Systèmes informatiques

Franck Guingne,
sur la base du cours d'Olivier Lecarme

Cours Licence 1; Semestre 2

2009–2010

Huitième cours : Personnalisation et extension du shell

Plan en cours

Historique des commandes

- utiliser l'historique des commandes permet d'éviter de saisir à nouveau des commandes complexes
- le mécanisme dèjà vu est propre à Zsh
- les deux mécanismes à venir utilisent le fichier d'historique
- son nom est donné par la variable `HISTFILE`
- la commande `fc` vient de Ksh
- la *substitution historique* vient de Csh

La commande `fc`

- le comportement de la commande est déterminé par ses options
  - `-l` affiche les commandes spécifiées
  - `-s` *`ancien nouveau`* remplace une chaîne par une autre avant d'exécuter la commande spécifiée
  - `-e` *`éditeur`* utilise l'éditeur pour modifier les commandes spécifiées
  - `-r` inverse l'ordre des commandes
- on spécifie les commandes par :
  - leur numéro, donné par la commande `fc -l`
  - leur rang en remontant dans l'historique (valeur négative)
  - une chaîne figurant dans la commande

Exemples d'utilisation

- `fc -l -1 -4` liste les quatre dernières commandes
- `fc -e ex sftp`
  - appelle l'éditeur EX pour modifier la dernière commande contenant la chaîne `sftp`
  - à la sortie de l'éditeur, soumet la commande modifiée
- `fc -s polymnie=deptinfo sftp`
  - prend la dernière commande contenant la chaîne `sftp`
  - y remplace la chaîne `polymnie` par la chaîne `deptinfo`
  - soumet la commande modifiée

Substitution historique

- mécanisme très complet, mais difficile à utiliser
- on réutilise des fragments de commandes précédentes pour composer une nouvelle commande
- la nouvelle commande est proposée pour soumission
- forme générale :
  `![`*`événement`*`][[:]`*`sélecteur`*`[:`*`modificateur...`*`]]`
- les crochets dénotent une partie facultative
- l'*événement* spécifie l'ancienne commande
- le *sélecteur* choisit un ou plusieurs mots de cette commande
- les *modificateurs* permettent de modifier les mots

Spécifications d'événements

- `!!` désigne la commande précédente
- `!`*`n`* est la commande numéro n
- `!-`*`n`* est la n-ième commande précédente
- `!`*`chaîne`* est la commande précédente commençant par la *chaîne*
- `!?`*`chaîne`*`?` est la commande précédente contenant la chaîne
- exemples :
  - `!sftp` : dernière commande commençant par `sftp`
  - `!?polymnie?` : dernière commande contenant `polymnie`

Sélecteurs

- `:0` désigne le nom de la commande
- `:^` le premier argument
- `:$` le dernier
- `:`*n* le *n*-ième
- `:`n_1`-`n_2 les mots n_1 à n_2
- `:*` tous les mots
- `:`*n*`*` les mots *n* au dernier
- `:`*n*`-` les mots *n* à l'avant-dernier
- `:-`*n* les mots 0 à *n*
- `:%` mot correspondant à la précédente `?`*chaîne*`?`
- exemples
  - `!!:0` : nom de la commande précédente
  - `!sftp:2*` : du deuxième au dernier argument de la dernière commande commençant par `sftp`

Modificateurs

- `:r` retire le suffixe final du premier mot sélectionné
- `:h` retire le dernier élément du chemin (du dernier `/` à la fin du mot)
- `:[g]s/`*chaîne*$_1$`/`*chaîne*$_2$ remplace la *chaîne*$_1$ par la *chaîne*$_2$; si `g` est present, remplace partout
- `:[g]&` répète le remplacement précédent
- `:q` met les mots entre guillemets
- exemples
  - `!152:2:h` : chemin menant au fichier du deuxième argument de la commande 152
  - `!147:*:s/Sample/Truc` : tous les arguments de la commande 147, avec substitution

Plan en cours

Adaptation de l'achèvement automatique

- le dictionnaire d'achèvement n'est pas le même pour le nom de la commande et les arguments
- on peut avoir des dictionnaires dépendant de la commande
- la commande `compctl` détermine le comportement de l'achèvement
- trois formes :
  - `compctl` *`options commandes`* ne s'applique qu'aux commandes énumérées
  - `compctl -D` *`options`* s'applique aux autres commandes
  - `compctl -C` *`options`* s'applique au nom de la commande

Options de la commande

- pour fournir un dictionnaire particulier :
  - `-f` noms de fichiers
  - `-c` noms de commandes
  - `-o` noms d'options du shell
  - `-v` noms de variables
  - `-j` noms de tâches
  - `-u` noms d'utilisateurs
- pour construire un dictionnaire :
  - `-k` *`tableau`* noms à prendre dans le tableau
  - `-g` *`modèle`* noms à prendre dans un modèle utilisant des jokers
  - `-K` *`fonction`* noms fournis par la fonction
  - `-H` *`n modèle`* noms cherchés dans les n dernières lignes du fichier d'historique

