Empir Software Eng (2017) 22:1372–1404
DOI 10.1007/s10664-016-9445-5

A repository of Unix history and evolution

Diomidis Spinellis[1]

Published online: 11 August 2016

Abstract The history and evolution of the Unix operating system is made available as a revision management repository, covering the period from its inception in 1972 as a five thousand line kernel, to 2016 as a widely-used 27 million line system. The 1.1GB repository contains 496 thousand commits and 2,523 branch merges. The repository employs the commonly used Git version control system for its storage, and is hosted on the popular GitHub archive. 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. In total, 973 individual contributors are identified, the early ones through primary research. The data set can be used for empirical research in software engineering, information systems, and software archaeology.

Keywords Software archeology · Unix · Configuration management · Git

1 Introduction

The Unix operating system stands out as a major engineering breakthrough due to its exemplary design, its numerous technical contributions, its impact, its development model, and its widespread use (Gehani 2003, pp. 27–29). The design of the Unix programming

Communicated by: Romain Robbes, Martin Pinzger and Yasutaka Kamei

The work has been partially funded by the Research Centre of the Athens University of Economics and Business, under the Original Scientific Publications framework (project code EP-2279-01) and supported by computational time granted from the Greek Research & Technology Network (GRNET) in the National HPC facility — ARIS — under project ID PA003005-CDOLPOT.

✉ Diomidis Spinellis
dds@aueb.gr

[1] Department of Management Science and Technology, Athens University of Economics and Business, Patision 76, 104 34 Athens, Greece

environment has been characterized as one offering unusual simplicity, power, and elegance (McIlroy et al. 1978; Pike and Kernighan 1984). On the technical side, features that can be directly attributed to Unix or were popularized by it include (Ritchie and Thompson 1978; Ritchie 1978; Johnson and Ritchie 1978):

– the portable implementation of the kernel in a high level language;
– a hierarchical file system;
– compatible file, device, networking, and inter-process I/O;
– the pipes and filters architecture;
– virtual file systems; and
– the shell as a user-selectable regular process.

A large community contributed software to Unix from its early days (Ritchie 1984; Salus 1994, pp. 65–72). This community grew immensely over time and worked using what are now termed open source software development methods (Raymond 2003, pp. 440–442; McKusick 1999, p. 46). Unix and its intellectual descendants have also helped the spread of:

– the C (Ritchie et al. 1978; Rosler 1984; Ritchie 1993) and C++ (Stroustrup 1984, 1994) programming languages;
– parser and lexical analyzer generators (Johnson and Lesk 1978) — *yacc* (Johnson 1975), *lex* (Lesk 1975);
– software development environments (Dolotta et al. 1978),
– document preparation tools (Kernighan et al. 1978) and declarative markup — *troff* (Ossanna 1979; Kernighan 1982), *eqn* (Kernighan and Cherry 1974), *tbl* (Lesk 1979b), the *mm* macros (Mashey and Smith 1976);
– scripting languages — *awk* (Aho et al. 1979), *sed* (McMahon 1979), *Perl* (Wall and Schwartz 1990);
– TCP/IP networking (Stevens 1990); and
– configuration management systems — *SCCS* (Rochkind 1975), *RCS* (Tichy 1982), *Subversion*, *Git*.

Unix systems also form a large part of the modern internet infrastructure and the web.

The importance of Unix as an engineering artefact motivates the preservation of its development history, which can then be used for empirical research in software engineering, information systems, and software archeology.

The availability of Unix source code has changed over the years. In the 1970s, when Unix came of out Bell Labs and became widely known in the scientific community (Ritchie and Thompson 1974), AT&T was still operating under a 1956 “consent decree” entered by Judge Thomas F. Meaney (Lewis 1956). This was the result of a complaint filed by the US Department of Justice Antitrust Division in 1949 against the Western Electric Company and AT&T, claiming that the companies were unlawfully restraining and monopolizing trade and commerce in violation of the Sherman Antitrust Act. Under the terms of the consent decree, Western Electric (a fully owned subsidiary of AT&T and 50 % owner of Bell Labs) was prohibited from manufacturing non-telecommunications equipment, and AT&T (owner of the other 50 % of Bell Labs) was forbidden to engage in business other than communication services (Lewis 1956). Consequently, AT&T could not market or license Unix for profit, and, therefore Unix was initially licensed royalty-free through simple letter agreements (Salus 1994, p. 60). Later however licenses became more intricate and restrictive, limiting the availability of its source code (Takahashi and Takamatsu 2013), which was carefully guarded as a trade secret (Libes and Ressler 1989, p. 20).

Luckily, important Unix material of historical importance has survived until today, often through magnetic tapes preserved in the hands of people realizing their significance. Also, key parts of the early Unix development, namely the systems running on the 16-bit PDP-11 and early versions of the 32-bit Unix (excluding System III, System V, and their successors), were released in 2002 by one of its right-holders (Caldera International) under a liberal license. The license, which covers the 16-bit Unix Editions 1–7 and 32-bit Unix 32V, allows the redistribution and use of the material in source and binary forms, with or without modification, subject to conditions similar to these of the original BSD license.

Combining these parts with software that was developed or released as open source software by the University of California at Berkeley and the FreeBSD Project provides coverage of the system's development over a period ranging from June 20th 1972 until today.

Curating and processing available source code snapshots as well as old and modern configuration management repositories allows the reconstruction of a new synthetic Git repository that combines under a single roof most of the available data. This repository documents in a digital form the detailed history and evolution of an important digital artefact over a period of 44 years. The contributions of the work presented here are:

– the release of a 1.1GB Git repository covering the history of Unix from 1972 until the modern time, which can be used for diverse research,
– the documentation of the authorship details of many parts of the early Unix source code in a machine-readable format,
– the provision of an open source project where additional authorship data and Unix systems can be added, and
– the development of techniques and tools for converting historical source code snapshots into a Git repository that can correctly track changes and authorship across releases.

This work expands on a presentation (Gall et al. 2014) and a four-page conference paper (Spinellis 2015) by including considerably more detailed information on the data and their generation process. The added material includes an overview of the code's licensing, the data set's key metrics (Table 1), a detailed description of the available software releases (Section 2.1), an expanded overview of data sources (Table 2) and available metadata (Section 2.2), GitHub integration (Section 2.3), known limitations (Section 2.4), the documentation of derived authorship data (Tables 3 and 4), details on the data import process and tools (Section 3.3), instructions for contributing to the project (Section 5), and a second example on using the data set (Fig. 13).

The following sections describe the Unix history repository's structure and contents (Section 2), the way it was created (Section 3), how it can be used (Section 4), and how it can be extended (Section 5). The paper concludes with ideas for further work (Section 6).

Table 1 Key repository metrics of the Unix and Linux history repositories

Metric	Unix history	Linux history
Start date	1972-06-20	1991-09-17
Start files	13	92
Start lines	4768	917,812
End files	63,049	51,396
End lines	27,388,943	21,525,436
Data size (`.git`)	1.1GB	1.0GB
Number of commits	495,622	611,735
Number of merges	2,523	48,821
Number of authors	973	18,465
Days with activity	13,004	5,126

Table 2 Data sources

Tag	Data source(s)
Research-V1	http://www.tuhs.org/Archive/PDP-11/Distributions/research/Dennis_v1/svntree-20081216.tar.gz
Research-V3	http://www.tuhs.org/Archive/PDP-11/Distributions/research/Dennis_v3/nsys.tar.gz
Research-V4	http://www.tuhs.org/Archive/PDP-11/Distributions/research/Dennis_v4/v4man.tar.gz
Research-V5	http://www.tuhs.org/Archive/PDP-11/Distributions/research/Dennis_v5/v5root.tar.gz
Research-V6	http://www.tuhs.org/Archive/PDP-11/Distributions/research/Dennis_v6/v6root.tar.gz
	http://www.tuhs.org/Archive/PDP-11/Distributions/research/Dennis_v6/v6src.tar.gz
	http://www.tuhs.org/Archive/PDP-11/Distributions/research/Dennis_v6/v6doc.tar.gz
BSD-1	http://www.tuhs.org/Archive/PDP-11/Distributions/ucb/1bsd.tar.gz
BSD-2	http://www.tuhs.org/Archive/PDP-11/Distributions/ucb/2bsd.tar.gz
Research-V7	http://www.tuhs.org/Archive/PDP-11/Distributions/research/Henry_Spencer_v7/v7.tar.gz
	http://www.tuhs.org/Archive/PDP-11/Distributions/research/Henry_Spencer_v7/v7.patches.tar.gz
Bell-32V	http://www.tuhs.org/Archive/VAX/Distributions/32V/32v_usr.tar.gz
BSD-3	http://www.tuhs.org/Archive/4BSD/Distributions/3bsd.tar.gz
BSD-4	file://CSRG-CD-ROMs/cd1/4.0
BSD-4_1_snap	file://CSRG-CD-ROMs/cd1/4.1.snap
BSD-4_1c_2	file://CSRG-CD-ROMs/cd1/4.1c.2
BSD-4_2	file://CSRG-CD-ROMs/cd1/4.2
BSD-4_3	file://CSRG-CD-ROMs/cd1/4.3
BSD-4_3_Tahoe	file://CSRG-CD-ROMs/cd2/4.3tahoe
BSD-4_3_Net_1	file://CSRG-CD-ROMs/cd2/net.1
BSD-4_3_Reno	file://CSRG-CD-ROMs/cd2/4.3reno
BSD-4_3_Net_2	file://CSRG-CD-ROMs/cd2/net.2
BSD-4_4	file://CSRG-CD-ROMs/cd3/4.4

Table 2 (continued)

Tag	Data source(s)
BSD-4_4_Lite1	file://CSRG-CD-ROMs/cd2/4.4BSD-Lite1
BSD-4_4_Lite2	file://CSRG-CD-ROMs/cd3/4.4BSD-Lite2
BSD-SCCS	file://CSRG-CD-ROMs/cd4
386BSD-0.0	http://www.oldlinux.org/Linux.old/distributions/386BSD/386bsd-0.0/floppies/3in/src/
386BSD-0.1	http://www.oldlinux.org/Linux.old/distributions/386BSD/0.1/386BSD/
386BSD-0.1-patchkit	ftp://www.tuhs.org/BSD/386bsd-patchkits/
FreeBSD-release/1.0	http://ftp-archive.freebsd.org/pub/FreeBSD-Archive/old-releases/i386/ISO-IMAGES/1.0/1.0-disc1.iso
FreeBSD-release/1.1	http://ftp-archive.freebsd.org/pub/FreeBSD-Archive/old-releases/i386/ISO-IMAGES/FreeBSD-1.1-RELEASE/cd1.iso
FreeBSD-release/1.1.5	http://ftp-archive.freebsd.org/pub/FreeBSD-Archive/old-releases/i386/ISO-IMAGES/FreeBSD-1.1.5.1/cd1.iso
FreeBSD-release/2...	https://github.com/freebsd/freebsd

Table 3 Manually-allocated contributions in Research Unix editions

Identifier	Name	Contributions
aho	Alfred V. Aho	awk, dbm, egrep, fgrep, libdbm
ark	Andrew Koenig	varargs
bsb	Brenda S. Baker	struct
bwk	Brian W. Kernighan	adv, awk, beg, beginners, ctut, ed, edtut, eqn, eqnchar, learn, m4, neqn, rat, ratfor, trofftut, uprog
cbh	Charles B. Haley	regen, setup, tar
csr	C. S. Roberts	tss
dan	D. A. Nowitz	uucp
dmr	Dennis Ritchie	a.out, ar, as, assembler, atan, bcd, c, cacm, cat, cc, cdb, check, chmod, chown, cmp, core, cp, ctime, ctour, date, db, dev, df, dir, dmr, dp, dsw, du, ed, exit, exp, f77, fc, fort, fptrap, getc, getty, glob, goto, hypot, if, init, iolib, iosys, istat, ld, libc, ln, login, ls, m4, man2, man3, man4, mesg, mkdir, mount, mv, nm, od, pr, ptx, putc, regen, rew, rf, rk, rm, rmdir, rp, secur, security, setup, sh, sin, sort, sqrt, strip, stty, su, switch, tp, tty, type, umount, unix, uprog, utmp, who, write, wtmp
doug	Doug McIlroy	diff, echo, graph, join, look, m6, sort, spell, spline, tmg
haight	Dick Haight	expr, find
jfm	J. F. Maranzano	adb
jfo	Joe Ossanna	azel, ed, getty, nroff, ov, roff, s7, stty, troff, wc
ken	Ken Thompson	ar, atan, atof, bas, bj, bproc, cacm, cal, cat, check, chess, chmod, chown, core, cp, dc, dd, df, dir, dli, dp, dsw, dtf, ed, exp, f77, fc, fed, form, fort, fptrap, getty, grep, hypot, implement, init, itoa, ken, libplot, ln, log, login, ls, mail, man, man2, man4, mesg, mkdir, moo, mount, mv, nlist, nm, od, password, plot, pr, qsort, rew, rf, rk, rm, rmdir, roff, rp, sa, sh, sin, sort, sqrt, stty, su, sum, switch, sync, sys, tabs, tp, ttt, tty, umount, uniq, unix, utmp, who, write, wtmp
lem	Lee E. McMahon	comm, cu, grep, qsort, sed
llc	Lorinda Cherry	bc, dc, deroff, eqn, eqnchar, fed, form, neqn
mel	Michael E. Lesk	iolib, learn, lex, ms, msmacros, refer, tbl, tmac, uucp
pjw	Peter J. Weinberger	awk, f77, libI77, libmp, mp
rhm	Robert Morris	atan, bc, crypt, dc, exp, factor, fed, form, libm, m6, man3, password, primes, sky, sqrt

