CP386: Assignment 4 – Spring 2022

Due on July 20, 2022

Before 11:59 PM

This is a group assignment (group of 2) and a GitHub repository would be used to manage the code development. In this assignment, we will try to practice the concept of deadlock avoidance and contiguous memory allocation.

General Assignment Notes:

For collaborating and managing the source code on GitHub

- You must create a repository on GitHub (make it public only after the assignment submission deadline passed), where you and your teammate would collaborate on this code development.
- You must maintain the README.md file for the repository to explain the following:
  - Project title
  - Makefile
  - Screenshots
  - Individual contribution (Function-wise)
  - Features
  - Tests cases (if any)
  - Use examples
  - About Developers
  - License to use your code

- Marks for you and your teammate's contribution to the project would be allotted based on the history of the commits to the GitHub. If no commit is found from your login ID, then a zero would be awarded to you.

Even if you are creating the GitHub project, you must submit the source code file to Myls page. For submitting your assignments to Myls page follow these requirements:

- Name your source code file as: combination of your Laurier ID number & your teammate's ID, an underscore, 'a' (for 'assignment'), then the assignment number in two digits. For example, if the user 100131001 and 100131233 submits Assignment 4, the name should be `100131001_100131233_FILE_NAME.c.txt`. No other file name format will be accepted. We require the .txt extension in the end because MyLS does not allow .c extension.
- Include in source code file: your and your teammate's GitHub login ID and the URL of the GitHub repository used for your project/algorithm development.
- For this assignment, you must use C99 language syntax. Your code must be compiled using make without errors. There will be a small bonus if it compiles without warnings. You must create with a make file and mention instructions to use it on GitHub page.
- Test your program thoroughly with the `gcc` compiler (version 5.4.0) in a Linux environment.

- If your code does not compile, then you will score a zero. Therefore, make sure that you have removed all syntax errors from your code.

Note:

1. GitHub repository is a must for this assignment. If you are not maintaining it, then you will score a zero in Assignment 4. Further, marks will be deducted for any of the questions in which these requirements are not fulfilled.
2. This is a group assignment and for creating groups, the instruction would be provided on Myls.

Question 1 (Marks: 60):

- In this program, you would be using different algorithms (first fit, best fit, worst fit) for contiguous memory allocation (refer to section 9.2 from text). This project will involve managing a contiguous region of memory of size MAX where addresses may range from 0 … MAX − 1. Your program must respond to three different requests:

    - Request for a contiguous block of memory
    - Release of a contiguous block of memory
    - Report the regions of free and allocated memory

- Your program will allocate memory using the first fit, best fit, and worst fit algorithm. This will require that your program keep track of the different allocations and holes representing available memory. When a request for memory arrives, it will allocate the memory from one of the available holes based on the allocation strategy (for the first allocation, all memory is available). If there is insufficient memory to allocate to a request, it will output an error message and reject the request
- Your program will also keep track of which region of memory has been allocated to which process. This is necessary to support the `Status' command and is also needed when memory is released via the `RL' command, as the process of releasing memory is passed to this command. If a partition being released is adjacent to an existing hole, be sure to combine the two holes into a single hole.
- The program should be invoked by passing the size initial amount of the memory via the command line. You would invoke your program as follows:

```
./allocation 1000000
```

- The program would run a loop (user enters Exit to stop its execution) and would take the user to enter commands for responding to a request of contiguous memory block allocation, the release of the contiguous block of memory, status/report of the regions of free and allocated memory blocks. The program should take the following commands:
    - RQ <process number> <size> <B>: 'RQ' command is the new process that requires the memory, followed by the amount of memory being requested, and finally the algorithm. Use the following flags for three approaches:
        - "F" for first fit algorithm
        - "B" for best fit algorithm
        - "W" for worst fit algorithm
    - RL <process number/name>: 'RL' command will release the memory that has been allocated to a process (e.g., P0).

- C: The `C’` command is used to compact the set of holes into one larger hole. For example, if you have four separate holes of size 550 KB, 375 KB, 1,900 KB, and 4,500 KB, your program will combine these four holes into one large hole of size 7,325 KB. There are several strategies for implementing compaction, one of which is suggested in Section 9.2.3. Be sure to update the beginning address of any processes that have been affected by compaction.
- Status: The `‘Status’` command used for reporting the status of memory is entered.
- Exit: The `‘Exit’` command is used to exit the loop and the program.
- A request for 20,000 bytes will appear as follows:

  ```
  command>RQ P0 20000 B
  ```

  The first parameter to the RQ command is the new process that requires the memory, followed by the amount of memory being requested, and finally the strategy. (In this program, “B” refers to the best-fit algorithm.)
- Similarly, a release will appear as:

  ```
  command>RL P0
  ```

- The other implementation details are at your discretion, and you are free to explore.
- You must write all the functions required to implement the contiguous memory allocation algorithms. Complete the program as per the following details so that you can have functionality as described above. Write all the code in a single C file.
- To run the code run command <make run> that would initialize the program with 1 MB (1,048,576 bytes) of memory.

The expected output is given as below:

```
$ make run
gcc -std=gnu99 -o allocation allocation.c
./allocation 1048576
Allocated 1048576 bytes of memory
allocator>RQ P0 200000 B
Successfully allocated 200000 to process P0
allocator>RQ P1 350000 B
Successfully allocated 350000 to process P1
allocator>RQ P2 300000 B
Successfully allocated 300000 to process P2
allocator>RL P0
releasing memory for process P0
Successfully released memory for process P0
allocator>Status
Partitions [Allocated memory = 650000]:
Address [200000:549999] Process P1
Address [550000:849999] Process P2

Holes [Free memory = 398576]:
Address [0:199999] len = 200000
Address [850000:1048575] len = 198576
allocator>RQ P3 120000 B
index = 0 delta = 80000 best delta = 1048577
index = 1 delta = 78576 best delta = 80000
Successfully allocated 120000 to process P3
```

```
allocator>Status
Partitions [Allocated memory = 770000]:
Address [200000:549999] Process P1
Address [550000:849999] Process P2
Address [850000:969999] Process P3

Holes [Free memory = 278576]:
Address [0:199999] len = 200000
Address [970000:1048575] len = 78576
allocator>RQ P4 150000 B
index = 0 delta = 50000 best delta = 1048577
Successfully allocated 150000 to process P4
allocator>RQ P5 80000 B
No hole of sufficient size
allocator>Status
Partitions [Allocated memory = 920000]:
Address [0:149999] Process P4
Address [200000:549999] Process P1
Address [550000:849999] Process P2
Address [850000:969999] Process P3

Holes [Free memory = 128576]:
Address [150000:199999] len = 50000
Address [970000:1048575] len = 78576
allocator>C
Compaction process is successful
allocator>Status
Partitions [Allocated memory = 920000]:
Address [0:149999] Process P4
Address [150000:499999] Process P1
Address [500000:799999] Process P2
Address [800000:919999] Process P3

Holes [Free memory = 128576]:
Address [920000:1048575] len = 128576
allocator>
```