

Course Number: EECIE18-S3303

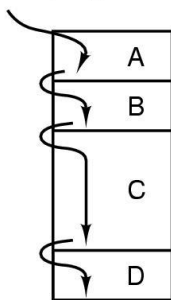
Course Name: Operating System – Processes and Threads

Lecturer: Dr. Sahar Abdul Aziz AL-talib

---

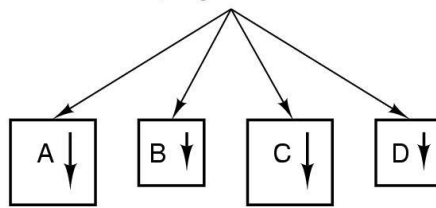
## The Process Model:

One program counter

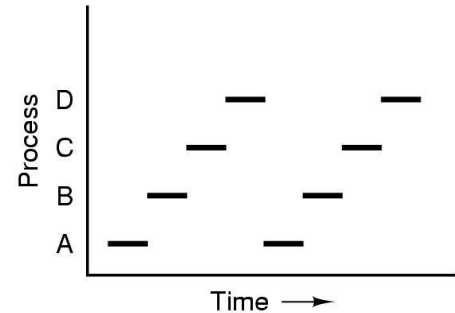


(a)

Four program counters



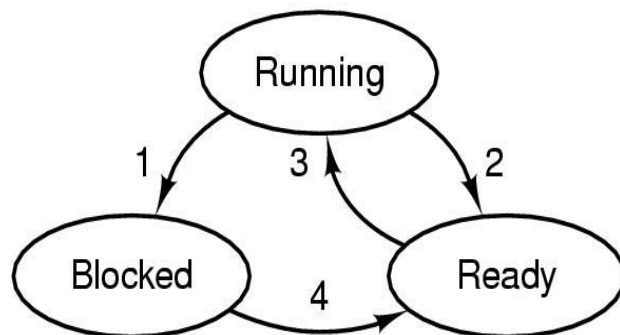
(b)



(c)

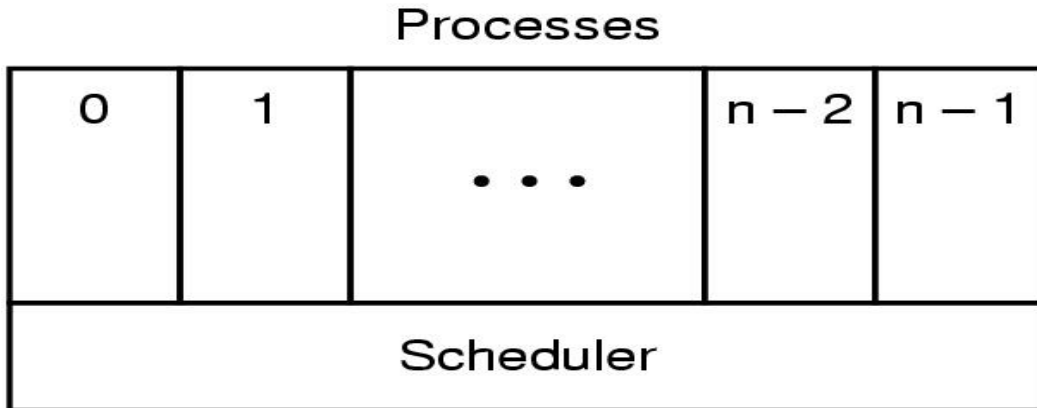
- Multiprogramming of four programs
- Conceptual model of 4 independent, sequential processes
- Only one program active at any instant

Process States:



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

- Possible process states
  - running
  - blocked
  - ready



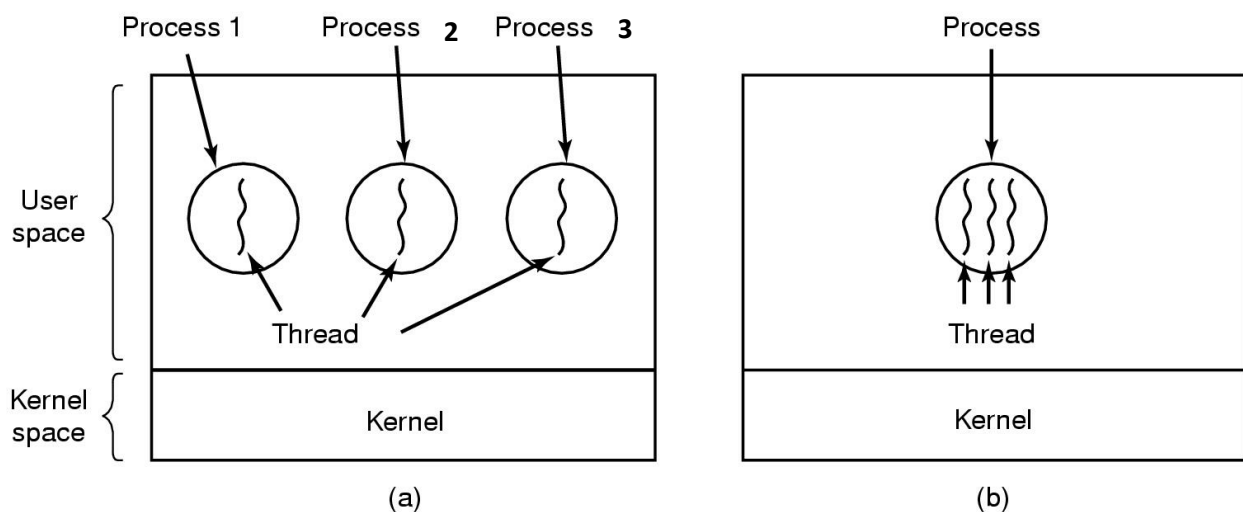
- **Lowest layer of process-structured OS**
  - handles interrupts, scheduling
- **Above that layer are sequential processes**

### Threads:

A thread is a basic unit of CPU utilization; it comprises a thread ID, program counter, a register set and a stack. It shares with other threads belonging to the same process its code section, data section, and other operating system resources, such as open files and signals.

- Process with single thread of control is called heavy weight process.
- Process with multiple threads of control can perform more than one task at a time.

### The thread Model:



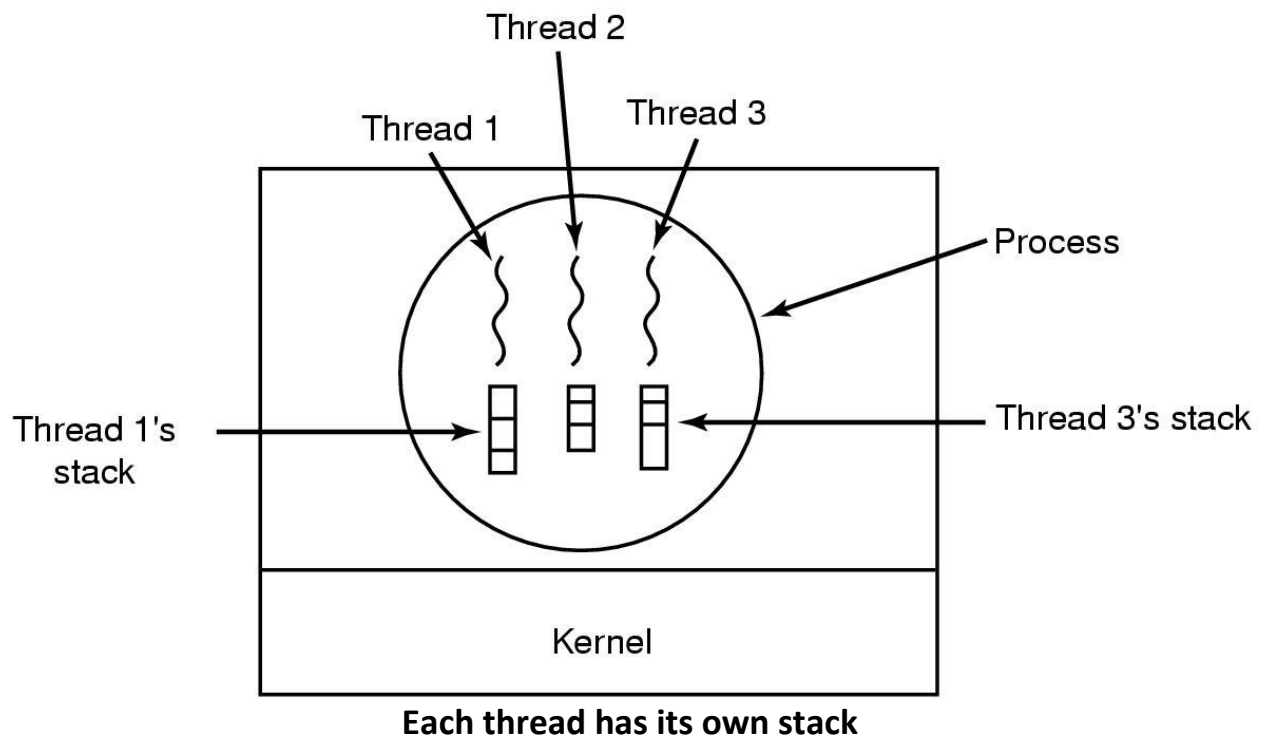
**(a) Three processes each with one thread**

