Linux Utilisation

Module 06 – Spécificités du shell Bash

1

Objectifs

- Faire des recherches précises
- Découvrir les expressions régulières pour des recherches encore plus fines

Être efficace avec le bash et rechercher des fichiers

2

Effectuer des recherches

<commande 1> | <commande 2>

```
user30@deb:~$ ls | grep .txt
contacts.txt
edition1.txt
Fait.txt
listenombre.txt
liste.txt
Tel-2018.txt
Tel-2020.txt

user30@deb:~$ ls | grep .txt | wc -l
7
```

grep [-eilnv] "expr« <nomfichier1> [<nomfichier2>]

```
user30@deb:~$ grep -in "dupont" Edition
13:Dupont Jean 111111
14:Dupont Jean 111111
15:Dupont Patrice 222222
16:Dupont Jojo 999999
17:Dupont Pierre 222222
```


<command> | grep [-eilnv] "expr"

```
user30@deb:~$ ls | grep -v "[Tt]el"
-rw-r--r-- 1 user30 user30  113 juil. 21 11:04 liste.txt
-rw-r--r-- 1 user30 user30   29 juil. 21 09:36 newcontacts
drwxr-xr-x 3 user30 user30 4096 juil. 22 14:38 perso
-rw-r--r-- 1 user30 user30    0 juil. 22 17:27 toto
drwxr-xr-x 2 user30 user30 4096 août  10 10:55 video
```

Recherches avancées

find <chemin> [critères de recherches] ([action à effectuer sur le résultat)]

Recherche simple avec Find

```
user30@deb:~$ find . -name "[Tt]el*"
/home/user30/tel2020
/home/user30/Tel-2018.txt
/home/user30/perso/tel2018
/home/user30/perso/tel2020
/home/user30/perso/tel2019
/home/user30/Tel-2020.txt
```

Quelques critères de recherches parmi les plus utilisés :

- -name "ficnom" Recherche en fonction du nom. Peut être générique, doit être protégé par des doubles quotes
- -user nom Recherche des fichiers par nom de l'utilisateur propriétaire
- -group groupe Recherche des fichiers par nom du groupe propriétaire
- -type [fdlcbp] Recherche par type de fichier. Cette recherche utilise en argument les lettres f, d, l , c, b ou p représentant les divers types de fichiers Unix

```
user30@deb:~$ find . -type d -name "*test*"
./dirtest01
./perso/dirtest02
```

- -size [+-]nb Recherche en fonction de la taille (possibilité de préciser k, M ou G). Le caractère - ou + permet d'indiquer "plus grand que" ou "plus petit que".
- -mtime nb Recherche des fichiers sur date de dernière modification, nb correspond au nombre de jours (1 pour la veille, 2 jours avant ...)

```
user30@deb:~$ find dirtest01/ -size +500K
dirtest01/fic1M
dirtest01/fic10M
user30@deb:~$ find dirtest01/ -size -100K
dirtest01/fic50k
dirtest01/fichier01.txt
user30@deb:~$ find dirtest01/ -type f -mtime -1
dirtest01/ficnew
```

En plus des critères de recherches, find est capable d'appliquer diverses actions sur les résultats.

- -print — Affiche le nom complet des fichiers trouvés. Il s'agit de l'action par défaut
- -printf — Affiche le résultat avec les possibilités de formatage de la fonction printf du langage C (cf man find)

-exec command {} \; — Exécute séquentiellement la commande *command* sur le résultat représentée par {} . Doit se terminer par \;

```
user30@deb:~$ find . -type f -mtime -1 -exec ls -li {} \;
15684 -rw-r--r-- 1 user30 user30 0 août  10 17:05 ficnew
```

```
find . -type f -name "Edition" -exec cp -v {} /tmp/{}.txt \;
'./Edition' -> '/tmp/./Edition.txt'
```

-ok command {} \; — même fonctionnement que -exec mais avec demande de confirmation à chaque exécution.

Des recherches sous Linux

Introduction aux expressions régulières

- définition ([1]) : Chaîne de caractères ou motif, décrivant un ensemble d'autres chaînes selon une syntaxe précise.
- synonymes : Expression Rationnelle, expression normale
- Surnom : regex (de l'anglais Regular Expression)
- Utilisations : compilateurs, éditeurs de textes, IDE, commandes d'administration …

https://fr.wiktionary.org/wiki/expression_rationnelle

- Les expressions régulières sont utilisées par les outils d'édition et de recherche dans les fichiers tels que vi, grep, sed, awk, ed, ex, et autres.
- Une expression régulière est construite à l'aide de différents caractères dotés d'une signification particulière.
- Les caractères d'expression régulière auront cette syntaxe : **`regex`**.
- Les caractères normaux utiliseront celle-ci : `caractère`

- `*` répétition du caractère qui précède de 0 à n fois

  `a*` correspond à : `<rien>` ou `a` ou `aa` ou `aaa` ...
- `\+` répétition du caractère qui le précède de 1 à *n* fois.

  `a\+` correspond à `a` ou `aa` ou `aaa` ...
- `\?` correspond à 0 ou 1 fois le caractère précédent.

  `a\?` correspond à `<rien>` ou `a`

`car\{m,M\}` un caractère répété au minimum `m` fois et au maximum `M` fois

`ab\{1,3\}c` correspond à : `abc` ou `abbc` ou `abbbc`

`car\{n,\}` un caractère répété au minimum n fois

`es\{2,\}ai` correspond à : `essai`, mais aussi `esssai`, `essssai` ...

