**Tìm hiểu Lập trình shell linux (P1)**

**I. Shell script là gì**

Shell là chương trình giao tiếp với người dùng. Có nghĩa là shell chấp nhận các lệnh từ bạn (keyboard) và thực thi nó. Nhưng nếu bạn muốn sử dụng nhiều lệnh chỉ bằng một lệnh, thì bạn có thể lưu chuỗi lệnh vào text file và bảo shell thực thi text file này thay vì nhập vào các lệnh. Điều này gọi là shell script.

Định nghĩa: **Shell script** là một chuỗi các lệnh được viết trong plain text file. Shell script thì giống như batch file trong MS-DOS nhưng mạnh hơn.

Tại sao phải viết shell script:

• Shell script có thể nhận input từ user, file hoặc output từ màn hình.

• Tiện lợi để tạo nhóm lệnh riêng.

• Tiết kiệm thời gian.

• Tự động làm một vài công việc thường xuyên.

**II. Hướng dẫn tạo và thực thi chương trình shell**

**step1**: Tạo file hello.sh (trong thư mục cd /home/test/) nội dung như sau:

sử dụng vi, emacs, gedit... để soạn thảo nội dung

#!/bin/bash

echo "hello world"

Dòng đầu tiên chúng ta luôn đặt #!/bin/bash, đây là cú pháp bắt buộc. Sau # được hiểu là comment, chú thích của các đoạn mã.

**step2**: Sau đó, để script có thể thực thi ta phải cấp quyền cho nó

chmod 0777 hello.sh

**step3**: Thực thi file shell.

// có thể chạy file bằng 1 số cách sau

–  bash hello.sh

–  sh hello.sh

–  ./hello.sh

**III. 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

**1. Một số biến hệ thống**

VD: file hello.sh

#!/bin/bash

echo "hello"

echo $BASH\_VERSION

echo $BASH

echo $HOME

echo $PATH

Kết quả: run ./hello.sh

hello

4.2.24(1)-release

/bin/bash

/home/test

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

**2. Biến người dùng, cú pháp, quy tắc đặt tên**

**cú pháp:**

tên\_biến=value

* **tên\_biến** phải bắt đầu bằng ký tự
* Không có dấu cách 2 bên toán tử = khi gán giá trị cho biến

#Đúng

a=1

#sai

a = 1

#sai

a= 1

* Tên biến có phân biệt chữ hoa, thường

#các biến sau đây là khác nhau

a=1

A=2

* Một biến không có giá trị khởi tạo thì bằng NULL
* Không được dùng dấu ?, \* để đặt tên các biến

**ECHO Để in giá trị của biến**

**Cú pháp:**

echo [option][string,variables…]

#example

echo $tên\_biến

* 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

//example

$ echo -e "Hello\tTest"

#output: Hello Test

$ echo -e "Hello\nTest"

#output

Hello

Test

**export:**

lệnh export được sử dụng để export và validate biến môi trường trong session hiện tại (session hiện tại ở đây có thể tính là login vào server qua SSH hoặc login trực tiếp trên console).

Lưu ý toàn bộ các biến môi trường sẽ bị phá hủy khi disconnect hoặc là logout.

3. bash\_profile

Đối với một user trên các hệ điều hành unix (linux, solaris) mỗi user đều sẽ có một file tạm gọi là profile để chứa các biến môi trường trong thư mục home:

* Linux: tên file là .bash\_profile
* Solaris: tên file là .profile

Mỗi khi user login phâ hệ shell của hệ điều hành sẽ tự động tải các biến môi trường thiết lập trong file profile.

# .bash\_profile

# Get the aliases and functions

if [ -f ~/.bashrc ]; then

. ~/.bashrc

fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH

alias ls="ls -FA"