Table 3 (continued)

Identifier	Name	Contributions
schmidt	Eric Schmidt	lex
scj	Stephen C. Johnson	cc, lint, mip, pcc, porttour, yacc
sif	S. I. Feldman	f77, make
srb	S. R. Bourne	adb, sh, shell
xtp	Greg Chesson	mpx, mpxcall, mpxio, pk[01]

2 Data Overview

The 1GB Unix history Git repository is made available for cloning on GitHub.[1] Currently[2] the repository contains 496 thousand commits and 2,523 merges from about 973 contributors (measured by counting unique email addresses). The contributors include 29 from the Bell Labs staff, 158 from Berkeley's Computer Systems Research Group (CSRG), 79 contributors of the 386BSD patch kit, and 691 from the FreeBSD Project. More metrics regarding the Unix history repository are listed in Table 1. For comparison purposes the table also includes details regarding the Linux kernel history repository.[3] The Unix history repository reported here differs from the Linux one in three ways: first it covers a significantly longer timespan, second after 1974 it contains code of a complete system (kernel and tools) rather than only a kernel, and third it represents the work of four diverse communities.

2.1 Available Operating System Releases

The repository starts its life at a tag identified as *Epoch*, which contains only licensing information and its modern README file. Various tag and branch names identify points of significance. A timeline of these releases based on their repository timestamps is depicted in Fig. 1.

The `Research-V`*X* tags correspond to six so-called *research* editions of Unix that came out of Bell Labs. These start with `Research-V1` (4768 lines of PDP-11 assembly) and end with `Research-V7` (1820 mostly C files, 324kLOC-lines of code). Following tradition, the numbers of these releases correspond to the edition of the manual (Libes and Ressler 1989, p. 5). For example, *Research–V7* is variously called *7th Edition* or *Version 7* Unix.

The 1st Edition (November 3, 1971[4] — `Research-V1`) contains only the kernel; the 60 user commands that came with it (Salus 1994, p. 41) are no longer available. Even the kernel, written in PDP-11 assembly language, has not survived in electronic form. It was derived from a group effort that took a scanned June 1972 280-page printout of 1st Edition UNIX source code and documentation (Bashkow 1972), and restored it to an incomplete but running system (Toomey 2010). Two representative pages of the printout are shown in Fig. 2.

The next four editions are also only partially available.

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

[2] Updates may add or modify material. To ensure replicability the repository's users are encouraged to fork it on GitHub or archive it.

[3] https://archive.org/details/git-history-of-linux.

[4] The dates provided here are given by Salus (1994, p. 43).

Table 4 Manually-allocated contributions in BSD Unix releases

Identifier	Name	Contributions
arn	Rich Newton	spice
arnold	Ken Arnold	curses, fortune, fortunes, libcurses
cbh	Charles B. Haley	ex, eyacc, mkstr, pascal, pi, public, px
cohen	Ellis Cohen	where
cvw	Chris Van Wyk	ideal
dlw	David Wasley	libI77uc
dop	Don O. Pederson	spice
eric	Eric Allman	me, memacros, portlib, sendmail, trek, tset
erics	Eric Shienbrood	more
frodo	T. J. Kowalski	fsck
honey	Peter Honeyman	pathalias
hpk	Howard Katseff	box, crazy, froc, last, sdb, sess, syswatch, toc, watch
jeff	Jeff Schriebman	biorhythm, colrm, flt40, linerm, procp, repeat, strip
jfr	John Reiser	as
jkf	John Foderaro	lisp
ken	Ken Thompson	apl, pi, px
kurt	Kurt A. Shoens	fix, fixit, fleece, fmt, funny, lock, mail, Mail, pq, reset, rmtree, ucb-mail, vpac
lem	Lee E. McMahon	gres
llc	Lorinda Cherry	diction
mark	Mark Horton	banner, chfn, curses, leave, libcurses, rewind, script, ul, w
mckusick	Kirk McKusick	gprof, num
mike	Mike Tilson	tmac, vcat
mja	Mike Accetta	enet, pty, tty_pty
ozalp	Ozalp Babaoglu	analyze, locore, vm, vmstat, vmunix
peter	Peter B. Kessler	gprof
presott	David Presotto	vgrind
rrh	Robert R. Henry	as
schmidt	Eric Schmidt	berknet, net, netcp, netlpr, netmail, netq, netrm
sif	S. I. Feldman	efl
sklower	Keith Sklower	arff, flcopy, libNS
tbl	Tom London	liszt
td	Tom Duff	tmac, vcat
toy	Michael Toy	33, libretro, num, rogue, shutdown, termcap, termlib
tuck	Richard Tuck	arff, flcopy

Table 4 (continued)

Identifier	Name	Contributions
wnj	Bill Joy	analyze, apropos, ashell, cat3a, chessclock, chownall, colcrt, collpr, cptree, cr3, csh, cshms, cxref, dates, diffdir, double, dribble, edit, ex, ex-1, expand, exrecover, exrefm, eyacc, fold, from, glob2, head, htmp, htmpg, htmps, iul, list, lntree, locore, ls, makeTtyn, man, manwhere, mkstr, msgs, nm, num, number, osethome, pascal, pascals, pcc, pi, pi0, pi1, pix, print, Print, puman, px, pxp, pxref, rout, see, sethome, sh, sidebyside, size, soelim, squash, ssp, strings, strip, termcap, termlib, tests, tra, transcribe, ttycap, ttycap2, Ttyn, ttytype, typeof, ulpr, vgrind, vi, vm, vmstat, vmunix, wc, whatis, whereis, whoami, whoison, xstr
x-br	Bill Reeves	tmac, vcat
x-clm	Colin L. Mc Master	ccat, compact, uncompact
x-dl	Douglas Lanam	apl
x-dw	David Willcox	indent
x-etc	Earl T. Cohen	finger
x-im	Ivan Maltz	ticktock
x-jp	Juan Porcar	locore, vm, vmunix
x-le	Len Edmondson	lastcomm
x-or	Olivier Roubine	dribble
x-rd	R. Dowell	spice
x-rh	Ross Harvey	apl
x-rt	Robert Toxen	tod

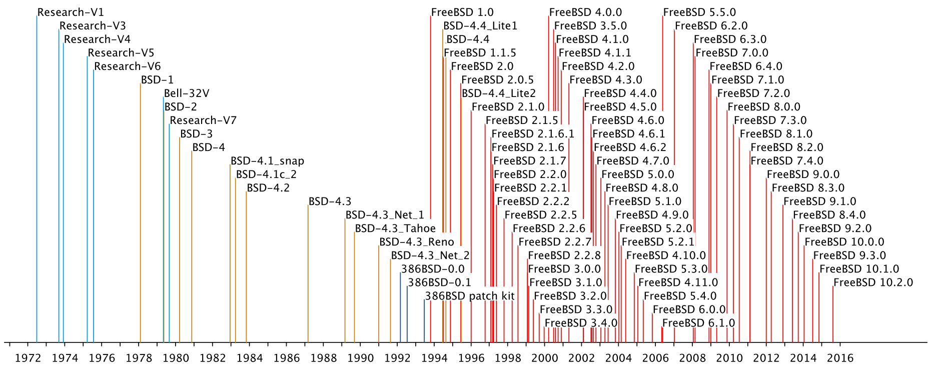
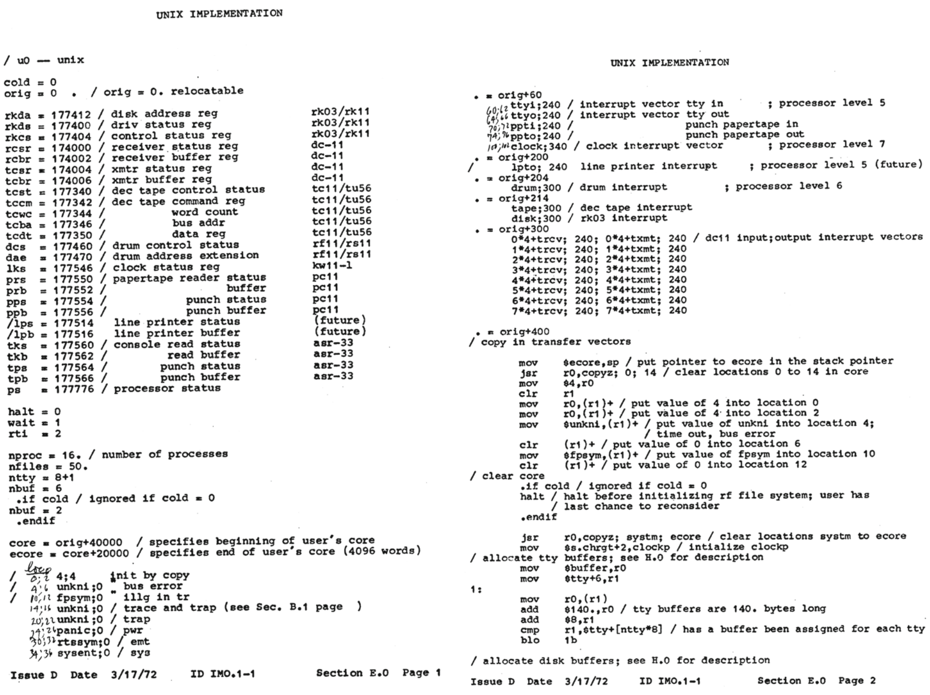


