Advanced Linux

Module 2

Embedded application development

Compiling simple application

- The compiler used for Linux systems is GCC (you can check the version by using `gcc --version`)
  Usually installed but if not, use (in Ubuntu) to install full suite:
  ```
  sudo apt install build-essential
  ```
- Compiling a single-file application developed in C
  ```
  gcc -o test test.c
  ```
- By using `-Wall` option, you can enable additional information
- Compiling multiple source files and linking them into one executable
  `gcc -c test1.c` (compile)
  `gcc -c test2.c` (compile)
  `gcc -o test test1.o test2.o` (link)
- gcc automatically invokes the linker `ld` when needed

Compiling static library

Links statically with the application

- Compiling library source files
  gcc -c test1.c (compile)
  gcc -c test2.c (compile)
  ar rcs libtest.a test1.o test2.o (create static library)
  gcc -o test test.c -L. -ltest (link statically)
- Does not depend on any other object file or library

Compiling dynamic library

In Linux world better known as *shared library* (equivalent to DLL in Windows lingua)

- Dynamically linked in run-time
- Compiling library source files
  gcc -c -fPIC test1.c (compile as position independent code)
  gcc -c -fPIC test2.c (compile as position independent code)
  gcc -shared test1.o test2.o -o libtest.so (create shared library)
  gcc -o test test.c -L. -ltest (link dynamically)
- You must export the directory in which the shared library is placed to the `LD_LIBRARY_PATH` environment variable
- Alternatively, the library can be moved to `/lib/` or `/usr/lib/`
- You can use `ldd` command to check which libraries are needed for an application

Using third-party libraries (1)

- On any Linux system, a standard C library is readily available and offers a large set of different functions to assist you in application development
- Aside the standard C library, there are thousands of other libraries for many fields
- Many of the libraries are available as packages in the distribution, generally found in two variants
  - `libfoo` – the package for the library itself (required to **execute** already compiled applications, not sufficient for building new ones)
  - `libfoo-dev` – the package which contains the headers and other configuration files necessary for **building** new applications

Using third-party libraries (2)

- When functions from a library are used, the proper header files of the library must be included in your source
  - Typically `#include <foo.h>` or `#include <foo/foo.h>`
  - It is expected that the headers are present in `/usr/include/`
  - Otherwise, you must specify where the headers are located during compilation

    ```
    gcc -o test test.c -I./libtest -L./libtest -ltest
    ```
- The easiest way to compile an application with the library is to use *pkg-config* facility, if it is supported by the library

  ```
  gcc -o test test.c $(pkg-config --cflags --libs)
  ```
- By default, the application is dynamically linked with the libraries

Advanced Linux

Module 2
Embedded application development
Practical Demonstration