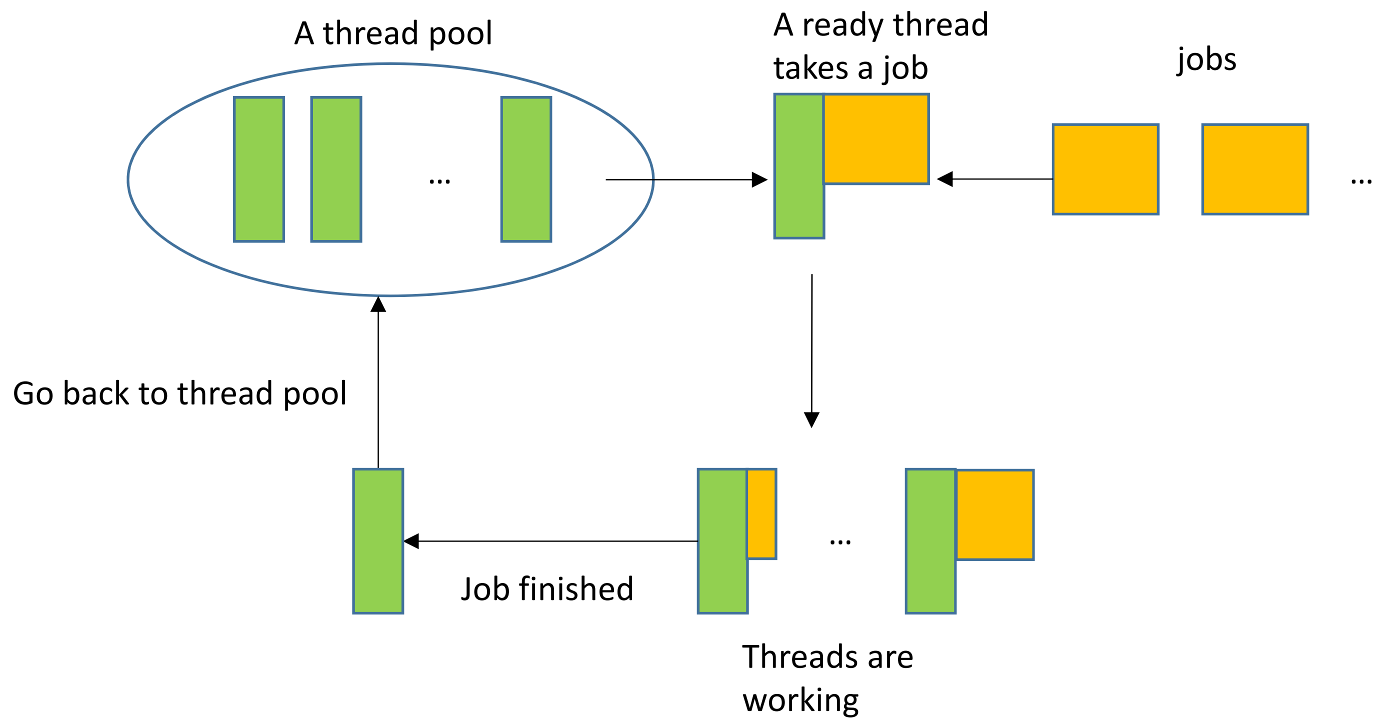
# Programming Assignment #4: Quick Sort with a Thread Pool

