OS HW3
Multi-Threading Programming

Operating System 108 Fall

Professor: W.J. TSAI

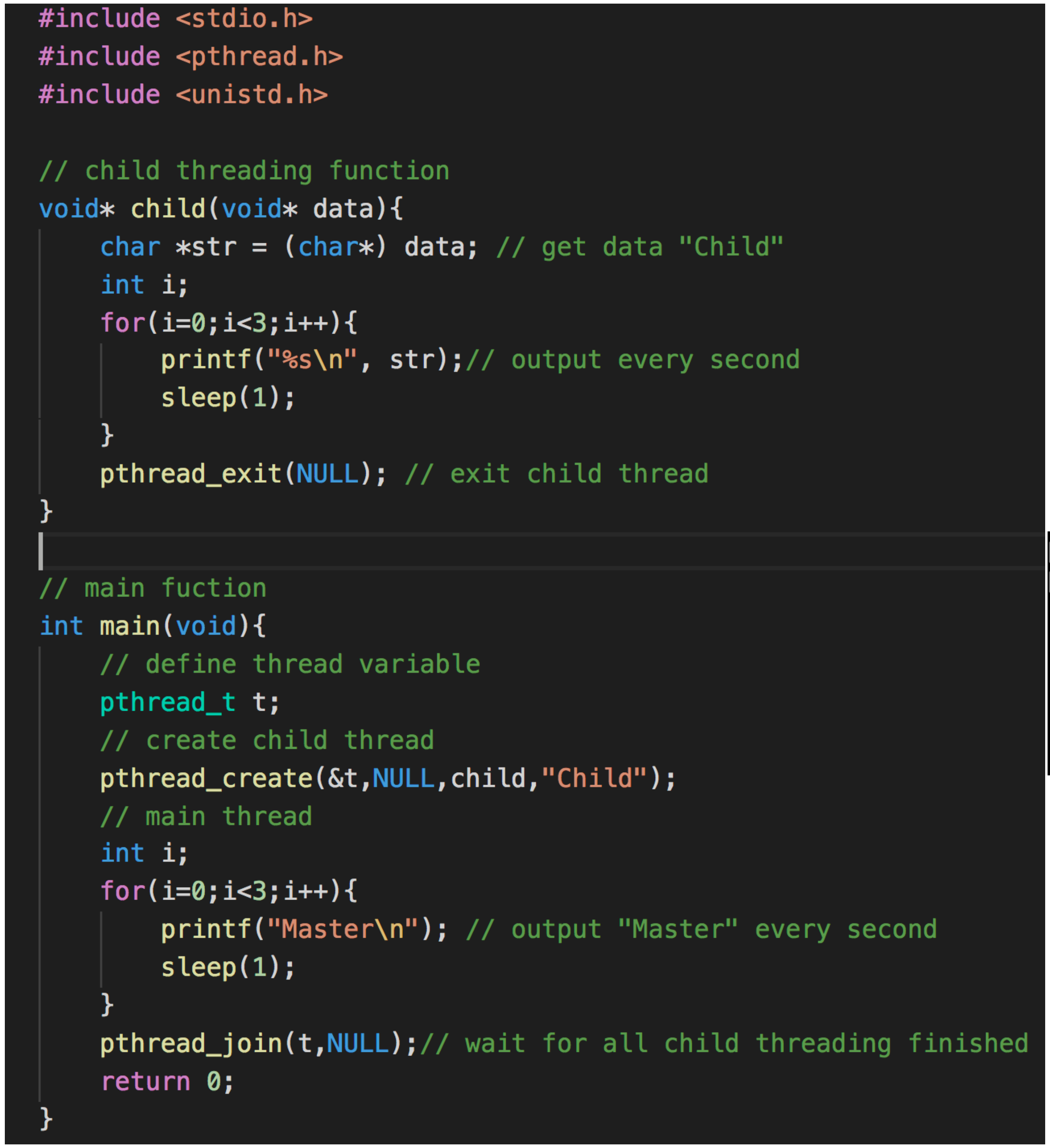
APIs

- Thread management: <pthread.h>
  - pthread_create
  - pthread_join
  - pthread_exit

