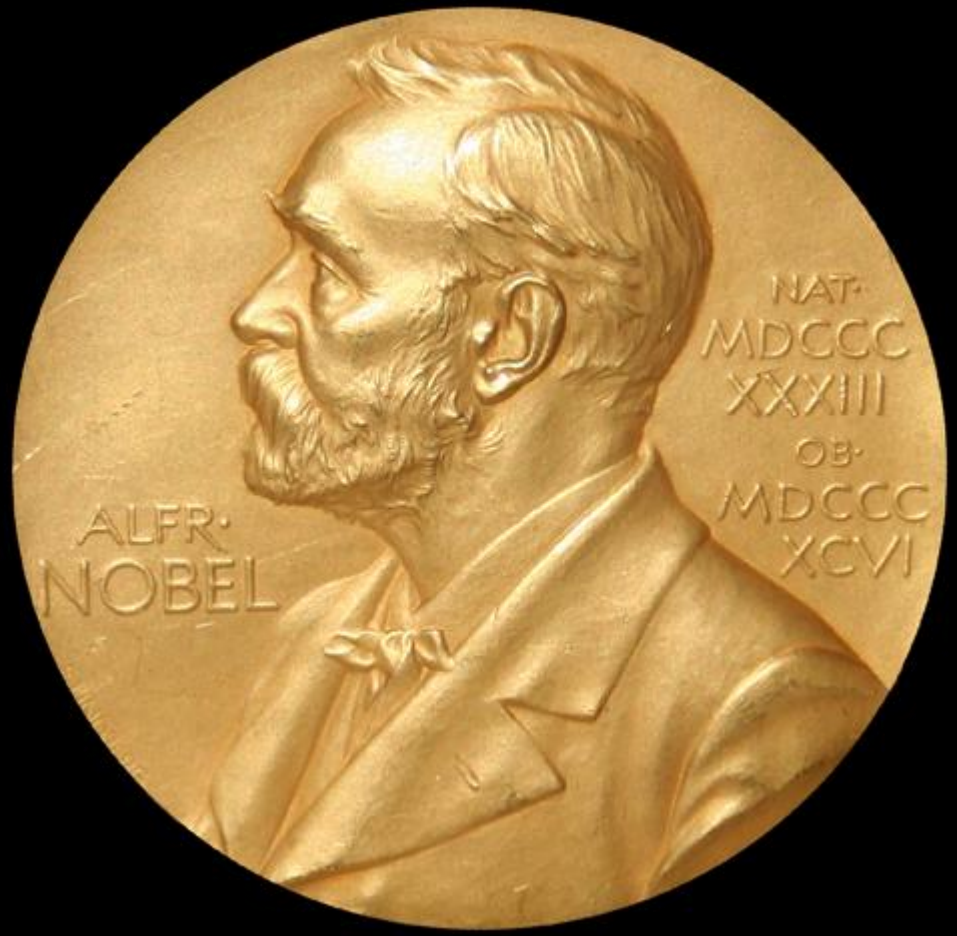
The v4 tape in the Unix history repo

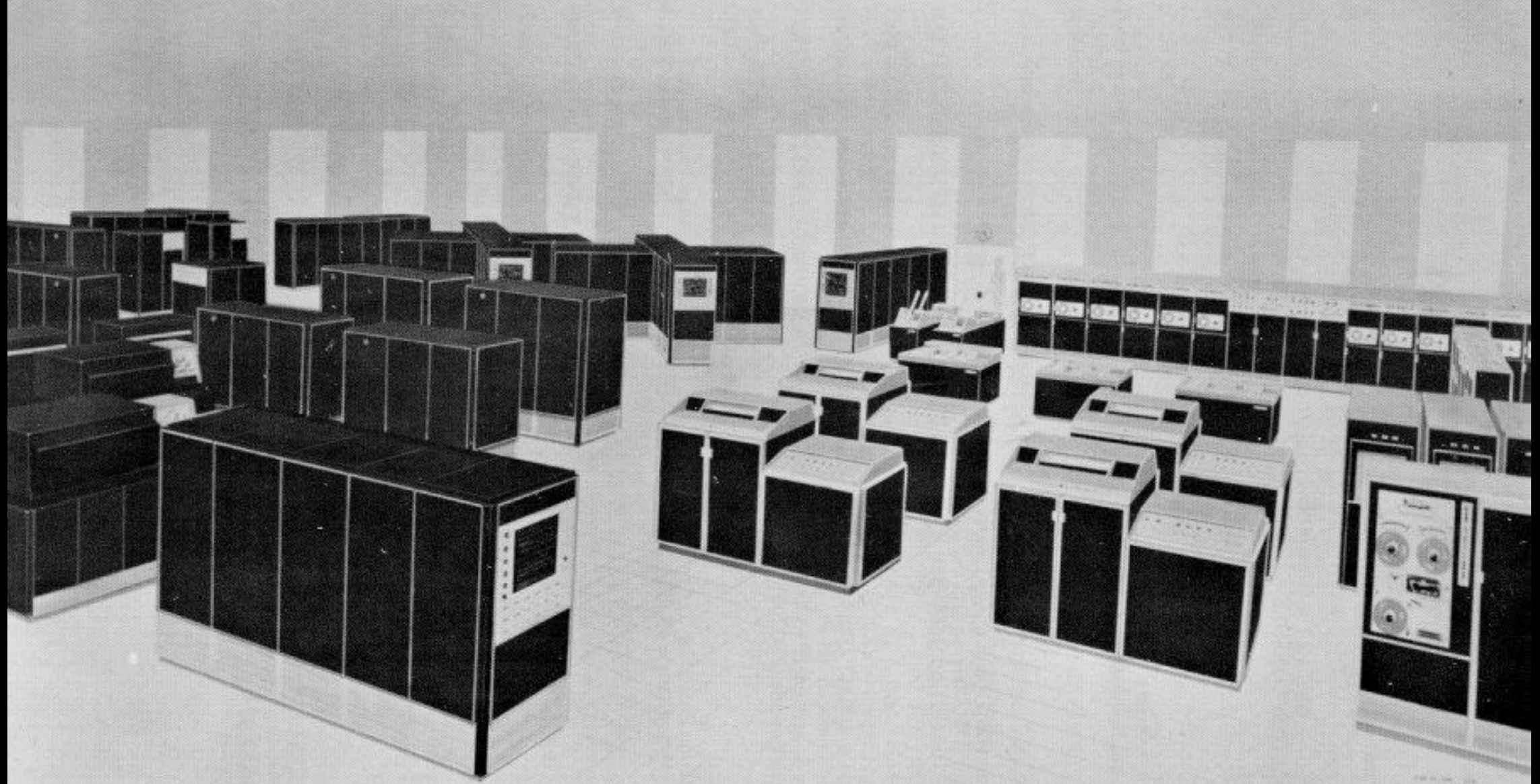
Diomidis Spinellis

Department of Management Science and Technology
Athens University of Economics and Business
&
Department of Software Technology
Delft University of Technology

www.spinellis.gr
DSpinellis
@CoolSWEng
@CoolSWEng@mastodon.acm.org
dds@aueb.gr


ALFR·
NOBEL
NAT·
MDCCC
XXXIII
OB·
MDCCC
XCVI


ACM
A.M. TURING AWARD



GE-645 SYSTEM


JANUS 71

UNIX PROGRAMMER'S MANUAL

K. Thompson

D. M. Ritchie

November 3, 1971

UNIX PROGRAMMER'S MANUAL

Second Edition

K. Thompson

D. M. Ritchie

June 12, 1972

UNIX PROGRAMMER'S MANUAL

Third Edition

K. Thompson

D. M. Ritchie

February, 1973

UNIX PROGRAMMER'S MANUAL

Fourth Edition

K. Thompson

D. M. Ritchie

November, 1973

UNIX PROGRAMMER'S MANUAL

Fifth Edition

K. Thompson
D. M. Ritchie

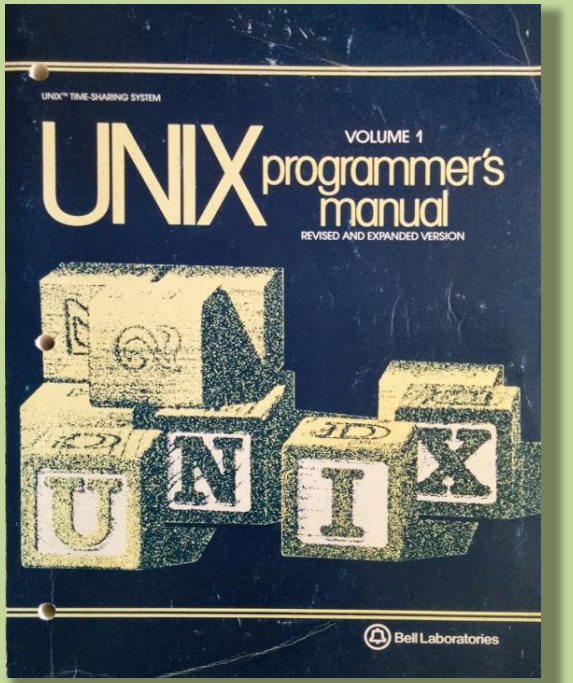
June, 1974

UNIX PROGRAMMER'S MANUAL

Sixth Edition

K. Thompson
D. M. Ritchie

May, 1975



Surviving materials

	v0	v1	v2	v3	v4
Kernel	✓	✓		✓	
Commands	◐	◐	◐		
Documentation				✓	✓
C compiler			✓	✓	

Bell Laboratories

600 Mountain Avenue
Murray Hill, New Jersey 07974
Phone (201) 582-3000

May 31, 1974

MR. MARTIN E. NEWELL
The University of Utah
College of Engineering
Salt Lake City, Utah 84112

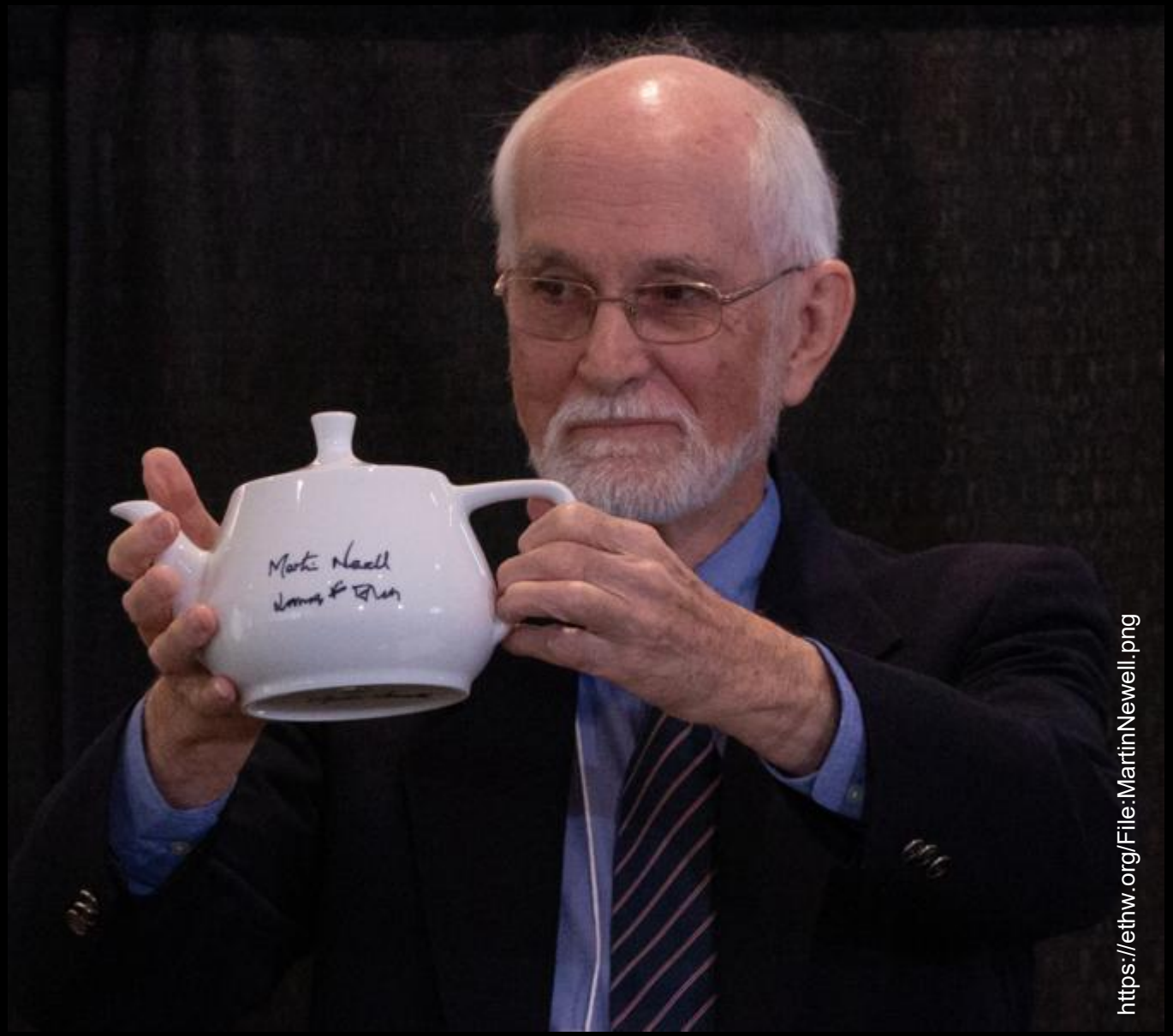
Dear Mr. Newell:

I am sorry for the delay in delivering UNIX. Unfortunately we've run out of copies of documentation. As soon as a new batch comes from the printers I will send you the system.

