Project 2 Thread Safe Malloc

Description of Thread Safe model:

As part of this project 2 we need to extend our best-fit malloc and free by making it thread safe. There are two ways to do it (Locking and Non Locking)

ts_malloc_lock() & ts_free_lock():

These two functions use locking to ensure synchronization between threads. There is a race condition to update free_head (i.e. the list which holds the free space) when allocating memory block from free list or freeing memory block. Therefore we use pthread_mutex_lock (free_list_lock) to avoid race conditions. We lock this lock before calling the critical section [i.e allocate_from_freelist() (in case of ts_malloc_lock()) and ts_free_lock()].

Since sbrk() (i.e. the system call to expand the heap) is not thread safe. We use sbrk_lock right before and after the call to protect any race condition. This is how we ensure that ts_mallock_lock() & ts_free_lock() is thread safe.

ts_malloc_nolock() & ts_free_nolock()

Here we make use of thread local storage to ensure that there is no race condition. Since race conditions happen only when many threads operate on the same memory, we assign a separate free list (i.e. free_head_nonlock) for each thread. This ensures that there is no common memory between threads. Therefore no race condition between multiple threads of ts_malloc_nolock() and ts_free_nolock(). But we still need to use sbrk_lock since sbrk() is not thread safe.

(Note: There could be scenarios where we might not return the best fit during malloc from the allocated free space using this approach. Since each thread has their own list of free space, there could be a scenario where the free space you are searching might not exist in the current thread but might be present in some other thread.)

Performance Report

ts_malloc_lock() & ts_free_lock()

```
ka266@vcm-30704:~/ece-650/ece-650-project1/ts_malloc/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.516794 seconds
Data Segment Size = 42973080 bytes
ka266@vcm-30704:~/ece-650/ece-650-project1/ts_malloc/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 1.036937 seconds
Data Segment Size = 42467144 bytes
ka266@vcm-30704:~/ece-650/ece-650-project1/ts_malloc/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.317004 seconds
Data Segment Size = 43184416 bytes
ka266@vcm-30704:~/ece-650/ece-650-project1/ts_malloc/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.279318 seconds
Data Segment Size = 42927568 bytes
ka266@vcm-30704:~/ece-650/ece-650-project1/ts_malloc/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.296099 seconds
Data Segment Size = 42788448 bytes
```

ts_malloc_nolock() & ts_free_nolock()

```
ka266@vcm-30704:~/ece-650/ece-650-project1/ts_malloc/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.156007 seconds
Data Segment Size = 42302216 bytes
ka266@vcm-30704:~/ece-650/ece-650-project1/ts_malloc/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.138253 seconds
Data Segment Size = 43636552 bytes
ka266@vcm-30704:~/ece-650/ece-650-project1/ts_malloc/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.133991 seconds
Data Segment Size = 42319768 bytes
ka266@vcm-30704:~/ece-650/ece-650-project1/ts_malloc/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.180804 seconds
Data Segment Size = 42274120 bytes
ka266@vcm-30704:~/ece-650/ece-650-project1/ts_malloc/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.234689 seconds
Data Segment Size = 43003528 bytes
```

Analysis of Performance Report:

Average for 5 tries	**Execution time**	**Data Segment Size**
Locking version	0.4892304 seconds	35132931.2 bytes
Non locking version	0.1687488 seconds	42707236.8 bytes

There are two basic trends in the performance report.

Comparison between Execution time:

Since the locking version locks critical sections during its execution. Multiple threads cannot operate on the data simultaneously in the locking version. However, in the non locking version each thread will have its own separate list and doesn't have to wait for any other thread. It can take full advantage of multithreading. Therefore, the non locking version has a lesser average execution time.

Comparison between Data Segment Size:

In the non-locking version, each thread will have its own free list. Hence we might not always find the best fit. Due to this we expand the heap more than the Locking version. Hence data segment size more for non-locking version compared to locking version