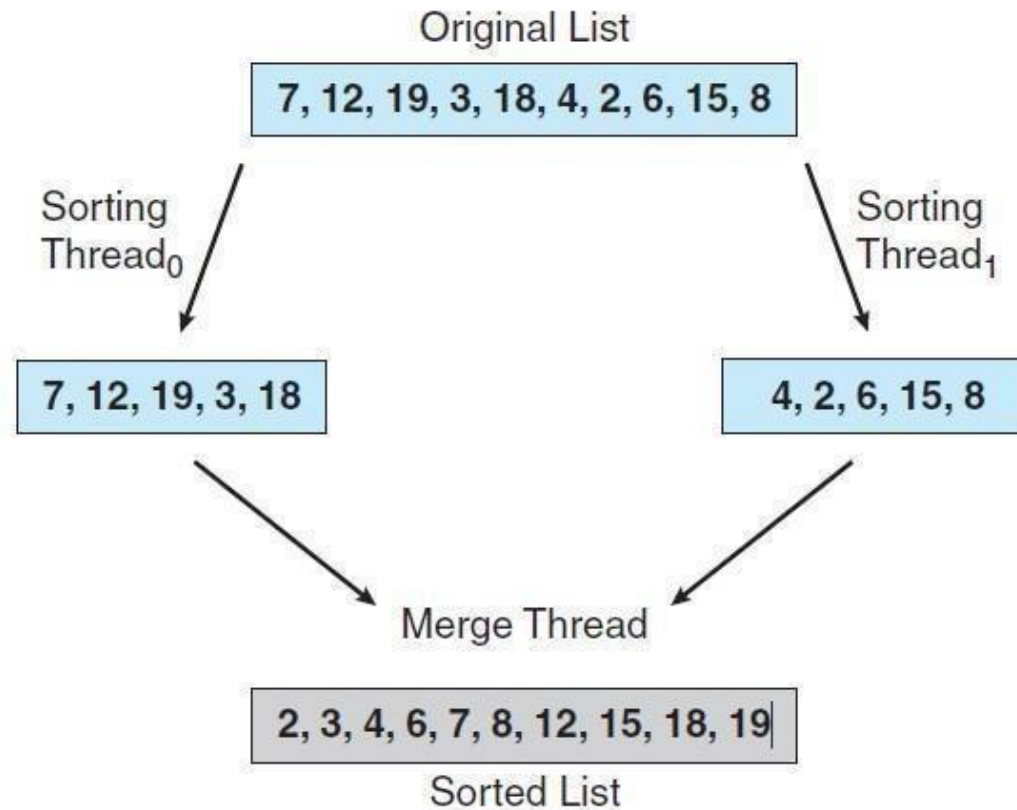


# Multithread Programming

# Multithreaded Sorting Application

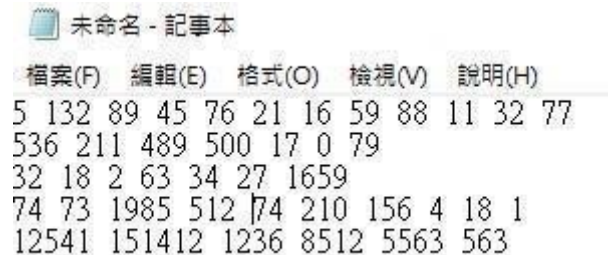
- In this project, you need to write a multithreaded sorting program that works as follow:
- A list of integers is divided into **two smaller lists of equal size**. You have to create **two separate threads (sorting threads)** to sort each sub-list using the **merge sort** algorithm.
- Then, the two sub-lists are merged into a single sorted list by a **third thread (merging thread)**.

# Multithreaded Sorting Application



# Multithreaded Sorting Application

- We will give you a .txt file which contains several lines of **integers** as your input test data. Each line represents one test case which is composed by several numbers. So, you should figure out **how to use argc, argv way to read in the test data line by line first.**
- E.g.



