**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

---- 🙥🕮🙧 ----



**BÁO CÁO BTL THUỘC HỌC PHẦN:**

**NGUYÊN LÍ HỆ ĐIỀU HÀNH**

**ĐỀ TÀI: NGHIÊN CỨU TÌM HIỂU VỀ QUẢN LÝ TẾN TRÌNH TRONG HỆ ĐIỀU HÀNH LINUX**

**GVHD: Ths.Nguyễn Thanh Hải**

**Nhóm - Lớp: 2 \_ 20212IT6025008**

**Thành viên: Nguyễn Viết An**

**Trần Bá Lập**

**Nguyễn Thị Loan**

**An Thị Ngọc Mai**

**Nguyễn Văn Nam**

*Hà Nội, ngày 15 tháng 05 năm 2022*

**MỤC LỤC**

*Trang*

**MỞ ĐẦU** 3

**Chương 1: GIỚI THIỆU VỀ HỆ ĐIỀU HÀNH LINUX** 4

1.1 Tổng quan về hệ điều hành 4

1.2 Hệ điều hành Linux 5

**Chương 2: TỔNG QUAN VỀ TIẾN TRÌNH** 7

2.1 Khái niệm về tiến trình 7

2.2 Các trạng thái của một tiến trình 7

2.3 Quan hệ giữa các tiến trình 8

2.4 Phương pháp tổ chức liên lạc giữa các tiến trình 8

**Chương 3: QUẢN LÝ TIẾN TRÌNH LINUX** 9

3.1. Các loại tiến trình chính trên Linux 9

3.2. Tại sao phải quản lý tiến trình 9

3.3. Tạo một tiến trình 10

3.4. Dừng một tiến trình 11

3.5. Giao tiếp giữa các tiến trình 12

3.6. Liên lạc giữa hai tiến trình 13

3.7. Lập lịch đa tiến trình 15

3.8 Giám sát và điều khiển các tiến trình 17

**KẾT LUẬN** 21

**MỞ ĐẦU**

Vài năm qua, Linux đã thực sự tạo ra một cuộc cách mạng trong lĩnh vực máy tính. Sự phát triển và những gì chúng mang lại cho máy tính thật đáng kinh ngạc: một hệ điều hành đa nhiệm, đa người dùng. Linux có thể chạy trên nhiều bộ vi xử lý khác nhau như: Intel, Motorola, MC68K, Dec Alpha. Nó tương tác tốt với các hệ điều hành: Apple, Microsoft và Novell. Không phải ngẫu nhiên mà ngành công nghệ thông tin Việt Nam chọn Linux làm hệ điều hành nền cho các chương trình ứng dụng chủ đạo về kinh tế và quốc phòng. Với mã nguồn mở, sử dụng Linux an toàn hơn các ứng dụng Windows. Linux đem đến cho chúng ta lợi ích về kinh tế với rất nhiều phần mềm miễn phí. Mã nguồn mở của hệ điều hành và của các chương trình trên Linux là tài liệu vô giá để chúng ta học hỏi về kỹ thuật lập trình vốn là những tài liệu không được công bố đối với các ứng dụng Windows.

Trong đồ án này, chúng em sẽ tìm hiểu một phần rất quan trọng trong hệ điều hành Linux đó là: quản lý tiến trình trong Linux. Qua quá trình nghiên cứu, tự tìm hiểu cùng với sự giúp đỡ của giáo viên hướng dẫn chúng em đã biết tạo tiến trình, dừng tiến trình, hiểu được nguyên tắc lập lịch, liên lạc, giao tiếp, giám sát và điều khiển các tiến trình trong hệ điều hành Linux.

Do kiến thức của bản thân còn hạn chế và chưa có kinh nghiệm thực tiễn nên nội dung bài nghiên cứu khó tránh những thiếu sót. Chúng em rất mong nhận được ý kiến đóng góp của thầy, cô giáo và những người quan tâm đến đề tài này để đề tài này của chúng em được hoàn thiện hơn nữa. Chúng em xin chân thành cám ơn!

*Chúng em xin chân thành cảm ơn!*

**CHƯƠNG 1**

**GIỚI THIỆU VỀ HỆ ĐIỀU HÀNH LINUX**

**1.1. Tổng quan về hệ điều hành.**

Hệ điều hành là một phần mềm chạy trên máy tính, dùng để điều hành, quản lí các thiết bị phần cứng và các tài nguyên phần mềm trên máy tính. Nó là phần căn bản nhất của tất cả các phần mềm hệ thống.

