CHUYÊN ĐỀ HỆ ĐIỀU HÀNH LINUX

Tuần 6: Cú pháp cơ bản trong ngôn ngữ bash (phần 2)

GVLT: NGUYỄN Thị Minh Tuyền

Nội dung

1. Các toán tử trên biến
2. Hàm
3. Thay thế lệnh
4. Debug

Nội dung

1. **Các toán tử trên biến**
2. Hàm
3. Thay thế lệnh
4. Debug

Các toán tử trên biến [1]

- `${var-val}` → `$var` nếu được định nghĩa, nếu không `val`
- Ví dụ:

```
$ a="ga"
$ echo "${a-meu}"
ga
$ unset a
$ echo "${a-meu}"
meu
```

```
$ set bu zo
$ echo "${2-meu}"
zo
$ set bu
$ echo "${2-meu}"
meu
```

Các toán tử trên biến [2]

- `${var?mess}` → `$var` nếu được định nghĩa, nếu không, hiển thị `mess` trên `stderr + exit 1`
- `${var+val}` → `val` nếu `var` được định nghĩa. Ngược lại, không hiển thị gì cả
- `${var=val}` → `$var` nếu được định nghĩa. Ngược lại, `val` + *phép gán*
- Ví dụ :

```
$ a="ga"
$ echo "${a?Undefined}"
ga
$ echo "${a+toto}"
toto
$ echo "${a=meu}"
ga
```

```
$ unset a
$ echo "${a? Undefined}"
bash: a: Undefined
$ echo "${a+toto}"

$ echo "${a=meu}"
meu
```

Các chuỗi con

- `${var:i}` → chuỗi con từ vị trí `i≥0`
- `${var:i:k}` → chuỗi con từ vị trí `i≥0` với độ dài `k`
- `${#var}` → độ dài của `$var`
- Ví dụ :

```
$ a="bonjour"
$ echo ${#a}
7
$ echo ${a:3}
jour
$ echo ${a:3:1}
j
```

Xoá đầu chuỗi khớp với pattern

- ${var#pattern} → xoá phần ngắn nhất đầu chuỗi tương ứng với pattern
- ${var##pattern} → xoá bỏ phàn dài nhất đầu chuỗi tương ứng với pattern
- Ví dụ :

```
$ path="/Users/tuyennguyen1/linux/td2.sh"
$ echo "${path#/Users/tuyen}"
nguyen1/linux/td2.sh
$ echo "${path#tuyen}"
/Users/tuyennguyen1/linux/td2.sh
$ echo "${path#*tuyen}"
nguyen1/linux/td2.sh
$ echo "${path##*/}"
td2.sh
```

Xoá bỏ phần cuối chuỗi khớp với pattern

- `${var%pattern}` → xoá bỏ phần ngắn nhất ở cuối chuỗi khớp với pattern
- `${var%%pattern}` → xoá bỏ phần dài nhất ở cuối chuỗi khớp với pattern

Thay thế sử dụng pattern

- `${var/pattern/txt}` → thay thế phần đầu tiên tương ứng với pattern bởi chuỗi `txt`
- `${var//pattern/txt}` → thay thế tất cả các chuỗi con khớp với pattern bởi chuỗi `txt`
- Ví dụ :

```
$ path="/Users/tuyennguyen1/linux/td2.sh"
$ echo "${path/td2/td3}"
/Users/tuyennguyen1/linux/td3.sh
$ echo "${path/linux/unix}"
/Users/tuyennguyen1/unix/td2.sh
$echo "${path/u/U}"
/Users/tUyennguyen1/linux/td2.sh
$ echo "${path//u/U}"
/Users/tUyennguUyen1/linUx/td2.sh
$ echo "${path//u*n/U}"
/Users/tUux/td2.sh
```

Bổ sung : đọc nhiều biến

`read` v_1 v_2 ... v_n

- đọc một dòng từ chuẩn vào,
- cắt thành các từ (cách nhau bởi khoảng trắng, theo $IFS),
- gán giá trị cho các biến :

  $word_1$ → v_1

  $word_2$ → v_2

  .

  .

  .

  $word_{n-1}$ → v_{n-1}

  phần còn lại của dòng → v_n

- Vì vậy, `read` v_1 đọc cả dòng vào v_1 mà không cắt chuỗi

Nội dung

1. Các toán tử trên biến
2. **Hàm**
3. Thay thế lệnh
4. Debug

Hàm

- Khai báo :

```
function_name ()
{
   instructions
}
```

- hay khai báo trên một dòng :

```
function_name() { instructions ;}
```

- Có thể thực hiện ngay trên cửa sổ dòng lệnh hoặc đặt ở bất kỳ đâu trong file script, trước khi hàm được triệu gọi.
- () luôn luôn đi liền nhau: để chỉ đây là khai báo hàm.

