Chương III: Tiến trình (Process)

- Khái niệm cơ bản
- Trạng thái quá trình
- Khối điều khiển quá trình (Process control block)
- Định thời quá trình (Process Scheduling)
- Các tác vụ đối với quá trình
- Sự cộng tác giữa các quá trình
- Giao tiếp giữa các quá trình

3.1. Khái niệm cơ bản

- Cái gì gọi các hoạt động của CPU?
  - Hệ thống bó (Batch system): jobs
  - Time-shared systems: user programs, tasks
  - Các hoạt động là tương tự => gọi là process

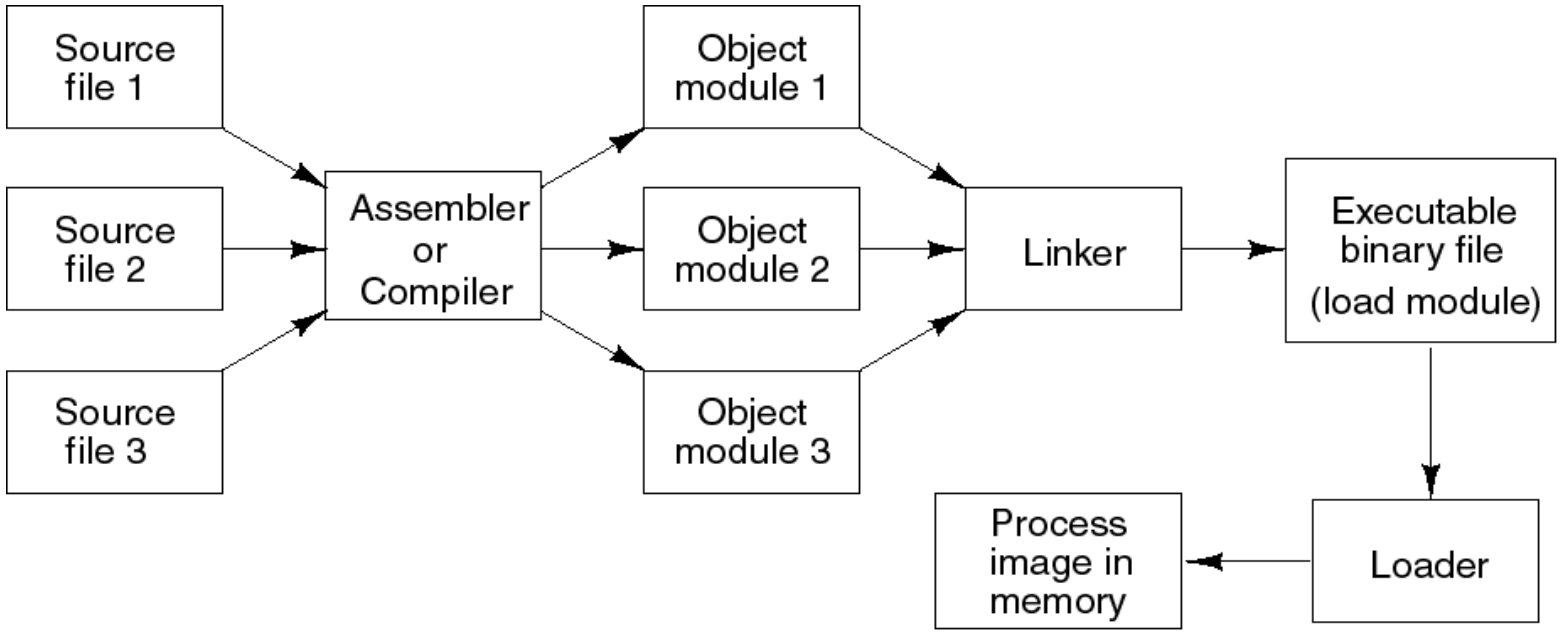
- *Quá trình* (process)
  - một chương trình đang thực thi

Một quá trình bao gồm

- *Text section* (program code), *data section* (chứa global variables)
- *program counter (PC), process status word (PSW), stack pointer (SP), memory management registers, …*

3.1. Khái niệm cơ bản

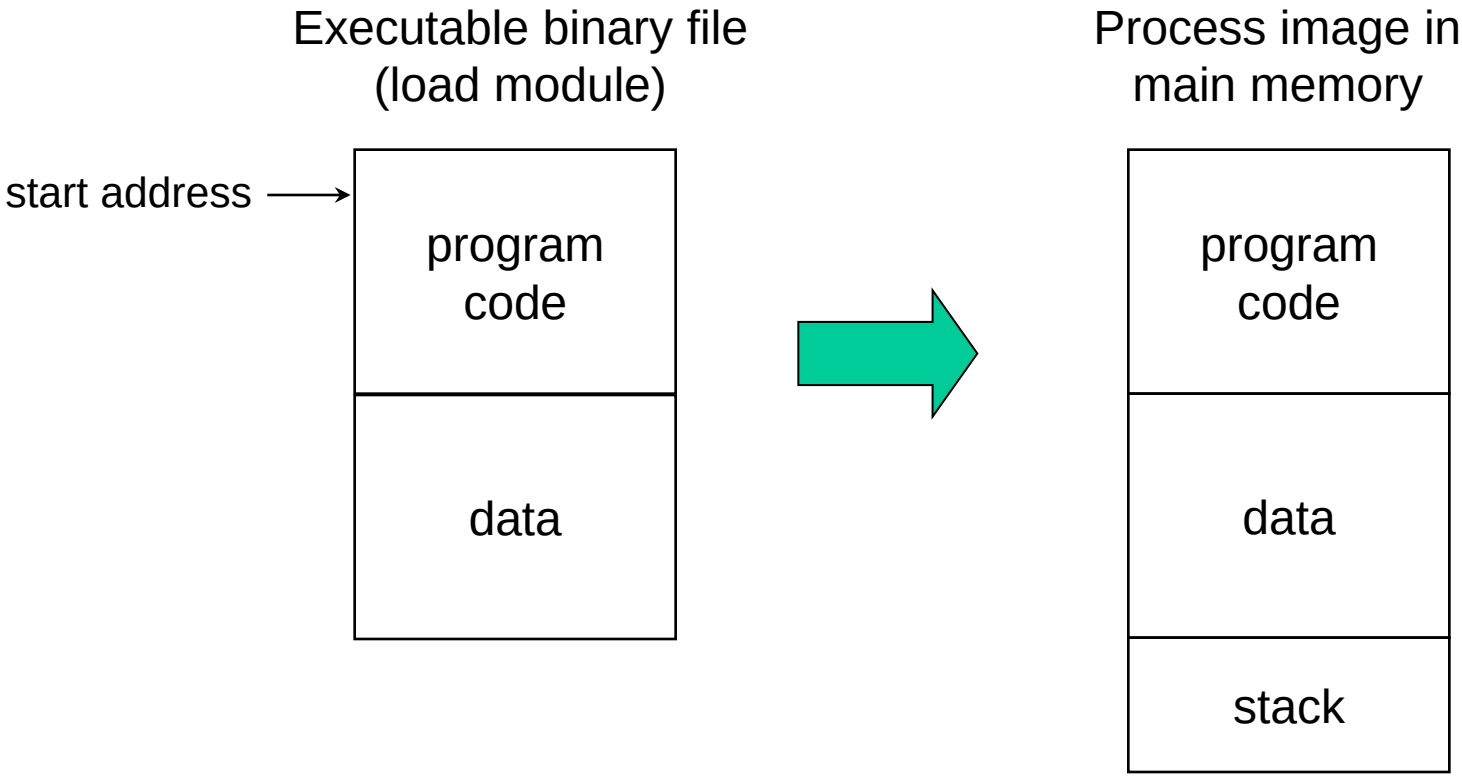
Các bước nạp chương trình vào bộ nhớ



3.1. Khái niệm cơ bản

chương trình => quá trình

- Dùng *load module* để biểu diễn chương trình thực thi được
- Layout luận lý của *process image*