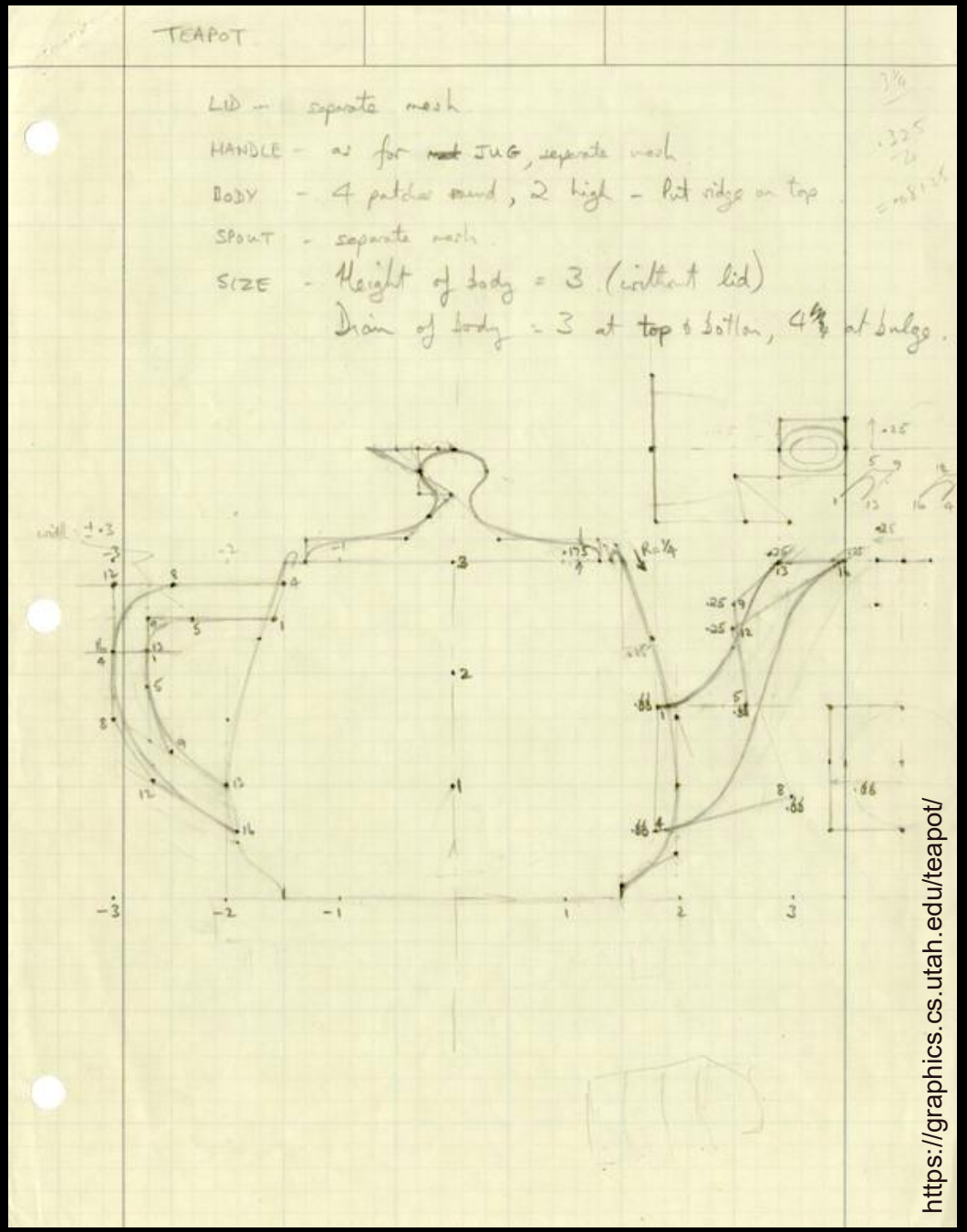
Sincerely,

KEN THOMPSON

KT-gam



https://ethw.org/File:MartinNewell.png



https://graphics.cs.utah.edu/teapot/


UNIX Original
from Bell Labs V4
(See Manual for format)
1200
800
400
Scotch
700
GP
3200 FCI

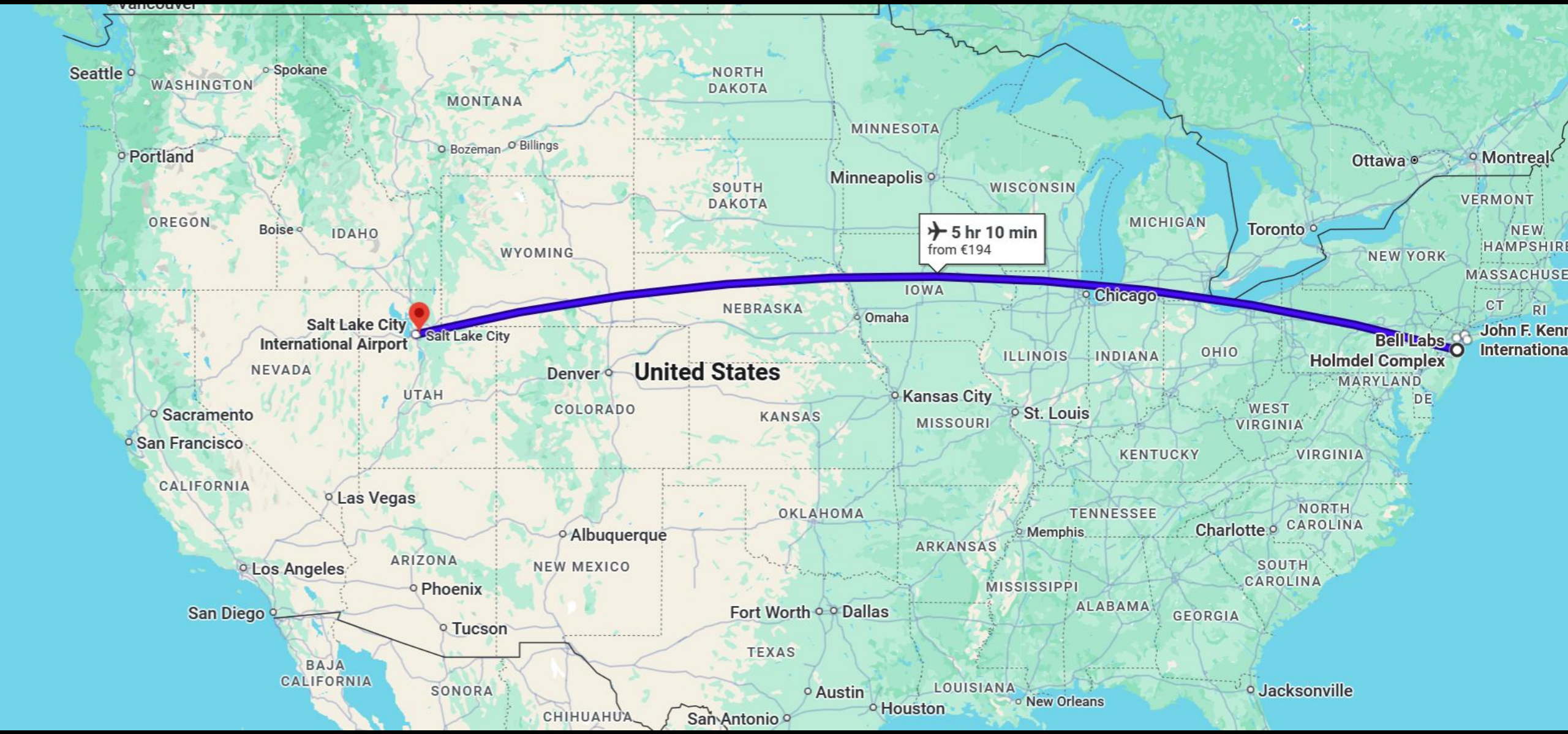
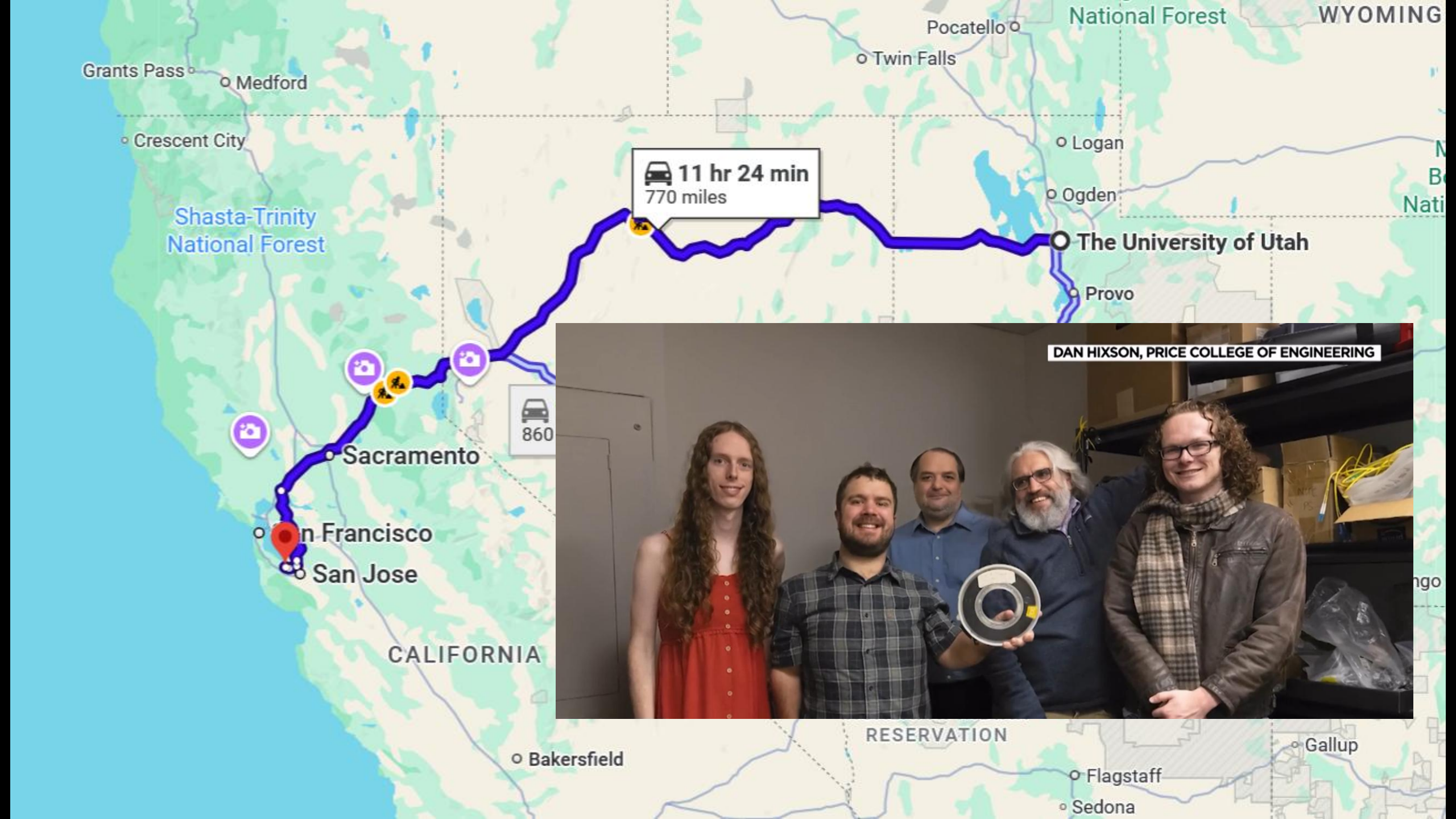
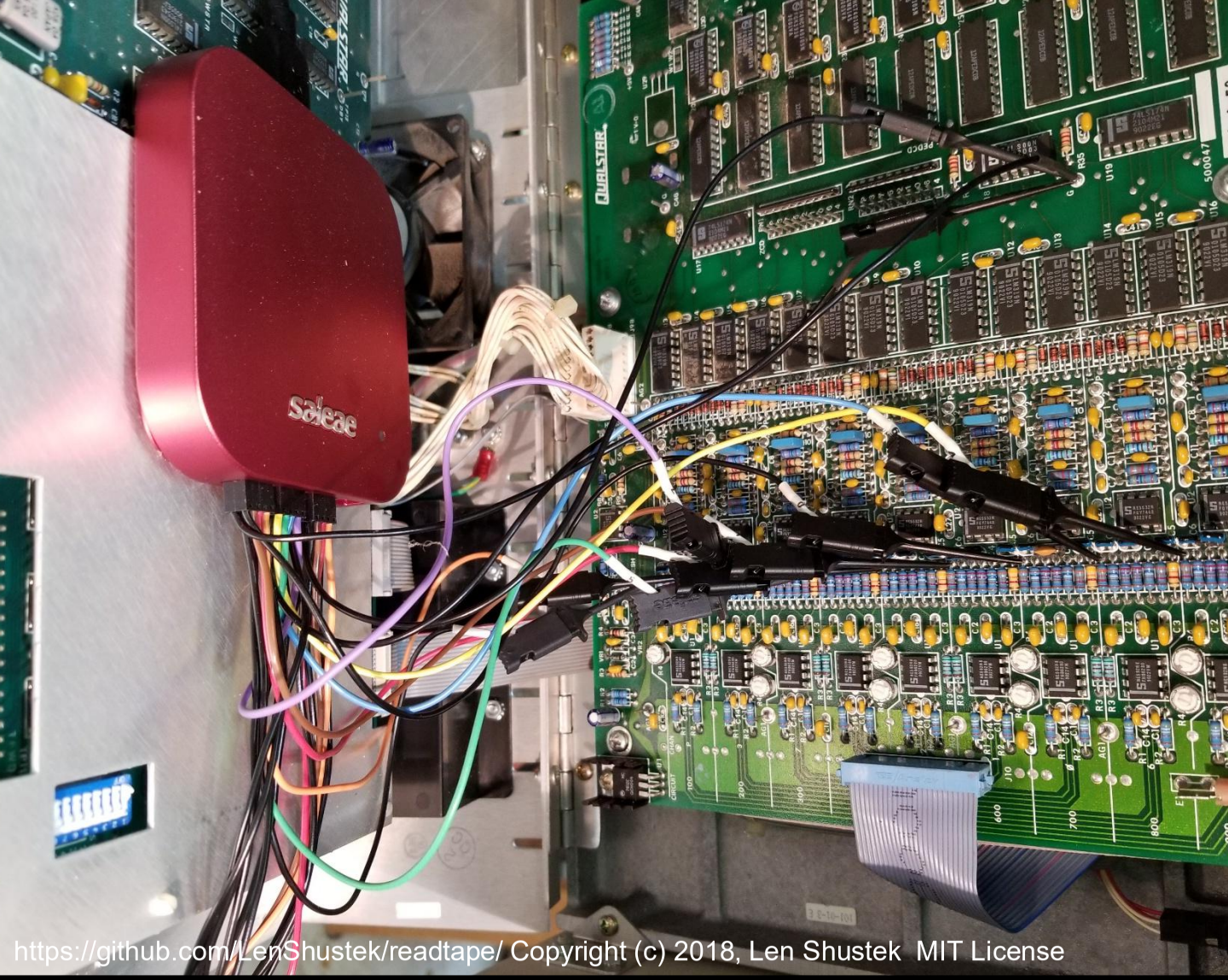
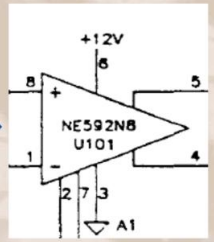

