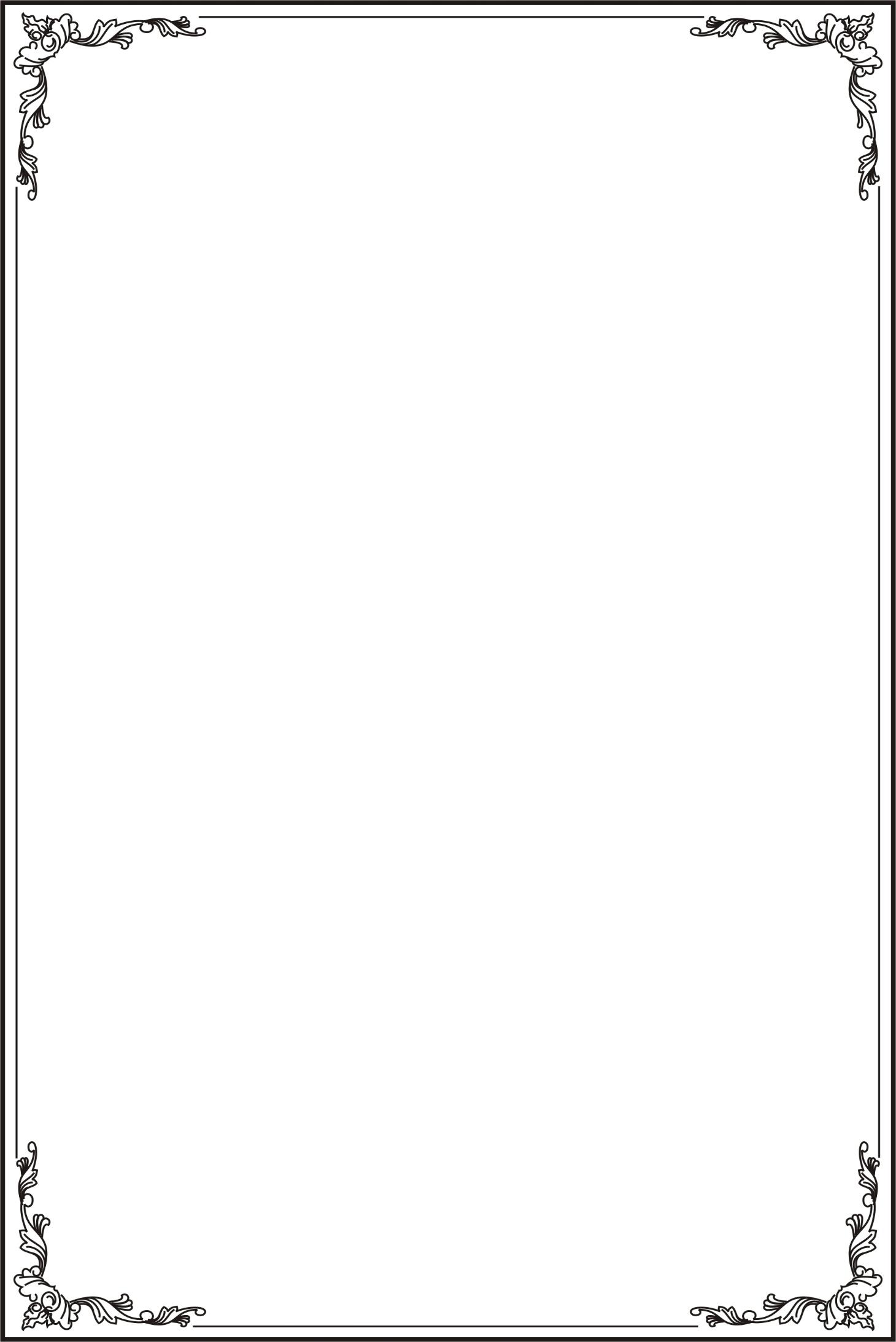
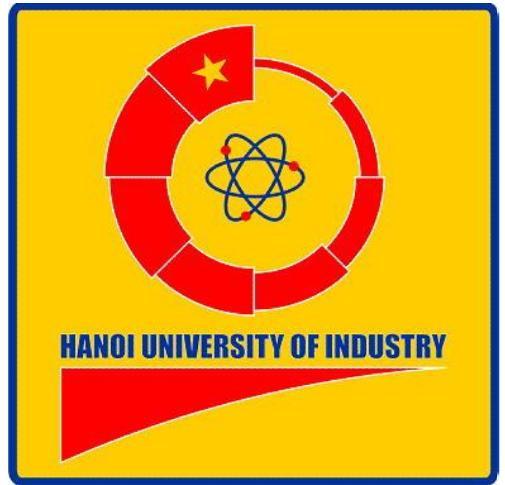
**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI\**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----