3.1. Khái niệm cơ bản

Khởi tạo quá trình

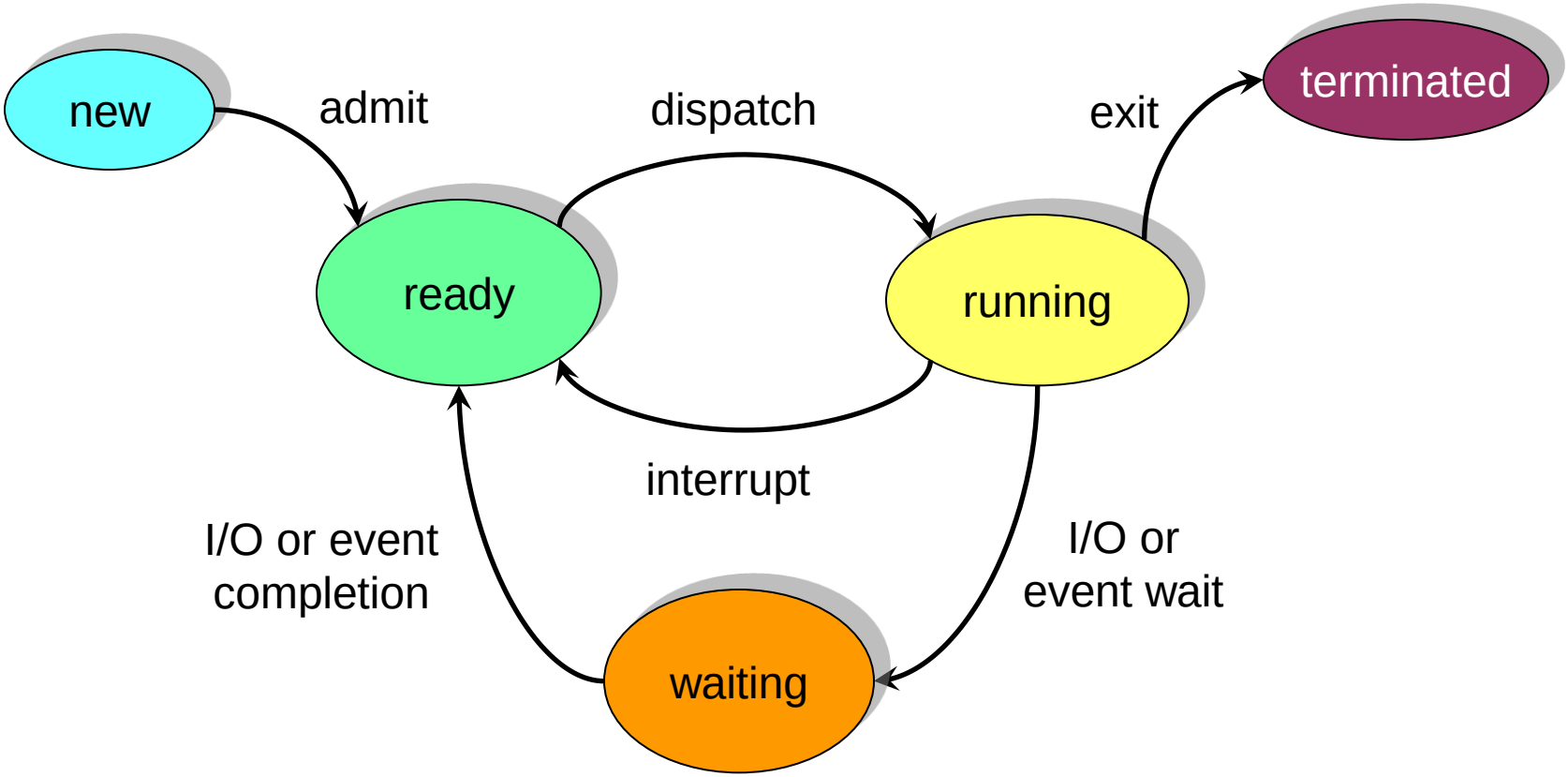
- Các bước hệ điều hành khởi tạo quá trình
  - Cấp phát một *định danh* duy nhất (process number hay process identifier, pid) cho quá trình
  - Cấp phát không gian nhớ để nạp quá trình
  - Khởi tạo khối dữ liệu *Process Control Block* (PCB) cho quá trình
    - PCB là nơi hệ điều hành lưu các thông tin về quá trình
  - Thiết lập các mối liên hệ cần thiết (vd: sắp PCB vào hàng đợi định thời,…)

3.2.Trạng thái quá trình

- Các *trạng thái của quá trình* (process states):
  - *new:* quá trình vừa được tạo
  - *ready:* quá trình đã có đủ tài nguyên, chỉ còn cần CPU
  - *running:* các lệnh của quá trình đang được thực thi
  - *waiting:* hay là *blocked*, quá trình đợi I/O hoàn tất, tín hiệu.
  - *terminated:* quá trình đã kết thúc.

3.2.Trạng thái quá trình

- Chuyển đổi giữa các trạng thái của quá trình



3.2.Trạng thái quá trình

Ví dụ

```
/* test.c */
int main(int argc, char** argv)
{
   printf("Hello world\n");
   exit(0);
}
```

Biên dịch chương trình trong Linux
`gcc test.c -o test`

Thực thi chương trình test
`./test`

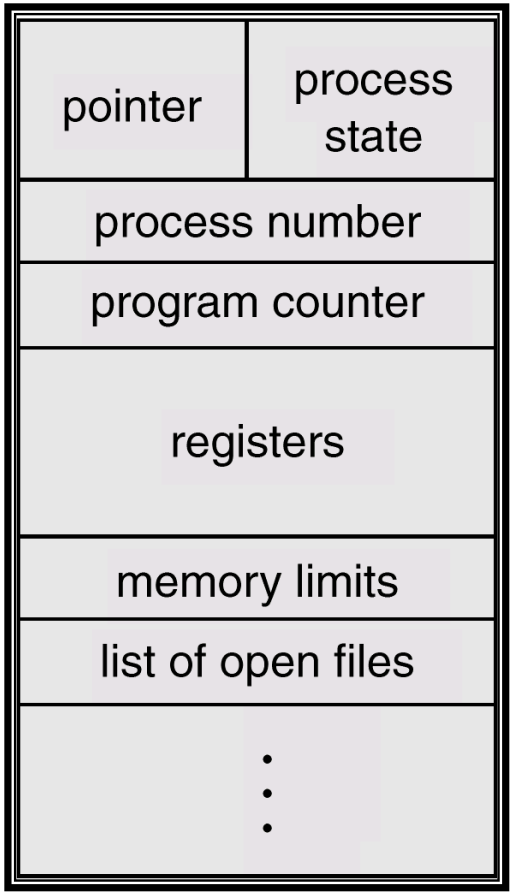
Trong hệ thống sẽ có một quá trình *test* được tạo ra, thực thi và kết thúc.

