03 - Userspace

Gaétan Richard
gaetan.richard@unicaen.fr

L2 – S4 2011/2012

1. Autour du système d'exploitation

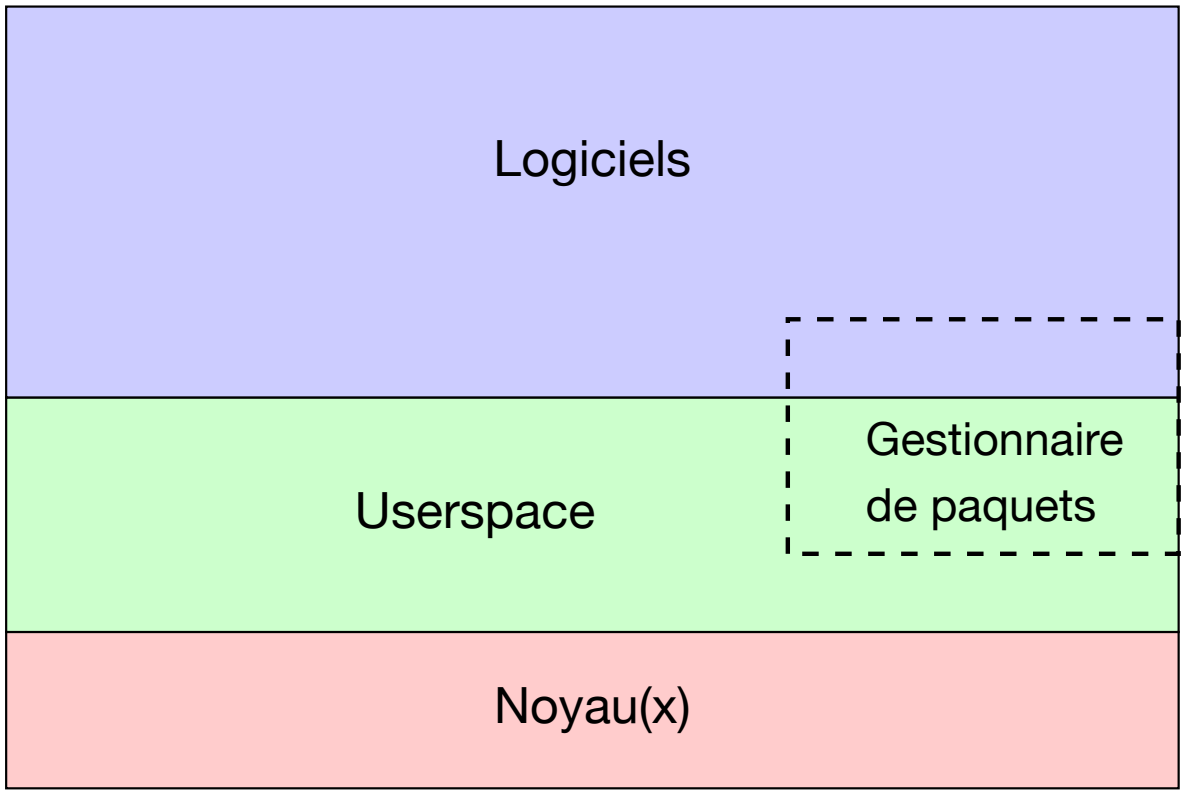
Architecture

Un ordinateur est composé :

- d'un (ou plusiuers) microprocesseur ;
- de mémoire vive (RAM) ;
- de mémoire morte (disque dur, SDD) ;
- de périphériques,
- …

Distribution :

Une distribution est un ensemble de logiciels formant un tout cohérent et prêts à installer.



Noyau

Il existe différents noyaux :

- Linux
- Mach (XNU)
- NT
- Noyau BSD
- Hurd
- …

Userspace

En sus du noyau, un système UNIX possède un ensemble d'utilitaires de base permettant de réaliser de nombreuses opérations.

Dans ce domaine, on rencontre principalement deux grandes visions : System V (GNU, …) et *BSD* (Mac OS X, …).

