



Phần 1: Thực hành

1. Makefile

Makefile là gì?

- Makefile là một file dạng script chứa các thông tin:
 - Cấu trúc project (file, sự phụ thuộc)
 - Các lệnh để tạo file
- Lệnh make sẽ đọc nội dung Makefile, hiểu kiến trúc của project và thực thi các lệnh

Ví dụ:

- Project chứa 4 file: Shape.java, Rectangle.java, Square.java, Main.java
- 2 đối tượng Square và Rectangle cùng cài đặt giao diện Shape
- File thực thi là Main.java

Nội dung các file được cho trong folder đi kèm. Khi đó file Makefile có nội dung sau: Main: main.class

```
java Main
```

```
main.class: Main.java Rectangle.class Square.class
```

```
javac Main.java
```

```
Rectangle.class: Rectangle.java Shape.class
```

```
javac Rectangle.java
```

```
Square.class: Square.java Shape.class
```

```
javac Square.java
```

```
Shape.class: Shape.java
```

```
javac Shape.java
```

Thứ tự thực hiện:

Khi bạn thực hiện lệnh make, chương trình make sẽ nhảy đến phần đầu tiên là Main với mục đích để tạo ra nó, để làm được điều đó make đi kiểm tra lần lượt nội dung sau dấu : xem tệp tin main.class đã tồn tại chưa. Tệp đầu tiên là main.class chưa có, cần phải tìm câu lệnh nào đó mà ở đó main.class được tạo ra, make tìm ra dòng thứ 3 và nó nhảy đến thực hiện câu lệnh ở dòng này để tạo ra main.class. Sau khi tạo ra main.class, make tiến hành các bước tương tự. Sau khi tất cả các tệp liên quan được tạo ra, make mới có thể tạo ra tệp chạy cuối cùng là Main.

Vậy make thực hiện theo thứ tự như sau:

- Tạo ra các file object trước (main.class)

- Tạo ra chương trình nhị phân cuối cùng từ các file object đã được tạo ra trước đó (sum)

Sau khi đã tạo các file với nội dung như trên, vào thư mục chứa các file trên thực hiện lệnh:

```
student@linux$make
```

Kết quả:

```
25
```

```
30
```

2. Biểu thức chính quy

- **Biểu thức chính quy** (*regular expression* - viết tắt *regex*) là một cú pháp đặc biệt được sử dụng để mô tả các mẫu ký tự. Trong Linux, biểu thức chính quy thường được sử dụng cùng với các chương trình: *grep*, *sed*, *vi*, ... để hỗ trợ các thao tác tìm kiếm và thay thế trong văn bản. Biểu thức chính quy là tập hợp các ký tự điều khiển (*metacharacter*) với những ý nghĩa đặc biệt và các ký tự thông thường.

Các ký tự điều khiển trong biểu thức chính quy

Ký tự	Mô tả
<code>^</code>	Đánh dấu đầu dòng
<code>\$</code>	Đánh dấu cuối dòng
<code>.</code>	Biểu diễn ký tự bất kỳ
<code>[]</code>	Biểu diễn một ký tự bất kỳ trong cặp dấu <code>[]</code>
<code>[^]</code>	Biểu diễn phép phủ định của nội dung trong <code>[]</code>
<code>[-]</code>	Biểu diễn một ký tự bất kỳ bên trong vùng được chỉ rõ bởi <code>[]</code>
<code>()</code>	Biểu diễn các ký tự trong <code>()</code> như một nhóm ký tự
<code>\<</code>	Đánh dấu đầu từ
<code>\></code>	Đánh dấu cuối từ
<code>\b</code>	Đánh dấu biên từ
<code>\B</code>	Đánh dấu xâu rỗng không ở biên từ
<code>\n</code>	Biểu diễn ký tự/nhóm ký tự nằm giữa cặp <code>()</code> thứ <i>n</i> trước đó
<code>?</code>	Phép lặp: Ký tự/nhóm ký tự đi trước có thể xuất hiện hoặc không
<code>+</code>	Phép lặp: Ký tự/nhóm ký tự đi trước xuất hiện ≥ 1 lần
<code>*</code>	Phép lặp: Ký tự/nhóm ký tự đi trước xuất hiện ≥ 0 lần
<code>{n}</code>	Phép lặp: Ký tự/nhóm ký tự đi trước xuất hiện đúng <i>n</i> lần
<code>{n,}</code>	Phép lặp: Ký tự/nhóm ký tự đi trước xuất hiện $\geq n$ lần
<code>{n,m}</code>	Phép lặp: Ký tự/nhóm ký tự đi trước có thể xuất hiện từ <i>n</i> đến <i>m</i> lần
<code> </code>	Phép hoặc

Dùng `\` trước các ký tự điều khiển để sử dụng chúng như những ký tự thông thường.

Ví dụ:

- $[^09] \equiv$ Tập các ký tự khác 0 và 9
- $[a-z45] \equiv \{a, b, c, \dots, z, 4, 5\}$
- $0|([1-9][0-9]^*) \equiv$ Biểu diễn tập các số tự nhiên
- tux Biểu diễn các dòng bắt đầu bằng *tux*
- $03(6)(24)\backslash 1 \equiv 036246$
- $colou?r \equiv \{\text{color, colour}\}$

Các phiên bản *regex* trong Linux

Có 3 phiên bản để biểu diễn cú pháp của biểu thức chính quy:

- BRE (Basic Regular Expressions)
- ERE (Extended Regular Expressions)
- PCRE (Perl Regular Expressions)

Các phiên bản ra đời sau đưa ra bộ cú pháp mới hơn để hỗ trợ tốt hơn cho việc biểu diễn biểu thức chính quy.

