Debugging under TOS

Objectives

- Explain the TOS testing system
- Explain some debugging techniques when you don’t have a debugger and when a program error typically crashes the whole system

Debugging under TOS

- Debugging: remove errors from a program.
- Usually you have a tool called a debugger that helps you to locate errors.
- Problem: in system programming there is no debugger! (since a debugger requires an OS).
- Conclusion: there is no debugger in TOS
- Question: how to find problems in TOS?

Test Cases

- TOS comes with several test cases that test the behavior of your implementation
- All these tests are located in `~/tos/test`
- Each test case has a name and it tests one particular feature of your implementation, e.g., `test_mem_1` tests the peek and poke functions
- Each test case is stored in a separate file, e.g., `~/tos/test/test_mem_1.c`
- If a test fails, the system will print an error code

Running Test Cases

- Compile TOS with test cases enabled. This can be accomplished by typing:

```
cd ~/tos
make tests
```

- Run the TOS Test Center (TTC)
  - Make sure .bochsrc includes the line:

```
com2: enabled=1, mode=socket, dev=localhost:8899
```

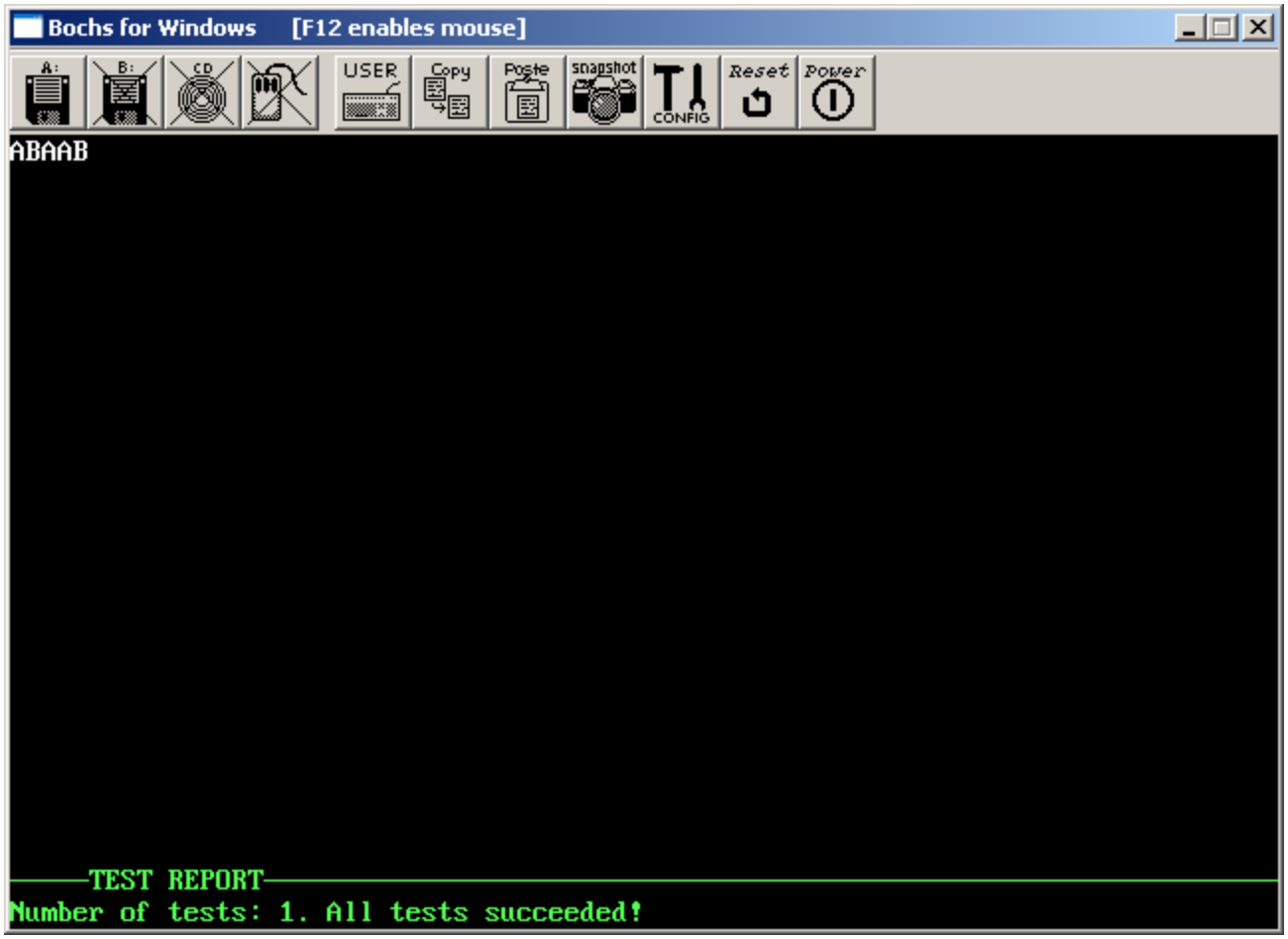
  - Launch TTC with:

```
java -jar ttc.jar
```