5 hr 10 min
from €194
Salt Lake City International Airport
Salt Lake City
Bell Labs Holmdel Complex
United States
Seattle
Spokane
WASHINGTON
Portland
MONTANA
Bozeman
Billings
NORTH DAKOTA
MINNESOTA
Minneapolis
SOUTH DAKOTA
WISCONSIN
MICHIGAN
Ottawa
Montreal
Toronto
VERMONT
NEW HAMPSHIRE
NEW YORK
MASSACHUSETTS
CT
RI
OREGON
Boise
IDAHO
WYOMING
IOWA
Chicago
NEBRASKA
Omaha
NEVADA
Denver
UTAH
COLORADO
Kansas City
KANSAS
ILLINOIS
INDIANA
OHIO
MARYLAND
DE
WEST VIRGINIA
St. Louis
MISSOURI
KENTUCKY
VIRGINIA
Sacramento
San Francisco
CALIFORNIA
Las Vegas
OKLAHOMA
TENNESSEE
NORTH CAROLINA
Memphis
Charlotte
Albuquerque
ARIZONA
NEW MEXICO
ARKANSAS
Los Angeles
Phoenix
MISSISSIPPI
SOUTH CAROLINA
ALABAMA
GEORGIA
San Diego
Tucson
Fort Worth
Dallas
TEXAS
BAJA CALIFORNIA
SONORA
CHIHUAHUA
Austin
San Antonio
Houston
LOUISIANA
New Orleans
Jacksonville

Photo Credit: Dan Hixson

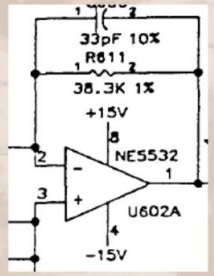

Grants Pass
Medford
Crescent City
Shasta-Trinity
National Forest
Pocatello
Twin Falls
National Forest
WYOMING
Logan
Ogden
11 hr 24 min
770 miles
The University of Utah
Provo
860
Sacramento
n Francisco
San Jose
CALIFORNIA
Bakersfield
RESERVATION
Gallup
Flagstaff
Sedona
DAN HIXSON, PRICE COLLEGE OF ENGINEERING


https://github.com/LenShustek/readtape/ Copyright (c) 2018, Len Shustek MIT License

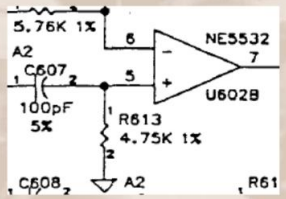

KENNEDY
9610
Computer History Museum



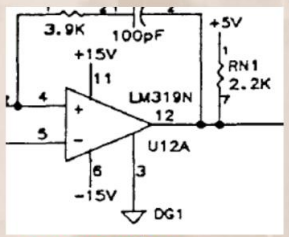
pre-amp



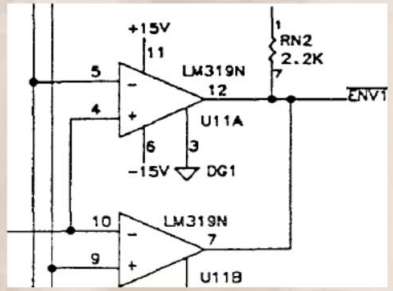
differential amp



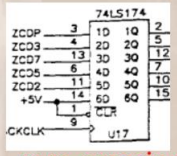
differentiator



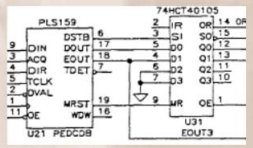
comparator



threshold detector

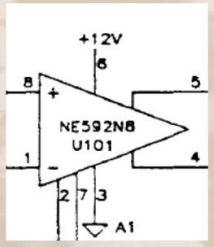


zero-crossing detector



data bit decoder

...and then block and error handling!



pre-amp

analog to digital conversion

detect flux transitions

detect bits

assemble into words, then blocks

OK ?

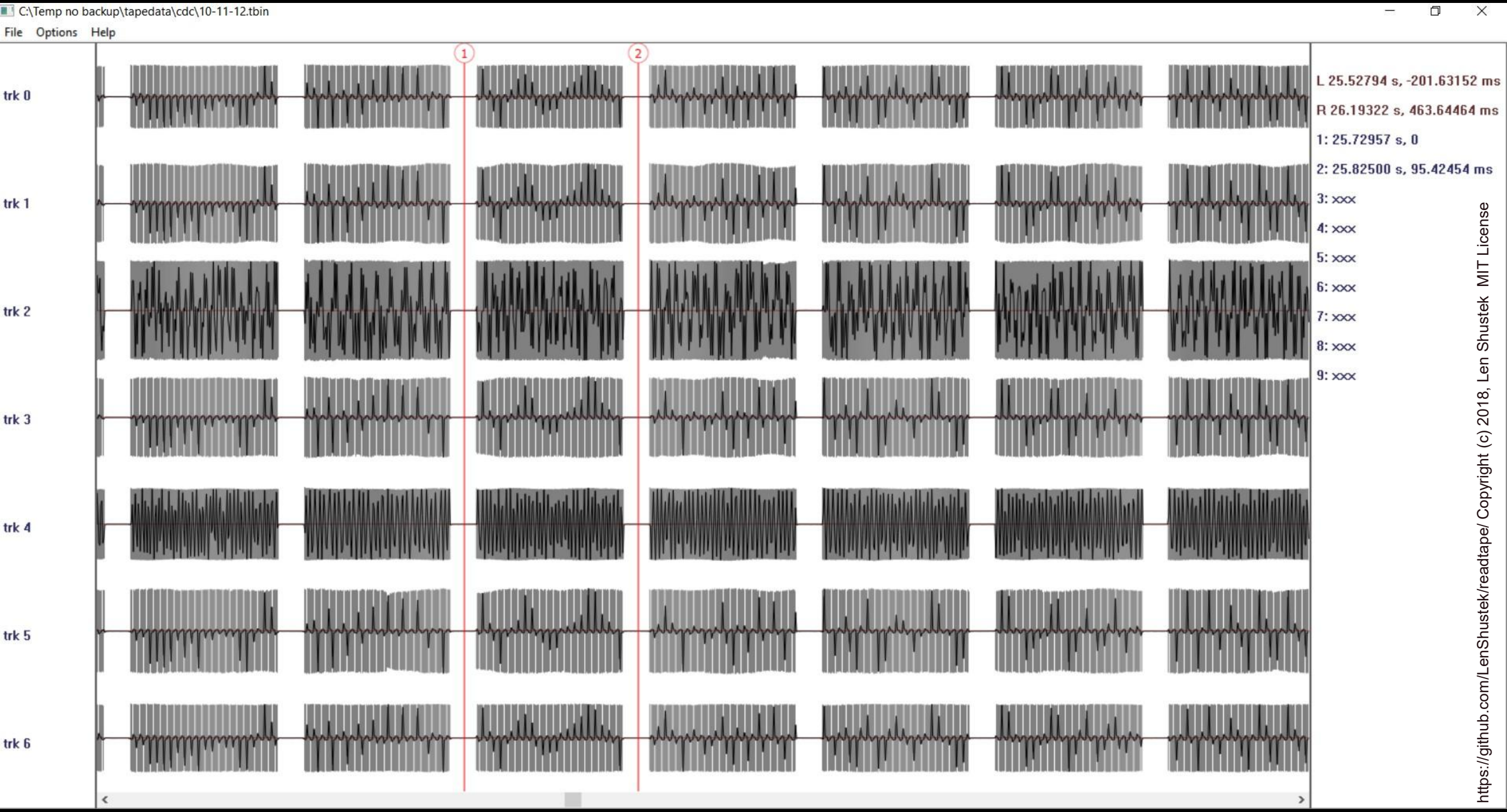
yes

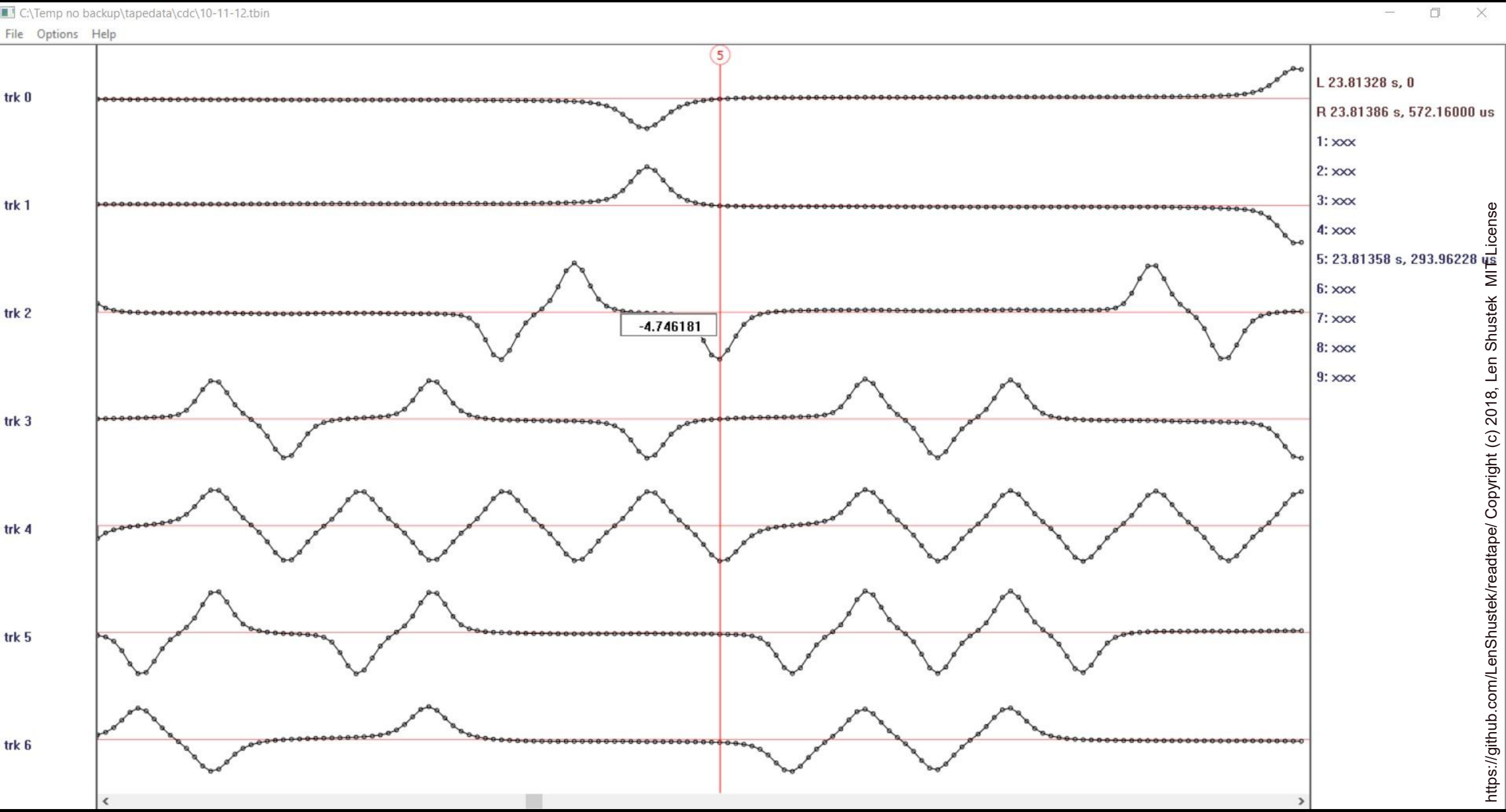
next block

no

in software, so we can retry without rereading the tape!

