

Abstract

Linux is an operating system in which the configuration of services is contained within plain text files, and within which said services are configured by editing the configuration text files manually. The aim of this project is to research, design and implement a tool that provides a means for the manipulation of the files used for the configuration of the Domain Name Service (DNS) and Dynamic Host Configuration Protocol (DHCP) network services to be carried out graphically. The project will look at the various approaches and enabling tools suitable for this purpose, and develop the most suitable.

Introduction

The origins of Linux lie with the development of the UNIX operating system. UNIX was developed in 1969 by researchers based at AT&T Bell Laboratories to provide an operating system tailored to the needs of a research environment; an efficient and affordable operating system designed to run on minicomputers with multitasking and multi-user capabilities (Petersen 2003).

Over the years, UNIX evolved to a portable, powerful and flexible operating system, used within research laboratories and enterprise computer systems around the world. It had been designed at a time when mainframes and minicomputers were the only computers available, and as these computers evolved in terms of power, capability and features, UNIX evolved alongside to provide the operating system support that was required. In a nutshell, UNIX is an operating system designed to run on big, powerful computers in order for the computers to carry out big, powerful tasks (Rosen et al. 2001). Technologies such as networking and the World Wide Web were originally developed with UNIX in mind, and for these reasons UNIX remains the most popular choice of operating system for network servers and mainframe systems (Petersen 2003).

The Linux operating system was originally conceived by Linus Torvalds, a research student based at the University of Helsinki in Finland. His intention was to develop a version of the UNIX operating system that could effectively run on a PC workstation. The first released version of the software, version 0.11, was widely distributed over the Internet, and in the following years Linux was added to and refined by programmers from around the world. In time, all of the main features and applications of the UNIX operating system were implemented into Linux (Petersen 2003). Due to its 'open-source' nature, the source code for the Linux kernel is freely available for users and programmers alike.

Due to the origins of UNIX as a server-oriented operating system, it has followed that as Linux has evolved and been developed further, it is increasingly also being used as a server operating system, particularly for smaller networks. However, in recent years as Microsoft has moved into the server market, Linux is increasingly losing out to Windows in terms of user-friendliness, in part due to the non-graphical way in which many of the services are set up and configured.

Two of the most important functions of a server are the DNS (Domain Name Service) and DHCP (Dynamic Host Configuration Protocol) services. DNS is a technique used to resolve hostnames of computers on a network to their IP (Internet Protocol) address. IP addresses are unique

numerical addresses that are used by the most widely used computer networking environments (including the Internet), and in these environments all communication between computers uses the IP addressing scheme. However, this is not really convenient, so DNS exists as a service on a network to store a list of hosts on a network with their IP address. When a host on a network tries to invoke communication with another host by specifying its hostname, the DNS server on the network is sent a query to resolve the IP address of the intended recipient from its specified hostname. This way, the IP address is found from the name of the computer.

DHCP is a technique that is used on a network to ensure that each host on the network has its own unique IP address. Essentially, rather than configuring on each computer its address on the network (known as static addressing), each host on the network sends a request to the designated DHCP server for an address. The DHCP server controls a 'pool' of available addresses according to configured parameters, and distributes addresses to hosts on the network as required.

In Linux, the configuration of options and services is contained within plain text files. This is unlike Windows, where the configuration files are typically encoded and not designed for direct editing. In Linux, the primary method of configuring options is to manually edit the configuration files, inherited from UNIX. This is partly due to the development of UNIX before the popularity of graphical interfaces; however another reason is the potentially complex nature of these services. Without constraints of aesthetic requirements, and the overhead of graphical libraries, the services can provide more powerful options and settings (Rosen et al. 2001).

More recently, distributions of Linux have moved increasingly towards graphical interfaces as a front end used to edit the text file. However, this is less so for the more complex server functions, such as DNS and DHCP.

