

A STUDY OF LINUX AND ITS VIABILITY IN MAINSTREAM COMPUTING

99E001I

An Interactive Qualifying Project Report

submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

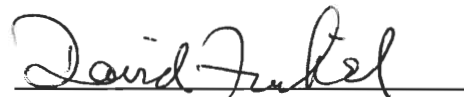
Jeffrey Cilley

Dennis Mattoon

Aaron Chandler-Worth

Date: June 7, 1999

Approved:



Professor David Finkel, Major Advisor

- 1. Linux
- 2. Viability
- 3. Open Source

Abstract

Linux's recent surge in popularity has caused many professionals to wonder if it is to become a serious competitor in the business operating system market. This project combines research with in-depth qualitative interviews of industry leaders to find the answer; today, Linux is not a suitable business solution but with its current rate of growth and refinement it will be. It has also revealed that Linux's potential success could change the fundamental software engineering paradigm that is in practice today.

TABLE OF CONTENTS

	Page
1. INTRODUCTION	1
2. BACKGROUND	2
3. INTERVIEWS	12
4. PREDICTIONS AND CONCLUSIONS	22
APPENDICES	
A1 The BSD License	23
A2 The GNU General Public License	24
A3 The X Consortium License	29
A4 The Artistic License	30
A5 The Official Open Source Definition	33
A6 The Initial Interview	35
A7 Mr. Brown	36
A8 Mr. Green	37
A9 Mr. Blue	38
A10 Mr. Orange, Mr. Pink, Mr. Purple	39
A11 Mr. White	40
BIBLIOGRAPHY	41

1. Introduction

If you have not heard about Linux yet, you will soon. Between 1997 and 1998 Linux experienced a 212% growth in unit sales, making it the fastest growing operating system in the market. Linux was featured in such prominent publications as *The New York Times* and *Forbes Magazine* and is known by most all students of computer science. With all of this attention, one would think that Linux would be in every home and office in America, but it is not. We posed the question, “Why?” We sought to discover what, specifically, society wants in an operating system and what it would take to integrate Linux into that society. To this end, we chose to interview captains of industry to get their opinions on what they feel Linux will need to become viable in mainstream computing.

A trickle-down axiom of computing is what led us to industry professionals. It is believed that the integration of software into society begins in academe, progresses to the business environment and finally trickles down into the home. It is already an established operating system among computer science students; this project explores its progression into the business arena.

2. Background

Linux is an operating system. To those who do not know, an operating system's purpose, in most cases, is to hide the technical details of a computer from the user. This abstraction is what makes computers useable by the average Joe. One of the things that sets Linux apart from other operating systems, and what most believe is its biggest strength, is the fact that it is Open Source. In other words, the source code that makes up the operating system is free to anyone who wants it. In recent months, Linux's popularity has increased by leaps and bounds due to factors beyond its Open Source nature. The goal of this project was to predict whether or not Linux's current popularity would pay off and turn the operating system into a viable alternative in mainstream computing. This goal was achieved through the use of interviews with industry leaders and the analysis of the opinions and facts they conveyed. The following background section will lead to the manner with which we conducted our interviews.

The philosophy behind free software has been around since the early days of computing, long before the official definition of Open Source software even existed. When the first version of the Unix operating system was developed in 1969, it was freely distributed with its complete source code. It was not until 1979 that AT&T first announced its plans to develop Unix commercially. [6] By this time, the proprietary software movement was overtaking the traditional software development method of the time and rapidly growing into the enormous industry that it is today. The Open Source movement might have been squashed before it had begun if not for a young researcher at MIT's AI lab named Richard Stallman. [2]

In 1984, Richard Stallman started the Free Software Foundation (FSF) and established the GNU project in the hopes of promoting free software. "The GNU project's goal was, simply put, to make it so that no one would ever have to pay for software." [1] He believed that the

source code for every program should be free. At the same time, however, he understood that businesses would attempt to exploit the work of others if source code was simply released to the public. In order to counteract this threat, GNU established the GNU General Public License (GPL). Until then no formal guidelines governing free software were as significant or influential. [2]

Anyone who wishes to use software released under the GPL must follow certain guidelines. The user has the ability to copy and distribute software released under the GPL as long as it is kept in its original form. The user can modify any software and distribute their version provided they include “prominent notices” of their alterations in each modified file, and they release it under the GPL free of charge to anyone who wishes to use it. The GPL prevents businesses from exploiting free software by requiring all source released under it and any works derived from that source to be freely available. Programmers now had the tool with which they could share their knowledge without losing credit for it. [2]

The creation of the GPL played a pivotal role in the evolution of free software. First, it guaranteed that every program released under it would be available free of charge with its source code. However, we must not think of free software only as free code. The GPL also gives programmers a guarantee that no one can claim ownership of their work. This gives all programmers the freedom to share knowledge with others. It is important to think of both meanings of the term free when talking about free software. As Stallman describes, we must “Think free as in free speech, not free beer”[4].

