Today’s class

- Threads, SMP, and Microkernels
- Principles of concurrency

UPPSALA UNIVERSITET

Informationsteknologi

Process

- Resource ownership - process includes a virtual address space to hold the process image
- Scheduling/execution- follows an execution path that may be interleaved with other processes
- These two characteristics are treated independently by the operating system

UPPSALA UNIVERSITET

Informationsteknologi

Process

- Dispatching is referred to as a thread or lightweight process
- Resource ownership is referred to as a process or task

UPPSALA UNIVERSITET

Informationsteknologi

Multithreading

- Operating system supports multiple threads of execution within a single process
- UNIX supports multiple user processes but only supports one thread per process
- Windows, Solaris, Linux, Mach, and OS/2 support multiple threads

UPPSALA UNIVERSITET

Informationsteknologi

Process

- In a multithreaded environment a process is defined as the unit of resource allocation and a unit of protection
- Have a virtual address space which holds the process image
- Protected access to processors, other processes, files, and I/O resources

UPPSALA UNIVERSITET

Informationsteknologi

Thread

- An execution state (running, ready, etc.)
- A saved thread context when not running
- An execution stack
- Some per-thread static storage for local variables
- Access to the memory and resources of its process

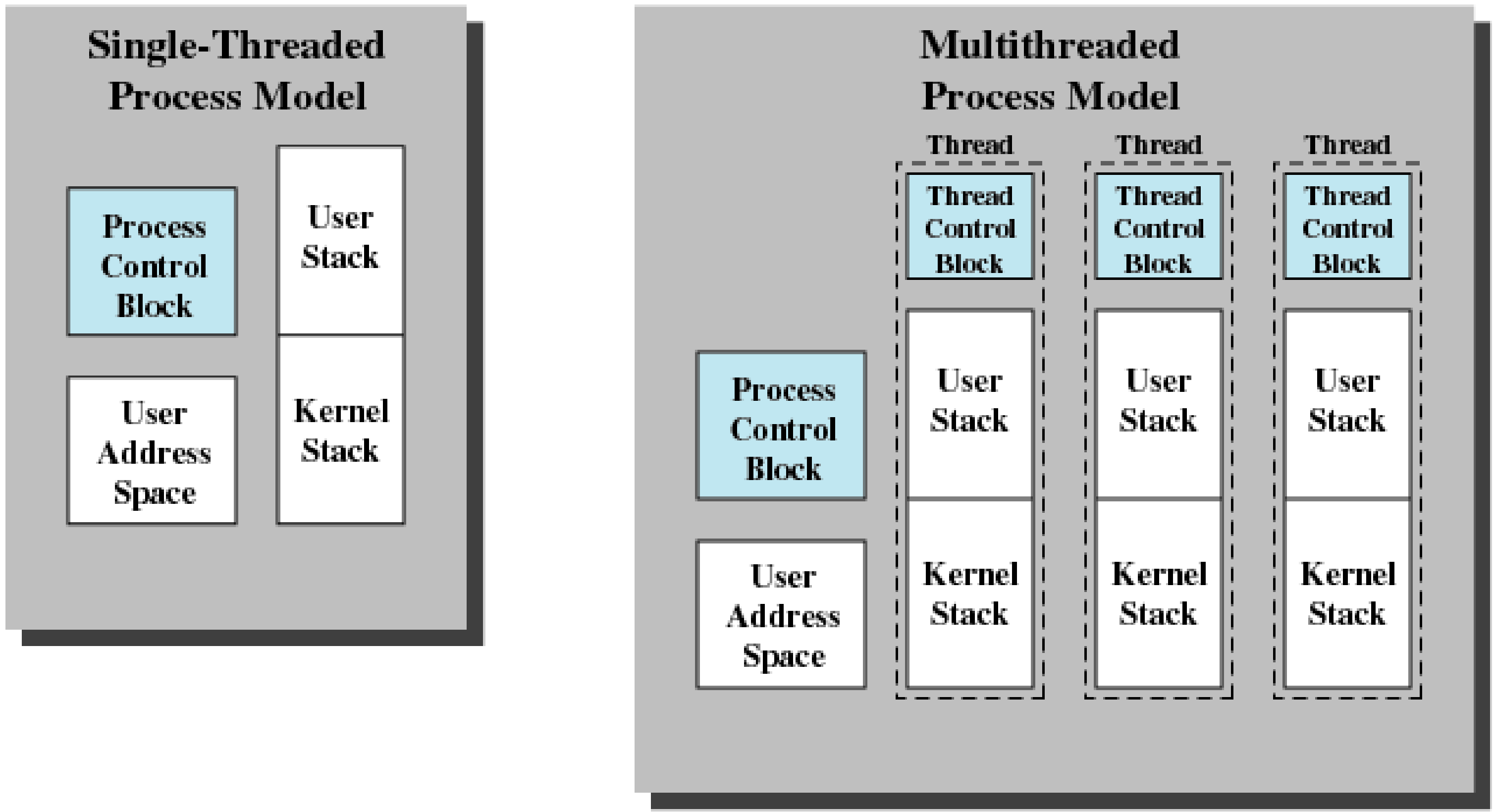


Figure 4.2 Single Threaded and Multithreaded Process Models

UPPSALA UNIVERSITET

Informationsteknologi

Benefits of Threads

- Takes less time to create a new thread than a process
- Less time to terminate a thread than a process
- Less time to 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

Uses of Threads in a Single-User Multiprocessing System

- Foreground to background work
- Asynchronous processing
- Speed of execution
- Modular program structure

