Khoa Nguyen
CS 362
01-30-2018

Random Testing Quiz

As I am provided with testme.c, this allows me to look at the code to see the logics that will give an error in the provided program. Based on my understanding, I see that the program starts out with state 0 and will increment its state by 1 every time it sees a matching character at each state level. Once the program reaches state 9, the random string comes to play as the program looks to see if the string is "reset" and that will generate the error. So my objective was to generate random characters and random strings such that the program could produce the error within the given timeframe of less than 5 minutes.

To do so, I started out with inputChar() to generate a random character. I made sure that my random range covers all the ASCII characters that the program was looking for. This was trivial and the program got to state 9 pretty quickly as the given range was small enough. Then, I followed the same logics for generating random strings. As I knew that the program was specifically looking for the string "reset" with all lower case characters, I wrote inputString() such that it would generate a string of length 6 with all randomized lower case characters and null-terminated. As there were 26 options for each position and there were 5 randomized positions, the number of permutations greatly increased as compared to inputChar() mentioned above. This resulted in longer running time before the program produced the error. Overall, it took 662 iterations for the program to get to state 9 and 21,552,889 iterations for the program to reach the error message in one of my test runs. The number of iterations varies greatly between runs due to randomization but the program has always been able to produce the error within 5 minutes on my local machine.

When I tested the running time on flip, the program ran significantly longer probably due to server load and configuration. So I changed my inputString() such that it would conditionally apply different randomization range to generate the string. This significantly reduced the number of iterations so that the program could run within 5 minutes on flip while still reserving the randomization element.