Fontys
UNIVERSITY OF
APPLIED SCIENCES

› FOR SOCIETY

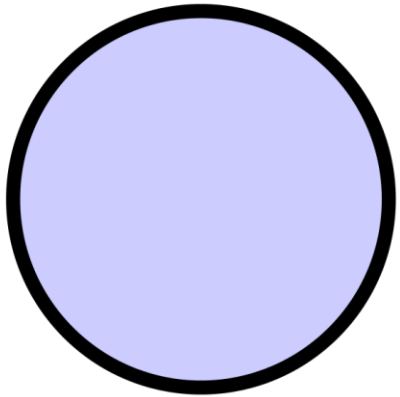
Chapter 4 – Threads
Lecture 4

Roadmap

- **Processes and Threads**
- Types of threads
- Windows
- Solaris
- Linux

Processes and Threads

Process



Thread

Windows Taakbeheer

Bestand Opties Beeld Help

Toepassingen | Processen | Services | Prestaties | Netwerk | Gebruikers

Processorgebruik

5 %

Geschiedenis van processorgebruik

Geheugen

5,13 GB

Geschiedenis van gebruik van fysiek geheugen

Fysiek geheugen (MB)

Totaal	16318
In cache	3974
Beschikbaar	11061
Vrij	7302

Systeem

Ingangen	54147
Threads	1560
Processen	101
Tijd actief	1:02:52:11
Doorvoeren (G	6 / 31

Kernelgeheugen (MB)

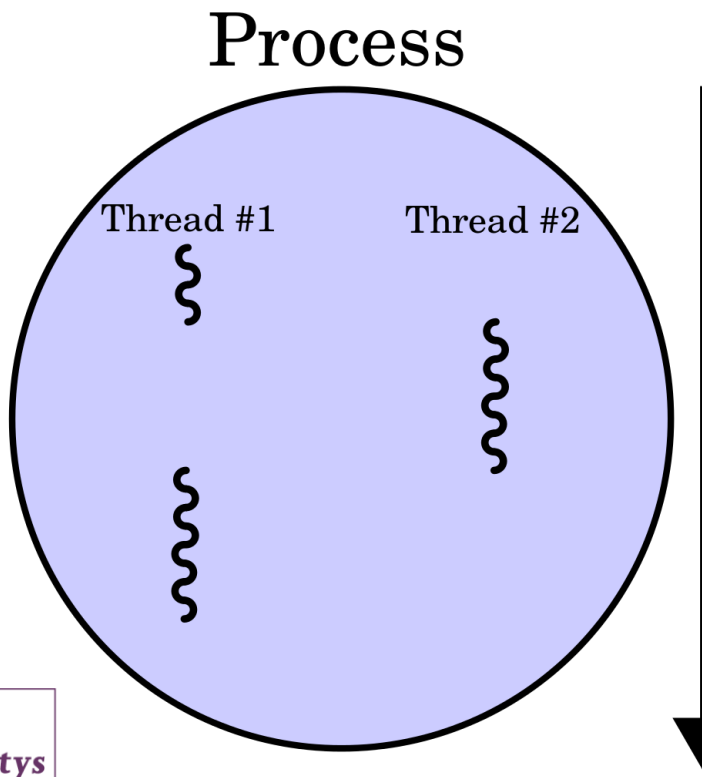
In wisselbestand	601
Niet in wisselbestand	175

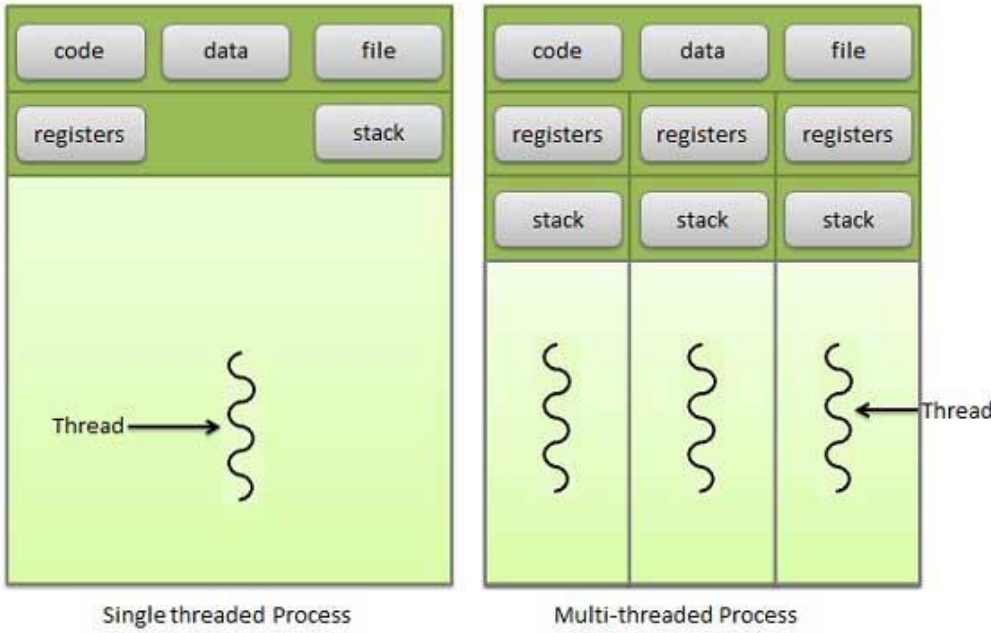
Broncontrole...

Processen: 101 | Processorgebruik: 5% | Fysiek geheugen: 32%

Fontys

Processes and threads





Multithreading

The ability of an OS to support multiple, concurrent paths of execution within a single process.

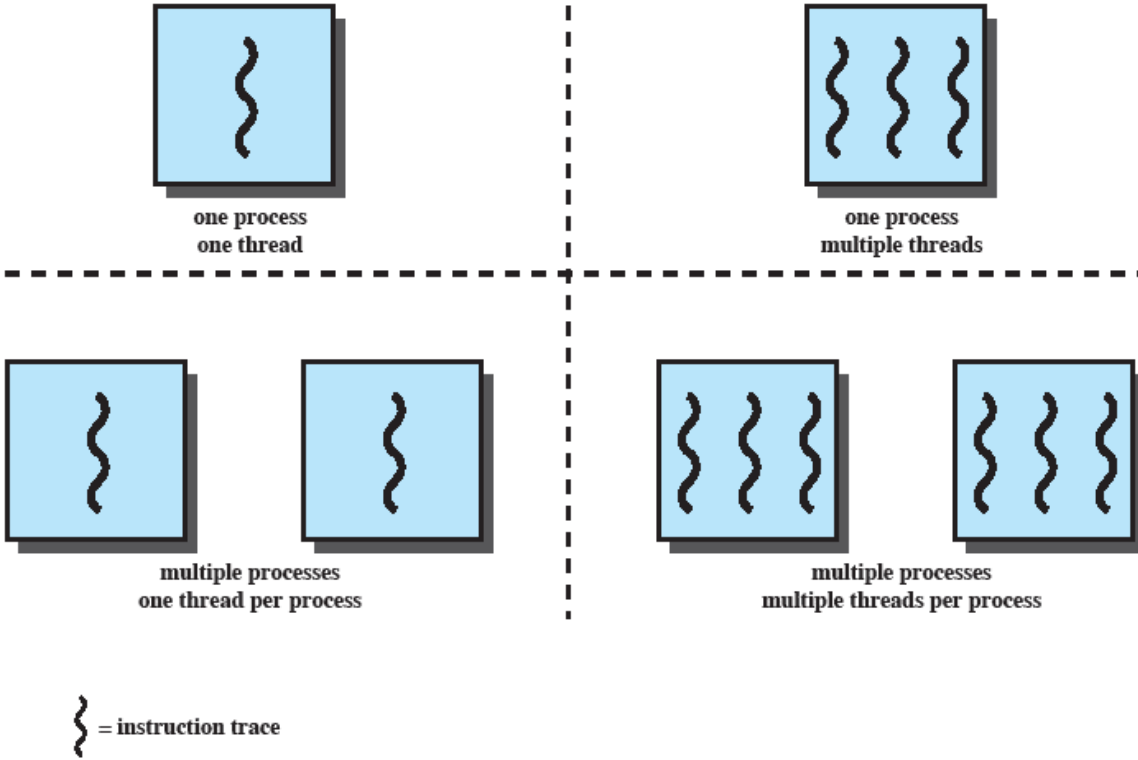


Figure 4.1 Threads and Processes [ANDE97]

Single Thread Approaches

- MS-DOS supports a single user process and a single thread.
- Some UNIX, support multiple user processes but only support one thread per process

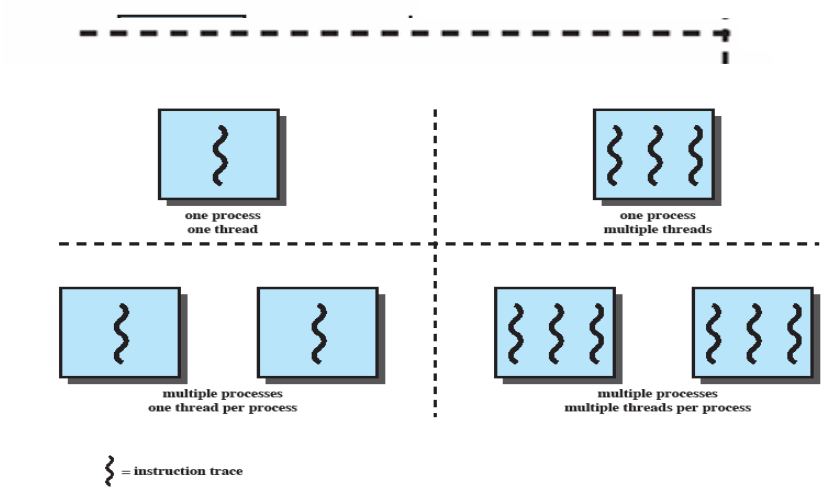


Figure 4.1 Threads and Processes [ANDE97]

Fontys

Multithreading

- Java run-time environment is a single process with multiple threads
- Multiple processes ***and*** threads are found in Windows, Solaris, and many modern versions of UNIX

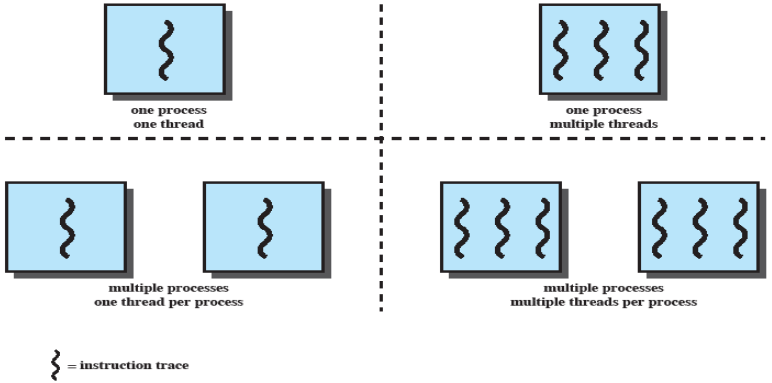


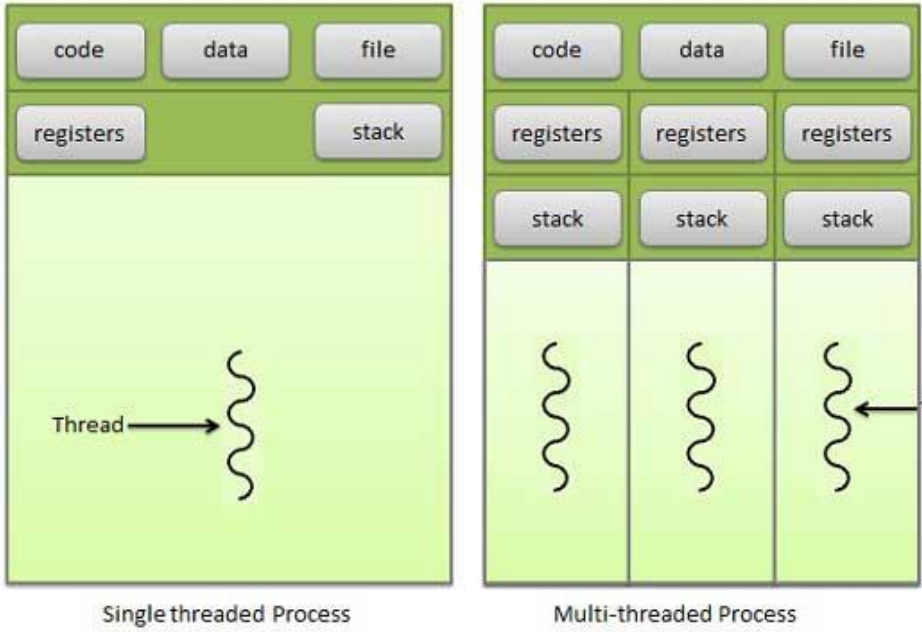
Figure 4.1 Threads and Processes [ANDE97]

