Samouczek 1

Gnu Debugger (GDB) i asembler

podstawy

Prerekwizyty[1]

Dostęp do maszyny w architekturze x86-64 z zainstalowanym systemem rodziny Linux, z zainstalowanym oprogramowaniem: gcc, gdb, binutils, make. Elementarna umiejętność korzystania z powłoki, np. bash.

Cele

Nabycie elementarnych umiejętności posługiwania się debuggerem.

GNU Debugger[2]

Zastosowanie: śledzenie programu podczas wykonania.

Ćwiczenie 1.

Pobierz i wypakuj plik, następnie przedź do katalogu `rec3`. W linii poleceń wydaj komendę:

```
$ make
```

spowoduje to zbudowanie plików wykonywalnych ze zródeł, na podstawie zawartości pliku `Makefile`. W swoim ulubionym edytorze tekstu wyświetl zawartość pliku `act1.c`. Wersję wykonywalną tego pliku będziesz analizować za pomocą gdb. W linii poleceń wydaj komendę:

```
$ gdb act1
```

spowoduje to wczytanie przez `gdb` pliku wykonywalnego `act1`, wraz z definicjami symboli (nazwy funkcji, zmiennych) i uruchomienie linii poleceń debuggera. W tym momencie program `act1` nie jest jeszcze uruchomiony. Wpisuj kolejne polecenia w linii poleceń gdb .

```
(gdb) break main
```

Zakłada tzw. pułapkę, czyli punkt w którym debugger powinien zatrzymać wykonanie programu. W tym przykładzie pułapka założona jest na wejściu do funkcji main.

---

[1] Poinformuj prowadzącego wykład lub ćwiczenia, jeśli nie spełniasz prerekwizytów

[2] Źródłem ćwiczeń jest kurs Introduction to Computer Systems na uniwersytecie Carnegie Mellon, USA http://www.cs.cmu.edu/afs/cs/academic/class/15213-m18/www/

```
(gdb) run 777
```

uruchamia program `act1` z argumentem `777`. Wykonanie programu wstrzyma się na wcześniej zdefiniowanej pułapce. Debugger powinien wypisać komunikat podobny do:

```
Breakpoint 1, main (argc=2, argv=0x7fffffffe378) at act1.c:5
5         int ret = printf("%s\n", argv[argc-1]);
```

Następnie wpisz
`(gdb) continue`
spowoduje to kontynuację działania programu aż do napotkania następnej pułapki. Ponieważ w tym wypadku takiej nie ma, program `act1` kończy działanie.
`(gdb) clear main`
usuwa wcześniej zdefiniowaną pułapkę
`(gdb) run Ala`
Ponownie uruchamia program `act1`, tym razem z parametrem `Ala`. Program kończy się bez zatrzymywania, pułapka została wcześniej skasowana.
`(gdb) disassemble main`
Deasembluje kod funkcji main.
`(gdb) print (char*) adres`
wypisze ciąg znaków spod adresu adres. Za adres możesz w tym wypadku podstawić liczbę po znaku `#` w wierszu z instrukcją `lea`. Czym jest ten napis?
Ustaw pułapkę na funkcję `main` i uruchom program (`break main`, później `run`). Napisz
`(gdb) print argv[1]`
Jaka wartość zostanie wypisana?
`(gdb) quit`
Kończy pracę debuggera. Musisz wyrazić zgodę na zakończenie analizowanego programu.

Ćwiczenie 2

Przejdź do analizy programu `act2`.

```
$ gdb act2
(gdb) break main
(gdb) run
(gdb) print /x $rsi
```

Ostatnie polecenie wypisze wartość rejestru `%rsi` w systemie szesnastkowym. Porównaj z `print $rsi`.
Wydrukuj również zawartość rejestru `%rdi`. Co zawierają te rejestry? Gdzie jest wartość `argc` a gdzie `argv`?
`(gdb) disassemble main`
Zauważ, że `main` wywołuje funkcję `stc`

```
(gdb) break stc
(gdb) continue
```

```
(gdb) run 18213
```
odpowiedz y
```
(gdb) continue
```
Która funkcja wykona się teraz?

```
(gdb) disassemble
(gdb) stepi
```

Komenda stepi powoduje wykonanie pojedynczej instrukcji asemblera. Możesz ją powtórzyć naciskając ENTER. Ogólnie, naciśnięcie ENTER powtarza ostatnią komendę. GDB pamięta historię komend, możesz je wybrać za pomocą klawiszy strzałek (w górę i w dół). Użyj klawisza TAB do autouzupełnienia polecenia (np `cont` i TAB uzupełni się do `continue`)

```
(gdb) disassemble
```
Zauważ przesunięcie strzałki => po lewej stronie ekranu na kolejną instrukcję funkcji.

```
(gdb) quit
```

Ściągę zawierającą przydatne komendy GDB znajdziesz tu.

Ćwiczenie 3

Wyświetl zawartość pliku act3.c, gdzie znajduje się główna część programu. Użyj debuggera GDB by dowiedzieć się, jakie dwie wartości liczbowe należy podać jako argumenty programu, by wypisał on komunikat `good args!`

Gdzie znajduje się treść funkcji `compare`? Wskazówka: zobacz plik `Makefile`.