(B) Thread (Luồng)

- Khái niệm tổng quan
- Các mô hình multithread
- Pthread (POSIX thread)
- Multithreading trong Solaris

Xem xét lại khái niệm quá trình

- Khái niệm quá trình truyền thống: quá trình gồm
  1. Không gian địa chỉ
     - chứa code, data (Unix: text section, data section)
  2. Một luồng thực thi duy nhất (single thread of execution)
     - program counter
     - các register
     - stack
  3. Các tài nguyên khác (các open file, các quá trình con,...)

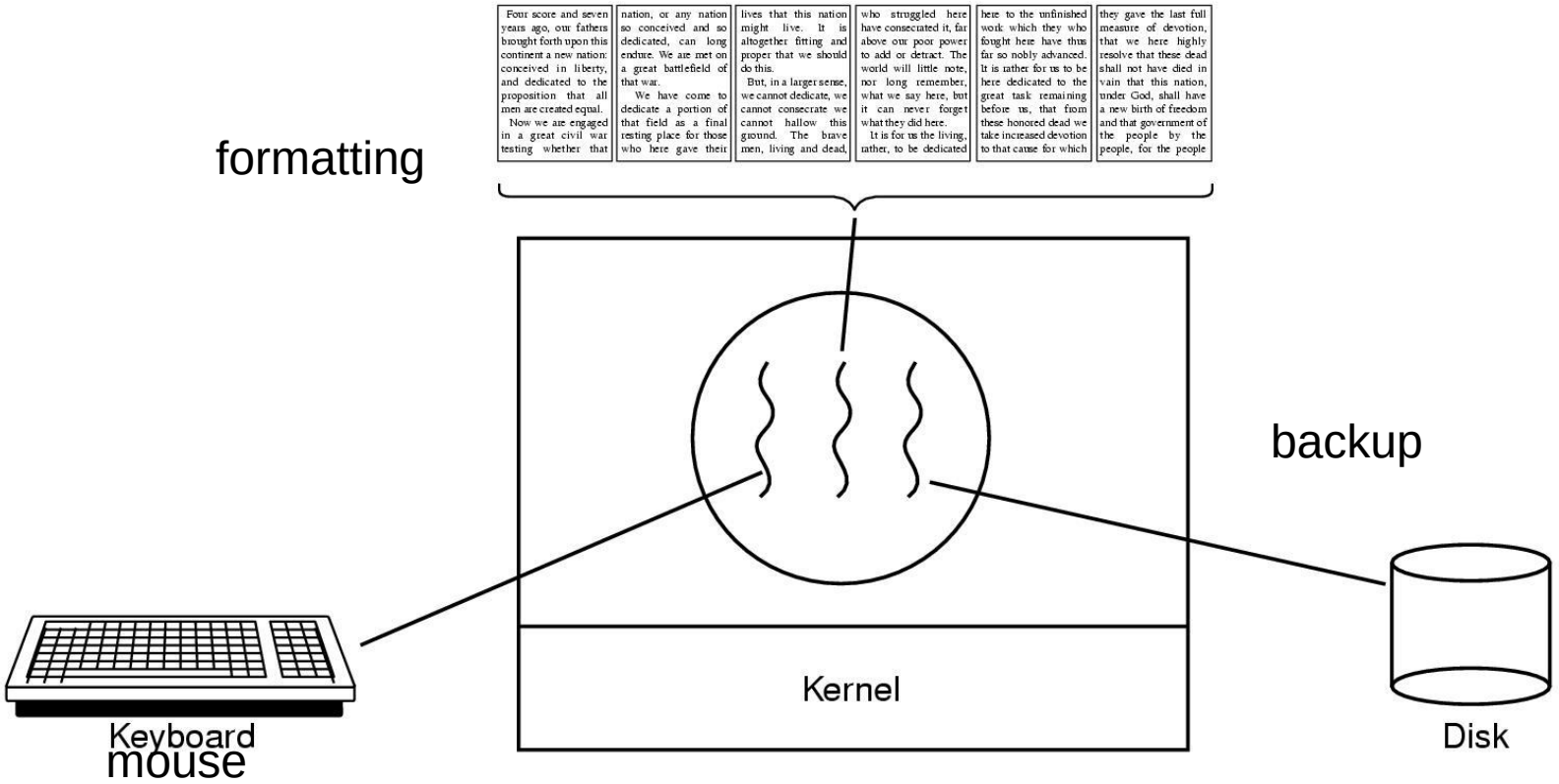
Mở rộng khái niệm quá trình

- Mở rộng khái niệm quá trình truyền thống bằng cách hiện thực nhiều luồng thực thi trong cùng một môi trường của quá trình.
- Quá trình gồm
  1. Không gian địa chỉ
  2. Một hay nhiều luồng thực thi, mỗi luồng thực thi (thread) có riêng:
     - program counter
     - các register
     - stack
  3. Các tài nguyên khác (các open file, các quá trình con,...)

Quá trình đa luồng

- Khi quá trình khởi đầu chỉ có main (hay initial) thread thực thi
  - Main thread sẽ tạo các thread khác.
- Các thread trong cùng một process chia sẻ code, data và tài nguyên khác (các file đang mở,...) của process.
- Quá trình *đa luồng* (multithreaded process) là quá trình có nhiều luồng.

