Foreword

The public sector is becoming more and more standardised in its choice of IT products. Statskonsult wishes to point out the disadvantages of an over-standardised policy, and to assess the alternatives to the current solutions.

This report assesses a special phenomenon in software development called "Open-Source Software". One example of open-source software is the operating system Linux.

The report assesses whether open-source software can provide the public sector with a freer choice of supplier, and whether it can reduce total IT costs.

The report was written over an extended period of time. During this time, Statskonsult tested a number of different types of open-source software. The majority of staff in the IT Department of Statskonsult, internal IT operations staff and other interested parties from other departments, participated in the test. We also requested opinions from other departments in the public sector. Mr. Endre Grøtnes was project leader.

The report was written originally in Norwegian and translated into English. A lot of background information was translated from English into Norwegian for the compilation of the Norwegian version. Due to the translation back and forth there might be some minor inconsistencies in the text, especially in chapter 3-5.

Any comments to the report would be welcomed, and comments can be sent to Statskonsult (directorate of public management) at the following e-mail address: postmottak@statskonsult.dep.no or endre.grotnes@statskonsult.dep.no .

Oslo, August 2001.

Jon Blaalid

1 Summary

The press, people in the IT industry, and politicians have all recently been looking at Linux as an alternative to Windows. For this reason, and because Statskonsult would like to reduce supplier-dependency and the cost of buying software, the following report has been drawn up.

Linux is an *open-source*, freely available operating system. An open-source program means that the source code of the program is freely available; it can be used, altered, improved, extended and redistributed by anyone.

Statskonsult has assessed the suitability of Linux and other open-source software in the light of information it has collated, discussions with the companies in the field and its own tests. As well as being able now to form a more balanced opinion as to the usability of open-source software in administration, we have documented how open-source software will affect the development of the IT infrastructure.

Statskonsult believes that Linux and other open-source software have great potential – particularly in the areas of infrastructure development, the design of standards, and in teaching and training. At present, there are insufficient end-user, open-source programs for administration to replace its current software. There are no accounting programs, or corporate control systems, for Linux, for example.

Open-source software is currently an important contributor to the development of the Internet infrastructure. The basic standards for transferring data between computers, for sending e-mail and translating Internet addresses are all available as open-source software. The standards have also been developed and specified through an open-source process.

The recommendations for open-source software can be summarised as follows.

?? The use of open-source software to reduce costs would be beneficial for companies where IT costs are incurred primarily through the purchase of software licences, and where the costs of training users to use new software are relatively low.
?? Linux is ideal as a server operating system, even for the public sector. We do not believe that open-source software for end-users can currently replace the commercial end-user alternatives.
?? The state should urge schools and the education sector in general to use Linux and other open-source software.
?? The state should support the development of open-source software. Support could be offered in the form of research and development funding. In allocating research and development funds, the state should require that the software developed be made available as open-source software.

2 Introduction

2.1 Background

In its survey called "IT in the public sector 1999",[1] Statskonsult obtained the following figures on Microsoft's dominating position as the standard IT platform in the public sector.

?? 96% use MS Word as their word-processor.
?? 91% of the companies in the survey had a Windows NT server, and Windows NT was the operating system used by 60% of all servers.
?? Over 60% of all client computers used Windows as the operating system.[2]
?? Outlook and MS Mail accounted for 44% of all e-mail clients.[3]

Due to the dependence of public administration on one supplier, as indicated by the statistics, and to the ever-increasing costs of IT in general, it is only natural to consider alternatives to the Windows/Office platform.

The press, people in the IT industry, and politicians have all recently been looking at Linux as an alternative to Windows. In this report, we have defined Linux as an open-source, freely available operating system.[4]

Statskonsult has chosen to evaluate open-source software because it appears to provide opportunities to:

?? reduce supplier-dependency in public administration;
?? reduce the costs of buying software for public administration.

Public administration will only begin to use open-source software if it is easy to buy, install and operate. There must also be a substantial range of software to run on the operating system, so that standard office duties and other tasks can be carried out on it. Open-source software (such as word-processors and e-mail clients) intended for end-users must also be equally as user-friendly as the equivalent proprietary[5] products before the public administration will start to use it.

Statskonsult is responsible for the standardisation of data communications and appurtenant infrastructure in public administration. We have therefore also assessed how open-source software can help promote joint standards for the

[1] Statskonsult report 2000:8. IT in the public sector 1999.
[2] IT in the public sector 1999 did not ask directly which client was used, but based on the answers concerning use of the server operating system and the number of PCs / terminals connected, we have estimated this figure as the minimum.
[3] This was before the public sectors changeover to Outlook.
[4] We will use the term "open-source software" for the type of software that will be discussed in this report. English terms include "Free Software" and "Open-Source Software". See also the chapter on "open-source software" for a more detailed description.
[5] In this report, we will be using the term "proprietary software" for all software that is not open-source software.

exchange of information within public administration and between public administration and the outside world.

2.2 Supplier independence

One of the reasons why Statskonsult is evaluating open-source software is to determine whether the use of such software could reduce supplier-dependency in the public sector. This section looks at the disadvantages of excessive dependency on one supplier.

The concept of supplier independence is often perceived as being real competition. That is, we, as buyers, can select from a number of suppliers and products when we want to buy a product or service. Dependence on one sole supplier can be perceived as a monopoly situation and is undesirable. All the rules governing purchases in the public sector are based on the principle that there are several suppliers and that real competition exists between them.

Monopoly situations may lead to slower product development, poorer services, higher prices and greater vulnerability because only one product is used. In NOU 2000:24 "A vulnerable society" it says: *"Some suppliers have virtually a monopoly on software development. The disadvantages of such monopolies are not only related to the power such monopolies command in the market, and the inflated costs that they generate due to the lack of competition, but also the dependence of the market on them and the scale of damage if something goes wrong."*

Product development. It is generally assumed that competition accelerates product development and yields products that are better suited to end-users' requirements. There is currently considerable debate in IT circles as to whether end-users are best served by buying products from different suppliers, or whether it is in their interest to buy all products from the same supplier. In this report, we take the view that plurality of suppliers gives users greater choice and, hence, a greater opportunity to find the products and services required.

Services. Competition is particularly important in this area in order to ensure that users obtain the service they want at a reasonable price. In a monopoly situation, users have to accept what they can get. In the majority of telecom and IT areas, monopolies have been dissolved in order to improve the competition for services. One example of this is the dissolution of the telecom monopoly. This has led to a free market where any party can offer telecom services, and prices have consequently dropped considerably. In this report, we take the view that competition for services is of unconditional benefit for users.

Monoculture. The extensive use of a single product leads to what we call "monocultures". Monocultures are very fragile when faults appear in the respective product. One example of this in the IT world is the fault that arose in Intel's processors recently. This led to computer retailers not being able to meet

the demand for new computers. Intel has virtually a monopoly on the manufacture of PC processors.

Statskonsult views open-source software as supplier-independent. The main reasons for this are:

?? different suppliers can supply the same product since all specifications, documentation and even the source code are available to everyone;
?? no patents have been taken out on the software, and there are no restrictions on the use of any part of the software by any party.

2.3 Aim of the report and how it was carried out

The aim of the report was to assess how suitable the various forms of open-source software was for the public sector.

We have concentrated on assessing potential applications of the software in public administration, the financial consequences of choosing open-source software, and how such software can reduce supplier-dependency.

