# LDD Bus Example: A Simple Bus Driver in the Linux Kernel

## Introduction

The lddbus.c file is part of the Linux Device Drivers, Third Edition (LDD3) repository by Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman. This file provides an example of how to implement a simple bus driver in the Linux kernel.

## Purpose

The main purpose of this code is to demonstrate how to create a custom bus type within the Linux kernel. A bus in the Linux kernel is a way to connect drivers with their corresponding devices, facilitating communication between them. This example showcases the essential components and steps required to set up such a bus, including bus registration, device registration, and driver registration.

## System Requirements

To compile and run this code, the following system requirements are necessary:

* A Linux operating system with kernel source code available.
* Basic knowledge of how to compile and insert kernel modules.
* Root access to insert and remove kernel modules.
* A development environment set up with necessary tools like make and gcc.

## Inputs

The inputs for this code include:

* Kernel source and headers: Necessary for compiling the module.
* Bus, device, and driver structures: Defined within the code to simulate the bus and its components.

## Outputs

The outputs of this code are:

* Bus registration: The custom bus is registered within the kernel.
* Device registration: A device is registered on the custom bus.
* Driver registration: A driver is registered that can handle devices on the custom bus.
* Kernel messages: Various messages printed to the kernel log to show the status and actions performed by the module.

## Detailed Explanation of the Code

Here’s a high-level overview of the important sections of the code:

### Header Inclusions

The code includes necessary header files for kernel module development.

### Bus Type Definition

A custom bus type is defined with the name "ldd".

### Device and Driver Structures

Device and driver structures are defined to simulate the bus and its components.

### Module Initialization and Exit Functions

* The module initialization function registers the bus, device, and driver.
* The exit function unregisters them to ensure proper cleanup.

### Module Information

Module information is provided, including license, author, and description.