first monday
PEER-REVIEWED JOURNAL ON THE INTERNET

Essence of Distributed Work: The Case of the Linux Kernel

by Jae Yun Moon and Lee Sproull

This paper provides a historical account of how the Linux operating system kernel was developed from three different perspectives. Each focuses on different critical factors in its success at the individual, group, and community levels. The technical and management decisions of Linus Torvalds the individual were critical in laying the groundwork for a collaborative software development project that has lasted almost a decade. The contributions of volunteer programmers distributed worldwide enabled the development of an operating system on the par with proprietary operating systems. The Linux electronic community was the organizing structure that coordinated the efforts of the individual programmers. The paper concludes by summarizing the factors important in the successful distributed development of the Linux kernel, and the implications for organizationally managed distributed work arrangements.

Contents

Introduction

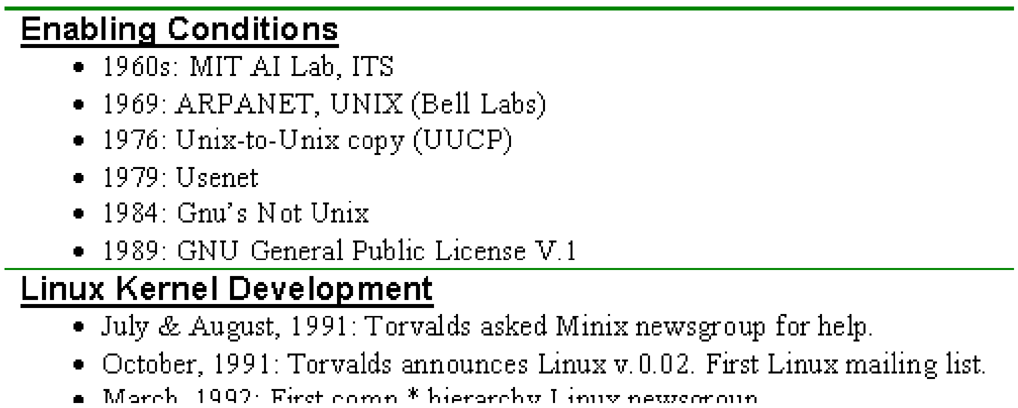
Complex tasks plus a global economy have impelled the creation of many distributed engineering and development groups supported by information and communication technologies [1]. Distributed groups range in duration from weeks to years; they range in size from fewer than ten people to more than a thousand; and they may have members located in two locations or many locations. Distributed engineering and development depends upon careful planning, coordination, and supervision. "Careful" does not necessarily imply micromanagement. Contributors who are geographically distant from one another inevitably operate under some degree of autonomy. The management challenge is to insure that members of geographically distributed engineering and development teams stay focused on shared goals, schedules, and quality. This challenge grows as the number of employees and sites increases. There have been some notable successes in large-scale distributed engineering and development. For example, the Boeing 777 marshaled 4,500 engineers working on three continents (Committee on Advanced Engineering Environments, 1999). But successful large-scale distributed engineering and development projects are rare.

That, in part, explains the business and media fascination with Linux. Linux is a PC-based operating system (OS) that has been produced through a software development effort consisting of more than 3,000 developers and countless other contributors distributed over 90 countries on five continents. It is difficult to provide a precise estimate of the number of programmers who have contributed to Linux. Published estimates range from several hundred to more than 40,000 (Shankland, 1998; Raymond, 1999). In its first three and a half years of development (November 1991 to July 1995) more than 15,000 people submitted code or comments to the three main Linux related newsgroups and mailing lists. In the next three and a half years, thousands continued to contribute code and hundreds of thousands of people joined electronic discussions about Linux philosophy, applications, competitors, business models, and so forth. In this paper we focus narrowly on people actually writing code for the operating system kernel.

As of December 1998, more than eight million users were running Linux on a wide variety of platforms. Linux was estimated to have 17% of server operating systems sales in 1998 and was projected to have a compound annual growth rate of 25%, 2.5 times greater than the rest of the market (Shankland, 1998; Berinato, 1999). It was widely regarded as being of very high quality and reliability. According to 1998 internal Microsoft documents assessing the potential threat from Linux, the Linux failure rate was two to five times lower than that of commercial versions of Unix (Valloppillil, 1998). In both 1997 and 1998 Linux won the Info World Product of the Year award for best operating system; in 1997 it won the InfoWorld Best Technical Support award. Such a successful large-scale distributed project would make any organization proud and its shareholders happy.

But the real fascination with Linux stems from the fact that it is not an organizational project. No architecture group developed the design; no management team approved the plan, budget, and schedule; no HR group hired the programmers; no facilities group assigned the office space. Instead, volunteers from all over the world contributed code, documentation, and technical support over the Internet just because they wanted to. This paper analyzes factors contributing to the Linux kernel story and explores how those factors could be brought into play in formal organizations that are managing distributed work [2].

A 1971 book about the Cuban missile crisis, called *Essence of Decision*, suggested the title and rhetorical structure for this paper. That book's author, Graham Allison, observed that no single perspective could provide an entirely satisfactory explanation of a complex social phenomenon. Thus, in the book he told the story of the missile crisis three times, each time from a different explanatory perspective. In this paper, after a brief description of Linux and its enabling conditions, we tell the Linux kernel story three times. (See Figure 1 for timeline). First we tell the story from a "great man" perspective, emphasizing the technical and management abilities of Linus Torvalds. We then tell the story a second time from a "hacker culture" perspective emphasizing what motivates individual hackers to work on this project. We then tell the story a third time from an "electronic community" perspective, emphasizing communication forums and role differentiation. We conclude with suggestions for how the factors emphasized in these three perspectives could pertain to distributed work within organizations.



Enabling Conditions

- 1960s: MIT AI Lab, ITS
- 1969: ARPANET, UNIX (Bell Labs)
- 1976: Unix-to-Unix copy (UUCP)
- 1979: Usenet
- 1984: Gnu's Not Unix
- 1989: GNU General Public License V.1

Linux Kernel Development

- July & August, 1991: Torvalds asked Minix newsgroup for help.
- October, 1991: Torvalds announces Linux v.0.02. First Linux mailing list.
- March, 1992: First comp.* hierarchy Linux newsgroup

- March, 1992: First comp.os.linux newsgroup
- March, 1994: Linux v.1.0. Parallel release structure. Credits File.
- February, 1996: Maintainers File
- June, 1996: Linux v.2.0

Figure 1. Timeline of Key Events

What's Linux

An operating system is a computer's central program that oversees resource allocation, program execution, and the control of input, output, file management, and peripheral devices. The kernel is the central module of the operating system - the part of the operating system that loads first, remains in the main memory of the system, and controls memory management, process and task management, and disk management. Common operating systems in use today are Microsoft Windows and Unix-based systems such as Sun Solaris™ and IBM AIX™.

Linux version 1.0, released in March 1994, offered most features of a typical Unix operating system, including multitasking, virtual memory, and TCP/IP networking. It had about 175,000 lines of code. Linux version 2.0, released in June, 1996, offered 64-bit processing; symmetric multiprocessing, which allows the simultaneous deployment of several chips in a system; and advanced networking capabilities. It had about 780,000 lines of code. Version 2.1.110, released in July, 1998, had about 1.5 million lines of code: 17% in architecture-specific code, 54% in platform-independent drivers, and 29% in core kernel and file systems.