We tested a number of distributions of the operating system Linux, the e-mail server, Sendmail, and the web server, Apache, all open-source programs. The report concentrated on the use of open-source software for servers, but some attention was also given to its use for client machines. We also assessed the use of open-source software as a means of promoting standardisation.

We obtained feedback from various important groups and individuals who use open-source software in the public sector. Certain examples of how open-source software is used are described in this report. We also established contact with parties offering a variety of services relating to open-source software.

We encouraged and received considerable feedback, enquiries, and contributions from outside companies while we were compiling the report. We have made every effort to consider this feedback in drawing up the report.

2.4 The report

Chapters 3–5 describe what open-source software is, different financial models for open-source software, and the most widely used open-source software.

Chapter 6 gives examples of open-source software in the public sector. Chapter 7 describes in brief the open-source software and services currently on the market in Norway. Chapter 8 discusses, in particular, the potential of open-source software to promote standardisation, and Chapter 9 assesses applications of a variety of open-source software. Chapter 10 summarises the conclusions and recommendations we reached.

3 Open-source software

INFORMATION, NO MATTER HOW EXPENSIVE TO CREATE, CAN BE REPLICATED AND SHARED AT LITTLE OR NO COST.

? *Thomas Jefferson*

Sharing and copying information is now even easier than it was in Thomas Jefferson's day. It is worth noting that (the source code of) software is also information, and that the quotation can equally be applied to software. It is the ease with which information can be disseminated to large user groups over long distances[6] that explains how open-source software had gained the significance it has.

3.1 What is open-source software?

Open-source-code software, or open-source software, as we will call it in this report, is not easy to describe in only a few words. This is because there are a number of variations and definitions of the concept. The following description is only a summary, not a definition. In the following section, we will describe our interpretation of the concept "open-source software" by presenting the main principles of the concept.

An open-source program means that the source code of the program is freely available; it can be used, altered, improved, extended and redistributed by anyone.

There are clear definitions of both "open-source software"[7] and "free software".[8] Our interpretation of "open-source software" is that which fulfils the terms described in Section 3.2.

3.2 Principles of open-source software

The characteristics of open-source software are that it gives users[9] the freedom to:

?? use the program however they wish, on as many computers as they wish, and in any situations they wish;

?? adapt the software to meet their requirements; improve the software, fix bugs, expand the software, and study how the software works;

[6] This can be interpreted as the availability of the Internet.

[7] The Open-Source Initiative's definition of open-source software is available at: http://www.opensource.org/osd.html

[8] The Free Software Foundation's definition of "free software" is available at: http://www.fsf.org/philosophy/free-sw.html

[9] In open-source software environments, "users" are interpreted as skilled program developers and not just "point and click" end-users. The user perspective in the field is very developer–oriented.

?? re-distribute the software to other users who can then, in turn, adapt it to their requirements.

To fulfil the requirements above, there is one more condition that must be met:
?? Users must have access to the source code of the software.

Having the source code to a program, which will usually be written in a high-level programming language like Java, C, or C++, is essential for users to be able to understand how the program is built up and works, and to be able to adapt it, expand it, or fix bugs in it.

To maintain the freedom described in this section, it is necessary to protect the software with licence conditions and/or by copyright. This may seem rather paradoxical, but if software is not protected by licence conditions (called "public domain" software), anyone can copyright it, redistribute it without the source code, or use the source code as the basis for proprietary software. A variety of open-source software licences is described in Section 3.4.

3.3 Differences between open-source software and other software

Open-source software is not the same as freeware. Freeware is the name of compiled[10] programs that are distributed free-of-charge. There is a great deal of freeware available – everything from small help programs to well-known programs such as Microsoft's Internet Explorer.

Commercial software sold without the source code being available, and without the right to change or redistribute it, is called proprietary software. It is the licence conditions, in particular, of such proprietary software that has prompted the development of open-source alternatives. The majority of licences impose restrictions on the number of computers the software can be installed on, the right to change the software, and the right to redistribute it.

The licence conditions for proprietary software grant users the right to use the software subject to specific terms, not to owning the software.

3.4 Brief description of open-source software licences

The subject of licence conditions for software is vast. We have chosen to illustrate, below, a number of examples of the contents of open-source software licences and to explain why the designers have drawn up such licence conditions.

[10] A compiled program can be read by the hardware that is to use the program, and not by people. A program that is written in a language that people can understand is compiled into a language that the computer can understand. If you distribute only the compiled program, it is only the computer that can read and understand it.

The first licence for open-source software was developed by Free Software Foundation and related to its GNU[11] software. A description of the principles of this licence can be found at: http://www.gnu.org/copyleft/copyleft.html. The GNU software licence is called "GNU General Public Licence" and has become known simply as GPL.[12] This licence is reproduced in full as an appendix to this report. GPL is the basis of the majority of other open-source software licences. The Open-source Initiative had stipulated the conditions that must be met for a licence to be called an open-source software licence.

Two examples of open-source software licences are attached as appendices to this report: GPL (Appendix III) and BSD (Appendix IV).

One of the main reasons creating licence restrictions for open-source software is to protect the software from losing its open-source character. As mentioned above, open-source software can be made proprietary software if it is not bound by licence conditions. Licence conditions may include, for example:

?? safeguarding the free characteristics described in Section 3.2
?? safeguarding the conditions that the original designers aspired to (that the names of the original designers always be present, that the licence be distributed with the software, etc.)
?? safeguarding the rights of the program designers and of those who have copyright to the name of the software
?? charging designers of new software based on open-source software to ensure that such new software is also open-source software
?? charging all designers of software that uses open-source software to ensure that such software is also open-source software.

The various licences differ most in relation to the last two points above. While GPL is very strict in stipulating that all software based on the original code must also be made freely available, BSD-related licences grant the right to design proprietary software based on the original source code.

Open-source software licences do not prevent companies from selling such software, nor from earning money from such activities. Those who sell open-source software, however, cannot prevent other parties from selling the same software. The licence conditions cannot be used to restrict anyone's right to copy, or to change the software, and any programs sold must always be accompanied by their respective source code. GPL maintains that alterations and improvements in the code must also be considered open-source, and that anyone can use such new code, too. A good example of this is the installation program "RPM" (RedHat Packet Manager), designed by the Linux distributor, RedHat. This program is also distributed by Caldera and SuSE. Thus, any improvements made by one distributor can be used by the other distributors.

[11] GNU is a recursive acronym for "Gnus not Unix"
[12] The original text is available at: http://www.gnu.org/copyleft/gpl.html

It must be clearly stated on the packaging of open-source software distributed, that distribution is covered by an open-source software licence. Everyone has the right to assemble packages of open-source software and to distribute them.

3.5 Examples of open-source software

There is a great deal of open-source software available. Many of the most popular programs on the Internet are open-source programs. Such programs are often developed in open projects. In this case, a project is a small core of developers who publish new versions of a program, and who coordinate and use the input of a large number of contributors to enhance the program. Below is a short description of some of these programs.

3.5.1 Linux

Linux is the program that grabbed the attention of everyone previously outside the culture of open-source software; it is the program that most people refer to when trying to explain what open-source software is.

Linux, or, more accurately, GNU/Linux, is an operating system similar to UNIX. Linux is designed to complay to the POSIX standard to be as compatible as possible with other UNIX systems.[13] A program developed for a POSIX operating system should, in theory, be usable on all operating systems following the POSIX standard.

Linux was originally designed for running on Intel-based PCs, but is now available for other platforms, such as Macintosh, Alpha[14] and Sun SPARC.

