Systemy Komputerowe w Sterowaniu i Pomiarach

Laboratorium 3

Aleksander Kruk
Michał Sobiech

15 listopada 2023

Zadanie 1 Pierwszy pakiet

Użyte komendy – w folderze ~/SKPS

Korzystamy z openwrt zainstalowanego na poprzednich laboratoriach
Ręczne pobranie i rozpakowanie `WZ_W03_przyklady.tar.xz`

`wget 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`	Pobranie sdk openwrt
`tar -xf openwrt-sdk-21.02.1-bcm27xx-bcm2711_gcc-8.4.0_musl.Linux-x86_64.tar.xz`	Rozpakowanie sdk
`mv openwrt-sdk-21.02.1-bcm27xx-bcm2711_gcc-8.4.0_musl.Linux-x86_64.tar.xz openwrt-sdk`	Skrócenie nazwy
`make menuconfig`	Uruchomienie konfiguracji w folderze openwrt-sdk

Wybrane opcje	Status
`Global Build Settings::Select all target specific packages by default`	✗
`Global Build Settings::Select kernel module packages by default`	✗
`Global Build Settings::Select all userspace packages by default`	✗
`Global Build Settings::Cryptographically sign package lists`	✗
`Advanced configuration options::Automatic removal of built directories`	✗

Użyte komendy

`export LANG=C`	Dodanie wymaganej zmiennej systemowej
`vi ~/SKPS/openwrt-sdk/feeds.conf.default`	Otwarcie pliku konfiguracyjnego
`src-link skps ~/SKPS/openwrt-sdk/demo1_owrt_pkg`	Dodanie do pliku konfiguracyjnego
`~/SKPS/openwrt-sdk/scripts/feeds update -a`	Aktualizacja pakietów
`~/SKPS/openwrt-sdk/scripts/feeds install -p skps -a`	Instalacja pakietów skps
`make menuconfig`	Uruchomienie konfiguracji

Wybrane opcje	Status
`Examples::demo1`	✓
`Examples::demo1mak`	✓

Użyte komendy

`make ~/SKPS/openwrt-sdk/package/feeds/skps/demo1/compile`	Kompilacja pakietów z przykładu
`make ~/SKPS/openwrt-sdk/package/feeds/skps/demo1mak/compile`	
`cd ~/SKPS/openwrt-sdk/bin/packages/aarch64_cortex-a72/skps/`	
`python3 -m http.server`	Uruchomienie serwera do transferu plików
`wget http://10.42.0.1:8000/demo1_1.0-1_aarch64_cortex-a72.ipk`	Pobranie pliku pakietu
`opkg install demo1_1.0-1_aarch64_cortex-a72.ipk`	Zainstalowanie pakietu na płytce
`demo1`	uruchomienie zainstalowanego pakietu

Zadanie 2 Pakiety „worms” i „buggy”

Użyte komendy – w folderze ∼/SKPS

Pobieramy ręcznie WZ_W03_przykład_extbr	
`mkdir ∼/SKPS/zad_2_owrt_pkg`	Utworzenie folderu na nasze pakiety
`vi ∼/SKPS/openwrt-sdk/feeds.conf.default`	Otwarcie pliku konfiguracyjnego
`src-link skps ∼/SKPS/zad_2_owrt_pkg`	Zamiana folderu z poprzedniego zadania
`∼/SKPS/openwrt-sdk/scripts/feeds install libncurses`	Instalacja ncurses potrzebnego do wormsa
`∼/SKPS/openwrt-sdk/scripts/feeds update -a`	Aktualizacja pakietów
`∼/SKPS/openwrt-sdk/scripts/feeds install -p skps -a`	Instalacja pakietów skps
`make menuconfig`	Uruchomienie konfiguracji

Wybrane opcje	Status
`Examples::worms`	✓
`Examples::buggy`	✓

Użyte komendy

`make ∼/SKPS/openwrt-sdk/package/feeds/skps/buggy/compile`	Kompilacja naszych pakietów
`make ∼/SKPS/openwrt-sdk/package/feeds/skps/worms/compile`	
`cd ∼/SKPS/openwrt-sdk/bin/packages/aarch64_cortex-a72/skps/`	
`python3 -m http.server`	Uruchomienie serwera do transferu plików
`wget http://10.42.0.1:8000/buggy_1.0-1_aarch64_cortex-a72.ipk`	Pobranie pliku pakietu buggy
`wget http://10.42.0.1:8000/worms_1.0-1_aarch64_cortex-a72.ipk`	Pobranie pliku pakietu worms
`opkg install buggy_1.0-1_aarch64_cortex-a72.ipk`	Instalacja buggy
`opkg install worms_1.0-1_aarch64_cortex-a72.ipk`	Instalacja worms

Zadanie 3 Debugowanie zdalne

Użyte komendy

`opkg update`	
`opkg install gdb`	Pobieramy debuggera
`opkg install gdbserver`	Pobieramy zdalnego debuggera
Po dodaniu flag gdb do plików makefile rekompilujemy buggiego	
`make ∼/SKPS/openwrt-sdk/package/feeds/skps/buggy/compile CONFIG_DEBUG=y V=s`	
`gdbserver :9000 usr/bin/bug[number]`	Uruchomienie debuggera zdalnego dla podanego programu
`∼/SKPS/openwrt-sdk/scripts/remote-gdb [adres płytki]:9000`	Start debuggowania
`∼/SKPS/openwrt-sdk/build-dir/target-aarch64_cortex-a72_musl`	
`/buggy-1.0/.pkdir/buggy/usr/bin/bug[number]`	

Program 1

Użyte komendy

`break main`	
`display table`	Obserwacja table i indeksu do iteracji
`display i`	
`s`	
Postępując linijka po linijce okazuje się że tablica nie jest inicjowana (jest pełna śmieci)	
Otrzymujemy sygnał SIGSEGV segmentation fault	

Program 2

Użyte komendy

`break main`	
`display table`	Obserwacja zmiennych table i indeksu do iteracji
`display i`	
`s`	
`continue`	
Pozwalając działać programowi obserwujemy że wartość zmiennej iteracji wychodzi poza rozmiar tablicy	

Program 3

Nie zdążyliśmy