This notion of free software did not appeal to everyone; some viewed the Free Software Foundation’s message as anti-business. Edward J Zander from Sun Microsystems, for example, is one who is openly critical of the free software movement. “Why should I give away what I

pay millions of dollars to develop? Why doesn't General Motors give its cars away for free?"[4] This view is very understandable, as businesses rely on having knowledge that their competitors don't, it is what gives them their edge. There is an inherent flaw in this analogy because it assumes that the materials required to write software cost money just like the materials required to build cars. However, unlike cars, where new materials are required for every product, multiple pieces of software can be written on the same machine. After a program is developed, infinite copies of the software can be produced for a very minimal amount of money. Basically, when you purchase a piece of software, you are paying only for the knowledge that it requires to develop it. Stallman and others like him have always taken a more scientific view of software. The idea is that the knowledge should be shared because if the technology advances more quickly, everyone benefits. However, the opinions of most of the corporate world are very similar if not identical to Edward J. Zander's, and until fairly recently no effort was made to polish free software's anti-business image.

In 1997, several prominent free software advocates met in California to discuss this problem.[2] The group, made up of Tim O'Reilly, VA Research president, Larry Augustin, Eric Raymond, and others, recognized that the Free Software Foundation's anti-business message may scare away some people, or at least prevent them from fully appreciating the benefits of the free software model. Through their discussions on how to remedy the situation, the term Open Source and the Open Source Definition were born.[2]

The definition is a set of guidelines that a piece of software must meet in order to be called Open Source. First and probably most important, the software must be released under an appropriate license such as the GPL (see appendix A2), the BSD (see appendix A1), the Artistic (see appendix A4), or the X Consortium license (see appendix A3). The BSD, the Artistic, and

the X Consortium licenses are very similar to the GPL; their official definitions are located in the appendices of this document. The program must be distributed with its source code included or must include documentation that gives a location where it can be downloaded free of charge via the Internet. The source code must not be obfuscated and intermediate forms such as output from a preprocessor or translator are not allowed. This simply means that the source code must not be blatantly released in a confusing form. The Open Source Definition originated from “The Debian Free Software Guidelines” which was drafted by Bruce Perens. [2] The Debian-specific references were removed and the Open Source Definition was born. The term “Open Source” was established largely on Eric Raymond’s insistence that a new marketing campaign be created in order to improve Open Source Software’s image in the business world.

Eric Raymond is a long time hacker and supporter of Open Source. In 1997, he wrote a paper entitled “The Cathedral and the Bazaar”[6] that was the first to truly examine the Bazaar Style of Open Source development as a practical programming model rather than just a theory. It helped transform Open Source from an idea that all software should be free, to an effective model for software development. His work was so influential that it convinced Netscape to release the source code for their web browser.[5] The paper touches on many issues such as differences in the Cathedral and Bazaar development models.

In the Cathedral model, one development team does all of the coding, testing, and debugging of the software - in other words, they do all the work. Because there are a limited number of developers and much of their time is spent debugging, this model tends to produce releases that are infrequent at best. By taking so long to release updates, programmers actually hinder their own progress by spending too much time debugging. If they were to “Release early [and] Release often”, [5] they could allow their users to identify bugs for them while they spend

their time in a more constructive fashion such as adding functionality. Allowing users to identify bugs will, almost inevitably, lead to users suggesting fixes for those bugs. As the number of users contributing to bug fixes grows, so does the project as a whole. If this process continues, you will have a community of developers all contributing to the advancement of the project, instead of a specific group of people trying to handle the job themselves. This sense of community is where the Bazaar Style development model draws its strength. [5]

With the relatively large testing and co-developer base the Bazaar method boasts, debugging is much quicker. The larger and more diverse the developer/tester base is, the greater the chances are that complicated bugs will be resolved quickly and efficiently. This quick bug-to-resolution turnaround time is a result of variation amongst users; each user looks at a bug from a slightly different point of view. Because of the varying skill sets among users, chances are that even the deepest bugs will appear shallow to someone. [5]

Faster debugging time is one example of how the Bazaar style's divide and conquer method excels over the Cathedral style. This is probably the most important Open Source pertinent theme that Raymond presents--the fact that "the closed-source world cannot win an evolutionary arms race with open-source communities that can put orders of magnitude more skilled time into a problem." [5] There is probably no one that understands this better than Linus Torvalds, the creator of Linux.

On August 25, 1991 a young graduate student from the University of Helsinki sent the following message to the comp.os.minix newsgroup. [1]

```
From: torvalds@klaava.Helsinki.FI(Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: 1991Aug25.205708.9541@klaava.Helsinki.FI
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki
```

```
Hello everybody out there using minix -
```

I'm doing a (free) operating system (just a hobby, won't be big and professional like GNU) for 386(486) AT clones. This has been brewing since April, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things). I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc.), and it probably never will support anything other than AT-harddisks, as that's all I have :-).

