



Quản lý tiến trình

TS Trương Diệu Linh



Giới thiệu

- Một tiến trình = thực thi của một chương trình được thực thi
- Mỗi tiến trình sẽ tương ứng với một tập các thông tin sau:
 - Một định danh (pid)
 - Một tiến trình cha (ppid)
 - Người sở hữu (uid) và nhóm (gid)
 - Câu lệnh khởi tạo tiến trình
 - Một đầu vào chuẩn (stdin), một đầu ra chuẩn (stdout), một kênh báo lỗi chuẩn (stderr)
 - Thời gian sử dụng CPU (CPU time) và mức độ ưu tiên
 - Thư mục hoạt động hiện tại của tiến trình
 - Bảng các tham chiếu đến các file được tiến trình sử dụng.
- Các tiến trình được sắp xếp để chia sẻ thời gian sử dụng CPU



Các kiểu tiến trình (1)

■ Các tiến trình hệ thống

- Thường thuộc về quyền root
- Không có giao diện tương tác
- Thường được chạy dưới dạng các tiến trình ngầm (daemon)
- Đảm nhiệm các nhiệm vụ chung, phục vụ mọi người sử dụng.
- Ví dụ:
 - **lpsched**: Quản lý các dịch vụ in ấn
 - **cron**: tự động thực hiện một lệnh/chương trình vào một thời gian xác định trước.
 - **inetd**: quản lý các dịch vụ mạng.



Các kiểu tiến trình (2)

■ Các tiến trình của người sử dụng

- Thực hiện các nhiệm vụ của một người dùng cụ thể
 - Thực hiện dưới dạng một shell tương ứng với một sự đăng nhập.
 - Thực hiện dưới dạng một lệnh thông qua shell
- Thường được thực hiện, quản lý bằng một terminal
- Ví dụ:
 - cp
 - vi
 - man
 - ...



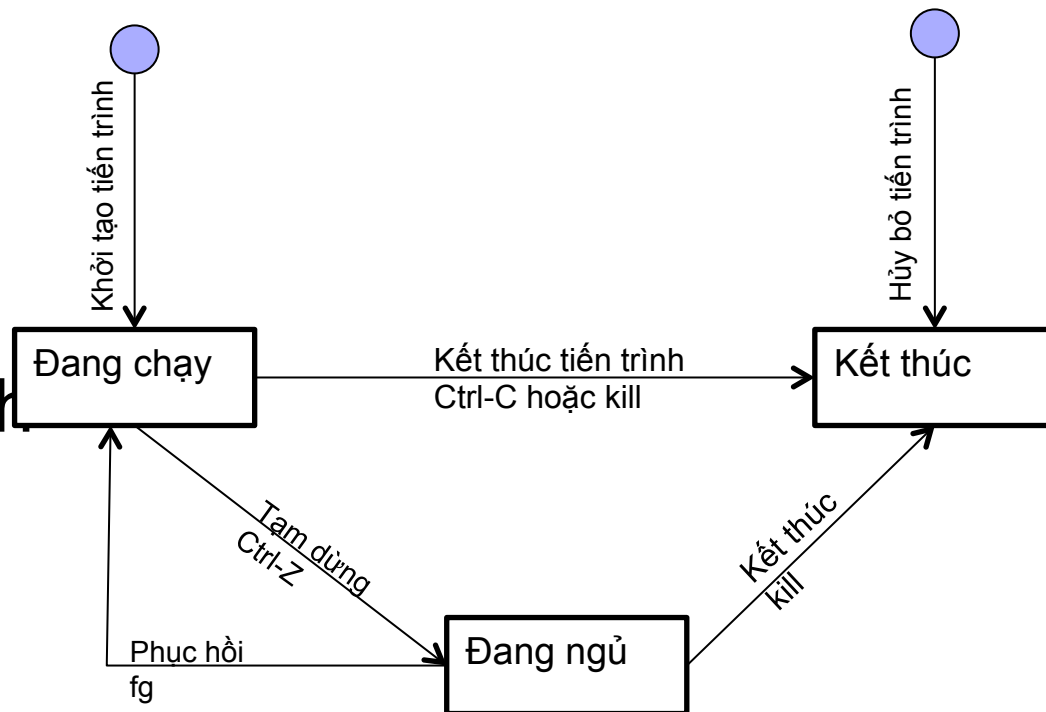
- Theo ngầm định, lệnh ps hiển thị các tiến trình thuộc về người sử dụng terminal.
- Sử dụng tùy chọn aux để hiển thị tất cả các tiến trình đang chạy trong máy.

