11 Debugowanie, gdb

Niniejsze ćwiczenia polegają przede wszystkim na znajdowaniu błędów z wykorzystaniem `gdb`. Ponieważ przykłady są stosunkowo proste, wprawne oko programisty jest wstanie wskazać błędy bez użycia takiego narzędzia – niemniej proszę by użyć debuggera, gdyż on jest przedmiotem tych ćwiczeń, a nie sam język C.

Ściągnij plik `http://srodowisko.tcs.uj.edu.pl/files/11-debuggery.zip`.

Ćwiczenie 1. Plik: `11power.c`. Program wylicza wartość 2^4 za pomocą rekurencyjnej funkcji potęgującej `power`

- Odpowiednio skompiluj i uruchom program w środowisku `gdb`.
- Prześledź krok po kroku działanie funkcji `power`.
- Wywołując komendę `step` zbyt wiele razy można wylądować wewnątrz funkcji `printf`. Nie chcemy wykonywać kodu krok po kroku nie mając dostępu do źródeł. Jak zatem wyjść z funkcji `printf` ale tak by nic więcej się nie wydarzyło, np. by program się nie zakończył?
- Bez modyfikacji programu wylicz za pomocą `power` wartość 31^5.
- Jaka jest zawartość stosu gdy `power` wywołane jest z debuggera?
- Uruchom program jeszcze raz. Jaka jest zawartość stosu gdy zwracana jest wartość 1 z funkcji `power`? Jakie są wtedy zmienne lokalne we wszystkich wywołaniach `power`?

Ćwiczenie 2. Plik: `11qsort.c`. Program wczytuje liczbę n a następnie n liczb, które ma posortować za pomocą ręcznie zaimplementowanego algorytmu quicksort. Niestety implementacja zawiera błąd i program kończy działanie komunikatem `Segmentation fault`.

Za pomocą `gdb` wskaż dokładniej co prowadzi do takiego komunikatu. Następnie, przeglądając zmienne i stos, wskaż gdzie znajduje się błąd.

Ćwiczenie 3. Plik: `11dbg.c`. Program ma za zadanie wczytać do 10 argumentów i je zsumować. Niestety, tak jak jest napisany, zawiera on wiele błędów. W szczególności, gdy zostanie uruchomiony, pojawia się komunikat: `stack smashing detected` sugerujący, że kluczowe fragmenty stosu zostały nadpisane.

Wykorzystując `valgrind` sprawdź zachowanie odnośnie pamięci. Skorzystaj z `gdb` wskaż i napraw błędy.

Ćwiczenie 4. Wykonaj wybrane zadanie z uzyciem graficznego debuggera, np. `KDbg` lub `Nemiver`.