# SKSP Lab 3

Mikalai Stelmakh
Anna Sztanga

**Wstęp:**

Uruchomiliśmy OpenWRT na RaspberryPi i połączyliśmy się z nim poprzez konsolę komputera w sali. Na komputer pobraliśmy OpenWRT z przygotowanym SDK spod linku: https://downloads.openwrt.org/releases/21.02.1/targets/bcm27xx/bcm2711/openwrt-sdk-21.02.1-bcm27xx-bcm2711_gcc-8.4.0_musl.Linux-x86_64.tar.xz

Rozpakowaliśmy pakiet. Uruchomiliśmykonfigurację poprzez make menukonfig. Wyłączyliśmy następujące parametry:

- W „Global Build Settings”
    - Select all target specific packages by default
    - Select all kernel module packages by default
    - Select all userspace packages by default
    - Cryptographically sign package lists
- W „Advanced configuration options”
    - „Automatic removal of build directories”

**Zadanie 1:**

Pobraliśmy i rozpakowaliśmy paczkę spod linku :
https://moodle.usos.pw.edu.pl/pluginfile.php/217384/mod_folder/content/0/WZ_W03_przyklady.tar.xz

Wykonaliśmy komendę „export LANG=C”, a następnie dopisaliśmy na koniec pliku „feeds.conf.default” dodaliśmy linijkę
„src-link skps /home/user/skps22_sztanga_stelmakh/lab3/demo1_owrt_pkg”.
Wykonaliśmy polecenia:

```
./scripts/feeds update -a
./scripts/feeds install -p skps -a
```

Po tym wszystkim ponownie uruchomiliśmy menuconfig i w „examples” zaznaczyliśmy do kompilacji pakiet „demo1”. Wykonaliśmy polecenie „make package/feeds/skps/demo1/compile” by wykonać kompilację. Nazwą pliku wynikowego było „demo1_1.0-1_aarch64_cortex-a72.ipk”. Przenieśliśmy ów plik na Raspberry używając gita. Zainstalowaliśmy paczkę z użyciem opkg:

```
opkg install demo1_1.0-1_aarch64_cortex-a72.ipk
```

Po wykonaniu tych wszystkich kroków można było uruchomić program demo1 poprzez wpisanie jego nazwy. Program faktycznie działał.

**Zadanie 2:**

Pobraliśmy „WZ_W03_przyklad_extbr.tar.xz” i rozpakowaliśmy. Utworzyliśmy katalog „zad2” w którym stworzyliśmy katalogi dla obu pakietów. Struktura tych katalogów była zrobiona na wzorze struktury z demo1. Do podkatalogów src skopiowaliśmy pliki źródłowe i ich makefile. Na tym samym poziomie co katalog src utworzyliśmy makefile naszych pakietów. Zostały one utworzone poprzez modyfikację makefila demo1. Najważniejszymi zmianami była nazwa pakietu oraz dodanie biblioteki ncurses dla pakietu worms (+libncurses). Dla pakietu buggy istotne było też dodanie flagi dla debugera (TARGET_CFLAGS += -ggdb3).

Do pliku „feeds.conf.default „ dodaliśmy kolejną linijkę z lokalizacją naszych nowych paczek. W menukonfig dodaliśmy zarówno worms jak i buggy. By wygenerować plik .ipk wykonaliśmy komendy:

```
/scripts/feeds update -a
/scripts/feeds install -p skps2 -a
```

Budowanie zakończyło się pomyślnie, więc przenieśliśmy plik na Raspberry, gdzie zainstalowaliśmy oba pakiety używając opgk, tak jak w zadaniu 1. Gra worms (a właściwie snake) działała poprawnie. Programy bug1, bug2 i bug3 miały oczywiste błędy w kodzie źródłowym.

