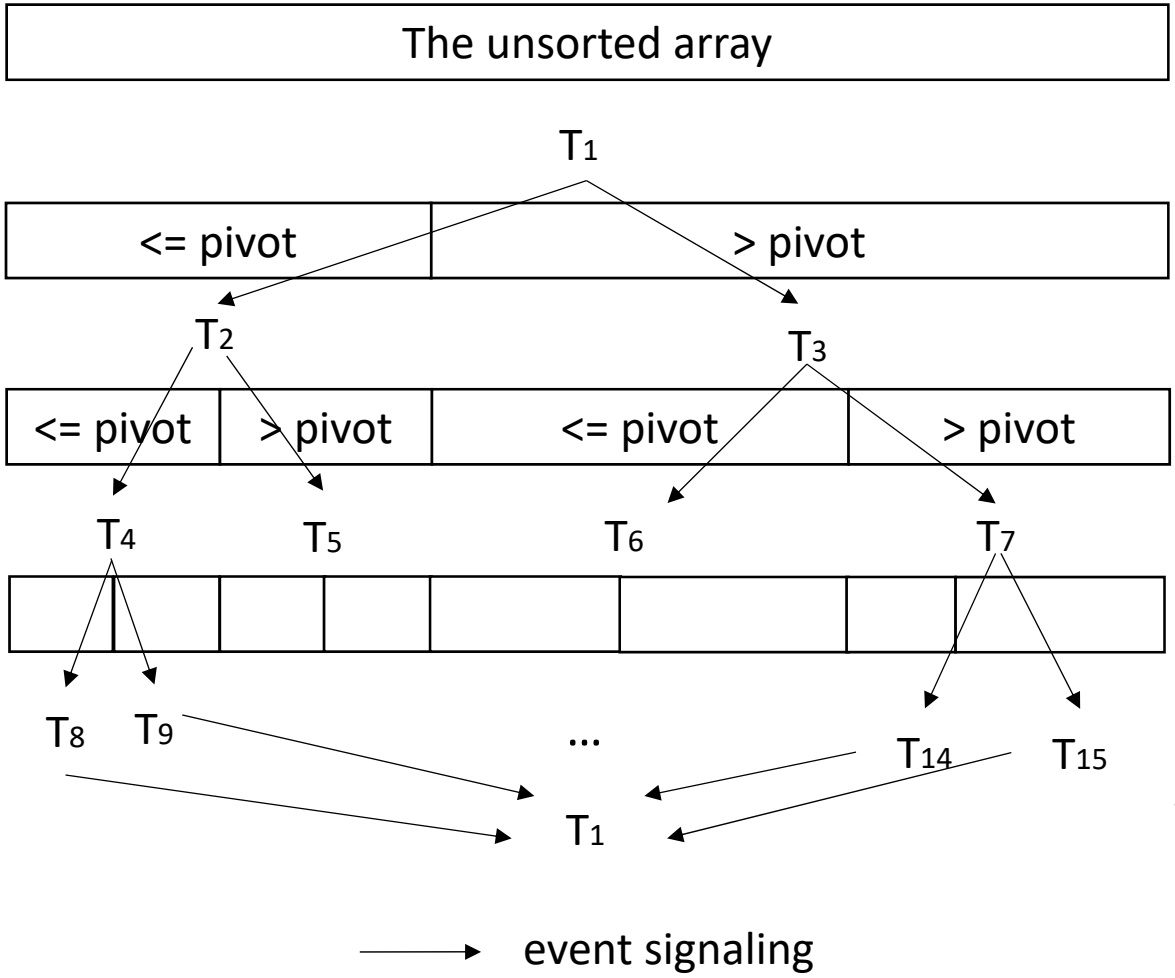
Operating Systems
Programming Assignment #3

Parallel Quicksort using Pthread

Porf. Li-Pin Chang

National Chiao-Tung University

Parallel Quicksort



T1: the master thread

T1: partitions array and signals (via semaphores) T2 and T3

T2: partitions array and signals T4 and T5

T4 : partitions array and signals T8 and T9

T8: sorts the array and signals T1

T1 reports completion when signaled by all the 4^{th}-level threads

APIs

- <pthread.h>

  Thread management

  - Pthread_create, pthread_exit
  - Do not use pthread_join, use semaphore instead.

- <semaphore.h>

  Semaphore operations

  - sem_init, sem_wait, sem_post, sem_getvalue, sem_destroy

Requirements

1. Prompt for the name of the input file
2. Read integers from the file
3. Do the sorting
4. Print the execution time of multi-thread sorting and single-thread sorting
   - MT sorting should be much faster than ST sorting
   - Their results must be exactly the same
5. Write the sorted array to a file
   - output1.txt → MT sorting
   - output2.txt → ST sorting

Requirements

- The cooperation among threads must be exactly the same as shown in the figure
- Create all threads in the beginning of your program
  - All created threads wait on their own semaphore (T1~T15) until they are signaled
  - The main program signals the master thread T1 to start
  - The master thread T1 signals the worker threads T2 and T3 and then waits until it has been signaled by all the bottom-level worker threads (T8~T15)
  - A worker thread waits until being signaled by its predecessor, does its job, and then signals its successor
- Fail to comply with this requirement will incur a score penalty
- Use bubble sort at the bottom level (T8~T15)

Input/output format

- Input file format:

<# of elements of array><space>\n

<all elements separated by space>

  - Largest input: 1,000,000 integers

- Output file format:

<sorted array elements separated by space>

Testing OS Environment

- Ubuntu 16.04, Ubuntu 14.04 or CS linux work station
  - Your code should compile successfully in one of the above environments