Fig. 1 Timeline of releases in the repository



```
                    UNIX IMPLEMENTATION


/ u0 -- unix

cold = 0
orig = 0  .  / orig = 0. relocatable

rkda = 177412 / disk address reg               rk03/rk11
rkds = 177400 / driv status reg                rk03/rk11
rkcs = 177404 / control status reg             rk03/rk11
rcsr = 174000 / receiver status reg            dc-11
rcbr = 174002 / receiver buffer reg            dc-11
tcsr = 174004 / xmtr status reg                dc-11
tcbr = 174006 / xmtr buffer reg                dc-11
tcst = 177340 / dec tape control status        tc11/tu56
tccm = 177342 / dec tape command reg           tc11/tu56
tcwc = 177344 /          word count            tc11/tu56
tcba = 177346 /          bus addr              tc11/tu56
tcdt = 177350 /          data reg              tc11/tu56
dcs  = 177460 / drum control status            rf11/rs11
dae  = 177470 / drum address extension         rf11/rs11
lks  = 177546 / clock status reg               kw11-l
prs  = 177550 / papertape reader status        pc11
prb  = 177552 /                  buffer        pc11
pps  = 177554 /            punch status        pc11
ppb  = 177556 /            punch buffer        pc11
/lps = 177514   line printer status            (future)
/lpb = 177516   line printer buffer            (future)
tks  = 177560 / console read status            asr-33
tkb  = 177562 /         read buffer            asr-33
tps  = 177564 /        punch status            asr-33
tpb  = 177566 /        punch buffer            asr-33
ps   = 177776 / processor status

halt = 0
wait = 1
rti  = 2

nproc = 16. / number of processes
nfiles = 50.
ntty = 8+1
nbuf = 6
 .if cold / ignored if cold = 0
nbuf = 2
 .endif

core = orig+40000  / specifies beginning of user's core
ecore = core+20000 / specifies end of user's core (4096 words)

     loop
/    0;2  4;4     init by copy
/    4;6  unkni;0 " bus error
/  10;12  fpsym;0 " illg in tr
   14;16  unkni;0 / trace and trap (see Sec. B.1 page  )
   20;22  unkni;0 / trap
   24;26  panic;0 / pwr
   30;32  rtssym;0 / emt
   34;36  sysent;0 / sys

Issue D  Date  3/17/72     ID IMO.1-1         Section E.0   Page 1
```

```
                    UNIX IMPLEMENTATION


 . = orig+60
  60;62  ttyi;240 / interrupt vector tty in        ; processor level 5
  64;66  ttyo;240 / interrupt vector tty out
  70;72  ppti;240 /                   punch papertape in
  74;76  ppto;240 /                   punch papertape out
 100;102 clock;340 / clock interrupt vector        ; processor level 7
 . = orig+200
/        lpto; 240  line printer interrupt      ; processor level 5 (future)
 . = orig+204
         drum;300 / drum interrupt           ; processor level 6
 . = orig+214
         tape;300 / dec tape interrupt
         disk;300 / rk03 interrupt
 . = orig+300
         0*4+trcv; 240; 0*4+txmt; 240 / dc11 input;output interrupt vectors
         1*4+trcv; 240; 1*4+txmt; 240
         2*4+trcv; 240; 2*4+txmt; 240
         3*4+trcv; 240; 3*4+txmt; 240
         4*4+trcv; 240; 4*4+txmt; 240
         5*4+trcv; 240; 5*4+txmt; 240
         6*4+trcv; 240; 6*4+txmt; 240
         7*4+trcv; 240; 7*4+txmt; 240

 . = orig+400
/ copy in transfer vectors

        mov    $ecore,sp / put pointer to ecore in the stack pointer
        jsr    r0,copyz; 0; 14 / clear locations 0 to 14 in core
        mov    $4,r0
        clr    r1
        mov    r0,(r1)+ / put value of 4 into location 0
        mov    r0,(r1)+ / put value of 4 into location 2
        mov    $unkni,(r1)+ / put value of unkni into location 4;
                        / time out, bus error
        clr    (r1)+ / put value of 0 into location 6
        mov    $fpsym,(r1)+ / put value of fpsym into location 10
        clr    (r1)+ / put value of 0 into location 12
/ clear core
        .if cold / ignored if cold = 0
        halt / halt before initializing rf file system; user has
             / last chance to reconsider
        .endif

        jsr    r0,copyz; systm; ecore / clear locations systm to ecore
        mov    $s.chrgt+2,clockp / intialize clockp
/ allocate tty buffers; see H.0 for description
        mov    $buffer,r0
        mov    $tty+6,r1
1:
        mov    r0,(r1)
        add    $140.,r0 / tty buffers are 140. bytes long
        add    $8,r1
        cmp    r1,$tty+[ntty*8] / has a buffer been assigned for each tty
        blo    1b

/ allocate disk buffers; see H.0 for description

Issue D  Date  3/17/72     ID IMO.1-1         Section E.0   Page 2
```

Fig. 2 Representative scanned pages from the 1st Edition Unix

- The 2nd Edition (June 12, 1972) source code has only survived in the form of fragments. These were manually restored by Warren Toomey, who pieced together data from a subset of a disk dump's DECtapes, that were extracted by Dennis Ritchie.[5] The fragments comprise the source code for some of the system's utilities. In addition, this edition's manual survives as a printed document.
- The 3rd Edition (February 1973 — `Research-V3`) contains only the Unix kernel: 7609 lines of which just 768 are written in PDP-11 assembly and the rest are written in C. This was the first Unix version to support pipes (Salus 1994, p. 50).
- The 4th Edition (November 1973 — `Research-V4`) contains only source markup for the manual pages: 18975 lines of *troff* code.
- The 5th Edition (June 1974 — `Research-V5`) is missing the source markup of the manual pages. This edition was officially made available to universities for educational use (Libes and Ressler 1989, p. 8).

The 6th Edition (May 1975 — `Research-V6`), is the first that appears in the repository in complete form, and the first that became widely available outside Bell Labs through licenses to commercial and government users. It was also the last bearing the names of Thompson and Ritchie on the manuals' title page. The 6th Edition is the one John Lions used for teaching two operating systems courses at the University of New South Wales in Australia. In 1977 Lions produced a booklet with an indexed 9073-line listing of the entire Unix kernel with an equal amount of commentary explaining its structure (Lions 1996). Although

[5] http://www.tuhs.org/Archive/PDP-11/Distributions/research/1972_stuff/.

this was initially sold by mail order, a year afterwards it was no longer available (Salus 1994, p. 130). Nevertheless, for the next two decades it circulated as multiple-generation *samizdat* photocopies (Lions 1996, p. ix), until in late 1995 the lawyers of Santa Cruz Operation, Inc. gave permission for its official publication.

The 7th Edition (January 1979 — `Research-V7`), includes many new influential commands, such as *awk* (Aho et al. 1979), *expr*, *find*, *lex* (Lesk 1975), *lint* (Johnson 1977), *m4* (Kernighan and Ritchie 1979), *make* (Feldman 1979), *refer* (Lesk 1979a), *sed* (McMahon 1979), *tar*, *uucp* (Nowitz and Lesk 1979), and the Bourne shell (Bourne 1979; 1978). It also supports larger file systems and more user accounts. It is the version that was widely ported to other architectures.

Unix 32V (or *32/V* — tagged `Bell-32V`) is the port of the 7th Edition Unix to the DEC/VAX architecture. It was created by John Raiser and Tom London, managed by Charlier Roberts, at Bell Labs in Holmdel in 1978. There seem to be two reasons why the port was not implemented by the original team. First, DEC's refusal to support Unix, favouring VMS instead, and, second, the complexity of the VAX instruction set, which apparently went against the values of the Unix patriarchs (Salus 1994, p. 154). The port took about three months to complete by treating the VAX as a large PDP-11 — keeping the existing swapping mechanism and ignoring the VAX's hardware paging capability (Libes and Ressler 1989, p. 12). In the fall of 1978 *Bell-32V* was sent to the University of California at Berkeley under a "special research agreement" (Salus 1994, p. 154).

BSD–X tags correspond to 15 snapshots released from Berkeley. Their contents are summarized in the following paragraphs, based on published descriptions (Salus 1994, pp. 142–145; McKusick 1999; McKusick and Neville-Neil 2004, pp. 3–13) and the manual examination of their contents. The first Berkeley Software Distribution (BSD) (tagged `BSD-1`), released in early 1978, contained the Unix Pascal System the *ex* line editor, and a number of tools. The Second Berkeley Software Distribution (2BSD, tagged `BSD-2`), included the full-screen editor *vi*, the associated terminal capability database and management library *termcap*, and many more tools, such as the *csh* shell. The 3BSD release (tagged `BSD-3`), released in late 1979, extended *Unix 32V* with support for virtual memory (Babaoğlu and Joy 1981) and the 2BSD additions. Subsequent releases (Salus 1994, pp. 164–167) included in the repository are marked with the following tags.

– `BSD-4` (4BSD — October 1980) was developed by the newly established Computer Systems Research Group (CSRG) working on a contract for the Defense Advanced Research Projects Agency (DARPA). The contract aimed at standardizing at the operating system level through the adoption of Unix the computing environment used by DARPA's research centers. The release included a 1k block file system, support for the VAX-11/750, enhanced email, job control, and reliable signals.
– `BSD-4_1_snap` (4.1BSD — December 1982), a snapshot of 4.1BSD probably before 4.1a, included performance improvements and auto-configuration support. This release was named 4.1BSD rather than 5BSD in response to objections by AT&T lawyers who feared the 5BSD name might be confused with AT&T's commercial Unix *System V* release. Subsequent BSD releases followed this numbering scheme.
– `BSD-4_1c_2` (4.1c2BSD — April 1983) was the last intermediary release preceding 4.2BSD. It was used by many hardware vendors to start their 4.2BSD porting efforts. It included TCP/IP networking, networking tools (*ftp*, *netstat*, *rlogin*, *routed*, *rsh* , *rwho*, *telnet*, *tftp*) from 4.1aBSD, and filesystem improvements, such as symbolic links, from 4.1bBSD. Sadly, 4.1aBSD and 4.1bBSD are not included in the CSRG CD set, which was used for obtaining the BSD snapshots for this work.

- `BSD-4_2` (4.2BSD — September 1983) was a major release of features tested in 4.1aBSD to 4.1cBSD. Compared to the preceding releases it improved networking support and added new signal facilities and disk quotas.
- `BSD-4_3` (4.3BSD — June 1986) came with performance improvements, a directory name cache, and the BIND internet domain name system server.
- `BSD-4_3_Tahoe` (4.3BSD Tahoe — June 1988) split the kernel into machine-dependent and machine-independent parts in order to include support for the CCI Power 6/32 minicomputer (code-named *Tahoe*). It also included improved TCP algorithms.
- `BSD-4_3_Net_1` (4.3BSD Networking Release — November 1988) is a subset of the code that does not include material requiring an AT&T license. It was released to help vendors create standalone networking products, without incurring the AT&T binary license costs. It included the BSD networking kernel code and supporting utilities.
- `BSD-4_3_Reno` (4.3BSD Reno — June 1990) supported virtual file system implementations through the *vnode* interface, Hewlett-Packard 9000/300 workstations, and OSI networking. It also incorporated a new virtual memory system adapted from Carnegie-Mellon's MACH operating system, a Network File System (NFS) implementation done at the University of Guelph, and an automounter daemon. Considerable material in this release was released by Berkeley with a license allowing the easy redistribution and reuse of those parts.
- `BSD-4_3_Net_2` (4.3BSD Networking Release 2 — June 1991) came with (what is now called) an open source reimplementation of almost all important utilities and libraries that used to require an AT&T license. It also included a kernel that had been cleaned from AT&T source code, requiring just six additional files to make a fully-functioning system. This was the version used by Bill Jolitz to create a compiled bootable Unix system for the 386-based PCs.
- `BSD-4_4_Lite1` (4.4BSD Lite — June 1994) was released following two years of litigation and settlement talks regarding the alleged use of proprietary AT&T material between: a) Unix System Laboratories (USL — a wholly owned subsidiary of AT&T that developed and marketed Unix) and (later) USL's new owner, Novell and b) Berkeley Software Design Incorporated (BSDI — a developer of commercially supported version of BSD Unix) and the University of California. As a result this release removed three files that were included in the *Net/2* release, added USL copyrights to about 70 files, and made minor changes to a few others. With these changes and according to the settlement's terms USL could not sue third parties basing their code on this release. Consequently, efforts such as FreeBSD and NetBSD rebased their work on this code base. The release also included additional work done on the system, such as support for the portal filesystem.
- `BSD-4_4`, released at the same time as 4.4BSD Lite, was an "encumbered" version of 4.4BSD-Lite that included the files requiring an AT&T license.
- `BSD-4_4_Lite2` (4.4BSD-Lite Release 2 — June 1995) was the last release made by CSRG before the group was disbanded. It included bug fixes and enhancements integrated through funding obtained from the distribution of 4.4BSD.

386BSD–X tags correspond to 386/BSD: version 0.0 (March 1992 — tagged `386BSD-0.0`) and version 0.1 (July 1992 — tagged `386BSD-0.1`). This was a derivative of the BSD Networking 2 Release targeting the Intel 386 architecture, developed by Lynne and William Jolitz, who wrote the six missing kernel files. A description of this system was published as a series of 18 articles in the *Dr. Dobb's Journal* (Jolitz and Jolitz 1991).

The `386BSD-0.1-patchkit` branch contains 171 commits associated with patches made to 386BSD 0.1 by a group of volunteers from mid-1992 to mid-1993. Patches contain their changes in Unix "context diff" format, and can therefore be applied automatically to the 386BSD distribution. Each patch is accompanied by a metadata file listing its title, author, description, and prerequisites.

FreeBSD–release/X tags and branches mark 69 releases derived from the FreeBSD Project. The names of tags and branches to be imported are obtained by excluding from the corresponding FreeBSD set, names matching one of the following patterns: `projects/`, `user/`, `master`, or `svn_head`. The FreeBSD Project started in early 1993 to address difficulties in maintaining 386/BSD through patches and working with its author to secure the future of 386/BSD (FreeBSD 2015). The focus of the project was to support the PC architecture appealing to a large, not necessarily highly technically sophisticated audience (McKusick and Neville-Neil 2004, p. 11). For legal reasons associated with the settlement of the USL case, while versions up to 1.1.5.1 were derived from the BSD Networking 2 Release, later ones were derived from the 4.4BSD-Lite Release 2 with 386/BSD additions. Two other BSD Unix descendants that could have been imported in the place of FreeBSD or in parallel with it are NetBSD and OpenBSD. FreeBSD was chosen, because it appears to be more popular that the other two as measured by the results obtained by Google search (17 million results for FreeBSD, 366 thousand results for OpenBSD, and 350 thousand results for Net BSD).

