## Introduction

The program being written I have named Linuxconf. It builds on the Ubuntu restricted devices program, and will be made available for the latest LTS Ubuntu release, 18.04 running the 4.15 Kernel.

This document explains the architecture the proposed system from a high level of abstraction. It is intended for people with an understanding of the process of compiling and inserting linux kernel modules using bash.

*The users who contribute configurations for the client users to download are referred to as contributors, the users who use these configurations to configure their systems are referred to as users,*

## Current System

This project builds on the current “additional drivers” program in Ubuntu which allows proprietary firmware to be installed using a graphical interface. It extends this by allowing Internet users with knowledge of the module build process to contribute git projects to install firmware on a per device basis.

### Functional Description

This project enables a user of Linux with limited technical knowledge to easily install Linux kernel modules using a graphical interface. It relies on the Linux community to suggest configurations for each device being installed based on the identifier of that device.

As an incentive for contributions the contributors will be able to link to their homepage / profile etc possibly with a location. When the users have successfully installed a kernel module the nearest contributor will have a link to their web page or git profile displayed with some basic information about them, including a link to a picture as well as a list of their skills. The user chosen will be based on the most successful contributions to the system.

In it’s initial configuration three types of customisation were planned, configuration of software available from the repositories using apt-get and yum, kernel parameter configuration (for the make config process), and firmware downloads, possibly using all methods for one device configuration.

I realized having installed a wireless ac adapter in Linux that the scope of devices configurable by this project can be massively increased by by allowing users to submit git projects for this system. This process doesn’t require recompiling the kernel, which takes time and may lead to an unstable system. When contemplating how to discuss issues with a particular configuration I realized that git hosts everything this system could need. Firmware files, Makefiles, and most importantly it can host a special file for the entire build process written in bash. The special build file will contain details of how to clone the target git repository, build the firmware using make and insert the module, returning a success or failure exit code once this is done.

*The special build file lives on the GitHub servers and instructs the client application on how to build the kernel module then insert it as part of the client process. It can be seen as similar to a Makefile.*

*After the module has been installed the user will be prompted on the success or failure of the kernel configuration on a per device basis, using a unique code via a website. A program to demonstrate the success or failure of a configuration can be launched, as part of the post install process. This could involve playing a sound to ensure the audio device works, or opening a webcam program like cheese to test the video camera.*

### User Community Description

Two types of user will take advantage of this system. The users will benefit from not have to use the shell to install kernel modules, which many people find obscure and intimidating. The contributors will suggest configuration processes referred to as recipes, with the incentive mentioned earlier.

### Technical Architecture

In order to use GitHub the system will look for a certain file, penguin.sh (I’m being obscure to keep the concept of this project quiet) as part of the project. The client system aspect will run this shell script to build and insert the module into the kernel, reporting success or failure to the central database.

Potentially any site could be linked to in the database. To quote

*https://www.infoworld.com/article/3184399/security/malware-finds-unwitting-ally-in-github.html*

*“****We do not allow anyone to use our platform for exploit delivery, such as hosting malicious executables, or as attack infrastructure, for example by organizing denial of service attacks or managing command and control servers.***

***My intention is to only use git commits older than a week to ensure that malicious Makefiles are not being used. Also building kernel modules in a chroot will protect the client user form malicious Makefiles, but not malicious kernel modules.***

The system relies on a MySQL database, probably maria DB, accessed using Java EE Servlets using get and post requests of JSON type parameters, running on tomcat8. Bash will also be used extensively to clone the git repository and insert the kernel modules, forked from the client program. Rather than writing communications protocols for communication between the C++ front end and the Java back end I propose using HTTP requests.

Major components:

* Backend Database.
* Java back end.
* C++ front end, available as a Debian package..
* Dynamic webpage for communicating success.
* Static webpage explaining the project for file download.
* Linux server to host the system

Data collected:

Device information.

User login information (using oath for git).

Information on the success or failure of a particular configuration.

Two tier architecture.

