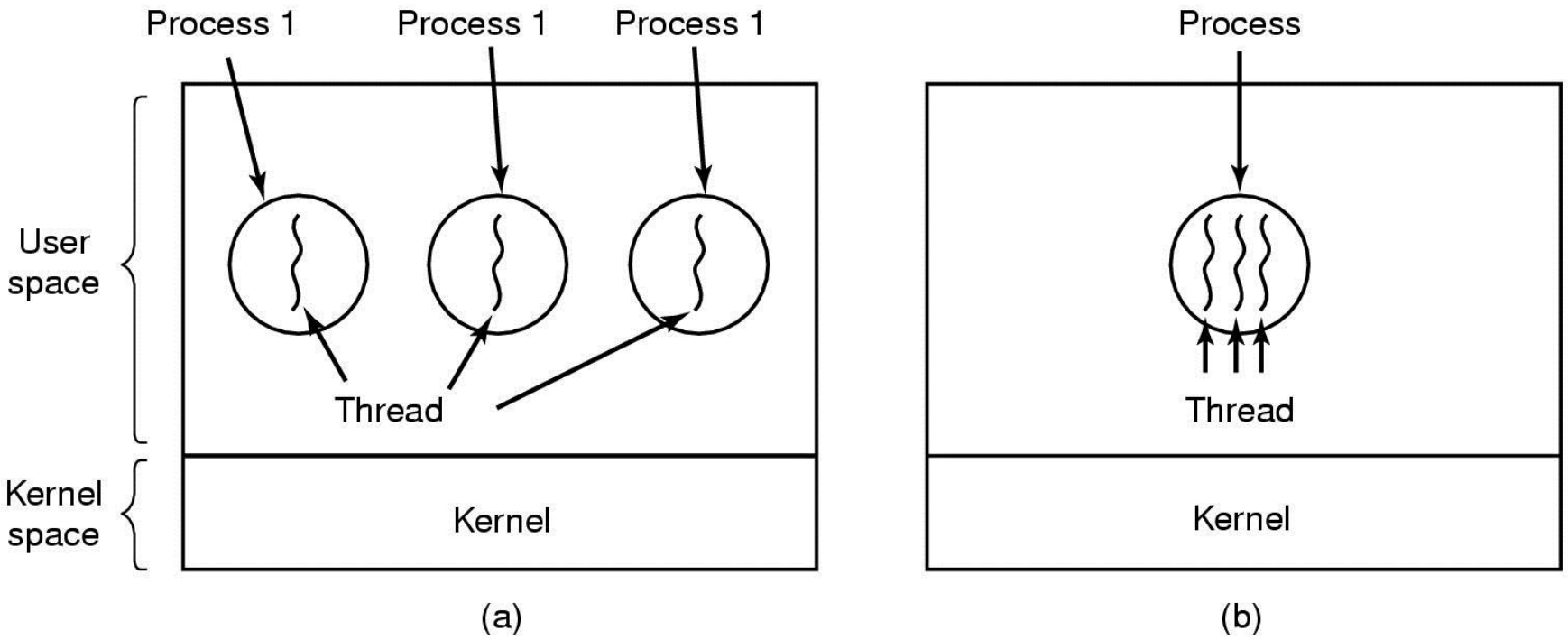
线程与线程模型

进程：地址空间、控制线程

资源分组处理、执行-〉线程（资源与执行分开）

进程：集中资源，资源被管理的单位；
线程：CPU上被调度执行的实体。（轻量级进程）

The Classical Thread Model



(a)Three processes each with one thread.(控制线程)

(b)One process with three threads.(地址空间相同，共享全局变量)

The Classical Thread Model (2)

Per process items	Per thread items
Address space	Program counter
Global variables	Registers
Open files	Stack
Child processes	State
Pending alarms	
Signals and signal handlers	
Accounting information	

The first column lists some items shared by all threads in a process.
The second one lists some items private to each thread.

线程模型

线程：试图实现的是，共享一组资源的多个线程的执行能力，以便这些线程为完成某一任务而共同工作。

线程与传统进程（只有一个线程的进程）一样有运行、阻塞、就绪和终止态。

每个线程有自己的PC、寄存器、堆栈。

堆栈：用来记录执行历史，其中一帧保存了一个已调用的但是还没有从中返回的过程。

The Classical Thread Model (3)