Ví dụ: Sử dụng thread



A word processor with three threads

Process & thread information

Per process items
Address space
Open files
Child processes
Signals & handlers
Accounting info
Global variables

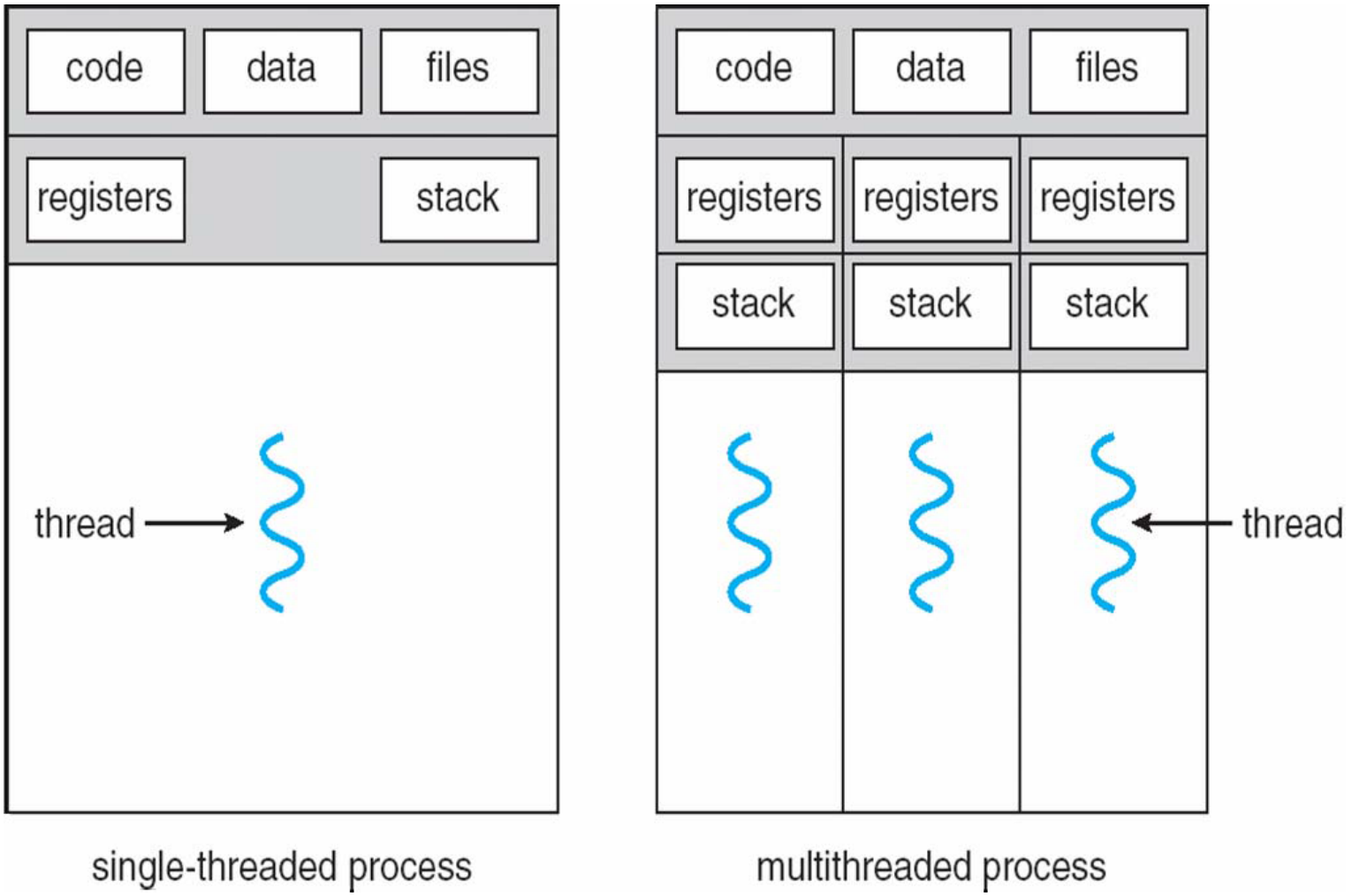
Per thread items
Program counter
Registers
Stack & stack pointer
State

Per thread items
Program counter
Registers
Stack & stack pointer
State

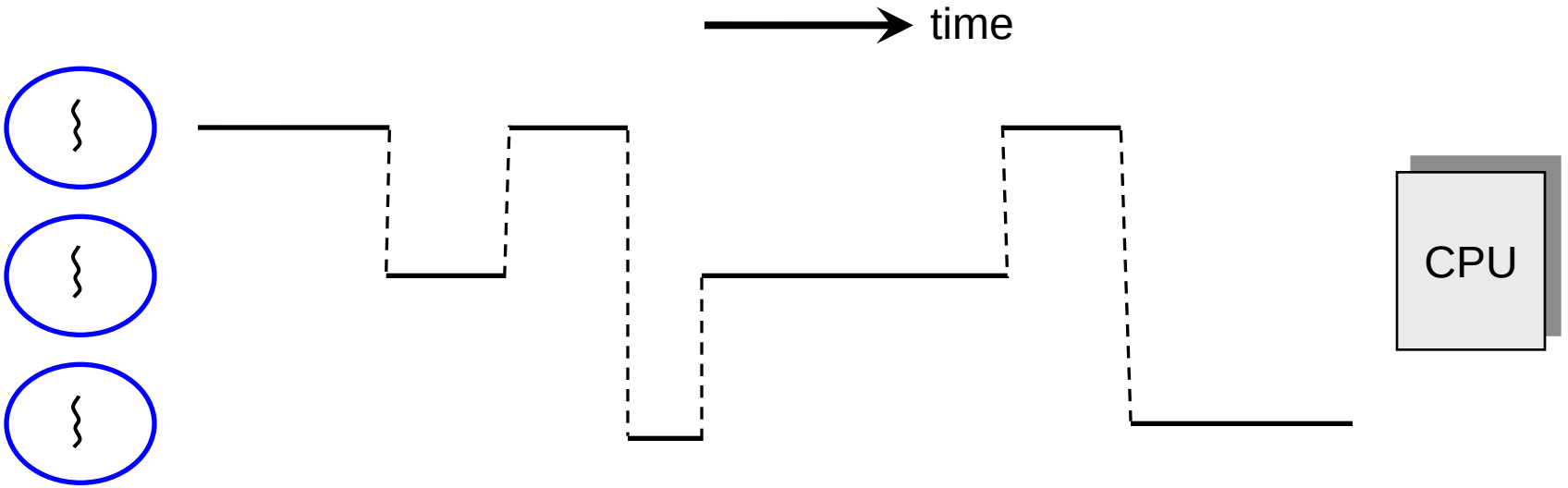
Per thread items
Program counter
Registers
Stack & stack pointer
State

