



Nhập môn hệ điều hành Unix

Nguyễn Hải Châu
Khoa Công nghệ Thông tin
Trường Đại học Công nghệ
Đại học Quốc gia Hà Nội
(Bài giảng tuần 5)



Nội dung

- Lập trình shell trên Unix-Linux
 - Các toán tử trong shell
 - Các cấu trúc điều khiển trong shell



Các toán tử string

Toán tử	Chức năng
<code>\${var:- word}</code>	Nếu biến tồn tại và xác định thì trả về giá trị của nó, nếu không thì trả về word
<code>\${var:= word}</code>	Nếu biến tồn tại và xác định thì trả về giá trị của nó, nếu không thì gán biến thành word, sau đó trả về giá trị của nó



Các toán tử string

<code>\${var:+ word}</code>	Nếu biến tồn tại và xác định thì trả về word, còn không thì trả về null
<code>\${var:?message}</code>	Nếu biến tồn tại và xác định thì trả về giá trị của nó, còn không thì hiển thị "bash: \$var:\$message" và thoát ra khỏi lệnh/tập lệnh hiện thời.
<code>\${var:offset[:length]}</code>	Trả về một chuỗi con của var bắt đầu tại offset của độ dài length. Nếu length bị bỏ qua, toàn bộ chuỗi từ offset sẽ được trả về.



Ví dụ minh họa toán tử string

Xét một biến shell tên là status được khởi tạo giá trị "defined". Sử dụng các toán tử string cho kết quả status như sau:

```
echo ${status:-undefined}
defined
echo ${status:=undefined}
defined
echo ${status:+undefined}
undefined
echo ${status:?Dohhh! undefined}
defined
```



Ví dụ (tiếp)

- Bây giờ sử dụng lệnh unset để xóa biến status, và thực hiện vẫn các lệnh đó, được output như sau:

```
unset status
echo ${status:-undefined}
undefined
echo ${status:=undefined}
undefined
echo ${status:+undefined}
undefined
unset status
echo ${status:?Dohhh! undefined}
bash:status Dohhh! Undefined
```



Các toán tử pattern-matching

Toán tử	Chức năng
<code>\${var#pattern}</code>	Xoá bỏ phần khớp (match) ngắn nhất của pattern trước var và trả về phần còn lại
<code>\${var##pattern}</code>	Xoá bỏ phần khớp (match) dài nhất của pattern trước var và trả về phần còn lại
<code>\${var%pattern}</code>	Xoá bỏ phần khớp ngắn nhất của pattern ở cuối var và trả về phần còn lại



Các toán tử pattern-matching

<code>\${var%%pattern}</code>	Xoá bỏ phần khớp dài nhất của pattern ở cuối var và trả về phần còn lại
<code>\${var/pattern/string}</code>	Thay phần khớp dài nhất của pattern trong var bằng string. Chỉ thay phần khớp đầu tiên. Toán tử này chỉ có trong bash 2.0 hay lớn hơn.
<code>\${var//pattern/string}</code>	Thay phần khớp dài nhất của pattern trong var bằng string. Thay tất cả các phần khớp. Toán tử này có trong bash 2.0 hoặc lớn hơn.



Ví dụ: Tách tên thư mục/tệp

```
#!/bin/bash
#####
myfile=/usr/src/linux/Documentation/ide.txt
echo `${myfile##*/}`=${myfile##*/}
echo `basename $myfile`=$(basename $myfile)
echo `${myfile%/*}`=${myfile%/*}
echo `dirname $myfile`=$(dirname $myfile)
```



Ví dụ: Tách tên thư mục/tệp

■ **Kết quả:**

```
`${myfile##*/}` = ide.txt
basename $myfile = ide.txt
`${myfile%/*}` = /usr/src/linux/Documentation
dirname $myfile = /usr/src/linux/Documentation
```



Các toán tử so sánh chuỗi

Toán tử	Ý nghĩa
<code>str1 = str2</code>	str1 bằng str2
<code>str1 != str2</code>	str1 khác str2
<code>-n str</code>	str có độ dài lớn hơn 0 (khác null)
<code>-z str</code>	str có độ dài bằng 0 (null)



