CS4218 Hackathon Report

Team_15

CAO SHENGZE
KIM HYUNG JON
YAP HAN CHIANG

1.Introduction

Regarding our tests:

We have written a Junit test class “ErrorRevealingTest.java” that contains test cases that reveal the presence of all the bugs we discovered.
In the bug report table below, each bug either has the corresponding test case name, or a command revealing the bug when executed on Team 1’s implementation of Shell. The names of the test cases in ErrorRevealingTest.java follow the format “testBugRevealing#”, where # is the index in the bug report table, for ease of identification.

Both ErrorRevealingTest.java and files created for use with the test are zipped with this report. The test files should be put at the same directory as the src folder. ErrorRevalingTest.java should be put in the package sg.edu.nus.comp.cs4218test.integration (together with the Team I’s integration test files).

Regarding avoiding duplicates:

From Lab 7, the definition of duplicates is given as: “behavior that exercises the same problem or an identical way of reproducing another bug (e.g., if a function f(String s){} does not handle special symbols in the input, then failures reported for cases f(“\”); f(“&”); f(“$”); are duplicates! Only one of them should be reported)”

Hence, to determine whether two bugs we found are duplicates or not, we focused at whether the program behavior that lead to the failure are identical. For example, although both bug 1 and bug 2 in our bug report below are related to numeric sort in SortApplication, we considered them separate bugs since the program behaviors responsible for those failures are different.

We have gone through all the tests we have found and are quite sure that none of the tests below are duplicate bugs.

Regarding system-related errors:

13 of the tests failed in our computers, apparently due to OS-related issues. While these are not bugs, we still mentioned them for the team’s information.

Regarding Team 1’s assumptions:

Some of the assumptions made by Team 1 appear to violate the specifications of the project description. For example, wc should accept multiple files; the pipeline should be allowed to use together with semicolon. In such cases, we did not accept these behaviors as implementations and classified them as bugs.

General method used for testing:

1. First step is to look through all the test cases that Team 1 has and check whether they all pass. We also tried to match their integration test cases to a pairwise table to see what is missing.
2. Then, we went through the applications one by one. For each application, we first observed its own performance (unit test), then we tried to use it together with other applications (integration test) to see whether we can find errors. We focused more on the pairs that do not appear in their own test cases.
3. Following step 2, we also copied our tests into the other team's test folder and ran, after making adjustments to remove build errors. We observed the result, and first determined whether the failures were happening because of different implementation (e.g. different way of counting the chars and lines, different way of output, incompatible parameters) or actually due to bugs.
4. We also conducted random testing by running the Shell directly and executing some applications or combinations of applications using pipes and command substitution, with custom files as input if necessary, to see if any unexpected behaviors occur.
5. During the process 1-4, if any test failed, we analyzed by simplifying the input to isolate and identify the exact conditions that cause the failure. For example, we observed the tests in Sort that threw unexpected exceptions and tried to find patterns in the failing inputs (while keeping in mind that there may be more than one causes, which could be distributed unevenly across the failing cases). Once we identified the exact conditions under which the program fails, we documented the bug.

2. Bug Report

Bug #	Description	Test Case	Comments
1	Sort: if the “-n” option is chosen, numbers come behind alphabets	see testBugRevealing1()	
2	Sort: if the “-n” option is chosen and at least one line in the file contains only whitespace(s), sort is unable to process the file and throws exception	see testBugRevealing2()	
3	Sort: if multiple files are used, the last line of nth file and the first line of n+1th file get merged to one	see testBugRevealing3()	
4	Sort: some special characters get come behind numbers or alphabets	see testBugRevealing4()	
5	Sort: numeric sort does not work on stdin (produces same result as non-numeric sort)	see testBugRevealing5()	
6	Sort: sometimes reads less empty lines than the file actually contains. Confirmed happening if file only contains empty lines or the only non-empty line in the file contains only whitespaces.	see testBugRevealing6()	
7	Wc: does not work with more than one files. Team 1 commented that their implementation is to consider only the last arg as filename, but this violates the project description that wc should accept “given files”.	see testBugRevealing7()	
8	Double quote: wrapping arguments containing whitespaces in double quotes to indicate that the string is a single argument and that the whitespaces should not be considered as argument separators does not work. Bug 21 is related.	sort "fileBugRevealing8 with space.txt”	

9	Double quote: Process of the double quote should be executed before the command, then echo a”b” should just print ab instead of regrading it as an invalid command.	echo a”b”	
10	Single quote: Process of the single quote should be executed before the command, then echo a’b’ should just print ab instead of regarding it as an invalid command.	echo a’b’	
11	Redirection: input would not work if the redirection is not at the last	echo a > 1.txt; cat < 1.txt 1.txt	
12	Pipeline and semicolon appearing together is not supported (Stated as assumption in the report, but pipeline and semicolon should be able to appear together)	echo a | cat; echo b | cat	
13	**Globbing is not implemented at all**	cat testDir/*	
14	Command substitution: does not work with more than 2 command substitutions	sort `cat cmdSubFile.txt` `head cmdSubFile4.txt`	
15	SedApplication: does not support regex, it just treat the regex as charSequence	echo a | sed s/[a]/b	
16	SedApplication: cannot use symbol “s” as separator symbol. The specification is: However, this separation symbol should not be used inside the regexp and the replacement string. But it does not say the s cannot be used.	echo a | sed ssasbs	
17	Cmd substitution with results of a pipe does not work	echo `echo a | cat`	
18	Pipe: cannot handle large amount of bytes e.g. result of cal 2017	cal 2017 | cat	
19	Cal: if 3 or more arguments are passed and the arguments are	cal 1 1 1	

	invalid, cal outputs nothing instead of throwing an exception		
20	Grep: cannot handle input from stdin with certain contents.	testBugRevealing20()	
21	Single quote: cannot transfer the space to a space character. The double quote has the same problem, but this is addressed in bug 8.	echo “a b”|grep ‘ ‘	

System-related errors:

These are problems that occur due to OS incompatibility, not bugs.

1. The test case testWcWithStdinWithOptionMForNonAscii() in WcApplicationTest may fail if the test machine does not contain the relative code for test. My computer, it would get 8 instead of 3. As I think it a System problem, I did not put it into the bug report.
2. The test case testPipeCatAndSedAndWc() may fail under windows as in windows a newline character is \r\n, which with length of 2 instead of 1. Lots of other integration test cases with Wc all have this problem as well.
3. The test case testGValidEchoDate() may fail if system language is not English. Some other test cases have this problem as well.