

Windows Processes

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

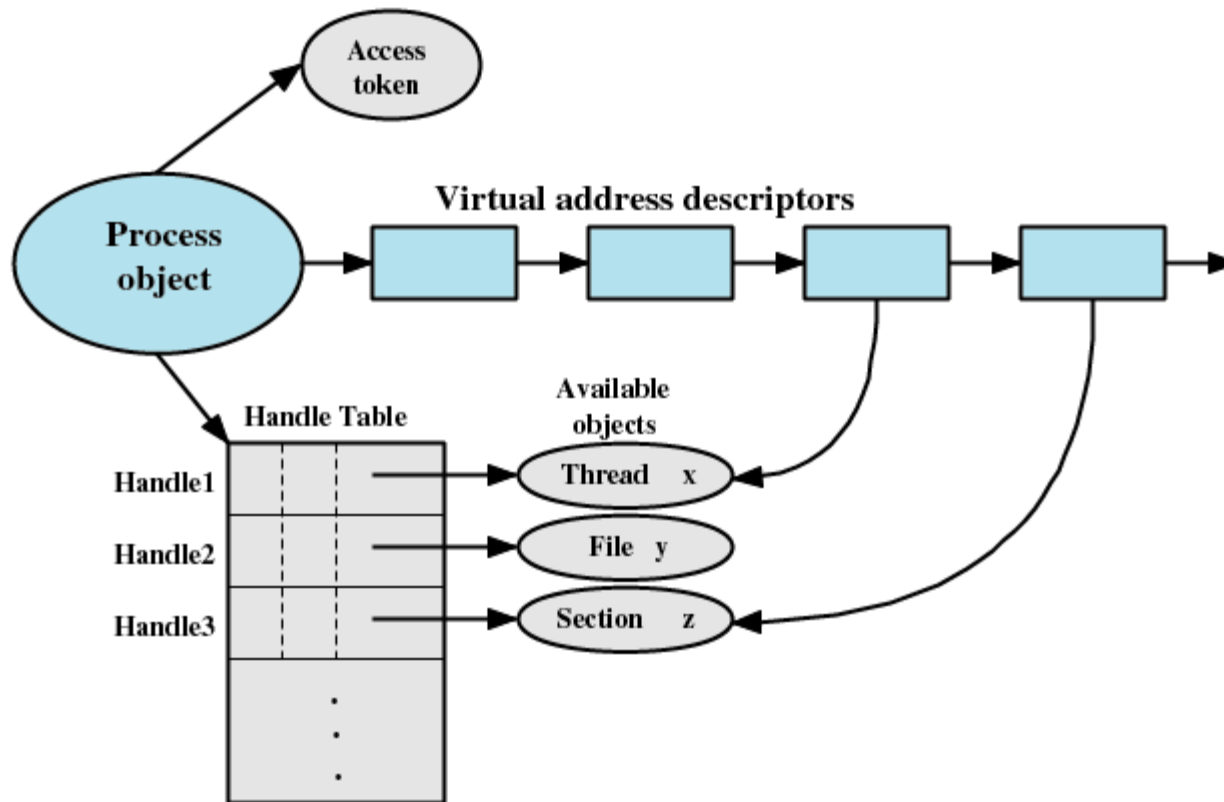
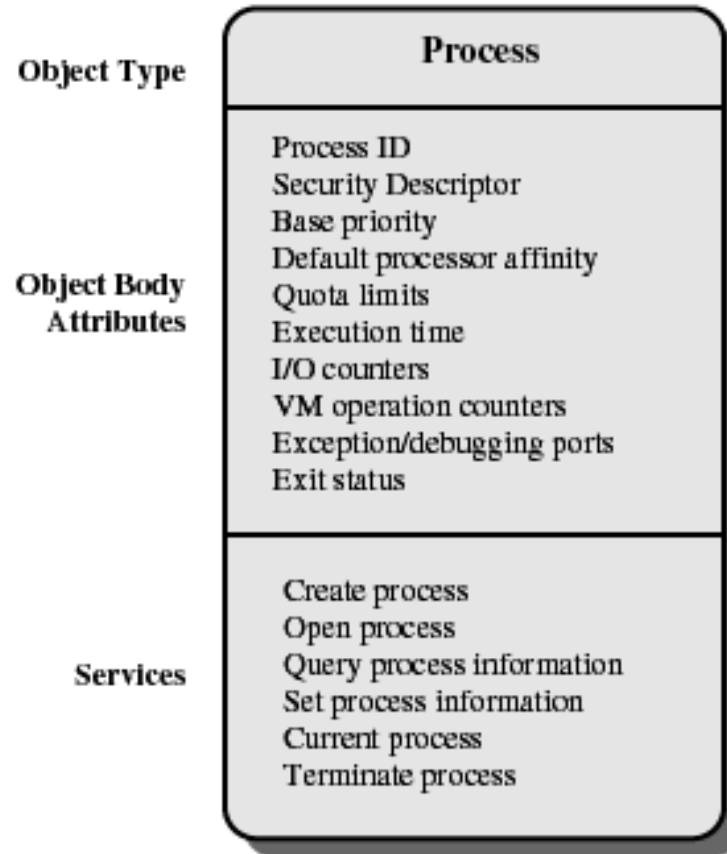


