# SR01 : A RETENIR POUR LE FINAL

## TD

1. Grep

- sert à rechercher des motifs spécifiques dans des fichiers ou des flux de données

2. Sed

- sert à éditer des textes : modifier, supprimer, insérer ou transformer

## GREP + REGEX

### options du GREP

```
grep -E // extended, pareil que egrep
grep -P // Perl regular expression --> compatible avec des expressions régulières
Perl
grep -G // basic, expression de base
grep -F // ce qui suit doit être interprété littéralement
grep -l // liste des fichiers avec match
grep -L // liste des fichiers sans match
grep -v // affichage des lignes ne contenant pas la chaîne précisée
grep -n // affiche les numéros de ligne
grep -c // compte le nombre de ligne
grep -o //  show only the part of a line matching PATTERN
grep -r // récursif : si rechercher dans répertoire
grep -i // majuscule = minuscule
grep -w // force pattern to match only whole words
grep -d // recurse, parcourt tous les fichiers des repertoires / sous-répertoires
grep -H // affiche le nom des fichiers avec la ligne ayant la correspondance
```

- On peut cumuler des options (par ex grep -vcE)
- on peut utiliser un pipe | par exemple sur un echo

```
[:punct:] // ponctuation
[:space:] // espaces
[:digit:] // chiffres
[:alpha:] // lettres
[:alnum:] // alphanumérique
```

- **Comment vérifier qu'une chaîne contient les deux**

```
echo "salut, " | grep [[:punct:]] | grep [[:space:]]
```

- **Comment vérifier qu'une chaîne contient un des deux**

```
echo "," | grep -E "[[:punct:]|[:space:]]"
// guillemets pour ne pas confondre le ou avec un pipe
// expression étendue donc -E
```

- **Afficher les téléphone qui commencent par 06 et qui ont 10 chiffres**

```
grep -E ^06[[:digit:]]{8}$ tel.txt
```

--> dès qu'on veut compter un ou des caractères on les mets entre crochets puis on met le nombre de fois attendu

- **Compter les numéros qui ne correspondent pas**

```
grep -vcE ^06[[:digit:]]{8}$ tel.txt
// -v = lignes qui ne correspondent pas
// -c = count
// -E = extended version
```

- **Un caractère ou un caractère**

```
"[a|b]"
[ab]
// si suite de caractère
(abc|def)
```

> Par exemple numéros qui commencent par 06 ou 07

```
grep -E "^0[6|7][[:digit:]]{8}$" tel.txt
// ou
grep -E ^0[67][[:digit:]]{8}$ tel.txt
```

- **On veut maintenant intégrer les signes de ponctuation**

```
// première approche
grep -E ^0[67][[:punct:]]?[[:digit:]]{2}\
[[:punct:]]?[[:digit:]]{2}\
[[:punct:]]?[[:digit:]]{2}\
[[:punct:]]?[[:digit:]]{2}$ tel.txt

// ? == {0,1}
```

```
// on peut simplifier

grep -E "^0[67]([[:punct:]]?[[:digit:]]{2}){4}$" tel.txt
// on répète quatre fois ce qui est entre parenthèse
// guillemets à cause des parenthèses
```

- **Trouver les URL de la forme lettre://lettre ou chiffre ou ponctuation**

```
grep -E ^[[:alpha:]]+://[[:alnum:]\.]+$ url.txt

// + == {1,} --> au moins une fois
// \. caractère spécial .
```

- **Trouver les URL de la même forme mais sans commencer ou finir par un point et sans avoir plusieurs points d'affilé**

```
grep -E "^[[:alpha:]]{4}://([[:alnum:]][\.]?[[:alnum:]]+)*$" url.txt

// * signifie 0 ou plusieurs occurrences
// premier alnum avec rien après signifie qu'il est obligatoire
```

- **Pareil mais avec numéro de port à la fin**

```
grep -E "^(ftp|http)://([[:alnum:]][\.]?[[:alnum:]]+)*(:[[:digit]]+)?$ url.txt
```

- **Compter le nombre de 'UNIX'**

```
grep -o 'UNIX' file.txt | wc -l
// word count -l = lines, -c = characters
```

REGEX

Voir fiche pour compléter

- \b : limite de mot (peut être au début et/ou à la fin)

# SED

- **Capturer le domaine d'une adresse mail**

```
echo <mail> | sed 's/.*@\(.*\)/domaine=\1/'
echo "mathilde.wallut@gmail.com" | sed 's/\(.*\)\.\(.*\)@\(etu\.*\)/prenom = \1
\nnom = \2 \ndomaine = \3/'
```