Fontys

One or More Threads in Process

Each thread has

- An execution state (running, ready, etc.)
- 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 (all threads of a process share this)



Fontys

Threads vs. processes

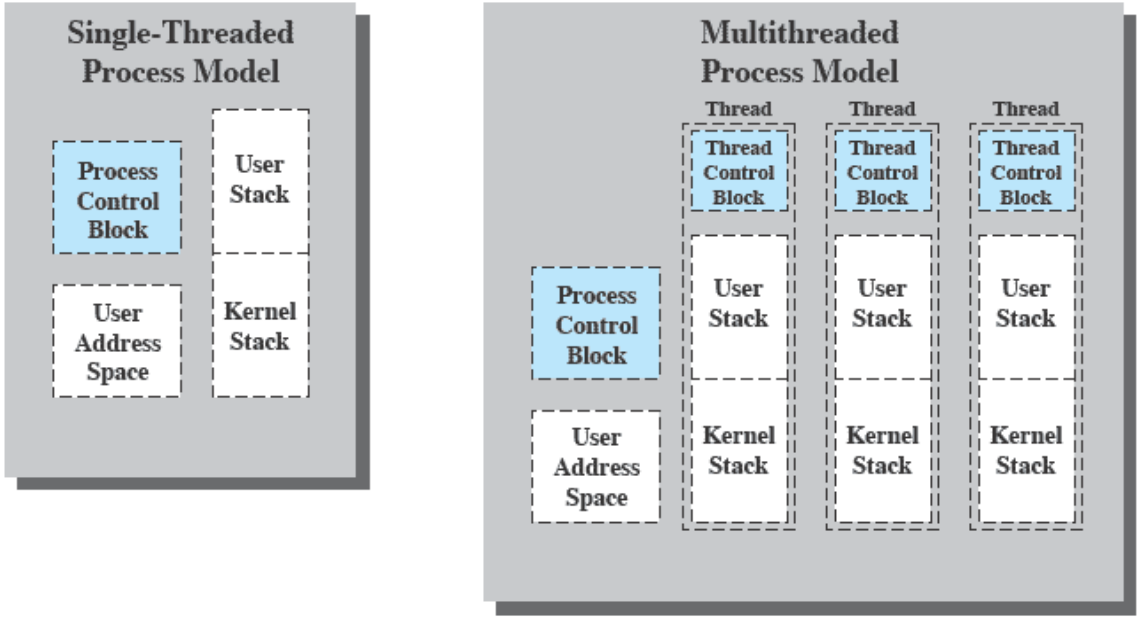


Figure 4.2 Single Threaded and Multithreaded Process Models

Benefits of Threads

- Takes less time to create a new thread
- Less time to terminate a thread
- Switching between two threads takes less time. Threads can communicate with each other
  (without invoking the kernel)
- Threads enhance efficiency in communication between programs

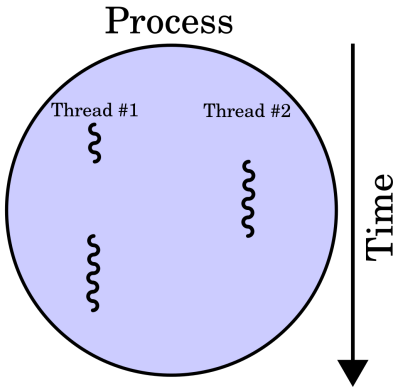
Fontys

Thread use in a Single-User System

- Foreground and background work
  - display // computation in spreadsheet
- Asynchronous processing
  - auto save in text editor
- Speed of execution
  - double buffering: parallelize processing with reading of next job
- Modular program structure
  - dynamically connect various producers with consumers of data

Fontys

Threads



- Several actions that affect all of the threads in a process
  - Suspending a process involves suspending all threads of the process
  - Termination of a process, terminates all threads within the process

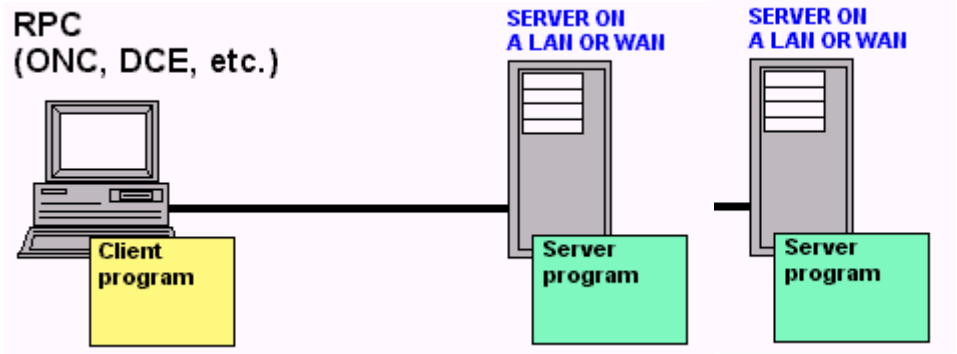
Fontys

Thread Execution States

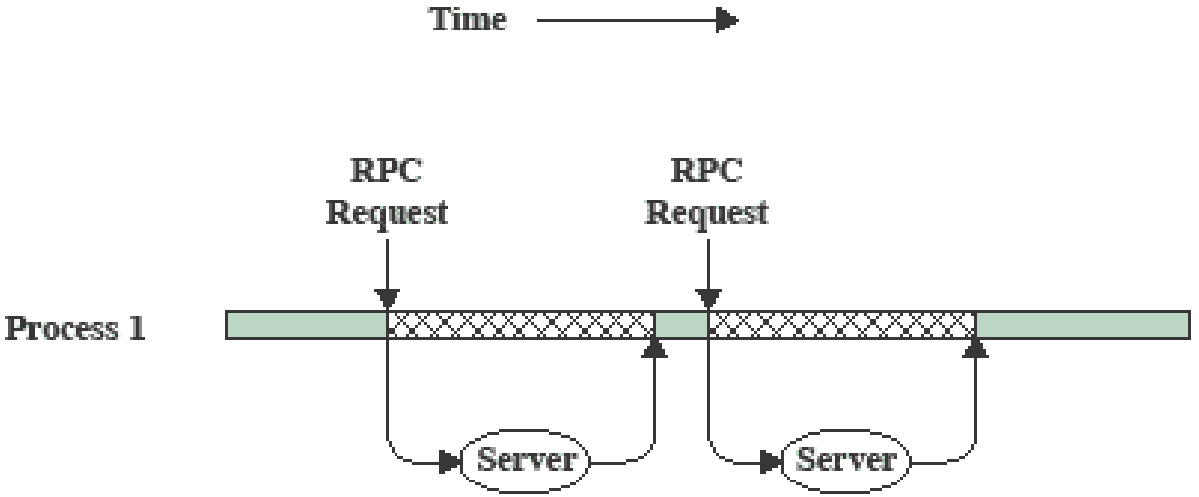
- Key states for a thread are:
  - Running
  - Ready
  - Blocked
- States associated with a change in thread state
  - Spawn (another thread)
  - Block
  - Unblock
  - Finish (thread)

