UNIX - Parties 3 & 4

Méta-caractères
&
Expressions régulières

Architecture – Systèmes – Réseaux
Introduction aux systèmes informatiques - UE11, M1101-1

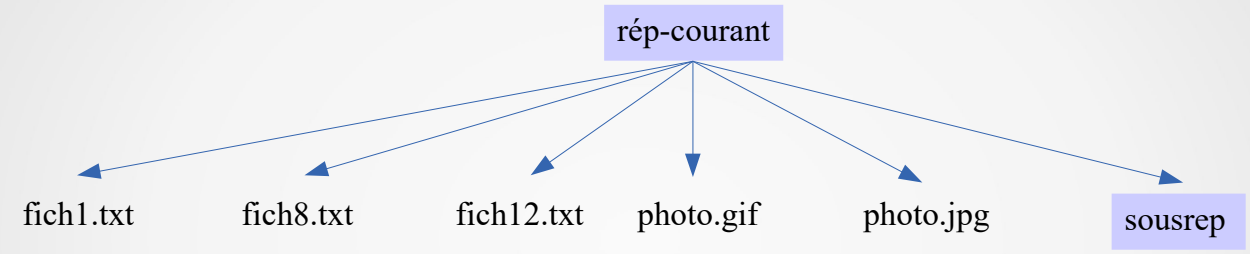
Objectifs

- Apprendre à utiliser LINUX
- Se simplifier la vie : comprendre la notion de méta-caractère et d'expression régulière
- Eléments de culture générale

Plan

- Méta-caractères (ou caractères spéciaux)
- Application : génération de noms de fichiers
- Expressions régulières
- Mise en pratique
  - Les filtres : la commande grep
  - Expressions régulières de base et étendues

Méta-caractères (caractères spéciaux), pourquoi ?



fichiers source

```
portejoi@k-prof-052% cp fich1.txt fich8.txt fich12.txt photo.gif
photo.jpg sousrep
```

répertoire destination

Long à écrire et pas très puissant ! ☹

Exemple : le méta-caractère *

* = n'importe quelle suite de caractères (0 à n)

```
portejoi@@k-prof-052% cp fich1.txt fich8.txt fich12.txt photo.gif
photo.jpg sousrep
```

tous les fichiers commençant par *photo.*

```
portejoi@@k-prof-052% cp *.txt photo.* sousrep
```

tous les fichiers finissant par *.txt*

Remplacement par le shell avant interprétation :

*.txt → fich1.txt fich10.txt fich20.txt
photo.* → photo.gif photo.jpg

Exemples d'utilisation du méta-caractère *

fich1.txt
fich10.txt
fich20.txt
photo.gif
photo.jpg
sousrep

*	fich1.txt fich10.txt fich20.txt photo.gif photo.jpg sousrep	* = *tout !*
f*0.txt	fich10.txt fich20.txt	
o	photo.gif photo.jpg sousrep	
p	photo.gif photo.jpg sousrep	* = *rien !*

Plus de méta-caractères (associés à des noms de fichiers)

fich1.txt
fich10.txt
fich22.txt
photo.gif
photo.jpg
sousrep

***** : toutes les chaînes de caractères, y compris la chaîne vide

Exemples vus précédemment

? : un caractère quelconque (au moins un et un seul)

fich?0.txt → fich10.txt
fich??.txt → fich10.txt fich22.txt

Plus de méta-caractères (associés à des noms de fichiers)

fich1.txt
fich10.txt
fich20.txt
photo.gif
photo.jpg
sousrep

[…] : un caractère quelconque appartenant à la liste

deux caractères séparés par un tiret (-) définissent une liste de caractères rangés par ordre alphabétique, premier et dernier inclus

[afsz]ou[afsz]rep → sousrep
phot[g-z].gif → photo.gif

[^…] : ^une liste de caractère à exclure

fich[^2468]0.txt → fich10.txt
fich[^2-8].txt → fich1.txt

Plus de méta-caractères (associés à des noms de fichiers)

{...,...} : une liste de chaînes de caractères

fich1.txt
fich10.txt
fich22.txt
photo.gif
photo.jpg
sousrep

fi{ch1,ch2}0.txt → fich10.txt
fi{ch1,ch2,ch22}.txt → fich1.txt fich22.txt

Combinaisons

fi{ch1,ch2}*.txt → fich1.txt fich10.txt fich22.txt
fich[0-5]?.txt → fich10.txt fich22.txt

Neutralisation d'un méta-caractère : \

Empêcher l'interprétation d'un caractère(1) : \ (antislash)

(1) : il existe d'autres caractères spéciaux pour l'interprétation (<, >, etc...)