# ---------------------------------------------------

# OS User: Oracle

# Application: Oracle Database

# Version: Oracle 12c release 2

#

# ---------------------------------------------------

ORACLE\_SID=gitlog01; export ORACLE\_SID

ORACLE\_UNQNAME=gitlog; export ORACLE\_UNQNAME

ORACLE\_BASE=/app/oracle; export ORACLE\_BASE

ORACLE\_HOME=$ORACLE\_BASE/product/12c/gitlog; export ORACLE\_HOME

#ORACLE\_PATH=/u01/app/common/oracle/sql; export ORACLE\_PATH

# SQLPATH=/u01/app/common/oracle/sql; export SQLPATH

ORACLE\_TERM=xterm; export ORACLE\_TERM

NLS\_DATE\_FORMAT="DD-MM-YYYY HH24:MI:SS"; export NLS\_DATE\_FORMAT

#TNS\_ADMIN=$ORACLE\_HOME/network/admin; export TNS\_ADMIN

ORA\_NLS11=$ORACLE\_HOME/nls/data; export ORA\_NLS11

PATH=.:${PATH}:$HOME/bin:$ORACLE\_HOME/bin

PATH=${PATH}:/usr/bin:/bin:/usr/bin/X11:/usr/local/bin

#PATH=${PATH}:/app/common/oracle/bin

export PATH

LD\_LIBRARY\_PATH=$ORACLE\_HOME/lib

LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:$ORACLE\_HOME/oracm/lib

LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:/lib:/usr/lib:/usr/local/lib

export LD\_LIBRARY\_PATH

CLASSPATH=$ORACLE\_HOME/JRE

CLASSPATH=${CLASSPATH}:$ORACLE\_HOME/jlib

CLASSPATH=${CLASSPATH}:$ORACLE\_HOME/rdbms/jlib

CLASSPATH=${CLASSPATH}:$ORACLE\_HOME/network/jlib

export CLASSPATH

THREADS\_FLAG=native; export THREADS\_FLAG

export TEMP=/tmp

export TMPDIR=/tmp

umask 022

ulimit -t unlimited

ulimit -f unlimited

ulimit -d unlimited

ulimit -s unlimited

ulimit -v unlimited

Người dùng có thể sửa đổi nội dung file bash\_profile sau đó logout ra và login lại, hoặc trực tiếp dùng lệnh source để load profile mà không cần phải logout:

[oracle@gitlog01 ~]$ source .bash\_profile

**IV: Các phép toán số học**

Shell cung cấp cho ta một số biểu thức toán học.

**Cú pháp**:

expr toán\_hạng\_1 toán\_tử toán\_hạng\_2

example:

# phép cộng

$expr 1 + 2

# phép trừ

$expr 5 - 1

# phép chia

$expr 8 / 3 # output =2 phép chia chỉ lấy phần nguyên

$expr 8 % 5 # output =3 phép chia lấy phần dư

$expr 10 \\* 2 # output = 20 phép nhân

Chú ý: Phải có dấu cách trước và sau toán tử.

# example sai cú pháp

$expr 1+2

$expr 5- 1

**Các dấu ngoặc**

* Tất cả các ký tự trong dấu ngoặc kép đều không có ý nghĩa ñnh toán, trừ những ký tự sau \ hoặc $
* Dấu nháy ngược (`): nghĩa là yêu cầu thực thi lệnh

#example

$ echo "ngay hom nay la: `date`"

#ouput: ngay hom nay la: Wed Apr 27 10:43:59 ICT 2016

$ echo `expr 1 + 2`

#output = 3

$echo "expr 1 + 2"

#ouput: expr 1 + 2

\*\*Kiểm tra trạng thái trả về của 1 câu lệnh **cú pháp**

$echo $?

– Trạng thái 0 nếu câu lệnh kết thúc thành công. – Khác 0 nếu kết thúc có lỗi

# xóa file không tồn tại

rm abc.txt #output messge:( rm: cannot remove `abc.txt': No such file or directory )

# kiểm tra trạng thái câu lệnh rm abc.txt

$echo $? #output 1 nghĩa là có lỗi

$ echo "ngay hom nay la: `date`"

#ouput: ngay hom nay la: Wed Apr 27 10:43:59 ICT 2016

$echo $? #output 0, nghĩa là thành công

**V: Cấu trúc điều khiển trong shell script**

Cũng giống như các ngôn ngữ lập trình khác, Shell Scripts cũng cung cấp các vòng lặp: "for", "while"; và lệnh rẽ nhánh "if", "case".