- Reference: https://computing.llnl.gov/tutorials/pthreads/

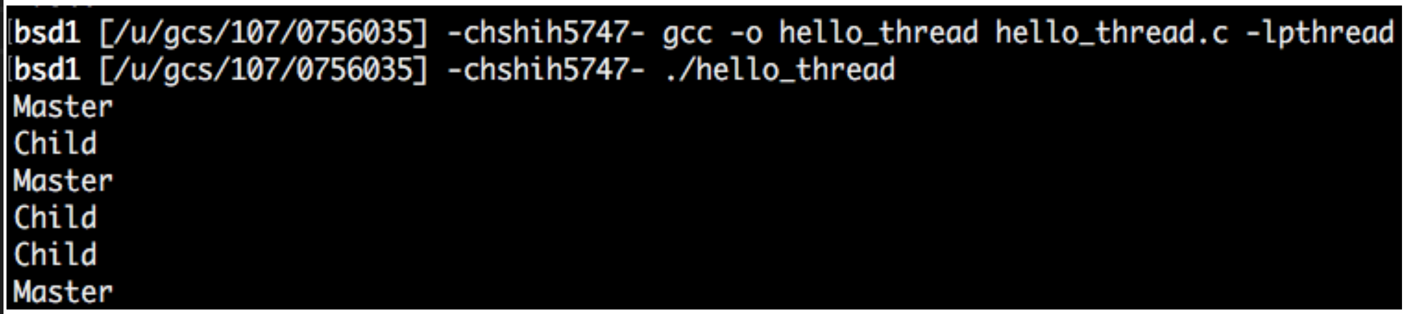
Example - Hello Thread

hello_thread.c



```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

// child threading function
void* child(void* data){
    char *str = (char*) data; // get data "Child"
    int i;
    for(i=0;i<3;i++){
        printf("%s\n", str);// output every second
        sleep(1);
    }
    pthread_exit(NULL); // exit child thread
}

// main fuction
int main(void){
    // define thread variable
    pthread_t t;
    // create child thread
    pthread_create(&t,NULL,child,"Child");
    // main thread
    int i;
    for(i=0;i<3;i++){
        printf("Master\n"); // output "Master" every second
        sleep(1);
    }
    pthread_join(t,NULL);// wait for all child threading finished
    return 0;
}
```

