Module 3

Cross-compilation and remote debugging

Cross-compilation (1)

- Different sorts of compilation types:
  - *native* – building code for the host that runs a compiler
  - *cross* – building code for different target
  - *canadian* – building a cross-compiler on a host that produces code for another target
  - many more exotic combinations (you name it)
- Install the cross-compiling toolchain for Raspberry Pi:

```
git clone --depth 1
https://github.com/raspberrypi/tools
```

- Edit the file `.bashrc` file in home directory (add the following line at the end of the file):

```
export PATH=$PATH:/path/to/rpi/toolchain/
```

Cross-compilation (2)

- Actualize the *bash* environment variables:
  run `. ~/.bashrc` or log out and log in
- Install wiringPi library:
  `git clone --depth 1 git://git.drogon.net/wiringPi`
- Go to wiringPi directory and cross-compile the library:
  `make CC=arm-linux-gnueabihf-gcc`
- Copy the build shared library to the local lib folder
- Finally, cross-compile a simple *blinking* application (use dynamic linking approach)
- Move the binary to the target and check if it works (add some printing info so you can verify its functionality)
  `scp blinking pi@192.168.23.x:/home/pi`
  `ssh pi@192.168.23.x`
  `./blinking` ← run on the target

Using GDB (1)

- Cross-compile with *debugging* information (i.e., adding -g3 option)
- Strip off debugging information on the target:
```
arm-linux-gnueabihf-strip -s -o add4target ./add
```
- Install GDB server on the target (if not already):
```
sudo apt-get install gdbserver
```
- Run the GDB server:
```
gdbserver :8000 ./add4target
```
- On the host, run cross-debugger and load program with debugging information:
```
arm-linux-gnueabihf-gdb ./add
(gdb) target remote 192.168.23.x:8000
```

Using GDB (2)

- Running GDB server without specifying a process to load:

```
gdbserver --multi :8000
```

- Run cross-debugger without any file and connect to target using *extended-remote* option:

```
(gdb) target extended-remote 192.168.23.x:8000
(gdb) set remote exec-file test4target
(gdb) file ./test
```

- To load a new process to debug, just use:

```
(gdb) set remote exec-file test4target2
(gdb) file ./test2
```

- To quit remote gdbserver, in cross-debugger environment run:

```
(gdb) monitor exit
```

Advanced Linux

Module 3

Cross-compilation and remote debugging

Practical Demonstration