Linux was created because Linus Torvalds was unsatisfied with the Minix OS and decided to write something that fit his needs. This is typical of the beginnings of most Open Source projects. Projects are usually started because the programmer either has a need for the software or because it is interesting to him or her. Commercial software projects on the other hand are motivated primarily by money. This is another one of the fundamental differences between the Open Source and commercial models for software development.

The driving force behind most commercial projects is largely financial. This means they have to produce software that makes a profit. For a piece of software to be successful commercially, trade-off's have to be made. One common trade-off is sacrificing reliability for a shorter time-to-market. When companies are in a hurry to release their product they will sometimes either not test as thoroughly as they could or even cut development short to make a particular release date. Because profits and deadlines govern most commercial projects, quality takes a backseat. Open Source projects on the other hand are not motivated by money and usually do not have to make any deadlines. By not being profit driven, Open Source projects do not have the same quality issues as commercial projects do. Software can be tested as vigorously as the developers feel necessary with no time constraints, resulting in more reliable software. A related issue here is the difference in motivations between Cathedral style commercial developers and those of the Open Source Bazaar style. [5]

Commercial developers work on projects determined by the companies. It is not uncommon that these developers have no interest in the projects they are working on. They contribute to a project because they are paid to do so and not because they are interested in its development and want to see it advance. This attitude often produces sub-par software, lacking in both quality and functionality. This is not an issue in the Bazaar Open Source model of software development. Developers contribute to or begin a project because they have a genuine interest in it and money is usually not a factor. If a developer does lose interest in a project, there are always several others that are eager to take his place. An issue unique to Open Source development that comes into play here is that of reputation. An Open Source programmer's reputation is directly related to the success of their software. This means that a developer will put more of their time and effort into a project if it means that he or she will become better known and respected among his or her peers. A hugely successful and well-known project like Linux will, and has, produce an equally well-known developer like Linus Torvalds. [5]

The success of Linux thus far is proof of the viability of the Open Source model. However, this success begs the question, if the Open Source style is so good, why was it quickly overtaken by the commercial methods of software development and why is it now becoming such a viable competitor? The answer is, simply stated, organization and communication. As Raymond writes, "I don't think it's a coincidence that the gestation period of Linux coincided with the birth of the World Wide Web, and that Linux left its infancy during the same period in 1993-1994 that saw the takeoff of the ISP industry and the explosion of mainstream interests in the Internet." [5] It would seem that the Bazaar Style of software development just needed a church parking lot in which to base itself--the Internet is that church parking lot.

It is still not clear whether or not Linux will succeed in the business market. It has seen a recent surge in popularity on the university level in recent years. The fact that the source code is available makes it a great learning tool for students. However, the university environment is a far cry from the business environment. This leads us into the next phase of our project. In order to make predictions of where Linux was headed, we needed to know how the business world looks at it. It is important to note that our objective was not merely to do a technical comparison of Linux and other available products. We really wanted to know how the business world perceives Linux. We decided that the best way to accomplish this was to go out and talk to people who work in a field in which they might have been exposed to Linux in order to discover what their opinions are.

To discover how people in the field felt about Linux, we needed to conduct interviews. Before doing so however, we had to determine how they would be carried out and there were a few different interviewing styles to choose from. The first type is the standardized interview. This technique involves creating a very strict set of questions before the first interview and conducting each interview without straying from the original script. The questions should be asked in the same order and most should be very simple. Simple questions should produce brief responses that can be easily compared when the data is analyzed. A Standardized interview gives the interviewer complete control over the proceedings and is very good for testing hypotheses. However, due to their rigid nature, it is unlikely that the researcher will acquire a great deal of new information from his interviewees. One of the advantages of standardized interviews is that they are simple enough to be conducted over the phone or through the mail which makes them very convenient and also not time consuming. The results can also be analyzed very easily because the responses are very simple. Because of these factors, the

researcher will usually be able to interview large quantities of people – this allows the findings to be applied to large populations. Another advantage of standardize interviews is that the interviewer does not have to worry about influencing the responses of the interviewee because they are all written before hand. [3]

Another technique that can be used is the in-depth qualitative method of interviewing. In-depth qualitative interviews require much less preparation time than the Standardized interviews because the researcher does not need to prepare a rigid script to follow. Compared to standardized interviews, in-depth qualitative interviews are loosely structured and seem more like a natural conversation. This gives the interviewer a great deal more flexibility than he or she would have with the standardized interview. This loosely organized style, because of its exploratory nature, allows the interviewee to give responses that are rich in detail and sometimes personal in nature. This will often lead to new lines of questioning that the researcher may not have previously thought about. The purpose of these interviews is not normally to prove a hypothesis, but simply to acquire information from the participants. The main advantage of in-depth qualitative interviews is that they will usually provide a wealth of information previously unknown to the researcher. A major disadvantage is that the researcher cannot interview a large quantity of people because each interview takes a great deal of time. It can also be very difficult to analyze the data because it varies greatly between participants. Because the interview is so interactive, it is very easy for the interviewer to bias the results by influencing the interviewee. If done correctly however, in-depth qualitative interviews have the potential to provide much more information to the researcher. [3]