UPPSALA UNIVERSITET

Informationsteknologi

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

UPPSALA UNIVERSITET

Informationsteknologi

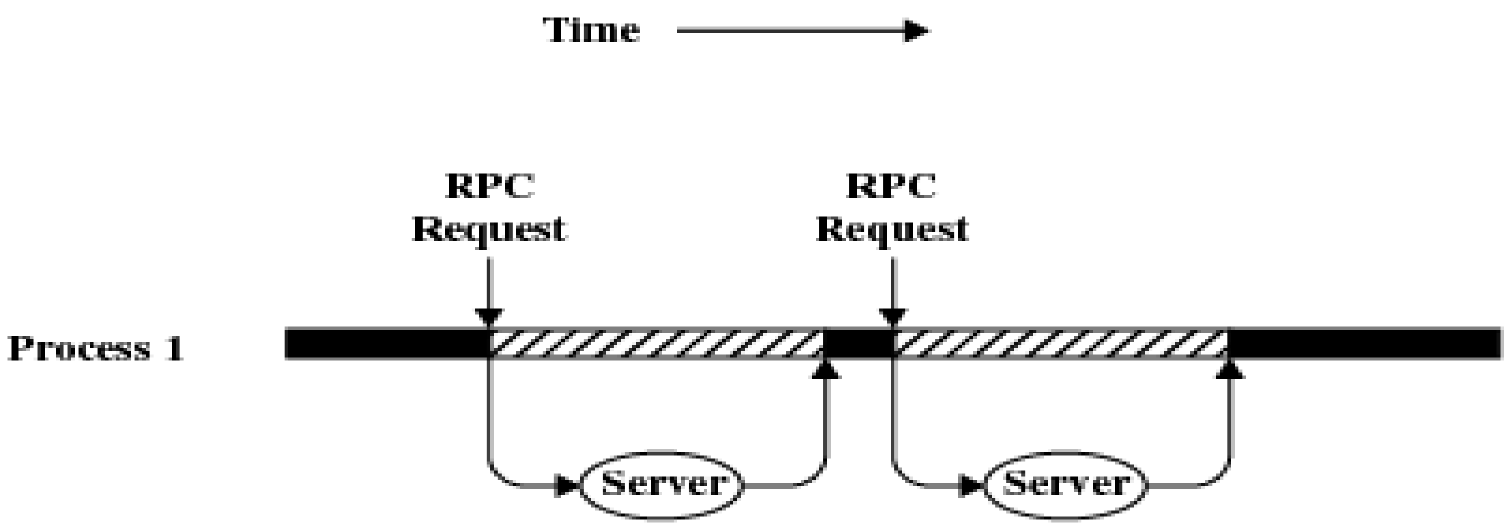
Thread States

- Key thread states are Running, Ready, and Blocked
- Operations associated with a change in thread state
  - Spawn
  - Block
  - Unblock
  - Finish
    - Deallocate register context and stacks

