Documented Unix Facilities Over 48 Years

Diomidis Spinellis
Department of Management Science and Technology
Athens University of Economics and Business
Athens, Greece
dds@aueb.gr

ABSTRACT

The documented Unix facilities data set provides the details regarding the evolution of 15 596 unique facilities through 93 versions of Unix over a period of 48 years. It is based on the manual transcription of early scanned documents, on the curation of text obtained through optical character recognition, and on the automatic extraction of data from code available on the Unix History Repository. The data are categorized into user commands, system calls, C library functions, devices and special files, file formats and conventions, games et. al., miscellanea, system maintenance procedures and commands, and system kernel interfaces. A timeline view allows the visualization of the evolution across releases. The data can be used for empirical research regarding API evolution, system design, as well as technology adoption and trends.

CCS CONCEPTS

• **Software and its engineering → Software evolution**;

ACM Reference Format:
Diomidis Spinellis. 2018. Documented Unix Facilities Over 48 Years. In *MSR '18: 15th International Conference on Mining Software Repositories, May 28–29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3196398.3196476

1 INTRODUCTION

The Unix operating system is being continuously developed from the same code base for almost over half a century. It stands out as a major engineering artefact due to its exemplary design, its numerous technical contributions, its impact, its development model, and its widespread use [2, pp. 27–29], [9]. The design of the Unix programming environment, which nowadays offers thousands of tools and libraries, has been characterized as offering unusual simplicity, power, and elegance [5, 7]. Consequently, empirical data regarding how the facilities Unix provides grew and changed over time can be used for empirical research on API evolution, system design, as well as technology adoption and trends.

Although one can study a system's evolution through its source code [1, 10], the very large size of modern systems can hinder the recognition of the relevant parts. Fortunately, another avenue is available for studying Unix systems, namely their documentation. From the first version of the Unix system until today, every release is accompanied by a complete reference manual, where all provided facilities (commands, APIS, file formats, and device drivers) are neatly organized into several corresponding sections (see Table 1). The central role of the reference manual in the Unix system is evidenced by the fact that early Unix versions coming out of AT&T Bell Labs were named after the edition of the accompanying manual. Some early editions of the manuals have not survived in a machine-readable format, but most are available in text markup that can be processed through scripts to extract relevant data.

The data set presented here is based on the printed and machine-readable Unix reference manuals released over a period of 48 years. It documents the evolution of 15 596 facilities through 93 versions of Unix. Section 2 outlines the provided data, Section 3 describes how the data were produced, and Section 4 sketches two examples of how the data can be used for quantitative and qualitative empirical studies.

ACM ISBN 978-1-4503-5716-6/18/05…$15.00
https://doi.org/10.1145/3196398.3196476

2 UNIX RELEASES AND THEIR FACILITIES

The primary data are made available in the form of 93 text files containing 405 726 records. The files are named after the associated Unix release, following the tags and branches nomenclature established in the Unix History Repository [9]. A record is a text line with tab-separated fields. Each record contains the number of the Unix manual section associated with a facility (1–9; see Table 1) followed by the facility's name, optionally followed by a URI identifying the facility's documentation in *troff* markup [6]. In total, the set contains data about 15 596 facilities pointing to 193 781 unique URIS, identifying 48 250 distinct manual page instances.

