Processes and Threads

From OSDev Wiki

Contents

- 1 Processes/Tasks
- 2 Threads
- 3 Task Models
- 4 Scheduling
- 5 See Also

Processes/Tasks

Processes are running programs, including code, data, heap, and stack. In most implementations (but not always), each process has its own virtual address space (i.e. its own logical address-to-physical memory mapping) and its own set of system resources (files, environment variable, etc.). It is fairly common to for each process to have several threads (see below). This way there is a process to maintain the address space and several threads to control the process' execution.

Threads

A **thread** is a control flow in an executable image. Threads can be "user level" (i.e., the process handles multiple threads within itself) or "kernel level" (i.e., the OS scheduler handles multiple threads within a single process). Two threads from the same process naturally share the same code and global data but are given different stacks so that they do not interfere with each other's local variable and may have their own call chain.

Task Models

- Monotasking Systems
- Multitasking Systems
- Real-Time Systems
- Continuation Systems

Scheduling

- Context Switching
- Scheduling Algorithms
- Multiprocessor Scheduling
- Blocking Process

See Also

- IPC

- Synchronization

Retrieved from "http://wiki.osdev.org/index.php?title=Processes_and_Threads&oldid=10142"
Category: Processes and Threads