Triệu gọi hàm và tham số

- Gọi hàm : `function_name parameters`

→ Một hàm sử dụng như một script :

Script titi :

```
#! /bin/bash
echo "Receive $1"
exit 0
```

Gọi :

```
$ ./titi foo
 Receive foo
```

Với một hàm :

```
$ toto () {
  echo " Receive $1"
  return 0
}
$ toto foo
Receive foo
```

Tham số

```
$ set foo
$ echo "$1"
foo
$ hop () { echo "$1" ;}
$ hop bar
bar
$ echo "$1"
foo
```

Tham số $0

- Tham số $0 là bất biến :

```
$ echo "$0"
bash
$ shift
$ echo "$0"
bash
$ pouet () { echo "$0" ;}
$ pouet
bash
```

- Giống như vậy trong script.

Đầu ra của hàm

- `return [x]` : ra khỏi hàm ngay lập tức
- `exit [x]` : ra khỏi script (hoặc shell) ngay lập tức
- x là mã kết thúc → $?
- Mặc định: mã kết thúc của lệnh cuối cùng được thực thi

```
foo () {...; return 0 ;}     # thanh cong
foo () {...; echo "ga";}     # thanh cong
foo () {...; false    ;}     # that bai
foo () {...; test...  ;}     # phu thuoc vao test
```

Không trả về giá trị

- Giống như các script, hàm thành công hoặc thất bại nhưng không trả về giá trị.
- Sử dụng hàm:

```
myfunc () { ..... ; return x ;}
if myfunc ga bu ; then ... ; fi
while myfunc zo meu ; do ... ; done
```

Điều hướng

- Ta có thể điều hướng vào/ra của hàm :

```
hop () { ls ;}
hop > toto
hop | sort
```

- Khối lệnh giữa các dấu { và }: cho phép điều hướng các dòng lệnh

```
{ echo "ga" ; echo "bu" ;} > toto
```

Phạm vi của biến

- Mặc định: Các biến là toàn cục:

```
$ hop () { a="foo" ;}
$ a="bar" ; hop ; echo "$a"
foo
```

- Ta có thể khai báo biến cục bộ trong hàm :

```
local var1 var2 ...
var x=value ...
```

- Ví dụ:

```
$ hop () { local a="foo" ;}
$ a="bar" ; hop ; echo "$a"
bar
```

Gán giá trị với `local`

```
$ a=hips
$ g() {
  local a="$1" b="$a"
  echo "$b"
}
$ g bu
hips
```

⚠ Trong `local`, các thay thế được thực hiện trước tiên

- Giải pháp : thực hiện 2 pha

```
$ g() {
    local a="$1"
    local b="$a"
    echo "$b"
}
```

```
# hoặc :
# local a="$1" b
# b="$a"
```

Chỉ dẫn khi dùng các tham số

- Chỉ dẫn :
  - ghi rõ các tham số trong chú thích
  - sử dụng các tham số trong các biến cục bộ.

```
afficher_age () # nom prenom age
{
    local nom="$1" prenom="$2" age="$3"
    echo "$prenom $nom a $age ans."
}
```

Ví dụ

```
possede_extension () # fichier extension
{
   local fichier="$1" ext="$2"
   local fichier_sans_ext="${fichier%$ext}"
   test "$fichier_sans_ext$ext" = "$fichier"
}
```

- Ví dụ sử dụng hàm :

```
fic="Mes images/IMG234.jpg"
if possede_extension "$fic" ".jpg" ; then
    echo "Le fichier $fic possède l'extension .jpg"
fi
```

Nội dung

1. Các toán tử trên biến
2. Hàm
3. **Thay thế lệnh**
4. Debug

Thay thế lệnh

- Cho phép thay thế việc triệu gọi một lệnh bởi những gì mà nó hiện thị. Có hai dạng thức:

  `$(command)` bash, nên sử dụng

  `'command'` sh, cũ

- bash thực hiện lệnh trong tiến trình con, sau đó thay thế bởi đầu ra chuẩn của nó.
- Ví dụ : `x=$(expr 5 + 7)`
- có cùng hiệu ứng với :

```
expr 5 + 7 >| tmp
read x < tmp
rm -f tmp
```