Hệ điều hành đóng vai trò trung gian giao tiếp giữa người sử dụng với phần cứng máy tính, cung cấp một môi trường cho người sử dụng và phát triển các ứng dụng của họ một cách dễ dàng.

Hệ điều hành là một phần quan trọng của hầu hết các hệ thống máy tính. Hệ điều hành chạy trong môi trường đặc biệt, gọi là chế độ nhân (Kernel mode hay Supervisor mode). Chế độ chạy này được hỗ trợ biển kiến trúc của CPU (bởi các lệnh máy đặc biệt) và nó ngăn người dùng truy cập vào phần cứng (quản lí phần cứng chuẩn xác cho nhiều người dùng đồng thời, còn gọi là chế độ được bảo vệ (protect mode)).

***Chức năng của hệ điều hành***

- Quản lí quá trình (Proccess manament)

- Quản lí bộ nhớ (Memory manament)

- Quản lí hệ thống lưu trữ

- Giao tiếp với người dùng (User interaction).

***Nhiệm vụ của hệ điều hành:***

+ Điều khiển quản lí trực tiếp các phần cứng như bo mạch chủ, bo mạch đồ họa và bo mạch âm thanh….

+ Thực hiện một số thao tác cơ bản trong máy tính như các thao tác đọc, viết tập tin, quản lí hệ thống tập tin (file system) và các kho dữ liệu.

+ Cung cấp giao diện cho người dùng, giao diện là môi trường giao tiếp cho phép người sử dụng trao đổi trong qua trình sử dụng.

+ Cung ứng một hệ thống lệnh cơ bản để điều hành máy. Các lệnh này gọi là lệnh hệ thống (system command).

+ Ngoài ra hệ điều hành, trong vài trường hợp cũng cung cấp các dịch vụ cơ bản cho các phần mềm ứng dụng thông thường như chương trình duyệt web hay soạn thảo văn bản….

***Các thành phần của hệ điều hành bao gồm:***

- Hệ thống quản lí tiến trình.

- Hệ thống quản lí bộ nhớ.

- Hệ thống quản lí nhập xuất.

- Hệ thống quản lí tập tin.

- Hệ thống bảo vệ.

- Hệ thống dịch lệnh.

- Quản lí mạng.

**1.2. Hệ điều hành Linux**

Linux là tên gọi của một hệ điều hành máy tính và cũng là tên hạt nhân của hệ điều hành. Đây có lẽ là một ví dụ nổi tiếng nhất của phần mềm tự do và của việc phát triển mã nguồn mở.

Phiên bản Linux đầu tiên do Linus Torvalds viết vào năm 1991, lúc ông còn là một sinh viên của Đại học Helsinki tại Phần Lan. Ông làm việc một cách hăng say trong vòng 3 năm liên tục và cho ra đời phiên bản Linux 1.0 vào năm 1994. Bộ phận chủ yếu này được phát triển và tung ra trên thị trường dưới bản quyền GNU General Public License. Do đó mà bất cứ ai cũng có thể tải và xem mã nguồn của Linux. “Linux” được sử dụng để chỉ Nhân Linux, nhưng tên này được sử dụng một cách rộng rãi để miêu tả tổng thể một hệ điều hành giống Unix (còn được biết dưới tên GUN/Linux) được tạo ra bởi việc đóng gói nhân Linux cùng với các thư viện và công cụ GNU, cũng như là các bản phân phối Linux.



*Torvalds – Người “cha đẻ” của Linux*.

Khởi đầu Linux được phát triển cho dòng vi xử lý 386. Hiện tại hệ điều hành này hỗ trợ một số lượng lớn các siêu thị máy tính và các thiết bị nhúng như là các máy điện thoại di động.

Ban đầu, Linux được phát triển và sử dụng bởi những người say mê. Tuy nhiên, hiện nay Linux đã có được sự hỗ trợ bởi các công ty lớn như IBM và Hewlett-Packard, đồng thời nó cũng bắt kịp được các phiên bản được các phiên bản Unix độc quyền và thậm chí là một thách thức đối với sự thống trị của Microsoft Windows trong một số lĩnh vực. Sở dĩ Linux đạt được những thành công một cách nhanh chóng là nhờ vào các đặc tính nổi bật so với các hệ thống khác: chi phí phần cứng thấp, tốc độ cao (khi so sánh với các phiên bản Unix độc quyền) và khả năng bảo mật tốt, độ tin cậy cao (khi so sánh với Windown) cũng như các đặc điểm về giá thành rẻ, không bị phụ thuộc vào nhà cung cấp. Một đặc tính nổi trội của nó là được phát triển bởi một mô hình phát triển phần mềm nguồn mở hiệu quả.