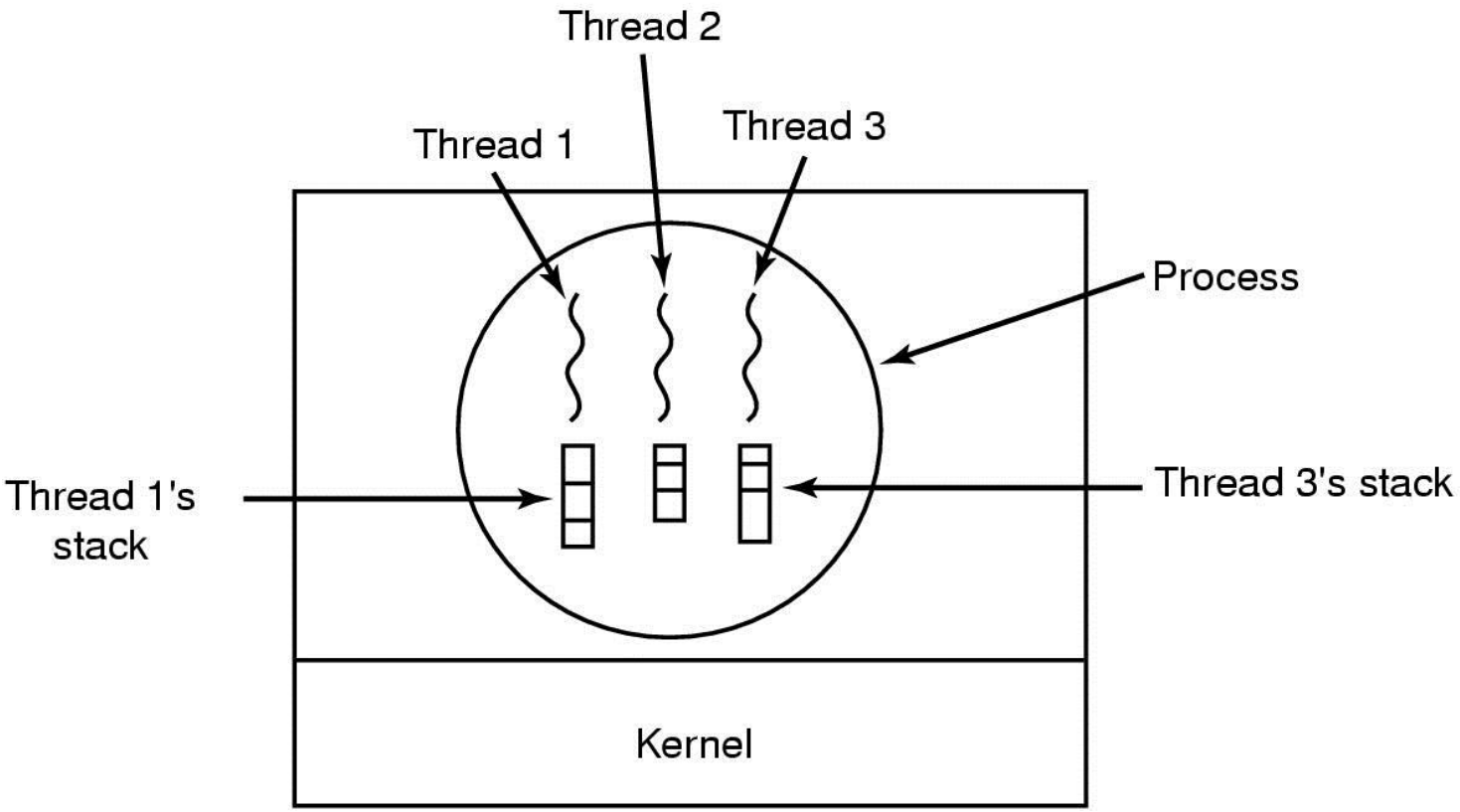


Figure 2-13. Each thread has its own stack.