- s = substitute
- expression entre ()\ est celle capturée qui va être substituée
- \n = nième groupe capturé
- (expression à caturer)

## Administration système

```
whoami // info login courant
id Malena // info de l'user Malena
who // info users connectés sur le serveur
tty // console depuis laquelle on est connecté
```

- **Fichiers de configuration**

```
/etc/passwd --> infos utilisateur
/etc/group --> infos du groupe
/etc/shadow --> mdp hachés des utilisateurs
/etc/gshadow --> mdp hachés des groupes
```

- passwd et group sont lisibles par tout le monde et sont filtrés pour raison de sécurité
- shadow et gshadow ne sont lisibles que par root et parfois par les membres appartenant au groupe shadow
- **Liste des utilisateurs du système**

```
// on cherche la première colonne du passwd
// on utilise cut pour couper la colonne
// délimiteur : (-d:)
// première colonne : -f1
cat /etc/passwd | cut : -d: -f1
```

- **Liste des utilisateurs du groupe**

```
cat /etc/group | cut -d: -f1
```

- **Mode root (super utilisateur)**

```
sudo -i
```

- **Créer un dossier Sauvegarde et sauvegarder les fichiers de config des groupes et des utilisateurs en les copiant dedans**

```
mkdir Sauvegarde
cp /etc/passwd /etc/group /etc/shadow /etc/gshadow Sauvegarde
```

- **Créer des groupes et des utilisateurs**

```
groupadd -g 1025 LINUX
groupadd -g 1026 WINDOWS

// création des répertoires parents de connexion des users
mkdir /home/{linux,windows}

// user + UID + GID + répertoire + shell
useradd mathou --uid 2024 --home /home/linux --create-home --groups LINUX --gid
LINUX --shell /bin/bash
```

- -u == --uid
- --gid LINUX == --gid 1025
- on peut aussi faire un gpasswd
- **définir les mdp des utilisateurs**

```
passwd mathou
```

- **confifigurer un mot de passe**

```
chage mathou
```

- **utilisation de chmod**
- lit "à l'envers", si on met qu'un chiffre change les permissions des autres, deux chiffres change les permissions des autres et du groupe, trois chiffres permissions de tout le monde
- On ne peut pas exécuter un fichier pour lequel on a les permissions d'exécution mais pas de lecture
- **répertoire /tmp**
- accessible en écriture pour tous
- le sticky bit est activé pour que tout le monde ne puisse pas modifier des fichiers qu'ils n'ont pas créés
- t sur la permission des autres

- un utilisateur peut seulement supprimer les fichiers qu'il a créé

> drwxrwxrwt 19 root root 16K Dec 21 18:58 tmp

```
// soit un fichier avec bit SUID activé
chmod u-x file.sh // on retire l'exécution à l'utilisateur

--> le s devient S car plus de permission d'exécution mais bit SUID activé
(fonctionne pareil pour le SGID et le sticky bit)
```

- **Créer un répertoire dont les fichiers sont détenus par le groupe users et ne peuvent être supprimés que par l'utilisateur qui les a créés**

```
mkdir Box
chown :users Box/
// le groupe propriétaire est désormais users l'utilisateur est inchangé

// w+x pour le groupe
// setgid permet aux fichiers de Box d'hériter du groupe du répertoire
chmod g+wxs Bow/

// sticky bit pour que seul le proprio d'un fichier puisse le supp
chmod o+t Box/
```

```
Useradd -m Dave
```

- Crée le user Dave avec son répertoire personnel (par défaut /home/dave) et les fichiers et répertoires du répertoire squelette sont copiés dedans. Un nouveau groupe est créé avec le même nom que le compte utilisateur
- tant qu'on n'a pas défini le mdp, le compte utilisateur est vérouillé (! dans cat /etc/shadow | grep dave)

```
cat /etc/passwd | grep dave
-> dave:x:1015:1019::/home/dave:/bin/sh
cat /etc/group | grep 1019
-> dave:x:1019:
// UID = 1015, GID=1019, pas de membre dans le groupe dave car le quatrième champ
de /etc/group est vide
```

- **Utilisation de plusieurs options à la fois**

```
Useradd -u 1035 -g sys_amdin -G web-admin,db_admin carol
```

- **Supprimer des comptes et des groupes**

```
userdel $user
groupdel $group
```

