



**Free and Open Source Software
Asia-Pacific (FOSSAP) Consultation**

9 – 11 February 2004, Kuala Lumpur, Malaysia

Introduction to Free/Open Source Software

**Nah Soo Hoe
Malaysia**

**International Open Source Network (IOSN)
An Initiative of UNDP's Asia-Pacific Development Information Programme
<http://www.iosn.net>**

Introduction to Free/Open Source Software

Nah Soo Hoe
Malaysian National Computer Confederation (MNCC) Open Source SIG
(MNCC-OSSIG)
nsh@pop.jaring.my

Introduction

The last few years have seen the rise in popularity of Free/Open Source Software (FOSS) not only in the developed countries, which traditionally lead the way in ICT adoption, but also (and especially so) in the developing countries. It is a phenomenon which has led many in the ICT industry to re-look and re-think about the way software is developed, distributed, utilised and commercialised. It is still too early to say whether FOSS will largely replace proprietary software but what has been ascertained is that FOSS is here to stay as a full-fledged player in the software arena alongside proprietary software in almost all areas of software utilisation and deployment.

This paper will attempt to introduce to the reader how FOSS came about, the philosophy behind it, its community-based development model, its main licensing models and discuss why it is especially useful and important to developing countries.

A Brief History of FOSS [1]

The term Free/Open Source Software is derived from the terms "free software" and "open-source software". Technically speaking these two terms are not identical. Open-source software (OSS) has its roots in free software and the movement surrounding it and is derived from it. To understand the history behind FOSS we need to understand what free software is and how it came about.

Free as in Freedom

When we refer to the term "free software" we mean "free" as in "freedom" and "liberty" and not the monetary meaning of "free". Free software then loosely (a more precise definition will be given later) refers to software in which it is easily and freely available for use and the user has the ability to modify it for his own needs. In addition, the modified software is allowed to be used by others. In order to achieve this freedom of usage and modification, the source code to the software has to be available.

The Early Days

This concept and practice of free software is not new. During the early days of computing, before shrink-wrapped commercial software became widely available, software was freely exchanged with source code by researchers and academics. In those days the computer users were mainly scientists, engineers and researchers. They had to write whatever software needed for their work. As such, they wrote and used their own software and also freely exchanged with other scientists and researchers software with the source code. These early programmers embodied a technical culture of computer enthusiasts. They were passionate about programming and the software that they *crafted*. They understood intimately the machines they were working on. These were the original programmers and hackers!. (The term *hacker* here is used in its original sense to denote a person who is interested in the inner workings of computer systems, hardware and/or software. She probes the innards of hardware/software technologies to learn as much as she can about them. Unfortunately in recent times this term has been used to denote what is more appropriately termed as a *cracker* - a person who exploits vulnerabilities in computer systems with some malicious

intentions. For this paper the original meaning of the term hacker will be used.)

The software implementing the technologies that run the Internet in its early days is a good example of free software. The Internet started with the US Department of Defence contracting US universities and research organisations to develop a wide area computer network with no single point of failure. During those days of the ARPANET (as the research network was then called), software which implemented the networking protocols and services was freely exchanged among the researchers. The basic networking protocol of the Internet, TCP/IP, was developed in this way. It is no exaggeration if we were to say that the Internet runs on free software and owes its success to it.

UNIX and PC Hackers

By the early 1980s the Internet had flourished in the universities in the US and there were two main groups of hackers - UNIX-based hackers and PC-based hackers. The UNIX operating system was developed originally at AT&T Bell Laboratories and subsequently AT&T gave away freely with source code the UNIX software to universities and research institutions. Many of these universities then introduced UNIX to their students and also hacked UNIX to include various enhancements and extensions. The University of California at Berkeley (UCB) was one of those which took UNIX and enhanced it to produce what is now popularly known as Berkeley UNIX. It was widely used in universities and TCP/IP was originally developed on Berkeley UNIX (or its derivatives) platforms running on Digital Equipment Corporation VAX minicomputers. These UNIX-based hackers were based in universities and research institutions. They developed many of the software for driving the Internet and there was widespread exchange of free software over the Internet.

