The Triumph of the New Jersey Style: How Architectural Elegance, Components, and Reuse Shaped the Evolution of Unix

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

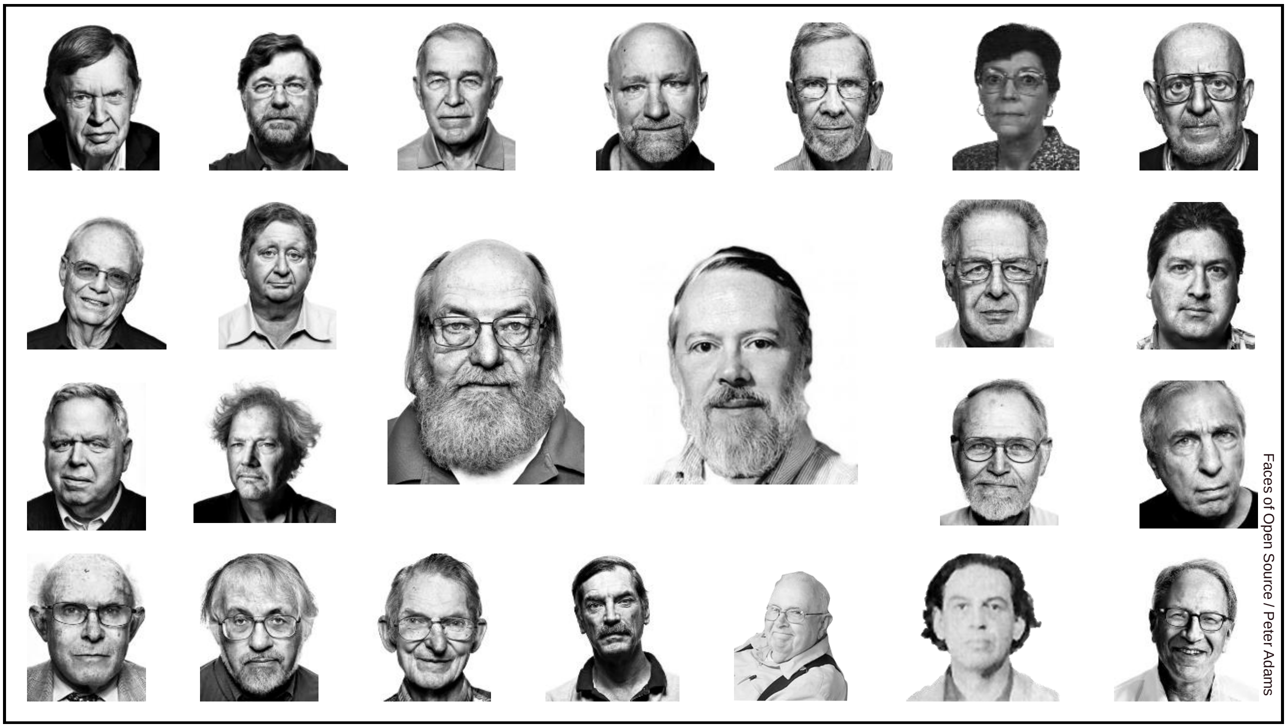
Paris Avgeriou
Institute for Mathematics and Computer Science
University of Groningen