Focus groups are in-depth qualitative interviews that are done with more than one participant at a time. Instead of a one on one conversation, they usually take the form of a

discussion. The interview focuses on the interaction between the participants. It is therefore important to have a topic that is interesting otherwise or it will not generate very much discussion. It also helps if the topic is relatively new so that the participants have not had time to formulate concrete opinions. The interviewer acts as a moderator and sometimes the discussion can stray from the desired topics. The researcher has to be careful to keep the participants focused on the desired area of discussion. [3]

For the first interview, we prepared a very structured sequence of questions to ask the interviewee. We hoped that we would be able to use the standardized technique because it is easier to analyze the data. We quickly realized that it was not producing information that was very useful. We then made a transition to a qualitative style of interviewing. It seemed more appropriate because we were more interested in the opinions of the people we were interviewing. An outline of our interview can be found in appendix A6.

In the following chapter we will provide an in-depth analysis of the interviews we conducted. It will include a description of the people who we interviewed and an overview of the topics covered. We will conclude with our interpretation of what these views suggest about the future of Linux and Open Source as a whole. Also, notes from each of our interviews can be found in appendices A7-A11 of this text.

3. Interviews

We set out to interview a variety of people who work in the computer industry. Our goal was to ascertain a common, generally accepted view of Linux and its place in the business environment. To this end we sought out a number of business professionals who were willing to engage with us and share their feeling and opinions towards Linux.

During our discussions, a majority of the participants requested to remain anonymous. To accommodate our interviewees and to remain consistent, we will refer to all of our contacts by a simple color. Our first contact, Mr. Brown, is a Unix system Administrator at a distinguished institution. The second, Mr. Blue, is an Information Technology Manager at a reputable University. Mr. Green owns his own business that specializes in the hosting and authoring of web pages and general software development. Mr. Orange, Mr. Pink, and Mr. Purple are all engineers at a major computer company. Mr. White is a Vice President of his department at a large communications company.

During the analysis phase of the data, which we obtained from our interviews, we found that the participants agree and disagree on many of the same points. Because we used the in-depth qualitative style of interviewing, points that were discussed in some interviews were not mentioned in the others. However, there were several key points that were brought up during each session. Due to the fact that we had a very small sample size, we thought that these common views would best represent the feelings among the majority. In the remaining portion of this chapter we will outline and discuss the significance of these integral points.

The issue of cost to the user was the first major point mentioned by each interviewee. Linux, unlike other commercial competitors, is available free of charge to anyone who has access to the Internet. For many corporations, the purchase price is only a minor concern when looking

at the total cost of ownership. The cost of tinkering and downtime account for the largest portion of the overall cost.

The cost of tinkering is a very important consideration when choosing an operating system. Encapsulated within tinkering are installation, configuration, and the maintenance of a new system. The installation of an operating system can range in difficulty anywhere from a user-friendly program that holds your hand through the whole process to an utterly baffling and backbreaking odyssey. In the corporate arena, time is money and odysseys are expensive. Configuration can also be very time consuming and unlike installation, it needs to be repeated every time the system changes or the demands on the system change. Maintenance is an ongoing process regardless of what operating system you choose. It is important to find an operating system that requires as little maintenance as possible once it is installed and configured properly.

The amount of time and money lost to downtime can vary from business to business. For a company that sells their products through the web, a server crash could cause hundreds or thousands of orders to be lost. The downtime cost in this case could be enormous. On the other hand, if a system internal to the company crashes, it might simply cause a few hours of lost productivity that would probably be only a minor inconvenience. If the system is down regularly, the hours of lost productivity and the lost business can be a severe handicap to a company, not to mention a hassle. Every business aims to keep the downtime of their system to a minimum. Since the need to limit downtime may differ between businesses, a company needs to choose an operating system which matches their specific tastes and needs. If downtime is not an issue, there is no justification for purchasing a fault tolerant system that costs an exorbitant amount of money. When downtime is an issue, the opposite would also be true.

It has already been mentioned that Linux has an insignificant startup cost, but it should be clear now that the initial cost is not the only thing to consider when purchasing an OS. From our interviews we found a general consensus on where Linux begins to breakdown in terms of cost. Mr. Brown felt that the initial cost of an operating system was a minor issue because most large businesses and institutions can afford to pay for software. He thought that the hassle of configuring Linux outweighed its benefits. Mr. Green, a small business owner, felt that the purchase price was significant and viewed the ability to configure Linux as an asset, not a hindrance. He also felt that Linux's low maintenance cost was very appealing. Mr. Blue felt that Linux was a big hassle and, as an IT manager, would be willing to pay for ease of installation and configuration. He also brought up the point that when some businesses see that Linux is free, they immediately jump to the conclusion that "you get what you pay for". Mr. Orange, Mr. Pink, and Mr. Purple all agreed that most businesses see Linux as a tinkering OS and that tinkering is viewed as a complicated and expensive process. Mr. White said that it would be nice not to have to pay for an OS but mentioned it was not an important issue. Mr. Brown believed that for most large businesses or institutions, the purchase price for software is not a significant factor.

