Chapter 4
Threads

Part **C**: Sections 4.**5**–*end*

Solaris

- uses four thread-related concepts:

Process	• Includes the user's address space, stack, and process control block: ➔ *see Chapter 3: UNIX*
User-level Threads	• A user-created unit of execution within a process
Lightweight Processes (LWP)	• A mapping between ULTs and kernel threads
Kernel Threads	• Fundamental entities that can be scheduled and dispatched to run on one of the system's processors

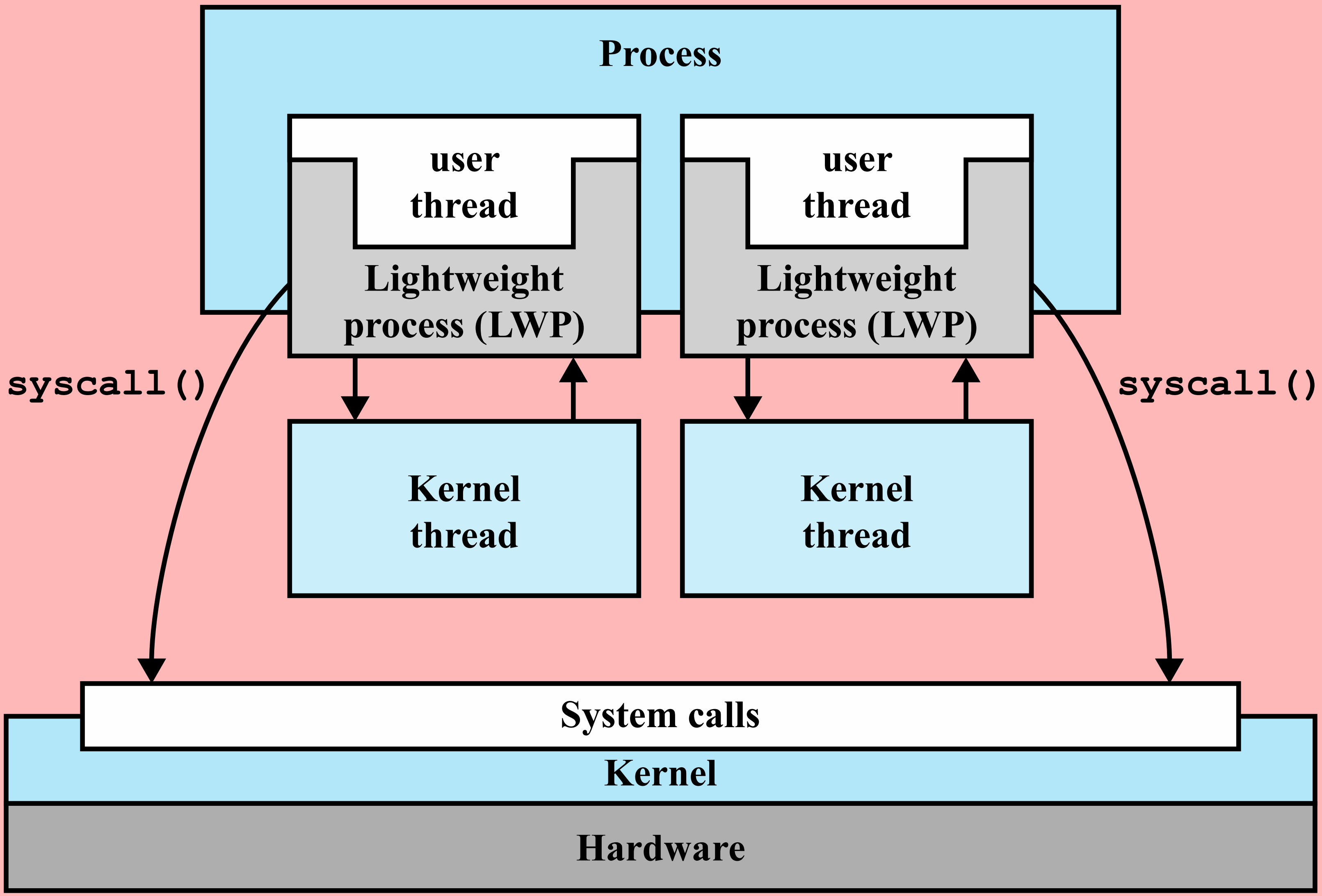


Figure 4.12 Processes and Threads in Solaris

A Lightweight Process (LWP) Data Structure Includes:

- An LWP identifier
- The priority of this LWP and hence the kernel thread that supports it
- A signal mask that tells the kernel which signals will be accepted
- Saved values of user-level registers
- The kernel stack for this LWP, which includes system call arguments, results, and error codes for each call level
- Resource usage and profiling data
- Pointer to the corresponding kernel thread
- Pointer to the process structure

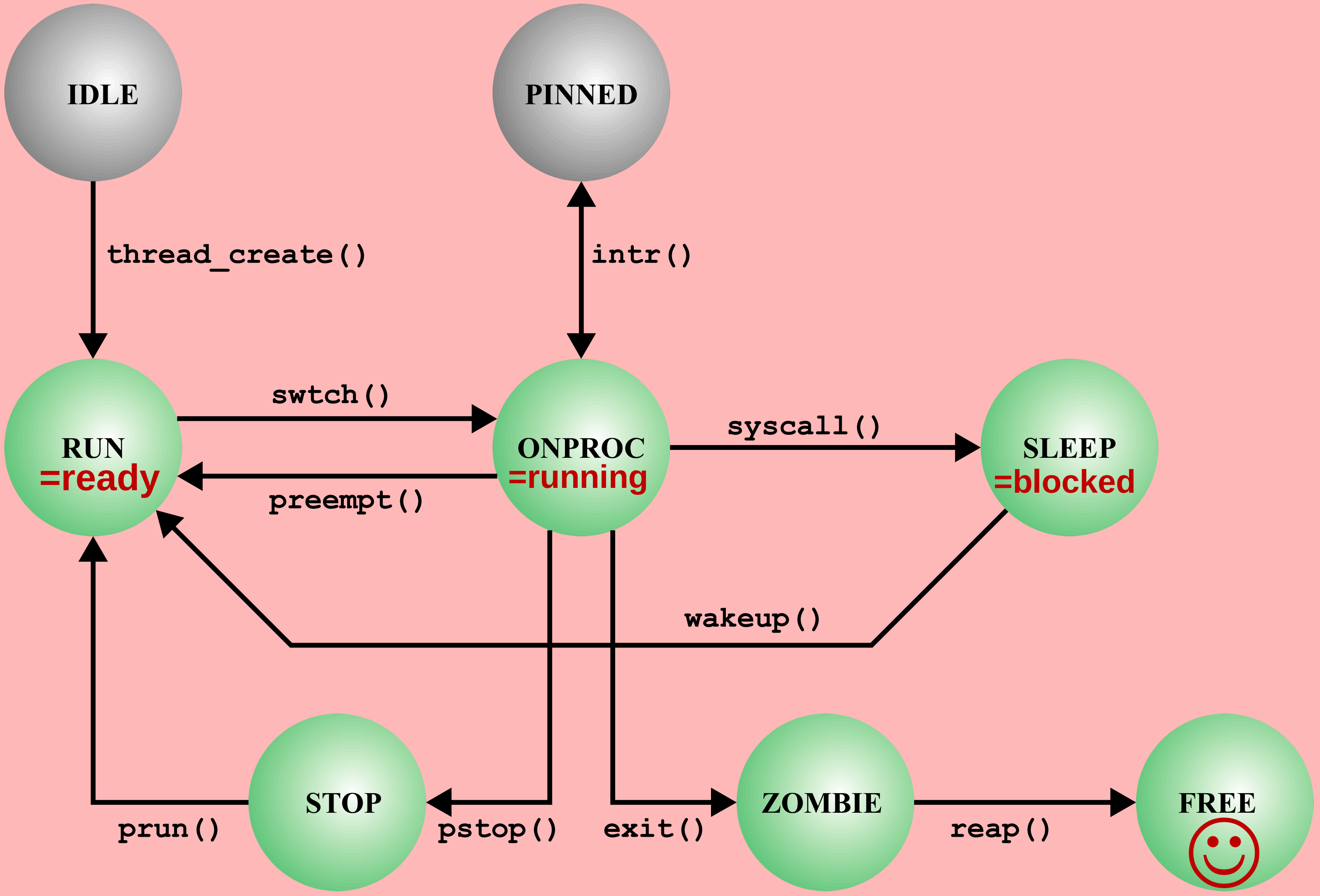


Figure 4.14 Solaris Thread States

Solaris:

Interrupts *as Threads*

- Most operating systems contain two fundamental forms of concurrent activity:

Processes (threads)	Cooperate with each other and manage the use of shared data structures by primitives that enforce mutual exclusion and synchronize their execution
Interrupts	Synchronized by preventing their handling for a period of time

- **Solaris unifies these two concepts into a single model**, namely kernel threads, and the mechanisms for scheduling and executing kernel threads
  - To do this, **interrupts are converted to kernel threads**

Solaris Solution

- Solaris employs a set of **pre-defined kernel threads to handle interrupts**
  - An interrupt thread has its own identifier, priority, context, and stack
  - The kernel controls access to data structures and synchronizes among interrupt threads using ***mutual exclusion*** mechanisms ➔ **see Chapter 5**.
  - Interrupt threads are assigned higher priorities than all other types of kernel threads

Linux *Threads*

- Linux does NOT recognize a difference between threads and processes
- User-level threads are mapped into kernel-level processes
- A new process is created by copying the attributes of the current process
- The new process can be *cloned* so that it shares resources
- The clone() call creates separate stack spaces for each process

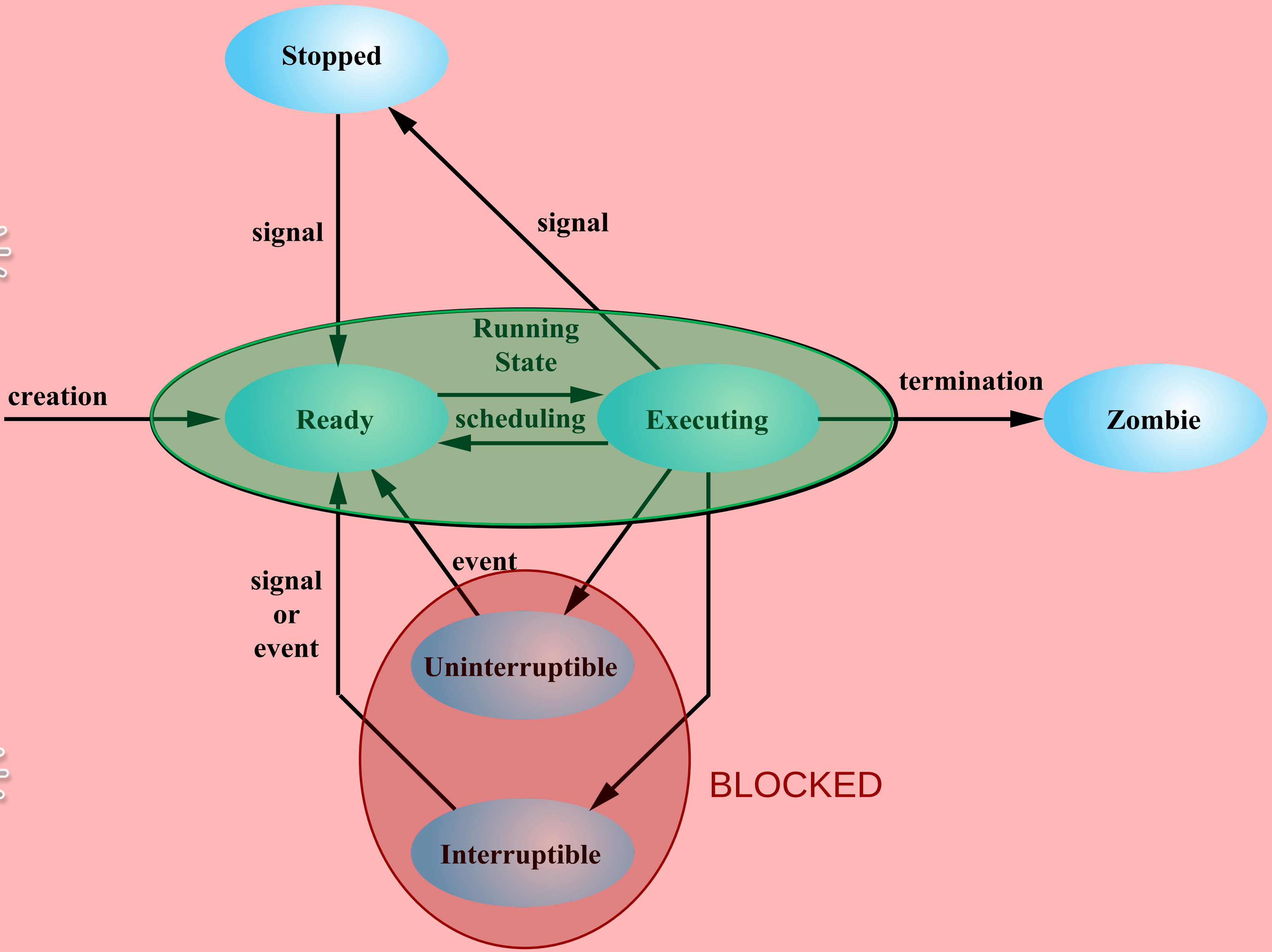


Figure 4.15 Linux Process/Thread Model

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

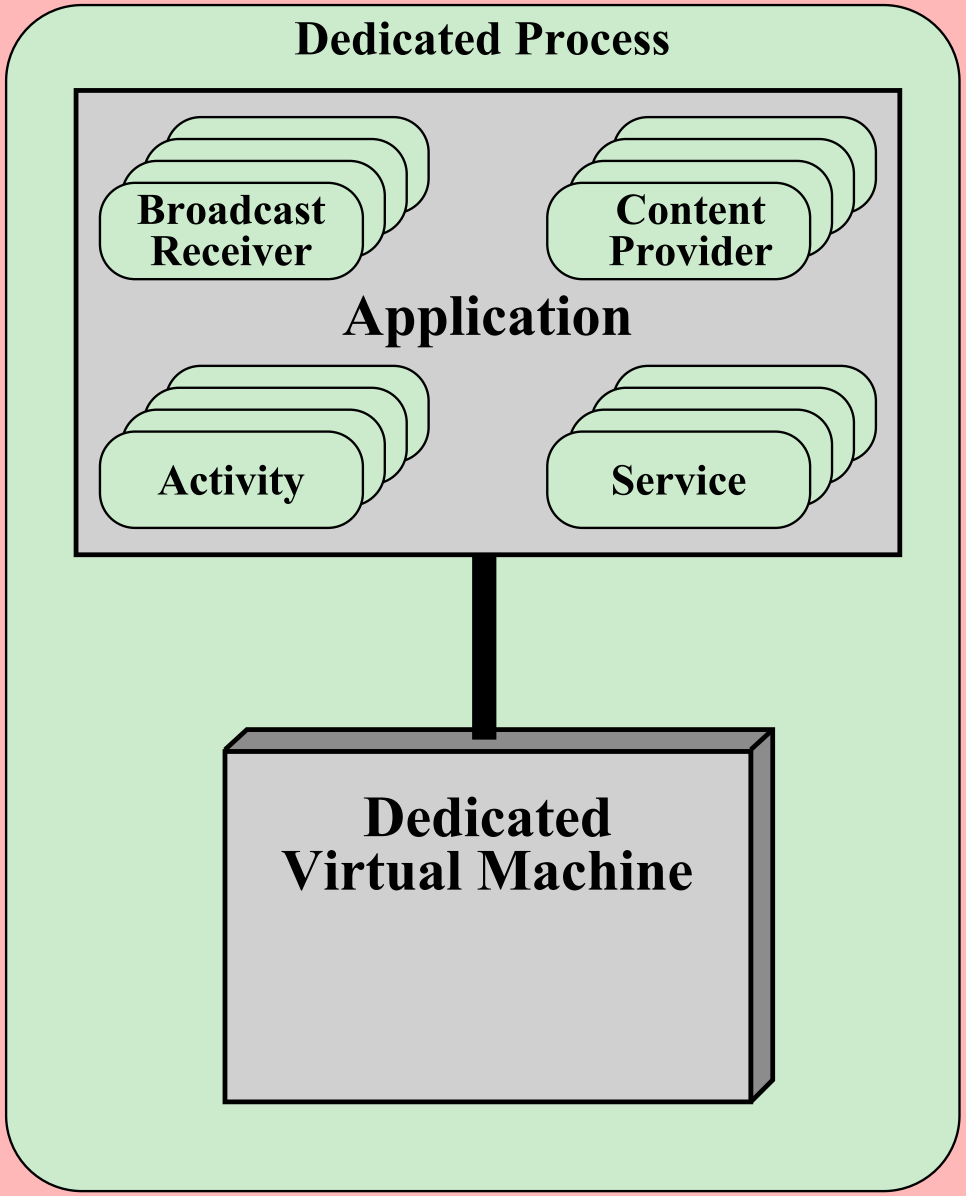


Figure 4.16 Android Application

Android: *Activities*

- An Activity is an application component that provides a screen with which users can interact in order to do something
- Each Activity is given a window in which to draw its user interface
- The window typically fills the screen, but may be smaller than the screen and float on top of other windows
- An application may include multiple activities
- When an application is running, one activity is in the foreground, and it is this activity that interacts with the user
- The activities are arranged in a last-in-first-out stack in the order in which each activity is opened
- If the user switches to some other activity within the application, the new activity is created and pushed on to the top of the back stack, while the preceding foreground activity becomes the second item on the stack for this application

