Bài làm

**1.Shell của UNIX/Linux**

Mọi thứ được thực hiện trên Unix đều bởi tiến trình. Vậy tạo ra tiến trình như thế nào? Cách thứ nhất là viết ra các chương trình mà các chương trình này biết cách tạo ra tiến trình (C/C++). Tuy nhiên cách này đòi hỏi nhiều hiểu biết và nỗ lực. Cũng như các hệ điều hành làm việc kiểu ảo khác, Unix hổ trợ một phương tiện xử lí lệnh làm giao diện giữa lệnh máy (mà người dùng đưa vào) và việc thực thi của lệnh đó (bởi Unix). Phương tiện đó gọi là shell. Từ khi ra đời Unix đã có vài kiểu shell, đó là Bourne, C, Korn shell. Thực ra shell làm gì? Tòan bộ mục đích của shell là để khởi động các tiến trình xử lí lệnh đưa vào: yêu cầu đưa (dòng) lệnh vào, đọc đầu vào, thông dịch dòng lệnh đó, và tạo ra tiến trình để thực hiện lệnh đó. Nói cách khác shell quét dòng lệnh đưa vào máy tính, cấu hình môi trường thực thi và tạo tiến trình để thực hiện lệnh.

Thực tế tìm hiểu shhell là để học ngôn ngữ lập trình, cho dù không phức tạp như C, hay các ngôn ngữ khác, nhưng cũng phải qua những đòi hỏi cần thiết. Trong Unix/Linux có các lọai shell khác nhau và có thể lựa chọn để dùng theo nhu cầu mà người dùng thấy phù hợp.

Linux/Unix tách biệt các ứng dụng, lệnh gọi các hàm chức năng của nhân thành những đơn thể rất nhỏ (tiến trình). Tuy nhiên, nhiều lệnh của Linux có thể kết hợp lại với nhau để tạo nên chức năng tổng hợp rất mạnh mẽ.

Ví dụ: $ ls -al | more

Lệnh trên được kết hợp bằng hai lệnh, ls liệt kê toàn bộ danh sách tệp và thư mục trên đĩa ra màn hình, nếu danh sách quá dài, ls chuyển dữ liệu kết xuất cho lệnh more xử lý hiển thị kết quả thành từng trang màn hình. Linux có cách kết hợp dữ liệu kết xuất của các lệnh với nhau thông qua cơ chế chuyển tiếp (redirect), ống dẫn (pipe).

Hầu hết các Shell trong Unix/Linux sử dụng một ngôn ngữ gần giống với C (điều này cũng dễ hiểu bởi trong thế giới Unix/Linux, C là ngôn ngữ lập trình thống trị). Ngôn ngữ Shell càng giống C thì lập trình viên hay người điều khiển Linux càng cảm thấy thân thiện với HĐH.

Các shell trên Unix/Linux:

1. sh: shell nguyên thủy áp dụng cho Unix, dòng shell sử dụng cấu trúc lệnh của C làm ngôn ngữ kịch bản. Được tạo ra đầu tiên bởi Bia Joy. Là shell thông dụng thứ hai sau bash shell.
2. bash: shell chủ yếu của Linux. Ra đời từ dự án GNU. bash (Viết tắt của Bourne Again Shell có lợi điểm là mã nguồn được công bố rộng rãi.
3. rc: shell mô rộng của csh với nhiều tương thích với ngôn ngữ C hơn. rc cũng ra đời từ dự án GNU.

