Chương 2.B Thread

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

Xem xét lại khái niệm quá trình

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

Mở rộng khái niệm quá trình

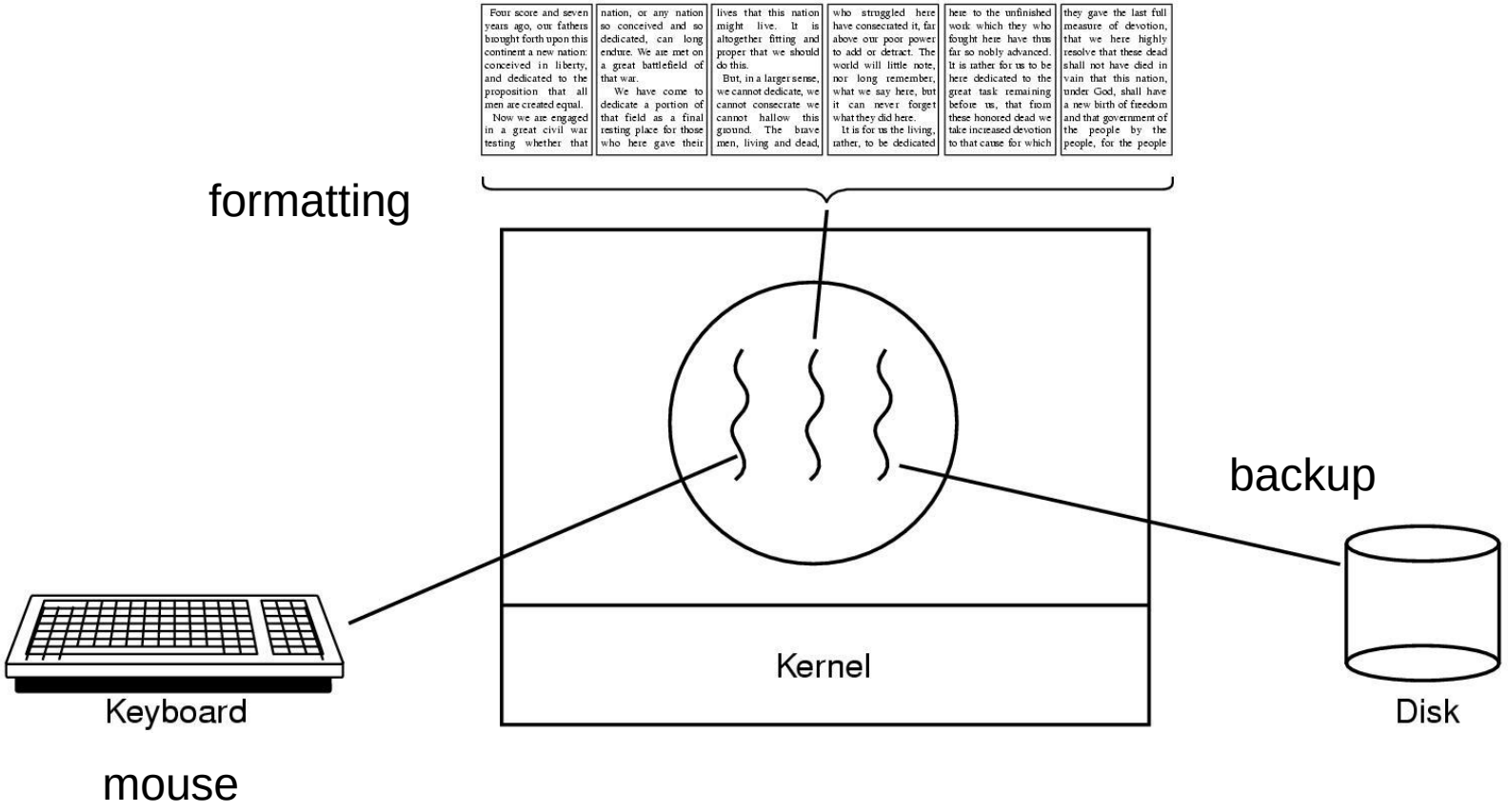
- Mở rộng khái niệm quá trình truyền thống bằng cách hiện thực nhiều luồng thực thi trong cùng một môi trường của quá trình.

- 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 (Multi-threaded process)

- 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.

Sử dụng thread



Trình soạn thảo văn bản với ba thread

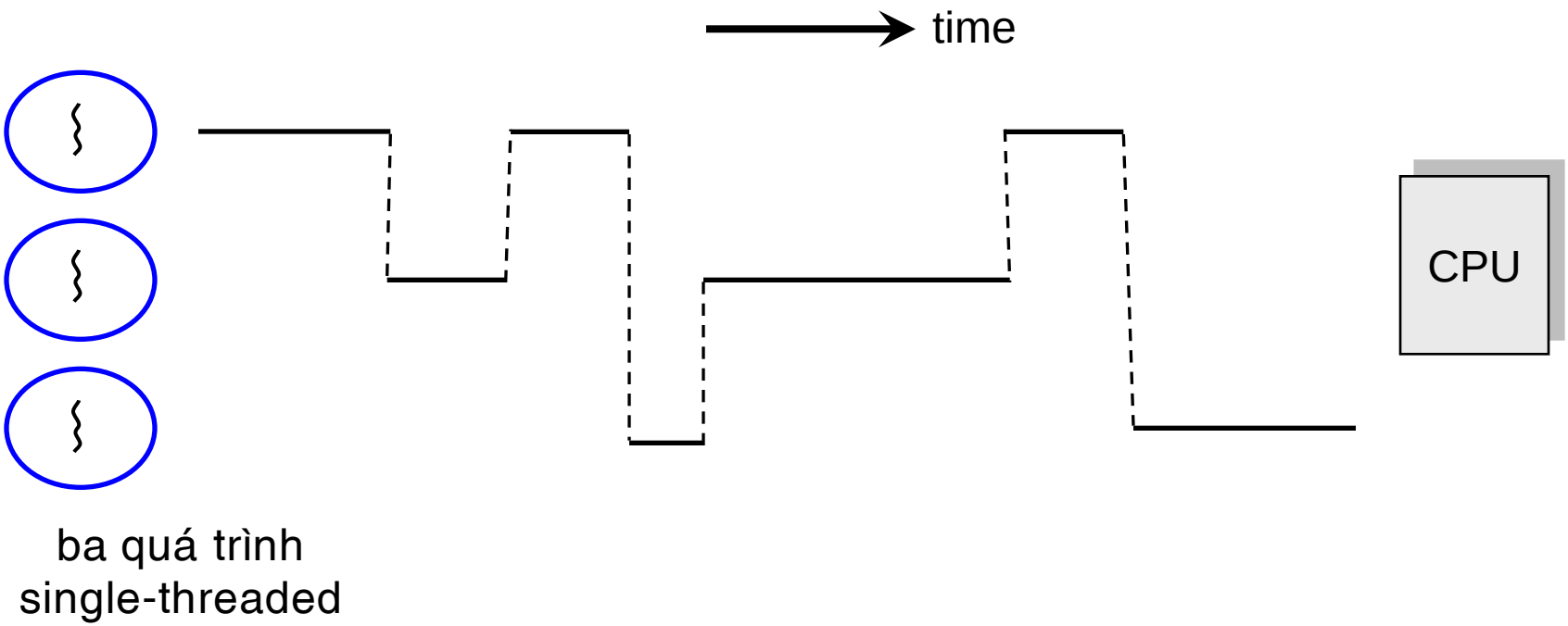
Process & thread information

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

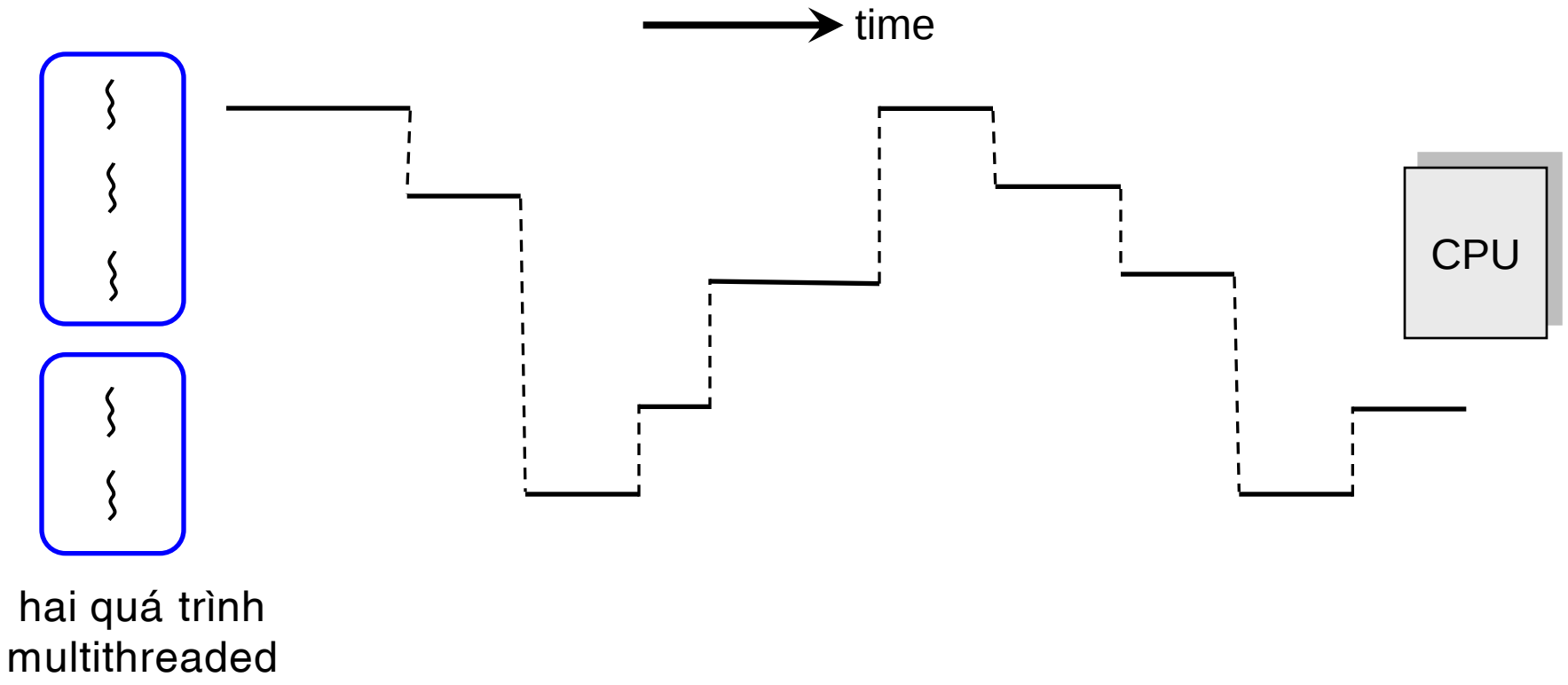
Per thread items	**Per thread items**	**Per thread items**
Program counter	Program counter	Program counter
Registers	Registers	Registers
Stack & stack pointer	Stack & stack pointer	Stack & stack pointer
State	State	State