* Inspired by Paul Pierce's Java-based project in 2004


C:\Temp no backup\tapedata\cdc\10-11-12.tbin
File
Options
Help
trk 0
trk 1
trk 2
trk 3
trk 4
trk 5
trk 6
1
2
L 25.52794 s, -201.63152 ms
R 26.19322 s, 463.64464 ms
1: 25.72957 s, 0
2: 25.82500 s, 95.42454 ms
3: xxx
4: xxx
5: xxx
6: xxx
7: xxx
8: xxx
9: xxx
https://github.com/LenShustek/readtape/ Copyright (c) 2018, Len Shustek MIT License


C:\Temp no backup\tapedata\cdc\10-11-12.tbin
File
Options
Help
5
trk 0
trk 1
trk 2
trk 3
trk 4
trk 5
trk 6
-4.746181
L 23.81328 s, 0
R 23.81386 s, 572.16000 us
1: xxx
2: xxx
3: xxx
4: xxx
5: 23.81358 s, 293.96228 us
6: xxx
7: xxx
8: xxx
9: xxx
https://github.com/LenShustek/readtape/ Copyright (c) 2018, Len Shustek MIT License

Modified	Size	Filename
2002-01-23	12.0K	LICENSE-CALDERA.pdf
2002-01-23 (transcription)	2.8K	LICENSE-CALDERA.txt
2025-12-19 11:04:36 -0800	1.1G	v4/v4.sal
2025-12-19 11:14:40 -0800	1.6G	v4/analog.tbin
2025-12-19 11:14:40 -0800	540B	v4/analog.csvtbin.log
2025-12-19 11:23:08 -0800	3.6K	v4/analog.peakstats_deskew.csv
2025-12-19 11:26:36 -0800	2.5M	v4/analog.tap
2025-12-19 11:26:36 -0800	5.7K	v4/analog.log
2025-12-19 11:26:36 -0800	3.7K	v4/analog.peakstats.csv

Obverse label (handwritten, probably by Jay Lepreau):

UNIX Original
from Bell Labs V4
(See Manual for format)

Obverse label (printed, color):

Scotch®
BRAND
700
GP
3200 FCI

https://archive.org/details/utah_unix_v4_raw

240 West Center Street
Orem, Utah 84057
801-765-4999 Fax 801-765-4481

January 23, 2002

Dear UNIX® enthusiasts,

Caldera International, Inc. hereby grants a fee free license that includes the rights use, modify and distribute this named source code, including creating derived binary products created from the source code. The source code for which Caldera International, Inc. grants rights are limited to the following UNIX Operating Systems that operate on the 16-Bit PDP-11 CPU and early versions of the 32-Bit UNIX Operating System, with specific exclusion of UNIX System III and UNIX System V and successor operating systems:

32-bit 32V UNIX
16 bit UNIX Versions 1, 2, 3, 4, 5, 6, 7

Caldera International, Inc. makes no guarantees or commitments that any source code is available from Caldera International, Inc.

The following copyright notice applies to the source code files for which this license is granted.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code and documentation must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

All advertising materials mentioning features or use of this software must display the following acknowledgement:

This product includes software developed or owned by Caldera International, Inc.

Neither the name of Caldera International, Inc. nor the names of other contributors may be used to endorse or promote products derived from this software without specific prior written permission.

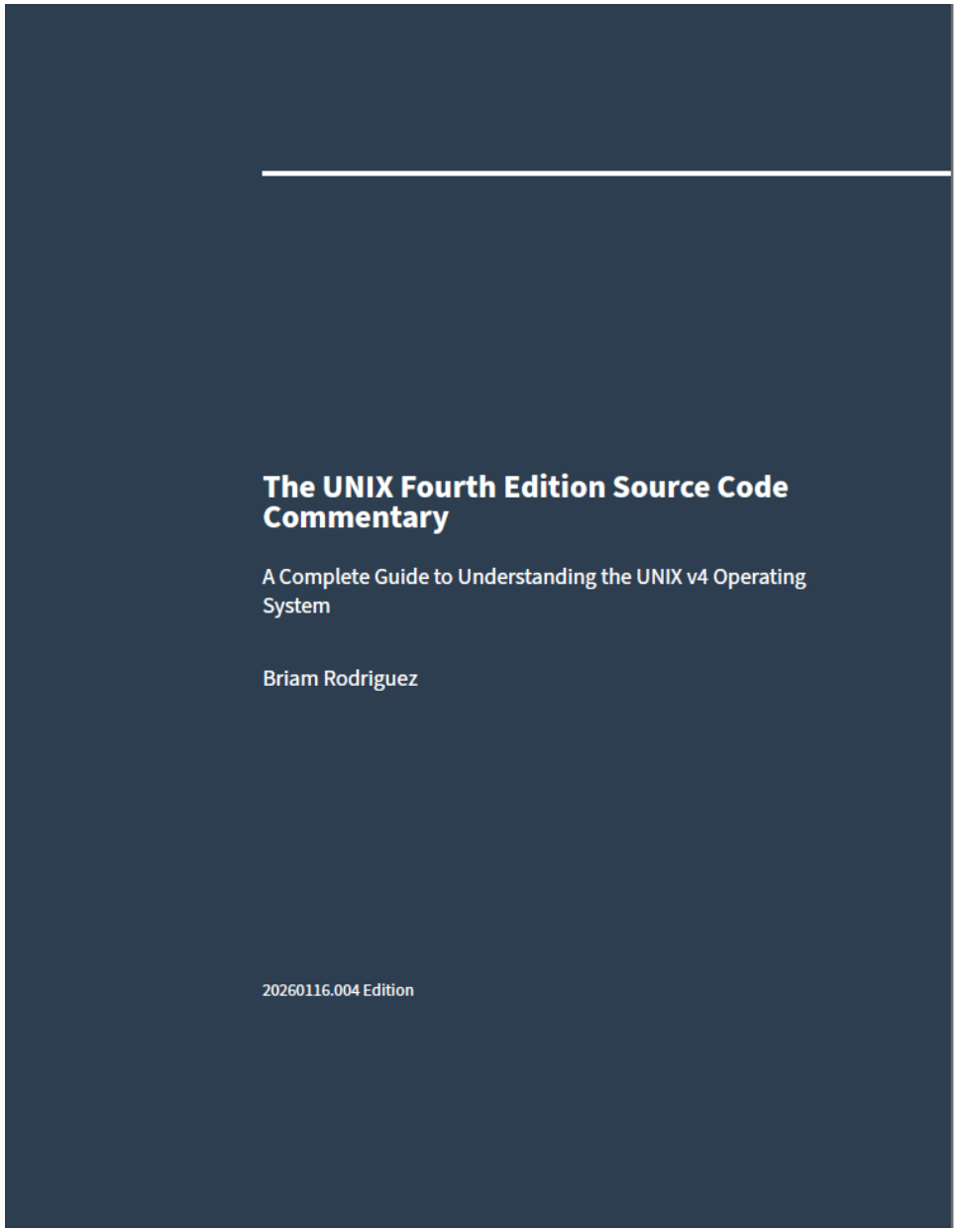
USE OF THE SOFTWARE PROVIDED FOR UNDER THIS LICENSE BY CALDERA INTERNATIONAL, INC. AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL CALDERA INTERNATIONAL, INC. BE LIABLE FOR ANY DIRECT, INDIRECT INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Very truly yours,

/signed/ Bill Broderick

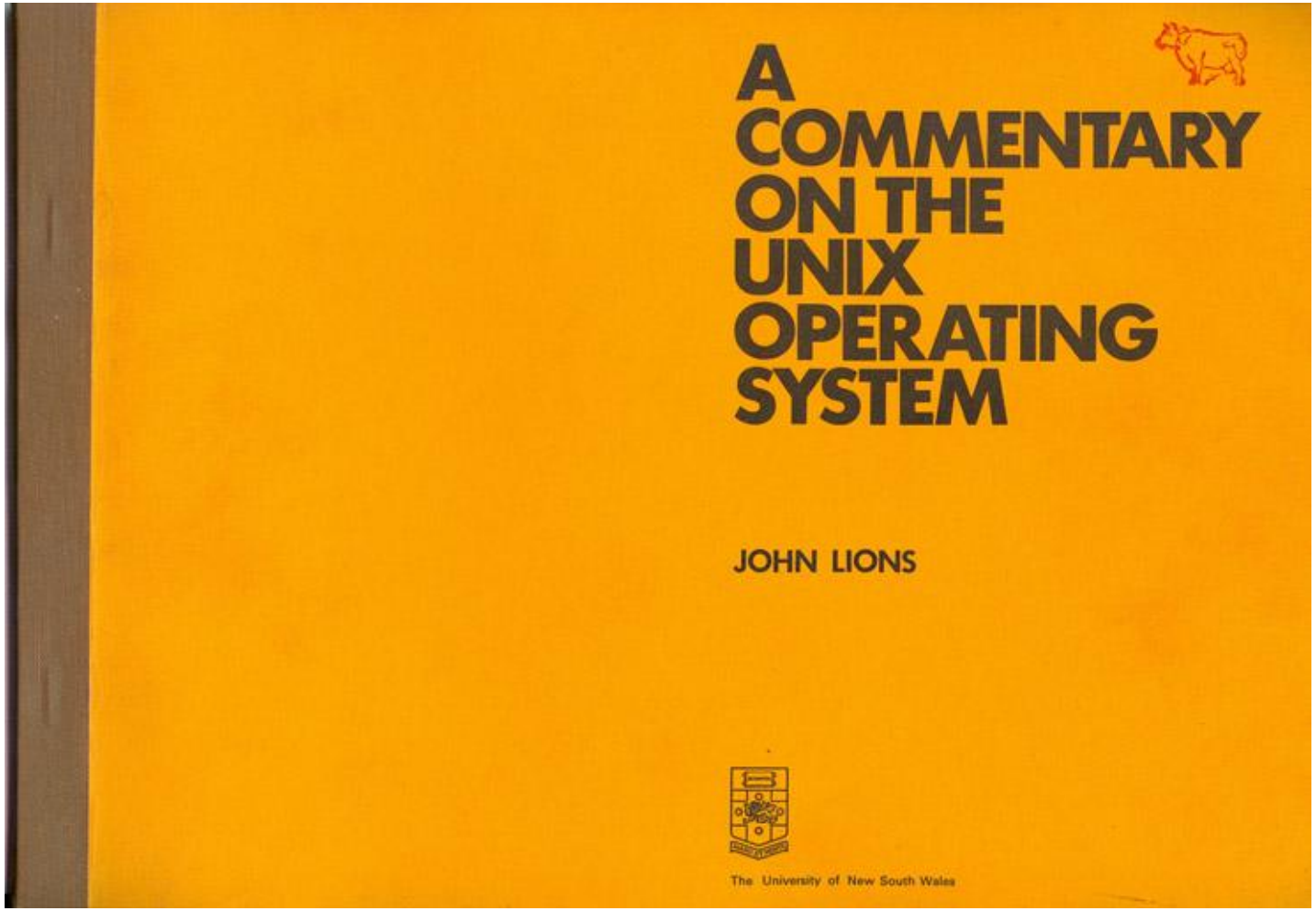
Bill Broderick
Director, Licensing Services