All of our interviewees agreed that in order to be successful in the corporate market, an OS had to be powerful. By powerful they mean that it will be able to perform all of the functions required of it. To meet the demands of such a general requirement specification, the operating system needs to be versatile. A versatile OS, is one that can support a wide variety of hardware and software. A machine that can only run one or two software packages is not very powerful and certainly not useful in a business market.

Support for hardware has always been a big issue when talking about Linux. It has always been the users who add support for hardware to Linux. When there were very few users,

it supported few pieces of hardware. However, as the user base has grown, the number of supported devices has grown and now, aside from the more exotic peripherals, Linux supports most hardware. In spite of Linux's current compatibility with a majority of the most commonly used hardware, most of our interviewees believe that it is still a problem.

The lack of software that Linux supports is still one of its main handicaps. There is no reason to use an operating system that doesn't support the software packages you need. It is for this reason that many businesses do not consider Linux to be a viable option. Historically, commercial software vendors have not ported their software to Linux because of its small number of users. Recently however, many vendors who have lost in the Microsoft market have begun to see Linux as an opportunity, hence the growing numbers of software packages available for Linux.

Integrated solutions have become the latest craze in business. An integrated solution has been the answer for many corporations facing the problems of hardware and software support. Solutions such as these take the worry out of purchasing an OS. This convenience is very appealing to managers because it cuts their work in half. Most of our interviewees foresee this as being the biggest market for Linux in the future. Currently there are very few vendors providing integrated solutions that include Linux as the OS. Those we interviewed however, believe that with more well known companies such as Dell joining the ranks of Linux based integrated solutions vendors, the more the industry will grow. With this growth in availability and support, Linux will become a more attractive alternative for corporations in search of integrated solutions.

The following is how our interviewees felt about the issues of support and integrated solutions. Through research, Mr. Brown and Mr. Green found that Linux did not support much of the software they required. Mr. Blue felt that integrated solutions were important simply

because they were more convenient, and as an IT manager he looks to minimize hassle wherever possible. Mr. Orange, Mr. Pink, and Mr. Purple all thought that Linux's lack of support for certain software packages was a hindrance. Mr. White agreed.

When it comes to OS choice, businesses will use whatever works, especially when it is easy to use. Until recently a business' choice of operating system was based strongly on its looks. This trend seems to be fading however, with more and more businesses choosing systems based on power and reliability rather than aesthetics and their *time to learn*.

Even though corporations are beginning to embrace Linux, there are still many who see it as a risk not worth taking. Most who feel this way do so because they find, according to a strong majority of our interviewees, that Linux is very difficult to use. To some extent they are right. The Graphical User Interface, or GUI, can be intimidating to some users. For the most part, businesses want an interface that is simple and hides all technical aspects from the user. Right now, the operating systems that have an interface which achieve this are notorious for their failures. According to some of our interviewees, businesses are beginning to realize this.

One of the things Linux needs in order to be widely accepted in business is a better graphical user interface. Currently there are several to choose from, but none achieve the level of abstraction that would make the OS *idiot proof*. The majority of our interviewees feel that once this interface has evolved into something intelligent enough for the average user, standards should be established. They also feel that once these standards are introduced, an even greater number of businesses will adopt Linux.

Standardizing a GUI would also open up a new and lucrative market for many companies. Companies that are used to writing software for operating systems that have a standard interface, such as that of Microsoft Windows 9x, would not find it difficult to port their

product to Linux. Currently, to write an application for a GUI under Linux there are certain requirements that the user needs to meet, but with a widely used standard interface the user is likely to already have any required component. As a result of this, more applications can be written for a much larger user base.

Mr. Green, Mr. Orange, Mr. Purple, and Mr. Pink all believed that a “user friendly GUI” was very important to Linux’s evolution. Mr. Brown did not feel that a GUI was terribly important to Linux’s success. At no point during our meetings with Mr. Blue or Mr. White did the issue of a standard graphical user interface come up.

When conducting the interviews, one of the questions posed was, “In addition to the lack of a standard interface, what prevents businesses from using Linux?” The responses we received were very conclusive. In addition to the aforementioned issue of cost, there are other issues that are of great concern and interest to corporations.

