

Bài thực hành số 4

QUẢN LÝ TIẾN TRÌNH TÍN HIỆU VÀ ĐỒNG BỘ

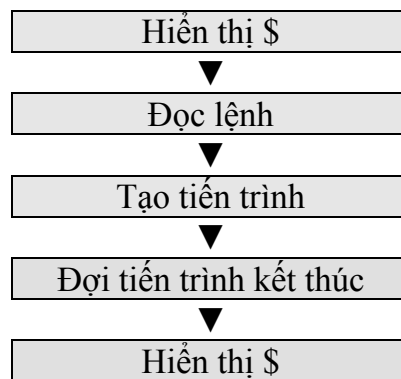
1 Nội dung thực hành

- Tiến trình và cách thức thực hiện tiến trình
- Quản lý các tiến trình
- Quản lý các tín hiệu

2 Quản lý tiến trình

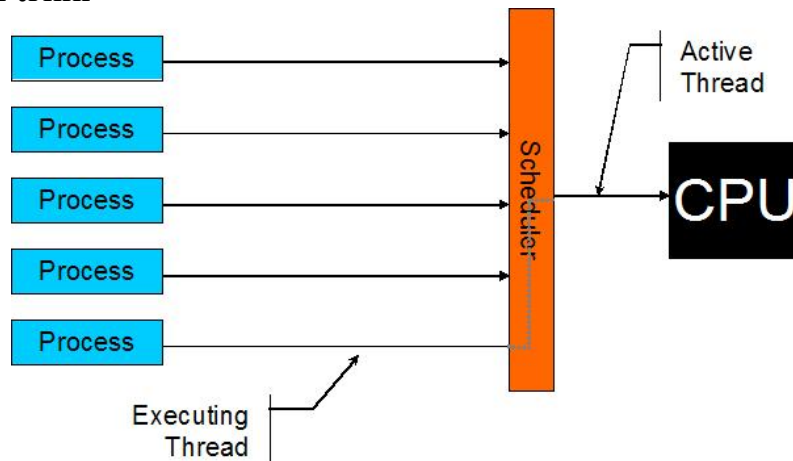
2.1 Mục đích của Shell

Shell là chương trình thông dịch lệnh.



Shell có thể đọc và thực hiện 1 tập tin gồm danh sách các lệnh cần thực hiện. Tập tin ở dạng này được gọi là shell_script hoặc procedure. Shell_script được thực hiện nhờ shell, và chính shell sẽ phát sinh và quản lý tất cả các tiến trình cần thiết để thực hiện công việc được mô tả trong shell_script.

2.2 Tạo tiến trình



Tiến trình được hiểu là việc thực hiện 1 công việc hay 1 chương trình trong 1 môi trường cụ thể trong hệ thống. Ta có thể phân biệt 2 loại tiến trình:

- Tiến trình hệ thống: là tiến trình không gắn với bất kỳ 1 terminal nào, nó được tạo ra vào thời điểm khởi động hệ thống hoặc vào các thời điểm cố định do người quản trị hệ thống đặt.
- Tiến trình do người sử dụng tạo ra

Các tiến trình được tổ chức theo dạng cây: Tiến trình cha → tiến trình con

Đối với người sử dụng, tiến trình cha là shell được tạo tại thời điểm bắt đầu phiên làm việc.

Cách thực hiện 1 shell_script như sau:

```
$chmod +x proc
```

```
$proc
```

hoặc: \$sh proc

2.3 Liệt kê các tiến trình

Lệnh ps cho phép liệt kê danh sách các tiến trình đang diễn ra:

```
$ps
```

PID	TTY	TIME	CMD
3884	pts/1	00:00:00	bash
3955	pts/2	00:00:00	more
3956	pts/5	00:00:05	sqlplus

Với :

PID số của tiến trình (process identity)

TTY terminal điều khiển tiến trình

TIME thời gian tích lũy thực hiện tiến trình (cumulative time)

COMMAND tên lệnh sinh ra tiến trình

Tiến trình số 1 là tiến trình init, trong đó có chức năng giám sát các terminal, là tiến trình cha của tất cả các tiến trình shell khi login.

Các tùy chọn cho lệnh ps:

- **a** : hiển thị tất cả các tiến trình của người dùng (all users)
- **u** : hiển thị các tiến trình cùng với tên người dùng và thời gian bắt đầu
- **x** : hiển thị các tiến trình mà không gắn kết với một terminal

Trong đó:

UID	số uid của người chủ tiến trình
PID	số của tiến trình (process identity)
PPID	số của tiến trình cha (parent process identity)
C	chỉ số sử dụng bộ xử lý (processor utilization for scheduling)
STIME	thời điểm bắt đầu tiến trình
TTY	terminal điều khiển tiến trình
TIME	thời gian tích lũy thực hiện tiến trình (cumulative time)
COMMAND	tên lệnh sinh ra tiến trình

2.4 Các lệnh có liên quan đến tiến trình

Dưới đây là danh sách các lệnh có liên quan đến tiến trình:

Lệnh	Ý nghĩa	Ví dụ
ps	Hiển thị các tiến trình đang chạy	\$ ps
ps -ag	Hiển thị thông tin về tất cả các tiến trình đang chạy	\$ ps -ag
ps aux	Hiển thị các tiến trình và người dùng tạo ra tiến trình	\$ ps aux
kill {PID}	Stop tiến trình PID	\$ kill 1012
killall {Process-name}	Stop tiến trình có tên Process-name	\$ killall httpd
kill 0	Stop tất cả các tiến trình ngoại trừ shell của người dùng	\$ kill 0
Linux-command &	Tiến trình sẽ chạy ngầm (background). Số PID của tiến trình ngầm ở trong biến \$!. Ta có thể đổi hướng vào/ra của tiến trình ngầm, tránh nhiễu màn hình khi ta làm việc khác.	\$ ls / -R wc -l &
free	Thống kê việc sử dụng bộ nhớ	\$free
pstree	Hiển thị cây quá trình	\$ pstree
[ctrl-z]	Làm treo (tạm ngưng) một tiến trình đang chạy	ấn phím [ctrl-z]
bg	Chuyển quá trình đang chạy sang chế độ chạy ngầm (background). Cần phải treo tiến trình trước khi thực hiện nó	\$bg
fg	Chuyển quá trình đang chạy ngầm (background) sang chế độ chạy bình thường (foreground).	\$fg
jobs	Liệt kê các tiến trình chạy ngầm	\$jobs
top	Hiển thị theo thời gian thực các tiến trình hiện đang chạy và các thông tin khác có liên quan:bộ nhớ, sử dụng CPU.	\$ top

☞ Ví dụ: output của lệnh top:

\$top

```

1:25am up 3:25, 1 user, load average: 0.00, 0.00, 0.00
52 processes: 51 sleeping, 1 running, 0 zombie, 0 stopped
CPU states: 0.0% user, 9.0% system, 0.0% nice, 90.9% idle
Mem: 126844K av, 44864K used, 81980K free, 37840K shrd, 2092K buff
Swap: 136512K av, 0K used, 136512K free 27032K cache
  
```

PID	USER	PRI	NI	SIZE	RSS	SHARE	STAT	LIB	%CPU	%MEM	TIME	COMMAND
1195	vivek	19	0	860	860	668	R	0	9.2	0.6	0:00	top
1	root	0	0	476	476	404	S	0	0.0	0.3	0:04	init
2	root	0	0	0	0	0	SW	0	0.0	0.0	0:00	kflushd
3	root	0	0	0	0	0	SW	0	0.0	0.0	0:00	kupdate
4	root	0	0	0	0	0	SW	0	0.0	0.0	0:00	kpiod
5	root	0	0	0	0	0	SW	0	0.0	0.0	0:00	kswapd
6	root	-20	-20	0	0	0	SW<	0	0.0	0.0	0:00	mdrecovery
408	bin	0	0	428	428	340	S	0	0.0	0.3	0:00	portmap
423	root	0	0	0	0	0	SW	0	0.0	0.0	0:00	lockd
424	root	0	0	0	0	0	SW	0	0.0	0.0	0:00	rpciod
433	root	0	0	560	560	472	S	0	0.0	0.4	0:00	rpc.statd
447	root	0	0	476	476	412	S	0	0.0	0.3	0:00	apmd
498	root	0	0	528	528	428	S	0	0.0	0.4	0:00	syslogd
507	root	0	0	764	764	388	S	0	0.0	0.6	0:00	klogd
521	nobody	0	0	624	624	520	S	0	0.0	0.4	0:00	identd
525	nobody	0	0	624	624	520	S	0	0.0	0.4	0:00	identd

2.5 Quản lý tiến trình ngầm

- Tạo tiến trình chạy ngầm (background):
\$lệnh &
 - Quản lý các tiến trình đang chạy ngầm:
\$set -m
 - Hiển thị trạng thái của các tiến trình ngầm:
\$jobs -l
 - Tiếp tục thực hiện các tiến trình ngầm sau khi cắt liên lạc với terminal:
\$nohup lệnh &
Các số liệu của tiến trình đưa ra stdout và stderr sẽ được ghi lên tập tin nohup.out
 - Đợi kết thúc tiến trình:
\$wait PID
 - Hủy tiến trình:
\$kill PID (phát sinh tín hiệu 15 - mặc định)
- Hoặc
- \$kill -n PID (phát sinh tín hiệu diệt tiến trình)
- Hoặc:
- \$kill %n
diệt tiến trình theo số thứ tự của nó trong danh sách các tiến trình chạy ngầm

2.6 Thiết lập cách thức cho shell_script

Lệnh set cho phép thiết lập cách thức chạy 1 shell_script

```
set -x    hiển thị dòng lệnh sau khi triển khai lệnh
set -v    hiển thị dòng lệnh trước khi triển khai lệnh
set -e    ra khỏi Shell_script sau khi gặp 1 lỗi
set -t    ra khỏi Shell_script sau lệnh tiếp
set -     xóa tác dụng của x và v
```

Việc thiết lập cách thức chỉ liên quan đến shell_script đang chạy. Các tùy chọn `-x` và `-v` có thể đưa vào dòng lệnh gọi shell_script:

```
$sh -v proc
$sh -x proc
```

Có thể sử dụng ký tự '#' để viết chú thích cho dòng lệnh trong Shell_script, nếu chú thích viết ngay sau lệnh trên cùng một dòng, ta phải cho ít nhất 1 dấu cách (space) vào trước ký tự '#'.