github.com/unix-v4-commentary/unix-v4-source-commentary



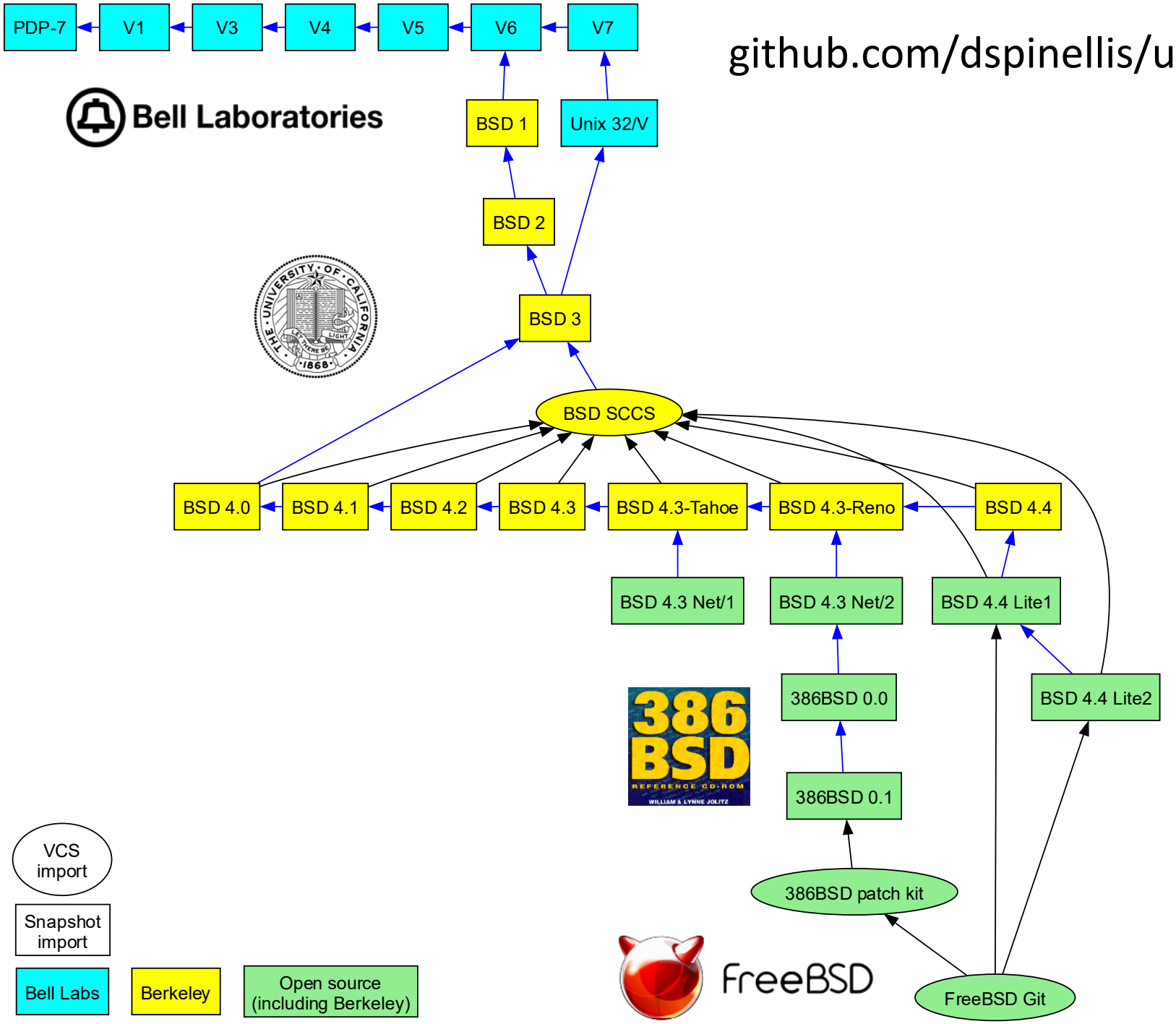
Contents

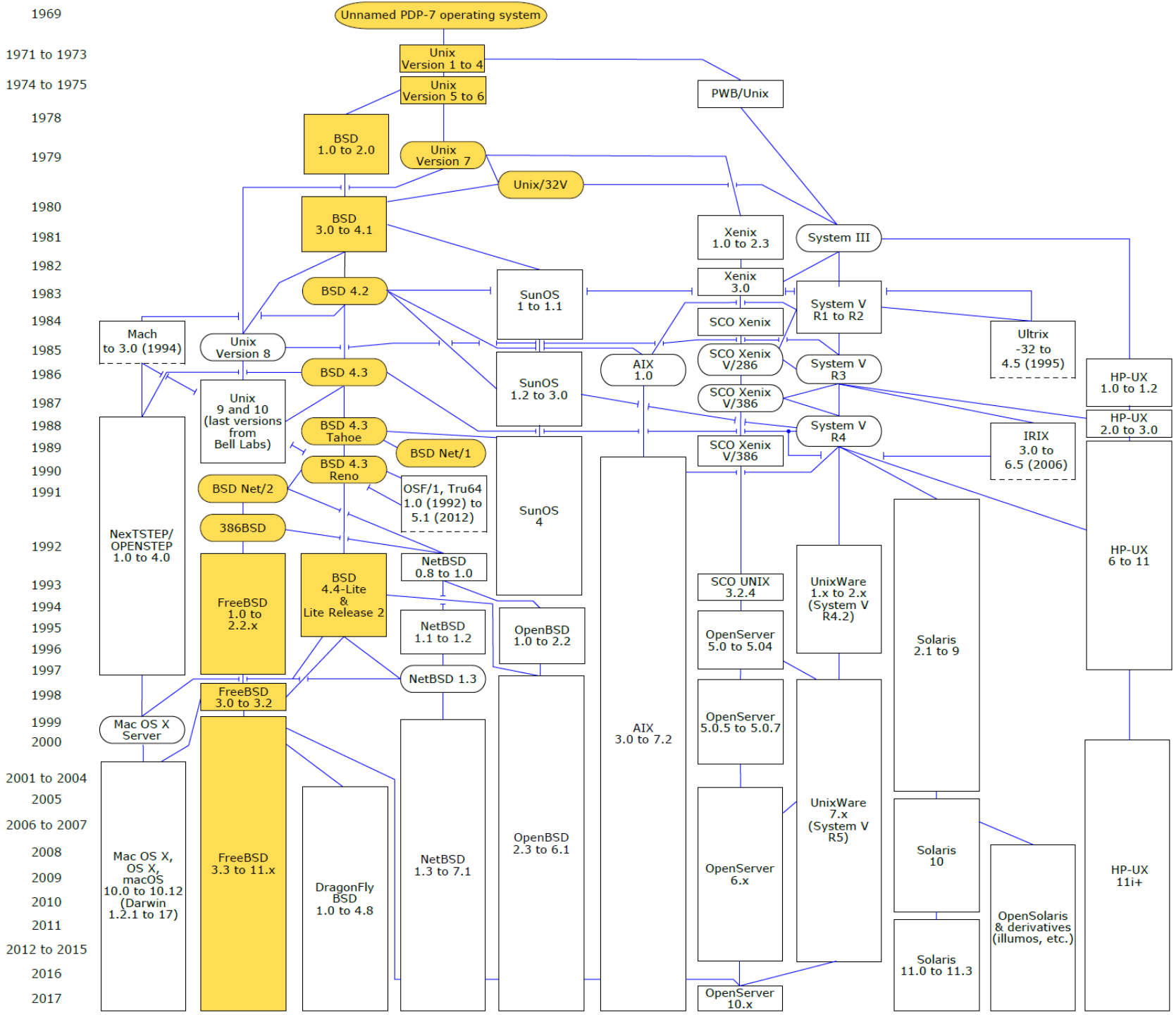
Sixth Research Edition (May 1975)

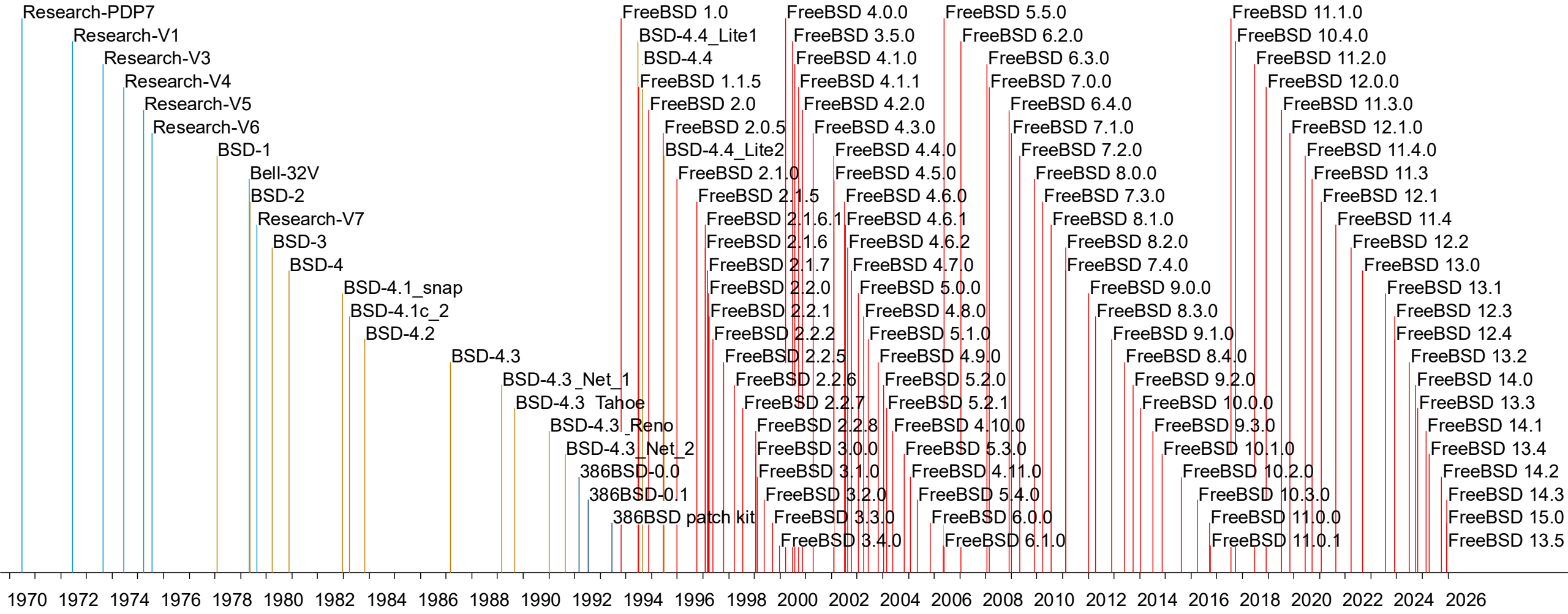



39C3

github.com/dspinellis/unix-history-repo/




1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2015
2016
2017
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
PWB/Unix
BSD 1.0 to 2.0
Unix Version 7
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
SunOS 1 to 1.1
System V R1 to R2
SCO Xenix
Mach to 3.0 (1994)
Unix Version 8
Ultrix -32 to 4.5 (1995)
SCO Xenix V/286
AIX 1.0
System V R3
BSD 4.3
SunOS 1.2 to 3.0
HP-UX 1.0 to 1.2
SCO Xenix V/386
HP-UX 2.0 to 3.0
Unix 9 and 10 (last versions from Bell Labs)
BSD 4.3 Tahoe
BSD Net/1
System V R4
IRIX 3.0 to 6.5 (2006)
SCO Xenix V/386
BSD 4.3 Reno
BSD Net/2
OSF/1, Tru64 1.0 (1992) to 5.1 (2012)
SunOS 4
386BSD
NetBSD 0.8 to 1.0
NextSTEP/ OPENSTEP 1.0 to 4.0
HP-UX 6 to 11
SCO UNIX 3.2.4
UnixWare 1.x to 2.x (System V R4.2)
FreeBSD 1.0 to 2.2.x
BSD 4.4-Lite & Lite Release 2
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
Solaris 2.1 to 9
NetBSD 1.3
FreeBSD 3.0 to 3.2
OpenServer 5.0.5 to 5.0.7
AIX 3.0 to 7.2
Mac OS X Server
UnixWare 7.x (System V R5)
Solaris 10
OpenServer 6.x
Mac OS X, OS X, macOS 10.0 to 10.12 (Darwin 1.2.1 to 17)
FreeBSD 3.3 to 11.x
DragonFly BSD 1.0 to 4.8
NetBSD 1.3 to 7.1
OpenBSD 2.3 to 6.1
OpenSolaris & derivatives (illumos, etc.)
HP-UX 11i+
Solaris 11.0 to 11.3
OpenServer 10.x