Linux was developed and is distributed under the terms of GPL. This means that Linux can be distributed freely as long as the source code accompanies the program. If any parties want to distribute changes and modifications they make to Linux, they must also distribute the source code for such changes. In this way, the GPL ensures that the Linux operating system remains open-source and that no one can create a proprietary version of it. See Chapter 4 for more information on Linux.

3.5.2 Apache

The Apache projet is a voluntary development project aimed at creating a robust, commercial-grade, featurful, and freely-available source code implementation of an HTTPserver (henceforward called, slightly inaccurate, a”web server”).

The Apache server is the most used web server on the Internet, and has approx. 60% of the market, according to the surveys carried out by Netcraft. There are

[13] POSIX has been published as “IEEE Standard 1003.1-1988 Portable Operating System Interfaces for Computer Environments” and as ISO/IEC 9945.1: 1990.
[14] This was formerly DEC Alpha, but is now Compaq Alpha.

currently over five million Apache web servers on the Internet.[15] For a full and up-to-date summary, please see http://www.netcraft.com/survey/

Apache can be used on a number of platforms and with many operating systems. Amongst the most common operating systems used by the Apache web server are: FreeBSD, Sun Solaris and Linux. Apache may be seen as a reference implementation for the HTTP standard for all platforms for which it was developed.

Apache has many adherents. Some of the companies in Norway that use Apache for their external web services are: Aetat, the Data Inspectorate (Datatilsynet), Statistics Norway (SSB), Statskonsult, Dagbladet (national daily), VG (national daily), Startsiden (website), Sol (ISP) Stocknet and Telenor.[16]

3.5.3 Bind

BIND (Berkeley Internet Name Demon) is an implementation of the DNS protocol (Domain Name System) and provides a freely-distributable reference implementation of the most important components of the DNS protocol, including: a name-server, a DNS library for translating names to and from IP addresses, and a tool for verifying that a DNS server is working according to the standard.

BIND is the predominant name-server on the Internet and is essential for using names, e.g. www.statskonsult.no, to find the IP address on a computer on the Internet.

3.5.4 Perl

Perl is a programming language with particularly good text-manipulation qualities. It is therefore particularly suitable for making interactive and dynamic web pages. The majority of commercial web pages offering dialogue with the users and access to database servers are developed with Perl. One example of this is Amazon.com.

Perl can be integrated in web-servers, thus improving processing speed substantially. All in all, the use of Perl makes communication with base systems much easier and faster.

3.5.5 Sendmail

Sendmail is the most popular server for Internet e-mail. Approx. 75% of all e-mail servers on the Internet are variations of Sendmail. Sendmail is freely available and can also be re-distributed freely.

[15] These figures are from Netcraft, October, 2000.
[16] These details were obtained through the use of Netcraft's services in October, 2000.

Sendmail offers three main types of services: e-mail routing (including various forms of authentication and encryption devices), e-mail storage and retriving (including support for POP and IMAP) and opportunities for expanding the e-mail program (expansion such as delivery confirmations and integration with a PKI).

Sendmail adheres to, and has been developed in accordance with, current e-mail standards within the IETF. The latest versions of Sendmail also support ESMTP (RFC 1869).

3.6 Development methods for open-source software

The method of developing open-source software is very different from that used for developing proprietary software. The book "The Cathedral & the Bazaar" (2) describes the development method for, the driving force behind, and the culture of, open-source software. It is important to know how open-source software is created in order to be able to understand the conceptual difference between open-source and proprietary software.

The majority of open-source software projects start when one or more people feel that there is a piece of software missing, or that an existing program is insufficient or inadequate. They start by making a new program, often based on current software where the source code is available. When there are sufficient components in the program that work, not when the whole program is ready, the development group publishes it to a newsgroup, or similar, so that other people can use it and provide feedback and suggestions for improvements. If enough people show interest in the program and provide such feedback, a new open-source software project is born.

All open-source software projects have an owner. The owner is the person who receives the suggestions for changes in the program, suggestions for new code, corrections, improvements, and who publishes new versions of the program. This is an on-going process. At the beginning of a project, new versions of a program may be released every day.

All source code is published, and everyone can use the published source code for existing or new open-source software projects.

The owner's task is to develop a simple and effective program structure, to deal with all the changes and new code suggested, and, of course, to develop new versions of the program to include the changes and code he thinks appropriate. This means that not all the code suggested is included, and not all the suggested functions are implemented. Every program has a "read me" file that specifies who has helped with what in the development of the program. It is important to give credit to the contributors.

It is important for a project to have as many users of the program as possible in order to acquire as many contributors as possible to enhance the program.

3.7 What type of software is best for developing open-source software?

What can be developed as open-source software, and what has been developed as open-source software? These are important questions, and the answers will show what open-source software is suitable for.

The range of software stretches from infrastructure software to end-user software. Infrastructure software might be anything from the implementation of different types of server software and operating system kernels to DNS and TCP/IP. End-user software here means software that the non-technical user uses, such as word-processors, spreadsheets, accounting programs and various types of trade-specific systems.

The development of open-source software has normally started on the infrastructure side of the software range. Bind, Sendmail and Linux are all good examples. Other programs such as Apache and Perl are about half way between the two, but cannot be classified as non-technical end-user software.

What is clear to see is that open-source software is often software that other programs have to run on top of, or it is a tool for developing other programs. According to Brian Behlendorf, principle developer of Apache, there are many reasons for open-source software having been developed on the infrastructure- and server-side of the software range. Some of these reasons are described below.

?? End-user applications are difficult to make. It is not only because a programmer has to relate to an ever-changing non-standardised window development environment, but also because the majority of developers are not good graphical interface developers. Good end-user software requires a good graphical user interface. This, in turn, requires a test laboratory, user studies, etc, which the open-source software environment does not usually have access to.

?? Culturally, open-source software has focused on network programs and operating systems.

?? The development of open-source software benefits most from an environment where incremental development is possible. Historically, this has been easier on the server side, rather than on the client side, since the structure and architecture of software are better known, and it is easier to design a basic operating program that can be enhanced later.

?? A great deal of open-source software has been written by technicians to solve a problem that arose whilst developing another program. The target group for the software was other technicians.

To develop a program, you need to be well acquainted with the problems that can arise, or to have a good specification of requirements. Traditionally, the open-source software environment has not been aware of the requirements of

the various non-technical end-users, and has not been given good requirement specifications. The way in which open-source software projects start, and programs are developed, also makes it difficult for users to contribute if they cannot contribute code. But in a development group for proprietary software, it is perfectly normal for participants to be users.

The following quotation is intended to illustrate the necessity of being well acquainted with an area in order to be able to develop good programs for it. It is a quotation from Larry Augustin, President and CEO of VA Linux Systems.[17]

"Open-source developers understand UNIX. This is part of what made it possible to create a better UNIX-Linux. In order to create a better MS Office, open-source developers need to understand MS Office in as much detail as they understood UNIX. My fear is that the open-source developer community doesn't understand Office. It can't create what it doesn't understand. What we need is more developers using windows and Office."[5]

"They cannot create what they don't understand". This is the quintessential message. If developers of open-source software have to create an accounting system, for example, they need to know a lot about accounting and all the rules and routines pertaining to it – as well as be good programmers.

3.8 Advantages of open-source software

The motives for using and developing open-source software differ. They include everything from philosophical and ethical reasons to purely practical reasons. In the following section, we present some of the advantages of using open-source software.

