Chương 5:
Quản trị tiến trình

1

Nội dung

Chương 4. Quản trị tiến trình

4.1. Khái niệm tiến trình

4.1.1. Sơ bộ về tiến trình
4.1.2. Sơ bộ về cấu trúc điều khiển của UNIX
4.1.3. Các hệ thống con trong nhân
4.1.4. Sơ bộ về điều khiển tiến trình
4.1.5. Trạng thái và chuyển trạng thái
4.1.6. Sự ngưng hoạt động và hoạt động trở lại của tiến trình
4.1.7. Sơ bộ về lệnh đối với tiến trình

4.2. Các lệnh cơ bản

4.2.1. Lệnh fg và lệnh bg
4.2.2. Hiển thị các tiến trình đang chạy với lệnh ps
4.2.3. Hủy tiến trình với lệnh kill
4.2.4. Cho máy tạm ngừng hoạt động (suspend) với lệnh sleep
4.2.5. Xem cây tiến trình với lệnh pstree
4.2.6. Lệnh thiết lập lại độ ưu tiên của tiến trình: nice và renice

2

Sơ bộ về tiến trình

- Chương trình là một file được lưu trên thiết bị lưu trữ và có thể thực thi khi chạy nó
  - Ví dụ: /sbin/shutdown, /sbin/init
- Tiến trình là ***một chương trình trong thời gian vận hành*** (đang chiếm một phần bộ nhớ RAM và CPU)

3

Sơ bộ về tiến trình

- Từ *một chương trình* có thể sinh ra *nhiều tiến trình* trong hệ thống. Mỗi tiến trình được xác định thông qua một số nguyên duy nhất gọi là PID (process identification)
- Các tiến trình đồng hành, dùng chung CPU
- Hệ điều hành phân chia thời gian để kiểm soát các tiến trình
- Thuật ngữ: *process - tiến trình - quá trình - task*

4

Phân loại tiến trình

- **User process**: các tiến trình được thực hiện bởi người dùng từ terminal (sau khi người dùng đăng nhập thành công vào hệ thống)
- **Daemon proccess**: các tiến trình cung cấp các dịch vụ hệ thống: hệ thống log, scheduling, printing, web-server, file-server, database server,... (được kích hoạt trước khi người dùng đăng nhập)

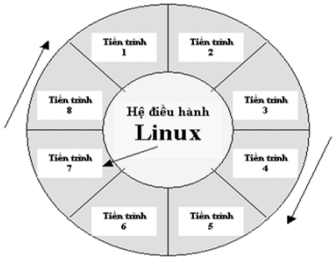
5

Sự thực thi của các tiến trình

- Ví dụ trong môi trường đồ họa (Graphic Mode), vừa có thể nghe nhạc lại vừa có thể soạn thảo văn bản. Trong chế độ Console Mode, vừa có thể chạy chương trình xử lý thuật toán nén file lại vừa có thể ra lệnh in văn bản ra máy in.
- **->Xử lý song song?**
- Thực tế, các tiến trình được thực thi một cách tuần tự chứ không song song. Mỗi thời điểm, CPU chỉ có khả năng xử lý được một chỉ thị lệnh duy nhất.

6

- Hầu hết các HĐH đều mô phỏng khả năng xử lý song song (***Parallel Processing***) bằng kỹ thuật điều phối tiến trình (***Time Schedule***). CPU sẽ được điều phối xoay vòng, mỗi tiến trình chiếm giữ một thời gian của CPU rất ngắn sau đó HĐH sẽ can thiệp và tạm dừng để CPU có khả năng làm việc với tiến trình khác.



7

Process State

Khi một tiến trình thực thi, nó thay đổi trạng thái theo từng tình huống. Tiến trình trong Linux có các trạng thái sau đây:

•**Running**: Tiến trình hoặc là đang thực thi (là tiến trình hiện hành trong hệ thống) hoặc là đang sẵn sàng để thực thi (đang chờ để được gán cho một trong các CPU của hệ thống).