Research-PDP7
Research-V1
Research-V3
Research-V4
Research-V5
Research-V6
BSD-1
Bell-32V
BSD-2
Research-V7
BSD-3
BSD-4
BSD-4.1_snap
BSD-4.1c_2
BSD-4.2
BSD-4.3
BSD-4.3_Net_1
BSD-4.3_Tahoe
BSD-4.3_Reno
BSD-4.3_Net_2
386BSD-0.0
386BSD-0.1
386BSD patch kit
FreeBSD 1.0
BSD-4.4_Lite1
BSD-4.4
FreeBSD 1.1.5
FreeBSD 2.0
FreeBSD 2.0.5
BSD-4.4_Lite2
FreeBSD 2.1.0
FreeBSD 2.1.5
FreeBSD 2.1.6.1
FreeBSD 2.1.6
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 3.1.0
FreeBSD 3.2.0
FreeBSD 3.3.0
FreeBSD 3.4.0
FreeBSD 4.0.0
FreeBSD 3.5.0
FreeBSD 4.1.0
FreeBSD 4.1.1
FreeBSD 4.2.0
FreeBSD 4.3.0
FreeBSD 4.4.0
FreeBSD 4.5.0
FreeBSD 4.6.0
FreeBSD 4.6.1
FreeBSD 4.6.2
FreeBSD 4.7.0
FreeBSD 5.0.0
FreeBSD 4.8.0
FreeBSD 5.1.0
FreeBSD 4.9.0
FreeBSD 5.2.0
FreeBSD 5.2.1
FreeBSD 4.10.0
FreeBSD 5.3.0
FreeBSD 4.11.0
FreeBSD 5.4.0
FreeBSD 6.0.0
FreeBSD 6.1.0
FreeBSD 5.5.0
FreeBSD 6.2.0
FreeBSD 6.3.0
FreeBSD 7.0.0
FreeBSD 6.4.0
FreeBSD 7.1.0
FreeBSD 7.2.0
FreeBSD 8.0.0
FreeBSD 7.3.0
FreeBSD 8.1.0
FreeBSD 8.2.0
FreeBSD 7.4.0
FreeBSD 9.0.0
FreeBSD 8.3.0
FreeBSD 9.1.0
FreeBSD 8.4.0
FreeBSD 9.2.0
FreeBSD 10.0.0
FreeBSD 9.3.0
FreeBSD 10.1.0
FreeBSD 10.2.0
FreeBSD 10.3.0
FreeBSD 11.0.0
FreeBSD 11.0.1
FreeBSD 11.1.0
FreeBSD 10.4.0
FreeBSD 11.2.0
FreeBSD 12.0.0
FreeBSD 11.3.0
FreeBSD 12.1.0
FreeBSD 11.4.0
FreeBSD 11.3
FreeBSD 12.1
FreeBSD 11.4
FreeBSD 12.2
FreeBSD 13.0
FreeBSD 13.1
FreeBSD 12.3
FreeBSD 12.4
FreeBSD 13.2
FreeBSD 14.0
FreeBSD 13.3
FreeBSD 14.1
FreeBSD 13.4
FreeBSD 14.2
FreeBSD 14.3
FreeBSD 15.0
FreeBSD 13.5
1970 1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018 2020 2022 2024 2026

```
$ git checkout FreeBSD-release/10.0.0
$ git blame -M -M -C -C ./lib/libc/gen/timezone.c

usr/src/libc/gen/timezone.c     (Dennis Ritchie     1979-01-10 14:58:45 -0500  76) static struct zone {
usr/src/libc/gen/timezone.c     (Dennis Ritchie     1979-01-10 14:58:45 -0500  77)     int    offset;
usr/src/libc/gen/timezone.c     (Dennis Ritchie     1979-01-10 14:58:45 -0500  78)     char    *stdzone;
usr/src/libc/gen/timezone.c     (Dennis Ritchie     1979-01-10 14:58:45 -0500  79)     char    *dlzone;
usr/src/libc/gen/timezone.c     (Dennis Ritchie     1979-01-10 14:58:45 -0500  80) } zonetab[] = {
lib/libc/gen/timezone.c         (Jordan K. Hubbard  1996-07-12 18:57:58 +0000  81)     {-1*60,    "MET",    "MET DST"},
[...]
lib/libc/gen/timezone.c         (Jordan K. Hubbard  1996-07-12 18:57:58 +0000  96)     {-1}
usr/src/lib/libc/gen/timezone.c (Bill Joy           1980-12-22 00:40:25 -0800  97) };
usr/src/lib/libc/gen/timezone.c (Bill Joy           1980-12-22 00:40:25 -0800  98)
usr/src/lib/libc/gen/timezone.c (Keith Bostic       1987-03-28 19:27:07 -0800 106) char *
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)         }
usr/src/lib/libc/gen/timezone.c (Keith Bostic       1987-03-28 19:27:07 -0800 119)
usr/src/lib/libc/gen/timezone.c (Keith Bostic       1987-03-28 19:27:07 -0800 120)     if (zone < 0) {
/* create one */
usr/src/lib/libc/gen/timezone.c (Bill Joy           1980-12-22 00:40:25 -0800 121)         zone = -zone;
usr/src/lib/libc/gen/timezone.c (Keith Bostic       1987-03-28 19:27:07 -0800 122)         sign = '+';
usr/src/lib/libc/gen/timezone.c (Keith Bostic       1987-03-28 19:27:07 -0800 123)     }
usr/src/lib/libc/gen/timezone.c (Keith Bostic       1987-03-28 19:27:07 -0800 124)     else
usr/src/lib/libc/gen/timezone.c (Keith Bostic       1987-03-28 19:27:07 -0800 125)         sign = '-';
lib/libc/gen/timezone.c         (Warner Losh        1998-01-21 21:46:36 +0000 126)     (void)snprintf(czone,
sizeof(czone),
lib/libc/gen/timezone.c         (Warner Losh        1998-01-21 21:46:36 +0000 127)         "GMT%c%d:%02d",sign,zone /
60,zone % 60);
lib/libc/gen/timezone.c         (Rodney Grimes      1994-05-27 05:00:24 +0000 128)     return(czone);
lib/libc/gen/timezone.c         (Rodney Grimes      1994-05-27 05:00:24 +0000 129) }
```

usr/source/s1/form.\.s rhm,llc
usr/source/s1/getty\.s dmr,ken,jfo
usr/source/s1/glob\.c dmr
usr/source/s1/goto\.c dmr
usr/source/s1/grep\.s ken,lem
usr/source/s1/ld.\.s dmr
usr/source/s2/mail\.c ken
usr/source/s2/sa\.c ken
usr/source/s2/sh\.c ken
usr/source/s2/sort\.c ken,doug
usr/source/s2/strip\.s dmr
usr/source/s2/sum\.s ken
usr/source/s2/tee\.c doug
usr/source/s2/tr\.c doug
usr/source/s2/uniq\.c ken
usr/source/s2/wc\.c jfo

Demo time!

https://unixv4.dev/

First available **complete** Unix distribution

	v1	v2	v3	v4
Released	Nov 71	Jun 72	Feb 73	Nov 73
Kernel	✓		✓	✓
Commands	◐	◐		✓
Man pages			✓	✓
C compiler		✓	✓	✓

Structured Programming

- Kernel implemented in “New B”
- 7883 lines New B
- 1054 lines PDP-11 assembly

```
main()
{
        extern schar;
        extern char end[], data[], etext[];
        int i, i1, *p;

        /*
         * zero and free all of core
         */

        UISA->r[0] = KISA->r[6] + USIZE;
        UISD->r[0] = 6;
        for(; fubyte(0) >= 0; UISA->r[0]++) {
                clearseg(UISA->r[0]);
                mfree(coremap, 1, UISA->r[0]);
        }
        mfree(swapmap, NSWAP, SWPLO);

        /*
         * set up system process
         */

        proc[0].p_addr = KISA->r[6];
        proc[0].p_size = USIZE;
        proc[0].p_stat = SRUN;
        proc[0].p_flag =| SLOAD|SSYS;
        u.u_procp = &proc[0];

        /*
         * set up 'known' i-nodes
         */

        sureg();
        LKS->integ = 0115;
        cinit();
        binit();
        iinit();
        rootdir = iget(ROOTDEV, ROOTINO);
        rootdir->i_flag =& ~ILOCK;
        u.u_cdir = iget(ROOTDEV, ROOTINO);
        u.u_cdir->i_flag =& ~ILOCK;
```

Language-Independent API

PIPE (II) 8/5/73 PIPE (II)

NAME

pipe – create a pipe

SYNOPSIS

(pipe = 42.)
sys pipe
(read file descriptor in r0)
(write file descriptor in r1)

pipe(fildes)
int fildes[2];

DESCRIPTION

The *pipe* system call creates an I/O mechanism called a pipe. The file descriptors returned can be used in read and write operations. When the pipe is written using the descriptor returned in r1 (resp. fildes[1]), up to 4096 bytes of data are buffered before the writing process is suspended. A read using the descriptor returned in r0 (resp. fildes[0]) will pick up the data.

It is assumed that after the pipe has been set up, two (or more) cooperating processes (created by subsequent *fork* calls) will pass data through the pipe with *read* and *write* calls.

The shell has a syntax to set up a linear array of processes connected by pipes.

Read calls on an empty pipe (no buffered data) with only one end (all write file descriptors closed) return an end-of-file. Write calls under similar conditions are ignored.

Data Structure Definition & Reuse

```
buf.h    file.h     inode.h   proc.h   seg.h     text.h   user.h
conf.h   filsys.h   param.h   reg.h    systm.h   tty.h
```

Device Driver Abstraction

IV. SPECIAL FILES

cat	phototypesetter interface
da	voice response unit
dc	DC-11 communications interface
dn	dn11 ACU interface
dp	dp11 201 data-phone interface
kl	KL-11/TTY-33 console typewriter
mem	core memory
pc	PC-11 paper tape reader/punch
rf	RF11/RS11 fixed-head disk file
rk	RK-11/RK03 (or RK05) disk
rp	RP-11/RP03 moving-head disk
tc	TC-11/TU56 DECtape
tiu	Spider interface
tm	TM-11/TU-10 magtape interface
vs	voice synthesizer interface
vt	11/20 (vt01) interface

Driver Interface

```
struct {
        int     (*d_open)();
        int     (*d_close)();
        int     (*d_strategy)();
        int     *d_tab;
} bdevsw[];

struct {
        int     (*d_open)();
        int     (*d_close)();
        int     (*d_read)();
        int     (*d_write)();
        int     (*d_sgtty)();
} cdevsw[];
```

UNIX PROGRAMMER'S MANUAL

Third Edition

K. Thompson

D. M. Ritchie

February, 1973

UNIX PROGRAMMER'S MANUAL

Fourth Edition

K. Thompson

D. M. Ritchie

November, 1973

SNO (I) 2/9/73 SNO (I)

NAME

sno – Snobol interpreter

SYNOPSIS

sno [file]

DESCRIPTION

Sno is a Snobol III (with slight differences) compiler and interpreter. *Sno* obtains input from the concatenation of *file* and the standard input. All input through a statement containing the label 'end' is considered program and is compiled. The rest is available to 'syspit'.

Sno differs from Snobol III in the following ways.

There are no unanchored searches. To get the same effect:

```
a ** b          unanchored search for b
a *x* b = x c   unanchored assignment
```

There is no back referencing.

```
x = "abc"
a *x* x         is an unanchored search for 'abc'
```

Function declaration is different. The function declaration is done at compile time by the use of the label 'define'. Thus there is no ability to define functions at run time and the use of the name 'define' is preempted. There is also no provision for automatic variables other than the parameters. For example:

definef( )

or

define f(a,b,c)

All labels except 'define' (even 'end') must have a non-empty statement.

If 'start' is a label in the program, program execution will start there. If not, execution begins with the first executable statement. 'define' is not an executable statement.

There are no builtin functions.

Parentheses for arithmetic are not needed. Normal precedence applies. Because of this, the arithmetic operators '/' and '*' must be set off by space.

The right side of assignments must be non-empty.

Either ´ or " may be used for literal quotes.

The pseudo-variable 'sysppt' is not available.

SEE ALSO

Snobol III manual. (JACM; Vol. 11 No. 1; Jan 1964; pp 21)

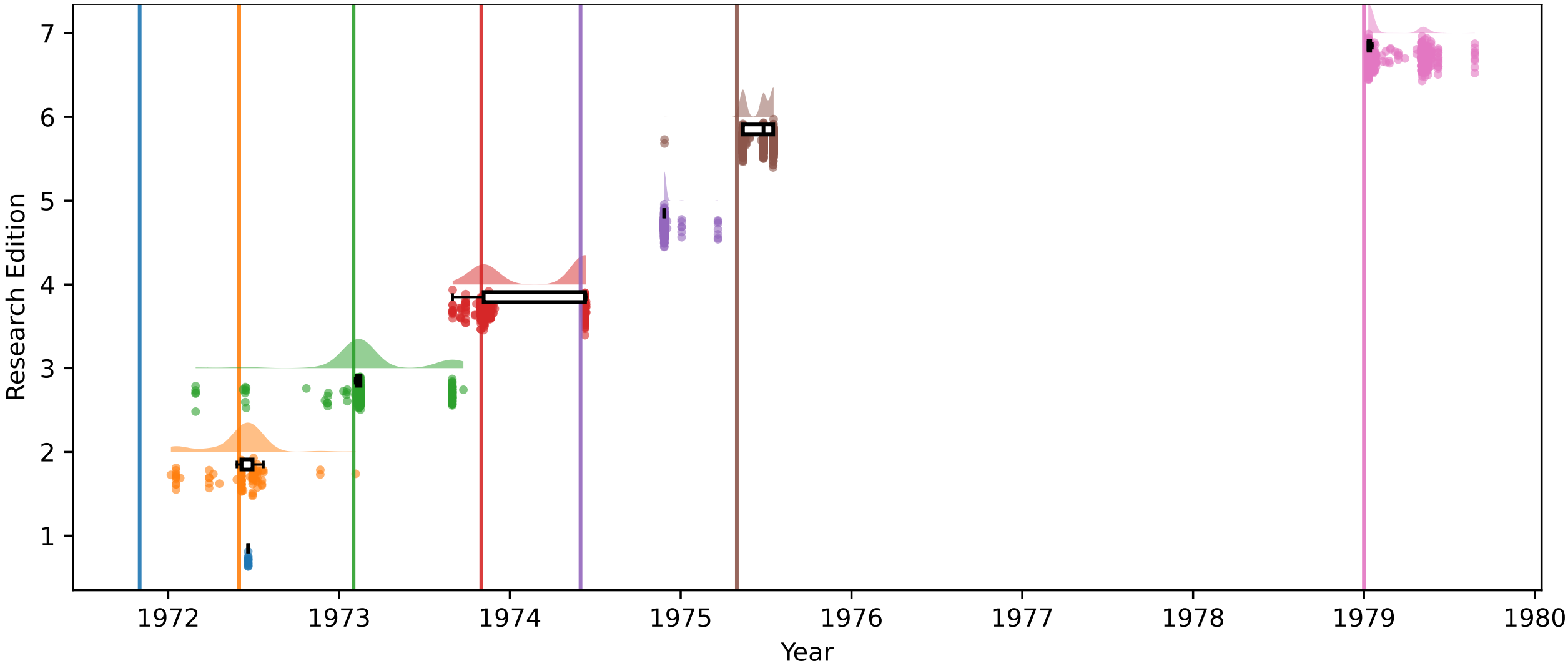
BUGS

I think writing it was just a quick entertainment for Ken. The "application" that has survived is a 1-page program that solves the Soma (or Instant Insanity) puzzle.