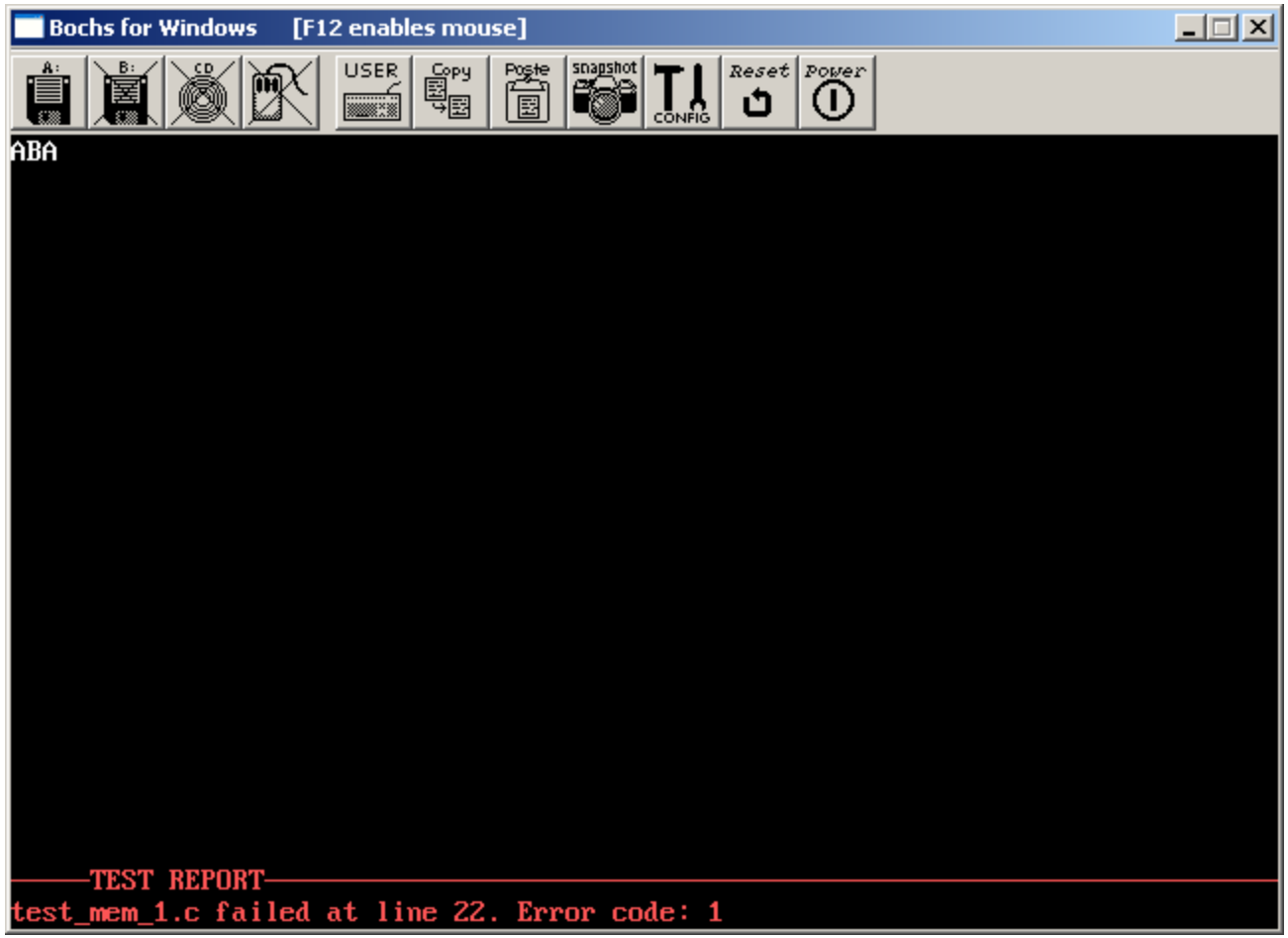
  - Start (or re-start) Bochs
- TOS contains the binaries of a reference implementation. You can run a test case against the reference implementation to see what the output of the test case should look like.
- This can be accomplished by typing:

```
cd ~/tos
make run_ref
```

