More System Calls
In order to allow user programs to use printf() and malloc(), you need to implement all of these system calls: ioctl(), fstat(), getpagesize(), sbrk(). Use the man pages and google to answer the following.

1. What command do you use to pull up the ioctl() man page? _________________________

2. What are the argument types passed into fstat()? _________________________________

3. What does sbrk() return if there is NOT an error? ______________________________

4. Can the argument passed to sbrk() be negative? ______________________________

Memory
In KOS, memory for user processes is main_memory[MemorySize]. In other words, it is a big array. In Lab 1, we only dealt with a single user process, so we gave it the entirety of main_memory to work with. In Lab 2, we are going to divide main_memory into eight chunks so that we can have eight different user processes.

5. Fill out the following chart in terms of MemorySize, main_memory, x, y, and z.

<table>
<thead>
<tr>
<th>User Addr</th>
<th>Kernel Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>proc1-&gt;base</td>
<td>0</td>
</tr>
<tr>
<td>buf1</td>
<td>x</td>
</tr>
<tr>
<td>proc2-&gt;base</td>
<td></td>
</tr>
<tr>
<td>buf2</td>
<td>y</td>
</tr>
<tr>
<td>proc3-&gt;base</td>
<td></td>
</tr>
<tr>
<td>buf3</td>
<td>z</td>
</tr>
</tbody>
</table>

A

B

C

What are the values of proc1->limit, proc2->limit, and proc3->limit?
Execve, Fork, Wait, Exit

In order to create and manage multiple processes, you will need to implement execve(), fork(), wait(), and exit().

6. What does execve() do? _______________________________________________________________

7. Look at the exec.c test (it’s in /cs/faculty/rich/cs170/test_execs/). When you run ./kos -a exec

   What is the filename of the first program being loaded into memory? ____________________

   What is the filename of the program being loaded into memory on the execve call? _______

Look at the following program.

parent_program.c :
int main() {
   char* child1[1] = {"child1_program"};
   char* child2[1] = {"child2_program"};
   int ret = fork();
   if (ret == 0) {
       execve(child1[0], child1);  
   }
   int ret2 = fork();
   if (ret2 == 0) {
       execve(child2[0], child2);  
   }
   int status;
   wait(&status);
   wait(&status);
   exit(0);
}

8. Who is executing each line? parent

9. How many programs are loaded into memory when parent_program runs?_____________________

10. Why does the parent call wait() twice?___________________________________________________

11. Who is guaranteed to exit last?_________________________________________________________