Threads

Giorgio Richelli
giorgio_richelli@it.ibm.com

Contents

- Introduction
- Fundamental Abstractions
- Lightweight Process Design
- User-Level Threads Libraries

Giorgio Richelli
giorgio_richelli@it.ibm.com

Introduction

- Motivation
  - Sharing common resources
    - Possible with processes, but with overheads
    - *fork()* is expensive
  - Multiprocessor architecture
    - The *process* is fundamentally a “single CPU” abstraction

Giorgio Richelli
giorgio_richelli@it.ibm.com

Multiple Threads and Processors

- True parallelism for multiprocessor architectures
- Multiplex if #T > #P
- Ideally:
  - if an application need 1 unit time with one thread version, it will only need 1/n unit time with a multithread version on a computer with n processors.

Giorgio Richelli
giorgio_richelli@it.ibm.com

Traditional UNIX system

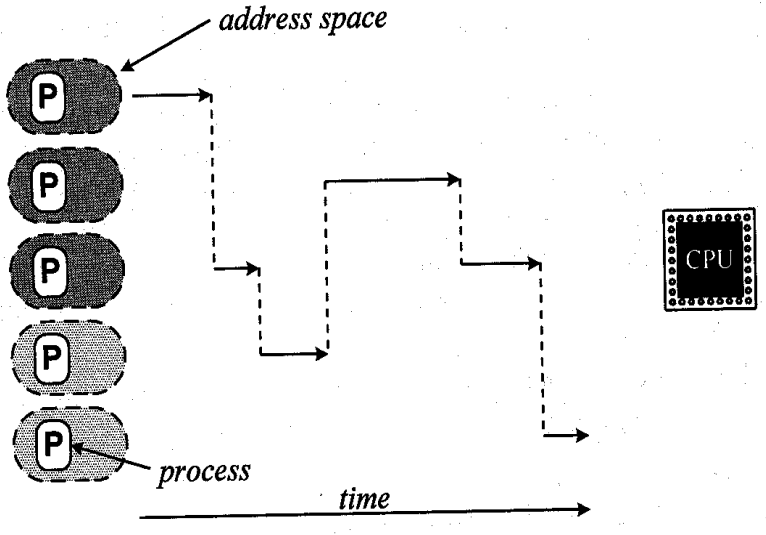


Figure 3-1. Traditional UNIX system—uniprocessor with single-threaded processes.