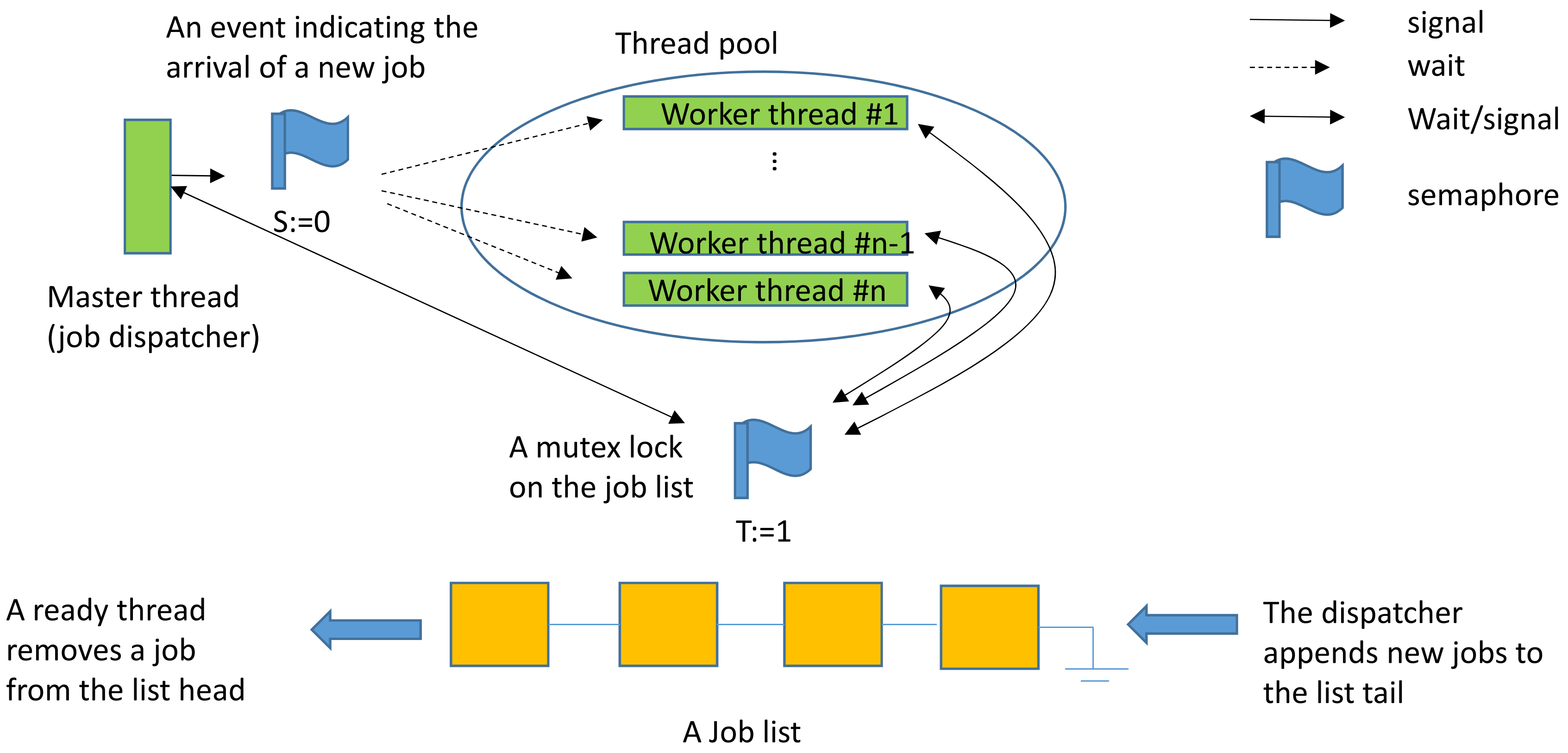
## Objective

- Multithreaded sorting using a thread pool
  - # of threads in the pool determines the max. degree of parallelism
- The problem definition is the same as that in the previous assignment, except that the binding of sorting jobs to threads is dynamic
- A job performs one of the following
  - Partitioning a large array into two small arrays
    - Left array < pivot < right array
  - Sorting a bottom-level array
    - Bubble sort, insertion sort, etc.

# The Concept of a Thread Pool



# A Reference Implementation



An event indicating the arrival of a new job
Thread pool
signal
wait
Wait/signal
semaphore
Worker thread #1
⋮
Worker thread #n-1
Worker thread #n
S:=0
Master thread (job dispatcher)
A mutex lock on the job list
T:=1
A ready thread removes a job from the list head
The dispatcher appends new jobs to the list tail
A Job list

## Procedure

1. Read data from the input file “input.txt”
2. n=1
3. Do the sorting with a thread pool of n threads
4. Print the execution time
5. Write the sorted array to a file
    - Filename: output_n.txt (e.g., output_3.txt if n=3)
6. n++; if n<=8 then goto 3

## Remarks

- Reuse your assignment 3
- The binding of jobs to threads must be dynamic
- All the 8 output files must be identical
- Execution time decreases as *n* increases
- Performance improvement saturates as *n* increases

## Input/Output Format

- Format of “input.txt”:

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

<all elements separated by space>

  - Largest input: the same as in assignment #3

- Output file format:

<sorted array elements separated by space>