à

`car\{n\}` un caractère répété n fois exactement

`[A-Z]\{3\}` correspond à une chaîne de caractères composée de 3 majuscules

```
user30@deb:~$ ls | grep -E ^[eE]\{2,5\}
eedition
Eedition
eeeeexemple

user30@deb:~$ ls | grep -E ^[eE]\{2,3\}
eedition
Eedition
Eeeeexemple

user30@deb:~$ ls | grep -E ^[eE]\{2,3\}[^eE]
eedition
Eedition
```

- `.` Un caractère quelconque
  - `.` peut-être `a b c d ù 3` etc...
  - `.*` correspond à une séquence de caractères quelconques

`[...]` Un caractère parmi la liste entre crochets

- `[abc12]` correspond à : `a` ou `b` ou `c` ou `1` ou `2`

`[...-...]` Un caractère parmi l'intervalle défini entre crochets

- `[A-Z]` correspond à une lettre majuscule
- `[A-Za-z]` correspond à toutes les lettres minuscules et majuscules
- `[0-9]` correspond à un chiffre

`[^…]` Un caractère sauf ceux de la liste entre crochets

`[^ab1]` correspond à tout sauf `a` ou `b` ou `1`

`^motif` Le caractère `^` symbolise le début d'une ligne.

`^B` une ligne qui commence par B

`^[a\t]` une ligne qui commence par a ou une tabulation

`^[ \t]*` une ligne qui commence par un espace ou une tabulation, répétée de 0 à *n* fois

`motif$` Le caractère `$` symbolise la fin d'une ligne.

`fin^` une ligne qui se termine par le mot fin

`[^aeiouy]$` une ligne se termine par autre chose qu'une voyelle en minuscule

`^$` une ligne vide

`^.*$` une ligne quelconque

On peut aussi rechercher un ou plusieurs caractères selon qu'ils font partie d'une "classe". Il peut s'agir de caractères en majuscules ou en minuscules, de chiffres, de caractères alphanumériques (chiffres ou lettres) ...

- `[[:alpha:]]` Caractère alphabétique
- `[[:lower:]]` Lettre minuscule
- `[[:upper:]]` Lettre capitale
- `[[:alnum:]]` Caractère alphanumérique
- `[[:digit:]]` Chiffre décimal
- `[[:space:]]` Espace blanc
- `[[:blank:]]` Espace ou tabulation


```
user30@deb:~$ ls | grep ^[[:upper:]]
Edition
Eedition
Fait.txt
Tel-2018.txt
Tel-2020.txt
```

```
user30@deb:~$ ls | grep  [[:digit:]]$
contacts2020
dirtest01
tel2018
tel2020
```

Bash Métacaractères : classes de caractères

Ces classes peuvent aussi être utilisées directement dans le shell bash. En ce cas, la syntaxe varie légèrement et retrouve les métacaractères.

```
user30@deb:~$ ls [[:upper:]]*
Edition  Fait.txt  Tel-2018.txt  Tel-2020.txt
```

```
user30@deb:~$ ls -d *[[:digit:]]
contacts2020  dirtest01     tel2020     tel2018
```

Regex : caractères d'échappement

`\n`	Nouvelle ligne	`\s`	Séparateurs (espace et tabulation)
`\r`	Retour chariot	`\S`	Tous sauf des séparateurs
`\t`	Une tabulation	`\w`	Tous les caractères alphanumériques + l'underscore
`\d`	Des chiffres		
`\D`	Tous sauf des chiffres	`\W`	Tous les caractères sauf les alphanumériques + l'underscore

Regex : caractères d'échappement

`\`	L'antislash seul est le banaliseur des caractères spéciaux des expressions régulières
`\u`	Transforme le caractère suivant en majuscule
`\U`	Transforme tous les caractères suivants en majuscules (jusqu'à `\E`)
`\l`	Transforme le caractère suivant en minuscule
`\L`	Transforme tous les caractères suivants en minuscules (jusqu'à `\E`)
`\E`	Termine les séquences `\L` et `\U`
`|`	"ou" logique

Sous-expressions et références arrières

Sous-expression : partie d'une expression régulière encadrée par `\(` `\)`

Référence arrière : pour réutiliser des sous-expressions, numérotées de 1 à 9. Appelés également avec antislash.

La référence arrière reprend exactement le résultat trouvé dans sa sous-expression de référence.

`a\([bc]\)d\1` correspond à `abdb` ou `acdc`, mais pas `acdb`

Regex avec grep et find

TP

Recherches et regex

La commande Locate

Recherche sur noms de fichiers avec locate

- Commande non standard
- Utilisation d'une base de données référençant noms et emplacements des fichiers
- Besoin de mise à jour après installation ou création de nouveaux fichiers
- La commande updatedb (droits d'administration requis) met à jour la base

```
user30@deb:~$ touch explocate
user30@deb:~$ locate explocate

root@deb:~# updatedb

user30@deb:~$ locate explocate
/home/user30/explocate
```

- Vous savez rechercher des mots ou expressions dans les fichiers
- Vous pouvez manipuler les métacaractères du Bash ou les Expressions Régulières pour améliorer l'efficacité de vos recherches
- Vous savez automatiser recherche et action grâce à la commande find