The first advantage is that it is free and freely available. This advantage is not unique to open-source software. As described in Section 3.3, there is a substantial amount of proprietary software freely available. That which distinguishes open-source software from other software and gives it its advantages is how an individual can make use of the main principles behind the development of open-source software. These are described below:

?? *The availability of the source code and the right to adapt this.* The availability of the source code makes it possible to adapt and improve the program as required. It also makes it possible to create versions for new computer platforms and to gain detailed knowledge as to how the program works. If the program needs to be corrected, a user can make the changes himself, assuming he knows how. The program can also be adapted to the local environment – for example, in a different language, if the suppliers cannot supply one already made.

[17] VA Linux Systems sells computers with Linux pre-installed and customised to clients' special requirements. They also offer a range of consultancy services for Linux. Their web address is: http://www.valinux.com/

?? *It provides the opportunity to redistribute bug fixes and modifications.* It facilitates the creation of new programs based on existing code; such programs can then be distributed freely. Users can correct any bugs and deficiencies in the program and can then redistribute the modified version. If a new version is better, this will normally be adopted by the original program, so that this original program improves.
?? *The program can be used in the way that best suits the individual user.* Many free programs have limitations on how they can be used. A common example is that a program cannot be used for commercial purposes. Such restrictions do not exist in respect of software distributed under GPL. If users want to use the program on new platforms (e.g. in embedded systems, etc.), this is also acceptable.
?? *There are no hidden "black boxes" in the program.* In proprietary software, some elements of functionality may be new. The program may have certain functions that have not been documented and which the parties who made the program can use. These are often created as undocumented system calls.
?? *The development of software does not depend on just one company.* If the developer of the program does not want to crate new versions, or does not want to enhance it for a user's platform, the user is entitled to ask another supplier to take over such a job, or to do it himself. In the case of Linux, there are many suppliers of the same product, and all of them, in some way, enhance the product and adopt the best ideas of what the other suppliers do. The development and maintenance of the program does not depend on one company, and supplier-dependency is thus minimised.

3.9 Disadvantages of open-source software

Here are some of the disadvantages of open-source software.

?? *There are no guarantees that a particular program will be developed, or that its development will be in the desired direction.* All software projects need human and financial resources to get going. You never know whether a product will be developed sufficiently for it to be usable, or whether there will be enough interest in the product. This is the case for all types of software development, but the level of uncertainty is much greater in respect of open-source software projects.

   In all software projects, there is a period extending from the time an idea is formed and work is started up to the time the actual product is available for use. This is the most precarious phase of an open-source software project. If the open-source software project is started without adequate corporate support, it may be difficult to produce sufficient good-quality code for the project to attract other programmers who can use and develop the product. Without adequate support, or programmers, the project will just die, or fade out.

?? *It is sometimes difficult to know whether a particular open-source software project actually exists and how far it has come.* There is little marketing of open-source software, so it is difficult for a user to know if there is an open-

source program to suit his needs.

?? *Problems relating to patent rights and intellectual property rights.* Rights of intellectual property, or, more accurately, the breach of such rights, may be a major problem for open-source software. It is difficult to know whether a particular algorithm, or method of solving a problem, has been patented. Software patents are a problem for everyone developing software, but the subject affects open-source software more than other software for the following reasons. The availability of the source code makes it easier to discern infringements of intellectual property rights. The majority of open-source software projects do not have the funds to call upon legal aid in disputes relating to infringements of intellectual property rights.

4 Linux

As described in Section 3.5.1, Linux, or, more accurately, GNU/Linux, is an operating system similar to UNIX. It is distributed in various ways from various suppliers, and is also available for download via the Internet.

The following chapter describes Linux in greater detail, the various Linux distributions, and the software available for Linux. It will be rather technical and will require readers to have some technical knowledge.

4.1 What is a Linux distribution?

A Linux distribution is a collection of many different program modules called packages, which, together, constitute a complete operating system. When we buy a Linux distribution from a distributor, what we actually get is a collection of packages that the distributor has put together for us. RedHat Linux consists of over 400 packages from many different development projects.

A Linux distribution is more than just an operating system. It is a collection of open-source software components that allow users to utilise the operating system more easily and intuitively. Some of the components in the distribution are: a graphical user interface, compilers, word-processors and various server programs. All these components are open-source software. The distribution also often include proprietary utilities such as office support programs and games.

The number of components in a Linux package and the frequently accompanying installation programs prompt most people to buy a package rather than to download the components and install them individually.

4.2 The Linux kernel

The most important component of Linux is the kernel. To be more accurate, the kernel *is* Linux. The kernel is the system that controls communications (or the interface) with other programs and hardware.

The kernel in an operating system manages the file-handling, I/O, signaling, separation of processes, reception of interrupt reports from the various hardware devices, and the communications with other hardware and parts of the operating system.

4.3 Design targets for Linux[18]

Linux was developed for the Intel 386 platform, and was not originally designed with portability in mind. The Linux kernel was not designed to be portable, but rather to be efficient.

The Linux kernel was developed by considering the common characteristics of existing hardware architecture and designing the operating system based on what was common for these architectures. The kernel is termed "monolithic", and is not a "micro-kernel", which was the main trend when the development of Linux began.

In enhancing the Linux kernel, Linus Torvalds has laid down certain principles, as follows.

?? General, common-sense design principles should be used rather than designing the product for portability. Good, efficient design and code will, themselves, lead to the system being portable.

?? Designers should, as far as possible, avoid building programmable interfaces into the kernel. This is because once a programmable interface has been built into the kernel, it must be maintained in new versions because users will have started programming it.

?? Designers should be careful in adding new characteristics to the system. They should consider such characteristics carefully before adopting them in the kernel, since the kernel affects the entire system.

?? The kernel must be as modular as possible for several reasons. One reason is that modular systems are easier to adapt to specific hardware architectures since the whole system does not need to be changed. Another reason is that people can work independently on different parts of the kernel.

4.4 Graphical user interface

Linux was originally without a graphical user interface, but there are various graphical user interfaces that can be used with the kernel to render it a complete and modern operating system. Linux has more than ten different graphical user interfaces. These are called "window managers".

The graphical user interfaces run on top of XFree86 – a free version of the X Windows System. The X System forms the infrastructure of all the Unix-like Windows management systems. The X System is easy to configure and, to a large extent, you can determine the appearance of the user interface. You can configure the X System so that it resembles both Windows 95/98 and Macintosh.

Currently, the two most popular user interfaces for Linux are KDE (the K Desktop Environment) and GNOME (the GNU Network Object Model

[18] The text in this section is an edited extract from an essay written by Linus Torvalds in his book "Open Sources".

Environment). All Linux distributions are distributed with one or more graphical user interfaces.

4.5 Compilers

To adapt source code and make a separate version, you need a compiler that translates source code to machine code. So the better the compiler is, the easier it is to make customised versions, or versions for other platforms. Linux uses the programming language called “C”. This comes with the compiler in all the Linux packages. The most common compiler is called “gcc” (GNU C Compiler).

4.6 Software development

Linux is an excellent platform for developing software. C and C++ compilers and assemblers come as standard with all Linux distributions. There are also development tools for newer languages like Perl and Python. The reason that so much emphasis is put on software for software developers is that the opportunity to customise software is one of Linux’s greatest advantages. Without good software development tools, the availability of the source code and the possibility of changing the code would have little practical benefit.

4.7 Licence terms for Linux

Linux has been developed and is distributed under a GNU General Public Licence (GPL). See also Section 3.4.