**Debugowanie:**

Używając opkg zainstalowaliśmy gdb oraz gdbserver na RaspberryPi. Uruchomiliśmy serwer:

gdb-server localhost:8000 bug1

Na komputerze zaś połączyliśmy się z serwerem gdb poprzez polecenie:

./scripts/remote-gdb 10.42.0.156:8000

*Bug1:*

Użyliśmy polecenia „directory” by wskazać debugerowi pliki źródłowe. Ustawiliśmy breakpoint na funkcji main. Wykonaliśmy zaledwie jeden krok i program zgłosił błąd. Po wyświetlaniu wartości pętli widać, że stało się to przy pierwszym wykonaniu. Wnioskujemy, że tablica do której są wpisywane wartości nie została zainicjowana.

```
Breakpoint 1, main () at bug1.c:9
9       bug1.c: No such file or directory.
(gdb) directory /home/user/skps22_sztanga_stelmakh/lab3/zad2/buggy/src
Source directories searched: /home/user/skps22_sztanga_stelmakh/lab3/zad2/buggy/
src:$cdir:$cwd
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/user/skps22_sztanga_stelmakh/openwrt-sdk/build_dir/targe
t-aarch64_cortex-a72_musl/buggy-1.0/.pkgdir/buggy/usr/bin/bug1

Breakpoint 1, main () at bug1.c:9
9         for(i=0;i<1000;i++) {
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x000000000040046c in main () at bug1.c:9
9         for(i=0;i<1000;i++) {
(gdb)
```

*Bug2:*

Postawiono serwer w taki sam sposób jak dla bug1. Również użyto komendy directory.

Ustawiono breakpoint na main a następnie kontynuowano. Program zatrzymał się na błędzie. Używając polecenia print sprawdziliśmy które to wykonanie pętli. Naszą teorią było, że pętla wykracza poza zakres tablicy i w 1008 iteracji trafia na pamięć do której nie ma dostępu. Sprawdziliśmy to więc testując kilka różnych pozycji w tablicy. Wykonaliśmy również backtrace.

```
Continuing.

Program received signal SIGSEGV, Segmentation fault.
main () at bug2.c:8
8           table[i]=i;
(gdb) print i
$1 = 1008
(gdb) info frame
Stack level 0, frame at 0x7fffffd00:
 pc = 0x40046c in main (bug2.c:8); saved pc = 0x7ff7f93190
 source language c.
 Arglist at 0x7fffffd00, args:
 Locals at 0x7fffffd00, Previous frame's sp is 0x7fffffd00
(gdb) p table[1002]
$2 = 1002
(gdb) p table[1010]
Cannot access memory at address 0x412008
(gdb) p table[108]
$3 = 108
(gdb) p table[1008]
Cannot access memory at address 0x412000
(gdb) p table[1007]
$4 = 1007
(gdb)
```

```
(gdb) p table[1008]
Cannot access memory at address 0x412000
(gdb) p table[1007]
$4 = 1007
(gdb) backtrace
#0  main () at bug2.c:8
(gdb)
```

*Bug3:*

Analogicznie do poprzednich punktów połączono się z serwerem i użyto polecenia „directory”.

Program po uruchomieniu dawał dziwne wyniki. Z kodu wynikałoby, że tablica s2 nie jest nigdy modyfikowana, a jednak jej wartość nie równała się początkowej. Ustawiliśmy więc watchpoint na początek drugiej tablicy by sprawdzić kiedy zaczyna być modyfikowana. Kiedy debuger wykrył zmianę wartości s2[0] zatrzymał wykonywanie. Sprawdziliśmy numer iteracji. Iterator pętli wynosił 11. Doszliśmy do wniosku, że to od tego momentu s2 zaczyna być niezamierzenie modyfikowane.

```
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/user/skps22_sztanga_stelmakh/openwrt-sdk/build_dir/targe
t-aarch64_cortex-a72_musl/buggy-1.0/.pkgdir/buggy/usr/bin/bug3

Breakpoint 1, main () at bug3.c:12
12          for(i=0;i<24;i++) {
(gdb) set breakpoint auto-hw off
(gdb) set can-use-hw-watchpoints 0
(gdb) watch s2[0]
Watchpoint 2: s2[0]
(gdb) c
Continuing.

Watchpoint 2: s2[0]

Old value = 97 'a'
New value = 74 'J'
main () at bug3.c:12
12          for(i=0;i<24;i++) {
(gdb) print i
$1 = 11
(gdb)
```