At the same time, the microprocessor revolution had given rise to the microcomputer with first 8-bit, then 16-bit CPUs. The IBM PC and its predecessor the Apple microcomputer were runaway successes and revolutionised the computer industry forever. Around these systems grew the PC-based hackers who worked on proprietary operating systems and environments like Apple DOS, CP/M and PC/MS-DOS. Many of these were hobbyists and self-taught enthusiasts. They exchanged software via telephone dial-up bulletin board systems (BBS), computer clubs and physical floppy disks. The BBS culture encouraged the use of freeware, shareware and public-domain software (see definitions in the section on "Definition of Free Software"). From the 1980s through to the early 1990s the microprocessors grew in power. PC/MS-DOS dominated PCs using Intel CPUs resulting in a new generation of hackers on PCs.

The UNIX Wars

In the meantime UNIX gained popularity especially amongst engineers and scientists who usually ran UNIX on powerful microprocessor-based workstations. These were much more expensive than the PCs but UNIX with its multi-user and multi-tasking capabilities was superior to the single-user, single-tasking, operating systems running on the PCs. However at this time UNIX was beginning to suffer from having too many different versions of it. By now AT&T had commercialised UNIX and had stopped giving away its source code free of charge. Instead AT&T licensed UNIX source code to many computer vendors and these began to produce their own versions of UNIX to differentiate their products. While they all profess to be UNIX at heart, each one of them was different enough to cause sufficient incompatibilities at both the operational and source code levels to result in a fragmented UNIX environment. The various vendors (including AT&T) sell their versions of UNIX as proprietary products under their own brand names e.g. AIX, Ultrix, HP-UX, Solaris, Xenix etc. These UNIX licenses were not cheap as compared to the licensing fees for running PC-based operating systems like PC/MS-DOS. To further add confusion to all this, Berkeley UNIX was also being actively developed and commercially marketed as Berkeley Systems Distribution (BSD) UNIX. Many of the vendors incorporated both the features of AT&T UNIX and BSD UNIX into their proprietary UNIXes.

As a result of these proprietary UNIX wars and the fragmented UNIX environment, PCs with ever increasing power, relative ease-of-use, cheaper costs and abundance of software made inroads into business and even scientific computing. PC-based local area networks (LANs) helped to overcome to a certain extent the PC's inherent single-user operating environment but by and large the PC still could not compete with UNIX workstations and minicomputers for many demanding business and scientific applications.

By the early 1990s, with the the arrival of 32-bit CPUs for PCs, ports of UNIX to the PC were made and a fully-functional UNIX system could run on PC hardware. However, while UNIX for PC-class machines were available commercially their software license prices were still expensive. In the meantime, the power and influence of PC/MS-DOS and later MS-Windows on the PC grew. PC-based Ethernet LANs became more sophisticated and proliferated, especially those running on Novell's proprietary Netware network operating system. Later Microsoft successfully introduced its own multi-tasking operating system, Windows NT, which further narrowed the gap with UNIX. Many in the industry thought that UNIX was on the way out to be replaced for good by these new operating systems and environments.

The Rise of the Free UNIXes and GNU/Linux

Around that time BSD UNIX was ported to the Intel 386-based architected PCs and from this several free and non-commercial UNIXes developed e.g. FreeBSD, NetBSD, OpenBSD. Still, on PCs, large numbers of young people were exposed mainly to MS-DOS/MS-Windows and these early free UNIXes attracted only a niche following. Earlier in 1983 the Free Software Foundation (FSF) was founded by Richard Stallman. One of the main projects of the FSF is the GNU project, to produce a UNIX operating system clone using only free software. Apart from the kernel, GNU managed to deliver almost all of the parts of the operating system and system software development tools and toolkit. Linux Torvalds, a student at Helsinki University, in 1991 started development of a free UNIX kernel for 386 PCs using the FSF toolkit. This project attracted many hackers over the Internet. It became a full-featured *free* UNIX work-alike. Most of the work was concentrated on the kernel and much of the other supporting software taken from GNU. This was the birth of Linux (or a more appropriate name, GNU/Linux as most of the non-kernel software came from the GNU Project). From the start Linux made it easy for MS-DOS users to co-exist and/or migrate to it. As a result Linux attracted many hackers and users, both from the UNIX and non-UNIX sides.