*Chim cánh cụt Tux- biểu trưng và vật may mắn của Linux.*

Tuy nhiên, hiện tại số lượng phần cứng được hỗ trợ bởi Linux vẫn còn rất khiêm tốn so với Windown vì các trình điều khiển thiết bị tương thích với Window nhiều hơn là Linux. Nhưng trong tương lai số lượng phần cứng được hỗ trợ cho Linux sẽ tăng lên. Hiện nay, Linux có nhiều bản phân phối khác nhau, một phần là bời vì mã nguồn mở của nó.

**CHƯƠNG 2**

**TỔNG QUAN VỀ TIẾN TRÌNH**

**2.1. Khái niệm về tiến trình**

Để hỗ trợ hoạt động đa nhiệm, hệ thống máy tính cần phải có khả năng thực hiện nhiều tác vụ xử lí đồng thời những việc điều khiển hoạt động song hành ở cấp độ phần cứng là rất khó khăn. Vì vậy các nhà thiết kế hệ điều hành đề xuất một mô hình song hành giả lập bằng cách chuyển đổi bộ xử lí qua lại giữa các chương trình để duy trì hoạt động của nhiều chương trình tại cùng một thời điểm. trong mô hình này, các chương trình trong hệ thống được tổ chức thành các tiến trình (process).

Như vậy, có thể coi *tiến trình* là một chương trình đang xử lí, nó sở hữu 1 con trỏ lệnh, tập các thanh ghi và các biến. Để hoàn thành nhiệm vụ của mình các tiến trình còn yêu cầu một số tài nguyên hệ thống như CPU, bộ nhớ và các thiết bị vào/ra...

**2.2. Các trạng thái của một tiến trình.**

Trong môi trường hệ điều hành Linux, một tiến trình có các trạng thái sau:

- Khởi tạo (new): Tiến trình đang được tạo lập

- Sẵn sàng (chờ): Tiến trình chờ cấp phát CPU để xử lí.

- Thực hiện (running): Tiến trình được xử lý

- Đợi (waiting): tiến trình phải dừng vì thiếu tài nguyên hoặc chờ 1 sự kiện

nào đó.

- Kết thúc (halt): tiến trình đã hoàn tất công việc cần xử lý.

**2.3. Quan hệ giữa các tiến trình.**

Các tiến trình hoạt động trong hệ thống tồn tại 2 mối quan hệ: độc lập và hợp tác (song hành).

**\*** *Quan hệ độc lập*: tiến trình được gọi là độc lập nếu hoạt động của nó không gây ảnh hưởng hoặc không bị ảnh hưởng của các tiến trình khác cũng đang hoạt động của hệ thống.

Tiến trình độc lập có những đặc trưng sau:

* Trạng thái của nó không bị chia sẻ với bất kì tiến trình nào khác.
* Việc thực hiện tiến trình là đơn định (kết quả chỉ phụ thuộc vào đầu vào).
* Tiến trình có thể tái hiện (lặp lại).
* Tiến trình có thể dừng hoặc bắt đầu lại mà không gây ảnh hưởng tới các tiến trình khác trong hệ thống.

***\**** *Quan hệ hợp tác*: tiến trình được gọi là hợp tác (song hành) nếu hoạt động của nó gây ảnh hưởng hoặc bị ảnh hưởng bởi các tiến trình khác cũng đang hoạt động trong hệ thống.

Tiến trình hợp tác có những đặc trưng sau:

* Trạng thái của nó bị chia sẻ cho các tiến trình khác.
* Việc thực hiện tién trình không đơn điệu (kết quả của tiến trình phụ thuộc vào dãy thực hiện tương ứng và không dự báo trước).

- Tiến trình không thể tái hiện.

**2.4. Phương pháp tổ chức liên lạc giữa các tiến trình**

Hệ điều hành xây dựng một hệ thống thông báo giữa các tiến trình dựa trên ba thao tác : Send message, receive message, communication link.

Dựa vào các thông báo này,các tiến trình có thể phối hợp để vào đoạn tới hạn bằng cách trao đổi thông báo cho nhau. Hệ thông báo sử dụng hai cách liên lạc :

+ Trực tiếp: bằng cách sửa hai thao tác gửi và nhận

+ Gián tiếp: dùng hộp thư trung gian

**CHƯƠNG 3**

**QUẢN LÝ TIẾN TRÌNH LINUX**

**3.1 Các loại tiến trình chính trên Linux:**

Trên hệ điều hành Linux, tiến trình được định danh bởi một PID (tiến trình ID) phục vụ cho việc giám sát và quản lý của hệ thống. Các loại tiến trình chính trên Linux :

*Tiến trình tương tác (Interactive processes)* là tiến trình khởi động và quản lý bởi shell, kể cả tiến trình forceground hay background.