Linux is now much more than an operating system. As the number of people interested in Linux grew, they formed user groups to share information and code in face-to-face meetings with local Linux users in addition to communicating through the Internet with any Linux user in the world. By July 2000, there were more than 400 Linux user groups in 71 countries (Linux User Groups Worldwide, http://lugww.counter.li.org). More Linux users meant more interest in applications to run on the operating system. People began developing their own applications and making them available on the Web. The August 1999 Linux Software Map database lists about 2,500 people who have contributed over 3,500 applications which range from word processors and mathematical applications to games (Dempsey, Weiss, Jones, and Greenberg, 1999). Approximately 300 people have also contributed over 20 megabytes of Linux documentation ranging from information sheets to full scale manuals. Over half of the documentation is in "How-to" documents, which are mini-tutorials on various tasks related to Linux development, implementation, and use (Dougherty, 2000).

Anyone can download all Linux files from the Internet for free. Thus, anyone who wants to can install a Linux OS, applications, and documentation on his/her PC without going through (or paying) a commercial organization. For people who don't have the time or skill for a "do-it-yourself" installation, however, several commercial firms, such as RedHat, Suse and Caldera, and non-profit organizations such as Debian, also sell and support low-cost CD-ROM distribution versions of Linux [3].

Enabling Conditions: Open Source and Internet

Two enabling factors underlie the Linux development as well as that of similar volunteer distributed software development projects [4]. The first is the social and legal conventions of "open source," a means for sharing software code. The second is the information and communications infrastructure of the Internet.

Open sharing of software code was a common practice in the MIT Artificial Intelligence Laboratory in the early 1960s and in similar laboratories at universities such as Stanford and Carnegie Mellon (Levy, 1984). Because there were very few computers at that time, people would start a program and leave it available for others using the machine after them to admire and improve upon. By the late 1960s, the scarcity of computers and the growing number of programmers and users led MIT to develop the Incompatible Timesharing System (ITS) for the Digital PDP-10, the large computer in use at the AI Lab. The system had no passwords and gave users the ability to browse both the source code [5] of the system itself as well as the personal programs and documents created by other users. In essence, the ITS facilitated open sharing of software code.

Also in the late 1960s, a small group of people within AT&T's Bell Laboratories, led by Ken Thompson and Dennis Ritchie, developed the Unix operating system (Lerner and Tirole, 2000; Salus, 1994). Initially, the system was used within AT&T for research purposes. Antitrust regulations of the time restricted AT&T's ability to expand its business into areas beyond the telephone industry. After Thompson presented the ideas represented in the system at the ACM Symposium on Operating Systems Principles in 1973 however, Unix began to spread throughout the research and academic community (Salus, 1994). AT&T licensed the Unix operating system to academic institutions for a nominal fee and distributed the system in source, but maintained a policy of no official support. Due to the lack of official support, users of the Unix operating system began to informally share their bug fixes and improvements with one another. In 1975, many users met at the first USENIX meeting, the purpose of which was to bring together various people to share their experiences. In the late 1970s, the Unix user-developers developed a new feature, known as the UUCP (Unix-to-Unix-Copy), that enabled them to exchange files and data over phone lines (Gaffin and HeitKotter, 1994). The feature was used to distribute information to the Unix community, and led to the beginning of Usenet [6]. In addition to the institutionalization of this open sharing culture, Unix also embodied various innovative principles that made the system portable and simple in design.

Advances in computing architecture by the early 1980s saw the demise of the PDP-10 series machines for which MIT's ITS had been developed (Stallman, 1998). New machines in the AI Lab were now equipped with proprietary operating systems, which, unlike the ITS, came in binary code wrapped with non-disclosure agreements that forbade sharing with other people. Richard Stallman, who had used ITS at the AI Lab, started the GNU (GNU's Not Unix) project in 1984 to develop a free operating system like ITS to support open sharing and collaboration. However, free access to source code for everyone meant that commercial software developers could exploit the free software to develop proprietary programs. Stallman instituted the Copyleft software license to guarantee sustained easy sharing of code by decreeing that all users of the program have the right to use it, copy it, modify it, and distribute their modifications. Additionally, Copyleft also explicitly forbids privatization of derivations from software distributed under copyleft; derived works must also be distributed under the copyleft license. The canonical copyleft license is the GNU General Public License (GPL), which also forbids packaging of proprietary software with GPL-licensed software (Free Software Foundation [FSF], 1991). Copyleft codified the open sharing practices of the early closely-knit group of programmers; software distributed under a Copyleft license is known as open-source software.

Whereas copyleft codified the social and legal norms of open sharing [7], the Internet provided a ubiquitous technology infrastructure that made it easy for programmers to communicate and share their code. While its predecessor, the Advanced Research Projects Agency Network (ARPANET) was a network of research laboratories in academia and industry funded by the U.S. Department of Defense (King, Grinter and Pickering, 1997), the Internet broadened the scope of connectivity through widespread emergence of commercial Internet service providers. It was no longer only the few privileged members of places like the AI Lab at MIT or Bell Labs who could share code and collaborate with others on improving their systems. The standard, robust communication system of the Internet made possible the exchange of both messages and source code among programmers worldwide. People could learn about a project and participate in open discussions about it through Usenet bulletin board discussions and electronic mailing lists. They could also download source code from the Internet and upload their modifications to the Net for others to critique and improve. The Internet made it possible to go beyond the exchange of source code among people who already knew one another. In the early days, the ITS facilitated sharing of code among programmers at the AI Lab; the ARPANET enabled collaboration and open sharing of code among programmers in select geographically-dispersed research institutions; and finally the Internet made it possible for people anywhere to share their code and ideas.