UPPSALA UNIVERSITET

Informationsteknologi

Remote Procedure Call Using Single Thread



(a) RPC Using Single Thread

Remote Procedure Call Using Threads

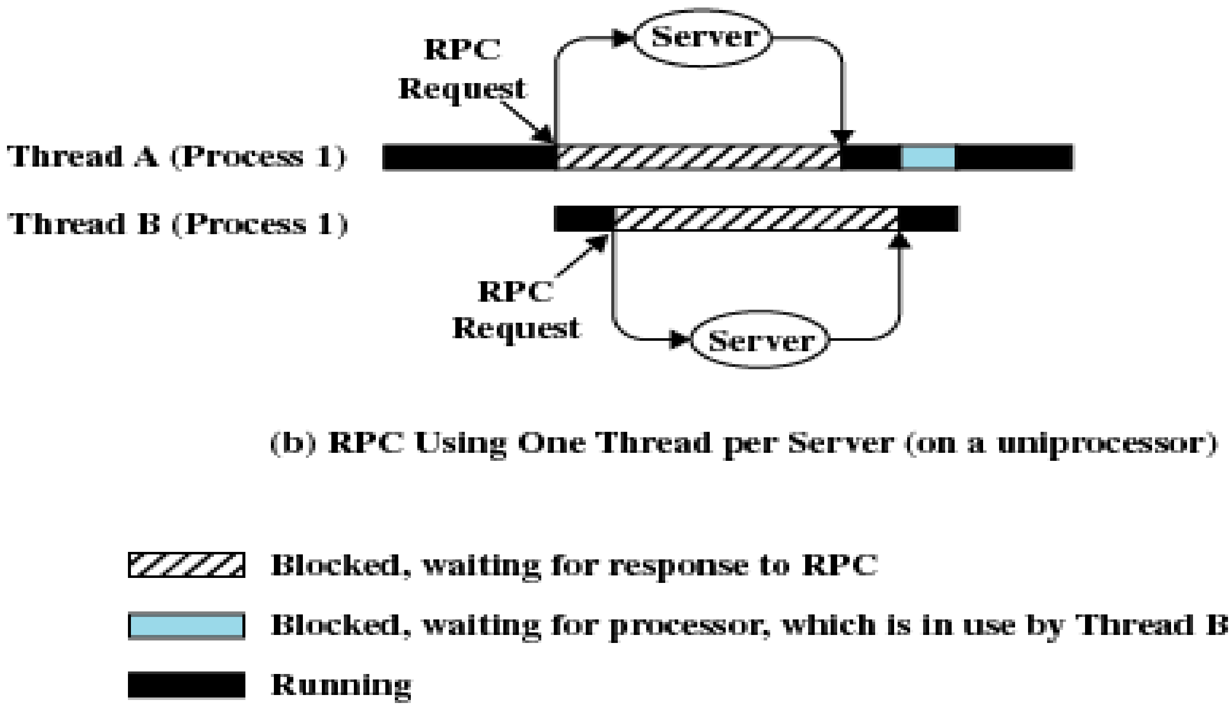


Figure 4.3 Remote Procedure Call (RPC) Using Threads

UPPSALA UNIVERSITET
Informationsteknologi

Multithreading

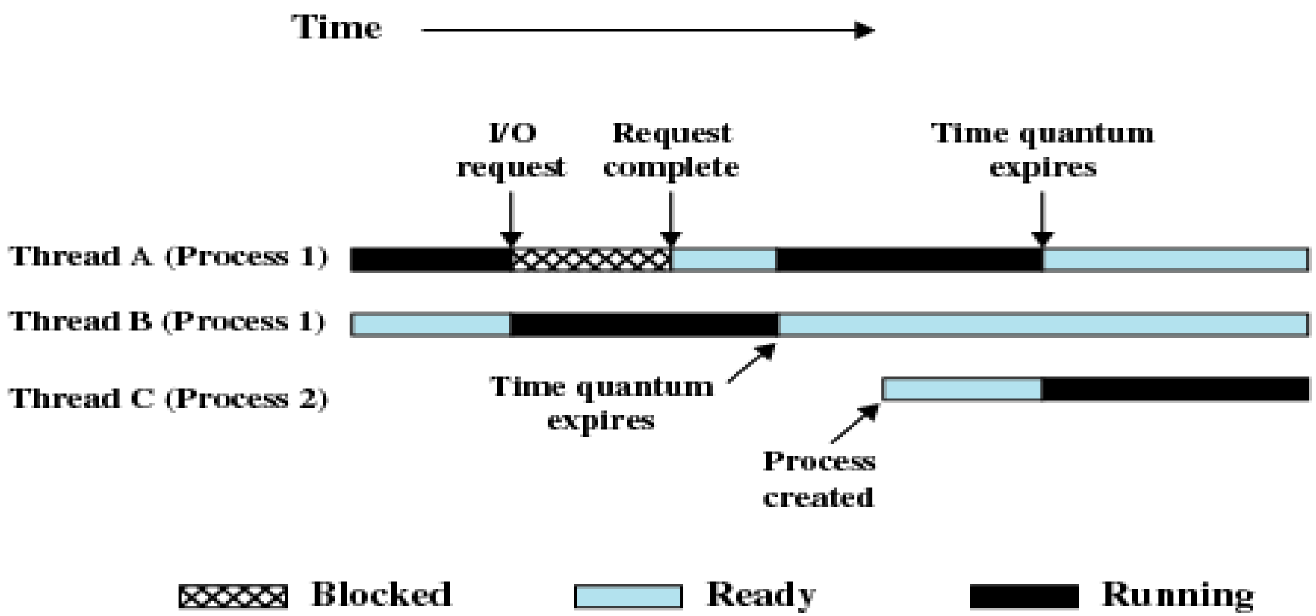


Figure 4.4 Multithreading Example on a Uniprocessor

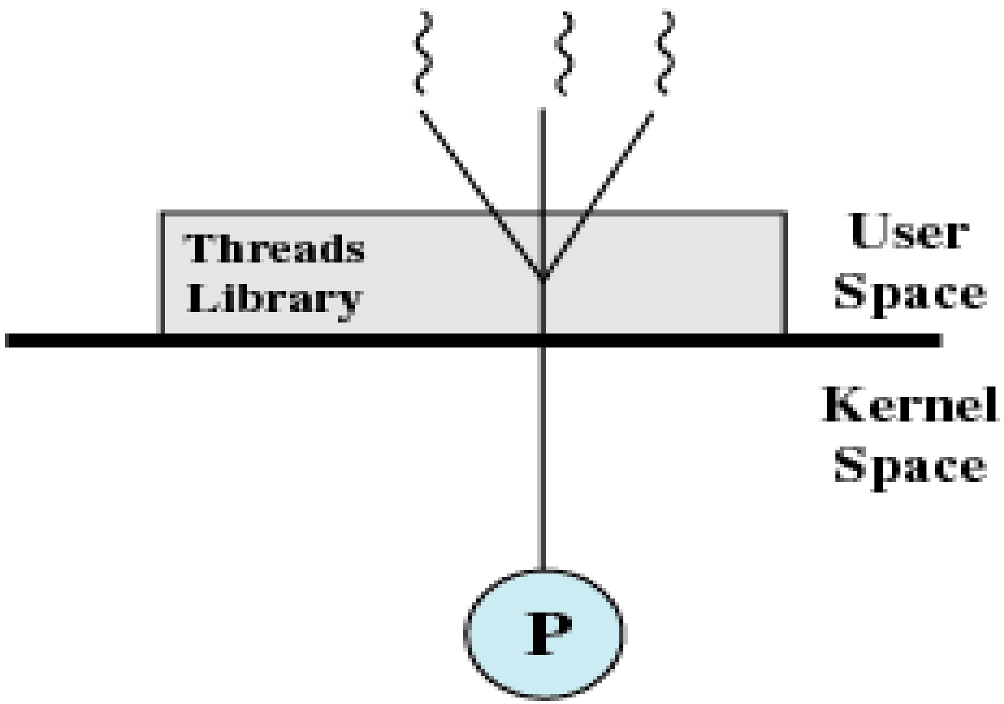
UPPSALA UNIVERSITET

Informationsteknologi

User-Level Threads

- All thread management is done by the application
- The kernel is not aware of the existence of threads

User-Level Threads



UPPSALA UNIVERSITET

Informationsteknologi

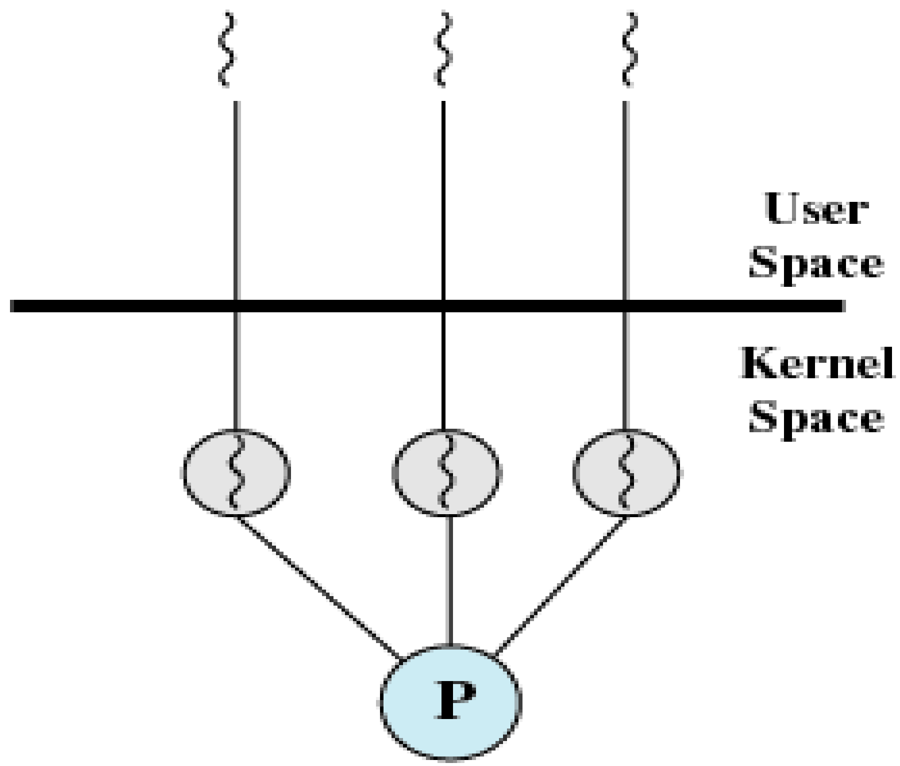
Kernel-Level Threads

- Windows is an example of this approach
- Kernel maintains context information for the process and the threads
- Scheduling is done on a thread basis