PID	TTY	TIME	CMD
2803	pts/1	00:00:00	bash
2965	pts/1	00:00:00	ps

```
USER      PID    %CPU    %MEM    VSZ     RSS    TTY      STAT    START   TIME    COMMAND
root       1      0.1     0.1    1104     460      ?        S       15:26   0:03   init[3]
...
ttanh    951      0.0     0.3    1728     996    pts/0    S       16:09   0:00   bash
ttanh    953      0.0     1.9    6860    4916    pts/0    S       16:09   0:00   emacs
ttanh    966      0.0     0.3    2704    1000    pts/0    R       16:23   0:00   ps aux
...
```

Trạng thái của tiến trình

- **S**: đang ngủ
- **R**: đang chạy
- **T**: dừng
- **Z**: không xác định





Lệnh kill

- Gửi một tín hiệu đến một tiến trình (định danh của tiến trình được xác định dưới dạng một tham số của lệnh).
 - Theo ngầm định, tín hiệu gửi đi là tín hiệu 15 (SIGTERM – kết thúc tiến trình)
 - \$ kill pid
 - Tùy chọn -l: liệt kê tất cả các tín hiệu có thể sử dụng.
- Lệnh killall: dùng để kết thúc tất cả các tiến trình của một câu lệnh thông qua việc truyền tên của câu lệnh dưới dạng một tham số.
 - killall tên_lệnh
- Quyền hủy tiến trình thuộc về người sở hữu tiến trình




Độ ưu tiên của các tiến trình

- Tất cả các tiến trình đều có độ ưu tiên ban đầu được ngầm định là **0**
- Mức độ ưu tiên của một tiến trình dao động trong khoảng từ **-19** đến **+19**
 - Chỉ người sử dụng có quyền root mới có thể giảm giá trị biểu diễn độ ưu tiên của tiến trình. Một người sử dụng thông thường chỉ có thể làm giảm độ ưu tiên của tiến trình thông qua việc tăng giá trị biểu diễn độ ưu tiên.
- Lệnh **nice** cho phép thay đổi độ ưu tiên của một tiến trình ngay khi bắt đầu thực hiện lệnh tương ứng với tiến trình.
 - `$ nice [-n Value] [Command [Arguments ...]]`
- Lệnh **renice** cho phép thay đổi độ ưu tiên của một tiến trình sau khi đã chạy.




Lệnh top

- Hiển thị và cập nhật các thông tin sau của các tiến trình đang chạy:
 - ☐ Phần trăm sử dụng CPU
 - ☐ Phần trăm sử dụng bộ nhớ trong
- \$ top [-d] delay
 - ☐ Tùy chọn -d cho phép xác định thời gian định kỳ cập nhật thông tin (tính theo giây).
- Lệnh top cho phép người sử dụng tương tác và quản lý các tiến trình
 - ☐ Phím « f » trong quá trình hoạt động của top cho phép thay đổi, lựa chọn các thông tin cần hiển thị



Chạy ở chế độ hiện (foreground và chạy ở chế độ ngầm (background) (1)

- Quá trình chạy ở chế độ hiện sẽ tiến hành theo những bước như sau:
 - Thực hiện quá trình « fork », nhân bản tiến trình cha (trong trường hợp thực thi các lệnh, đó sẽ là tiến trình shell)
 - Thực hiện quá trình « wait », đưa tiến trình cha vào trạng thái ngủ (sleep).
 - Thực hiện quá trình « exec », thực thi tiến trình con.
 - Sau khi tiến trình con thực thi xong, một tín hiệu « đánh thức » sẽ được gửi đến tiến trình cha.
 - Do quá trình chạy như trên => trong quá trình thực hiện tiến trình con, người sử dụng không thể tương tác với tiến trình cha.