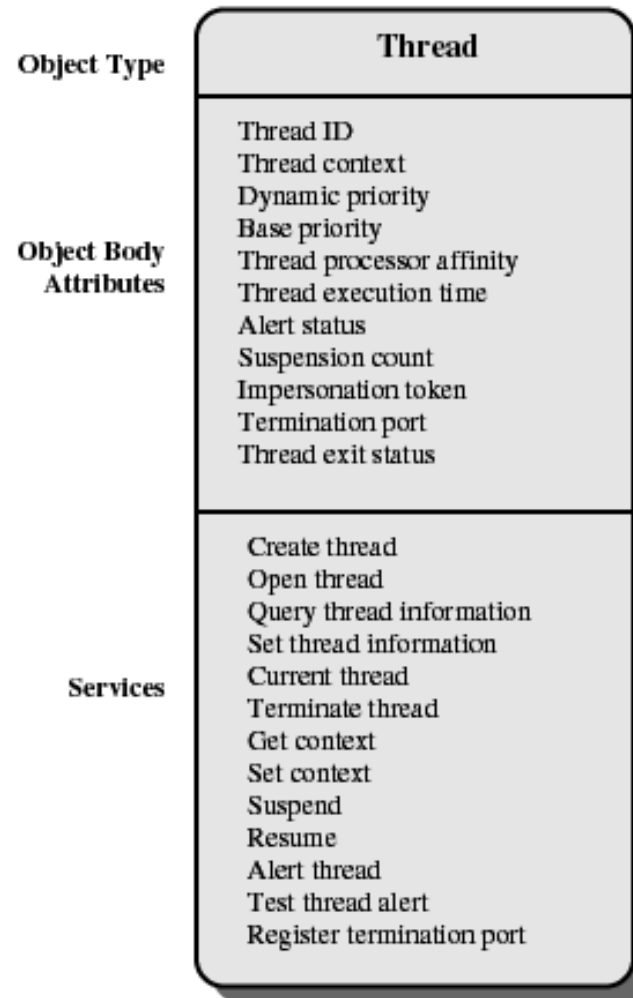
Figure 4.12 A Windows Process and Its Resources