Ces deux visions présentent de nombreux points commun ainsi que des différences marquées. Cependant, de nombreux efforts sont fait pour unifier les deux approches et les rendre compatibles.

2. Retour sur les expansions

Expressions

On les construit à partir des bases suivantes

- les caractères ;
- **?** (un caractère) ;
- ***** (un nombre quelconque de caractères) ;
- **[ ]** (un caractère parmi ceux à l'intérieur) ;
- **[!]** (un caractère sauf ceux à l'intérieur) ;
- **** (protège un caractère précédent).

Note : entre crochets, il est possible d'utiliser un tiret pour dénoter un intervalle (ex : **[a-d]** est équivalent à **[abcd]**).

Exemples

En direct

Substitutions

Dans les substitutions, il est possible d'utiliser des expressions comme définies précédemment. Dans ce cas la notion de plus court / plus long motif apparaît.

Exemples :

```
$ test=”une chaine assez longue”
$ echo ${test#*n}
e chaine assez longue
$ echo ${test##*n}
gue
$
```

Retour sur les substitutions et les protections

Substitutions :

- **${*var*}** : même chose que sans les accolades ;
- **${#*var*}** : longueur du contenu de la variable *var* ;
- **${*var*%*motif*}** : valeur de la variable *var* auquel au retire le plus petit suffixe *motif* ;
- **${*var*#*motif*}** : valeur de la variable *var* auquel au retire le plus petit préfixe *motif* ;
- **${*var*%%*motif*}** et **${*var*##*motif*}** : même chose que précédemment avec le plus grand.

Remarque : Si on veut parler du caractère *, il faut le protéger.

3. Commandes basiques

Date

Fonction :

- affiche la date ;
- (avec un argument et les bon droits) change la date.

Exemples :

```
$ date
Lun 25 jan 2010 17 :07 :26 CET
$ date 1710
date : bind : Permission denied
date : settimeofday (timeval) : Operation not permitted
```

Pwd

Fonction : affiche le répertoire courant.

Exemple :

```
$ pwd
/Users/grichard/Enseignement/2009-2010/L2-Systeme/TD
$
```

Uname

Fonction : affiche le système d'exploitation courant.

Options : *-a* plus d'informations

Exemples :

```
$ uname
Darwin
$ uname -a
Darwin britten.local 10.6.0
Darwin Kernel Version 10.6.0 :
Wed Nov 10 18 :11 :58 PST 2010;
root :xnu-1504.9.26~3/RELEASE_X86_64 x86_64
```

True / False

Fonction :

- **true** : retourne vrai ;
- **false** : retourne faux ;

Exemples :

```
$ if true ; then echo ”vrai” ; else echo ”faux” ;fi
vrai
$ if false ; then echo ”vrai” ; else echo ”faux” ;fi
faux
$ true && echo toto
toto
$ true || echo toot
$
```

Wc

Fonction : affiche le nombre de lignes / de mots / de caractères du fichier donné en argument ou de l'entrée standard.
Options : `-l` affiche juste le nombre de lignes

Exemples :

```
$ wc CM-02.tex
    1154    2937   26513 CM-02.tex
$ wc -l CM-02.tex
    1154 CM-02.tex
$
```

Units

Fonction : convertisseur de mesures

Exemples :

```
$  units
500 units, 54 prefixes
You have : 10 gallons
You want : litres
* 37.854118
/ 0.026417205
$
```

4. Affichage

Echo

Fonction : affiche le texte donné en argument.

Exemples :

```
$ echo $HOME est loin
/Users/grichard est loin
$
```

Cat

Fonction : affiche le contenu du fichier donné en argument ou l'entrée standard.

Options : -n (resp. -b) numérote toutes les lignes (resp. les lignes non blanches).

Exemples :

```
$ cat -n sav
     1 toto
     2 yu
$
```

More / less

Fonction : affiche l'argument ou l'entrée standard en s'arrêtant si nécessaire.

Note : less dispose de fonctionnalités supplémentaires, en particulier celle de pouvoir « revenir » dans l'affichage.

Head

Fonction : affiche le début du contenu du fichier donné en argument ou l'entrée standard.