UPPSALA UNIVERSITET

Informationsteknologi

Kernel-Level Threads



(b) Pure kernel-level

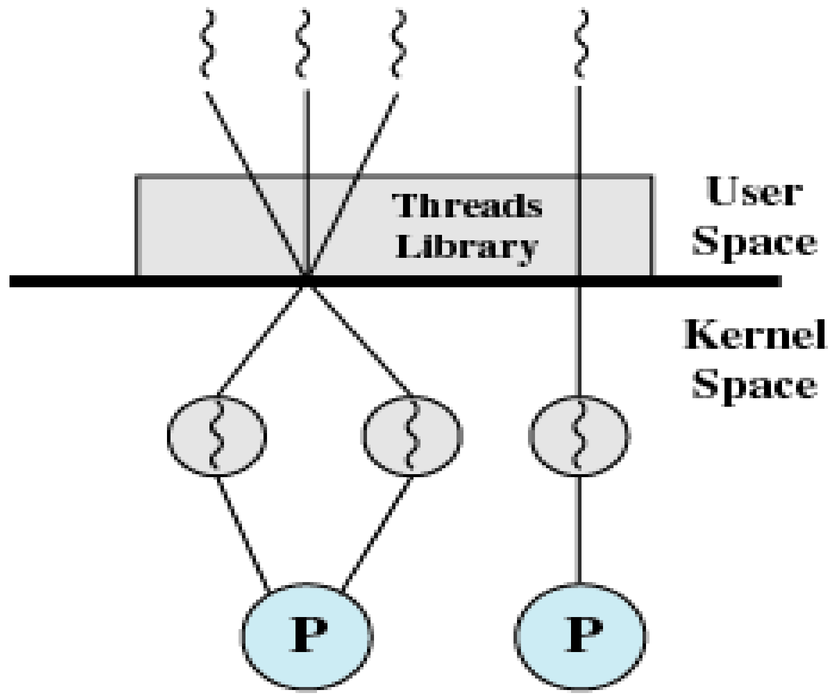
UPPSALA UNIVERSITET

Informationsteknologi

Combined Approaches

- Example is Solaris
- Thread creation done in the user space
- Bulk of scheduling and synchronization of threads within application

Combined Approaches



(c) Combined

UPPSALA UNIVERSITET

Informationsteknologi

Categories of Computer Systems

- Single Instruction Single Data (SISD) stream
  - Single processor executes a single instruction stream to operate on data stored in a single memory
- Single Instruction Multiple Data (SIMD) stream
  - Each instruction is executed on a different set of data by the different processors

UPPSALA UNIVERSITET

Informationsteknologi

Categories of Computer Systems

- Multiple Instruction Single Data (MISD) stream
  - A sequence of data is transmitted to a set of processors, each of which executes a different instruction sequence. Never implemented
- Multiple Instruction Multiple Data (MIMD)
  - A set of processors simultaneously execute different instruction sequences on different data sets

Informationsteknologi

UPPSALA UNIVERSITET

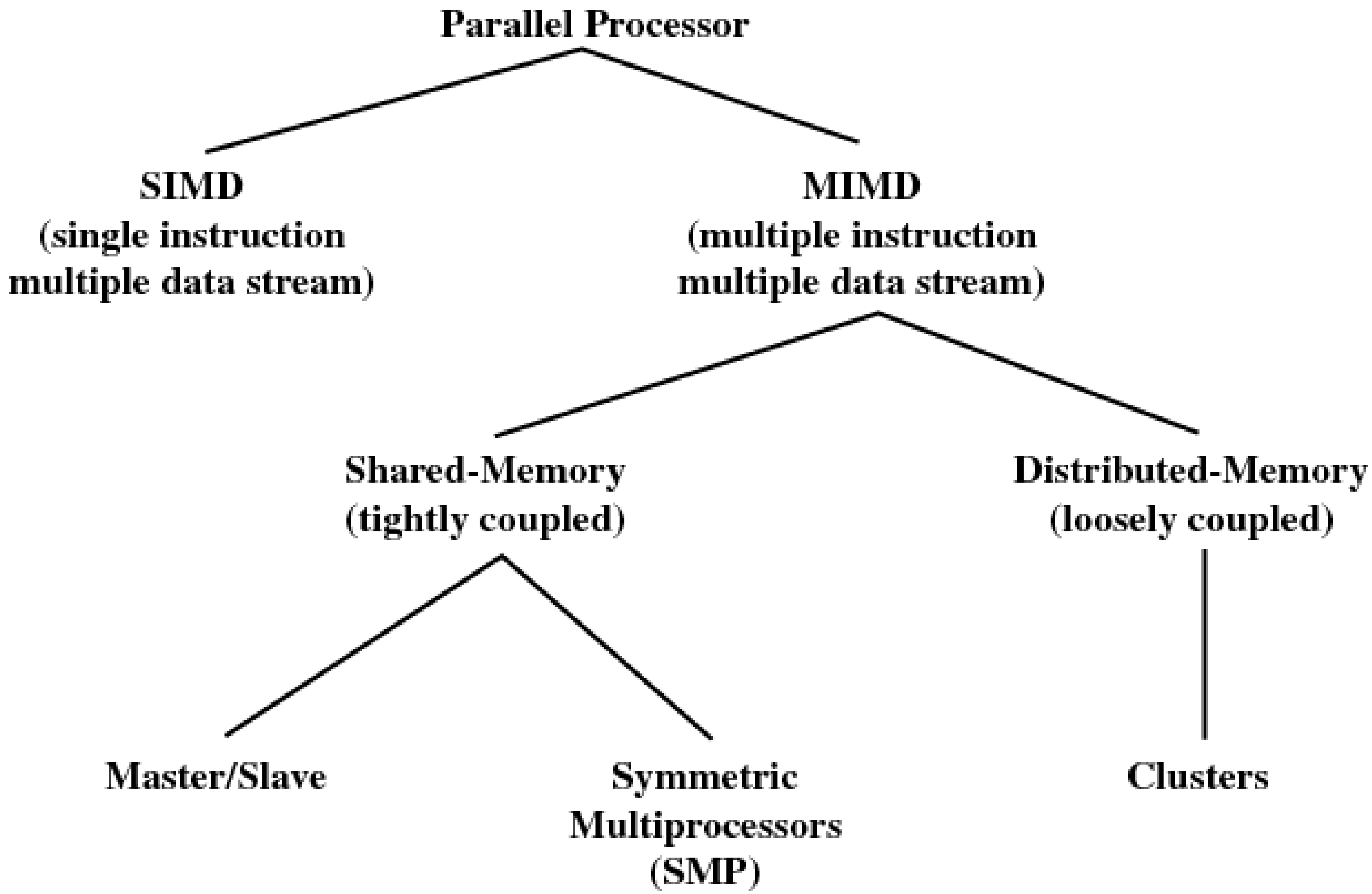


Figure 4.8 Parallel Processor Architectures

UPPSALA UNIVERSITET

Informationsteknologi

Symmetric Multiprocessing

- Kernel can execute on any processor
- Typically each processor does self-scheduling form the pool of available process or threads