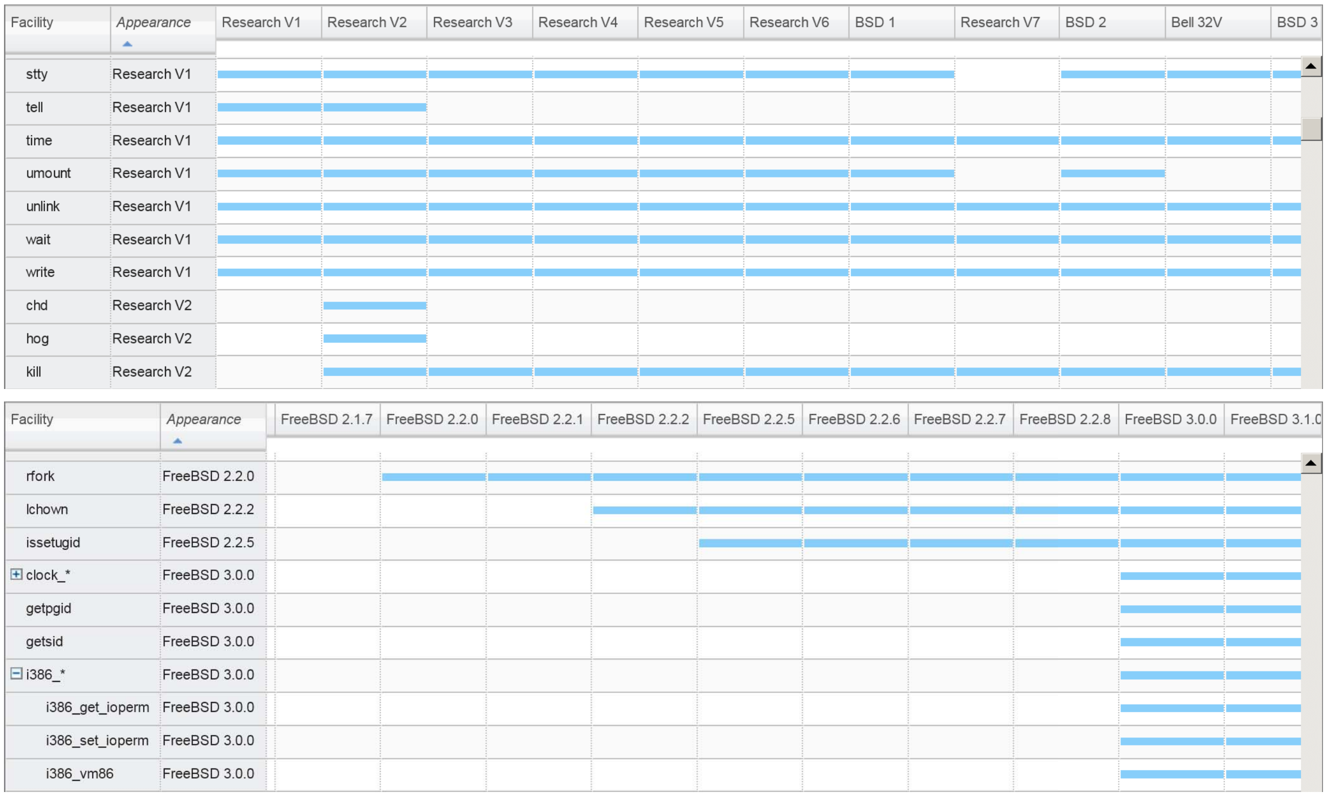
As an example, the following lines show the documentation files associated with the label command (:), the archiver (*ar*), and the assembler (*as*), as documented in Section I of the 1973 Third Edition Unix manual.

```
1       :       Research-V3/man/man1/:.1
1       ar      Research-V3/man/man1/ar.1
1       as      Research-V3/man/man1/as.1
```

By prepending the Unix History Repository GitHub permalink base URL "https://github.com/dspinellis/unix-history-repo/blob" to an entry's URI, one can obtain a URL for viewing the documentation markup source code for the corresponding entry.

A separate text file, named *timeline* associates each of the data files with the year, month, and day of the corresponding release. For instance, the following entries of the *timeline* file list the dates associated with the Sixth and Seventh Research Editions and the first Berkeley Software Distribution.

```
Research-V6 1975 07 18
BSD-1 1978 02 01
Research-V7 1979 08 26
```



Facility	Appearance	Research V1	Research V2	Research V3	Research V4	Research V5	Research V6	BSD 1	Research V7	BSD 2	Bell 32V	BSD 3
stty	Research V1	▬	▬	▬	▬	▬	▬	▬		▬	▬	▬
tell	Research V1	▬	▬									
time	Research V1	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬
umount	Research V1	▬	▬	▬	▬	▬	▬	▬		▬		
unlink	Research V1	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬
wait	Research V1	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬
write	Research V1	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬
chd	Research V2		▬									
hog	Research V2		▬									
kill	Research V2		▬	▬	▬	▬	▬	▬	▬	▬	▬	▬