Giorgio Richelli
giorgio_richelli@it.ibm.com

Multithreaded Processes

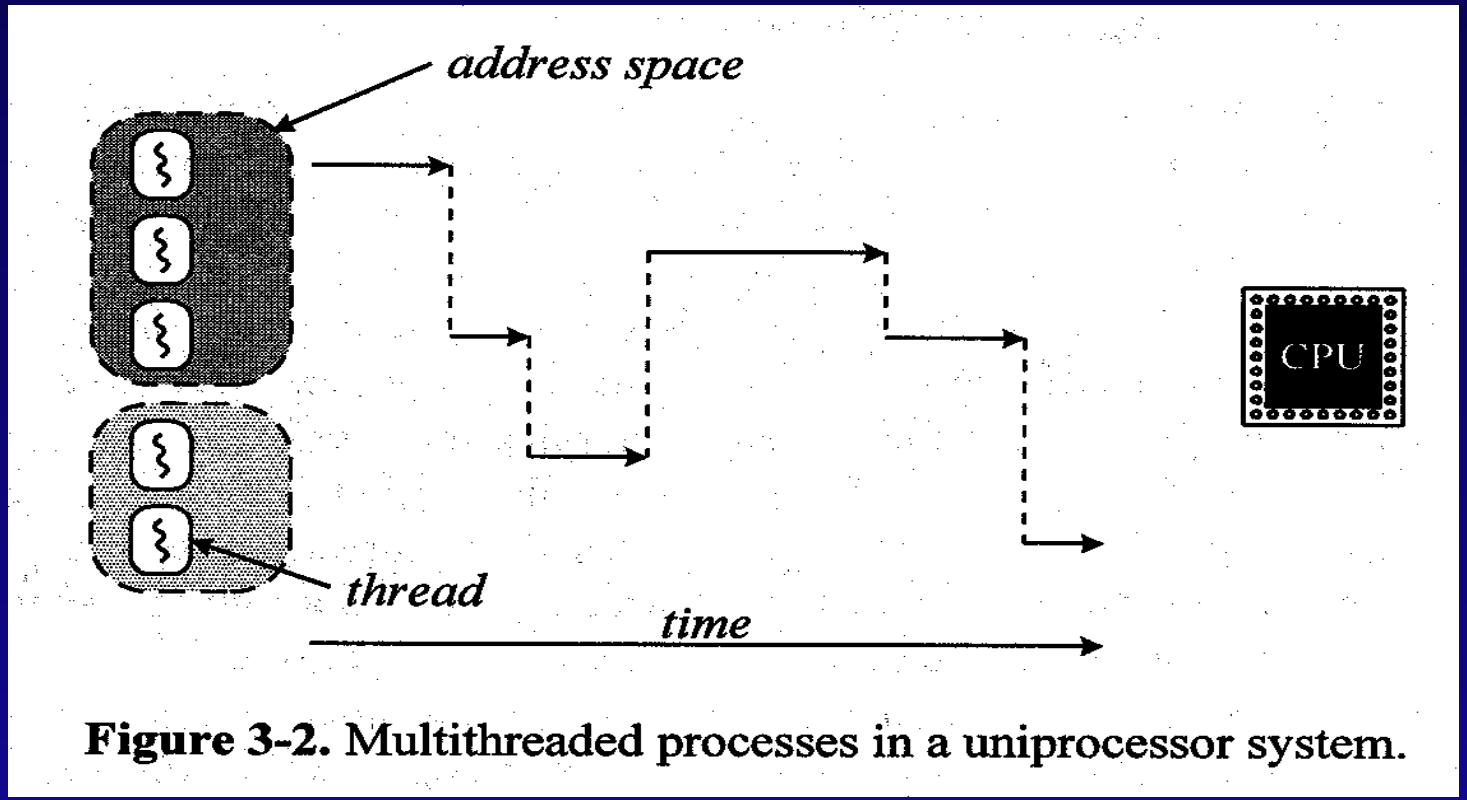


Figure 3-2. Multithreaded processes in a uniprocessor system.

