

The University of Queensland
School of Information Technology and Electrical Engineering
Semester One, 2011
COMP2303 / COMP7306 - Assignment 2
Due: 11pm 16 April, 2011
Marks: 50
Weighting: 25% of your overall assignment mark (COMP2303)

Introduction

The aim of this assessment is to develop and test your debugging skills using the compiler and debugger. The exercises come in two parts: the *Binary Bomb* and *Quicksand*. In these exercises you will be presented with puzzles relating to the behaviour of programs. You will be given marks for the number of puzzles you solve correctly and the number of attempts you take to do so.

This assignment must be done on *moss*. Neither the *Binary Bomb* nor *Quicksand* will run on any other system.

This is an *individual assignment*. You should feel free to discuss aspects of C programming and the assignment specification with fellow students. You should not actively help (or seek help from) other students with the exercises.

Binary Bomb

This exercise is focused on the use of `gdb`. The bomb consists of a number of phases which you must defuse by entering the correct passphrase. You will need to discover this phrase by examining the bomb program in a debugger. The operation of the bomb has been demoed in lectures.

Important programs

- `getbomb` - retrieves *your* version of the bomb. If you run this command it will give you the same bomb (or an updated bomb with the same answers as the one you had).
- `bomb` - executes the bomb and prompts for the passphrase to defuse the current phase. You can use `PASS` to move onto the next phase. When you enter a phrase you will be prompted to type `OK` before your guess will be recorded. You can quit/kill/restart the bomb at any time before this with no penalty. **We recommend that you run your bomb inside `gdb`.**
- `gdb`

Marks

Your attempts will be recorded automatically when you run the bomb on *moss*. Each phase of the bomb is worth a maximum of 5 marks. The mark you receive for each phase depends on the number of attempts you make to defuse that phase. Specifically, if it takes n wrong attempts to defuse a phase your mark will be:

$$5 \times 0.8^n$$

Bomb Tips

- Remember that you must enter input before a phase starts to run. Enter anything to get started but be sure not to 'OK' your guess until you are sure you have it correct.
- You should be familiar with the commands `p`, `break`, `run`, `cont`, `next step`, `up`, and conditional breakpoints.
- This is not a race. It is very easy to make silly mistakes due to lack of concentration.

Quicksand

Quicksand commands

This exercise is about the compiler rather than the debugger. You will explore the operation of a program using simple statements such as `printf`. You must identify (**Not fix**) lines which are either wrong in the source or are being corrupted (deliberately) in the compilation process.

- `get` - retrieves *your* version of the source files. If you run this command, then it will overwrite any changes you have made with a clean copy.
- `test` - compiles your source files and performs a series of tests.
- `guess` - records your guess as to the location of a bug.
- `status` - displays how many bugs you have correctly identified and how many guesses that you have taken so far.

More details about quicksand can be found on the man page. `man quicksand`.

Marks

Your attempts will be recorded automatically when you run `quicksand guess` on *moss*. Repeated guesses or guesses involving invalid tags will not count against you. Each time you correctly identify a bug tag, quicksand will give you a mark based on how many incorrect guesses you have made since your last correct tag. If you identify a tag after n incorrect guesses your mark will go up by:

$$\frac{25}{11} \times 0.9^n$$

Guessing not will reduce the marks you already have.

Quicksand Tips

- Remember that you are looking for lines which do not do what they are supposed to do. These lines could do
 - more than they are supposed to,
 - less that they are supposed to
 - something completely different.

They could be written incorrectly or Quicksand could be corrupting them during the compile process.

- Remember to distinguish between symptoms and causes. The place where a problem becomes obvious is not always the cause. For example, if:

$c=b-a$

is supposed to result in c storing 5, this statement would not be at fault if $a==17$ and $b==2$.

- It is possible to identify the faulty lines without making incorrect guesses. If you suspect a line is buggy, think about how you can test your theory.