The development of Unix

By Kasper Edwards[*]

Departmnent of Technology and Social Sciences, Technical University of Denmark
Building 303 East, room 150, 2800 Lyngby, Denmark. (email: ke@its.dtu.dk)

ABSTRACT

This paper tells the story of the development of the Unix time sharing system. The development at AT&T and the MULTICS roots are uncovered. The events are presented in chronological order from 1969 to 1995. The Berkeley Software Distribution (BSD) are presented as well as the Free Software Foundation and other.

1.1 Introduction

This thesis about Linux, however Linux is called a Unix clone in the sense that it looks like, and are designed on the same principles as Unix. Both Unix and Linux are POSIX (Portable Operating System Interface) compliant (described in paragraph 3.29). In short POSIX describes the Unix user interface, i.e. commands and their syntax. Some Unix'es are certified POSIX compliant but no one have yet been willing to pay a third party company to test the POSIX compliance of Linux. It should be noted that the speed of which new versions of Linux appears makes it almost impossible to maintain a certification.

A note on spelling, Unix is though out this thesis written Unix in stead of UNIX with capitals. This is done because one of the original developers (Doug McIlroy) [Salus 1994:ix] of Unix once noted that spelling Unix with capitals had been a grave error.

1.2 Table of contents

1.3 The sources used to produce this chapter

This paragraph provides a brief description of the literature that provides the foundation for this chapter. Peter Salus book "A quarter century of Unix" is used extensively in the chronology of the Unix evolution. Salus book provides detailed descriptions of how Unix came about. Peter Salus has strong ties to the Unix community and is the editor of the quarterly journal "Computing Systems" and has a column in the USENIX newsletter *;login:*.

Two papers by Dennis Ritchie is used "The Development of the C Language" and "The Evolution of the Unix Time-sharing system". Dennis Ritchie worked close with Ken Thompson in the development of Unix. Dennis Ritchie provides insights into the gory details of the development process.

Kirk McKusick have participated in the development of the Berkeley Unix and been a member of the Unix development team.

Richard M. Stallman is the founder of the Free Software Foundation and has had and still has strong influence on the free software community.

Other authors are used in this chapter but not to an extent that warrants an introduction.

1.4 Vocabulary

Assembler   Symbolic machine code.

Binary   Representation of data often described as a sequence of binary digits (0 and 1). Resembles a physical representation of currents or directions of magnetization.

Binary program   Representation of a program as machine code.

BSD Unix   Berkely Software Distribution Unix

Code	A lose term describing a program written in a programming language i.e. source code.
Compiler	A program that translates from a programming language to object code.
Debugger	Software that helps the programmer trace errors.
DEC	Digital Equipment Corporation.
Linker	A program that is able to combine objects to a binary program.
Library	A set of binary programs that other programs can link to examples are mathematical functions like cos, sin and tan.
Machine Code	A sequence of machine instructions in a binary representation.
Machine - instruction	A single instruction that a specific computer can execute. A programmer regards this the smallest "building block"
Object code	A representation of program pieces that can be combined by a linker to a binary program. The pieces may have been written in different programming languages.
Open Source	Software to which the source code is provided.
Platform	A family of hardware that are compatible for example the X86 family of Intel processors. As operating systems abstract the hardware away, it has become custom to define the platform by the operating system.
Port	Taking a program from one type of computer to use it on another type of computer. Porting a program often requires the program to be rewritten to accommodate the language of the target computers processor.
POSIX	Portable Operating Systems Interface, a standard describing the interface of an operating system.
Shell	A separate program that interprets user commands and passes the instructions to the operating system. It provides a simple interface where one can manipulate the file system that is copy, delete and execute files.
Source code	The code for a program that was used for compilation, the source of the compilation.
Unix System V	A different version of Unix, developed by AT&T.
Unix Vx	The original Unix developed by Bell Labs Computer Systems Research Group.

1.5 Discussion of terms

Terms, words and phrases that are used ambiguously or contain the views of different political sides will be discussed in this paragraph. The use, meaning and definition that will be adopted in this thesis will be argued.

For example, terms like Open Source and Free Software contain a large discussion of which to use. There has been a battle raging in the community about which term to adopt in the public promotion of this special way of developing software.

1.5.1 Free software and open source

The term Free Software was coined by Richard M. Stallman (Described in The Free Software Foundation paragraph 3.32 page 17) in the early 1980's. The term has been used extensively by the Free Software Foundation which have made extensive argumentation about the use of the term and why this is the correct term to use.

Open Source is a term that was constructed as a reaction to the confrontational attitude that had been apparent with The Free Software Foundation and Richard Stallman on the one side arguing for the complete freedom of software and others with a pragmatic view. The pragmatic wing argued that companies had to be involved for this development model to be a wide spread success. It so happened that Netscape had decided to release parts of the source code for their Communicator Browser and was looking for advice from the community. A meeting took place in Palo Alto (Calif.) in February 1998 and the term Open Source was chosen [OpenSource.Org 1999:/history.html].

The Open Source License is more business friendly and allows for some degree of control of the source code by the original developer, that is, company or private. This is the hart of the argument, should the term appeal to the freedom of the users or should there be a possibility of a business-related trademark.

Today the community could be divided into three groups with some overlap in between. One group originates from the philosophies of the Free Software Foundation these are hard-liners that only accept licenses that provide the total freedom of software to the user. It is allowed to make money from sales but not to impose any restrictions on the user and he's freedom to use or modify the software. The users are however obligated to return any changes made to the community.

The other group is the people who likes the term Open Source. This group is more lax on total freedom of software and wants to embrace the businesses so the development model can be used on a wider scale. It seems as if the Open Source group is winning the PR race against the Free Software Foundation, in the sense that all most all press releases and articles use the Open Source term.

Then there are the very large group that use both terms and doesn't separate their different meanings. Most people are aware of the Open Source term but have not heard of the Free Software Foundation and don't care much about the political implications.

In the effort of trying to avoid a political and/or a philosophical discussion that is beside the scope of this dissertation the following convention of the use of terms will be adopted:

The choice of term will depend on the license used on the particular piece of software. Software that uses a version of the Gnu Public License (GPL license defined by the free software foundation) will be referred to as Free Software. The GNU license is cover in depth in chapter 3.32.2. GNU is an recursive acronym that means Gnu Not Unix.

Software that is OSI Certified will be refered to as Open Source Software (OSS). OSI is the Open Source Initiative, a non-profit organisation that works to promote Open Source Software.

Software that has stronger restrictions in license will be referred to as proprietary software.

1.5.2 Different standards

In this paragraph there will be a discussion and definition of the different terms that relate to standards, this is taken from the work of Quarterman & Wilhelm 1993:51.
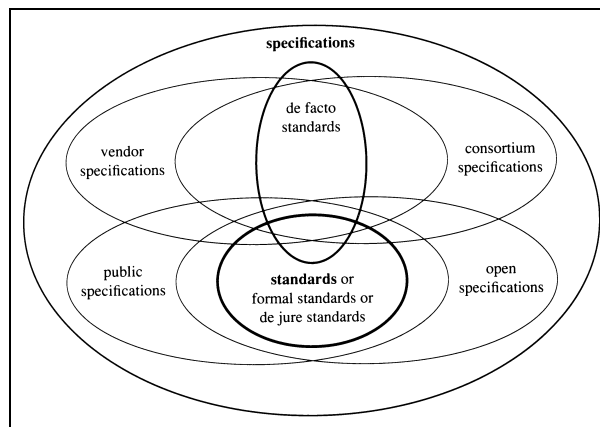
A *standard* is a specification that is produced produced by a formal process by a formal (accredited) standards body. The term *specification* is used for specifications that are not standards, the term *standard* is used for formal standards only. A formal may be called *a de jure standard* to destinguish it from a specification.

A de *facto standard* is a specification that is widely implemented and used. This has nothing to do with how such a document is produced, and either a standard or a specification may be a de facto standard. A formal standard may be a de facto standard.

A specification produced by an open process is an *open specification*. An *open process* is one that controls the future content of the specification, and that permits input by any interested or affected party. Formal standardisation processes are designed to be open processes, and mostly succeed to a large extent in being such. Vendor consortia, such as X/Open, OSF or UI produce other open specifications. There are some contention over weather some such consortium are truly open, since consortia may permit direct input from only member organisations that have paid membership fees. A consortium may be as open as a formal standardising process. An example is the group specifying the TCP/IP protocols, this is neither a consortium nor a formal process.

