**Report (ECE 650 Project 2)**

Name: Yichen Huang

NetID: yh348

Date: 02/09/2023

**Overview: Description of thread-safe model**

The implementation includes two versions: lock and no lock. First of all, a lock should be initialized with the line of code highlighted in figure 1.

Text

Description automatically generated

Figure 1. Code of initialization for a lock

For lock version, the code of best fit in the first project is used. The implementation way is to add the highlighted line of code in figure 2-1 at the beginning of both malloc and free.

Text

Description automatically generated

Figure 2-1. Code for lock version (at the beginning)

Then add the highlighted line of code in figure 2-2 before each “return” in malloc and free.

Text

Description automatically generated

Figure 2-2. Code for lock version (at the end)

For no lock version, the original code of best fit in the project 1 is also used. The way to implement is to modify the type of head and tail from “block\_tag\*” to “\_\_thread block\_tag\*”, and use head and tail in new type in the implementation. Figure 3 demonstrates the type modification for both head and tail.

Text

Description automatically generated

Figure 3. Head and tail in new version

The other modification is to add lock and unlock just like for lock version for the function where sbrk() is called.

Text

Description automatically generated

Figure 4. Modification related to sbrk() function for no lock version

**Performance Result Presentation**

With locking and no-locking version of thread-safe malloc/free functions, different results related to

* Execution time: the run-time of the program
* Allocation efficiency: the total size of the data segment after running the test

are shown in Table 1.

Table 1. Results of First Fit and Best Fit

|  |  |  |
| --- | --- | --- |
|  | Lock | No Lock |
| Execution Time (s) | 1.665985 | 0.270894 |
| Data Segment Size (bytes) | 42663704 | 43903200 |

The data printed in terminal has been shown in figure 5 and figure 6.

**Graphical user interface, text

Description automatically generated**

Figure 5. Results of locking version

Graphical user interface, text

Description automatically generated

Figure 6. Results of no locking version

**Results Analysis**

About execution time, time for locking version is approximately 6 times longer than time for no locking version. The reason is that, for lock version, only one single free list is shared by all threads because of locking; but for no lock version, several threads can operate simultaneously.

About data segment size, there is no significant difference between locking and no locking version. Although single free list used for lock version and multiple free list used for no lock version, the block hit rate seems have no evident difference. If on single free list a block cannot hit, on a multiple free lost it also cannot hit. However, based on analysis, memory reuse rate for no locking version will be worse than for locking version, since separate free list for different threads may have adjacent free blocks which cannot be merged together. Thus, the data segment size of locking version is slightly smaller than no locking version.

Thus, the trade-off here is that, in the case of execution time is more important, no locking version is evidently better; when memory size is regarded as the more significant factor, locking version should be chosen.