Quá trình có ba thread

Chia sẻ CPU giữa các thread (1/2)

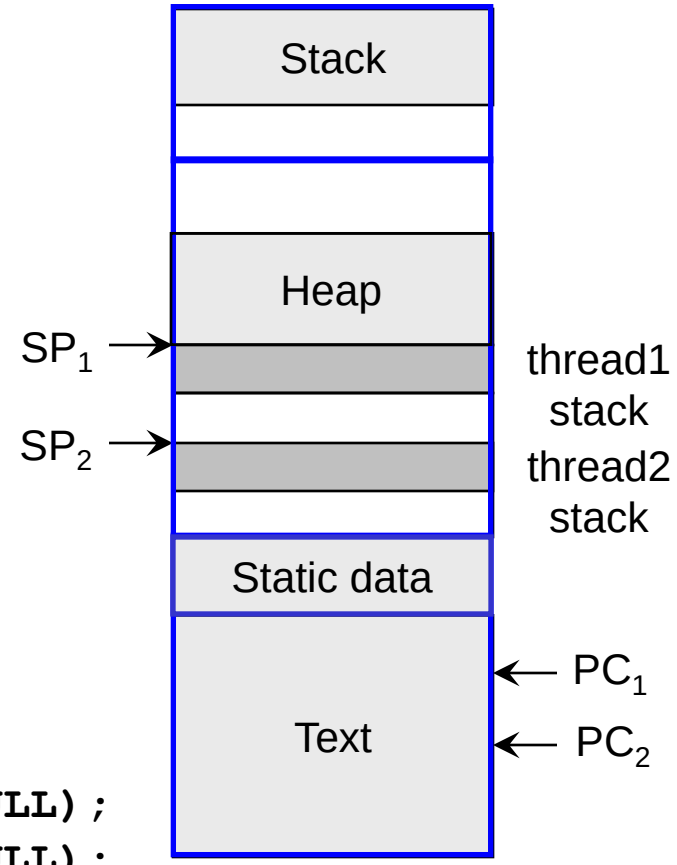


Chia sẻ CPU giữa các thread (2/2)