All branches with a *–Snapshot–Development* suffix denote commits that have been synthesized from a time-ordered sequence of a snapshot's files, while tags with a *–VCS–Development* suffix mark the point along an imported version control history branch where a particular release occurred.

2.2 Available Metadata

The repository's history includes commits from the earliest days of the system's development, such as the ones listed in Fig. 3. Commits that have been synthesized from snapshots and author-to-file maps, rather than imported from other revision control systems, can be recognized by the "`Synthetic commit`" phrase that appears in the commit's comment. Such commit comments follow exactly the preceding format, identifying the snapshot from which the commit was synthesized (four Research Editions in this case) and the file corresponding to the commit's time stamp.

Note that the commits derived from snapshot data are timestamped with the modification time of each file in the snapshot (see Fig. 3). This means that they represent only the file's final change and state in the development of the given release. Furthermore, the timestamp may be incorrect in cases where the file's modification time was changed after it was last written by its author. This is almost certainly the case in the very early Unix Research Editions.

Merges between releases that happened along the system's evolution, such as the development of 3BSD from 2BSD and Unix 32/V, are also correctly represented in the Git repository as graph nodes with two parents (see Fig. 8).

More importantly, the repository is constructed in a way that allows *git blame*, which annotates source code lines with the version, date, and author associated with their first appearance, to produce the expected code provenance results. For example, checking out the *BSD–4* tag, and running `git blame -M -M -C -C` on the kernel's *pipe.c* file will show lines spanning the 5th, 6th, and the 7th Research Edition developed at Bell Labs, as well as 4BSD developed at Berkeley (see Fig. 4). These lines are derived from snapshot

```
commit c4b1db0397c78e91b554e3edff3350a8c80781b1
Author: Ken Thompson <ken@research.uucp>
Date:   Mon May 7 01:23:11 1979 -0500

    Research V7 development
    Work on file usr/sys/sys/nami.c

    Synthesized-from: v7
commit 08d62191ab22882194e5f7004b3c00fb39d99193
Author: Ken Thompson <ken@research.uucp>
Date:   Fri Jul 18 04:09:14 1975 -0500

    Research V6 development
    Work on file usr/sys/ken/nami.c

    Synthesized-from: v6
commit 90798d6e3caec237bab95d22f0650047c3e9d431
Author: Ken Thompson <ken@research.uucp>
Date:   Thu Jan 2 19:25:11 1975 -0500

    Research V5 development
    Work on file usr/sys/ken/nami.c

    Synthesized-from: v5
commit a8c0fddc39968d4669a1f75a5121b4acd8f9c699
Author: Ken Thompson <ken@research.uucp>
Date:   Thu Aug 30 19:30:51 1973 -0500

    Research V3 development
    Work on file sys/ken/nami.c

    Synthesized-from: v3
```

Fig. 3 A log of file changes across Research Unix releases

files (probably) written by Ken Thompson in 1974, 1975, and 1979, and by Bill Joy in 1980. This feature allows the automatic (though computationally expensive) detection of the code's provenance at any point of time. Similarly, the *git log* command can also trace file changes across successive Unix releases. An example can be seen in Fig. 3, which was obtained by running `git log --follow -M20 -C20./usr/sys/sys/nami.c` on the checked out version of *Research-V7*.

```
78a8403693 usr/sys/ken/pipe.c     (Ken Thompson 1975-07-17 10:33:37 -0500  48)  iput(ip);
78a8403693 usr/sys/ken/pipe.c     (Ken Thompson 1975-07-17 10:33:37 -0500  49)  return;
78a8403693 usr/sys/ken/pipe.c     (Ken Thompson 1975-07-17 10:33:37 -0500  50)  }
9dd2619e6d usr/sys/sys/pipe.c     (Ken Thompson 1979-01-10 15:19:35 -0500  51)  u.u_r.r_val2 = u.u_r.r_val1;
9dd2619e6d usr/sys/sys/pipe.c     (Ken Thompson 1979-01-10 15:19:35 -0500  52)  u.u_r.r_val1 = r;
2c5a749b29 usr/sys/ken/pipe.c     (Ken Thompson 1974-11-26 18:13:21 -0500  53)  wf->f_flag = FWRITE|FPIPE;
2c5a749b29 usr/sys/ken/pipe.c     (Ken Thompson 1974-11-26 18:13:21 -0500  54)  wf->f_inode = ip;
2c5a749b29 usr/sys/ken/pipe.c     (Ken Thompson 1974-11-26 18:13:21 -0500  55)  rf->f_flag = FREAD|FPIPE;
2c5a749b29 usr/sys/ken/pipe.c     (Ken Thompson 1974-11-26 18:13:21 -0500  56)  rf->f_inode = ip;
2c5a749b29 usr/sys/ken/pipe.c     (Ken Thompson 1974-11-26 18:13:21 -0500  57)  ip->i_count = 2;
9dd2619e6d usr/sys/sys/pipe.c     (Ken Thompson 1979-01-10 15:19:35 -0500  58)  ip->i_mode = IFREG;
7fc472a9e2 usr/src/sys/sys/pipe.c (Bill Joy     1980-11-09 08:01:07 -0800  59)  ip->i_flag = IACC|IUPD|ICHG|IPIPE;
```

Fig. 4 Identification in a single file of commits spanning multiple snapshots

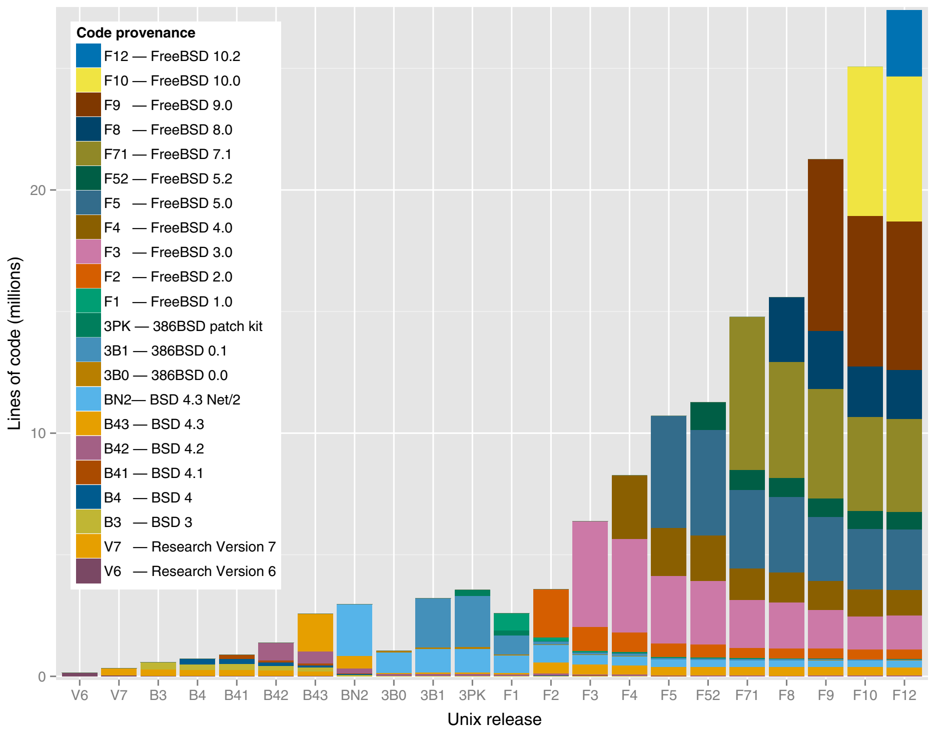


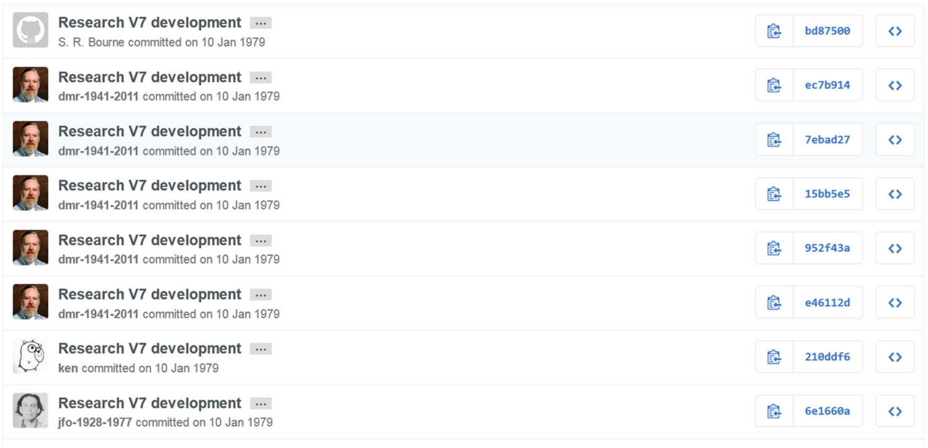
Fig. 5 Code growth and provenance across representative Unix releases

As can be seen in Fig. 5, a modern version of Unix (FreeBSD 10.2) still contains visible chunks of code from 4.3BSD, 4.3BSD Net/2, and all releases starting from FreeBSD 2.0. Interestingly, the Figure also shows that code developed during the 18-month dash to create an open source operating system out of the code released by Berkeley — 386BSD and FreeBSD 1.0 — does not seem to have survived.

The oldest significant code in the 2016 version of FreeBSD (10.2.0) appears to be an 18-line sequence in the C library file `timezone.c`. This was found by running the *git blame* command on it, which takes a bit more than two minutes to complete on a modern PC. The output (see Fig. 6) includes code with changes spanning three decades. The oldest part can also be found in the 7th Edition Unix file with the same name and a time stamp of January 10th, 1979 — 36 years ago.

```
lib/libc/gen/timezone.c         (Ed Schouten       2009-12-05 19:31:38 +0000 107) _tztab(int zone, int dst)
lib/libc/gen/timezone.c         (Rodney Grimes     1994-05-27 05:00:24 +0000 108) {
lib/libc/gen/timezone.c         (David E. O'Brien  2002-02-01 01:08:48 +0000 109)     struct zone     *zp;
lib/libc/gen/timezone.c         (David E. O'Brien  2002-02-01 01:08:48 +0000 110)     char    sign;
usr/src/lib/libc/gen/timezone.c (Bill Joy          1980-12-22 00:40:25 -0800 111)
usr/src/lib/libc/gen/timezone.c (Keith Bostic      1987-03-28 19:27:07 -0800 112)     for (zp = zonetab; zp->offset != -1;++zp)
                                                                                                                /* static tables */
usr/src/lib/libc/gen/timezone.c (Keith Bostic      1987-03-28 19:27:07 -0800 113)             if (zp->offset == zone) {
usr/src/libc/gen/timezone.c     (Dennis Ritchie    1979-01-10 14:58:45 -0500 114)                     if (dst && zp->dlzone)
usr/src/libc/gen/timezone.c     (Dennis Ritchie    1979-01-10 14:58:45 -0500 115)                             return(zp->dlzone);
usr/src/libc/gen/timezone.c     (Dennis Ritchie    1979-01-10 14:58:45 -0500 116)                     if (!dst && zp->stdzone)
usr/src/libc/gen/timezone.c     (Dennis Ritchie    1979-01-10 14:58:45 -0500 117)                             return(zp->stdzone);
usr/src/libc/gen/timezone.c     (Dennis Ritchie    1979-01-10 14:58:45 -0500 118)             }
```

Fig. 6 The oldest surviving code in a 2016 version of FreeBSD Unix (lines 114–118)



Research V7 development
S. R. Bourne committed on 10 Jan 1979
bd87500

Research V7 development
dmr-1941-2011 committed on 10 Jan 1979
ec7b914

Research V7 development
dmr-1941-2011 committed on 10 Jan 1979
7ebad27

Research V7 development
dmr-1941-2011 committed on 10 Jan 1979
15bb5e5

Research V7 development
dmr-1941-2011 committed on 10 Jan 1979
952f43a

Research V7 development
dmr-1941-2011 committed on 10 Jan 1979
e46112d

Research V7 development
ken committed on 10 Jan 1979
210ddf6

Research V7 development
jfo-1928-1977 committed on 10 Jan 1979
6e1660a

Fig. 7 Integration of the repository with current GitHub accounts

2.3 GitHub Integration

All commits included in the repository are associated with a single internet-standard (Resnick 2008) email address that can be linked to GitHub accounts. Old-style UCCP addresses (e.g. `research!srb`) are expressed in domain-name format (`srb@research.uucp`). Where more contributors are associated with a commit these are identified through `Co-Authored-By:` header-like lines added to the commit's comment. For example, most unaccounted early commits are attributed as instructed in the following quote (Ritchie 1984).