Dennis

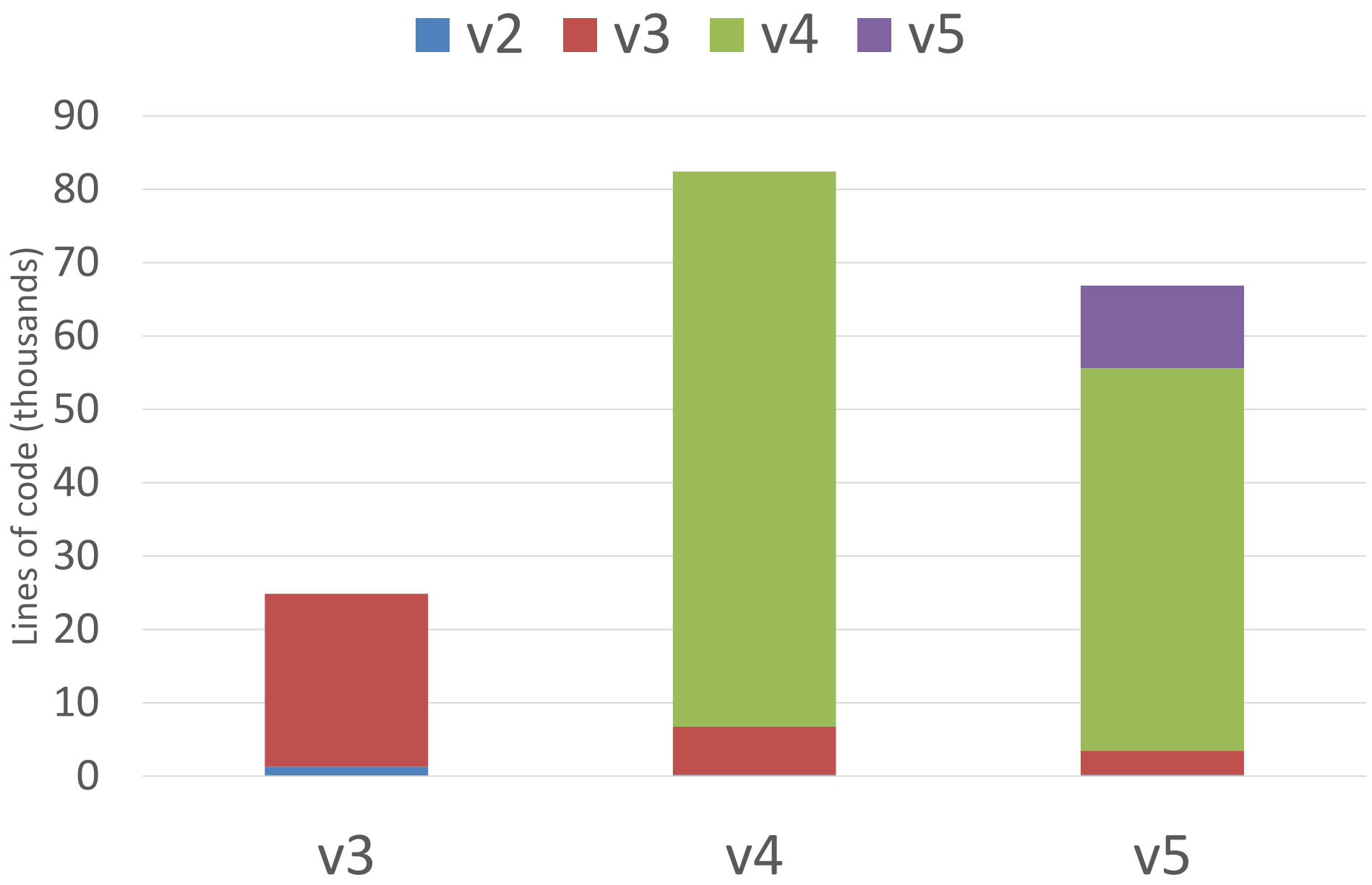
```
define  remove
  front */'1'* = ''
  back */'1'* = ''
  right */'1'* = ''
  left */'1'* = '' /(return)

define  test(s)
  s */'2'* *ft/'1'* *bk/'1'* *lt/'1'* *rt/'1'* /f(xxx)
  front ** ft /s(freturn)
  back ** bk /s(freturn)
  left ** lt /s(freturn)
  right ** rt /s(freturn)
  front = ft front
  right = rt right
  left = lt left
  back = bk back /(return)

define  build(s)
  $s *1/'1'* *2/'1'* *3/'1'* *4/'1'* *5/'1'* *6/'1'*
/f(xxx)
  $(s '1') = 1 2 3 4 5 6
  $(s '2') = 1 2 6 5 3 4
  $(s '3') = 1 2 4 3 6 5
  $(s '4') = 1 2 5 6 4 3
  $(s '5') = 3 4 1 2 6 5
  [...]
  $(s '24') = 6 5 3 4 1 2 /(return)

start a = 'wbrrrg'
  b = 'wbbrgg'
  c = 'wbgrwg'
  d = 'wgwrrb'

  build('a')
  build('b')
  build('c')
  build('d')

  a = '1'
l1  x1 = $('a' a)
  test(x1) /f(xxx)
  b = '1'
l2  x2 = $('b' b)
  test(x2) /f(t2)
  c = '1'
l3  x3 = $('c' c)
  test(x3) /f(t3)
  d = '1'
l4  x4 = $('d' d)
  test(x4) /f(t4)
    syspot = x1
    syspot = x2
    syspot = x3
    syspot = x4
    remove()
t4  d = d + '1'
    d '25' /f(l4)
    remove()
t3  c = c + '1'
    c '25' /f(l3)
    remove()
t2  b = b + '1'
    b '25' /f(l2)
    remove()
t1  a = a + '4'
    a '13' /f(l1)
end syspot = 'Done...'
```

Dating the tape



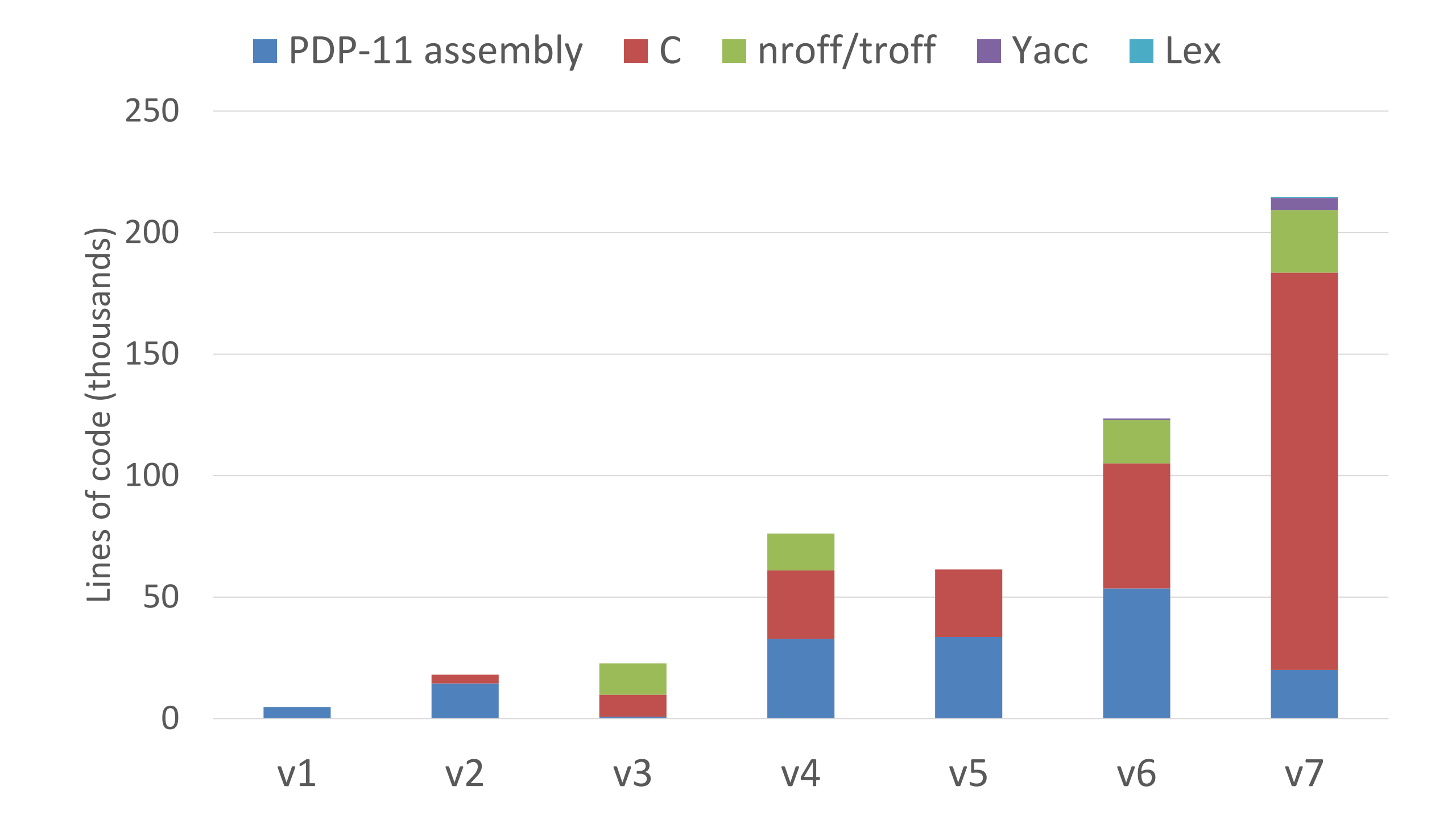
Inherited code

```
for ref in Research-V{3,4,5}-Snapshot-Development ; do
  echo $ref
  # For all the edition's files
  git ls-tree -r --name-only $ref |
    # Exclude administrative files introduced in the repo.
    grep -Ev 'README|LICENSE|\.pdf|\.ref' |
    # Run git-blame on each.
    xargs -I '{}' git blame -M -M -C -C $ref -- '{}' |
    awk '{print $1}' |
    sort |
    # Sum lines for each commit
    uniq -c |
    # Obtain lines and provenance of each commit; output totals.
    awk '{("git show " $2 "| awk '\''/Synthesized-from:/{print $2}'\''") |
       getline ver; total[ver] += $1 }
     END {for (v in total) print v, total[v]}'
done
```


v2
v3
v4
v5
Lines of code (thousands)
90
80
70
60
50
40
30
20
10
0
v3
v4
v5

Evolution of language use

```
for ref in Research-V{1,2,3,4,5,6,7}-Snapshot-Development ; do
  echo -n $ref | sed 's/Research-V/v/;s/-.*/\t/'
  # For all the edition's files
  git ls-tree -r --name-only $ref |
    # Exclude administrative files introduced in the repo.
    grep -Ev 'README|LICENSE|\.pdf|\.ref' |
    # Run git-show on each.
    while read path ; do
      echo -n "$path "
      git show $ref:$path | wc -l
    done |
    awk '$1 ~ /\./ {
      n = split($1, arr, /\./);
        suff = arr[n];
        if (suff ~ /^[0-9][sm]?$/) suff = "r";
        if (suff ~ /^[ch]$/) suff = "C";
        if (suff !~ /\//)
          lines[suff] += $2
      }
      #END {for (s in lines) if (lines[s] > 500) print s, lines[s]}
      END { OFS = "\t"; print lines["s"], lines["C"], lines["r"], lines["y"], lines["l"]; }'
done
```

