Thread-Safe Malloc Library Report

Section 1: Requirements and summary of development.

The main purpose of this assignment is to design the thread-safe malloc function and the corresponding free function of two strategies in C language. The Linux environment is Ubuntu 18.04.6 LTS during the development, and Valgrind is used to solve memory leaks, and GDB is used for debugging. The development process is mainly composed of design and development, testing and debugging, results and analysis.

Section 2: Design, implementation and test

As for the design, in order to implement thread-safe malloc and free functions that allow for concurrency, it's crucial to be mindful of race conditions. Which can lead to incorrect or non-deterministic results. To prevent this, it's necessary to control the possible interleaving of thread executions. Here I used pthread library to implement lock version and using a integer sbrk to indicate a no lock version.
In implementation, I use a LinkedList to track free blocks to avoid race, and using _thread to ensure that each thread has its own LinkedList and can only make changes to its respective LinkedList. And the value of sbrk determined whether using the lock mode or no lock mode. As the brk is not thread safe, here I used thread_lock to this value.
For testing, first tested each function, and then debugged through the test files in the provided homework2-kit after there are basically no problems.

Section 3: Performance Results & Analysis

Version	Execution Time /s	Data Segment Size /bytes
Lock	0.08	43038784
Non-Lock	0.06	43203840

As for the execution time. The non-lock version of the code runs faster than the lock version because the non-lock version only places the sbrk value in the lock operation, whereas the lock version places the entire original malloc/free code within the lock operation. As a result, the non-lock version allows more code to run concurrently outside of the lock operation, which ultimately results in faster performance.
As for the data segment size, the result is quite near as the two versions code both used best-fit strategy. And there are many threads to malloc.