Cours Unix 4

Michel Mauny

INRIA ROCQUENCOURT

ETGL

Le cours 3 est disponible sur http://quincy.inria.fr/courses/unix/

Plan du cours 4

Un dernier mot sur les processus: la primitive exec

```
exec [commande [arg ...]]
```

La commande exec exécute commande **à la place** du processus (shell) courant:

- pas de création de nouveau processus
- arg, ... deviennent les arguments de commande
- si commande n'existe pas, le shell termine avec un code de retour non nul (127, par exemple).

Seuls les descripteurs de fichiers 0, 1 et 2 sont transmis à la commande.

Sans argument, les redirections imposées à exec sont permanentes et appliquées au shell courant.

La primitive exec: exemple

```
$ ksh
   $ exec 1>/tmp/out
   $ pwd
   $ ^D
$ cat /tmp/out
 /home/tom/cours/etgl/slides
```

Environnement: `stty`

Change ou imprime la configuration du terminal.

- `stty -a` imprime toutes les caractéristiques
- `stty eof` c *end-of-file* (`^D`)
- `stty erase` c effacement du dernier caractère entré (`^?`, ou `^H`)
- `stty kill` c effacement de la ligne entière (`^U`)
- `stty intr` c caractère d'interruption (`^C`)
- `stty quit` c envoi du signal `QUIT` (`^\`)
- ...

Expressions régulières

Des **motifs** représentant des **ensembles de chaînes**

L'expression régulière	représente par exemple	mais pas
`abc`	`abc`	`abd`
`ab*c`	`ac`, `abc`, `abbc`	`abdc`
`a.c`	`abc`	`ac`

Les expressions régulières sont utilisées par de nombreux outils Unix (`grep`, `ed`, `sed`, `vi`, `emacs`, ...).

Attention: le caractère générique `*` du shell se comporte différemment du caractère `*` des expressions régulières.

Expressions régulières

Description mathématique:

- un **alphabet** $\mathcal{A}$ (pour faire des mots ou chaînes)
- le **mot vide** (noté ϵ)

Une expression régulière r est ou bien:

- l'expression vide ϵ
- une lettre a de l'alphabet $\mathcal{A}$
- une concaténation d'expressions régulières $r_1 r_2$
- une alternative $r_1 \mid r_2$
- l'itération $r*$

Expressions régulières

Une expression régulière r désigne un ensemble de mots E. On note $\mathcal{L}(r) = E$.

- $\mathcal{L}(\epsilon) = \{\epsilon\}$
- $\mathcal{L}(a) = \{a\}$, où $a \in \mathcal{A}$
- $\mathcal{L}(r_1 r_2) = \{m_1 m_2$ avec $m_1 \in \mathcal{L}(r_1)$ et $m_2 \in \mathcal{L}(r_2)\}$
- $\mathcal{L}(r_1 \mid r_2) = \mathcal{L}(r_1) \cup \mathcal{L}(r_2)$
- $\mathcal{L}(r*) = \{\epsilon\} \cup \mathcal{L}(r) \cup \mathcal{L}(rr) \cup \ldots$
  $= \bigcup_{i=0}^{\infty} \mathcal{L}(r^i)$

Expressions régulières

Exemples:

- $\mathcal{L}(ab) = \{m_1 m_2$ avec $m_1 \in \{a\}$ et $m_2 \in \{b\}\} = \{ab\}$
- $\mathcal{L}(ab* c) = \{ac, abc, abbc, abbbc, \ldots\}$

On peut alors définir:

- `.` comme l'alternative sur toutes les lettres de l'alphabet $a_1 \mid a_2 \mid \ldots$
- $r+$ comme $rr*$
- *etc.*

Expressions régulières

Syntaxe des expressions régulières utilisées par ed, vi, grep, sed:

- `.` désigne un caractère quelconque
- $c*$ représente 0 ou plus répétitions du caractère précédent
- `[...]` représente l'un quelconque des caractères à l'intérieur des crochets
- `[^...]` représente un caractère quelconque **différent** de ceux à l'intérieur des crochets
- c{`n,m`}`}` représente un nombre compris entre n et m itérations de ce qui précède

