Operating System Principles
COSC 1112/1114
Assignment 2 - Segmented Memory Allocator

Assessment Type	Individual assignment. Submit online via Canvas > Assignment 2. Marks awarded for meeting requirements as closely as possible. Clarifications/updates may be made via announcements or relevant discussion forums.
Due Date	Friday, 23:59, 16 October
Marks	40

1. Overview

To a large extent, programming is about management of memory as it's one of the key resources that we have at our disposal as programmers. What we want you to understand the pros and cons of different memory allocation strategies by simulating and comparing them given a set of memory requests. Please note that you may choose to implement your solution to this assignment in C or in C++. However, you will be using low-level operating systems calls implemented in plain C.

2. Learning Outcomes

This assessment relates to the learning outcomes of the course which are:

CLO 3. Explain the objective and functions of modern operating systems, explain memory hierarchy and cost-performance trade-offs, explain the operation, implementation and performance of modern operating systems, and the relative merits and suitability of each for complex user applications

CLO 4. Compare and contrast the common algorithms used for both pre-emptive and non-pre-emptive scheduling of tasks in operating systems, such a priority, performance comparison, and fair-share schemes. Contrast kernel and user mode in an operating system

CLO 5. Evaluate and report appropriate design choices when solving real-world problems

CLO 6. Analyse the key trade-offs between multiple approaches to operating system design.

3. Preliminary

The basic system calls used to allocate memory in UNIX-like operating systems are brk() and sbrk(). These functions allocate and free memory by shifting the "program break" or heap frontier up and down. The heap starts at low addresses and so shifting the program break upwards to allocate memory and shifting it downwards to free memory. The function prototypes for these are:

```
int brk(void * addr);
```

This function sets the program break to be the value passed in by addr. If the setting of this values is successful, brk() will return 0.

```
void * sbrk(intptr_t increment);
```

This function adds or subtracts from the current program break to allocate memory (if a positive number is provided) and to free memory (if a negative number is provided).

In this assignment, you will use these functions to allocate memory; to make it simple, you are not required to free the memory once allocated.

Assignment Task

Allocation and Deallocation Lists: You will use two linked lists to manage memory allocation:

- A linked list of memory blocks allocations - allocMBList
- A linked list of memory blocks that have been allocated but deallocated later [1]- freedMBList

In each of the lists, you need to keep track of the memory block information (e.g., the starting address in memory and the size).

> *Note you are not expected to implement the linked lists yourself. At this point you should know how linked lists work and should be able to use an already existing library. As such there are no marks awarded for the linked list itself.*

Memory Allocation: Given a piece of information (like *your_first_name*), the memory allocation will

1. search for a memory block in freedMBList (i.e., a memory block allocated previously and deallocated later) which is suitable according to a particular allocation strategy (detailed below). If the memory block is available, store the information in the memory block, update both freedMBList and allocMBList; if not, go to 2.
2. allocate a new memory block using *sbrk* or *brk* to store the information; update the allocMBList accordingly.

Memory Deallocation: Given a piece of information currently stored in a memory block, when it needs to be deallocated, the content currently in the memory block will be deleted. This can be done by deleting the corresponding node from allocMBList and adding a new node in freedMBList.

> *Note the memory blocks in freedMBList should be merged into one memory block if they are consecutive in memory.*

Allocation Strategies: As a part of memory allocation, you will need to implement the following allocation strategies:

- *first fit:* find the first memory block in freedMBList that is big enough; you split the block and return to the user a chunk that is big enough to meet their needs. Then, you add the rest of the block back onto freedMBList.
- *Best fit:* find the memory block in freedMBList whose size is the closest match to the allocation need; you split the block and return to the user a chunk that is big enough to meet their needs. Then, you add the rest of the block back onto freedMBList.
- *Worst fit:* find the largest memory block in freedMBList; you split the block and return to the user a chunk that is big enough to meet their needs. Then, you add the rest of the block back onto freedMBList.