A *public specification* is a specification that is available to the public but was not originally produced by an open process. The producer of an public specification will usually require control over future versions of the public specification, or at least a frozen form of a specific version of the public specification, before using it in producing a standard or open specification.

An *implementation* conforms to a standard or a specification. Most of the standards and specifications relevant to open systems specify interfaces. Implementations that conform to them may wary in their internal detail as long as they supply the appropriate interfaces. This provides dependability for the application developer.

Picture 1: The relation between standards and specifications. Source: Quarterman & Wilhelm 1993:52.

2 Software licenses

In general, a license is a legal construction that is used when the owner does not wish to surrender full consumption of rights but wishes to sell certain rights of use of the product..

The license is the agreement the user has to accept if he wants to use a piece of software. The license states the terms on which the software may be used i.e. it may or may not be redistributed without permission and so forth. The license is a legally binding document that gives the owner of the software the right to prosecute any violation in a court of law. The license is thus a powerful instrument that states the rules of use for the software

Software comes with many types of licenses Microsoft has one, the Apache project uses another and Netscape Mozilla (the free version of the Netscape browser) a third type of license and many other. While many different licenses exist, it is possible to describe some categories that the majority will fall into.

Commercial software is all types of licenses and software that are used in a commercial sphere. The label commercial software is independent of the license used and thus can be applied to all types of licenses.

This means that the gradient from free (GPL) to commercial makes no sense. It is possible to view the different license forms, as done in the following. The different features must however be made explicit in order to clarify how to understand what is free or non-free software. The term free or non-free may even show to make very little sense. This then becomes a discussion of what features in a license that are important.

The following description of different licenses represents what is commonly perceived as the different license forms in terms of free, non-free software.

2.1.1 Commercial software

This is the Microsoft style license, the EULA (End User License Agreement) that a user has to accept before installation of a new piece of Microsoft software. In this type of license the commercial developer retain all rights for redistribution and modification. The software has to be bought and is usually only available in a binary form. The commercial developer makes profit from the sale and after sale of the software. The software may not be redistributed nor are changes allowed. This type of software very rarely comes with the source code.

2.1.2 Limited trial software

Usually distributed for free or at very low cost, however this is software that is retarded in some way compared to the retail versions. Often this type of software is either distributed in version with limited functionality or with full features and limited time of use. Redistribution is allowed. Examples are software distributed with magazines or on the web that have limited trail period of 30 days. The program will stop functioning after the trail period has expired; this is called a time bomb.

2.1.3 Freeware

The term Freeware is not clear-cut in its definition, but it usually refers to software that may be redistributed without cost. Modification is not allowed and the source code is seldom provided or available. This should not be confused with the term Free Software that uses the GPL license, se paragraph 2.1.11 for a description of the GPL License.

2.1.4 Shareware

Distributed for free this type of license also gives the user the right to redistribute. The license states that the user must buy the software after a certain period of time. There is no time bomb but the user will usually be notified that the evaluation period has expired and that

the user should consider buying the software. The widely used file compression program "WinZip" is one such program.

2.1.5   Non-commercial use
This license allows the user to use the software in a non-commercial environment, i.e. only private use. Usually the license permits redistribution by non-commercial users, making changes is not allowed. If a company wishes to deploy the software, the company has to buy the proper license. The Netscape Browser has been distributed under such licensing terms.

2.1.6   Royalty free binaries
This is software that may be distributed and re-distributed for free but in a binary only form, source code is not distrubuted. It is not allowed to make changes to the software. The Microsoft Internet Explorer is an example of this type of software and license.

2.1.7   Royalty free libraries
A library is a set of programs that other programs can link to examples are mathematical functions like cos, sin and tan. Libraries may be distributed and re-distributed in both binary and source form. No matter the distribution form the user are not allowed to make changes to neither source - nor binary code.

2.1.8   BSD-license license
This is the license used in the BSD-Unix project. The binary - and source code are distributed for free and redistributed are allowed. It is allowed to modify the code and redistribute it as a binary only thus forming a separate work. Creating new software bases on huge parts of another piece of software is called "forking the code". Stricter licensing can be applied to the forked code. There is however a catch, derived versions has to state that the code originated from UC Berkeley, this must also be stated when advertising [Perens 1999:183]. The BSD style license typically does not allow other than selected people to contribute to the code [Valloppillil 1998:59]. The reason for allowing forking of the code can be found in way the BSD project was funded. Initially the BSD-Unix development was funded by government grants, which made the results public property, and thus everybody should be allowed to create a commercial closed source product from this source. The BSD-style licenses are considered to comply with the Open Source Definition as the OSD does not require the derived work to carry same license as the original [Perens 1999:183].

2.1.9   Apache-style license
This is a derivative of the BSD-style license the apache-style license does however allow other than the central development group to contribute to the code. As with the BSD-style license, the Apache-style license is considred to comply with the Open Source Definition.

2.1.10   The Open Source Definition
Used by the OSI (Open Source Initiative) as the minimal requirement for obtaining the OSI certificate. The Open Source Definition requires that binary - and source code are freely redistributable. It is allowed for the code to fork and it is nor required that derived work carries the same license, and thus create a product with a much less permissive license. It is allowed to modify the source code. The Apache-style licenses are considered to comply with

the Open Source Definition Definition as the OSD does not require the derived work to carry same license as the original [Perens 1999:183].

2.1.11 The Gnu Public License (GPL)

The GPL is created by the Free Software Foundation the full version can be found in appendix 1. The GPL is written as a political manifesto as well as a license, many lines in the document is spend explaining the reasons for the license.

The GPL offer another feature that the Open Source Definition does not. It prevents forking of the code by obligating that changes are returned to the author of the software. Also if a work are formed on the basis of GPL'ed software this software too has to carry the GPL license. The smallest amount of GPL'ed code in a piece of software will result in the software being covered by the GPL. This a sort of viral effect; everything that is contaminated by GPL becomes GPL'ed. This prevents people from changing the license to something less permissive, closing the source for the sake of own profit. The source code always has to be available, this also applies to derived work in effect ensuring that changes and improvements are not lost and returned to the community.

2.1.12 Other licenses

The multiple other types license that lie in the range between commercial software and the Gnu Public License. Netscape have created the NPL, Netscape Public License and Mozilla have created the MPL, Mozilla Public License. All these in-betweens are created to give the author the largest degree of control and still convince the public to contribute to the project, often a delicate balance.

3 History of the Unix Evolution

I therefore begin this empirical chapter with the history of Unix told in chronological order. Parts of this story has been taken from Peter H. Salus' book "A Quarter Century of Unix", however Salus' book follows the actors at the expense of the chronological detail. I have structured the events chronological and added materials from other publications, a large amount of these have been available through the Internet.

The emphasis in this chapter will be on the development of Unix and how there came to be free Unix-like alternatives such as the BSD Unix, Berkeley Free Software Distributions and Linux.

3.1 Necessary history of the AT&T

To understand the evolution of Unix, a few words about the early history of the AT&T are required.

In 1949 the Antitrust Division of the US department of Justice under the Truman administration filed a complaint against Western Electric Company, Inc. and the American Telephone and Telegraph Company. The claim was that the two companies were acting together in restraint of trade in violation of the Sherman Antitrust Act.

In 1956 after extensive negotiation Judge Tomas F. Meaney decreed that AT&T and Western Electric was not allowed to engage in manufacture, sale or lease of any equipment other than

telephone or telegraph equipment. There was some exemptions to this decree an important on was that AT&T and Western Electric was allowed to perform experiments for the purpose of testing or developing new common carrier communications services [Salus 1994:56-57]. This exception made the development of the Unix operating system possible and is the reason behind the early license terms.

The structure of the companies was that Western Electric was a wholly subsidiary of AT&T and the Bell Telephone Laboratories was a jointly owned subsidiary of Western Electric and AT&T (50/50) [Salus 1994:57].

In the Bell Telephone Labs there was a conservative attitude, there were no need to provoke further action from the department of Justice. Thus no experimenting with different product and lawsuits were undertaken. It also meant that the collaboration with General Electric and Massachusetts Institute of Technology (MIT) in the MULTICS project (explained in detail in paragraph 3.3) was a research experiment. The internal efforts of the Bell Labs computing research group were experimental and thus the results from research projects could not be sold and profited from. At least until the decree had been removed.

The effect was that AT&T could not sell or profit from the Unix operating system. Universities that were interested in the Unix operating system could have it under lacks terms.

