Report

Student Name: Huidan Tan

NetID: ht175

Implementation

Malloc and free are subject to race condition when there're multi-thread on one process. **Race condition** means "system's substantive behavior is dependent on the sequence or timing of other uncontrollable events. It becomes a bug when one or more of the possible behaviors is undesirable." To avoid this situation, two types of method are implemented--accordingly `Lock-based synchronization` and `Thread-local storage`

For Lock-based synchronization, all threads share one `free-list`. This `shared free-list` may cause **Race condition**. To avoid that, `pthread_mutex_lock` and `pthread_mutex_unlock` in c library are used. This way ensures that there's only one thread at a time in a critical section. Other thread needs to wait for lock to be released to modify the `free-list`. The code is as below.

```
void *ts_malloc_lock(size_t size){
    //lock
    pthread_mutex_lock(&lock);
    int sbrk_control = 0;
    void * ans=bf_malloc(size,sbrk_control,&lock_head);
    pthread_mutex_unlock(&lock);
    return ans;
}

void ts_free_lock(void *ptr){
    pthread_mutex_lock(&lock);
    bf_free(ptr,&lock_head);
    pthread_mutex_unlock(&lock);
}
```

Compared lock-based version, for Thread-local storage, each thread has its own `free-list`. `pthread_mutex_lock`

is not needed for the `free-list`. However, sbrk function is not thread-safe. Therefore, pthread_mutex_lock is used only for sbrk function. The code is as below.

```
 pthread_mutex_lock(&lock);
 p=sbrk(size+sizeof(meta_t));
 pthread_mutex_unlock(&lock);
```

Performance Result

Strategy/Measurement	Execution Time	Data Segment Size

Strategy/Measurement	Execution Time	Data Segment Size
Lock-based synchronization	0.237501 seconds	42494496 bytes
Thread-local storage	0.170355 seconds	42610976 bytes

Result Analysis

- **speed**

  Lock-based synchronization strategy spends more time than Thread-local storage strategy. This because Lock-based synchronization locks all malloc and free code when one thread is modified the free-list. However, for Thread-local storage strategy, it only locks the sbrk function. Therefore, it allows more code to run simultaneously.

- **Data segement size**

  Lock-based synchronization has less data Segment Size. This might because all threads share one `free-list`. merge may happen more frequently.