Quá trình có ba thread

Quá trình đơn & đa luồng

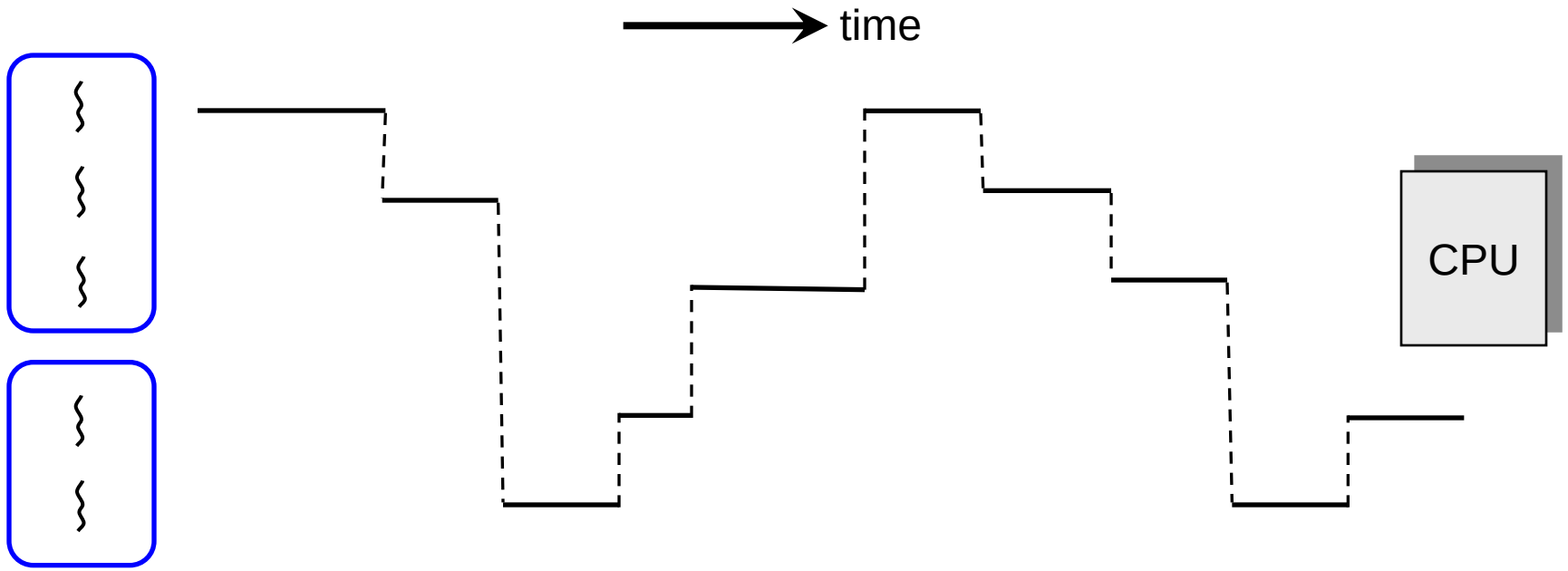


Multiplexing CPU giữa các thread



ba quá trình
single-threaded

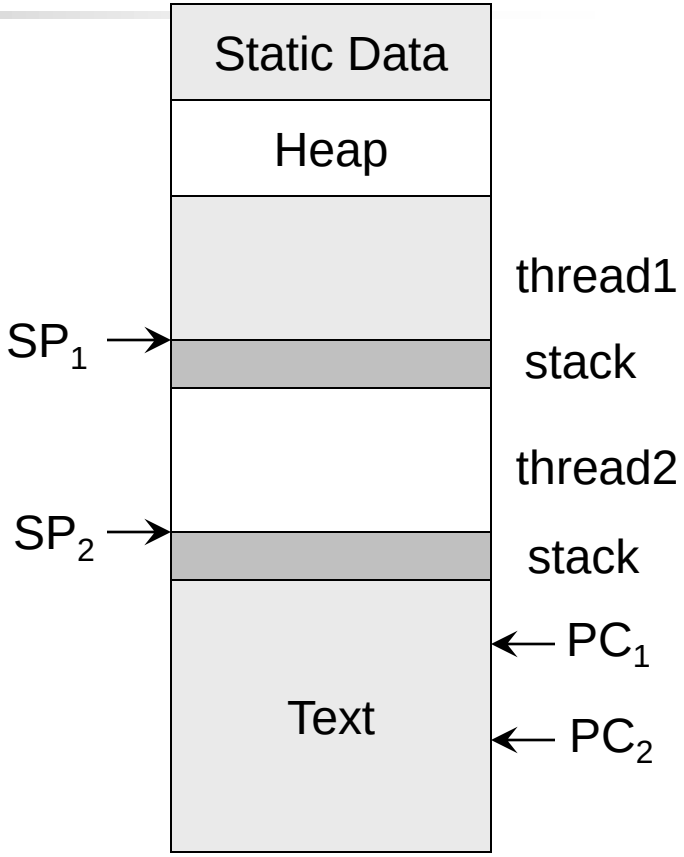
Multiplexing CPU giữa các thread (tt.)



hai quá trình multithreaded

Ví dụ: Pthread

```
#include <stdio.h>
void* thread1(){
    int i;
    for (i = 0; i < 10; i++){
        printf("Thread 1\n");
        sleep(1);
    }
}
void* thread2(){
    int i;
    for (i = 0; i < 10; i++){
        printf("Thread 2\n");
        sleep(1);
}
int main(){
    pthread_t th1, th2;
    pthread_create(&th1, NULL, thread1, NULL);
    pthread_create(&th2, NULL, thread2, NULL);
    sleep(20);
    return 0;
}
```



Sơ đồ bộ nhớ

Chương trình này khi chạy có bao nhiêu thread?