Windows Process Object



(a) Process object

Windows Thread Object



(b) Thread object

Windows 2000 Thread States

- Ready
- Standby
- Running
- Waiting
- Transition
- Terminated

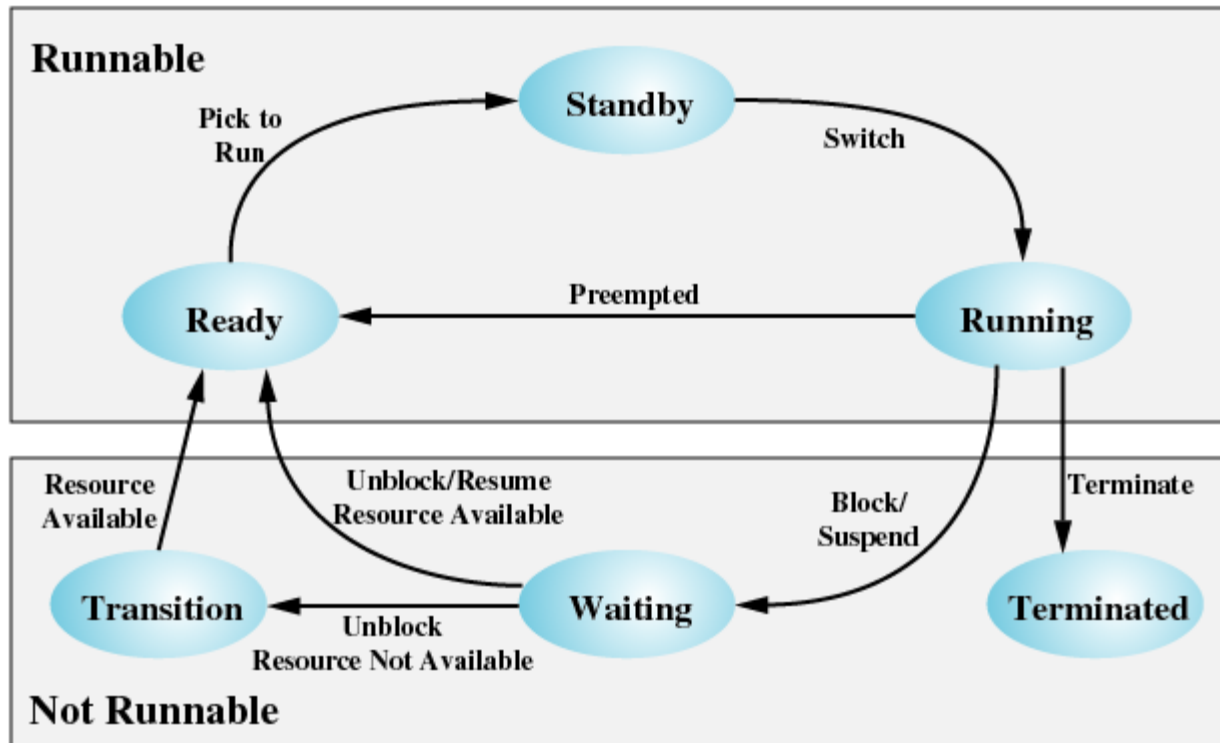


Figure 4.14 Windows Thread States

Solaris

- Process includes the user's address space, stack, and process control block
- User-level threads
- Lightweight processes (LWP)
- Kernel threads