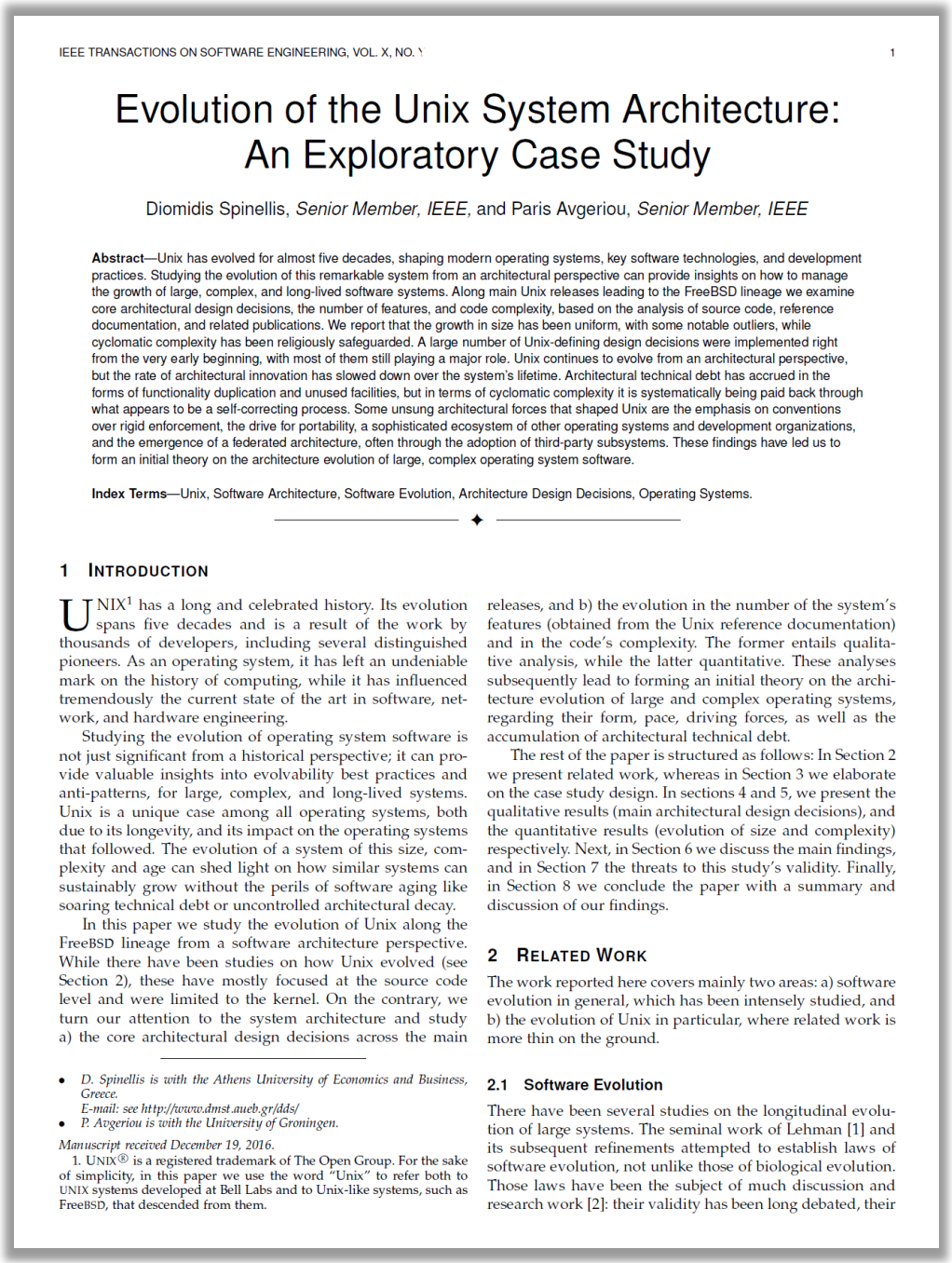

PDP-11 assembly
C
nroff/troff
Yacc
Lex
Lines of code (thousands)
250
200
150
100
50
0
v1
v2
v3
v4
v5
v6
v7

Thank you!

github.com/dspinellis/unix-history-repo/

www.spinellis.gr
@CoolSWEng
@CoolSWEng@mastodon.acm.org
dds@aueb.gr

Backup slides



Evolution of the Unix System Architecture: An Exploratory Case Study

Diomidis Spinellis, *Senior Member, IEEE*, and Paris Avgeriou, *Senior Member, IEEE*

Abstract—Unix has evolved for almost five decades, shaping modern operating systems, key software technologies, and development practices. Studying the evolution of this remarkable system from an architectural perspective can provide insights on how to manage the growth of large, complex, and long-lived software systems. Along main Unix releases leading to the FreeBSD lineage we examine core architectural design decisions, the number of features, and code complexity, based on the analysis of source code, reference documentation, and related publications. We report that the growth in size has been uniform, with some notable outliers, while cyclomatic complexity has been religiously safeguarded. A large number of Unix-defining design decisions were implemented right from the very early beginning, with most of them still playing a major role. Unix continues to evolve from an architectural perspective, but the rate of architectural innovation has slowed down over the system's lifetime. Architectural technical debt has accrued in the forms of functionality duplication and unused facilities, but in terms of cyclomatic complexity it is systematically being paid back through what appears to be a self-correcting process. Some unsung architectural forces that shaped Unix are the emphasis on conventions over rigid enforcement, the drive for portability, a sophisticated ecosystem of other operating systems and development organizations, and the emergence of a federated architecture, often through the adoption of third-party subsystems. These findings have led us to form an initial theory on the architecture evolution of large, complex operating system software.

Index Terms—Unix, Software Architecture, Software Evolution, Architecture Design Decisions, Operating Systems.

1 Introduction

UNIX[1] has a long and celebrated history. Its evolution spans five decades and is a result of the work by thousands of developers, including several distinguished pioneers. As an operating system, it has left an undeniable mark on the history of computing, while it has influenced tremendously the current state of the art in software, network, and hardware engineering.

Studying the evolution of operating system software is not just significant from a historical perspective; it can provide valuable insights into evolvability best practices and anti-patterns, for large, complex, and long-lived systems. Unix is a unique case among all operating systems, both due to its longevity, and its impact on the operating systems that followed. The evolution of a system of this size, complexity and age can shed light on how similar systems can sustainably grow without the perils of software aging like soaring technical debt or uncontrolled architectural decay.

In this paper we study the evolution of Unix along the FreeBSD lineage from a software architecture perspective. While there have been studies on how Unix evolved (see Section 2), these have mostly focused at the source code level and were limited to the kernel. On the contrary, we turn our attention to the system architecture and study a) the core architectural design decisions across the main releases, and b) the evolution in the number of the system's features (obtained from the Unix reference documentation) and in the code's complexity. The former entails qualitative analysis, while the latter quantitative. These analyses subsequently lead to forming an initial theory on the architecture evolution of large and complex operating systems, regarding their form, pace, driving forces, as well as the accumulation of architectural technical debt.

The rest of the paper is structured as follows: In Section 2 we present related work, whereas in Section 3 we elaborate on the case study design. In sections 4 and 5, we present the qualitative results (main architectural design decisions), and the quantitative results (evolution of size and complexity) respectively. Next, in Section 6 we discuss the main findings, and in Section 7 the threats to this study's validity. Finally, in Section 8 we conclude the paper with a summary and discussion of our findings.

2 Related Work

The work reported here covers mainly two areas: a) software evolution in general, which has been intensely studied, and b) the evolution of Unix in particular, where related work is more thin on the ground.

2.1 Software Evolution

There have been several studies on the longitudinal evolution of large systems. The seminal work of Lehman [1] and its subsequent refinements attempted to establish laws of software evolution, not unlike those of biological evolution. Those laws have been the subject of much discussion and research work [2]: their validity has been long debated, their

- *D. Spinellis is with the Athens University of Economics and Business, Greece.*
  E-mail: see http://www.dmst.aueb.gr/dds/
- *P. Avgeriou is with the University of Groningen.*

Manuscript received December 19, 2016.

1. UNIX® is a registered trademark of The Open Group. For the sake of simplicity, in this paper we use the word "Unix" to refer both to UNIX systems developed at Bell Labs and to Unix-like systems, such as FreeBSD, that descended from them.

Diomidis Spinellis and Paris Avgeriou. Evolution of the Unix system architecture: An exploratory case study. *IEEE Transactions on Software Engineering*, 47:1134–1163, June 2021. doi:10.1109/TSE.2019.2892149

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

CrossMark

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

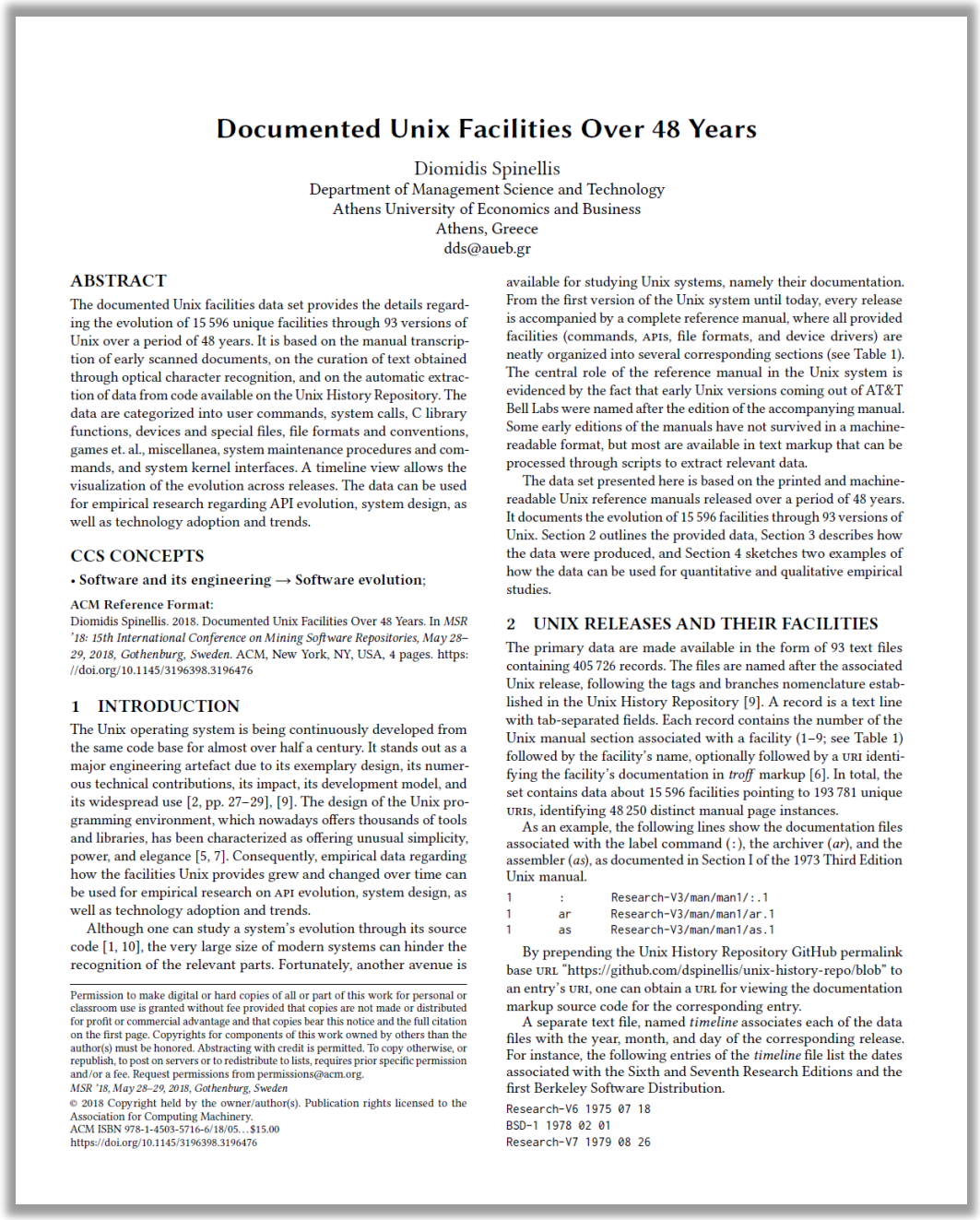
Communicated by: Romain Robbes, Martin Pinzger and Yasutaka Kamei

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

Diomidis Spinellis. A repository of Unix History and evolution. *Empirical Software Engineering*, 22(3):1372–1404, 2017. doi:10.1007/s10664-016-9445-5



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.

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
```


MSR '18, May 28–29, 2018, Gothenburg, Sweden

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

Diomidis Spinellis. Documented Unix facilities over 48 years. In *MSR '18: Proceedings of the 15th Conference on Mining Software Repositories*, pages 58–61, New York, NY, USA, May 2018. Association for Computing Machinery. doi:10.1145/3196398.3196476

Free open edX course (MOOC): Unix Tools: Data, Software and Production Engineering

---

Grow from being a Unix novice to Unix wizard status! Process big data, analyze software code, run DevOps tasks and excel in your everyday job through the amazing power of the Unix shell and command-line tools.

https://www.spinellis.gr/unix

edX

TU Delft