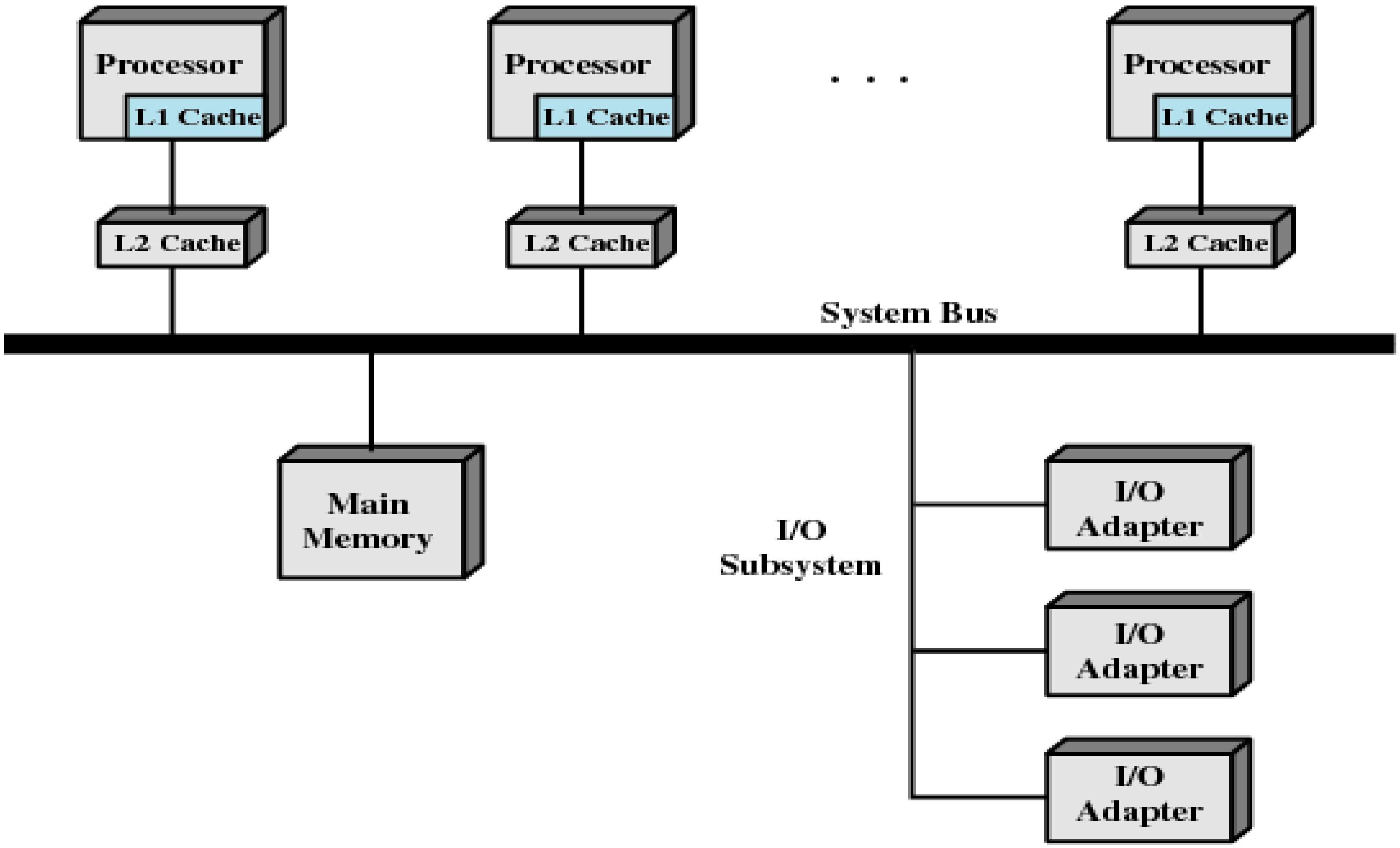


Figure 4.9 Symmetric Multiprocessor Organization

UPPSALA UNIVERSITET

Informationsteknologi

Multiprocessor Operating System Design Considerations

- Simultaneous concurrent processes or threads
  - Kernel routines need to be re-entrant, to allow several processors to execute the same kernel code simultaneously
- Scheduling
  - May be performed by any processor, so conflicts must be avoided
- Synchronization
- Memory management
- Reliability and fault tolerance

UPPSALA UNIVERSITET

Informationsteknologi

Microkernels

- Small operating system core
- Contains only essential core operating systems functions
- Many services traditionally included in the operating system are now external subsystems
  - Device drivers
  - File systems
  - Virtual memory manager
  - Windowing system
  - Security services