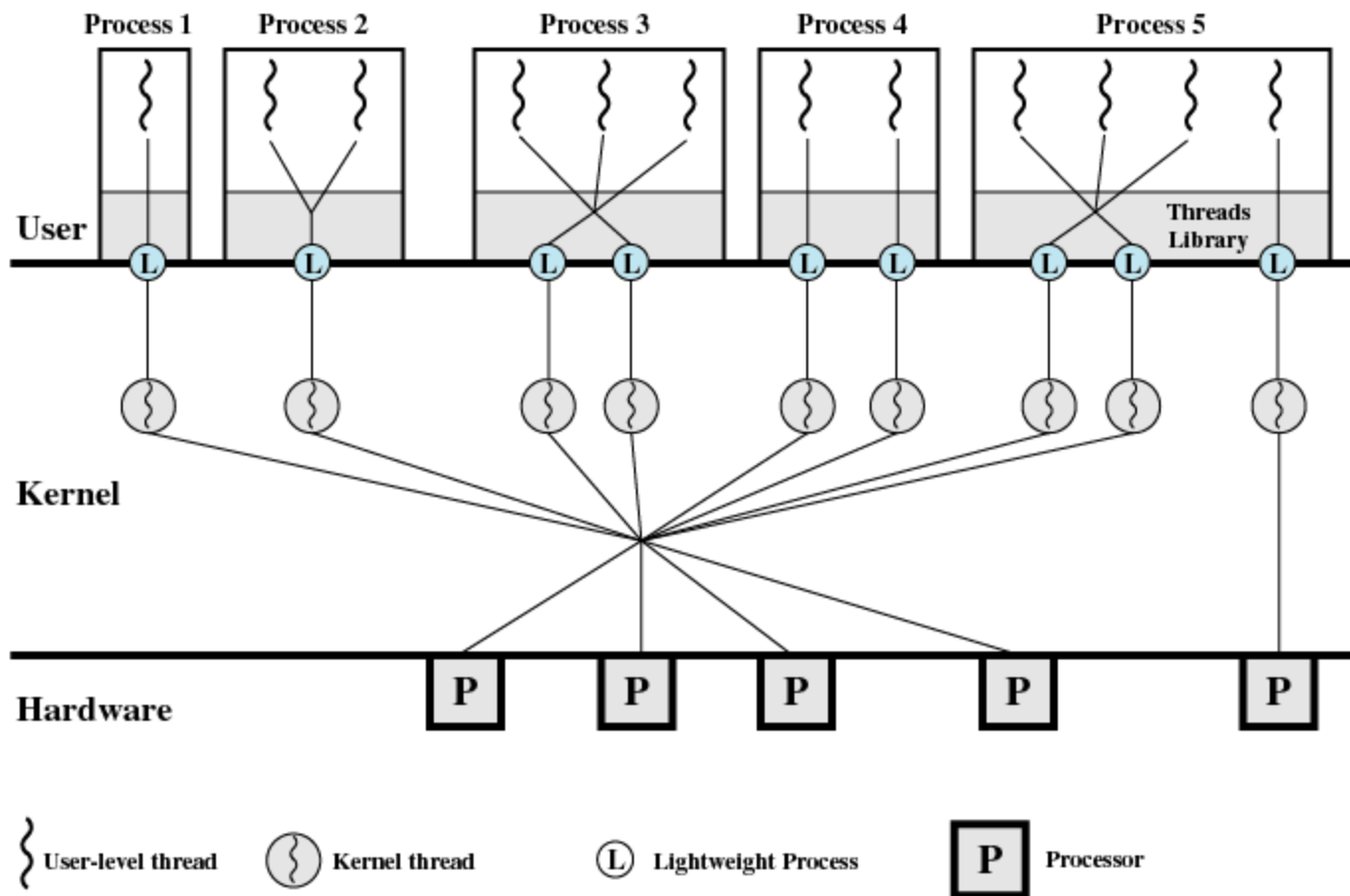
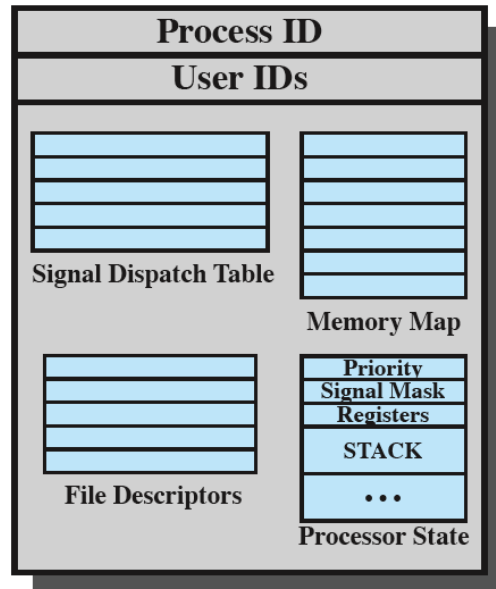


Figure 4.15 Solaris Multithreaded Architecture Example

UNIX Process Structure



Solaris Process Structure

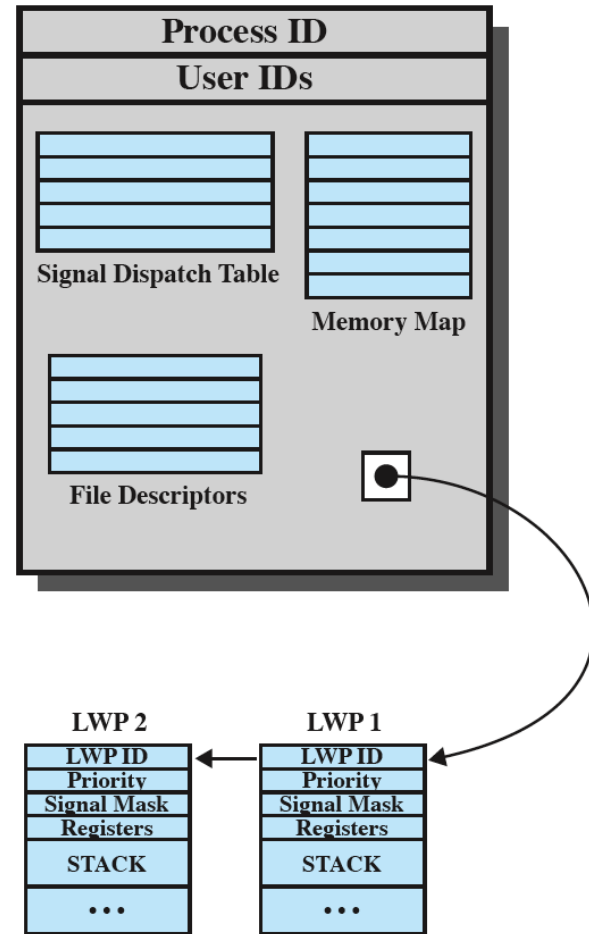


Figure 4.16 Process Structure in Traditional UNIX and Solaris [LEWI96]

