.ens-cachan.fr/~goubault/

De la même façon, la commande `renice` permet de modifier la priorité d'ordonnancement d'un ou de plusieurs processus en cours d'exécution.

```
$ renice +5 -p pid
```

— Quelques commandes utiles

Commande	Rôle
tty	Afficher le nom du terminal associe à l'entrée standard
nohup	Exécuter un programme en le rendant insensible aux déconnexions
disown	Rendre un programme déjà en cours d'exécution insensible aux déconnexions
at	Lancer l'exécution d'un processus ultérieurement

3.1 Exercice 4.5 : les priorités

Le programme `yes` permet d'afficher y indéfiniment.

1. Ouvrez un terminal, et lancer `yes` pour le constater.

2. Stoppez le processus `yes`.
3. Relancez `yes` en tâche de fond en redirigeant la sortie du programme dans le vide.
4. Ouvrez un autre terminal, et un lancer un `top` pour observer l'activité du processus correspondant. Sur une machine multi-processeurs, faîtes <Shift>+1 pour afficher l'occupation de tous les processeurs.
5. Lancez deux autres `yes` en tâche de fond en redirigeant la sortie du programme dans le vide
6. Que constatez-vous ?
7. Diminuez au maximum la priorité de deux des processus correspondants.
8. Que constatez-vous ?
9. Stoppez tous les processus `yes`

3.2 Exercice 4.6 : sortie standard et terminaux

1. Ouvrez deux terminaux et disposez les côte à côte.
2. Déterminer le nom du fichier connecté sur l'entrée standard du premier terminal
3. Dans le second terminal, lancer la commande `yes` en arrière plan et en redirigeant la sortie vers le fichier connecté sur l'entrée standard du premier terminal
4. Que constatez- vous ?
5. Stoppez le processus `yes`.

4 Un peu plus loin en réseau

Vous pouvez obtenir le nom de la machine sur laquelle vous êtes connectés grâce à la commande `hostname`.

```
[utilisateur@serveur]$ hostname
serveur
```

Vous pouvez obtenir l'adresse ip de la machine sur laquelle vous êtes connectés grâce à l'option `-i` :

```
[utilisateur@serveur]$ hostname -i
127.0.0.1
```

Pour obtenir de plus amples informations sur les interfaces réseaux de la machine sur laquelle vous êtes connectés, vous pouvez utiliser la commande `ifconfig`, si vous êtes connectés en ethernet :

```
[utilisateur@serveur]$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:2A:05:B3:F9:11
          inet adr:127.0.0.1  Bcast:127.0.0.255  Masque:255.255.255.0
          adr inet6: fe77::12f:551d:fe43:8da1/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6390655 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2370461 errors:0 dropped:0 overruns:0 carrier:2
          collisions:0 lg file transmission:1000
          RX bytes:1001151350 (0.93 GiB)  TX bytes:471550316 (0.43 GiB)
```

Vous retrouverez ainsi son adresse au format IPv4 ainsi que celle au format IPv6, la quantité de données transmises par cette interface en émission et en réception.

Pour déterminer si une machine est connectée au réseau et obtenir des informations complémentaires vous pouvez utiliser la commande `ping` (valable seulement si les paquets réseaux envoyés ne sont pas bloqués par un pare-feu par exemple)

http://www.ens...th-info.univ-paris5.fr/cbi/

```
[utilisateur@serveur]$ ping -c 4  www.google.fr
PING www.l.google.com (66.249.92.104) 56(84) bytes of data.
64 bytes from par03s01-in-f104.1e100.net (66.249.92.104): icmp_seq=1 ttl=53 time=1.53 ms
64 bytes from par03s01-in-f104.1e100.net (66.249.92.104): icmp_seq=2 ttl=53 time=1.41 ms
64 bytes from par03s01-in-f104.1e100.net (66.249.92.104): icmp_seq=3 ttl=53 time=1.38 ms
64 bytes from par03s01-in-f104.1e100.net (66.249.92.104): icmp_seq=4 ttl=53 time=1.35 ms

--- www.l.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 1.356/1.421/1.532/0.080 ms
```

Ici, on a envoyé 4 requêtes vers un des serveurs de Google, on obtient son adresse IPv4, son nom qualifié et différents temps de latences. Ainsi, le temps moyen d'un aller/retour par le réseau est de 1.421 ms ; c'est une latence très faible (c'est presque comme si la machine était dans la pièce voisine ;-))

Selon un principe analogue, la commande `traceroute` permet de tracer la route des différentes machines qui se trouvent sur le réseau entre votre machine et la machine ciblée.

4.1 Exercice 4.7

1. Ouvrez deux terminaux ; dans le premier lancer un `top` et dans le second, récupérez le nom votre votre machine en utilisant la commande `hostname`.
2. Echangez le nom de votre machine avec un groupe voisin et connectez vous à sa machine grâce à la commande `ssh` en utilisant votre identifiant.
3. Lancez la commande `yes` dans votre terminal et observez le top sur l'écran du groupe voisin.
4. Que constatez-vous ?

https://www.ens.math-info.univ-paris5.fr/cbi/