Ví dụ chương trình sử dụng 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

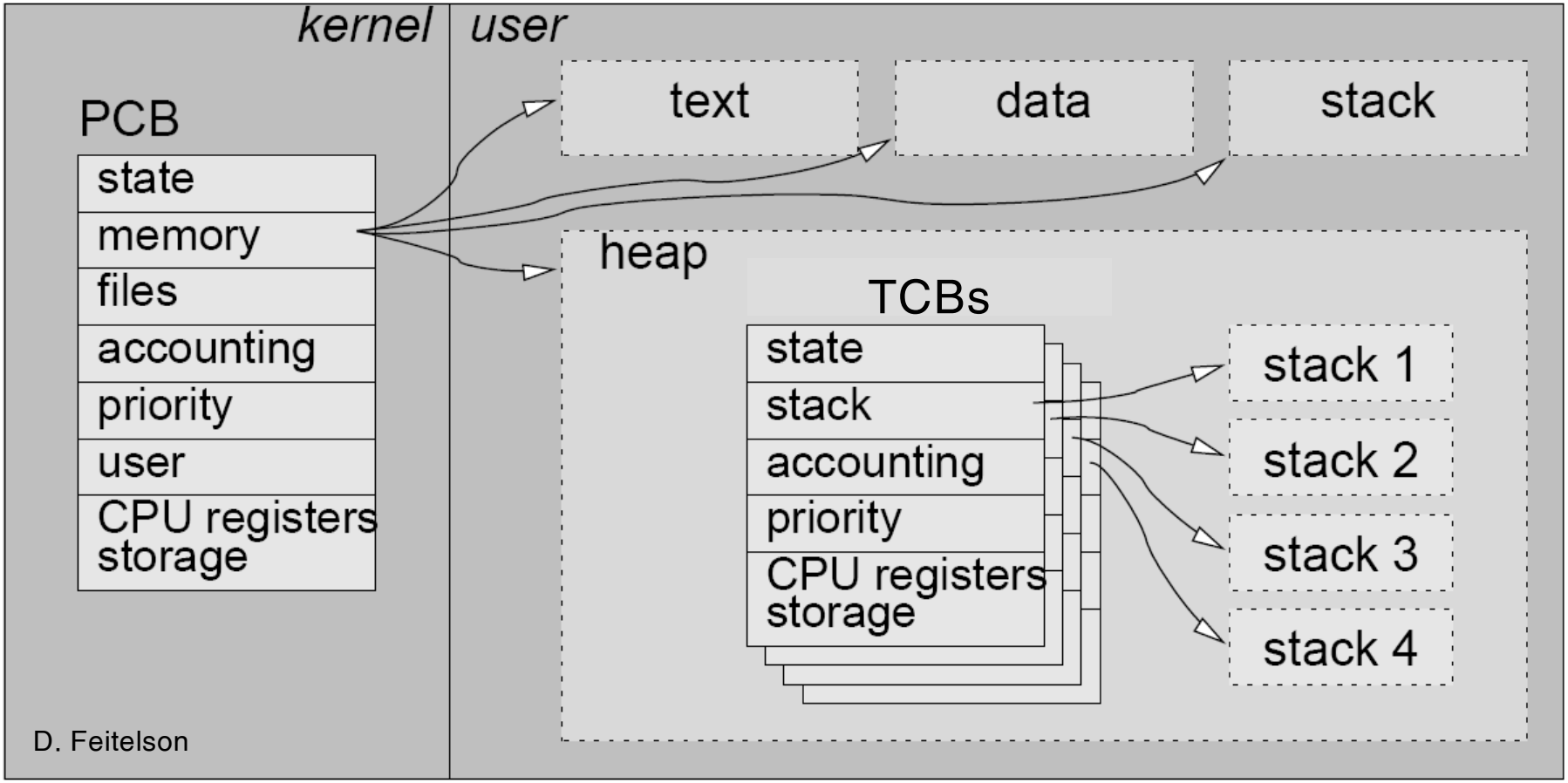
- Tính đáp ứng cao cho các ứng dụng tương tác
- Chia sẻ tài nguyên giữa các thread: vd memory
- Tiết kiệm chi phí hệ thống
  - Chi phí tạo/quản lý thread nhỏ hơn so với quá trình
  - Chi phí chuyển ngữ cảnh giữa các thread nhỏ hơn so với quá trình
- Tận dụng được đa xử lý (multiprocessor)
  - Mỗi thread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

User thread (1/4)

- Một *thư viện thread* (thread library, run-time system) được hiện thực trong user space để hổ 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 (2/4)

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

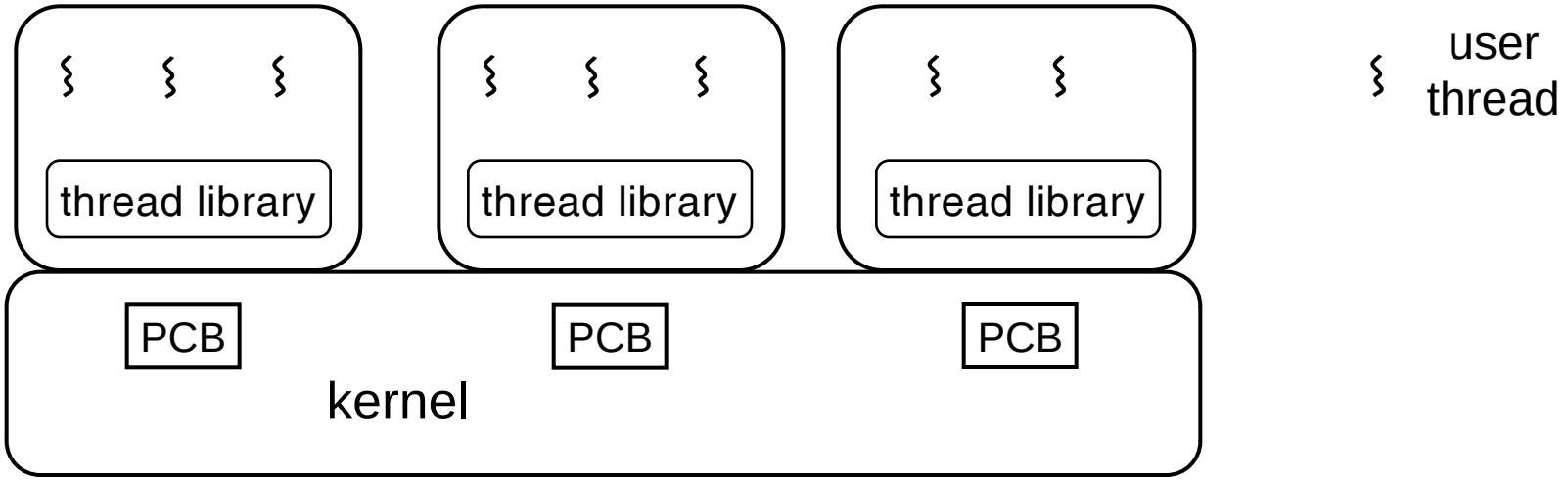


User thread (3/4)

