**READ ME**

**Matthew Kalitasr NetID:mhk85**
**JOSHUA RASLOWSKY NetID: jtr96**

## How to build project:

To build server: ***gcc -Wall -pedantic sorter_server.c -o sorter_server –pthread***

To build client: ***gcc -Wall -pedantic sorter_client.c -o sorter_client –pthread***

Wall and pandetic are so it won't compile if warnings are present changing certain warnings to errors! And not allowing compile (Better Compile method)
Also both Extra credits are done and work -h 4 will create a thread limit on client

**OUR Protocol of communication**

For each sequence described below there is a common rules for request message format. Request message format consists of:
- First 20 characters (**BLOCK1**) describe request type as defined in macro below:
This is used to set session and (cookie)

#define GET_ID_REQUEST "@RQ_GET_ID@"
#define SORT_REQUEST "@RQ_SORT@"
#define DUMP_REQUEST "@RQ_DUMP@"

All other unused characters in BLOCK1 should be spaces.

Next 5 characters (**BLOCK2**) describe client session id. If session id value length < 5 then all unused characters should be spaces.

Next block (**BLOCK3**) that describes argument to the request is optional, length of it not strictly defined.
Request message should end with special token (**BLOCK4**) as defined in macro below:

#define END_OF_MESSAGE "@MSG_END@"

**Get session id (extra credit) sequence:**
CLIENT → SERVER: [**BLOCK1 =** @RQ_GET_ID@][EMPTY 5 cells][**BLOCK4 =** @MSG_END@]

After the client sent get id request, server evaluates next available sequence id and replies with:
SERVER → CLIENT: [**BLOCK2 =** value of session id][**BLOCK4 =** @MSG_END@]

**Sort sequence:**
Sort sequence consists of two types of sort request:
- request with data of one entry to be sorted (without ending token **BLOCK4**)
- request without data to indicate that there is no more entries in the file to be sorted (with ending token)
This is how we accomplished the sort from end to end

Below are algorithms of sorting sequence:
CLIENT → SERVER: [**BLOCK1 =** @RQ_SORT@][**BLOCK2 =** value of session id][**BLOCK3 =** data of one entry from the file][No ending token = indicates that other entries are expected]

SERVER → CLIENT: [**BLOCK3 =** OK][**BLOCK4 =** @MSG_END@]
blocks #1,2 are missing, answer from server to client is just to indicate that server received data and expecting more

…. other messages sent similar to previous 2 and finally finishing message:
CLIENT → SERVER: [**BLOCK1 =** @RQ_SORT@][**BLOCK2 =** value of session id][**BLOCK4 =** @MSG_END@]

Finishing message informs server that sorting can be performed and no entries from current socket will be sent.

**Dump sequence (get sorted results):**
**We choose to dump after all files were sent by client and not to implement a wait for imput and dump**
CLIENT → SERVER: [**BLOCK1 =** @RQ_DUMP@][**BLOCK2 =** value of session id][**BLOCK4 =** @MSG_END@]

After receiving request as above server will start continuously send sorted data to client and client will repeat with confirmation that everything is OK:

SERVER → CLIENT: [**BLOCK3 =** Data for one entry]
CLIENT → SERVER: [**BLOCK3 =** OK]
…. while all data is not transferred to client

How to build project:
To build server: ***gcc -Wall -pedantic sorter_server.c -o sorter_server -pthread***
To build client: ***gcc -Wall -pedantic sorter_client.c -o sorter_client -pthread***

Run as with arguments as described in specification – first server application, then client.