Command and Control System Software Development

Ricky Velasquez
Kennedy Space Center
Major: Computer Engineering
Fall Session
Date: 17 11 2017

Command and Control System Software Development

Ricky Velasquez[1]

Wichita State University, Wichita, KS 67260

Nomenclature

API	= Application Programming Interface
CPU	= Central Processing Unit
IDE	= Integrated Development Environment
KSC	= Kennedy Space Center
LCS	= Launch Control System
NASA	= National Aeronautics and Space Administration
SLS	= Space Launch System

I. Introduction

Kennedy Space Center has been the heart of human space flight for decades. From the Apollo Program to the Space Shuttle Program, and now to the coming Space Launch System (SLS) and Orion, NASA will be a leader in deep space exploration for mankind. Before any rockets blast off, there is significant work to be done in preparation for launch. People working on all aspects of spaceflight must contribute by developing new technology that has yet to participate in a successful launch, and which can work with technology already proven in flight. These innovations, whether hardware or software, must be tried and true, and includes the projects to which interns contribute to.

For this internship, the objective was to create a data recording system for the developers of a LCS section that records certain messages in the traffic of the system. Developers would then be able to use these recordings for analysis later on, either manually or by an automated test. The tool would be of convenience to a developer as it would be used if the system's main data recorder was not available for tests.

II. Approach

A. Preparation

Before beginning the project, training for different software tools used on the project had to be completed as well as familiarization with the development environment. There were many existing files to review and coding standards to follow. While getting acquainted with everything, the test tool requirements and design specifications were provided. With the project now outlined, the team then started off with pseudo-code to get a better idea of how to approach the recorder.

B. Data Recorder Tool

The outline that was given to us was clear and straightforward. The user needed to be able to start the recorder and stop the recorder. The user could start the recorder with additional arguments that change the data that is recorded but there was to be no interaction between the user and program besides running and quitting the program. Then, the tool was to store all of the recorded data in a file that could be easily parsed for later analysis.

[1] Command & Control System Software Development Intern, NE-XS, Kennedy Space Center. Wichita State University

General Requirements:

- User manually starts and stops the recorder
- User may provide arguments to record certain data
- No user input during runtime
- Should not consume inappropriate amounts of resources
- Record data types with certain attributes
- Take appropriate action when errors occur (disk space, system connection, etc.)

Since there were two people responsible for this project, the other person being Michael Wallace, the work was split evenly. It was agreed that the program would initially be divided into a section for recording the data and another section for storing the data. The data storage section required code that needed to be built from the ground up, whereas the recording section would make use of an existing Application Programming Interfacing (API). These sections would then be integrated towards the end of the internship.

Extracting information required review of the existing API to handle the data. The sheer volume of files made it quite the task. The first objective when reviewing was to understand how each API method works. After reviewing the code and running a test of the API, certain methods were identified to be useful in recording data, as well as interfacing with the system. The data that was being retrieved could be of different types, which determined how the data was separated into different classes. These methods were to be used in the recorder and ensured that the data was of the correct type and properly handled. This process would continue for the remainder of the project as issues emerged, requirements changed, or new things were discovered.

A successful run of the tool would be:

1) User starts the program through the command line
    a) Optional: User can pass in arguments to receive specific data
2) Tool tries to connect to the traffic of the system
3) Once connected, the tool listens to the traffic and retrieves messages
4) Messages are written to file for storage
5) User quits the tool
    a) Disconnect from system
    b) Flush and close files

III. Conclusion

The data recording tool allows the user to record the data from the system's message traffic without any user input after initialization. Having a tool that is more lightweight and requires less user interaction saves the developer time, as time is becoming more valuable as future missions draw near. There are a few improvements that can be made to the tool itself. Examples of such improvements would be refinement of the code so it is less bulky and finding ways to reduce processor usage during runtime. The tool should be a welcome addition for the developers. This project has been my first experience with professional software development and I have learned so many things from the employees and my fellow interns. I have seen rockets launch and land and have seen the wonder it brings to people, old and young. I am glad to say that I made a contribution to NASA and I am leaving having learned new things about spaceflight and all that goes into it. I hope to return to KSC one day, except next time, instead of as an intern, it will be as an astronaut.

IV. Acknowledgements

This has been an incredible experience and I owe it to some very special people who made it possible. First, I would like to thank Caylyne Shelton for taking the time to interview me and considering me for this position. She has been a great mentor, helping me with any questions I had and making sure the interns had everything needed to complete the projects. I would also like to thank Jamie Szafran who was there to help throughout the internship and had cool opportunities, outside of work, for us to participate in. Caylyne and Jamie were very much the dynamic duo. Next, I would like to thank Jason Kapusta who was the technical point of contact for this project. He was there to help with the tool’s progress whenever we needed it. Finally, I would like to thank the interns I worked with as they made every work day one to look forward to. I will always remember my internship at Kennedy Space Center.