Expressions régulières

Syntaxe des expressions régulières utilisées par ed, vi, grep, sed:

- `^`r représente le début de ligne
- r`$` représente la fin de ligne
- `\`c annule l'interprétation du caractère c
- c`+` représente une occurrence ou plus
- c`?` représente zéro ou une occurrence
- r_1`|`r_2 représente l'alternation
- `(`r`)` sert à grouper

Exemples

- `^...$` représente toutes les lignes contenant exactement 3 caractères
- `^$` représente toutes les lignes vides
- `[1-9]\.[1-9]` représente toutes les chaînes de 3 caractères: deux chiffres séparés par un point
- `^[a-zA-Z]*:.*` représente toutes les lignes commençant par une chaîne alphabétique éventuellement vide, suivie du caractère ":"

La commande grep

Filtre sélectionnant les lignes correspondant à une expression régulière.

```
grep [option] regexpr [fichier ...]
```

Options:

- `-i` ignore la casse
- `-c` n'imprime que le nombre de lignes sélectionnées
- `-v` imprime les lignes **non représentées** par l'expression régulière

Code de retour:

- 0: des lignes ont été sélectionnées
- 1: aucune ligne ne correspond

La commande grep

La commande egrep interprète ("comprend") les métacaractères ?, +, {, |, (, et)

La commande grep ne les comprend pas.

La commande fgrep ne comprend aucun métacaractère (elle ne cherche que des chaînes, et ne les interprète pas comme des motifs).

La commande sed

sed (*stream editor)* est une sorte d'éditeur à la ed, mais **non interactif**.

```
sed [options] [script-file] [file-list]
```

Options:

- `-f` lit les commandes à appliquer depuis `script-file`
- `-n` n'imprime pas les lignes de `file-list` dans la sortie standard

La commande sed

Étapes de traitement par sed:

- sed lit une ligne de l'entrée
  - sed lit la première commande du script, et, si il y a filtrage, exécute l'action
  - sed lit la prochaine commande, et, si filtrage, exécute l'action correspondante
  - Ce processus est répété jusqu'à ce que toutes les commandes du script soit traitées
- idem pour la ligne suivante
- et ainsi de suite

Note: sed **ne modifie pas** les fichiers qu'il traite, il imprime des résultats dans la sortie standard.

La commande sed

Exemples:

- `sed '/abc/ p' file` imprime le contenu de `file` ainsi que les lignes contenant la chaîne `abc`. (On peut utiliser l'option `-n` pour n'avoir que les lignes contenant le motif.)
- `sed '5,15d' file` efface les lignes de 5 à 15 de la sortie standard
- `sed '/abc/d' file` omet d'imprimer les lignes contenant le motif.
- `sed 's/motif/chaine/' file` substitue dans chaque ligne la première occurrence de `motif` par `chaine`

La commande sed

- `sed '10q' file` affiche les 10 premières lignes et s'arrête
- `sed '/motif/q'` affiche toutes les lignes jusqu'à la première filtrée par `motif` et s'arrête
- `sed '/motif/d' file` identique à `grep -v 'motif' file`

La commande sed

- `sed '/motif1/s/motif2/chaine/g' file` n'effectue la substitution que sur les lignes contenant une occurrence de `motif1` (et autant de fois que nécessaire sur la même ligne)
- `sed '/moti1f/s//chaine/g' file` si `motif2` est omis, vaut `motif1` par défaut
- `sed -n '10,20s/motif/chaine/p' file` n'imprime pas les lignes par défaut, sauf celles qui sont filtrées (lignes de 10 à 20)

La commande sed

- `a\text` ajoute `text`, où chaque retour à la ligne est précédé par \
- `i\text` insère `text`, où chaque retour à la ligne est précédé par \
- `q` s'arrête. Si pas d'option `-n`, la ligne est tout de même imprimée.
- `r filename` ajoute le contenu de `filename`
- `w filename` écrit la ou les lignes sélectionnées dans `filename`