Fontys

Example: Remote Procedure Call



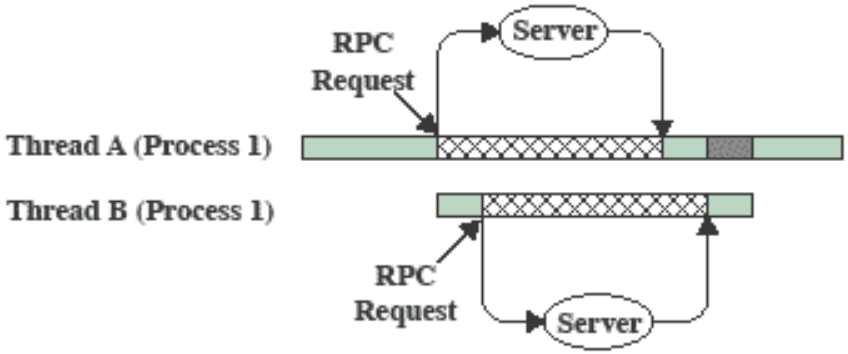
RPC: Using Single Thread



(a) RPC Using Single Thread

Fontys

RPC Using: One Thread per Server



(b) RPC Using One Thread per Server (on a uniprocessor)

Blocked, waiting for response to RPC

Blocked, waiting for processor, which is in use by Thread B

Running

Fontys

Multithreading on a Uniprocessor

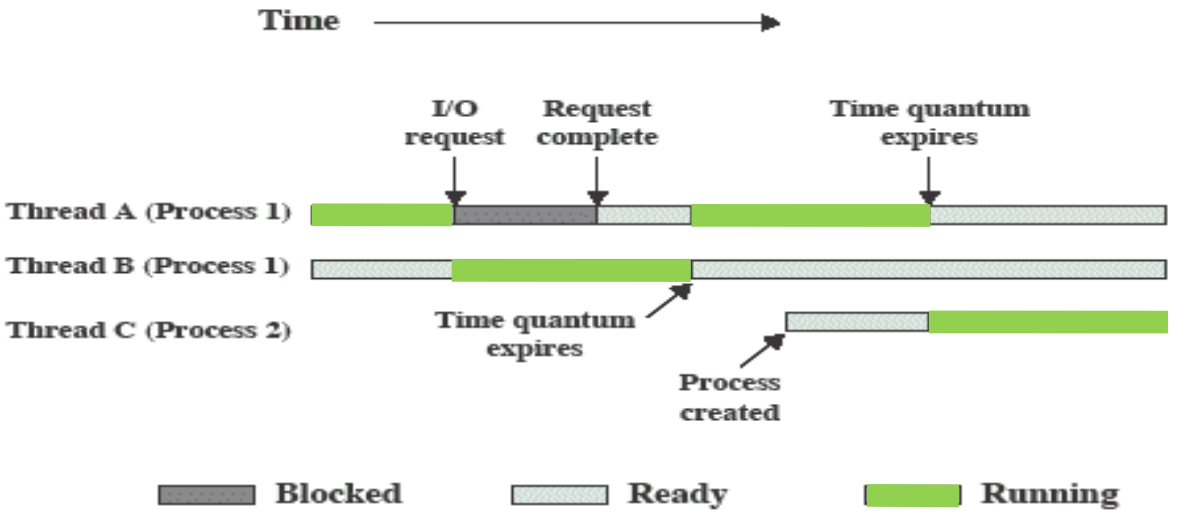


Figure 4.4 Multithreading Example on a Uniprocessor

Fontys

Roadmap

- Processes and Threads
- **Types of threads**
- Windows
- Solaris
- Linux

Types of Threads

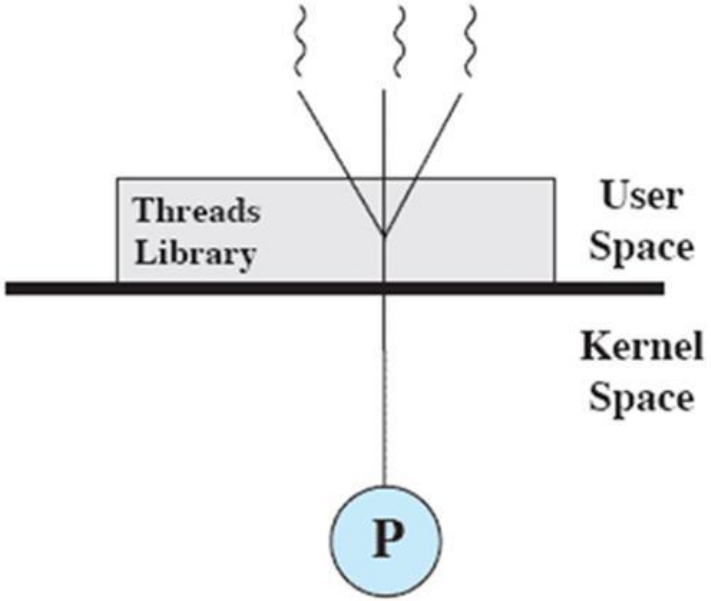
- User Level Thread (ULT)

- Kernel level Thread (KLT) also called:
  - kernel-supported threads
  - lightweight processes.