Solaris Lightweight Data Structure

- Identifier
- Priority
- Signal mask
- Saved values of user-level registers
- Kernel stack
- Resource usage and profiling data
- Pointer to the corresponding kernel thread
- Pointer to the process structure

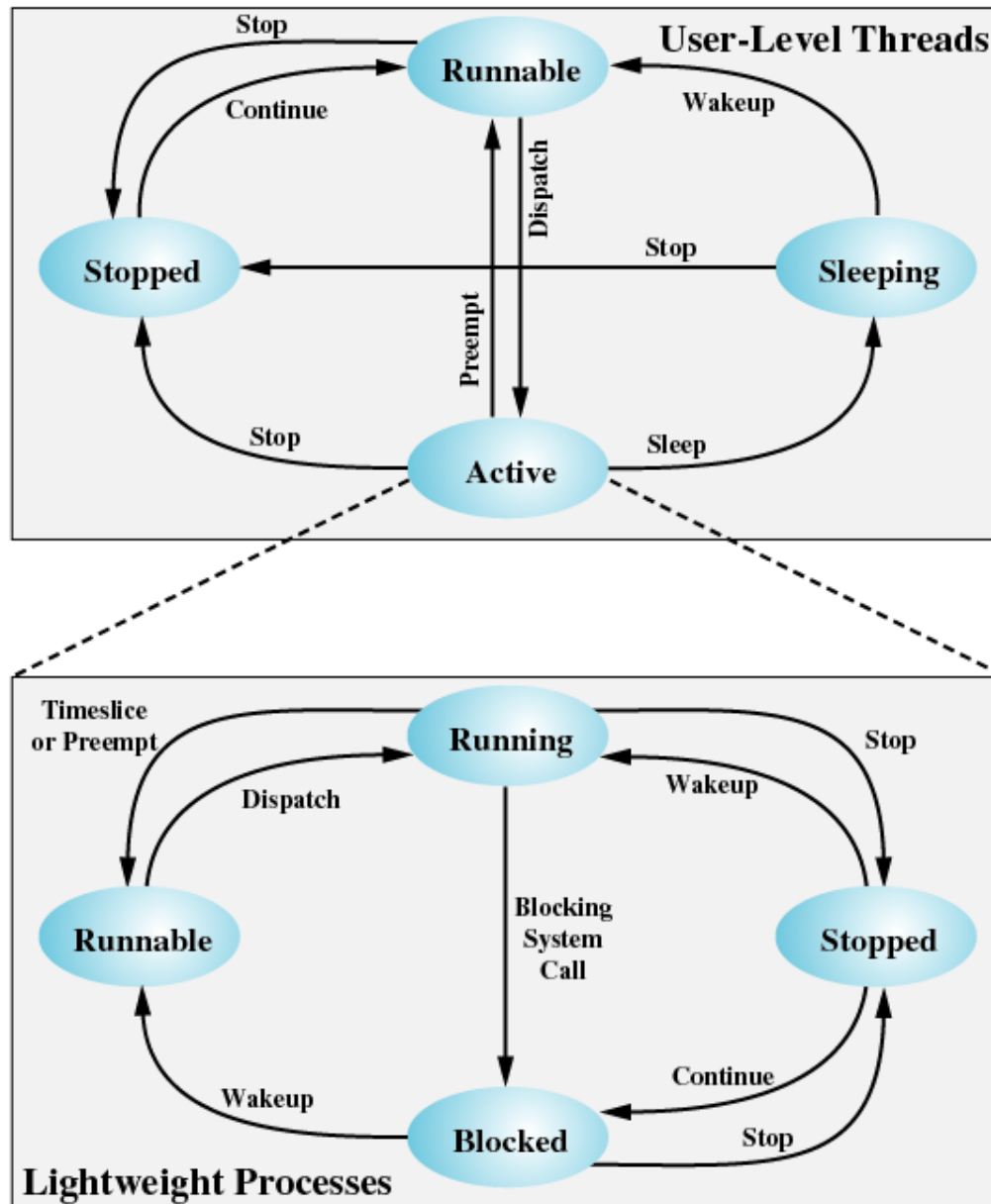


Figure 4.17 Solaris User-Level Thread and LWP States

Linux Task Data Structure

- State
- Scheduling information
 - normal or real-time, priorities
- Identifiers
- Interprocess communication
- Links
- Times and timers
- File system
- Address space
- Processor-specific context

