XỬ LÝ VĂN BẢN

Phạm Nguyên Khang, Đỗ Thanh Nghị
pnkhang@cit.ctu.edu.vn

Nội dung

- Xem nội dung
- Sắp xếp
- Dịch, lựa chọn
- Tìm kiếm
- Cắt văn bản
- Khác
- sed
- awk

Xem nội dung

- cat — hiển thị nội dung 1 file
- head –n — hiển thị n dòng đầu tiên của 1 file
- tail –n — hiển thị n dòng cuối cùng của 1 file
- more — hiển thị cả file có phân trang
- less — tương tự more, nhưng cho phép quay lui, tìm kiếm
- wc — thống kê, đếm số ký tự, từ và hàng
  - -c — đếm số ký tự
  - -w — đếm số từ
  - -l — đếm số hàng

Sắp xếp

- sort [OPTIONS] [tập tin]
  - Sắp xếp nội dung tập tin theo 1 thứ tự nào đó
  - Nếu không chỉ rõ tập tin nào, sort sẽ sắp xếp luồng nhập chuẩn (sử dụng ống dẫn)
- OPTIONS
  - -n trường đang quan tâm là số (mặc định là chuỗi)
  - -r xếp theo thứ tự giảm dần
  - -tx x là ký tự phân cách (mặc định là ký tự TAB)
  - +bd bd là trường bắt đầu (kể từ 0)
  - -kt kt là trường kết thúc
- Ví dụ:
  - sort auto
  - sort -t: -n +4 -5 auto
  - sort -t: +0 -1 +3n -4 auto

Dịch

- **tr** [OPTIONS] tập_hợp_1 [tập_hợp_2] <file
  - Chuyển từng ký tự trong tập hợp 1 thành ký tự tương ứng trong tập hợp 2
  - Ví dụ: tr [a-z] [A-Z] </etc/passwd
- OPTIONS:
  - -d xóa bỏ ký tự nếu nó nằm trong tập hợp 1

Trích chọn

- cut [OPTIONS] file
  - Trích chọn 1 số trường trong file file
  - OPTIONS
    - -dx x là ký tự phân cách, mặc định là TAB
    - -fds ds là sách các trường cách nhau bằng dấu phẩy, tính từ 1
  - Ví dụ:
    - cut -d: -f1,5 /etc/passwd
    - cut -d: -f1-3 /etc/passwd
- uniq
  - Xóa bỏ những dòng trùng nhau, chỉ giữ lại 1

Tìm kiếm

- Việc tìm kiếm một từ hay nhiều từ trong một văn bản có thể được thực hiện bằng các dùng lệnh **grep**, **fgrep** hay **egrep**.
- Các từ khóa được dùng trong việc tìm kiếm được kết hợp từ các ký tự. Sự kết hợp này được gọi là *biểu thức chính quy*.
- Biểu thức chính quy cũng được dùng trong các ứng dụng khác như **sed** và **vi**.
- grep biểu_thức file

Biểu thức chính quy cơ bản

Ký tự	Ý nghĩa
x (hay bất cứ ký tự nào khác)	Chứa 1 ký tự x
\<KEY	Từ bắt đầu bằng KEY
WORD\>	Từ kết thúc bằng WORD
^	Bắt đầu dòng
$	Kết thúc dòng
[Range]	Một đoạn ký tự ASCII
[^c]	Không chứa ký tự c
\[	Ký tự [
cat*	Chứa ca hoặc cat hoặc catt, …
.	Bất kỳ ký tự nào

Biểu thức chính quy mở rộng

Ký tự	Ý nghĩa
A1|A2|A3	Chứa A1 hoặc A2 hoặc A3
cat+	Chứa cat hoặc catt, hoặc cattt, …
cat?	Chứa ca hoặc cat, hoặc catt, …

Họ lệnh grep

- grep cơ bản:
  - Sử dụng các *biểu thức chính quy cơ bản* để tìm kiếm
- egrep:
  - Sử dụng các *biểu thức chính quy mở rộng* để tìm kiếm
- fgrep (fast grep):
  - Không hỗ trợ biểu thức chính quy.