•**Waiting**: Tiến trình đang chờ một sự kiện hoặc một tài nguyên. Linux phân biệt hai loại waiting process:

- Interruptible waiting processes *có thể bị ngắt* bởi các tín hiệu
- Uninterruptible waiting processes chờ đợi một điều kiện nào đó về phần cứng và *không thể bị ngắt* theo bất kỳ điều kiện nào khác.

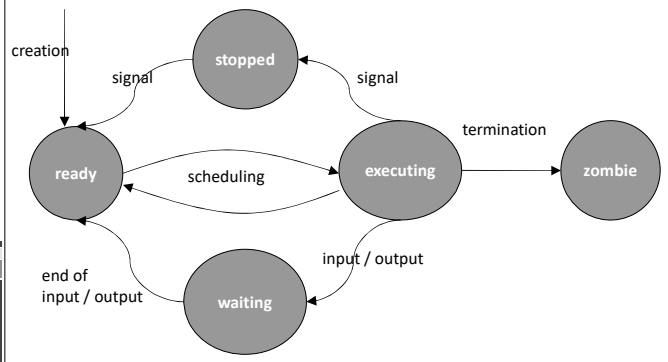
8

Process State

- **Stopped**: Một tiến trình có thể dừng lại, sau khi nhận được một tín hiệu. Một tiến trình đang được debugged có thể ở trạng thái dừng.
- **Zombie**: Là một tiến trình đã dừng hẳn, vì một số nguyên nhân, vẫn còn cấu trúc dữ liệu task-struct trong task vector. Đồng nghĩa với dead process.

9

Process State



10

Tiến trình trong Windows

- Hiển thị cửa sổ Task Manager
  - Ctrl + Alt + Del
  - Right Click trên thanh taskbar, chọn Start Task Manager
  - Chọn tab “Processes”
  - Xem các tiến trình đang thực hiện trong hệ thống
  - Mở chương trình Notepad, 2 lần, xem sự thay đổi trong cửa sổ “Processes”

11

Tiến trình trong Linux

- Xem thông tin tiến trình
- Loại bỏ tiến trình

12

Xem thông tin về tiến trình trong Linux

- Sử dụng lệnh **top** để theo dõi trạng thái các tiến trình được cập nhật liên tục (mỗi 3 giây)
- Sử dụng lệnh **ps** để xem trạng thái các tiến trình tại một thời điểm (snapshot)
- Lệnh **pstree**

13

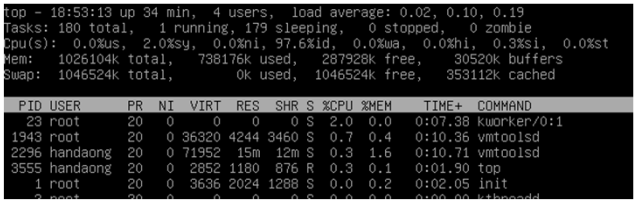
Lệnh top

- Lệnh top cung cấp một khung nhìn thời gian thực về các tiến trình đang thực thi của hệ thống
- Lệnh top cho biết chương trình gì đang chiếm bộ nhớ (hoặc hệ thống có dung lượng bộ nhớ là bao nhiêu).
- Thoát khỏi lệnh top: gõ q
- Tham số:

-d : delay khoảng thời gian refresh giữa 2 lần

-n : number - chạy number lần rồi ngưng

14



- PID -- Process Id
- USER -- User Name
- PR -- Priority - The priority of the task.
- NI -- Nice value
- S -- Process Status

'D' = uninterruptible sleep
'R' = running
'S' = sleeping
'T' = traced or stopped
'Z' = zombie

15

Xem thông tin về tiến trình - Lệnh ps

- Lệnh ***ps***: snapshot of the current processes.
- Lệnh ***ps*** có rất nhiều tùy chọn,
- Tuỳ chọn

l : hiển thị ở dạng long list

a : là yêu cầu liệt kê hết tất cả các tiến trình.

w : hiển thị ở dạng wide output

x : xem cả các process không gắn với terminal (daemon)

