Systemy komputerowe w sterowaniu i pomiarach (SKPS)
Instrukcja do laboratorium 3

mgr inż. Dawid Seredyński, dr hab. inż. Wojciech Zabołotny
Ostatnia aktualizacja: 04.04.2022

Informacje wstępne

Cel laboratorium

Celem laboratorium jest praca z OpenWRT, w szczególności:
- zapoznanie się z SDK
- zbudowanie własnych pakietów (za pomocą SDK)
- debugowanie pakietu (zdalne, przez gdb)

Laboratorium 3 jest za 6 punktów, w tym:
- 1 - wejściówka (UWAGA: osoby, które otrzymają 0 punktów z wejściówki mogą nie zostać dopuszczone do dalszej części laboratorium)
- 5 - praca na zajęciach

Zakres wejściówki

1. OpenWRT - SDK, tworzenie pakietu, Makefile, instalacja pakietów (opkg)
2. debugowanie (gdb) - co można zrobić za pomocą poleceń: backtrace, print, display, break, step, directory, continue, info, watch, delete, set, show.

Plan laboratorium 3:

1. Zbudowanie i uruchomienie pakietu dostarczonego w materiałach do wykładu.
2. Stworzenie pakietów OpenWRT na bazie dostarczonych kodów źródłowych.
3. Zdalne debugowanie programów za pomocą gdb.

Praca na zajęciach

W trakcie zajęć proszę na bieżąco tworzyć raport z wykonanych prac. Kolejne etapy należy przedstawiać prowadzącemu.

Oznaczenia użyte w tym dokumencie

Polecenia, które są wpisywane w konsoli **Linuxa** na komputerze **host (PC)** oznaczone są jasnoszarym tłem, np.:
`ls -la /`

Polecenia, które są wpisywane w konsoli **Linuxa** na komputerze **RPi** oznaczone są malinowym tłem (raspberry, czyli malina), np.:
`ls -la /`

Materiały

Wykład WZ_W03 z SKPS:
https://moodle.usos.pw.edu.pl/pluginfile.php/217384/mod_folder/content/0/WZ_W03_Rozszerzanie_BR_OpenWRT.pdf

Zestaw przykładowych pakietów, w tym prosty pakiet dla OpenWRT:
https://moodle.usos.pw.edu.pl/pluginfile.php/217384/mod_folder/content/0/WZ_W03_przyklady.tar.xz

Pakiety dla Buildroot, które należy przerobić na pakiety OpenWRT:
https://moodle.usos.pw.edu.pl/pluginfile.php/217384/mod_folder/content/0/WZ_W03_przyklad_extbr.tar.xz

Wiadomości dot. tworzenia pakietów:
https://openwrt.org/docs/guide-developer/packages

Dodawanie pakietów:
https://openwrt.org/docs/guide-developer/feeds#feed_configuration

Debugowanie:
https://openwrt.org/docs/guide-developer/gdb

Zadania

1. Pierwszy pakiet

Proszę pobrać, rozpakować i zbudować pakiet demo1 dostępny w paczce *WZ_W03_przyklady.tar.xz*.
Sposób dodania pakietu do OpenWRT z wykorzystaniem SDK jest opisany na wykładzie *WZ_W03*.

Następnie proszę zainstalować pakiet na OpenWRT (należy ściągnąć plik .ipk i zainstalować przez opkg).
UWAGA: podczas próby ponownej instalacji tego samego pakietu (np. po dokonaniu zmian i przebudowaniu) opkg może odmówić wykonania operacji, jeśli numer wersji pakietu (określony w Makefile'u) będzie taki sam jak poprzednio. W związku z tym należy wcześniej usunąć pakiet za pomocą opkg lub instalować z opcją *--force-reinstall*.