Facility	Appearance	FreeBSD 2.1.7	FreeBSD 2.2.0	FreeBSD 2.2.1	FreeBSD 2.2.2	FreeBSD 2.2.5	FreeBSD 2.2.6	FreeBSD 2.2.7	FreeBSD 2.2.8	FreeBSD 3.0.0	FreeBSD 3.1.
rfork	FreeBSD 2.2.0		▬	▬	▬	▬	▬	▬	▬	▬	▬
lchown	FreeBSD 2.2.2				▬	▬	▬	▬	▬	▬	▬
issetugid	FreeBSD 2.2.5					▬	▬	▬	▬	▬	▬
⊞ clock_*	FreeBSD 3.0.0									▬	▬
getpgid	FreeBSD 3.0.0									▬	▬
getsid	FreeBSD 3.0.0									▬	▬
⊟ i386_*	FreeBSD 3.0.0									▬	▬
i386_get_ioperm	FreeBSD 3.0.0									▬	▬
i386_set_ioperm	FreeBSD 3.0.0									▬	▬
i386_vm86	FreeBSD 3.0.0									▬	▬

Figure 1: Timeline view of system call evolution in Research Unix Editions and early BSD distributions (top) and early FreeBSD releases (bottom)

Table 1: Manual Sections and Documented Facilities

Section Number and Title		Elements	URIs
1	User commands	1 494	34 836
2	System calls	457	12 472
3	C library functions	8 643	70 931
4	Devices and special files	1 224	25 111
5	File formats and conventions	315	9 208
6	Games et. al.	187	2 322
7	Miscellanea	106	1 779
8	System maintenance procedures and commands	995	26 990
9	System kernel interfaces	2 173	14 621

As a tool for performing qualitative studies (see Section 4.2), a web site[1] provides a timeline view of each manual section. The horizontal axis of the view lists all tracked releases ordered by the date of their appearance. The vertical axis lists all facilities (e.g. commands or system calls) ordered by the release in which they first appeared and then alphabetically. The timeline view can be scrolled horizontally to move along releases and vertically to move across facilities. Facilities can be ordered alphabetically or by their release's date. The names of facilities and releases, and the release where a facility first appeared do not scroll so as to provide the required context. Where possible, the timeline is hyperlinked to the corresponding manual pages, which are displayed using *manview*[2] and *jroff*.[3] Two examples, depicting system call evolution in the Research Unix Editions and early BSD and FreeBSD releases can be seen in Figure 1.

Where many facilities share the same prefix, they are grouped together and can be expanded and collapsed through a corresponding icon. This can be seen in some FreeBSD 3.0 system calls listed in the bottom part of Figure 1. The calls prefixed with `clock_` appear collapsed, while those prefixed with `i386_` are expanded. In total, 542 parent nodes house 7 597 collapsed entries.

3 DATA GENERATION

The data set was generated by processing data available in diverse formats through manual labour and custom scripts.

3.1 Data Provenance

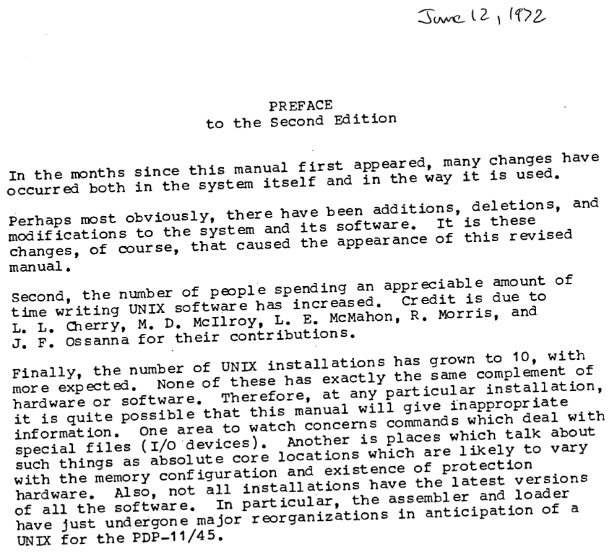
The data sources used are the printed versions of Unix manuals available on the Unix Heritage Society archive [11, 12] and the Unix History Repository [9]. The repository makes available the history and evolution of the Unix on GitHub,[4] covering the period from its inception in 1970 as a 13 thousand line unnamed prototype written

[1]https://dspinellis.github.io/unix-history-man/

[2]https://github.com/dspinellis/manview

[3]https://github.com/roperzh/jroff

[4]https://github.com/dspinellis/unix-history-repo


June 12, 1972

PREFACE
to the Second Edition

In the months since this manual first appeared, many changes have occurred both in the system itself and in the way it is used.