- Chuỗi trạng thái của quá trình test như sau (trường hợp tốt nhất):
  - new
  - ready
  - running
  - waiting (do chờ I/O khi gọi printf)
  - ready
  - running
  - terminated

3.3.Process control block

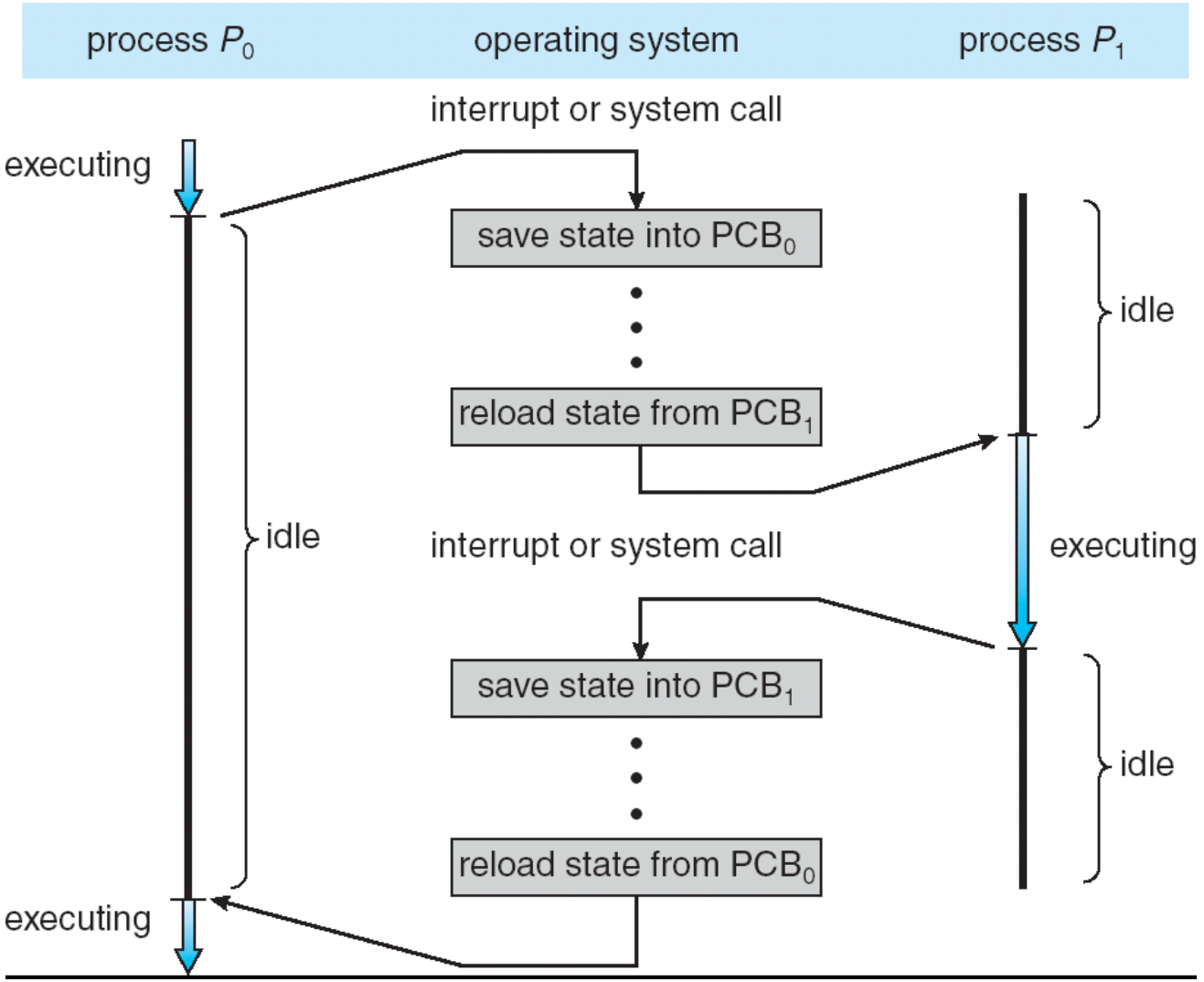
- Đã thấy là mỗi quá trình trong hệ thống đều được cấp phát một *Process Control Block* (PCB)
- PCB là một trong các cấu trúc dữ liệu quan trọng nhất của hệ điều hành và gồm:

  - Trạng thái quá trình: new, ready, running,…
  - Bộ đếm chương trình
  - Các thanh ghi
  - Thông tin lập thời biểu CPU: độ ưu tiên, …
  - Thông tin quản lý bộ nhớ
  - Thông tin tài khoản: lượng CPU, thời gian sử dụng,
  - Thông tin trạng thái I/O