Ưu điểm của thread

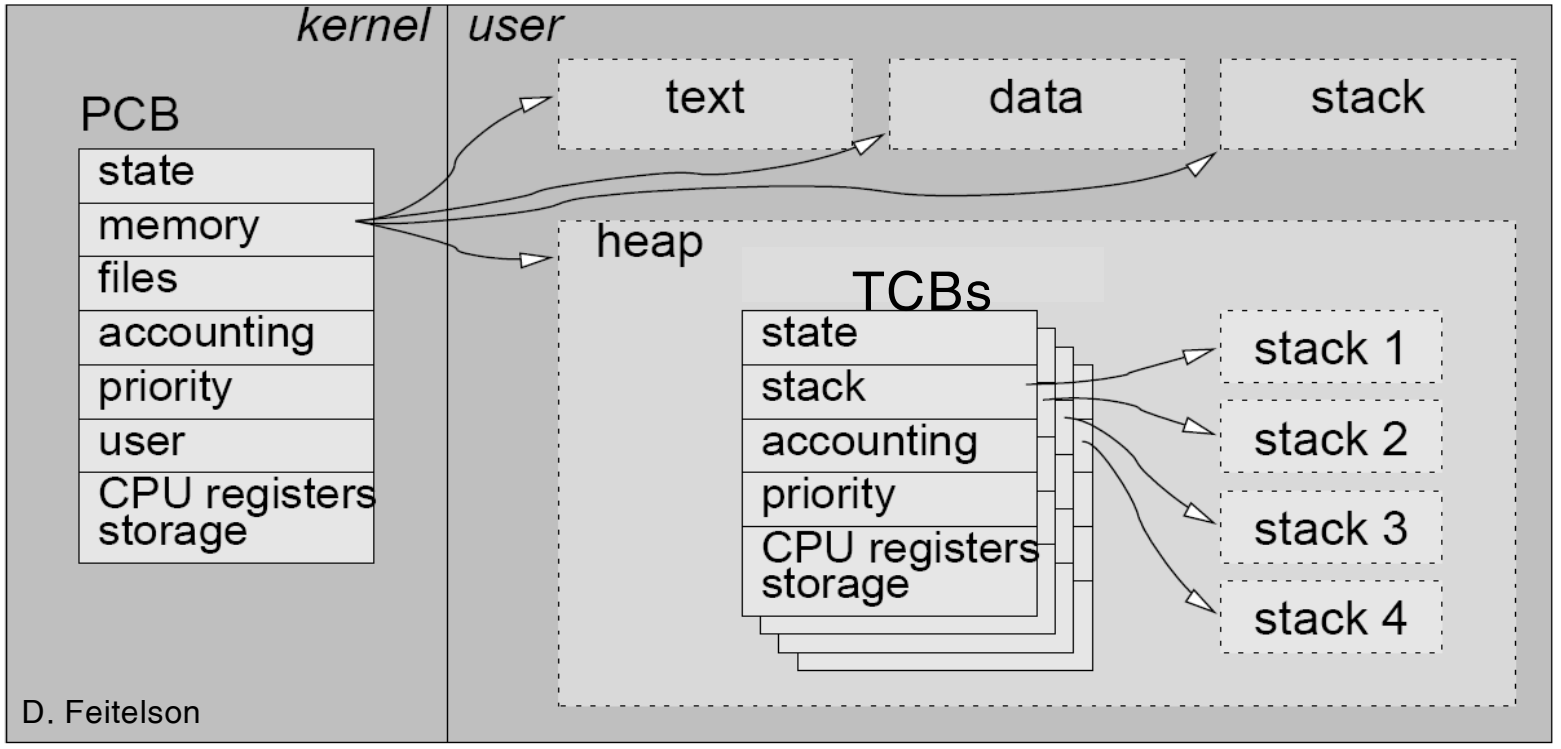
- **Tính đáp ứng** (responsiveness) cao cho các ứng dụng tương tác multithreaded
- **Chia sẻ** tài nguyên (resource sharing)
  - ví dụ memory
- **Tiết kiệm** chi phí hệ thống (economy)
  - Chi phí tạo/quản lý thread nhỏ hơn so với quá trình
  - Chi phí chuyển ngữ cảnh giữa các thread nhỏ hơn so với quá trình
- **Tận dụng** được đa xử lý (multiprocessor)
  - Mỗi thread chạy trên một processor riêng, do đó tăng mức độ song song của chương trình.

User thread

- Một *thư viện thread* (thread library, run-time system) được hiện thực trong user space để hỗ trợ các tác vụ lên thread
  - Thư viện thread cung cấp các hàm khởi tạo, định thời và quản lý thread như
    - **thread_create**
    - **thread_exit**
    - **thread_wait**
    - **thread_yield**
  - Thư viện thread dùng *Thread Control Block* (TCB) để lưu thông tin về user thread (program counter, các register, stack)

User thread (tt.)