Linux is open-source software, but anyone has the right to assemble a distribution and sell it. This, of course, would be subject to meeting the licence terms laid down in the GPL. Linux distributors cannot limit the terms of the GPL, or restrict further distribution by other parties in any way. If you want to install Linux on more than one computer, you only need to buy one distribution (CD), as you can use this to install the program on all the computers. There is one exception to this rule. To increase the value of their own packages, some suppliers include commercial software. The commercial software may have limitations as to how many computers it can be installed on. This should be clear, however, from the documentation enclosed in the distribution.

4.8 Variety of distributions

We have looked at four different Linux distributions and assessed their different characteristics. The packages are priced differently and have different accompanying software. The purpose of this evaluation was not to crown a winner, but to see whether the different packages add anything to Linux.

Below is a summary of some of the characteristics of the packages.

Distribution / Characteristics	Caldera	Corel	RedHat	SuSE
Price	NOK 345	$ 89 + $ 63 shipping NOK 1,260	$ 80 + $ 38 shipping NOK 977	NOK 345
Type	Caldera Open Linux 2.3 Standard	Corel Linux OS Deluxe	RedHat Linux 6.1 Deluxe	SuSE Linux 6.4
How purchased	Internet / via Akademika	Internet / direct from Corel	Internet / direct from RedHat	Internet / via Akademika
Delivery time	2 days	7 days	5 days	2 days
First/time installation	Partition Magic, Lizard	Corel Install Express	linuxconf	YAST 2
Installation of new package	COAS	Corel package installer	RPM	RPM, SaX
Server software included	Apache, Samba, Sendmail, DB2	Apache, Samba,	Apache, Samba, Sendmail, MySQL	Apache, Samba, Sendmail, MySQL
Office support software included	Star Office, Applixware, Wordperfect (trial)	WordPerfect	Star Office	Star Office, Applixware, Wordperfect (trial)
Operation and administration	COAS			
User-support included	90 days e-mail	30 days e-mail and tel.	90 days web 30 days tel.	60 days e-mail and tel.
Other user-support available	Telephone support at cost			

5 Financial aspects of open-source software

This chapter describes how you can earn money with open-source software, how the public sector can finance the development of open-source software and how the use of open-source software can reduce overall IT costs.

5.1 Business methods for open-source software

It is interesting to consider how it is possible to earn money through open-source software. Open-source software is freely available and can be downloaded free-of-charge. Any earnings therefore have to be made in ways other than selling the actual open-source software. This section describes various business models for earning money from open-source software.

5.1.1 Creating a brand name and distributing open-source software

This is the most obvious way of earning money from it. Companies such as RedHat and SuSE assemble distributions of Linux and sell them with limited user-support and printed manuals. Users do not pay for the actual software, but for the assembly of the various programs, user-support and for the packing and distribution. Buyers like to relate to a brand name and buy products from such companies, rather than buying similar but unbranded products.

5.1.2 Selling open-source software designed mainly in-house.

As in the case described in the section above, it is really a brand name which constitutes the product. Additionally here, though, it is the company selling the open-source software that is also the main developer of the software. Thus, it is this company that has the most expertise (since it knows the source code best) and which, consequently, has a marketing advantage when selling services such as user-support and special customisations. One of the disadvantages of open-source software is that developers spend a lot of money creating the software but are not remunerated directly for these costs. The critical sales point is to know the product better than competitors do and always to keep a few months ahead with new enhancements to the code. Cygnus, the main developer of GCC (Gnu C Compiler), the compiler, is a good example of a company that survives by having more expertise than its competitors have.

5.1.3 Proprietary products that increase the value of open-source software

Many companies design both open-source software and proprietary software based on open-source software. The open-source software is intended for the consumer market, while the proprietary software is intended for the corporate market. One example of this is Sendmail. Sendmail comes in a proprietary version and an open-source software version, but the former includes any new functionality four to eight months before the latter.

5.1.4 Consultancy services

This is a traditional activity. Give away the product and sell services such as training, operations and maintenance. All the distributors of Linux assume most of their income will be made through the sale of services. This is also one of the most important arguments used for supporting open-source software. IT companies must concentrate on offering good services to users since they cannot base their survival on income from licences; the services provided therefore improve.[19]

5.1.5 Selling accessories

This may appear trivial, but the sale of "accessories" is important for the open-source software market. The publishing company, O'Reilly, supports the development of open-source software by paying the main developers of the software. They recuperate this outlay by selling books on Linux, Perl and other subjects. Other parties in this category are those selling hardware with open-source software pre-installed, or that sell commercial software for Linux. Curiously enough, there is a big market for various kinds of Linux mascots.

5.2 Financing open-source software development

This section deals with the financing of the development of open-source software. It will form the basis of the proposals that the report puts forward regarding how the public sector can support the development of open-source software (see Chapter 10). It is based primarily on the EU-sponsored report on open-source software.[4]

5.2.1 Externally financed development

The development of open-source software is here financed by external parties. There are many reasons why someone may be willing to finance the development of open-source software. Many contributors are companies that earn their livelihood from open-source software in some way or other (see Section 5.1).

A major contributor is the public sector, which through funds for research and development can sponsor the development of new software for special applications. Typical contributors in Norway might be the Research Council of Norway and the Norwegian Industrial and Regional Development Fund.

Other contributors are those companies that sell accessories and that stand to benefit from the development of a particular open-source program. With Linux, there is a substantial amount of proprietary software (especially office support software) that needs drivers or other software to work correctly under Linux. Owners of this proprietary software are willing to pay for the development of the necessary open-source drivers and software.

[19] We have hitherto not seen any documentation to indicate that customer services have become better or worse as a result of users using open-source software.

5.2.2 Internally financed development

Companies that sell open-source software, or services directly linked to open-source software, will support the development of open-source software in order to expand the market for such products. If they are good, the market will expand and will probably lead to greater earnings. All the big distributors of Linux contribute in this way. Paying for the development of the respective product themselves, companies hope to create a larger market and, thus, to increase the possibilities of making money.

5.3 Significance of open-source software on total IT costs

Has open-source software any significant effect on a company's total IT costs? This question was one of the reasons for Statskonsult wanting to assess open-source software. It is impossible to give a general answer to this question, since all companies have different requirements when buying new IT equipment. However, we should like to give a general description of how a company can assess its own IT costs and concentrate on those areas where there are differences between open-source and proprietary software.

We have not assessed the usefulness or suitability of IT purchases for any company. This is something that all companies should do, however, though it is impossible to do on a general basis. What we will be looking at here is the costs in a costs/benefit assessment.

The term "total IT costs" is here intended to mean the costs of purchase and use of a given IT system throughout the period the IT system is used. We have divided the "lifetime" of a given IT purchase into periods: information gathering, purchase, implementation, training, day-to-day use, maintenance, and transfer to new technology. We will be looking at all these periods in detail.

?? *Information gathering.* This phase involves compiling information on the product to be purchased. It may prove difficult to find information on open-source software, since the distributors do not usually have their own marketing department. The information that exists may be more accurate than that for proprietary software since open-source software manufacturers generally do not earn money on the sale of their products. Furthermore, the code is available for inspection.

   The public sector can ensure through its purchase routines that information on open-source software is just as readily available as information on other types of software.

?? *Acquisition.* Purchases made in the public sector must adhere to the rules laid down for public sector acquisitions. These rules are intended to view all potential suppliers equally. If a company considers an item for purchase

based purely on its price, open-source software will obviously come out best, since such products have a very low initial cost. It is worth noting that a considerable amount of software is acquired with the purchase of new hardware, or through the upgrading of existing software, so it is impossible to draw up a calculation to cover all companies even in this isolated field.