Exemples d'utilisation

- `compctl -g '*.ps *.eps *.epsf' ghostview gs`
  les commandes `ghostview` et `gs` achèvent leurs arguments parmi les fichiers ayant les suffixes indiqués
- `compctl -j -P % -x 's[-] p[1]' -k signals - kill`
  la commande `kill` complète ses arguments à partir des noms de tâches ; le signe `%` est ajouté automatiquement

Les variables prédéfinies

- les *variables prédéfinies* sont affectées ou utilisées par Zsh
- ce sont des variables d'environnement ordinaires, au nombre d'une centaine
- les plus simples sont également reconnues par les shells plus simples

Variables affectées par Zsh

- `#` : nombre de paramètres du script
- `$` : numéro de processus de ce shell
- `*` : tableau des paramètres
- `@` : liste des paramètres
- `HOST` : nom de la machine
- `LINENO` : numéro de ligne dans le script
- `OLDPWD` : nom du répertoire précédent
- `PWD` : nom du répertoire courant
- `RANDOM` : nombre aléatoire entre 0 et 32767
- `USERNAME` : nom d'utilisateur

Variables utilisées par Zsh

- `RANDOM` : germe du générateur aléatoire
- `CDPATH` : liste de noms de chemins pour abréger la commande `cd`
  - si un nom après `cd` ne contient pas de `/`
    et n'est pas visible dans le répertoire courant
    on le prend dans cette liste
- `HISTFILE` : nom du fichier de sauvegarde de l'historique des commandes
- `HISTSIZE` : taille maximale de ce fichier
- `HOME` : valeur par défaut pour `cd`, utilisée aussi par beaucoup de commandes
- `LANG` : langue utilisée pour les messages, les nombres, la date, etc.
  - la commande `locale -a` permet de connaître les langues disponibles

Variables utilisées par Zsh

- `PS1` : invite principale, affichée avant la lecture de chaque commande
  - la chaîne peut contenir des commandes de présentation ou d'affichage de valeurs
  - `%d` répertoire courant
  - `%~` même chose, sous la forme `~`*`nom`* si possible
  - `%c` dernier composant du répertoire courant
  - `%!` numéro d'événement courant
  - `%m` nom de la machine
  - `%U...%u` souligner
  - `%B...%b` caractères gras
  - `%T` heure
  - `%n` nom de l'utilisateur
- `PS2` : invite secondaire, si la ligne est incomplète
- `RPS1` : invite principale, à droite de la fenêtre

Options d'exécution

- les *options d'exécution* sont booléennes
- mises en fonction par `setopt`
- hors fonction par `unsetopt`
- majuscules et minuscules équivalents
- les caractères `_` sont ignorés
- le préfixe `no` nie l'option
- `setopt noglob` est équivalent à `unsetopt glob`
- une centaine d'options
- `setopt` liste celle qui sont en fonction

Options principales

- `all_export` : exporter automatiquement toutes les variables
  - à n'utiliser que provisoirement, dans un script
- `auto_cd` : si on appelle une commande qui n'existe pas, mais qu'il existe un répertoire de ce nom, faire `cd` vers ce répertoire
- `auto_list` : lister les différents choix d'un achèvement automatique
- `auto_menu` : après un deuxième `TAB`, lister les différents choix
- `auto_name_dirs` : après la commande `toto=`*`répertoire absolu`*, `~toto` devient le nom de ce répertoire
- `cd_able_vars` : si l'on essaie d'aller vers un réperoire non absolu et introuvable, essayer en le préfixant par `~`

Options principales (suite)

- `clobber` : permettre que `>` efface un fichier existant et que `>>` en crée un
- `correct` : corriger les fautes de frappe pour les commandes
- `correct_all` : corriger aussi pour les arguments
- `equals` : la notation `=commande` est remplacée par le nom absolu de la commande
- `glob` : faire fonctionner le mécanisme des jokers
- `hist_verify` : permettre de corriger la commande avant de la soumettre dans une substitution historique
- `ignore_eof` : ne pas terminer le shell en fin de fichier
- `menu_completion` : en cas d'achèvement ambigu, proposer tout de suite le premier choix, puis le suivant au prochain `TAB`
- `zle` : permettre l'édition des commandes

Plan en cours

Définition de fonction

- forme de la définition :
  ```
  function nom {
      liste de commandes
  }
  ```
- la fonction est conservée directement dans le shell
- on l'appelle comme une commande ordinaire, mais cela ne crée pas de sous-shell
- paramètres utilisables avec la notation `$1 $2` etc.
- sortie de la fonction par la commande `return`
- exemple :
  ```
  function bof { cd ../truc ; }
  ```
- les blancs sont nécessaires
- le point-virgule peut être remplacé par un passage à la ligne
- la commande `unfunction` supprime la définition