From the start, the Linux model of software development was quite different from the traditional way in which a relatively small, tightly-knit group of people worked on software in a controlled and coordinated fashion. Instead Linux was hacked on by huge numbers of volunteers coordinated only through the Internet. Quality was maintained by releasing code very often and getting fast feedback from many users. In this manner only good quality code and useful features were used. The Linux kernel was released under the GNU General Public License (GPL) and so it is Free Software. (More details on the GPL will be discussed in the section on FOSS Licenses.)

The Internet WWW Explosion

Around the early-1990s, the Internet exploded with the introduction of the World Wide Web (WWW). Its multimedia support and ease-of-use resulted in non-techies going online. Suddenly hackers which created much of the underlying Internet technologies were much sought after for their technical skills in this "new" frontier. UNIX-based computers found wide deployment as Internet servers. Many Internet services were originally developed on UNIX and distributed as free software and so UNIX-based servers function well as Internet servers. The free UNIXes and work-alikes running on the Intel x86 architectures, e.g. xBSD and Linux, found widespread popular use because they couple affordability and availability with the functionality of a full-fledged UNIX system. The subsequent dot com boom years further fueled the demand for such systems and skills. The Internet boom thus contributed to the widespread use of free software and the renewed surge in interest in UNIX and work-alike systems.

Open Source Software (OSS)

With the success of the Internet, some people (mainly the techies) started to introduce free software used on the Internet to the business organisations where they work. However, very often they were met with negative sentiments about using free software by the management. To these people, who were used to paying thousands of dollars for the software sold by proprietary vendors, the term "free software" is perceived as software of poor quality and unreliable.

There was another problem in that many in the business community viewed that the free software community/movement as championed by the FSF was hostile to businesses and the proprietary software used widely by these businesses. As a result management in corporations and businesses were reluctant to use free software.

To try and overcome these problems associated with the usage of the term "free software", the term "open-source software" or "open source" was coined in 1998 and the Open Source Initiative (OSI) established [2]. This creation of the Open Source movement can be seen as an attempt by some people in the free software community to take a less confrontational approach towards working with proprietary software and the business community. They were willing to take on a pragmatic approach in trying to engage the management and business communities to get them to understand and appreciate the software that were developed by the free software community. They realised that this can be an effective way to get managerial and business buy-in and hence lead to more adoption of free software in corporations and other establishments.

Free Software or Open Source Software?

Open-source software has its roots in free software. The Free Software Foundation has defined what constitutes Free Software and it shares the same basic philosophy and outlook as Open Source Software (see the sections on their definitions below). However, the FSF favours the retained usage of the term "Free Software" to reflect the "freedom" aspect of the philosophy behind it. It argued that by changing the name to Open Source Software, the ideals expounded by Free Software will not be emphasised and subsequently may even be lost. This may result in a tendency for people to compromise and switch back to software which does not fulfill the Free Software definition.

The FSF, true to its roots as a social movement, rejects some of the more compromising stand which the Open Source movement has adopted towards software commercialism and proprietariness. However, while the two movements have ethical and philosophical differences, they are essentially on the same side in the battle against proprietary software.

One should note that many of the Open Source Licenses are recognised by the FSF as Free Software. In everyday usage to all intents and purposes we can treat OSS and Free Software to be the same. Many people are now using term Free/Open Source Software (FOSS) to refer to open source and free software in general. For the rest of this paper we shall do likewise.