However, in 1982 Judge Green proposed a decree which key features were "the divestiture of the operating companies from the remainder of AT&T" [Salus 1994:190]. Meanig that the operating companies were split up in several minor companies intended to compete with each other. Following the decree it was now legal for AT&T to engage in other activities than telephone or telegraph. AT&T welcomed this new profit opportunity with new licenses and higher prices for the Unix operating system.

3.2   Unix is a trademark

In all most every document about Unix there is a remark stating that "Unix is a registered trademark of Unix System Laboratories" or before 1983, "Unix is a registered trademark of AT&T Bell Laboratories" [Quarterman and Wilhelm 1993:40].

The word *Unix* is trademarked, the source code for the Unix operating system is licensed, copyrighted and a trade secret – all three at once. The license permits someone other than Unix System Laboratories (USL) to use the source code or the binary Unix. A license implies control of modifications to both the implementation and the interface [Quarterman and Wilhelm 1993:40]. By defining the code as a trade secret, it is possible to show/give/license the code to specific parties without violating the right to later patent the same code.

Many different vendors have based their operating systems on Unix, and some still are. Due to the trademarks and various licenses this has to be done with care. Those who wanted to make their own Unix used the source code from USL and adopted it to fit their own proprietary hardware. Sometimes they added their own proprietary software. Examples of this are Ultrix (from DEC), HP-UX (from HP), SunOS (from Sun) and AIX (from IBM) [Quarterman and Wilhelm 1993:41].

Such vendors pay a license fee to USL for the use of the source code and for each copy of the system sold. USL retains the complete control of the resulting software packages the vendors create and the product is not a Unix operating system unless USL agrees to let the vendor call it Unix.

There are other systems with the same programming interface as Unix that is the interface the person using the system uses are identical on these systems. The commands and their use are identical so the different systems that use same programming interface look and feel identical - like Unix. Their internal structure may vary i.e. the source code is different but the interface are the same. The source code to these Unix'es is different from the USL Unix and there is not a line of the original code in [Quarterman and Wilhelm 1993:41]. Examples of such operating systems are Linux, 4.4 BSD, GNU Hurd, OSF/1, Free BSD and NET BSD.

3.3 1967

Ken Thompson joined the MULTICS (Multiplexed Information and Computing System), a joint research project between General Electric (GE), Bell Telephone Labs (BTL) and Massachusetts Institute of Technology (MIT). The project tried to create a multi user operating system, which could accommodate a thousand users at the same time. [Salus 1994:5]. This would require a highly reliable computer with a matching operating system capable of continuously running 7 days a week and 24 hours a day. [Corbató & Vyssotsky 1965:1] In those days the reliability alone was all most impossible to archive. A large amount of money and time was spent on the project, which did not produce the required results, and in the beginning of 1969 BTL withdrew from the project [Salus 1994:8; Mahoney 1998:1]. The MULTICS project did however produce a lot of good ideas, of which some later where used in the development of Unix.

Picture 2: An orbit from the Space Travle game [Salus 1994:133].

3.4 1969

Ken Thompson had written a game called Space Travel [Salus 1994:9] for the MULTICS machine, actually the game was an astronomical simulations program, not merely a game. However, without the MULITCS project there was no computer so Ken Thompson started to look for an alternative computer. First Space Travel was ported to a GECOS computer but that was an unsatisfactory solution since the display was jerky and a single game cost about 75$ in CPU time [Ritchie 1984:2]. It did not take long time for Ken Thompson to find a little used PDP-7 computer, a Digital Equipment Corporation (referred to as DEC) computer, with an excellent graphic processor which could be used for Space Travel.

Picture 3: A successor to the PDP-4, the PDP-7 used smaller, more conventional system units and was well received in laboratory and data acquisition applications. The machine featured DEC's first mass-storage based operating system (DECsys for DECtape). Ultimately, 120 PDP-7s were produced and sold. Source: http://www.digital.com/timeline/1964-3.htm.

Ken Thompson and Dennis Ritchie began to rewrite Space Travel to fit the PDP-7 computer, this proved to be a greater task than expected. The cast-off PDP-7 with 340 display only provided an assembler and a paper tape loader. (A paper tape loader is used to load programs from paper tape to the computer, this was in the old days where hard- and floppy drives were very expensive and in some cases not available.) This meant that Ken Thompson and Dennis Ritchie could not go directly to the PDP-7 and start to program. They had to write an operating system that could handle the Space Travel game – loading the files and controlling the cursor on the monitor. Having only an assembler meant there was no operating system that could load the files and no tools for which one could make programs. The PDP-7 could load and execute a program from tape and no more.

Ken Thompson and Dennis Ritchie used a GECOS computer where they wrote the assembler for the PDP-7 and then carried the paper tape from the GECOS to the PDP-7 [Salus 1994:9; Ritchie 1984:2]. One must understand that it was a very complicated task to write the program in assembler. Soon after Space Travel ran on the PDP-7, there was a rudimentary operating system and Ken Thompson and Dennis Ritchie began to further develop this emerging operating system.

In the summer of '69 Ken Thompson spend a month rewriting a shell, an assembler, an editor and the operating system. By the end of the summer the PDP-7 was all most self-supporting, e.g. it was now possible to write programs directly on the PDP-7 using the shell and editor and execute the programs.

3.5 1970

Dennis Ritchie, Ken Thompson and Joe Ossanna had tried several times to convince BTL to purchase a computer for the research group, but so far without any luck. In those days (1969-1970) buying a computer meant spending well over $100.000 [Salus 1994 94:33]. So the research group still had to use the borrowed PDP-7 for their experiments.

By the beginning of 1970 Unix was now showing a very promising file system, and some interesting tools to use with the file system it was thus becoming a growing concern for the research group. Primitive by today's standard, it was capable of providing a more congenial programming environment than its alternatives [Ritchie 1984:8]. The Unix project had shown

more promise [Ritchie 1996:6] than the MULTICS project and this had happened on a smaller computer and with little or no funding.

In early 1970 DEC introduced the PDP-11 at the cost of "only" $65.000. The application for the PDP-11 differed from the earlier besides the lower price, that it described text processing as an area of research [Ritchie 1984:8].

The PDP-11had 24K bytes of core memory, 16K for the operating system and 8K for user programs. It also had a disk with 1K blocks, each block had a size of 512K bytes, the files were limited to 64K bytes.
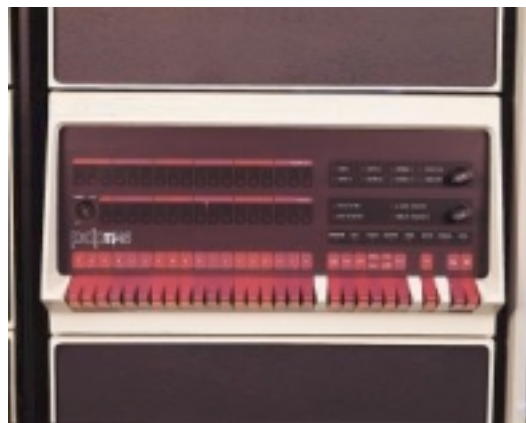
The money was granted and the PDP-11 arrived in the summer of 1970. The PDP-11 was so new a product that no disc was available until December 1970. The mean time was spent writing a rudimentary core-only version of Unix on a cross-assembler on the PDP-7 [Ritchie 1984:8]. This was done by transporting paper tape with the PDP-11 assembler from the PDP-7 to the PDP-11. Having no disc the file system was present only in the memory of the PDP-11. Once the disc arrived the system was quickly completed. The first PDP-11 Unix represented little advance from the PDP-7, mostly it was a matter of translation.

Having promised a word processor the research group began to translate the *roff* text formatter from the PDP-7 to the PDP-11. In early summer 1970, editor and formatter in hand the research group felt prepared to fulfil the word processor charter by offering to supply a text-processing service to the Patent department [Ritchie 1984:8]. The *roff* was adapted to the needs of the Patent department, numbered lines, mathematical symbols etc.

3.6 1971

During the late half of 1971 the research group was supporting three typist from the Patent department, who spent the day busily typing, editing and formatting patent applications, and meanwhile tried to carry its own work. Unix had gained a reputation of supplying interesting services on modest hardware requirements [Ritchie 1984:8].

Dennis Richie quoted in Salus 1994:36 : "We knew there was a scam going on – we'd promised a word processing system, not an operating system". But the text processing effort was a success: The Patent Department of BTL became the first Unix user sharing the PDP-11 with the research group. Even more important: Bell's Patent Department subsequently took over the PDP-11 running Unix and turned funds over to the Research Group, with which a PDP-11/45, see Picture 4, was purchased.