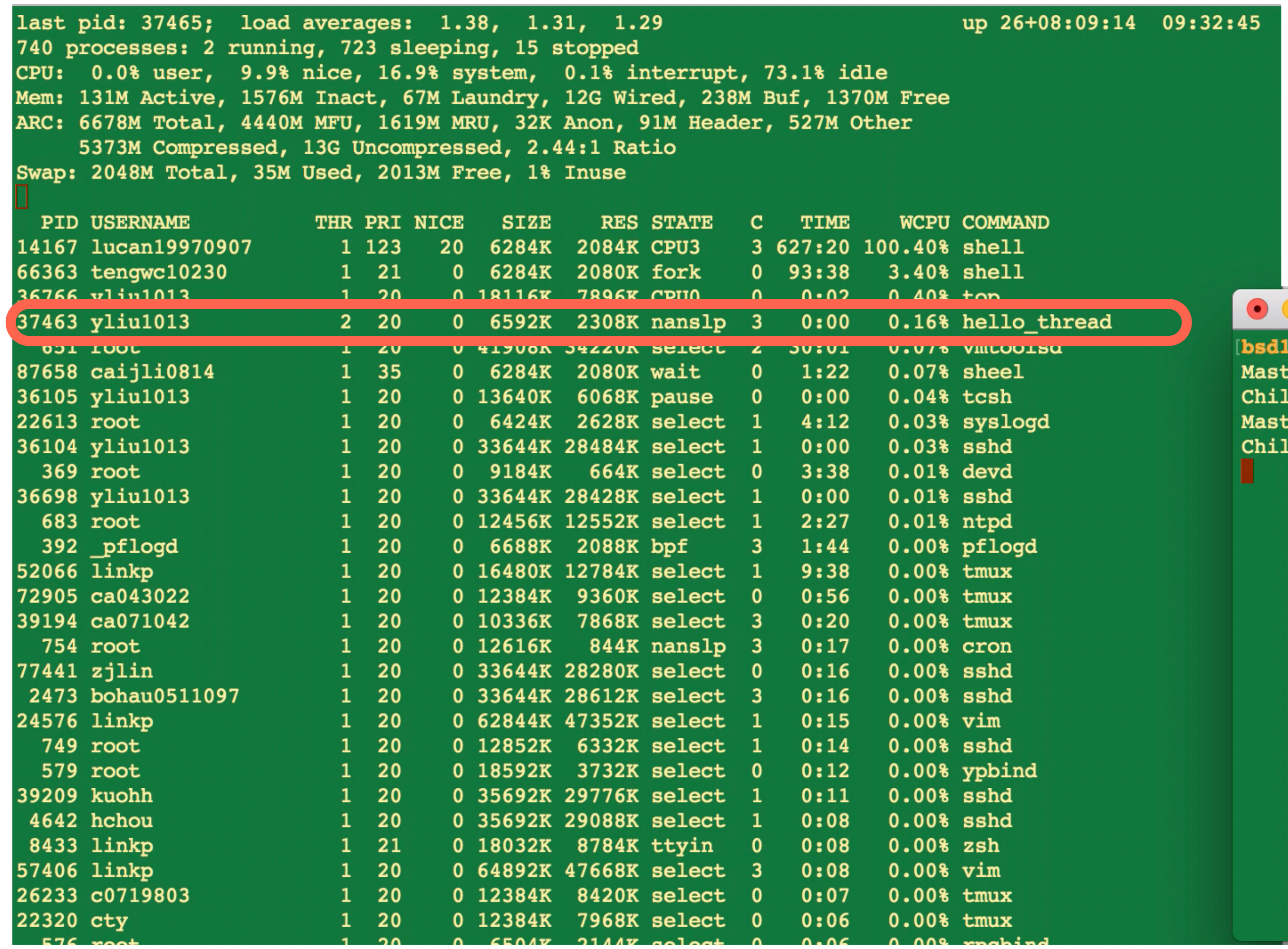
Output



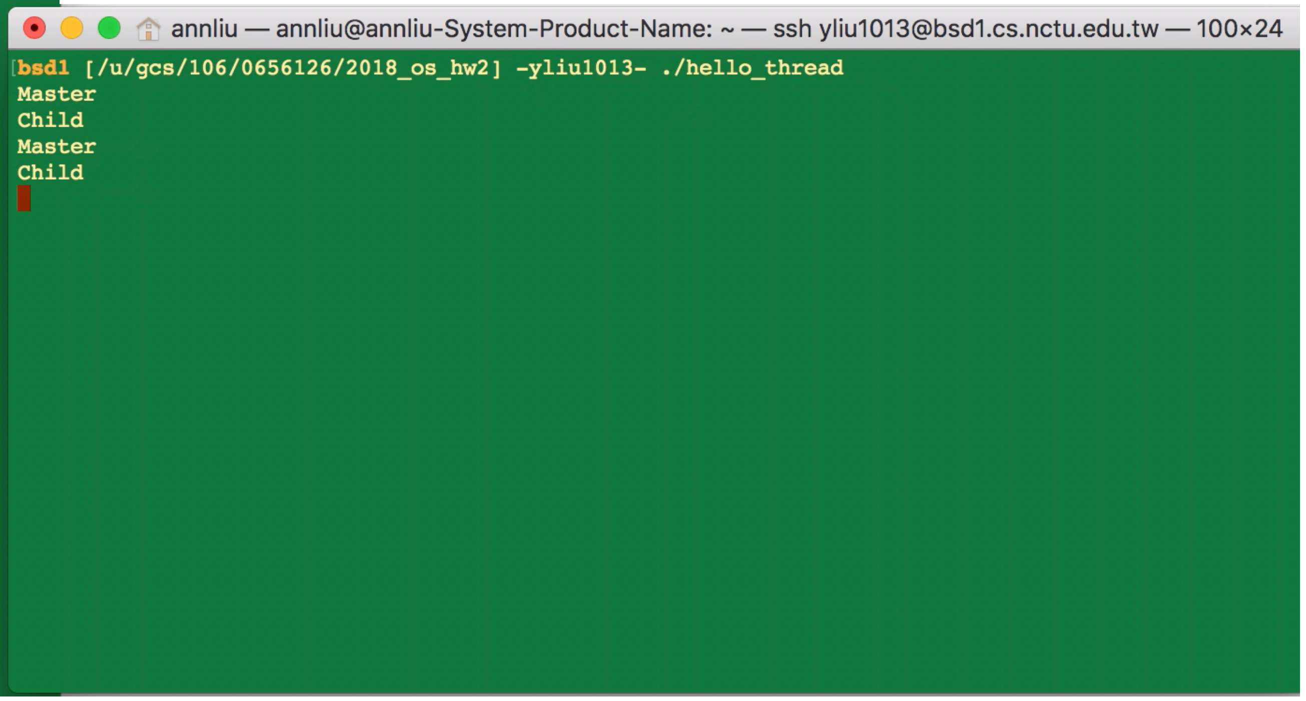
```
bsd1 [/u/gcs/107/0756035] -chshih5747- gcc -o hello_thread hello_thread.c -lpthread
bsd1 [/u/gcs/107/0756035] -chshih5747- ./hello_thread
Master
Child
Master
Child
Child
Master
```

Tools for showing thread

- **NCTU Workstation** > top
  (THR means the num of threads under the process)