Làm việc với lệnh grep

- Cú pháp:
  - grep [option] PATTERN FILE

Grep	Các tùy chọn chính
-c	Đếm số dòng thỏa mãn mẫu PATTERN
-f	Mẫu tìm kiếm được lấy từ tập tin
-i	Không phân biệt chữ hoa chữ thường
-n	Hiển thị số thứ tự của dòng thỏa mãn mẫu PATTERN
-v	Hiển thi tất cả các dòng không thõa mãn mẫu
-w	Tìm chính xác mẫu

Grep

- Ví dụ:
  - grep -v “^$” /etc/passwd
  - Hiển thị tất cả các dòng không rỗng của tập tin /etc/passwd

egrep và fgrep

- Sử dụng tương tự như grep
- egrep sử dụng biểu thức chính quy mở rộng
- Ví dụ: egrep "linux|^image" FILE

  Tìm các dòng có chứa từ linux hoặc bắt đầu bằng image
- fgrep không hỗ trợ biểu thức chính quy
- Ví dụ: fgrep "cat*" FILE

  Tìm các dòng có chứa chuỗi cat*

Tách file

- split -n file
  - Tách file file thành nhiều file con, mỗi file con có n dòng
  - Tên file con được đặt tên từ xaa đến xaz

Các lệnh khác

- cmp, diff
  - So sánh 2 file
- paste: nối từng hàng của 2 file lại với nhau
- join: nối từng hàng của 2 file theo 1 trường nào đó
  - join -1 FIELD -2 FILED FILE1 FILE2
- tee:
  - Copy đầu ra của lệnh trước xuống file, và chuyển đầu ra thành đầu vào của lệnh sau

sed

- Trình soạn thảo luồng (**s**tream-oriented **ed**itor)
- Thông dịch lệnh và thực thi lệnh
- Kết quả in ra dòng xuất chuẩn (màn hình)
- Cú pháp
  - **sed** [options] 'lệnh' FILE
  - **sed** [options] **-f** script FILE
  - Trong đó:
    - Một lệnh có dạng: [địa chỉ][,địa chỉ][!]lệnh[tham số]
    - script là file chứa lệnh
- Ví dụ:
  - **sed 's/xx/yy/g'** FILE
    - thay thế tất cả các chuỗi xx thành yy
  - sed '/BSD/d' FILE
    - xóa tất cả các dòng có chứa từ BSD
  - sed -n '/^BEGIN/,/^END/p' FILE
    - In các dòng nằm giữa BEGIN và END
  - sed '/SAVE/!d' FILE
    - Xóa tất cả các dòng không chứa từ SAVE
  - sed '/^BEGIN/,/^END/s/xx/yy/g' FILE
    - Thay thế xx thành yy trong khoảng từ BEGIN đến END

sed

- Địa chỉ có thể là:
  - Số dòng, ví dụ: 3 (dòng số 3)
  - Mẫu: đặt trong cặp //, ví dụ: /BEGIN/
- **!** đặt sau địa chỉ có nghĩa là trừ phần địa chỉ ra
- Lệnh
  - s/mau/thaythe/g thay thế **mau** thành **thaythe**
    - Nếu không có g, chỉ thay thế một lần cho 1 dòng
    - Có thể sử dụng _ hoặc : để thay thế cho /
    - Ví dụ: **s_mau_thaythe_g** hoặc **s:mau:thaythe:g**
  - d xóa
  - p in (sử dụng với option -n)

sed

- sed '/^$/d' FILE
  - Xóa tất cả các dòng trống
- sed 's/.*/cp & &.copied/' FILE
  - Tạo danh sách các lệnh copy để copy các file
  - & = mẫu so khớp
  - Có thể sử dụng cặp \(\) để nhóm các mẫu, để tham chiếu đến các nhóm này ta dùng **số thứ tự nhóm**
  - Ví dụ: Nếu FILE chứa
    - abc.jpg
    - 123.png
  - sed 's/\(.*\)\.jpg/convert \1.jpg \1.gif:g' FILE

sed

- s/mau/thaythe/flags
  - Ngoài cờ g, lệnh s còn thể được sử dụng với
    - Số nguyên n (vd 2) lệnh s sẽ được thực hiện với mẫu thứ n
    - Số nguyên và g (vd: 2g) thực hiện lệnh s từ mẫu thứ 2 trở đi
    - p in kết quả sau khi thay thế (nên sử dụng
    - với option -n)
    - w tênfile ghi kết quả ra file tênfile
  - Có thể kết hợp nhiều cờ lại với nhau nếu có ý nghĩa
    - Ví du: sed -n 's/a/A/2pw /tmp/file' FILE

sed

- Sử dụng nhiều lệnh
  - sed -e 'lệnh 1' -e 'lệnh 2' … FILE
  - Thay vì
    - sed 's/a/A/' FILE | sed 's/b/B/'
  - Ta có thể sử dụng
    - sed -e 's/a/A/' -e 's/b/B/' FILE
- Có thể nhóm các lệnh bằng cặp dấu ngoặc {}

```
sed -n ' /begin/,/end/ {        s/#.*//
                                s/[ ^I]*$//
                                /^$/ d
                                p
} '
```