Thay thế trong chuỗi

```
f="ga.c"

echo "File $f has $(wc -l < $f) lines"
          ↓
echo "File ga.c has $(wc -l < ga.c) lines"
          ↓
echo "File ga.c has 25 lines"
```

Xoá dòng

- Trong khi thực hiện thay thế lệnh, bash xoá những dòng trống cuối cùng :

```
$ echo -e "\n\n bonjour \n\n"


 bonjour


$ a=$(echo -e "\n\n bonjour \n\n")
$ echo "'$a'"
'

 bonjour '
$
```

Thay thế bằng một file

- Để thay thế bằng nội dung một file:

  `$(cat < fichier)`

  `$(< fichier)` (ngắn gọn hơn)

- Ví dụ:

```
$ echo "Voici le contenu du fichier text.txt" > text.txt
$ cat text.txt
Voici le contenu du fichier text.txt
$ text=$(cat text.txt)
$ echo $text
Voici le contenu du fichier text.txt
```

Thay thế lồng nhau và thay thế khối lệnh

- Ta có thể thay thế các lệnh lồng nhau :

```
msg="taille = $(wc -c < $(which test)) octets"
        ↓
msg="taille = $(wc -c < /usr/bin/test) octets"
        ↓
msg="taille = 30272 octets"
```

- Ta có thể thay thế một khối lệnh:

```
meu=$(echo "ga" ; echo "bu" ; echo "zo")
case $(ls -t | head -1) in ... esac
```

Kết quả một hàm

- `return` được dành cho thành công/thất bại. Các hàm có thể đưa ra kết quả ở chuẩn ra:

```
my_function () # parameters
{
    ...
    echo "le résultat"
    return 0 # ta co the bo qua
}
```

- Ta lấy kết quả bằng cách sử dụng thay thế lệnh:

```
res=$(my_function parameters)
```

- Hàm chỉ hiển thị `"le résultat"`

Ví dụ 1: cộng, nhân, tổng bình phương

```
plus () # x y
{
    local x="$1" y="$2"
    expr "$x" + "$y"
}

mult () # x y
{
    local x="$1" y="$2"
    expr "$x" '*' "$y"  # Chú ý dấu ''
}
norme_carre () # x y
{
    local x="$1" y="$2"
    plus $(mult "$x" "$x") $(mult "$y" "$y")
}

x=3 ; y=4 ; z=$(norme_carre "$x" "$y") # 25
echo $z
```

Ví dụ 2 : số lớn nhất

```
max () # liste d'entiers
{
    if test $# -lt 1 ; then echo "" ; return ; fi
    local i max="$1"
    shift
    for i in $* ; do
   if test "$max" -lt "$i" ; then max="$i" ; fi
    done
    echo "$max"
}

m=$(max 5 7 20 6 8)  #20
echo $m
```

Ví dụ 3 : từ dài nhất

Độ dài một biến : ${#nom_variable}

```
plus_long () # liste de mots
{
    local i max=""
    for i in "$@" ; do
   if test ${#max} -lt ${#i} ; then max="$i" ; fi
    done
    echo "$max"
}

plus_long "el" "ello" "hello" "hello everybody"
```

Nội dung

1. Các toán tử trên biến
2. Hàm
3. Thay thế lệnh
4. **Debug**

Debug

- Hiển thị danh sách các hàm :

```
declare -F
```

- Hiển thị thân hàm:

```
declare -f function_name
```

thành công nếu hàm tồn tại.

- Xoá hàm :

```
unset -f function_name
```

- ta có thể định nghĩa lại một hàm, `unset` không bắt buộc.

Thông tin trên biến

- Hiển thị danh sách các biến :

```
declare -p
```

- Hiển thị nhiều thông tin hơn cho một biến:

```
declare -p variable_name
```

thành công nếu biến tồn tại.

- Xoá một biến :
- `unset -v variable_name`

Lần dấu tự động [1]

- Để yêu cầu bash hiển thị tất các các lệnh mà nó đã thực thi : `set -x`

Huỷ bỏ : `set +x`

```
$ ga="home"; bu="nguyen"
$ set -x
$ if test "$ga" = "$bu" ; then echo "ok" ; fi
+ test home=nguyen
+ echo ok
ok
$ set +x
```

Lần dấu tự động [2]

- Kích hoạt hoặc huỷ kích hoạt chế độ debug trong cửa sổ lệnh hoặc trong script :

```
set -x ... set +x
```

- Cấu hình script để nó hiện thị chế độ lần dấu :

```
#! /bin/bash -x
```

- Thực thi script ở chế độ lần dấu mà không thay đổi script :
- `$ bash -x ./myscript`

Câu hỏi ?