3 Quản lý các tín hiệu

3.1 Các tín hiệu

Trong khi thực hiện một shell_script, các tín hiệu sau có thể phát sinh:

```
signal 0    ra khỏi shell (exit of the shell)
signal 1    cắt liên lạc với terminal (disconnection)
signal 2    Ngắt (ví dụ phím DEL)
signal 3    Quit (Ctrl |)
signal 9    Hủy tiến trình
signal 10   Kết thúc logic một tiến trình
```

Trong một chương trình ứng dụng, bằng cách dùng lệnh trap, ta có thể định nghĩa việc cần xử lý khi một tín hiệu phát sinh. Lệnh này cho phép gán một công việc xử lý cho bất cứ một tín hiệu nào.

3.2 Lập trình phím DEL

- Lệnh trap không đối số sẽ liệt kê danh sách các tín hiệu và các xử lý tương ứng.
- Cú pháp gán một công việc xử lý cho phím DEL:
\$trap 'các lệnh' 2
- Xóa bỏ tác dụng phím DEL:
\$trap '' 2
- Gán chức năng mặc định cho phím DEL :
\$trap 2

4 Bài tập

Sinh viên phải đọc kỹ phần lý thuyết và làm các ví dụ trên trước khi làm phần bài tập

4.1 Bài tập 1

Sử dụng câu lệnh ps, thực hiện các yêu cầu sau:

1. Hiển thị tất cả các tiến trình của người dùng hiện đang làm việc
2. Hiển thị tất cả các tiến trình hiện đang chạy
3. Đọc và hiểu các output trên

4.2 Bài tập 2

Giải thích output của câu lệnh ps sau:

```
$ps -l &  
[1] 478
```

4.3 Bài tập 3

Thực hiện tạo một tiến trình và một tiến trình chạy ngầm (background). Cho nhận xét về 2 tiến trình trên. (cho ví dụ minh họa và nhận xét)

4.4 Bài tập 4

Tạo một tiến trình chạy ngầm với câu lệnh ps. Khi tiến trình trên kết thúc, một thông báo kết thúc sẽ xuất hiện : “END.” (không sử dụng shell script)

4.5 Bài tập 5

Tuần tự thực hiện các yêu cầu sau:

1. Thực thi lệnh top
2. Treo tiến trình đó
3. Tạo một tiến trình mới mà ở đó nó sẽ bị trì hoãn 2 lần 60 giây trước khi thực hiện lưu output của lệnh **ls -l** vào tập tin flist
4. Chuyển tiến trình ở bước 3 sang chế độ chạy ngầm (background)
5. Liệt kê các tiến trình đang chạy ngầm
6. Hủy tiến trình ở bước 1
7. Kiểm tra lại để thấy tiến trình ở bước 3 là hoàn tất.
8. Cho nhận xét việc thực hiện các công việc 1 -> 7

4.6 Bài tập 6

Liệt kê tất cả các tiến trình hiện đang chạy theo từng trang màn hình

4.7 Bài tập 7

Tạo một shell script có tên uncount hiển thị dãy chữ số :

```
6 5 4 3 2 1
```

trong các khoảng thời gian 5 giây (hiển thị một số/5giây), mà nếu ta gõ phím DEL thì nó sẽ hiển thị chữ số kế tiếp

4.8 Bài tập 8

Liệt kê các tiến trình ngầm hiện đang chạy

4.9 Bài tập 9

Liệt kê các tiến trình ngầm đã kết thúc

4.10 Bài tập 10

Xác định số tín hiệu phát sinh khi thực hiện lệnh:

1. \$kill 408
2. \$kill -9 521

4.11 Bài tập 11

Hãy viết shell script : LisFileDel file1 file2

Chức năng:

- hiển thị nội dung các tập tin có tên trong danh sách đối số
- nếu gõ phím DEL, bỏ qua tập tin đang hiển thị, bắt đầu tập tin kế tiếp
- khôi phục chức năng mặc định của phím DEL khi kết thúc

4.12 Bài tập 12

Hãy viết shell script : trap2

Chức năng:

- thực hiện một chu trình hiển thị một thông báo:
“Shutdown in n minutes” n có giá trị từ 5 đến 1
- mỗi khi gõ phím DEL, lập tức hiển thị thông báo tiếp theo
- khôi phục chức năng mặc định của phím DEL khi kết thúc