



BACKEND DEVELOPERS WITH SKILLS IN **LARAVEL** **NODEJS**
 FRONTEND DEVELOPERS WITH SKILLS IN **VUEJS** **SASS**
 SMART PHONE DEVELOPERS WITH SKILLS IN **REACT-NATIVE**



VIBLO

Search Viblo



Nguyen Thanh Tung B

Follow

Published May 15th, 12:10 PM

Learn sed

Editors' Choice [tuts](#) [stream editor](#) [Terminal](#) [Unix](#) [Sed](#) [command](#)

May 15th, 12:10 PM

100

5

0

Add to my series

Report

Introduction

Sed 1 stream editor có tác dụng biến đổi text từ 1 input stream (1 file hoặc đầu vào của 1 pipeline - lấy từ kết quả đầu ra của 1 chuỗi câu lệnh chẳng hạn). Sed thường được trang bị sẵn trên những hệ điều hành nhân Unix. Bạn có thể dùng sed trên Ubuntu, MacOS mà không phải cài đặt thêm gì. Cá nhân mình gần đây có nhu cầu dùng sed để biến đổi đầu ra của 1 số câu lệnh, sửa, thay thế text để thu được thông tin mình mong muốn. Bài viết này sẽ nói về 1 số cách sử dụng câu lệnh sed trên MacOS (do phiên bản sed trên MacOS (BSD version) và trên các hệ điều hành nhân Unix (GNU .) khác có 1 số khác biệt)

Overview

Cú pháp thông thường của 1 lệnh sed là

```
sed [option] [command] [file]
```

Trong trường hợp bạn muốn thực hiện edit đối với standard input thay vì file thì không cần phải để tham số `[file]` nữa.

Một số option phổ dụng

-i

Thực hiện việc edit text inplace, trực tiếp vào input stream thay vì chỉ in kết quả sau khi edit text ra standard output. Điều này có nghĩa là bạn có thể trực tiếp thay đổi nội dung file thay vì chỉ in kết quả sau khi edit text ra màn hình console.

Ví dụ ta có 1 file `test.txt` với nội dung

```
hello
world
```

Nếu bạn muốn thay đổi chữ `l` thường thành `L` hoa thì có thể dùng lệnh sau. (cú pháp `s/original_text/substitute_text/g`) chuyên dùng để thay thế text.

```
$ sed "s/l/L/g" test.txt
```

Kết quả sẽ là đoạn output dưới được in ra màn hình

```
heLlO
worLd
```

Tuy nhiên nội dung file `test.txt` không thay đổi. Trong trường hợp bạn muốn thay đổi nội dung trong file đó đồng thời muốn giữ lại nội dung file cũ vào 1 file khác để đề phòng bất trắc thì có thể làm như sau



```
$ sed -i .bak "s/l/L/g" test.txt
```

Khi đó nội dung file `test.txt` sẽ bị thay đổi, các chữ `l` sẽ bị đổi thành `L`, còn nội dung cũ sẽ được lưu lại trong 1 file backup tên là `test.txt.bak`. Nếu bạn tự tin vào câu lệnh biến đổi text của mình thì có thể không cần file backup bằng cách chỉ định tham số của option `-i` là 1 string rỗng

```
$ sed -i "" "s/l/L/g" test.txt
```

-e

Bổ sung thêm command cho list command.

Bạn có thể thực hiện nhiều phép thay thế text trên 1 câu lệnh sed như ví dụ sau:

```
$ echo "abcdefgh" | sed -e "s/b/B/g" -e "s/f/F/g"
aBcdeFgh
```

-f

Thay vì viết một chuỗi các command với option `-e` bạn có thể lưu các command này vào 1 file và sử dụng option `-f` để load các command từ file đó ra

Ví dụ bạn có thể lưu 2 command sed trên vào 1 `test.sed` với nội dung như sau

```
s/b/B/g
s/f/F/g
```

rồi dùng câu lệnh với option `-f` để cho ra kết quả tương tự

```
$ echo "abcdefgh" | sed -f test.sed
```

-E

Cho phép sử dụng biểu thức regex mở rộng thay vì biểu thức regex cơ bản. Đây là 1 option mà mình thường xuyên sử dụng vì nó cho phép thực hiện các phép thay thế text phức tạp. Phiên bản regex mở rộng cho phép sử dụng các meta-characters `?`, `+`, `{`, `|`, `(`, `,`, `)` và khi muốn thực hiện matching với các meta-characters này phải thêm backslashed `\`.

```
# Basic regex
$ echo "01234{5}6789" | sed "s/[0-9]{5}/P/g"
0123P6789
# Extended regex
$ echo "01234{5}6789" | sed -E "s/[0-9]{5}/P/g"
P{5}6789
```

Nếu bạn muốn kết quả trả về giống với phiên bản dùng basic regex khi dùng option `-E` hãy thêm backslashed `\` vào trước `{` cũng như `}` trong phần command của câu lệnh `sed`

```
$ echo "01234{5}6789" | sed -E "s/[0-9]\{5\}/P/g"
0123P6789
```

-n

Loại bỏ action mặc định in kết quả ra màn hình. Nếu trong phần command không có các giá trị đặc biệt như `p` thì sau khi bạn thực hiện xong câu lệnh `sed` sẽ không thấy gì được in ra màn hình console. Option này hữu ích trong việc tùy chỉnh in ra những dòng mà bạn mong muốn.

```
$ printf "12\n34\n56\nHERE\n78\n90\n" | sed -n '4p'
HERE
```

Sed script command

Đây là phần cốt lõi của lệnh `sed`. Dựa theo các chỉ thị trong phần command này mà lệnh `sed` thực hiện các thao tác tương ứng.

