Multi-Session Monitoring Research Report

Sam Kaufman
Mentor: Jeremy Enos

August 4, 2016

Abstract

Over the course of the last eight weeks, I have worked with an open source project called asciinema. This software allows a user to record a terminal session and replay it either in a terminal or in a browser. I have built an automated launcher on the Blue Waters network's h2ologin node, and learned a lot about how the program works. I've begun to build a live session streaming capability, as well as a logfile parser that will allow for command recognition and cataloging. I will continue to develop this software going forward, working with the open source developers to create tools that will benefit both the Blue Waters team and the asciinema project.

Methodology

I began this project by comparing five different open source terminal recorders. I ran tests to determine their performance under a range of circumstances that we need them to handle effectively, including resizing the terminal window, using the "screen" command, and using terminal-based text editors (vim, emacs). I also considered the log file format of each, as well as additional features built with the recorder and their specific utility in the context of our needs.

After deciding on asciinema, I went to work sifting through the source code, figuring out what each of the individual commands did and how they interacted with one another. This was challenging at first because, at the time, the project's main branch was in a language called Go. An elegant compiled languange, I spent several days learning the basics of its syntax, data types and class structure. Progress was slow because, though concise and well made, the project is almost entirely devoid of comments. I've learned that this is a common frustration, in particular when working with open source projects.

I reached out to the developers to gather more information and gain some guidance towards our own development goals. About half-way through the SPIN summer program, I discovered that asciinema's main supported branch was switched from Go to an updated Python version. I then pivoted and started

working with the Python source. This made progress move more quickly because I already knew Python, even though I was starting essentially back at square one. After that, I spent the bulk of my time going through the Python source code while reading up on the libraries and functions that were used.

Results

I decided to use asciinema, an open source project available on Github, as our starting point. Asciinema's "rec" command, passed as an argument on the command line (i.e. asciinema rec [filename]) performed the most reliably under the tested conditions. In addition, it outputs the terminal log in a JSON file, which is easily parsible and portable across many different platforms. A big selling point for us is asciinema's additional capabilities.

Its "play" command allows you to replay any logfile generated previously by the "rec" command. This is nifty for short recordings, but I found that it was easy to break and not particularly useful for longer sessions because it lacks any playback controls such as fast forwarding or pausing. The most valuable command for our purposes besides "rec" is the "upload" command, which takes a logfile as an argument (i.e. asciinema upload [filename]) and generates HTML with a robust ASCII-based player. It then automatically uploads this webpage to the asciinema website and generates a unique URL. Though for our security purposes this function will need to be repurposed to hosting on NCSA's own servers, the basic capability of in-browser playback is exactly what we identified as a key feature.

After deciding on asciinema, I implemented an automated launch in .bashrc on the h2ologin node. This silently launches the asciinema record command when I ssh into h2ologin and exits silently when the connection is closed. This is an important feature for eventually implementing our software on the bastion host, because we don't want it to annoy or inconvenience any of our users and especially not our sysadmins.

So far, I have become familiar with the asciinema launcher, recorder and pseudo-terminal scripts. I've made some progress towards building a live streaming capability. However, after discussing with the main developer Marcin, I learned that he is planning on adding a live view command in September. Jeremy and I decided I should start working on a JSON file parser and bash command recognition script to allow their development to progress smoothly. I have since begun building a rudimentary script that parses the JSON log files and compares them to the current user's executables.

Learning Outcomes

This project served as a trial-by-fire for me and my programming knowledge, serving as an incredible learning opportunity. In addition, I found that simply being around the NCSA and talking to staff and my fellow interns led to learning

in many fields unrelated to my project. These outcomes can be organized into a number of different thematic areas, including:

Supercomputer Design

One of my earliest lessons from Jeremy focused on how the network that allows users and administrators to connect into the NCSA's HPC resources are configured. I expanded on this knowledge by studying the Blue Waters network logical map. I've learned about the Gemini network within Blue Waters, as well as the infrastructure that enables scientific computing around the world. By reading through info pages on the Blue Waters portal, I've also learned about all of the different cores that are involved and the file system, to name a few.

Golang

Though it ended up not actually being useful to the project, I did learn and become fairly comfortable with a modern (c. 2007) and elegant compiled language, which I will likely encounter again at some point in my career.

Python

Though I had a working understanding of Python's syntax and some built-in functions and libraries, my knowledge base has expanded by probably an order of magnitude in the last few weeks. Much of my working time has been spent reading man pages on libraries and the corresponding functions that are used throughout the asciinema source code. I've come to understand Python's class structure much better, and have become familiar with a number of libraries that are vital to building applications. Due to asciinema's CLI-based nature, many of these libraries are invaluable for building tools that interact with Unix-based systems.

Unix/Bash

Asciinema interfaces intimately with the Unix command line, and thus I've had to become much more familiar with many different features. I've learned more about stdin/stdout, piping, the file system, processes

Git/Github

Asciinema and a few of the other session recorders are hosted on Github. In order to use the software effectively, I've learned much more about different git commands and version control in general.

Further Work

Going forward, I plan on building out the JSON file parser and eventually integrate it into a SQL database that will catalog logfiles with metadata including user, timeframe, a comprehensive list of commands issued and their associated timing data. I will likely let the main developer create the live viewing capability, then retrofit that and the upload command to our specific needs.