UPPSALA UNIVERSITET

Informationsteknologi

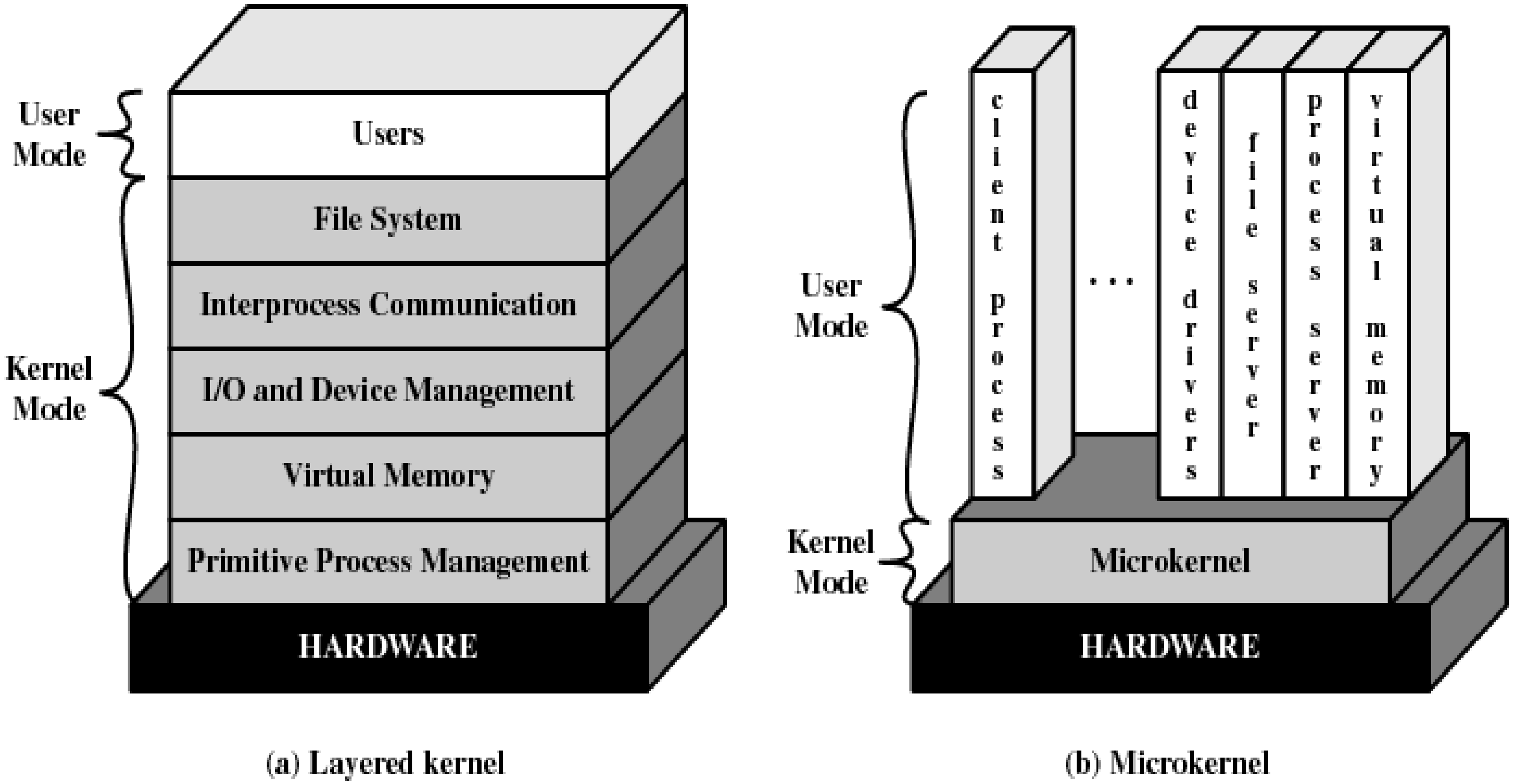


Figure 4.10 Kernel Architecture

UPPSALA UNIVERSITET

Informationsteknologi

Benefits of a Microkernel Organization

- Uniform interface on request made by a process
  - Don't distinguish between kernel-level and user-level services
  - All services are provided by means of message passing
- Extensibility
  - Allows the addition of new services
- Flexibility
  - New features added
  - Existing features can be subtracted

UPPSALA UNIVERSITET

Informationsteknologi

Benefits of a Microkernel Organization

- Portability
  - Changes needed to port the system to a new processor are changed in the microkernel - not in the other services
- Reliability
  - Modular design
  - Small microkernel can be rigorously tested

UPPSALA UNIVERSITET

Informationsteknologi

Benefits of Microkernel Organization

- Distributed system support
  - Message are sent without knowing what the target machine is
- Object-oriented operating system
  - Components are objects with clearly defined interfaces that can be interconnected to form software

UPPSALA UNIVERSITET

Informationsteknologi

Microkernel Design

- Low-level memory management
  - Mapping each virtual page to a physical page frame

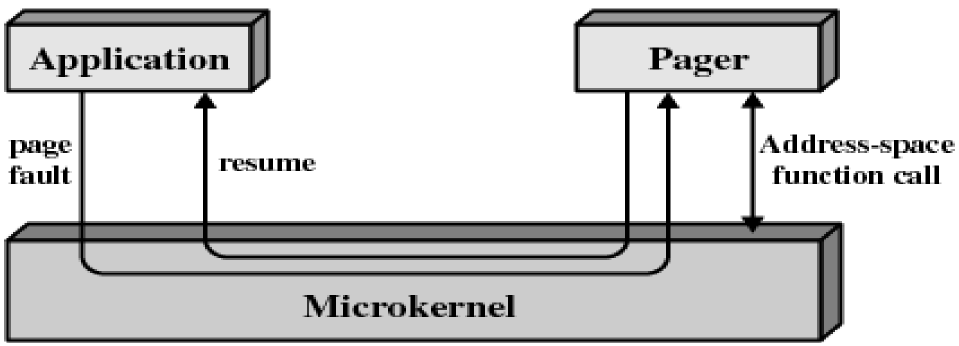


Figure 4.11 Page Fault Processing

UPPSALA UNIVERSITET

Informationsteknologi

Microkernel Design

- Interprocess communication
  - Basic mechanism is a *message*
  - A *port* is a queue of messages destined for a particular process
- I/O and interrupt management
  - Hardware interrupts handled as messages
  - I/O ports included in address space

UPPSALA UNIVERSITET

Informationsteknologi

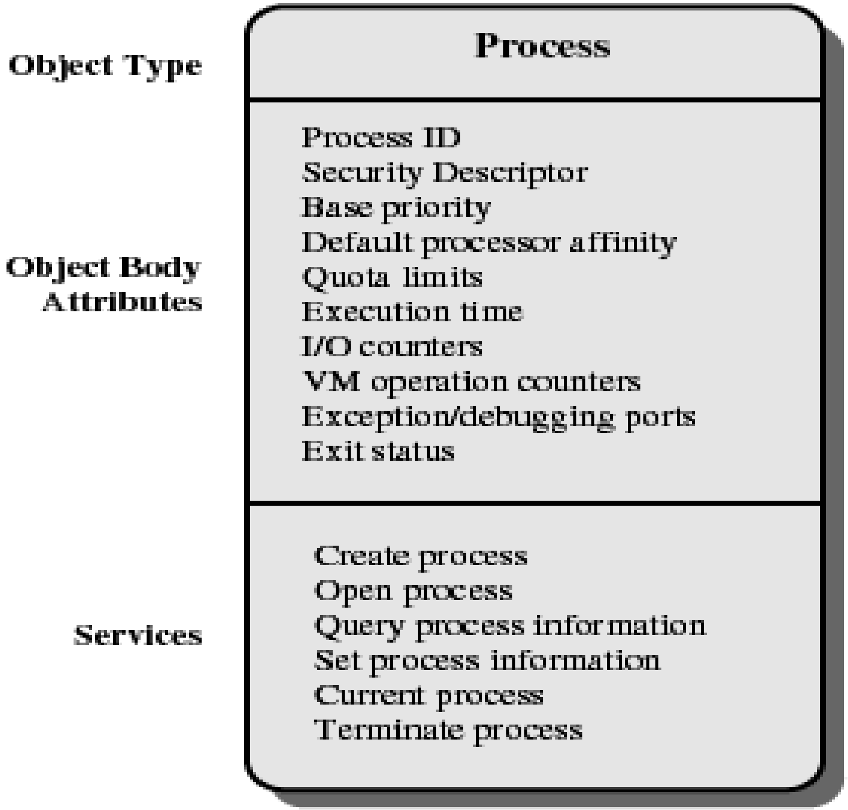
Windows Processes

- Implemented as objects
- An executable process may contain one or more threads
- Both processes and thread objects have built-in synchronization capabilities