**(b) One process with three threads**

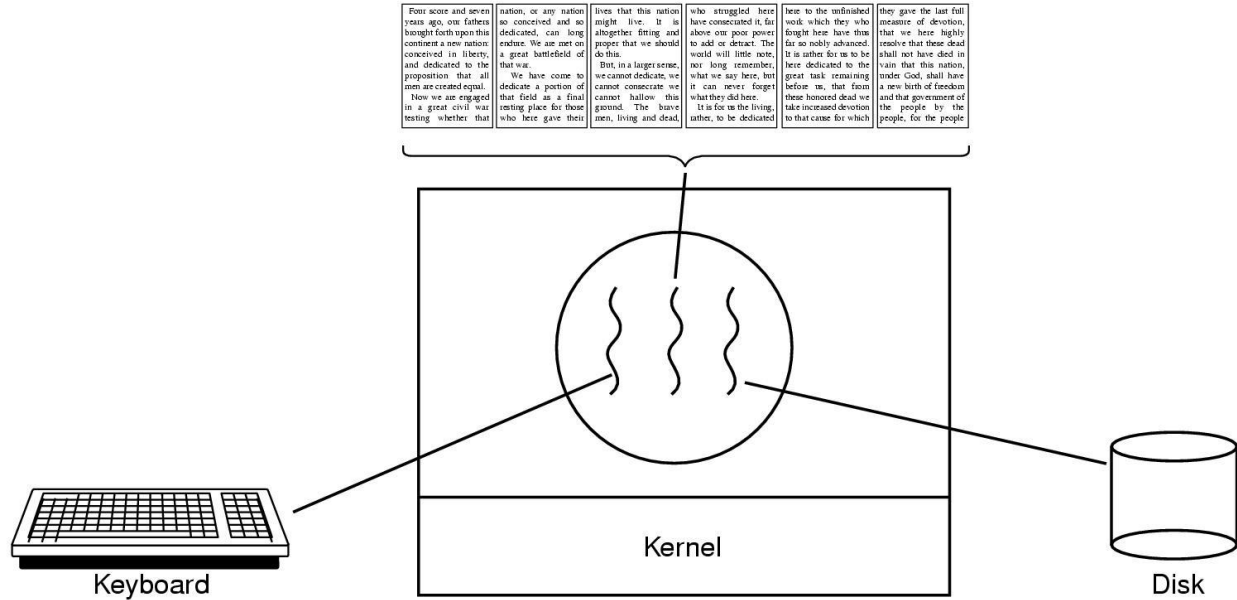
<b>Per process items</b>	<b>Per thread items</b>
Address space	Program counter
Global variables	Registers
Open files	Stack
Child processes	State
Pending alarms	
Signals and signal handlers	
Accounting information	