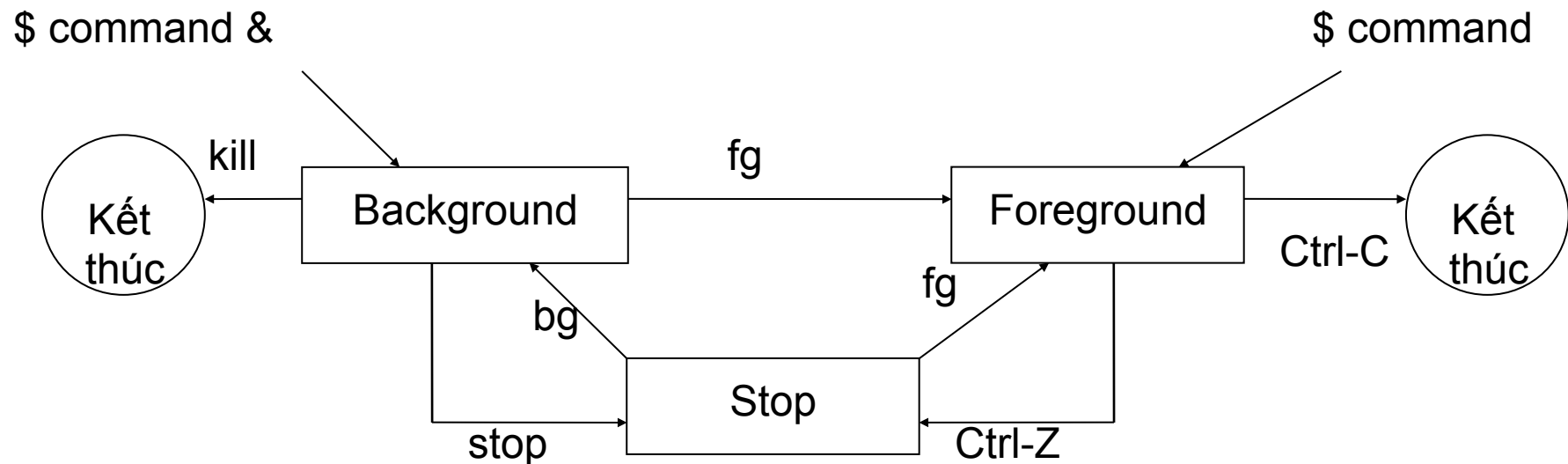


Chạy ở chế độ hiện (foreground và chạy ở chế độ ngầm (background) (2)

- Quá trình chạy ở chế độ ngầm cho phép thực thi tiến trình cha và tiến trình con một cách độc lập.
- Ví dụ: `$ emacs &`
- Sau khi thực hiện lệnh trên, emacs sẽ chạy ở chế độ ngầm, người sử dụng có thể tiếp tục sử dụng console để thực thi các lệnh khác

Quản lý tác vụ

- Một tác vụ = việc thực hiện một câu lệnh. Một tác vụ có thể liên quan đến một nhóm các tiến trình (một tiến trình cha và tập các tiến trình con của nó)
- Không thể có nhiều hơn 1 tác vụ chạy ở chế độ hiện (foreground)
- Có thể có nhiều hơn 1 tác vụ chạy ở chế độ ngầm (background)





Ví dụ

```
$ emacs &
```

```
[1] 756
```

```
$ fg %1
```

```
$ kill -9 756
```



Lọc thông tin

- `$cat file_path [...]`
 - Hiển thị nội dung 1 hoặc nhiều file
- `$head -n file_path`
 - Hiển thị n dòng đầu tiên của 1 file
- `$tail -n file_path`
 - Hiển thị n dòng cuối cùng của 1 file
- `$wc file_path`
 - Hiển thị số dòng, số từ, số ký tự trong 1 file thông qua các tùy chọn `-l`, `-w`, `-c`




grep : Tìm kiếm các dòng trong tệp

`$grep [-options] expreg [files]`

- ☐ Tìm kiếm trong file hoặc trong đầu vào chuẩn các dòng có chứa các ký tự hoặc xâu thỏa mãn một số điều kiện nào đó.