UPPSALA UNIVERSITET

Informationsteknologi

Windows Process Object



(a) Process object

UPPSALA UNIVERSITET

Informationsteknologi

Windows Thread Object

Object Type: **Thread**

Object Body Attributes:
- Thread ID
- Thread context
- Dynamic priority
- Base priority
- Thread processor affinity
- Thread execution time
- Alert status
- Suspension count
- Impersonation token
- Termination port
- Thread exit status

Services:
- Create thread
- Open thread
- Query thread information
- Set thread information
- Current thread
- Terminate thread
- Get context
- Set context
- Suspend
- Resume
- Alert thread
- Test thread alert
- Register termination port

(b) Thread object

UPPSALA UNIVERSITET

Informationsteknologi

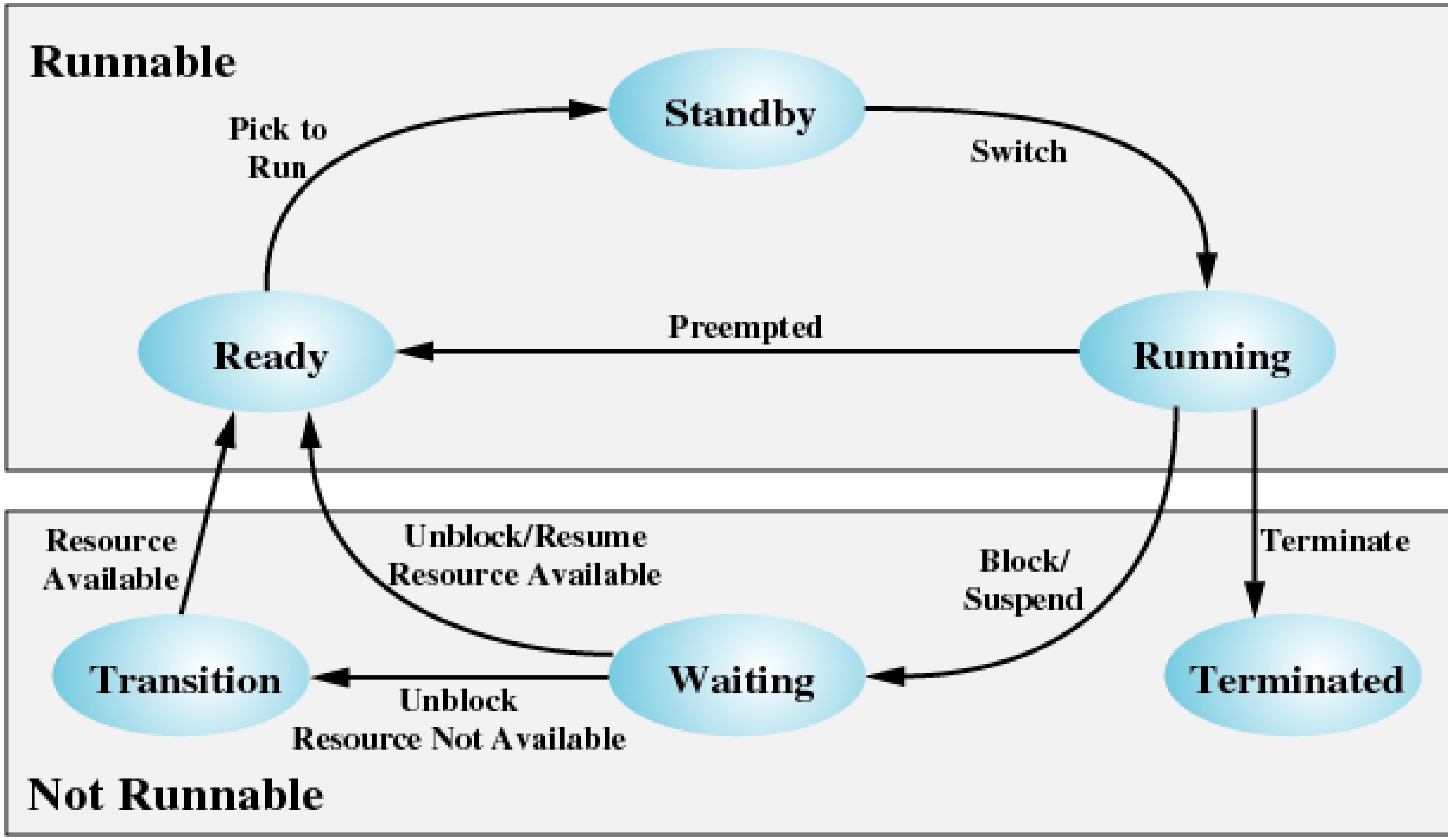


Figure 4.14 Windows Thread States

UPPSALA UNIVERSITET

Informationsteknologi

Linux Task Data Structure

- State
- Scheduling information
- Identifiers
- Interprocess communication
- Links
- Times and timers
- File system
- Address space
- Processor-specific context