www.spinellis.gr
@CoolSWEng

1

Unix 50th Anniversary 1969–2019

2


Faces of Open Source / Peter Adams

3


JESUS SAV
Fill up with the Holy Ghost & Fire

4

5

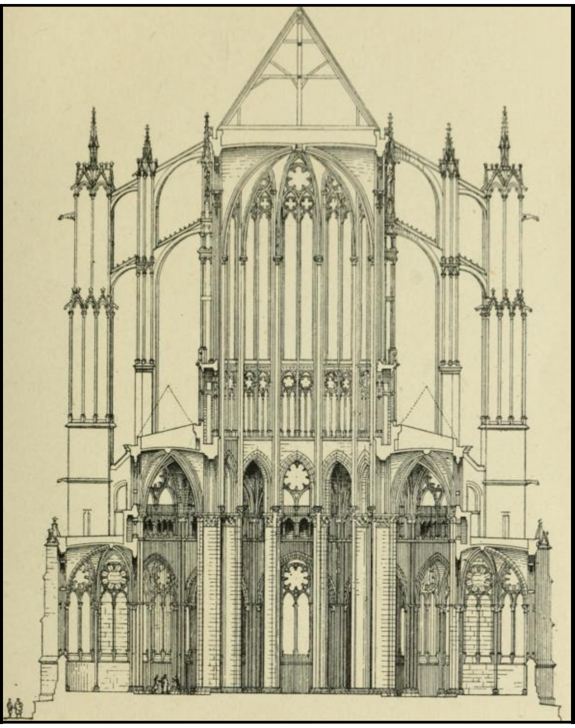
6


FORECLOSURE
You Tube
digg

7

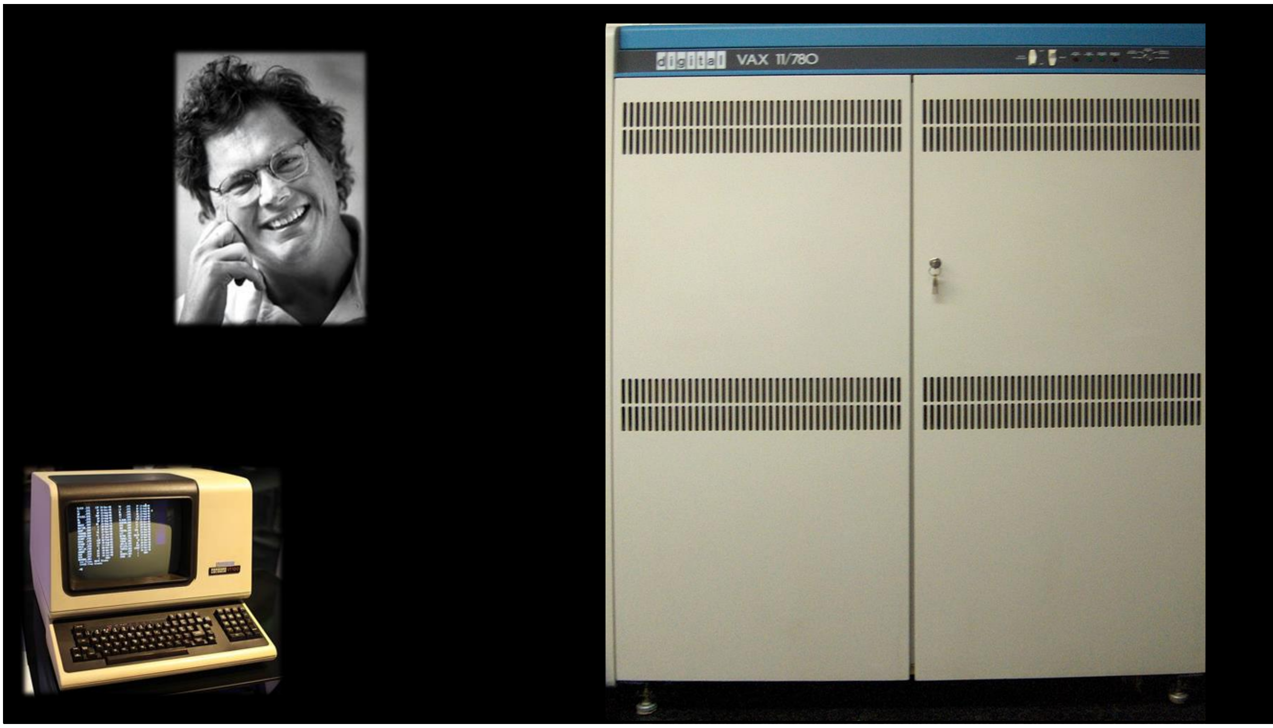

NO DEAL? NO PROBLEM!
JUST HOOT
we Voted Leave

8



9

10



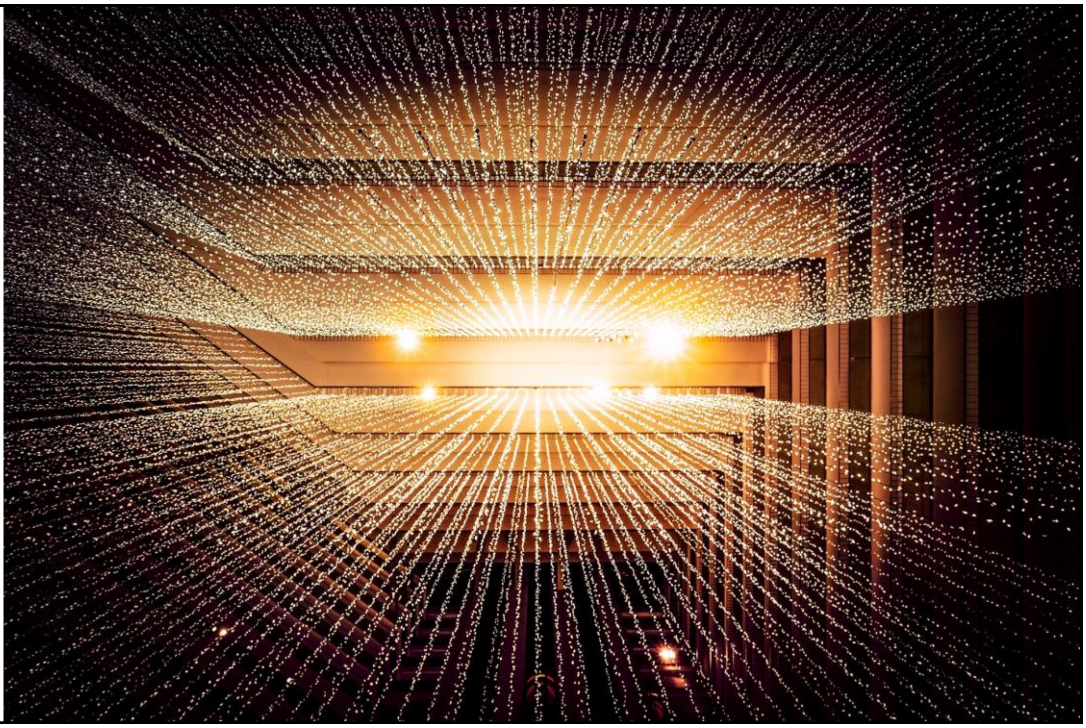
11

12

History

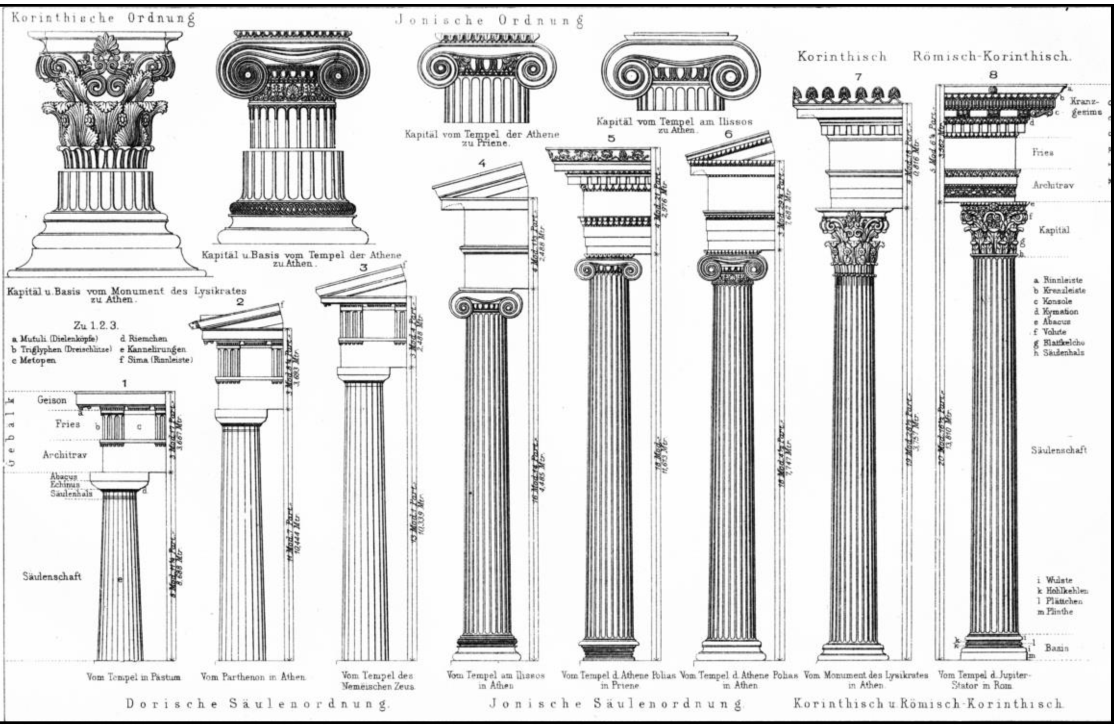
13

Data Sources



14

Architectural Design Decisions



15

Quantitative Results

16

Theory of OS Architectural Evolution

17

History

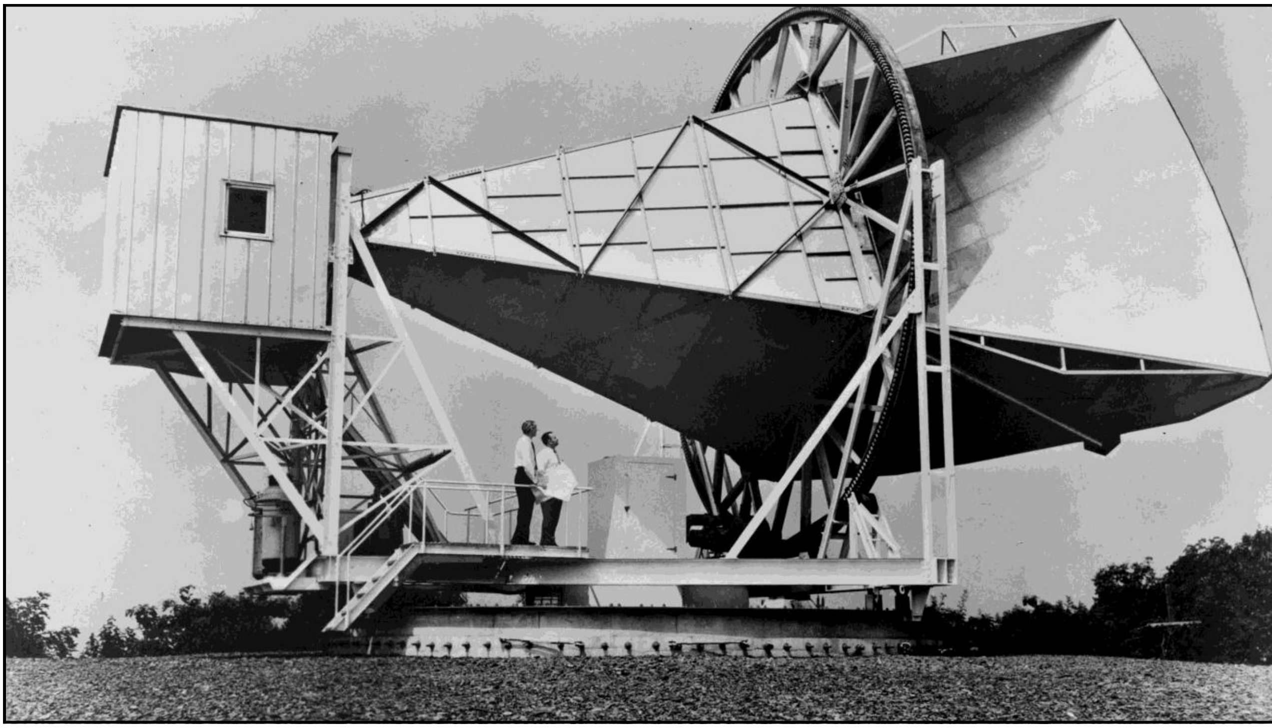
18


ALFR·
NOBEL
NAT·
MDCCC
XXXIII
OB·
MDCCC
XCVI

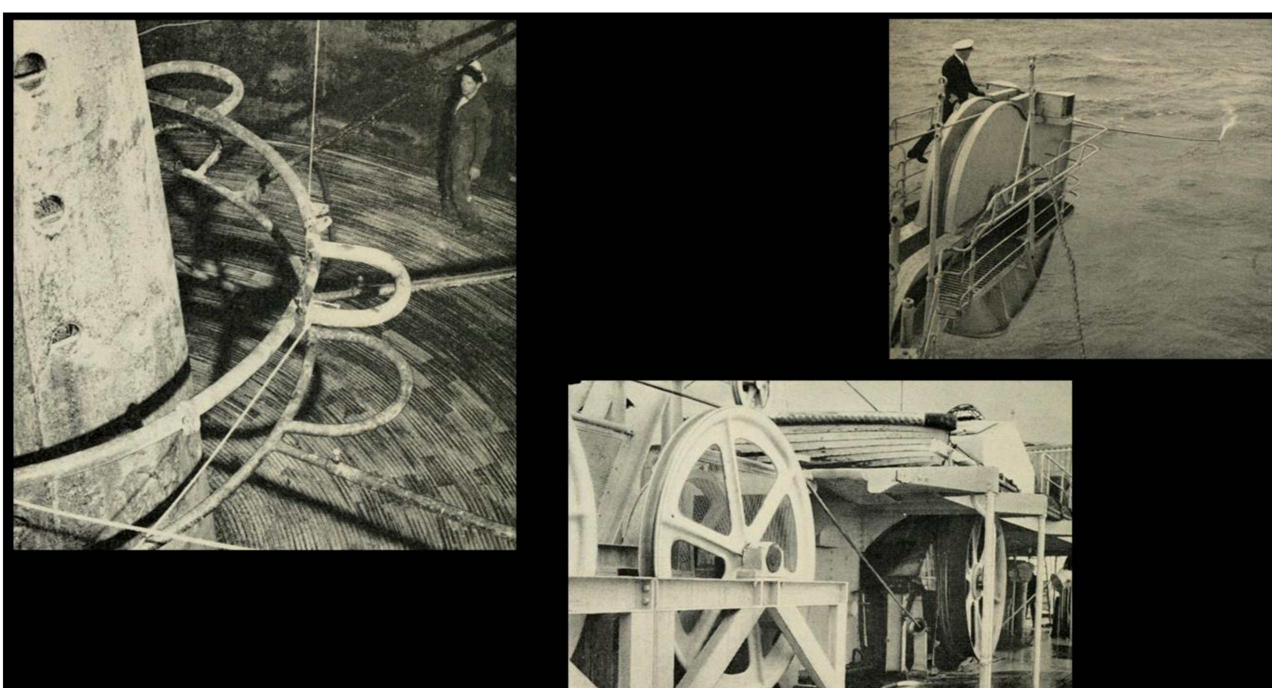
19


ACM
A.M. TURING AWARD

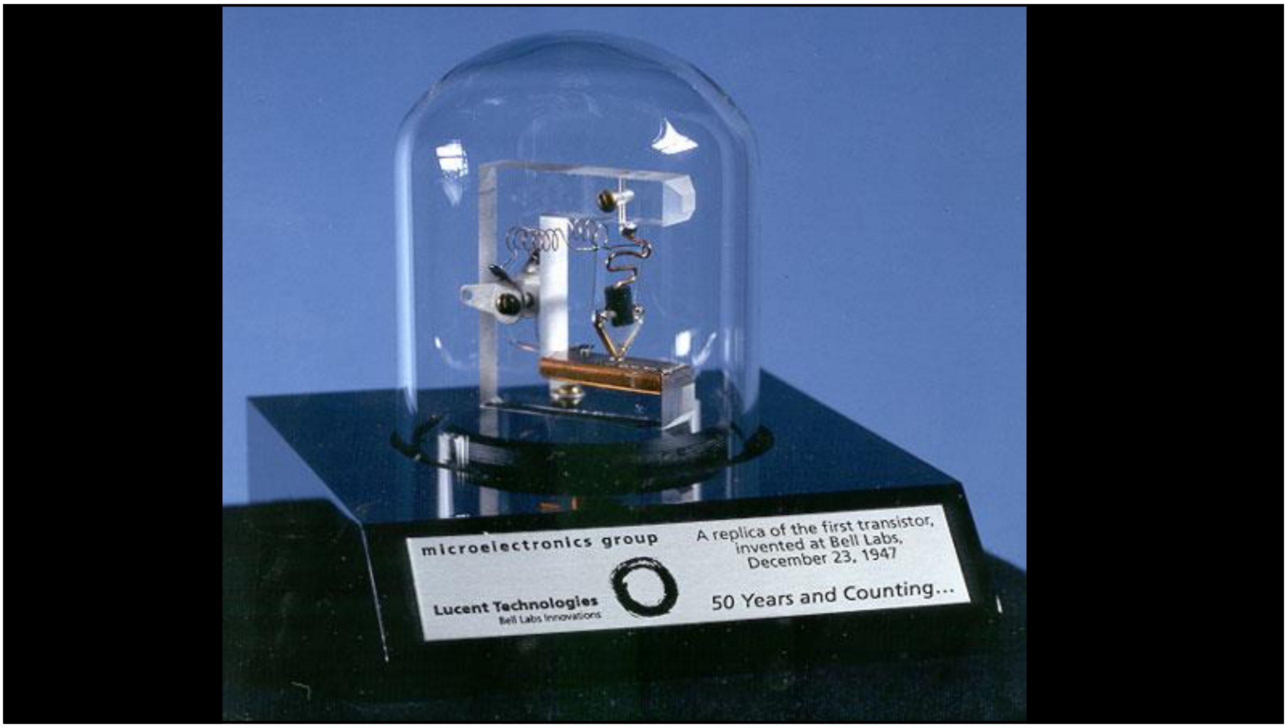
20



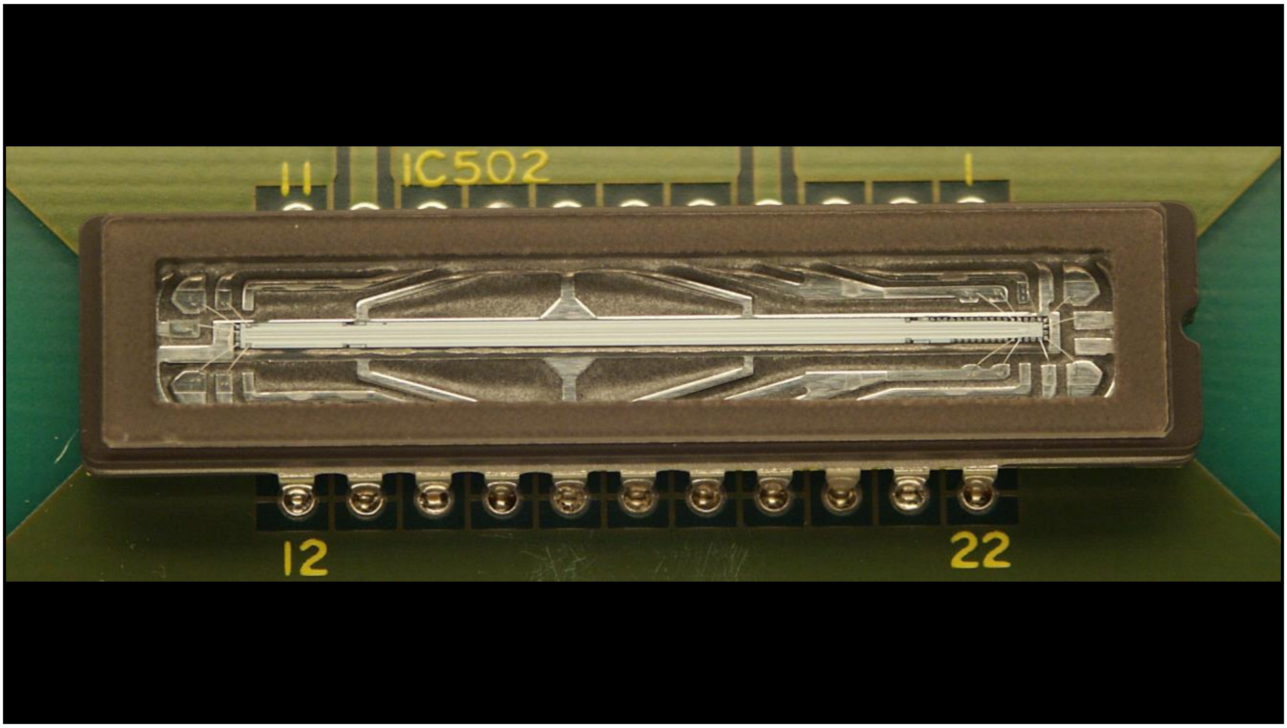
21



22


microelectronics group
Lucent Technologies
Bell Labs Innovations
A replica of the first transistor,