未命名 - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

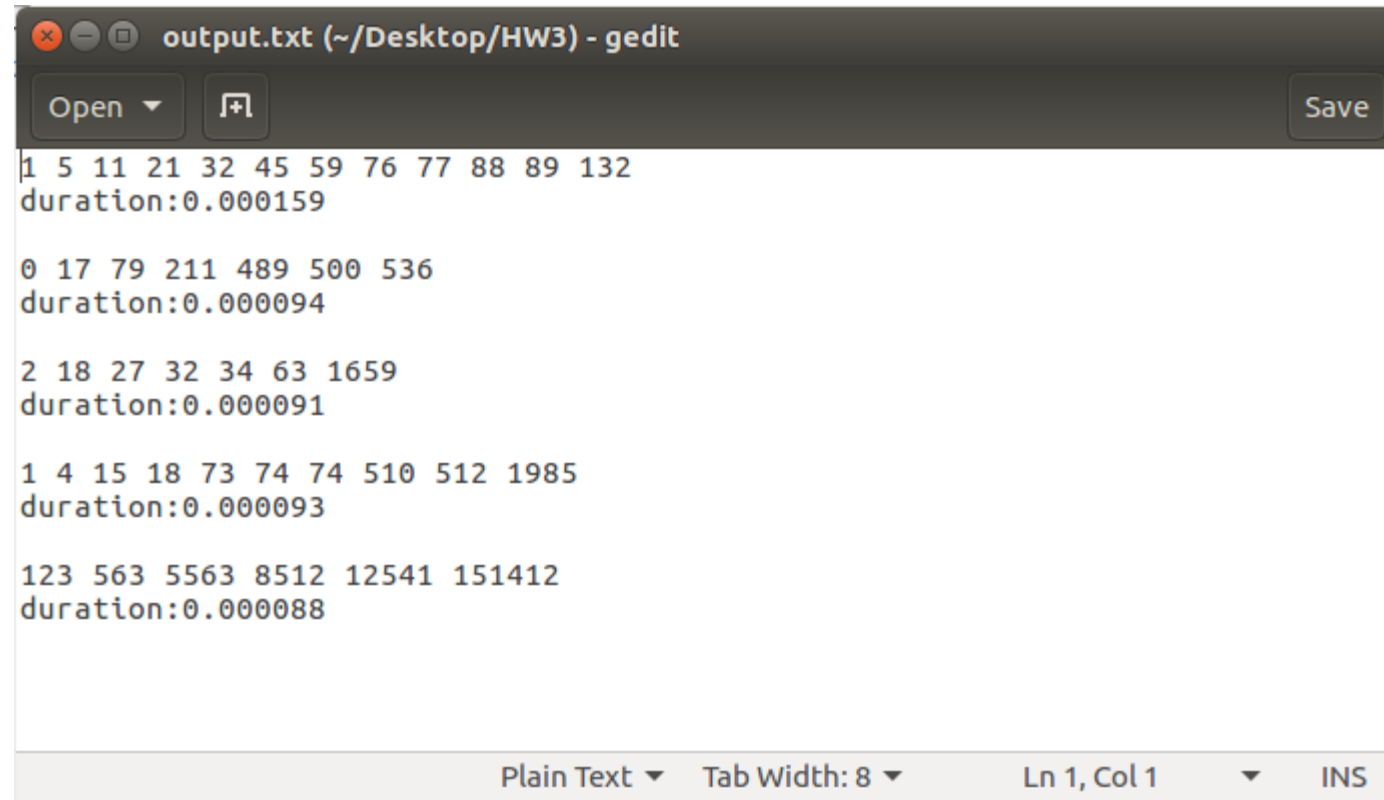
```
5 132 89 45 76 21 16 59 88 11 32 77
536 211 489 500 17 0 79
32 18 2 63 34 27 1659
74 73 1985 512 74 210 156 4 18 1
12541 151412 1236 8512 5563 563
```

# Multithreaded Sorting Application

- Then, you need to implement the merge sort using multithreaded programming: **two threads for sorting, and one for merging.**
- Make sure that the third thread (merging thread) get started after both sorting threads are done, which means you have to **keep the merging thread waiting until the sorting threads finish their jobs by pthread condition wait.**
- Trace **pthread.h** first.
- Finally, you should **output an output.txt file for your sorting result and your running time of each test case.**

# Multithreaded Sorting Application

Your output file should be like the following one:



The screenshot shows a gedit window titled "output.txt (~/Desktop/HW3) - gedit". The window contains five lines of text, each representing a sorting operation. Each line consists of a sequence of numbers followed by a duration. The numbers are: "1 5 11 21 32 45 59 76 77 88 89 132", "0 17 79 211 489 500 536", "2 18 27 32 34 63 1659", "1 4 15 18 73 74 74 510 512 1985", and "123 563 5563 8512 12541 151412". The durations are: "duration:0.000159", "duration:0.000094", "duration:0.000091", "duration:0.000093", and "duration:0.000088". The window has a dark theme and includes standard window controls (close, minimize, maximize) and buttons for "Open", "Save", and "Add". The status bar at the bottom indicates "Plain Text", "Tab Width: 8", "Ln 1, Col 1", and "INS".

```
1 5 11 21 32 45 59 76 77 88 89 132
duration:0.000159

0 17 79 211 489 500 536
duration:0.000094

2 18 27 32 34 63 1659
duration:0.000091

1 4 15 18 73 74 74 510 512 1985
duration:0.000093

123 563 5563 8512 12541 151412
duration:0.000088
```

# Multithreaded Sorting Application

- The output filename must be **output.txt**.
- Please output your result with **the same format as the input** we gave you.
- We will use more difficult test cases to test your program.
- **At most 10000 integers for one test case.**
- You can find some test cases to test your program by yourself.
- You should **implement under Linux**.

# Multithreaded Sorting Application

- Only **merge sort** is acceptable.
- Must use **argc, argv** way to input the test file.
- Use **gcc hw3.c -pthread -o hw3.o** to compile your code.
- Your command to run and output your code **must** be **./hw3.o testcase.txt output.txt**



# Required File

- Hw3\_{studentID}.zip :
  - `hw3.c` (90%)
  - 10% for each hidden test cases, so you need to pass 9 test cases for full score. So pass the test case which we give you will not cost any score.
  - `hw3.o`
  - `output.txt`
  - `hw3_report.pdf` (10%)
  - Tell us how you implement your homework and show us your time and result with some screenshots.

# Multithreaded Sorting Application

- 0 will be given to cheaters, so don't copy & paste your friend's code directly.
- Make sure that you totally understand your code 😊.
- Deadline: 5/5(Tue) 23:59
- We will randomly pick 1/4 of all students to demo in person after the midterm.

# Cautions

- Please remind that your **archive file** must be **.zip**
- There is no hint this time.
- The number of hidden test cases is not as same as the test cases which we give you, so make sure that your code can deal with any number of test cases.
- Your report should be more clear, so this means that you can't just explain your code in one or two words. Also your report should include your **code screenshot**, **result screenshot** and **code explanation**.