*Tiến trình theo lô (Batch processes)* là tiến trình không gắn liền với bàn điều khiển (terminal) và được nằm trong hàng đợi để lần lượt thực hiện.

*Tiến trình ẩn trên bộ nhớ (Daemon process)* là các tiến trình chạy ẩn bên dưới hệ thống (background). Các tiến trính này được khởi động từ đầu. Đa số các chương trình server cho các dịch vụ chạy theo phương thức này. Đây là các chương trình này được gọi lên bộ nhớ, đợi (thụ động) các yêu cầu từ chương trình khách (client) để trả lời các socket xác định, tên của nó thường được kết thúc bằng ký tự “d”.

Khi một tiến trình sinh ra 1 tiến trình khác thì:

- Tiến trình ban đầu được gọi là tiến trình cha và được định danh bởi PPID

(Parent PID).

- Tiến trình mới được gọi là tiến trình con.

Tương tác giữa tiến trình cha và tiến trình con :

- Khi tiến trình con đang chạy thì tiến trình cha sẽ chờ.

- Khi tiến trình con hoàn thành thì tiến trình cha sẽ tiếp tục thực thi và tiến

trình con sẽ được kết thúc.

**3.2. Tại sao phải quản lý tiến trình.**

Trong hệ thống luôn tồn tại nhiều luồng tiến trình. Mặt khác trong hệ thống có những tài nguyên hữu hạn khả năng phục vụ nhưng trong một khoảng thời gian nào đó có nhiều tiến trình muốn sử dụng tài nguyên đó.

=> Dẫn đến xung đột, bế tắc xảy ra, giao thông trong máy bị đình trệ dẫn đến treo máy. Vì vậy cần quản lý tiến trình.

*\* Nhiệm vụ của quản lý tiến trình:*   
 - Tạo lập, hủy bỏ tiến trình.   
 - Tạm dừng, tái kích hoạt tiến trình.   
 - Tạo cơ chế thông tin liên lạc giữa các tiến trình.  
 - Tạo cơ chế đồng bộ hóa giữa các tiến trình.   
*\* Mục tiêu:*    
 - Hạn chế tối đa xung đột và bế tắc xảy ra, đưa ra giải pháp nếu xảy ra các

tình huống đó.  
 - Tận dụng tối đa khả năng của CPU (bài toán lập lịch).