- Cấu trúc dữ liệu và memory layout để hiện thực user thread

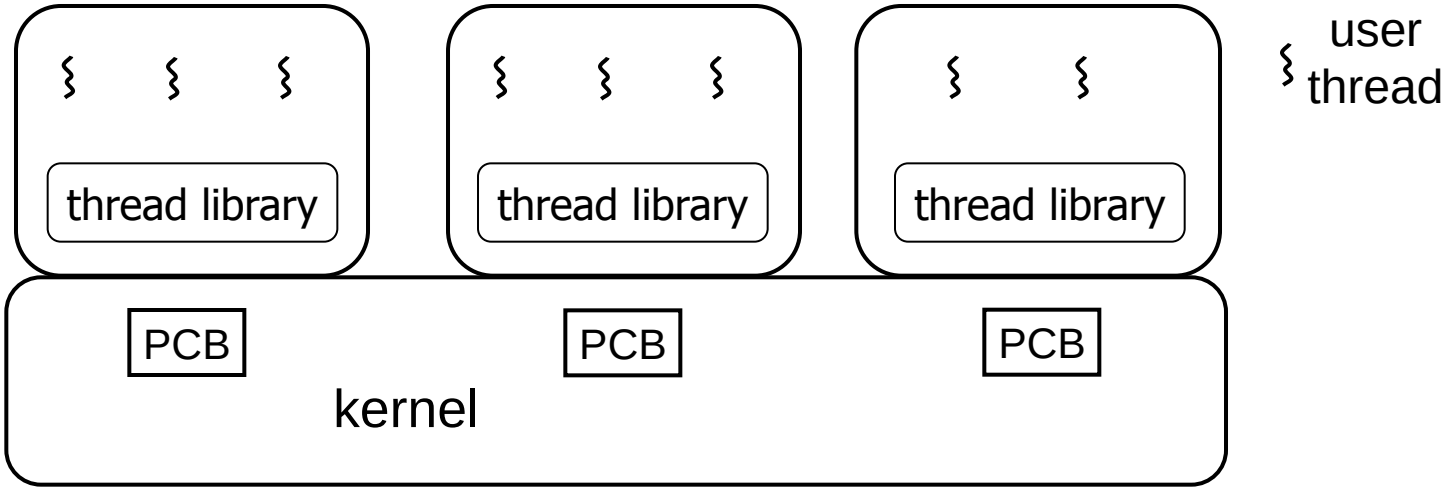


User thread (tt.)

- Kernel không biết sự có mặt của user thread
  - Kernel chỉ biết PCB của quá trình
- Ví dụ thư viện user thread
  - POSIX Pthread

User thread (tt.)

- Vấn đề: hệ điều hành chỉ cấp phát duy nhất một PCB cho mỗi process (→ main/initial thread).
  - *Blocking problem:* Khi một thread trở nên blocked thì mọi thread khác của process cũng sẽ trở nên blocked.

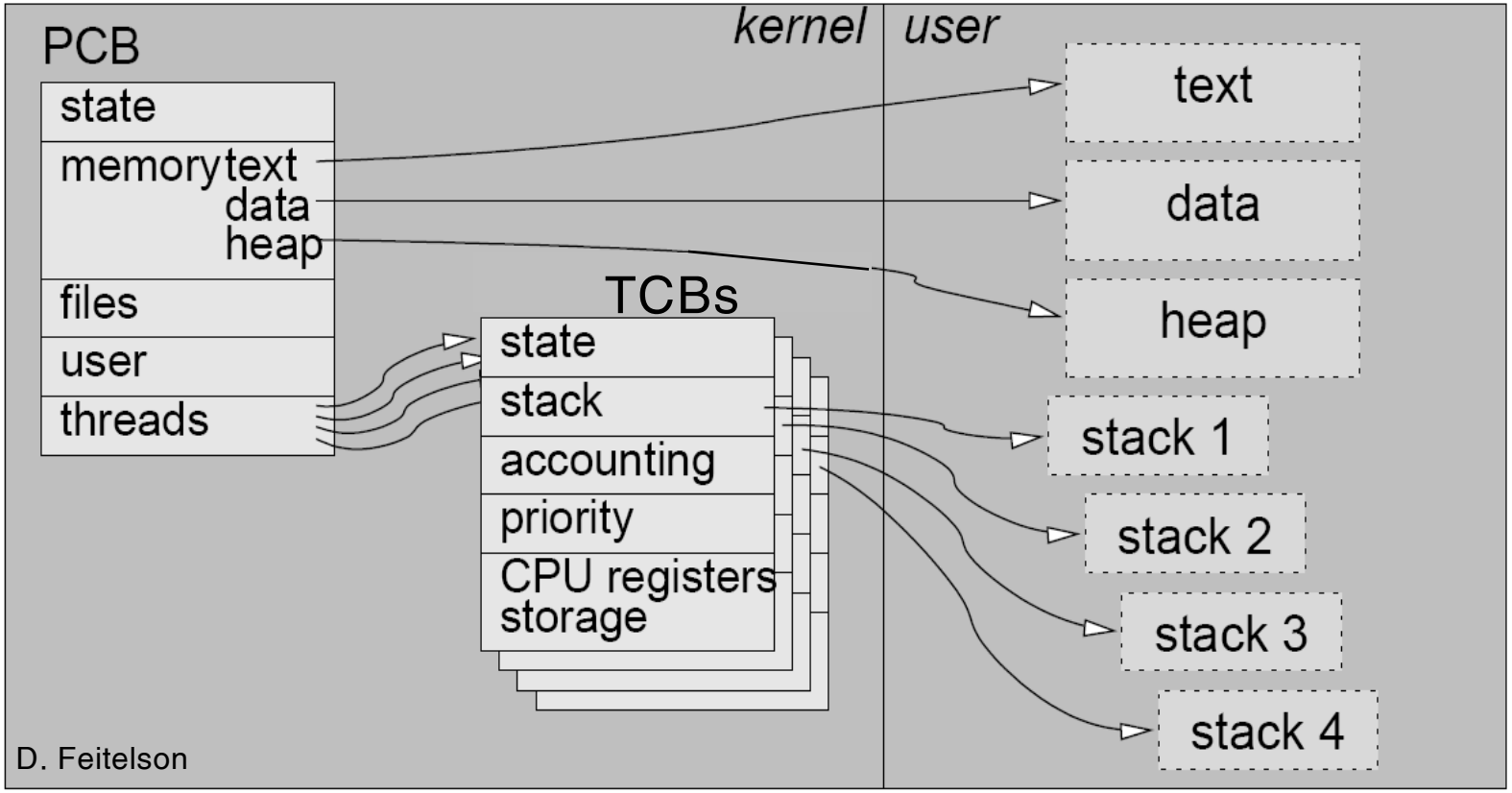


Kernel thread

- Khi kỹ thuật multithreading được hệ điều hành trực tiếp hỗ trợ
  - Kernel quản lý cả process và các thread – **kernel thread**
  - Việc định thời CPU được kernel thực hiện trên thread

Kernel thread (tt.)

- Cấu trúc dữ liệu và memory layout để hiện thực kernel thread



Kernel thread (tt.)

