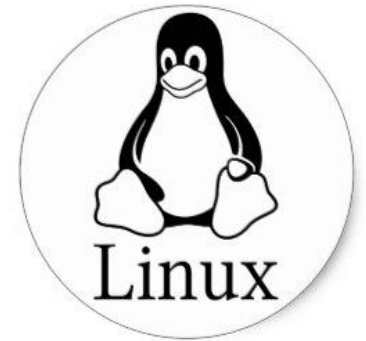


# LINUX

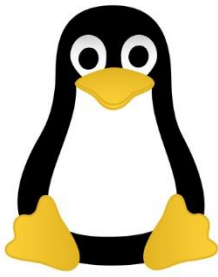


## LẬP TRÌNH SHELL

TS. TRẦN HẢI ANH

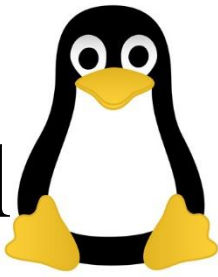
Freedom. Choices. Beautiful.

# Lập trình shell là gì



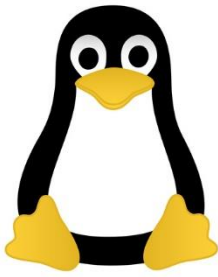
- Shell là trình thông dịch lệnh của Linux
  - Thường tương tác với người dùng theo từng câu lệnh.
  - Shell đọc lệnh từ bàn phím hoặc file
  - Nhờ hạt nhân Linux thực hiện lệnh
- Shell script
  - Các chương trình shell, bao gồm chuỗi các lệnh.

# Soạn và thực thi chương trình shell



- Sử dụng mọi trình soạn thảo dạng text:
  - vi, emacs, gedit
  - Nội dung bao gồm các câu lệnh được sử dụng trên dòng lệnh của Linux
  - Các câu lệnh trên cùng 1 dòng phải phân tách bằng dấu ;
- Thiết lập quyền thực thi cho chương trình shell
  - `chmod o+x ten_file`
- Thực thi
  - `bash ten_file`
  - `sh ten_file`
  - `./ten_file`

# Ví dụ shell đơn giản



- **\$vi first**

```
# My first shell script
```

```
clear
```

```
echo "Hello $USER"
```

```
echo "Today is \c ";date
```

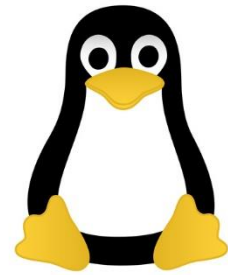
```
echo "Number of user login : \c" ; who | wc -l
```

```
echo "Calendar"
```

- **\$ chmod 755 first**

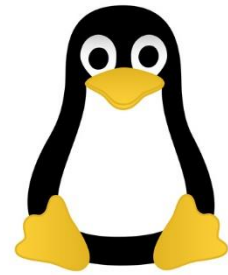
- **\$/first**

# Biến trong shell

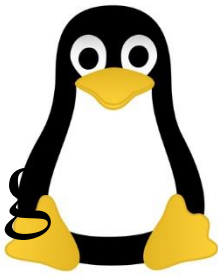


- Trong Linux shell có 2 loại biến:
    - Biến hệ thống:
      - Tạo ra và quản lý bởi Linux.
      - Tên biến là CHỮ HOA
    - Biến do người dùng định nghĩa
      - Tạo ra và quản lý bởi người dùng
      - Tên biến là chữ thường
    - Xem hoặc truy nhập giá trị các biến:
      - \$tên\_biến
      - echo \$HOME
      - echo \$USERNAME
- Phải có dấu \$ trước tên biến

# Một số biến hệ thống



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]\$\n	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



