

# CS6888 Program Analysis and its Application #HW3

Jianqiang Chen(jc3ze)

## Task1 Setup

Step 1: compile the tcas.c

*# gcc tcas.c -o tcas -ftest-coverage -fprofile-arcs*

It confuses me gcc-6 will have 16 fail tests and gcc-7 will have 12 fail tests.

Step 2: run the test in the runtests.sh

Step 3: for each test:

- Remove the .gcda file (it will record last tests)

- store the .gcov file to /output/fail if the test fails(returns "F"), store the coverage information to /output/failPrint as txt file;

- store the .gcov file to /output/success if the test fails(returns "P"); store the coverage information to /output/successPrint as txt file;

In total 12 failed tests, 1596 success tests for 1609 tests.

Code "task1.py" is attached.

## Task 2: Implement fault isolation technique

Step 1: traversal all files in /output/fail, for every line, if it doesn't start with "-" or "#####", store the line number in a dict. Every fail test will refresh this dict. Store the statement in another dict, which key is the line number, value is the line statement.

Step 2: traversal all files in /output/success, for every line, if it doesn't start with "-" or "#####", store the line number in a dict. Every success test will refresh this dict. Store the statement in another dict, which key is the line number, value is the line statement.

Step 3: calculate the S(s) for each executing statement s in fail tests.

Code "task2.py" is attached.

### Task 3: Run experiments and report findings

```
top 1: line number: 134 statement:      alt_sep = DOWNWARD_RA;
#12 #1586 #1596 11.0068879148
top 2: line number: 87 statement:      result = Own_Above_Threat() && (Cur_Vertical_Sep >= MINSEP) && (Up_Separation >= ALIM());
#12 #1591 #1596 11.0037570445
top 3: line number: 104 statement:     result =  Own_Above_Threat() && (Up_Separation >= ALIM());
#12 #1591 #1596 11.0037570445
top 4: line number: 131 statement:     else if (need_upward_RA)
#12 #1595 #1596 11.0012523482
top 5: line number: 133 statement:     else if (need_downward_RA)
#12 #1595 #1596 11.0012523482
top 6: line number: 136 statement:      alt_sep = UNRESOLVED;
#12 #1595 #1596 11.0012523482
top 7: line number: 139 statement:     return alt_sep;
#12 #1595 #1596 11.0012523482
top 8: line number: 142 statement: int main(argc, argv)
#12 #1595 #1596 11.0012523482
top 9: line number: 146 statement:     if(argc < 13)
#12 #1595 #1596 11.0012523482
top 10: line number: 151 statement:    initialize();
#12 #1595 #1596 11.0012523482
```

The result is the above picture. I have tested my scripts in small size gcov files and it works well.

I think the fault may exist in 131-142 line since the these lines have the most tops. And it will cause the fault tests in the main function.

#### 3 Variants:

Tacs\_f1.c: in line 122

// Mutant: change (&& ((&&)|)) to all ||

if (enabled || ((tcas\_equipped || intent\_not\_known) || !tcas\_equipped))

Total test: 1609 Total success: 1453 Total fail: 155

```
top 1: line number: 104 statement:      result =  Own_Above_Threat() && (Up_Separation >= ALIM());
#155 #1451 #1453 154.002063274
top 2: line number: 131 statement:      else if (need_upward_RA)
#155 #1452 #1453 154.001375516
top 3: line number: 133 statement:      else if (need_downward_RA)
#155 #1452 #1453 154.001375516
top 4: line number: 136 statement:      alt_sep = UNRESOLVED;
#155 #1452 #1453 154.001375516
top 5: line number: 139 statement:      return alt_sep;
#155 #1452 #1453 154.001375516
top 6: line number: 142 statement: int main(argc, argv)
#155 #1452 #1453 154.001375516
top 7: line number: 146 statement:      if(argc < 13)
#155 #1452 #1453 154.001375516
top 8: line number: 151 statement:      initialize();
#155 #1452 #1453 154.001375516
top 9: line number: 152 statement:      Cur_Vertical_Sep = atoi(argv[1]);
#155 #1452 #1453 154.001375516
top 10: line number: 153 statement:      High_Confidence = atoi(argv[2]);
#155 #1452 #1453 154.001375516
```

I changed the decision condition of alf\_sep\_test function, and the top 1 – top 5 all in this function. So this is a pretty good bug finding!