Definition of Free Software

The definition of Free Software, as stated by the Free Software Foundation (FSF) [3], is software in which one has:

- the freedom to run the program, for any purpose
- the freedom to study how the program works, and adapt it to your needs
- the freedom to re-distribute copies so you can help others

- the freedom to improve the program, and release your improvements to the public

It is important to realise that to perform most of the above, access to the source code of the software is needed.

In addition there are other categories of software which some people very often confuse them with free software: freeware, shareware and public domain software [4].

Freeware and shareware are not free software even though these categories of software allow you to freely distribute and copy the software. While freeware is distributed free-of-charge the source code is usually not available and even if it is, modifications and re-distribution of the modified code is not permitted. Shareware is software which is chargeable after a period of usage (usually at a relatively low price) and modifications and re-distribution of the modified code are not permitted.

Public domain software is a special type of free software in which the author has given up all his claims of copyright to it and so the source code is available and modifications and re-distribution are freely allowed.

Definition of Open Source Software

The definition of Open Source Software, as stated by the Open Source Initiative (OSI) [5], is software which has a license that:

- allows free re-distribution
- allows its source code to be freely and easily available
- allows modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software
- caters to the integrity of the author's source code
- must not discriminate against any person or group of persons or fields of endeavour
- does not restrict other software
- is technology neutral
- is not specific to a product
- allows distribution of license

It can be seen that the OSS definition is not identical to the FSF definition. The OSS definition resembles a more pragmatic approach to free software licensing. However, many free software licenses can fit under both licensing regimes..

Free/Open Source Software Licensing

The OSI has listed over 30 licenses which it considers as Open Source Licenses [6]. The FSF has recognised many (but not all) of these Open Source Licenses as Free Software Licenses too [7]. These FOSS software licenses fall into 2 main categories:

- Copylefted
- Non-copylefted

Copyleft Software [8]

One of the greatest fears of Free Software advocates is that if the software licensing terms are not set correctly, someone may convert a free software to non-free software. Once this is done, the software is probably lost forever to the free software world. To prevent this, a type of licensing called the Copyleft is used.

The creator/owner maintains copyright on the software. In addition to this copyright, he puts in certain restrictions to prevent the conversion to non-free software. An example of such a restriction may be to state in the licensing terms that anyone who re-distributes the software, with or without changes, must pass along the freedom to further copy and change it. This guarantees that every user has freedom. An example of a Free Software license that is copylefted is the GNU General Public License (GPL) [9]. Linux is distributed under a GNU GPL license.

The GPL is a widely used license for FOSS and it is viral in nature in that any piece of software which incorporates some code from a GPL software will have to be distributed under a GPL license too.

The GPL is a strong proactive license to safeguard the selfish interests of free software. So like a normal proprietary software license it tries to protect the interests and intentions of the original people behind the software in question. One may argue that with this the GPL restricts freedom on the part of the developer in order to protect the software's own freedom guarantees.

Non-Copyleft Software

There are other FOSS Licenses which have much more permissive licensing terms than the copylefted licenses. These allow the user permission to re-distribute and modify the software and permission to add additional restrictions to its further distribution and modification. This means that it is possible for someone to take such software and make it proprietary with or without modifications. Examples of such licenses are the X11 (and XFree86) license, BSD License and Apache License.

Many programmers prefer to distribute their software under a liberal licensing regime like the BSD License since it allows their software to be incorporated easily into other software both FOSS as well as proprietary.

Community-Style Software Development

Many FOSS projects (see for example those listed on Sourceforge [10]) use a community-style development process in which while there are one or more project leaders, almost anyone can sign-up and contribute code to the project. The contributed code can be viewed by the people in the project as well as other users. In this fashion, there is peer review of the code almost immediately and any weaknesses or poor code is usually spotted and corrected before it is too late. This style of software development may be contrasted with the traditional model whereby only a relatively small number of developers are able to contribute code to the project and widespread peer review of the code under development is not forthcoming.