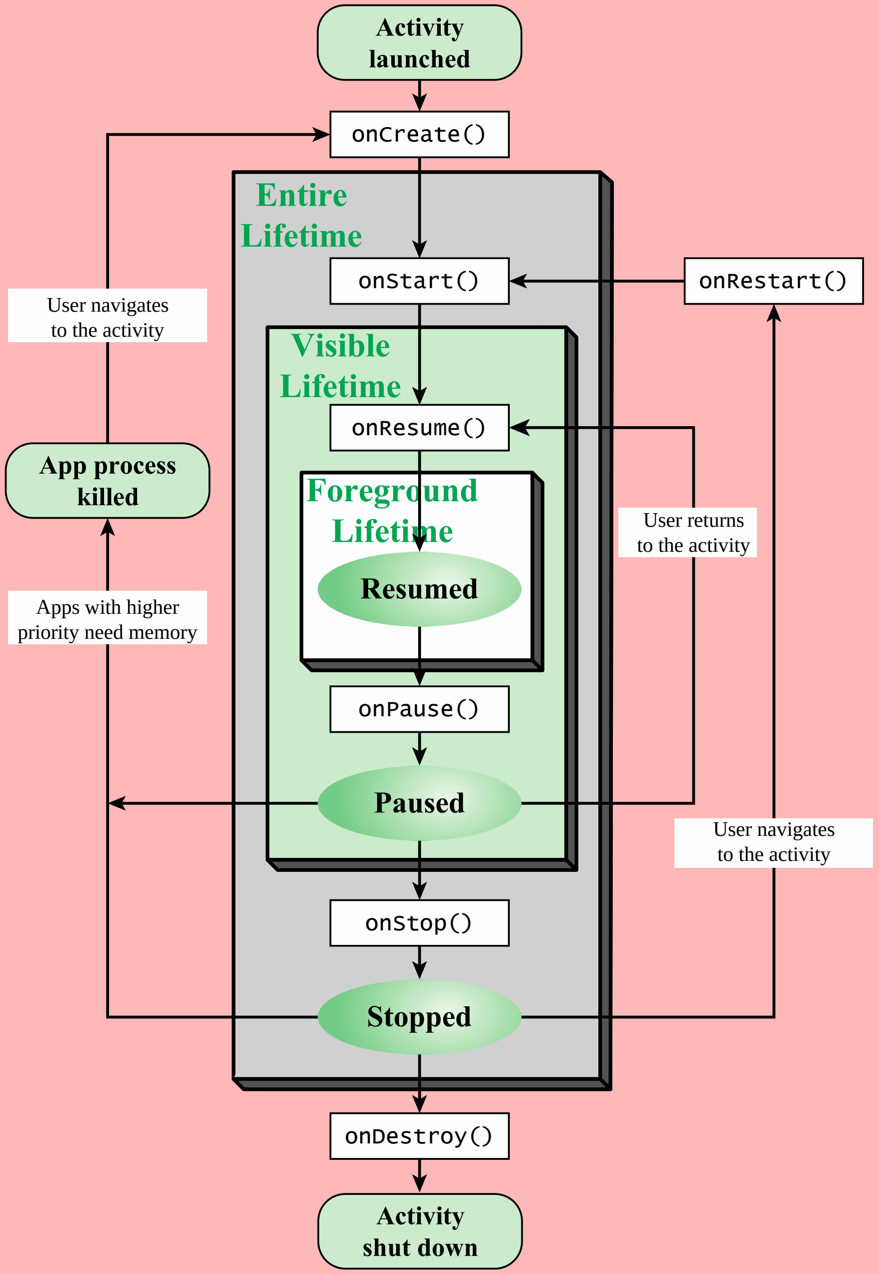
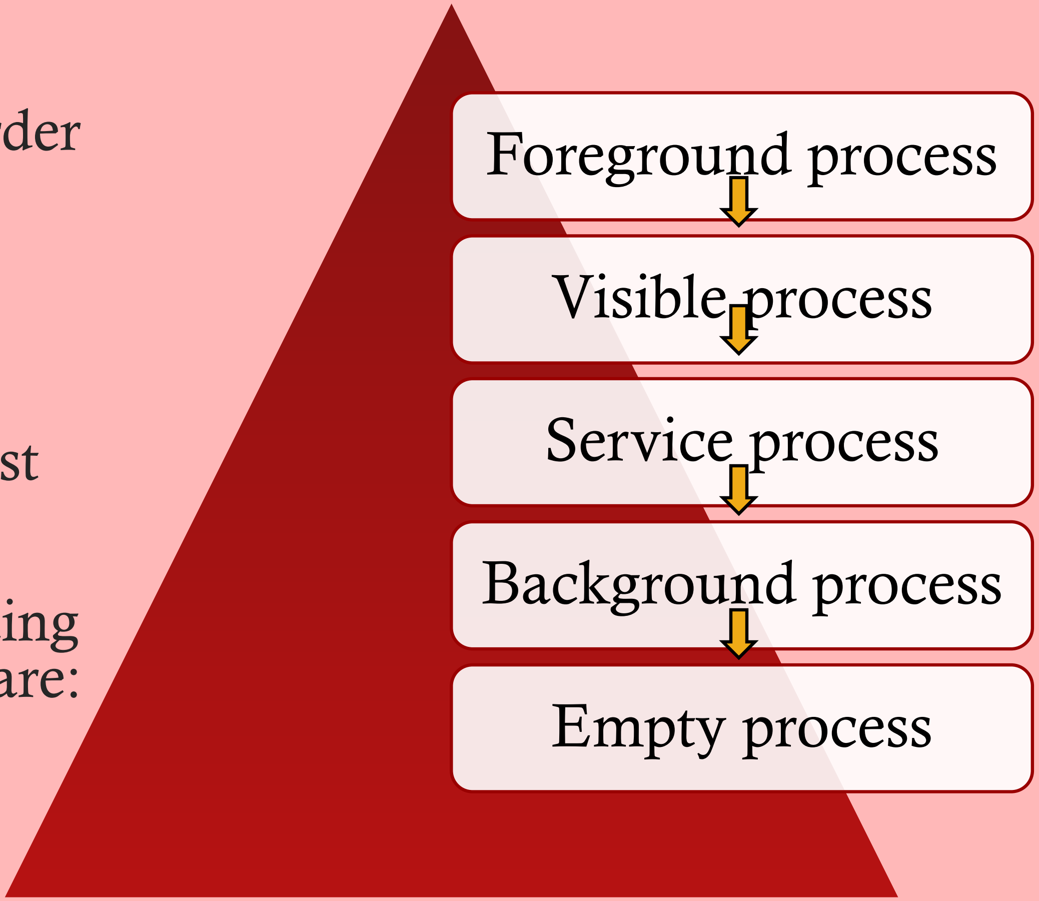


Figure 4.17 Activity State Transition Diagram

Processes and *Threads*

- A precedence hierarchy is used to determine which process or processes to kill in order to reclaim needed resources
- Processes are killed beginning with the lowest precedence first
- The levels of the hierarchy, in descending order of precedence are:



Mac OS X Grand Central Dispatch (GCD)

- Provides a **pool** of available **threads**
- User-level Designers can designate portions of applications, so-called "*blocks*", that can be dispatched independently and run concurrently
- **Concurrency** is based on the number of cores available and the thread capacity of the system