fich1.txt
fich1*.txt
fich?2.txt
fich\??.txt

f*1

fich1*.txt → fich1*.txt
fich\??.txt → fich?2.txt

touch f*1 crée un fichier nommé f*1

Neutralisation d'une chaîne de caractères

L'encadrer avec l'apostrophe (') ou l'apostrophe double (")

fich1.txt
fich1*.txt
fich?2.txt
fich\??.txt

f*1

fich'\ ??'.txt → fich\??.txt
'fich\ ??'.txt → fich\??.txt

fich"\??".txt → fich\??.txt
"fich\??".txt → fich\??.txt

Voir expressions régulières

Les expressions régulières (er)

Mécanisme très proche du mécanisme de génération de noms de fichiers, utilisé pour la description générique de chaînes de caractères

Utilisées par :

- un grand nombre d'utilitaires (grep, sed, awk pour UNIX)
- des langages (perl...)
- des éditeurs de texte (vi, emacs...)

**Caractères spéciaux employés dans les expressions régulières : | . * + ? ^ $ ( ) [ ] { } **

Caractères spéciaux pour les er

char	Le seul caractère *char*	
sp	Le seul caractère spécial *sp*	
.	Un seul caractère quelconque (attention : ? pour fichiers)	
[gkl]	Un des caractères placés entre []	(ie g, k ou l)
[^gkl]	Un caractère autre que g, k ou l	(^ = négation)
[a-z]	Tous les caractères de a à z	(ordre alphabétique)
[^a-z]	Tous les caractères sauf ceux compris entre a et z	

Les expressions régulières atomiques (era)
Exemples :

a	§ le caractère a
[i-n]	§ un seul caractère parmi i, j, k, l, m, n
.	§ un seul caractère quelconque
\.	§ le caractère .
[^0-9]	§ un seul caractère autre qu'un chiffre

Les expressions régulières - Construction

expression régulière atomique = brique élémentaire de l'er

expression régulière = combinaison d'era

3 opérations élémentaires :

- Concaténation (quoi ?)
- Quantification (combien ?)
- Ancrage (où ?)

ER - La concaténation (quoi ?)

Une *er* est souvent obtenue par concaténation (juxtaposition) d'*era*

Par exemple, abc signifie *a* suivi de *b* suivi de *c*

Remarque : pas d'opérateur spécifique

ER - La concaténation (quoi ?) – Exemples

abc	§ la chaîne abc
[Oo]ui	§ la chaîne Oui ou la chaîne oui
[A-Z][0-9]..	§ une majuscule suivie d'un chiffre § suivi de deux caractères quelconques ==> *A8b5* *Z444* mais pas *h7fu*
[a-z][0-9].\.	§ une minuscule suivie d'un chiffre § suivi d'un caractère quelconque suivi § d'un point ==> *a8b.* *h6..* mais pas *Z67f*

ER – La quantification (combien?)

era, une expression régulière atomique

era*	Tout mot de 0 à n caractères vérifiant *era*
era+	Tout mot de 1 à n caractères vérifiant *era*
era?	Tout mot de 0 à 1 caractère vérifiant *era*
era{n}	Tout mot de n caractères vérifiant *era*
era{n_1,n_2}	Tout mot de n_1 à n_2 caractères vérifiant *era*
era{n_1, }	Tout mot d'au moins n_1 caractères vérifiant *era*
era{ ,n_2}	Tout mot de 0 à n_2 caractères vérifiant *era*

Remarque : Notation {n,m} générale mais lourde

==> simplification :
- * pour {0, }
- + pour {1, }
- ? pour {0,1}

ER - La quantification (combien ?) – Exemples

a* § a répété 0 à n fois
==> *rien* ou *a* ou *aa* ou *aaa*...

a.* § a suivi de n'importe quelle chaîne (même vide)
==> *a* *ou* *ab* ou *abc* ou *abbbd*...

a+ § a répété 1 à n fois
==> *a* ou *aa* ou *aaa*...

a? § a répété 0 ou 1 fois
==> *rien* ou *a*

ER - La quantification (combien ?) – Exemples, suite

a{2} § a répété 2 fois
==> *aa*

[a-b]{2} § a ou b répété 2 fois
==> *aa* ou *ab* ou *bb* ou *ba*

ER – L'ancrage (où ?)

