CS 503 - Fall 2014

Lab 4
Memory Management
Aligned getmem

Due Sunday, October 5th, 2014 at 11:59 PM

Objectives:

By the end of this lab you will be able to:

- Understand how to allocate heap memory using getmem
- Create a new system call aligned_getmem which allocates heap space which is aligned on a particular byte multiple.

Background

Today XINU contains a function for allocating memory from the heap: getmem which takes as input a size in bytes and allocates a contiguous block of memory by scanning the memlist structure (also referred to as the free list) to find the first region of usable memory that is large enough to satisfy the requested size. The getmem function then returns the address of the memory block. While the caller of getmem is guaranteed to either receive a memory address or SYSERR, he/she has no control over the memory address returned. In certain instances, there maybe a requirement that the address of the allocated memory be *aligned* to a multiple of a particular number. For example, for loading and storing integers in memory it is typically faster if the integers are stored at a multiple of the word size of the machine.

Your job is to create a new system call: aligned_getmem with the following prototype:

```
char    *aligned_getmem(uint32 bytes, uint32 alignment);
```

which allocates *bytes* bytes of memory at an address that is a multiple of *alignment*.

Setup

In /homes/cs503/xinu there is a file called xinu-fall2014-lab4.tar.gz that contains a start to the code. Unpack:

```
tar zxvf /u/u3/cs503/xinu/xinu-fall2014-lab4.tar.gz
```

This will create a directory called xinu-fall2014-lab4.

Along with the main code for XINU, this tarball contains the following files (additional explanation of the contents of the files is in the following sections).

- system/aligned_getmem.c - function declaration for the aligned_getmem function.
- include/prototypes.h - modified prototypes.h file which includes the aligned_getmem function.

What's in system/aligned_getmem.c

The aligned_getmem.c file contains an empty function declaration for the aligned_getmem function. Your job is to fill in code to allocate memory at an address of specified alignment or return SYSERR if the allocation cannot be performed.

Additional Requirements:

- The value of *alignment* must be a power of 2 and the value of *bytes* must be a multiple of alignment. The aligned_getmem function should return SYSERR if the parameters are not correct.
- If the value of *alignment* or *bytes* is less than 8, the values must be rounded up to 8.
- If the value of *alignment* is less than 8 but not a power of 2 (e.g. 6) then return SYSERR. So for values of less than 8 the behavior should be:
  - 0 - return SYSERR
  - 1 - round up to 8
  - 2 - round up to 8
  - 3 - return SYSERR
  - 4 - round up to 8
  - 5 - return SYSERR
  - 6 - return SYSERR
  - 7 - return SYSERR
- Make sure to check that the number of bytes is a multiple of alignment after doing all necessary rounding.
  For example:
  - The value for passed in for *alignment* is 2 and the value passed in for *nbytes* is 20.
  - The value 20 is a multiple of 2 but the value 2 for *alignment* is rounded to 8.
  - After rounding 20 is no longer a multiple of *alignment* since 20 is not a multiple of 8.
  - If this happens return SYSERR
- The number of bytes and the value for alignment must be greater than 0. Return SYSERR if either is 0.
- Make sure to remove all debug output from your system calls. When the TAs run your submitted code, calling a system call directly should not produce any output.
- Provide a set of test cases to ensure that your code works as required. Put these test cases in main.c
- The TAs will be replacing main.c with their own test cases after running your submitted test cases. Make sure you do not define any dependent variables in main.c
- If your submitted code does not compile (either the exact submitted code or the code after the TA's replace any test case files), you will receive zero (0) points for code execution. If this happens, you will be allowed to resubmit for half credit only.

- Please run "make clean" prior to submission so that you don't submit object files
- NOTE: When you make xinu for this lab the make file will generate two files in the compile directory:
  - xinu - this is the file you will download to the xinu backend (you will NOT use xinu.xbin)
  - xinu.elf - this is the executable and linkable format version of the xinu binary. This is not the format to use when sending to the backends. It is useful for low level (assembly) debugging only.
- Some tips on writing an affective report can be found here.

Grading Criteria

The lab will be graded out of 100 total points:

- 25 Points - How well your test cases verify that your code has met the requirements.
- 50 Points - The results from the TA's test cases.
- 25 Points - How well your report answers the discussion questions and your test case explanation.

What to turn in

Submit using turnin command your complete source code (all of XINU) including the any files you added to complete the lab. Please make sure to remove all printed debug output and run make clean prior to submission. In the 'system' directory include a PDF file with a write up discussing:

- The details behind your implementation. As part of this discussion write answers to the following questions:
  - How does your solution ensure allocation of memory in an aligned manner?
  - Describe your test cases. How to they ensure that your code correctly meets the requirements?
  - Does it make sense to allow an *alignment* value that is not a power of 2, of so why if not why not?
  - The getmem system call is very similar to the POSIX malloc system call. In POSIX, there is a corresponding free system call which performs a very similar operation to the freemem system call in XINU, however the freemem system call in XINU requires the caller to specify not only the memory address to free, but also the size of memory to free (take a look at system/freemem.c). The POSIX free does not require the caller to specify the size (only the address). Why does XINU require the size to be specified when freemem is called? What modifications would need to be made to XINU to not require the caller to specify a size when the call freemem?

To turn in your lab use the following command

```
turnin -c cs503 -p lab4 xinu-fall2014-lab4
```

assuming xinu-fall2014-lab4 is the name of the directory containing your code.

If you wish to, you can verify your submission by typing the following command:

```
turnin -v -c cs503 -p lab4
```

Do not forget the -v above, as otherwise your earlier submission will be erased (it is overwritten by a blank submission).

Note that resubmitting overwrites any earlier submission and erases any record of the date/time of any such earlier submission.

We will check that the submission timestamp is before the due date; we will not accept your submission if its timestamp is after the due date. Do NOT submit after 11:59 PM Eastern Time.