# Assignment 6

Georgiy Krylov

November 3, 2024

## ASSIGNMENT IS TO BE COMPLETED INDIVIDUALLY BY ALL STUDENTS!

## 1 Description

This assignment is to develop a simulation of contiguous memory allocation. **The assignment is Due by 11:59 p.m. on Sunday, 10th of November 2024 (one minute before Monday).**

## 2 Task

One of the simplest ways to allocating memory is to service the processes requesting memory with a chunk of contiguous memory of the requested size. In this model of memory allocation, the chunks are reserved for the process requested it until the process terminates [1]. Upon termination, the memory is made available for allocation again. Initially the whole memory is available for allocation. The task of this assignment is to implement three algorithms that can be used for this:

1. *First-fit*: The first chunk of available memory that is big enough is used to service the allocation request.
2. *Worst-fit*: The biggest chunk of available memory is used to service the allocation request.
3. *Best-fit*: The smallest chunk of available memory that is big enough is used to service the allocation request.

More details are available in Chapter 9.2.2 of the book [1], or in the slides uploaded to D2L. At the very beginning of the algorithms' operation, just one hole of maximum available memory size exists. Throughout the operation of the algorithms, before any process is terminated, you are to try allocating memory from this hole. Suppose the first process terminates and the memory occupied by this process is released. Now, you have two holes, and you need to decide which one to use to service the next allocation request, based on the algorithm.

Do not forget that if any of two chunks are adjacent, they need to be merged to become one bigger chunk. For example, you may have two chunks of free memory: one with the addresses ranging between 0 and 20, and another one, with the addresses ranging between 20 and 40. These two chunks should be grouped together into a chunk with addresses ranging between 0 and 40, to be able to service a bigger process.

Consider the following input format:

```
N 1 500
T 7
T 1
S
```

This file describes the process #1 being allocated (using **N** command), requesting 500 bytes. If your algorithm can service this request, the chunk of free memory should be reduced by 500 bytes, and an entry in process table should be created. Otherwise, an error message (like **"Process # failed to allocate x memory"**) should be printed, but the program should keep executing the requests. The program then attempts to terminate the process #7 (using **T** command). As the process #7 was not created before, the program should print an error message (like **Process # failed to free memory**, and keep executing the requests. If the process was on the process table, the process should have been terminated and resources should have been released.

The summary should be printed upon reading the **S** command. The summary contents are to reflect the total number of processes created, total allocated memory, total number of processes terminated, total size of freed memory, final memory available across all the chunks, final smallest and largest fragmented memory chunks, the number of failed requests, and the number of free memory chunks.

Your program to accept the maximum memory size specified as a command line argument "-s#", and the algorithm using "-f", "-b", "-w". See the sample starter code provided.

You are not allowed to implement a "hard-coded" solution for this assignment (e.g. make assumptions about the size). For this assignment, if you submit the code that passes all tests and contains no "hard-coded" parts, you will get the full mark.

# 3 Input/Output format

See the provided test suite. If you place your .c files and .h files in the "code" folder and type "make", the code should be compiled. To test your code, you can type "make all", which will delete an executable called **program1**, delete the contents of the **.student_out** directory, and run all the tests. The individual tests can be performed by typing "make 1", "make 2", .., up to "make 14". Empty output from the diff program means the test has passed successfully. Sample input files are available in the **.in** directory, the expected output

is available in the **.out** directory, and your output will be generated to the **.student_out** directory.

## 4 Submission instructions

Please submit just your C and H files to D2L Assignment box. Make sure your code compiles and runs. Make sure your code follows the specified input/output format. You must use C programming language to solve this assignment. For this assignment, your grade will be proportional to the percentage of the tests passing successfully.

## NOTE: THE INPUT AND OUTPUT OF YOUR PROGRAM IS SPECIFIED SUCH THAT THE MARKER CAN AUTO TEST YOUR SUBMISSIONS. PLEASE FOLLOW THE SPECIFICATIONS! USE THE MAKEFILE).

## References

[1] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts, 10th Edition*. Wiley, 2018.