- Chú ý: ^I = ký tự TAB
- Xem thêm: http://www.grymoire.com/Unix/Sed.html

awk

- Giả sử ta có file coins.txt như sau:

```
gold     1    1986  USA                American Eagle
gold     1    1908  Austria-Hungary    Franz Josef 100 Korona
silver   10   1981  USA                ingot
gold     1    1984  Switzerland        ingot
gold     1    1979  RSA                Krugerrand
gold     0.5  1981  RSA                Krugerrand
gold     0.1  1986  PRC                Panda
silver   1    1986  USA                Liberty dollar
gold     0.25 1986  USA                Liberty 5-dollar piece
silver   0.5  1986  USA                Liberty 50-cent piece
silver   1    1987  USA                Constitution dollar
gold     0.25 1987  USA                Constitution 5-dollar piece
gold     1    1988  Canada             Maple Leaf
```

- awk '/gold/' coins.txt
  - Liệt kê các dòng có chứa từ gold

awk

- awk '/gold/ {print $5,$6,$7,$8}' coins.txt
  - Tìm những dòng chứa từ gold và in các trường 5, 6, 7, 8
  - Kết quả:
    - American Eagle
    - Franz Josef 100 Korona
    - ingot
    - Krugerrand
    - Krugerrand
    - Panda
    - Liberty 5-dollar piece
    - Constitution 5-dollar piece
    - Maple Leaf
- Cú pháp của awk là
  - awk ‘<mẫu> {lệnh}’ FILE
  - Nếu không có <mẫu> thì phạm vi áp dụng là cả file

awk

- Lệnh print
  - In các tham số ra màn hình
  - Nếu không có tham số sẽ in toàn bộ dòng hiện hành ra màn hình, do đó
    - awk '/gold/' = awk '/gold/{print}' = awk '/gold/ {print $0}'
- $n: trường thứ n (mặc định các trường cách nhau bằng khoảng trắng, ta có thể sử dụng option -Fx để chỉ định x là ký tự phân cách)
- $0: toàn bộ dòng
- Ta có thể làm nhiều thứ phức tạp hơn nữa với awk
  - awk 'if ($3 < 1980) print $3, " ", $5, $6, $7, $8}' coins.txt
  - Kết quả:
    - 1908 Franz Josef 100 Korona
    - 1979 Krugerrand

awk

- In tổng số loại tiền đang có
  - awk 'END {print NR,"coins"}' coins.txt
  - Kết quả: **13 coins**
- Từ khóa END cho biết phần trong cặp {} chỉ thực hiện 1 lần sau khi duyệt qua hết file
- Tương tự như thế từ khóa BEGIN cho biết phần trong cặp {} chỉ thực hiện 1 lần khi bắt đầu duyệt file
- NR: tổng số dòng của file coins.txt
- NF: tổng số trường của dòng hiện hành
- Cú pháp tổng quát của awk là:

```
awk 'BEGIN {lệnh bd}
    <mẫu 1> {lệnh 1}
    <mẫu 2> {lệnh 2}
...
END {lệnh kt}'
```

awk

- Ví dụ:
  - awk '/gold/ {ounces += $2} END {print "value = $" 425*ounces}' coins.txt
- Có thể lưu các lệnh của awk trong file cmd.awk và gọi
  - awk –f cmd.awk