Cấu trúc của 1 `sed script command` như sau

```
[addr]X[options]
```

`[addr]` là phần địa chỉ đánh dấu phạm vi mà câu lệnh `sed` tác động tới. Phần `[addr]` có thể là 1 dòng, hoặc 1 phạm vi bao gồm nhiều dòng hoặc biểu thức regex. `x` là câu lệnh sẽ được thực hiện trên phạm vi chỉ định bởi `[addr]` `[options]` được sử dụng trong 1 số trường hợp. Bạn có thể thực hiện nhiều sed script command bằng cách liên kết chúng qua dấu `;` Ví dụ:

```
# delete content from line 2 to line 4
$ printf "12\n34\n56\n78\n90\n" | sed "2,4d"
12
90

# print content until found string "HERE"
$ printf "12\n34\n56\nHERE\n78\n90\n" | sed "/^HERE/q"
12
34
56
HERE
# print
```

Dưới đây chúng ta sẽ dạo qua 1 lượt các command chính được dùng trong `sed`

Often-Used Commands

`a\` append

Thêm 1 vào sau dòng

```
$ printf "12\n34\n56\n" | sed 'a\
Check
'

12
Check
34
Check
56
Check

# or another way

$ printf "12\n34\n56\n" | sed 'a\ '$'\n''Check'$'\n'
12
Check
34
Check
56
Check
```

Ngay đằng sau `a\` yêu cầu phải có kí tự xuống dòng, đó lại lí do tại sao ta phải có `$'\n'`. Ngoài ra để thêm 1 dòng thì phần text thêm vào cũng cần có kí tự xuống dòng nên đó là lí do phải có đoạn `$'\n'` thứ 2. Nếu không có đoạn `$'\n'` thứ 2 thì kết quả sẽ như sau.

```
$ printf "12\n34\n56\n" | sed 'a\ '$'\n''Check'
→
12
Check34
Check56
```

c\ change

Thay đổi text của 1 hay nhiều dòng thành 1 text khác

```
$ printf "12\n34\n56\n78\n90\n" | sed '3,4c\ '$'\n' 'Check'$'\n'
12
34
Check
90
```

d delete

Xoá dòng

```
$ printf "12\n34\n56\n78\n90\n" | sed '3,4d'
12
34
90
```

n no-action

Nếu chế độ auto-print chưa bị disable thì in dòng đó ra và chuyển tới dòng kế tiếp

```
$ printf "12\n34\n56\n78\n90\n12\n" | sed 'n;n;s/[0-9]*/HI/g'
12
34
HI
78
90
HI
```

Trong ví dụ này chúng ta chỉ thực hiện lệnh thay thế cho các dòng là bội số của 3 thông qua việc giữ nguyên các dòng còn lại bằng cách dùng 2 lần command **n**

{ }

Group các command tác động lên dòng. Nó hữu dụng khi bạn muốn thực hiện 1 nhóm các command lên 1 dòng hoặc 1 phạm vi dòng nhất định thay vì toàn bộ các dòng.

```
$ printf "12\n34\n56\n78\n90\n" | sed '3,4{s/[0-9]/H/;s/[0-9]//}'
12
34
HI
HI
90
```

s substitute

Thay thế 1 đoạn text bằng 1 đoạn text khác. Đây có thể nói là command hữu dụng nhất của sed. Trong các ví dụ trên mình cũng thường xuyên dùng command này rồi 😊

```
$ printf "12\n34\n56\n78\n90\n" | sed 's/[0-9]/H/;'
H2
H4
H6
H8
H0
```

Address

Chỉ định 1 dòng

```
$ printf "1\n2\n3\n4\n5\n" | sed "4d"
1
2
3
5
```

Chỉ định bằng 1 phạm vi dòng

```
$ printf "1\n2\n3\n4\n5\n" | sed "2,4d"
1
5
```

Chỉ định bằng regex

Ví dụ: thay thế text **foo** thành **baz** với dòng có chứa text **hello**

```
$ printf "foo\nfont\nhello foo\nhello\n" | sed "/hello/ s/foo/baz/"
foo
font
hello baz
hello
```

Chỉ định bằng cách phủ định lại 1 address

Ví dụ: thay thế text **foo** thành **baz** với dòng KHÔNG chứa text **hello**

```
$ printf "foo\nfont\nhello foo\nhello\n" | sed "/hello/! s/foo/baz/"
baz
font
hello foo
hello
```

Have problems with [tuts](#), [stream editor](#) or [Terminal](#)? [Ask on Viblo »](#)

Related

[Learn sed \(part 2\).](#)

[Nguyen Thanh Tung B](#)

👁 28 🗒 0 💬 0 ⬆ 1

[Ag - The Silver Searcher](#)

[Ho Hoang Lam](#)

👁 24 🗒 2 💬 0 ⬆ 6

More from Nguyen Thanh Tung B

[Ruby hijacking](#)

[Nguyen Thanh Tung B](#)

👁 24 🗒 0 💬 0 ⬆ 2

[Learn sed \(part 2\).](#)

[Nguyen Thanh Tung B](#)

👁 28 🗒 0 💬 0 ⬆ 1


[Threads, Processes, Parallel Programing in Ruby \(part...](#)

[Threads, Processes, Parallel Programing in Ruby \(part...](#)


Comments

Write

Preview



Write a response...



Post Comment

No comments, yet.

RESOURCES


- Posts


Questions
- Videos


Tags
- Authors


Tools

LINKS

- 

Facebook
- 

GitHub
- 

Browser extension
- 

Atom plugin

MOBILE APP