Successful Test



Unsuccessful Test



Notes on TOS Test Cases

- The assignment slides indicate which test cases should be run for that particular assignment.
- If a test cases fails, it will print out an error code. The HTML page `~/tos/test/messages.html` explains all the error codes.
- If a test case fails, it often helps to study the implementation of the test case to understand what it is doing. Note that some helper code is located in `~/tos/test/common.c`
- If all test cases succeed, it doesn't necessarily mean that the implementation is bug free (testing vs. verification)
- If TOS crashes (without printing any error codes), you'll have to employ a debugging technique explained on the following slides.
- Always run previous test cases. If one test succeeds today, it may fail tomorrow due to some changes you made.

Debugging hints

- If something goes wrong in TOS, the whole machine usually crashes.
- In that case, the first priority is to locate the line in your program that causes the crash.
- This can be done by carefully inserting an endless loop into your program:

```
statement_1;
statement_2;
Crash_causing_statement;
statement_3;
statement_4
while(1);
```

⟹ System crashes

```
statement_1;
statement_2;
while(1);
Crash_causing_statement;
statement_3;
statement_4;
```

⟹ System does not crash

```
statement_1;
statement_2;
 Crash_causing_statement;
while(1);
statement_3;
statement_4;
```

⟹ System crashes

Debugging hints

- Once the statement that causes the crash has been isolated, the next step is to understand why it crashes.
- This requires us to know the values of C-variables.
- Use `kprintf()` to print the value of C-variables.

Debugging hints

- Another powerful debugging tool are assertions.
- An assertion defines a condition that you expect to be true at a certain program location.
- The assertion is tested at runtime.
- If the assertion evaluates to FALSE, a detailed error message is given.
- TOS provides a courtesy implementation of assertions (i.e., you don't have to implement it).
- However: the assertions provided in TOS assume a working `output_string()` function. This means you can only use assertions once you have implemented this basic output function.
- Assertions are implemented through function `assert()` defined in `~/tos/include/assert.h`
- `assert.h` is automatically included if you include `kernel.h`

Assertion Example (1)

```
Node* elem;
elem = alloc_data_item();
assert(elem != (Node*) 0);
```

- This piece of code is based on the example given earlier for dynamic memory management techniques
- This assertion will fail if there should not be any more free data items

Assertion Example (2)

```
void move_cursor(WINDOW *wnd, int x, int y)
{
      assert(x >=0 && x < wnd->width);
      …
}
```

- It is often useful to check input parameters.
- The code above checks that input parameter ‘x’ is within the allowed boundaries.

Assertion Tips

- Use many assertions (assertions are your friend!).
- Never remove assertions once you have added them to your program.
- `assert()` is very useful in testing the validity of input parameters of functions.
- `assert(0)` always fails. Useful to mark locations in your program that should never be reached (e.g. default case of a switch-statement).

TOS Test Center (TTC)

User's Manual

TTC Feature List

- Simplification of test case execution.
- Saving of selected test cases at shutdown.
- Highlighting and scrolling to error line.
- Multiple test case selection.
- Screenshots of successful executions for each test case in Bochs.