Linus Torvalds

In October 1991, Linus Torvalds, a computer-science graduate student at the University of Helsinki, announced on the Internet that he had written a "free version of a minix-lookalike for AT-386 computers" and would make it available to anyone who was interested (See Figure 2 for Torvalds' announcement message). Minix was a simplified version of Unix written and maintained largely as a teaching tool by a European computer science professor, Andrew Tanenbaum. It was widely used in computer science courses, even though its license cost $79.95 and restricted its redistribution. As a result of Minix use by computer science students, the professor was bombarded with requests and suggestions for improvement. He was reluctant to implement many of them, however, and some students were frustrated with his unresponsiveness [8]. Torvalds, a frustrated Minix user, noted in his announcement message that all of the code in his operating system was freely-available to be copied by others. He also volunteered to add functions written by others, if they were also freely distributable. In the next thirty months, Torvalds released 90 additional versions of his OS, culminating on March 14, 1994, with the release of version 1.0.

```
Date:         Sat, 5 Oct 1991 05:41:06 GMT
Reply-To:     INFO-MINIX@UDEL.EDU
Sender:       INFO-MINIX-ERRORS@PLAINS.NODAK.EDU
Comments:     Warning -- original Sender: tag was
info-minix-request@UDEL.EDU
From:         Linus Benedict Torvalds <torvalds@KLAAVA.HELSINKI.FI>
Subject:      Free minix-like kernel sources for 386-AT

Do you pine for the nice days of minix-1.1, when men were men and
wrote their own device drivers? Are you without a nice project and
just dying to cut your teeth on a OS you can try to modify for your
needs? Are you finding it frustrating when everything works on minix?
No more all-nighters to get a nifty program working? Then this post
might be just for you :-)

As I mentioned a month(?) ago, I'm working on a free version of a
minix-lookalike for AT-386 computers. It has finally reached the
stage where it's even usable (though may not be depending on what you
want),and I am willing to put out the sources for wider distribution.
It is just version 0.02 (+1 (very small) patch already), but I've
successfully run bash/gcc/gnu-make/gnu-sed/compress etc under it.

Sources for this pet project of mine can be found at nic.funet.fi
(128.214.6.100) in the directory /pub/OS/Linux. The directory also
```

```
contains some README-file and a couple of binaries to work under
linux (bash, update and gcc, what more can you ask for :-). Full
kernel source is provided, as no minix code has been used. Library
sources are only partially free, so that cannot be distributed
currently. The system is able to compile "as-is" and has been known
to work. Heh. Sources to the binaries (bash and gcc) can be found at
the same place in /pub/gnu.

ALERT! WARNING! NOTE! These sources still need minix-386 to be
compiled (and gcc-1.40, possibly 1.37.1, haven't tested), and you
need minix to set it up if you want to run it, so it is not yet a
standalone system for those of you without minix. I'm working on it.
You also need to be something of a hacker to set it up (?), so for
those hoping for an alternative to minix-386, please ignore me. It is
currently meant for hackers interested in operating systems and 386's
with access to minix.

The system needs an AT-compatible harddisk (IDE is fine) and EGA/VGA.
If you are still interested, please ftp the README/RELNOTES, and/or
mail me for additional info.

I can (well, almost) hear you asking yourselves "why?". Hurd will be
out in a year (or two, or next month, who knows), and I've already
got minix. This is a program for hackers by a hacker. I've enjouyed
doing it, and somebody might enjoy looking at it and even modifying
it for their own needs. It is still small enough to understand, use
and modify, and I'm looking forward to any comments you might have.

I'm also interested in hearing from anybody who has written any of
the utilities/library functions for minix. If your efforts are freely
distributable (under copyright or even public domain), I'd like to
hear from you, so I can add them to the system. I'm using Earl Chews
estdio right now (thanks for a nice and working system Earl), and
similar works will be very wellcome. Your (C)'s will of course be
left intact. Drop me a line if you are willing to let me use your
code.

    Linus

PS. to PHIL NELSON! I'm unable to get through to you, and keep
getting "forward error - strawberry unknown domain" or something.
```

Figure 2. Announcement of Linux v.0.02 (October 1991)

Torvalds certainly did not set out to create a worldwide distributed OS development project. But he exhibited both technical and management capabilities and decisions without which Linux would not have grown. He makes it easy to build a "great man theory" of the success of Linux. Known as a "damn fine programmer" (Raymond, 1999) and an "arch hacker" (Moody, 1997), Torvalds' programming skills are widely admired. Even today up to half the code in key parts of the kernel have been written by Linus (Yamagata, 1997). And in the early days, if he didn't write the code himself, he edited code by others so heavily "as to be totally unrecognizable" by its original authors (Torvalds, 1992, March 8).

Beyond programming skill, Torvalds' chief technical contribution lay in designing a portable, modular architecture for the kernel. Torvalds wrote the initial Linux for his own PC, which had an Intel 386 architecture. By 1993 Linux had been completely rewritten once to run on a Motorola 68K architecture and it was about to be rewritten yet again to run on a DEC alpha architecture. The prospect of having to maintain a completely separate code base for every new machine architecture was unacceptable to Torvalds. Thus he redesigned the Linux kernel architecture to have one common code base that could simultaneously support a separate specific tree for any number of different machine architectures (Torvalds, 1999b). The architecture greatly improved the Linux kernel portability through establishing systematic modularity. A modular system minimized the need for communication between different components of the kernel and made it possible to write code in parallel on different portions of the kernel (Torvalds, 1999a; 1999b) [9]. Modularity decreased the total need for coordination and meant that necessary remaining coordination could be deferred.

In creating a worldwide distributed OS development project, Torvalds' management decisions and skills were just as important as his technical ones. Perhaps Torvalds' most important management decision was establishing, in 1994, a parallel release structure for Linux (1994, March 15). Even-numbered releases were reserved for relatively stable systems and focused only on fixing bugs; odd-numbered releases were the development versions on which people could experiment with new features. Once an odd-numbered release series incorporated sufficient new features and became sufficiently stable through bug fixes and patches, it would be renamed and released as the next higher even-numbered release series and the process would begin again [10]. The parallel release structure allowed Torvalds to simultaneously please two audiences who are often in conflict with one another. Users who rely upon a Linux OS to support their production computing want a stable, reliable system in which new releases introduce only well-tested new functionality, no new bugs, and no backward compatibility problems. Developers, by contrast, want to try out new ideas and get feedback on them as rapidly as possible. The parallel structure offered both relative stability and certainty to users and rapid development and testing to programmers. Table 1, which displays the release history of Linux, illustrates the disparate release rate between the two release trees.

Release Series	Date of Initial Release	Number of Releases	Time to Final Release in Series	Duration of Series
0.01	9/17/91	2	2 mos.	2 mos.
0.1	12/3/91	85	27 mos.	27 mos.
1.0	3/13/94	9	1 mo.	12 mos.
1.1	4/6/94	96	11 mos.	11 mos.
1.2	3/7/95	13	6 mos.	14 mos.
1.3	6/12/95	115	12 mos.	12 mos.
2.0	6/9/96	34	24 mos.	32 mos.
2.1	9/30/96	141	29 mos.	29 mos.
2.2	1/26/99	14	9 mos.	--
2.3	5/11/99	60	12 mos.	--

Table 1. Linux Release History [11]

Despite his position of importance there is little that is imperious or dictatorial in Torvalds' communication style. Initially he was (appropriately) quite deprecatory about his project, "just a hobby, won't be big and professional like gnu" (Torvalds, 1991, August 25). He never orders anyone to do anything and even his suggestions are mild-mannered. Typical suggestions are of the form: "hint, hint, Linus wants to get out of doing it himself;^)" (Torvalds, 1995, June 29) or "my personal priority is not that kind of behaviour, so it would be up to somebody else to implement this.. (hint hint)" (Torvalds, 1996, April 16). Yet, there is no confusion about who has decision authority in this project. Torvalds manages and announces all releases - all 569 of them to date. He acts as a filter on all patches and new features - rejecting, accepting, or revising as he chooses. He can single-handedly decide to redo something completely if he wishes. In the early days he personally reviewed every contribution and communicated via personal e-mail with every contributor. As Torvalds says, "[There is] one person who everybody agrees is in charge (me)"

(Yamagata, 1997).

Hackers

In October 1991, Linus Torvalds announced on the Internet that he had written a "program for hackers by a hacker" (See Figure 2 again for the October 1991 announcement message). Hackers are people who enjoy "exploring the details of programmable systems and how to stretch their capabilities," and appreciate programming over theorizing (Jargon Dictionary, 2000). Torvalds encouraged people to look at his program and modify it for their own needs and to send him their code to add to the system. People who had been frustrated with the restricted features of Minix accepted Torvalds' invitation. By the end of the month, ten people had the system working on their machines (Torvalds, 1991, October 31). Some offered to work on different features of the program and began sending him contributions.

Within two months of Torvalds' October 1991 announcement, about thirty people had contributed close to 200 reports of errors and problems in running Linux, contributions of utilities to run under Linux, and drivers and features to add to Linux. When Torvalds released version 0.11 in December 1991, he credited contributions by three people in addition to himself (Torvalds, 1991, December 3). In version 0.13, released in February 1992, the majority of patches were written by people other than Torvalds. By July 1995, more than 15,000 people from 90 countries on five continents had contributed comments, bug reports, patches, and features [12]. Why did these people accept Torvalds' invitation to test and modify Linux?