Experiments and Discussions: Implement the memory allocation and deallocation as required above for different allocation strategies. Run them on two data sets: **first-names.txt** and **middle-names.txt** (downloadable from Assignment 2 on Canvas, originally from https://github.com/dominictarr/random-name)

[1] Note deallocation means the content in the memory block is deleted so that the memory block can be used to store other information. It is not relinquishing the memory block to system.

In each data set, each line contains a name. Each name is a piece of information which needs to be stored in memory. You should follow the introductions below:

1. For the first 1000 names, directly allocate memory blocks and update allocMBList,
2. Randomly delete 500 names, update allocMBList and freedMBList,
3. Then, read from the data set the next 1000 names; for each name, allocate memory blocks following “Memory Allocation” above; update allocMBList and freedMBList.
4. Randomly delete 500 names, update allocMBList and freedMBList,
5. Repeat 3 and 4 until the end of the data set.

Run each allocation strategy on each data set and your code should save the following information in an output file at the end of processing:

a. The total memory allocated using *sbrk* or *brk.*
b. The starting address and size of all nodes in freedMBList, each node per line.
c. The starting address, size and content of all nodes in allocMBList, each node per line.

Discuss what is the optimization objective of allocation strategy. Compare the collected information, can you conclude which allocation strategy is better? Explain why using the collected information as evidence to support your arguments.

4. Submission Requirement and Marking Guide

Source Code, Makefile and README (30 marks)

- The submission is a single zip file “COSC1114/1112-A2-Code-Sxxxxx.zip” which should include: *source code(s)*, *Makefile*, and *README* (Sxxxxx is your student id). (1 mark)
- README file should briefly introduce each source code file, and present how to compile the source code and how to execute the code on each data set. (1 marks)
- The input and output file name are not hardcoded (1 mark)
- The executive code can be generated by running the Makefile where -Wall option is required. This is something you should have learnt how to do in your previous studies. (1 marks)
- The source code(s) should contain enough comments to explain the major steps (1 marks)
- Implementation Correctness:
  (i) The source code(s) can be successfully complied and executed according to README. (3 marks)
  (ii) On the basis of full mark in (i), the source code regarding how the memory information is managed using the two lists (i.e., allocMBList and freedMBList as instructed in “Assignment Tasks > Experiments and Discussions > 1-5”) is correct (12 marks)
  (iii) On the basis of full mark in (i)(ii), the outcomes are reasonable/correct. (10 marks)

Important Marking Information for Source Code, Makefile and README

— The source code regarding how the memory information is managed using the two lists (i.e., allocMBList and freedMBList as instructed in “Assignment Tasks > Experiments and Discussions > 1-5”) is incorrect or has major flaw in logic, you will have up to 10 marks out of the 30 marks in this section.

— If the source code regarding the first (or worst/best) fit allocation strategy is incorrect or has major flaw in logic, you will lose 5 marks for each strategy.

Written Report: (10 marks)

The written report will include the source code, experiment results and discussions on them. You must submit your report via the provided Turnitin submission link.

- The written report must be named "COSC1114/1112-A2-Report-Sxxxxxx.pdf" where Sxxxxxx is your student id. (1 mark)
- It should include a cover page and a table of contents with clear headings. If needed, you can include well formatted tables and graphs in the report. (1 mark)
- Include all source codes in the written report (not image such as screenshot) (1 mark)
- Report the experiment results instructed by "Assignment Tasks > Experiments and Discussions > 1-5" for each allocation strategy on each data set. (6 mark)
- All materials outside of provided course materials that are accessed must be referenced in a bibliography. (1 mark)

Important Marking Information for Written Report

- If the conclusions and discussions lack support of experimental evidence but is based on solid research, you will have up to 5 marks in written report.
- if the conclusions and discussions have major flaw, you will have up to 5 marks in written report.

5. Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarized, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods,
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviors, including:

- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students

For further information on our policies and procedures, please refer to https://www.rmit.edu.au/students/student-essentials/rights-and-responsibilities/academic-integrity