Picture 4:The PDP-11/45 was an excellent computational tool for large multi-user, multi-task installations. Through memory management, memory could be expanded to 128K, which included a combination of bi-polar and MOS memory. Other features included a greatly expanded floating point processor. Source: http://www.digital.com/timeline/1971-3.htm.

With the advent of the PDP-11/45, the system began to grow. It put on weight in terms of instructions, subroutines and games [Salus 1994:38]. It was about to become the first version of Unix.

The first version came about as a demand for a programmer's manual. Doug McIlroy, head of research group, bent arms to get the manual, he saw the manual as a way of maintaining the coherence and integrity of the Unix operating system. Doug McIlroy said about the manual [Salus 1994:39-40]:

> Cleaning something up so you can talk about it is really quite typical of Unix. Every time another edition of the manual would be made, there would be a flurry of activity and, when you down the uglies, you'd say, "We can't put this in print and you'd take features out and put features in order to make them easier to talk about."

The first version of the "Unix programmers manual" is dated November 3, 1971. It was a comprehensive manual describing every command, subroutine, system call, special file, file format and user maintained programs. Fifteen years later Berkeley's version of Unix (4.3 BSD) were arranged the same way [Salus 1994:43].

There is an interesting point to the first Unix manual, the introduction contains a list of the owners of the different programs. The tradition of giving credit where credit is due in the Unix community begins here. When a user is logged in to a Unix system the file being used e.g. a text document or the source code of a program show who has been manipulating the file. Later this feature has been removed in some larger corporations such as IBM, DEC and HP but still exists in many parts of the Unix community.

3.7 1972

The second version of Unix, V2, appeared in June 1972 the preface had an important note: "The number of Unix installations has grown to 10". Impressive when taken into account that the system was unsupported by BTL, it was however free and the source code was provided. The research group continued their work on Unix.

One of the Unix installations was found at New York Telephone. They got a DEC PDP-11/20 which was intended to control three multiplexers. The 1972 version Unix system where very rough and it took some effort to make the system usable. Several people could work on the system at the same time and every time someone was to run an untested program they would shout "Dangerous Program!" and all people working would save their work. In those days there was no such thing as a printer queue so people would shout "Line printer" this was done on a first to shout, first to serve basis [Salus 1994:44].

In general 1972 was a very creative period of time and many changes happened. Most of the C language was defined in that period [Ritchie 1996:9].

3.8 1973

February 1973 the third edition of Unix, V3, was finished. There had been a lot of improvements, Unix was the essence of flux and the rate of change so large that the first parts of the manual had to be rewritten at the same time the last part was being written [Salus 1994:40].

Early this year the essential parts of modern C where complete, the language and compiler was strong enough to permit a complete rewriting of Unix in the C language [Ritchie 1996:9].

November this year the fourth edition, V4, was released [Salus 1994:43]. As a consequence of the rewriting of Unix to fit the different available computers Ken Thompson and Dennis Ritchie rewrote the Unix kernel in the C programming language for the PDP-11 [Ritchie 1996:9]. The idea was to make Unix easily portable to other computers. C was a high level language in the sense that it was not written to accommodate any specific hardware. The C compiler took care of any hardware specifics and the compilers relied on a set of libraries and the underlying assembler.

Writing in C made the source code easy to read and modify. C was a spin off from the Unix development in the sense that it was created to provide a pleasant working environment in which to make programs. C eventually became a separate research project which lead to the standardisation by ANSI in the late 1980'ies, for a more exhaustive reference on the C development history see "The Development of the C language" by Dennis Ritchie.

The rewriting of Unix in C is probably one of the most significant events in the evolutionary process of Unix. It meant that Unix could be ported to any hardware in a matter of months [Lewis 1994].

3.9 1974

May this year the V5 Unix was released [Salus 1994:43]. Unix source code was distributed freely to universities (due, at least in part, to the US trade restrictions on ATT). As a result, Unix gained favour within the academic/research community and formed the basis for Operating Systems instructions in the leading universities.

US government filed a separate antitrust suit against AT&T, Western Electric and Bell Labs. This complaint "alleged monopolisation" and sought divestiture from AT&T of the Bell Operating Companies and the dissolution of Western Electric [Salus 1994:189].

The First meeting for Unix users was arranged by Lou Katz at the Columbia University. Columbia University had been a recipient of the first distribution. The meeting was held May 15 [Salus 1994:65]. The program for the day was interesting:

Unix users meeting agenda may 15, 1974

10:30	Start
	Description of the several installations and their use of Unix
12:00	Lunch
1:00	Ken Thompson speaks
2:00	Interchange of Unix hints, problems solutions, bugs
3:00	Interchange of DEC hints, problems solutions, bugs
4:00	Free for all discussion

About two dozen people from a dozen institutions appeared. One must know that there had yet to be published any article about Unix.

Two months later Ken Thompson and Dennis Ritchie wrote an article in CACM, this was very well received.

3.10 1975

V6 was the first version of Unix that was widely available outside Bell Labs especially in universities. The V6 was distributed with the source code, as was most operating systems at that time. This was due to numerous bugs and problems with different types of computer configuration. Just to get the system up and running there often had to be done some editing of the source code.

There are some dispute about the release date of V6, Quarterman & Wilhelm 1993:32 claims that V6 was released in 1976, Salus 1994:43 quotes the Unix Programmers Manual (The documentation distributed with Unix), states the release date was May 1975. Quotes from the Programmers Manual seem to have the greater weight and will be considered the fact, the difference is however small and will not have any bearing on this thesis.

The first edition of "UNIX NEWS" (Number 1) was published July 30. This edition was 11 pages, 37 institutions subscribed, most of which was universities. In appendix 1 the list of subscribers can be found.

Second edition of "UNIX NEWS" was published that same year the number of subscribers had risen to 60 indicating the growth of the Unix community.

3.11 1977

The Computer Systems Research Group (CSRG) at the University of California at Berkeley (UCB) released the first version of the Berkeley Software Distribution, 1BSD. This distribution was meant for use with Unix V6 on a PDP-11 computer. This was not a complete operating system, but mostly application programs. One of the programs was **vi**, the first full screen text editor [Quarterman & Wilhelm 1993:33] the other was a Unix Pascal System (programming language) [Salus 1994:143]. About 30 copies were distributed of the 1BSD.

In May-June the last edition of "UNIX NEWS" was published. AT&T owned the Unix trademark and would not allow it to be used by other "UNIX NEWS" therefore changed name to "USENIX" which was not a trademark. An inspired way of circumventing the restriction imposed by the UNIX trademark.

3.12 1978

2BSD was released **vi** and the pascal system had become more robust. Also **termcap** was born and put on the 2BSD. About 75 tapes were shipped [Salus 1994:143]. The 2BSD were written for Unix V6 and/or V7 on PDP 11 computers. There were new software added and modifications made from time to time including Mail, Pascal and commands like **more**, **csh** and **ex**. The 2BSD were shipped with different licenses depending on the buyer's Unix license [Salus 1994:159].

3.13 1979

The 7th version of Unix was very influential on several levels. Unix V7 was released from Bell Labs in June 1979 [Salus 1994:146] and was the last version distributed by the Bell Labs Computer Systems Research Group. For many, this is called the "last true Unix" [Lewis 1994:1]. This version was developed to be portable to various hardware architectures [A1.51]. This version was vastly improved in both reliability, file system size, maximum number of user accounts and there was a lot of new commands that would ease the administration. The V7 Unix was released with a full C compiler and all most 1200 pages of

documentation and a more sophisticated shell. There where however a major drawback, the system had worse performance than the V6 Unix, so the users had to improve the system.

Berkeley personal changed the size of the data blocks of the file system, this was later in April 1980; later again this was ported by someone at Unisoft to the PDP-11/70. Others from Berkeley in December 1979 moved the buffers out of kernel address space, and changed the **stdio** library on the DEC VAX computer. One at UC San Francisco later ported these changes. In University of New South Wales they had begun to create a new procedure for directory path names. There were contributions from various sources both commercial, private, universities and the Bell Labs them selves, the users had improved the performance tremendously [Salus 1994:146-148]. All these changes and improvements where incorporated into releases of both BSD and AT&T Bell Labs Unix.

Until Unix V7 schools and universities had used Unix extensively in different courses on programming of operating systems design. But with the release of Unix V7 AT&T announced its intention to commercialise Unix so the license prohibited the use of Unix V7 source code from being studied in classes, in order to keep its status as a trade secret [Tannenbaum in Salus 1994:152].