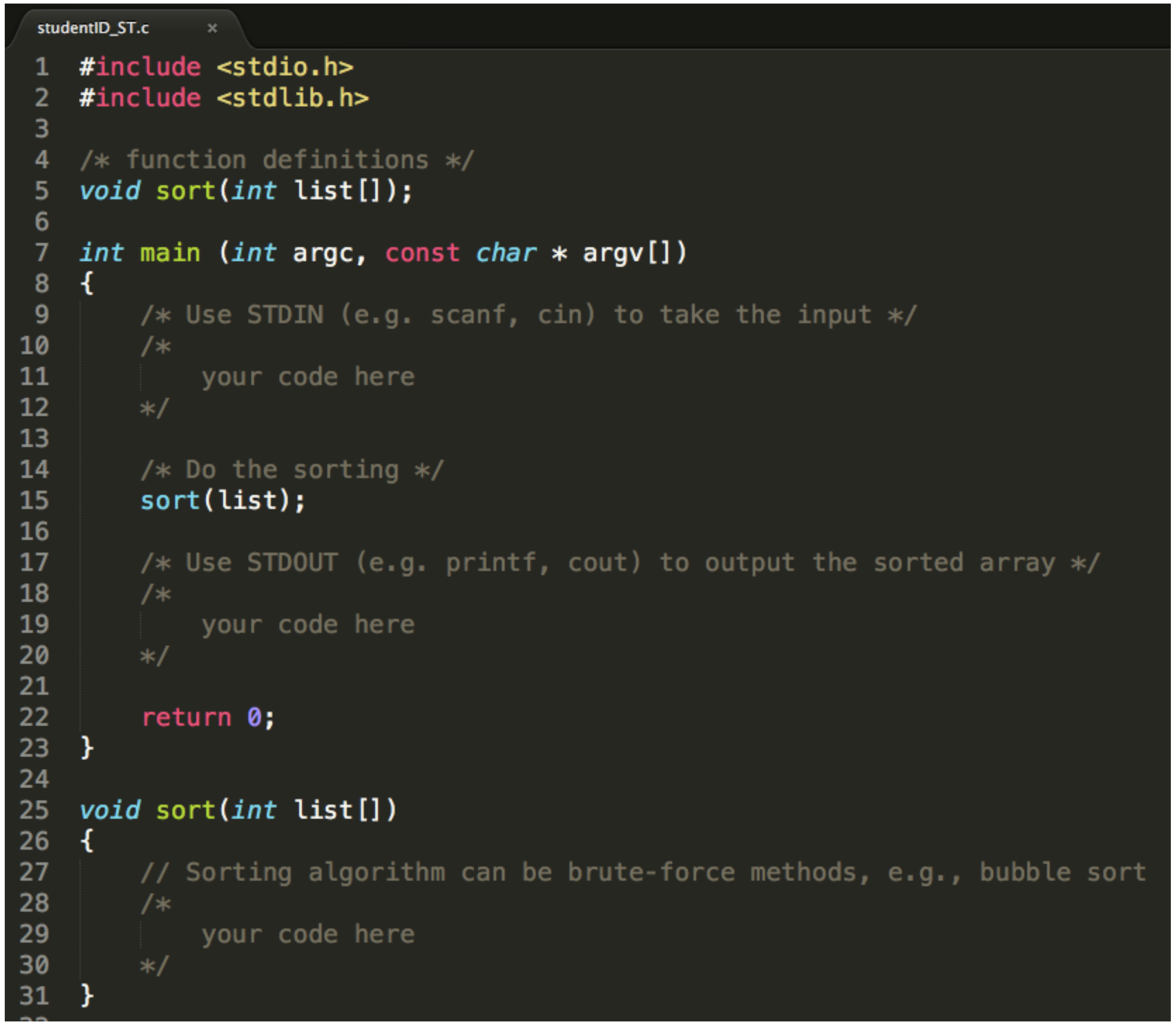
```
last pid: 37465;  load averages:  1.38,  1.31,  1.29                up 26+08:09:14  09:32:45
740 processes: 2 running, 723 sleeping, 15 stopped
CPU:  0.0% user,  9.9% nice, 16.9% system,  0.1% interrupt, 73.1% idle
Mem: 131M Active, 1576M Inact, 67M Laundry, 12G Wired, 238M Buf, 1370M Free
ARC: 6678M Total, 4440M MFU, 1619M MRU, 32K Anon, 91M Header, 527M Other
     5373M Compressed, 13G Uncompressed, 2.44:1 Ratio
Swap: 2048M Total, 35M Used, 2013M Free, 1% Inuse

  PID USERNAME         THR PRI NICE   SIZE    RES STATE   C   TIME    WCPU COMMAND
14167 lucan19970907      1 123   20  6284K  2084K CPU3    3 627:20 100.40% shell
66363 tengwc10230        1  21    0  6284K  2080K fork    0  93:38   3.40% shell
36766 yliu1013           1  20    0 18116K  7896K CPU0    0   0:02   0.40% top
37463 yliu1013           2  20    0  6592K  2308K nanslp  3   0:00   0.16% hello_thread
  651 root               1  20    0 41908K 34220K select  2  30:01   0.07% vmtoolsd
87658 caijli0814         1  35    0  6284K  2080K wait    0   1:22   0.07% sheel
36105 yliu1013           1  20    0 13640K  6068K pause   0   0:00   0.04% tcsh
22613 root               1  20    0  6424K  2628K select  1   4:12   0.03% syslogd
36104 yliu1013           1  20    0 33644K 28484K select  1   0:00   0.03% sshd
  369 root               1  20    0  9184K   664K select  0   3:38   0.01% devd
36698 yliu1013           1  20    0 33644K 28428K select  1   0:00   0.01% sshd
  683 root               1  20    0 12456K 12552K select  1   2:27   0.01% ntpd
  392 _pflogd            1  20    0  6688K  2088K bpf     3   1:44   0.00% pflogd
52066 linkp              1  20    0 16480K 12784K select  1   9:38   0.00% tmux
72905 ca043022           1  20    0 12384K  9360K select  0   0:56   0.00% tmux
39194 ca071042           1  20    0 10336K  7868K select  3   0:20   0.00% tmux
  754 root               1  20    0 12616K   844K nanslp  3   0:17   0.00% cron
77441 zjlin              1  20    0 33644K 28280K select  0   0:16   0.00% sshd
 2473 bohau0511097       1  20    0 33644K 28612K select  3   0:16   0.00% sshd
24576 linkp              1  20    0 62844K 47352K select  1   0:15   0.00% vim
  749 root               1  20    0 12852K  6332K select  1   0:14   0.00% sshd
  579 root               1  20    0 18592K  3732K select  0   0:12   0.00% ypbind
39209 kuohh              1  20    0 35692K 29776K select  1   0:11   0.00% sshd
 4642 hchou              1  20    0 35692K 29088K select  1   0:08   0.00% sshd
 8433 linkp              1  21    0 18032K  8784K ttyin   0   0:08   0.00% zsh
57406 linkp              1  20    0 64892K 47668K select  3   0:08   0.00% vim
26233 c0719803           1  20    0 12384K  8420K select  0   0:07   0.00% tmux
22320 cty                1  20    0 12384K  7968K select  0   0:06   0.00% tmux
  576 root               1  20    0  6504K  2144K select  0   0:06   0.00% rpcbind
```