awk

- Ví dụ:

```
/gold/    { num_gold++; wt_gold += $2 }      # Get weight of gold.
/silver/  { num_silver++; wt_silver += $2 }  # Get weight of silver.
END { val_gold = 485 * wt_gold;              # Compute value of gold.
      val_silver = 16 * wt_silver;           # Compute value of silver.
      total = val_gold + val_silver;
      print "Summary data for coin collection:";  # Print results.
      printf ("\n");
      printf ("   Gold pieces:                %2d\n", num_gold);
      printf ("   Weight of gold pieces:      %5.2f\n", wt_gold);
      printf ("   Value of gold pieces:      %7.2f\n",val_gold);
      printf ("\n");
      printf ("   Silver pieces:              %2d\n", num_silver);
      printf ("   Weight of silver pieces:    %5.2f\n", wt_silver);
      printf ("   Value of silver pieces:    %7.2f\n",val_silver);
      printf ("\n");
      printf ("   Total number of pieces:     %2d\n", NR);
      printf ("   Value of collection:       %7.2f\n", total); }
```

awk

Summary data for coin collection:

```
Gold pieces:                9
Weight of gold pieces:      6.10
Value of gold pieces:       2958.50

Silver pieces:              4
Weight of silver pieces:    12.50
Value of silver pieces:     200.00

Total number of pieces:     13
Value of collection:        3158.50
```

awk

- Lệnh printf
  - Tương tự như printf của C
  - printf("<format_code>", <parameters>)
- Awk nâng cao
  - Cú pháp đầy đủ của awk:
    - awk [-F<ch>] {pgm} | { -f <pgm_file> } [<vars>] [- | <data_file>]
    - ch — ký tự phân cách trường (mặc định là khoảng trắng)
    - pgm — lệnh (chương trình awk)
    - pgm_file — file chứa lệnh
    - Vars — danh sách các biến cần khởi tạo
    - - — sử dụng dòng nhập làm đầu vào
    - data_file — tên file đầu vào

awk

- Ví dụ trong file **sumary.awk** có đoạn

```
END { val_gold   = pg * wt_gold
   val_silver = ps * wt_silver
   ...
}
```

- Khi gọi awk ta có thể truyền 2 biến pg và ps
  - awk -f summary.awk pg=485 ps=16 coins.txt
  - Chú ý: không có khoảng trắng trước vào sau dấu =
- Mẫu:
  - Sử dụng /biểu thức chính quy/
  - Ví dụ /^The/: dòng bắt đầu bằng từ The

awk

- Mẫu (tt)
  - $1 ~ /^France$/ dòng có trường đầu tiên là France
  - $1 !~ /^Norwey$/ dòng có trường đầu tiên không phải là Norwey
  - /^Ireland/,/^Summary/ tất cả các dòng từ dòng bắt đầu bằng Ireland cho đến dòng bắt đầu bằng Summary
  - NR == 10 dòng thứ 10
  - NR == 10, NR==20 từ dòng thứ 10 đến dòng 20 (11 dòng)
  - Có thể sử dụng !=, <, <=, >, >=
    - NF == 0 dòng trống (không có trường nào)
  - Có thể sử dụng &&, || và các dấu ngoặc
    - ((NR >= 30) && ($1 == "France")) || ($1 == "Norway")

awk

- Biến
  - var = 1776 và var = "1776" là như nhau
  - var = "something" và var = something là khác nhau
  - Với var = something, (var == 0) luôn trả về true
- Biến dựng sẵn
  - NR: dòng hiện hành, khi đến cuối file NR là tổng số dòng
  - NF: số trường của dòng hiện hành
  - $1, $2, …, $NF là các trường từ 1 đến NF
  - $0: cả dòng
  - FILENAME: tên file
  - FS: ký tự ngăn cách các trường
  - RS: ký tự ngăn cách dòng (mặc định là newline)
  - OFS: ký tự ngăn cách trường của kết quả (mặc định là khoảng trắng
  - ORS: ký tự ngăn cách dòng của kết quả
- Có thể gán giá trị cho biến trường ví dụ
  - $2 = "toto"

awk

- Mảng (array)
  - Mảng chỉ số
    - ar[1], ar[2]
    - Chỉ số của phần tử đầu tiền là 1
  - Bảng băm
    - marks[“Kim”], prices[“gold”]
  - Duyệt mảng

    for (var in array)

    làm gì đó trên array[var]
  - Xóa phần tử
    - delete array[chỉ số]
  - Xóa cả mảng
    - delete array

awk

- Các phép toán
- Cấu trúc if, for, while
- Tương tự như C
  - if (<condition>) <action 1> [else <action 2>]
  - while (<condition>) <action>
- Xem thêm: http://www.vectorsite.net/tsawk.html