The reader will not, on the average, go far wrong if he reads each occurrence of 'we' with unclear antecedent as 'Thompson, with some assistance from me.'

A simple web-based search engine and a process outlined in the project's README file, allow current GitHub users to associate their past commits with their current GitHub account through the email address listed in the commit. This can be seen in Fig. 7: S. R. Bourne's commit (top) is not associated with a GitHub account, Ken Thompson's commit (second from the bottom) is associated with his current GitHub account, while the commits by Dennis Ritchie and J. F. Ossanna are associated with posthumously-created *in memoriam* accounts. Through direct emails and a message posted on *The Unix Heritage Society* mailing list past authors were encouraged to link their current GitHub accounts to their past commits. Although some have responded enthusiastically, the response was not overwhelming.

2.4 Known Limitations

Researchers using the provided data set should note some limitations regarding its coverage and fidelity. Where applicable these are discussed in detail in other parts of this work.

Many of the data set's limitations are associated with releases that are imported through snapshots (depicted by square boxes in Fig. 8). These are the following.

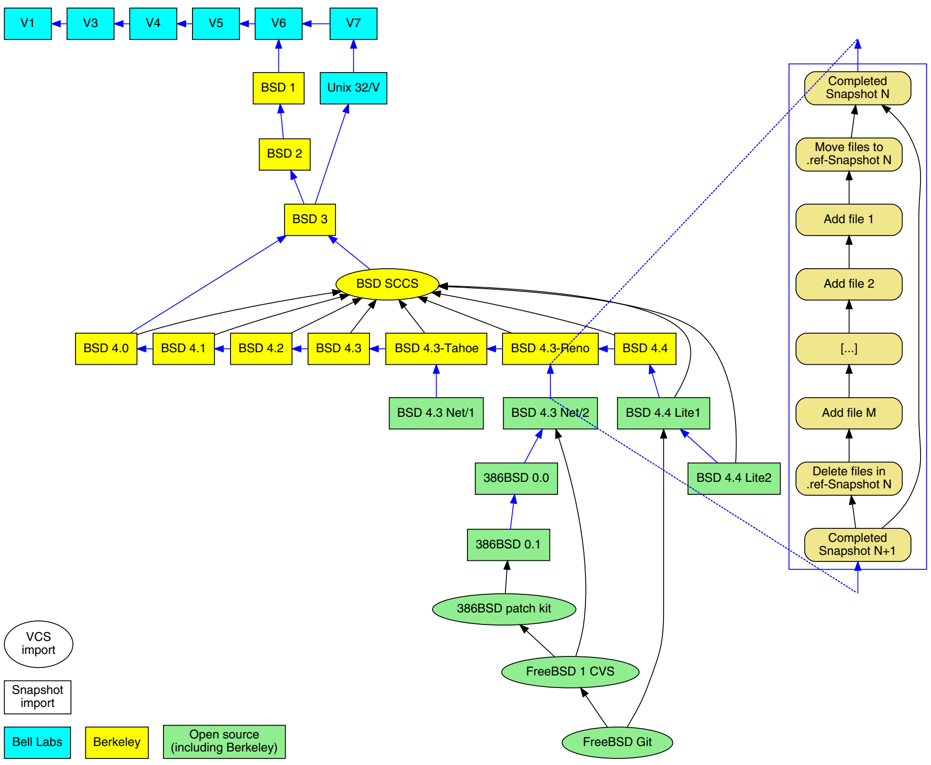


Fig. 8 Imported Unix snapshots, repositories, and their mergers. (On the *right*: a model of the synthetic commits between any two snapshots.)

- Only a single commit is associated with each file. This corresponds to the version of the released file. Other file changes up to the point of the preceding release have been lost forever.
- The file's commit time is derived from the file's modification time. Thus it may not correspond to the time the file's author last modified it, but to changes performed *en masse* at a later point of time.
- File authorship has been attributed manually as part of the work described here. It may thus be incorrect or incomplete.
- Commits lack the author's comment describing the change. The commit comments are automatically generated during the import process, and only detail metadata associated with each commit.
- Subjective decisions have been made by this work's author regarding which snapshot files to include in the repository and which to exclude. For example, binary files and formatted manual pages are excluded. Also excluded are copies of files that exist both in a source directory and in their installed position (for example `/usr/src/sys/h/param.h` and `/usr/include/sys/param.h`). In this case only the source code copy of the file is included. These decisions are clear cut; others, such as the exclusion of the Ingres database because it was deemed to be a separate project, are arguable.

Other limitations apply to the data set as a whole.

- Numerous evolution branches, such as the Second Research Edition, 2BSD, System III, Solaris, and OpenBSD, are missing, because they appear to have been permanently lost, or due to licensing restrictions or lack of time to import them.
- Early imported releases are missing data (e.g. source code or documentation), as described in Section 2.1.
- In software developed by multiple authors, such as *awk*, only the first author is identified in the commit metadata. The other authors are identified through header-like comment entries. This is a known limitation of Git.
- The same author may be identified in various commits with different email addresses. For example, `rgrimes@cdrom.com` and `rgrimes@FreeBSD.org` (Rodney Grimes) or `torek@ee.lbl.gov` and `torek@ucbvax.Berkeley.EDU` (Chris Torek). This is a well known problem in email analytics (Bird et al. 2006). No attempt was made to address this issue. However, the problem can be addressed through side channels, such as authors claiming their old email addresses through GitHub (see Section 2.3) or by populating and using Git's *mailmap* facility.
- There is not easy way to distinguish between branches and tags that have been created by this project and those that have been imported from the corresponding systems.
- The processing of *git blame* when run on FreeBSD releases 5.4 to 7.0 (inclusive) stops at a CVS to Subversion conversion dated April 20th, 2005.

3 Data Collection and Processing

The goal of the work reported here is to consolidate data concerning the history of Unix in a form that helps the study of the system's evolution, by entering them into a modern revision repository. This involves collecting the data, curating them, and synthesizing them into a single Git repository.

The software and data files that were developed as part of this project, are available online,[6] and, with appropriate network, CPU, and disk resources, they can be used to recreate the repository from scratch.

3.1 Primary Data

The project is based on three types of data (see Fig. 8 and the corresponding data sources listed in Table 2). First, **snapshots of early released versions**, which were obtained from the Unix Heritage Society archive (Toomey 2009), the CD-ROM images containing the full source archives of CSRG,[7] the OldLinux site, and the FreeBSD archive. These data are represented in the Unix history repository as synthetic commits, based on manually-added and extracted metadata. Second, **past and current repositories**, namely the CSRG SCCS repository, the FreeBSD 1 CVS repository, and the Git mirror of modern FreeBSD development.

[6] https://github.com/dspinellis/unix-history-make.

[7] https://www.mckusick.com/csrg/.

```
# 2. http://www.cs.bell-labs.com/who/doug/index.html
# "Text- and data-processing utilities:
# spell, diff, sort, join, graph, speak, etc."
usr/src/cmd/diff.*              doug
usr/src/cmd/graph\.c            doug
usr/src/cmd/join\.c             doug
usr/src/cmd/spell/.*            doug
bin/spell                       doug

# 3. [Morris] was also the author of the series of crypt programs
# that came with early Unix, including the final one distributed with the
# Seventh Edition
# http://cm.bell-labs.com/cm/cs/who/dmr/crypt.html
usr/man/man1/crypt\.1           rhm
usr/man/man3/crypt\.3           rhm
usr/src/cmd/crypt\.c            rhm
usr/src/libc/gen/crypt\.c       rhm

# 5. Volume 2 of the manual (supplementary documents)
# Based on the authors listed in each document
usr/doc/adb/.*                  jfm,srb
usr/doc/adv.ed/.*               bwk
usr/doc/assembler               dmr
usr/doc/awk                     aho,pjw,bwk
```

Fig. 9 Example specifications of file authorship

These data were imported into the repository as commits matching the original ones. The last, and most labour intensive, source of data was **primary research**, which is discussed in the next section. Information regarding merges between source code bases was obtained from a BSD family tree maintained by the NetBSD project.[8]

3.2 Authorship

The release snapshots do not provide information regarding their ancestors and the contributors of each file. Therefore, these pieces of information had to be determined through primary research. The authorship information was mainly obtained:

- by reading author biographies, research papers, books (Libes and Ressler 1989, pp. 29–36), internal memos, and old documentation scans;
- by reading and automatically processing source code and manual page markup;
- by communicating via email with people who were there at the time;
- by posting a query on the Unix *StackExchange* site;[9]
- by looking at the location of files; in early Research editions the kernel source code was split into `/usr/sys/dmr` (Dennis Ritchie) and `/usr/sys/ken` (Ken Thompson); and
- by propagating authorship from research papers and manual pages to source code and from one release to others. (Interestingly, the 1st and 2nd Research Edition

[8]http://ftp.netbsd.org/pub/NetBSD/NetBSD-current/src/share/misc/bsd-family-tree.

[9]http://unix.stackexchange.com/questions/64025/who-are-these-bsd-unix-contributors.

Fig. 10 Author names as listed in Unix documentation files

```
./adb/tut:.AU "MH2F-207" "3816"
./adb/tut-J. F. Maranzano
./adb/tut:.AU "MH2C-512" 7419
./adb/tut-S. R. Bourne
./adb/tut-.AI
--
./adv.ed/ae0:.AU "MH 2C518" 6021
./adv.ed/ae0-Brian W. Kernighan
./adv.ed/ae0-.AI
--
./assembler:.AU
./assembler-Dennis M. Ritchie
./assembler-.AI
--
./awk:.AU "MH 2C-522" 4862
./awk-Alfred V. Aho
./awk:.AU "MH 2C-518" 6021
./awk-Brian W. Kernighan
./awk:.AU "MH 2C-514" 7214
./awk-Peter J. Weinberger
./awk-.AI
```

manual pages have an "owner" section, listing the person (e.g. *ken*) associated with the corresponding system command, file, system call, or library function. This section was not there in the 4th Edition, and resurfaced as the "Author" section in BSD releases.)

Precise details regarding the source of the authorship information are documented in the project's files that are used for mapping Unix source code files to their authors and the corresponding commit messages.

The authorship information for major releases is stored in files under the project's `author-path` directory. These contain lines with a regular expressions for a file path followed by the identifier of the corresponding author (Fig. 9). Multiple authors can also be specified. The regular expressions are processed sequentially, so that a catch-all expression at the end of the file can specify a release's default authors.

Listing 1 Retrieving authorship information from documentation files

```
# Location of the Volume 2 documentation
cd archive/v7/usr/doc

# Find all files
find . -type f |
# List those containing the .AU macro
xargs fgrep .AU |
# Create path regular expressions
sed -n 's/^\.\/\([^-\/:]*\)\([:/]\).*/\/usr\/doc\/\1\2\.*/p' |
# Eliminate wildcard for single files
sed 's/:\.\*//;s/\///' |
# Remove duplicates
sort -u

# Find all files
find . -type f |
# List two lines of context around the .AU macro
xargs fgrep -A 2 .AU
```

Fig. 11 Specifications of author details

```
# Email address template
%A $@research.uucp

# Id (used in path maps):Full name:email
aho:Alfred V. Aho
bsb:Brenda S. Baker
bwk:Brian W. Kernighan
csr:C. S. Roberts
dan:D. A. Nowitz
dmr:Dennis Ritchie
doug:Doug McIlroy
jfm:J. F. Maranzano
jfo:Joe Ossanna
[...]
schmidt:Eric Schmidt:schmidt@ucbvax.Berkeley.EDU
```

As an example on how file authorship was collected and processed, consider the authors of the documentation files comprising Volume 2 of the *Unix Programmer's Manual* in the 7th Research Edition. These files contain the names of their authors using the *troff* markup macro `.AU`. The path regular expressions for the corresponding files were obtained through the shell commands shown in Listing 1 lines 4–13. The output were lines similar to what appears on the left column of Fig. 9. Then, the author names were listed with the commands shown in lines 15–18 of Listing 1. The generated output, such as the one appearing in Fig. 10, was then used to fill-in by hand the author identifiers appearing on the right column of Fig. 9. The authorship could then be propagated to the corresponding source code and Volume 1 manual pages.

To avoid repetition, a separate file with a `.au` suffix is used to map author identifiers into their names and emails (Fig. 11). One such file has been created for every community associated with the system's evolution: Bell Labs (`bell.au`), Berkeley (`berkeley.au`), 386BSD (`386bsd.au`), and FreeBSD (`freebsd.au`). For the sake of authenticity, emails for the early Bell Labs releases are listed using the UUCP (Quarterman and Hoskins 1986) top-level pseudo-domain, e.g. `ken@research.uucp`.