Fontys

User-Level Threads

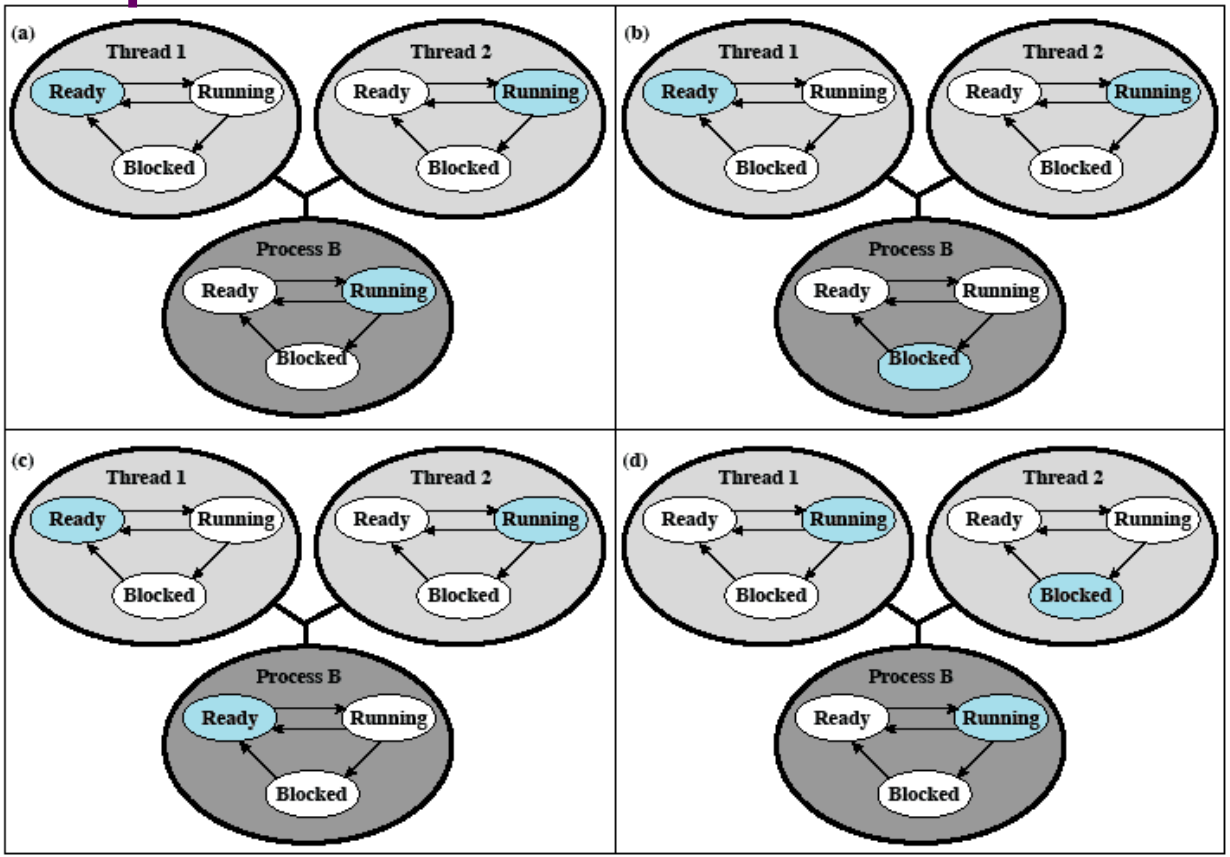
- All thread management is done by the application
- The kernel is not aware of the existence of threads



(a) Pure user-level

Fontys

Relationships between ULT Thread and Process States



Colored state
is current state

Figure 4.7 Examples of the Relationships Between User-Level Thread States and Process States

Advantages – Disadvantages ULT's

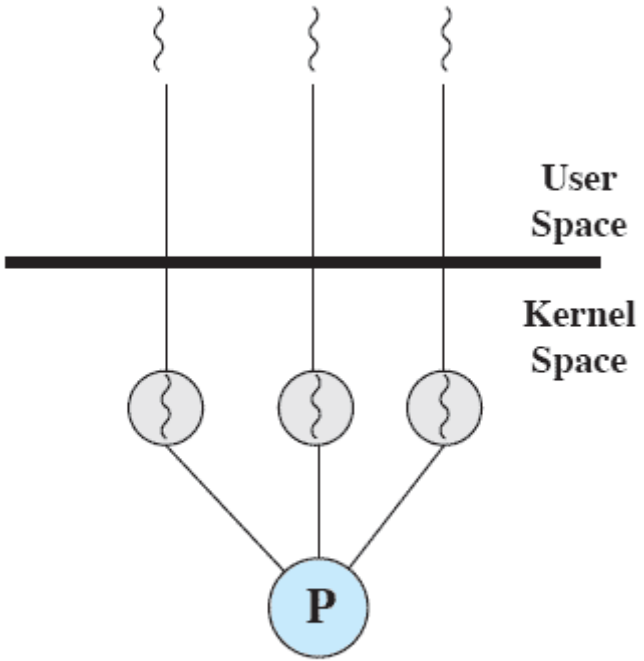
Advantages

- Thread switching does not require kernel mode privileges
- Scheduling can be application specific
- ULTs can run on any OS

Disadvantages

- In a typical OS, many system calls are blocking
- In a pure ULT strategy, a multithreaded application cannot take advantage of multiprocessing

Kernel-Level Threads

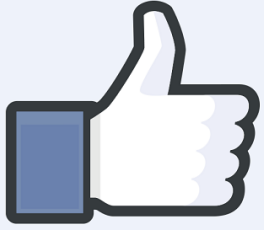


(b) Pure kernel-level

- Kernel maintains context information for the process and the threads
  (No thread management done by application)
- Scheduling is done on a thread basis

Windows is an example of this approach

Advantages of KLT



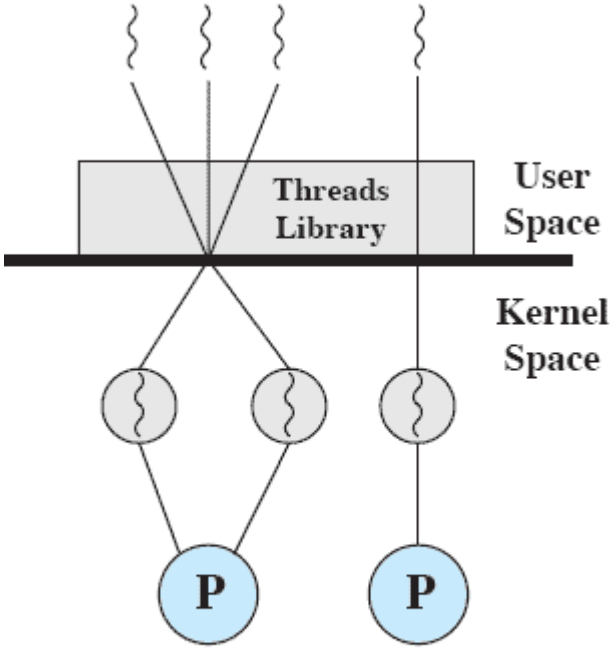
- The kernel can simultaneously schedule multiple threads from the same process on multiple processors.
- If one thread in a process is blocked, the kernel can schedule another thread of the same process.
- Kernel routines themselves can be multithreaded.