与线程有关的库函数

thread_create：新线程标识符

thread_exit：线程结束

thread_wait：等待exit的同步

thread_yield：自动放弃CPU（时钟中断只对进程有效，对于线程，需要“自觉”放弃CPU，以便分时和并行，其他线程才会有机会运行）

线程与fork，见书中例子。

线程的使用

线程：在应用中同时发生多种活动，其中某些活动随着时间的进展会被阻塞，将其分解成可以准并行的多个顺序线程。

共享同一地址空间和所有可用数据的能力，正是多进程（具有不同的地址空间）不能胜任此类工作的原因。

线程附带资源不多，比进程更容易创建和撤销。（100倍）

CPU密集型 v.s. I/O密集型

降低阻塞

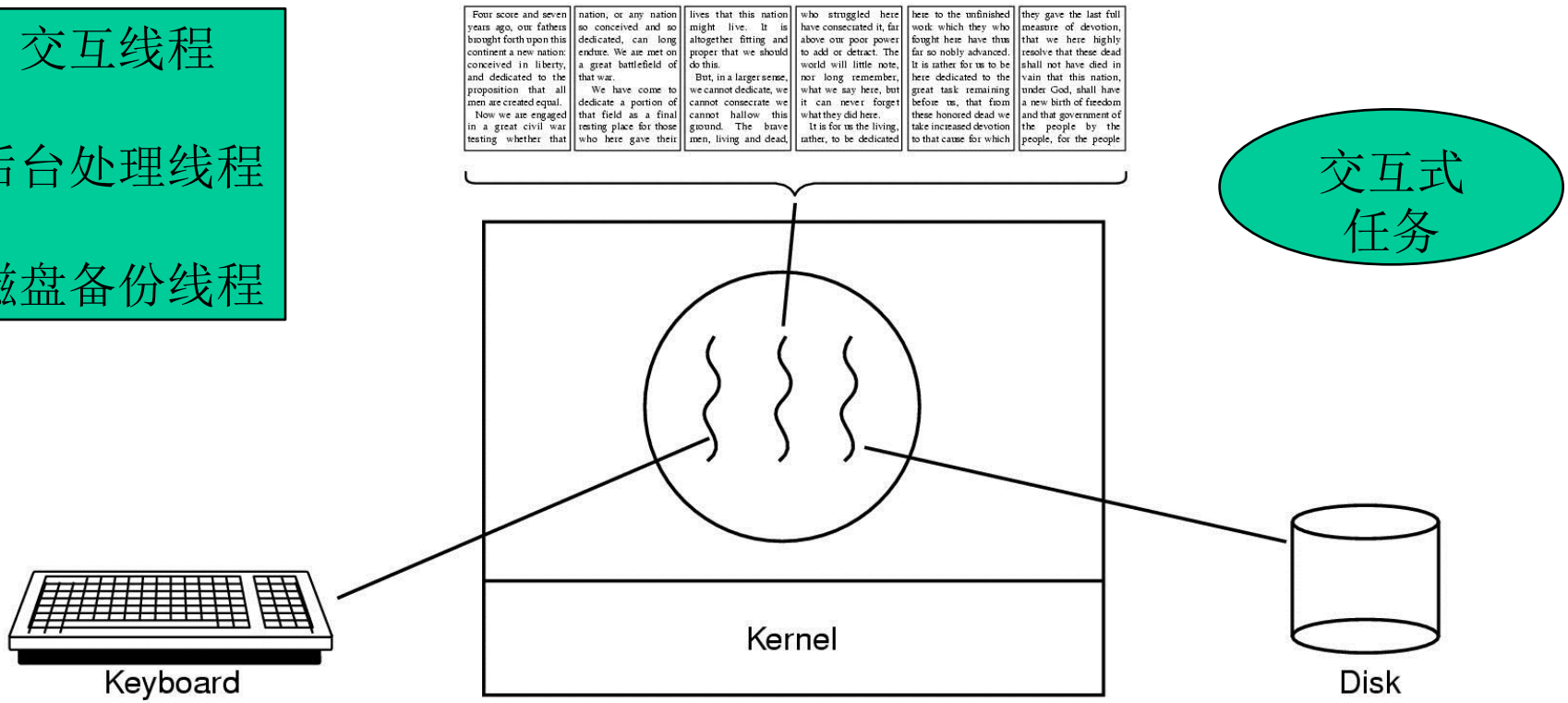
```
while(1)
{
    if(flag==1)
        break;
    sleep(1);  //浪费CPU?
}
```

Thread Usage (1)