The FreeBSD author identifier map, required for importing the early CVS repository, was constructed by extracting the corresponding data from the project's modern Git repository, which includes the full names of modern committers. In addition, the Unix *finger* command was used on a computer hosting FreeBSD Project developers, to obtain the full names of another 60 contributors. In total the commented authorship files (897 rules) comprise 1215 lines, and there are another 988 lines mapping author identifiers to names.

3.3 Processing

The processing of the project's data sources has been codified into a 190-line `Makefile`. The processing involves five steps: data fetching, tool construction, data unpacking, data cleaning, and repository creation. The following paragraphs summarize how each step is performed.

Data fetching involves copying and cloning about 11GB of images, archives, and repositories from remote sites. Some of the snapshots used are available as compressed *tar* or *cpio* archives (sometimes split into multiple files), while others are available as (or can be converted into) CD-ROM images.

Under **tool construction** an archiver required for processing old PDP-11 archives on modern platforms is compiled from source. The archiver's code stems from 2.9BSD. It was subsequently modified to work on non- PDP-11 architectures.[10] Further modifications introduced as part of the work reported here include changes to make it preserve the modification time of the extracted files and adjustments to allow its warning-free compilation under Linux.

The **data unpacking** of the archives is mainly performed using *tar* and *cpio*. In addition, three 6th Research Edition directories are combined into one and all 1BSD archives are unpacked using the old PDP-11 archiver. Furthermore, the 8 and 62 386BSD floppy disk images are combined into two separate files. Finally, all CD-ROM images are made accessible so that they can be processed as file systems. This is done by mounting them via a loop-back device, which makes their contents (read-only) accessible as regular files.

The **data cleaning** involves tasks required to bring the data into a state suitable for the repository import tools. These are:

- restoring the 1st Research Edition kernel source code files, which were obtained from printouts through optical character recognition, into a format close to their original state;
- patching some 7th Research Edition source code files;
- removing metadata files and other files that were added after a release, to avoid obtaining erroneous time stamp information;
- patching corrupted SCCS files;[11]
- creating a Git repository representing the 386/BSD patch kit patches by successively applying each patch to 386/BSD 0.1 and committing the result;
- using a custom Perl script to remove CVS symbols assigned to multiple revisions in the early FreeBSD CVS repository;
- deleting CVS *Attic* files clashing with live ones; and
- converting the early FreeBSD CVS repository into a Git one using *cvs2svn*.

Finally, the synthesis of the various data sources into the single **Unix history Git repository** is performed by two scripts: A Perl script to feed Git with data and a shell script to invoke it for each data set.

The 780-line Perl script (`import-dir.pl`) can export the (real or synthesized) commit history from a single data source (snapshot directory, SCCS repository, or Git repository) in the *Git fast export* format. The script takes as input a number of obligatory and optional arguments. These are used to specify:

- whether to import data from a release snapshot directory, an SCCS repository, or a Git repository;
- the directory to import, the corresponding branch and version, and the timezone associated with the imported snapshot files;

[10] ftp://ftp.tuhs.org.ua/PDP-11/Tools/Tapes/newoldar.c.

[11] https://github.com/jonathangray/csrg-git-patches/.

- the mapping of files to contributors (see Fig. 9);
- the mapping between contributor login names and their full names and email (see Fig. 11);
- the Git commit or commits from which the import will appear to be merged;
- a path to prepend to file paths or Git branches being committed — this homogenizes the imported data;
- a leading path to strip from file paths being committed — again, this homogenizes the imported data;
- a list of Git identifiers whose files will be incorporated into the import as reference files (see the right of Fig. 8);
- a date specifying that reference files dated before that date are to be deleted — this will be explained later on;
- lists of files to ignore — this is used to exclude from the import duplicate files (existing in the source and the installation directory), CD-ROM name map files, SCCS directories, binary executables, other installed files, and third-party packages that were only present in the distributions for a brief period of time (e.g. Ingres);
- a list of files to ignore during the import, but merge at its end — this is used to handle SCCS files that appeared between two BSD snapshots;
- a cutoff date for the imported commits (this is used to ignore BSD SCCS commits after 1996 — the year following the last imported BSD snapshot);
- the name of a file to write unmatched paths — these are paths matched by a wildcard in the file to contributor map, and are used for manually improving the map's quality;
- a regular expression specifying the files to process — this is used for implementing a fast import test process over a small subset of all data.

The command produces output in the so-called *Git fast import* format; a simple text-based stream format that many Git tools use to import and export data. An excerpt of this format can be seen in Listing 2, though its contents will be explained later.

An interesting part of the repository representation is how snapshots are imported and linked together in a way that allows *git blame* to identify old code in newer file versions. Snapshots are imported into the repository as sequential commits based on the time stamp of each file. When all files have been imported, the repository is tagged with the name of the corresponding release. At that point these files could be deleted, and the import of the next snapshot could begin. Note that the *git blame* command works by traversing backwards a repository's history and using heuristics to detect code moves and copies within or across files. Consequently, deleted snapshot files would create a discontinuity between snapshots, and prevent the tracing of code between them.

Instead, before the next snapshot is imported, all the files of the preceding snapshot are moved into a hidden reference look-aside directory named `.ref`. (See the expanded synthetic commit series appearing on the right of Fig. 8). They remain there, until all files of the next snapshot have been imported, at which point they are deleted. Because every file in the `.ref` directory matches exactly an original file, *git blame* can determine how source code moves from one version to the next via the `.ref` file, without ever displaying the `.ref` file. To further help the detection of code provenance, and to increase the representation's realism, each release is represented as a merge between the branch with the incremental file additions (*–Development*) and the preceding release.

Listing 2 Example of generated Git fast import data

```
# 315830189 ../archive/3bsd/usr/src/cmd/ex/ex_addr.c
blob
mark :3
data 5190
 /* Copyright (c) 1979 Regents of the University of California */
 #include "ex.h"
 #include "ex_re.h"
  [...]

# Start development commits from a clean slate
commit refs/heads/BSD-3-Snapshot-Development
mark :10
author Bill Joy <wnj@ucbvax.Berkeley.EDU> 287674317 -0800
committer Bill Joy <wnj@ucbvax.Berkeley.EDU> 287674317 -0800
data 99
Start development on BSD 3
Create reference copy of all prior development files
(Synthetic commit)
merge Bell-32V
merge BSD-2
M 100644 1468bde18e292c07e5d30ecbd7fd2b91a60e4626 .ref-Bell-32V/usr/include/stat.h
M 100644 1468bde18e292c07e5d30ecbd7fd2b91a60e4626 .ref-Bell-32V/usr/include/sys/stat.h
M 100644 816685f1f60f44dfaed7e673294b9d07a12114e5 .ref-Bell-32V/usr/man/man2/open.2
 [...]

# 315830189 ../archive/3bsd/usr/src/cmd/ex/ex_addr.c
commit refs/heads/BSD-3-Snapshot-Development
mark :13
author Bill Joy <wnj@ucbvax.Berkeley.EDU> 315830189 -0800
committer Bill Joy <wnj@ucbvax.Berkeley.EDU> 315830189 -0800
data 75
BSD 3 development
Work on file usr/src/cmd/ex/ex_addr.c
(Synthetic commit)
M 100644 :3 usr/src/cmd/ex/ex_addr.c
 [...]

# Release
commit refs/heads/BSD-Release
mark :3700
author Bill Joy <wnj@ucbvax.Berkeley.EDU> 315928541 -0800
committer Bill Joy <wnj@ucbvax.Berkeley.EDU> 315928541 -0800
data 78
BSD 3 release
Snapshot of the completed development branch
(Synthetic commit)
from :3699
merge Bell-32V
merge BSD-2
D .ref-Bell-32V
D .ref-BSD-2

tag BSD-3
from :3700
tagger Bill Joy <wnj@ucbvax.Berkeley.EDU> 315928541 -0800
data 91
Tagged 3 release snapshot of BSD with 3
Source directory: ../archive/3bsd
(Synthetic tag)

done
```

The small example of the *Git fast import* data seen in Listing 2 demonstrates the concepts described in the preceding paragraphs. The data stream begins with the contents of files that will be stored in the repository. These are specified using the *blob* command (lines 1–8). For debugging purposes the name and timestamp of the file from which the data were taken are first listed as a # line comment (line 1). The embedded *mark* command (line 3) associates the number 3 with the contents. These are specified through the *data* command (line 4). Its argument specifies the number of bytes supplied, which follow after a newline (e.g. lines 5–8).

Following the definitions of data elements come the commits associated with the import in the *BSD-3-Snapshot-Development* branch. The first commit (lines 10–24) creates a reference copy of the previous snapshot's files by moving them to a hidden directory (`.ref-Bell-32V`) by means of the *M — filemodify* command (lines 21–24). This changes the path of the old blob object identified through its SHA-1 hash to the one specified. The number `100644` (an octal representation similar to the Unix file mode) specifies that this is a normal (non-executable) file. The associated branch is given as an argument to the *commit* command (line 11). The snapshot being imported (in this case *BSD-3*) is identified as a merge of two preceding snapshots (*Bell-32V* and *BSD-2*) using the *merge* command (lines 19–20). The name and email associated with the commit's author and committer and the corresponding timestamps (in seconds since 1970 and UTC offset) are given in lines 13–14, while the commit's message is specified with a *data* command in lines 15–18.

Then comes a series of commits that add files to the repository. The commits are ordered according to the timestamps of the corresponding files. The example listed in lines 26–36 creates the file `ex_addr.c`. This is again specified with an *M — filemodify* command, which now refers to the blob (3) with the file's contents. The branch, author, committer, file mode, and commit message are specified in the same way as in the previous commit.

The last commit in a snapshot import (lines 38–51) marks a logical point on the *BSD-Release* branch. This is defined as a merge between the last commit in the *BSD-3-Snapshot-Development* branch (marked as 3699 and identified with the *from* command in line 47) and the two preceding snapshots (*Bell-32V* and *BSD-2* — lines 48–49). At this point two *D — filedelete* commands remove the refence file copies that were created at the beginning (lines 50–51).

Finally, a *tag* command (lines 53–59) associates a symbolic name with the release, and the *done* command (line 61) signals the stream's end.

A 620-line shell script (`import.sh`) creates the Git repository and calls the Perl script with appropriate arguments to import each one of the approximately 30 available historical data sources (see Table 2). As an example, consider the following (slightly simplified) invocation.

```
perl ../import-dir.pl $VERBOSE -m Bell-32V,BSD-2 \
  -c ../author-path/BSD-3 -n ../berkeley.au -r Bell-32V,BSD-2 \
  -i ../ignore/BSD-3 -u ../unmatched/BSD-3 $ARCHIVE/3bsd \
  BSD 3 -0800 |
git fast-import --stats --done --quiet
```

The preceding shell command runs the import script over the `3bsd` snapshot to create version 3 of the BSD branch. This will appear as a merge between the tags `Bell-32V` and `BSD-2`, whose files will also be retained until all the snapshot's files have been imported. The file `author-path/BSD-3` specifies the authorship of each file and `berkeley.au` the details of the authors. Files listed in `ignore/BSD-3` will not be imported and files matched with a wildcard (`.*`) authorship pattern will be listed in `unmatched/BSD-3`. The command's output is piped into the `git fast-import` command to convert it into actual Git commits.

For a period in the 1980s, only a subset of the files developed at Berkeley were under SCCS version control. During that period the Unix history repository contains imports of both the SCCS commits, and the snapshots' incremental additions. At the point of each release, the SCCS commit with the nearest time stamp is found and is marked as a merge with the release's incremental import branch. These merges can be seen in the center of Fig. 8.

The import shell script also inserts into all imported versions of Unix diverse licensing files and a file named `README.md` which, among other things, contains the Git SHA sum of the software that created the repository and a timestamp of the import process. Provided the data sources are not modified, this allows the Unix repository to be uniquely identified in a replicable fashion.

The shell script also runs 30 tests that compare the repository at specific tags against the corresponding data sources, verify the appearance and disappearance of look-aside directories, and look for regressions in the count of tree branches and merges and the output of *git blame* and *git log*.

Before pushing the created repository to GitHub, *git* is called to garbage-collect and compress the repository from its initial 6.1GB size down to the distributed 1.1GB.

4 Data Uses

The Unix history repository can be used for empirical research in software engineering, information systems, and software archeology. Through its unique uninterrupted coverage of a period of more than 40 years, it can inform work on software evolution and handovers across generations. With thousandfold increases in processing speed and million-fold increases in storage capacity during that time, the data set can also be used to study the co-evolution of software and hardware technology.