# Định nghĩa các biến của người dùng

- Cú pháp:

tên\_biến=giá\_trị

- In giá trị của biến

```
echo $tên_biến
```

- Ví dụ:

```
no=10
```

```
echo $no
```

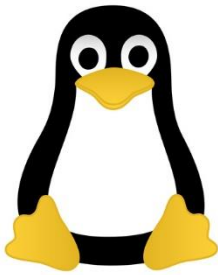
# Quy tắc đặt tên biến



- Tên biến phải bắt đầu bằng ký tự
  - HOME
  - SYSTEM\_VERSION
  - no
  - vech
- Không được để dấu cách hai bên toán tử = khi gán giá trị cho biến
  - no=10 # là đúng
  - no =10 # là sai
  - no = 10 #là sai

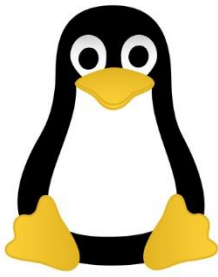


# Quy tắc đặt tên biến

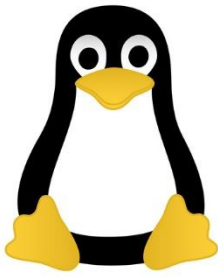


- Tên biến có phân biệt chữ hoa, thường
  - Các biến sau đây là khác nhau:  
no=10  
No=11  
NO=20  
nO=2
- Một biến không có giá trị khởi tạo thì bằng NULL
- Không được dùng dấu ?, \* để đặt tên các biến

# Ví dụ



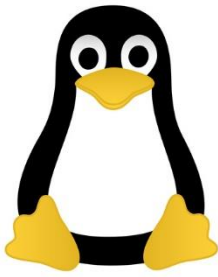
```
$ vi variscript
#
#
# Script to test MY knowledge about variables!
#
myname=Vivek
myos = TroubleOS
myno=5
echo "My name is $myname"
echo "My os is $myos"
echo "My number is myno, can you see this number"
```



# Lệnh echo

- Cú pháp:  
echo [option] [string, variables...]
- In một số ký tự đặc biệt trong tham số với tùy chọn -e:
  - \a alert (bell)
  - \b backspace
  - \c suppress trailing new line
  - \n new line
  - \r carriage return
  - \t horizontal tab
  - \\ backslash
- Ví dụ:  
**\$ echo -e "An apple a day keeps away \a\t\tdoctor\n"**

# Các phép toán số học



- Để thực hiện các phép tính toán số học cần dùng câu lệnh:

`expr biểu_thức_số_học`

Các toán tử: `+`, `-`, `\*`, `/`, `%`

- Ví dụ:

`expr 1 + 3`

`expr 2 - 1`

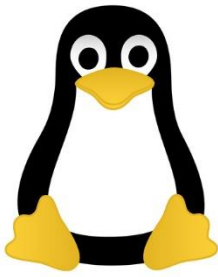
`expr 10 / 2`

`expr 20 % 3`

`expr 10 \* 3` # phép nhân là `\*` .

`echo `expr 6 + 3`` # đánh giá giá trị biểu thức `6+3` và in ra.

# Các dấu ngoặc



- Dấu ngoặc kép “ ”
  - 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 ` (cùng nút với dấu ~)
  - Yêu cầu thực hiện lệnh

VD:

```
$ echo "Today is `date`"
```

```
Today is Tue Jan ...
```

# Trạng thái kết thúc câu lệnh



- 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 biết trạng thái kết thúc câu lệnh trước đó
- Ví dụ

`rm unknow1file`

Nếu không có file này, hệ thống thông báo

rm: cannot remove `unkowm1file': No such file or directory

Nếu thực hiện lệnh:

`$ echo $?`

Sẽ in ra giá trị khác 0.