**1. Cú pháp rẽ nhánh If** Cú pháp:

if điều\_kiện

then

câu lệnh 1

…

fi

**if...else...fi**

Cú pháp:

if điều\_kiện then

câu\_lệnh\_1

….

else

câu\_lệnh\_2

fi

**Vòng lặp For**

Cú pháp:

for { tên biến } in { danh sách }

do

# Khối lệnh

# Thực hiện từng mục trong danh sách cho đến cho đến hết

# (Và lặp lại tất cả các lệnh nằm trong "do" và "done")

done

#hoặc sử dụng for

for (( expr1; expr2; expr3 ))

do

# Lặp cho đến khi biểu thức expr2 trả về giá trị TRUE

done

example

# for 1

for i in 1 2 3 4 5

do

echo $i

done

#output: 1 2 3 4 5

#for 2

for (( i = 0 ; i <= 5; i++ )) # bao quanh bằng (())

do

echo $i

done

#ouput 1 2 3 4 5

**3. Vòng lặp While**

while [Điều kiện]

do

command1

command2

command3 .. ....

done

example demo1.sh

#!/bin/sh

echo "Nhap vao cac so can tinh tong, nhap so am de exit"

sum=0

read i

while [ $i -ge 0 ] # nếu i >= 0

do

sum=`expr $sum + $i`

read i # nhận giá trị từ người dùng

done

echo "Total: $sum."

Kết quả sau khi chạy ./demo1.sh

#ouput

./demo1.sh

Nhap vao cac so can tinh tong, nhap so am de exit

1

5

4

-1

Total= 10.

**VI: Lệnh test**

Lệnh test được dùng để kiểm tra một biểu thức là đúng hay không và trả lại

–  0 nếu biểu thức đúng

–  khác 0 sai

Cú pháp:

test biểu\_thức HOẶC [biểu thức]

**Các phép toán kiểm tra**

| **Mathematical Operator in,Shell Scrip** | **Meaning** | **Normal Arithmetical/ Mathematical Statements** |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | 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.

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 |

**Toán tử logic**

| **Operator** | **Meaning** |
| --- | --- |
| ! expression | Logical NOT |
| expression1,-a,expression2 | Logical AND |
| expression1,-o,expression2 | Logical OR |

**kiểm tra file, thư mục**

| **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 |

**Lập trình Shell (P2)**

Trong phần trước chúng ta đã tìm hiểu khái niệm bash script là gì, cách hoạt động, cú pháp khai báo biến và tham số, cũng như viết một câu lệnh điều kiện như thế nào. Trong phần 2 sẽ trình bày cách sử dụng vòng lặp, các trình thay thế, và cách khai báo function trong shell như thế nào.

**Vòng lặp**

Như các bạn thừa biết, vòng lặp để lặp đi lặp lại một hoặc nhóm câu lệnh nào đó cho đến khi gặp điều kiện dừng vòng lặp lại. Có rất nhiều loại vòng lặp trong bash script.

**Vòng lặp for**

Các bạn tạo file test.sh như phần trước và cấp quyền execute cho nó bằng lệnh chmod +x test.sh.

Ví dụ ta cần tính giai thừa của 5, cú pháp như sau:

#!/bin/bash

number=5

result=1

for ((i=1; i<=$number; i++))

do

result=$(($result \* $i))

done

echo $result

Sau khi chạy lệnh > ./test.sh kết quả trả về là 120 = 5!

**Vòng lặp for với khoảng số**

Để lặp số dùng vòng lặp for, trong shell còn có một cách viết khách cách thông thường đó là dùng khoảng số:

kq=1

for number in {1..5}

do

kq=$(($kq \* $number))

done

echo $kq

Kết quả trả về cũng là 120

Chưa hết, trong shell các bạn còn biết nó có thể in kết quả của một command line nào đó ra file hoặc hiển thị trên terminal. Ví dụ muốn in ra danh sách các file, thư mục trong thư mục hiện tại ta dùng vòng lặp for như sau:

for i in $(ls)

do

echo $i

done

Các bạn có thể chạy thử để thấy cả đống thư mục và file trong thư mục đang chạy test.sh được list ra.