|  |
| --- |
| **3.3. Tạo một tiến trình.**  Tiến trình con được tạo ra bằng system call fork(), có prototype như sau:    System call fork() làm việc bằng cách tạo ra 1 tiến trình mới với PID mới, và nhân bản dữ liệu từ tiến trình cha sang tiến trình con  Giá trị trả lại là 0 cho tiến trình con và dấu hiệu pid cho tiến trình cha. Giá trị sẽ là -1 nếu không tạo được tiến trình mới. Vì vậy, tiến trình con sẽ kế thừa toàn bộ các biến (bao gồm cả biến môi trường), giá trị hiện tại của các biến, các mô tả file và stack frame của tiến trình cha. Sau lời gọi fork(), 2 tiến trình sẽ đồng thời tồn tại và chúng sẽ tiếp tục chạy sau thời điểm fork() return.  Sau fork() cả 2 tiến trình cha và con đồng thời đang tồn tại, với tiến trình cha, fork() sẽ return 1 số nguyên dương là PID của tiến trình con mà nó vừa tạo ra. Với tiến trình con, fork() sẽ return giá trị 0, tiến trình con có thể lấy dược PID của mình bằng system call getpid(). System call fork() trả về -1 khi bị lỗi.      **3.4. Dừng một tiến trình.**  Kết thúc một tiến trình có thể được thực hiện theo vài cách khác nhau. Thông thường, từ một lệnh console-based, gửi CTRL + C bằng gõ phím (mặc định là ký tự ngắt) sẽ hủy lệnh. Nó làm việc khi tiến trình đang chạy trong chế độ Foreground.  Nếu một tiến trình đang chạy trong chế độ Background, thì đầu tiên bạn cần nhận ID công việc (job ID) bằng cách sử dụng lệnh ps và sau đó bạn có thể sử dụng lệnh **kill** để dừng tiến trình như sau:  $ps -f  UID PID PPID C STIME TTY TIME CMD  amrood 6738 3662 0 10:23:03 pts/6 0:00 first\_one  amrood 6739 3662 0 10:22:54 pts/6 0:00 second\_one  amrood 3662 3657 0 08:10:53 pts/6 0:00 -ksh  amrood 6892 3662 4 10:51:50 pts/6 0:00 ps -f  $kill 6738  Terminated    Có hai lệnh **kill** đáng chú ý nhất :  + **kill -9** : kết thúc tiến trình ngay lập tức.  + **kill -15** : gửi tín hiệu kết thúc tiến trình chờ tiến trình thực hiện cleanup và kết thúc (Trong một số trường hợp kill -15 không thể tắt được tiến trình, buộc phải dùng kill -9 để dừng tiến trình).  $kill -9 6738  Terminated    Ngoài ra, để kết thúc một hoặc nhiều tiến trình theo tên hoặc thuộc tính của tiến trình ta dùng lệnh **pkill.**  VD : **kill** tất cả các tiến trình Java trên máy chủ :    **3.5. Giao tiếp giữa các tiến trình.**  Linux cung cấp một số cơ chế giao tiếp giữa các tiến trình gọi là IPC (Inter-Process Communication):  + Signals handling – Trao đổi bằng tín hiệu  + Message Queues – Trao đổi thông qua hàng đợi tin nhắn  **3.5.1. Signals handling (Trao đổi bằng tín hiệu)**  Tín hiệu được sử dụng để báo hiệu các sự kiện không đồng bộ cho một hoặc nhiều tiến trình. Mỗi tín hiệu có thể kết hợp hoặc có sẵn bộ xử lý tín hiệu (signal handler). Tín hiệu sẽ ngắt ngang quá trình xử lý của tiến trình, bắt hệ thống chuyển sang gọi bộ xử lý tín hiệu ngay tức khắc. Khi kết thúc xử lý tín hiệu, tiến trình lại tiếp tục thực thi.  ***\**** *Gửi tín hiệu tới các tiến trình*  - Các nguồn gửi signals : phần cứng, từ kernel  - Các cách gửi tín hiệu gửi tín hiệu đến tiến trình :  Từ bàn phím : Ctrl+C(gửi tín hiệu INT( SIGINT ) đến tiến trình, ngắt ngay tiến trình (interrupt)), Ctrl+Z(gửi tín hiệu TSTP( SIGTSTP ) đến tiến trình, dừng tiến trình (suspend)), Ctrl+/(gửi tín hiệu ABRT( SIGABRT ) đến tiến trình, kết thúc ngay tiến trình (abort))  Từ dòng lệnh : Lệnh kill được dùng để dừng một tiến trình. Nó có thể gửi bất kỳ tín hiệu signal nào tới một tiến trình, nhưng theo mặc định nó gởi tín hiệu 15, TERM (là tín hiệu kết thúc chương trình).  *\* Nhận và xử lý tín hiệu*  - Khi một tiến trình nhận một tín hiệu, nó có thể xử sự theo một trong các cách sau: Bỏ qua tín hiệu, Xử lý tín hiệu theo kiểu mặc định, Tiếp nhận tín hiệu và xử lý theo cách đặc biệt của tiến trình.  **3.5.2.** **Mesage Queue (Trao đổi thông qua hàng đợi tin nhắn)**  Message queue là cơ chế IPC có sẵn trong Linux. Mỗi một khối dữ liệu được truyền đi được xác định một kiểu (TYPE) cụ thể và người nhận có thể nhận được các dữ liệu đó tùy theo kiểu của dữ liệu.   * msgget (): trả về ID định danh cho message queue mới được tạo hoặc trả về định danh của một message queue đã tồn tại cùng với một giá trị khóa nhận dạng của nó. * msgsnd (): Hàm này được sử dụng để đưa dữ liệu vào message queue. * msgrcv (): Hàm này giúp lấy dữ liệu ra khỏi một message queue. * msgctl (): Hàm này cũng giúp thực hiện thao tác điều khiển một message queue.   **3.6. Liên lạc giữa hai tiến trình.**  Ống dẫn liên lạc là một cơ chế cơ bản để liên lạc gián tiếp giữa các tiến trình. Đó là các file đặc biệt (FIFO), ở đó các thông tin được truyền đi 1 đầu và thoát ra ở một đầu khác. Một số đặc điểm của "ống dẫn":  **-** Các ống dẫn chỉ mang tính chất tạm thời, chỉ tồn tại trong thời gian thực hiện của  một tiến trình tạo ra nó.  **-**  Muốn tạo ra một ống dẫn phải bắt đầu bằng một lệnh đặc biệt: pipe().  **-**  Nhiều tiến trình có thể viết và đọc trên cùng một ống dẫn. Tuy nhiên, không có  một cơ chế nào để phân biệt thông tin cho các tiến trình ở đầu ra.  **-** Dung lượng ống dẫn bị hạn chế (khoảng 4KB). Do đó khi chúng ta cố gắng viết  khi ống dẫn bị đầy thì sẽ gặp phải trường hợp tắc nghẽn.  **-** Các tiến trình liên lạc qua ống dẫn phải có mối quan hệ họ hàng và các ống dẫn nối  phải được mở trước khi tạo ra các tiến trình con.  **-** Không thể tự thay đổi vị trí thông tin trong ống.  **Thao tác với "ống dẫn liên lạc".**  ● Tạo một ống dẫn:  int p\_desc[2];  int pipe(p\_desc);   * Giá trị trả về là 0 nếu thành công, -1 nếu thất bại.   p\_desc[0] : chứa các số hiệu mô tả nhờ đó có thể đọc trong ống dẫn.  p\_desc[1] : chứa các số hiệu mô tả nhờ đó có thể viết trong ống dẫn.   * Như vậy, việc viết trong p\_desc[1] là để truyền dữ liệu trong ống và việc   đọc trong p\_desc[0] để nhận chúng.  **Ví dụ:**  #include  #include  main(){  int i, ret, p\_desc[2];  char c;  pipe(p\_desc);  write(p\_desc[1], "AB", 2);  for (i=1; i<=3; i ++){  ret=read(p\_desc[0], &c, 1);  if (ret == 1)  printf(" Gia tri: %c\n", c);  else  perror("Loi ong dan rong");  }  }  Ví dụ trên chỉ ra rằng ta có thể truyền và nhận thông tin trên ống dẫn. Chúng ta đã dùng hàm read() và write() để viết (truyền) và đọc (nhận) trên ống dẫn.  **3.7. Lập lịch đa tiến trình.** |
| Cơ chế lập lịch cho tiến trình của Linux được thực hiện bởi 2 thông số ứng với mỗi tiến trình:  • process priority (chỉ số ưu tiên của tiến trình)  • scheduling policy (giúp hệ điều hành quyết định xem tiến trình được chạy  bao lâu và khi nào phải nhường quyền chạy cho tiến trình khác).  Hệ điều hành Linux có các scheduling policy cho việc hỗ trợ:  • Lập lịch cho real-time process : SCHED\_FIFO, SCHED\_RR  • Lập lịch cho normal process: SCHED\_NORMAL, SCHED\_IDLE và  SCHED\_BATCH.  **3.7.1. Scheduling policy cho real-time process**  **3.7.1.1. SCHED\_RR (round-robin):**  Các tiến trình chạy tuân theo cơ chế round-robin (time-sharing): mỗi tiến trình được cấp một khoảng thời gian (hay khe thời gian) cố định được gọi là time slice. Khi một tiến trình chạy, nó sẽ chiếm giữ CPU cho đến khi xảy ra một trong các sự kiện sau:  + Thời gian tiến trình chạy bằng time slice được cấp cho nó.  + Tiến trình tự động nhường CPU, trường hợp này xảy ra khi tiến trình gọi một blocking system call hoặc gọi system call sched\_yield().  + Tiến trình bị kết thúc.  + Tiến trình bị chiếm quyền bởi một tiến trình có độ ưu tiên cao hơn.  Với trường hợp đầu, tiến trình sau khi bị mất quyền chiếm giữ CPU sẽ bị đẩy vào cuối của queue với cùng các tiến trình cùng priority khác. Với trường hợp 4, tiến trình bị chiếm quyền được đẩy vào đầu của queue, sau khi tiến trình độ ưu tiên cao hơn chạy hết time slice, nó sẽ tiếp tục thực thi cho đến khi đi hết time slice của mình.  **3.7.1.2. SCHED\_FIFO (first in- first out):**  Chính sách SCHED\_FIFO khác SCHED\_RR ở điểm là không sử dụng time slice. Theo đó tiến trình đang chiếm dụng CPU sẽ tiếp tục chạy cho đến khi có một trong các sự kiện sau:  -Tiến trình tự động nhường CPU, tương tự như chính sách SCHED\_RR ở trên.  -Tiến trình bị kết thúc.  -Tiến trình bị chiếm quyền bởi một tiến trình có độ ưu tiên cao hơn.  Với trường hợp đầu, tiến trình sau khi bị chiếm quyền sẽ bị đẩy vào cuối queue của các tiến trình cùng priority. Với trường hợp 3, tiến trình bị chiếm quyền được đẩy vào đầu của queue, sau khi tiến trình có độ ưu tiên cao hơn dừng chiếm dụng CPU, nó sẽ tiếp tục thực thi.  **3.7.2. Scheduling policy cho normal process**  **3.7.2.1. SCHED\_NORMAL:**  SCHED\_NORMAL (còn gọi SCHED\_OTHER) là scheduling policy mặc định cho các tiến trình không đòi hỏi cơ chế real-time (normal process). SCHED\_NORMAL policy tuân thủ cơ chế hoạt động chia sẻ thời gian (time-sharing) round-robin tương tự chính sách SCHED\_RR.  Điểm khác biệt là SCHED\_NORMAL không fix cứng time slice cho mỗi tiến trình mà độ dài time slice của tiến trình thay đổi theo tính toán của bộ lập lịch của Linux kernel. Cụ thể, time slice sẽ phụ thuộc vào 2 yếu tố: nice value (nice value càng thấp (ví dụ -20) thì time slice càng cao) và thời gian chờ của tiến trình (tiến trình ở trạng thái runable mà đang chờ đến lượt chạy càng lâu thì time slice càng nhiều). Cơ chế tính time slice này đảm bảo độ công bằng (fair) giữa tất cả các normal process trong hệ thống.  **3.7.2.2. SCHED\_BATCH**  Scheduling policy SCHED\_BATCH được đưa vào Linux kerne từ version 2.6.16, có thể dùng để lập lịch cho các batch process. Chính sách **này tương tự như SCHED\_NORMAL, chỉ khác một điểm là các tiến trình SCHED\_BATCH sẽ ít được đánh thức để xem xét lập lịch hơn các tiến trình SCHED\_NORMAL.**  **3.7.2.3. SCHED\_IDLE**  **Scheduling policy SCHED\_IDLE được đưa vào Linux kernel từ version 2.6.23. Chính sách lập lịch này cũng tương tự như policy SCHED\_NORMAL nhưng làm cho tiến trình có độ ưu tiên (tương đương với nice value) rất thấp. Nice value không có ý nghĩa gì với policy này. Có thể nói các tiến trình có scheduling policy này có độ ưu tiên thấp hơn các tiến trình SCHED\_NORMAL thông thường, nó chỉ được lập lịch khi không còn normal process nào yêu cầu chạy.** |

