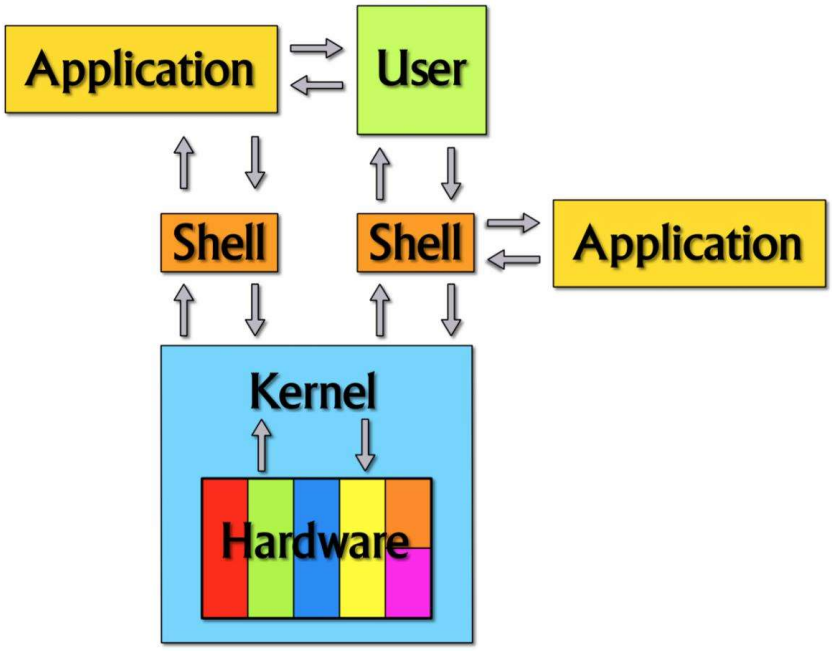
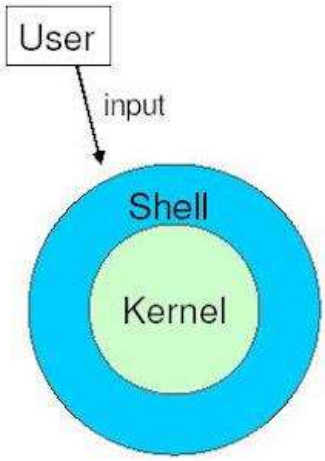
LẬP TRÌNH SHELL

Giới thiệu

- Khái niệm về shell
- Biến trong Shell
- Tùy biến môi trường
- Điều khiển công việc
- Viết chương trình Shell

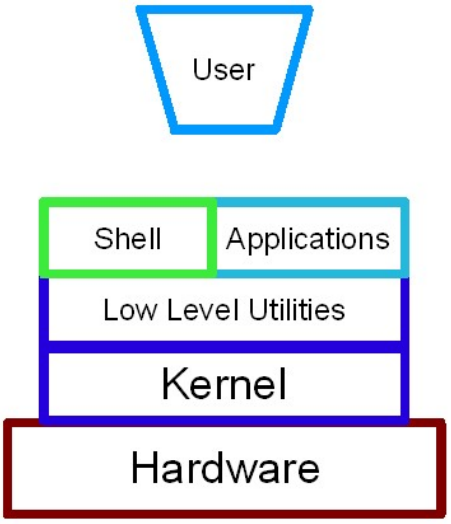
Lập trình shell





Khái niệm về shell

- Shell là chương trình thông dịch lệnh của hệ điều hành Linux
  - Tương tác với người dùng theo từng câu lệnh;
  - Shell đọc lệnh từ bàn phím hoặc tệp tin;
  - Nhờ nhân của hệ điều hành Linux để thực thi lệnh.
- Shell script: là chương trình shell, bao gồm một tập các lệnh