As one concrete example, Fig. 12 depicts trend lines of some interesting code metrics along 36 major releases of Unix. It demonstrates the evolution of code style and programming language use over very long timescales. This evolution can be driven by software and hardware technology affordances and requirements, software construction theory, and even social forces. The Figure was obtained with R's local polynomial regression fitting function. The dates in the Figure have been calculated as the average date of all files appearing in a given release. As can be seen in it, over the past 40 years the mean length of identifiers has steadily increased from 4 characters to 7 and mean length of file names has increased from 6 characters to 11. We can also see less steady increases in the number of comments and decreases in the use of the *goto* statement, as well as the virtual disappearance of the *register* type modifier. Based on these observations made in an exploratory study (Spinellis et al. 2015) a follow-up work (Spinellis et al. 2016) used the Unix history repository to examine seven concrete hypotheses. By extracting, aggregating, and synthesizing metrics from 66 snapshots in the period covered by the repository it was found that over the years developers of the Unix operating system appear to have evolved their coding style in tandem with advancements in hardware technology, promoted modularity to tame rising complexity, adopted valuable new language features, allowed compilers to allocate registers on their behalf, and reached broad agreement regarding code formatting. The reported

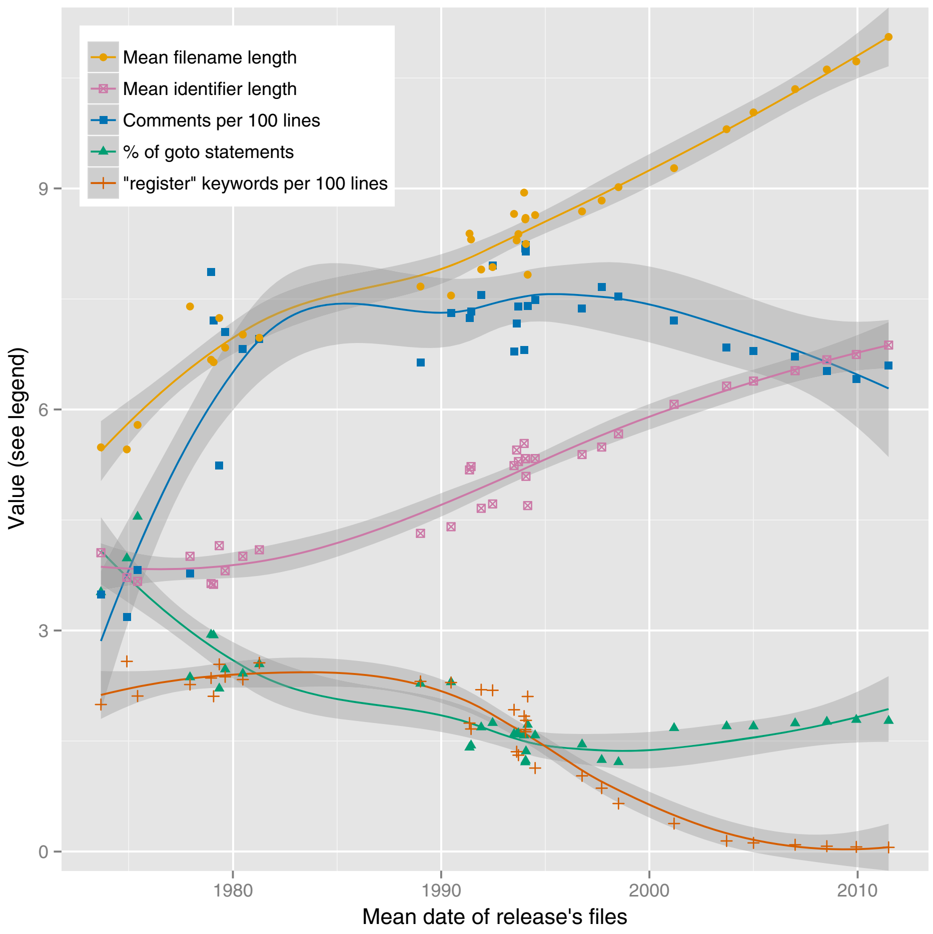


Fig. 12 Code style evolution along Unix releases

work also showed that many trends point toward increasing code quality through adherence to numerous programming guidelines, that some other trends indicate adoption that has reached maturity, and that in the area of code commenting progress appears to have stalled.

As a second example, Fig. 13 shows the distribution of minimum and maximum lifespan estimates of a line of code. The estimates were obtained as follows. First *git blame* (with `-w -C -C -C` parameters) was run on all (1.5 million) source code files of 71 Unix releases selected in the repository. This task used considerable computing resources: 9.9 core years CPU time, 3,815 cores, 7.6 TB RAM, and 588 GB of disk space. Its execution was made possible by running it, as a set of tasks scheduled through SLURM (Yoo et al. 2003), on a supercomputer (IBM NeXtScale nx360M5, Intel Xeon E5-2680v2 10C 2.8 GHz, Infiniband FDR14, 8,520 cores, 170 TFLOP/s). The run associated, with each line of code of each release, a timestamp indicating the time the line was last modified. By identifying the first release where a line of code stopped appearing, it was possible to estimate the minimum and maximum bounds of that line's lifespan in its initial form. The line was considered to "die" (it was removed or modified) sometime between the preceding release and the one where

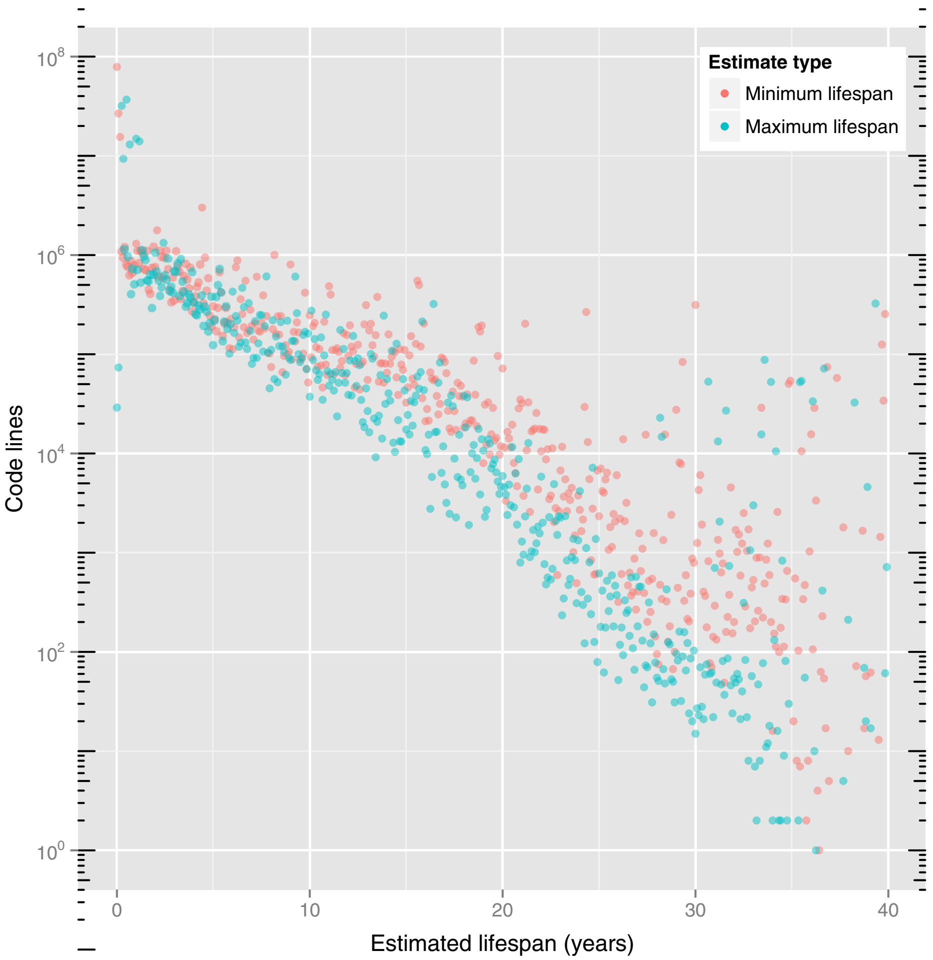


Fig. 13 Exponential decay of Unix source code

it stopped appearing. In total, minimum estimates were obtained for 194 million lines and maximum estimates for 167 million lines. (The minimum estimates also include lines that survived until the last available release.) Linear regression on the logarithm of the surviving lines of code l and their lifespan t indicates ($R^2 = 0.73$; $p = 2.2 \times 10^{-16}$) that the code's decay matches the following exponential model.

$$l = e^{-0.26 \times t}$$

Based on the lifespan's median value, we can bound the half-life of a line of code somewhere between 0.8 and 9.7 years.

Apart from the preceding two concrete examples, many more areas of research present themselves. The move of the software's development from research labs, to academia, and to the open source community can be used to study the effects of organizational culture on software development. In that area an additional branch from Unix 32/V with System III, System v, and *illumos* could trace the evolution of Unix in corporate hands and its transition to another open source community.

The repository can also be used to study how notable individuals, such as Turing Award winners (Dennis Ritchie and Ken Thompson) and captains of the IT industry (Bill Joy and Eric Schmidt), actually programmed. Another phenomenon worthy of study concerns the longevity of code, either at the level of individual lines, or as complete systems that were at times distributed with Unix (Ingres, Lisp, Pascal, Ratfor, Snobol, TMG), as well as the factors that lead to code's survival or demise.

Finally, because the data set stresses Git, the underlying software repository storage technology, to its limits, it can be used to drive engineering progress in the field of revision management systems.

5 Contributing Extensions

The Unix history repository is managed as an open source project. The project can benefit from the addition of authorship information and entirely new data sources. Both can be contributed as changes to the repository containing the creation code; ideally as GitHub pull requests.

Adding authorship data for code that is imported via snapshots involves adding the author's login identifier, full name, and email (if different from the default for the corresponding community) in the author file associated with the repository: `386bsd.au`, `bell.au`, `berkeley.au`, or `freebsd.au`. The fields are colon-separated; see Fig. 11. Then, records must be added in the repository's authorship data file, which is located in the `author-path` directory. Each record consists of a regular expression that matches one or more files in the repository, followed by the login identifier of the files' author (see Fig. 9). For example the following two lines identify Alfred Aho as the author of all files in the `libdbm` directory and Doug McIlroy as the author of the file `join.c`. (Note the escaped "`.`"). Records are matched from top to the bottom of the file, so more specific patterns should be listed before more general ones.

```
usr/src/libdbm/.*             aho
usr/src/cmd/join\.c           doug
```

Both files allow comments starting with a "#" character. The associated comments and commit messages should clearly indicate the attribution's justification, e.g. a pointer to a publication or a timestamped excerpt of a personal communication. Following the change, the consistency of the added data should be verified by running `import.sh -VI`, the Unix history repository should be rebuilt, and the differences in the `unmatched` directory files should be closely examined to verify that they match the change made.

Adding a completely new release data source is more involved. First, note that the corresponding data should be legally available for further redistribution. For example, although various snapshots of System III and beyond seem to be floating around the internet, including them in the repository is not currently possible, because Caldera's license explicitly excludes them. In brief, the steps required are the following.

- Add a `Makefile` rule that will download the data and expand it in a directory within `archive` directory.
- Add the directory's path as a dependency to the `Makefile`'s `all` rule.

- If the data involves a snapshot, add authorship information as detailed in this section's preceding paragraphs.
- Add a list of files that should not be imported (e.g. because they are executables compiled from the repository's source code) in the `ignore` directory. Alternatively, add in `import.sh` a command to generate this list.
- Add a statement in `import.sh` that will graft the data in the appropriate place of the Unix history repository tree.
- Rebuild the history repository.
- Verify that the checked out version of the new data source from the history repository matches the original data.
- Verify through manual inspection that `git blame` and `git log` produce the expected results (especially across releases), and that branches and merges are correctly represented.
- Add corresponding verification rules in the `verify` function located in `import.sh`.