Giorgio Richelli
giorgio_richelli@it.ibm.com

Multiprocessor

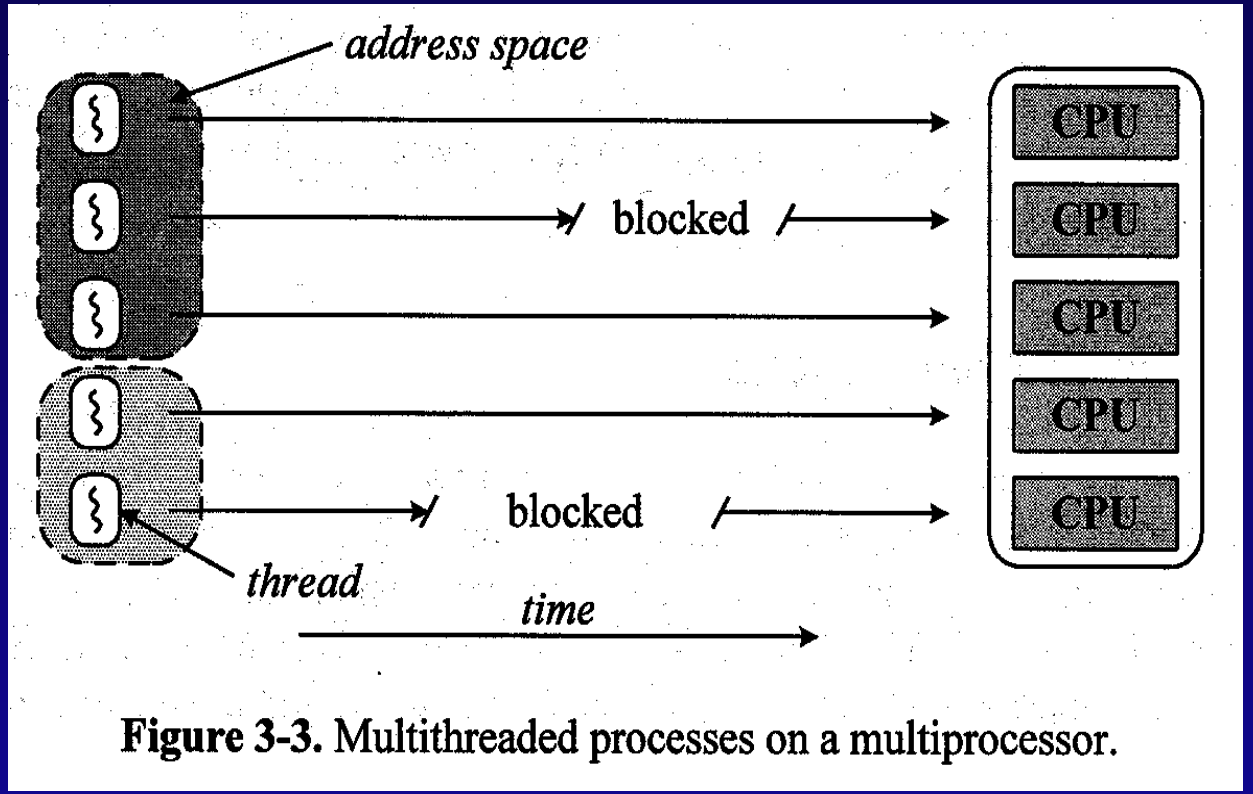


Figure 3-3. Multithreaded processes on a multiprocessor.

Giorgio Richelli
giorgio_richelli@it.ibm.com

Concurrency & Parallelism

- Concurrency:
  - The *maximum parallelism* it can achieve with an *unlimited number* of processors.
- Parallelism:
  - The actual degree of parallel execution achieved, limited by the number of physical processors.
- User/System concurrency

Giorgio Richelli
giorgio_richelli@it.ibm.com

Fundamental Abstraction

- A *process* :
  - is a compound entity that can be divided into two components:
    - a set of threads
    - a collection of resources.

Giorgio Richelli
giorgio_richelli@it.ibm.com

Fundamental Abstraction

- A *thread* :
  - a dynamic object that represents a control point in the process and that executes a sequence of instructions.
  - Shared resources
  - Private objects:
    - pc, stack, register context

Giorgio Richelli
giorgio_richelli@it.ibm.com

Processes

- Have a virtual address space which holds the process image
- Protected access to processors, other processes, files, and I/O resources