Ví dụ: Xâu chứa ký tự 'i' hoặc 'a' được biểu diễn trong:

- BRE: $[i\backslash a]$
- ERE: $[i|a]$

Có thể bổ sung tùy chọn khi thực hiện một lệnh nào đó để lựa chọn phiên bản *regex* sẽ sử dụng.

Ví dụ: Lệnh *grep* sử dụng các tùy chọn:

- G: Đọc các xâu theo phiên bản BRE
- E: Đọc các xâu theo phiên bản ERE

- **Pipe** (giao tiếp giữa các tiến trình)

Đôi khi các tiến trình cần trao đổi thông tin cho nhau để xử lý. Một cơ chế được sử dụng khá phổ biến trong Linux là *pipe*, sử dụng chỉ thị $|$ cho phép đầu ra của tiến trình bên trái ký hiệu $|$ là đầu vào của tiến trình bên phải.

Ví dụ 1: *student@linux ~ \$ ls -R /etc | more*

Kết quả: Nội dung của thư mục */etc* được hiển thị ra màn hình theo từng trang

Ví dụ 2: *student@linux ~ \$ echo "hoa phong lan" | cat > hoa.txt*
student@linux ~ \$ cat hoa.txt

Kết quả: hoa phong lan

- **grep** là một công cụ hữu hiệu trong Linux để tìm kiếm trong nội dung tệp tin các dòng phù hợp với mẫu tìm kiếm.

Cú pháp: *grep [options] 'regex' fileName*

grep [options] 'regex' fileName > outputFile

(ghi nội dung tìm kiếm ra tệp *output-*

File)

Trong đó *[options]*:

- *'regex'*: là biểu thức chính quy biểu diễn mẫu tìm kiếm.
- *[options]*:
 - * -i: Tìm kiếm không phân biệt hoa thường
 - * -v: Tìm các kết quả không chứa các xâu biểu diễn bởi *'regex'*

- * -w: Đưa ra các dòng có chứa chuỗi biểu diễn bởi *'regex'* là một từ (tìm kiếm chính xác từ đó)
- * -c: Đếm số dòng chứa chuỗi *'regex'*
- * -e: Đếm số lần xuất hiện của chuỗi *'regex'* trong tệp tin
- * -A5: Đưa ra 5 dòng sau kết quả
- * -B5: In ra 5 dòng trước kết quả
- * -C5: In ra 5 dòng trước và sau kết quả

Lệnh *grep* có thể sử dụng kết hợp với các lệnh khác: *ls*, *cat*, *echo*, ...

Ví dụ:

- *student@linux ~ \$ grep 'nt' /etc/group* (in ra các dòng chứa chuỗi *'nt'* trong tệp tin */etc/group*)
- *student@linux ~ \$ ls /etc | grep a\$* (in ra các tệp tin/thư mục nằm trong thư mục */etc* có tên kết thúc bằng chữ cái *'a'*)
- *student@linux ~ \$ ls * | grep -E '[i|a]+'* (in ra các tệp tin/thư mục nằm trong thư mục hiện tại có tên thỏa mãn: chữ cái *'i'* hoặc *'a'* xuất hiện ít nhất 1 lần)
- *student@linux ~ \$ cat /etc/passwd | grep '\bover\b'* (in ra các dòng có chứa từ *'over'* trong tệp tin */etc/passwd*)
- *student@linux ~ \$ echo 'There is a book' | grep -E '[her]+'*

Phần 2: Bài tập thực hành

Viết các biểu thức chính quy biểu diễn:

- Tập các số tự nhiên
- Tập các số thực
- Tập các số tự nhiên chia hết cho 5
- Tập các số tự nhiên chia hết cho 5 có độ dài chẵn
- Tập các dòng bắt đầu bằng chữ cái *'T'* và chứa ít nhất 2 chuỗi *"this is an exercise"*

Kết hợp *ls* với *grep* để thực hiện:

- Tìm kiếm các tệp tin/thư mục nằm trong thư mục */etc* có chứa ít nhất 2 chữ cái *'a'*. Lưu kết quả ra tệp *output1.txt*
- Tìm kiếm các tệp tin/thư mục nằm trong thư mục */etc* bắt đầu bằng chữ cái *'b'* và không chứa chữ cái *'c'*. Lưu kết quả ra tệp *output2.txt*

Sử dụng trình soạn thảo *vi* để soạn thảo văn bản có nội dung trong tệp tin *vd01.pdf*. Sau đó tiến hành thực hiện các nội dung sau:

- Xóa đi các dòng trống trong văn bản
- Sao chép đoạn đầu tiên và dán vào cuối văn bản
- Chèn thêm 1 dòng ở đầu văn bản có nội dung: *"Bài thực hành VI"*
- Lưu lại văn bản đã sửa

Sử dụng trình soạn thảo *emacs* để soạn thảo một đoạn chương trình java có nội dung trong tệp tin *vd02.pdf*. Áp dụng biểu thức chính quy, thay thế tất cả các xâu chú thích *// comment* bằng */* comment */*

Sử dụng trình soạn thảo *emacs* soạn thảo đoạn văn bản *html* có nội dung trong tệp tin *vd03.pdf*. Áp dụng biểu thức chính quy, thay thế tất cả các thẻ *<meta content />* bằng thẻ *<meta content > </meta>*

Phần 3: Liên lạc

STT	Họ và tên	Email	ĐT
1	Hà Mỹ Linh	halinh.hus@gmail.com	