**2. SỬ DỰNG SHELL NHƯ NGÔN NGỮ LẬP TRÌNH**

Trước hết bạn dùng lệnh

$cat > first.sh hay các trình soạn thảo như vi hay emacs (hoặc mc) để soạn nội  
dung tập tin first.sh như sau:  
# ! /bin/ sh  
# first.sh  
# Script nay se tìm trong thư mục hiện hành các chuỗi  
# mang nội dung main( ) , nội dung của fìle sẽ được hiển thị ra màn hình nếu tìm thấy.  
for file in \*  
do  
 if grep -l 'main( ) ' $file  
 then  
 more $fỉle  
 fi  
done  
exit 0

Không như chú thích của C, một dòng chú thích (comment) trong ngôn ngữ shell bắt đầu bằng ký tự #. Tuy nhiên Ở đây có một chú thích hơi đặc biệt đó là #!/bin/sh. Đẩy thực sự không phải là chú thích. Cặp ký tự #! là chỉ thị yêu cấu shell hiện tại triệu gọi shell sh nằm trong thư mục /bin. Shell sh sẽ chịu trách nhiệm thông dịch các lệnh nằm trong tập tin script được tạo.

Chỉ thị #! Còn được dùng để gọi bất kì chương trình nào ta muốn chạy trước khi script tiếp theo được dịch. Lệnh exit bảo đảm rằng script sau khi thực thi sẽ trả về mã lỗi, đây là cách mà hầu hết các chương trình nên làm, mặc dù mã lỗi trả vế ít khi được dùng đến trong trường hợp thực hiện tương tác trực tiếp từ dòng lệnh. Tuy nhiên, nhận biết mã trả về của một đoạn script sau khi thực thi, lại thường rất có ích nếu bạn triệu gọi script từ trong một script khác. Trong đoạn chương trình trên, lệnh exit sẽ trả về 0, cho biết script thực thi thành công và thoát khỏi shell gọi nó. Mặc dù khi đã lưu tập tin script với tên .sh, nhưng UNIX và Linux không bắt buộc điều này. Hiếm khi Linux sử dụng phần đuôi mở rộng của tập tin làm dấu hiệu nhận dạng, do đó tệp tệp script có thể là tùy ý. Tuy vậy .sh vẫn là cách chúng ta nhận ngay ra một tập tin có thể là script của shell một cách nhanh chóng.

**3. CÚ PHÁP NGÔN NGỮ SHELL**

Chúng ta đã thấy cách viết lệnh và gọi thực thi tập tin scirpt. Phần tiếp theo nay dành cho bạn khám phá sức mạnh của ngôn ngữ lập trình shell. Trái với lập trình bằng trình biên dịch khó kiểm lỗi và nâng cấp, lập trình script cho phép bạn dễ dàng sửa đổi lệnh bằng ngôn ngữ văn bản. Nhiều đoạn script nhỏ có thể kết hợp lại thành một script lớn mạnh mẽ và rất hữu ích. Trong thế giới UNIX và Linux đôi lúc gọi thực thi một chương trình, bạn khó mà biết được chương trình được viết bằng script hay thực thi theo mã của chương trình nhị phân, bởi vì tốc độ thực thi và sự uyển chuyển của chúng gần như ngang nhau. Phần này chúng ta sẽ học về:

• Biến: kiểu chuỗi, kiểu số, tham số và biến môi trường  
• Điều kiện: kiểm tra luận lý Boolean bằng shell  
• Điều khiển chương trình: if, elif, for , while, until, case  
• Danh shell  
• Hàm  
• Các hình nội tại của shell  
• Lấy về kết quả của một lệnh

3.1. Sử dụng biến

Thường bạn không cần phải khai báo biến trước khi sử dụng. Thay vào đó biến sẽ được tự động tạo và khai báo khi lần đầu tiên tên biến xuất hiện, chảng hạn như trong phép gán. Mặc định, tất cả các biến đều được khởi tạo và chứa trị kiểu chuỗi (string). Ngay cả khi dữ liệu mà bạn đưa vào biến là một con số thì nó cũng được xem là định dạng chuỗi. Shell và một vài lệnh tiện ích sẽ tự động chuyển chuỗi thành số để thực hiện phép tính khi có yêu cầu. Tương tự như bản thân hệ điều hành và ngôn ngữ C, cú pháp của shell phân biệt chữ hoa chữ thường, biến mang tên foo, Foo, và FOO là ba biến khác nhau.