invented at Bell Labs,
December 23, 1947
50 Years and Counting...

23


11
IC502
1
12
22

24

Reprinted with corrections from *The Bell System Technical Journal*,
Vol. 27, pp. 379–423, 623–656, July, October, 1948.

A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist[1] and Hartley[2] on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and the savings possible due to the statistical structure of the original message and due to the nature of the final destination of the information.

The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one *selected from a set* of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design.

If the number of messages in the set is finite then this number or any monotonic function of this number can be regarded as a measure of the information produced when one message is chosen from the set, all choices being equally likely. As was pointed out by Hartley the most natural choice is the logarithmic function. Although this definition must be generalized considerably when we consider the influence of the statistics of the message and when we have a continuous range of messages, we will in all cases use an essentially logarithmic measure.

The logarithmic measure is more convenient for various reasons:

1. It is practically more useful. Parameters of engineering importance such as time, bandwidth, number of relays, etc., tend to vary linearly with the logarithm of the number of possibilities. For example, adding one relay to a group doubles the number of possible states of the relays. It adds 1 to the base 2 logarithm of this number. Doubling the time roughly squares the number of possible messages, or doubles the logarithm, etc.

2. It is nearer to our intuitive feeling as to the proper measure. This is closely related to (1) since we intuitively measures entities by linear comparison with common standards. One feels, for example, that two punched cards should have twice the capacity of one for information storage, and two identical channels twice the capacity of one for transmitting information.

3. It is mathematically more suitable. Many of the limiting operations are simple in terms of the logarithm but would require clumsy restatement in terms of the number of possibilities.

The choice of a logarithmic base corresponds to the choice of a unit for measuring information. If the base 2 is used the resulting units may be called binary digits, or more briefly *bits*, a word suggested by J. W. Tukey. A device with two stable positions, such as a relay or a flip-flop circuit, can store one bit of information. N such devices can store N bits, since the total number of possible states is 2^N and $\log_2 2^N = N$. If the base 10 is used the units may be called decimal digits. Since