?? *Implementation.* This covers installation and initialisation of the system so that it is ready for use throughout the company. Large companies selling proprietary software are now better equipped for this work than companies that sell open-source software, because they have traditionally been more involved in this phase. If big companies are to use open-source software, the sellers of open-source software must set up a system able to handle distribution of large volumes.

?? *Training.* Training costs should not be influenced at all by whether the software is open-source or not. There is currently a bigger range of training courses for proprietary software than there is for open-source software. The price is not different, but it is easier to find a course that is suitable for an individual company with proprietary software. There are also many more users who know proprietary software than those who know open-source software, so some costs would be incurred for training if a company chose to change system.

?? *Day-to-day use and maintenance.* This is often the phase that incurs the greatest proportion of total IT costs. Advocates of open-source software claim that their software makes maintenance cheaper, but we have not found any documentation to prove this yet.

?? *Transition to a new system.* New IT systems are often built to replace old systems. If a company has the source code (not necessarily open-source code), the transfer to a new system will often be easier, since the respective company will have a better idea of how the former system was implemented. The most important thing in this phase is that the system is possible to document and that the data from the previous system can be used on the new one.

We can see from an assessment of the various phases of the lifetime of an IT system, that it is only the purchase phase that clearly stands out in favour of open-source software. One of the major costs of moving over to open-source software is the training of the users of the new system. In the other phases, it is difficult to see any particular differences in the costs of using open-source software compared to other type of software.

Our conclusion on the issue of whether the public sector can reduce its costs by changing over to use open-source software is as follows. The use of open source software to reduce costs would be beneficial for companies where IT costs are incurred primarily through the purchase of software licences, and

where the costs of training users to use “new” software are relatively low.

6 Examples of the use of open-source software in public administration

Apart from the use of Linux and Apache for web services, we have few examples of the application of open-source software in public administration. The examples we do have come from the education sector.

6.1 The University of Oslo

The University of Oslo uses open-source software in many ways. Many of the faculties provide end-users with access to Linux. Many run Linux on their servers, which sometimes run highly advanced technical applications. Apache is widely used as a web-server, and Sendmail and Bind are the principal infrastructure programs.

The above-mentioned programs are perfectly common applications of open-source software. Another use of Linux which, until now, has been uncommon, but which is very interesting, is that of using it as a thin client. This means that Linux is only used for handling user interfaces from programs running on the servers.

All user applications are stored on the server. These may be common office support programs or more technical programs. The servers run Windows or NT. The client is used only for communicating with the server and for displays. Users see the customary Windows screens, but the operating system is Linux and it is this which is generating the screen displays. This operating method avoids the need to install Windows on all the end-users’ PCs.

Thus, by replacing Windows with Linux, but keeping the old software, it is possible to save considerably on licence costs. The disadvantage, or perhaps the advantage, of this is that you need to change to an environment with thin clients, so end-users lose direct control of what is on their respective computer.

In environments where a company wants to introduce thin clients, but not to let the users have direct access to local software, it is a good idea to replace Windows with Linux on all the clients. More and more universities and technical colleges are using Linux in order to introduce thin clients.

6.2 Høle Primary School

Høle Primary School has installed Linux on old PCs that it has obtained from the business community. The PCs were without an operating system, so this had to be installed. The school chose Linux and thus saved considerably on licence costs.

7 Commercial services for open-source software in Norway

7.1 Purchasing software

It is easy to buy open-source software in Norway. We ordered the software from Akademika and the package arrived at our door two days later. A standard invoice was enclosed so we could settle up in the usual way. There are many shops in Norway that sell open-source software via the Internet, and many that sell Linux distributions over the counter.

It is also possible to order Linux distributions directly from the suppliers. This takes longer and, taking into account the costs of shipping and import duty, is more expensive than buying from suppliers in Norway. The advantages are that you can obtain versions that are not available in Norway, and you can obtain the latest versions more quickly.

Commercial software for Linux is sold in the same way as other commercial software. Large companies like IBM, SGI and Oracle all produce Linux versions of their software. They also have a support department for helping those customers who choose to run their software under Linux.

7.2 Purchase of computers with ready-installed software

Many suppliers now sell their computers with open-source software already installed. Such computers are usually set up with Linux, but server software such as Apache and Sendmail may also be installed.

7.3 Consultancy services

Consultancy services are provided by companies whose sole activity is consultancy, and by the suppliers of commercial software. The consultancy companies include large international companies that provide “everything”, and small niche companies who support only Linux and associated software.

7.4 Courses and training

More and more companies are offering training courses in the use of Linux and other open-source software. Such courses are often held by companies that sell software for Linux and Unix, and by universities and technical colleges. There are still, however, fewer training courses for Linux than there are for Windows.

8 Use of open-source software as a standardisation and development tool

Standards and standardisation are important components of IT development. Open-source software will have a significant influence on future standardisation. Some parties involved in IT standardisation, such as the IETF, reflect the culture of open-source software, while other parties wish to use open-source software as a tool for the distribution and better interoperability of standards.

8.1 The IETF and open-source software

The way in which the IETF (Internet Engineering Task Force) is organised and functions is a good example of "Bazaar" development principles, as put forward by Eric Raymond [1].

The IETF defines open standards for the fundamental protocols on the Internet, and standards for various shared applications such as security, transport services and administration. All IETF standards are free and readily available on the Internet.

In order for the IETF standard to continue to remain a standard, there must be, within a given time, at least two different implementations of the standard that work together. Open-source software is important in this respect, as it can spread implementations of the standard and can test various products against each other.

The IETF is open to everyone; you become a "member" simply by participating in discussion groups and meetings. Everyone is treated as an individual and not as a representative from a company or public authority.

Both the standardisation process and the standards are based on openness.

8.2 Internet and open-source software reference implementations

A reference implementation of a standard is an implementation that other parties can refer to, or test something against, when creating interoperable products based on this standard. A reference implementation should implement all the properties of a standard and in the correct way. Apache is an example of a reference implementation of HTTP (the standard). Internet browsers must be able to communicate with Apache in order to be able to claim that they are a correct implementation of the standard.

Part of the success of the Internet is that the products based on the Internet standards are interoperable. This creates a well-integrated whole. Most of the fundamental protocols used by the Internet exist as open-source software reference implementations. Reference implementations are often released early

and command a large part of the market. Other products have to be interoperable with the reference implementation in order to gain a foothold in the market. Examples of this are: TCP/IP, DNS, SMTP and HTTP.

8.3 Open-source software as a catalyst for the spread of new standards

In order to develop products based on standards, it is important that implementations of standards exist that the developers can build on. Open-source software plays an important role in this respect. If the standard has been implemented as an open-source software reference implementation, developers can use this implementation of the standard to develop new products. If only proprietary implementations of the standard exist, developers wishing to develop new products based on the standard must first implement their own version of the standard, or buy a licence to use a proprietary version. This slows down new developments and the process of adopting new standards.

In connection with the development of new standards to support the implementation of the EU directive on electronic signatures, the EU Commission has requested the development of open reference implementations of some of the already agreed standards from ETSI. This is intended to facilitate product development and research in the field.

In Norway, the Norwegian Research Council and the National Business and Regional Development Fund might be suitable organisations for promoting the use of open-source software in development work. The Norwegian Standardisation Association might be the right party to promote open-source software in the field of IT standardisation.

