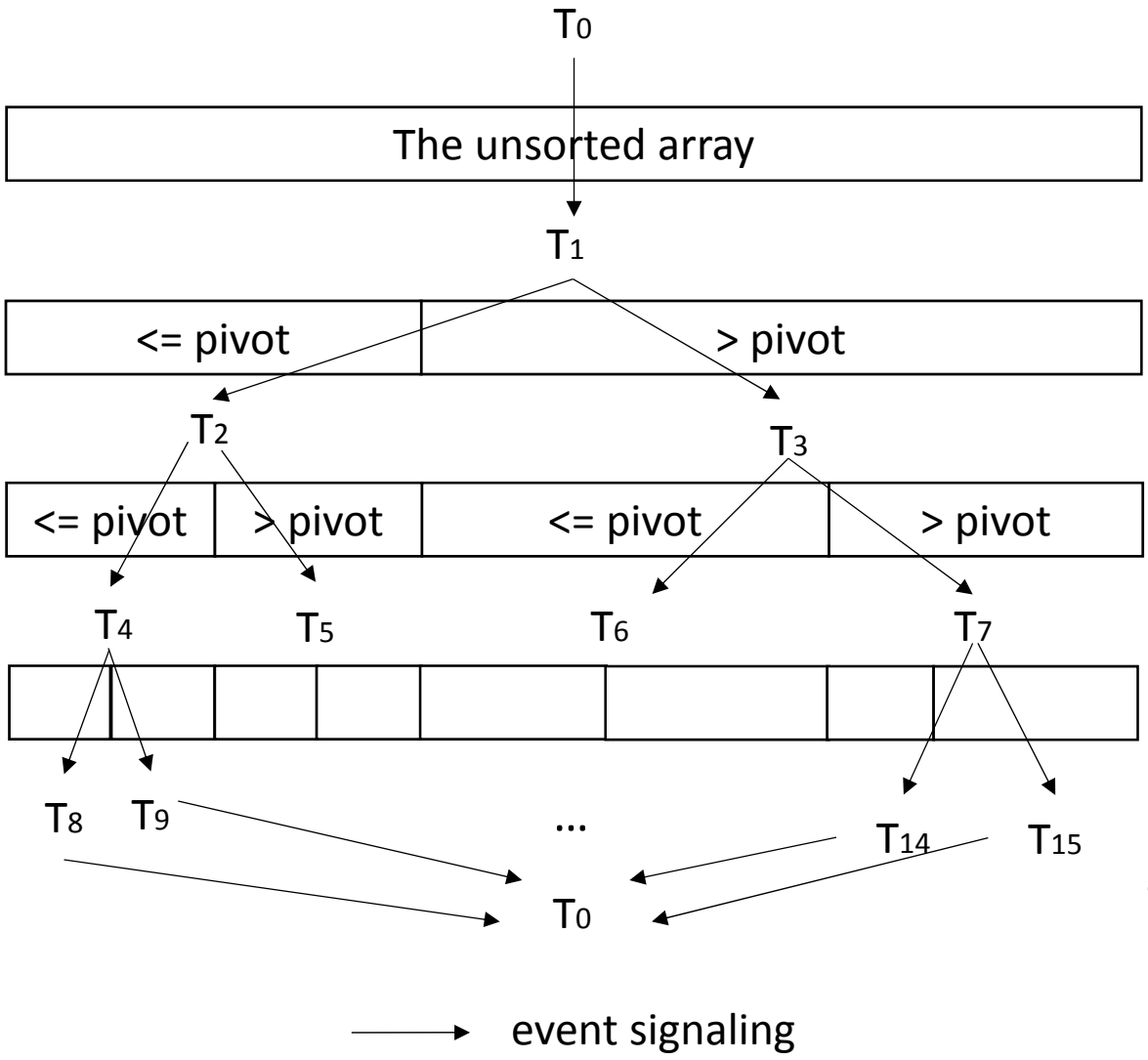
# Operating Systems
# Programming Assignment #3
# Parallel Quicksort

# Parallel Quicksort




T0: the mother task

T1: partitions array and creates T2 and T3

T2: partitions array and creates T4 and T5

T4 : partitions array and creates T8 and T9

T8: sorts the array and signals T0 via a semaphore

T0 reports done when signaled by all the $4^{th}$-level tasks

# 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
  - Each child thread waits on a semaphore until it is signaled (T1~T15)
  - The mother thread waits until she has been signaled by all the bottom-level threads (T8~T15)
  - Stupid, but please comply with this specification
- Sorting at the bottom level can be brute-force methods, e.g., bubble sort

# 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>