- Khi multithreading được hỗ trợ bởi kernel
  - Khởi tạo và quản lý các thread chậm hơn
  - Tận dụng được lợi thế của kiến trúc multiprocessor
  - Thread bị blocked không kéo theo các thread khác bị blocked.

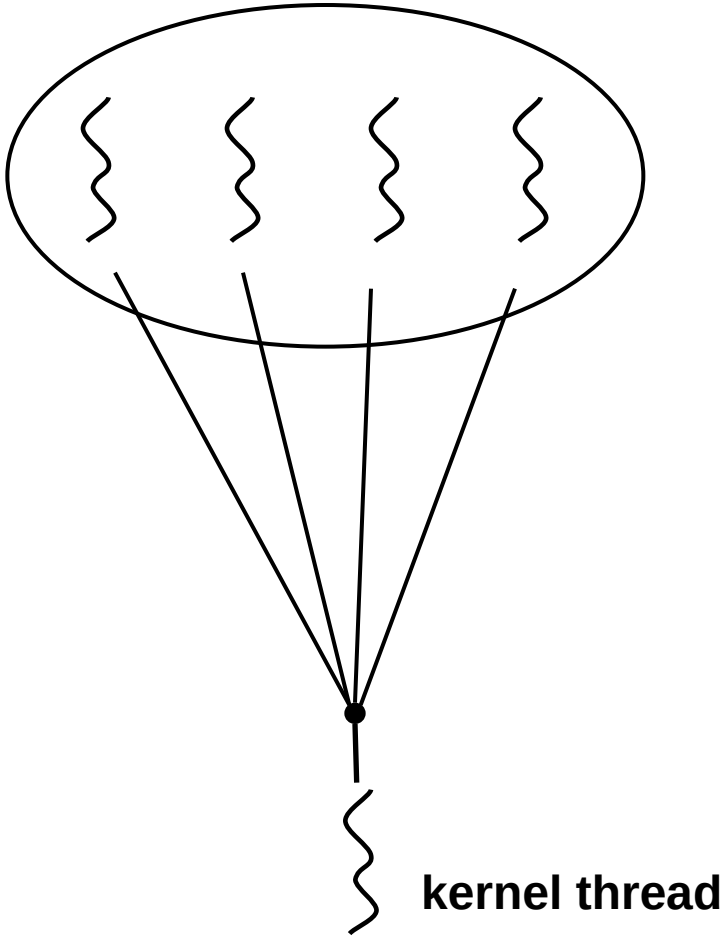
- Một số hệ thống multithreading
  - Windows 9x/NT/200x
  - Solaris
  - Linux

Hiện thực thread

- Nhắc lại **kernel thread** – thread được hệ điều hành quản lý
- Multithreading có thể hiện thực theo một trong các mô hình sau
  - Mô hình *many-to-one*
  - Mô hình *one-to-one*
  - Mô hình *many-to-many*

Mô hình many-to-one

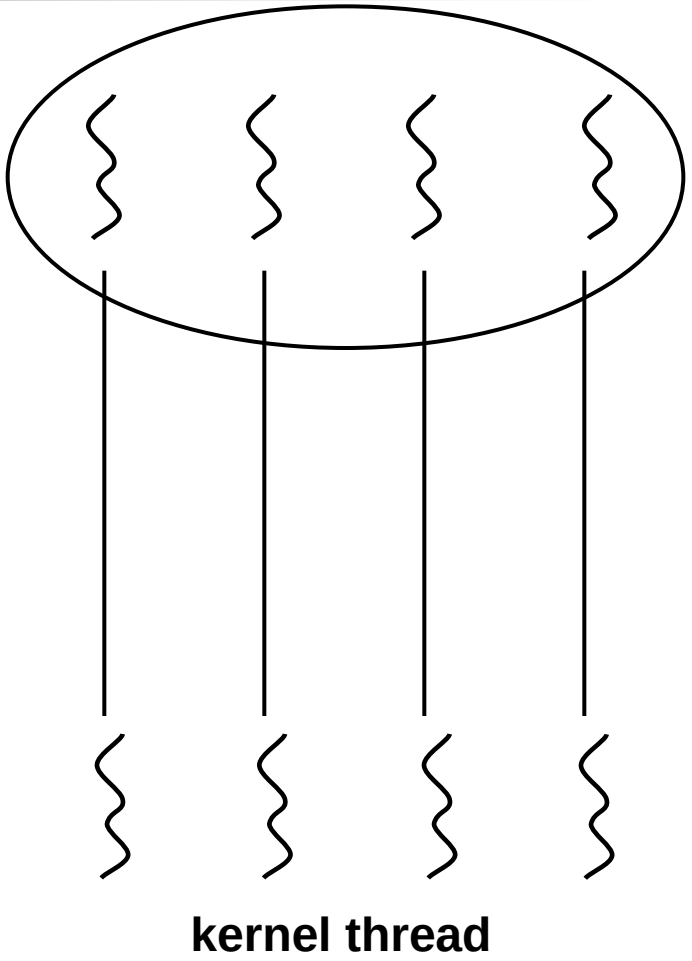
- Nhiều user-level thread "chia sẻ" một kernel thread để thực thi
  - Việc quản lý thread được thực hiện thông qua các hàm của một thread library được gọi ở user level.
  - Blocking problem: Khi một thread trở nên blocked thì mọi thread khác của process cũng sẽ trở nên blocked.
- Có thể được hiện thực đối với hầu hết các hệ điều hành.