U : xem process của user cụ thể

u : thể hiện ở dạng user format

16

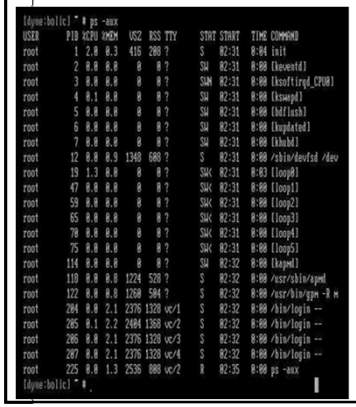
Lệnh pstree

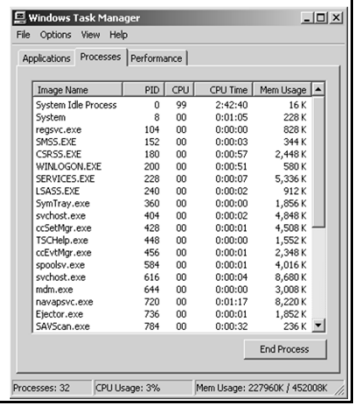
- Hiển thị process ở dạng cây
- Tham số:

-p : hiển thị cả PID

17

Process Viewer & Task Manager





18

Phân loại tiến trình

- Có các tiến trình đang vận hành
  - **Foreground** - Tiền cảnh
  - **Background** - Hậu cảnh

19

Tiến trình tiền cảnh

- Mô tả: Khi đang trên dấu nhắc của hệ thống (# hay $) và gọi thực thi một chương trình thì chương trình này sẽ trở thành tiến trình đi vào hoạt động dưới sự kiểm soát của hệ thống.
- Dấu nhắc hệ thống sẽ không hiển thị trong khi tiến trình đang chạy. Chỉ khi nào tiến trình hoàn thành tác vụ và chấm dứt thì HĐH (Shell) sẽ trả lại dấu nhắc để người dùng tiếp tục thực thi các tác vụ khác.

➢ Đây là cơ chế của tiến trình hoạt động ở chế độ **TIỀN CẢNH**

20

Tiến trình tiền cảnh

- Ví dụ:

  # ls –aR / > allfiles.txt

  Lệnh sẽ thực thi công việc liệt kê toàn bộ tập tin và thư mục (tham số R-Recursive) của HĐH bắt đầu từ thư mục gốc / vào file allfiles.txt
- Quá trình liệt kê này diễn ra có thể lâu và hiện ra trực tiếp trên màn hình. Sau khi lệnh trên thực hiện xong thì HĐH lúc này mới trả lại dấu nhắc cho người dùng → Cần đến khả năng **HẬU CẢNH** của HĐH.

21

Tiến trình hậu cảnh

- Mô tả: Nhằm mục đích đưa những tiến trình chiếm nhiều thời gian (hoặc ít tương tác với người dùng) ra hoạt động ở hậu cảnh (chạy ngầm bên trong hệ thống không cần xuất hiện)
- Thao tác: Ta chỉ cần cho dấu "**&**" sau mỗi câu lệnh
- Dấu nhắc của hệ thống hiển thị để sẵn sàng triệu gọi một chương trình khác (tiến trình trước vẫn đang chạy)
- Ví dụ:

```
# ls -aR / > allfiles.txt &
[1] 23978
```

  - Tiến trình được đưa vào hậu cảnh (thứ 1) với mã số PID là 23978

22

Tạm dừng tiến trình

- Nếu tiến trình nào đó đang chạy và cần đưa vào hậu cảnh (do phải chờ đợi việc kết thúc của tiến trình ấy lâu và khi thực thi lệnh không dùng dấu "**&**") => Bấm **Ctrl + Z**
- Khi một chương trình đang chạy và nhận được tín hiệu ngắt do bấm tổ hợp phím Ctrl + Z, tiến trình được tạm dừng và đưa vào hậu cảnh. Tuy ở hậu cảnh, nhưng tiến trình này đang trong tình trạng **PAUSE** và nó chỉ thực thi tiếp khi cho phép.
- Ví dụ:

```
# ls -aR / > allfiles.txt
^Z
[1]+       Stopped              ls -aR / > allfiles.txt
#
```

- Lệnh ***ps -af*** để xem đầy đủ thông tin về các tiến trình đang chạy.