- **Modification du mdp**

```
ls -l /usr/bin/passwd
-> -rwsr-xr-x 1 root root 42096 mag 17  2015 /usr/bin/passwd
// La commande passwd a le bit SUID activé (le quatrième caractère de cette
ligne), ce qui signifie que la commande est exécutée avec les privilèges du
propriétaire du fichier (donc root). C’est ainsi que les utilisateurs ordinaires
peuvent modifier leur mot de passe.
```

# MAKEFILE

Soit la structure d'un programme suivante :

```
mystr.c
mystr.h

mystrchrn.c
mystrchrn.h

mystrinv.c
mystrinv.h

mystrncat.c
mystrncat.h

mystrcpy.c
mystrcpy.h

mystrupdown.c
mystrupdown.h

teststring.c --> main

dans les .h :

#ifndef fichier
#define fichier
#include <stdio.h>
#include <string.h>
// struct ou fonctions
#endif //fichier
```

1er jet du makefile :

```
teststring : teststring.o mystr.o mystrchrn.o mystrinv.o mystrncat.o \
        mystrcpy.o mystrupdown.o
        gcc -o tesstring teststring.o mystr.o mystrchrn.o mystrinv.o \
                mystrncat.o mystrcpy.o mystrupdown.o

teststring.o : teststring.c mystr.h mystrchrn.h mystrinv.h mystrncat.h \
        mystrcpy.h mystrupdown.h
        gcc -c teststring.c

mystr.o : mystr.c mystr.h
        gcc -c mystr.c

mystrchrn.o : mystrchrn.c mystrchrn.h
        gcc -c mystrchrn.c

mystrinv.o : mystrinv.c mystrinv.h
        gcc -c mystrinv.c

mystrncat.o : mystrncat.c mystrncat.h
        gcc -c mystrncat.c

mystrcpy.o : mystrcpy.c mystrcpy.h
        gcc -c mystrcpy.c

mystrupdown.o : mystrupdown.c mystrupdown.h
        gcc -c mystrupdown.c

clean :
        rm teststring teststring.o mystr.o mystrchrn.o mystrinv.o \
                mystrncat.o mystrcpy.o mystrupdown.o
```

première amélioration :

```
OBJS = teststring.o mystr.o mystrchrn.o mystrinv.o mystrncat.o \
        mystrcpy.o mystrupdown.o

teststring : $(OBJS)
        gcc -o testring $(OBJS)

teststring.o : teststring.c mystr.h mystrchrn.h mystrinv.h mystrncat.h \
        mystrcpy.h mystrupdown.h
        gcc -c teststring.c

mystr.o : mystr.c mystr.h
        gcc -c mystr.c

mystrchrn.o : mystrchrn.c mystrchrn.h
        gcc -c mystrchrn.c
```

```
mystrinv.o : mystrinv.c mystrinv.h
        gcc -c mystrinv.c

mystrncat.o : mystrncat.c mystrncat.h
        gcc -c mystrncat.c

mystrcpy.o : mystrcpy.c mystrcpy.h
        gcc -c mystrcpy.c

mystrupdown.o : mystrupdown.c mystrupdown.h
        gcc -c mystrupdown.c

clean :
        rm teststring $(OBJS)
```

2eme amélioration :

```
CC:=gcc
OBJS = teststring.o mystr.o mystrchrn.o mystrinv.o mystrncat.o \
        mystrcpy.o mystrupdown.o
TARGET = teststring
HEADERS = mystrinv.h, mystrncpy.h, mystrchrn.h, mystrncat.h, mystrupdown.h
```

## Questions de cours

- **différence entre un appel système et une fonction normale**
  --> Une fonction normale est exécutée en mode utilisateur tandis qu'un appel système est exécuté en mode système donc avec plus de privilèges quant à l'accès aux ressources

- **différence entre Linux et GNU/Linux**
  --> Linux fait référence au noyau (kernel) du système d'exploitation, il gère entre autres la mémoire, les processus de bas niveau ou autres. GNU/Linux fait référence à l'ensemble du système d'exploitation, incluant le noyau Linux et les outils GNU.

- **différence entre appel système exec() et system()**
  --> exec() remplace le processus courant, les instructions situées après exec() dans le programme ne s'exécuteront donc pas. L'appel system() ne fait pas de recouvrement mais crée un processus fils qui se charge d'exécuter la commande passée en paramètre.

- **comment exécuter un process dont la taille est supérieure à la RAM**
  --> On peut utiliser une mémoire virtuelle dans laquelle on stocke à un instant donné les données pas nécessaires à l'exécution du process. La RAM ne contient alors que les données absolument utiles pour l'exécution.