Mô hình one-to-one

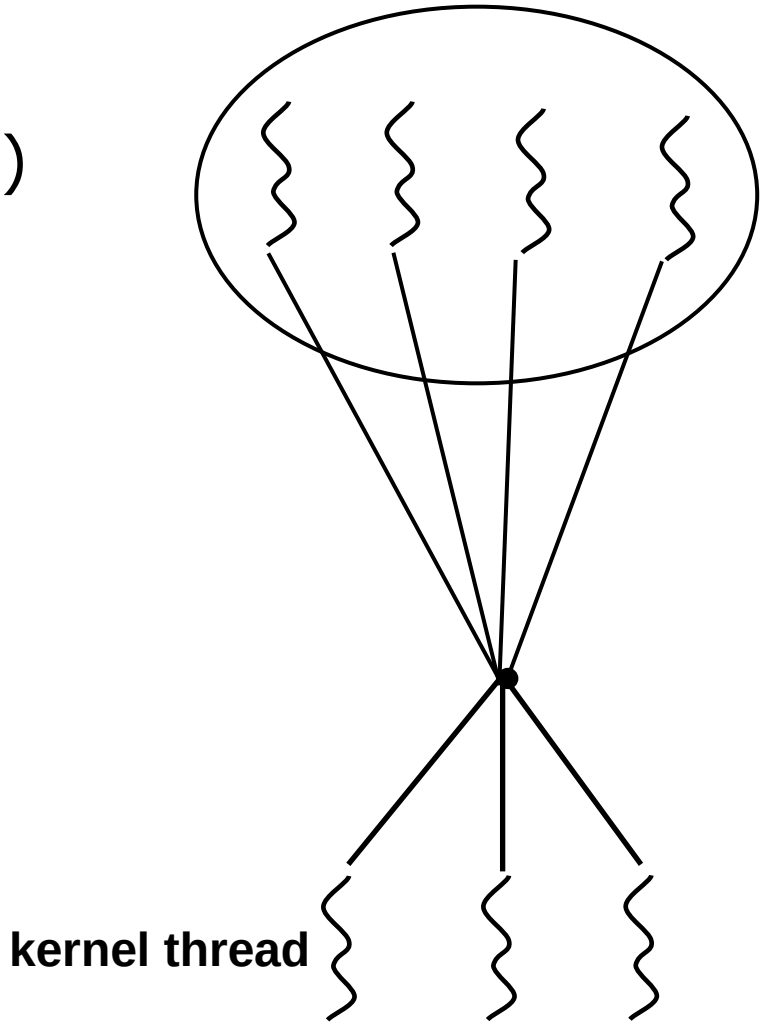
- Mỗi user-level thread thực thi thông qua một kernel thread riêng của nó
  - Mỗi khi một user thread được tạo ra thì cũng cần tạo một kernel thread tương ứng
- Hệ điều hành phải có cơ chế cung cấp được nhiều kernel thread cho một quá trình

  Ví dụ: Windows NT/2000



Mô hình many-to-many

- Nhiều user-level thread được phân chia thực thi (multiplexed) trên một số kernel thread.
  - Tránh được một số khuyết điểm của hai mô hình many-to-one và one-to-one
- Ví dụ
  - Solaris 2
  - Windows NT/2000 với package ThreadFiber



