# Cours 2
# Principe et fonctionnement d'un système informatique

Nicolas Carrara

Parcours MASE/SCC/SISL licence MIASHS, S3, Lille 3.

Année 2018-2019

# Les flux de redirections | la sortie standard

A la suite d'une commande, au lieu d'afficher la sortie standard dans la console, on peut

- Dump les résultats dans un nouveau fichier : >
- ou à la fin d'un fichier : >>.

# Les flux de redirections | la sortie d'erreur

On peut aussi faire la distinction entre la sortie standard et la sortie d'erreurs :

- `2>` (ou `2>>`) pour les erreurs.
- `2>&1` pour rediriger les erreurs de la même façon que la sortie standard (fusion).

# Les flux de redirection | exemples

- `ls -l > resultat_ls.txt`
- `cat fileunknown.csv > std.txt 2> err.log`
- `cat fileunknown.csv >> std.txt 2>&1`

# Les flux de redirections : les entrées

On peut spécifier l'entrée d'une commande avec un fichier ou depuis le clavier

- < : depuis un fichier
- << : depuis le clavier.

Exemples :

- `cat < a_afficher.txt`
- `sort -n <<`. Trier des nombres, taper FIN pour terminer la saisie.

# Les flux de redirections : chaîner les commandes

On peut aussi rediriger vers une autre commande avec la barre : |. Par exemple :

- `ls -l | grep '-rw-rw-r--'`
- `cat fichier.txt | grep 'salut'`

# Regex

Une expression régulière (regex, Stephen Cole Kleene 1956) est :

- une chaîne de caractère.
- Elle décrit un ensemble de chaîne de caractères possible.
- Les wildcards (*,[] et ?) sont une forme très simplifiée des regex.

# Regex | Exemples

- Le pattern ex-(a ?e|æ|é)quo : ex-équo, ex-equo, ex-aequo et ex-æquo
- Le pattern 6,66*$ : 6,6, 6,666 , 6,6666 , ...

# Regex | Les quantificateurs

- ? : zéro ou une occurence de l'expression précédente
  - Chiens ? : Chien, Chiens
  - 11 ?0 ? : 110, 11 ,10
- * : zéro ou plusieurs occurence de l'expression précédente
  - gooo*gle : google, gooogle, goooogle ...
- + : une ou plusieurs occurences de l'expression précédente
  - goo+gle : google, gooogle, goooogle ...
- . : un seul caractère
  - ...DarkVador... : XxXDarkVadorXxX, 666DarkVadorXxx, MdrDarkVador_59

# Regex | Autres opérateurs

- () : groupement de caractères
  - Etudiant(es) ? : Etudiant, Etudiantes
- | : le choix entre deux expressions
  - a|b : a,b
  - Etudiant(es|s) : Etudiants, Etudiantes
- [] : liste de caractères
  - [0123456789] : 0,1,2 ...
  - [^*aeiouy*] : 0,b,z, ! ...
- $\{n\}$ : précise le nombre d'occurences de l'expression
  - $bo\{2\}h!$ : booh !
  - $bo\{2,3\}h!$ : booh !, boooh !
  - $bo\{3,\}h!$ : boooh !, booooh !, . . . , booo $\infty$ ooooh !

# Regex | caractères spéciaux

- ^ : début de ligne
- $ : fin de ligne.
- \ : métacaractère
  - \$ : le caractère dollard.
  - \) : la parenthèse fermante.
- On peut aussi utiliser [] pour échapper les métacaractères (sauf^) :
  - [.?*] : . ? ou *
  - [\^*abc*] :^,a,b ou c.

# Regex | classe de caractère

Elles dépendent du moteur de Regex. Exemples :

- Caractère alphanumérique en POSIX : `[:alnum:]`,
- en java $\backslash p\{Alnum\}$
- et en ASCII : A-Za-z0-9

Utilisation :
Voir la page wikipédia pour plus de détails sur les classes.

# Regex | le standard POSIX

POSIX est une famille de normes techniques. Deux types de normes pour les regex

- BRE (« Basic Regular Expression ») : $\{\}$,(), ?,+ ne sont pas des métacaractères. Utiliser $\backslash$ pour les rendre méta.
- ERE (« Extented Regular Expression ») : tout ce dont on a parlé précédement.

# Regex | Les utilitaires

Différentes commandes linux utilisent des regex :

- grep : historiquement :g/re/p sur ed (éditeur de texte), global regexp print.
- sed : stream editor, édite un flux séquentiel de données textuelles (ligne par ligne).
- rename : renommer un fichier.

Par defaut, grep et sed sont en BRE. Modifiable vers ERE avec une option : -E pour grep et -r pour sed. Rename est en ERE.

Python, Java, vim ... possèdent des moteurs de Regex.

# grep

Exemples d'execution de grep

- cat fichier.txt | grep 'bonjour'
- cat fichier.txt | grep '\(bonjour\)|\(bonsoir\)'
- cat fichier.txt | grep -E '(bonjour)|(bonsoir)'

# sed

Les commandes :

- s : substitute, remplace un pattern par un autre.
- g : global, applique la modification sur toutes les occurences du pattern.

Exemples d'execution de sed

- sed 's/day/night/g' old >new
- sed 's/[0123456789]/ceci_est_un_chiffre/' old >new
- sed 's/\(oe\)/œ/' <test.txt
- sed -r 's/(oe)/œ/' <test.txt

# rename

Les options :

- -n : n'applique par le changement, l'affiche seulement.
- -f : force la réecriture des fichiers
- -v ; verbose

Exemples d'execution de rename :

- `rename -n 's/.htm\$/.html\$/' *`
- `rename -nvf 's/id\_([0-9]{2})/id=\$1/' *.csv`

# Exercice (difficulté facile)

Trouver les expressions régulières pour les ensembles suivant :

- Les voyelles.
- Les chiffres et les nombres entiers (positive ou negatifs).

Quelles commandes pour :

- Afficher les lignes qui contiennent le mot "SHS" de tous les fichiers du repertoire courant.
- Changer toutes les virgules du fichier test.csv en espace.
- Faire une tabulation (\t) après chaque instruction "for" du fichier debug.py.

# Exercice (difficulté moyenne)

Trouver les expressions régulières pour les ensembles suivant :

- Les phrases qui commencent par "Je".
- Les nombres pairs.
- Les nombres binaires.
- Les prix strictement inférieurs à 1000euro, en euro ou en dollard.

Trouver la commande pour changer le nom des fichiers du format a_en_fonction_de_o.png vers abscisse="a"_ordonnee="o".png. a et o sont des variables. Par exemple degrés_en_fonction_de_temps.png.

# Exercice (difficulté haute)

Trouver les expressions régulières pour les ensembles suivant :

- Les citations, avec des simples quotes ou double quotes.
- Les phrases terminant par un mot en majuscule.
- Les mots de passe : 8 caractères minimum, au moins une lettre, un chiffre et un caractère spécial.