■ Các tùy chọn

- ☐ -c : chỉ đưa ra tổng số dòng thỏa mãn điều kiện
- ☐ -l : chỉ đưa ra tên file
- ☐ -v : chỉ đưa ra các dòng mà điều kiện tìm kiếm không thỏa mãn
- ☐ -i : không phân biệt chữ hoa, chữ thường
- ☐ -n : chỉ đưa ra số thứ tự của dòng
- ☐ -w : thực hiện tìm kiếm với quá trình so sánh được thực hiện đối với từng từ của chuỗi nhập vào (các từ được phân cách nhau bằng các ký tự không phải là chữ cái, không phải là chữ số và không phải là dấu gạch dưới _)



Một số ký tự đặc biệt

- grep sử dụng một số ký tự đặc biệt trong câu lệnh:
 - . Biểu diễn 1 ký tự bất kỳ
 - * Lặp lại ký tự ở vị trí trước
 - ^ bắt đầu 1 dòng
 - \$ kết thúc 1 dòng
 - [...] xác định danh sách hoặc một khoảng các ký tự cần tìm kiếm
 - [^..] các ký tự không tìm kiếm
- Chú ý: để tránh nhầm lẫn, nên đặt các ký tự biểu diễn điều kiện trong ngoặc kép.



Ví dụ

- `$grep "tuananh" /etc/passwd`
 - Tìm tất cả các dòng có chứa xâu "tuananh"
- `$grep "^t" /etc/passwd`
 - Tìm trong file /etc/passwd tất cả các dòng bắt đầu bằng ký tự "t"
- `$grep [^t] /etc/passwd`
 - Tìm tất cả các dòng không bắt đầu bằng ký tự "t"
- `$ls -l /etc | grep "^d"`
 - Hiển thị tất cả các thư mục con của /etc



Bài tập

- Viết lệnh xem danh sách các tiến trình
- Tìm dòng có chứa số hiệu của tiến trình bash
- Dùng lệnh ls liệt kê nội dung thư mục hiện tại
 - Viết lệnh trích ra danh sách các tệp nằm trong thư mục hiện tại mà chủ sở hữu có quyền đọc, ghi, thực thi.



cut : Xác định các cột

`$cut -options [files]`

■ Tùy chọn

- ☐ `-c<Số ký tự>` xác định các ký tự
- ☐ `-f<Số trường>` xác định các trường
- ☐ `-d<phân cách>`

■ Ví dụ

- ☐ `$cut -c5 file` #hiển thị ký tự thứ 5
- ☐ `$cut -c5-10 file` #hiển thị ký tự thứ 5 đến ký tự thứ 10
- ☐ `$cut -d: -f1 /etc/passwd` #hiển thị tên tất cả người sử dụng của hệ thống



sort: sắp xếp nội dung

- `$sort -options tên_file`

- Các tùy chọn

- ☐ **-b**: bỏ qua các dấu cách ở đầu mỗi trường
- ☐ **-d** : sắp xếp chỉ dựa vào các ký tự trong bảng chữ cái và chữ số (ký tự, chữ số, dấu cách)
- ☐ **-r** : đảo ngược thứ tự sắp xếp
- ☐ **-f** : không phân biệt chữ thường chữ hoa
- ☐ **-t x** : ký tự x được sử dụng làm ký hiệu ngăn cách giữa các trường
- ☐ **-u** : xóa các dòng trùng nhau.
- ☐ **-n** sắp xếp dựa vào các chữ số
- ☐ **-kx** sắp xếp theo trường thứ x




Ví dụ

- carnet.txt

- maurice:29:0298334432:Crozon
 - marcel:13:0466342233:Marseille
 - robert:75:0144234452:Paris
 - yvonne:92:0133444335:Palaiseau

- \$sort -n -t : -k2 carnet.txt

- ☐ Thực hiện quá trình sắp xếp theo trường thứ 2



Thay đổi nội dung file

■ split

- Cắt một file ra thành nhiều file nhỏ hơn

- Ví dụ:

 - `split -10 /etc/passwd smallpasswd`

■ tr

- Thay thế một chuỗi bằng một chuỗi khác có cùng độ dài


- Ví dụ:

 - `$cat /etc/passwd | tr ":" "#"`



So sánh các file

- `$cmp file1 file2`
 - So sánh file1 và file2
- `$diff file1 file2`
 - Tìm sự khác nhau giữa file1 và file2 (các file dưới dạng văn bản)
 - Kết quả hiển thị dưới dạng các dòng



Các cách thực hiện nối tiếp các câu lệnh

- `cmd1;cmd2`
- `cmd1 && cmd2`
- `cmd1 | cmd2`



Các kiểu thực thi

■ Thực thi nhiều lệnh độc lập

- Sử dụng ký tự `;` để thực thi nhiều lệnh liên tiếp, các lệnh này hoạt động độc lập với nhau.