3.3.Process control block

Lưu đồ chuyển CPU từ quá trình này đến quá trình khác

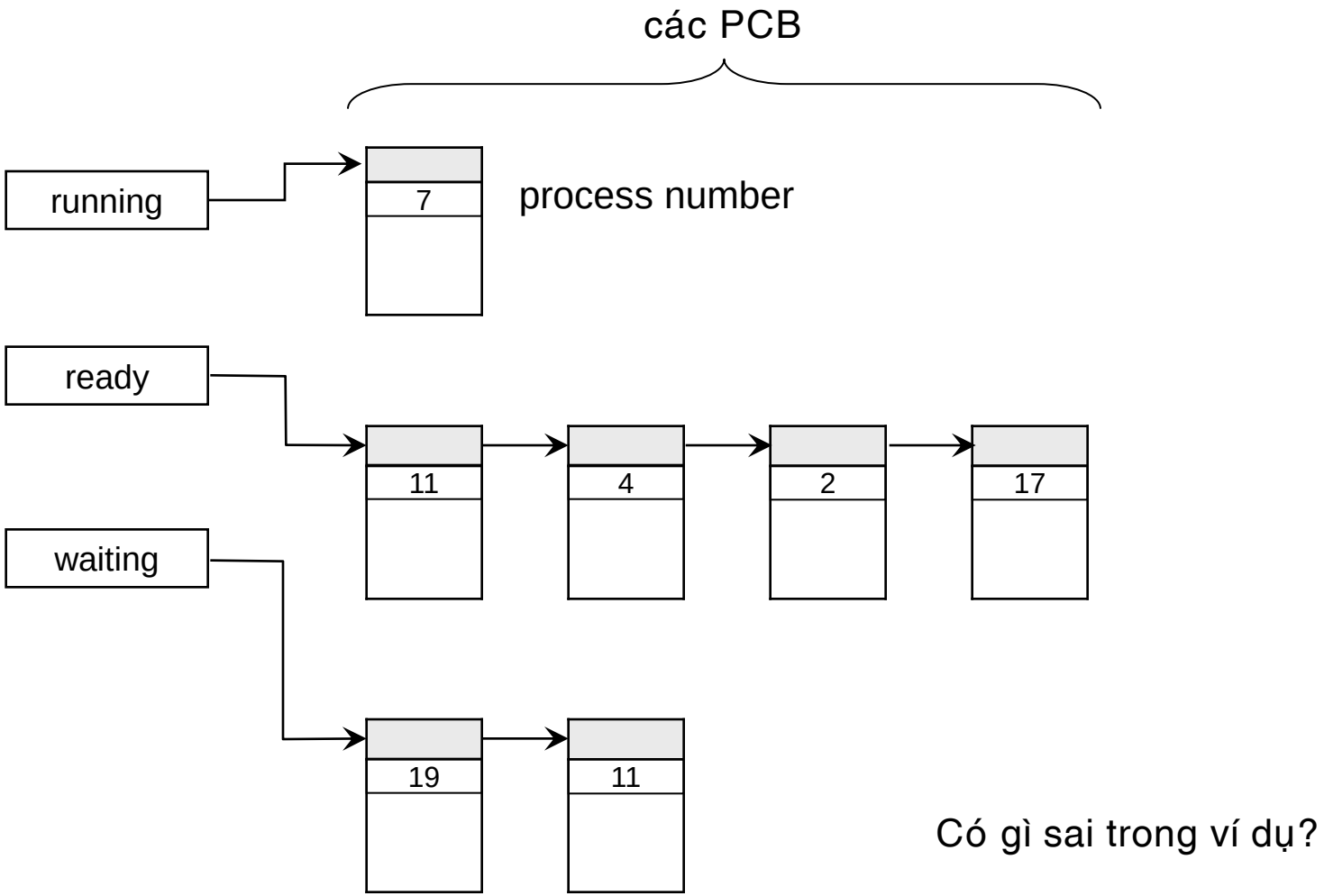


Yêu cầu đối với hệ điều hành về quản lý quá trình

- Hỗ trợ sự thực thi luân phiên giữa nhiều quá trình
  - Hiệu suất sử dụng CPU
  - Thời gian đáp ứng
- Phân phối tài nguyên hệ thống hợp lý
  - tránh deadlock, trì hoãn vô hạn định,…
- Cung cấp cơ chế giao tiếp và đồng bộ hoạt động các quá trình
- Cung cấp cơ chế hỗ trợ user tạo/kết thúc quá trình

Quản lý các quá trình: các hàng đợi

- Ví dụ

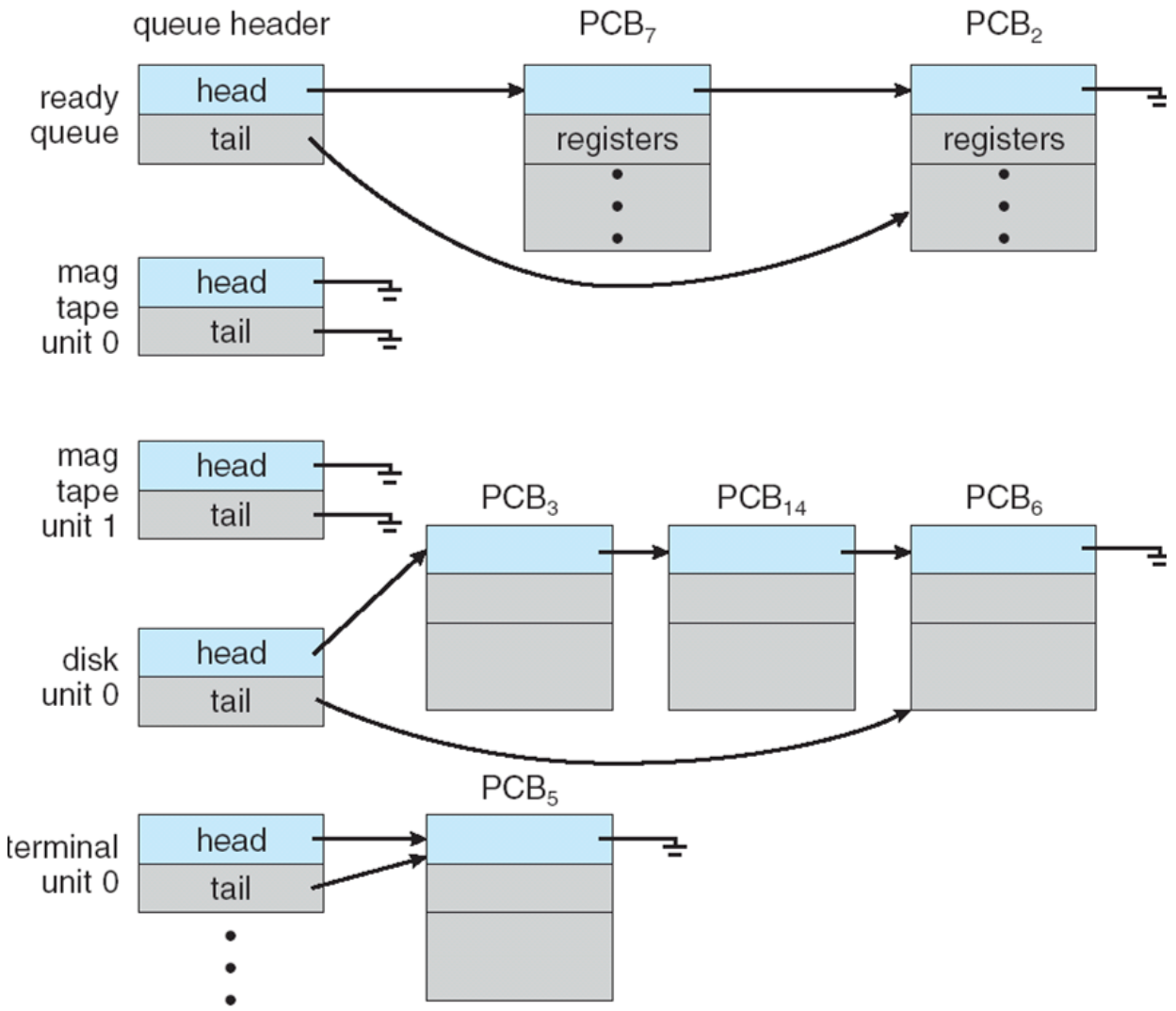


3.4. Định thời quá trình (Process Scheduling)