The licenses prompted the University of California at Berkeley to create its own variant: BSD Unix, they had of course at this time released the 1BSD and 2BSD but these were not complete Unix operating systems just applications (see paragraph 3.11 and 3.12).

The new license resulted in the release of 3BSD, the 3BSD was based on a pre-release of Unix/32V that was modified to fit the DEC VAX machine, many things such as demand paging of main memory were added [Salus 1994:157; Quarterman & Wilhelm 1993:33]. The 3BSD wad not just utilities and applications it was a complete Unix operating system including all the software from 2BSD.

This year a USENIX conference was held that had special influence on the community of user groups. From the confrence:

> "some guy stood up and they boooed him off the stage because he was a marketing consultant or something" [Salus 1994:193].

This was typical for the USENIX community, representing the users, not the commercial interests. There had all ready been some complaining about this from different people and some sort of reaction was forming. The reaction was the creation of the /usr/group user group in 1980.

3.14 1980

/usr/group was formed and represented the commercial interests in Unix. This split up from USENIX represents sort of a split between the growing number of binary users and the source hungering scientific community. It was the /usr/group that initiated the standards effort of the Unix operating system. In 1984 /usr/group had produced a standard [Salus 1994:193]. The commercial Unix developers have no desire to publish the source code since this would expose their work and allow everyone to learn from it. There was though a desire to produce standards that would allow applications to run on different Unix systems.

3.15 1983

AT&T released the commercial Unix System V. BSD version 4.2 was released, this was probably the most influential BSD release [A1.51; Quarterman & Wilhelm 1993:33]. Unix

System V was a different strand of Unix, compared to vanilla Vx Unix, it was however developed by AT&T.

There were different research Unix'es in use inside AT&T, these included PWB, MERT (not really a Unix system), CB Unix and Unix/RT [Quarterman & Wilhelm 1993:35]. System V was developed from system III, which managed most internal systems at AT&T in 1982.

As a result of the 1974 antitrust case against AT&T (see 3.9 and 3.9), Judge Green proposed a decree that split the operating companies from the rest of AT&T. The result was that Western Electric was dissolved and the various operating companies formed the "Baby Bells". The spilt up permitted AT&T to enter the computer hardware and software businesses [Salus 1994:190; Quarterman & Wilhelm 1993:34]. This possibility to capitalise from Unix was met with the release of Unix system V [Salus 1994:190]. Unix system V was developed by Unix Systems Lab (USL) at AT&T.

3.16 1984

/usr/group proposed a standard that was adopted by the membership this standard was soon to become inadequate due to divergent developments of SystemV, 4.xBSD and Xenix [Salus 1993:203].

3.17 1985

Unix V8 released [Salus 1994:40]

The /usr/group committee merged with the newly formed IEEE POSIX working group (Institute of Electrical and Electronics Engineers, Portable Operating Systems), the /usr/group's standard was adopted as a first draft [Salus 1994:203].

3.18 1986

Unix V9 released [Salus 1994:40].

3.19 1987

AT&T realeased Unix System V r3. This is the release that various major hardware vendors such as HP (HP-UX) and IBM (AIX), most notably, finally felt business pressure forcing them to have a "Unix" OS for their hardware, based their Unix version on. BSD v 4.3 was released.

Late this year AT&T purchased a sizeable percentage of Sun Microsystems, along with this announcement AT&T also announced that Sun would get preferential treatment in the development of Unix and software from AT&T Unix Systems Labs (USL). In the same breath Sun announced that its next operating system would not be an extension of SunOS (originally derived from Berkeley Unix) but derived from SystemV release4 [Salus 1994:217]. This of course sent shivers down the spine of good deal of the Unix world. The scientific community felt that Sun turned their back on them. Manufactures of hardware feared that Sun now would gain a superior competitive advantage. Armando Stettner (of DEC) quoted in Salus 1994:216:

> "Sun was everyone's most aggressive competitor. We saw Sun's systems were a direct replacement for the VAX. Just think: the alliance combined our most

aggressive and innovation competitor with the sole source of system software – the balance had shifted"

3.20 1988

The result of the alliance between AT&T/USL and Sun was a meeting at DEC's headquarters between Apollo Computer Inc., DEC, Gould Electronics, Honeywell-Bull, InfoCorp, HP, MIPS, NCR, Silicon Graphics, UniSoft and Unisys. These were referred to as the Hamilton Group, DEC's headquarters are at 100 Hamilton Avenue.

This meeting produced a telegram that was sent to James E. Olson, CEO of AT&T:

"As licensees of AT&T software and supporters of an open Unix standard, we are concerned about recent announcements between AT&T and SUN Microsystems. These announcements have created concern within our companies and customers regarding the future of Unix as an open standard. We feel it is important to get a better understanding of these issues before the upcoming "Uniforum" conference and we request a meeting between our corporate manager responsible for Unix strategy and mr. Cassoni during the week of January 25 1998….." [Salus 1994:217]

Vittorio Cassoni was the senior vice president of AT&T's Data Systems Division. The meeting had no positive result where the group was concerned

The group invited IBM to join and after a series of semi-secret meetings the Open Software Foundation was announced who's purpose was to produce an AT&T code free Unix operating system [Salus 1994:217].

As a counter strike Sun and AT&T formed a counter consortium: Unix International, that was dedicated to the marketing of SystemV.

By the end of this year OSF had produced the standard for a user interface MOTIF, this was well received.

3.21 1990

ATT issued System V release 4 as a new standard unifying Unix variants. This was the result of Sun and ATT's co-operation. However, other vendors (especially DEC, HP, IBM) felt threatened by this collaboration between the two most important Unix developers and united to create OSF - the Open Software Foundation. Intriguingly, some people have suggested that the name OSF were chosen because it could also mean "Oppose Sun Forever" [A1.51], this has, however, been refuted by Amando Stettner who suggested the OSF organisation [Email from Amando Stettener] . System V relese4 was very influential and merged many features from POSIX, BSD, SunOS. All most all vendors of Unix now support this programming interface. These interfaces are specified in the system V interface definitions SVID [Quarterman & Wilhelm 1993:34].

3.22 1991

OSF-1 released. DEC is the only major vendor to adopt OSF; however, other vendors such as IBM have adopted parts. This was also the year that freely distributable Unix clones such as Linux and FreeBSD started [A1.51]. Linus Thorvalds released the first version of Linux and posts a call for help to develop the operating further on the news group comp.os.minix.

The economy worsened, Bull, IBM DEC, and Siemens were all loosing money.

AT&T sold their share of SUN.

3.23 1992

Sun develops its Solaris OS: a System V release 4 derivative with support for symmetric multiprocessing (i.e. multiple CPUs) [A1.51].

3.24 1993

Unix International went out of business and OSF had abandoned several of its promised products. Hope had vanished and nobody any longer talked of an AT&T license free OSF Unix [Salus 1994:218].

3.25 1994

Client/Server computing and the Internet have become buzzwords. In particular, delivery of information to clients via the Internet has recently become popular. Such delivery is highly interconnected with other system administration tasks [A1.51].

3.26 1995

Linux, a Unix clone written by Linus Torvalds was now being actively developed by a growing Internet-based community of hackers. Returning Unix to its roots, the source code was/is freely available. Although originally developed only for Intel hardware, ports are actively underway to Alpha, Sparc and MIPS hardware.

3.27 Up till now, 1999

The past years have been interesting, Unix who was considered the king of advanced servers have been met with fierce competition. On two fronts the competition has been - and still is raging, easy to use server software represented by Microsoft Windows NT. And on the other side the advanced server who boasts stability, security and configurability: Linux, FreeBSD, BSD Unix and other Unix-like operating systems. Not much systematic work has been done to document this battle of operating systems.

There is however a trend, Linux is the only Unix-like operating system that is gaining market shares, all other Unix'es has either lost ground or stagnated. Windows NT is still expanding market share, but the rate of increase has been dropping.

There is other interesting trends in the market place. Since early 1999 Linux have had a media boom, every computer magazine has now got a Linux section. This publicity of Linux has increased its momentum and the Linux-awareness in the computer industry. Many of the prominent computer manufactures has this year announced and some released versions of their software for Linux. IBM now supports Linux on their Netfinity enterprise servers, that is full 24/7 support.

Digital's high end processor; the Alpha processor, is not going to be supported by MS Windows 2000 as it is now supported by Windows NT. Market analysts say that this will not pose a treat to the Alpha since most runs other operating systems than Windows NT [a5.50 a5.33]. Note: Compaq is now developing the Alpha.