Ví dụ:

$ xinahao=hello  
$ echo $xinchao

Output: Hello  
$ xin chao= "I am here"  
$echo $xin chao  
Output: I am here  
$ xinchao=12+l  
$echo $xin chao  
Output: 12+1

3.2. Điều kiện

Nền tảng cơ bản trong tất cả ngôn ngữ lập trình, đó là khả năng kiểm tra điều kiện và đưa ra quyết định rẽ nhánh thích hợp tùy theo điều kiện đúng hay sai. Trước khi tìm hiểu cấu trúc điều khiển của ngôn ngữ script, ta hãy xem qua cách kiểm tra điề kiện. Một script của shell có thể kiểm tra mã lỗi trả về của bất kỳ lệnh nào có khá năng triệu gọi từ dòng lệnh, bao gồm ả những tập tin lệnh script khác. Đó là lý do tại sao chúng ta thường sử dụng lệnh exit ở cuối mỗi scipt khi kết thúc.

Dước đây là cách sử dụng lệnh lệnh if, tuy đơn giản nhưng được sử dụng nhiều nhất. if kiểm tra điều kiện đúng hoặc sai để thực thi biểu thức thích hợp

Ví dụ:

if [-f hello.c ]  
then  
 . . .  
fi

Lưu ý, phải đặt khoảng trắng giữa lệnh [] và biểu thức kiểm tra, và dĩ nhiên giữa một lệnh và tham số truyềncho lệnh phải phân cách nhau bằng khoảng trắng để trình biên dịch có thể hiểu.

So sánh chuỗi

So sánh Kết quả

stringl = string2 true nếu 2 chuỗi bằng nhau (chính xác từng ký tự)

string1 != string2 true nếu 2 chuỗi không bằng nhau

-n string1 true nếu string1 không rổng

-z stringl true nếu string1 rỗng (chuổi null)

***So sánh toán học***

So sánh Kết quả

expression1 -eq expression2 true nếu hai biểu thức bằng nhau

expression1 -ne expression2 true nếu hai biểu thức không bằng nhau

expression1 -gt expression2 true nếu biểu thức expression1 lớn hơn expression2

expreesion1 -ge expression2 true nểu biểu thức expression1 lớn hơn hay bằng

expression2

expression1 -lt expression2 true nếu biểu thức expression1 nhỏ hơn expression2

expression1 -le exprebbion2 true nếu biểu thức expression1 nhỏ hơn hay bằng

expression2

!expression true nếu biểu thức expression là false (toán tử not)

***Kiểm tra điều kiện trên tập tin***

-d file true nếu file là thư mục

-e file true nếu file tồn tại trên đĩa

-f file true nếu file là tập tin thông thường

-g file true nếu set-group-id được thiết lập trên file

-r file true nếu file cho phép đọc

-s f ile true nếu kích thước file khác 0

-u file true nếu set-ser-id được áp đặt trên file

-w file true nếu file cho phép ghi

-x file true nếu file được phép thực thi

3.3. Cấu trúc điều khiển

Shell cung cấp cấu trúc lệnh điều khiển rất giống với các ngôn ngữ lặp trình khác đó là if, elif, for, while, until, case. Đối với một vài cấu trúc lệnh (ví dụ như case), shell đưa ra cách xử lý uyển chuyển và mạnh mẻ hơn. Những cấu trúc điếu khiển khác nếu có thay đổi chỉ là những thay đổi nhỏ không đáng kể.

**Lệnh for**: Sử dụng for để lặp lại một số lần với các giá trị xác định. Phạm vi lặp có thể nằm trong một tập hợp chuỗi chỉ định tường minh bởi chương trình hay là kết qủa trả về từ một biến hoặc biểu thức khác.