9 Assessment of the applications of open-source software

9.1 Assessment methods for open source software

In choosing our assessment criteria, we have differentiated open-source software intended for servers and infrastructure, and open-source software intended primarily for use by normal end-users.

We have concentrated on the following factors when assessing open-source software for servers and infrastructure.

?? *Acquisition.* The software must be easy to obtain.
?? *Installation.* The software must be easy to install – both for the first time and when installing upgrades.
?? *Support.* Users must have access to professional software support.
?? *Administration and operation of the system.*
    ?? Flexibility, simplicity and the possibility of customisation.
    ?? Compatibility with other systems.
    ?? Market position
    ?? Operating costs
?? *Standards.* Does the program adhere to current standards?
?? *Training.* Users must have access to training in the use of the system.
?? *Usability for the public sector.* Is the program usable by the public sector?

We concentrated on the following factors when assessing open-source software for end-users (the operating system with programs such as word-processors and e-mail).

?? *Acquisition.* The software must be reasonably priced and easy to obtain.
?? *Installation.* The software must be easy to install – both for the first time and when installing upgrades. New programs must be easy to install and remove.
?? *User support.* User support must be available for the programs installed.
?? *Training.* Training must be available for learning how to use different programs.
?? *User-friendliness.* The programs must be equally as user-friendly as corresponding commercial software. In respect of office support programs, it is natural to choose Microsoft Office as a source of comparison.
?? *Language.* The programs must be available in Norwegian.
?? *Operating costs.* Costs of running the programs over the long-term.
?? *Usability for the public sector.* Is the program usable by the public sector?

We selected the criteria based on what we considered to be the most important factors covering every stage of acquisition and use of software. Based on these

criteria, it had to be possible to analyse the total costs of obtaining and using the software.[20]

9.2 Which branches of public administration are best suited to using open-source software?

Public administration covers a vast array of fields. Not all of these fields are well suited to using open-source software. Departments and sectors with large, complex, custom-built programs, and departments dependent on the use of custom-built administration systems are not well suited to using open-source software. This is because i) such open-source software does not exist, and ii) such software is not suitable for development as open-source software.

Open-source software is better suited for use in environments where word-processing, e-mail and Internet are the main activities. This category covers many ministries, some directorates, and large areas of the education sector. All these parties could use open-source software. The education sector uses a wide variety of software, while the directorates and ministries do not.[21] The education sector thus seems more adept at using new software than the ministries and directorates. Furthermore, one of the goals of the education sector is to provide training – essential for using new software. The training staff at technical colleges and universities are also highly skilled in the field of IT. All in all, it is the education sector which would appear to be the best place to start using open-source software in public administration.

We do not believe that open source software for end-users can currently replace the commercial alternatives due to the better functionality and integration capabilities in the commercial alternatives.

9.3 What types of open-source software is suitable for public administration?

As with other software, there are significant differences in user-friendliness and usability of open-source software. In this report, we have chosen to assess software for end-users and servers. Software for end-users includes graphics, word-processing, spreadsheets, browsers and e-mail clients. Server software includes e-mail, web page set-up, and standard file management.

9.3.1 End-user software

There are several good word-processors and spreadsheets for Linux. The disadvantage is that many of them are not open-source software, so they have the same disadvantages as those of other proprietary software. The word-processors and spreadsheets that exist as open-source software regrettably lack the holistic and integration characteristics that the proprietary products have. In respect of browsers and e-mail clients, many products are available for running

[20] This is often termed "total cost of ownership (TCO)" for IT products.
[21] This can be extracted from the background figures in the report "IT in the public sector 1999"

under both Linux and Windows. Not all of these products exist as open-source software, but the cost of a licence to use them is often small, or it is free-of-charge.

There are currently no word-processors, spreadsheets and e-mail clients developed as open-source software that have the same completeness and functionality as the commercial products. Considering the majority of users will require training and a period of adjustment when changing over to new products, such products should have approximately the same functionality as the old ones. Price is less significant for end-users in this respect. If staff are to undergo training and have not become dependent on the functionality of proprietary products, open-source software is ideal. We believe that open-source software intended for end-users will be most appropriate for the education sector and for particularly well-motivated and interested users.

9.3.2 Server software

The situation is different for server software. Most software for developing and running web-services is already open-source software. There are also many open-source server programs for e-mail. In these areas, there is clearly great potential for the use of open-source software in public administration. Unfortunately, however, it is also in just these areas that developers of proprietary software offer their software free-of-charge, or at discounted prices, so that changing over to open-source software will not necessarily save much money. It may be wise, however, to use open-source software to prevent excessive dependence on proprietary software, which is particularly likely to occur in ministries and directorates.

Open-source server software is functionally and administratively equally as good as proprietary alternatives. Suppliers may not, however, be able to provide the same level of support. This situation is changing, though, since many large suppliers are now offering support for open-source server software such as Apache, Samba and Sendmail.

Our opinion is that Linux is well suited for running on the servers in public administration – particularly so if Apache is also run as a web- server.

10 Conclusions, measures and recommendations

10.1 Conclusions

The public sector does not benefit from being too dependent on just one type of software. Every business should, as far as possible, be free to choose the software it believes to be the most suitable for its tasks. This means that businesses need to be acquainted with different products and services. *Open-source software is a cheap way for users to get to know new software.*

Public administration should be aware of how to benefit from open-source software. Having access to the source code means that more people can get to know the fundamental technology of the respective product, and this may make it easier to develop and spread new technology. *Open-source software will become increasingly important, particularly in the fields of education, research and development.*

Open-source software is predominant in connection with the implementation of Internet infrastructure. Both the development and the implementation of standards for the Internet are based on the ideas and thoughts of open-source software. *The Norwegian authorities should be aware of the possibilities that open-source software provides for rapid distribution of new standards in the market.*

Parties developing open-source software via the Internet need to have a very clear understanding of what needs to be developed. *The development of server and infrastructure software as open-source software is thus ideal.* The reason for this is that there are often clear specifications and standards that the developers can use. It is usually the case that the parties who are engaged to implement a standard were previously involved in defining it. End-user programs are not so well suited for development as open-source software, since there are no standards for these and it is more difficult to define a clear specification. This problem is also reflected in what open-source software that is avalable.

It is not always the best technology that wins through. This is important to remember. There are many non-technical reasons why people choose one operating system, or device, rather than another. These might be, for example, the marketing of the product, design, price, availability of accessories (for operating systems, this might be the programs available, and the hardware supported), what users associate with the product and which other users use it.

Our assessment of Linux and open-source software is thus not merely a technical assessment, but is a holistic assessment that takes into consideration users' experience and the investments that the public sector has already made in IT.

It cannot be taken for granted that open-source software is cheaper than commercial software. *In our view, the use of open source software to reduce costs would be beneficial for companies where IT costs are incurred primarily through the purchase of software licences, and where the costs of training users to use "new" software are relatively low.*

Linux is primarily an operating system that was intended for use on servers. This is reflected in the method of marketing the majority of Linux distributions and in the documentation that accompanies the programs.[22]

Linux is a variant of Unix, so it is natural that Linux is used primarily in those situations that other Unix operating systems were formerly used. Unix systems were customarily used on servers, and we can see that Linux is also used primarily for the same purpose. Those environments in which Unix was used will be the first to use Linux.

Linux could play an important role particularly in the education sector as an operating system for both servers and end-users. By encouraging the use of different software in training activities, users will benefit from broader experience and will be in a better position to choose software themselves later.

Linux is now a good operating system for servers and would be well suited for use by the public sector. We do not believe that open source software for end-users can currently replace the commercial due to the better functionality and integration capabilities in the commercial alternatives..

