Thread Program Collection

Two Service Directories:

lib: A directory containing the Command Line parsing library

Number of CPUs: A directory containing a program that gives the number of CPUs as a result

Seven Program Directories containing the following five programs:

1. ASimpleThread: A program that creates and initializes five Threads and run them concurrently.
2. SingleThreadedServer: A computing intensive program that does not use threads. It just executes the task sequentially
3. ThreadPerTaskServer: A computing intensive program that uses an unlimited number of threads
4. ThreadPool: A computing intensive program that uses a configurable thread pool
5. ThreadPoolImplementation: A program showing the way a Thread Pool works.
6. SocketWithThread: A program to test the combination of sockets and threads.
7. SocketWithThreadAndPool: A program to test the combination of sockets, threads and thread pools.

Path to navigate through the proposed programs.

1. ASimpleThread shows what a Thread is, how to activate and run Threads
2. SingleThreadServer demonstrates the execution of a workload made up of a set of tasks, each consisting of a processing part and of a sleeping part. The balance between processing and sleeping can be configured. The program shows that the global load deriving from the execution of multiple instances of the same task grows up proportionally with respect to the number of tasks.
3. ThreadperTaskServer deals with the case in which the tasks are distributed over distinct threads. In principle, the distribution of processing-intensive tasks on distinct threads does not cause a growth of the global execution time as the different threads execute on different CPUs. The case of sleeping-intensive tasks is particularly evident. The total execution time is not a function of the number of tasks to be executed. Overlap is almost perfect.
4. ThreadPool deals with the case in which the user wants to control the number of threads. Instead of generating a new thread when needed, which would take resources, a fixed number of threads are activated at start time and dynamically allocated to the tasks when free. The saving derives from the absence of thread generation, from the reduced memory size and from the more efficient scheduling. The drawback is that the tasks are carried out in parallel bursts.
5. ThreadPoolImplementation shows the way a Thread Pool is implemented. In particular it shows that a Thread Pool is not a “magic” operating system based solution, whereas on the contrary it relies on a queue data structure (BlockingQueue) that blocks both processes that try to enqueue elements when the queue is full and processes that try to dequeue elements when the queue is empty.
6. SocketwithThreads shows the case in which a server is listening to a socket, blocked on accept. Upon reception of a connection request the server creates a thread to serve the request. That allows increasing the capability of servers to serve large numbers of requests. Of course, the global performance is limited.
7. SocketWithThreadsandPool shows the case in which the server does not wildly create threads to serve requests. On the contrary it assigns requests to a number of threads belonging to a pool.