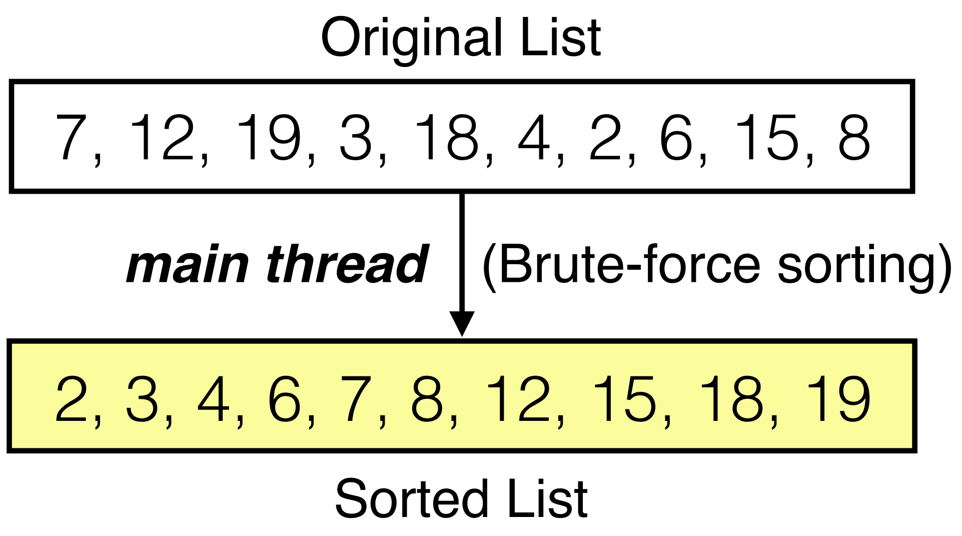
annliu — annliu@annliu-System-Product-Name: ~ — ssh yliu1013@bsd1.cs.nctu.edu.tw — 100×24

```
bsd1 [/u/gcs/106/0656126/2018_os_hw2] -yliu1013- ./hello_thread
Master
Child
Master
Child
```