BÀI TẬP LỚN

MÔN HỌC: NGUYÊN LÝ HỆ ĐIỀU HÀNH

**ĐỀ TÀI: NGHIÊN CỨU TÌM HIỂU VỀ QUẢN LÝ TIẾN TRÌNH TRONG HĐH LINUX**

**Giảng viên hướng dẫn: Nguyễn Bá Nghiễn**

**Sinh Viên: Hoàng Minh Đức(NT)**

**Nguyễn Văn Dũng**

**Lê Nguyên Giáp**

**Hoàng Đức Hiếu**

**Dương Thế Đoàn**

**Nhóm thực hiện: Nhóm 1**

**Lớp: IT6025.4**

***Hà Nội, 2022***

**MỤC LỤC**

[**1.**](#_heading=h.3znysh7) **Giới thiệu về hệ điều hành linux** 3

[**1.1**](#_heading=h.2et92p0) **Tổng quan về hệ điều hành.** 3

[**1.2**](#_heading=h.tyjcwt) **Hệ điều hành linux.** 4

[**2.**](#_heading=h.3dy6vkm) **Khái niệm về tiến trình.** 7

[**3.**](#_heading=h.1t3h5sf) **Các trạng thái của một tiến trình** 8

[**4.**](#_heading=h.4d34og8) **Quan hệ giữa các tiến trình** 9

[**5.**](#_heading=h.2s8eyo1) **Quản lý tiến trình trên linux** 10

[**5.1**](#_heading=h.17dp8vu) **Tại sao phải quản lý tiến trình** 10

[**5.2**](#_heading=h.1y810tw) **Tạo một tiến trình** 10

[**5.3**](#_heading=h.2xcytpi) **Dừng một tiến trình** 11

[**5.4**](#_heading=h.1pxezwc) **Giao tiếp giữa các tiến trình** 11

[**5.5**](#_heading=h.46r0co2) **Liên lạc giữa hai tiến trình** 13

[**5.6**](#_heading=h.2250f4o) **Lập lịch đa tiến trình** 15

[**5.7**](#_heading=h.243i4a2) **Liên lạc giữa tiến trình cha và tiến trình con** 17

Lời 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 ta sẽ tìm hiểu một phần rất quan trọng trong hệ điều hành Linux đó là: quản lý bộ nhớ trong Linux. Một hệ điều hành muốn chạy ổn định thì phải có một cơ chế quản lý bộ nhớ hiệu quả. Cơ chế này sẽ được trình bày một cách chi tiết trong đồ án và có kèm theo các chương trình minh họa.

Qua đề tài này, chúng em đã tiếp thu được nhiều kiến thức cơ bản về quản lý tiến trình trong HĐH Linux. Bởi vì lượng kiến thức còn thiếu sót, đề tài này vẫn còn nhiều thiếu sót, mong thầy cô và các nhóm góp ý để đề tài của chúng em hoàn thiện hơn. Nhóm chúng em xin trân thành cảm ơn, những góp ý của Thầy Nguyễn Bá Nghiễn trong quá trình chúng em hoàn thành đề tài này.

Trân trọng cảm ơn

***Nhóm thực hiện đề tài***

# 1. Giới thiệu về hệ điều hành linux

**1.1. Tổng quan về hệ điều hành.**

* Phần mềm máy tính chia ra làm hai loại: đó là các phần mềm hệ thống, quản lý hoạt động của bản thân máy tính, và các chương trình ứng dụng, giải quyết các yêu cầu của người dùng.
* Phần căn bản nhất của tất cả các phần mềm hệ thống gọi là Hệ điều hành.
* Hệ điều hành là một phần mềm chạy trên máy tính, dùng để điều hành, quản lý các thiết bị phần cứng và các tài nguyên phần mềm trên máy tính.
* 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 lý phần cứng chuẩn xác cho nhiều người dùng đồng thời, còn gọi là chế độ được bảo vệ (protect mode)).
* Chức năng chủ yếu của hệ điều hành đó là: Quản lí chia sẻ tài nguyên (CPU, bộ nhớ trong, bộ nhớ ngoài…) và Giả lập một máy tính mở rộng.
* Ngoài ra còn có thể chia chức năng của hệ điều hành theo bốn chức năng là:
* Quản lý quá trình (Process management)
* Quản lý bộ nhớ (Memory management)
* Quản lý hệ thống lưu trữ
* Giao tiếp với người dùng (User interaction).
* Nhiệm vụ của hệ điều hành:
* Điều khiển quản lý 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 lý hệ thống tập tin (file system) và các kho dữ liệu.
* Cung ứng một hệ thống giao diện sơ khai cho các ứng dụng thường là thông qua một hệ thống thư viện hàm chuẩn để hệ điều hành các phần cứng mà từ đó các ứng dụng có thể gọi tới.
* 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 lý tiến trình.
* Hệ thống quản lý bộ nhớ.
* Hệ thống quản lý nhập xuất.
* Hệ thống quản lý tập tin.
* Hệ thống bảo vệ.
* Hệ thống dịch lệnh.
* Quản lý mạng.

**1.2. Hệ điều hành linux.**

- Linux là hệ điều hành mô phỏng Unix, được xây dựng trên phần nhân (kernel) và các gói phần mềm mã nguồn mở. Linux được công bố dưới bản quyền của GPL (General Public Licence).

- Linux cũng là một hệ điều hành cho nhiều nền máy tính khác nhau nhưng trước tiên là cho PC nền Intel. Đó là hệ điều hành có hàng trăm nhà lập trình tham gia thiết kế và xây dựng, với mục tiêu là tạo ra một bản giống UNIX hoàn toàn không lệ thuộc vào phần mềm có đăng ký tác quyền nào và cả thế giới có thể sử dụng thoải mái.

- Thành phần cơ bản của Linux là hạt nhân Linux ( thường được gọi là nhân Linux-Linux kernel), là nhân hệ điều hành được phát triển bởi Linus Torvalds được công bố lần đầu tiên vào 9/1991 với phiên bản 0.0.1.

- Ngày 5/4/1991, Linus Torvalds, chàng sinh viên trường Đại học Helsinki, Phần Lan đã bắt tay vào viết những dòng lệnh đầu tiên của Linux.



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

Tháng 9/1991 phiên bản Linux 0.0.1 , phiên bản đầu tiên được Torvalds công bố với 10239 dòng lệnh. Phiên bản 0.0.2 cũng được công bố 1 tháng sau đó.

Năm 1992, Torvalds cho phát hành Linux dưới dạng mã nguồn mở của giấy phép GPL, cho phép mọi người có thể download về để xem mã nguồn mở để cùng chung tay phát triển.

Năm 1993, Slackware, hệ điều hành đầu tiên được phát triển dựa trên mã nguồn mở Linux đã ra đời. Phiên bản mới nhất của Slackware được phát hành 5/2010 . 14/3/1994 Sau 3 năm làm việc, Torvalds đã cho ra mắt phiên bản hoàn thiện đầu tiên, Linux 1.0 ra đời với 176.250 dòng lệnh. Sau đó 1 năm, phiên bản 1.2 ra đời với 310.950 dòng lệnh.

Ngày 3/11/1994 Red Hat Linux, phiên bản 1.0 được giới thiệu. Đây là một trong những phiên bản được thương mại hóa đầu tiên dựa trên Linux.

Năm 1996, hình ảnh chú chim cánh cụt được lấy làm biểu tượng của Linux.

Năm 1998, Linux được IBM, Compaq, Oracle..quan tâm đầu tư phát triển….

Năm 2007, các hãng máy tính HP, ASUS., Dell… cũng bắt đầu bán ra các sản phẩm laptop được cài sẵn Linux.

1/2009, số người dùng Linux đạt tới con số 10 triệu người.

Hiện nay sau 20 năm tồn tại phát triển, Linux đã được đông đảo mọi người biết đến và sử dụng, vốn chỉ từ hệ điều hành hơn 10 ngàn câu lệnh, phiên bản mới nhất 2.6.38 đã được phát hành với 14.294.493 dòng lệnh đánh dấu chặng đường tồn tại và phát triển của Linux.

**2. 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).

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

- Tiến trình là một đơn vị công việc trong một hệ điều hành phân chia thời gian hiện đại.

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

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

- Running(đang chạy): Tiến trình chiếm quyền xử lí CPU dùng tính toán hay thực thi các công việc của mình.

- Waiting(chờ): Tiến trình bị HĐH tước quyền xử lí CPU và chờ đến lượt cấp phát khác.

- Suspend(tạm dừng): HĐH tạm dừng tiến trình, tiến trình được đưa vào trạng thái ngủ ( sleep), khi cần thiết và có nhu cầu HĐH sẽ đánh thức(wake up) hay nạp lại mã lệnh của tiến trình vào bộ nhớ, cấp phát tài nguyên CPU để tiến trình tiến trình có thể hoạt động.

Tại dòng lệnh có thể bấm Ctrl –Z để tạm dừng một tiến trình sau đó dùng lệnh “pg” để đưa vào hậu trường để sau đó dùng “fg” để chuyển lên mặt trước.

**4. Quan hệ giữa các tiến trình.**

* Các tiến trình hoạt động trong 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 tiến trình là không đơn định(kết quả phụ thuộc dãy thực hiện tương ứng và không dự báo trước).
* Tiến trình không thể tái hiện.

