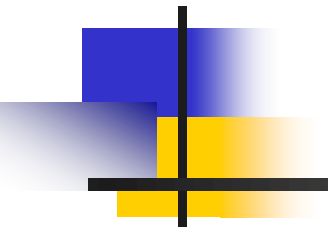
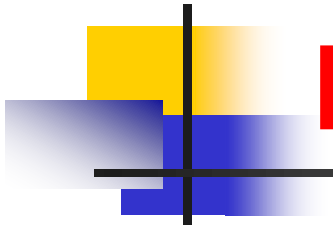
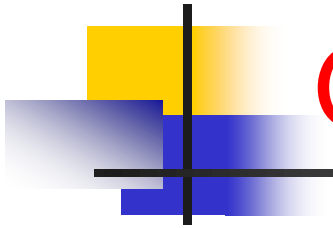


# Bài 6: Lập trình Shell



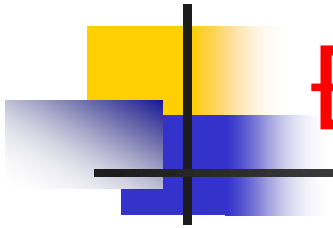
# Nội dung

1. Giới thiệu
2. Một số thành phần cơ bản của lập trình Shell
3. Cấu trúc điều khiển
4. Các hàm



# Giới thiệu lập trình Shell

- Shell là bộ biên dịch lệnh có vai trò kết nối giữa nhân và người dùng
- Shell đọc lệnh từ bàn phím hoặc file và nhờ nhân Linux thực hiện
- Shell script là các chương trình shell bao gồm chuỗi các lệnh:
  - ✓ Nhận đầu vào từ người dùng, file
  - ✓ Tự động hóa các hành động (các lệnh)
  - ✓ Sử dụng các cấu trúc mạnh: vòng lặp, rẽ nhánh, …
  - ✓ Script cài đặt phần mềm cho phép cấu hình



# Đặc trưng của scrip

- Chương trình viết bằng ngôn ngữ shell
  - Các shell khác nhau không tương thích hoàn toàn: nó phụ thuộc vào shell: csh, sh, …
  - Chạy được (excutable)

# Cấu trúc của shell script

- Chỉ thị tên shell
  - Dòng dầu tiên bắt đầu bằng #!/<path_name_to_shell>

    Ví dụ: #!/bin/bash hoặc #!/bin/sh
- Chú giải (comment)

  - Dấu #: bắt đầu chú giải đến hết dòng, biên dịch sẽ bỏ qua dòng chú giải
  - Có thể đặt comment ở bất cứ đâu
  - Chỉ thị shell là một comment đặc biệt
- Dòng lệnh (code): các hàm hoặc các lệnh shell

# Soạn và thực thi chương trình shell

- Sử dụng mọi trình soạn thảo dạng text
  - gedit, vi, emacs, …
  - Nội dung bao gồm các lệnh được sử dụng trên dòng lệnh của Linux
  - Các câu lệnh trên cùng 1 dòng phải phân tách bởi dấu ;
- Thiết lập quyền thực thi cho chương trình shell

  **chmod u+x file_name**
- Thực thi: có ba cách

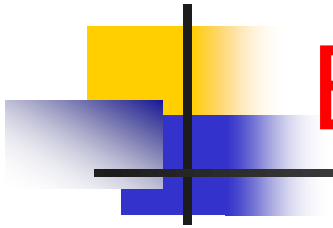
  **bash file_name.sh** hoặc **sh file_name.sh** hoặc **./file_name.sh**

# Ví dụ chương trình shell

```
#!/bin/bash
echo " Xin chao "
echo "Ban ten la gi: "
read name
echo "Chao ban $name"
```

# Một số thành phần trong lập trình shell

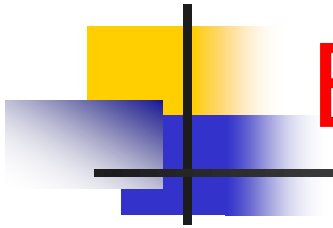
- Các biến
- Các phép toán số học
- Các tham số đối dòng lệnh
- Vào ra dữ liệu
- Biểu thức điều kiện
- Cấu trúc điều kiển: lặp, rẽ nhánh