The first of these issues is that of switching. Those businesses that already have systems up and running must evaluate the cost, hassle, and advantage of switching to a new and better system. Often, this is such an arduous task, that many companies do not even bother to switch systems. Instead, they spend their time trying to make outdated software and hardware work for them. This becomes a cycle leading to more time and money spent on tinkering and maintenance. Businesses often overlook this cost because it is spread out over such a long period of time. Although switching is a hassle, not switching can prove to be even more costly in the long run.

The second issue mentioned is the lack of technical support associated with Linux. Businesses know that downtime can be catastrophic. When there is a problem, it needs to be fixed as quickly as possible. Businesses rely on those organizations and people who can solve

their technical problems quickly and efficiently. Until recently, Linux has not had any organized method of technical support available to their users.

At present, there are several companies that offer their own distributions of Linux. While Linux distributors cannot charge for the Linux kernel and other utilities within their distribution covered by free licenses, they can charge for utilities they write themselves or those from other software makers whose software is not covered. For many distributors, the sale of Linux is not their primary source of income. Several Linux distributors provide technical support at a fee. When a company purchases Linux they pay for convenience in packaging and included pieces of commercial software. California based Linux distributor Caldera, for example, offers their Linux distribution as a “Complete solution for small and medium-sized businesses.” Companies cannot make a profit from selling Linux itself, but they can make a profit by selling distributions that include more than just Linux. Notice they are selling “solutions” not just software. The other distributors make their money by offering similar services.

In response to the previous section’s question, “...what prevents businesses from using Linux?” it was Mr. Brown’s feeling that switching from a current operating system to a new one was not worth the hassle involved. On the topic of technical support, Mr. Brown did not feel that it was an important issue because most large institutions have knowledgeable employees that could solve most any problem without the aid of technical support. Conversely, Mr. Blue and Mr. White both felt that technical support was one of the more important issues needs to be addressed before Linux can be successful. Mr. Green felt similarly on the topic of technical support. Each Mr. Orange, Mr. Purple, and Mr. Pink believed that the issue of technical support

carries much weight in companies decisions. Everyone agreed that Linux's software support was lacking but Mr. Green was alone in his opinion that Linux's hardware support was sufficient. "I want someone I can point a finger at" is a common feeling among most businesses according to Mr. Orange. The issue that this statement brings to light is that of liability. When systems fail, there is the potential for monumental financial loss and the majority of corporations want someone who they can blame. Unfortunately, due to its licensing, no one can be held liable for Linux's failure, should it occur. For some businesses, this issue alone eliminates Linux as a possible alternative. It is unlikely that any company will ever accept liability for Linux because it is Open Source. Anyone can modify the source of Linux and could intentionally add malicious code that could cause a serious system failure. The company could be forced to pay for many fraudulent claims. This is and will continue to be a major roadblock for Linux.

Mr. Blue thought that both liability and support were major roadblocks that Linux would have to overcome. During our meeting with Mr. Orange, Mr. Pink, and Mr. Purple, the realization that "you can't fire a volunteer" was raised. Each believed that, because no one can be held accountable for Linux's failures, should they occur, corporations would not feel comfortable switching. Neither Mr. Green nor Mr. Brown believed that liability was an important issue to address.

In spite of a few roadblocks, Linux is gaining in popularity. When asked why this was the case, everyone we spoke to had very similar answers. One of the reasons that Linux is gaining in popularity is a result of the Microsoft backlash.

According to the people we spoke to, many businesses that chose Microsoft for their business solutions are beginning to realize that sacrificing a nice interface is an acceptable price

to pay for reliability; in time it is likely that Linux will have both. This realization was not born out of spite for Microsoft but rather a general feeling of growing dissatisfaction of their product.

Another reason for this growing popularity is increased corporate support. Companies that tried to compete with Microsoft and were defeated see Linux as a second chance for success and are backing Linux. IBM for example, whose OS/2 system was beaten by Microsoft in the operating system market, is now releasing their database products for Linux. IBM is not the only one looking for a second chance. Corel, maker of the WordPerfect suite, was once the undisputed leader in the word processor market. With the release of Microsoft Office, Corel's WordPerfect suite was relegated to storage closets and bargain bins. They now offer a Linux version free for non-commercial use and it has become one of the most widely used word processors for Linux. Other companies that have a strong user base within the Microsoft community such as Oracle, Sybase, and Informix have released Linux versions of their products.

A third explanation for Linux's popularity is seen in academe. At many technology-oriented universities, Linux has become the operating system of choice among students of computer science. One of our interviewees, Mr. Blue, made the point that trends in businesses are, more often than not, started in the academic arena. If a large number of students embrace Linux then when those students complete their education and go out into the business world, they will take with them their affinity for Linux. As a result, they are likely to use the system in business. Mr. Blue went further to say that trends in home computing are often determined by trends in business computing. Over time, says Mr. Blue, academic use will trickle down to wide spread business use which will in turn lead to home use. This is, of course, provided that Linux evolves into a well-rounded operating system.