One possible motivation for working on the Linux project is that hackers by nature enjoy solving interesting technical problems and building new programs (Raymond, 1999, p. 233). Torvalds labeled himself a hacker and said that he "enjoyed" developing the operating system in his October 1991 message and suggested that "somebody else might enjoy looking at it" (See Figure 2 again). To hackers, who are generally interested in "sexy" and "technically sweet" problems (Raymond, 1999, p. 72), an operating system represents an alluring challenge. Programming an operating system means a hacker has tamed the computer hardware and stretched its functionality. In his October announcement, Torvalds appealed to the hacker's need for such challenges when he asked if they were "without a nice project and just dying to cut [their] teeth on a OS."

People also wrote code to solve particular personal problems. Torvalds himself developed Linux so that he could run a Unix-like operating system on his own PC. In his October announcement Torvalds noted that Linux was available for anybody "to modify for [their] needs." When people accepted Torvalds' invitation to try the first Linux release, they wrote their own device drivers to support their choice of hardware and peripherals. For instance, a German programmer using Linux developed the German keyboard driver that was included in the December 1991 release (Thiel, 1991). To this day, people work on areas that they know and care about (Raymond 1999; p. 32). No one tells anyone what to work on [13].

However, whereas intrinsic pleasure and personal problem solving may be enough to motivate people to write Linux code for their own use, they do not satisfactorily explain why people contributed that code to a larger effort. People might contribute because they expect others will do so in return (Kollock, 1999; Raymond, 1999). When Torvalds announced his free operating system, he was also "interested in hearing from anybody who has written any of the utilities/library functions for minix" to add to his system. People posted bug reports hoping others would fix them; people contributed patches expecting that others would post patches to other problems with the system. The Gnu GPL ensured both that their contributions would be made accessible to everyone else and that everyone else's contributions would be accessible to them. This ensured the establishment of a stable 'generalized exchange system' (Kollock, 1999) in which people could expect returns to their contributions.

This generalized exchange system, however, could have broken down if everyone waited for someone else to contribute (Kollock, 1998). In the Linux case, programmers also contributed because they wanted to be known for writing good code (Raymond, 1999). Ackerman and Palen (1996) offered a similar explanation when they suggested that MIT undergraduates contribute to a voluntary online help system to develop their reputation as "clueful". An economic explanation suggested that accrued reputation has a positive impact on the programmers' careers (Lerner and Tirole, 2000). However, this account fails to explain the motivation of early hackers of the Linux operating system. The potential career opportunities to be gained from building up a reputation as a skilled programmer in the Linux operating system project were not present from the start.

Good code was acknowledged in a variety of ways. The separate management of production and development kernels meant that people contributing good code received rapid positive feedback as their contributions were incorporated in the development kernel in a short period of time. Some programmers left personal signatures within the source code so that others looking at the source code could recognize their work. The GPL ensured that freely contributed code would not be exploited by others. Torvalds also frequently acknowledged the contributions of others. In his original October 1991 announcement message, Torvalds credited another programmer for code he had used in the first release ("thanks for a nice and working system Earl <Chews>"). In his January 1992 release, he acknowledged the significant contributions of three other programmers (Torvalds, 1992, January 9). As the number of contributors grew, it became impossible for Torvalds to acknowledge all those who had contributed code with each release. With the official release of version 1.0 in 1994, Torvalds acknowledged and thanked more than 80 people by name (Torvalds, 1994, March 14). Version 1.0 was also the first version to include a 'credits' file that listed the various contributors and their role in developing and maintaining the Linux kernel. The programmers themselves were responsible for requesting to be added to the credits file if they considered themselves to be "worth mentioning" (Torvalds, 1993, December 21). In the early days, Torvalds credited programmers personally. The credits file marked a transition to a process in which programmers could credit themselves according to a shared understanding of what constituted creditable contributions.

However, whereas reputation as a skilled programmer was the most important motivation for contributing code, it was not the only way of gaining reputation. Credit could also come from significant contributions of documentation and other information useful to developers and users of Linux. In fact, in an early message Torvalds acknowledged a contributor for compiling the Linux information sheet (Torvalds, 1992, January 9), and stated that one did not "need to be a *kernel* developer to be on the credits list" (Torvalds, 1993, December 21).

Community

In July and August, 1991, Linus Torvalds posted messages to a Usenet group to which he belonged asking the group for help on a project. He asked where to find a particular set of operating system standards, what things group members liked and disliked about minix, and what features they would want in a free minix look alike. The August message received five replies within two days of its posting (See Figure 3 for the August message). Torvalds did not create his original project in social isolation. He was a member of an on-going active community of programmers - electronically organized in a Usenet group, comp.os.minix, with thousands of members [14]. It was to this group that Torvalds turned for advice, suggestions, and code. It was to this group that Torvalds announced his initial project. It was from this group that early Linux contributors were drawn. And it was out of this group that the first Linux group was created.