Single-threaded Sorting


studentID_ST.c

```c
#include <stdio.h>
#include <stdlib.h>

/* function definitions */
void sort(int list[]);

int main (int argc, const char * argv[])
{
    /* Use STDIN (e.g. scanf, cin) to take the input */
    /*
        your code here
    */

    /* Do the sorting */
    sort(list);

    /* Use STDOUT (e.g. printf, cout) to output the sorted array */
    /*
        your code here
    */

    return 0;
}

void sort(int list[])
{
    // Sorting algorithm can be brute-force methods, e.g., bubble sort
    /*
        your code here
    */
}
```

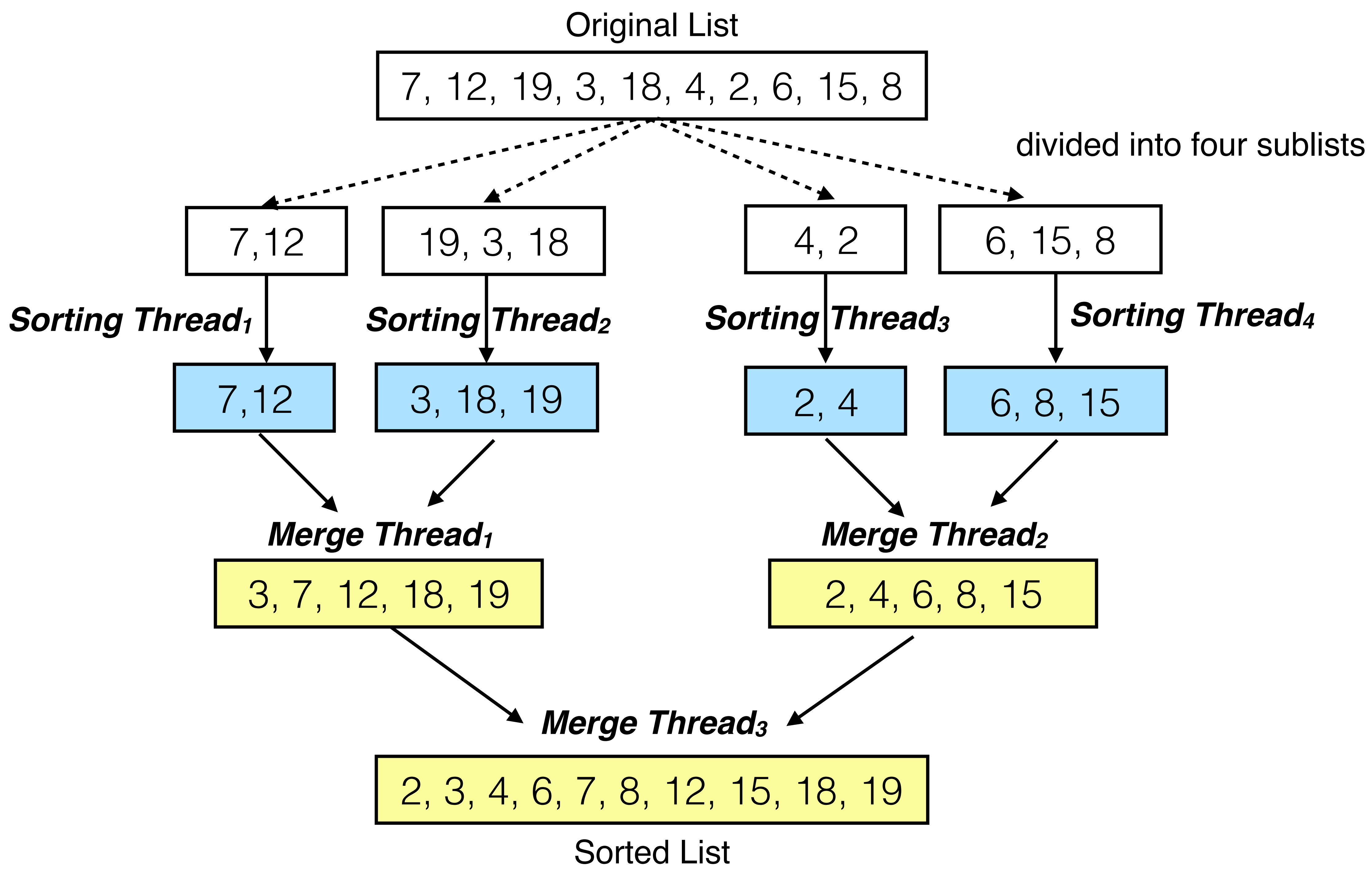


- You should implement:

1. **STDIN (e.g. scanf, cin)**
2. **Sort function**
   (use brute-force methods, e.g., bubble sort)
3. **STDOUT (e.g. printf, cout)**