**Vòng lặp while**

Vẫn trong bài toán tính giai thừa của 5, với while ta làm như sau:

# Tính giai thừa của 5

kq=1

i=1

while [ $i -le 5 ]

do

kq=$(($kq \* $i))

((i++))

done

echo $kq

Trong đó : -le là kiểm tra xem biến i <= 5

Cũng không khó hiểu lắm về thuật toán vì nó khá giống các ngôn ngữ lập trình khác, chỉ khó nhớ cái cú pháp viết, các bạn cần chú ý vì viết sai là sẽ báo lỗi ngay.

**Các trình thay thế trong shell**

Hiểu đơn giản đó là cách để bạn có thể xử lý các ký tự đặc biệt, ví dụ:

a=10

b=20

echo -e "Value of a is $a \n and value of b is $b"

Kết quả in ra sẽ là

Value of a is 10

and value of b is 20

Vậy -e cho phép bạn hiểu ký tự \n là ký tự xuống dòng. Bỏ -e để thấy sự khác biệt. Tương tự với \n bạn có thể thao tác với các ký tự đặc biệt khác.

**Thay thế lệnh**

Hãy xem qua ví dụ sau

DATE=`date`

echo "Date is $DATE"

Kết quả : Date is Thứ sáu, 14 Tháng 9 năm 2018 05:50:00 +07

Từ ví dụ bạn có thể thấy cú pháp `command` (dấu nháy dưới phím Esc) cho phép bạn lấy kết quả của 1 command gán vào 1 biến.

**Trình thay thế biến**

Các bạn xem ví dụ sau:

#!/bin/sh

echo ${var:-"Variable is not set"}

echo "1 - Value of var is ${var}"

echo ${var:="Variable is not set"}

echo "2 - Value of var is ${var}"

unset var

echo ${var:+"This is default value"}

echo "3 - Value of var is $var"

var="Prefix"

echo ${var:+"This is default value"}

echo "4 - Value of var is $var"

echo ${var:?"Print this message"}

echo "5 - Value of var is ${var}"

Kết quả

Variable is not set

1 - Value of var is

Variable is not set

2 - Value of var is Variable is not set

3 - Value of var is

This is default value

4 - Value of var is Prefix

Prefix 5 - Value of var is Prefix

Việc thực hiện thay thế giá trị của biến được thể hiện trong bảng sau đây:

| **Mẫu** | **Mô tả** |
| --- | --- |
| ${var} | Thay thế giá trị của var |
| ${var:-word} | Nếu var là null hoặc unset, thì word được thay thế cho var. Giá trị của var không thay đổi. |
| ${var:=word} | Nếu var là null hoặc unset, var được thiết lập là giá trị của word . |
| ${var:?message} | Nếu var là null hoặc unset, message được in là lỗi. Lệnh này kiểm tra xem các biến có được thiết lập đúng không. |
| ${var:+word} | Nếu var được thiết lập, word được thay thế cho var. Giá trị của var không thay đổ |

**Function**

cũng như các ngôn ngữ lập trình khác, shell cũng cho phép khai báo function để thực hiện một nhóm các lệnh và trả về cho chúng ta kết quả có thể sử dụng tùy từng mục đích. Syntax khá đơn giản:

function\_name () {

list of commands

}

Ví dụ :

said () {

echo "Hello Phuc"

}

said

Kết quả: Hello Phuc

Cách sử dụng có vẻ khá giống javascript đúng không? Truyền tham số cho function thì hơi khác một chút, bạn phải làm như sau:

said () {

echo "Hello $1 $2"

}

said Phuc Phuong

Kết quả: Hello Phuc Phuong

Vậy làm sao chúng ta có thể return kết quả từ một hàm đây:

sum () {

a=$(($1 + $2))

return $a

}

sum 10 20

ret=$?

echo "value is $ret"

Kết quả: value is 30

Gọi một function trong một function khác cũng khá đơn giản

number\_one () {

echo "This is the first function"

number\_two

}

number\_two () {

echo "This is now the second function"

}

number\_one

Kết quả:

This is the first function

This is now the second function

**Lập Lịch Tasks Trên Linux Sử Dụng Crontab**

Nếu bạn đang có một website mà nó "nặng nề" trên webserver, bạn có lẽ muốn chạy một vài tiến trình ví dụ như tạo ra những hình nhỏ hoặc thu thập dữ liệu ngầm. Theo cách này, nó không thể hiển thị với một giao diện người dùng. Linux có một chương trình tuyệt vời dành cho việc này. Nó cho phép các tác vụ chạy một cách ngầm tự động bên dưới đều đặn trong một khoảng thời gian. Bạn cũng có thể sử dụng nó để tự động sao lưu, đồng bộ files, lập lịch cập nhật và nhiều hơn thế nữa. Chào mừng đến với thế giới của crontab.