Writing code is a solitary activity and, as noted in the previous section, some people may have written Linux code just for the joy of it or to solve a personal problem with no thought of contributing to a larger endeavor. But the hackers described in the previous section did not just upload code to the Internet randomly. Their code was motivated, organized, and aggregated by and through features of the Linux electronic community. Two attributes of this community have been particularly important: its group communication structure and its role structure [15].

```
Date:          Sun, 25 Aug 1991 20:57:08 GMT
Reply-To:      INFO-MINIX@UDEL.EDU
Sender:        INFO-MINIX-ERRORS@PLAINS.NODAK.EDU
Comments:      Warning -- original Sender: tag was
info-minix-request@UDEL.EDU
From:          Linus Benedict Torvalds <torvalds@KLAAVA.HELSINKI.FI>
Subject:       What would you like to see most in minix?

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones. This has been brewing
since april, and is starting to get ready. I'd like any feedback on
things people like/dislike in minix, as my OS resembles it somewhat
(same physical layout of the file-system (due to practical reasons)
among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to
work. This implies that I'll get something practical within a few
months, and I'd like to know what features most people would want.
Any suggestions are welcome, but I won't promise I'll implement them
:-)

Linus (torvalds@kruuna.helsinki.fi)

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

Figure 3. Announcement of Torvalds' Project (August 1991)

Linux mailing lists and Usenet groups provided a map of the growing and changing Linux territory so that people could know what new code had been written, where to send their code or comments, and where to find information that interested them. The first Linux mailing list, Linux-activists, was created in October, 1991, and had about 400 subscribers by January, 1992. Today there are more than 700 Linux mailing lists [16]. Comp.os.linux was formed in March, 1992, the first of 14 comp.os.linux.* Usenet groups devoted to Linux formed in the next two years [17]. Beginning in 1994, an additional hierarchy of Linux groups, linux.* was formed, which by 1998 had 369 groups (See Table 2 for list of groups and mailing lists).

Hierarchy	Number of groups	Constituent Groups
comp.os.*	16	comp.os.linux.advocacy comp.os.linux.alpha comp.os.linux.announce (moderated: Mikko Rauhala) comp.os.linux.answers (moderated: Matt Welsh) comp.os.linux.apps comp.os.linux.development.apps

		comp.os.linux.development.system
		comp.os.linux.hardware
		comp.os.linux.m68k
		comp.os.linux.misc
		comp.os.linux.networking
		comp.os.linux.portable
		comp.os.linux.powerpc
		comp.os.linux.security
		comp.os.linux.setup
		comp.os.linux.x
Corporate groups[1]	At least 19	Linux.debian.* linux.redhat.*
Linux.act.*	60	Linux.act.kernel etc.
Linux.dev.*	53	linux.dev.doc etc.
Others	At least 120[2]	Newsgroups worldwide, Non-English language newsgroups, miscellaneous newsgroups in the alt.* hierarchy

Table 2. Linux Newsgroups [18]

The linux-kernel mailing list organizes the behavior of kernel programmers [19]. Feature freezes, code freezes, and new releases are announced on this list. Bug reports are submitted to this list. Programmers who want their code to be included in the kernel submit it to this list. Other programmers can then download it, test it within their own environment, suggest changes back to the author, or endorse it [20]. From June, 1995, to April, 2000, about 13,000 contributors posted almost 175,000 messages to the linux-kernel list. Here then is how individual behavior is organized into a product with social utility. Coordination of the individual programmers is achieved through the modular structure of the Linux kernel. Close collaboration among large numbers of programmers does not occur across the broad community, but may occur at times among smaller numbers of programmers working on the same module.

The previous two sections might convey the impression that the Linux development project consisted of Linus Torvalds and an undifferentiated band of happy hackers. In fact the community has developed a differentiated role structure that both reflects and supports its activities. The two most important roles within this community are *credited developer* and *maintainer.* Beginning with the v1.0 release in 1994, new releases of the kernel were accompanied by a "credits file," which publicly acknowledges people who have contributed substantial code to the kernel. The initial credits file listed 80 contributors. By the release of Linux 2.0 in June 1996, there were 190 acknowledged contributors. By July 2000, the list had grown to approximately 350 contributors from over 30 countries worldwide. The role of credited developer however is not restricted to people who are listed in the credits file. People who have contributed extensively to the Linux kernel but do not add themselves to the credits file are also recognized by the community.

The maintainer role was first formally acknowledged in February, 1996, when the maintainers file was announced (Cox, 1996, February 21). The first maintainer's file contained three names; today it contains 147 names. Designated maintainers are responsible for particular modules of the kernel. They review linux-kernel mailing list submissions (bug reports, bug fixes, new features) relevant to their modules, build them into larger patches, and submit the larger patches back to the list and to Torvalds directly [21]. Over the years Torvalds and the community have come to know and trust the technical competence of these maintainers, most of whom still "work on Linux for free and in their spare time" (Linux-kernel mailing list FAQ, http://www.tux.org/lkml). Indeed, one maintainer, Alan Cox, has the complete responsibility for overseeing the stable tree. Torvalds still announces major stable tree releases, but this is a *pro forma* gesture [22].

Table 3 displays the message contribution profile for the linux-kernel mailing list. Note that approximately 2% of the contributors contributed more than 50% of the messages, an indicator of the differentiated role structure and contribution profile of the community. Of these 254 contributors, 30% are credited developers and 19% are designated maintainers.

Messages	Contributors	% of Total (contributors)	Cumulative Message count	Unique Messages contributed	% of Total (messages)
Less than or eq 10	10925	84.07%	27861	27861	15.99%
11-100	1816	13.97%	81047	53186	30.52%
100+	134	1.03%	100046	18999	10.90%
200+	46	0.35%	111380	11334	6.50%
300+	16	0.12%	116943	5563	3.19%
400+	15	0.12%	123551	6608	3.79%
500+	10	0.08%	129034	5483	3.15%
600+	5	0.04%	132300	3266	1.87%
700+	5	0.04%	136050	3750	2.15%
800+	3	0.02%	138540	2490	1.43%
900+	2	0.02%	140377	1837	1.05%
1000+	18	0.14%	174257	33880	19.44%

Table 3. Message Contribution Profile [23]

Because the linux-kernel mailing list is the central organizing forum for kernel developers and supports a very heavy volume of message traffic, people need a clear understanding of how to contribute to the list. Maintainers of the linux-kernel mailing list FAQ create that understanding. The FAQ is the document that explains the rules of the road for kernel development to newcomers ("how do I make a patch?" "how do I get my patch into the kernel?"). It also reiterates the norms and values of the community ("A line of code is worth a thousand words. If you think of a new feature, implement it first, then post to the list for comments.") In addition to learning through the FAQ, newcomers can learn directly from the more experienced and skilled developers on the list through direct observation and questions posted to the list. Newcomers can develop into skilled Linux kernel programmers through such informal training (Brown and Duguid, 1991; Cox, 1998). Twelve people maintain the FAQ, all of whom are also credited developers or maintainers. The mailing list, with its thousands of contributors, differentiated role structure, dedicated role incumbents, and rules of the road, is more than just a list. As the FAQ explains, "Remember the list is a community."

Lessons for Organizations