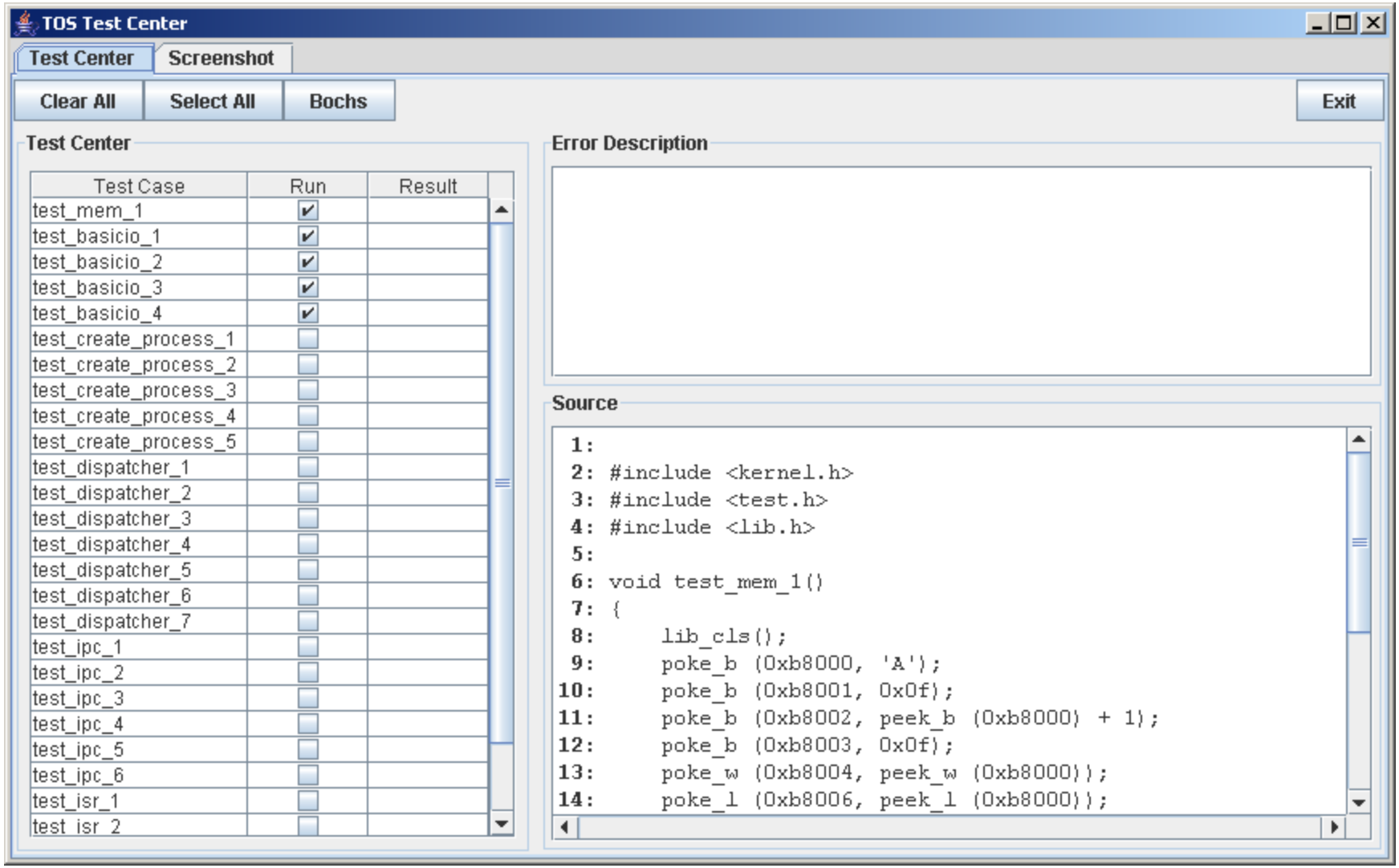
Running the TTC

- Compiling the TTC requires a Java JDK.
- The TTC will automatically compiled together with TOS.
- If compilation succeeded, there will be a file called `~/tos/ttc.jar`
- TTC can be launched via:

```
cd ~/tos
java –jar ttc.jar
```

- Note: TTC always needs to be launched *before* running Bochs.