Tacs\_f2.c: in line 146

// Mutant: change < to >

if(argc > 13)

Total test: 1609 Total success: 1566 Total fail: 42

```
top 1: line number: 134 statement:      alt_sep = DOWNWARD_RA;
#42 #1556 #1566 41.007019783
top 2: line number: 87 statement:      result = Own_Above_Threat() && (Cur_Vertical_Sep >= MINSEP) && (Up_Separation >= ALIM());
#42 #1561 #1566 41.0038289726
top 3: line number: 104 statement:     result = Own_Above_Threat() && (Up_Separation >= ALIM());
#42 #1561 #1566 41.0038289726
top 4: line number: 131 statement:     else if (need_upward_RA)
#42 #1565 #1566 41.0012763242
top 5: line number: 133 statement:     else if (need_downward_RA)
#42 #1565 #1566 41.0012763242
top 6: line number: 136 statement:      alt_sep = UNRESOLVED;
#42 #1565 #1566 41.0012763242
top 7: line number: 139 statement:     return alt_sep;
#42 #1565 #1566 41.0012763242
top 8: line number: 142 statement: int main(argc, argv)
#42 #1565 #1566 41.0012763242
top 9: line number: 147 statement:     if(argc > 13)
#42 #1565 #1566 41.0012763242
top 10: line number: 152 statement:    initialize();
#42 #1565 #1566 41.0012763242
```

I change the line 146 in the main function. In the top 9 is exactly the fault I made. Though the rank is not so high and same as the original fault provided by our professor. I think that line is only responsible for "Error: Command line arguments". So doesn't affect normal tests.

Tacs\_f3.c: in line 120

// Mutant: change alt\_sep = UNRESOLVED to alt\_sep = UNRESOLVED + 1

alt\_sep = UNRESOLVED + 1;

Total test: 1609 Total success: 1596 Total fail: 12

```
top 1: line number: 134 statement:      alt_sep = DOWNWARD_RA;
#12 #1586 #1596 11.0068879148
top 2: line number: 87 statement:      result = Own_Above_Threat() && (Cur_Vertical_Sep >= MINSEP) && (Up_Separation >= ALIM());
#12 #1591 #1596 11.0037570445
top 3: line number: 104 statement:     result = Own_Above_Threat() && (Up_Separation >= ALIM());
#12 #1591 #1596 11.0037570445
top 4: line number: 131 statement:     else if (need_upward_RA)
#12 #1595 #1596 11.0012523482
top 5: line number: 133 statement:     else if (need_downward_RA)
#12 #1595 #1596 11.0012523482
top 6: line number: 136 statement:      alt_sep = UNRESOLVED;
#12 #1595 #1596 11.0012523482
top 7: line number: 139 statement:     return alt_sep;
#12 #1595 #1596 11.0012523482
top 8: line number: 142 statement: int main(argc, argv)
#12 #1595 #1596 11.0012523482
top 9: line number: 146 statement:     if(argc < 13)
#12 #1595 #1596 11.0012523482
top 10: line number: 151 statement:    initialize();
#12 #1595 #1596 11.0012523482
```

The output is same as the tcas.c. I guess this line doesn't make change to the tracks. It only has invisible impact on the program.

**Conclusion:** fault isolation skill is useful and efficient. It could help us narrow the scope and focus on high-risk lines. However, it could not tell us exact line is the line cause the problem, and may not able to find the invisible impact fault line. So large impact fault is easier to find with this tool, small impact fault is harder to find.