Ví dụ:

#!/bin/sh  
for foo in bar fud 13  
do  
 echo $foo  
done  
exit 0

Output:

$./ for\_loop.sh  
bar  
fud  
13

**Lệnh while:** Lệnh while cho phép thực hiện lặp vô hạn khi điều kiện kiểm tra vẫn còn đúng.

Ví dụ:

#!/bin/sh  
echo "Enter password"  
read trythis  
while [ "$trythis" != "secret" ]; do  
 echo "Sorry, try again"  
 read trythis  
done  
exit 0

Output:

$./password.sh

Enter password:

abc

Sorry, try again

secret #gõ đúng

**Lệnh case:** Lệnh case có cách sử dụng hơi phức tạp hơn các lệnh đã học, nhưng lệnh case rất linh động. case cho phép thực hiện so khớp nội dung của biến với một chuỗi mẫu pattern nào đó. Khi một mẫu được so khớp, thì (lệnh) statement tương ứng sẽ được thực hiện. Hãy lưu ý đặt hai dấu chầm nhảy ;; phía sau mỗi mệnh đề so khớp pattern, shell dùng dấu hiệu này để nhận dạng mẫu pattern so khớp tiếp theo mà biến cần thực hiện

Ví dụ:

#!/bin/sh  
echo "Is it morning? Please answer yes or no"  
read timeofday  
case "$timeofday" in  
 "yes") echo "Good Morning";;  
 "no" ) echo "Good Afternoon";;  
 "y" ) echo "Good Morning";;  
 "n" ) echo "Good Afternoon";;  
 \* ) echo "Sorry, answer not recognised";;  
esac  
exit 0

Cách thực hiện: Sau khi người dùng nhập vào câu trả lời, lệnh case sẽ lấy nội dung của biến $timeofday so sánh với từng chuỗi. Khi gặp chuỗi thích hợp nó sẽ thực thi lệnh đằng sau dấu) và kết thúc (không tiếp tục so khớp với các mẫu khác). Ký tự đại diện \* cho phép so khớp với mọi loại chuỗi. \* thường được xem như trường hợp so sánh đúng cuối cùng nếu các mẫu so sánh trước đó thất bại. Bạn có thể xem \* là mệnh đề default trong lệnh switch của C hay case … else của Pascal.

3.4. Danh shell thực thi lệnh (Lists)

**Danh sách AND (&&):** Danh shell AND cho phép thực thi một chuỗi lạnh kề nhau, lệnh sau chỉ thực hiện khi lệnh trước đã thực thi và trả về mã lỗi thành công. Cú pháp sử dụng như sau:  
Statement1 && statement2 && statement3 && . . .

Ví dụ:

#!/bin/sh  
touch file\_one  
rm -f file\_two  
if [ -f file\_one ] && echo "hello" && [ -f file\_two ] && echo "there"  
then  
 echo -e "in if"  
else  
 echo -e "in else"  
fi  
exit 0

Output:

$./and\_list.sh  
hello  
in else

**Danh sáchl OR ( || ):** Danh shell OR cũng tương tự với AND là thực thi một dãy các lệnh, nhưng nếu có một lệnh trả vế true thì việc thực thi ngừng lại. Cú pháp như sau:  
statementl || statement2 || statement3 && . . .

Ví dụ:

#!/bin/sh  
rm -f file\_one  
if [ -f file\_one ] || echo "hello" || echo "there"  
then  
 echo "in if"  
else  
 echo "in else"  
fi  
exit 0

Output:

$./and\_list.sh  
hello  
in if

3.5. Hàm (function)

Tương tự các ngữ trình khác, shell cho phép bạn tự tạo hàm hay thủ tục để triệu gọi bên trong script. Mặc dù bạn có thể gọi các script con khác bên trong script chính, chúng tương tự như việc gọi hàm. Tuy nhiên triệu gọi các script con thường tiêu tốn tài nguyên và không hiệu quả bằng triệu gọi hàm.