# Câu lệnh đọc dữ liệu đầu vào



- Đọc dữ liệu từ bàn phím và ghi và biến
- Cú pháp:

Read variable1

```
$ vi sayH
#
#Script to read your name from key-board
#
echo "Your first name please:"
read fname
echo "Hello $fname, Lets be friend!"
```

Run it as follows:

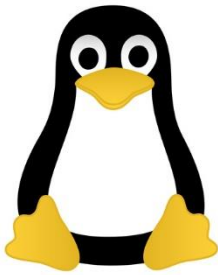
```
$ chmod 755 sayH
```

```
$ ./sayH
```

```
Your first name please: vivek
```

```
Hello vivek, Lets be friend!
```

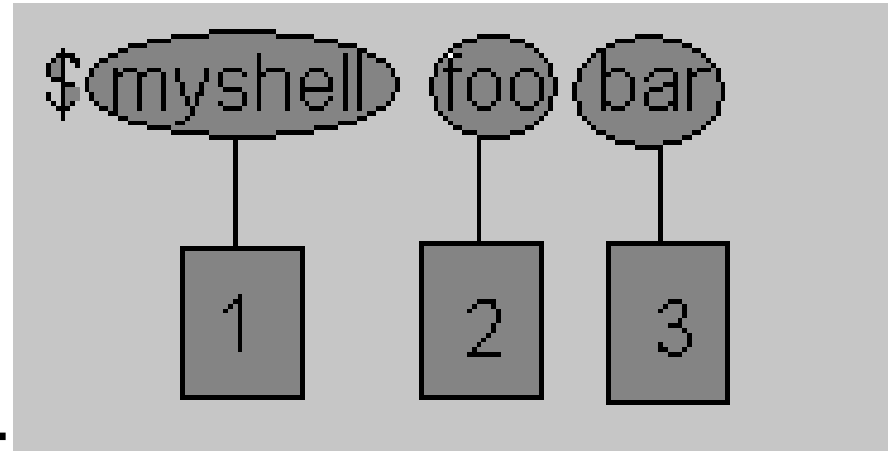
# Các tham số dòng lệnh



- Một chương trình shell có thể có các tham số dòng lệnh

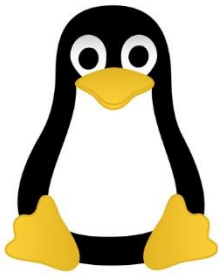
\$myshell foo bar

- Tham chiếu:
  - tên lệnh: \$0
  - các tham số: \$1, \$2.
  - Số các tham số: \$#





# Cấu trúc rẽ nhánh if



- Cú pháp:

if điều\_kiện

then

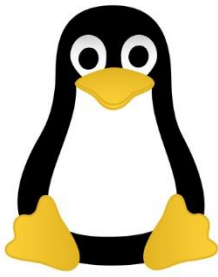
    câu lệnh 1

    ...

fi

Câu lệnh 1 được thực hiện khi điều\_kiện là đúng hoặc trạng thái kết thúc của điều\_kiện là 0 (kết thúc thành công).

# Cấu trúc rẽ nhánh if

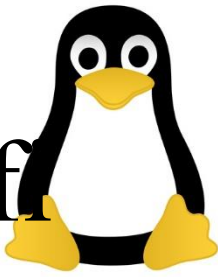


- Ví dụ, tệp showfile có nội dung:

```
#!/bin/sh
#
#Script to print file
#
if cat $1
then
echo -e "\n\nFile $1, found and successfully echoed"
fi
```

- Thực thi tệp:  
    [\\$./showfile foo](#)
- \$1 cho giá trị foo

# Cấu trúc rẽ nhánh if ... else ... fi



- Cú pháp

```
if điều_kiện then
```

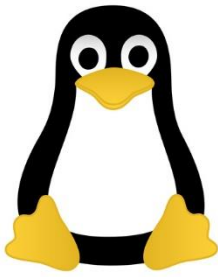
```
    câu_lệnh_1
```

```
    ....
```

```
else
```

