Lecture 4: Process Management – Part 3

Barnaby Martin

barnaby.d.martin@dur.ac.uk

Multiprogramming

The advantages of multiprogramming / implementing process co-operation may include:

- Computation speed-up: if there are multiple CPUs and/or cores.
- Convenience: single user wants to run many tasks.
- Information sharing: for example shared files.
- Modularity: programming (e.g. OOP)

Multiprogramming supports the co-operation of processes.

Multiprogramming

The aim of multiprogramming is to **maximize** CPU utilization.

With multiprogramming several processes are kept in memory concurrently.

When one process is **waiting**, the operating system executes (to **running**) one of the processes sitting in the **ready** queue.

CPU Scheduling is the method used to support multiprogramming in process management.

Multiprogramming - further considerations

Multiple CPUs, Cores, and Threading:

CPU - one, two, or more.

Cores vs. CPUs.

Threading / hyper-threading.

See Task Manager / Resource Monitor (in Windows).

Complexity.

Threads

A **thread** is a basic unit of CPU utilization (akin to a subset of a process).

A thread shares some attributes (e.g. priority) with its peer threads (the same process) but may execute different code.

Benefits of Using Threads

The benefit of using threading includes the following:

- **Responsiveness:** the process (therefore program) continues running even if part of it (i.e. a thread) is blocked or performing a lengthy operation.
- **Resource sharing:** threads share the resources of their common process.
- **Economy:** allocating memory and resources for process creation is costly.
- **Multiprocessor architectures:** a single-threaded process can only run on one processor.
- **Performance:** Cooperation of multiple threads in same job delivers higher throughput and improved performance.

Support for Threads

User Threads

Thread management in user programs is supported by the libraries:

- Java threads
- Win32 (Windows API) threads

Kernel Threads

Thread management in operating system programs is supported by a kernel module.

Multithreading Models

Many-to-One

- Many user application threads to one kernel thread.
- Programmer controls the number of application threads.

One-to-One

One user application thread to one kernel thread.

Many-to-Many

- N user application threads to $\leq N$ kernel threads.
- Defined by the capabilities of the operating system.

Example: Windows

Windows may schedule threads using a priority-based pre-emptive scheduling algorithm:

- Ensuring highest-priority thread always runs.

A thread selected by the dispatcher will run until:

- Pre-empted by a higher-priority thread.
- It terminates (completes or some other event causing it to terminate).
- The time quantum ends.
- A system call is made (privileged).

Windows Priority Scheme

The dispatcher in Windows uses a 32-level priority scheme:

- The **higher** the level, the **higher** the priority.

The priorities are divided into two main classes:

- The Variable Class:
  - Threads with priorities between 1 to 15
  - A thread's priority can be altered
- The Real-time Class:
  - Threads with priorities between 16 to 31
- Additionally, a thread running at priority of 0 is used in memory management.

Windows Priorities

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

Column: Process priority class

Row: Relative priority of thread in process

Windows Dispatcher

The dispatcher uses a queue for each scheduling priority.

The dispatcher traverses the set of queues looking for the highest priority thread to run.

When no thread is found in any of the queues the dispatcher will select the idle thread for execution.

Variable Class Threads

When a thread's time quantum runs out, the thread is interrupted and dispatched to the ready queue.

- Priority lowered: never below base priority.

When a thread is released from a wait operation it is dispatched to the ready queue.

- Priority is raised: based on what the thread was waiting for.