Expressions Rationnelles ou « régulières » : ER

RegEx : Regular Expression

Pour **grep**, **expr**, **sed**, **awk**

et autres

Expressions Rationnelles

- Une ER est une chaîne de caractères
- Ensembles de caractères et/ou méta-caractères qui correspondent ou spécifient des modèles
  - caractères en signification littérale
  - ancres qui désignent la position dans la ligne de texte **^** ou **$**
  - modificateurs : *** [**

ER : caractère '*'

Une expression suivie du caractère '*' désigne 0 ou plusieurs occurrences des caractères désignés par l'expression.

La signification de '*' ne diffère donc pas de la signification habituelle lors de l'interprétation des commandes par le *shell*

« 1133* » correspond à '11' + un ou plusieurs '3' ainsi que d'autres caractères :

113, 1133, 111312 ...

ER : caractère '.'

Le point '.' correspond à un seul caractère, sauf le retour à la ligne (équivalent du '?' du langage de commande)

« 13. » correspond à '13' + au moins un caractère (incluant un espace):

1133, 11333 mais pas 13 (un caractère supplémentaire manquant)

ER : caractères '^' et '$'

- La puissance '^' marque le début d'une ligne mais, quelque fois, selon le contexte, inverse la signification d'un ensemble de caractères dans une ER
- Le signe dollar '$' à la fin d'une ER marque la fin d'une ligne
  - « XXX$ » correspond à XXX à la fin d'une ligne
  - « ^$ » correspond à des lignes vides

ER : caractères '[' et ']'

[...] englobent un ensemble de caractères pour réaliser une correspondance dans une seule ER.

- **[xyz]** : caractères **x**, **y** ou **z**.
- **[c-n]** : tout caractère compris entre **c** et **n**
- **[B-Pk-y]** : tout caractère compris entre **B** et **P** et entre **k** et **y**
- **[a-z0-9]** : toute lettre en minuscule et tout chiffre

ER : caractère \

- L'antislash **'\'** protège ou échappe un caractère spécial, ce qui signifie que le caractère est interprété littéralement
- Un **\$** renvoie la signification littérale du caractère **$** plutôt que sa signification ER de fin de ligne. De même un **** a la signification littérale de ****

Exemples

- ^ab
- [0-9]
- [adh-kA-Z]
- ^\.
- [-d]

ER : caractères \< \>

- Les signes « inférieur » et « supérieur » échappés

  « \<...\> » indiquent les limites du mot.

- Ces signes doivent être échappés, sinon ils n'ont que leur signification littérale.
  - « \<le\> » correspond au mot « le » mais pas aux mots « les », « leur », « belle », etc.

ER : caractères \< \>

```
bash$ cat fichiertexte
This is line 1, of which there is only one instance.
This is the only instance of line 2.
This is line 3, another line.
This is line 4.

bash$ grep 'the' fichiertexte
This is line 1, of which there is only one instance.
This is the only instance of line 2.
This is line 3, another line.

bash$ grep '\<the\>' fichiertexte
This is the only instance of line 2.
```

ER étendues : egrep, awk, perl, ...

- Le point d'interrogation **'?'** correspond à 0 ou 1 instance de la précédente ER. Il est généralement utilisé pour correspondre à des caractères uniques.
- Le signe plus **'+'** correspond à une ou plusieurs occurrences de la précédente ER. Il joue un rôle similaire à **'*'**, mais ne correspond pas à zéro occurrence.

ER étendues : \(...\) et \n

Les commandes **grep** (selon versions) et **awk** permettent de désigner des sous-chaînes de caractères à l'aide des deux type d'expressions suivants:

- **\(expression\)** désigne la même chose que l'expression elle-même
- **\n** où **n** est un entier, désigne la chaîne de caractères qui correspond à la nième expression entourée précédemment par **\(** et **\)**