- 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 (4/4)

- 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 sẽ không tiến triển được



Kernel thread (1/3)

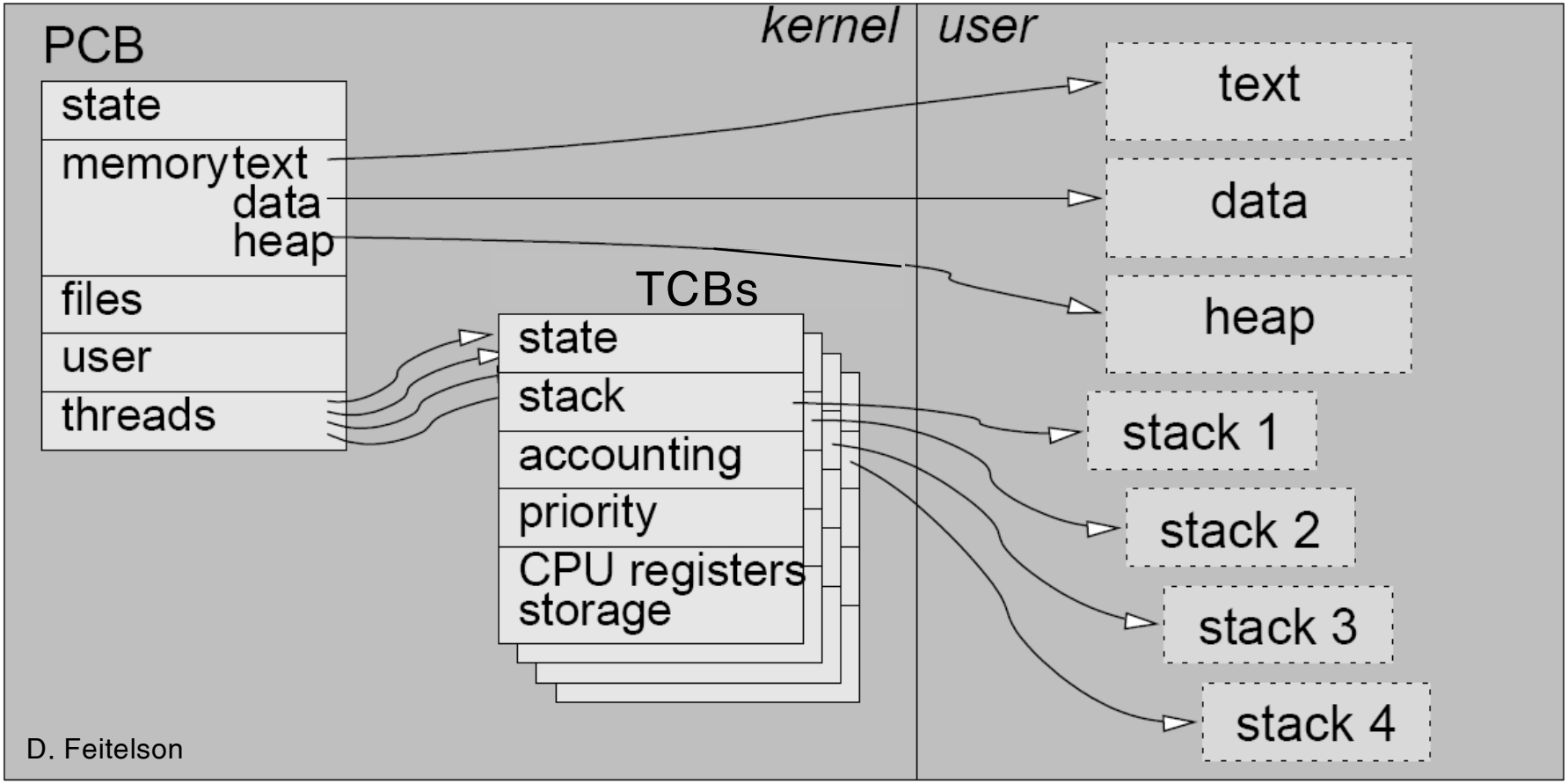
- Khi kỹ thuật multithreading được hệ điều hành trực tiếp hỗ trợ
  - Kernel quản lý cả process và các thread – **kernel thread**
  - Việc định thời CPU được kernel thực hiện trên thread

Kernel thread (2/3)

- Khi multithreading được hỗ trợ bởi kernel
  - Khởi tạo và quản lý các thread chậm hơn so với user thread do system call overhead chuyển user mode ↔ kernel mode
  - Tận dụng được lợi thế của kiến trúc multiprocessor
  - Dù một thread bị blocked, các thread khác của quá trình vẫn có thể tiến triển
- Một số hệ thống multithreading
  - Windows 9x/NT/200x
  - Solaris
  - Linux

Kernel thread (3/3)