Options : `-n count` affiche les *count* premières lignes.

Exemples :

```
$ head -n 10 CM-01.tex
\documentclass[french]{beamer}

\usetheme{Gaet}

\usepackage{amsthm}

\usepackage{colortbl}
\arrayrulecolor{blue}

\usepackage{stmaryrd,amsmath}
$
```

Tail

Fonction : affiche la fin du contenu du fichier donné en argument ou l'entrée standard.

Options :

- `-n count` affiche les *count* dernières lignes.
- `-n +count` affiche les lignes à partir de la ligne *count*.

Exemples :

```
$ tail -n 4 /var/log/system.log
Jan 26 09 :11 :34 britten [0x0-0x428428].net.sourceforge.skim-app.skim[2391] : wa
Jan 26 09 :12 :48 : --- last message repeated 5 times ---
Jan 26 09 :12 :38 britten [0x0-0x428428].net.sourceforge.skim-app.skim[2391] : wa
Jan 26 09 :15 :18 : --- last message repeated 5 times ---
$
```

Diff

Fonction : affiche les différences entre deux fichiers.

Exemple :

```
$ diff CM-03.tex CM-3.tex.old
363c363
< \begin{frame}[fragile]
---
> \begin{frame}
371,372c371,373
<   \bftext{Option :} \verb+-u+ affiche le contexte.
<
---
...
$
```

5. Filtres

Recode

Fonction : Change le système de codage (fonctionne également avec un fichier comme argument)

Exemple :

```
$ recode data..base64
régulier
csOpZ3VsaWVyC
$ recode latin1..utf-8 fichier.html
$
```

Grep

Fonction : Permet de sélectionner les lignes possèdent un motif particulier de l'entrée standard.

Options :

- `-v` inverse le comportement ;
- `-B n`,`-A n` affiche également les *n* lignes précédentes (suivantes).

Exemple :

```
$ cat /var/log/system.log | grep invalid
Jan  5 10 :48 :05 britten [0x0-0x139139].com.apple.Preview[9065] : invalid AES bloc
Jan  5 10 :48 :05 britten [0x0-0x139139].com.apple.Preview[9065] : invalid AES bloc
Jan 10 18 :00 :51 britten kernel[0] : jnl : disk1s1 : close : journal
0x6f4ed10, is invalid.  aborting outstanding transactions
...
$
```

Cut

Fonction : Permet de sélectionner des champs particuliers parmi les lignes de l'entrée standard.

Options :

- `-d delim` utilise *delim* comme délimiteur ;
- `-f list` renvoit les champs présents dans *liste*.

Exemple :

```
$ cat /etc/passwd | cut -d ' :' -f 3,6
...
60 :/var/empty
65 :/var/empty
67 :/var/empty
70 :/Library/WebServer
71 :/var/empty
...
$
```

Sort

Fonction : trie les lignes d'un fichier passé en argument ou de l'entrée standard.

Options :

- `-n` considère l'entrée comme des entiers ;
- `-u` supprime les doublons.

Exemple :

```
$ ls -f | sort
.
..
case.sh
for.sh
if.sh
log
script.py
$
```

Uniq

Fonction : supprime les doublons d'un fichier trié

Option : `-c` affiche le nombre de lignes identiques.

Exemple :

```
$ cat /var/log/system.log | cut -f 2 -d ' ' | sort -n | uniq

10
11
...
25
26
$
```

Tr

Fonction : change les caractères.

Exemples :

```
$ echo valise | tr ae àé
vàlisé
$ echo ”abcdefghij?,?azerty” | tr ”a-e?” ”12345x”
12345fghijx,x1z5rty
$
```

Sed

Fonction : effectue des substitutions de texte et plus encore.

Options : `-i` faire le remplacement dans le fichier.