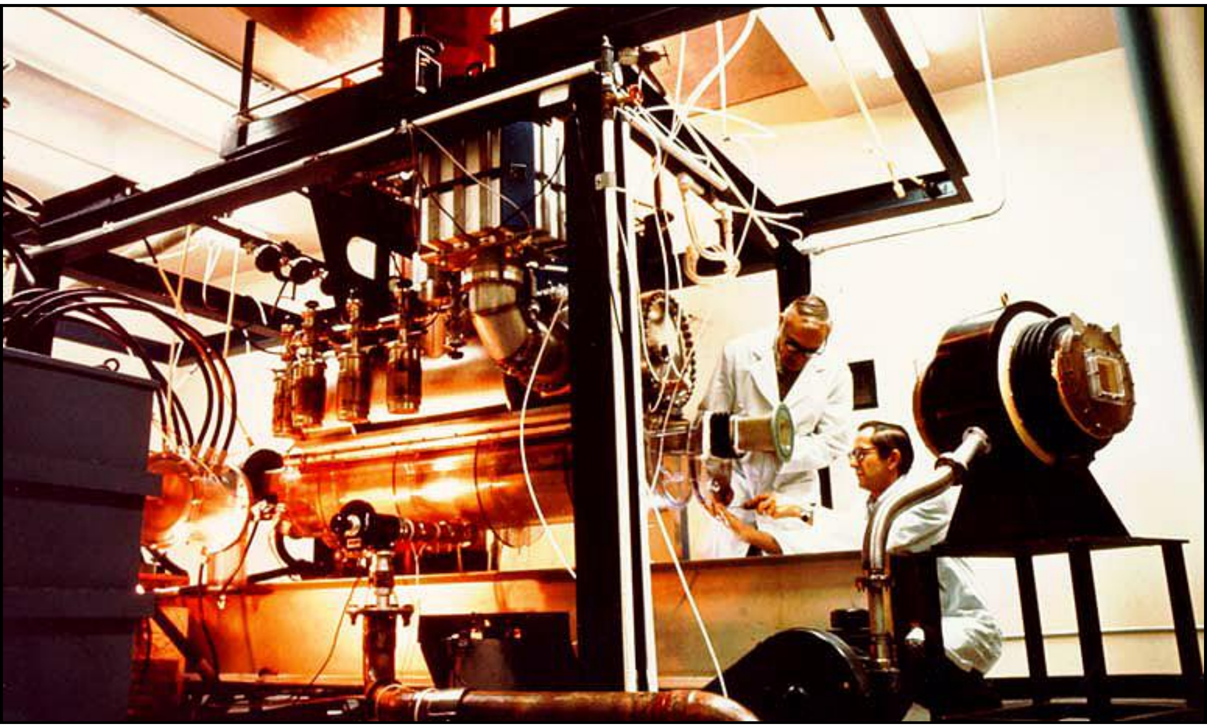
$$\begin{aligned}\log_2 M &= \log_{10} M / \log_{10} 2 \\ &= 3.32 \log_{10} M.\end{aligned}$$

[1]Nyquist, H., "Certain Factors Affecting Telegraph Speed," *Bell System Technical Journal*, April 1924, p. 324; "Certain Topics in Telegraph Transmission Theory," *A.I.E.E. Trans.*, v. 47, April 1928, p. 617.

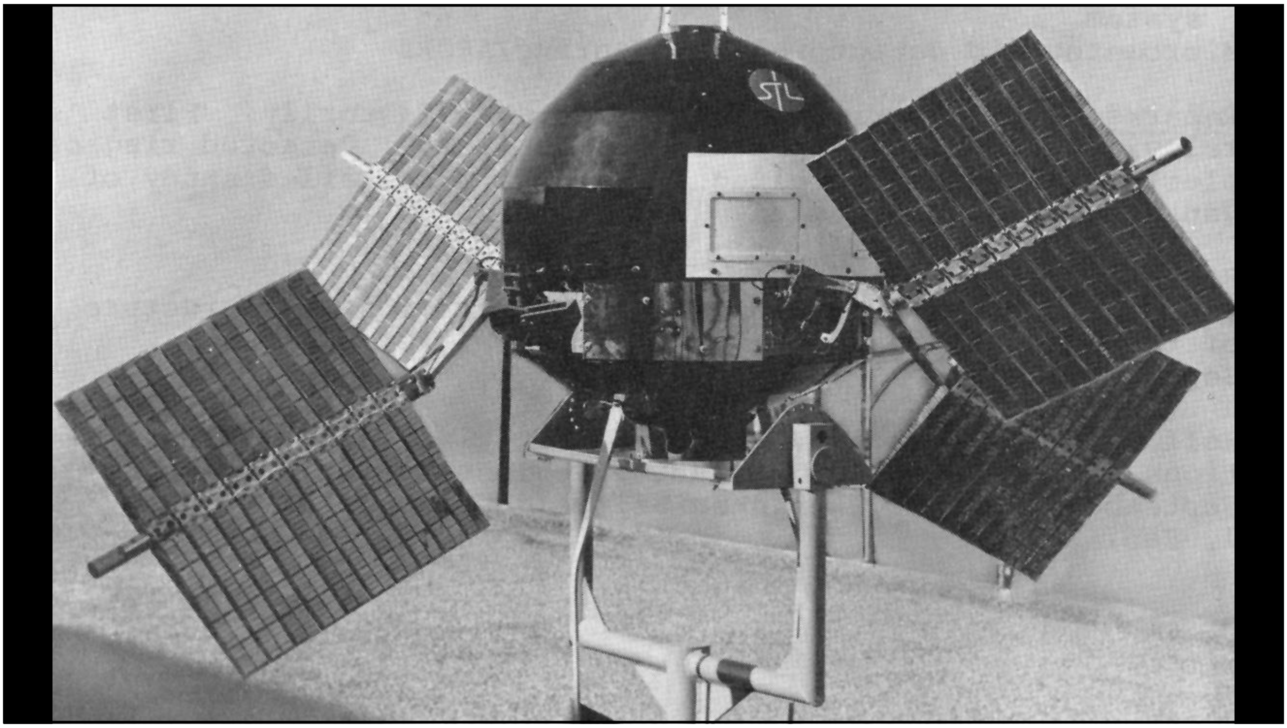
[2]Hartley, R. V. L., "Transmission of Information," *Bell System Technical Journal*, July 1928, p. 535.