- **DO NOT USE FILE I/O !**

Multithreaded Sorting



Multithreaded Sorting

- A list of integers is divided into four smaller lists.
- Four separate threads (***sorting thread$_1$* ~ *sorting thread$_4$***) sort each sublist using any brute-force methods (e.g., bubble sort).
- The four sublists are then merged into a single sorted list by three threads (***merging thread$_1$* ~ *merging thread$_3$***).
- You should implement:

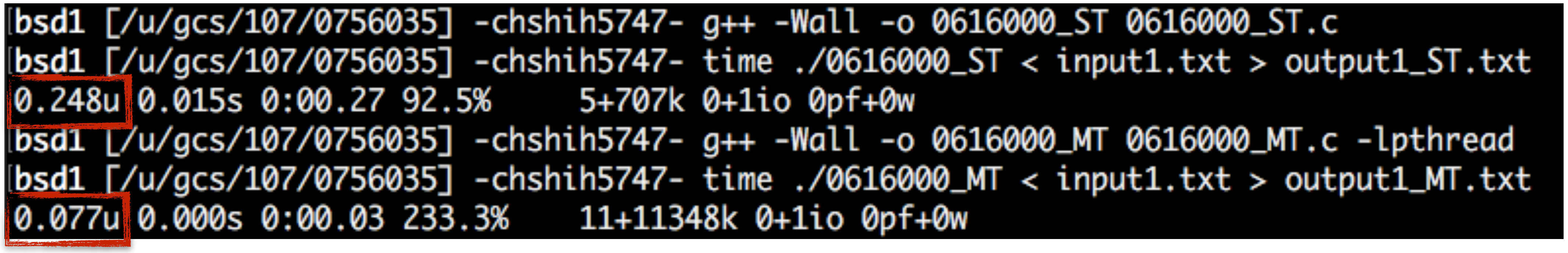
  1. **STDIN (e.g. scanf, cin)**
  2. **Sorting thread function**
     (Use brute-force methods, e.g., bubble sort)
  3. **Merge thread function**
     (Use simple merge sort for merging two sublists)
  4. **Thread management**
  5. **STDOUT (e.g. printf, cout)**

- **DO NOT USE FILE I/O !**

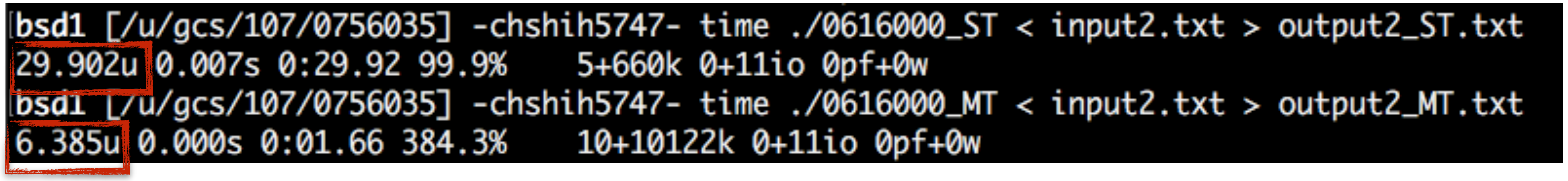
Compile & Run Commands

- Compile:
  (single-thread) $ g++ -Wall -o studentID_ST studentID_ST.c
  (multi-thread) $ g++ -Wall -o studentID_MT studentID_MT.c **-lpthread**

- Run:
  (single-thread) $ time ./studentID_ST < input1.txt > output1_ST.txt
  (multi-thread) $ time ./studentID_MT < input1.txt > output1_MT.txt