- `$cp public/* perso; rm -r public`

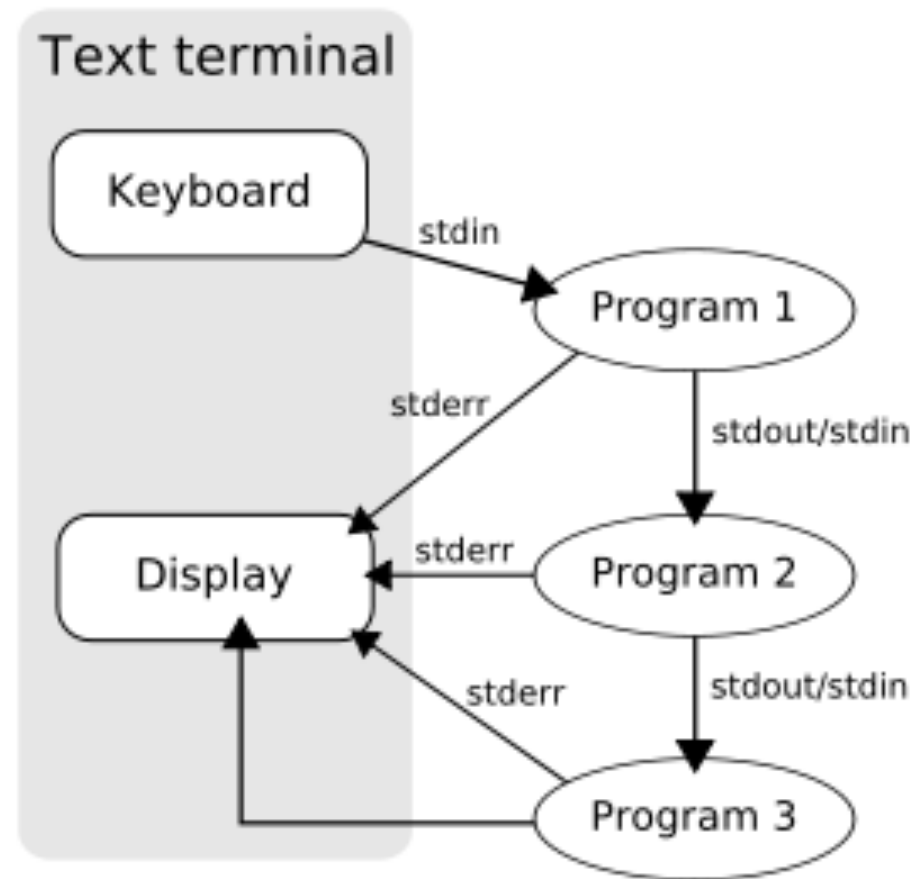
■ Thực thi nhiều lệnh phụ thuộc nhau

- Sử dụng ký hiệu **`&&`** để thực thi nhiều lệnh liên tiếp, các lệnh này phụ thuộc nhau, lệnh sau chỉ được thực hiện nếu lệnh trước không gặp lỗi.

- `$cp public/* perso && rm -r public`

Cơ chế đường ống

- Cơ chế đường ống giữa hai tiến trình cho phép định hướng lại đầu ra của tiến trình thứ nhất trở thành đầu vào của tiến trình thứ hai
- Cơ chế đường ống được thiết lập bằng cách sử dụng ký tự: |
 - \$ cmd1 | cmd2





Chuyển hướng các kênh chuẩn

- Mỗi tiến trình sở hữu:
 - Một đầu vào chuẩn (ngầm định là bàn phím)
 - Một đầu ra chuẩn (ngầm định là terminal)
 - Một kênh báo lỗi chuẩn (ngầm định là terminal)
- Chuyển hướng đầu vào chuẩn (<)

```
$ tee < test.txt
```
- Chuyển hướng đầu ra chuẩn (>, >>)

```
$ ls > file_1  
$ ls >> test.txt
```
- Chuyển hướng kênh báo lỗi

```
$ rm prog.c 2> /dev/null  
$ gcc prog.c 2>> erreur.txt
```