Perhaps most obviously, there have been additions, deletions, and modifications to the system and its software. It is these changes, of course, that caused the appearance of this revised manual.

Second, the number of people spending an appreciable amount of time writing UNIX software has increased. Credit is due to L. L. Cherry, M. D. McIlroy, L. E. McMahon, R. Morris, and J. F. Ossanna for their contributions.

Finally, the number of UNIX installations has grown to 10, with more expected. None of these has exactly the same complement of hardware or software. Therefore, at any particular installation, it is quite possible that this manual will give inappropriate information. One area to watch concerns commands which deal with special files (I/O devices). Another is places which talk about such things as absolute core locations which are likely to vary with the memory configuration and existence of protection hardware. Also, not all installations have the latest versions of all the software. In particular, the assembler and loader have just undergone major reorganizations in anticipation of a UNIX for the PDP-11/45.

Figure 2: Scanned page from the Second Edition Unix Manual

in PDP-7 assembly language, to 2018 as a widely-used 32 million line production-quality operating system. It has been created by synthesizing with custom software 24 snapshots of systems developed at Bell Labs, the University of California at Berkeley, and the 386BSD team, two legacy repositories, and the modern repository of the open source FreeBSD system.

The data's timeline starts with the so-called "Research" editions that came out of Bell Labs, continues with the Berkeley Software Distributions, and finishes with versions of the FreeBSD operating system distribution that continues its development until today. We cannot study Unix versions that derive from the Research editions via AT&T System V, such as Solaris, AIX, and HPUX, because most of the corresponding code remains proprietary. Also, the evolution of Research editions into *Plan 9* [8] was not examined, due to the system's limited adoption. Other systems deriving from the BSD source code base are NetBSD, which focuses on widespread architecture portability, especially among embedded devices, and OpenBSD, which focuses on security. Although FreeBSD, NetBSD, and OpenBSD differ in terms of vision and technologies, all frequently exchange among them code and ideas. The provided data concern the evolution in the popular FreeBSD line, to capitalize on the author's inside knowledge of FreeBSD.

3.2 Data Processing

The data set was generated using three methods: manual typing and editing, bespoke scripts, and a general-purpose script.

Some early editions of the Unix manual are only available as scanned documents (see Figure 2 containing the famous "the number of UNIX installation has grown to 10, with more expected" quote). The corresponding data for those were created from the text made available through optical character recognition and then hand-edited to correct errors, such as the replacement of the letter l (el) for the digit 1 (one). Given the small number of facilities in the early editions of Unix, some of the data sets were used as a base to manually type-in and edit data corresponding to a previous or next edition.

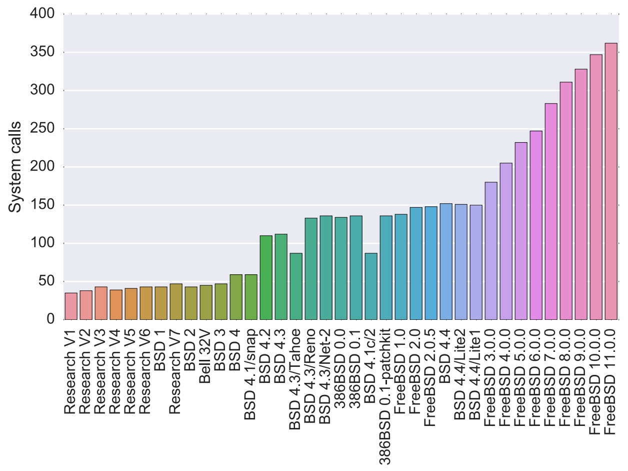


Figure 3: Evolution in the number of system calls across key releases

In releases of Unix where the source code is available, the supported facilities can be obtained by processing the source code tree. However, in the first 30 years of the system's lifetime the manual pages tended to move from one place of the source code tree to another. For this reason, around thirty custom shell scripts were written to generate and verify the list of facilities for 17 such releases. In some cases the results needed further cleaning by hand. The data set's generation steps and commands are documented in the form of Git commit comments in the data files for these releases. For modern (mainly FreeBSD) versions the data are obtained through a 130-line shell script that lists the available facilities for each major FreeBSD release.

