CS569-2015Spring Assignment1 Report

Hongyan Yi   yih@onid.oregonstate.edu

Apr.21.2015

1. Introduction

I wrote one algorithm **mergeFile.c**. It works in this way: If give me two files A.txt and B.txt, both with several characters. The algorithm will merge these two files to a new file C.txt as well as keep the characters in file C in alphabet order. Also, I wrote a **harness.c** by using CBMC tool to check the mergeFile algorithm.

2. Was the program correct or buggy? What was hard to specify? Was there functionality you could not specify?

My algorithm mergeFile.c is correct, I've run it independently before I wrote the harness.c. At the beginning of writing harness.c, I've tried to ignore CBMC related code, such as *__CPROVER_assume( ), nondet_char( ),* use other statements to instead of them, and compile directly in VS2013, and it worked well. But after I use CBMC related code in my harness.c, and run it with CBMC it always show *VERIFICATION FAILED*, even with simple command: *cbmc mergeFile.c harness.c -DSIZE1=2 -DSIZE2=1 --unwind 4*. I'm not sure the failed reason.

3. How long did it take to verify with different loop bounds? How did turning on/off bounds and pointer checker affect cbmc runtime?

It is obvious that the bounds and pointer checker will cost time but not affect too much, see the following two tables:

1) Time consuming with different loop bounds

No.	Commnd	Time
1	cbmc mergeFile.c harness.c -DSIZE1=2 -DSIZE2=1 --unwind 3	0.078s
2	cbmc mergeFile.c harness.c -DSIZE1=2 -DSIZE2=1 --unwind 4	0.12
3	cbmc mergeFile.c harness.c -DSIZE1=2 -DSIZE2=1 --unwind 5	0.25
4	cbmc mergeFile.c harness.c -DSIZE1=2 -DSIZE2=1 --unwind 6	0.359

2) Time consuming with bounds and pointer checker

No.	Commnd	Time
1	cbmc mergeFile.c harness.c -DSIZE1=2 -DSIZE2=1 --unwind 3--pointer-check --bounds-check --all-claims	0.093
2	cbmc mergeFile.c harness.c -DSIZE1=2 -DSIZE2=1 --unwind 4--pointer-check --bounds-check --all-claims	0.141
3	cbmc mergeFile.c harness.c -DSIZE1=2 -DSIZE2=1 --unwind 5--pointer-check --bounds-check --all-claims	0.256
4	cbmc mergeFile.c harness.c -DSIZE1=2 -DSIZE2=1 --unwind6--pointer-check --bounds-check --all-claims	0.375

4. Discuss the ability of the harness to find the bugs you introduced, and how to address the problem if it did not, including (if possible) a revised harness to find them.

I think the CBMC output will help me revise my harness to a better one. For example, when I run *cbmc mergeFile.c harness.c -DSIZE1=2 -DSIZE2=1 --unwind 3 --all-claims*, the results showed as flowing, will let me know that where it is failed.

```
** Results:
[] free called for new[] object: OK
[main.assertion.1] free argument is dynamic object: OK
```

```
[main.assertion.2] free argument has offset zero: OK
[main.assertion.3] double free: OK
[main.assertion.4] free called for new[] object: OK
[main.assertion.5] free argument is dynamic object: OK
[main.assertion.6] free argument has offset zero: OK
[main.assertion.7] double free: OK
[main.assertion.8] free called for new[] object: OK
[main.assertion.9] assertion c[k] >= tail: FAILED
[main.assertion.10] assertion k == SIZE1 + SIZE2: FAILED
[main.assertion.11] free argument is dynamic object: OK
[main.assertion.12] free argument has offset zero: OK
[main.assertion.13] double free: OK
[main.assertion.14] free called for new[] object: OK
[MergeFile.assertion.1] free argument is dynamic object: OK
[MergeFile.assertion.2] free argument has offset zero: OK
[MergeFile.assertion.3] double free: OK
[MergeFile.assertion.4] free called for new[] object: OK
[MergeFile.assertion.5] free argument is dynamic object: OK
[MergeFile.assertion.6] free argument has offset zero: OK
[MergeFile.assertion.7] double free: OK
[MergeFile.assertion.8] free called for new[] object: OK
[MergeFile.assertion.9] free argument is dynamic object: OK
[MergeFile.assertion.10] free argument has offset zero: OK
[MergeFile.assertion.11] double free: OK
[MergeFile.assertion.12] free called for new[] object: OK
[main.unwind.0] unwinding assertion loop 0: OK
[main.unwind.1] unwinding assertion loop 1: OK
[MergeFile.unwind.0] unwinding assertion loop 0: FAILED
[MergeFile.unwind.1] unwinding assertion loop 1: FAILED
[MergeFile.unwind.2] unwinding assertion loop 2: FAILED
[MergeFile.unwind.3] unwinding assertion loop 3: FAILED
[MergeFile.unwind.4] unwinding assertion loop 4: OK
[main.unwind.2] unwinding assertion loop 2: FAILED
```