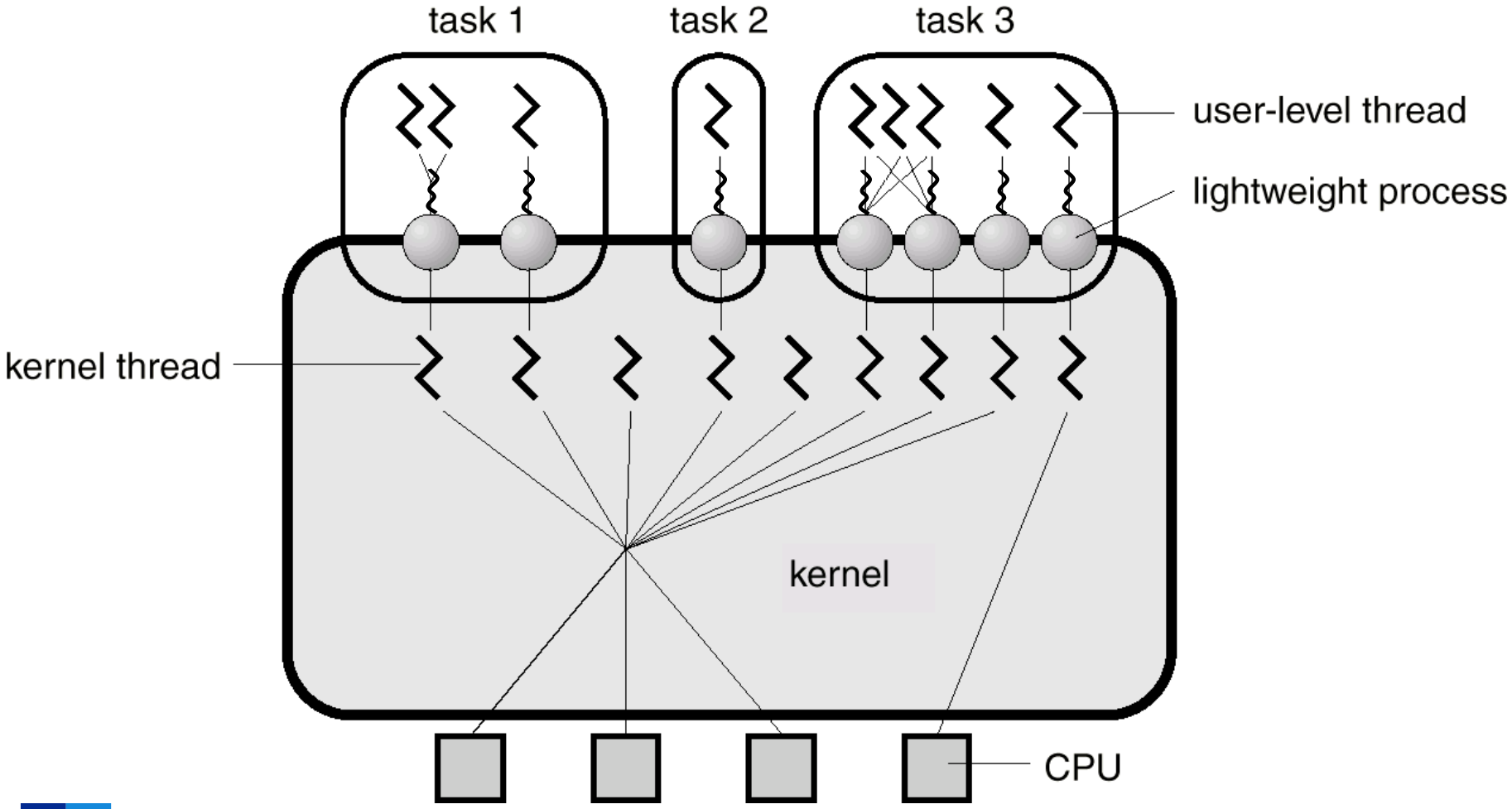
Pthread

- Chuẩn POSIX (IEEE 1003.1c) đặc tả API cho các thủ tục tạo thread và đồng bộ thread
- Phổ biến trong các hệ thống UNIX/Linux
- Là một thư viện hỗ trợ user-level thread
- Tham khảo thêm ví dụ về lập trình thư viện Pthread với ngôn ngữ C trong hệ thống Unix-like, trang 140, "Operating System Concepts", Silberschatz et al, 6th Ed, 2003.
- Biên dịch và thực thi chương trình multithreaded C trong Linux
  - **$ gcc source_file.c -lpthread –o output_file**
  - **$ ./output_file**

Thread trong Solaris

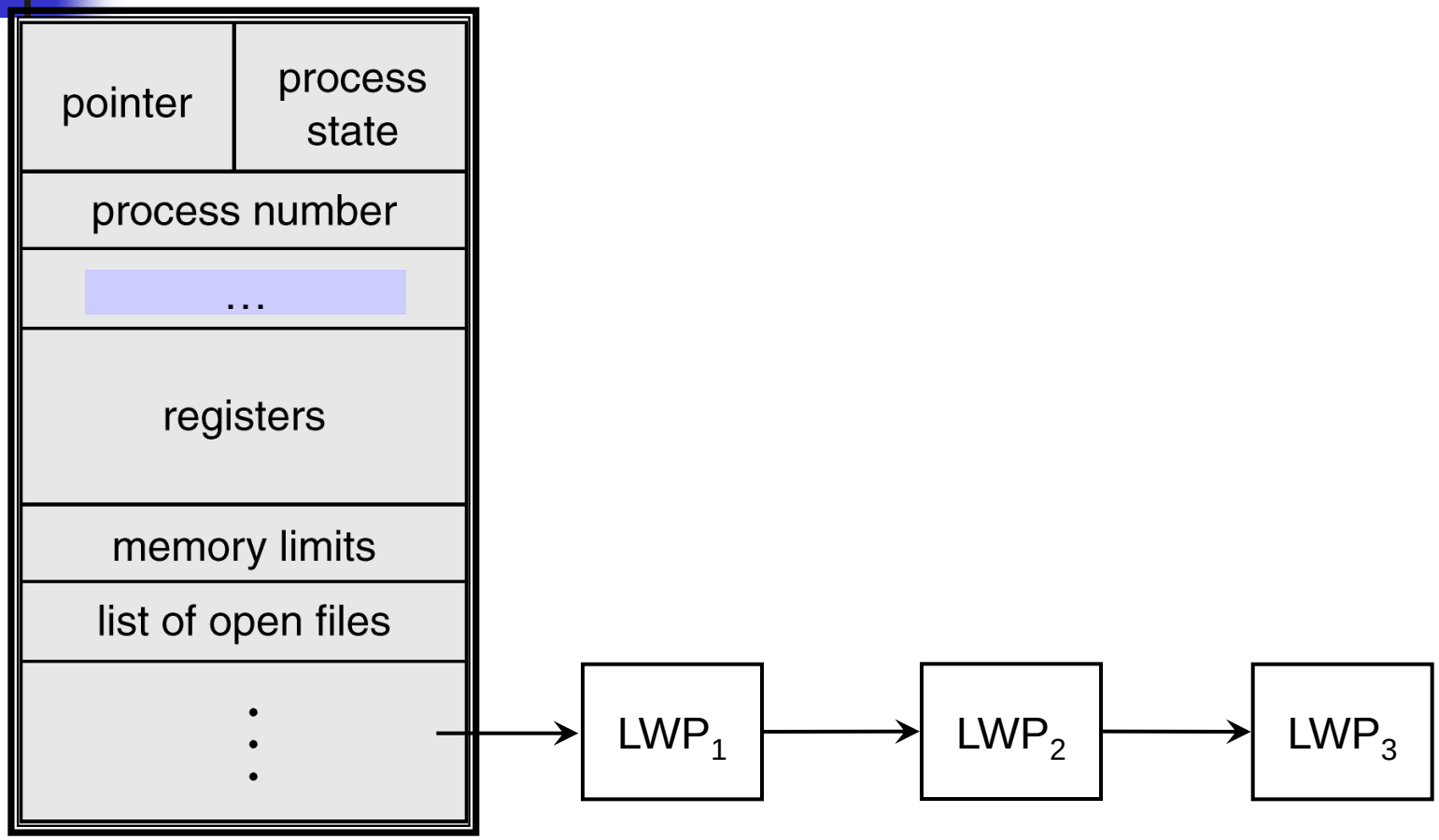
- User-level threads
  - Pthread và UI-thread
- *Lightweight process* (LWP)
  - Mỗi process chứa ít nhất một LWP
  - Thư viện thread có nhiệm vụ phân định user thread vào các LWP
    - User-level thread được gắn với LWP thì mới được thực thi.
  - Thư viện thread chịu trách nhiệm điều chỉnh số lượng LWP
- Kernel-level threads
  - Mỗi LWP tương ứng với một kernel-level thread
  - Ngoài ra, hệ thống còn có một số kernel thread dành cho một số công việc ở kernel (các thread này không có LWP tương ứng)
  - Đối tượng được định thời trong hệ thống là các kernel thread

Thread trong Solaris (tt.)



many-to-many

Thread trong Solaris (tt.)



Quá trình trong Solaris