Debugging
GDB

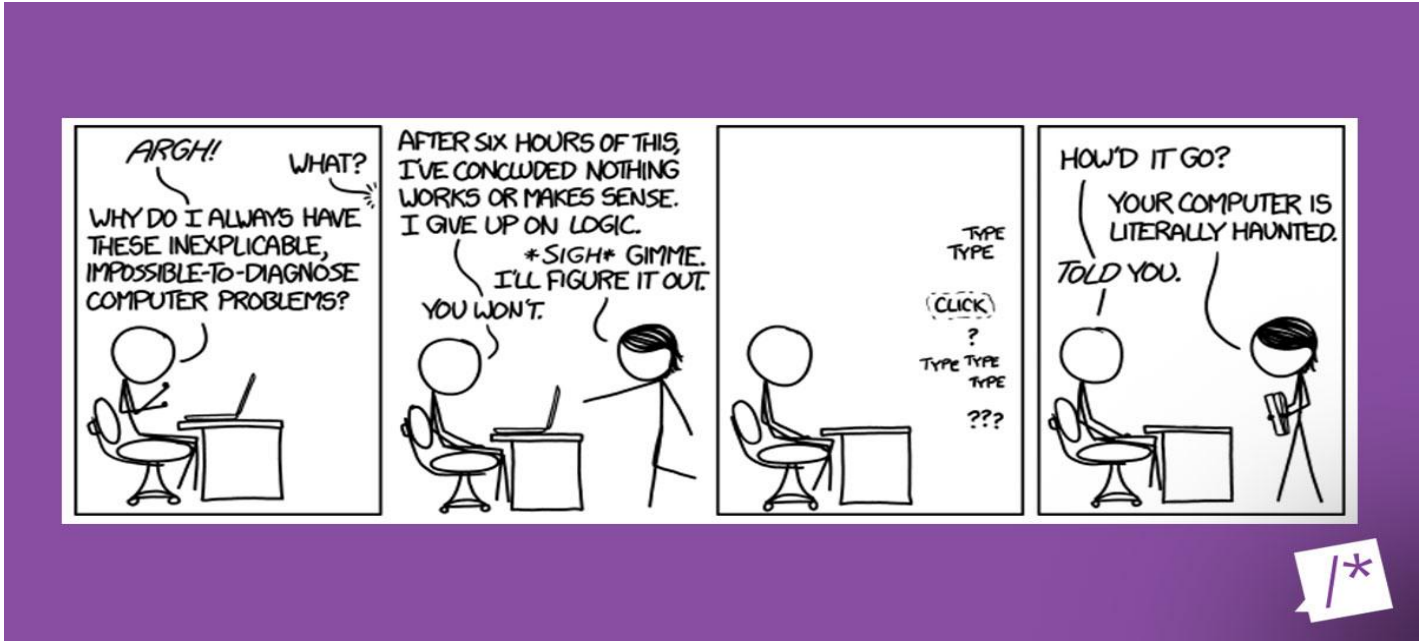
Εργαστήριο Λογισμικού - HY255
Ορέστης Χιωτάκης

GDB - GNU Debugger

- Το debugging είναι από τις πιο χρονοβόρες διαδικασίες στην παραγωγή Software/Hardware
- Δωρεάν και Open Source [link]
- Μπορεί να τρέξει σχεδόν παντού (εγκατεστημένος by-default σε πολλά συστήματα Linux)
- Υποστηρίζει μεγάλη ποικιλία γλωσσών (C 💗, C++, D, Fortran, Assembly, Go, Rust, Pascal κ. α)
- Τρέχει στο τερματικό (πιο γρήγορος, δεν χρειάζεται desktop environment)
- Υπάρχουν διάφορα front-end, δηλαδή, UI υλοποιήσεις (πχ. Seer)
- Υποστήριξη πολλών και διαφόρων ενεργειών για γρήγορο και αποτελεσματικό debugging
- Ιδιαίτερα χρήσιμος όταν “παίζουμε” με διαχείριση μνήμης και pointers (C)!!!
- Αύξηση της παραγωγικότητας και μείωση του χρόνου παραγωγής κώδικα

GDB - GNU Debugger

credits



ΠΑΡΑΔΕΙΓΜΑΤΑ

Παράδειγμα 0 - Simple Segmentation Fault

Ενημερώνουμε κατάλληλα τον compiler.

