7th Homework problem for Operating Systems - Buddy Buffers

This assignment is worth 40 points. Note that late work will not be accepted for this program and no other assignments will be accepted after this date. Also note that this program is a lot more work than any of the others – do not put starting on it!

Make a simple buddy buffer manager with 4 functions for an operating system and a test driver.

It will give out buffers from a fixed size area. The available memory for the buffer manager will be initially set to 10 of the maximum buffer sizes (511 words)

It will have a function which gives out buffers in sizes (which are powers of 2 less 1) from 7 words to 511 words. The other word in the buffer will be a control word for use by the buffer manager. The requester will provide a block size and the routine will return an address of a buffer (probably for most of you a relative address from the start of an array holding the buffers) or -1 (cannot provide a buffer of that size because of lack of space). Note that the routine may need to split a buffer into two buddies to get a proper sized buffer. This splitting may need to cascade. For example if there are only 511 word blocks available, to get a 7 word block, a 511 word block will have to be split into two 255 word buddy blocks. Then one of the 255 word buddy blocks will have to split into two 127 word buddy blocks, etc. down to two 7 word buddy blocks, one of which will be given to the requester. The returned address will be the word after the control word. The first call to request a buffer will be required to call your initialization routine to set up the linked lists defined below. Another thing to check is a request for illegal block sizes – if you get a request for 700 words you need to return an error status (a -2 for illegal request). If, for example, you get a request for a 5 word block, you have an option (but you must be consistent), you can either return a pointer (or index) to a 7 word block or return an error status (a -2 for illegal request).

It will have another function which the user uses to returns the buffers. The only input will be an address. No status will be returned. This routine must recombine buffers that are less than the maximum size if it can. Note that this recombining may cascade, that is two 7 word buffers may combine into a 15 word buffer which intern could combine with another 15 word buffer into a 31 word buffer, which may or may not combine with another 31 word buffer. This combining could go all the way up to a 511 word buffer, but 511 word buffers are never combined.

The routine must set a flag when the buffer pool’s memory is tight (less than 2 maximum sized buffers exist). It must clear this flag if the buffer pool is no longer tight.

Part of the control word is to contain a size of the buffer. Another part of the control word contains the address of the next buffer of that size in the buffer pool (or 0/null in there are none). You will need pointers to the first one of these. Initially the only buffers will be the maximum sized ones. As stated above, larger buffers may need to be split to provide smaller ones. After the first splitting happens, there will be chains of various sized buffers which will grow and contract as requests for buffers are made and other buffers are returned.

A third function will return a tight / OK status of the buffer pool. Remember tight is when there are two or fewer maximum sized buffers (there may be lots of small buffers, and even if the sum of these small buffers exceeds two maximum sized buffers, tight is to be returned).

A 4th function will be a debug function that will return the number of buffers in each buffer chain.

You will also need a driver that tests out all of the functions and prints results as it goes. The first thing this test program should do is call the status and debug status functions and then print out the expected values (not tight and ten 511 word buffers) and the actual values. Then request a 700 word block and verify that you get an illegal request status back (-2). The next thing it should do is request a 7 word buffer, verify it got it and then call the status functions again. It should then print out the expected (not tight, nine 511 word buffers, and one of each of the other sizes) and the actual status. Then return the buffer and print out the expected (not tight, ten 511 word buffers, and no other buffers) and actual status. Next in a loop request ten 511 word buffers and check out output statuses. Then request another buffer of any size (should get a -1 status). Return the buffers and output expected and actual status. Next make up some other tests to verify the proper functioning of your program. Make sure you check that no improper combining of non-buddy buffers occurs and that proper combining does.

I will need to see the code you have written (test driver as well as buffer manager code) as well as the printout from your test driver (the test printout should go to a file and the file is what I want to see). You will need to make sure you test all of the code by using up buffers and make sure recombine works etc. by outputting the results from the status and debug functions as described above. Writing a good test routine may be just as difficult as any of the other functions. Place your buffer manager (the source code), your test driver (the source code), a data file with a sample run output, and an observations in the Dropbox for assignment 6.

Note that languages with no pointers like Java will not work too well to do this assignment, C / C++ and some other languages will work nicely. For that reason if you are using Java or a language with no pointers or if you just do not like using pointers, you can return an offset from the start of your buffer pool in place of a pointer.

Please put comments at the top of your program with your name, the date, the assignment number, and a brief description of the program. I also want to see comments within the program.

Note, if you need more space for your control data, you may use two words instead of one. In that case, your buffer sizes will be: 6, 14, 30, 62, 126, 254, and 510.

Notes:

1. Do not send me screen shots of your output, it must be to a “\*.txt” file.

2. The testing is to be done by a test driver, do not have the user input data via a menu.

3. This problem is much harder than any of the others, do not put it off.

4. Your test of the program must demonstrate that it works!

5. Zip the files together and put the zip file in the DropBox for assignment #7.