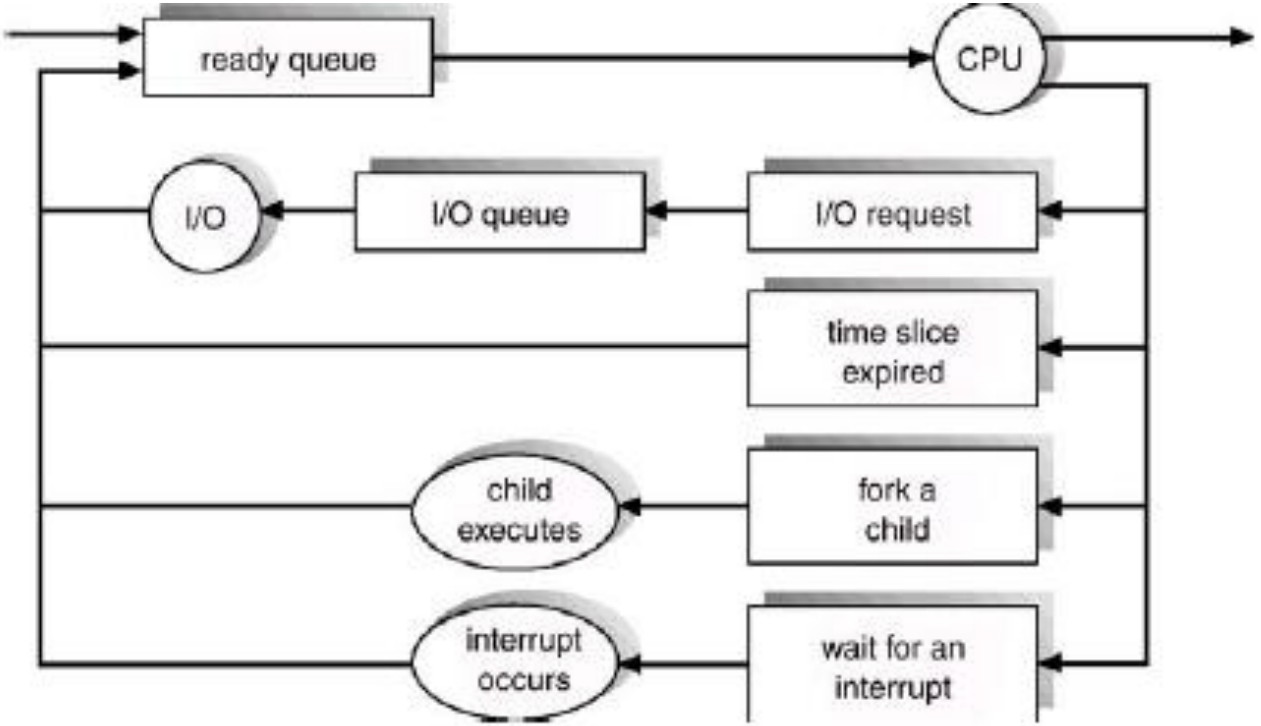
- Tại sao phải định thời?
  - Đa chương (Multiprogramming)
    - Có vài quá trình chạy tại các thời điểm
    - Mục tiêu: tận dụng tối đa CPU
  - Chia thời(Time-sharing)
    - Users tương tác với mỗi chương trình đang thực thi
    - Mục tiêu: tối thiểu thời gian đáp ứng
- Một số khái niệm cơ bản
  - Các *bộ định thời* (scheduler)
  - Các *hàng đợi định thời* (scheduling queue)

Các hàng đợi định thời (Scheduling queues)

- Hàng đợi công việc-Job queue
- Hàng đợi sẵn sàng-Ready queue
- Hàng đợi thiết bị-Device queues
- …



Các hàng đợi định thời (Scheduling queues)



Lưu đồ hàng đợi của định thời quá trình

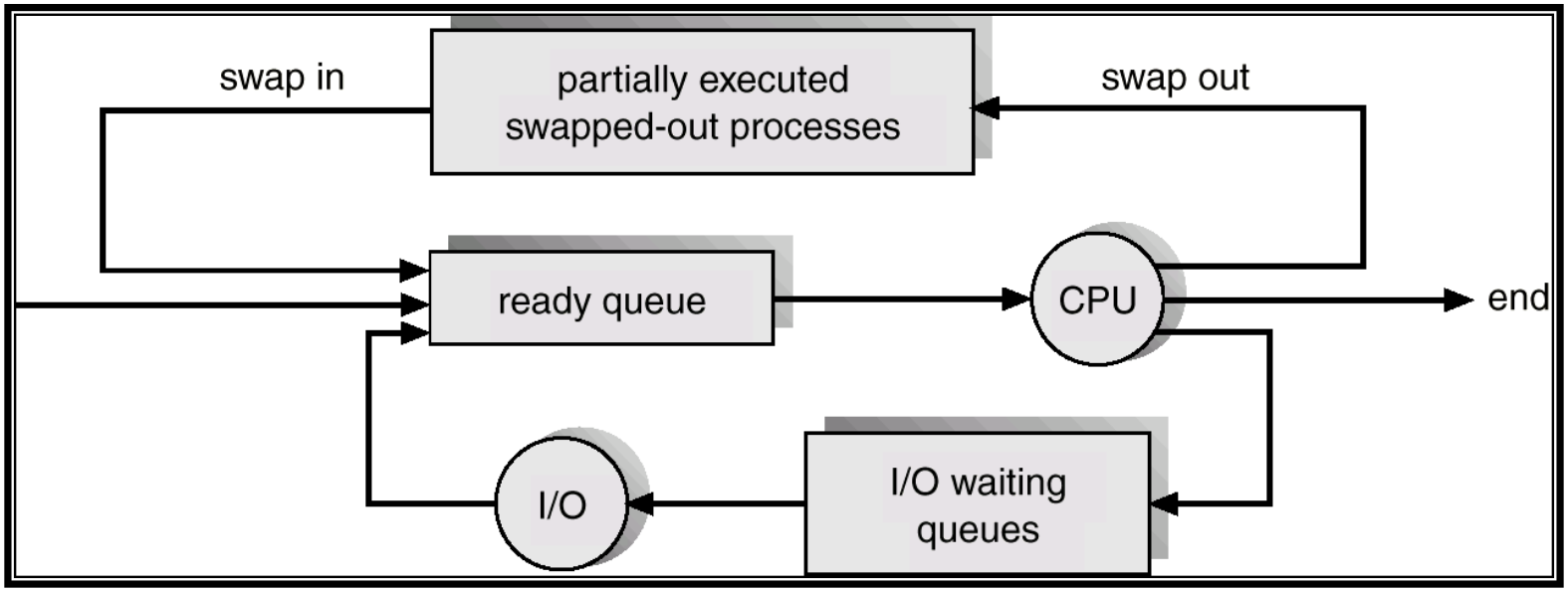
3.5. Bộ định thời (Scheduler)

- Bộ định thời công việc (Job scheduler) hay bộ định thời dài (long-term scheduler)
- Bộ định thời CPU hay bộ định thời ngắn
- Các quá trình có thể mô tả như:
  - Quá trình hướng I/O (I/O bound process)
  - Quá trình hướng CPU (CPU bound process)

  Thời gian thực hiện khác nhau => kết hợp hài hòa giữa chúng

Bộ định thời trung gian(medium-term scheduling)

- Đôi khi hệ điều hành (như time-sharing system) có thêm medium-term scheduling để điều chỉnh mức độ đa chương của hệ thống
- *Medium-term scheduler*
  - chuyển quá trình từ bộ nhớ sang đĩa (swap out)
  - chuyển quá trình từ đĩa vào bộ nhớ (swap in)