⇢ *gcc* **-g** filename.xxx *-o progname* (debug symbols)

Τρέχουμε τον gdb με όρισμα το πρόγραμμα που θέλουμε να κάνουμε debug

⇢ **gdb** progname

Στη συνέχεια, περνάμε τα απαραίτητα arguments (αν χρειάζονται)

⇢ (gdb) **run** arg1 arg2 arg3 … (ή κατευθείαν **gdb** --args progname arg1 arg2 …, **r**)

Γενικά, στον gdb υπάρχουν διάφορες κλάσεις από εντολές. Για να πάρουμε βοήθεια (documentation) σχετικά με την κάθε κλάση, καθώς και τις εντολές που εντάσσονται σε αυτή

⇢ (gdb) **help** (ή πιο συγκεκριμένα, **help** classX)

Παράδειγμα 0 - Simple Segmentation Fault

Αφήνουμε το πρόγραμμα να τρέξει (“μέσα” στον gdb) ώσπου να σταματήσει λόγω κάποιου λάθους. Ο gdb μας πληροφορεί σε ποια γραμμή στον κώδικα είναι το πρόβλημα, καθώς και σε ποια συνάρτηση. Στη συνέχεια, μπορούμε να τυπώσουμε τα περιεχόμενα των μεταβλητών (στο παρόν stack-frame!!!, περισσότερα στο τέλος των διαφανειών!!!).

⇢ (gdb) **print** variable (**p**)

Έτσι λοιπόν, μπορούμε να ελέγξουμε τις τιμές την ώρα του “εγκλήματος”, να διαπιστώσουμε που βρίσκεται το λάθος και να το διορθώσουμε! Τελειώνοντας, σταματάμε την λειτουργία του gdb.

⇢ (gdb) **q**

Διορθώνουμε το πρόγραμμα, το τρέχουμε για να βεβαιωθούμε ότι όντως διορθώσαμε το λάθος (ή ότι δεν υπάρχουν κι άλλα), και τέλος, αφαιρούμε το **-g** από τις παραμέτρους που δίνουμε στον gcc.

Παράδειγμα 1 - Simple Infinite Loop

Σε αντίθεση με το προηγούμενο παράδειγμα, εδώ το πρόγραμμά μας δεν διακόπτεται, αλλά, κολλάει σε ένα infinite loop. Στην προκειμένη περίπτωση, χρησιμοποιούμε breakpoints. Ο gdb θα τρέξει το πρόγραμμα χωρίς καμία διακοπή, μέχρι να συναντήσει κάποιο breakpoint. Βάζουμε πρώτα τα breakpoints και μετά καλούμε την run!!!

⇢ (gdb) **b** line ή function (**b** function ή **b** main.c:28, **b** 14)

Πως μπορούμε να δούμε τι breakpoints έχουμε ορίσει μέχρι στιγμής;

⇢ (gdb) **info breakpoints** (κάθε breakpoint έχει ένα ακέραιο αναγνωριστικό, **info b**)

Αν θέλουμε να τα απενεργοποιήσουμε (οποιαδήποτε στιγμή στο debugging).

⇢ (gdb) **disable** breakpoint# (**disable** 2)

Παράδειγμα 1 - Simple Infinite Loop

Όταν το πρόγραμμα σταματήσει εξαιτίας ενός breakpoint, μπορούμε να τρέξουμε διάφορες εντολές για να συλλέξουμε περεταίρω πληροφορίες (πχ: print, όπως είδαμε και προηγουμένως).
Μπορούμε να συνεχίσουμε την εκτέλεση γραμμή-γραμμή, χειροκίνητα.
⇢ (gdb) **next** (ή **step**. Η next, εκτελεί τις συναρτήσεις σαν να είναι μία εντολή, ενώ η step, μπαίνει μέσα στη συνάρτηση και την εκτελεί γραμμή-γραμμή, **n**, **s**).
Πλέον, παίζουμε με next, κι όποτε χρειαστεί κάνουμε print. Παρατηρούμε κάτι περίεργο/ασυνήθιστο; Ίσως κάποιο μοτίβο; Αφού βρούμε το λάθος, τερματίζουμε πάλι το πρόγραμμα όπως είδαμε πριν [...]

Παράδειγμα 2 - Infinite Loop advanced

Μία άλλη προσέγγιση. Πρέπει να σταματήσουμε το infinite loop.

⇢ (gdb) <**Ctrl-C>** (SIGINT, ο gdb δεν θα τερματίσει το πρόγραμμα οριστικά!!!)

Τώρα πρέπει να δούμε σε ποια συνάρτηση βρισκόμαστε (και πως φτάσαμε εκεί).

⇢ (gdb) **backtrace** (οι αριθμοί δίπλα στο '#' είναι τα αντίστοιχα stack-frames, **bt**)

Το πρόγραμμα έχει φορτωμένο το stack frame της τελευταίας συνάρτησης που καλέστηκε. Αν δεν βρισκόμαστε στη συνάρτηση που βρίσκεται η μεταβλητή που μας ενδιαφέρει (πχ: στη main(...)), τότε, πρέπει να πάμε στο stack frame της συγκεκριμένης συνάρτησης, ώστε να βλέπουμε τις σωστές τιμές.

⇢ (gdb) **frame** frameno (**f)**

Μετά μπορούμε να χρησιμοποιήσουμε διάφορες εντολές (πχ: print, info args, info locals κτλ).