Mr. Brown and Mr. Green believed that Linux needs commercial software support before businesses will adopt Linux. Mr. Blue felt that software support would grow proportional to the number of users because there would be an increased demand from the community. When it comes to Linux's growing popularity, Mr. White believes that this is mostly due to the Microsoft backlash while Mr. Orange, Mr. Purple, and Mr. Pink all feel that this is only a minor factor. These three also felt that Linux's current popularity in the academic environment would be a major contributor to its future in the business market.

4. Predictions and Conclusions

Based upon the knowledge and information that we obtained from our research and our interviews, we found that today Linux is not yet a suitable business solution. Linux has already progressed into the academic arena, but where Linux is currently lacking are the areas which are of the most interest to businesses. Presently, Linux's outstanding reliability can unfortunately, not make up for its shortcomings in ease-of-use and technical, hardware, and software support. This is, of course, not to say that Linux fails in any one of these areas. If Linux's current rate of growth and refinement continues, the operating system will surmount its current obstacles and indeed become a viable alternative in mainstream computing.

Appendix A1: The BSD License

The BSD License

Copyright (c) 1998, Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- All advertising materials mentioning features or use of this software must display the following acknowledgement:
This product includes software developed by the University of California, Berkeley and its contributors.
- Neither name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Appendix A2: The GNU General Public License

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 675 Mass Ave, Cambridge, MA 02139, USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too. When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it. For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights. We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software. Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all. The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this

License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix A3: The X Consortium License

X Consortium

Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Appendix A4: The Artistic License

Artistic License

The "Artistic License"

Preamble

The intent of this document is to state the conditions under which a Package may be copied, such that the Copyright Holder maintains some semblance of artistic control over the development of the package, while giving the users of the package the right to use and distribute the Package in a more-or-less customary fashion, plus the right to make reasonable modifications.

Definitions:

- "Package" refers to the collection of files distributed by the Copyright Holder, and derivatives of that collection of files created through textual modification.
- "Standard Version" refers to such a Package if it has not been modified, or has been modified in accordance with the wishes of the Copyright Holder.
- "Copyright Holder" is whoever is named in the copyright or copyrights for the package.
- "You" is you, if you're thinking about copying or distributing this Package.
- "Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.)
- "Freely Available" means that no fee is charged for the item itself, though there may be fees involved in handling the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you duplicate all of the original copyright notices and associated disclaimers.

2. You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.

3. You may otherwise modify your copy of this Package in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

- a) place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as ftp.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.
- b) use the modified Package only within your corporation or organization.
- c) rename any non-standard executables so the names do not conflict with standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.
- d) make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or executable form, provided that you do at least ONE of the following:

- a) distribute a Standard Version of the executables and library files, together with instructions (in the manual page or equivalent) on where to get the Standard Version.
- b) accompany the distribution with the machine-readable source of the Package with your modifications.
- c) accompany any non-standard executables with their corresponding Standard Version executables, giving the non-standard executables non-standard names, and clearly documenting the differences in manual pages (or equivalent), together with instructions on where to get the Standard Version.
- d) make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own.

6. The scripts and library files supplied as input to or produced as output from the programs of this Package do not automatically fall under the copyright of this Package, but belong to whomever generated them, and may be sold commercially, and may be aggregated with this Package.

7. C or perl subroutines supplied by you and linked into this Package shall not be considered part of this Package.

8. The name of the Copyright Holder may not be used to endorse or promote products derived from this software without specific prior written permission.

9. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The End

Appendix A5: The Official Open Source Definition.

The Open Source Definition

(Version 1.4)

Open source doesn't just mean access to the source code. The distribution terms of an open-source program must comply with the following criteria:

1. Free Redistribution

The license may not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license may not require a royalty or other fee for such sale. (rationale)

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of downloading the source code, without charge, via the Internet. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed. (rationale)

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software. (rationale)

4. Integrity of The Author's Source Code.

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software. (rationale)

5. No Discrimination Against Persons or Groups.

The license must not discriminate against any person or group of persons. (rationale)

6. No Discrimination Against Fields of Endeavor.

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research. (rationale)

7. Distribution of License.

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties. (rationale)

8. License Must Not Be Specific to a Product.

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution. (rationale)

9. License Must Not Contaminate Other Software.

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software. (rationale)

10. Conforming Licenses and Certification.

- Any software that uses licenses that are certified conformant to the Open Source Definition may use the Open Source trademark, as may source code explicitly placed in the public domain. No other license or software is certified to use the Open Source trademark.