UPPSALA UNIVERSITET

Informationsteknologi

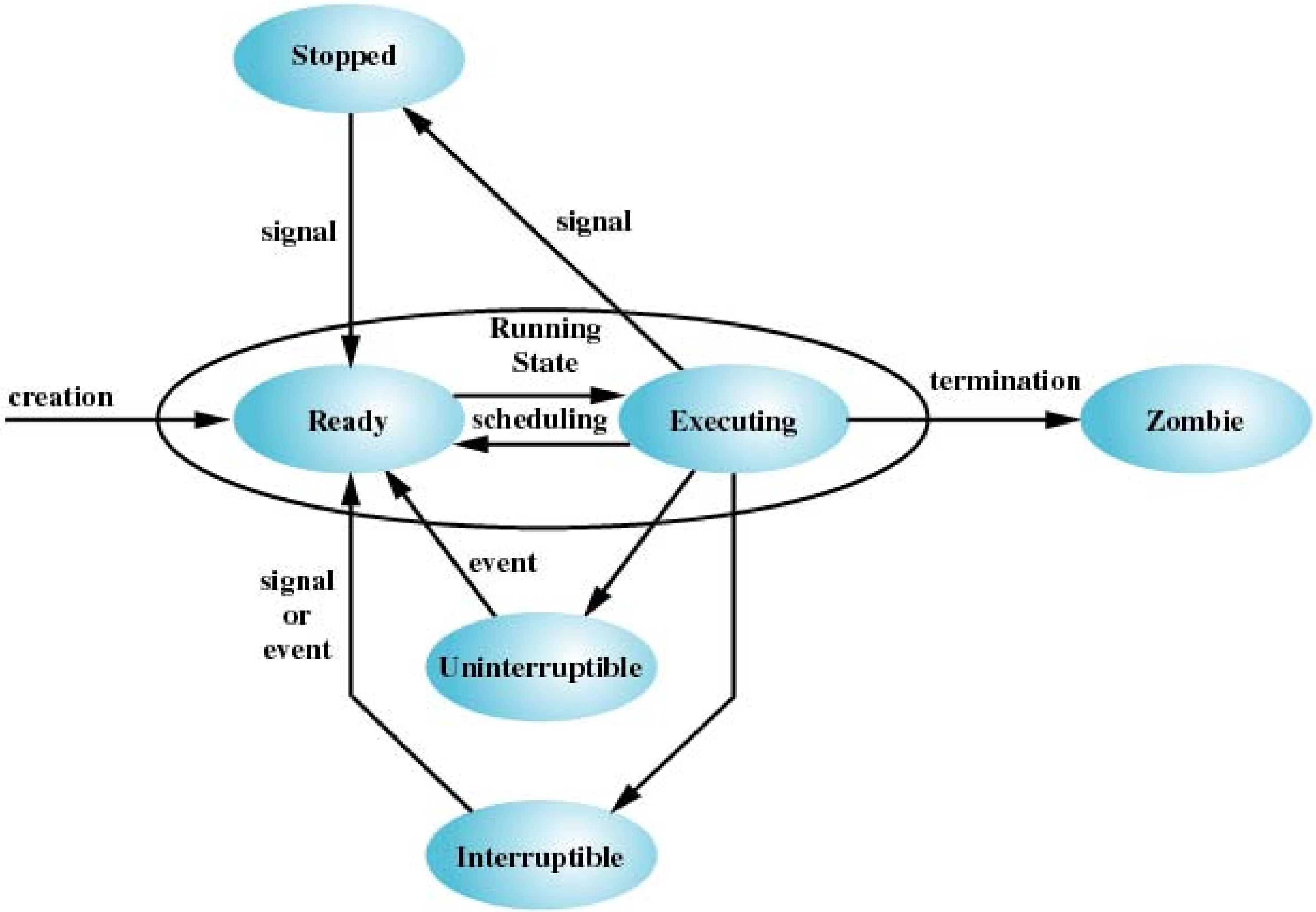


Figure 4.18 Linux Process/Thread Model

UPPSALA UNIVERSITET

Informationsteknologi

Concurrency

- Multiple applications
  - Multiprogramming
- Structured application
  - Application can be a set of concurrent processes
- Operating-system structure
  - Operating system is a set of processes or threads

UPPSALA UNIVERSITET

Informationsteknologi

Concurrency

Table 5.1 Some Key Terms Related to Concurrency

critical section	A section of code within a process that requires access to shared resources and which may not be executed while another process is in a corresponding section of code.
deadlock	A situation in which two or more processes are unable to proceed because each is waiting for one of the others to do something.
livelock	A situation in which two or more processes continuously change their state in response to changes in the other process(es) without doing any useful work.
mutual exclusion	The requirement that when one process is in a critical section that accesses shared resources, no other process may be in a critical section that accesses any of those shared resources.
race condition	A situation in which multiple threads or processes read and write a shared data item and the final result depends on the relative timing of their execution.
starvation	A situation in which a runnable process is overlooked indefinitely by the scheduler; although it is able to proceed, it is never chosen.

UPPSALA UNIVERSITET

Informationsteknologi

Difficulties of Concurrency

- Sharing of global resources
- Operating system managing the allocation of resources optimally
- Difficult to locate programming errors

UPPSALA UNIVERSITET

Informationsteknologi

A Simple Example

```
void echo()
{
  chin = getchar();
  chout = chin;
  putchar(chout);
}
```

UPPSALA UNIVERSITET

Informationsteknologi

A Simple Example

Process P1

```
.
chin = getchar();
.
chout = chin;
putchar(chout);
.
.
```

Process P2

```
.
.
chin = getchar();
chout = chin;
.
putchar(chout);
.
```

UPPSALA UNIVERSITET

Informationsteknologi

Race Condition

- A race condition occurs when multiple processes or threads read and write data items so that the final result depends on the order of execution of instructions in the multiple processes or threads.

UPPSALA UNIVERSITET

Informationsteknologi

Operating System Concerns

- Keep track of various processes
- Allocate and deallocate resources
  - Processor time
  - Memory
  - Files
  - I/O devices
- Protect data and resources
- Output of process must be independent of the speed of execution of other concurrent processes

Table 5.2 Process Interaction

Degree of Awareness	Relationship	Influence that one Process has on the Other	Potential Control Problems
Processes unaware of each other	Competition	•Results of one process independent of the action of others •Timing of process may be affected	•Mutual exclusion •Deadlock (renewable resource) •Starvation
Processes indirectly aware of each other (e.g., shared object)	Cooperation by sharing	•Results of one process may depend on information obtained from others •Timing of process may be affected	•Mutual exclusion •Deadlock (renewable resource) •Starvation •Data coherence
Processes directly aware of each other (have communication primitives available to them)	Cooperation by communication	•Results of one process may depend on information obtained from others •Timing of process may be affected	•Deadlock (consumable resource) •Starvation

UPPSALA UNIVERSITET
Informationsteknologi

Competition Among Processes for Resources

- Mutual Exclusion
  - Critical sections
    - Only one program at a time is allowed in its critical section
    - Example only one process at a time is allowed to send command to the printer
- Deadlock
- Starvation

UPPSALA UNIVERSITET
Informationsteknologi

Requirements for Mutual Exclusion

- Only one process at a time is allowed in the critical section for a resource
- A process that halts in its noncritical section must do so without interfering with other processes
- No deadlock or starvation

UPPSALA UNIVERSITET

Informationsteknologi

Requirements for Mutual Exclusion

- A process must not be delayed access to a critical section when there is no other process using it
- No assumptions are made about relative process speeds or number of processes
- A process remains inside its critical section for a finite time only