Ví dụ:

#!/bin/sh  
foo() {  
 echo "Function foo is executing"  
}  
echo "script starting"  
foo  
echo "script ended"  
exit 0

Output:

$./my\_function.sh  
script starting  
Function foo is executing  
script ending

**Cách truyền tham số:** Shell không có cách khai báo tham số cho hàm như cách của C, Pascal hay các ngôn ngữ lập trình thông thường khác. Việc truyền tham số cho hàm tương tự như truyền tham số trên dùng lệnh. Ví dụ để truyền tham số cho foo(), ta gọi hàm như sau  
foo "paraml", "param2", param3 . . .

Ví dụ:

#!/bin/sh  
yes\_or\_no() {  
 echo "In function parameters are $\*"  
 echo "Param 1 $1 and Param2 $2"  
 while true  
 do  
 echo -n "Enter yes or no"  
 read x  
 case "$x" in  
 y | yes ) return 0;;  
 n | no ) return 1;;  
 \* ) echo "Answer yes or no"  
 esac  
done  
}  
echo "Original parameters are $\*"  
if yes\_or\_no "Is your name” “ $1?"  
then  
 echo "Hi $1"  
elif  
 echo "Never mind"  
fi  
exit 0

Output:

$,/get\_name.sh HoaBinh SV  
Original parameters are HoaBinh SV  
In function parameters are Is your name HoaBinh  
Param 1 Is your name param 2 HoaBinh  
Is your name HoaBinh ?  
Enter yes or no : yes  
Hi HoaBinh, nice name

3.6. Các lệnh nội tại của shell

**Break:** Tương tự ngôn ngữ C, shell cung cấp lệnh break để thoát khỏi vòng lập for,while hoặc until bất kề điều kiện thoát của các lệnh này có diễn ra hay không.

Ví dụ:

#!/bin/sh  
rm -rf fred\*  
echo > fred1  
echo > fred2  
mkdir fred3  
echo > fred4  
for file in fred\*  
do  
 if [ -d "$file" ]; then  
 break;  
 fi  
done  
echo first directory fred was $file  
exit 0

Đoạn script trên dùng lệnh for để duyệt toàn bộ tên của tập tin và thư mục hiện hành bất đầu bằng chuỗi fred. Khi phát hiện thư mục đầu tiên trong danh shell các tập tin, sẽ in ra tên thư mục và dùng break rể thoát khỏi vòng lặp (không cần duyệt tiếp các tập tin khác).

**Cotinue:** Lệnh continue thường được dùng bên trong vòng lặp, continue yêu cầu quay lại thực hiện bước lặp kế tiếp mà không cần thực thi các khối lệnh còn lại.

Ví dụ:

#!/bin/sh  
rm -rf fred\*  
echo > fred1  
echo > fred2  
mkdir fred3  
echo > fred4  
for file in fred\*  
do  
 if [ -d "$file" ]; then  
 continue  
 fi  
 echo file is $file  
done  
exit 0

Đoạn script trên dùng lệnh for để duyệt toàn bộ tên của tập tin và thư mục hiện hành bắt đầu bằng chuỗi fred. Nếu kiểm tra tên tập tin là một thư mục, thì continue yêu cầu quay lại duyệt tiếp file khác. Ngược lại lệnh echo sẽ in ra tên tệp.

**Printf**: Lệnh printf của shell tương tự printf của thư viện C. Mặc dù vậy, cơ bản printfcủa shell có một số hạn chế sau: không hỗ trợ định dạng số có dấu chấm động (float) bởi vì tất cả các tính toán của shell đấu dựa trên số nguyên. Dầu sổ \ dùng chỉ định các hiển thị đặc biệt trong chuỗi (xem bảng dưới). Dấu % dùng định dạng số và kết xuất chuỗi. Dưới đây là danh sách các ký tự đặc biệt có thể dùng với dấu \, chúng được là chuỗi thoát.

