Linux Simple I/O Library Alire Crate for GNAT Ada

Revision 1
5 August 2021

by Philip Munts
President, Munts Technologies
http://tech.munts.com

Linux Simple I/O Library

The **Linux Simple I/O Library** (aka **libsimpleio**) is an attempt to encapsulate (as much as possible) the ugliness of Linux I/O device access. It provides services for the following types of I/O devices:

- **Industrial I/O Subsystem** A/D (Analog to Digital) Converter Devices
- **Industrial I/O Subsystem** D/A (Digital to Analog) Converter Devices
- GPIO (General Purpose Input/Output) Pins
- Raw HID (Human Interface Device) Devices
- I^2C (Inter-Integrated Circuit) Bus Devices
- PWM (Pulse Width Modulated) Output Devices
- **Remote I/O Protocol** Devices
- Serial Ports
- SPI (Serial Peripheral Interface) Bus Devices
- **Stream Framing Protocol** Devices
- TCP and UDP over IPv4 Network Devices
- Watchdog Timer Devices

libsimpleio exports a small number of C wrapper or shim functions. These shim functions present a more coherent API (Application Programming Interface) than Linux kernel **`ioctl()`** services and the myriad other different Linux device I/O API's. The **libsimpleio** shim functions are designed to be easily called from Ada, C++, C#, Java, Free Pascal and other programming languages.

The **`man`** pages specifying the **libsimpleio** API (Application Programming Interface) are available for viewing at **http://git.munts.com/libsimpleio/doc/libsimpleio.html**.

Ada Binding for the Linux Simple I/O Library

The Ada binding consists of several software component layers.

The bottom software component layer consists of the ***C shim functions*** discussed in the previous section.

The next software component layer consists of ***binding packages*** that declare the C shim functions as Ada procedures. Each of the binding packages corresponds to a single C source file (*e.g.* package **`libadc`** corresponds to **`libadc.c`**). Each of the C shim functions are declared as external Ada procedures using **`pragma Import`**. The Ada procedure names do not necessarily match the C function names (*e.g.* the C function **`ADC_Open()`** is declared as Ada procedure **`libadc.Open`**). Many of the binding packages also declare constants as well (*e.g.* **`DIRECTION_INPUT`** in **`libgpio.ads`**).

With very few exceptions, you will never need to directly call any of the procedures nor reference any of the constants declared in the **`libxxx`** binding packages.

The next software component layer consists of ***object packages*** that declare OOP (Object Oriented Programming) object types and methods for each of the I/O subsystems. This layer uses Ada interface types, access-to-interface types, and private tagged records extensively.

For example, the package **`GPIO`** defines an interface type **`PinInterface`**, an access to **`PinInterface`** type named **`Pin`**, and primitive operation subprograms **`Get`** and **`Put`**.

The child package **`GPIO.libsimpleio`** declares a private tagged record type **`PinSubclass`** that *implements* **`GPIO.PinInterface`**, subprograms **`Get`** and **`Put`** that are required to implement **`GPIO.PinInterface`**, and a constructor function **`Create`** that returns an **`GPIO.Pin`** access value. *Every package that implements* ***`GPIO.PinInterface`*** *will also declare a constructor function* ***`Create`*** *that returns* ***`GPIO.Pin.`***

This architecture allows code similar to the following fragment:

```
MyPins : array (1 .. 3) of GPIO.pin;

GPIO(1) := GPIO.libsimpleio.Create(...);
GPIO(2) := GPIO.UserLED.Create(...);
GPIO(3) := GPIO.PWM(...);
```

The three GPIO pins can be stored in the same array and manipulated in exactly the same manner even though the hardware implementation for each pin is radically different.

The topmost software component layer consists of ***device packages*** that implement support for particular I/O devices and are built upon the lower layers. Most of the device packages correspond to integrated circuits, such as the **PCA9534 I²C GPIO expander**. A few implement support for boards or modules, such the **Grove Temperature Sensor** module.

About this Crate

In a normal **libsimpleio** installation (from a Debian package or built from a **libsimpleio** source distribution), the C shim functions are compiled into static and dynamic libraries named **`libsimpleio.a`** and **`libsimpleio.so`**, that are installed into the system library directory **`/usr/local/lib/`**. The Ada binding is installed as a precompiled library project at **`/usr/local/share/libsimpleio/ada/lib/`**. A normal **libsimpleio** installation also includes some **`udev`** rules and hotplug helper scripts that set permissions and create symbolic links in **`/dev/`**.

This crate does **not** use the normal libsimpleio shared library **`libsimpleio.so`** nor any other part of the normal libsimpleio installation. Furthermore, this crate does **not** include the **`udev`** rules and hotplug scripts. It also does **not** include some other items contained in a normal libsimpleio installation, such as some experimental Ada bindings for MySQL, ncurses, and ZeroMQ, and a few packages that depend on the AdaCore Ada Web Server library.

This crate includes a project for building a test program **`test_hello`** that uses the **`Logging.libsimpleio`** package to print a message to the console and pass the same message to the **`syslog`** facility. You can view many more example programs at: **http://git.munts.com/libsimpleio/ada/programs**.

This crate includes all of the functionality of the **`mcp2221`** and **`remoteio`** crates. Unlike those two crates, which can be built for either Linux or Windows 64-bit targets, this crate can **only** be built for Linux targets.

Cross-Compiling

You can cross-compile this crate or a program using this crate for a Linux target computer by just copying the target configuration project file (**`.cgpr`**) from the cross-toolchain to **`default.cgpr`** in the Alire project directory. For example, to cross-compile for a Raspberry Pi 1 microcomputer running ***MuntsOS Embedded Linux***, using the MuntsOS cross-toolchain for the Raspberry Pi 1, just copy **`RaspberryPi1.cgpr`** to **`default.cgpr`**.

Web Links

Linux Simple I/O Library:
https://github.com/pmunts/libsimpleio

Linux Simple I/O Library API specification:
http://git.munts.com/libsimpleio/doc/libsimpleio.html

Linux Simple I/O LIbrary Ada example programs:
http://git.munts.com/libsimpleio/ada/programs

MuntsOS Embedded Linux:
https://github.com/pmunts/muntsos