A separate 57-line shell script is used to create the timeline of releases based on the most recent Unix History Repository commit for each release. The timeline view is created through a 506-line Perl script, 121 lines of hand-written JavaScript, and 926 thousand lines of auto-generated JavaScript utilising the SlickGrid control.[5]

4 USING THE DATA AND ITS VISUALIZATION

The Unix facilities evolution data can be used both in their raw format for quantitative analysis, and through their timeline visualization to obtain qualitative insights. Possible application areas include empirical research regarding API evolution, system design, as well as technology adoption and trends.

4.1 The Evolution of System Calls and File Formats

As two examples of how the provided data set can be used to perform quantitative studies, consider the evolution in the number of system calls and of documented file formats across key system releases.

In the evolution in the number of system calls (Figure 3) three periods can be readily discerned. While the system was developed

[5]https://github.com/andrewr88/SlickGrid

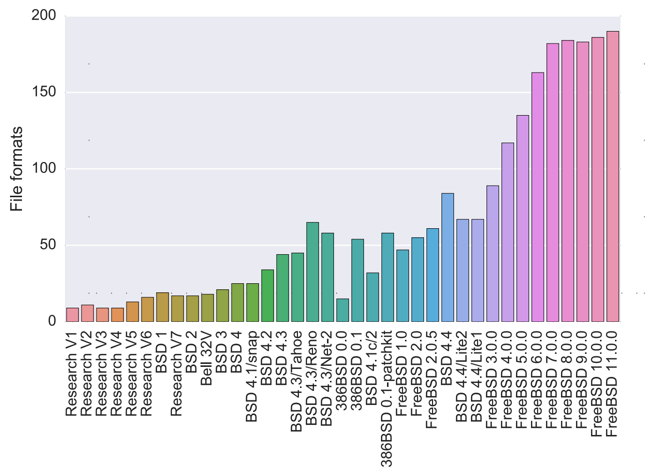


Figure 4: Evolution in the number of file formats across key releases

at Bell Labs and over the early Berkeley Editions (which did not include a modified kernel) the number of system calls remained relatively stable. This number swelled substantially with the support of local and remote inteprocess communication and the Internet protocol family (ARP, IP, TCP, UDP, and ICMP) in 4.2 BSD. Then another period stability followed, until the FreeBSD releases which, as one can see, began to introduce more system calls in each release. The reasons and effects associated with this increase require further qualitative examination, but could be correlated with the pressure for increased performance, portability requirements, a wider user base, or competition from similar Linux facilities.

The evolution of documented file formats (Figure 4) paints a very different picture. Files with a documented format, such as the password file or the domain name system resolver configuration, are often used in Unix as a way to provide facilities with minimal API support. The files can be processed by a variety of text tools [3] and modified using any programmer editor. Although their number appears to be increasing for decades, this increase appears to have almost stopped from FreeBSD 5.0 and onwards. Explanations for this trend could be that this particular method can no longer support the requirements of modern complex, large, and sophisticated systems or that data size issues and associated performance requirements lead to the use of other data storage mechanisms, such as relational databases.

4.2 API Evolution

As an example of qualitative insights that can be obtained using the data set's timeline view, consider the examination of deprecated system calls. System call deprecation can break backward compatibility, but can help keeping the system's API clean and orthogonal. It would therefore be interesting to see these forces in action.

Deprecated calls can be seen in the tabular view as lines that stop extending to the right, as is e.g. the case with the *tell* system call at the top part of Figure 1. By scrolling through the system call evolution table it becomes evident that when the Fifth Edition Unix got released, a number of system calls became deprecated. These included *cemt*, *fpe*, *ilgins*, *intr*, *quit*, and *rele*, By reading the Third Edition manual, it becomes evident that all these calls deal with interrupts. Three catch traps stemming from the execution of emulated (*cemt*) and illegal (*ilgins*) instructions as well as floating point exceptions (*fpe*). Others, control how interrupts are handled: they catch or inhibit keyboard-generated interrupts (*intr*) or quit (*quit*) signals and they implement timer-runout swaps (*rele*).