Sample Screenshot

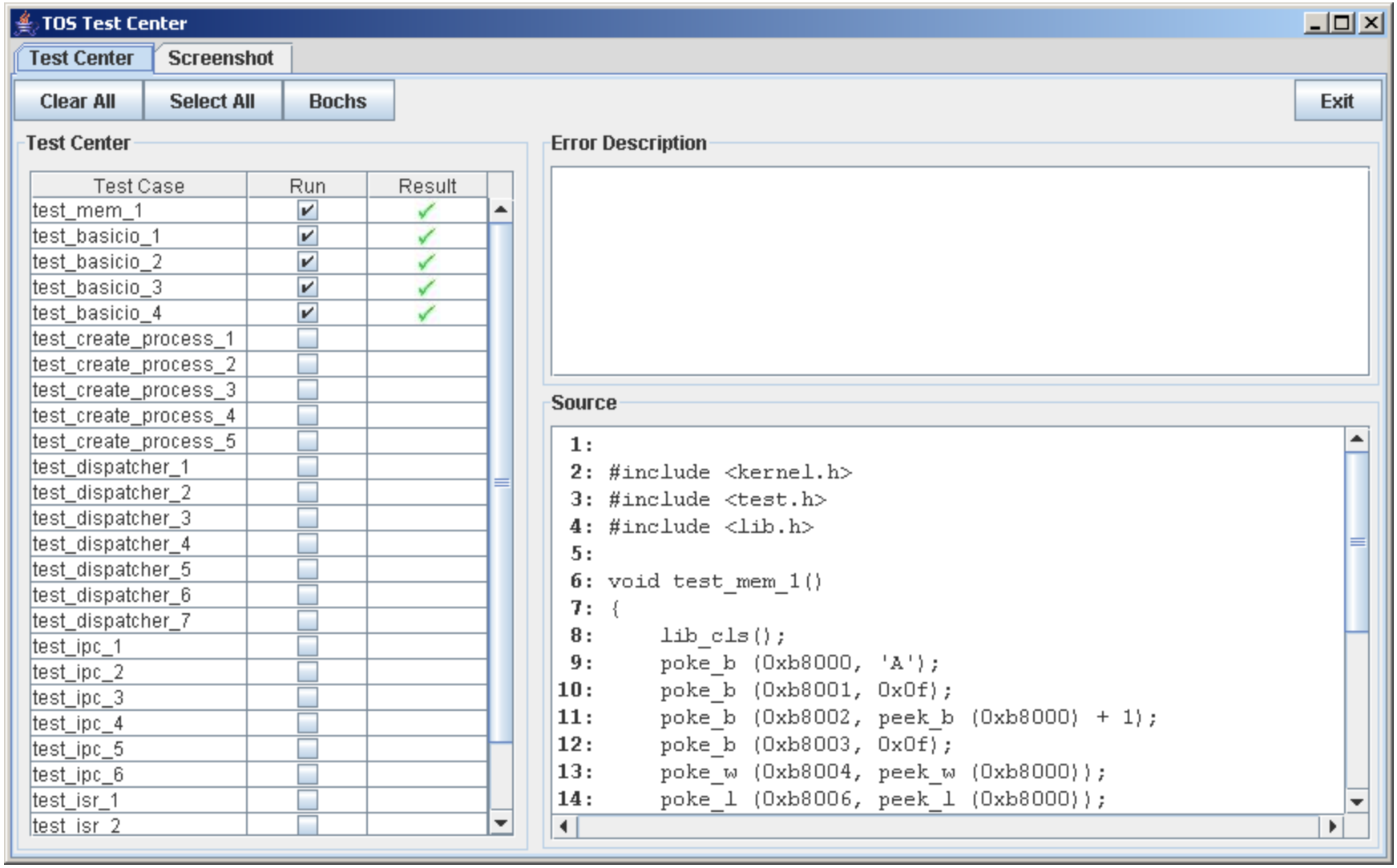


Using the TTC

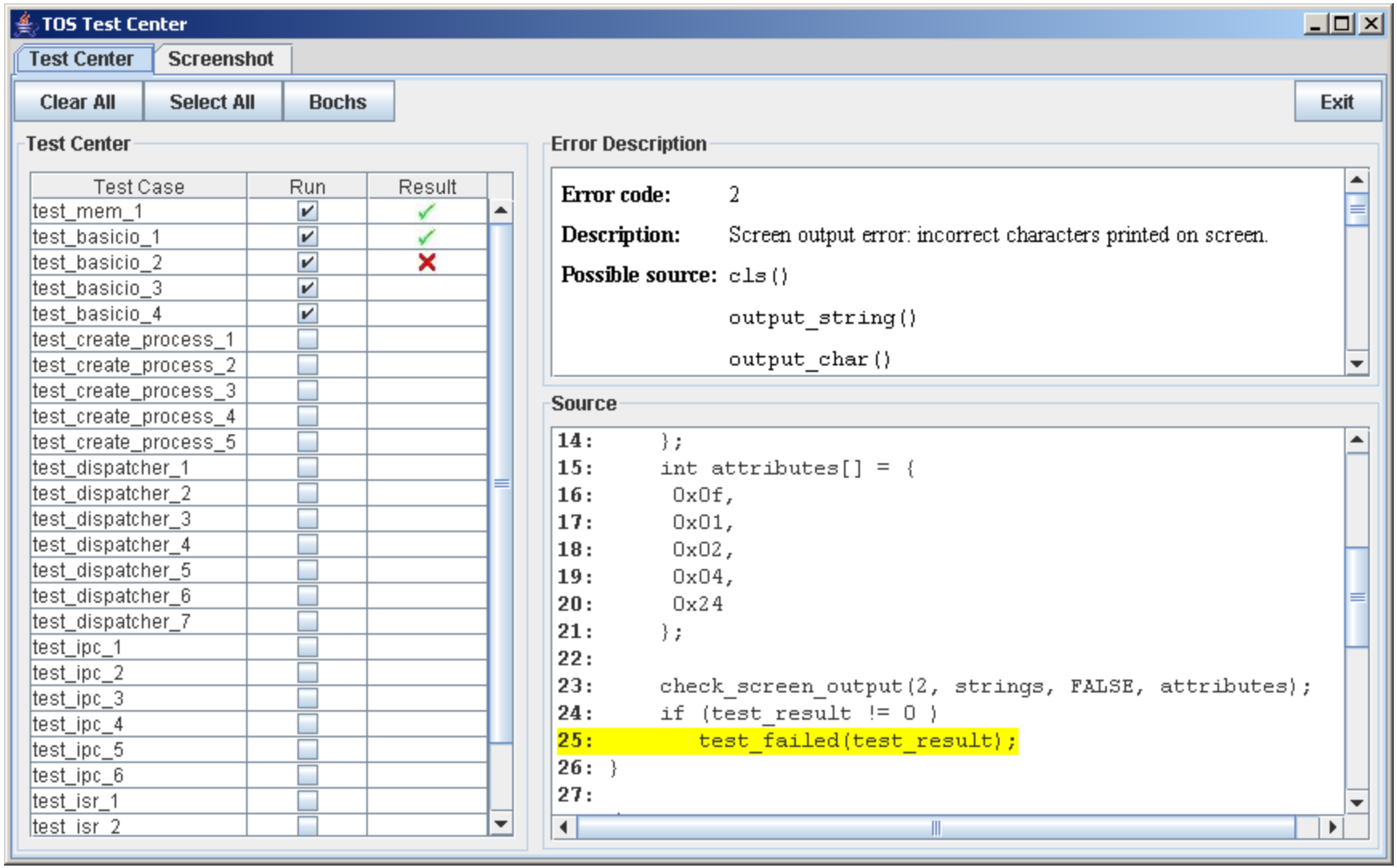
- First select the test cases you want to run by checking the appropriate boxes under the Run column of the TTC.
- Then click on the Bochs button* to execute Bochs for these test cases.
  - If a test case passes the test, you will see a green check mark on the Result column for that test case.
  - If you do not pass a test case, you will see a red X mark on the Results column. Along with this, you will see the type of error that was detected on the Error Description field. Finally, you Source field will scroll to the line that caused the error and the line will get a yellow highlight.

*Bochs button will not work if TTC reports "Could not find .bochsrc"

Successful Run



Unsuccessful Run



Multiple Test Cases

- If a test case is passed and there are more selected, then Bochs and TTC continue top-down to the next test case without stopping. This allows multiple test cases to be tested with one run.
- The TTC will stop at the first erroneous test case and will display the errors as shown in the previous slide.

How to Select Many Consecutive Test Cases

- First click on the first test case you want to run in the Run column.
- Then hold the Shift key down and click on the last case you want to run.
- All the test case from the start till end of your selection will be checked.
- To unselect them, hold the Shift button again and click on the same test case you just clicked on.
- Play around to found out more way to select multiple test cases.
- WARNING: DO NOT USE THE Ctrl KEY. It will cause an array index out of bounds exception for some unknown reason. So please avoid it. ☺