Threads SMP Microkernels

DESKTOP-H800RKQ

February 13, 2024

Contents

1 Processes and Threads

Main characteristics of a process are:

- resource ownership
  - space to hold process info
  - control over resources
- scheduling/execution
  - process follows trace through program
  - has state and priority

1.1 Multithreading

Multithreading: ability of the OS to support multiple, concurrent paths of execution within a single process
In a multithreaded environment, a process has:

- virtual address space for process image (resource allocation)

- protected access to resources (protection)

Threads each have:

- state (running, ready, etc)
- saved thread context when not running
- execution stack
- static storage for local variables
- access to the memory and resources of process (shared with other threads)

When one thread alters memory, other threads see the results if and when they access that item. One thread's access permissions cascade to other threads in the same process.
Key benefits of threads:

1. faster to create thread in a process than a new process
2. faster to terminate a thread than a process
3. less time to switch between threads within a process than between processes
4. communication between threads more efficient

Threads are useful for:

1. splitting foreground/background work
2. asynchronous processing
3. speed of execution
4. modular program structure

Scheduling and dispatching is done on threads, not processes.

1.2 Thread Functionality

Key states for a thread are running, ready, and blocked (no suspend).
Four basic thread operations associated with a change in thread state:

1. **spawn**: caused by process spawn or another thread, then placed on ready queue (register context and stack space)
2. **block**: when waiting for an event
3. **unblock**: when event occurs, moved to ready queue
4. **finish**: thread completes and is deallocated

With multiple threads, blocked threads can wait simultaneously, for threads within the same process and in different processes.
Must synchronize the activity of various threads so that they do not interfere with each other.

2 Types of Threads

2.1 User-Level and Kernel-Level Threads

For pure user-level:

- all thread management is done by the

application and the kernel is not aware of them

- done using threads libraries that manage thread control and state