In this fall of 1999 Silicon Graphics Inc. (SGI) maker of the IRIX operating system, a Unix like operating system, announced that SGI would contribute their XFS high performance file system to Linux. The contribution would be in form of the source code for the XFS file

system licensed as GPL [A5.14]. The XFS file system allows for exceptional large hard drive storage and data high transfer rates between hard drive and computer.

3.28 Summing up the Unix development history

In the period from 1969 to 1982 where the split up allowed AT&T to profit from Unix, it evolved quickly and became a coherent well performing operating system. When AT&T began to charge big money for licenses there was a fierce reaction from the user community, centred at Berkeley, the users would not accept the terms. As reaction the Berkeley Software Free Distributions came about: The BSD Unix, that is 3BSD, the third version was the first to deliver a complete functioning operating system.

Some conclusion to be offered from the chronology:

1. The speed of the Unix evolution happened only because many people contributed to the development.

2. The quality i.e. stability of the Unix operating system was a consequence of the availability of the source code that allowed people to fix bugs.

3. People got involved because Unix could be obtained a very little cost in the beginning.

4. The culture of sharing software and tips was a factor that aided to the rapid development of Unix.

5. The diffusion of Unix to the universities meant that young talent got to know Unix and that efforts where made to further develop Unix.

6. Unix got fragmented because licensing allowed companies to develop Unix without sharing the changes and new standards.

7. The users had much influence on the design an evolution of the Unix operating system.

8. Infrastructure may have had an impact. In the early days few people had access to computers and less had access to means of communication between computers i.e. modem or network.

3.29 The value of the source code

Several times in the preceding paragraphs it has been noted that the early versions of Unix came with the source code. The source code refers to the human understandable sequence of instructions that make up a computer program. The source code is the blueprint of a program, it shows statement for statement what a program does and how.

The source code makes it possible to understand the functioning and malfunctioning of a program and thus elaborate and/or fix bugs in the program. Today it is not a common thing to receive the source code for a program; instead the buyer receives the binary program, that is the compiled version of the program. The binary version can be used, but not easily modified. Modifying the binary version is more like reverse engineering, a very difficult task.

Today the average user has no need for the source code most software needs can be satisfied by off the shelf software packages like the Microsoft Office suite that satisfies all most every need for office produktivity. The same goes for the Windows operating system most people are content with the use and have no desire to modify the source code from Microsoft.

In the old days of Unix not many applications existed and those that did were crude and often developed for a small specific purpose. The Unix motto were: Write programs that do one thing and do them well. However, instead of starting from scratch programmers extended the

functionality of existing programs to fit new needs, provided there were reusable parts. For this the source code was crucial. Without the source code the programmer was doomed to start from scratch. Sometimes the source code carried a license that did not permitted that the code could be copied. But the availability of the source code provided valuable insight into the design of the specific type of program and the programmer could start with a reference and some good ideas as to how to create the program.

Thus, the source code is invaluable to the user who wishes to alter a program. The source code provides the possibility to modify and improve a program to fit the needs of the user. The availability of software and support, or rather the lack of, in the first many years of Unix meant that users and system administrators had to fix themselves the errors they encountered. Often some of the hardware on the specific location was not supported in the initial release and the administrator had to make the changes himself. New types of hardware continued to be developed and the administrator had to create for himself the modifications that allowed new hardware to function – the source code was invaluable.

Regarding the mainstream user this value has diminished over time as the developers of hardware and operating systems have upped their efforts to support the latest hardware. This goes particularly for the MS Windows operating system that supports all most every piece of hardware no matter how obscure. The Unix community to this date still suffers from marginal support compared to MS Windows [A7.20]. The source code thus still holds considerable value in the Unix community the same goes for the Linux community.

3.30 POSIX

POSIX is a standard defined by The Open Group - a consortium of hardware and software companies as well as private organisations. POSIX (Portable Operating System Interfaces X (The X denotes the relation to Unix)) describes how to program operating systems so that they easily can be ported to different hardware. The effect is that one can write the source code for an operating system in C and the compile on different compilers designed to different platforms [Simonsen 1999].
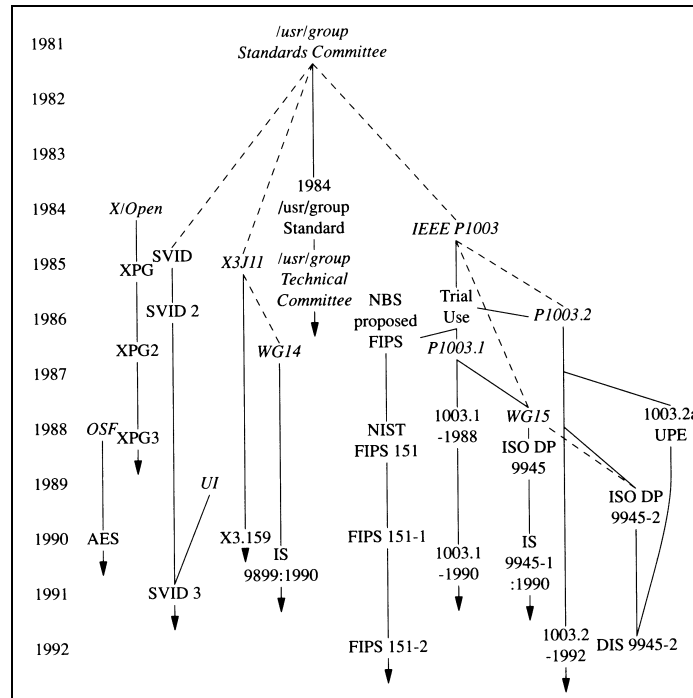
It is important to realise that POSIX is not Unix it is an interface not an implementation. POSIX has been referred to as "a generalisation of Unix".

3.30.1 Brief history of POSIX

From the /usr/group 1984 standard (really a specification) to the approved International Standard 9945-1 (POSIX 1003.1 Operating System Interface), standards related to Unix went from idea to reality in four to six years.

Picture 5shows the standards development tree that gives an indication how some of the standards and specifications have grown from each other. Many other groups and standards exist, but this gives an indication of the many different groups and standards [Quarterman & Wilhelm 1993:57].

The standards shown in Picture 5 are the 1984 /usr/group standard and its descendants, plus the three POSIX standards, plus the three major consortium specifications. All of the POSIX1003, X3J11 (the C language standard), and AT&T used the 1984 /usr/group standard as a base document. All of these documents and X/Open's XPG influenced each other. OSF's AES came too recently to affect other documents directly, but OSF members had participants in all the relevant standards committees [Quarterman & Wilhelm 1993:57].

Picture 5: The evolution of standards and how the different standards affected each other. Source: Quarterman & Wilhelm 1993:58.

Most of the standards that defined Unix were, and still are, produced as voluntary consensus standards. One reference describes voluntary standardisation as "the deliberate acceptance by a group of people having common interests or background of a quantifiable metric that influences their behaviour and activities by permitting a common interchange " [Cargill 1989:13 in Quarterman & Wilhelm 1993:58]. Some of these standards may later be required by law for certain purposes, and then would become mandatory or regulatory standards. It is, however, important to understand that the organisations that produce standards almost never require that anyone to use them.

3.30.2 The POSIX Motivation

The motivation for formalising and standardising the Unix application Programming Interface, API, can be divided into three categories:

1. Application portability
2. Interoperability
3. User portability

Application portability in general means that an application can be used on different types of hardware i.e. "ported" to other hardware. In order to archive this, the application should comply with a standard interface and the hardware should support the same standard interface. An application that is written for portability is cost-effective since it will not have to be rewritten to conform to a different standard interface in order to port it to other hardware. The two kinds of portability that are most common are source code portability and binary portability, other kinds of portability exists, a description can be found in Quarterman & Wilhelm 1993:53-57. Source code portability exists when the source code for an application is written for a standard interface defining source code. It is then possible to compile the source code on different systems and run the application on these systems. Binary portability exists when the application is written in such a way that the binary

Last printed 16-07-01 09:29

program can be run on different computers without modification. Binary portability is desirable because it allows the user to use the application on different types of computers without having to compile or modify.

Interoperability between different computers means that information can travel between computers, in other words information portability exists. Information portability makes it possible to share data. This can be done in many ways, one is to carry a disk with the data from one computer to another. This is information portability not interoperability since extensive user intervention is required (placing the data on the disk and carrying the disk). Interoperability exists only when data can travel between computers in a way that is transparent for the user, i.e. it makes no difference to the user whether the required data is stored locally or on another computer across a network. A standard protocol ensures that computers on the network can communicate and that the applications also are able to communicate.