2. Pakiety "worms" i "buggy"

Proszę pobrać i rozpakować paczkę *WZ_W03_przyklad_extbr.tar.xz*. Paczka ta zawiera dwa pakiety dla Buildroot:
- *buggy* - trzy programy z błędami
- *worms* - gra "Worms" (a właściwie "Snake") w ncurses

W ramach zajęć należy stworzyć dwa nowe pakiety dla OpenWRT na podstawie dwóch ww. pakietów dla Buildroot.
UWAGA: pakiet worms jest zależny od *ncurses*. Należy zainstalować *ncurses* w SDK poleceniem
`./scripts/feeds update -a`

3. Debugowanie zdalne

Na RPi należy zainstalować *gdb* i *gdbserver* za pomocą managera pakietów.

Proszę dla trzech programów z pakietu *buggy* znaleźć błędy za pomocą debuggera gdb.

Na RPi uruchamiamy *gdbserver*, a na host uruchamiamy skrypt *remote-gdb* (zgodnie z instrukcją na stronie openwrt.org).

Aby dodać katalog ze źródłami, należy użyć polecenia directory w terminalu *gdb*.
Aby zakończyć pracę *gdbserver* należy w zdalnym terminalu *gdb* (na host) wpisać:
`monitor exit`

W ramach doświadczeń z gdb należy wykonać:
- ustawienie breakpointu

- pracę krokową
- podgląd wartości zmiennej (jednorazowy i przy każdym kroku)
- podgląd stosu
- backtrace
- wykorzystanie watchpoint'ów w programie bug3, aby sprawdzić kiedy następuje zapisanie wartości pod niewłaściwym adresem, np. w s1[10].

UWAGA: ARM wspiera tylko 1 H/W watchpoint. Aby włączyć S/W watchpoint, należy wykonać polecenia (w zdalnym terminalu *gdb*):
`set breakpoint auto-hw off`
`set can-use-hw-watchpoints 0`

Krótkie objaśnienie niektórych poleceń *gdb*

Fragmenty z dokumentacji *gdb* (pełny opis jest dostępny w terminalu *gdb*, polecenie *help*):

`info` - Generic command for showing things about the program being debugged.
Examples:
`info breakpoints, info b` -- Status of specified breakpoints (all user-settable breakpoints if no argument).
`info watchpoints` -- Status of specified watchpoints (all watchpoints if no argument).

`show` - Generic command for showing things about the debugger.
Examples:
`show breakpoint` -- Breakpoint specific settings.
`show can-use-hw-watchpoints` -- Show debugger's willingness to use watchpoint hardware.
`show directories` -- Show the search path for finding source files.

`set` - Evaluate expression EXP and assign result to variable VAR.
Usage: set VAR = EXP
Examples:
`set breakpoint` -- Breakpoint specific settings.
`set can-use-hw-watchpoints` -- Set debugger's willingness to use watchpoint hardware.

`backtrace, where, bt`
Print backtrace of all stack frames, or innermost COUNT frames.

`print, inspect, p` - Print value of expression EXP.
Usage: print [[OPTION]... --] [/FMT] [EXP]

`display` - Print value of expression EXP each time the program stops.
Usage: display[/FMT] EXP
/FMT may be used before EXP as in the "print" command.

`break, brea, bre, br, b` - Set breakpoint at specified location.

`step, s` - Step program until it reaches a different source line.
Usage: step [N]
Argument N means step N times (or till program stops for another reason).

`directory` - Add directory DIR to beginning of search path for source files.

`continue, fg, c` - Continue program being debugged, after signal or breakpoint.
Usage: continue [N]
If proceeding from breakpoint, a number N may be used as an argument,
which means to set the ignore count of that breakpoint to N - 1 (so that
the breakpoint won't break until the Nth time it is reached).

`watch` - Set a watchpoint for an expression.
Usage: watch [-l|-location] EXPRESSION
A watchpoint stops execution of your program whenever the value of
an expression changes.

`delete, del, d` - Delete all or some breakpoints.
Usage: delete [BREAKPOINTNUM]...
Arguments are breakpoint numbers with spaces in between.
To delete all breakpoints, give no argument.
Also a prefix command for deletion of other GDB objects.