Documentation for Virtual bus for Linux Device Drives

Introduction:

This code defines a Linux kernel module that implements a custom bus along with associated devices and drivers. The module interacts with the Linux device model to manage devices and drivers that adhere to the bus type. This defines the implementation of virtual bus in linux device drivers. A virtual bus, in the context of operating systems and device drivers, refers to a software abstraction layer that emulates the behavior of a physical bus within the operating system environment. It provides device driver development, system flexibility enhancement and providing a standard interface for communication between hardware devices and device drivers and also in an OS environment.

Purpose:

The purpose of this code is to create a custom virtual bus in the Linux kernel environment. This bus will allow devices and drivers that confirm to this custom bus specification to be registered and managed within the kernel. It also handles the udev events, matching devices to drivers, and exporting a version attribute. It demonstrates how to register a custom bus type, devices associated with that bus, and drivers for those devices. The purpose of the virtual bus in LDD is to provide a standard, abstract interface that facilitates driver development, promotes modularity and portability, and also simplify the management and interaction with hardware devices.

System Requirements:

- Linux kernel headers, appropriate Linux kernel version supporting bus registration and related APIs
- The Linux kernel version should support the features used in the module.

Input:

In this code, user input does not directly take as it operates within the linux kernel environment. So, the inputs are indirectly provided through the kernel's device and driver management APIs. Input to the module typically involves interaction with the kernel subsystems and potentially udev events which is triggered by the device management.

Output:

It provides some outputs such as,

- Kernel log messages for debugging and informational purposes.
- The bus which is registered.
- Devices and drivers can be registered and unregistered dynamically.
- The version of the module is exposed through attributes.
- Error messages. If there are failures during bus registration, attribute creation, or device registration.
- Debug message which indicating the release of the bus.