Fontys

Combined Approaches

- Thread creation done in the user space
- Bulk of scheduling and synchronization of threads by the application

Example is Solaris



(c) Combined

Fontys

Roadmap

- Processes and Threads
- Types of threads
- **Windows**
- Solaris
- Linux
- Android

Different Approaches to Processes

- Differences between different OS's support of processes include
  - How processes are named
  - Whether threads are provided
  - How processes are represented
  - How process resources are protected
  - What mechanisms are used for inter-process communication and synchronization
  - How processes are related to each other

Fontys

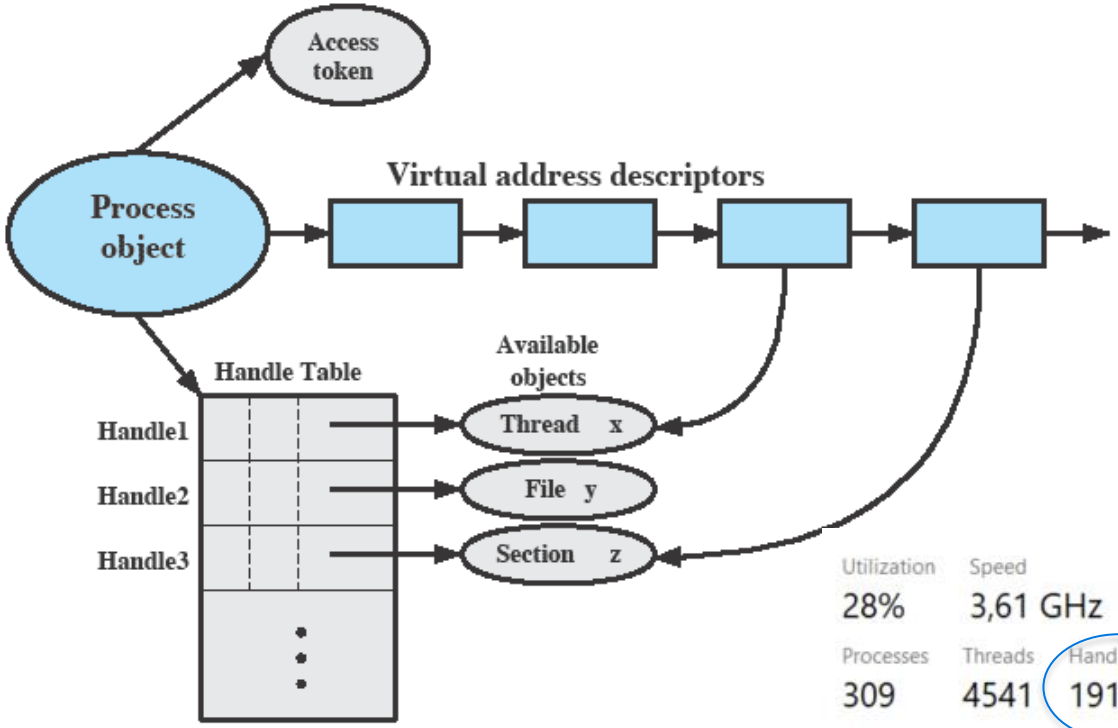
Windows Processes

Processes and services provided by the Windows Kernel are relatively simple and general purpose

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

Fontys

Relationship between Process and Resources



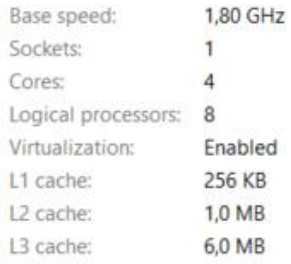
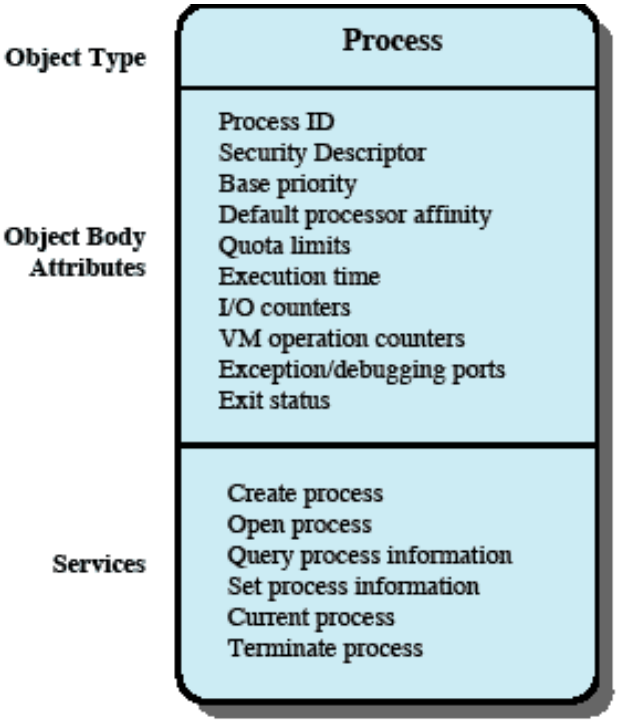


Figure 4.12 A Windows Process and Its Resources

Fontys

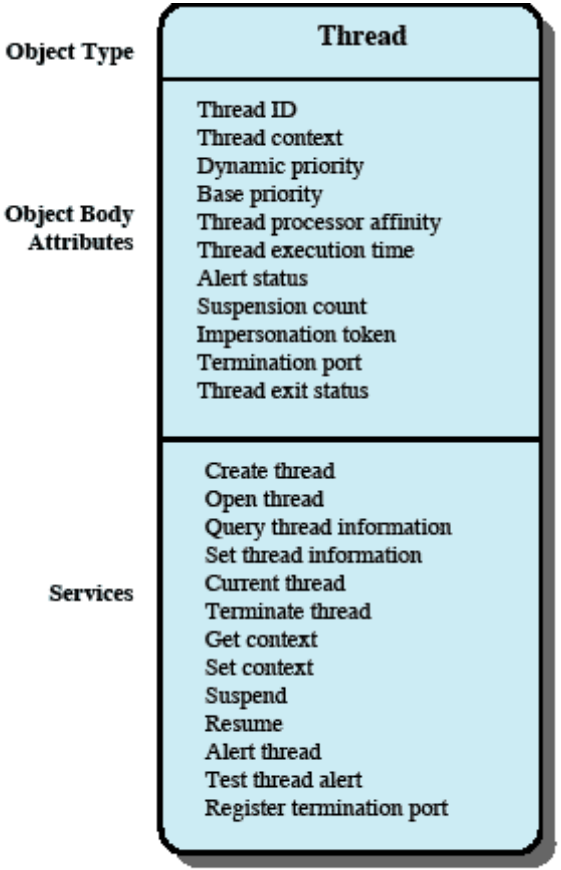
Windows Process Object



(a) Process object

Fontys

Windows Thread Object



(b) Thread object

Fontys

Thread States

- Ready
- Standby
  - To run next on specific proc
- Running
- Waiting
- Transition
  - Unblocked but waiting for mem.
- Terminated

