**CENG 305 – Operating Systems – Fall 2021**

**Project 2** - (Due 4/01/2022)

**Subject:** A Simple Memory Management Unit for a Multithreaded Program

You cannot upload this assignment to any web page, forum site or send somebody who does homework for money. If you take such an attitude, you will be in opposition to the law on the protection of personal data and violation of copyrighted verbatim. In addition, disciplinary action will be arranged by the Rules and Regulations Governing Student Disciplinary Actions in Institutions of Higher Education.

In this assignment, you are going to implement a multithreaded program that simulates a contiguous memory management unit. Your program will simulate dynamic memory allocation in an operating system. Your program will include three memory allocation strategies: **first-fit, best-fit, and worst-fit.** These are strategies most used to select a free hole from the set of available holes. Your program will simulate the allocation of a physical memory of a given size by using those three strategies. Since there is no paging, a contiguous space of memory will be allocated to a user program. Your program will inform the user about current number of holes and total size of the internal fragmentation. In addition, it will give error messages if an allocation is not possible due to **external fragmentation** or insufficient memory space.

You should assume that the physical memory is broken into fixed-sized blocks of **4 KB** and there are **4 threads** in the system. In this system, you have 4 threads which use the same global Memory. In the program, you may suppose that those 4 threads launch or terminate some processes and your program will allocate and deallocate memory space for them. Since the threads may modify the same data structure simultaneously, you need to take necessary precautions to prevent race condition.

Your program will be called **mmu** and will have the following parameters.

>> **mmu** <size\_of\_memory> <file\_name> <alloc\_strategy>

<size\_of\_memory> is the total size of memory in KB.

<file\_name> is a string value that shows the base name for the input files. For “filename.txt” as a parameter, thread 0 will read “filename.txt\_0”, thread 1 will read “filename.txt\_1”, threads 2 will read “filename.txt\_2” and finally threads 3 will read “filename.txt\_3”.

<alloc\_strategy> is an integer value and used for first-fit =1, for best-fit=2, and for worst-fit=3.

Therefore:

>> **mmu** 1024 process.txt 1

will manage the memory of size 1024 Kilobytes for first-fit strategy.

We know that the physical memory is broken into fixed-sized blocks of 4 KB and will be allocated in units based on this block size. That is, if the size of memory is 1024 KB then there are 256 blocks (frames) available for processes memory allocations.

The following is an example input file “filename.txt\_0”:

B 1 255

B 2 256

B 3 254

E 2

B 4 512

B 5 40

B 6 40

E 5

For example, the process with ID 1 needs to allocate 255 KB memory, 255/4 = 63.50 so 64 frames of memory will be used by that process. After running this process, there will be 192 blocks available in the memory.

Each line starts with a letter B or E. If the first letter is B, that means a program is begun (started) and we need to allocate memory for this program. Next comes the program (process) id and then the size (in KB) of the program. If the first letter in a line is E, that means a program will end (finish). The ID of the program that is terminated is indicated next. For example, “B 1 255” means that a process with id 1 is started and the size of the process is 255 KB. That is, we must allocate memory for this process (if enough space is available, otherwise print an error message). The unit of all program sizes is KB.

Your algorithm should keep the track of allocated and free space in the memory.

Your algorithm will produce two outputs. Firstly, you need to create a file named “memory.txt” which shows the latest situation of your memory with process IDs in the program. Secondly, your program will show the list of holes in the memory after all the requested allocations and deallocations are made to the screen. Note that some allocation requests may not be satisfied, in this case there should be an informative error message. The list of holes will be printed out in the following format: <start-address> <size>.

For example, the following is the output of “**mmu** 1024 processes.txt 1”:

64 40

84 176

192 256

All values are in KB. It says that there are 3 holes. The first hole, for example, starts at 64th frame and its size is 40 KB.

The memory is broken into 4 KB fixed-size blocks. Hence, we can have some internal fragmentation. An internal fragmentation should not be considered as part of a hole, since it cannot be utilized for another program. This is another information you need to print out in your program.

**Sample output for the command line:**

>> mem\_sim 1024 processes.txt 1

Program Launched, First-Fit strategy will be used

B 1 255 -> 64 frames will be used, remaining #frames: 192

B 2 256 -> 64 frames will be used, remaining #frames: 128

B 3 254 -> 64 frames will be used, remaining #frames: 64

B 7 300 -> ERROR! Insufficient memory

E 2 -> 64 frames are deallocated, available #frames: 128

B 4 512 -> 128 frames will be used, ERROR! External fragmentation

B 5 40 -> 10 frames will be used, remaining #frames: 118

B 6 40 -> 10 frames will be used, remaining #frames: 108

E 5 -> 10 frames are deallocated, available #frames: 118

Total free memory in holes: 118 frames, 472 KB

Total memory wasted as an internal fragmentation: 3 KB

Total number of rejected processes due to external fragmentation: 1

Total number of rejected processes due to insufficient memory: 1

Holes:

64 40

84 176

192 256

The content of “memory.txt:” (? shows a hole)

|  |
| --- |
| 1 0-63  ? 64-73  6 74-83  ? 84-127  4 128-255 |

**Report**:

Evaluate each algorithm individually and also compare the three strategies with each other in terms of number of external fragmentations and memory allocation fails. Discuss the results in your report.

The project will be done in Linux operating system using C programming language.

You will submit your project online through aybuzem.ybu.edu.tr. Your soft-copy report will also be included in your directory that you submit as a package (a zip file).

|  |
| --- |
| You cannot upload this assignment to any web page, forum site or send somebody who does homework for money. If you take such an attitude, you will be in opposition to the law on the protection of personal data and violation of copyrighted verbatim.  Cheating and studying together with the other project groups is strictly prohibited. A group should never show their codes to another group whether in seeking aid or giving it. A student may ask fellow student general questions which do not pertain to a particular programming assignment but may not ask any programming aid.  Disciplinary action to be taken against cheating is arranged by the Rules and Regulations Governing Student Disciplinary Actions in Institutions of Higher Education. |

Good luck!