**Giới Thiệu**

Crontab (cron xuất phát từ "chronos", theo tiếng Hy Lạp có nghĩ là thời gian, tab là viết tắt của table), được tạo ra trên Unix và hoặc những hệ điều hành nhân Unix, thường dùng để lập lịch cho các câu lệnh để thực thi định kỳ. Để xem được những crontab nào đang chạy trên hệ thống của bạn, bạn có thể mở terminal và chạy:

sudo crontab -l

Để chỉnh sửa danh sách các crontab này, bạn có thể chạy:

sudo crontab -e

Nó sẽ mở trình soạn thảo mặc định (có thể là vi hoặc pico, có thể thay đổi trình soạn thảo mặc định nếu bạn muốn) để chúng ta thao tác với crontab. Nếu bạn lưu lại và thoát trình soạn thảo, tất cả cronjobs được lưu lại trong crontab. Cronjobs được viết theo định dạng sau:

\* \* \* \* \* /bin/execute/this/script.sh

Như bạn đã nhìn thấy, có 5 dấu sao. Những dấu sao này thể hiện những thành phần khác nhau của ngày theo thứ tự sau:

* minute - phút (0 - 59)
* hour - giờ (0 - 23)
* day or month - ngày trong tháng (0 - 31)
* month - tháng (1 - 12)
* day of week (0 - 6 ~ Sunday - Saturday)

**Thực thi mỗi phút**

Nếu bạn để toàn dấu sao, nó có nghĩa là "mỗi". Nó có lẽ hơi khó hiểu một chút. Hãy sử dụng ví dụ trước đó:

\* \* \* \* \* /bin/execute/this/script.sh

Nó chứ toàn dấu sao nên nó sẽ thực thi

Theo thứ tự từ trái sang phái các dấu \* có ý nghĩa như sau:

* Mỗi phút
* Trong mỗi giờ
* Trong mỗi ngày
* Trong mỗi tháng
* Và mỗi ngày trong tuần

Nói ngắn gọn, kịch bản này được thực thi mỗi phút một lần, không có ngoại lệ.

**Thực thi vào 1AM mỗi Thứ Sáu**

Vậy nếu chúng muốn lập lịch cho kịch bản chạy vào 1 giờ sáng mỗi Thứ Sáu, chúng ta cần lệnh sau:

0 1 \* \* 5 /bin/execute/this/script.sh

Kịch bản này sẽ được thực thi khi giờ hệ thống:

* minute - phút: 0
* of hour - của giờ: 1
* of day of month - Của ngày trong tháng: (every day of month)
* of month - Của tháng: \* (every month)
* and weekdays - Và ngày trong tuần: 5 (=Friday)

**Thực thi vào 1AM từ Thứ Hai đến Thứ Sáu**

Nếu chúng ta muốn lập lịch chạy từ vào 1 giờ sáng từ Thứ Hai đến Thứ Sáu, chúng ta cần lệnh sau:

0 1 \* \* 1-5 /bin/execute/this/script.sh

Kịch bản này sẽ được thực thi khi giờ hệ thống:

* minute - phút: 0
* of hour - của giờ: 1
* of day of month - Của ngày trong tháng: (every day of month)
* of month - Của tháng: \* (every month)
* and weekdays - Và ngày trong tuần: 1-5 (=Monday-Friday)

**Lập lịch một cách ngắn gọn**

Điều gì xảy ra nếu bạn muốn chạy thứ gì đó cứ 10 phút một lần? Chúng ta có thể làm như sau:

0,10,20,30,40,50 \* \* \* \* /bin/execute/this/script.sh

Nhưng crontabs cho phép bạn thiết lập tốt hơn:

\*/10 \* \* \* \* /bin/execute/this/script.sh

Nó làm điều tương tự như câu lệnh ở trên. Bạn có thể làm toán chứ?