Soạn thảo shell script

- Soạn thảo chương trình shell
  - Sử dụng trình soạn thảo văn bản để tạo chương trình shell script (.sh)
  - Nội dung chương trình bao gồm tập các câu lệnh theo cú pháp lệnh trên Linux
  - Các câu lệnh *trên cùng một dòng* được ngăn cách bởi dấu chấm phẩy (;)
  - Thiết lập quyền thực thi cho tệp shell script: chmod u+x <tên tệp>
- Soạn thảo shell script
  - $ nano hello.sh : tạo một file script tên hello.sh và soạn thảo bằng trình soạn thảo nano

Gọi thực hiện shell script

- bash <Tên tệp> [arguments] hoặc
- sh <Tên tệp> [arguments] hoặc
  - ./<Tên tệp>
- Các tệp tin liên quan
  - /etc/profile – system initialization file
  - ~/.bash_profile(.bashrc) – Khởi tạo biến cá nhân
  - ~/.bash_history – Lưu lại các lệnh đã gõ

Chương trình shell đơn giản

- Ví dụ chương trình shell đơn giản displayInfo.sh

```
#!/bin/bash
clear
echo "Hello: $USER"
echo "Today is: "; date
echo "Number of user login:"; who | wc –l
echo "Calendar: "; cal
```

- Thiết lập quyền truy cập
  - chmod 755 displayInfo.sh
- Chạy chương trình shell
  - ./ displayInfo.sh

Biến trong shell

- Sử dụng trong lập trình shell và điều khiển môi trường
- Gán giá trị cho biến: variable_name=value
  - Không cần khai báo biến trước
- Truy cập vào giá trị của biến: $variable_name
- Ví dụ:

```
$ foo="hello there"
$ echo $foo
hello there
```

Các biến trong shell

- Trong Linux có 2 loại biến

Biến hệ thống	Biến do người dùng định nghĩa
Tạo ra và quản lí bởi hệ điều hành Linux;	Tạo ra và quản lí bởi người dùng;
Tên biến viết hoa	Tên biến viết thường

- Xem và truy xuất giá trị của biến
  - $<Tên biến>
  - echo $HOME
  - echo $USERNAME
  - Phải có kí hiệu $ trước tên biến

Biến người dùng

- Biến được nhận qua lệnh read hoặc
- Biến được định nghĩa
  - Cú pháp: <Tên biến>=<giá trị>
- Chú ý:
  - Không cần khai báo biến trước trong shell;
  - Tên biến phân biệt chữ hoa và chữ thường;
  - Tên biến phải bắt đầu bằng kí tự;
  - Không sử dụng kí tự đặc biệt trong tên biến
  - Biến không gán giá trị có giá trị là NULL;
  - Không sử dụng kí tự đặc biệt trong tên biến.
  - Không sử dụng dấu cách trong định nghĩa biến

Biến trong shell

- Ví dụ:

```
max=100
min=10
nhiet_do=35
```

- Hiển thị giá trị của biến:
  - Cú pháp: echo $<Tên biến>
  - ví dụ:

```
echo $max
echo $nhiet_do
```

- Chú ý: *không được để khoảng trắng ở hai bên toán tử gán (=)*

Các biến môi trường

- HOME – Thư mục nhà
  - Ví dụ: $ echo $HOME --> Kết quả: /home/tom (nếu người dùng có tên là tom)
- PATH – Đường dẫn tới các tập tin thực thi
- USER – Tên người dùng đăng nhập
  - Ví dụ: $ echo $USER --> tom
- printenv : lệnh in ra các biến môi trường

Các biến môi trường

- Một số biến môi trường:
  - BASH_VERSION: Tên version của shell;
  - HOME: Tên thư mục home của người sử dụng;
  - LOGNAME: Tên đăng nhập của người sử dụng;
  - OSTYPE: Kiểu hệ điều hành;
  - PATH: Thiết lập đường dẫn cho hệ thống;
  - PWD: Thư mục hiện hành của người sử dụng.