Linux States of a Process

- Running
- Interruptable
- Uninterruptable
- Stopped
- Zombie

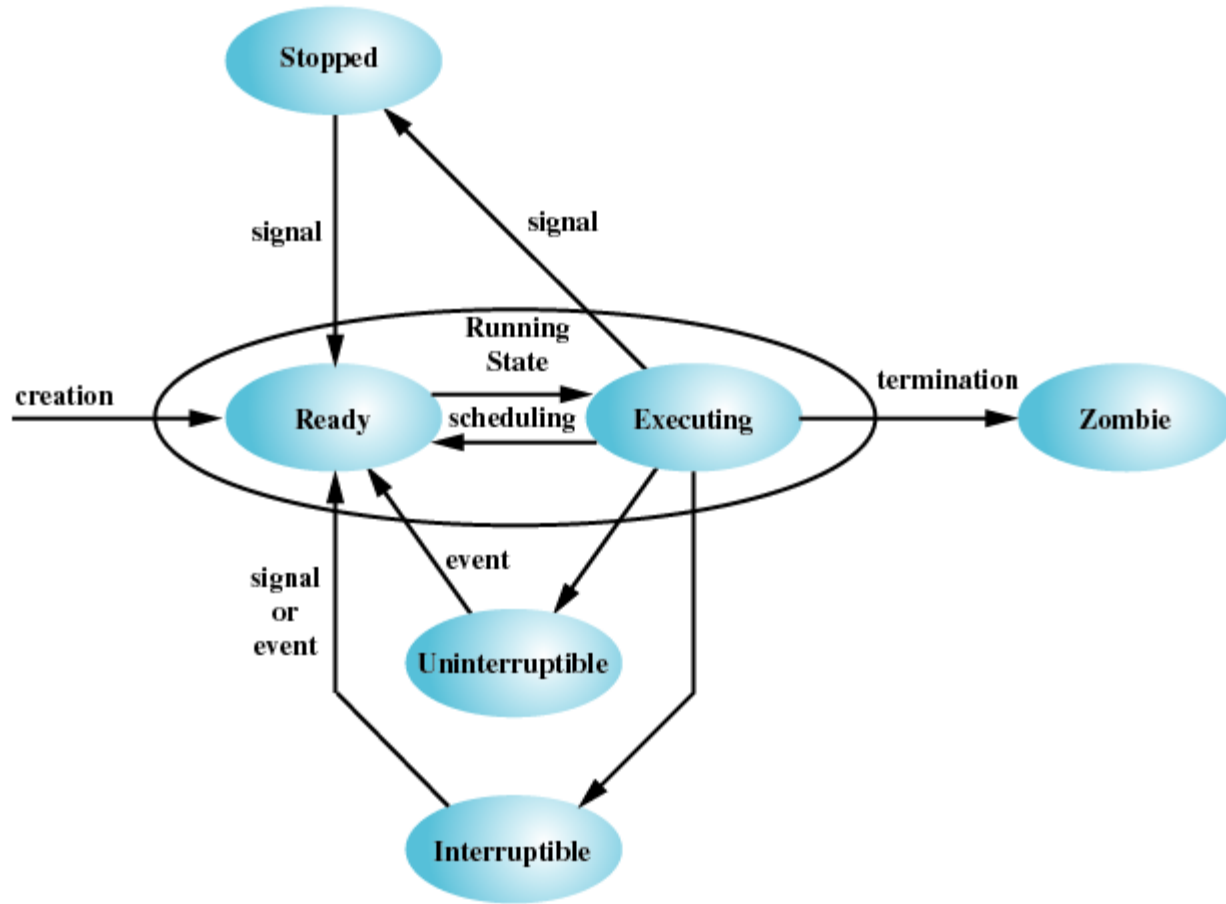


Figure 4.18 Linux Process/Thread Model