Lab3 - Dezvoltatrea si depanarea programelor C sub Linux

**3.2.1 compilare si rulare**

-> apelare compilator gcc:

**gcc -Wall program.c -o program**

**-Wall = warning all -> activeaza afisarea tuturor advertismentelor**

**-o program -> output --> ne permite esa alegem numele programului executabil ( prin convensie programale executabile NU au extensie)**

OBS: comanda minima : gcc program.c --> in cazul asta programul compilat s.n. a.C

-> rularea unui program compilat:

**./program**

./ 🡪 indica interpretorului de comenzi faptul ca acesta se gaseste in folderul curent. In mod normal interpetorul cauta in folderele din vb de mediu $PATH

**3.2.2 depanarea programelor**

**3.2.2.1 compilare pentru depanarea**

**-g --> mai multa informatie in mom in care depanam programul**

🡪 in program se adauga informatii legate de codul sursa si putem inspecta valoare unei vb in timpul depanarii

**gdb program** -> depanarea propriu zisa;

-> incarca programul si il suspenda inainte de executi; se pot efectua comenzi de depanare

3.2.2.2 comenzi de date la depanare

**break source.c:lineNo** --> setarea unui breakpoint -> se suspenda executia programului atunci cand se ajunge la lineNo

**run** --> porneste executia programului;

argumentele programului se transmit aici!

**continue** --> se reia executia programului ce continua pana cand gaseste un breakpoint

**step** --> executa urmatoarea linie din cod

**next** --> exec urm linie din cod doar ca daca aceasta e un apel de functie, se va face tot apelul intr-un singur pas --> nu se intra efectiv in functie cu depanarea

**backtrace/bt** --> afiseaza stiva de apeluri in pct curent al programului

**print var** --> afiseaza valoarea variabilei var

**3.2.2.3 depanarea post-mortem**

-> examinarea starii unui program din momentul in care crapa – se compileaza in prealabil cu optiunea -g pt a adauga info referitoare la codul sursa

**🡪ulimit -c unlimited** --> activare core dump -> activarea starii unui program cand acesta crapa --> are efect doar pentru terminalul curent in care a fost rulata( se ruleaza o singura data)

* se ruleaza apoi programul, reproducand conditiile ce l-au facut sa crape --> dupa crash apare un folder core in folderul curent

🡪 se poate incarca in gdb cu comanda :

**gdb program core** --> incarcarea unui core dump in gdb

la incarcarea unui core dump, executia programului e inghetata => **putem** examina stiva de apeluri/ valorile vb cu **backtrace** sau **print** **, DAR NU** putem avansa executia programului cu **continue**, **step**, **next**

**3.2.2.4 depanarea prin mesaje**

-> mesajele afisate pt depanare ar trebui scrise separat sau in stderr

-> apeluri precum printf()/fprinf() scriu textul afisat intr-un buffer si pot ajunge pe ecran cu intarziere **==> putem forta afisarea cu fflush()**

-> afisarea de info intr-un log (jurnal) e o varianta de investigare – uneori erorile se reproduc doar pe calculatorul unor cilenti si nu al programatorului

**3.2.3 depistarea problemelor de tip memory leak**

--> Problema: **dealocarea memoriei** ( nu avem garbage collector in c)

**valgrind** --> permite depistarea problemelor de dealocare a memoriei

**valgrid program arg1 arg2** --> testarea unui program cu valgrind

🡪 valgrind anunta daca s-au gasit probleme de tip memory leak

**valgrind --leak-check=full --show-leak-kinds=all program arg1 arg2**

🡪se vor afisa linile la care s-a alocat memorie ce nu a mai fost eliberata

**3.3 Intrebari recapitulative**

1. cu ce comanda se compileaza un program C sub Linux?

r: **gcc -Wall program.c -o program**

2. De ce e nevoie sa scriem ./ inaintea numelui unui program pentru a-l rula?

r**: pentru a indica interpretorului de comenzi faptul ca programul se gaseste in folderul curent ( interpretorul in mod normal cauta programele in folderele din vb de mediu $PATH)**

3. Ce este un **breakpoint** in contextul depanarii?

r: **e o intructiune ce permite suspendarea executiei programului atunci cand se ajunge la linia pe care e setat**

4. Ce optiune trebuie sa adaugam la comanda de compilare pentru ca un executabil sa contina info de debug?

r: **-g**

--> **depanare propriu zisa : gdb program**

5. ce face utilitarul valgrind?

r: **anunta daca in program s-au gasit probleme de tip memory leak;** permite depistarea problemelor de dealocare a memoriei

valgrind program arg1 arg2

valgrind --leak-check=full --show-leak-kinds= all program arg1 arg2 ==> afiseaza si linile la care s-a alocat memorie ce nu a mai fost eliberata