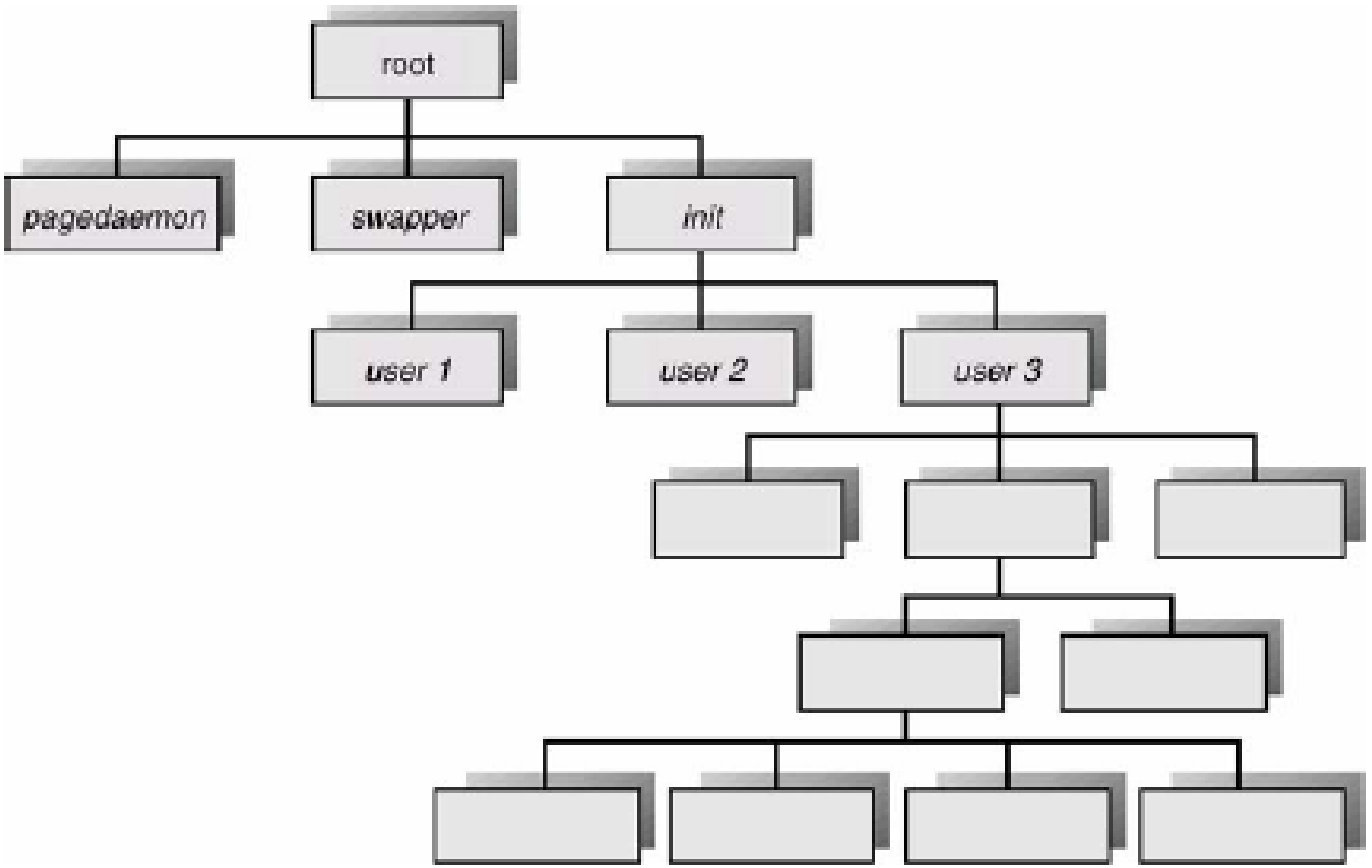


3.6. Các tác vụ đối với quá trình

- Tạo quá trình mới (process creation)
  - Một quá trình có thể tạo nhiều quá trình mới thông qua một lời gọi hệ thống ***create-process*** (vd: hàm fork trong Unix)
    - Ví dụ: (Unix) Khi user đăng nhập hệ thống, một command interpreter (shell) sẽ được tạo ra cho user
  - Quá trình được tạo là quá trình *con* của quá trình tạo (quá trình *cha*). Quan hệ cha-con định nghĩa một *cây quá trình.*

Cây quá trình trong Linux/Unix

- Ví dụ

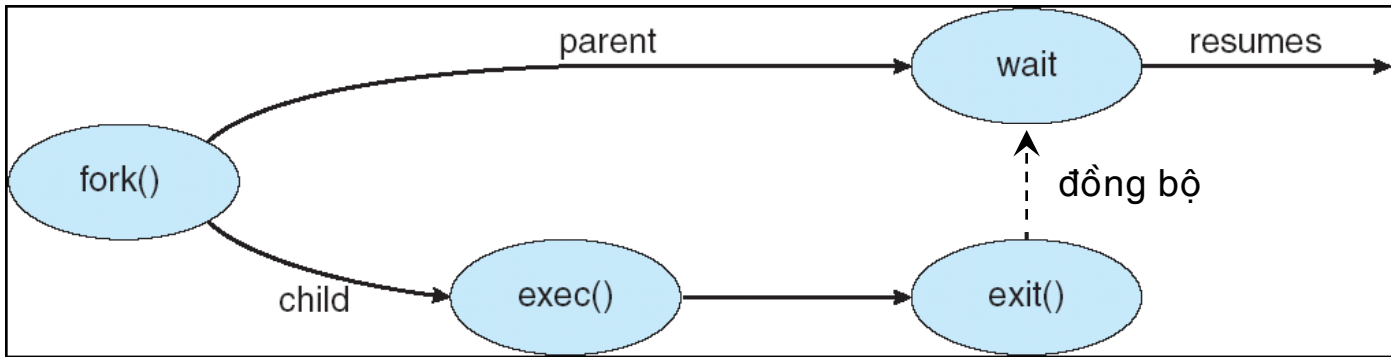


3.6.Các tác vụ đối với quá trình

- Tạo quá trình mới
  - Quá trình con nhận tài nguyên: từ HĐH hoặc từ P cha
  - Chia sẻ tài nguyên của quá trình cha
    - Quá trình cha và con chia sẻ mọi tài nguyên
    - Quá trình con chia sẻ một phần tài nguyên của cha
  - Trình tự thực thi
    - Quá trình cha và con thực thi đồng thời (concurrently)
    - Quá trình cha đợi đến khi các quá trình con kết thúc.

Về quan hệ cha/con

- Không gian địa chỉ (address space)
  - Không gian địa chỉ của quá trình con được nhân bản từ cha
  - Không gian địa chỉ của quá trình con được khởi tạo từ template.
- Ví dụ trong UNIX/Linux
  - System call fork() tạo một quá trình mới
  - System call exec() dùng sau fork() để nạp một chương trình mới vào không gian nhớ của quá trình mới



Ví dụ tạo process với fork()

```
#include <stdio.h>
#include <unistd.h>
int main (int argc, char *argv[]){
   int  pid;
   /* create a new process */
   pid = fork();

   if (pid > 0){
      printf("This is parent process");
      wait(NULL);
      exit(0);
   }
   else if (pid == 0)
   {
      printf("This is child process");
      execlp("/bin/ls", "ls", NULL);
      exit(0);
   }
   else {
        printf("Fork error\n");
        exit(-1);
   }
}
```

```
void main(){
        printf (“hi”);
        fork ();
        printf (“Hello”);
        fork ();
        printf (“Bye”);
}
```

Hỏi chương trình in ra các dòng chữ nào trên màn hình.

```
void main() {
        int pid;                        1
        printf (“hi”);                  2
        pid = fork ();                  3
        If( pid == 0) {                 4
                fork ();                5
                printf (“Hello”);       6
        } else                          7
                printf (“Bye”);         8
}
```

Hỏi chương trình in ra các dòng chữ nào trên màn hình.

