Report

Yidan Sun NetID:ys303

1. Implement

These two thread-safe versions of `malloc()` and `free()` are based on best-fit code from project 1. There is a linked list to store free spaces. When malloc, we search through the linked list at first, if we could find a proper space for allocation, we use this space and edit the linked list, otherwise, we call `sbrk()`. When free, we search through the linked list and find a proper space to insert the to-free space.

Lock Version

Since the linked list can be accessed by all threads, any search or modification of the linked list could be affected by other threads. Therefore, all code in `malloc()` and `free()` are critical part, we need to lock at the start of the function, and unlock just before `return`, otherwise, there will be overlapped regions.

As a result, we actually allow no concurency happen in this lock version.

Nolock Version

For nolock version, thread-local storage is used. Since the linked list is shared by all threads, we could declare our head node by `__thread Node* head_nolock_`. In this way, each thread will have their own head and there will not be race conditions. And we should note that `sbrk()` function is not thread-safe and we need to acquire a lock immediately before calling `sbrk()` and release a lock immediately after calling `sbrk()`.

In this nolock version, concurency is allowed to happen. Multiple threads might happen at the same time and there is no race condition because of thread-local storage. In this way, we could save a lot of time.

2. Measurement

	Time	**Data Segment Size**
Lock Version	1.05 s	42409504 bytes
Nolock Version	0.46 s	42743040 bytes

Lock Version spends more time than Nolock Version

Since no concurency allowed in Lock Version, all threads need to go through one by one as if there were only one thread. However, in Nolock Version all threads could happen together. As a result, Lock version needs more time than Nolock Version.

Lock Version needs fewer space than Nolock Version

Lock Version data segment size is smaller than nolock version, but there is no big difference. The reason I guess is that Lock Version only use one linked list therefore it could have the maximum space utilization. But for Nolock Version, there are multiple linked lists such that ree space in one thread cannot be shared by another thread. Therefore, nolock version needs more space in total.