Lập trình shell

- Một số lệnh trong LT Shell
- Shell script
- Các biến
- Biểu thức so sánh
- Câu lệnh điều khiển
- Hàm(Functions)

MỘT SỐ LỆNH LẬP TRÌNH SHELL

- Lệnh echo
  - Cú pháp: echo [tùy chọn] [string, variable]
  - Lệnh echo hiển thị thông tin ra thiết bị chuẩn, thông tin hiển thị có thể là một hằng xâu kí tự, giá trị một biến hoặc một biểu thức.
  - Ví dụ: echo “Hello”, $myname
- Một số kí tự đặc biệt
  - \b: Kí tự backspace (xóa lùi);
  - \n: Kí tự dòng mới;
  - \r: Kí tự về đầu dòng;
  - \t: Kí tự tab.

Biểu thức số học

- Các toán tử: +, -, *, /, %
- Biểu thức
  - $ let a=2+1 hoặc let "a = 2 +1"
  - $ echo $a
  - Kết quả khi chạy script: 3
  - Chú ý không chừa khoảng trắng trong biểu thức nếu không sử dụng dấu ngoặc kép
  - Có thể chừa khoảng trắng hoặc không khi sử dụng dấu ngoặc kép.
- Ví dụ
  - let "area = $len * $width"
  - let "percent = $num / 100"
  - let "remain = $n % $d"

Biểu thức so sánh

- Biểu thức so sánh chuỗi

Biểu thức	Kiểm tra (đúng -> true)
[$str1 = $str2]	str1 bằng str2
[$str1 != $str2]	str1 khác str2
[-n $str]	str có độ dài lớn hơn 0
[-z $str]	độ dài bằng 0 (null)

Biểu thức so sánh

- Biểu thức so sánh số học

Biểu thức	Kiểm tra
[$a –eq $b]	a bằng b
[$a –ge $b]	a lớn hơn hoặc bằng b
[$a –gt $b]	a lớn hơn b
[$a –lt $b]	a bé hơn b
[$a –le $b]	a bé hơn hay bằng b
[$a –ne $b]	a khác b

Biểu thức so sánh trên tệp

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 một thư mục hay một 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 nếu file là một thư mục thì cho phép tìm kiếm trên file
-O file	file của người dùng hiện tại
-G file	file thuộc một trong các nhóm người dùng hiện tại là thành viên
file1 -nt file2	file1 mới hơn file2
file1 -ot file2	file1 cũ hơn file2

Chuỗi

- Các chuỗi được đặt trong cặp dấu nháy kép "" hoặc cặp dấu nháy đơn ''
  - Ví dụ: echo “Hello $myname” : Lệnh này sẽ hiển thị lời chào với tên được lưu trong biến myname.
  - Dấu nháy đơn ‘’: Các biến đặt trong cặp nháy đơn sẽ bị vô hiệu hóa
  - Ví dụ: echo 'Hello $myname' : Lệnh này sẽ hiển thị dòng thông báo: Hello $myname.
- Trạng thái kết thúc câu lệnh: Linux mặc định trả về:
  - Trạng thái 0 nếu lệnh kết thúc thành công;
  - Khác 0 nếu lệnh có lỗi.
  - Kiểm tra trạng thái kết thúc lệnh: $?
    - Ví dụ: $cd document
    - $ echo $? --> kết quả: 0 nếu cd vào thư mục document thành công, ngược lại cho giá trị 1

CÁC LỆNH CƠ BẢN

- Lệnh đọc dữ liệu từ bàn phím
  - Cú pháp: read <Tên biến>
  - Lệnh đọc dữ liệu từ bàn phím và lưu vào biến được chỉ định bởi tên biến.
- Ví dụ:

```
echo "Enter Your name:"
read name
echo "Hello: $name"
```