Bài tập

- Viết câu lệnh liệt kê nội dung của thư mục hiện tại và xuất kết quả ra tệp kq.txt
- Viết lệnh tìm tệp “*.txt” từ thư mục gốc
 - Sửa lại câu lệnh để:
 - Xuất kết quả nối tiếp tiếp vào kq.txt
 - Xuất báo lỗi ra file log.txt
- Liệt kê ra file result.txt danh sách các tệp trong thư mục hiện tại mà chủ sở hữu có quyền đọc, ghi
- Tạo thư mục d2 trong thư mục d1 trong thư mục hiện tại bằng cách ghép các câu lệnh



Bài tập

1) Kết quả thực hiện lệnh ls đang là như sau:

```
$ ls -l
```

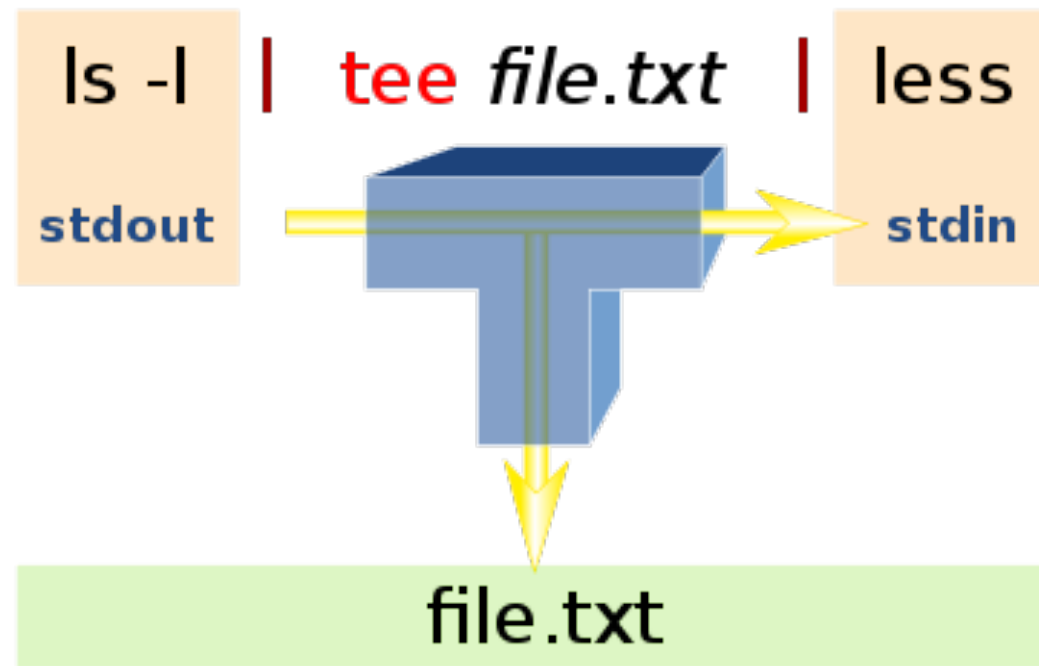
```
-rw-r--r-- 1 tuananh user1 8 Feb 10 1:12 test.txt
```

Viết câu lệnh để mọi người dùng có thể thực thi tệp test.txt và người dùng thuộc nhóm user1 có thể xóa tệp này

2) Viết câu lệnh tạo file tien_trinh.txt chứa danh sách các tiến trình đang chạy

3) Cho một file văn bản có tên day_log.txt. Dùng lệnh cat để ghi NỘI TIẾP nội dung file này vào file month_log.txt và xóa file day_log.txt một khi quá trình ghi kết thúc thành công.

tee command





Bai tap

- Cho file dulieu.dat

Phạm Đình Hảo? Ginggon91@gmail.com?20090991

Phạm Duy Hoàn? 20091134@students.hust.edu.vn? 20091134

Nguyễn Xuân Hoàng? nguyenxuanhoang281@gmail.com? 20091171

Đặng Văn Khánh?khanh20091171@gmail.com? 20091432

Lê Hoàng Kiên? Uno_s2@yahoo.com? 20091505

Ngô Hải Linh? Hoacotu326@yahoo.com? 20091596

--Kết thúc tệp--

- Viết câu lệnh Linux để trích lấy danh sách họ tên sinh viên.
- Viết câu lệnh Linux để trích chọn Tên của sinh viên có mã số X.
- Viết câu lệnh sắp xếp các sinh viên theo số hiệu SV

Sắp xếp danh sách sv theo thứ tự từ điển của tên.