15.12.2019

Ensar KAYA
21502089

Faruk ŞİMŞEKLİ
21502464

Project 3

In this project, we have implemented a thread-safe memory management library. This library will manage a given chunk of free memory by an application. The library will handle the requests to allocate and deallocate memory in the contiguous chunk of free space, that is created by the application. The library will be able to use hole-list approach. In this way, it can use 3 different methods: first-fit, best-fit, and worst-fit. After the implementation of the library, we have done some experimentations to measure the time and see the memory management.

	First-Fit	Best-Fit	Worst-Fit
# of alloc : 100	0.3	0.5	0.5
# of alloc :1000	19	21	25
# of alloc :10000	3082	3451	4691

Figure 1. Time - Number of allocations in different allocation methods
(*) Values in the cells have the unit milliseconds.

In this experiment, we have tried various values for the number of allocation from the memory. When number of allocation increases memory gets bigger as well thinking about the worst case that can happen in random number generator. For example, when number allocation is 100, the memory is 10KB, when it is 1000, memory is 1MB, and when it is 10000, memory is 100MB. In this experiment, we have created some number of random numbers and made the memory allocations accordingly. Then, we have freed the allocations that have odd index so that we have the pattern in the memory as occupied - available - occupied - available … After that we have tried to allocate some memory from the available holes using different allocation methods such as first-fit, best-fit, and worst-fit. And we have measured the time taken by each method. You can see the results in *Figure 1*.

Before the experiment, we were expecting that best-fit and worst-fit would take more time because they would have to traverse the whole memory to find the right spot to allocate. This is not the case for first-fit. Whenever it finds a bi enough hole, it takes it. So, it makes sense that first-fit takes less time than best-time and worst-time. The thing we could not understand is that why worst-fit takes more time that best-fit does. Both of them see the all memory. As you can see when number of allocations is 10000, the difference is quite dramatic, it is almost %33. Their implementation Only reason we could come up is the fat of possibility. We have tried this case several times, still the difference did not change much. Even though, the probability of this to happen is quite small, it may happen or there is some other reason we could not find.

After this point, we also have seen that our implementation of memory allocation methods can handle requests whose range is from 1 byte to 10MB. Even though it is mentioned maximum request will be 2MB, we can handle more. Also we can handle various chunk sizes whose range is from 1KB to 100MB. Even though the limits mentioned in the specifications are different we have tried to make our implementation more robust and powerful.

Our library also provides a thread-safe environment for threads. And with the synchronization, we are able to save the correct form of the memory. In terms of time, it does not make much of a difference though.