A further look at the timeline depiction of system call evolution also indicates that the Fourth Unix Edition introduced the *signal* system call. This is a more general facility, which in that edition could handle twelve different signals, including the ones that were previously handled by dedicated system calls. Significantly, the *signal* system call unifies the handling of hardware-triggered traps (*cemt*, *ilgins*, *fpe*, *emt*), with that of software-generated ones (*intr*, *quit*). Consequently, one can reason that the introduction of *signal* is a case of system call refactoring, where the handling of several special cases was subsumed by a more abstract facility. Interestingly, the table also shows that *signal* co-existed with the less general system calls in the Fourth Edition; an early example of gradual deprecation to avoid the instability caused by the changes' *ripple effects* [4].

Data Availability. The data set and its generation scripts are available on GitHub: https://github.com/dspinellis/unix-history-man/.

Acknowledgements. The project associated with this work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732223.

REFERENCES

[1] Vaggelis Atlidakis, Jeremy Andrus, Roxana Geambasu, Dimitris Mitropoulos, and Jason Nieh. 2016. POSIX Abstractions in Modern Operating Systems: The Old, the New, and the Missing. In *Proceedings of the Eleventh European Conference on Computer Systems*. ACM, 19.

[2] Narain Gehani. 2003. *Bell Labs: Life in the Crown Jewel.* Silicon Press, Summit, NJ.

[3] Brian W. Kernighan and Rob Pike. 1984. *The UNIX Programming Environment.* Prentice Hall, Englewood Cliffs, NJ.

[4] Tyler McDonnell, Baishakhi Ray, and Miryung Kim. 2013. An Empirical Study of API Stability and Adoption in the Adroid Ecosystem. In *Proceedings of the 2013 IEEE International Conference on Software Maintenance (ICSM '13).* IEEE Computer Society, Washington, DC, USA, 70–79. https://doi.org/10.1109/ICSM.2013.18

[5] M. D. McIlroy, E. N. Pinson, and B. A. Tague. 1978. UNIX Time-Sharing System: Foreword. *The Bell System Technical Journal* 57, 6 (July-August 1978), 1899–1904.

[6] J. F. Ossanna. 1979. NROFF/TROFF User's Manual. In *UNIX Programmer's Manual. Volume 2—Supplementary Documents* (seventh ed.). Bell Telephone Laboratories, Murray Hill, NJ.

[7] R. Pike and Brian W. Kernighan. 1984. Program Design in the UNIX System Environment. *AT&T Bell Laboratories Technical Journal* 63, 8 (Oct. 1984), 1595–1606.

[8] Rob Pike, Dave Presotto, Sean Dorward, Bob Flandrena, Ken Thompson, Howard Trickey, and Phil Winterbottom. 1995. Plan 9 from Bell Labs. *Computing Systems* 8, 2 (1995), 221–254.

[9] Diomidis Spinellis. 2017. A Repository of Unix History and Evolution. *Empirical Software Engineering* 22, 3 (2017), 1372–1404. https://doi.org/10.1007/s10664-016-9445-5

[10] Diomidis Spinellis, Panos Louridas, and Maria Kechagia. 2016. The Evolution of C Programming Practices: A Study of the Unix Operating System 1973–2015. In *ICSE '16: Proceedings of the 38th International Conference on Software Engineering*, Willem Visser and Laurie Williams (Eds.). Association for Computing Machinery, New York, 748–759. https://doi.org/10.1145/2884781.2884799

[11] Warren Toomey. 2009. The Restoration of Early UNIX Artifacts. In *Proceedings of the 2009 USENIX Annual Technical Conference (USENIX'09).* USENIX Association, Berkeley, CA, USA, 20–26.

[12] Warren Toomey. 2010. First Edition Unix: Its Creation and Restoration. *IEEE Annals of the History of Computing* 32, 3 (July/Sept. 2010), 74–82. https://doi.org/10.1109/MAHC.2009.55