Ankara Yildirim Beyazit University
CENG 305 – Operating Systems – Fall 2020

Project 3 - (Due 22/01/2021)

Subject: Memory Management – Simulation of dynamic memory allocation for Contiguous Memory Systems

In this assignment, you are going to implement a contiguous memory management program that 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 commonly 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 to various user programs using these 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.

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

```
>> mem_sim <size_of_memory> <file_name> <alloc_strategy>
```

<alloc_strategy> is an integer value and for first-fit =1, for best-fit=2, and for worst-fit=3.

Therefore;
```
>> mem_sim 1024 processes.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 second parameter <file_name> specifies a file in which details of processes that will run on the computer are explained. The following is an example input file:

```
B       1       298
B       2       300
B       3       300
B       4       299
E       2
B       5       400
B       6       100
B       7       100
E       6
B       8       160
B       9       118
```

For example, the process with ID 1 needs to allocate 298 KB memory, 298/4 = 74.50 so 75 frames of memory will be used by that process. After running this process there are 181 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 the means a program will end (finish). The id of the program terminated is indicated next. For example, "B 1 298" means that a process with id 1 is started and the size of the process is 298 KB. That means we have to allocate memory for this process (if possible; if enough space is available). The unit of all program sizes is KB.

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

The output of your program will be the list of holes in memory after all the requested allocations and deallocations are made. 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 "mem_sim 1024 processes.txt 1" :

```
75 25
125 25
254 1
```

All values are in KB. It says that there are 3 holes. The first hole, for example, starts at address 75 KB and its size is 25 KB. No empty lines will be in the output.

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 the another information you need to print out in your program.

Sample output for the command line:

```
>> mem_sim 1024 processes.txt 1

Program Launched
B      1      298    -> 75 frames will be used, remaining #frames: 181
B      2      300    -> 75 frames will be used, remaining #frames: 106
B      3      300    -> 75 frames will be used, remaining #frames: 31
B      4      299    -> ERROR! Insufficient memory
E      2             -> 75 frames are deallocated, available #frames: 106
B      5      400    -> 100 frames will be used, ERROR! External fragmentation
B      6      100    -> 25 frames will be used, remaining #frames: 81
B      7      100    -> 25 frames will be used, remaining #frames: 56
E      6             -> 25 frames are deallocated, available #frames: 81
B      8      160    -> 40 frames will be used, ERROR! External fragmentation
B      9      118    -> 30 frames will be used, remaining #frames: 51

Total free memory in holes: 51 frames, 204 KB
Total memory wasted as an internal fragmentation: 5 KB
Total number of rejected processes due to external fragmentation: 2
Total number of rejected processes due to insufficient memory: 1

Holes:
75 25
125 25
254 1
```

Report:
Evaluate each algorithm individually and also compare the three strategies with each other. 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).

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 a 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!