CSCI 2132
Software Development

Lab 5:

gcc and gdb tools

Instructor: Vlado Keselj

Faculty of Computer Science

Dalhousie University

Lab Overview

- Learning more about `gcc` and about `gdb`
- Using a program as an example
- Invoking debugger in `emacs`
- More gcc options
- Another C program

Step 1: Login and Lab Setup

- Login to bluenose
- Create `lab5` directory in SVN and submit

Step 2: gcc -Wall option

- copy `~prof2132/public/hello.c` to your current directory
- remove (or comment out) `return 0;`
- compile as usual: `gcc -o hello hello.c`
- compile using: `gcc -Wall -o hello hello.c`
- put back `return 0;`
- compile again using `-Wall` option
- Add `hello.c` to SVN

Step 3: gcc -g option

- Copy `~prof2132/public/numbers.c` to your current directory
- Compile: `gcc -g -o numbers numbers.c`
- Add `numbers.c` to SVN

Step 4: gdb

- Run: `gdb numbers`
- Enter `run` or simply `r`
- Enter: `8`
- Program will exit normally

Step 5: Breakpoints

- In gdb enter `l` or `list`
- `break 8` or `b 8`
- Set a breakpoint at line 20
- `run` or `r`
- `print value` or `p value`
- `step` or `s` twice
- On program prompt, enter `7`
- Enter a command to check value of variable `k`
- `continue` or `cont` or `c`
- Print the value of the variable `value`
- Repeate the previous two steps
- Quite debugger: `quit` or `q`

Step 6: Calculated Sequence

- Record values of variable `value`
- Enter the values into http://oeis.org to find out the sequence

Step 7: gdb in emacs

- Open `numbers.c` using `emacs`
- Compile using option `-g`
- Try the following window-splitting commands:

```
C-x 2  C-x o  C-x 1  C-x 3  C-x 1  C-x 3
```

- Enter: `M-x` and then `gdb`
- Enter: `gdb -i=mi numbers`
- `b 8  b 20  r`
- `s  s  C-x o`
- Enter input,: 8 (for example)
- `C-x 0  C-x 3  C-x *gud-numbers*`
- Continue with gdb commands tried before

Step 8: gcc option -std

- Modify the 'for' loop in `numbers.c`
- Try: `gcc -o numbers numbers.c`
- Try: `gcc -std=c99 -o numbers numbers.c`
- Try compiling with `-g` annd `-Wall` options

Step 9: Compiling binary.c

- Copy `~prof2132/public/binary.c` to the current dir

```
gcc -g -o binary binary.c
```

Step 10: Running gdb on ‘binary’

- Run `gdb binary` in emacs, enter 35
- Use: `break main`
- Enter: `run`
- `step`
- `print array`
- `step` (two times)
- Enter: `35`
- Enter: `print key`
- `step`
- Fix the bug
- Use: `kill` to stop the program

Step 11: Finding Another Bug

- Compile and run the program again in gdb
- Enter 35 again
- Output: `35 is at location 5.` (wrong)
- Use `delete 1` in gdb to remove previous breakpoint
- `break binary_search` to set a new breakpoint
- `run`
- `print *array@10` to print elements
- `print len` and `print key`
- `display lower`
- `display upper`
- `display middle`
- Keep entering `step` until the loop is finished
- Use `print array[middle]`

- However, `step` branches into `return middle` statement
- Fix the bug in line 45

Step 12: Testing Program

- Add `binary.c` to SVN

Step 13: End of Lab

- You can work on the assignment or practice programming questions
- Do not forget to commit all required files to SVN.