

Sprawozdanie laby3

Pierwszy pakiet

- pobraliśmy sdk dla RP4
- pobraliśmy plik z moodle zgodnie z instrukcją
- rozpakowaliśmy plik do odpowiedniej lokalizacji i z niego używaliśmy tylko folderu demo1_owrt_pkg w którym był już przygotowany makefile i paczka
- do skompilowania paczki demo1 użyliśmy sdk dla RP4 na naszym komputerze w następujący sposób:

- dodanie ścieżki do pliku feeds.conf.default:

```
src-link skps ścieżka/demo1_owrt_pkg
```

- update sdk

```
./scripts/feeds update -a
```

- instalacja paczki

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

- dodanie paczek w nconfig
- skompilowanie paczek demo1 i demo1mak
- przrzucamy pliki na RP4 za pomocą http server
- instalacja plików przy użyciu opkg
- test poprzez wywołanie komendy np: demo1

Pakiety worms i buggy

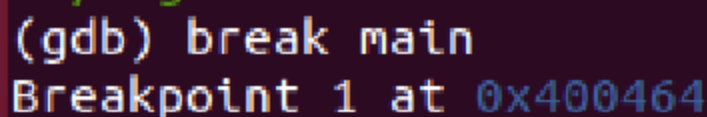
- zmiana całej struktury paczki danej, na folder bugs z podfolderem src z kodem, w folderze bugs makefile
- zmiana struktury worms analogicznie
- zmienione makefile znajdują się na repozytorium
- Zainstalowanie libncurses w sdk ponieważ pakiet worm potrzebuje tej biblioteki
- Instalacja analogiczna do instalacji pakietów demo z punktu pierwszego
- uruchomienie worms → działają
- uruchomienie bugs → nie działają (działają nie poprawnie)

Debugowanie

- pobraliśmy server gdb i gdbserver na RP4
- uruchomiliśmy gdbserver na RP4 `gdbserver :9012 /usr/bin/bug3`
- uruchomiliśmy remote gdb z odpowiednim skryptem `./scripts/remote-gdb <ip>9012`
`./build_dir/target-aarch64_cortex-a72_musl/bugs-2.1/.pkgdir/bugs/usr/bin/bug3`
- testowaliśmy różne funkcjonalności gdb jak zostało napisane w instrukcji, na różnych plikach bug, poniżej screeny z różnymi przykładami
- przy każdym odpalaniu gdb używaliśmy komend (napisanych poniżej) w celu dodania możliwości utworzenia wielu breakpoint-ów

```
set breakpoint auto-hw off  
set can-use-hw-watchpoints 0
```

- wykonanie
 - ustawienie breakpointu:



```
(gdb) break main  
Breakpoint 1 at 0x400464
```

- praca krokowa i podgląd wartości zmiennej przy każdym kroku:

```
(gdb) display i
1: i = 0
(gdb) step
12      in bugs-2.0/bug3.c
1: i = 1
```

- podgląd wartości zmiennej:

```
(gdb) print i
$1 = 3
```

- podgląd wartości stosu

```
(gdb) x $sp
0x7fffffffdd40:  0xffffffffd50
```

- backtrace

```
(gdb) backtrace
#0  main () at bugs-2.0/bug3.c:12
(gdb) █
```

- wykorzystanie watchpointów

```
breakpoint 1, 0x0000000000000000
(gdb) watch i
Hardware watchpoint 2: i
(gdb) c
Continuing.
```

