Bài 5: Biểu thức chính quy

Hà Mỹ Linh

Ngày 17 tháng 9 năm 2019

Nội dung

Giới thiệu

Định nghĩa

Biểu thức chính quy (*Regular Expressions*) là cú pháp cho phép mô tả các chuỗi ký tự.

Chức năng giống Wildcards nhưng có hiệu quả hơn rất nhiều.
Trong Linux, sử dụng 3 phiên bản

- BRE (*Basic Regular Expressions*)
- ERE (*Extended Regular Expressions*)
- PRCE (*Perl Regular Expressions*).

Các quy tắc chính

Các quy tắc của biểu thức chính quy:

- Sử dụng các kí tự điều khiển $. \wedge \$ [] - \{\} ? * + () | \backslash$
- Để biểu diễn kí tự điều khiển như kí tự thường, thêm $\backslash$ vào đằng trước
- $.$ biểu diễn kí tự bất kì, $\wedge$ và $\$$ biểu diễn đầu dòng và cuối dòng
- $\backslash b$ biểu diễn đầu và cuối từ, $\backslash s$ chỉ dấu cách hoặc tab
- $[characters]$ biểu diễn kí tự bất kì được liệt kê, $\wedge$ đặt bên trong mang nghĩa là khác
- Phép lặp: $? * + \{n\} \{n, m\} \{, m\} \{n, \}$
- Phép lấy tích ghép, phép hợp $|$
- () dùng để nhóm các chuỗi con.

Ví dụ: trò chơi tìm từ

Bài tập

Tìm từ tiếng Anh bắt đầu bởi chữ cái t và kết thúc bởi *.txt*.

Gợi ý:

- Trong Linux có một tệp lưu lại các từ tiếng Anh ở */usr/share/dict/words*
- Sử dụng lệnh **grep** để tìm.

$[A-Z]$ VS POSIX class

Vấn đề

Để liệt kê các chữ in hoa, thông thường ta dùng $[A-Z]$, tuy vậy rất nhiều trường hợp, cách liệt kê này không cho ra kết quả đúng.

Nguyên nhân:

- Ban đầu các chữ cái được liệt kê theo kiểu mã ASCII: *ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghi-jklmnopqrstuvwxyz*
- Sau khi bảng mã được mở rộng, các chữ cái được liệt kê theo thứ tự từ điển *AaBbCcDdEeFfGgHhIiJjKkLlMmN-nOoPpQqRrSsTtUuVvWwXxYyZz*.

$[A-Z]$ VS POSIX class

Các cách khắc phục:

- Thay vì sử dụng dấu – để liệt kê, ta sử dụng các lớp POSIX, ví dụ [:upper:] (xem ở phần Wildcards)
- Khai báo lại biến $LANG như sau

  [user@machine] $ LANG=POSIX

  Để kiểm tra giá trị của $LANG ta có thể làm như sau

  [user@machine] $ **echo** $LANG

Tra ngược lại (*backreference*)

- Tìm hiểu lệnh sau

```
$grep "\([a-z]\)\([a-z]\)\([a-z]\)\2\1" word.txt
```

- Mỗi một mẫu được mở đầu bằng \(và được đánh dấu phạm vi trong cặp \(và \)
- Có thể nhớ đến 9 mẫu

Sự khác nhau giữa phiên bản BRE và ERE

	BRE	ERE
Ktđk	. ∧ $ []*	nhận thêm (){}?+ |
Chèn \	(){}?+ | thành ktđk	ktđk thành kí tự thường
Ví dụ	**echo** "AB" | **grep** A\{1\}	**echo** "AB" | **grep** -E A{1}

Giới thiệu grep

grep (*global regular expression print*) là công cụ để tìm và in ra những dòng chứa chuỗi khớp với biểu thức chính quy.

Cú pháp | **grep** [options] regex [file] |

Các option thường dùng

- *-G*, *-E*, *-P* khai báo phiên bản BRE, ERE, PRCE
- *-i --ignore-case* không phân biệt in hoa in thường
- *-v --invert-match* in ra những dòng không chứa chuỗi khớp
- *-c --count* in ra số dòng thỏa mãn
- tìm hiểu thêm về *-l*, *-L*, *-n* và *-h*.

Giới thiệu grep

grep (*global regular expression print*) là công cụ để tìm và in ra những dòng chứa chuỗi khớp với biểu thức chính quy.

Cú pháp | **grep** [options] regex [file] |

Các option thường dùng

- *-G*, *-E*, *-P* khai báo phiên bản BRE, ERE, PRCE
- *-i --ignore-case* không phân biệt in hoa in thường
- *-v --invert-match* in ra những dòng không chứa chuỗi khớp
- *-c --count* in ra số dòng thỏa mãn
- *-e*: in ra số lần trùng khớp trong văn bản
- *-o*: xuất ra chỉ những phần văn bản trùng khớp trong tập tin, thay cho việc phải xuất ra cả 1 dòng
- *-b*: in ra vị trí mẫu trùng khớp trong văn bản
- tìm hiểu thêm về *-l*, *-L*, *-n* và *-h*.

Bài tập

Bài 1. Lệnh **egrep** và **zgrep** có gì khác so với **grep**.

Bài 2. Tìm kiếm các tệp nén có trong */bin/* với lệnh **find** và **locate**.

Bài 3. Tạo tệp tin *phonenumber.txt* mà mỗi dòng là một số điện thoại di động, viết biểu thức chính quy để tìm các số điện thoại của Viettel trong danh sách đó.

Bài 4. Tạo thư mục *play* trong thư mục người dùng, tạo các tệp *f1.txt*, *f2.txt*, *text.dov*, *txt.txt*, sau đó dùng lệnh **rename** đổi đuôi *.txt* thành *.text*

Bài 5. Tạo tệp *dates.txt* trong đó mỗi dòng là một thứ trong tuần (*Thu Hai*, *Thu Ba*, ..., *Thu Bay*) dùng lệnh **sed** để thay từ *Thu* thành *Cac thu*.

Dấu backslash

- Dùng để ngắt dòng lệnh thành nhiều dòng
- Dùng để biến kí tự đặc biệt thành kí tự thông thường

Dấu nháy trong Linux

- Có 3 loại dấu nháy
  - dấu nháy đơn (mạnh) (*single quote*) ’: những gì nằm trong dấu nháy đơn có ý nghĩa không đổi
  - dấu nháy kép (yếu) (*double quote*) "": bất cứ gì nằm trong dấy nháy kép được xem là những ký tự riêng biệt
  - dấu nháy lùi (*back quote*) ‘: thực thi lệnh
- Để in ra một chuỗi ký tự với các ký tự đặc biệt, ta dùng nháy đơn

  $**echo** ’the characters $ # * ? are special characters!’
- Làm việc với tên tệp chứa khoảng trắng hoặc kí tự đặc biệt ta dùng nháy đơn
- Dấu nháy kép có công dụng giống nháy đơn nhưng công nhận một số ký tự đặc biệt như $ * hoặc ? nên bị coi là "yếu" hơn $**echo** "Your home directory is $HOME"

Dấu nháy (2)

- Dấu nháy giống nhau lồng nhau

```
$echo "the double quotes are \" \" "
```

- Dấu nháy khác nhau lồng nhau

```
$echo "Don't quote me"
```

Lệnh sed (stream editor)

- Lệnh thay thế chuỗi kí tự, lọc văn bản
- Cú pháp chính:
  sed 's/pattern/replace_string/' file
  - pattern: Một chuỗi hoặc một biểu thức chính quy
  - /g: thêm tham số g vào sau sẽ thay thế các xuất hiện của mẫu cho tới cuối văn bản
  - /Ng: Thay thế từ xuất hiện thứ N tới cuối văn bản
  - /N: chỉ thay thế xuất hiện thứ N

Lệnh sed

- Ví dụ thay *day* thành *night* trong tệp word1.txt thành trong tệp word2.txt ta làm như sau

```
$sed 's/day/night/' word1.txt > word2.txt
```