# Biến

Trong linux shell có 3 loại biến

- **Biến môi trường (biến hệ thống, biến shell chuẩn)**
  - Tạo ra và quản lý bởi linux
  - Tên biến là chữ hoa: **HOME, PWD, USER, SHELL, BASH_VERSION**
  - Để xem tất cả các biến môi trường hiện có dùng lệnh **env**
- **Biến tự động (tham số vị trí)**
  - Các biến do Shell đã được định nghĩa trước
  - Có 10 biến tự động: $0, $1, …., $9
- **Biến do người dùng định nghĩa**
  - Tạo ra và được quản lý bởi người dùng
  - Tên biến là chữ thường



# Biến

Cách truy cập hoặc hiển thị giá trị các biến:

- **$variable_name**
- **echo $HOME** # hiển thị thư mục mặc định của người dùng
- **echo $HOSTNAME** # hiển thị tên máy tính

# Biến môi trường

| System Variable | Meaning |
|---|---|
| BASH=/bin/bash | Our shell name |
| BASH_VERSION=1.14.7(1) | Our shell version name |
| COLUMNS=80 | No. of columns for our screen |
| HOME=/home/vivek | Our home directory |
| LINES=25 | No. of columns for our screen |
| LOGNAME=students | students Our logging name |
| OSTYPE=Linux | Our Os type |
| PATH=/usr/bin:/sbin:/bin:/usr/sbin | Our path settings |
| PS1=[\u@\h \W]\$ | Our prompt settings |
| PWD=/home/students/Common | Our current working directory |
| SHELL=/bin/bash | Our shell name |
| USERNAME=vivek | User name who is currently login to this PC |

# Biến tự động

| Biến | Ý nghĩa |
|---|---|
| **$0** | Tên file của script hiện tại |
| **$n** | Những biến tương ứng với các tham số mà một script được gọi<br>($1 biểu diễn tham số đầu tiên, $2 biểu diễn tham số thứ 2, …) |
| **$#** | Số tham số tương ứng với một script |
| **$* hoặc $@** | Tất cả các đối số được trích dẫn, nếu một script gọi đến hai đối số thì $* tương ứng với $1 và $2 |
| **$?** | Trạng thái exit của câu lệnh cuối cùng được thực hiện. Nếu cậu lệnh thành công thì trạng thái nhận giá trị là 0, ngược lại nhận giá trị khác 0. |

# Biến người dùng

- Cú pháp:

  **variable_name=value**

- In giá trị các biến

  **echo $variable_name**

- Ví dụ:
  - **var=10**
  - **echo $var**

# Quy tắc đặt tên biến

- Ký tự đầu tiên phải là một chữ cái hoặc dấu gạch chân "_", không có ký tự cách
  - **sum, cost**
  - **distance_two_points**
- Không được để dấu cách hai bên toán tử = khi gán giá trị cho biến
  - Ví dụ:
    - **var=10** # đúng
    - **var =10** # sai do có dấu cách giữa biến và dấu "="
    - **var = 10** # sai do có dấu cách giữa biến và dấu "="
- Phân biệt chữ hoa, chữ thường: **var=10; Var=11; VAR=12;**
- Không dùng các ký tự "?" "*" để đặt tên các biến
- Một biến không có giá trị khởi tạo thì bằng NULL

# Các phép toán số học

- Để thực hiện các lệnh tính toán số học cần dùng câu lệnh:
  **expr biểu_thức_số_học  (expr $biến_1 toán tử $biến_2)**
  các toán tử: +, -, *, /, %
- Ví dụ: a=1; b=2;
  - `expr $a + $b`
  - `expr $a – $b`
  - `expr 10 / 2`
  - `expr 10 % 3`
  - `expr 10 \* 3`
  - var=`expr $a + $b`
  - echo $var #  in ra giá trị var

# Các dấu ngoặc

- **Ký tự trích dẫn yếu “ ” và ký tự trích dẫn mạnh ` `**
  - Tất cả các ký tự trong dấu ngoặc kép “ ” đều không có ý nghĩa tính toán, trừ những ký tự sau \ hoặc $
  - Dấu nháy ngược ` ` có ý nghĩa yêu cầu thực hiện lệnh

Ví dụ:

- ✓ **echo “Today is `date`”**

  Today is Mon April ...