The Linux client will be written in C++, using gtk and qt libraries for displaying in gnome and KDE respectively. Feedback on the success or failure of a device can be done in a webpage. Hosting can be provided from the Kent University servers, with the option to migrate this to the Linux Foundation servers if the project takes off. The code will be open source, which may be a legal requirement if GPL v3 licensing is used.

## Goals, Objectives, and Rationale for New or Significantly Modified System

### Project Purpose

Create a brand new system based on the current system, Ubuntu additional drivers.

### System Goals and Objectives

Make it easier for novice Ubuntu users to install device drivers,

### Proposed System

A graphical interfaced program which identifies devices on a Ubuntu system and downloads the software from GitHub to make the devices function.

#### System Scope

The system will only work for Ubuntu Linux 18.04 (LTS) running on the 14.15 kernel. Further distribution support can be added when this project is released to the Linux community.

#### Business Processes Supported

The system will be of most use to home users with little experience of configuring Linux, that said it is likely even capable system administrators will use it as it alleviates the need to find the correct firmware projects on git and perform a manual install.

#### High-Level Functional Requirements

Allow user to contribute git project to system.

Clone git repository from GitHub.  
Execute shell script.  
Insert kernel module.  
Provide feedback for user.  
Provide link to a contributing users website.

#### Summary of Changes

The system will perform the same functions as the Ubuntu additional drivers package, but allow this to apply to many more devices by allowing skilled internet users to contribute software without the oversight of the Ubuntu Kernel team.

## Factors Influencing Technical Design

### Relevant Standards

The use of GET and POST HTTP standards, the JSON standard for communication.

### Assumptions and Dependencies

Build-essential packages will need to be installed on the client machine. It is assumed the user has the knowledge to use the internet to download the package and run it from within a browser window.

### Constraints

The client system is running Ubuntu 18.04 running kernel 4.15. Further distribution support is beyond the scope of this project.

The end user must have an internet connection, to download the package and during the programs execution.

The success of an implementation depends on a suitable configuration being produced for that device.

### Design Goals

Incentive to contribute is crucial.

User interface must be easy to use.

Vetting of configurations to ensure malicious code is not used is crucial, considering chroot.

System must be extendable for future use, for future LTS kernels and different Linux distributions.

Ease of use to contribute at GitHub (add bash install script).

Rating configurations must be tamper proof.

## Proposed System

### High-Level Operational Requirements and Characteristics

The user downloads the client program as a Debian package from the host server. The user then installs the client package and is directed to launch the client program on the user’s machine.

The client program identifies the device by it’s device ID. It contacts the server and if a configuration has been submitted for that device the server returns a link to the GitHub project for the client to download and then install via a special run script (penguin.sh).

*Using Oauth contributors can be verified as users of git without revelling their login credentials. I’m suggesting using git hub as it covers many features I would have to build for this project. These include hosting code and files, as well as a comments thread on each repository and pull requests to suggest changes to current implementations.*

The client program builds the kernel modules. It then attempts to insert the newly built kernel module into the kernel, returning a success or failure result. In the case where an install has failed the client program tries again with a different configuration.

*Potentially the client could upload the error logs to the database from where I can use Oauth to post the error messages as a comment on the git project page under my user name.*

In situations where the computer needs to be restarted the client runs again as an at daemon process.

*https://en.wikipedia.org/wiki/At\_(Unix)*

After the client kernel module has been inserted a user review page appears. The user is welcome to try their new hardware to see if it works (we could fork a process for webcam, audio test etc as part of this?). Once the user has declared a device works it is up-voted in the server database. In situations where the device does not work the device Is down-voted in the server database. In cases where an internet connection is not available the user will be invited to connect via phone tethering or Ethernet.

Contributing configurations Is done via a website, which communicates with GitHub using Oauth to verify the git contributor to the project owns the git repository and project being referenced by the system.

Configurations are triaged by other contributing users to ensure they are not malicious. If a configuration is marked as malicious constantly I will personally review it. Keeping the voting process to other contributors implies a level of proficiency in the review process. (As things are being built in a chroot and the module is inserted into the kernel is this really necessary?).