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

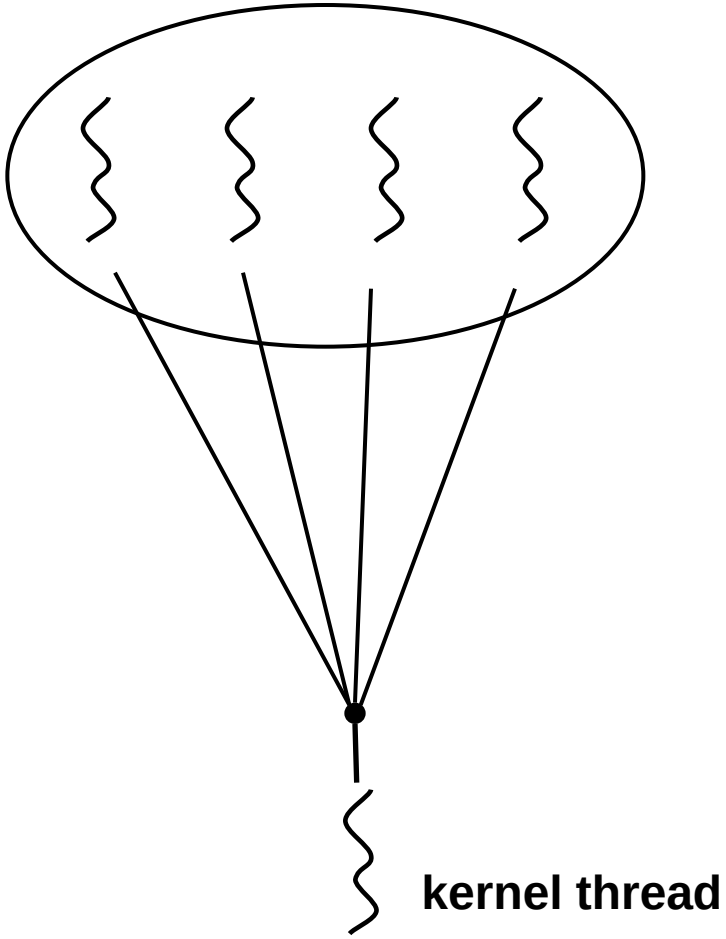


Hiện thực user thread

- Nhắc lại **kernel thread** – thread được hệ điều hành quản lý
- User(-level) 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*

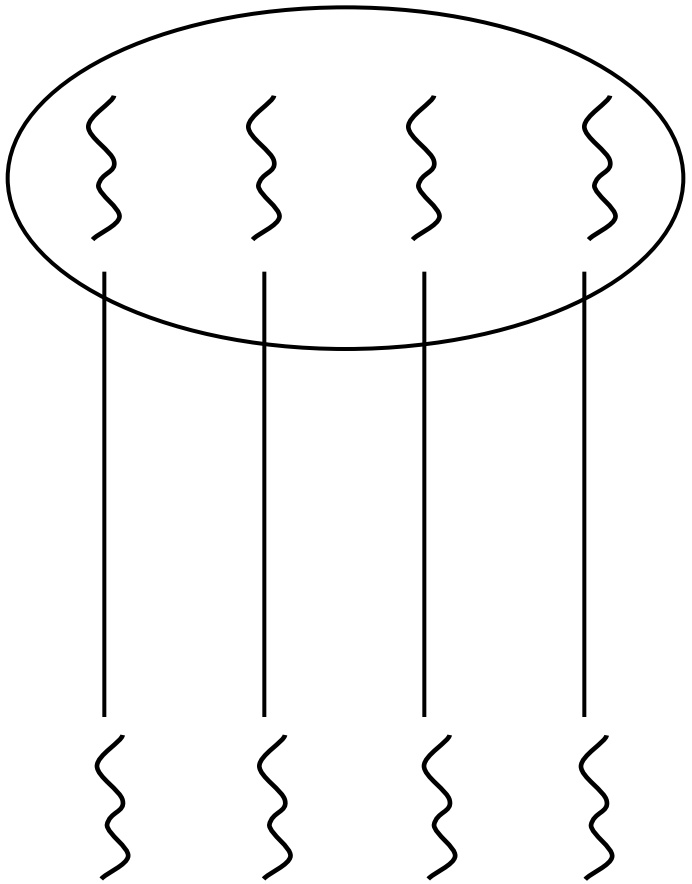
Mô hình many-to-one

- 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 không tiến triển được.
- Có thể được hiện thực đối với hầu hết các hệ điều hành.