Items shared by all threads in a process

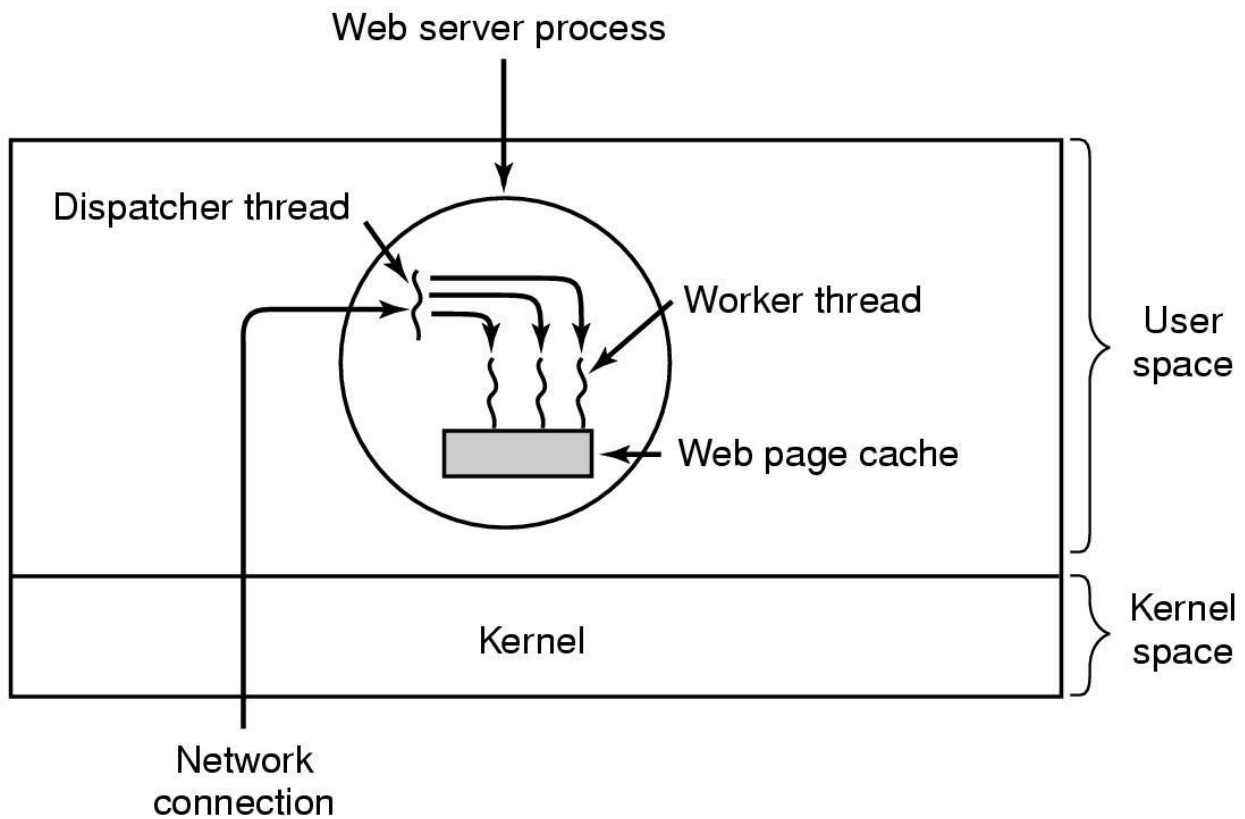
Items private to each thread



## Thread Usage Examples:



**Figure : A word processor with three threads**



**Figure: A multithreaded Web server**

```

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

```

(a)

**(a) Dispatcher thread**

```

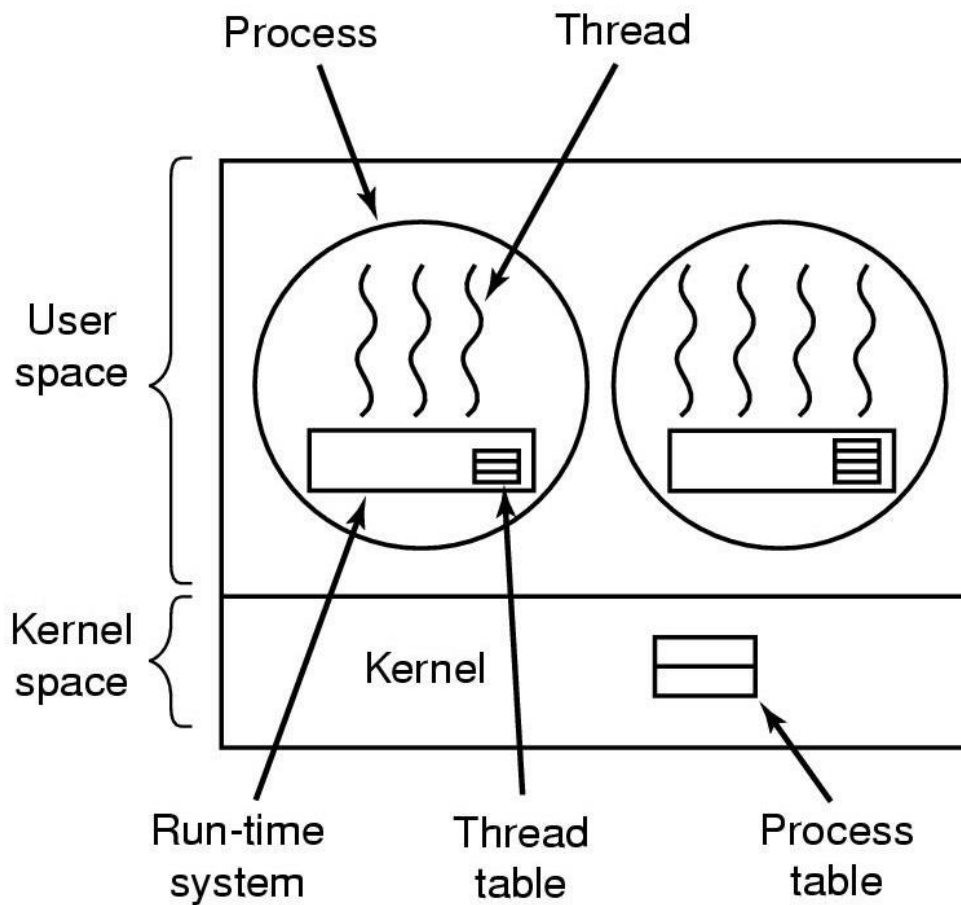
while (TRUE) {
  wait_for_work(&buf)
  look_for_page_in_cache(&buf, &page);
  if (page_not_in_cache(&page)
    read_page_from_disk(&buf, &page);
  return_page(&page);
}