**5. Quản lý tiến trình trên linux.**

**5.1. Tại sao phải 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).

| Tiến trình là một môi trường thực hiện, bao gồm một phân đoạn lệnh và một phân đoạn dữ liệu. Cần phân biệt với khái niệm chương trình chỉ gồm tập hợp lệnh.  Trên hệ điều hành Linux, tiến trình được nhận biết thông qua số hiệu của tiến trình, gọi là pid. Cũng như đối với user, nó có thể nằm trong nhóm. Vì thế để phân biệt ta nhận biết qua số hiệu nhóm gọi là PRGP. Một số hàm của C cho phép lấy được những thông số này:  int getpid() : trả về giá trị int là pid của tiến trình hiện tại  int getppid() : trả về giá trị int là pid của tiến trình cha của tiến trình hiện tại  int getpgrp() : trả về giá trị int là số hiệu của nhóm tiến trình  int setpgrp() : trả về giá trị int là số hiệu nhóm tiến trình mới tạo ra  Ví dụ  Lệnh : printf("Toi la tien trinh %d thuoc nhom %d",getpid(),getgrp());  Kết quả sẽ là: Toi là tien trinh 235 thuoc nhom 231  **5.2. Tạo một tiến trình.**  int fork() tạo ra một 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. Theo nguyên tắc cơ bản của hệ thống, tiến trình con và cha sẽ có cùng đoạn mã. Đoạn dữ liệu của tiến trình mới là một bản sao chép chính xác đoạn dữ liệu của tiến trình cha. Tuy nhiên tiến trình con vẫn khác tiến trình cha ở pid, thời gian xử lý, ...  **5.3. Dừng một tiến trình.**  Lệnh kill của Shell có thể dùng để chấm dứt hoạt động của một tiến trình.  Ví dụ như khi muốn dừng tiến trình 234 ta dùng lệnh: kill 234 C cũng có lệnh kill như sau:  int kill(pid, sig);  int pid; là dấu hiệu nhận biết của một tiến trình.  int sig; hằng tín hiệu giao tiếp tiến trình.  **5.4. Giao tiếp giữa các tiến trình.**  Việc giao tiếp giữa các tiến trình được thực hiện thông qua các tín hiệu chuẩn của hệ thống. Tín hiệu là một sự ngắt quãng logic được gửi đến các tiến trình bởi hệ thống để thông báo cho chúng về những sự việc không bình thường trong môi trường hoạt động của chúng (như lỗi bộ nhớ, lỗi vào ra). Nó cũng cho phép các tiến trình liên lạc với nhau. Một tín hiệu (trừ SIGKILL) có thể được xem xét theo ba cách khác nhau:  Tiến trình có thể được bỏ qua: Ví dụ chương trình có thể bỏ qua sự ngắt quãng của người sử dụng hệ thống (đó là sự bỏ qua khi một tiến trình đang được sử dụng ở phần nền.  Tiến trình có thể được thực hiện: Trong trường hợp này, khi nhận được 1 tín hiệu, việc thực hiện 1 tiến trình được chuyển về một quy trình do người sử dụng xác định trước, sau đó trở lại nơi nó bị ngắt.  Lỗi có thể được tiến trình trả về sau khi nhận được tín hiệu này.Dưới đây là một số tín hiệu thường gặp:   * SIGHUP:   + Tín hiệu này được phát đến các tiến trình vào lúc cuối khi mà nó tự ngắt.  + Nó cũng được phát đến mọi tiến trình có tiến trình chính tự ngắt.   * SIGINT: Tín hiệu này được phát đến các tiến trình khi ta ra lệnh ngắt. * SIGQUIT: Tương tự như trên khi ta gõ vào ^D. * SIGILL: Lệnh không hợp lệ, tín hiệu được phát ra khi phát hiện 1 lệnh không đúng ở cấp độ vật lý (ví dụ như 1 tiến trình thực hiện một lệnh mà máy tính chống có lệnh này). * SIGTRAP: Tín hiệu được phát ra sau mỗi lệnh trong trường hợp tiến trình có sử dụng lệnh ptrace(). * SIGIOT: Bẫy được phát khi có các vấn đề về vật lý.   - SIGEMT: Bẫy của lệnh phát, được phát ra khi có lỗi vật lý trong khi thực hiện.  - SIGFPE: Được phát ra khi có lỗi về tính toán như một số có dấu phẩy nối có định dạng không hợp lý. Gần như luôn chỉ ra lỗi khi lập trình.  - SIGKILL: Trang bị để kết thúc tiến trình. Không thể bỏ qua hoặc cắt tín hiệu này.  - SIGBUS: Được phát khi gặp lỗi trên bus.  - SIGSEGV: Được phát ra khi gặp lỗi trên phân đoạn sự truy cập dữ liệu bên ngoài phân đoạn dữ liệu được cấp phát cho tiến trình.  - SIGSYS: Đối số không đúng cho hệ thống gọi.  - SIGPIPE: Viết trên một ống dẫn không mở để đọc.  - SIGALRM: Phát ra khi đồng hồ của một tiến trình ngừng lại. Đồng hồ được hoạt động bằng lệnh alrm().  - SIGTERM: Được phát ra khi một tiến trình kết thúc bình thường. Cũng có thể dùng để dừng 1 hệ thống để kết thúc tất cả các tiến trình hoạt động.  **5.5. Liên lạc giữa hai tiến trình.**  Từ một chương trình đơn giản dưới đây sử dụng các lệnh phát và nhận tín hiệu, sau đó giúp liên lạc giữa hai tiến trình.  Nội dung của ví dụ là sự liên lạc giữa một tiến trình cha và một tiến trình con thông qua các tín hiệu đã được trình bày phần trước.  #include  #include  void fils\_atc()  {  printf(" Tien trinh bi loai bo !!!\n");  kill(getpid(), SIGINT);  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void fils()  {  signal(SIGUSR1, fils\_atc);  printf(" Hinh thanh tien trinh moi. Nhung chuan bi loai bo tien trinh nay !!\n");  while(1);  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  main()  {  int ppid, pid;  if ((pid = fork())==0) fils();  else  {  sleep(3);  printf(" Chap nhan !! Tien trinh se bi loai bo.\n");  kill(pid, SIGUSR1);  }  }  Trong ví dụ trên, tiến trình con có sử dụng hàm signal(SIGUSR1, fils\_atc). Hàm này có tác dụng mỗi khi tiến trình con nhận được tín hiệu SIGUSR1 thì hàm fils\_atc() sẽ được thực thi.  Như vậy ở ví dụ trên một tiến trình con đã được tạo ra nhưng nó lại không muốn tiếp tục tồn tại. Do vậy sau khi tạm dừng lại sleep(3), tiến trình cha đã gởi đến cho tiến trình con một tín hiệu là SIGUSR1 bằng lệnh:  kill(pid, SIGUSR1);  Ở tiến trình con, tín hiệu SIGUSR1 đã được gán với hàm fils\_atc(). Hàm này ra một thông báo báo hiệu tiến trình này sắp chết rồi tự gởi đến chính mình (tiến trình con) tín hiệu SIGINT, tín hiệu ngắt tiến trình. Và tiến trình con đã chết.  kill(getpid(), SIGINT);  Một số nhược điểm khi liên lạc trực tiếp bằng tín hiệu:  - Một tín hiệu có thể bị bỏ qua, kết thúc một tiến trình hoặc bị chặn lại. Đó là lý do chính đưa ra các tín hiệu không thích ứng được để tiến hành liên lạc giữa các tiến trình. Một thông điệp điệp dưới hình thức tín hiệu có thể sẽ bị mất nếu nó được nhận lúc loại tín hiệu này tạm thời bị bỏ qua.  - Một vấn đề khác là các tín hiệu có quyền rất lớn, khi đến chúng làm ngắt quãng công việc hiện tại. Ví dụ việc nhận một tín hiệu trong khi tiến trình đang đợi một sự kiện (mà có thể đến khi sử dụng các lệnh open(), read(), ...) làm cho việc thực thi hàm bị chệch hướng. Khi trở lại, lệnh chính bị ngắt gởi lại một thông điệp báo lỗi mà hoàn toàn không xử lý được.  Ngoài việc liên lạc trực tiếp như ở ví dụ trên, còn cho phép một phương pháp liên lạc giữa các tiến trình khác, đó là liên lạc qua "đường ống".  **5.6. Lập lịch đa tiến trình.**  Ống dẫn liên lạc  Ống dẫn 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.  **5.7. Liên lạc giữa tiến trình cha và tiến trình con.**  Trong ví dụ dưới đây, một tiến trình tạo ra một ống dẫn, tạo ra một tiến trình con, viết một văn bản vào ống dẫn.Tiến trình con thừa hưởng ống dẫn và các ký hiệu mô tả của ống dẫn, thực hiện đọc trong ống dẫn:  *#include*  *#include*  *void code\_fils(int number) {*  *int fd, nread;*  *char texte[100];*  *- 31-*  *fd=number;*  *printf(" So hieu mo ta la %d\n",fd);*  *switch (nread=read(fd, texte, sizeof(texte)))*  *{*  *case -1:*  *perror("Loi doc.");*  *case 0:*  *perror("EOF");*  *default:*  *printf("Van ban nhan duoc co %d ky tu: %s\n",fd, texte);*  *}*  *}*  *main() {*  *int fd[2];*  *char chaine[10];*  *if (pipe(fd)==-1)*  *{ perror("Loi khoi tao pipe.");*  *exit(1);*  *}*  *switch (fork()) {*  *case -1:*  *perror(" Loi khoi tao tien trinh.");*  *break;*  *case 0:*  *if (close(fd[1])==-1)*  *perror(" Error.");*  *code\_fils(fd[0]);*  *exit(0);*  *}*  *close(fd[0]);*  *if (write(fd[1]),"hello",6)==-1) perror("Loi truyen.");*  *}*  Kết quả chương trình:  So hieu mo ta la: 5  Van ban nhan duoc co 6 ky tu: hello  Chú ý rằng, tiến trình con đọc trong ống dẫn mà không viết ở đó nên nó bắt đầu bằng cách đóng phần viết fd[1] để tiết kiệm các tín hiệu mô tả của tổ hợp. Tương tự, vì tiến trình cha chỉ sử dụng phần viết nên nó đóng phần đọc lại (fd[0]). Sau đó tiến trình cha viết vào ống dẫn 6 ký tự và tiến trình con đã đọc chúng. |
| --- |