



## Bài 5: Giới thiệu về lệnh đường ống, biểu thức chính quy

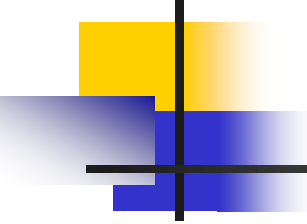
---



# Nội dung

---

1. Lệnh đường ống (pipe command)
2. Lọc dữ liệu (trích xuất dữ liệu) với lệnh grep, cut
3. Biểu thức chính quy
4. Sự kết hợp giữa biểu thức và grep
5. Tìm kiếm và thay thế với lệnh sed



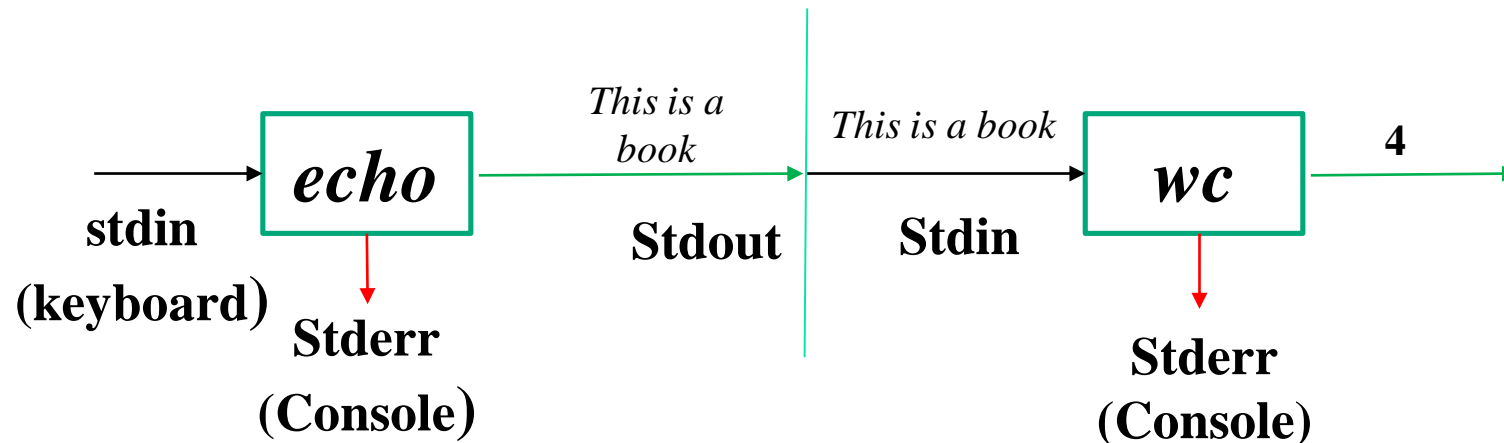
# Lệnh đường ống (pipe command)

---

- Đường ống (pipe) dùng để chuyển hướng dữ liệu trong Linux, cho phép người dùng sử dụng hai hoặc nhiều lệnh sao cho đầu ra của một lệnh đóng vai trò đầu vào trực tiếp của lệnh tiếp theo.
- Ưu điểm: kết nối trực tiếp các lệnh, dữ liệu được truyền liên tục mà không cần chuyển qua các tệp văn bản tạm thời hoặc màn hình hiển thị
- Các đường ống chỉ thực hiện 1 chiều tức là luồng dữ liệu chuyển hướng từ trái sang phải qua đường ống.

# Lệnh đường ống (pipe command)

- Một đường ống được biểu diễn bởi ký tự “|”
- Các đường ống giúp kết hợp nhiều lệnh cùng lúc và chạy liên tục với *cú pháp*: **command\_1 | command\_2|...|command\_n**
  - Ví dụ: *echo “This is a book” | wc -w*





## Lọc dữ liệu (1)

---

Để tìm kiếm các thông tin khớp với mẫu cần tìm người ta dùng các công cụ lọc dữ liệu: **grep** và **cut**

- Lọc dữ liệu theo hàng: sử dụng lệnh **grep**

**cú pháp: grep [options] <patterns> <file>**

- Ví dụ: *grep huong /etc/passwd* hoặc *grep -n huong /etc/passwd*
- Tùy chọn: *-n*: hiển thị số dòng, *-i*: dữ liệu lọc không phân biệt chữ hoa, chữ thường hoặc *-ni*