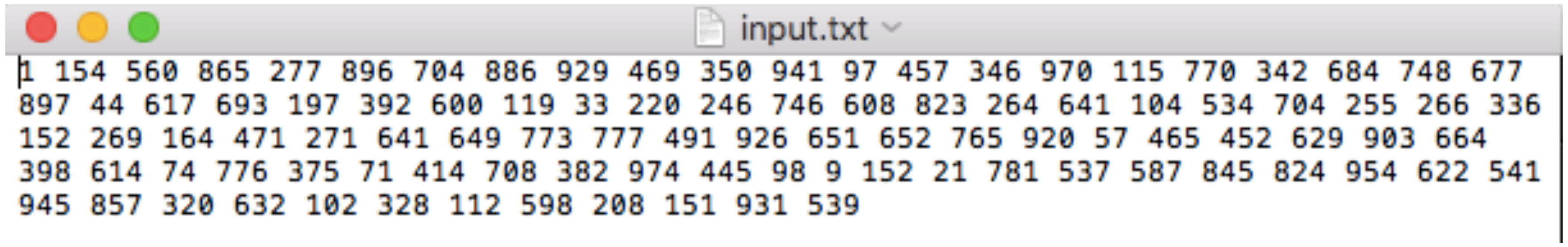
```
bsd1 [/u/gcs/107/0756035] -chshih5747- g++ -Wall -o 0616000_ST 0616000_ST.c
bsd1 [/u/gcs/107/0756035] -chshih5747- time ./0616000_ST < input1.txt > output1_ST.txt
0.248u 0.015s 0:00.27 92.5%     5+707k 0+1io 0pf+0w
bsd1 [/u/gcs/107/0756035] -chshih5747- g++ -Wall -o 0616000_MT 0616000_MT.c -lpthread
bsd1 [/u/gcs/107/0756035] -chshih5747- time ./0616000_MT < input1.txt > output1_MT.txt
0.077u 0.000s 0:00.03 233.3%    11+11348k 0+1io 0pf+0w
```



```
bsd1 [/u/gcs/107/0756035] -chshih5747- time ./0616000_ST < input2.txt > output2_ST.txt
29.902u 0.007s 0:29.92 99.9%    5+660k 0+11io 0pf+0w
bsd1 [/u/gcs/107/0756035] -chshih5747- time ./0616000_MT < input2.txt > output2_MT.txt
6.385u 0.000s 0:01.66 384.3%    10+10122k 0+11io 0pf+0w
```

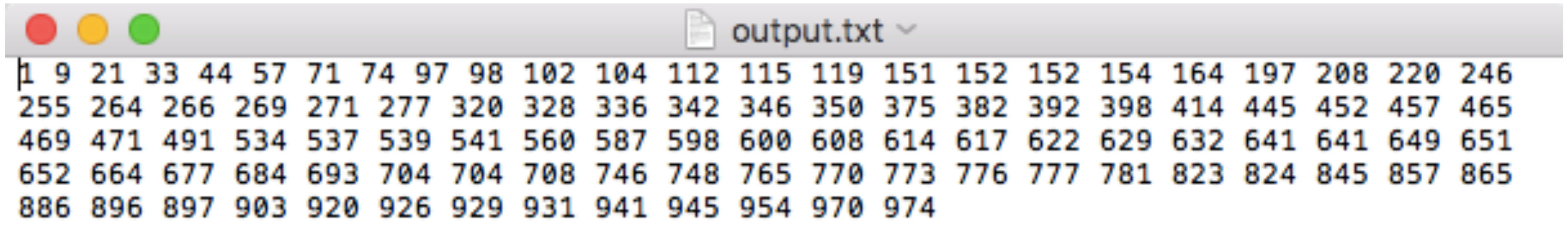
Input/output format

- Input format:
  <all elements separated by space>
  - Largest input: 1,000,000 integers



```
1 154 560 865 277 896 704 886 929 469 350 941 97 457 346 970 115 770 342 684 748 677
897 44 617 693 197 392 600 119 33 220 246 746 608 823 264 641 104 534 704 255 266 336
152 269 164 471 271 641 649 773 777 491 926 651 652 765 920 57 465 452 629 903 664
398 614 74 776 375 71 414 708 382 974 445 98 9 152 21 781 537 587 845 824 954 622 541
945 857 320 632 102 328 112 598 208 151 931 539
```

- Output format:
  <sorted array elements separated by space>



```
1 9 21 33 44 57 71 74 97 98 102 104 112 115 119 151 152 152 154 164 197 208 220 246
255 264 266 269 271 277 320 328 336 342 346 350 375 382 392 398 414 445 452 457 465
469 471 491 534 537 539 541 560 587 598 600 608 614 617 622 629 632 641 641 649 651
652 664 677 684 693 704 704 708 746 748 765 770 773 776 777 781 823 824 845 857 865
886 896 897 903 920 926 929 931 941 945 954 970 974
```

Requirements

- The sorting threads should use the same sorting algorithm as the single-thread program.
- Multi-thread sorting should be much faster than Single-thread sorting, and their results must be exactly the same
- Write your codes in **c/c++**
- You need to hand in one single-thread version and the other multi-thread version. Put studentID_ST.c(.cpp) ,studentID_MT.c(.cpp) and studentID_report.pdf into the same compressed file without folder. The type of compressed file must be “**studentID_hw3.zip**”
- Use **NCTU workstation** as your environment

Grading

- Total score: 100 pts. **COPY WILL GET 0 POINT!**
- Single-thread program: 20 pts (correctness)
- Multi-thread program: 60 pts (correctness & speedup)
- Report: 20 pts
- Incorrect file format: -10 pts
- Deadline: **2019/11/28 (THU) 23:59**
  Late submission will get 0 point!