**3.8. Giám sát và điều khiển các tiến trình.**

Để giám sát và điều khiển các tiến trình ta sử dụng một số câu lệnh cơ bản như sau: ps, pstree, top

***•*** **#ps <option>:** liệt kê các tiến trình hiện đang chạy trên hệ thống, với nhiều chế độ lọc và hiển thị có sẵn thông qua flag và đối số.



Một số lệnh **ps** hay sử dụng:

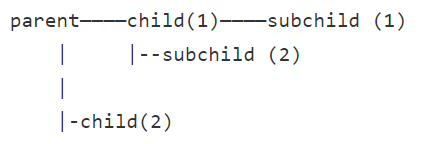
|  |  |
| --- | --- |
| **Lệnh ps** | **Công dụng** |
| ps -e -u userName | Lọc theo người dùng |
| ps -C name | Lọc theo tên tiến trình |
| ps -ef -p 1234,5678,9012 | Lọc theo ID tiến trình |
| ps -ef | Hiển thị tất cả các tiến trình |
| ps -e -forest | Hiển thị kết quả theo kiểu cây phân cấp |
| ps -e -o pid,uname,pcpu,pmem,comm | Hiển thị các cột cụ thể |
| p-o pid=Process,ruser=RealUser,comm=Command | Đổi tên tiêu đề cột |
| ps -f -U root -u root | Hiển thị tất cả các tiến trình root |
| ps -e --sort=-pcpu -o pid,pcpu,comm | Sắp xếp các tiến trình theo cách sử dụng |

**Ý nghĩa các trường:**

****

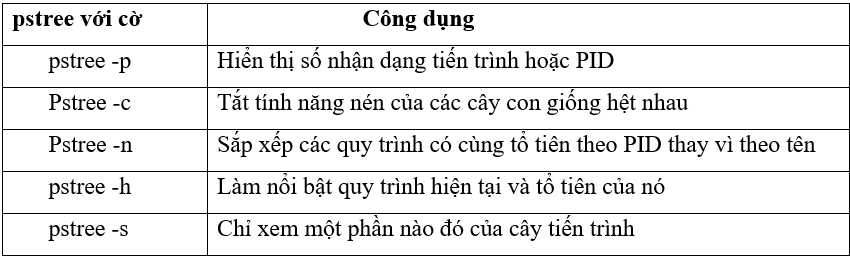
**• #Pstree <option>:** tương tự như ps, nhưng thay vì liệt kê các tiến trình đang chạy, nó hiển thị chúng trong một cây.