Exemples :

```
$ echo ”valise a balle” | sed s/al/bm/
vbmise a balles
$ echo ”valise a balle” | sed s/al/bm/g
vbmise a bbmles
$
```

6. Expressions régulières

Présentation

Dans certains cas, on souhaite pourvoir décrire facilement des *schémas* indiquant des ensembles de noms. Pour cela, on introduit la notion d'**expression régulière** qui repose sur les objets suivants :

- **[ ]** comme dans les expression du shell ;
- **|** pour désigner le ou ;
- **.** pour désigner un caractère quelconque ;
- **( )** pour grouper ;
- **+** pour indiquer une ou plusieurs répétions ;
- ***** pour indiquer des éventuelles répétitions ;
- **?** pour indiquer une présence éventuelle.

Attention : Dans le cadre des expressions régulières, les symboles ? et * n'ont pas la même signification que pour le shell.

Classes de caractères

Il existe également une notation usuelle pour désigner de façon portable et simple des classes de caractères :

- `[ :space :]` Espace blanc ou séparateur de ligne ou de paragraphe ;
- `[ :alnum :]` Caractère alphanumérique ;
- `[ :digit :]` Chiffre décimal ;
- `[ :alpha :]` Caractère alphabétique ;
- `[ :lower :]` Lettre minuscule ;
- `[ :upper :]` Lettre majuscule.

Exemples

- `-?[[ :digit :]]]+(,[[ :digit :]]*)?`
- `.*\.t(xt|ex)`
- ...

Extensions

Certain programmes acceptent également les syntaxes suivantes :

- **{ , }** pour indiquer le nombre de répétitions ;
- **\1**, **\2**, …pour parler du $n^{\text{ième}}$ éléments rencontré.

Note : La dernière extension perd la rapidité des expressions régulières. Il faut donc l'utiliser avec parcimonie.

Egrep

Principe : La commande grep comprend les expressions régulières. Pour cela, il suffit de l'appeler avec l'option -e ou d'utiliser l'alias **egrep**.

Note : Certaines versions de **grep** passent automatiquement en mode expressions régulières sous certaines conditions.

Exemple :

```
$ cat /var/log/system.log | egrep "08 :[0-9]* :[0-9]*.* USER_PROCES"
Jan  3 08 :49 :41 britten loginwindow[42673] : USER_PROCESS : 42673 console
Jan  4 08 :25 :44 britten loginwindow[43441] : USER_PROCESS : 43441 console
Jan  4 08 :30 :03 britten loginwindow[34] : USER_PROCESS : 34 console
Jan  4 08 :32 :27 britten loginwindow[34] : USER_PROCESS : 34 console
Jan  4 08 :43 :46 britten loginwindow[34] : USER_PROCESS : 34 console
...
```

Sed (bis)

Principe : sed accepte les expressions régulières et permet d'utiliser les valeurs **\1**, …dans les remplacements.

Note : grep demande de protéger les parenthèses par des échappements (\)

Exemple :

```
$ sed "s/a\(.*\)a\(.*\)b/\2\1/"
a234a6b
6234
$
```

Awk

Principe : awk est un outils très performant de manipulation de texte qui dispose de son propre langage.

Note : Nous n'en parlerons malheureusement pas plus en détail dans le cours.

7. Éditeurs

Caractéristiques

Les éditeurs modernes présentent un certains nombre de fonctionnalités :

- coloration syntaxique ;
- indentation automatique ;
- correcteur orthographique ;
- …

Emacs

Histoire : Écrit en 1976 par Richard Stallman. La version la plus connue est GNU Emacs.

Principe :

Un éditeur de texte en mode édition qui possède un très grand nombre de raccourcis et d'extensions au travers du langage LISP.

Documentation :

Il est possible d'obtenir de la documentation :

- de façon externe ;
- à l'aide de la commande **info emacs** ;
- dans le logiciel à l'aide des outils d'aide (`ctrl-h`).

Vi

Histoire : Écrit par Bill Joy en 1976. Il en existe maintenant de très nombreuses versions. Fait partie de la spécification UNIX.

Principe :
Un éditeur de texte très léger qui alterne entre le mode commande et le mode édition.

Documentation :

- de façon externe ;
- à l'aide de la commande **man vi** ;
- dans le logiciel à l'aide des outils d'aide (`:help`).

Vim / gvim

Ce sont certaines des nombreuses versions de vi. Très souvent, la commande vi pointe vers l'un d'entre eux.