**Từ khóa đặc biệt**

Với dấu sao đầu tiên, bạn có thể sử dụng những từ khóa sau thay cho con số:

@reboot Chạy một lần mỗi khi khởi động lại

@yearly Chạy một lần mỗi năm "0 0 1 1 \*"

@annually (Tương tự @yearly)

@monthly Chạy mỗi tháng một lần "0 0 1 \* \*"

@weekly Chạy mỗi tuần một lần "0 0 \* \* 0"

@daily Chạy một lần mỗi ngày "0 0 \* \* \*"

@midnight (Tương tự @daily)

@hourly Chạy một lần mỗi giờ "0 \* \* \* \*"

Và bỏ trống những phần còn lại. Ví dụ:

@daily /bin/execute/this/script.sh

**Lưu Crontab output**

Mặc định, cron lưu output của kịch bản "/bin/execute/this/script.sh" trong hòm thư (mailbox) của người dùng. Nhưng tốt hơn là lưu output vào từng file log. Như ví dụ dưới đây:

\*/10 \* \* \* \* /bin/execute/this/script.sh >> /var/log/script\_output.log 2>&1

Linux có thể báo cáo nhiều cấp độ khác nhau. Có đầu ra chuẩn STDOUT và đầu ra lỗi STDERR. STDOUT đánh dấu là 1, STDERR đánh dấu là 2. Vậy, câu lệnh dưới đây báo cho Linux lưu STDERR trong STDOUT, tạo mội luồng dữ liệu cho thông báo và lỗi.

2>&1

Bây giờ, chúng ta có một output stream, chúng ta có thể lưu dữ liệu vào file. Ký hiệu ">" sẽ ghi đè file, trong khi ">>" sẽ thêm vào file. Trong trường hợp này, chúng ta muốn ghi thêm vào file

>> /var/log/script\_output.log

**Gửi crontab output qua email**

Mặc định cron lưu output trong hòm thư của người dùng (trường hợp này là root) trong hệ thống cục bộ. Nhưng bạn cũng có thể cấu hình crontab để gửi toàn bộ output đến một địa chỉ thực sự bằng việc thiết lập một crontab bắt đầu bằng

MAILTO="example@domain.com"

Nếu bạn muốn nhận chỉ crontab output trong mail, hãy chắc chắn gói này được cài đặt:

aptitude install mailx

và Đổi cronjob như sau:

\*/10 \* \* \* \* /bin/execute/this/script.sh 2>&1 | mail -s "Cronjob ouput" yourname@yourdomain.com

**Xóa crontab output**

Nó thật dễ dàng:

\*/10 \* \* \* \* /bin/execute/this/script.sh > /dev/null 2>&1

Chỉ dẫn mọi thứ đến "null device", giống như một hố đen. Trong những hệ điều hành nhân Unix, /dev/null là một file đặc biệt hủy bỏ tất cả dữ liệu viết lên nó.

**Lưu ý**

Rất nhiều kịch bản được thử trên môi trường BASH với biến môi trường PATH đã được thiết lập. Với cách này, nó cho phép kịch bản của bạn chạy trên shell. Nhưng nếu chạy cron nơi mà biến PATH khác đi, kịch bản sẽ không thể tìm thấy file thực thi và không chạy được. Việc thiết lập biến PATH không phải một phần trong crontab, nó là trách nhiệm của người thực thi, vì vậy sẽ hữu ích hơn nếu đặt một biến PATH ở đầu file (ngay phía sau MAILTO)

**Kết Luận**

Trước đây, mỗi khi cần làm một công việc gì trên Server (chủ yếu là sao lưu dữ liệu), đều phải kết nối đến nó và thao tác một cách thủ công. Nhưng từ lúc biết được crontab, công việc đã dễ dàng di rất nhiều. Tôi đã thiết lập sao lưu tự động hằng ngày và không bao giờ còn lo về vấn đề này nữa. Đây chỉ là một ví dụ điển thình trong những việc mà tôi đã làm với crontab, thật là tuyệt vời đúng không nào? Tôi mong các bạn cũng sẽ tận dụng được crontab để giảm bớt được những thao tác thủ công mà ta phải làm đi làm lại hằng ngày, hàng giờ khi làm việc với Unix/Linux.