Veel silumisest

Silumise valikud

Paber ja pliiats, kummipardike (*rubber ducky*), …

print laused ja logimine (väärtused, tegevused)

Staatiline analüüs (*linter*)

gdb (+ kasutajaliidesed)

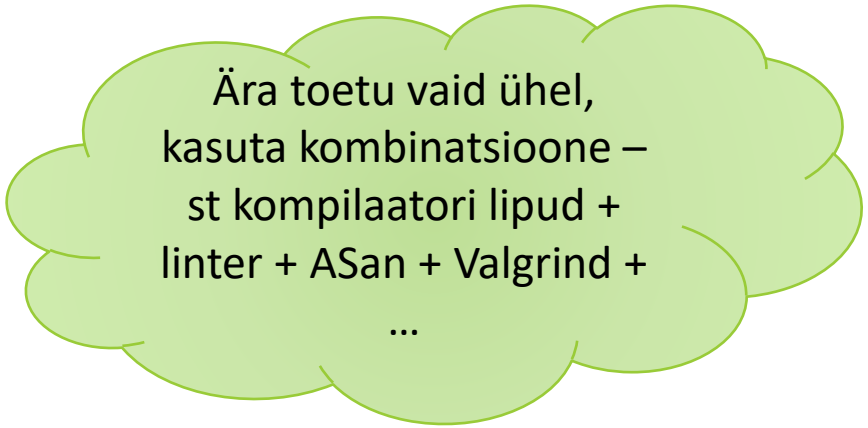
gcc lipud

AddressSanitizer (ASan)

Valgrind

IDE tööriistad

…

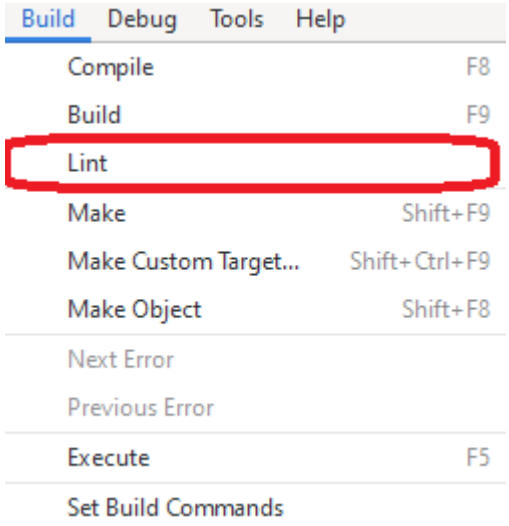


Linterid

Tutvusime Programmeerimine 1 aines

Teostab lähtekoodi staatilist analüüsi

Tüüpilised tööriistad: `cppcheck` ja `clang-tidy`
- `cppcheck` on paigaldatud laboriarvutites
- Geany kasutab vaikimisi tööriistana `cppcheck`
- Enda debian-põhisele Linuxile install: `sudo apt install cppcheck`
- Toetatud Linux, Windows, macOS



Ka GCC sisaldab staatilist analüsaatorit viimastes versioonides
- Kasutamiseks lisada lipp `-fanalyzer`
- Laboris käsurealt kompileerides määra kompilaatoriks `gcc-12`
- https://gcc.gnu.org/onlinedocs/gcc/Static-Analyzer-Options.html

Levinud praktika: sama koodifaili analüüsitakse kahe-kolme-… erineva staatilise analüsaatoriga

gdb

Koodi jooksutamine rida-rea haaval

Väärtuste vaatamine (ja muutmine) jooksvalt

Viimased jooksutatud read enne kokkujooksu

Jne

Kasutame näiteks läbi `geany-debugger` pistikprogrammi
https://blue.pri.ee/ttu/koodimisjuhendid/siluri-kasutamine-geanys/

Süviti kasutamiseks vaja käivitada käsurealt või kasutada mõnda põhjalikumat eeskihti.

gcc lipud

-Wall – enamik hoiatusi (komplektina)

-Wextra – täiendavad hoiatused (komplektina)

-Wconversion – hoiatused varjatud teisenduste kohta

-pedantic – range ISO C standardi jälgimine

-Wunreachable-code – kood milleni pole võimalik jõuda

-Wfloat-equal – murdarvude võrdlemise hoiatused

-Wshadow – samanimeliste muutujate korduv deklareerimine

-Werror – käsitle hoiatusi kui vigu (peatab kompileerimise hoiatuse korral)

-fanalyzer – GCC enda tööriist staatilise analüüsi tegemiseks

-g – silumiseks vajalik sümbolite tabel

…

Valgrind

Tuvastab vigu programmi töö ajal

- Algväärtustamata muutujad
- Mäluaadressidelt lugemine / kirjutamine, mis ei kuulu programmile
- NULL-viitade kasutamine
- Kehtetute viitade kasutamine
- Mälulekked, vabastamata ressursid
- Korduvad vabastamised
- …

Näitab vea kirjeldust, koodirida mis vea põhjustas, funktsiooni kutsete ajalugu, …

NB! Rakendus testimise ajal ligi 20x aeglasem

http://valgrind.org/docs/manual/manual.html

Ligipääs valgrindile

Laboriarvutites

- Paigaldatud, seadistatud ja valmis kasutamiseks
- Sealjuures ka kaugelt ligi (SSH, RDP)

Linux enda arvutis

- Debiani põhisel süsteemil: `sudo apt install valgrind`

MacOS

- Jah (amd64) aga ei (arm64), või siis kunagi tulevikus..

Windows

- Pole saadaval

Kasutamine

Kompileerimiseks vajalik lipp -g

- `gcc **-g** –Wall –Wextra –Wconversion -o my_program codefile.c`
- Ilma lisaliputa ei ole võimalik näha mis real viga juhtus

Kasutamine (minimaalselt – näitab vigu aga mitte detaile):

- `valgrind ./my_program`

Kasutamine (täiendava informatsiooniga):

- `valgrind --tool=memcheck --leak-check=full --show-leak-kinds=all ./myprogram`

Loe ka „Fun with valgrind“ – inglise keeles.

AddressSanitizer (ASan)

Tuvastab sarnase komplekti vigu nagu Valgrind
- Tuvastab täiendavalt pinu ja globaalmälu kasutamise vigu
- Ei suuda tuvastada algväärtustamata muutujate kasutamist

NB! Ei asenda täielikult Valgrindi – tavaliselt testitakse mõlemaga!

Parem ühilduvus operatsioonisüsteemidega (sh Windows, BSD, MacOS?)

Töötab kiiremini kui Valgrind (rakendus umbes 2x aeglasem tavapärasest, st ~10x kiirem kui Valgrind)

Kasutamine: Lisada kompilaatorile lipud `"-g -fsanitize=address"`

NB! Sedasi kompileeritud rakendust ei saa jooksutada Valgrindis! Tööriistad lähevad konflikti!

https://github.com/google/sanitizers/wiki/AddressSanitizer