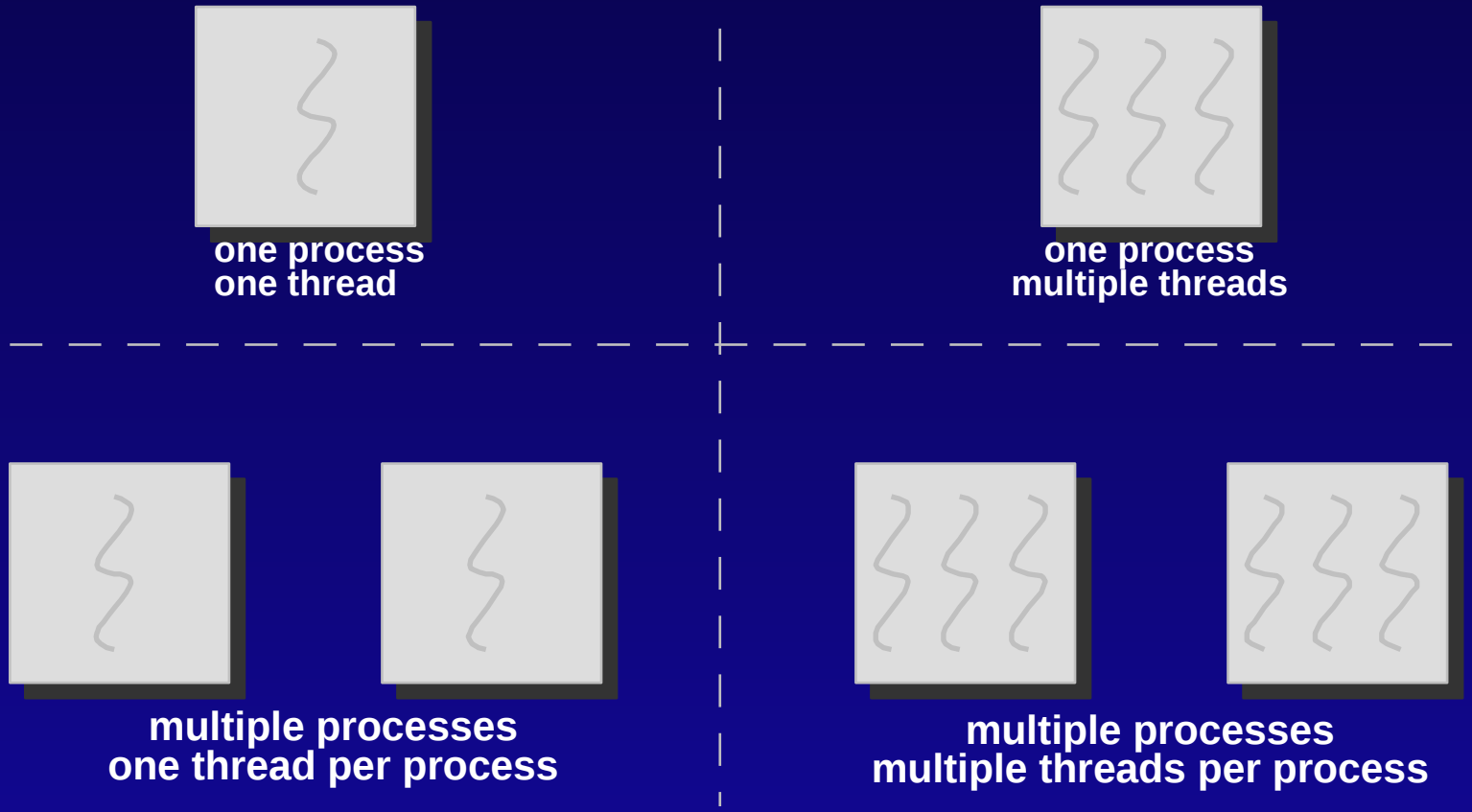
Giorgio Richelli
giorgio_richelli@it.ibm.com

Threads

- Has an execution state (running, ready, etc.)
  - OpSys saves thread context when not running
- Has an execution stack
- Has some static storage for local variables
- Has access to the memory and resources of its process (all threads of a process share it)

Giorgio Richelli
giorgio_richelli@it.ibm.com

Threads and Processes



Giorgio Richelli
giorgio_richelli@it.ibm.com

Benefits of Threads

- It’s faster to:
  - create a new thread than a process
  - terminate a thread than a process
  - switch between two threads within the same process
- Since threads within the same process share memory and files, they can communicate with each other without invoking the kernel

Giorgio Richelli
giorgio_richelli@it.ibm.com

Threads

- Suspending a process involves suspending all threads of the process since all threads share the same address space
- Termination of a process, terminates all threads within the process

Giorgio Richelli
giorgio_richelli@it.ibm.com