Παράδειγμα 2 - Infinite Loop advanced

Αν για κάποιο λόγο τελειώσουμε με το debug στην συνάρτησή μας, τότε, πρέπει οπωσδήποτε να επιστρέψουμε στην τελευταία συνάρτηση που είχε σταματήσει ο gdb, δηλαδή, στη συνάρτηση με το frame 0. Στη συνέχεια, θέλουμε να εκτελέσουμε ολόκληρη την συνάρτηση. Γενικά, αυτό γίνεται πολύ εύκολα μέσω της εντολής:(αντί για next ή (ακόμα χειρότερα) step).

⇢ (gdb) **finish** (ή **continue**, **c**)

Με αυτές τις γνώσεις, προσπαθήστε να βρείτε το λάθος σε κάθε ένα πρόγραμμα.

Ένα πάρα πολύ καλό tutorial για όσους θέλουν να τα ασχοληθούν περαιτέρω→ link

Stack Frame

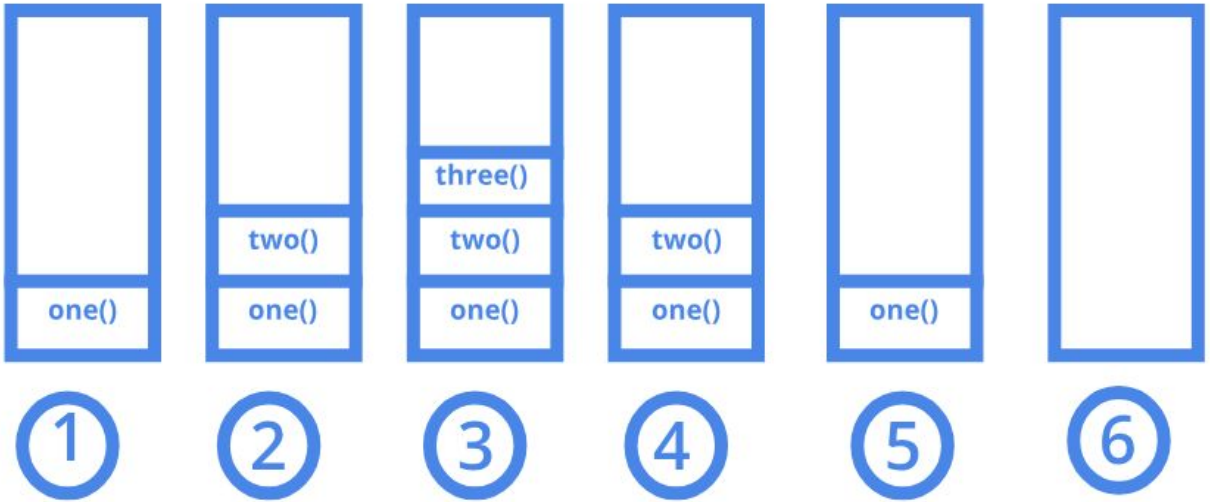
- Κάθε συνάρτηση έχει το δικό της χώρο μέσα στη μνήμη! (frame).
- Μέσα στο frame της, έχει όλες τις μεταβλητές της (τοπικές μεταβλητές)
- Όταν μία συνάρτηση καλεί μία άλλη, το frame της καινούργιας θα μπει πάνω από το frame (stack) αυτής που θα την καλέσει (caller - callee) - Δείτε το επόμενο slide!
- Έτσι, αν έχουμε καλέσει (πχ) από τη main() την do_task(...), και όσο εκτελείται η do_task(...) θέλουμε να δούμε τι τιμή έχει μία μεταβλητή από τη main(), τότε, πρέπει να μεταβούμε στο frame της main(), να κάνουμε τα print μας, και μετά να γυρίσουμε στο frame της do_task(...), ώστε να συνεχίσει η εκτέλεση του κώδικα από τον gdb.
- Μέσα σε οποιοδήποτε frame, γράψτε “info locals, info args, info vars”. Πειραματιστείτε!

```
function one(){
  two();
}

function two() {
  three();
}

function three() {
  console.trace("Call
              Stack");
}
```

Call Stack



“But you're out of your mind,” they said with a shrug.
“The customer's happy; what's one little bug?”
But he was determined. The others went home.
He spread out the program, deserted, alone.
The cleaning men came. The whole room was cluttered
With memory-dumps, punch cards. “I'm close,” he muttered.
The mumbling got louder, simple deduction,
“I've got it, it's right, just change one instruction.”
It still wasn't perfect, as year followed year,
And strangers would comment, “Is that guy still here?”
He died at the console, of hunger and thirst.
Next day he was buried, face down, nine-edge first.
And the last bug in sight, an ant passing by,
Saluted his tombstone, and whispered, “Nice try.”

GNU Humor