23

Đánh thức tiến trình

- Dùng lệnh ***jobs*** để hiển thị trạng thái các tiến trình trong hậu cảnh:

```
$ jobs
[1] +      Stopped              ls –aR / > allfiles.txt
```

- Kết quả cho thấy tác vụ [1] đang ở trạng thái dừng. Để tiến trình trên tiếp tục hoạt động ở hậu cảnh, sử dụng lệnh ***bg:***

```
$ bg 1
ls –aR / > allfiles.txt
$ jobs
[1] +      Running              ls –aR / > allfiles.txt
```

- Để tiến trình tiếp tục hoạt động ở chế độ tiền cảnh, sử dụng lệnh **fg**

24

Hủy tiến trình

- Có những trường hợp như: Tiến trình bị treo hoặc lặp trong một vòng lặp vô hạn => Cần phải **Hủy tiến trình**
- Nếu không hủy kịp thời => Chiếm tài nguyên hệ thống vô ích (chậm hệ thống)
- Sử dụng lệnh ***kill*** để tiến hành hủy bỏ tiến trình. Lệnh ***kill*** đi sau với tham số là số hiệu của tiến trình (**PID**)
- Lệnh ***kill*** thường hay đi chung với lệnh ***ps –af***

25

- Ví dụ:

```
# ls –R / > allfiles.txt
^z
# ps –af
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	3822	3821	0	Arp19	tty1	00:00:00	[bash]
root	2453	2452	30	11:03	pts/3	00:00:01	ls –R /
root	2458	2459	10	11:03	pts/3	00:00:00	ps –af

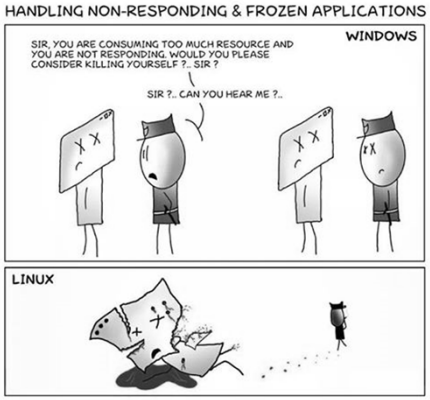
```
# kill 2453
# ps –af
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	3822	3821	0	Arp19	tty1	00:00:00	[bash]
root	2458	2459	10	11:03	pts/3	00:00:00	ps –af

- Đối với một số tiến trình có cấp độ ưu tiên cao (*High Priority*), không thể sử dụng lệnh ***kill*** mặc định để có thể dừng tiến trình được ⇨ Sử dụng thêm tham số "**-9**" để có thể hủy được tiến trình có cấp độ ưu tiên cao

```
# kill –9 2453
```

26



27

Giao tiếp giữa các tiến trình

- Các tiến trình cần phải giao tiếp với nhau để trao đổi thông tin.
- Như lệnh ***ls*** dùng để liệt kê về thông tin của tập tin và thư mục ra màn hình nhưng lệng ***ls*** trên không có tính năng dừng màn hình (nếu số dòng vượt quá 25 dòng). Tuy nhiên, lệnh ***more*** lại có thể làm được điều này ➢ Có thể kết hợp hai lệnh này lại thông qua chỉ thị "**|**" để thực thi cơ chế đường ống.
- Ví dụ sau minh họa quá trình chuyển dữ liệu do lệnh ***ls*** xuất ra sang cho lệnh ***more*** xử lý và phân trang bằng đường ống
- $**ls –R | more**

```
/
bin
boot
dev
etc
--More--
```

25 dòng tự động ngắt - sử dụng phím SpaceBar để tiếp tục

28