Others have written about lessons from Linux for commercial software development projects (e.g., Raymond, 1999). Here we consider how factors important in the Linux case might apply more generally to distributed work in and across organizations (also see Markus, Manville and Agres, 2000). It might seem odd to derive lessons for formal organizations from a self-organizing volunteer activity. After all, the employment contract should ensure that people will fulfill their role obligations and act in the best interest of the organization. Yet, particularly in distributed work, employees must go beyond the letter of their job description to exhibit qualities found in the Linux developers: initiative, persistence, activism. We suggest that the enabling conditions for Linux (the Internet and open source) usefully support these conditions. We then consider how factors emphasized in each of the three versions of the Linux story (great man and task structure, incentives for contributors, and communities of practice) can facilitate organizational distributed work.

Clearly easy access to the Internet or its equivalent is a necessary precondition for the kind of distributed work represented by Linux. Developers used the Internet both for easy access to work products (to upload and download files) and for easy communication with other developers (to ask and answer questions, have discussions, and share community lore). Both capabilities are surely important. And they are simple. It is noteworthy that, despite the technical prowess of Linux developers, they relied upon only the simplest and oldest of Internet tools: file transfer, e-mail distribution lists, and Usenet discussion groups. Even with today's wider variety of more sophisticated Web-based tools, Linux developers continue to rely on these tools for coordinating their efforts. These tools are simple; they are available worldwide; they are reliable.

The organizational equivalent of copyleft is a second precondition for the kind of distributed work represented by Linux. Both the formal and informal reward and incentive systems must reward sharing and discourage hoarding (See Constant, Kiesler and Sproull, 1996, and Orlikowski, 1992, for discussions of incentives for information sharing in organizations). Moreover work products should be transparently accessible so that anyone can use and build upon good features and anyone can find and fix problems. We do not underestimate the difficulty of creating the equivalent of copyleft for organizational work products. Failing to do so, however, can hobble distributed work.

Does every successful large-scale distributed project require one "great person" to be in charge? Clearly this was important in the Linux case. Yet other successful open-source development projects have had different leadership models (e.g., Fielding, 1999; Wall, 1999). The capabilities that a singular leader brings to a project - clear locus of decision-making, singular vision, consistent voice - are important. But, in principle, they can be achieved through a variety of leadership models.

Task decomposition has been an important organizational principle for at least the past forty years (e.g., March and Simon, 1958). Modularity is an extreme form of task decomposition, one that may be particularly suited to distributed work. Modular task decomposition reduces coordination costs to an irreducible minimum. Moreover it allows for redundant development which can increase the probability of timely, high-quality solutions. That is, multiple groups can work on the same module independently of one another. The first (or best) solution is selected for adoption.

Parallel release structures that support both stability and rapid development are not commonly deployed in organizational development projects. More typical are linear structures or phased structures that hand off a project from developers to clients or maintainers. Organizations are typically viewed as having to manage a trade off between exploration and exploitation (March, 1991). On-going parallel release structures could generate both more innovation and more continuous improvement.

In the Linux case, people were motivated to work for pleasure and to improve their personal work situation. They were motivated to contribute their work to others for the same reasons and for reputational credit. Careful attention to motivation and incentive structures are extremely important in distributed work projects. "Careful" need not mean "heavy handed," however. Encouraging people to sign their work can be a low-cost, high-payoff motivator. Publicly naming contributors can be another. This approach is analogous to one employed in the Xerox system for technical service representatives in which they are encouraged to submit their "fixes" to machine problems that are not adequately handled by their formal documentation (Bell, Bobrow, Raiman and Shirley, 1998).

Finally, Linux developers were members of and supported by vigorous electronic communities of practice. Creating and sustaining such communities can importantly contribute to distributed work. Electronic communities require both (simple) computer tools and social tools. We discussed computer tools under enabling conditions, above. The social tools include differentiated roles and norms. It is not enough to enable electronic communication among people working on a distributed project. In a project of any size people must understand and take on differentiated electronic roles. These roles, with their corresponding obligations and responsibilities, should be explicitly designated and understood by all. Indeed, one category of community norms is the expectations associated with role behaviors. More generally, norms are the "rules of the road" for the particular electronic community. Because distributed projects cannot rely upon the tacit reinforcements that occur in face-to-face communications, persistent explicit reminders of norms are necessary in the electronic context (See Sproull and Patterson, 2000 for more on this topic).

The scope of computer-supported distributed work will continue to increase in organizations. The lessons of open-source software development are surely not applicable to all distributed work. Indeed one fruitful avenue for further work will be to delineate features of projects for which these lessons are more or less applicable. Still Linux and its kin will continue to offer fascinating cases of distributed work in their own right and fruitful sources of lessons for organizational distributed work.

About the Authors

Jae Yun Moon is a doctoral candidate in the program in Information Systems at the Stern School, New York University. Her research interests include collective action and coordination in online communities, organizational memory systems and knowledge management, and human computer interaction and consumer behavior in electronic commerce.
E-mail: jmoon@stern.nyu.edu

Lee Sproull is the Leondard N. Stern School Professor of Business at the Stern School, New York University. She is currently director of the Stern School Digital Economy Initiative, a comprehensive initiative combining education programs, research, and industry partnerships. She is an internationally-recognized sociologist whose resesarch centers on the implications of computer-based communication technologies for managers, organizations, communities, and society. She has conducted research in Fortune 500 firms, scientific communities, municipalities, universities, software development teams, households, and electronic groups. In all of these settings she has documented how technology induces changes in interpersonal interaction, group dynamics and decision-making, and organizational or community structure. She has been a visiting scholar at Xerox PARC, Digital Cambridge Research Lab, and Lotus Development Corporation and has published the results of her research in eight books and more than sixty articles.
E-mail: lsproull@stern.nyu.edu

Acknowledgments

We would like to thank participants in the Electronic Communities seminar at Stern (Spring 2000), participants at the NSF Distributed Work Workshop (August 2000), Sara Kiesler, Gloria Mark, Wanda Orlikowski, and Bob Sproull for their valuable comments on the paper. We would like to thank members of the Linux community, in particular John A. Martin and Matti Aarnio, for their help in understanding the community. An earlier draft of this paper was presented at the Distributed Work Workshop, Carmel, Calif., August 2000.

References

M.S. Ackerman and L. Palen, 1996. "The Zephyr help instance: Promoting ongoing activity in a CSCW system," *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '96)*, pp. 268-275.

G.T. Allison, 1971. *Essence of decision: Explaining the Cuban missile crisis.* Boston: Little, Brown.

D.G. Bell, D.G. Bobrow, O. Raiman, and M.H. Shirley, 1998. "Dynamic documents and situated processes: Building on local knowledge in field service," In: T. Wakayama, S. Kannapan, C.M. Khoong, S. Navathe, and J. Yates (editors). *Information and process integration in enterprises: Rethinking documents.* Norwell, Mass.: Kluwer, pp. 261-276.

S. Berinato, 1999. "Linux shows 25% annual growth rate," *ZdNet: eWeek* (1 April), accessed 11 July 2000 at http://www.zdnet.com/zdnn/stories/news/0,4586,1014254,00.html