This community-style method of development does not mean that there is chaos and confusion in the code or there are multiple versions of it. The owners of the project will decide which code gets committed to the actual software itself, after taking into consideration the views and opinions of the contributors. The main advantage of this style of development is that anyone who has the interest and ability has a chance to participate in the project and may even move up to be one of the leaders of the project at a later stage. Also, anyone can start off a project and host it on the Internet to enable other people all over the world to participate in it. Whether the project attracts any following and flourishes or not is left to its own merit i.e. its usefulness, technical capabilities, etc.

It is possible for some one who is not happy with the way a FOSS project is progressing to fork the project i.e. form a break-away project and come up with another version of the same software which may not be compatible with the original version. The ability to fork a project prevents dominance by a single entity in a project. Sometimes it happens that a substantial number of followers and users of the project do not agree with the way the project is heading or is being implemented and the forking process enables them to forge their own version of the software. One potential danger of allowing this is of course ending up with incompatible versions of the same software. While this is possible, in practice it usually does not happen as the contesting parties try very hard to come to an agreement or compromise. It is only after serious attempts at re-conciliation have failed that a forked project takes birth. After that experience has shown that the two projects will co-exist for some time, and after that one of them will die off due to lack of popularity and usage or they will be folded back into a single project. It is thus akin to a natural selection process whereby only the stronger software survive. In this way the forking can be viewed as something good in that the outcome is a better software.

General Benefits of FOSS [11]

Here we will highlight some of the more important benefits of using FOSS. Many of these benefits come about because the FOSS licenses allow easy availability and access to source code as well as the right to modify and re-use and re-distribute the modifications.

Freedom to Learn, Re-distribute and Enhance the Software

With FOSS, anyone who is willing can learn from the software, re-distribute for others to use if it suits them and enhance the software if found necessary. If one does not have the required skills to do that, one can always engage other people who have the capability to do it. This means that while the software author may still own the copyright to the software, the user is not at his mercy, unlike proprietary software. The user's interests are protected.

Prevention of Technology Lock-in and Promotes Adherence to Open Standards

Single vendor and technology lock-ins are prevented since the code is there for all to see and so no single group or vendor can implement some secret or proprietary technology. Instead it ensures that open standards are promoted and preserved. The compliance to open standards helps to bring about fairer competition based on technical merit and everyone or organisation, big or small, can participate.

Open file and data formats are adhered to by FOSS. This is extremely important for users who have huge databases and data stores. The government is one such user. It is thus to the benefit of a country if its government specifies FOSS for its usage.

More Robust and Reliable Software

Many FOSS software development encourages peer review and feedback of the software. This will lead to more robust and reliable software since many people are looking at the code and a greater proportion of bugs and problems can be caught and fixed earlier and faster. It

can also result, arguably, in better security as obvious security weaknesses and bugs can be caught early.

Promotes Positive Traits

The participation of the world-wide community through the Internet in the usage, testing and in the development of FOSS itself promotes an environment for positive competition, self-learning, exploring and co-operation. This is especially true for the technical people whereby through the Internet a particular piece of software and its usage can be thoroughly discussed, its technical features and implementation debated, bugs, fixes and work-arounds reported and new features suggested. The authors of the software often benefit from all the input too and they in turn come up with a better piece of work. Usually, inferior code is not tolerated since the source code is there for all to view and any weaknesses can be pointed out immediately. The robustness of software like Apache and FreeBSD bears testimony to this.

Affordable Software

While many of the FOSS Licenses [6, 7] do not prohibit charging for the software, most of the major ones are freely available for downloading over the Internet or for distribution on CD-ROMs with no licensing fees. This obviously benefits poorer societies such as those in the Third World in providing useful, good and affordable software running on affordable computers and networks. Many of the mainstream FOSS are comparable with, or in some cases better than, expensive commercial proprietary software, e.g. the Apache web server, the Mozilla web browser, the PHP web scripting language, the MySQL database system, the GNU GCC C/C++ compiler and the OpenOffice suite of office applications.