```

(b)

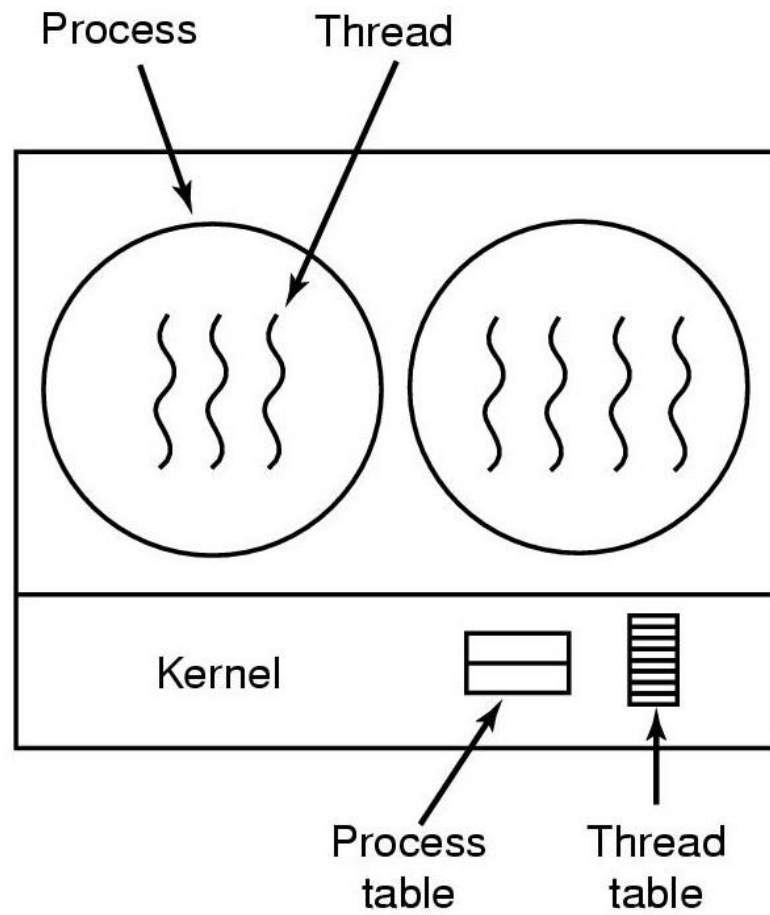
**(b) Worker thread**

**Implementing Threads in User Space:**



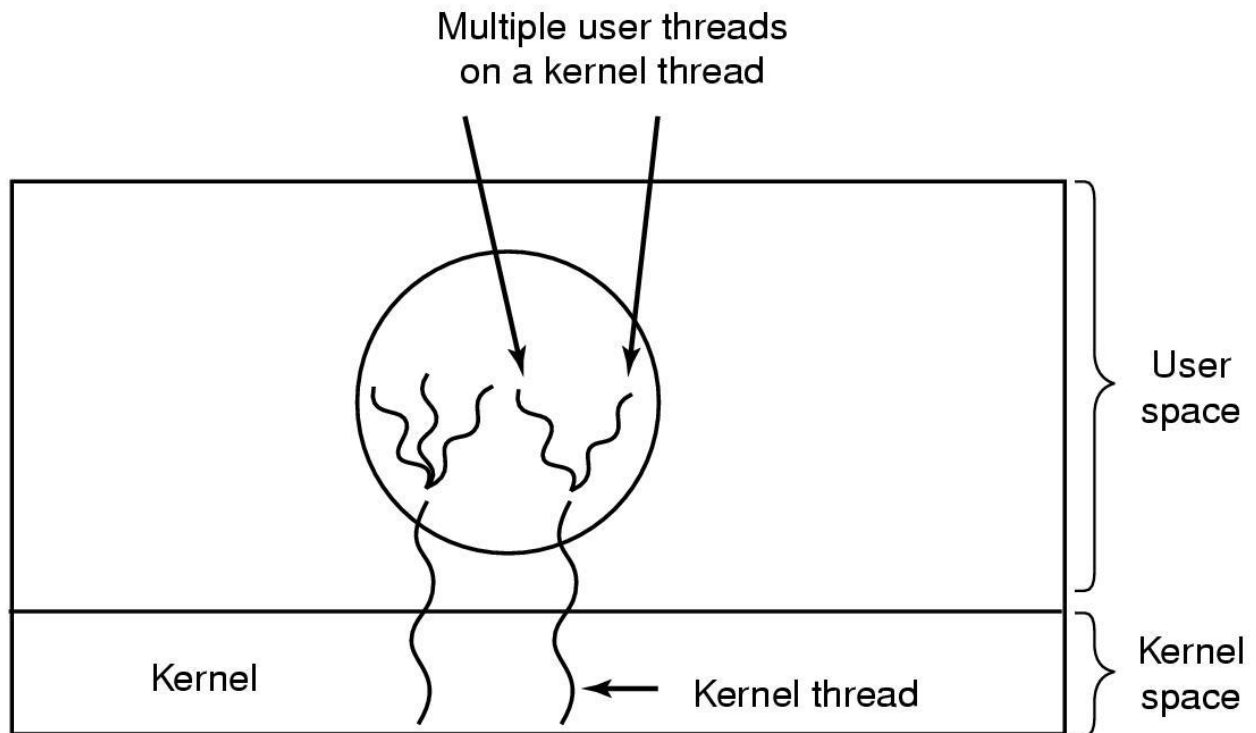
**Figure: A user-level threads package**