- ✓ *var= `pwd `*

  *echo $var*

# Trạng thái kết thúc

- Linux mặc định trả về
  - Trạng thái 0 nếu câu lệnh kết thúc thành công
  - Khác 0 nếu kết thúc có lỗi
- Kiểm tra trạng thái kết thúc một câu lệnh
  - $?: cho biến trạng thái kết thúc câu lệnh trước đó
- Ví dụ:

  rm file1

  Nếu không có file này hệ thống sẽ thông báo : rm cannot remove “file1”: No such file or directory

  Thực hiện echo **$?** sẽ in ra giá trị khác 0

# Các tham số dòng lệnh

- Tham số dòng lệnh là các giá trị được đưa vào như là 1 tùy chọn (option) ở câu lệnh chạy

  Ví dụ: ./ test.sh a b c

  - a,b,c là các đối dòng lệnh trong đó: a là đối dòng lệnh thứ nhất, b là đối dòng lệnh thứ 2, c là đối dòng lệnh thứ 3,…
- Tham chiếu:

  Tên lệnh: $0

  Các tham số: $1, $2, $3, …

  Số các tham số: $#

# Câu lệnh đọc dữ liệu đầu vào

- Đọc dữ liệu từ bàn phím và ghi vào biến

  Cú pháp: **read <options> variable_names**

  ví dụ: *read name*

  hoặc *read –p “Enter yourname: ” name*

# Cấu trúc điều khiển

- **Cấu trúc rẽ nhánh if**

Cú pháp

*if [ conditions ]*

*then*

*<statements>*

*elif [conditions] then <statements>...*

*else <statements>*

*fi*

# Cấu trúc điều khiển

| Mathematical Operator in Shell Script | Meaning | Normal Arithmetical/ Mathematical Statements | But in Shell | |
|---|---|---|---|---|
| | | | For test statement with if command | For [ expr ] statement with if command |
| -eq | is equal to | 5 == 6 | if test 5 -eq 6 | if [ 5 -eq 6 ] |
| -ne | is not equal to | 5 != 6 | if test 5 -ne 6 | if [ 5 -ne 6 ] |
| -lt | is less than | 5 < 6 | if test 5 -lt 6 | if [ 5 -lt 6 ] |
| -le | is less than or equal to | 5 <= 6 | if test 5 -le 6 | if [ 5 -le 6 ] |
| -gt | is greater than | 5 > 6 | if test 5 -gt 6 | if [ 5 -gt 6 ] |
| -ge | is greater than or equal to | 5 >= 6 | if test 5 -ge 6 | if [ 5 -ge 6 ] |

**NOTE:** == is equal, != is not equal.

# Cấu trúc điều khiển

- **Cấu trúc rẽ nhánh case**

Cú pháp

```
case <variable>
in
    value_1)
            <statements>;;
    value_2)
            <statement>;;
    value_3)
            <statements>;;
    ...
    *) # the exceptions
    exit;;
esac
```

# Vòng lặp for

**Cú pháp 1:**

*for <variable> in <list>*

*do*

    *<statements>*

*done*

**Cú pháp 2:**

*for (( initialization command ; stop conditions ;  next command ))*

*do*

    *<statements>*

*done*

# Vòng lặp for

**Ví dụ 1:**

```
#!/bin/bash
for i  in 1 2 3 4 5 6
do
        echo “$i”
done
```

**Ví dụ 2:**

```
#!/bin/bash
for (( i = 1; i <= 5; i++ ))
do
        echo “In ra lan thu $i”
done
```

# Hàm trong lập trình shell

- **Cú pháp**:

  *function-name()*

  *{*

  *command1*

  *command2*

  *…*

  *commandN*

  *return*

  *}*

- **Gọi hàm:**

  *function-name arg1 arg2 arg3 argN*

# Ví dụ:

```
#!/bin/bash
        sayHello()
        {
                echo “Hello $LOGNAME, Toi la An!”
        }
        sayHello
```