交互线程

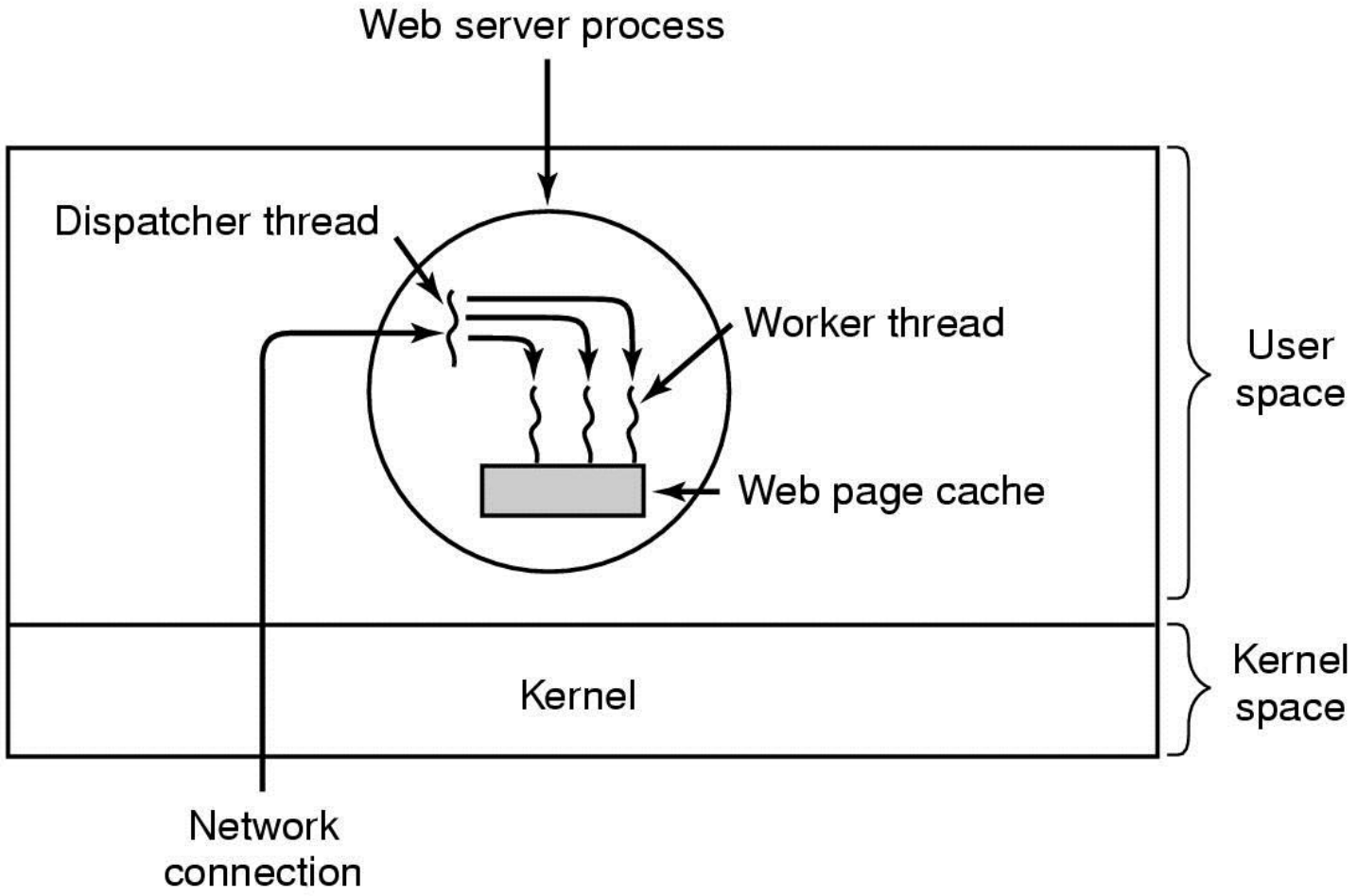
后台处理线程

磁盘备份线程



A word processor with three threads.

Thread Usage (2)



A multithreaded Web server.

Thread Usage (3)

```
while (TRUE) {
    get_next_request(&buf);
    handoff_work(&buf);
}
```

(a)

```
while (TRUE) {
    wait_for_work(&buf)
    look_for_page_in_cache(&buf, &page);
    if (page_not_in_cache(&page))
        read_page_from_disk(&buf, &page);
    return_page(&page);
}
```

(b)

A rough outline of the code.
(a) Dispatcher thread. (b) Worker thread.

Thread Usage (4)

Model	Characteristics
Threads	Parallelism, blocking system calls
Single-threaded process	No parallelism, blocking system calls
Finite-state machine	Parallelism, nonblocking system calls, interrupts

Three ways to construct a server.

POSIX Threads (1)

Thread call	Description
Pthread_create	Create a new thread
Pthread_exit	Terminate the calling thread
Pthread_join	Wait for a specific thread to exit
Pthread_yield	Release the CPU to let another thread run
Pthread_attr_init	Create and initialize a thread's attribute structure
Pthread_attr_destroy	Remove a thread's attribute structure

Some of the Pthreads function calls.

POSIX Threads (2)

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#define NUMBER_OF_THREADS     10

void *print_hello_world(void *tid)
{
        /* This function prints the thread's identifier and then exits. */
        printf("Hello World. Greetings from thread %d0, tid);
        pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
        /* The main program creates 10 threads and then exits. */
        pthread_t threads[NUMBER_OF_THREADS];
        int status, i;

        for(i=0; i < NUMBER_OF_THREADS; i++) {
                printf("Main here. Creating thread %d0, i);
                status = pthread_create(&threads[i], NULL, print_hello_world, (void *)i);

                if (status != 0) {
                        printf("Oops. pthread_create returned error code %d0, status);
                        exit(-1);
                }
        }
        exit(NULL);
}
```

An example program using threads.