Bỏ lệnh dòng 4, dòng 7. Chuong trình in ra các dòng chữ nào.

3.6.Các tác vụ đối với quá trình (tt)

- Tạo quá trình mới ✓
- Kết thúc quá trình
  - Quá trình tự kết thúc
    - Quá trình kết thúc khi thực thi lệnh cuối và gọi system routine exit
  - Quá trình kết thúc do quá trình khác (có đủ quyền, vd: quá trình cha của nó)
    - Gọi system routine abort với tham số là pid (process identifier) của quá trình cần được kết thúc
  - Hệ điều hành thu hồi tất cả các tài nguyên của quá trình kết thúc (vùng nhớ, I/O buffer,…)

3.7. Cộng tác giữa các quá trình

- Trong quá trình thực thi, các quá trình có thể *cộng tác* (cooperate) để hoàn thành công việc
- Các quá trình cộng tác để
  - Chia sẻ dữ liệu (information sharing)
  - Tăng tốc tính toán (computational speedup)
    - Nếu hệ thống có nhiều CPU, chia công việc tính toán thành nhiều công việc tính toán nhỏ chạy song song
  - Thực hiện một công việc chung
    - Xây dựng một phần mềm phức tạp bằng cách chia thành các module/process hợp tác nhau
- Sự cộng tác giữa các quá trình yêu cầu hệ điều hành hỗ trợ cơ chế giao tiếp và cơ chế đồng bộ hoạt động của các quá trình

Bài toán người sản xuất-người tiêu thụ (producer-consumer)

- Ví dụ cộng tác giữa các quá trình: *bài toán producer-consumer*
  - *Producer* tạo ra các dữ liệu và *consumer* tiêu thụ, sử dụng các dữ liệu đó. Sự trao đổi thông tin thực hiện qua buffer
    - *unbounded buffer:* kích thước buffer vô hạn (không thực tế).
    - *bounded buffer:* kích thước buffer có hạn.
  - Producer và consumer phải hoạt động đồng bộ vì
    - Consumer không được tiêu thụ khi producer chưa sản xuất
    - Producer không được tạo thêm sản phẩm khi buffer đầy.

3.8.Giao tiếp liên quá trình (Interprocess communication-IPC)

- *IPC* là cơ chế cung cấp bởi hệ điều hành nhằm giúp các quá trình
  - giao tiếp với nhau
  - và đồng bộ hoạt động

  mà không cần chia sẻ không gian địa chỉ

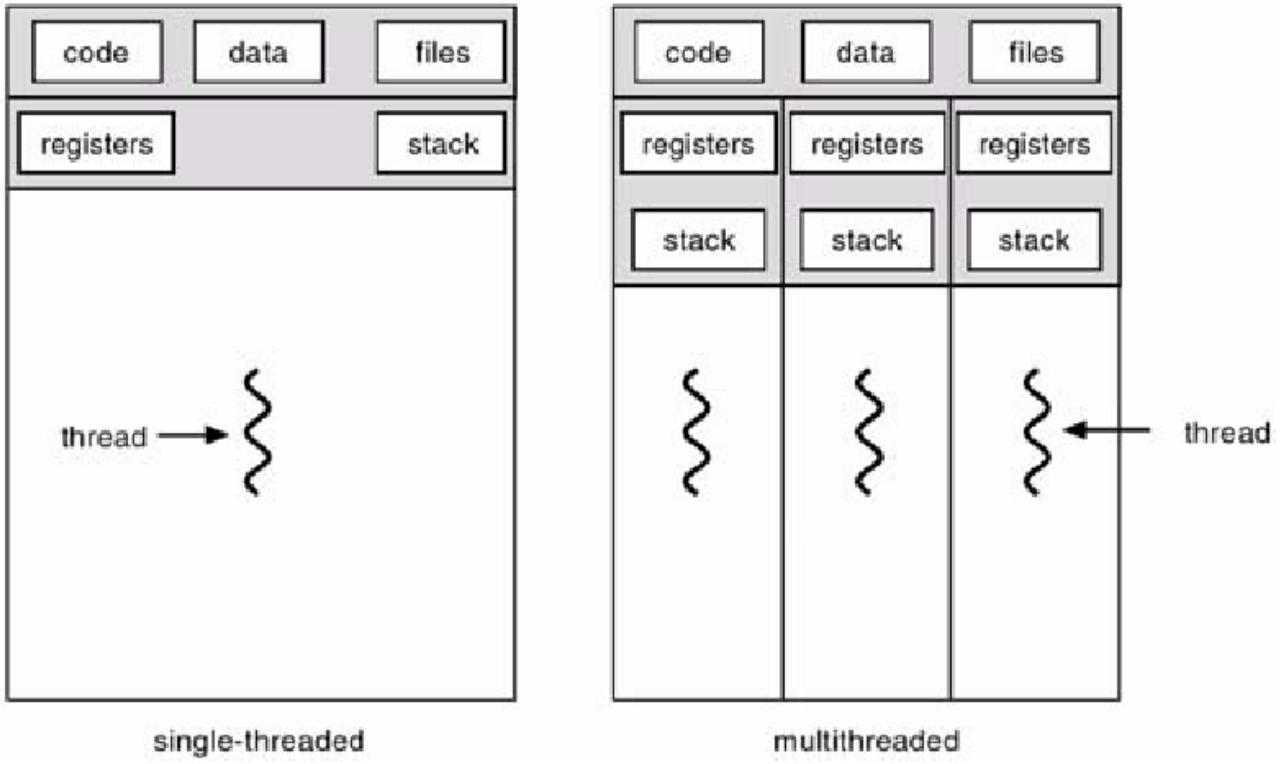
- IPC có thể được cung cấp bởi message passing system

Hệ thống truyền thông điệp
Message passing system

- Làm thế nào để các quá trình giao tiếp nhau? Các vấn đề:
  - *Đặt tên (Naming)*
    - Giao tiếp trực tiếp
      - send(P, msg): gửi thông điệp đến quá trình P
      - receive(Q, msg): nhận thông điệp đến từ quá trình Q
    - Giao tiếp gián tiếp: thông qua *mailbox* hay *port*
      - send(A, msg): gửi thông điệp đến mailbox A
      - receive(B, msg): nhận thông điệp từ mailbox B
  - *Đồng bộ hóa (Synchronization)*: blocking send, nonblocking send, blocking receive, nonblocking receive
  - *Tạo vùng đệm (Buffering)*: dùng queue để tạm chứa các message
    - Khả năng chứa là 0(Zero capacity hay no buffering)
    - Bounded capacity: độ dài của queue là giới hạn
    - Unbounded capacity: độ dài của queue là không giới hạn