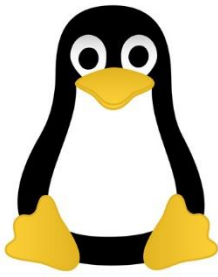
```
    câu_lệnh_2
```

```
fi
```



# Lệnh test

- Lệnh test được dùng để kiểm tra một biểu thức là đúng hay không và trả lại
  - 0 nếu biểu thức đúng
  - $<>0$ , trường hợp còn lại
- Cú pháp:  
Test biểu\_thức  
[biểu\_thức]
- Biểu thức có thể bao gồm:
  - Số nguyên
  - Các kiểu tệp
  - Xâu ký tự



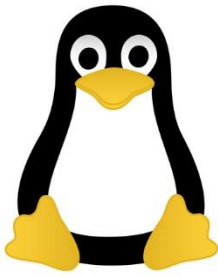
# Lệnh test

- Các phép toán kiểm tra

For Mathematics, use following operator in Shell Script

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.

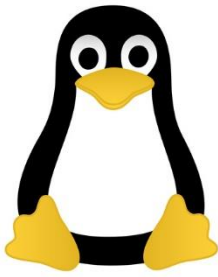


# Lệnh test

- Các phép so sánh chuỗi

For string Comparisons use

Operator	Meaning
string1 = string2	string1 is equal to string2
string1 != string2	string1 is NOT equal to string2
string1	string1 is NOT NULL or not defined
-n string1	string1 is NOT NULL and does exist
-z string1	string1 is NULL and does exist

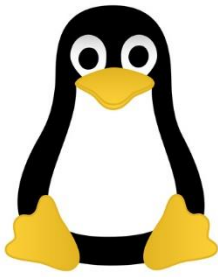


# Lệnh test

- Các phép kiểm tra file, thư mục

**Shell also test for file and directory types**

Test	Meaning
-s file	Non empty file
-f file	Is File exist or normal file and not a directory
-d dir	Is Directory exist and not a file
-w file	Is writeable file
-r file	Is read-only file
-x file	Is file is executable

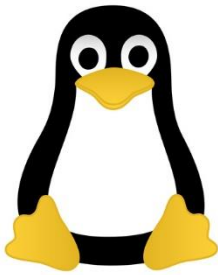


# Lệnh test

- Các phép toán logic:
  - NOT: !
    - ! Biểu\_thức
  - AND: -a
    - Biểu\_thức\_1 -a biểu\_thức\_2
  - OR: -r
    - Biểu\_thức\_1 -r biểu\_thức\_2



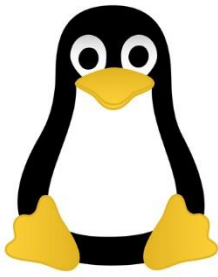
# Lệnh test



- Ví dụ tệp ispositive:

```
#!/bin/sh
#
# Script to see whether argument is positive
#
if test $1 -gt 0
then
echo "$1 number is positive"
fi
```

- \$ ./ispositive 5  
*5 number is positive*



# Cấu trúc lặp for

- Cú pháp

```
for { variable name } in { list }  
do
```

Các câu lệnh

```
done
```

Hoặc:

```
for (( expr1; expr2; expr3 ))  
do
```

Các câu lệnh

```
done
```

- Ví dụ tệp testfor

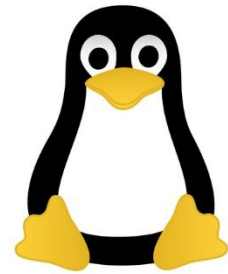
```
for i in 1 2 3 4 5
```

```
do
```

echo "Welcome \$i times"

```
done
```

# Cấu trúc lặp while



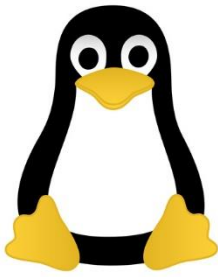
- Cú pháp

```
while [ condition ]  
do
```

```
command1  
command2  
command3
```

```
done
```

```
#!/bin/sh  
#  
#Script to test while statement  
#  
#  
if [ $# -eq 0 ]  
then  
    echo "Error - Number missing form command line argument"  
    echo "Syntax : $0 number"  
    echo " Use to print multiplication table for given number"  
exit 1  
fi  
n=$1  
i=1  
while [ $i -le 10 ]  
do  
    echo "$n * $i = `expr $i \* $n`"  
    i=`expr $i + 1`  
done
```