Các toán tử so sánh số học

Toán tử	Ý nghĩa
<code>-eq</code>	bằng
<code>-ge</code>	lớn hơn hoặc bằng
<code>-gt</code>	lớn hơn
<code>-le</code>	nhỏ hơn hoặc bằng
<code>-lt</code>	nhỏ hơn
<code>-ne</code>	khác



Điều khiển luồng trong shell

- Các cấu trúc điều khiển luồng của bash bao gồm:
 - **if** – Thi hành một hoặc nhiều câu lệnh nếu có điều kiện là true hoặc false.
 - **for** – Thi hành một hoặc nhiều câu lệnh trong một số cố định lần.
 - **while** – Thi hành một hoặc nhiều câu lệnh trong khi một điều kiện nào đó là true hoặc false.
 - **until** – Thi hành một hoặc nhiều câu lệnh cho đến khi một điều kiện nào đó trở thành true hoặc false.
 - **case** – Thi hành một hoặc nhiều câu lệnh phụ thuộc vào giá trị của biến.
 - **select** – Thi hành một hoặc nhiều câu lệnh dựa trên một khoảng tùy chọn của người dùng.



Cấu trúc rẽ nhánh if

```

if điều kiện
then
    Các câu lệnh
[elif điều kiện
    Các câu lệnh]
[else
    Các câu lệnh]
fi
  
```



Toán tử AND và OR logic

- **command1 && command2**
- Câu lệnh **command2** chỉ được chạy khi và chỉ khi **command1** trả về trạng thái là số 0 (true).
- **command1 || command2**
- Câu lệnh **command2** chỉ được chạy khi và chỉ khi **command1** trả lại một giá trị khác 0 (false).



Ví dụ

```

if cd /home/kwall/data
then
    if cp datafile datafile.bak
    then
        # more code here
    fi
fi
rm myf && echo "File is removed
successfully" || echo "File is not removed"
  
```



Ví dụ (tiếp)

```

if [ $1 -gt 0 ]
then
    echo "$1 is positive"
elif [ $1 -lt 0 ]
then
    echo "$1 is negative"
elif [ $1 -eq 0 ]
then
    echo "$1 is zero"
else
    echo "Opps! $1 is not number, give number"
fi
  
```



Kiểm tra điều kiện với test

Toán tử	Điều kiện true
-d file	file tồn tại và là một thư mục
-e file	file tồn tại
-f file	file tồn tại và là một file bình thường (không là thư mục hay file đặc biệt)
-r file	file cho phép đọc
-s file	file tồn tại và khác rỗng
-w file	file cho phép ghi
-x file	file khả thi hoặc một thư mục cho phép tìm kiếm tên file
-O file	file của người dùng hiện tại
-G file	file thuộc một trong các nhóm người dùng hiện tại là thành viên



Ví dụ

```
IFS=:
for dir in $PATH;
do
  echo $dir
  if [ -w $dir ]
  then
    echo -e "\tYou have write permission in $dir"
  else
    echo -e "\tYou don't have write permission in $dir"
  fi
```



Ví dụ (tiếp)

```
if [ -O $dir ]; then
  echo -e "\tYou own $dir"
else
  echo -e "\tYou don't own $dir"
fi
if [ -G $dir ]; then
  echo -e "\tYou are a member of $dir's group"
else
  echo -e "\tYou aren't a member of $dir's group"
fi
done
```



Vòng lặp for

for var in <danh sách>

do

Các câu lệnh có sử dụng \$var

done

hoặc

for ((expr1; expr2; expr3))

do

.....

Lặp lại các toán tử nằm giữa do và done cho đến khi expr2 nhận giá trị TRUE

done



Ví dụ

for docfile in doc/*.doc

do

cp \$docfile \${docfile%.doc}.txt

done

for ((i = 1; i <= 5; i++)) **do**

for ((j = 1; j <= 5; j++)) **do**
echo -n "\$i "

done

done



Ví dụ

```
#!/bin/bash
```

```
for i in 1 2 3 4 5
```

```
do
```

```
  echo "Welcome $i times"
```

```
done
```

```
#!/bin/bash
```

```
for (( i = 0; i <= 5; i++ ))
```

```
do
```

```
  echo "Welcome $i times"
```

```
done
```