PA3 - Part 2 - Test Case Document

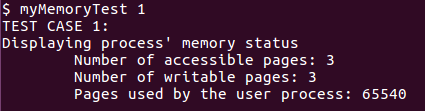
Paolo Ratti Tamayo - A20352801

Jiateng Sun - A20368601

The system call myMemory(), displays the number of pages that are accessible, writable, and the current physical memory usage of the current process.

To test the system call myMemory() we created 8 different test cases.

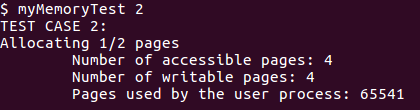
Test Case 1



We call the myMemory() system call.

This tests the functionality of myMemory().

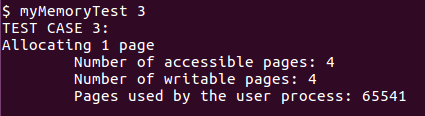
Test Case 2



First, we allocate 2048 bytes, which is half the size of a page. Then, we call myMemory().

This tests that when allocating memory that is less than the size of a page, we are still allocating a full page.

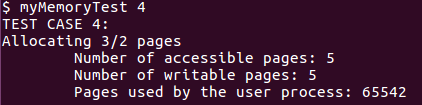
Test Case 3



First, we allocate 4096 bytes, which is exactly the size of a page. Then, we call myMemory().

This tests that when allocating memory that is exactly the size of a page, we are only allocating a page.

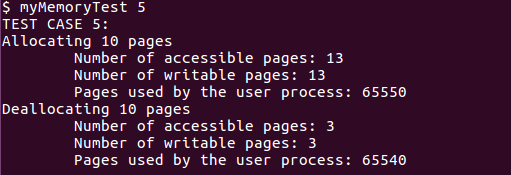
Test Case 4



First, we allocate 6144 bytes, which is the size of a page and a half. Then, we call myMemory().

This tests that when allocating memory that is larger than the size of a page. More pages will be created accordingly.

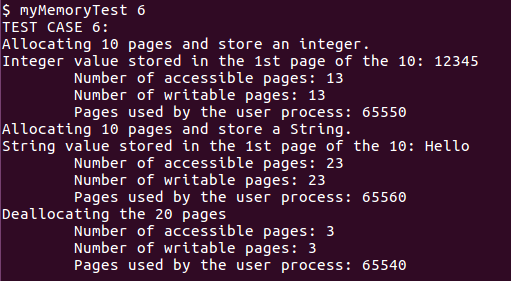
Test Case 5



First, we allocate 10 pages and call myMemory(). Then, we deallocate the 10 pages and call myMemory again.

This tests that myMemory() works with both allocation and deallocation.

Test Case 6



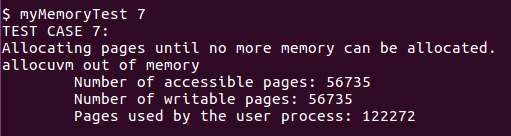
First, we allocate 10 pages and store an integer at the beginning of the allocated space and call myMemory().

Then, we allocate 10 more pages and store a string at the beginning of the second allocated space and call myMemory().

Finally, deallocate the 20 pages and call myMemory().

This tests that myMemory() will work even when different types of data are inserted into the allocated pages.

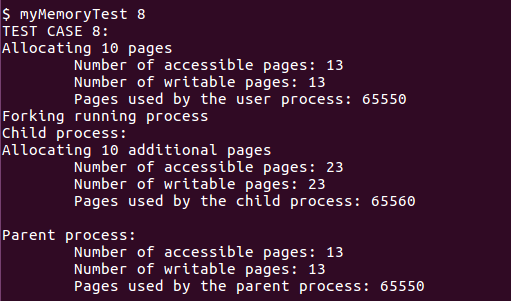
Test Case 7



We iteratively allocate pages until no more memory can be allocated. Then we call myMemory().

This tests that myMemory() can display memory usage when a process’s heap is full.

Test Case 8



First, we allocate 10 pages and call myMemory(). Then, we fork the running process, and call the wait system call in the parent process. In the child process, we allocate an additional 10 pages and call myMemory(). After the child process has finished execution, the parent will resume execution and call myMemory().

This tests that myMemory() can display the different memory usages from multiple processes.

**Explain why do you only use those test cases**

We are only using these test cases, because we consider that these test cases successfully cover most of the ways in which memory can be allocated and deallocated. This should be enough to prove that myMemory() is properly displaying the memory usage of a process.