# Cấu trúc case

- Cú pháp

*case \$variable-name in  
pattern1) command*

*... ..  
command;;*

*pattern2) command*

*... ..  
command;;*

*patternN) command*

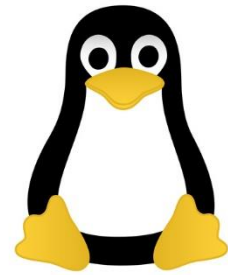
*... ..  
command;;*

*\*) command #default*

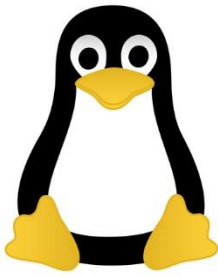
*... ..  
command;;*

*esac*

# Cấu trúc case



```
# if no vehicle name is given
# i.e. -z $1 is defined and it is NULL
#
# if no command line arg
if [ -z $1 ]
then
    rental="*** Unknown vehicle ***"
elif [ -n $1 ]
then
# otherwise make first arg as rental
    rental=$1
fi
case $rental in
    "car") echo "For $rental Rs.20 per k/m";;
    "van") echo "For $rental Rs.10 per k/m";;
    "jeep") echo "For $rental Rs.5 per k/m";;
    "bicycle") echo "For $rental 20 paisa per k/m";;
    *) echo "Sorry, I can not gat a $rental for you";;
esac
```



# Bài tập 1

- Viết chương trình cho biết tên năm âm lịch của một năm dương lịch cho trước. Yêu cầu chương trình nhận năm dương lịch tại dòng lệnh.
- Ví dụ:
  - \$lunar\_year 2004  
Giap Than
  - \$lunar\_year 2007  
Dinh hoi
  - \$lunar\_year 2013  
Quy ty

# Bài tập 1



- Năm âm lịch gồm Can và Chi

- Can (10): Giáp, Ất, Bính, Đinh, Mậu, Kỷ, Canh, Tân, Nhâm, Quý

- Chi (12): Tý, Sửu... Tuất, Hợi.

Mỗi năm Can tăng thêm 1, Chi tăng thêm 1 so với năm trước

Biết là 2013 là Quý Tỵ

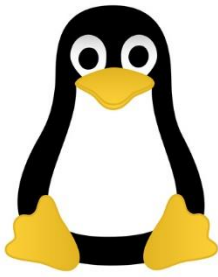
```
$lunar_year 2013
```

# Bài tập 2



- Viết một chương trình thực hiện chức năng của lệnh `ls`, tuy thế lệnh mới sẽ liệt kê các thư mục con trước rồi mới đến các tệp
- Viết chương trình gọi chương trình `lunar_year` và in ra bảng các năm dương lịch từ 1990 đến 2020 và tên năm âm lịch của chúng





# Bài tập 3

- Tạo chương trình nộp bài “nop\_bai” hoạt động như sau
  - Khi người dùng đăng nhập vào hệ thống với tên người dùng, ví dụ là tuanh, chương trình cho phép:
    - Nếu người dùng chạy
      - \$nop\_bai tep1 tep2
      - Chương trình copy các tệp vào thư mục: /home/baitaplinux/tuananh.
    - Người dùng này không sửa được bài của người dùng kia.
    - Bản thân người chủ tệp cũng không vào sửa bản nộp trực tiếp được
    - Người dùng có thể nộp lại bản mới (xóa bản cũ) khi chạy lại lệnh nop\_bai
    - Ghi nhật ký vào file log.txt các lần chương trình nop\_bai được chạy: ai chạy, ngày giờ nào, câu lệnh gì