Dominic Furano
January 2014
CS 5959 Writing Solid Code

Through thorough testing using several utilities, I have formed the opinion that tail is a solid utility and suitable for use in important roles as far as stability is concerned. Note that I did not test tail's file monitoring ability. The testing utilities I used were Bart Miller's group's fuzzer, my own fuzzers, the Murphy kernel shim, Valgrind, and Clang's runtime undefined behavior sanitizer. I managed to find the same version of tail from GNU that is used in the CADE lab, version 1.22, so I could do some compiling and testing at home.

I first started testing using Bart Miller's fuzzer. The tail utility didn't crash under this stress. I then tested tail using a fuzzer I wrote which just outputs 1 billion random characters. Tail handled this as well. I wrote a second fuzzer which outputted 1 billion sequential ints. This second fuzzer did not crash the utility, but twice, while I was testing on CADE, the tail process was killed by the kernel. I suspect it was the out of memory killer that did this. My suspicion is mainly based on the observation that I noticed several gigabytes of memory were being used by the tail process while running this test at home. This could be an issue if tail was ever to be used in a situation where it had to reliably work with very large files on the order of gigabytes.

While testing at home using fuzzing, I used a tail binary that I compiled with Clang 3.4 with default optimization the undefined behavior sanitizing flags. At no point were any runtime errors reported.

As a side note on fuzz-testing, ftp is terrible at handling erroneous inputs! It did some weird stuff when subjected to my random character fuzzer including creating a ton of weird files in the testing directory and launching an unrelated application.

For the second round of testing I used the Murphy utility in a cursory way. I didn't have the time to gain an in-depth understanding of the functionality of the tool. That being said I was able to make tail hang with a all the gremlins activated that are located in the gremlin example file. I wont draw any conclusions from this because of my lack of knowledge of what these gremlins mean.

I used Valgrind for my fourth tool. I ran tail through fuzzing and several generated files while using Valgrind and there were no memory leaks. Tail freed all memory allocated and seems pretty solid in that respect.

As far as I can tell, with the testing I have done in the past week, tail seems a fairly robust utility for its intended purpose of reading the end of log files. If files become too large, tail becomes a huge memory hog and unreliable as the operating system may choose to kill it.