Objectives

The main objective of this project is to develop a Graphical User Interface (GUI) tool to be used on a Linux network server that will:

1. Check for DNS and DHCP configuration files
2. Create the necessary files if they do not yet exist
3. Read the current DNS and DHCP configurations from the configuration files, whether previously configured manually or with the tool.
4. Enable changes to the configuration of the services to be made
5. Allow additional options/parameters to be included within the configuration.
6. Save the configuration to the server
7. Allow control over the starting and stopping of the services
8. Allow successful operation of the services on a computer network without the need for shell commands and manual file editing

This objective will be carried out using research to ascertain the best method for enabling the GUI, and the best tool for its development. The GUI will then be designed, implemented and tested.

Conclusions

The project was successful, and the stated project objectives have been met, as demonstrated in the test results and detailed below. However, the conduct of the project did not follow the original time plan. The unexpected failure of the KDevelop software and the C++ libraries caused delays in the implementation of the program, and time was lost in the early stages practising with these. The choice of contingency turned out to be a good one, however, and the Kylix for Delphi software proved to be perfectly adequate for the project.

In addition to the delays experienced due to the problems described above, the project was slow to start in the early feasibility stages due to the author's conduct. Meetings with the project supervisor were not scheduled in time, and the problems with KDevelop and C++ were identified later into the project duration than would have been ideal. It was the author's original intention to carry out much of the Feasibility research during the summer period before Year 3; this did not happen however due to constraints of work/time.

Due to these reasons, the design and implementation of the project was carried out over fewer weeks, so the hours worked each week increased accordingly.

The design and implementation remained consistent with the original plan, and the step-by-step method shown in the design and implementation stages ensured the project was completed on time and was relatively problem-free. The only main problem was the unfamiliarity of the Delphi language, causing slow progress early on in the Implementation and use of functions within DNS that were dropped in favour of more suitable functions for DHCP. This does not affect the end user however, and the coding would only be changed were the program to be developed further.

The program itself provides a useful tool for configuring DNS and DHCP for a computer network. It provides graphical support for the main functions used with each, while still allowing the user to configure additional options manually within the text file. The stated objectives are met:

- 1. Check for DNS and DHCP configuration files**
- 2. Create the necessary files if they do not yet exist**

Both the DNS and DHCP parts of the program test for, and if necessary create, the configuration files used for both services. For DNS, a default option for the zone directory is also written to the file.

- 3. Read the current DNS and DHCP configurations from the configuration files, whether previously configured manually or with the tool.**

The program is able to read the files regardless of the format used. No assumptions are made as to the contents of each line, and each line is checked against supported functions. The program is able to differentiate between tab and space characters used to manually space information in the files, and other subtleties in format a user may use.

- 4. Enable changes to the configuration of the services to be made.**
- 5. Allow additional options/parameters to be included within the configuration.**
- 6. Save the configuration to the server.**

The program allows the user to change values read from the file, by presenting the values and writing changes made by the user to the file. Options and parameters not supported by the program are ignored and read around, allowing the user to manually add these options to the text files. The changes made are saved to the files.

- 7. Allow control over the starting and stopping of the services.**
- 8. Allow successful operation of the services on a computer network without the need for shell commands and manual file editing**

The program allows the user to control the daemons using the interface. Using this program, the DNS and DHCP services successfully operate on a network, as shown in the testing, without the need to configure the services or their execution outside of the program.

As the program was developed using Kylix rather than KDevelop, portability is more difficult. KDevelop was originally selected for several reasons, and among these are its use of standard KDE libraries and its ability to generate RPM files. Each of these improves the portability of the software developed using KDevelop. As Kylix uses non-standard visual libraries, porting software requires porting the libraries as well as the compiled code. In addition, as the Delphi language is only used in Kylix, distributing the source code requires Kylix to compile the program. This is a limitation not found with C++ code, as compilers and KDE libraries are included as standard with most Linux distributions.

The failure to install Kylix successfully as root meant that write permissions for all were set for the configuration files. In a real network, this would not be acceptable.