6 Further Work

Many things can be done to increase the repository's faithfulness and usefulness. Given that the build process is shared as open source code, it is easy to contribute additions and fixes through GitHub pull requests. The most useful community contribution would be to increase the coverage of imported snapshot files that are attributed to a specific author. Currently, about 81 thousand snapshot commits (10 % out of a total of 496 thousand commits) are getting assigned an author through a default rule. Similarly, there are about 40 authors (primarily early FreeBSD ones, responsible for 4,974 commits — 1.6 % of the total) for which only the identifier is known. Both are listed in the build repository's `unmatched` directory, and contributions are welcomed. Furthermore, the BSD SCCS and the FreeBSD CVS commits that share the same author and time-stamp can be coalesced into a single Git commit. Support can be added for importing the SCCS file comment fields, in order to bring into the repository the corresponding metadata. Finally, and most importantly, more branches of open source systems can be added, such as *Plan 9 from Bell Labs*, NetBSD OpenBSD, DragonFlyBSD, and *illumos*. Ideally, current right holders of other important historical Unix releases, such as System III, System V, NeXTSTEP, and SunOS, will release their systems under a license that would allow their incorporation into this repository for study.

Acknowledgments The author thanks the many individuals who contributed, directly or indirectly, to the effort. John Cowan, Brian W. Kernighan, Larry McVoy, Doug McIlroy, Jeremy C. Reed, Aharon Robbins, and Marc Rochkind helped with Bell Labs login identifiers. Clem Cole, John Cowan, Era Eriksson, Mary Ann Horton, Warner Losh, Kirk McKusick, Jeremy C. Reed, Ingo Schwarze, Anatole Shaw, and Norman Wilson helped with BSD login identifiers and code authorship information. The historical and current material used in the repository was made available thanks to efforts by the Free BSD Project, Lynne Greer Jolitz, William F. Jolitz, Kirk McKusick, and the Unix Heritage Society. The early Unix editions were released under an BSD-style license thanks to the efforts of Bill Broderick, Paul Hatch, Dion L. Johnson II, Ransom Love, and Warren Toomey. The BSD SCCS import code is based on work by H. Merijn Brand and Jonathan Gray. The *newoldar* program is a result of work by Brandon Creighton and Dan Frasnelli. The First Research Edition Unix was restored by Johan Beiser, Tim Bradshaw, Brantley Coile, Christian David, Alex Garbutt, Hellwig Geisse, Cyrille Lefevre, Ralph Logan, James Markevitch, Doug Merritt, Tim Newsham, Brad Parker, and Warren Toomey.

References

Aho AV, Kernighan BW, Weinberger PJ (1979) Awk—a pattern scanning and processing language. Softw Pract Exper 9(4):267–280

Babaoğlu O, Joy W (1981) Converting a swap-based system to do paging in an architecture lacking page-referenced bits. In: Proceedings of the Eighth ACM symposium on operating systems principles SOSP '81. ACM, New York, pp 78–86. doi:10.1145/800216.806595

Bashkow TR (1972) Study of UNIX. Bell Laboratories memo MH-8234-TRB-mbh. Available online at http://bitsavers.informatik.uni-stuttgart.de/pdf/bellLabs/unix/PreliminaryUnixImplementationDocument_Jun72.pdf. Current September 2015

Bird C, Gourley A, Devanbu P, Gertz M, Swaminathan A (2006) Mining email social networks. In: Proceedings of the 2006 International Workshop on Mining Software Repositories, ACM, New York, NY, USA, MSR '06, pp 137–143. doi:10.1145/1137983.1138016

Bourne SR (1978) The UNIX shell. Bell Syst Tech J 56(6):1971–1990

Bourne SR (1979) An introduction to the UNIX shell. In: UNIX programmer's manual, volume 2—supplementary documents, 7th edn. Bell Telephone Laboratories. Murray Hill

Dolotta TA, Haight RC, Mashey JR (1978) The programmer's workbench. Bell Syst Tech J 56(6):2177–2200

Feldman SI (1979) Make—a program for maintaining computer programs. Softw Pract Exper 9(4):255–265

FreeBSD (2015) FreeBSD Handbook. The FreeBSD Documentation Project, revision 47376 edn, available online, https://www.freebsd.org/doc/handbook/index.html

Gall H, Menzies T, Williams L, Zimmermann T (2014) Software Development Analytics (Dagstuhl Seminar 14261). Dagstuhl Reports 4(6):64–83. doi:10.4230/DagRep.4.6.64. http://drops.dagstuhl.de/opus/volltexte/2014/4763

Gehani N (2003) Bell labs: life in the crown jewel. Silicon Press, Summit

Johnson SC (1975) Yacc—yet another compiler-compiler. Computer Science Technical Report 32. Bell Laboratories, Murray Hill

Johnson SC (1977) Lint, a C program checker. Computer Science Technical Report 65. Bell Laboratories, Murray Hill

Johnson SC, Lesk ME (1978) Language development tools. Bell Syst Tech J 56(6):2155–2176

Johnson SC, Ritchie DM (1978) Portability of C programs and the UNIX system. Bell Syst Tech J 57(6):2021–2048

Jolitz WF, Jolitz LG (1991) Porting UNIX to the 386: a practical approach. Designing a software specification. Dr Dobb's J 16(1)

Kernighan B, Lesk M, Ossanna JJ (1978) UNIX time-sharing system: Document preparation. Bell Syst Techn J 57(6):2115–2135

Kernighan BW (1982) A typesetter-independent TROFF. Computer Science Technical Report 97. Bell Laboratories, Murray Hill, available online at http://cm.bell-labs.com/cm/cs/cstr/97.ps.gz

Kernighan BW, Cherry LL (1974) A system for typesetting mathematics. Computer Science Technical Report 17. Bell Laboratories, Murray Hill

Kernighan BW, Ritchie DM (1979) The M4 macro processor. In: Unix Programmer's Manual (1979) UNIX Programmer's Manual. Volume 2– supplementary documents, 7th edn. Bell Telephone Laboratories, Murray Hill

Lesk M (1979a) Some applications of inverted indexes on the Unix system. In: Unix Programmer's Manual (1979) UNIX Programmer's Manual. Volume 2–Supplementary Documents, 4th edn. Bell Telephone Laboratories, Murray Hill

Lesk ME (1975) Lex—a lexical analyzer generator. Computer Science Technical Report 39. Bell Laboratories, Murray Hill

Lesk ME (1979b) TBL—a program to format tables. In: Unix Programmer's Manual (1979) UNIX Programmer's Manual. Volume 2–Supplementary Documents, 7th edn. Bell Telephone Laboratories, Murray Hill

Lewis A (1956) AT&T settles antitrust case; shares patents. New York Times 16:1

Libes D, Ressler S (1989) Life with UNIX. Prentice Hall, Englewood Cliffs

Lions J (1996) Lions' commentary on Unix 6th edition with source code. Annabooks, Poway

Mashey JR, Smith DW (1976) Documentation tools and techniques. In: Proceedings of the 2Nd international conference on software engineering ICSE '76. IEEE Computer Society Press, Los Alamitos, pp 177–181

McIlroy MD, Pinson EN, Tague BA (1978) UNIX time-sharing system: foreword. Bell Syst Tech J 57(6):1899–1904

McKusick MK (1999) Twenty years of Berkeley Unix: from AT&T-owned to freely redistributable. In: DiBona C, Ockman S, Stone M (eds) Open sources: voices from the open source revolution, O'Reilly, pp 31–46

McKusick MK, Neville-Neil GV (2004) The design and implementation of the FreeBSD operating system. Addison-Wesley, Reading

McMahon LE (1979) SED—a non-interactive text editor. In: Unix Programmer's Manual (1979) UNIX Programmer's Manual. Volume 2–Supplementary Documents, 7th edn. Bell Telephone Laboratories, Murray Hill

Nowitz DA, Lesk ME (1979) A dial-up network of UNIX systems. In: Unix Programmer's Manual (1979) UNIX Programmer's Manual. Volume 2–Supplementary Documents, 7th edn. Bell Telephone Laboratories, Murray Hill

Ossanna JF (1979) NROFF/TROFF user's manual. In: Unix Programmer's Manual (1979) UNIX Programmer's Manual. Volume 2–Supplementary Documents, 7th edn. Bell Telephone Laboratories, Murray Hill

Pike R, Kernighan BW (1984) Program design in the UNIX system environment. AT&T Bell Lab Tech J 63(8):1595–1606

Quarterman JS, Hoskins JC (1986) Notable computer networks. Commun ACM 29(10):932–971

Raymond ES (2003) The art of Unix programming. Addison-Wesley

Resnick P (2008) Internet message format. RFC 5322, RFC Editor. doi:10.17487/RFC5322. http://www.rfc-editor.org/rfc/rfc5322.txt

Ritchie DM (1978) A retrospective. Bell System Technical Journal 56(6):1947–1969

Ritchie DM (1984) The evolution of the UNIX time-sharing system. AT&T Bell Lab Tech J 63(8):1577–1593

Ritchie DM (1993) The development of the C language. ACM SIGPLAN Not 28(3):201–208. preprints of the History of Programming Languages Conference (HOPL-II)

Ritchie DM, Thompson K (1974) The UNIX time-sharing system. Commun ACM 17(7):365–375

Ritchie DM, Thompson K (1978) The UNIX time-sharing system. Bell Syst Tech J 57(6):1905–1929

Ritchie DM, Johnson SC, Lesk ME, Kernighan BW (1978) The C programming language. Bell Syst Tech J 57(6)

Rochkind MJ (1975) The source code control system. IEEE Trans Softw Eng SE 1(4):255–265

Rosler L (1984) The evolution of C — past and future. Bell Syst Tech J 63(8)

Salus PH (1994) A quarter century of UNIX. Addison-Wesley, Boston

Spinellis D (2015) A repository with 44 years of Unix evolution. In: MSR '15: Proceedings of the 12th working conference on mining software repositories. IEEE, pp 462–465. doi:10.1109/MSR.2015.6. http://www.dmst.aueb.gr/dds/pubs/conf/2015-MSR-Unix-History/html/Spi15c.html, best Data Showcase Award

Spinellis D, Louridas P, Kechagia M (2015) An exploratory study on the evolution of C programming in the Unix operating system. In: Wang Q, Ruhe G (eds) ESEM '15: 9th International symposium on empirical software engineering and measurement. IEEE, pp 54–57. doi:10.1109/ESEM.2015.7321190, (to appear in print). http://www.dmst.aueb.gr/dds/pubs/conf/2015-ESEM-CodeStyle/html/SLK15.html

Spinellis D, Louridas P, Kechagia M (2016) The evolution of C programming practices: a study of the Unix operating system. In: Visser W, Williams L (eds) ICSE '16: Proceedings of the 38th international conference on software engineering. Association for Computing Machinery, New York, pp 1973–2015. doi:10.1145/2884781.2884799, (to appear in print). to appear

Stevens WR (1990) UNIX network programming. Prentice Hall, Englewood Cliffs

Stroustrup B (1984) Data abstraction in C. Bell Syst Tech J 63(8):1701–1732

Stroustrup B (1994) The design and evolution of C++. Addison-Wesley, Boston

Takahashi N, Takamatsu T (2013) UNIX license makes Linux the last missing piece of the puzzle. Ann Bus Admin Sci 12:123–137

Tichy WF (1982) Design, implementation, and evaluation of a revision control system. In: Proceedings of the 6th international conference on software engineering. IEEE

Toomey W (2009) The restoration of early UNIX artifacts. In: Proceedings of the 2009 USENIX annual technical conference USENIX'09. USENIX Association, Berkeley, pp 20–26

Toomey W (2010) First edition Unix: its creation and restoration. IEEE Ann Hist Comput 32(3):74–82. doi:10.1109/MAHC.2009.55

Wall L, Schwartz RL (1990) Programming Perl. O'Reilly and Associates, Sebastopol

Yoo AB, Jette MA, Grondona M (2003) SLURM: Simple Linux utility for resource management. In: Feitelson D, Rudolph L, Schwiegelshohn U (eds) JSSPP 03: 9th International workshop on job scheduling strategies for parallel processing. Springer, Berlin Heidelberg, pp 44–60. doi:10.1007/10968987_3, (to appear in print). lecture Notes in Computer Science Volume 2862

Diomidis Spinellis is a Professor in the Department of Management Science and Technology at the Athens University of Economics and Business, Greece. From 2009 to 2011 he served as the Secretary General for Information Systems at the Greek Ministry of Finance. He is the author of two awardwinning books, *Code Reading and Code Quality: The Open Source Perspective*, as well as more than 200 widely-cited scientific papers. He served for a decade as a member of the *IEEE Software* editorial board, authoring the regular "Tools of the Trade" column. Dr. Spinellis has written the *UMLGraph* tool and code that ships with Mac OS X and BSD Unix. He holds an MEng in Software Engineering and a PhD in Computer Science, both from Imperial College London. Dr. Spinellis is a senior member of the ACM and the IEEE. From January 2015 he is serving as the Editor-in-Chief for *IEEE Software*.