HW1 Malloc Library Part2
Thread-Safe Malloc

This file will explain different ways to implement thread-safe malloc function and compare the difference between their behavior.

1 Implementation

The implementation of the malloc function share the same data structure with the implementation in HW1. However, to make it Thread-safe two methods have been used, the lock version and the unlock version.

1.1 The lock version

To avoid race condition, a lock have been used. If all the threads are sharing the same free linked list, then only one thread can edit the free linked list at the same time. The simplest way to do it is only allowing one thread to free or malloc at the same time. So a pthread_mutex_t lock have been added as a global variable. The lock is set to lock at the beginning of the ts_malloc_lock() and the ts_free_lock() function and is set to unlock before the return of those two function.

1.2 The unlock version

In the unlock version, every threads have their own free linked list. A local variable

__thread memBlock * multiFreeHead

is used to give each thread different free head. Inside the ts_malloc_nolock() and the ts_free_nolock() function, the multiFreeHead was used instand of freeHead. Different threads have to maintain their own free list, and cannot use others free space. Also, since sbrk()function is not thread-safe, a lock called sbrklock have to be used for the sbrk() function.

2 Result

table 2.1 execution time and fragmentation of different test case

Index	lock		unlock	
	exec time	fragmentation	exec time	fragmentation
1	1.549220	42484152	0.482658	42566128
2	0.719563	42446304	0.355158	42708104
3	1.026070	42351504	0.211435	42184912
4	0.495000	42173744	0.423229	42537512
5	0.735129	42164160	0.599574	42775440
Average	0.9049964	42323972.8	0.4144108	42554419

Note: this result is base on the default parameter.

3 Analysis

According to table 2.1 it is clear that the lock method takes more time then the unlock method. This is because all the threads share the same free list in the lock version, and only one thread can call the free or malloc function at a time. So one thread has to wait another thread to finish its function and this time makes the execution time longer. In the unlock version, on the other hand, have multiple free list and multiple thread can free and malloc at the same time. Different threads are totally independent, so no waiting time is needed before call a function.

However, according to the table 2.1 it is obvious that the fragmentation of the unlock version is higher than the lock version. This is because that in the unlock version, different threads cannot share others free space. For example, even thread A malloc 2GB of space and free all of them, thread B cannot use any of those space but only to sbrk() a new space. However, in the lock version, all the thread shares the same link list so thread A can use the space that free by thead B. Thus, more fragmentation is needed in the unlock version.

4 Other Approaches and Improvements

There are many other ways to implement this thread save malloc and free. For example, for the lock version, rather than lock all the malloc and free function, a thread save linked list can be use to implement the function. There are one more thing about the no lock version, two thread cannot free the same block, else it may cause some conflict.

5 the result picture of thread_test_measurement

```
hread
gcc -O3 -ggdb3 -I../ -DLOCK_VERSION -L../ -Wl,-rpath=../ -o thread_test_malloc_free_change_thread thread_test_malloc_free_change_t
hread.c -lmymalloc -lrt -lpthread
gcc -O3 -ggdb3 -I../ -DLOCK_VERSION -L../ -Wl,-rpath=../ -o thread_test_measurement thread_test_measurement.c -lmymalloc -lrt -lpt
hread
yl475@login-teer-01 [production] ~/HW2/thread_tests $ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 1.549220 seconds
Data Segment Size = 42484152 bytes
yl475@login-teer-01 [production] ~/HW2/thread_tests $ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.719563 seconds
Data Segment Size = 42446304 bytes
yl475@login-teer-01 [production] ~/HW2/thread_tests $ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 1.026070 seconds
Data Segment Size = 42351504 bytes
yl475@login-teer-01 [production] ~/HW2/thread_tests $ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.785364 seconds
Data Segment Size = 42171560 bytes
yl475@login-teer-01 [production] ~/HW2/thread_tests $ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.735129 seconds
Data Segment Size = 42164160 bytes
yl475@login-teer-01 [production] ~/HW2/thread_tests $ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.495000 seconds
Data Segment Size = 42173744 bytes

gcc -O3 -ggdb3 -I../ -DNOLOCK_VERSION -L../ -Wl,-rpath=../ -o thread_test_malloc_free_change_thread thread_test_malloc_free_chang
_thread.c -lmymalloc -lrt -lpthread
gcc -O3 -ggdb3 -I../ -DNOLOCK_VERSION -L../ -Wl,-rpath=../ -o thread_test_measurement thread_test_measurement.c -lmymalloc -lrt -
pthread
yl475@login-teer-01 [production] ~/HW2/thread_tests $ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.482658 seconds
Data Segment Size = 42566128 bytes
yl475@login-teer-01 [production] ~/HW2/thread_tests $ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.355158 seconds
Data Segment Size = 42708104 bytes
yl475@login-teer-01 [production] ~/HW2/thread_tests $ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.211435 seconds
Data Segment Size = 42184912 bytes
yl475@login-teer-01 [production] ~/HW2/thread_tests $ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.423229 seconds
Data Segment Size = 42537512 bytes
yl475@login-teer-01 [production] ~/HW2/thread_tests $ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.599574 seconds
Data Segment Size = 42775440 bytes
yl475@login-teer-01 [production] ~/HW2/thread_tests $ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.137308 seconds
Data Segment Size = 42136592 bytes
yl475@login-teer-01 [production] ~/HW2/thread_tests $
```