1

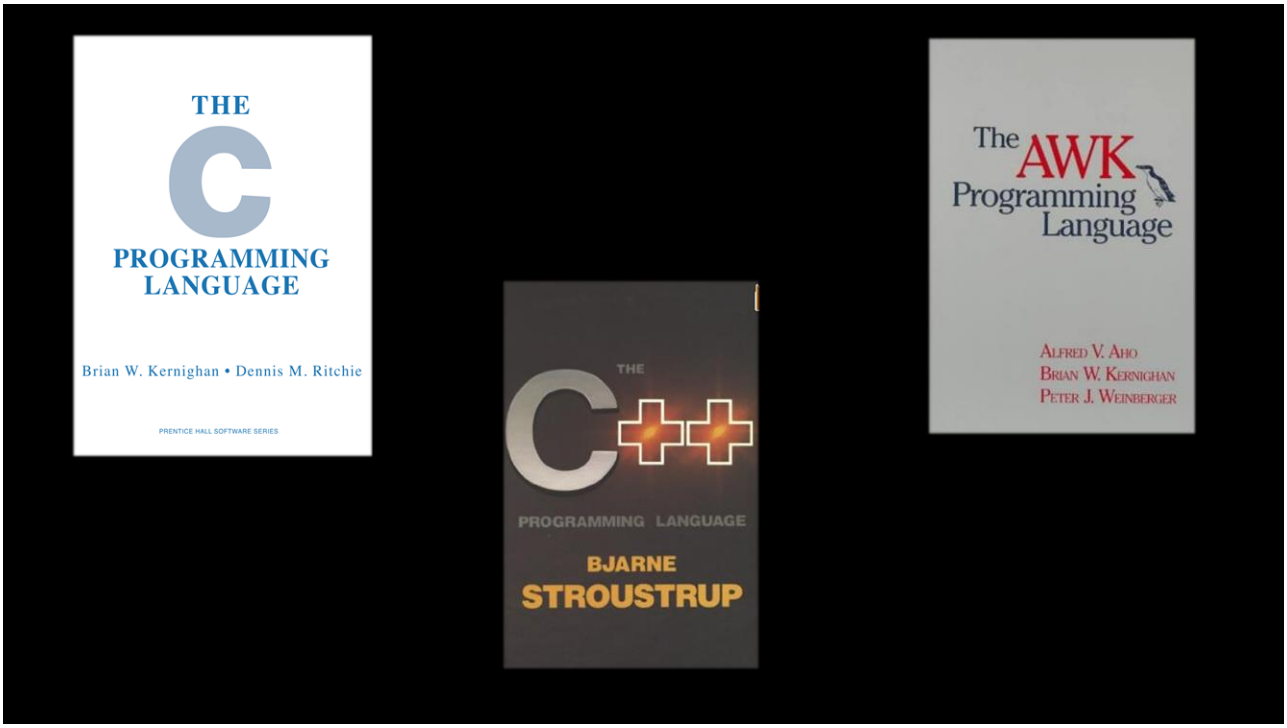
25



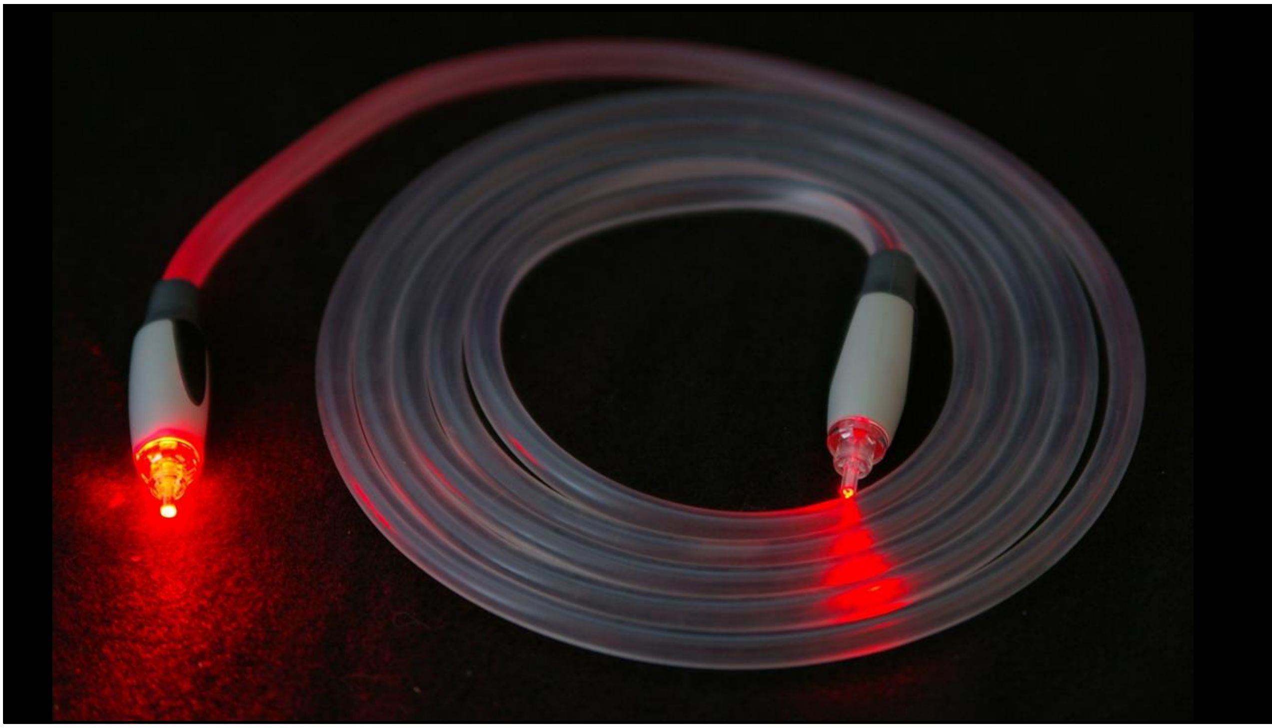
26



27


THE
C
PROGRAMMING
LANGUAGE
Brian W. Kernighan • Dennis M. Ritchie
PRENTICE HALL SOFTWARE SERIES
THE
C++
PROGRAMMING LANGUAGE
BJARNE
STROUSTRUP
The AWK
Programming
Language
Alfred V. Aho
Brian W. Kernighan
Peter J. Weinberger