Cấu trúc cơ bản của pstree:

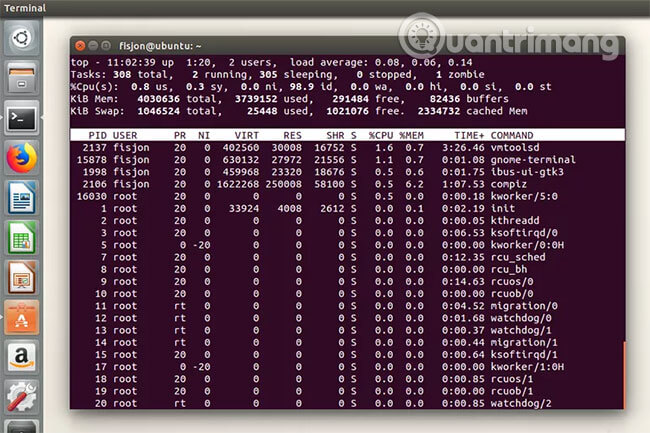




Sử dụng **pstree** với một số cờ :



**• Top:** hiển thị trạng thái của các tiến trình đang hoạt động trên hệ thống.

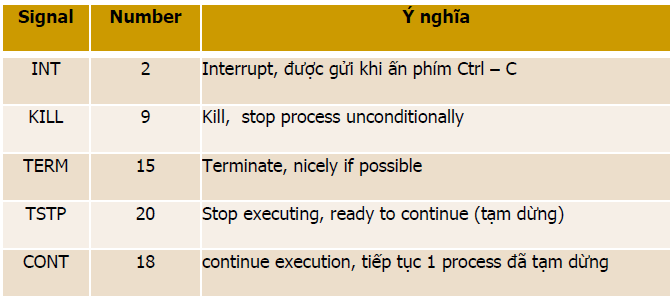


Các thông số chính của lệnh **top :**

|  |  |
| --- | --- |
| **Thông số** | **Ý nghĩa** |
| PID | tiến trình ID của các tiến trình |
| User | User thực thi tiến trình |
| %CPU | Phần trăm CPU mà tiến trình sử dụng |
| %Mem | Dung lượng RAM |
| Time+ | Thời gian hoạt động của tiến trình từ khi bật |
| Command | Chương trình chạy |

**● Gửi tín hiệu cho một tiến trình đang chạy**

**# kill <signal | number> <PID >**



**● Thay đổi thông số Priority**

Sử dụng lệnh **nice**, **renice**

**Nice**: dùng để thay đổi nice number của các process tại thời điểm start time

**# nice [–n number] [command]**

Ví dụ: # nice –n -10 vi /root/data.txt

**Renice**: Thay đổi thông số nice number của các process đã chạy

**# renice priority PID [[-g] group] [[-u] user]**

Ví dụ: # renice -2 203 Set nice number is -2 to PID=203

**● Can thiệp vào hoạt động**

&: Cho một job hoạt động ở background.

Ví dụ: # ls –l –R / > /root/list.txt & Ứng dụng ls sẽ chạy nền bên dưới.

**● Ngưng và tạm ngưng Job**

Ctrl C: Ngưng job đang thực thi. Sau khi ấn Ctrl C ta có thể dùng câu lệnh jobs để hiển thị trạng thái của các tiến trình đang chạy.

Ctrl Z: Tạm ngưng job đang thực thi. Sau khi ấn Ctrl Z ta có thể dùng 2 câu lệnh:

bg: tiếp tục job vừa ngưng ở trạng thái background.

fg: tiếp tục job vừa ngưng ở trạng thái foreground.

**KẾT LUẬN**

Sau đề tài này, chúng em đã nắm được các trạng thái của một tiến trình, cách quản lí các tiến trình thông qua bộ mô tả tiến trình cũng như các mối quan hệ giữa các tiến trình. Qua đó, tìm hiểu cụ thể cách tạo ra và kết thúc một tiến trình.

TÀI LIỆU THAM KHẢO

1. Giáo trình Nguyên lý hệ điều hành; Khoa CNTT Trường Đại học Công nghiệp Hà Nội, 2016.
2. Hà Quang Thụy, Nguyễn Chí Thành; Giáo trình hệ điều hành UNIX-LINUX; NXB Đại học Quốc Gia Hà Nội, 2004.

[1] <https://viblo.asia/p/quan-ly-tien-trinh-trong-linux-YWOZrynvKQ0>

[2] <http://www.cit.ctu.edu.vn/~dtnghi/linux/6-quan_ly_tien_trinh.pdf>