Systems Software

Assigment 2

Filippo Ghirardini

Freie Universität Berlin

ghira@zedat.fu-berlin.de

Threads

November 17, 2024

Overview

Process

What is a **process**?

- Independent execution with separate memory spaces
- Heavyweight due to memory and resource isolation
- Higher overhead for inter-process communication (IPC)

Thread

What is a **thread**?

- An independent sequence of instructions within a process
- Shares memory and resources with other threads in the same process
- Lightweight and efficient for communication
- Easier context switching compared to processes

Why threads?

- **Shared memory**: ideal for tasks requiring frequent communication or shared data
- **Lower overhead**: minimizes resource usage and context-switching costs
- **Fine-grained concurrency**: suitable for parallel tasks within a single application (e.g., GUI and computations)
- **Blocking I/O operations**: threads can perform blocking I/O without halting the entire application

Why processes?

- **Independency**
- **Information**: processes carry more state information than threads
- **Safety**: due to threads sharing the same address space, an illegal operation performed by a thread can crash the entire process

Definition

A **user** level thread is implemented by the user-level software and it's created and managed by the library through OS APIs.
A **kernel** level thread is implemented by the kernel itself and managed by it. It has it's own context: name, group and priority.

User-level advantages

- **Easy**: can be created and managed more rapidly
- **Portable**: can be implemented across different OS
- **Switch**: context switching can be done without going into kernel mode

Kernel-level advantages

- **Parallelism**: they allow real parallel execution in multi-core machines
- **Continuity**: different threads can run even if one is blocked
- **Resources**: they have direct access to kernel resources

Array

Advantages

- **Fast** random access
- **Efficient** memory usage when the # of threads is fixed

Disadvantages

- **Scalability** is limited by fixed size
- **Inefficient** if the array is sparse

List

Advantages

- **Flexibility** allows easy insertion and deletion

Disadvantages

- **Slow** compared to arrays
- **Memory overhead** due to the pointers

Tree

Advantages

- **Efficient** for priority or hierarchical organization
- **Fast** for insertion, deletion and search compared to lists

Disadvantages

- **Complex** to implement and mantain

Inverted table

Advantages

- **Efficient** for states and priority organization
- **Fast** for lookup and categorization

Disadvantages

- **Complex**
- **Overhead** due to the maintenance of consistency

TCB storage

Thread Control Blocks (TCBs) are typically stored within the operating system's **kernel address space**.
Storing TCBs in the kernel ensures that thread management is *secure*, *efficient*, and *isolated* from user-space processes. This centralized management allows the kernel to effectively handle thread scheduling, context switching, and synchronization.

Sources

- Operating System Concepts, Silberschatz, A., Galvin, P. B.
- Modern Operating Systems, Tanenbaum, A. S.