## Lọc dữ liệu (2)

---

- Lọc dữ liệu theo cột: sử dụng lệnh **cut**  
**cú pháp: cut -d 'delimiter' -f (field number) <file>**  
trong đó delimiter là dấu phân cách giữa các trường dữ liệu như:  
dấu cách (*space*), dấu “;” hay dấu “:”
  - Ví dụ: *cat matrix.txt*  
*cut -d “ ” -f1 matrix.txt* (trả về cột thứ nhất của matrix)

```
matrix.txt
1 2 3 4 5
6 7 8 9 10
11 12 13 14
```



## Lọc dữ liệu (3)

---

- **Câu hỏi:**

- Hiển thị tên của tất cả các người dùng có khả năng chạy bash shell trên Linux?
- Đếm số lượng người dùng có thể chạy bash shell?



# Một số bộ lọc thường dùng

| Tên               | Ý nghĩa                            |
|-------------------|------------------------------------|
| <b>sort (-n)</b>  | Sắp xếp dữ liệu                    |
| <b>head -n</b>    | In ra n dòng dữ liệu từ trên xuống |
| <b>tail -n</b>    | In ra n dòng dữ liệu từ dưới lên   |
| <b>wc -w (-l)</b> | Đếm từ, dòng                       |
| <b>uniq</b>       | Loại bỏ các dòng trùng nhau        |





# Biểu thức chính quy

---

- 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 có 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 ký tự điều khiển . ^ \$ [] – { } ? \* ( ) | \
- Để biểu diễn ký tự điều khiển như ký tự thường, người ta thêm \ vào đằng trước ký tự đặc biệt
- . Biểu diễn ký tự bất kỳ
- ^ và \$ biểu diễn bắt đầu dòng và kết thúc dòng
- \b biểu diễn đầu và cuối từ
- \s chỉ dấu cách hoặc tab



# Các quy tắc chính

---

- [characters] biểu diễn ký tự bất kỳ được liệt kê trong [],
- ^ đặt bên trong dấu [] mang nghĩa là **khác**
- Phép lặp: ? \* {n}, {n, m}, {., m} {n,}
- Phép lấy tích ghép, phép hợp |
- () dùng để nhóm các chuỗi con



# Các quy tắc chính

---

- Trong Linux người ta dùng lệnh **grep** để tìm các chuỗi khớp với biểu thức chính quy
- Ví dụ: Tìm tất cả các tệp tin trong thư mục hiện hành kết thúc bởi chữ cái “a:

```
ls * | grep a$
```



## Sự khác nhau giữa phiên bản BRE và ERE

|        | BRE                            | ERE                             |
|--------|--------------------------------|---------------------------------|
| Ktdk   | . ^ \$ [] *                    | nhận thêm () {} ? +             |
| Chèn \ | () {} ? +   thành ktdk         | ktdk thành kí tự thường         |
| Ví dụ  | <b>echo "AB"   grep A\{1\}</b> | <b>echo "AB"   grep -E A{1}</b> |



# Giới thiệu grep

---

- Cú pháp: `grep [options] regex [file]`
- Các option thường dung
- `-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

Bài tập: Tìm hiểu thêm về `-l`, `-L`, `-n`, `-h`



# Dấu backslash \

---

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



# Dấu nháy trong Linux

---

- ❖ Thường sử dụng hai loại dấu nháy
  - Dấu nháy đơn (mạnh) (*single quote*) ‘ ’
  - Dấu nháy kép (yếu) (*double quote*) “ ”

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 tệp chứa khoảng trắng hoặc ký 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 director is \$HOME”**





# Lệnh sed

---

- Trong Linux **sed** là một trình soạn thảo luồng được sử dụng như một bộ lọc có chức năng tìm kiếm và thay thế chuỗi ký tự.

cú pháp: **sed <options> 'commands' FILE**

- ví dụ: tìm kiếm và thay thế ký tự “*day*” thành “*night*” trong tệp test.txt và ghi vào tệp mới new\_test.txt

**\$sed 's/day/night/' test.txt > new\_test.txt**