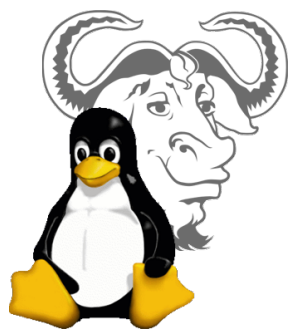
Administration d'un système GNU / Linux

05 — Expressions régulières et groupes

Anthony Labarre

上海师范大学

24 octobre 2024



Groupes de capture

Groupes

- `grep` sur une expression régulière nous donne toutes les lignes qui contiennent un motif correspondant à l'expression;

Groupes

- `grep` sur une expression régulière nous donne toutes les lignes qui contiennent un motif correspondant à l'expression;
- Bien souvent, on veut récupérer le motif, pas la ligne entière;

Groupes

- `grep` sur une expression régulière nous donne toutes les lignes qui contiennent un motif correspondant à l'expression;
- Bien souvent, on veut récupérer le motif, pas la ligne entière;
- Pour cela, on utilise les **groupes (de capture)**, et un autre programme nommé `pcregrep`;

Groupes: définition

- Un **groupe** dans une expression régulière est une partie de l'expression que l'on veut isoler;

Groupes: définition

- Un **groupe** dans une expression régulière est une partie de l'expression que l'on veut isoler;
- On le définit simplement avec des parenthèses;

Groupes: définition

- Un **groupe** dans une expression régulière est une partie de l'expression que l'on veut isoler;
- On le définit simplement avec des parenthèses;

Exemple

Pour isoler les années, mois et jours d'une date, on définit les trois groupes suivants:

(année) (mois) (jour)

```
(\d{4})-([0-1]\d)-([0-3]\d)
```

Groupes: extraction

Une fois les groupes définis, `pcregrep` les extrait avec l'option `-oN`, où `N` est le numéro du groupe.

Exemple

Extrayons divers morceaux des dates du fichier `test.txt`:

```
$ pcregrep -o1 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# toutes les années
```

Groupes: extraction

Une fois les groupes définis, `pcregrep` les extrait avec l'option `-oN`, où `N` est le numéro du groupe.

Exemple

Extrayons divers morceaux des dates du fichier `test.txt`:

```
$ pcregrep -o1 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# toutes les années
```

```
$ pcregrep -o2 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# tous les mois
```

Groupes: extraction

Une fois les groupes définis, `pcregrep` les extrait avec l'option `-oN`, où `N` est le numéro du groupe.

Exemple

Extrayons divers morceaux des dates du fichier `test.txt`:

```
$ pcregrep -o1 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# toutes les années
```

```
$ pcregrep -o2 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# tous les mois
```

```
$ pcregrep -o3 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# tous les jours
```

Groupes: extraction

Une fois les groupes définis, `pcregrep` les extrait avec l'option `-oN`, où `N` est le numéro du groupe.

Exemple

Extrayons divers morceaux des dates du fichier `test.txt`:

```
$ pcregrep -o1 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# toutes les années
```

```
$ pcregrep -o2 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# tous les mois
```

```
$ pcregrep -o3 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# tous les jours
```

```
$ pcregrep -o3 -o2 -o1 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# tous les dates au format DDMMYYYY
```

Groupes: extraction

Une fois les groupes définis, `pcregrep` les extrait avec l'option `-oN`, où `N` est le numéro du groupe.

Exemple

Extrayons divers morceaux des dates du fichier `test.txt`:

```
$ pcregrep -o1 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# toutes les années
```

```
$ pcregrep -o2 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# tous les mois
```

```
$ pcregrep -o3 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# tous les jours
```

```
$ pcregrep -o3 -o2 -o1 "(\d{4})-([0-1]\d)-([0-3]\d)" test.txt
# tous les dates au format DDMMYYYY
```

Essayez avec `echo $(date -I) | pcregrep ...`

Lookarounds

Interdictions et exigences

- Les *lookarounds* permettent de préciser ce qui doit (ou non) se trouver autour d'une expression.