A standard user interface enables user portability. Once users have learned the standard interface they can easily user the different types of hardware that supports the user interface [Quarterman & Wilhelm 1993:56]. Since the writing of the book by Quarterman & Wilhelm the costs of hardware have fallen dramatically, computers are now both cheaper and more powerfull at the same time. The cost of educating a user to use a new or different user interfase is large compared to the cost of software. A user would have to go through a course on the new user interface and one must expect a significat lower productivity in a period of time following the transition. Thus the desire for user portability is even greater today and given the tendency for further increasing wages and still more cheap hardware user portability must be expected to gain further importance in the future.

3.31 Berkeley Unix

In February 1973 Ken Thompson and Dennis Ritchie submitted a paper for the ACM Symposium on Operating Systems Principles, which was accepted. The presentation was given by Ken Thompson was also very well received by the audience. A revised version of the paper was later released in Communications of the ACM, that was 1974, this paper was also well received and got labelled "Elegant" by the editor. [Salus 1994:54]

Attending the symposium was Professor Robert Fabry of the University of California at Berkeley. Professor Fabry was impressed with what he had seen and decided to try at put together a joint purchase by Computer Science, Statistics and Mathematics of a DEC PDP 11/45 computer. The joint purchase was a success and soon after Professor Fabry ordered a tape with Unix from Ken Thompson. The installation of Unix on the new computer proved to be more difficult that expected so Ken Thompson had to help. This was done by phone Ken Thompson in one end and a computer operator at Berkeley in the other, along with this a 300 bps acoustic modem was used to transmit from the Berkeley computer to Ken Thompsons computer. Ken Thompson would then remotely debug the crash dumps from Berkeley [Salus 1994:137].

Since the PDP11/45 was an even split between three departments there were different operating system requirements, two of the departments wanted to run DEC's RSTS operating system. This meant that 8 hours a day the computer would run Unix and the remaining 16 hours it would run RSTS.

The Berkeley Unix is significat in the sense that the first non-AT&T Unix came from Berkeley, this was the 3BSD. As noted earlier the CSRG at Berkeley had allready released

the 1 and 2BSD at the time the 3BSD were released. The first two realeases were just software for the Unix opereting system.

The 3BSD marked a turning point from the complete dominance at the operating systems market by AT&T the the release of free operating system.
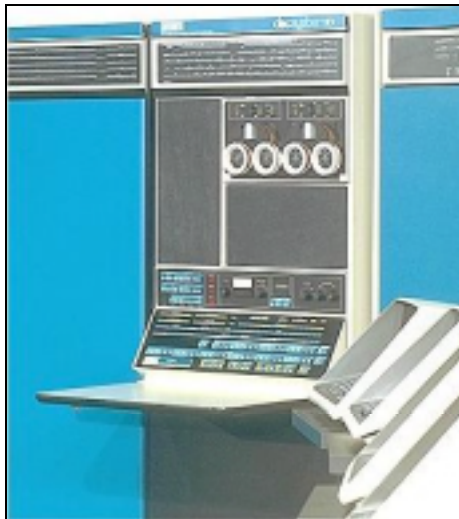
3.32 The Free Software Foundation

This following chapter is derived mainly from Stallman 1999.

Richard M. Stallman, in the following referred to as RMS, started working at MIT Artificial Intelligence Lab in 1971, in those days sharing of software was the rule, not the exception. It was more or less comparable to sharing recipes for cooking. The AI Lab used the ITS (Incompatible a Timesharing System) an operating system written by the lab itself. ITS was running on a DEC PDP-10 see Picture 6.

The scientific environment at universities and industrial research labs, were a large part of the market in those times. As a consequence of the academic habit of sharing results, computer programs were also shared. This probably comes from the non-commercial interests of Universities, which are mostly paid by government and not expected to earn money on software. The wages for the programmers/researchers were already paid, and there was no need to generate income on sale of software [Tanks to Keld J. Simonsen of DKUUG, Danish Unix User Group for this note].

Furthermore it was the custom of computer manufacturers to distribute their hardware bundled with software and source, so the local site could maintain the system. There were a lot of bugs in the software then, and the computers were extremely expensive, so there was often a large staff to support the machines. Thus expertise was available for the source maintenance on-site, and every improvement were very valuable, given the high price of the systems.

Picture 6: The 36-bit PDP-10 was program-compatible with the PDP-6 and approximately twice as powerful. Designed to perform conversational timesharing, batch-processing and real-time operations equally well and simultaneously, the PDP-10 achieved great popularity with the commercial timesharing utilities, university computer centers and research laboratories. Source: http://www.digital.com/timeline/1967-1.htm.

In the early 1980s the MIT AI Lab community collapsed, mostly due to two reasons 1) In 1981 a spin off company called Symbolics had hired close to all the hackers at the AI Lab 2) DEC discontinued the development and production of the PDP-10. This meant that the ITS

operating system got obsolete with one swift stroke. This was because the ITS was programmed in assembler for the PDP-10 and could not easily be ported to another computer [Stallman 1999:53]. The depopulated community at the AI Lab vas unable to maintain it self and the ITS operating system and hence when a new computer was bought, the administration decided to use DECs non-free operating system instead of ITS.

RMS was offended by the licensing agreement that had to be signed. In his own words: "This meant that the first step in using a computer was to promise not to help your neighbour. A Cooperating community was forbidden. The rule made by the owners of proprietary software was. "If you share with your neighbour, you are a pirate. If you want changes, beg us to make them" [Stallman 1999:54].

Faced with no community and no way to continue as before RMS had to make A Stark Moral Choice [Stallman 1999:55]. He could either join the proprietary software world feeling that he would betray he's fellow hacker. Or he could write programs for the common good. RMS decided the latter to make the software sharing community possible again, starting off with the basic thing a computer community need – an operating system.

Accustomed to the Unix operating system RMS chose to make his new operating system compatible with Unix so that it would be portable and easy for users to switch to. [Stallman 1999:56]. In those days an operating system meant the kernel but also tools, debuggers, text editor all the stuff needed to use a computer and develop software. The name GNU was chosen for the operating system, GNU is a recursive acronym for GNU's not Unix.

3.32.1 Free as in freedom

RMS quit his job and founded the Free Software Foundation in 1984. Free was described a free as in freedom, it has nothing to do with price [Stallman 1999:56]. It's all about freedom, therefore, is the definition of free software. A program is free software, for you, a particular user, if:

- You have the freedom to run the program for any purpose.

- You have the freedom to modify the program to suit your needs.

- You have the freedom redistribute copies either gratis or for a fee.

- You have the freedom distribute modified versions of the program, so that the community can benefit from your improvements.

    (From Stallman 1999:56.)

The interesting thing is, contrary to wide spread beliefs, that there is no contradiction between free and the selling of software. People can sell, distribute and charge all the money in the world, as long as they allow other people the freedom as described above. A lot of the software developed by the Free Software Foundation is in fact distributed on CD´s for which there is charged money. This is one way to raise money for free software development.

3.32.2 GNU Software and the GNU system

As noted, an operating system is a very complex entity constituted by various programs, so it's an enormous task to develop a new one from scratch. RMS, knowing the grandeur of this undertaking, decided to adapt and use the existing pieces of free software that was available [Stallman 1999:57]. For example, the TeX editor was adopted as the principal text formatter.

Adopting other software than the software developed by the Free Software Foundation meant incorporating other than GNU software. The GNU system comprises of both GNU Software

and other software developed by other people for their own purposes [Stallman 1999:57]. This poses no problems for the GNU system since these non-GNU programs are free software. It should be noted that the term GNU system is referring to the operating system made by GNU software.

After quitting his job, Professor Winston the then head of the MIT AI Lab proposed that RMS could continue to use the lab's facilities, which was gladly accepted. The reason for not keeping the job was that the work produced as an employee would belong to the MIT AI Lab, which was very undesirable.

One of the things needed to develop software was a multi-platform compiler i.e. a compiler that could compile the same programing language on several platforms (types of hardware) Having heard of the Free University Compiler Kit RMS contacted the author asking to use the compiler in the GNU system. The answer was that Free University Compiler Kit was free but NOT the compiler [Stallman 1999:57]. As a consequence RMS decided to develop a C compiler for the Free Software Foundation, this was ofcourse free. The result was the GCC GNU C Compiler, which is used as the Linux standard compiler the GCC is maintained and kept up to date by the Free Software Foundation.

