Quản lý tiến trình

TS Hà Quốc Trung

Giới thiệu

- Một tiến trình = một sự thực thi của một chương trình
- 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)
  - 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
    - …

cuu duong than cong . com

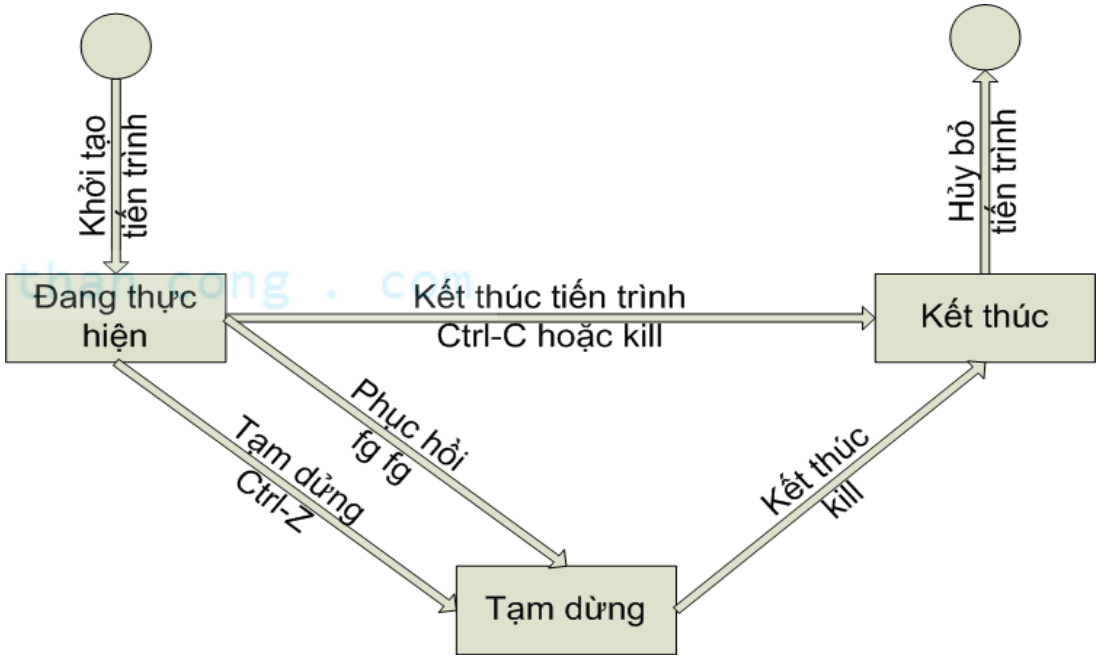
Lệnh ps

- Hiển thị các tiến trình
  - 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.


```
$ ps
 PID TTY          TIME CMD
2803 pts/1    00:00:00 bash
2965 pts/1    00:00:00 ps
$ ps aux
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)
  - Tùy chọn -9: gửi tín hiệu 9 (SIGKILL – hủy tiến trình)
  - 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.
- 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]
  - 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 (thay đổi độ ưu tiên, gửi các tín hiệu, …)

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`

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

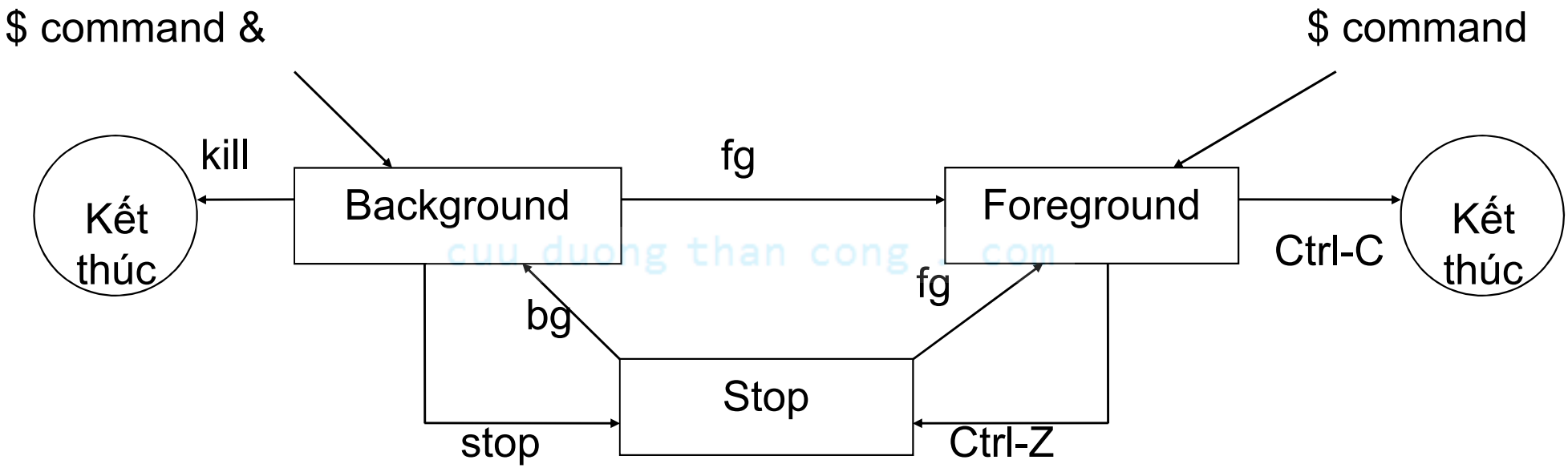
- Quá trình chạy ở chế độ hiện sẽ tiến hành theo những bước như sau:
  - Thực hiện quá trình « fork », nhân bản tiến trình cha (trong trường hợp thực thi các lệnh, đó sẽ là tiến trình shell)
  - Thực hiện quá trình « wait », đưa tiến trình cha vào trạng thái ngủ (sleep).
  - Thực hiện quá trình « exec », 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
$ stop 756
# or $ stop %1
$ bg 756
# or $ bg %1
$ kill 756
# or $ kill %1
```

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 > /dev/lp
$ ls >> test.txt
```

- Chuyển hướng kênh báo lỗi

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

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
- Ví dụ
  - `$ls -l | more #affiche page par page`
  - `$ls -l | tee log.txt #duplique la sortie`