Tiểu trình (luồng) - Thread

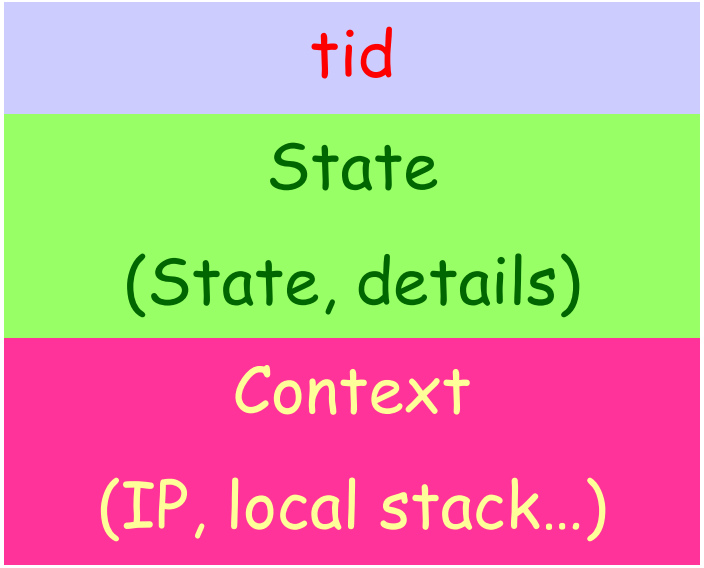
- Tiểu trình(tiến trình nhẹ-lightweight process): là một đơn vị cơ bản sử dụng CPU, gồm:
  - Thread ID, PC, Registers, stack và chia sẻ chung code, data, resources (files)



PCB vàTCB trong mô hình multithreads

PCB

Thread Control Block
TCB

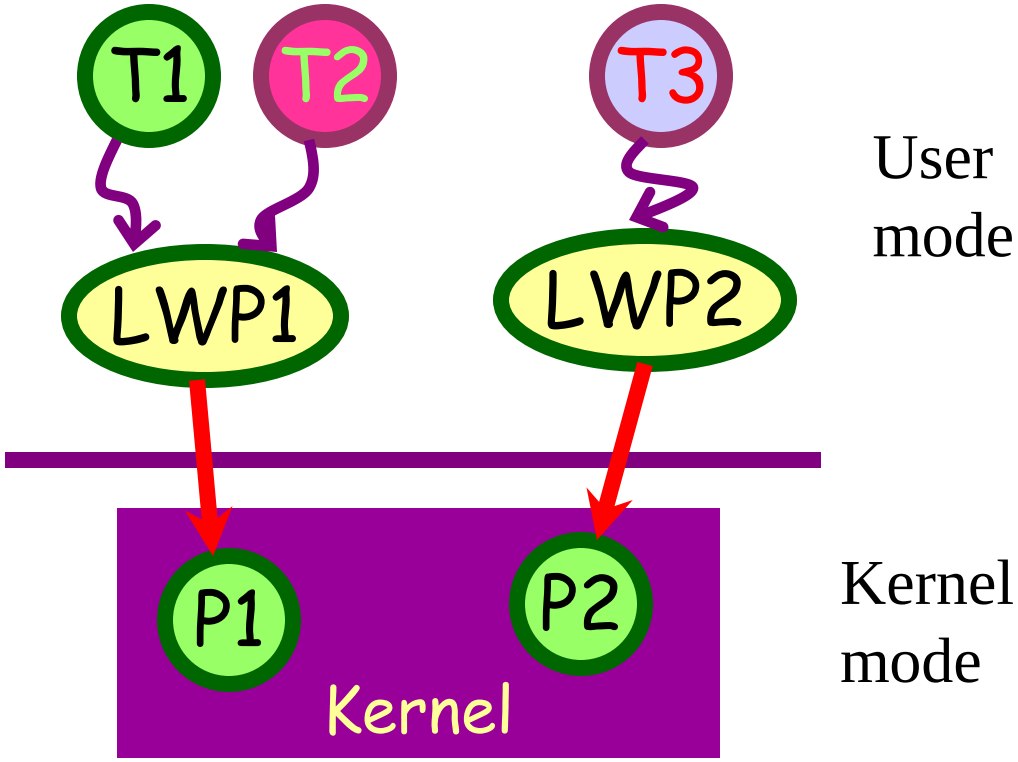


Lợi ích của tiến trình đa luồng

- Đáp ứng nhanh: Cho phép chương trình tiếp tục thực thi khi một bộ phận bị khóa hoặc một hoạt động dài
- Chia sẻ tài nguyên: tiết kiệm không gian nhớ
- Kinh tế: tạo và chuyển ngữ cảnh nhanh hơn tiến trình

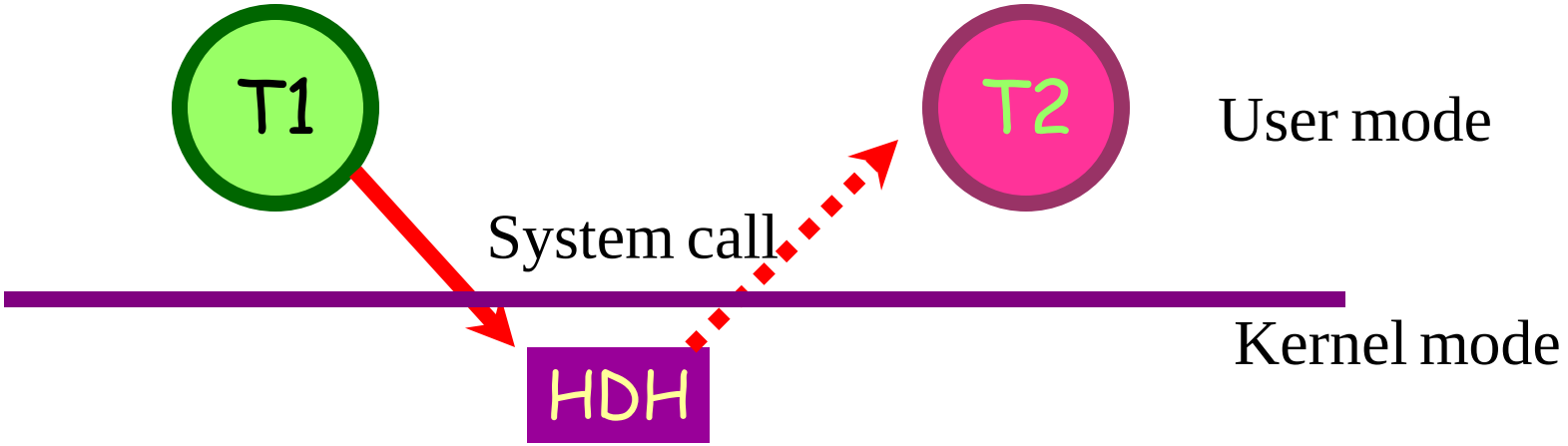
  VD: trong Solaris 2, tạo process chậm hơn 30 lần, chuyển chậm hơn 5 lần
- Trong multiprocessor: có thể thực hiện song song

Tiểu trình người dùng (User thread)



Khái niệm tiểu trình được hỗ trợ bởi một thư viện hoạt động trong user mode

Tiểu trình hạt nhân(Kernel thread)



Khái niệm tiểu trình được xây dựng bên trong hạt nhân