J.S. Brown and P. Duguid, 1991. "Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation," *Organization Science*, volume 2, number 1, pp. 40-57.

Committee on Advanced Engineering Environments, National Research Council, 1999. *Advanced engineering environments: Achieving the vision, phase 1.* Washington, D.C.: National Academy Press.

D. Constant, S. Kiesler, and L. Sproull, 1996. "The Kindness of strangers: On the usefulness of weak ties for technical advice," Organization Science, volume 7, number 2, pp. 119-135.

A. Cox, 1996. "Maintainers and source submission procedures. Linux kernel source code (v.1.3.68)," (21 February 1996), accessed 15 July 2000 at ftp://tsx-11.mit.edu/pub/linux

A. Cox, 1998. "Cathedrals, bazaars and the town council," *Slashdot*, (13 October 1998), accessed 13 July 2000 at http://slashdot.org/features/98/10/13/1423253.shtml

B.J. Dempsey, D. Weiss, P. Jones, and J. Greenberg, 1999. "A quantitative profile of a community of open source Linux developers," (6 October 1999), accessed 11 July 2000 at http://metalab.unc.edu/osrt/develpro.html

D. Dougherty, 2000. "Interview transcript: Matt Welsh," *O'Reilly Network* (25 February 2000), accessed 18 June 2000 at http://www.oreillynet.com/pub/a/linux/2000/02/25/interview/welshtranscript.html

G. DuBois, 2000. "Open-source StarOffice earns early praise," *Zdnet: eWeek* (20 July), accessed 19 September 2000 at http://www.zdnet.com/eweek/stories/general/0,11011,2605874,00.html

R.T. Fielding, 1999. "Shared leadership in the Apache project," *Communications of the ACM*, volume 42, number 4, pp. 42-43.

Free Software Foundation, 1991. "GNU General Public License," accessed 22 June 2000 at http://www.gnu.org/copyleft/gpl.html

A. Gaffin and J. Heitkotter, 1994. "Usenet History," In: *EFF's extended guide to the Internet: A round trip through global networks, life in cyberspace, and everything* ... accessed 19 September 2000 at http://www.eff.org/papers/eegtti/eeg_toc.html#SEC88

A. Gopal, 1999. "Communication, processes and contracts in offshore software development," (19

November 1999), Unpublished doctoral dissertation proposal, Carnegie Mellon University, Pittsburgh.

Jargon Dictionary, 2000. accessed 12 September 2000 at http://info.astrian.net/jargon

J.L. King, R.E. Grinter, and J.M. Pickering, 1997. "The Rise and fall of Netville: The Saga of a cyberspace construction boomtown in the great divide," In: S. Kiesler (editor). *Culture of the Internet.* Mahwah, N.J.: Lawrence Erlbaum Associates, pp. 3-34.

P. Kollock, 1998. "Social dilemmas: The Anatomy of cooperation," *Annual Review of Sociology,* volume 24, pp. 183-214.

P. Kollock, 1999. "The Economies of online cooperation: Gifts and public goods in cyberspace," In: M.A. Smith and P. Kollock (editors). *Communities in cyberspace.* London: Routledge, pp. 220-239.

K. Kuwabara, 2000. "Linux: A Bazaar at the end of chaos," *First Monday,* volume 5, number 3 (March), accessed 13 July 2000 at http://firstmonday.org/issues/issue5_3/kuwabara/index.html

J. Lerner and J. Tirole, 2000. "The Simple economics of open source," (25 February 2000), accessed 13 July 2000 at http://www.people.hbs.edu/jlerner/simple.pdf

S. Levy, 1984. *Hackers: Heroes of the computer revolution.* Garden City, N.Y.: Anchor Press/Doubleday.

P. MacDonald, 1992. "Re: Is Linux reliable?" *comp.os.minix* (13 January 1992), accessed 27 June 2000 at http://listserv.nodak.edu/scripts/wa.exe?A0=minix-l&D=0

J.G. March, 1991. "Exploration and exploitation in organizational learning," *Organization Science,* volume 2, number 1, pp. 71-87.

J.G. March and H.A. Simon, 1958. *Organizations.* New York: Wiley.

M.L. Markus, B. Manville, and C.E. Agres, 2000. "What makes a virtual organization work?" *Sloan Management Review,* volume 42, number 1 (Fall), pp. 13-26.

R.P. Merges, 1997. "The End of friction? Property rights and contract in the "Newtonian" world of on-line commerce," *Berkeley Technology Law Journal,* volume 12, number 1, accessed 4 October 2000 at http://www.law.berkeley.edu/journals/btlj/articles/12_1/Merges/html/reader.html

E. Moglen, 1999. "Anarchism triumphant: Free software and the death of copyright, *First Monday,* volume 4, number 8 (August), accessed 5 October 2000 at http://firstmonday.org/issues/issue4_8/moglen/

G. Moody, 1997. "The Greatest OS that (n)ever was," *Wired,* volume 5, number 8 (August), pp. 122-125, 154-156, 158, 164, and accessed 27 June 2000 at http://www.wired.com/wired/5.08/linux_pr.html

Netscape Press Relations, 1998. "Netscape announces plans to make next-generation Communicator source code available free on the net," (22 January), accessed 5 October 2000 at http://www.netscape.com/newsref/pr/newsrelease558.html

T. O'Reilly, 1999. "Lessons from open-source software development," *Communications of the ACM,* volume 42, number 4, pp. 32-37.

W. Orlikowski, 1992. "Learning from Notes: Organizational issues in groupware implementation," *Proceedings of the 1992 Computer Supported Cooperative Work,* accessed 14 October 2000 at http://www.acm.org/pubs/citations/proceedings/cscw/143457/p362-orlikowski/

J. Ousterhout, 1999. "Free software needs profit," *Communications of the ACM,* volume 42, number 4, pp. 44-45.

D.E. Powell, 2000. ".comment: Judgment day for the GPL?" *LinuxPlanet* (26 June), accessed 4 October 2000 at http://www.linuxplanet.com/linuxplanet/reports/2000/1/

E.S. Raymond, 1999. *The Cathedral and the bazaar: Musings on Linux and open source by an accidental revolutionary.* Sebastopol, Calif.: O'Reilly & Associates.

P.H. Salus, 1994. *A Quarter century of UNIX.* Reading, Mass.: Addison-Wesley.

S. Shankland, 1999. "Red Hat shares triple in IPO," *CNET News.com* (11 August), accessed 16 October 2000 at http://www.canada.cnet.com/news/0-1003-200-345929.html

S. Shankland, 1998. "Linux shipments up 212 percent," *CNET News.com* (16 December), accessed 28 June 2000 at http://news.cnet.com/category/0-1003-200-336510.html

L. Sproull and J. Patterson, 2000. *Computer support for local community.* New York: New York University, Center for Information Intensive Organizations.

R. Stallman, 1999. "The GNU operating system and the free software movement," In: C. DiBona, S. Ockman, and M. Stone (editors). *Open sources: Voices from the open source revolution.* Sebastopol, Calif.: O'Reilly & Associates, pp. 53-70.

R. Stallman, 1998. "The GNU project," accessed 11 July 2000 at http://www.gnu.org/gnu/thegnuproject.html

A. Tanenbaum, 1992. "Unhappy campers," *comp.os.minix* (3 February 1992), accessed 5 October 2000 at http://listserv.nodak.edu/scripts/wa.exe?A2=ind9202A&L=minix-l&P=R5793&D=0

A. Tanenbaum, 1991. "Re: Most requested features in MINIX," *comp.os.minix* (4 February 1991), accessed 27 June 2000 at http://listserv.nodak.edu/scripts/wa.exe?A0=minix-l&D=0

W. Thiel, 1991. "Keyboard.S with German keyboard," *Linux-activists* (21 November 1991), accessed 27 June 2000 at ftp://tsx-11.mit.edu/pub/linux

L. Torvalds, 1999a. "The Linux edge," *Communications of the ACM,* volume 42, number 4, pp. 38-39.

L. Torvalds, 1999b. "The Linux edge," In: C. DiBona, S. Ockman, and M. Stone (editors). *Open sources: Voices from the open source revolution.* Sebastopol, Calif.: O'Reilly & Associates, pp. 101-111.

L. Torvalds, 1996. "Re: Unices are created equal, but ..." *Linux-kernel* (16 April 1996), accessed 15 October 2000 at http://www.uwsg.iu.edu/hypermail/linux/kernel/9604.1/0771.html

L. Torvalds, 1996. "Re: torvalds@cs.helsinki.fi = /dev/null ???" *Linux-kernel* (19 February 1996), accessed 5 October 2000 at http://www.uwsg.iu.edu/hypermail/linux/kernel/9602/0529.html

L. Torvalds, 1995. "Re: Shared interrupts - PCI," *Linux-kernel* (29 June 1995), accessed 15

October 2000 at http://www.uwsg.iu.edu/hypermail/linux/kernel/9506/0194.html

L. Torvalds, 1994. "Linux code freeze," *Linux Journal* (May), accessed 26 June 2000 at http://www2.linuxjournal.com/lj-issues/issue1/2733.html

L. Torvalds, 1994. "New order," *Linux-activists* (15 March 1994), accessed 27 June 2000 at ftp://tsx-11.mit.edu/pub/linux

L. Torvalds, 1994. "Linux 1.0 - A better UNIX than Windows NT," *comp.os.linux.announce* (14 March 1994), accessed 27 June 2000 at http://www.cs.helsinki.fi/u/mjrauhal/linux/cola.html

L. Torvalds, 1993. "Re: Credits file," *Linux-activists* (21 December 1993), accessed 27 June 2000 at ftp://tsx-11.mit.edu/pub/linux

L. Torvalds, 1992. "Re: Writing an OS - questions!!" *Linux-activists* (5 May 1992), accessed 10 November 2000 at http://www.li.org/linuxhistory.php

L. Torvalds, 1992. "Linux 0.95," *Linux-activists* (8 March 1992), accessed 27 June 2000 at ftp://tsx-11.mit.edu/pub/linux