Tham số dòng lệnh

- Một chương trình shell có nhiều tham số dòng lệnh, người sử dụng có thể truy cập và lấy giá trị từ các tham số dòng lệnh của chương trình Shell
  - Tên lệnh: $0
  - Các tham số của chương trình: $1, $2,…$9
  - Số các tham số: $#
  - Ví dụ: "echo your program: $0"
- Ví dụ:
  - $ bash cong.sh 4 9

Các biến đặc biệt

- $$ - ID của tiến trình hiện tại
- $? - Trạng thái kết thúc của tiến trình cuối cùng
  - giá trị 0: Tiến trình kết thúc thành công
  - giá trị 1: Tiến trình kết thúc không thành công

Chương trình shell

- Chương trình là dãy các dòng lệnh shell được đặt trong một file văn bản (được soạn thảo theo soạn thảo văn bản),
- Các dòng lệnh bắt đầu bằng dấu # chính là dòng chú thích, bị bỏ qua khi thực hiện chương trình shell.
- Thông thường các bộ dịch lệnh shell là sh (/bin/sh) hoặc bash (/bin/bash)
- Để thực hiện một chương trình shell ta có các cách sau đây:
  - $ bash <<tên chương trình> [các tham số] hoặc
  - Đổi mod của chương trình: $ chmod u+x <tên chương trình>

  và chạy chương trình $ <tên chương trình> [các tham số]

Hello program

```
$ cat > hello.sh
#!/bin/bash
# This is a comment: simple hello shell script
echo "Enter your name:"
read name
echo "Hello $name, have a nice day!"
^D
$ ./hello.sh
bash: ./hello.sh: Permission denied
$ chmod +x hello.sh
$ ./hello.sh
Enter your name:
Tuan
Hello Tuan, have a nice day
```


Câu lệnh điều kiện

- IF statement

```
if [ expression1 ]; then
   statement1
elif [ expression2 ]; then
   statement2
else
   statement3
fi
```

- Example

```
if [ $1 ="" ]; then
   echo "Enter value:"
   read num
else
   let "num = $1"
fi
```

Ví dụ if .. else

```
#!/bin/bash
# Script to test if..elif...else
#
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
```


Câu lệnh điều kiện (cont)

- CASE statement

```
case $var in
val1)
   statement1;;
val2)
   statement2;;
*)
   statement3;;
esac
```

- Example

```
case $1 in
1)
   echo "One";;
2)
   echo "Two";;
*)
   echo "Unknown";;
esac
```

- Hai dấu chấm phẩy (;;) tương đương với lệnh break

Câu lệnh select

- Liệt kê menu lựa chọn, người sử dụng chọn các giá trị liệt kê thông qua các con số
- Cú pháp:

```
select value [in list]
do
    statements that manipulate $value
done
```


Ví dụ:

```
#!/bin/bash
clear
select i in one two three
do
 if [ $i ]; then
        echo "ban chon $i"
 else
        echo "ban chon khac"
 fi
        echo "Chon tiep hay bam ^C de ket thuc"
done
```

```
1) one
2) two
3) three
#? 1
ban chon one
Chon tiep hay bam ^C de ket thuc
```

Câu lệnh lặp - for

FOR statement:

```
for var [in list]; do
   statements
done
```

Example:

```
let "sum = 0"
for num in 1 2 3 4 5; do
   let "sum = $sum + $num"
done
echo $sum
```


Câu lệnh lặp - for

```
for (( expr1; expr2; expr3 ))
do
        .....
                        ...
        repeat all statements between do and
        done until expr2 is TRUE
done
```

```
#!/bin/bash
let s=0
for ((i=0;$i<=10;i++))
do
   let s=$s+$i
done
echo "Tong các so tu 1 den 10 la $s"
```

Ví dụ

- Hủy/Cấp quyền chạy cho tất cả các tập tin có phần mở rộng .sh
- Tạo một thư mục và di chuyển tất cả các tập tin có phần mở rộng .txt vào thư mục này.