Appendix A6: The initial interview

- 1) Introduction
 - a) Personal introductions
 - i) Tell them about ourselves and a little bit about the project
 - b) Explanations of proceedings
 - i) Ask if they mind us taping the interview
 - ii) Inform them that we will not use their name unless they want us to
 - iii) Explain to them how the interview will proceed and how we would like them to respond
- 2) Questions
 - a) Questions about their system
 - i) What system is your corporation currently using?
 - (1) - If they are not using Linux -
 - (a) What tasks do you perform with your system?
 - (i) Is it used for general business applications (office applications)?
 - (ii) Do you use it for specific tasks such as development
 - (b) Were you considering other Operating Systems
 - (i) - If they respond yes -
 1. What were the other systems you were considering?
 2. How were they brought to your attention, how did you hear about them?
 3. What strong points did they have that made you consider them as options?
 4. – If Linux was a consideration –
 - a. What were its strong points?
 - b. What were the factors that made you decide not to go with Linux?
 - c. – If Linux was not a consideration –
 - i. Why did you not consider Linux?
 - (ii) – If you only considered one operating system –
 1. What led you to believe that yours was the only viable option?
 - (2) – If you did decide to use Linux –
 - (a) What tasks do you perform with it?
 - (i) What tasks does it perform well?
 - (ii) What didn't like about the other systems you were considering?
 - (b) What were your deciding factors
 - 3) Conclusion, "Thank you"
 - a) Answer any questions they might have for us.
 - b) Thank them for their time and head home.

Appendix A7: Mr. Brown

- Uses Unix for his server base, (web, mail, news, dns)
- Not confident in moving from Unix
- Stability, programmability
- Started on Unix with a grant from digital, not worth the headache of moving
- Unix is cheap for large businesses and institutions
- Linux does not support the software that he needs to run (Maple, SAS given as ex.)
 - Does not fulfill needs
- Linux versions of software not as powerful as Unix versions
- Unix offers better performance on Alpha
- PC much cheaper – Unix not supported on PC
- Depends on what software is available
- Linux will come along
- Is hard to convert Linux -> pain in the butt
- Not enough software
- Technical support is a minor issue – comforting to know you have it when you need it.
- Software determines OS choice
- Thought it was interesting

Appendix A8: Mr. Green

- uses NT/Linux
- NT – Web server
- Linux – Mail Server (gateway)
- Linux is cheap
- * open source *
- Device drivers are hard to install in Unix, he likes Linux's modular kernel
- Frustrations with Microsoft
- The growth is there for Linux
- Linux needs corporate support before businesses will follow
- Businesses use whatever works
- Linux needs software support
- Hardware is cheap, so the fact that Linux can run effectively on old hardware is not much an issue
- Needs a user friendly GUI
- OS needs to be powerful, easy to use, and it shouldn't mess up
- Windows – easier development platform
- Learning tool – Linux

Appendix A9: Mr. Blue

- The fact that it is free but he felt that you often get what you pay for
- As an IT manager you look for:
 - Robust
 - Memory management
- Integrated solutions
 - Software and hardware from the same source
 - More robust
- Market demand – how to create market demand
- Liability
- SUPPORT
- Start with student support – grow to industry support
- University environment different from commercial environment
- Minimize hassle
- Safety in numbers
- Where are OS's going - Networking

Appendix A10: Mr. Orange, Mr. Pink, and Mr. Purple

- Linux will become a force
- Open-source in general will become a force
- Open-source programmers get fame rather than money
- “I want someone I can point a finger at”
- His boss asked him to look into Linux
- Less up front cost
- Businesses will pay money for good service
- “Tinkering costs money”
- Linux seen as a “tinkering” OS
- Linux unpolished
- Time is expensive
- Needs applications
- Linux great as a Server Machine
- “mindshare”
- Unix experts onboard
- Linux will overtake all of Unix
- Great in the Hobbyist environment
- Time is important
- Not a desktop OS
- Service Issue
- Can’t fire a volunteer
- Right now – leading edge
- Companies see Linux as a second shot

Appendix A11: Mr. White

- Linux will grow from business desktop
- People are sick of Microsoft
- Linux not necessarily a great operating system
- College students like it
- Large part of popularity is backlash against Microsoft
- More technical people lean towards Linux
- Lack of support will hinder Linux
 - Has to have some type of organized support
- There has to be a governing body
- Standards and Support mechanism
- Home market will grow from business market
- Use what works
- College -> work -> home
- Home computers -> networks

Bibliography

1. Berger, Robert. *Microscared: The Challenge of Open Source Software*. Linux Magazine, Spring 1999, vol. 1 num. 1. <http://www.linux-mag.com>
2. DiBona, Chris, Sam Ockman, Mark Stone. *Open Sources, Voices from the Open Source Revolution*. O'Reilly and Associates. January 1, 1999.
3. Doyle, James K. *Handbook for IQP Advisors and Students – Chapter 11: Introduction to Interviewing Techniques*. February 4, 1999.
<http://www.wpi.edu/Academics/Depts/IGSD/IQPHbook>.
4. Harmon, Amy. *The Rebel Code*. The New York Times, February 21, 1999.
5. Raymond, Eric S. *The Cathedral and the Bazaar*. November 22, 1998.
6. Seltzer, Larry. *Software Returns to its Source*. PC Magazine, March 23, 1999, vol. 18 num. 6.
<http://www.pcmag.com>.