L. Torvalds, 1992. "Linux information sheet" (non monthly posting), *comp.os.minix* (9 January 1992), accessed 14 July 2000 at http://listserv.novak.edu/archives/minix-l.html

L. Torvalds, 1991. "Last call for diffs for 0.11," *Linux-activists*, (3 December 1991), accessed 27 June 2000 at: ftp://tsx-11.mit.edu/pub/linux

L. Torvalds, 1991. "Re: [comp.os.minix] Free minix-like kernel sources for 386-AT," *comp.os.minix* (31 October 1991), accessed 12 July 2000 at http://listserv.nodak.edu/archives/minix-l.html

L. Torvalds, 1991. "What would you like to see most in minix," *comp.os.minix* (25 August 1991), accessed 27 June 2000 at http://listserv.nodak.edu/scripts/wa.exe?A0=minix-l&D=0

V. Vallopillil, 1998. "Open source software: A (new?) development methodology," (11 August 1998), accessed 28 June 2000 at http://www.opensource.org/halloween/halloween1.html

Virtual Worlds Group, 2000. "Virtual Worlds Web Site FAQ," (24 August 2000), accessed 5 October 2000 at http://www.vworlds.org/faq.asp

L. Wall, 1999. "The Origin of the camel lot in the breakdown of the bilingual Unix," *Communications of the ACM*, volume 42, number 4, pp. 40-41.

B.M. Wiesenfeld, S. Raghuram, and R. Garud, 1999. "Communication patterns as determinants of organizational identification in a virtual organization," *Organization Science*, volume 10, number 6, pp. 777-790.

H. Yamagata, 1997. "The Pragmatist of free software: Linus Torvalds interview," (30 September 1997), accessed 10 November 2000 at http://www.tlug.gr.jp/linus.html

R. Young, 1999. "Giving it away: How Red Hat software stumbled across a new economic model and helped improve an industry," In: C. DiBona, S. Ockman, and M. Stone (editors). *Open sources: Voices from the open source revolution.* Sebastopol, Calif.: O'Reilly & Associates, accessed 14 October 2000 at http://www.oreilly.com/catalog/opensources/book/young.html

Notes

1.They have also led to an increase in individual workers working at home or on the road (Wiesenfeld, Raghuram, and Garud, 1999). And they have led to an increase in outsourcing and offshoring (Gopal, 1999). Neither of these topics is the focus of this paper.

2. The bulk of the analysis of the Linux kernel development is based on archives of newsgroups and mailing lists that were devoted to the discussion of Linux development, and documentation within the Linux kernel source code. This analysis was corroborated when necessary through information obtained through published accounts of interviews with Linux kernel programmers, as well as first-hand e-mail contact with several active developers in the Linux community.

3. For analysis of the business implications of Linux see Young (1999).

4. Other software projects which have been developed by a group of volunteer programmers distributed geographically include the Apache Web server (Fielding, 1999), Perl programming language (Wall, 1999), GNU Emacs editor (Stallman, 1999), the fetchmail e-mail client program (Raymond, 1999), and the Tcl/tk scripting language (Ousterhout, 1999). A large proportion of Internet infrastructure is also the product of such collaborations. Sendmail, which enables people to send and receive e-mail, and BIND (Berkeley Internet Name Domain), the technology that makes it possible to navigate Web pages using natural language addresses instead of numbers, are among these (O'Reilly, 1999). Several commercial programs have also adopted an open source collaborative development style. These include but are not limited to Netscape Mozilla, Sun Microsystems Star Office and Microsoft Virtual Worlds (DuBois, 2000; Netscape Press Relations, 1998; Virtual Worlds Group, 2000).

5. The source code of a software program is a set of instructions that can be understood and modified by programmers. Compiling the source code produces binary code that is executable on the platform for which it was compiled. However, such code is not readily comprehended by people and thus cannot be usefully modified by them.

6. Usenet is a system of interconnected computer networks in which people form 'newsgroups' to discuss themes of common interest. Newsgroups function as a selective broadcasting system - people post to newsgroups where they hope to find readers with a common interest.

7. The enforceability of the GPL has not yet been determined in the legal courts. However, the GPL as a statement of informal norms and good practices does carry power, even with companies, which tend to honor the GPL (Powell, 2000). The FSF also polices the proper use of software code that has been distributed under GPL license. Detailed exposition of the debate surrounding the legal enforceability of the Copyleft licenses is beyond the scope of this chapter. For a view that GPL is not legally enforceable, see Merges, 1997; for an opposing view see Moglen, 1999.

8. In response to "most requested features in Minix," replies from the professor included "never." "No. too much hair. Mucks up kernel and MM too much." "Ha, ha" (Tanenbaum, 1991, February 4).

9. The modular architecture was in fact one of the reasons why it was easier to implement new features - unlike Minix, the features did not need to be in perfect coordination with the rest of the kernel for them to be incorporated into a working kernel for others to test (MacDonald, 1992). This characteristic of the kernel was greatly enhanced with the addition of loadable kernel modules in Linux 2.0 in early 1996.

10. Torvalds did not invent the concept of parallel release series. But he was the first to open his

development series to the entire world.

11. Versions 2.2.x and 2.3.x were still current as of May 2000.

12. We identified the contributors by extracting headers of archived mail and newsgroup messages. Although some people with multiple e-mail addresses were resolved manually, some might have been missed in the process.

13. In the early days, contributors were mostly hackers interested in working on an operating system for fun. Even though commercial firms also develop drivers and products for the Linux system, only 6% of maintained kernel modules and drivers (8/129) are maintained by people whose job requires it as indicated for Linux version 2.3, released in May 1999. An interview study of developers found that only 10% of those interviewed reported that their job in some way involved Linux (Kuwabara, 2000, and personal communication).

14. In 1992 there were 43,000 readers of the newsgroup (Tanenbaum, 1992, February 3). By 1995 that number had fallen to 25,000 (Minix information sheet, http://www.cs.vu.nl/~ast/minix.html).

15. Other important attributes and practices, which we omit because of space constraints, include norms, boundary maintenance, generational reproduction, and socialization.

16. Linux mailing lists (http://oslab.snu.ac.kr/~djshin/linux/mail-list/index.shtml), Linux Resource Exchange (http://www.linuxrx.com/Lists/Lists.perl).

17. The first Linux Usenet group was actually alt.os.linux, which was formed in January, 1992, to move the growing Linux discussion off of the minix group.

18. The list of newsgroups was taken on April 25, 2000 from the Newsgroups page of the Linux Portal Site (http://www.linuxlinks.com/Newsgroups/Miscellaneous) and the Liszt Newsgroups site (http://www.liszt.com/news).

19. Prior to the creation of the linux-kernel mailing list, the first Linux kernel mailing list, *Linux activists*, served this organizing function.

20. Programmers are cautioned not to send code via e-mail directly to Torvalds because he will be unlikely to see it (Linux-kernel mailing list FAQ, http://www.tux.org/lkml/#s1-14).

21. As Torvalds explained, "What happens is that if I get a patch against something where I'm not the primary developer (against a device driver or a filesystem I have nothing to do with, for example), I am less likely to react to that patch if it doesn't come from the person who is responsible for that particular piece of code. If I get a networking patch, for example, it's a _lot_ more likely to get a reaction if it is from Alan Cox, and then it usually goes in with no questions asked. Similarly ..., if it is a patch against the SCSI subsystem, I much prefer to have it from somebody I know works on SCSI (Eric, Drew, Leonard...)" (Torvalds, 1996 February 19).

22. Red Hat, a Linux distribution firm, funds Cox in addition to other key Linux developers (Shankland, 1999). Thus Cox's role in the Linux community is in large measure subsidized by Red Hat.

23. The source is the linux-kernel mailing list, from June 1995 to April 2000.

Editorial history

Paper received 25 October 2000; accepted 1 November 2000.

Contents Index

Essence of Distributed Work: The Case of the Linux Kernel by Jae Yun Moon and Lee Sproull
First Monday, volume 5, number 11 (November 2000),
URL: http://firstmonday.org/issues/issue5_11/moon/index.html