Benefits of Open Source to Developing Countries

In addition to the above general benefits, for less developed and less-technically advanced countries, FOSS offers several other benefits. All developing countries strive to become more technology driven and proficient with the hope of improving their economies. Almost all modern technology products need software to drive it and make it work. Thus, software today has become the *single* most important component in a technical product as without the appropriate software the product will not work properly. It is therefore crucial that developing countries can cope with this increased dependence on software and software-creating resources for their technology and economic advancements. FOSS and its associated models and eco-systems provide a relatively affordable and feasible means to achieve this due to the availability of source code and the opportunity to build on top of the work of others as well as the international co-operative community development environment.

Provides a Conducive Software Development Environment

FOSS promotes an environment which a country, striving to become a technology-driven one, should have. The ability to "look under the hood" and learn due to the availability of source code is very important to developing countries. The fact that FOSS has its roots in the hacker culture of the early UNIXes and Internet formative years means that the FOSS development environment (and indeed the user environment too) is conducive for the promotion of software and technology creativity and innovation.

Ability to Learn, Innovate and Invent

Many of the successful FOSS are systems programs like kernels, system utilities, network drivers, debuggers, compilers etc. and typically a less-developed country will lack people good in these areas. However, system programs and their development are crucial to implement any technology and hence FOSS promotes an environment conducive for technical and system development. Initially the local people are encouraged to look at the code and learn from it, then later on possibly improve or customise it to suit their special needs. Ultimately once they possess enough skills and experience, they can start off new FOSS development

projects of their own and invite the rest of the world to participate. This ability to *Learn, Innovate and inVEnt, or LIVE*, is in the author's opinion one of the most compelling reasons for a country to embrace FOSS.

Less Dependence on US-Centric Software

Currently most of the proprietary off-the-shelf software that we use are from US vendors like Oracle and Microsoft. As such we are totally dependent on these companies in our ICT implementations and projects, some of which are of critical national importance. Worse still, because the source code is usually not made available, there is no way for anyone to make an independent audit and check on the code. Since our purchases are usually relatively small it is unlikely that these large international software companies, purely business driven, will listen to what we want done for the software. This dependence on software and technologies where we can have no control or say can be overcome to a large extent if FOSS is used more, especially by the Government.

Builds Critical Mass of Good Coders

Active participation in FOSS projects by the technical people of a country will build up a critical mass of good coders and also help stimulate and build up the local software industry. Localised versions of software can be produced which will help immensely in countries where English is not too widely used.

Conclusion

This document has served as a brief introduction to FOSS. It has examined its history, the definitions of both Free Software and Open Source Software as defined by the FSF and OSI respectively, the Copyleft licensing concept, as well as the benefits of FOSS. It is hoped that by reading this introductory text on FOSS, the reader may be in a better position to understand what FOSS is and why it is very important to developing countries in particular.

References

1. "A Brief History of Hackerdom", Eric Raymond, <http://www.catb.org/~esr/writings/hacker-history/>
2. "History of the OSI", <http://www.opensource.org/docs/history.php>
3. "The Free Software Definition", <http://www.fsf.org/philosophy/free-sw.html>
4. "Categories of Software", <http://www.fsf.org/philosophy/categories.html>
5. "The Open Source Definition", <http://www.opensource.org/docs/definition.php>
6. "Open Source Licenses", <http://www.opensource.org/licenses>
7. "Various Licenses and Comments about Them", <http://www.fsf.org/licenses/license-list.html>
8. "What is Copyleft", <http://www.fsf.org/licenses/licenses.html#WhatIsCopyleft>
9. The GNU General Public License, <http://www.fsf.org/licenses/gpl.html>.
10. Sourceforge, <http://sourceforge.net/>
11. "Free and Open Source Software - Origins, Benefits, Myths and Reality", http://opensource.mimos.my/fosscn2003cd/paper/full_paper/nah_soo_hoe.sxw