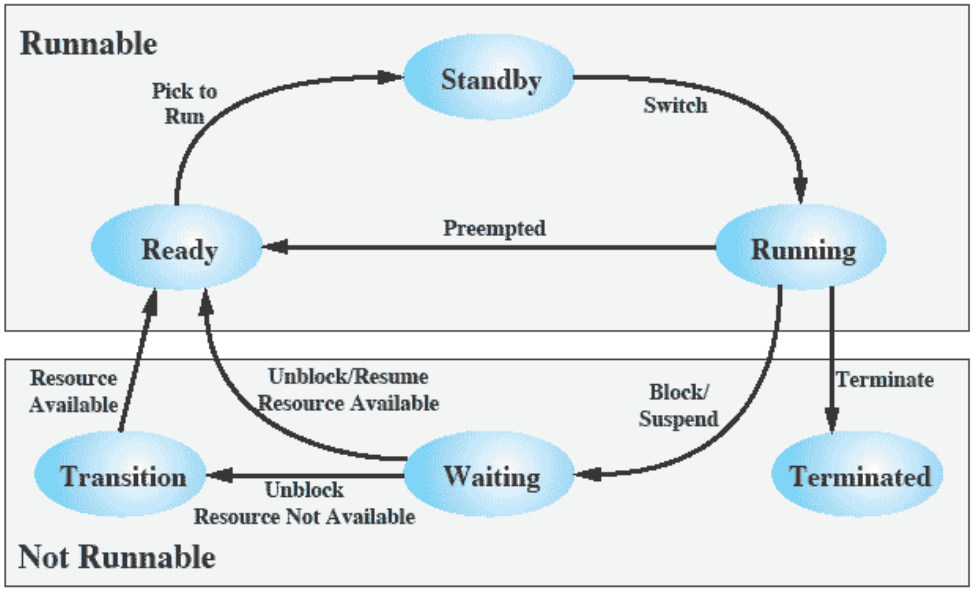


Figure 4.14 Windows Thread States

Windows SMP Support

- Threads can run on any processor
  - But an application can restrict affinity
- Soft Affinity
  - The dispatcher tries to assign a ready thread to the same processor it last ran on.
  - This helps reuse data still in that processor's memory caches from the previous execution of the thread.
- Hard Affinity
  - An application restricts threads to certain processor

Fontys › FOR SOCIETY

Roadmap

- Processes and Threads
- Types of threads
- Windows
- **Solaris**
- Linux
- Android

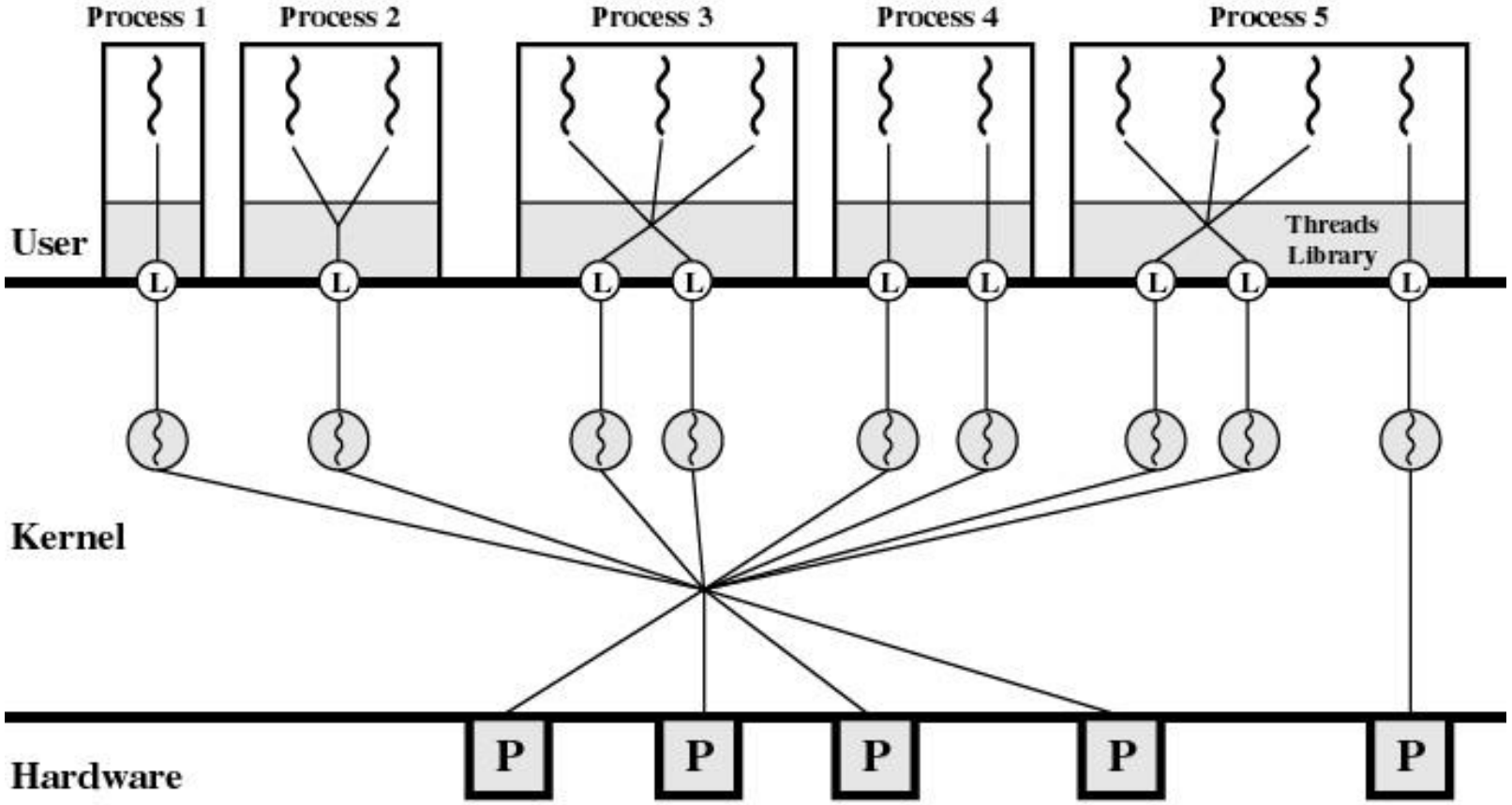
Solaris Process

Solaris makes use of 4 separate thread-related concepts to provide flexibility in exploiting processor resources:

1) Process: includes the user's address space, stack, and process control block.
2) User-level threads: a user-created unit of execution within a process.
3) Lightweight processes: a mapping between ULTs and kernel threads.
4) Kernel threads (includes interrupts)

Fontys

Solaris Multithreading: example



Fontys

Solaris Thread States

- RUN
  - The thread is runnable
- ONPROC
  - The thread is executing
- SLEEP
  -The thread is blocked
- STOP
  - The thread is stopped
- ZOMBIE
  - The thread has terminated
- PINNED
  - A thread assigned to a particular core is swappable

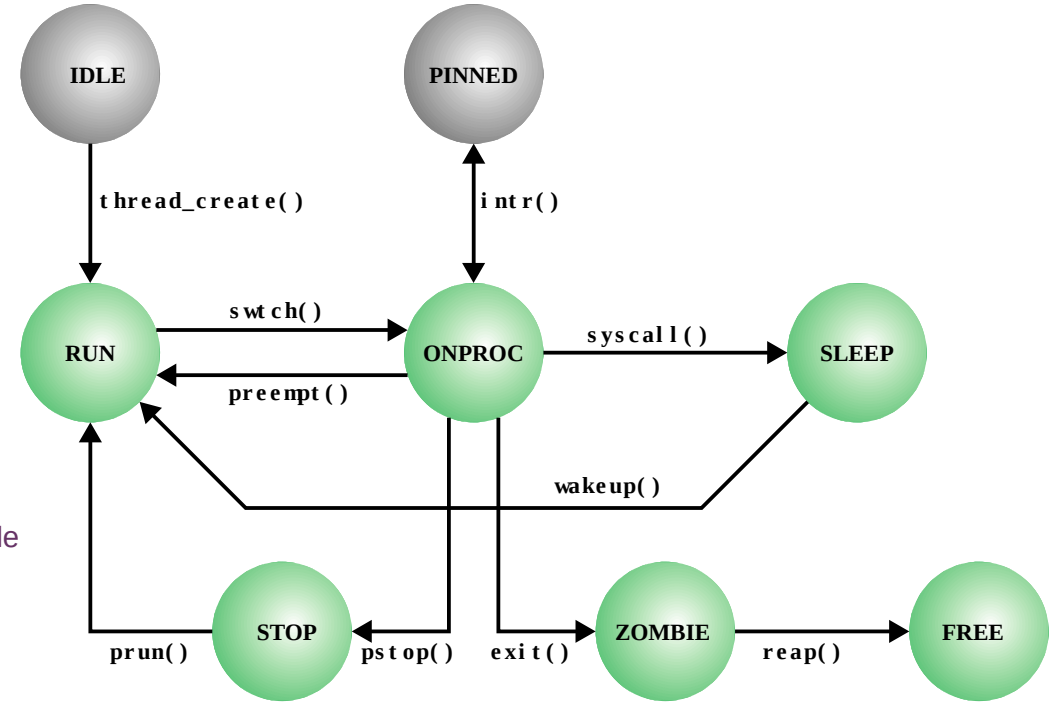


Figure 4.14 Solaris Thread States

Fontys › FOR SOCIETY

Roadmap

- Processes and Threads
- Types of threads
- Windows
- Solaris
- **Linux**
- Android

Linux Process/Thread Model

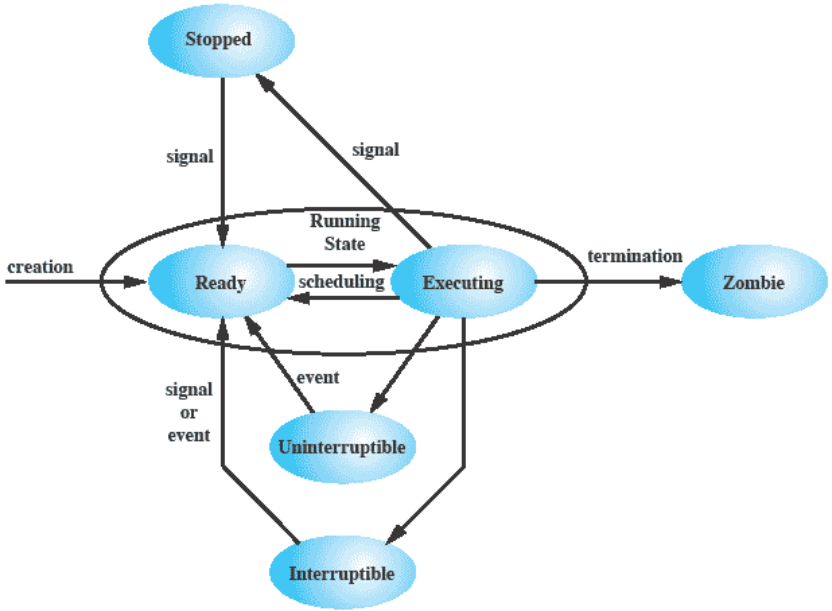


Figure 4.18 Linux Process/Thread Model

- Running
- ready or executing
- Interruptible
  - - blocked for IO
- Uninterruptible
  - - blocked/waiting for hardware conditions
- Stopped
  - - halted (waiting for other process)
- Zombie
  - - terminated

Fontys › FOR SOCIETY

Roadmap

- Processes and Threads
- Types of threads
- Windows
- Solaris
- Linux
- **Android**

Android – Process and thread management

- An Android application is the software that implements an app
- Each Android application consists of one or more instance of one or more of four types of application components
- Each component performs a distinct role in the overall application behavior, and each component can be activated independently within the application and even by other applications
- Four types of components:
  - Activities
  - Services
  - Content providers
  - Broadcast receivers

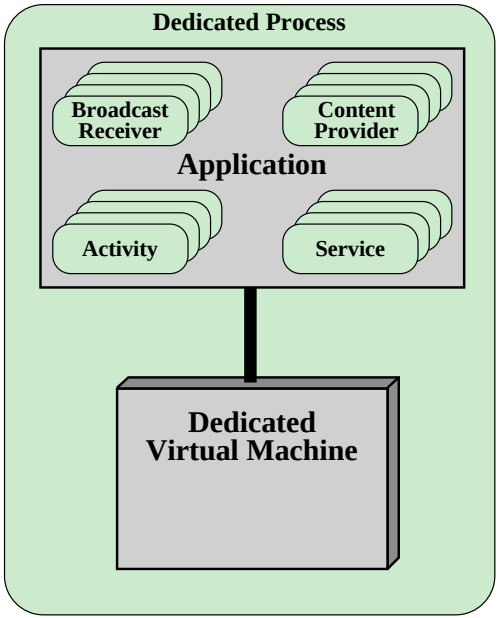


Figure 4.16 Android Application

Questions?

Fontys