3.32.3 The birth of The Free Software Foundation

Before writing the GNU C Compiler, RMS wrote the GNU Emacs. GNU Emacs is an advanced text editor which, today, can be used to interact with many different programs. Examples are email clients, compilers and other programs that require a text input. RMS started to work on GNU Emacs in September 1984 and in the beginning of 1985 it was usable [Stallman 1999:58].

GNU Emacs had such quality that other people wanted to use GNU Emacs. At that time very few people had access to the internet, and those who had, was tied to a crude text based interface. HTTP and the World-Wide Web (WWW) did not exist until 1991 when Tim Berners-Lee of CERN developed the idea of the browser. That meant that electronic distribution was a poor option, it was however used. An anonymous ftp access was set up at MIT. Still there where many interested users with no access, these had to be serviced. RMS sat up a simple postal service so that people mail and order the software. It was then announced that people could order the program for fee of 150$. In fact this was the first free software distribution business to exist.

In the first short period RMS handled the distribution him self but later that year The Free Software Foundation was created and the distribution business turned over to it. The Free Software Foundation was and is a tax-exempt charity foundation that makes money by distributing free software and by accepting donations, most income comes from the sales of free software [Stallman 1999:60]. Now the Free Software Foundation handles sales and distribution of copies of free software and related services such as printed manuals, t-shirts and CD-ROMs with source and binaries etc. Special services are also provided, that is, compilation software to the platform of choice, which is called a Deluxe Distribution.

Another way of making a living from free software is the selling of services related to the free software. Proven examples are teaching in the use of the software [Stallman 1999:61], development of software, customising of the software. One might call this consulting on the basis of the intimate knowledge acquired in the development process.

3.32.4 Licensing

The essence of the GNU system and the Free Software Foundation was, and is, to provide free software for every user for this to happen there has to be some sort of licensing that ensures this. Some might remember the late 1980's and early 1990's when it was very popular to make public domain software, that is, software that is not copyrighted. When public domain software leaves the author it is free software but there is not a single thing that ensures that it will remain this way. Another programmer or company might improve the public domain software and then make it proprietary. It can also happen that free software wind up in proprietary software and thus making the free proprietary. The X-windows system, developed at MIT under very permissive license agreements, is such an example. X-windows were released as free software but was soon adopted by many computer companies and wound up in their software in binary form only and with the usual strict non-disclosure license. Free software got proprietary, the developers of X-windows knew this and wanted market penetration and a very large user base more than freedom for the users. One of the consequences was that many users used the proprietary version of X supplied by the vendor and had no freedom [Stallman 1999:59]. The X-windows system is now owned by The Open Group and is non-free [Simonsen 2].

Others used the free version of X there were issues of compatibility between the free and the proprietary version. Today the free version is the dominating party supported by the Xfree86 consortium.

The Free Software Foundation thus needed a licensing scheme that could overcome these problems. The method adopted was the copy*left* as opposed to copy*right*, using the copyright law; the idea is to keep the software free.

Copylefted programs grant users permission to run, copy, distribute and modify the program. It is even allowed to modify the program and redistribute it, but it is not permitted to add restriction and thereby impose propriety rights on the code. The copyleft states that modified version must also be free and the changes must be available to the community, if the modified version is published. This way it is possible to share programs and let the programs grow as users improve and publish their improvements [Stallman 1999:60].

The combination of free and non-free software would result in a free piece of software. The license states that once a program, that is the individual lines of code, have been copylefted the program will remain copylefted. This viral effect makes sure that the copyleft licence is spread with the lines of code. If introduced to another program that program will become copylefted. The result is that proprietary software will not include free software, unless the owners of the rights want to make the whole program free.

There are different implementations of the copyleft, the one used for software is the GNU GPL – The GNU General Public License. Other kinds of copyleft exist and are used in specific circumstances, manuals for example use a much simpler version of copyleft than the GNU GPL [Stallman 1999:60].

The GNU Library GPL is an example of another type of copyleft. In general a library is a set of small binary programs that performs different tasks for other programs. The library gives other programs access to the various tasks and thus provides a service for the applications. When a program is run it makes use of these libraries so the question is raised: How to copyleft libraries? One could argue that the use of a library in a proprietary program constitutes an inclusion of the library code in the proprietary program and thus, following the GNU GPL, making the proprietary program copyleft due to the viral effect.

This would of course the undesired effect the proprietary programs either would not be made for the library and thus the free operating system. Or different libraries would be developed, planting seeds of incompatibility. Both alternatives are undesirable and the GNU Library GPL was designed. The GNU Library GPL gives the permission to link to (for programs to use) the libraries. Still the libraries themselves are covered so that they will remain free.

3.32.5 Summing up The Free Software Foundation

Building on the ruins of the old MIT AI Lab software sharing Richard Stallman created the Free Software Foundation, who's sole purpose was and is to provide free software. Today the Free Software Foundation have all most succeeded in this undertaking, there is a free alternative to the proprietary software the GNU/Linux system. The GNU/Linux system is made from some of the GNU tools and the Linux kernel. FreeBSD is also a free Unix-like operating system.

Richard Stallman remains a central character in the Free Software Foundation, and is a controversial person in the free software community. Richard Stallman has been a driving force in shaping a new software sharing culture. It must be noted that many people have participated in the development of software from The Free Software Foundation.

Richard Stallman continues to influence the free software community and the press as well in various ways. The strong views of Richard Stallman have alienated him to some parts of the community The Open Source Community have public discussions [Leonard 1998]

3.33 Refrences

Cargile, Carl F. (1989),
"Information Technology Standardization: Theory, Process, and Organizations", Digital Press, Bedford, MA.

Corbató, F.J. & Vyssotsky, V.A. (1965),
"Introdustion and Overview of the Multics System", 1965 Fall Joint Computer Confrence.

Honderich, Ted (ed.) "The Oxford Companion to Philosophy", 1995, Oxford University Press.

Mahoney, Michael S. (1998),
"The UNIX oral history projekt", AT&T Bell Laboratories.

McKusic, Marshall K. (1999),
 "Twenty Years of Berkeley Unix - From AT&T Owned til Freely Redistributable" in "Open Sources - Voices from the Open Source Revolution", Ed's Chris Dibona, Sam Ockman and Mark Stone, O'Reilly & Associates 1999.

Lewis, Pierre P. (1994),"A very brief look at Unix history",
http://www2.shore.net/~jblaine/vault/plh.html.

Leonard, Andrew (1998)
"The Richard Stallman Saga, Redux", Salon Magazine, Sept. 11, 1998
http://www.salonmagazine.com/21st/feature/1998/09/11feature.html

Munkholm, Hanne; Troelsen, Caper; Jensen, Eva D. (1997)

"The Linux Way", rapport made for the Operating Systems course at the Copenhagen
Engeneering School (Københavns Teknikum).

OpenSource.Org (1999),
"History of the Open Source effort", http://www.opensource.dk/mirror/history.html.

Perens, Bruce (1999)
"The Open Source Definition" in "Open Sources - Voices from the Open Source
Revolution", Ed's Chris Dibona, Sam Ockman and Mark Stone, O'Reilly & Associates 1999.

Ritchie, Dennis M. (1984),
"The Evolution of the Unix Time-sharing system", AT&T Bell Laboratories Technical
Journal 63 no.6 part 2, October, pp. 1577-93

Ritchie, Dennis M, (1996),
"The Development of the C Language", Bell Labs / Lucent Technologies.

Salus, Peter H. (1994),
"A Quarter Century of UNIX", Addison-Wesley Publishing, Inc.

Simonsen, Keld Jørn (1999),
Presentation at the Skåne Sjælland Linux User Group, 12/10-99, www.sslug.dk.

Stallman, Richard (1999),
"The GNU Operating System and the Free Software Movement" in "Open Sources - Voices
from the Open Source Revolution", Ed's Chris Dibona, Sam Ockman and Mark Stone,
O'Reilly & Associates 1999.

Valloppilli, Vinod
"The Halloween document" an internal Microsoft memorandum leaked to Eric S. Raymond
who subsequently published an annotated version at
http://www.opensource.org/halloween/halloween1.html.

Quarterman, John S. & Wilhelm, Sussane (1993),
"Unix, POSIX and open systems - The open Standards Puzzle", Addison -Wesley Publishing Company, Inc. 1993

Change history

16 July 01   The paragraph containing "Oppose Sun Forever" was rewritten to emphasise that it is a pun.