Câu lệnh lặp (cont)

WHILE statement:
while [expression]; do
statements
done

Example:

```
let "num = 0"
while [ $num -lt 10 ]; do
  echo $num
  let "num= $num + 2"
done
```

Hàm

- Tập hợp các đoạn mã code thực hiện một chức năng nào đó
- Các đoạn mã code viết trong hàm có thể tái sử dụng được nhiều lần trong chương trình thông qua các lời gọi hàm
- Khai báo và định nghĩa hàm

```
tên_hàm{
   Các câu lệnh
   [ return giá_trị ]
}
```


Hàm

- Lời gọi hàm:
  - Tên_hàm [tham_số_1 tham_số_2 ....]
  - Các tham số của hàm là các tham số vị trí được đại diện bởi $1, $2, S3 trong thân hàm khi định nghĩa hàm.
  - Các tham số của hàm khác với các tham số dòng lệnh của script mặc dù cũng dùng các kí hiệu $1, $2, $3.....
- Giá trị trả về của hàm
  - Hàm trong shell có thể trả về giá trị bằng câu lệnh return
  - Giá trị trả về của hàm có thể nhận được thông qua biến ? ($?) khi gọi hàm

Giá trị trả về của hàm

- Ví dụ: viết shell script: chao.sh in ra màn hình câu: Chao ban

```
1#!/bin/bash
2hello(){
3    echo "Chao ban!"
4}
5
6hello
```

  - Chạy script: bash chao.sh
  - Kết quả trên màn hình: Chao ban

Truyền tham số cho hàm

- Viết shell script tinh.sh

```
1#!/bin/bash
2tong(){
3    let ket_qua=$1+$2
4    echo "$1 + $2 = $ket_qua"
5}
6
7tong 4 5
```

- Chạy script: bash tinh.sh
  - Kết quả: 4 + 5 = 9

Ví dụ: Hàm có kết quả trả về

- Viết shell script tinh.sh

```
1#!/bin/bash
2tong(){
3    let ket_qua=$1+$2
4    return $ket_qua
5}
6
7tong 4 5
8echo "4 + 5 = $?"
```

- Chạy script: bash tinh.sh
  - Kết quả 4 + 5 = 9

Tham số dòng lệnh, tham số của hàm

- Viết shell script tinh.sh

```
1#!/bin/bash
2tong(){
3# $1, $2 là các tham số của hàm
4# không phải là các tham số dòng lệnh của script
5    let ket_qua=$1+$2
6    echo "$1 + $2 = $ket_qua"
7}
8
9# $1, $2 là các tham số dòng lệnh của script
10tong $1 $2
```

- Chạy script: bash tinh.sh 4 5
  - Kết quả: 4 + 5 = 9

- Viết shell script giải pt bậc 1
  - Viết không cần hàm
  - Viết dưới dạng hàm
- Hủy/Cấp quyền chạy cho tất cả các tập tin có phần mở rộng .sh
  - Sử dụng câu lệnh select để đưa ra các lựa chọn hủy hay cấp quyền
  - Viết hàm hủy/cấp quyền với một tham số (nhận một trong hai giá trị hủy hay cấp)
- Viết hàm max trả về số lớn nhất trong hai số
- Viết hàm min trả về số bé nhất trong hai số

Ví dụ

```
1 #!/bin/bash
2 #Ham huy cap quyen tap tin
3 HuyCapQuyen(){
4     if [ $1 == "huy" ]; then
5         p="-x"
6     else
7         p="+x"
8     fi
9     for file in ./*.sh
10    do
11        chmod u$p $file
12    done
13    ls -l *.sh
14 }
```

Ví dụ

```
15
16 select i in cap huy
17 do
18    if [ $i ]; then
19       HuyCapQuyen $i
20    else
21       echo "Chon nham nhe"
22    fi
23    echo "Chon lai hay nhan ctrl +C de thoat"
24 done
```