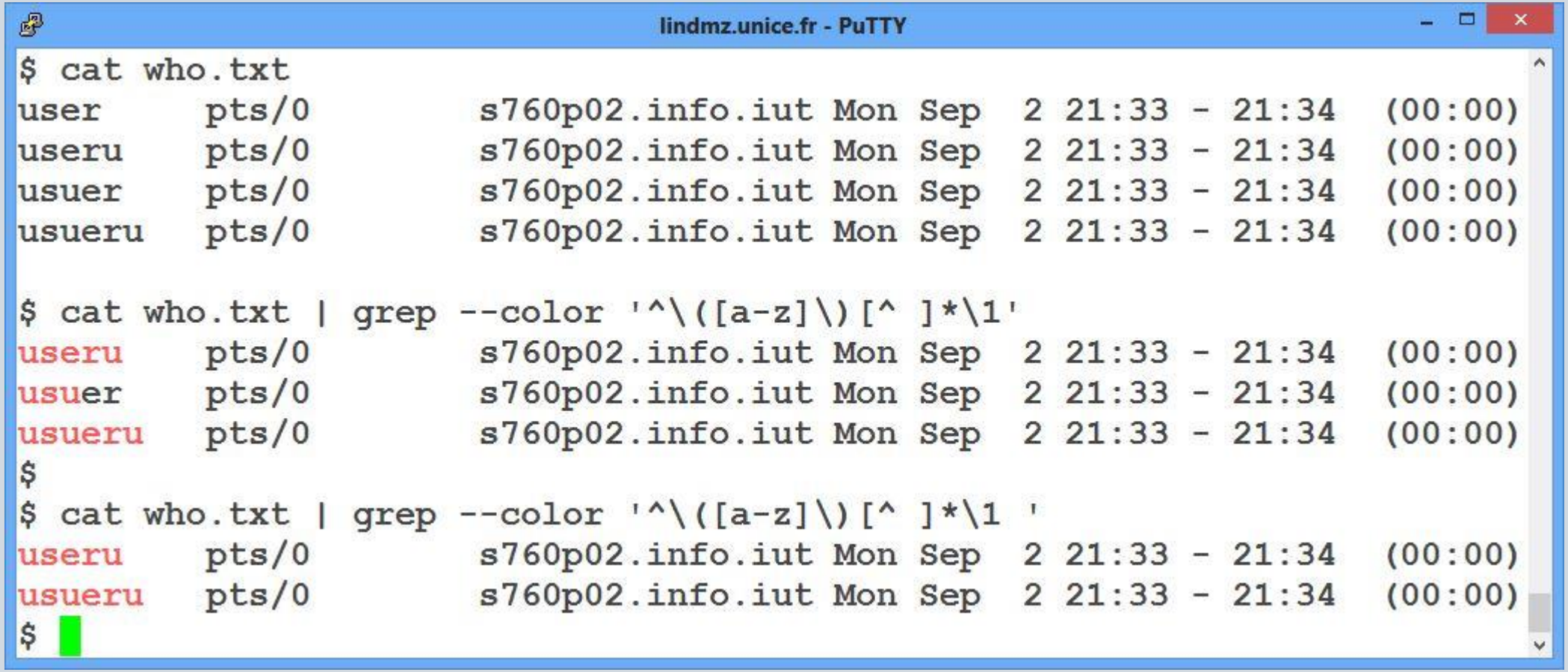
ER étendues : \(...\) et \n

```
$ who   | grep '^\([a-z]\)[^ ]*\1 '
```

Utilisateurs connectés au *host* dont le nom de *login* commence et se termine par la même lettre

Attention, il faut un espace après \1

ER étendues : \(...\) et \n



ER étendues : \{...\}

- **\{m\}** désigne exactement **m** occurrences
- **\{m,\}** désigne au minimum **m** occurrences
- **\{m,n\}** désigne entre **m** et **n** occurrences (bornes incluses)

```
$ echo -e "1\n12\n123\n1234"
1
12
123
1234
$ echo -e "1\n12\n123\n1234" | grep '^[0-9]\{3\}$'
123
```

ER étendues : \{...\}

```
syska@dmz-linserv:~$ grep '.\{4\}'
aaa
aaaa
aaaa
aaaaaa
aaaaaa
```