ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

CS255 - Programming Lab

Ενότητα: Tutorials

Άγγελος Μπίλας

Τμήμα Επιστήμης Υπολογιστών

Tutorial 3 - GDB Basics

GDB, the GNU Project debugger, allows you to see what is going on *inside* another program while it executes -- or what another program was doing at the moment it crashed.

GDB can do four main kinds of things (plus other things in support of these) to help you catch bugs in the act:

Start your program, specifying anything that might affect its behavior.
Make your program stop on specified conditions.
Examine what has happened, when your program has stopped.
Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.
The program being debugged can be written in Ada, C, C++, Objective-C, Pascal (and many other languages). Those programs might be executing on the same machine as GDB (native) or on another machine (remote). GDB can run on most popular UNIX and Microsoft Windows variants.

Compile your program for gdb

gcc -ansi -pedantic -Wall -g

Run your program using gdb

gdb myprogram

GDB will start and you will see a prompt (similar to bash's). Type
r -any_flags_needed any_args_needed
to start the execution of your program.

another alternative is
gdb myprogram --args -myprograms_flags myprograms_args
this way simply typing r will pass the arguments passed with --args to your program, saving you from some typing.

Note that r and run are the same command. From now on we will refer to such shortcuts with the following syntax, r[un]

Inspect code

l[ist] [lineno] will show you the programs source code around line *lineno*. If *lineno* is not given it will list around the current line.

If you want to see the code of another file issue l[ist] filename:lineno

Breakpoints

Breakpoints help you inspect your program's state at certain points.

Set

To set a breakpoint at line *lineno* issue b[reak] lineno
To set a breakpoint at every *my_function* invocation issue b[reak] my_function

Similar to l[ist] if you want to set a breakpoint at another file issue b[reak] filename:lineno

You can also set conditional breakpoints like this b[reak] my_function if a==0 && b==1

Show

Use i[nfo] b[reakpoints] to list all available breakpoints.

Use i[nfo] b[reakpoints] breakpointno to show information about *breakpointno*

Enable/Disable

You can enable and disable *breakpointno* issuing en[able] breakpointno and disable breakpointno

Remove

You can delete *breakpointno* issuing d[elete] breakpointno. Issue d[elete] to delete all available breakpoints.

Backtrace

bt or backtrace will show you the call stack that got you here. You can trace the call path that gave you a crash for instance (get a picture of how we got here). With backtrace you can investigate the arguments passed to the functions that finally called the function we where are running at the crash/breakpoint.

Continue/Step/Next/Finish

When a breakpoint is reached the program's execution stops to let you inspect its state. To continue the execution you can use:

1. c[ontinue] and the program will execute till its end or till it reaches another breakpoint.
2. f[inish] to execute this function until it returns and stop again.
3. s[tep] to execute the next line
4. n[ext] similar to s[tep] but doesn't step into function calls. If a function call is reached it executes it and then stops.

Note that s[tep] and n[ext] can be given an argument like s[tep] [count] and n[ext] [count]. This way you can tell gdb to execute this commands *count* times.

Print

From GDB you can print variables/structures, actually almost everything.

Use p[rint] with C-like arguments. For instance p my_struct->value or even p my_struct and gdb will print all the struct's fields.

Interrupt execution

Pressing Ctrl-C will break the programs execution allowing you to inspect its current state (useful when you get stack in infinite loops). After you are done You can continue the program's execution like it was a breakpoint.

Quit

Just type q and Enter

Notes

- Pressing the UP key goes through previously executed commands.
- Pressing Enter executes the last command.
- To visualize the debugging experience use gdbtui or gdb -tui or cgdb

References:

- http://www.gnu.org/software/gdb/

Authored by: Foivos S. Zakkak

Άδειες Χρήσης

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης