CG2271 Real Time Operating Systems
2013/14 Semester I
Term Assignment

1. Introduction

   In this term assignment you will implement a memory manager with three memory management policies – first fit, best fit and worst fit – and assess the policies on their run-time and memory fragmentation properties.

   You are provided with several source files:

   main.c – Contains the testing routines for your memory manager.
   memmang.h – Header file to configure your memory manager.
   memmang.c – Where you will implement your memory manager.
   linkedList.h – Header file for linked list routines
   linkedList.c – Linked list routines to manage memory manager information nodes.
   nodes.h, nodes.c – Constant-time node allocation algorithms.

   You will work on this assignment in your regular lab team of 2.

2. Deliverables and Deadline

   You will submit the following files, zipped up in a single archive called Axxxxxxx.zip (where Axxxxxxx is the matric number of one memory of your team) containing:

   a. A project report using the format provided in cg2271assgrpt.doc.
   b. Your memmang.c file. All implementation must be done in this file. DO NOT include any other source file. If you do they will be deleted and ignored.

   Please upload your zip file into the "Assignment Submission" folder on IVLE by 2359 on 22 November 2013.

   As 22 November 2013 falls in reading week, you are advised to complete this assignment by 15 November.

3. Development Environment

   You are free to use any development environment including gcc, Visual Studio or Xcode. Note that to compile this code properly you will need to compile in ALL the provided C files, and have all the provided header files in the same directory as the C files.

4. The Assignment

Before attempting this assignment you should revise Lecture 7, in particular pages 26 to 36.

The linkedList.h and linkedList.c files implement linked lists to manage the free and allocated memory as shown in the lecture notes. The lists are sorted in starting address order to facilitate coalescing. You are provided with:

a. The TMemoryNode data structure.

This data structure maintains information about free and allocated nodes, and the linked list is maintained in starting-address order. The fields of this data structure are:

startAddress: The starting address of the memory segment.
len: The length of this segment in bytes.
allocated: 0 if this segment is free, 1 if it has been allocated.
prev, next: Double-linked list pointers.

b. The functions shown in the table below:

Function Name	Parameters	Returns
newNode	Starting address of memory segment. Length of memory segment in bytes. Whether the memory segment is allocated (1) or not (0)	Pointer to a new TMemoryNode structure.
insertNode	Node to be inserted into the linked list.	Nothing. Node is inserted into the memory management linked list.
testAdjacent	Node to conduct adjacency test on.	Tests if location and location->next are adjacent nodes and whether location->next is allocated. Returns 0 if location and location->next can be coalesced, false otherwise.

mergeNodes	Node to coalesce with neighbor	location and location->next are coalesced

The memmang.c file contains three routines that you need to complete. This is the only file you need to modify for the entire assignment. The three routines that you must complete are:

Function Name	Parameters	Returns
findFreeMemory	# of bytes of free memory required	Pointer to TMemoryNode describing free memory larger than or equal to # of bytes of free memory required.
NSAlloc	# of bytes of free memory required	Returns starting address of free memory of the length specified in the parameters, or NULL if there is no more free memory to allocate (or if the amount of free memory left is insufficient)
NSFree	Starting address of memory segment to free.	Memory segment is freed, memory management linked list is duly updated, with coalescing if possible.

Your assignment is to complete these three functions. In particular findFreeMemory should located free memory using the three memory allocation strategies – first fit, best fit and worst fit.

This routine has been divided up into three sections. Implement each policy in its respective section.

Verify that your implementations are correct. You may have to write your own test-routines.

5. Experiment 1 – Running Times

   To run the first experiment:

   a. Open the memmang.h file for editing.
   b. Locate the #define called "EXP_TYPE". This defines the type of experiment to run.
   c. Set the value to RUNTIME.
   d. Locate the #define called "MEMMANG_TYPE". This defines the type of memory allocation to use. Set this to MEMMANG_FIRSTFIT.
   e. Compile your program and run it. Locate the lines "Average Time" and "Variance" and record the readings in your report.
   f. Repeat step e. 4 more times (i.e. 5 times in total). Record your readings in the report.
   g. Set MEMMANG_TYPE to MEMMANG_BESTFIT. Repeat steps e. and f.
   h. Set MEMMANG_TYPE to MEMMANG_WORSTFIT. Repeat steps e. and f.

Based on your recordings answer the questions in the Experiment 1 section of your report.

6. Experiment 2 – Fragmentation

   To run the second set of experiments:

   a. Set EXP_TYPE to ALLOCFREE
   b. Set MEMMANG_TYPE to MEMMANG_FIRSTFIT
   c. Compile the program and run it. Locate the lines "Total Free: xxx Largest Block: yyy Fragmentation: zzz" (where xxx, yyy and zzz are the actual readings)
   d. Record xxx, yyy and zzz in your report.
   e. Set MEMMANG_TYPE to MEMMANG_BESTFIT. Repeat steps c. and d.
   f. Set MEMMANG_TYPE to MEMMANG_WORSTFIT. Repeat steps c. and d.

   Based on your recordings answer the questions in the Experiment 2 section of your report.