Interdictions et exigences

- Les *lookarounds* permettent de préciser ce qui doit (ou non) se trouver autour d'une expression.
- Exemples d'utilisation:

Interdictions et exigences

- Les *lookarounds* permettent de préciser ce qui doit (ou non) se trouver autour d'une expression.
- Exemples d'utilisation:
  - les noms des fichiers dont l'extension est `.c`;

Interdictions et exigences

- Les *lookarounds* permettent de préciser ce qui doit (ou non) se trouver autour d'une expression.
- Exemples d'utilisation:
  - les noms des fichiers dont l'extension est `.c`;
  - les nombres qui ne sont **pas** négatifs;

Interdictions et exigences

- Les *lookarounds* permettent de préciser ce qui doit (ou non) se trouver autour d'une expression.
- Exemples d'utilisation:
  - les noms des fichiers dont l'extension est `.c`;
  - les nombres qui ne sont **pas** négatifs;
  - ...;

Interdictions et exigences

- Les *lookarounds* permettent de préciser ce qui doit (ou non) se trouver autour d'une expression.
- Exemples d'utilisation:
  - les noms des fichiers dont l'extension est `.c`;
  - les nombres qui ne sont **pas** négatifs;
  - ...;
- Ils utilisent la syntaxe des groupes, mais on peut les utiliser sans groupes;

Interdictions et exigences

- Les *lookarounds* permettent de préciser ce qui doit (ou non) se trouver autour d'une expression.
- Exemples d'utilisation:
  - les noms des fichiers dont l'extension est `.c`;
  - les nombres qui ne sont **pas** négatifs;
  - ...;
- Ils utilisent la syntaxe des groupes, mais on peut les utiliser sans groupes;
- Ils **ne font pas** partie de l'expression;

Positive lookahead

- But: accepter une expression **si elle est suivie de quelque chose**;
- On écrit `(?=expr)` après l'expression régulière;

Positive lookahead

- But: accepter une expression **si elle est suivie de quelque chose**;
- On écrit `(?=expr)` après l'expression régulière;

Exemple (les nombres suivis de `.exe`)

```
\d+(?=.exe)
```

Positive lookahead

- But: accepter une expression **si elle est suivie de quelque chose**;
- On écrit `(?=expr)` après l'expression régulière;

Exemple (les nombres suivis de `.exe`)

```
\d+(?=.exe)
```

Résultat:

"J'aime les fichiers nommés 12345.exe, mais pas les fichiers nommés 12345.zip"

Negative lookahead

- But: accepter une expression **si elle est n'est pas suivie de quelque chose**.
- On écrit `(?!expr)` après l'expression régulière;

Negative lookahead

- But: accepter une expression **si elle est n'est pas suivie de quelque chose**.
- On écrit `(?!expr)` après l'expression régulière;

Exemple (les adjectifs masculins en "u")

```
[a-z]*u(?!e)
```

Negative lookahead

- But: accepter une expression **si elle est n'est pas suivie de quelque chose**.
- On écrit `(?!expr)` après l'expression régulière;

Exemple (les adjectifs masculins en "u")

```
[a-z]*u(?!e)
```

Résultat:

"Il est venu mais elle n'est pas venue."

Positive lookbehind

- But: accepter une expression **si elle suit quelque chose**.
- On écrit `(?<=expr)` avant l'expression régulière;

Positive lookbehind

- But: accepter une expression **si elle suit quelque chose**.
- On écrit `(?<=expr)` avant l'expression régulière;

Exemple (les femmes mais pas les hommes)

```
(?<=Mme. )[A-Z][a-z]*
```

Positive lookbehind

- But: accepter une expression **si elle suit quelque chose**.
- On écrit `(?<=expr)` avant l'expression régulière;

Exemple (les femmes mais pas les hommes)

```
(?<=Mme. )[A-Z][a-z]*
```

Résultat:

"Nous acceptons Mme. Durand, mais pas Mr. Dubois, ni Alexandre ou Marie."

Negative lookbehind

- But: accepter une expression **si elle est ne suit pas quelque chose**.
- On écrit `(?<!expr)` avant l'expression régulière;

Negative lookbehind

- But: accepter une expression **si elle est ne suit pas quelque chose**.
- On écrit `(?<!expr)` avant l'expression régulière;

Exemple (les noms qui ne suivent pas un nom)

```
(?<![A-Z][a-z]*)[A-Z][a-z]*
```

Negative lookbehind

- But: accepter une expression **si elle est ne suit pas quelque chose**.
- On écrit `(?<!expr)` avant l'expression régulière;

Exemple (les noms qui ne suivent pas un nom)

```
(?<![A-Z][a-z]*)[A-Z][a-z]*
```

Résultat:

"my name is Bond . James Bond."