Expressions Régulières

---

1. Introduction aux Expressions Régulières

Définition

- Les **expressions régulières (ER)** sont des descriptions symboliques utilisées pour définir des motifs de caractères.
- Utilisées dans les outils Linux (`grep`, `sed`, `awk`, etc.) et des langages comme Perl, elles servent pour rechercher ou manipuler des chaînes de caractères.

Différences avec la génération de noms

- Les ER sont **plus puissantes** que le mécanisme de génération de noms de fichiers, qui est limité aux caractères génériques (`*`, `?`, `[]`).

Utilisations courantes

- Rechercher des motifs spécifiques dans des fichiers.
- Valider des formats (emails, numéros de téléphone, etc.).
- Remplacer du texte dans des fichiers.

---

2. Conventions des ER

Deux types principaux

1. **Expressions Régulières de Base (BRE)** : Syntaxe limitée, utilisée dans des outils comme `grep` standard.
2. **Expressions Régulières Étendues (ERE)** : Syntaxe avancée, requiert souvent l'option `-E` pour `grep`.

Caractères spéciaux

- Ces caractères doivent être échappés (`\`) pour être utilisés littéralement :
- `| . * + ? ^ $ ( ) [ ] { } \`

---

3. Expressions Régulières Atomiques (ERA)

Définition

- Les ERA sont les **éléments de base** pour construire des ER. Elles définissent des **ensembles de caractères**.

Principaux motifs d'ERA

Motif	Signification	Exemple
`ch`	Correspond exactement au caractère `ch`	`a` correspond à "a".
`.`	Correspond à **tout caractère**	`a.b` correspond à "acb".
`[a-z]`	Tout caractère entre `a` et `z`	`[a-c]` correspond à "a", "b".
`[^a-z]`	Tout sauf les caractères entre `a` et `z`	`[^0-9]` exclut les chiffres.
`[a-zA-Z0-9]`	Alphanumérique (lettres et chiffres)	`[a-zA-Z]` pour lettres.
`\sp`	Correspond au caractère spécial `sp`	`\.` correspond au caractère `.`.

---

4. Construction des ER

Concaténation

- **Définition :** La juxtaposition de motifs sans opérateur.
- **Exemples :**
  - `[A-Z][0-9]` : Une majuscule suivie d'un chiffre.
  - `abc` : Les caractères "abc" dans cet ordre.

Quantification

- **Définition :** Permet de définir combien de fois un motif doit apparaître.

Quantifieur	Signification	Exemple
`*`	0 ou plusieurs répétitions	`A*` correspond à "", "A", "AAA".
`+`	1 ou plusieurs répétitions	`A+` correspond à "A", "AAA".
`?`	0 ou 1 répétition	`A?` correspond à "", "A".
`{n}`	Exactement `n` répétitions	`A{3}` correspond à "AAA".
`{n,m}`	Entre `n` et `m` répétitions	`A{1,3}` correspond à "A", "AA".

Ancrage

- **Définition :** Fixe l'emplacement d'un motif dans une ligne.
- **Symboles :**
  - `^` : Début de ligne.
  - `$` : Fin de ligne.
- **Exemples :**
  - `^Linux` : Les lignes commençant par "Linux".
  - `Linux$` : Les lignes terminant par "Linux".
  - `^$` : Les lignes vides.

---

5. Combinaisons des ER

L’alternative (`|`)

- **Définition :** Correspond à "Ceci OU Cela".
- **Exemples :**
  - `Linux|Unix` : Le mot "Linux" ou "Unix".
  - `^[A-Z]|[0-9]{3}$` : Commence par une majuscule OU finit par trois chiffres.

Groupage (`()`)

- **Définition :** Permet de regrouper des motifs et d’appliquer des quantifieurs sur tout le groupe.
- **Exemples :**
  - `(abc){2}` : "abcabc".
  - `(Linux|Unix)+` : Une ou plusieurs occurrences de "Linux" ou "Unix".

---

6. Classes et caractères spéciaux

Caractères spéciaux échappés

Caractère	Signification
`\n`	Nouvelle ligne
`\t`	Tabulation
`\103`	Code ASCII octal 103

Classes prédéfinies

Classe	Signification	Exemple
`[[:alpha:]]`	Lettres (majuscules et minuscules)	`[[:alpha:]]` : "a", "B".
`[[:digit:]]`	Chiffres	`[[:digit:]]` : "1", "9".
`[[:lower:]]`	Lettres minuscules	`[[:lower:]]` : "a", "z".
`[[:upper:]]`	Lettres majuscules	`[[:upper:]]` : "A", "Z".
`[[:space:]]`	Espaces	`[[:space:]]` : " ", `\t`.

---

7. Exemple détaillé : Recherche Prénom-Nom

Objectif

- Trouver des prénoms suivis de noms dans un texte.

Décomposition

1. **Prénom :** `[[:upper:]][[:lower:]]+`
    - Une majuscule suivie de lettres minuscules.
2. **Nom :** `[[:upper:]]{2,}`
    - Deux lettres majuscules ou plus.
3. **Séparation :** `(^|[[:space:]])` et `([[:space:][:punct:]]|$)`
    - Début/fin de ligne ou espace/ponctuation.

Expression complète

```
grep -E '(^|[[:space:]])[[:upper:]][[:lower:]]+
[[:upper:]]{2,}([[:space:][:punct:]]|$)' fichier.txt
```

Exemples pratiques

Texte :

```
Rayan MIRI, Hicham AMMARI, Meryem CHAKIR.
```

Résultat :

- Correspond à : `Rayan MIRI`, `Hicham AMMARI`, `Meryem CHAKIR`.

---

8. Commandes Pratiques

`grep`

- **Description :** Recherche de motifs dans un fichier.
- **Syntaxe :**
- `grep 'motif' fichier`
- **Exemples :**
    - `grep '^Linux' fichier.txt` : Lignes commençant par "Linux".
    - `grep '[0-9]' fichier.txt` : Lignes contenant des chiffres.

`sed`

- **Description :** Modification des fichiers en ligne.
- **Syntaxe :**
- `sed -n '/motif/p' fichier`
- **Exemple :**
    - Supprimer les lignes vides :
    - `sed '/^$/d' fichier.txt`

`awk`

- **Description :** Extraction et transformation de données.
- **Syntaxe :**
- `awk '/motif/' fichier`
- **Exemple :**
    - Trouver les lignes contenant "Linux" :

- `awk '/Linux/' fichier.txt`