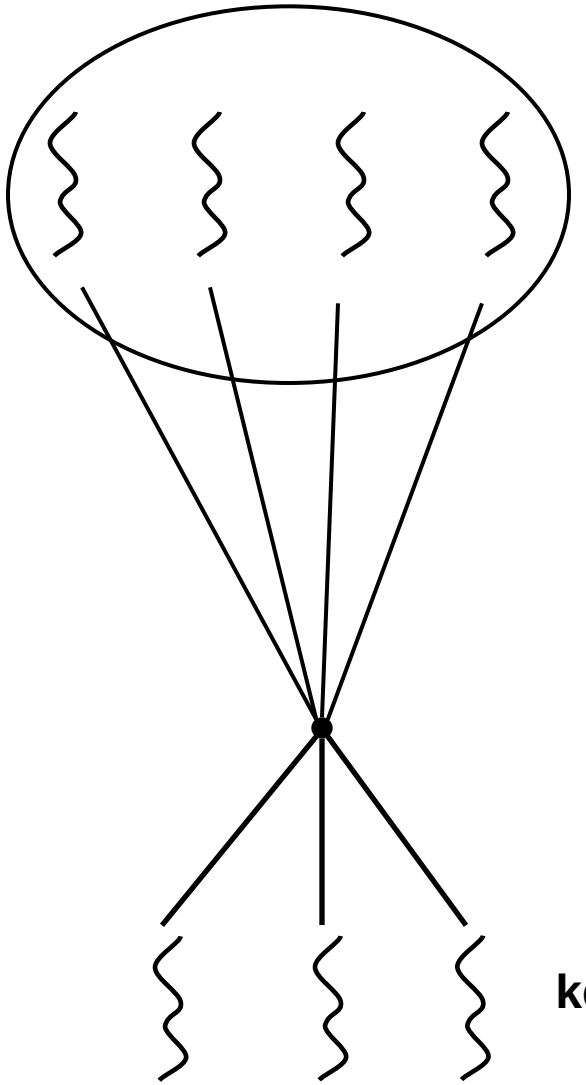
Mô hình one-to-one

- Mỗi user 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 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.
  - Kết hợp ưu điểm của user-level thread và của kernel-level thread
- 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.

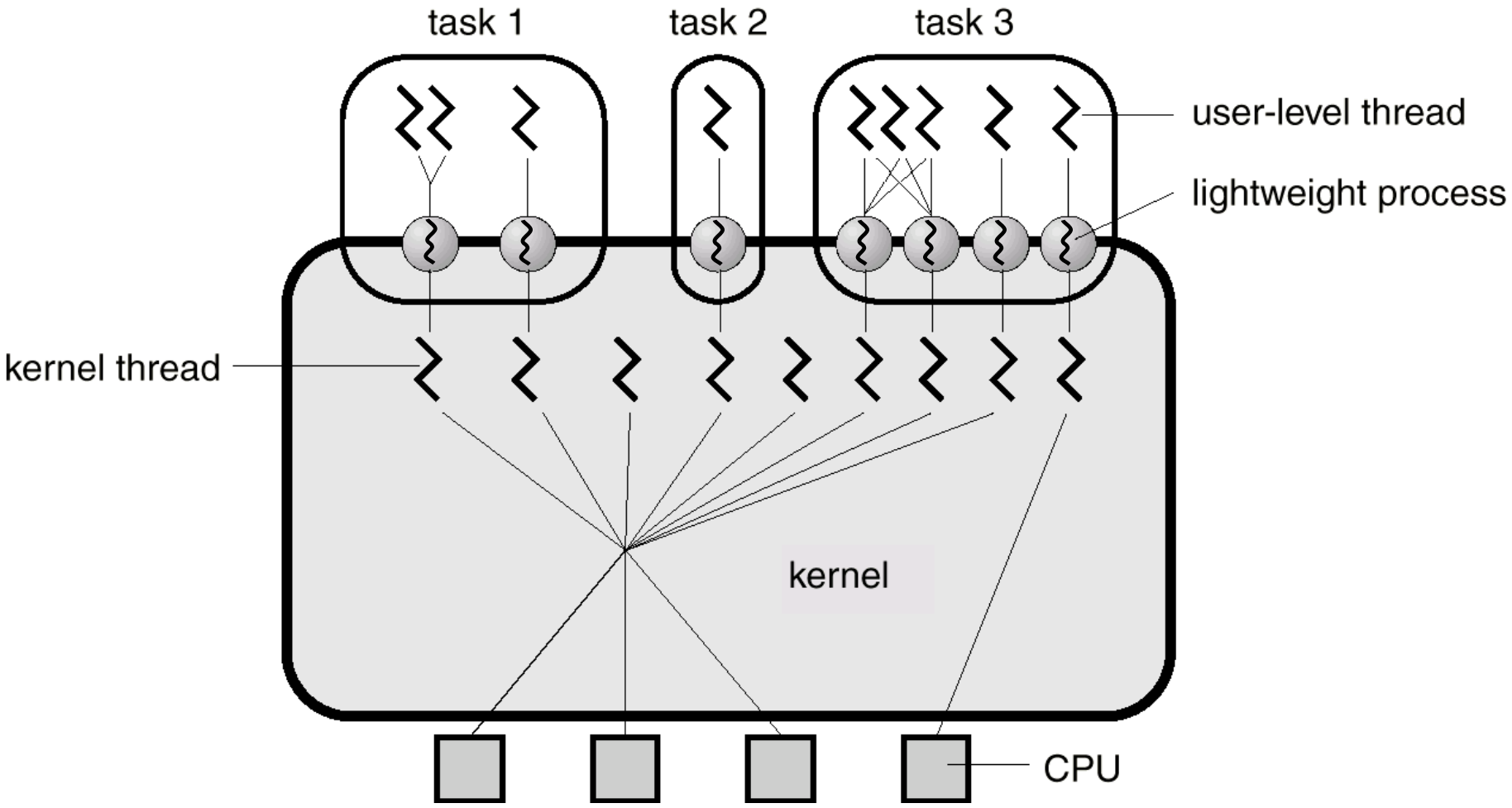
- Biên dịch và thực thi chương trình multithreaded C trong Linux

```
$ gcc source_file.c -lpthread -o output_file
$ ./output_file
```