Chuỗi thoát (escape sequence) Ý nghĩa

\\ Cho phép hiển thị ký tự \ trong chuỗi

\a Phát liếng chuông (beep)

\b Xóa backspace

\f Đẩy dòng

\n Sang dòng mới

\r Về đầu dòng

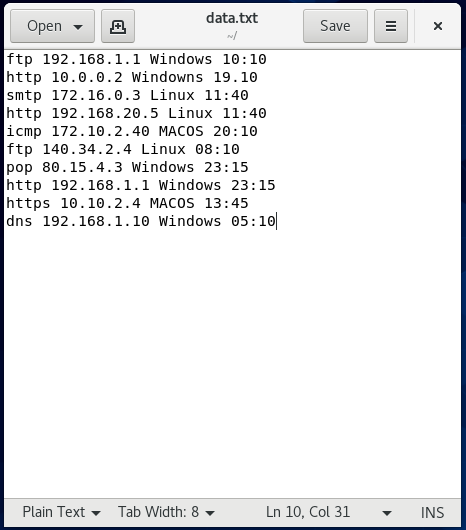
\t Căn tab ngang

\v căn tab dọc

**Return:** Lệnh return dùng để trả về giá trị của hàm. Lệnh return không tham số sẽ trả về mã lỗi của lệnh vừa thực hiện sau cùng.

4. Demo

Ta có 1 firewall lưu trữ các kết nối ra vào trong hệ thống, firewall đó đã lưu lại dữ liệu kết  
nối dưới tên là data, có cấu trúc như sau:



1. Viết 1 hàm nhanthongtin() để cho hệ thống có thể lấy các thông tin chép vào file data. (Trong tình huống viết giả lập này, các em thay hệ thống nhập thông tin vào file data theo cấu trúc dạng như trên).  
2. Sau khi đã có file dữ liệu data làm ở câu trên, hãy viết 1 hàm tên là xuatthongtin() để xuất ra toàn bộ thông tin đang nằm trong file data.   
3. Hãy viết hàm tên là: getIPfromOS() cho người dùng nhập vào 1 hệ điều hành, xuất ra thông tin các địa chỉ IP tương ứng của hệ điều hành.   
4. Hãy viết hàm tên là getIPfromProtocol cho người dùng nhập vào 1 giao thức, hãy xuất ra thông tin về địa chỉ IP đã truy cập.   
5. Hãy viết hàm tên là CountProtocol nhập vào giao thức, hãy đếm xem giao thức này được kết nối bao nhiêu lần.  
6. Hãy viết hàm tên là findOS để kiểm tra xem trong file data, có bao nhiêu hệ điều hành truy cập. Lưu tên hệ điều hành này vào 1 mảng.   
7. Hãy viết hàm Info() làm công việc: từ mảng đã làm ở câu trên, hãy đếm xem, mỗi hệ điều hành như vậy truy cập bao nhiêu lần. Chép vào 1 file tên là thongke, theo dạng như sau:   
Windows 5  
Linux 3  
8. Hãy thực hiện xuất ra thông tin sau trên chương trình chính  
“Hay thuc hien lua chon:”  
“1. Nhap thong tin vao file data  
“2. Xuat toan bo thong tin trong file data”  
“3. Nhap vao he dieu hanh de biet dia chi IP”  
“4. Nhap vao giao thuc de biet IP”  
“5. Nhap vao giao thuc de biet so lan ket noi”  
“6. Thuc hien thong ke”  
“7. Ket thuc”  
Lưu ý: khi người dùng nhập vào số 7 mới cho kết thúc chương trình. Khi chọn số 6, sẽ cho thực hiện 2 hàm findOS và info.

Dưới đây là mã nguồn của từng câu. Để dễ hiểu chúng ta tạo từng phần của mã lệnh, sau đó có thể kết hợp chúng lại như một tổng thể duy nhất.