Alias

- un *alias* est une abréviation pour une commande complète
- `alias` *`nom=chaîne`*
- la commande seule donne la liste des alias
- si la définition de l'alias se termine par un blanc, Zsh cherche un alias dans le mot qui suit
- `unalias` *`nom`* supprime la définition
- un alias est plus simple qu'une fonction, et n'a pas de paramètres
- Exemples :

```
alias ls="ls -CF"
alias rm="rm -i"
```

- pour utiliser la commande cachée par l'alias, on la précède par `\`

Attributs des variables

- les *attributs* des variables permettent de déclarer leurs caractéristiques
- la commande `typeset` sert à :
  - créer une (ou plusieurs) variable(s)
  - fixer certains de ses attributs
  - éventuellement lui affecter une valeur
- forme : `typeset [attributs] [nom[=valeur]]...`
- les attributs sont des indicateurs, en fonction si précédés du signe `-`, hors fonction si précédés du signe `+`

Attributs des variables (suite)

- `L` : justifier à gauche
- `L`*n* : justifier sur *n* caractères
- `R` : justifier à droite (et `R`*n*)
- `RZ` : remplacer les espaces de tête par des zéros
- `u` : convertir en majuscules
- `l` : convertir en minuscules
- `i` : entier
- `r` : valeur non modifiable
- `x` : exporter automatiquement
- etc.

Scripts d'initialisation

- un *script* est un fichier de commandes pour un shell
- les plus importants sont exécutés automatiquement pendant le processus de démarrage de Zsh, dans l'ordre suivant :
  - `/etc/zsh/zshenv` est toujours exécuté, pour fixer des caractéristiques locales
  - `$HOME/.zshenv` est exécuté pour tout type de shell
  - si c'est un *shell de connexion*, Zsh exécute `/etc/zsh/zprofile` puis `$HOME/.zprofile`
    - un shell de connexion est lancé par `login`, `rlogin` ou `ssh`
  - si c'est un *shell interactif*, Zsh exécute `/etc/zsh/zshrc` puis `$HOME/.zshrc`
    - un shell exécutant un script n'est pas interactif
  - si c'est un shell de connexion, Zsh exécute `/etc/zsh/zlogin` puis `$HOME/.zlogin`

Scripts d'initialisation (suite)

- en fin de session du shell, Zsh exécute `/etc/zsh/zlogout` puis `$HOME/.zlogout`
- beaucoup de ces fichiers n'existent généralement pas :
  - les fichiers locaux sont rarement utilisés
  - il est rare qu'on ait besoin à la fois de `zprofile` et de `zlogin`
    - un shell de connexion est toujours interactif, l'inverse n'est pas vrai
  - les utilisations de `zlogout` sont rares
- `zshenv` définit des alias et des fonctions, choisit des fonctions
- `zprofile` définit des variables d'environnement
- `zshrc` définit d'autres variables et choisit des options spécifiques

Exemples de fichiers d'initialisation

- fichier `/etc/zsh/zshenv` :

```
export PATH="/usr/local/bin :/usr/local/sbin :\
/bin :/usr/bin :/usr/sbin :/usr/bin/X11 :/usr/X11R6/bin
/usr/games :/sbin :$HOME/bin"
export LANG="fr_FR"
export VISUAL=ex
export EDITOR=ex
umask 022
limit coredumpsize 0
```

- fichier `/etc/zsh/zprofile` :

```
PATH=''$PATH :$HOME/bin''
export PATH
```

Exemples de fichiers d'initialisation (suite)

- fichier `/etc/zsh/zshrc` :

```
alias ls='ls -classify -tabsize=0 -literal\
-color=auto -show-control-chars -human-readable'
alias cp='cp -interactive'
alias mv='mv -interactive'
alias rm='rm -interactive'
alias ll='ls -l'
alias la='ls -a'
alias lla='ls -la'
alias c='clear'
alias less='less -quiet'
alias s='cd ..'
alias df='df -human-readable'
alias du='du -human-readable'
unsetopt beep
```

Exemples de fichiers d'initialisation (fin)

- fin du fichier `/etc/zsh/zshrc` :

```
unsetopt hist_beep
unsetopt list_beep
unsetopt clobber
unsetopt ignore_eof
unsetopt rm_star_silent
setopt auto_remove_slash
setopt glob_dots
export HISTORY=100
export HISTFILE=$HOME/.history
zstyle ' :completion :*' matcher-list ''
'm :a-z=A-Z'
zstyle ' :completion :*' max-errors 3 numeric
zstyle ' :completion :*' use-compctl false
autoload -U compinit
compinit
```

Exemples de fichiers d'initialisation (suite)

- fichier `/etc/zsh/zlogin` :

```
uname -a
uptime
mesg y
```

- fichier `/etc/zsh/zlogout` :

```
clear
```