10.2 Ways of increasing the use of open-source software

There are many ways of increasing the use of open-source software in public administration. This section describes just a few of them.

?? Make the use of open-source software mandatory
?? Support the use of open-source software
?? Distribute and develop open-source software
?? Support research and development of open-source software

10.2.1 Make the use of open-source software mandatory

The mandatory use of open-source software would be very effective in some circumstances. The obligation to use a particular technology is always, however, rather risky. Is it the right technology? Are the business areas that

[22] The following quotation is from the printed documentation accompanying Caldera OpenLinux 2.3: "*Today, Linux is first and foremost a server operating system. Although many applications are now appearing that allow Linux to be used as a primary workstation or desktop system, most users of Linux focus on the server capabilities of the operating system.*"

have been instructed to use the technology ready to implement it? Are the prospective users sufficiently motivated to use the new technology?

In general, it is better to avoid making the use of a particular technology mandatory, and rather to point out the advantages and disadvantages of it, so that the target group can itself choose whether or not to implement the technology.

10.2.2 Support the use of open-source software

By organizing the enviroment better you can support the use of open-source software. There are various ways of doing this. You can simplify the acquisition process, providing training in the use of open-source software, or you can provide help with its operation.

The optimisation of the environment for using open-source software will promote greater use of such software, yet without forcing anyone to use it against his will.

10.2.3 Distribute and develop open-source software

One way of supporting the use of open-source software is for the state itself to distribute its own version of Linux with other appurtenant open-source software. This would make it easier for companies to start using open-source software. On the other hand, it may upset the market if the state starts distributing its own Linux distributions. The content of the package is also an important issue. Another issue is whether the cost of such distribution would be justifiable, considering that it is free to redistribute existing Linux distributions.

One less exclusive solution would be for the public sector to copy and distribute existing Linux distributions, so that those parties wishing to try out open-source software could easily obtain it by applying to a public authority.

The public sector should also consider whether it could supply software that it currently has ownership rights to, as open-source software.

10.2.4 Support research and development of open-source software

By supporting the development of open-source software, you help to enhance the infrastructure and to make the technology available for everyone. As mentioned above, it is becoming a trend that new standards are being implemented as open-source software. If the contributors of funds for research and development were to demand that the result of the research had to be made available as open-source software, the research would be easier for others to make use of. We believe that in allocating research and development funds, it should be a requirement that the software developed be made available as open source software.

10.3 Recommendations

?? Linux is a product that the public sector should support in order to promote the development of a potential alternative to Microsoft's operating systems. Linux is currently best suited for use as a server operating system.

?? The state should support the development of open-source software to promote alternatives to current software. New open-source software could promote the enhancement of current software and prevent the public sector from becoming too focused on one direction. Support could be offered in the form of research and development funding.

?? In allocating research and development funds, the state should require that the software developed be made available as open source software.

?? The public sector should also consider whether it could publish source code that it currently has ownership rights to as open-source software.

?? The state should urge schools and the education sector in general to use Linux and other open source software. This is because schoolchildren and students must acquire as much knowledge as possible about a variety of products in order to build up a solid foundation on which to be able to choose at a later date the products they believe are the most suitable.

?? To save on licence costs, used PCs that are given to schools could be equipped with open-source software.

?? The infrastructure should be based on open standards implemented as open-source software. The public sector should lay down requirements for the use of open standards implemented as open-source software in the infrastructure it uses. The use of open standards and solutions based on open-source software could be advisable in connection with the establishment of infrastructure for the distribution and handling of digital signatures.

Appendix I References

[1] Chris DiBona, Sam Ockman & Mark Stone (Editors): Open-sources – Voices from the Open-source Revolution.
O'Reilly & Associates, 1999. ISBN 1-56592-582-3

[2] Eric S. Raymond: The Cathedral & the Bazaar.
O'Reilly & Associates, 1999. ISBN 1-56592-724-9

[3] The Open Source Initiative. The open-source definition, 1999.
Available at: http://www.opensource.org/osd.html

[4] Free Software / Open-source: Information Society Opportunities for Europe? Version 1.2 April 2000.
Working group on Libre software[23]
Available at: http://eu.conecta.it/paper.pdf

[5] Linux Journal. June 2000 "The new beginning"

[6] NOU 2000:24 "A Vulnerable Society"

[7] Open-source: the unauthorized white papers. Donald K. Rosenberg
M&T books, 2000. ISBN 0-7645-4660-0

[23] The work group for Libre Software was assembled on the initiative of the EU Commission's commissioner for the area "Information Society".

Appendix II Abbreviations

BIND	Berkeley Internet Naming Daemon
BSD	Berkeley Software Distribution
DNS	Domain Name System
ESMTP	Extended Simple Mail Transfer Protocol
ETSI	European Telecommunications Standards Institute
GNOME	the GNU Network Object Model
GNU	Gnu's Not Unix
GPL	GNU General Public Licence
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IEC	International Electrotechnical Committee
IEEE	The Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IMAP	Internet Message Access Protocol
IP	Internet Protocol
KDE	The K Desktop Environment
OSI	Open Systems Interconnection
PKI	Public Key Infrastructure
POP	Post Office Protocol
POSIX	Portable Operating System Interface for unIX
RFC	Request For Comments
RPM	RedHat Package Manager
SMTP	Simple Mail Transfer Protocol
TCO	Total Cost of Ownership
TCP	Transmission Control Protocol

Appendix III GPL (GNU General Public Licence)

GNU GENERAL PUBLIC LICENCE

Version 2, June 1991

Preamble

The licences for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licence 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 Licence 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 Licence 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 Licences 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 licence 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 licences, 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.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This Licence 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 Licence. 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 Licence; 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 Licence and to the absence of any warranty; and give any other recipients of the Program a copy of this Licence 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 Licence.

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 Licence. (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 Licence, 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 Licence, 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 Licence.

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 non-commercial 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 Licence. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this Licence. However, parties who have received copies, or rights, from you under this Licence will not have their licences terminated so long as such parties remain in full compliance.

5. You are not required to accept this Licence, 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 Licence. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this Licence 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 licence 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 Licence.

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 Licence, they do not excuse you from the conditions of this Licence. If you cannot distribute so as to satisfy simultaneously your obligations under this Licence and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent licence 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 Licence 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 licence 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 licencee cannot impose that choice.

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

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 Licence 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 Licence incorporates the limitation as if written in the body of this Licence.

9. The Free Software Foundation may publish revised and/or new versions of the General Public Licence 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 Licence 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 Licence, 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

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

one line to give the program's name and an idea of what it does.
Copyright (C) yyyy name of author

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public Licence as published by the Free Software Foundation; either version 2 of the Licence, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public Licence for more details.

You should have received a copy of the GNU General Public Licence along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public Licence. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright
interest in the program ‘Gnomovision’
(which makes passes at compilers) written
by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public Licence does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this

is what you want to do, use the GNU Library General Public Licence instead of this Licence.

Return to GNU’s home page.

FSF & GNU inquiries & questions to gnu@gnu.org. Other ways to contact the FSF.

Comments on these web pages to webmasters@www.gnu.org, send other questions to gnu@gnu.org.

Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA

Updated: 3 Jan 2000 rms

Appendix IV The BSD licence (Berkeley Software Distribution)

Copyright (c) <YEAR>, <OWNER>
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.

?? Neither the name of the <ORGANIZATION> 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 COPYRIGHT HOLDERS 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.