Câu 1: Như thường lệ phần đầu của script thường là lời triệu gọi shell thực thi sh, hàm này giúp ta nhập thêm thông tin vào file data.txt mình đã tạo sẵn và chứa dữ liệu theo đề bài.

|  |
| --- |
| #!/bin/bash  nhapthongtin(){      echo "Nhap"      read nhap      echo $nhap >> data.txt  } |

Câu 2: Hàm while giúp ta đọc toàn bộ dữ liệu trên file đến khi hết dữ liệu để đọc theo dòng, đồng thời xuất ra màn hình.

|  |
| --- |
| xuatthongtin(){      echo "Xuat file data.txt"      while read line      do          echo ${line}      done < data.txt  } |

Câu 3: Câu “`echo $line | awk 'BEGIN{FS=" "}{print $3}'`” giúp ta tách được từ thứ 3 từ dòng, để giúp ta so sánh với từ ta đã nhập.

|  |
| --- |
| getIPfromOS(){          echo "Nhap he dieu hanh de xuat IP"          read tim1          while read line          do              data=`echo $line | awk 'BEGIN{FS=" "}{print $3}'`              if [ "$data" = "$tim1" ]              then                  data=`echo $line | awk 'BEGIN{FS=" "}{print $2}'`                  echo $data              fi          done < data.txt  } |

Câu 4: Tương tự như câu 3, hàm này giúp ta tách được từ thứ nhất và thứ hai để so sánh.

|  |
| --- |
| getIPfromProtocol(){          echo "Nhap giao thuc de xuat IP"          read tim2          while read line          do              data=`echo $line | awk 'BEGIN{FS=" "}{print $1}'`              if [ "$data" = "$tim2" ]              then                  data=`echo $line | awk 'BEGIN{FS=" "}{print $2}'`                  echo $data              fi          done < data.txt  } |

Câu 5:

|  |
| --- |
| CountProtocol(){          echo "Nhap giao thuc de bet so lan ket noi"          read tim2          dem=0          while read line          do              data=`echo $line | awk 'BEGIN{FS=" "}{print $1}'`              if [ "$data" = "$tim2" ]              then                  data=`echo $line | awk 'BEGIN{FS=" "}{print $2}'`                  let dem++              fi          done < data.txt          echo $dem  } |

Câu 6:

|  |
| --- |
| findOS(){      dem=0      while read line          do              data=`echo $line | awk 'BEGIN{FS=" "}{print $3}'`              a[$dem]=$data              let dem++          done < data.txt      dem1=0      for((i=0;i<${#a[\*]};i++))      do          kq=0          for((j=0;j<$i;j++))          do              if [ "${a[$i]}" = "${a[$j]}" ]              then                  kq=1              fi          done      if [ $kq -eq 0 ]      then          b[$dem1]=${a[$i]}          let dem1++      fi      done      echo "So he dieu hanh"      echo ${b[\*]}  } |

Câu 7:

|  |
| --- |
| Info(){      for((i=0;i<${#b[\*]};i++))      do          dem=0          echo ${b[$i]}          while read line          do              data=`echo $line | awk 'BEGIN{FS=" "}{print $3}'`              if [ "$data" = "${b[$i]}" ]              then                  let dem++              fi          done < data.txt          echo $dem          echo ${b[$i]} $dem >> thongke.txt      done  } |

Câu 8: Chương trình chính bắt đầu tại đây, sau khi đã cài đặt hoàn tất.

|  |
| --- |
| while true  do      echo "----------------------------------"      echo "Phan Dong Thien Long - 3118410242"      echo "Hay thuc hien lua chon"      echo "1.Nhap thong tin vao file data"      echo "2.Xuat toan bo thong tin trong file data"      echo "3.Nhap vao he dieu hanh de biet dia chi IP"      echo "4.Nhap vao giao thuc de biet dia chi IP"      echo "5.Nhap vao giao thuc de biet so lan ket noi"      echo "6.Thuc hien thong ke"      echo "7.Ket thuc"      read chon      case $chon in          1)          clear          nhapthongtin;;          2)          clear          xuatthongtin;;          3)          clear          getIPfromOS;;          4) clear          getIPfromProtocol;;          5) clear          CountProtocol;;          6) clear          findOS          Info;;          7) break;;      esac  done |

Hình sau là giao diện chương trình:

|  |
| --- |
|  |