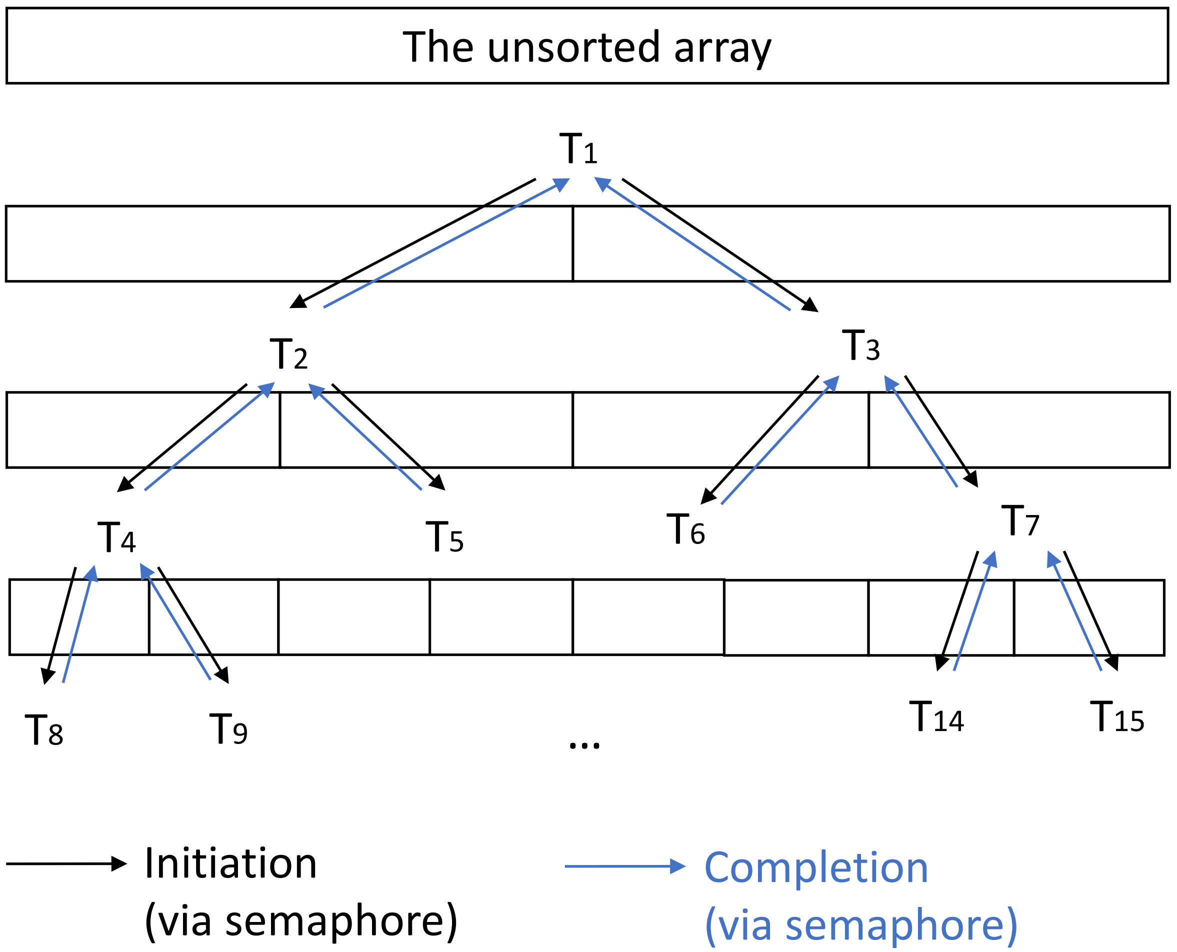
Operating Systems Programming Assignment #3

Parallel Merge Sort using Pthread

Prof. Li-Pin Chang

National Chiao-Tung University

Parallel Merge Sort



T1: the master thread

1. Divides the array into two equal sub-arrays
2. Signals T2 and T3 (via semaphores) to sort the two sub-arrays
3. Waits on T2 and T3 (via semaphores)
4. Merges the two sorted sub-arrays
5. Generate an output file

T8~T15:

1. Do bubble sort on their own sub-arrays
2. Signal their upper-level threads (via semaphores)

APIs

- <pthread.h>

  Thread management

  - Pthread_create, pthread_exit

- <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
  - Each of T1~T15 waits on its own semaphore
  - The main program signals the master thread T1 to start
  - T1 signals the 2nd-level threads T2 and T3 to start, and so on
  - Lower-level threads notify upper-level threads via semaphores; do not use pthread_join()
- Use Bubble sort at the bottom level (T8~T15)
  - For observation of speed-up

Requirements

- Single-thread sorting
  - Use one single thread to do the same sorting, but no thread parallelism
  - 3 levels of array partitioning, bubble sort at the bottom level, and merge sub-arrays on return
  - Should be noticeably slower than the multithreaded version

- Fail to comply with the requirements will incur a score penalty
- You get 0 point if you call qsort() at any place in your program

Input/output format

- Input file format:

<total # of integers><space>\n

<all integers separated by space>

    - Largest input: 1,000,000 integers
    - Generate your own file for testing

- Output file format:

<sorted integers separated by space>

Header of your .c or .cpp

```
/*
Student No.: <your student id>
Student Name: <your name>
Email: <your email>
SE tag: xnxcxtxuxoxsx
Statement: I am fully aware that this program is not
supposed to be posted to a public server, such as a
public GitHub repository or a public web page.
*/
```

Testing OS Environment

- Ubuntu 18.04
- Install as a VM or on a physical machine