Deux caractères spéciaux :

- ^ désigne le début de la ligne s'il est placé au début de l'er
- $ désigne la fin de la ligne s'il est placé à la fin de l'er

ER – L'ancrage (où ?) – Exemples

^linux	==>	*linux* en début de ligne
linux$	==>	*linux* en fin de ligne
^linux$	==>	une ligne ne contenant que *linux*
^$	==>	une ligne vide

ER – Combinaison d'er – Exemples

Deux opérateurs permettent de combiner les er :

- | désigne l'alternative
- ( ) désigne le groupage

Architecture – Systèmes – Réseaux
Introduction aux systèmes informatiques - UE11, M1101-1

ER – Combinaison d'er – Exemples

linux|unix ==> le mot *linux* ou le mot *unix*

^linux|^unix ==> *linux* ou *unix* mais en début de ligne

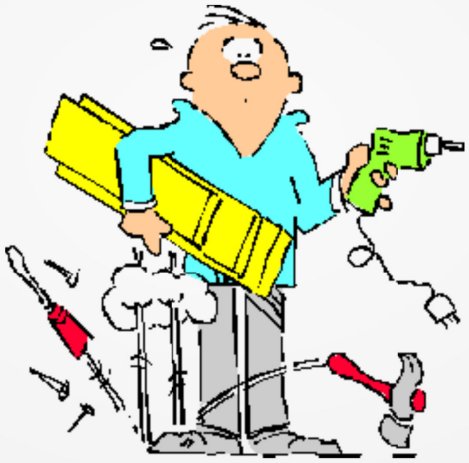
[A-Z]{8}|[0-9]{4} ==> 8 majuscules ou 4 chiffres

tsoin{2} ==> tsoinn

(tsoin){2} ==> tsointsoin

(bla|tsoin){2} ==> blatsoin ou tsoinbla ou blabla ou tsointsoin

Dans la pratique...



La commande grep (filtre) – Exemple 1

Principe : rechercher des informations dans un fichier *ligne par ligne*

⚠ Commande très utile ! ⚠

bonjour
mon
ami
de
toujours

fichier.txt

```
portejoi@@k-prof-052% grep jour fichier.txt
bonjour
toujours
portejoi@@k-prof-052%
```

grep sans argument fichier : entrée au clavier terminée par <ctrl c>

La commande grep – Plus d'exemples

grep read prog.f : toutes les lignes du fichier prog.f contenant la chaîne *read*

grep '*' prog.f : toutes les lignes du fichier prog.f contenant un caractère *

grep '[0-9]' * : toutes les lignes contenant un caractère numérique dans tous les fichiers du répertoire courant

grep -l 'call sub' * : tous les fichiers du répertoire courant contenant la chaîne *call sub*, et affichage de leur nom (*-l*)

Remarque : le caractère quote (') sert à encadrer l'er ou une chaîne

La commande grep – Plus d'options

-n : fait précéder chaque ligne affichée par son numéro dans le fichier

-v : toutes les lignes du fichier sauf celles contenant l'expression

-l : que les noms de fichiers dont au moins une ligne correspond à la recherche

-i : pas de distinction entre majuscules et minuscules

-c : affiche seulement le nombre de lignes contenant l'expression

NB : expressions régulières de base et er étendues

Vues précédemment : er étendues

Dans les er de base, + *? { } ( )* | n'ont pas de signification spéciale
(par défaut sous Linux : er de base)

==> les précéder de \ pour leur interprétation

Voir TD-TP

Principaux filtres UNIX (par ordre croissant de possibilités)

grep : recherche des chaînes de caractères conformes à un motif donné par une er de base et les affiche.

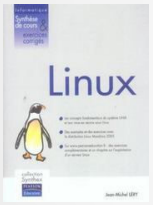
egrep : recherche des chaînes de caractères conformes à un motif donné par une er étendue et les affiche (grep -e).

sed : Petit langage de programmation permettant l'écriture de scripts. Cherche des chaînes de caractères conformes à un motif donné par une er et les modifie.

awk : langage de programmation spécialisé permettant des traitements complexes par des programmes très courts.
Permet de rechercher, modifier, formater, compter, afficher, traduire, etc...

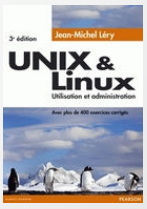
Bibliographie

Armspach Jean-Paul, Colin Pierre, Ostré-Waerzeggers Frédérique. 1999. LINUX - Initiation et utilisation. 2ème édition. Edition DUNOD.



Léry Jean-Michel. 2006. Linux. 3e édition 2011. Collection Synthex. Edition PEARSON.

Léry Jean-Michel. 2011. UNIX & Linux - Utilisation et administration. Avec plus de 400 exercices corrigés. 3e édition. Edition PEARSON.



Bosc Marcel. http://www-info.iutv.univ-paris13.fr/~bosc