Thread trong Solaris (1/3)

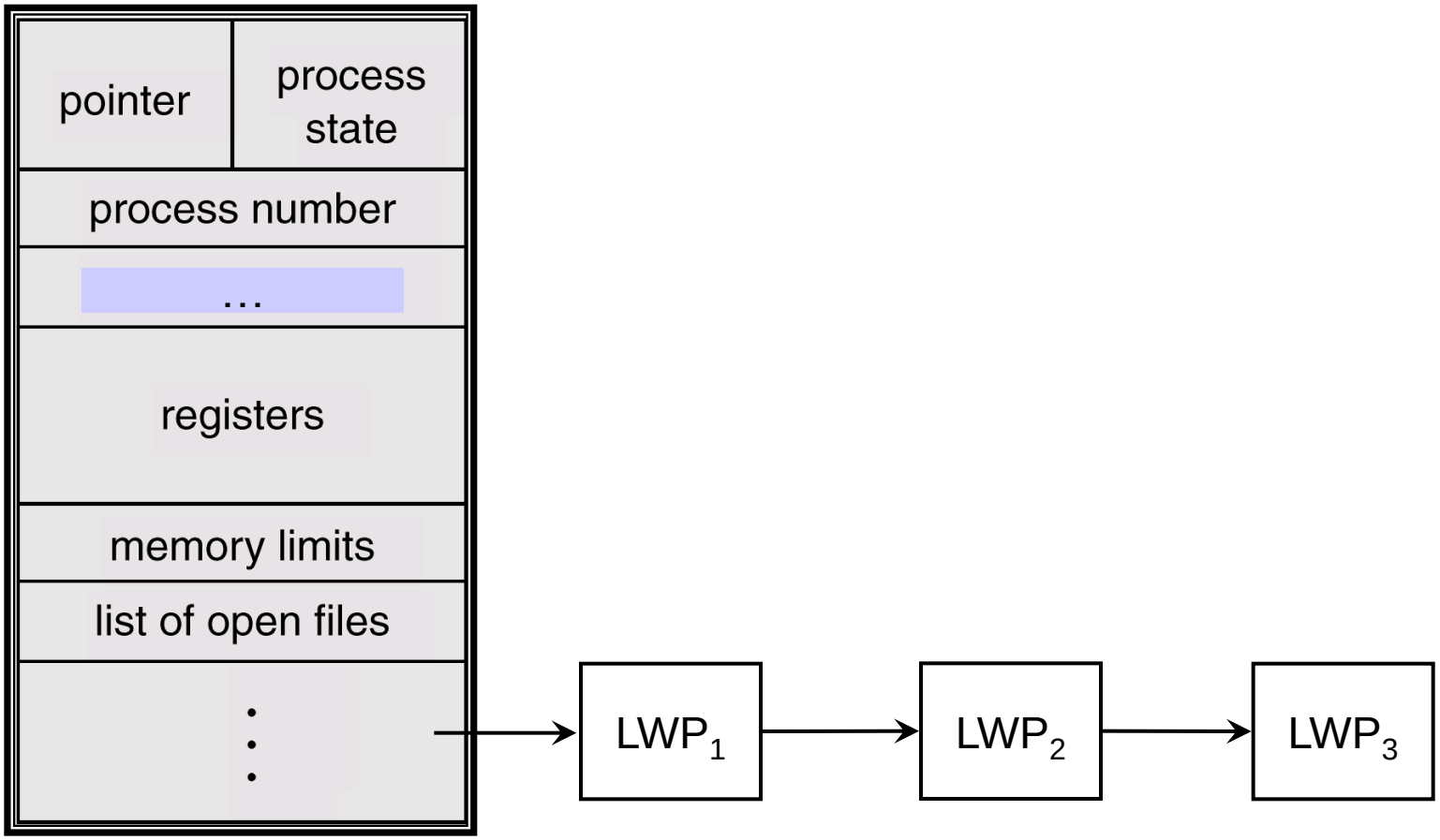
- User-level threads
  - Pthread và UI-thread
- *Lightweight process* (LWP)
  - Mỗi process chứa ít nhất một LWP
  - Thư viện thread có nhiệm vụ phân định user thread vào các LWP
    - User thread được gắn với LWP thì mới được thực thi.
  - Thư viện thread chịu trách nhiệm điều chỉnh số lượng LWP
- Kernel(-level) thread
  - Mỗi LWP tương ứng với một kernel 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 (2/3)



many-to-many

Thread trong Solaris (3/3)



Quá trình trong Solaris

Read more

- **POSIX thread programming**, *Blaise Barney, Lawrence Livermore National Laboratory*
  - https://computing.llnl.gov/tutorials/pthreads/