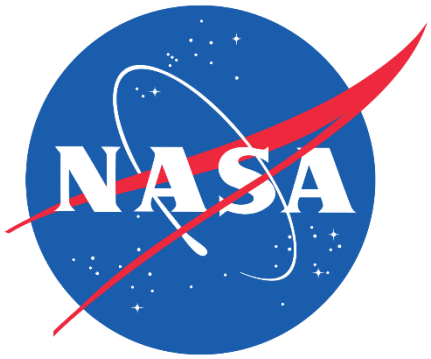
NASA KSC Intern Final Report – 2017-11-14





# Command and Control Software Development
# Memory Management

Student Name: Austin Pope Joseph
Academic Level: Senior
Academic Major: Electrical Engineering
Academic Minor: Mathematics
Academic Institution: Portland State University

Mentor Name: Jamie Szafran
Mentor Job Title: Computer Engineer, AST
Org Code/Branch: NE-XS/Software Branch
Division: NE-X Exploration Systems & Operations Division

## I. Nomenclature

GSDO – Ground Systems Development and Operations
KSC – Kennedy Space Center
COTS – Commercial Off the Shelf
CSCI – Computer Software Configuration Item
CIS – Continuous Integration Server
SCM – Software Configuration Management (application)
Mock File – Files that contain code meant to emulate methods in libraries the system under test would call.
Directly Lost (memory leak) – When the pointer to a block of allocated memory goes out of scope without deallocating the memory it pointed to.
Indirectly Lost (memory leak) – Similar to directly lost except the pointer points to an object that used to be pointed at the lost memory block.
Possibly Lost (memory leak) – The result of an interior pointer. The memory management program can't be certain the memory was handled correctly.
malloc()/new – Methods to dynamically allocate memory in C and/or C++.
free()/delete – Methods to deallocate dynamic memory created by malloc()/new.

## II. Introduction

This internship was initially meant to cover the implementation of unit test automation for a NASA ground control project. As is often the case with large development projects, the scope and breadth of the internship changed. Instead, the internship focused on finding and correcting memory leaks and errors as reported by a COTS software product meant to track such issues.

Memory leaks come in many different flavors and some of them are more benign than others. On the extreme end a program might be dynamically allocating memory and not correctly deallocating it when it is no longer in use. This is called a direct memory leak and in the worst case can use all the available memory and crash the program. If the leaks are small they may simply slow the program down which, in a safety critical system (a system for which a failure or design error can cause a risk to human life), is still unacceptable.

The ground control system is managed in smaller sub-teams, referred to as CSCIs. The CSCI that this internship focused on is responsible for monitoring the health and status of the system. This team's software had several methods/modules that were leaking significant amounts of memory. Since most

of the code in this system is safety-critical, correcting memory leaks is a necessity.

## III. Objectives-

The objective of the internship was to find and correct memory handling errors. While that may seem fairly straightforward, many of the errors that were encountered were very difficult to find and, once found, they were even more difficult to fix without changing the function of the original code.

## IV. Approach

Ideally, the intern would pull reports from the CIS, which will include detailed memory management error reports. The CIS provides a formatted report that will show what kind of memory error the particular method/module is producing as well as where the memory is being allocated and a history of where it is being utilized. The intern would then locate the files implementing the code and study them, looking for code that isn't handling memory optimally. Once the code is located, the intern would correct it, test it in the development environment, have it peer reviewed, and finally promote it to the active development code in the SCM. Due to the scope of possible error sources, this process can be incredibly difficult.

### The CIS, Memory Management, and Finding Errors

The first step is to find the memory leak. There are four common types of memory leaks: directly lost, indirectly lost, still reachable, and possibly lost (as reported by our memory management software).

Directly lost and indirectly lost are the most serious kinds of leaks. The most common cause of these problems is the use of malloc() or new without the corresponding use of free() or delete to deallocate the memory when it is no longer in use. Since many methods create new objects by design it is not as simple as deleting the object when the method goes out of scope. The function that called for the creation of an object will produce an error when it attempts to access the deleted object. Usually, in these cases the intern would need to rationalize when the last time these objects were likely to be used and make certain that the code that handles that is correctly deleting them.

The other common case for directly and indirectly lost memory leaks is when an object is created and then, before it is used, some kind of exception or error is thrown and the method creating the object terminates without cleaning up the memory. To correct these it is often enough to add a control statement that deallocates the memory on a failure state.

Still reachable and possibly lost errors are much more difficult to track down. While throughout the internship many of these errors were cleared up, there was no good practice to find them methodically. The best practice seems to be: parse through all of the code that is using the methods and all of the cursory code that is being called in said methods and pay attention to pointer assignment.

**Testing and Code Reviews**

The development environment has all of the tools required to build the CSCI runtime software, their respective unit tests, and mock files (used for the unit tests). After a memory leak has been corrected, it is important that the corrected code still functions as it was intended and doesn't introduce more errors into the system.

This can be accomplished by building the affected code locally on the computer and running the required memory handling and unit tests. Our memory management software puts out reports in an .xml format and while the CIS has a good XML reader that displays this information nicely, the best way to check these reports on a development machine is to load the report into a web browser. To streamline the testing process I wrote a shell script that accepts a test suite binary and outputs a ***.OUT file that can be quickly loaded into Firefox.

Once the code has been tested, the unit test report is clean, and the memory error report is clean, it is time to start a code review. At this point the code changes are checked into our code review software, the reports are uploaded, and one or more reviewers are chosen to inspect the changes. Once the code has been reviewed and accepted it can then be promoted to a higher branch within the SCM.

**Promotion**

This is the final step in the process. Once all of the changes have been reviewed, the intern can now promote the code to the active development build area in configuration management. This requires a work order that must be created/assigned from our project tracking software.

## V. Conclusion

As of November 2nd, I have been successful in reducing the number of directly lost memory leaks by eighty-five, indirectly lost leaks by thirty-three, possibly lost by eight and still reachable errors by twelve. In addition, I have made considerable alterations to code in almost every single module inside the CSCI that was assigned to me.

The most notable code changes involved writing a daemon to handle multi-process test cases and a signal handler to shut down the daemon safely after the tests were concluded. This internship has increased my understanding of the C++ language ten-fold and I have a much better grasp on the general development of a software product.

## VI. Acknowledgments

I could not have gotten as far as I did without the help of James Hinchey. He was an incredible help teaching me how to get the development environment up and running on my development machine as well as an incredible resource to ask technical questions.

My mentor, Jamie Szafran, has always been very available to answer questions that I have had throughout the internship. When she was unable to answer the questions herself, she was very quick to find someone who could; namely, JJ Serrano and Michael Reed.