### Implementing Threads in the Kernel:



**Figure: A threads package managed by the kernel**

## Hybrid Implementations:

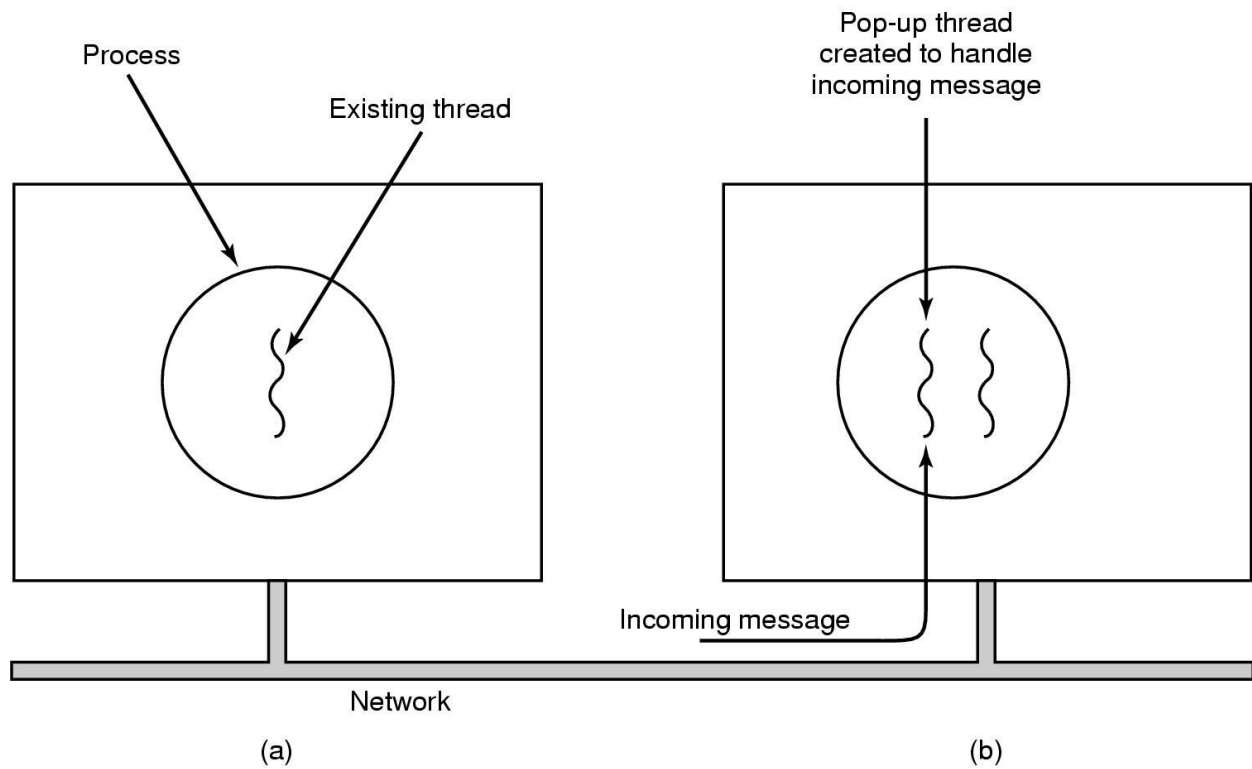


**Figure: Multiplexing user-level threads onto kernel- level threads**

## Scheduler Activations:

- Goal – mimic functionality of kernel threads
  - gain performance of user space threads
- Avoids unnecessary user/kernel transitions
- Kernel assigns virtual processors to each process
  - lets runtime system allocate threads to processors
- Problem:
  - Fundamental reliance on kernel (lower layer)
  - calling procedures in user space (higher layer)

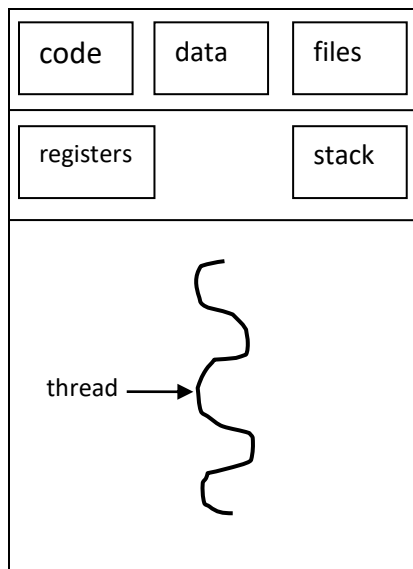
## Pop-Up Threads:



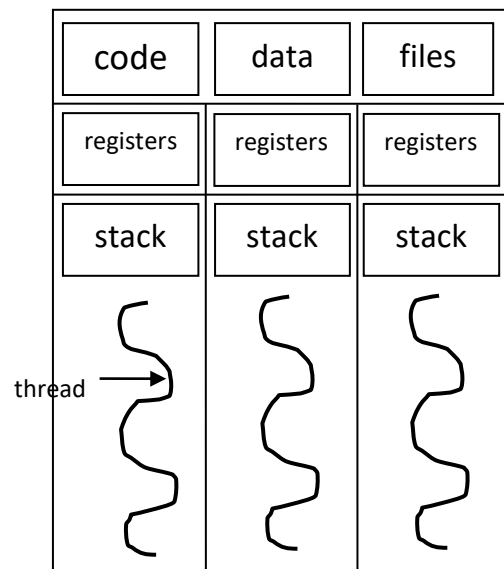
**Figure: Creation of a new thread when message arrives**

**(a) before message arrives**

**(b) after message arrives**



Single-threaded process



Multi-threaded process

A single application may be required to perform several similar tasks. Since process creation is time consuming and resource intensive, so it is more efficient to use one process that contains multiple threads instead of using multiple processes.

Many operating system kernels are now multithreaded; several threads operate in the kernel and each thread performs a specific task, such as managing devices or interrupt handling.

#### Benefits of Multithreading:

1. Interactive application may allow a program to continue running even if part of it is blocked or is performing a lengthy operation.
2. Resource sharing: by default, threads share the memory and the resources of the process to which they belong.
3. Economy: because threads share resources of the process to which they belong, it is more economical to create and context-threads. In general, It is much more time consuming to create and manage processes than threads.
4. Utilization of multiprocessor architectures: the benefits of multithreading can be greatly increased in a multiprocessor architecture, where threads may be running in parallel on different processors.