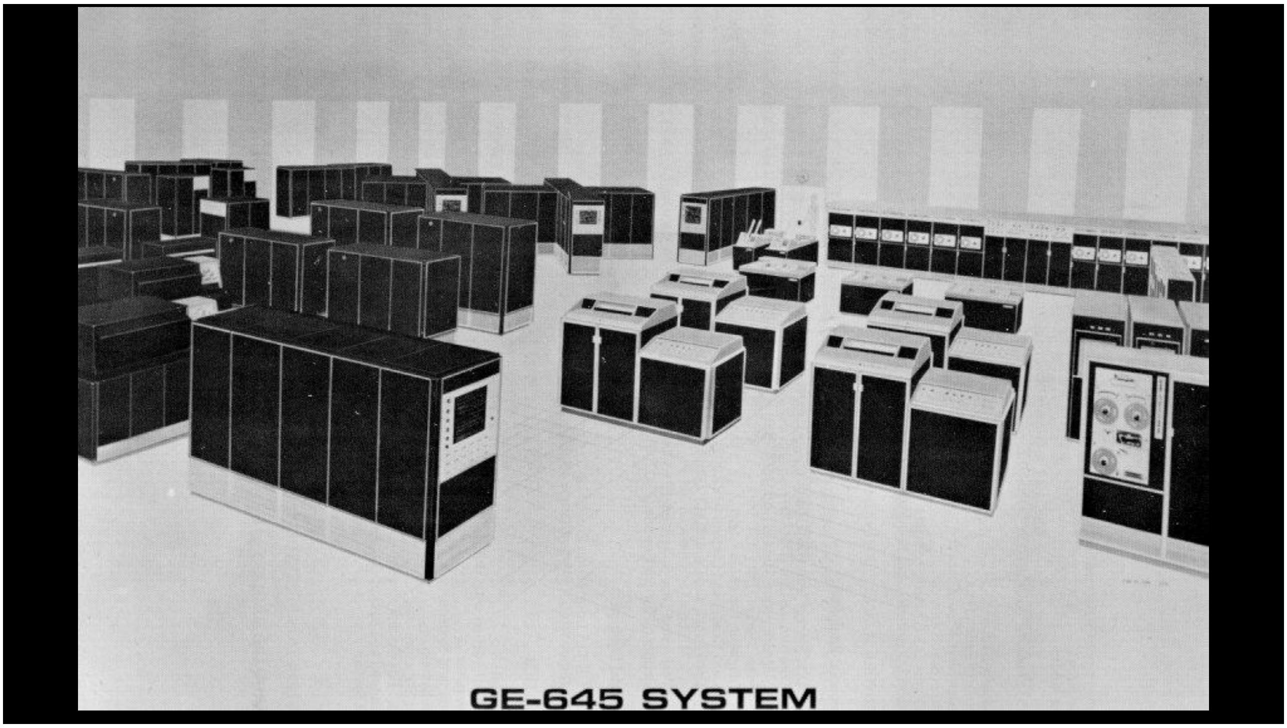
28



29

30

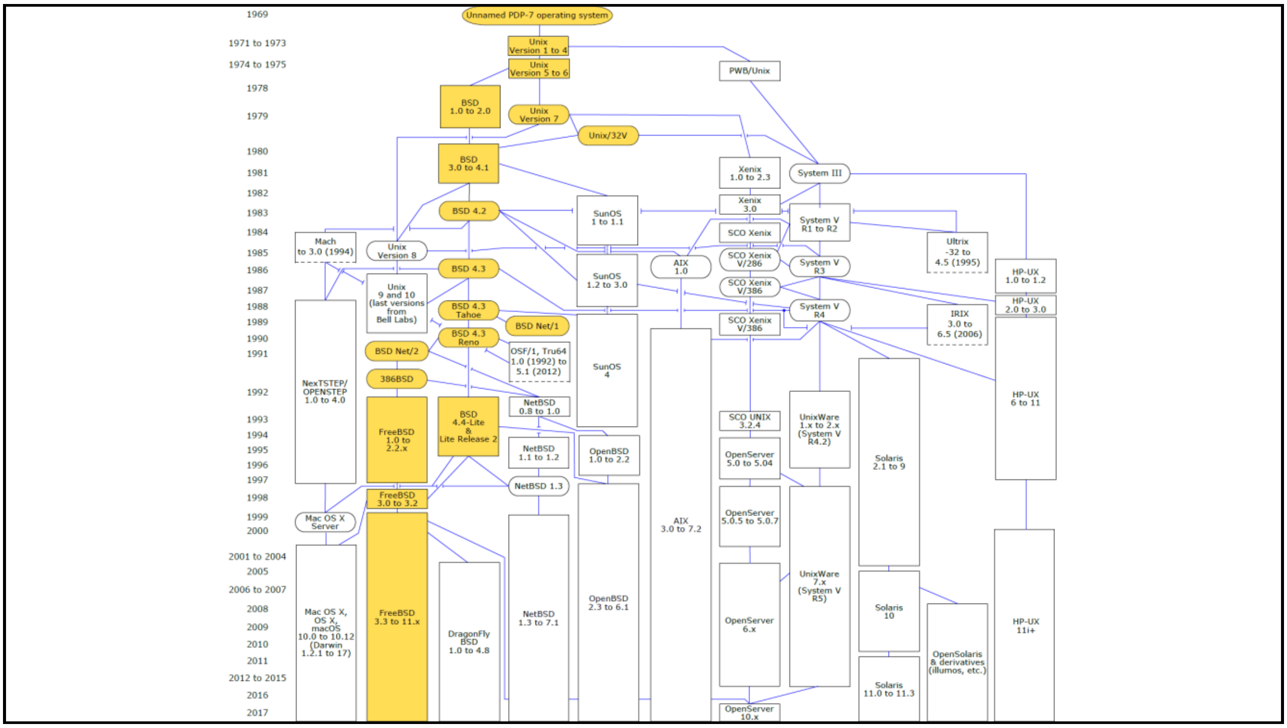
31



32

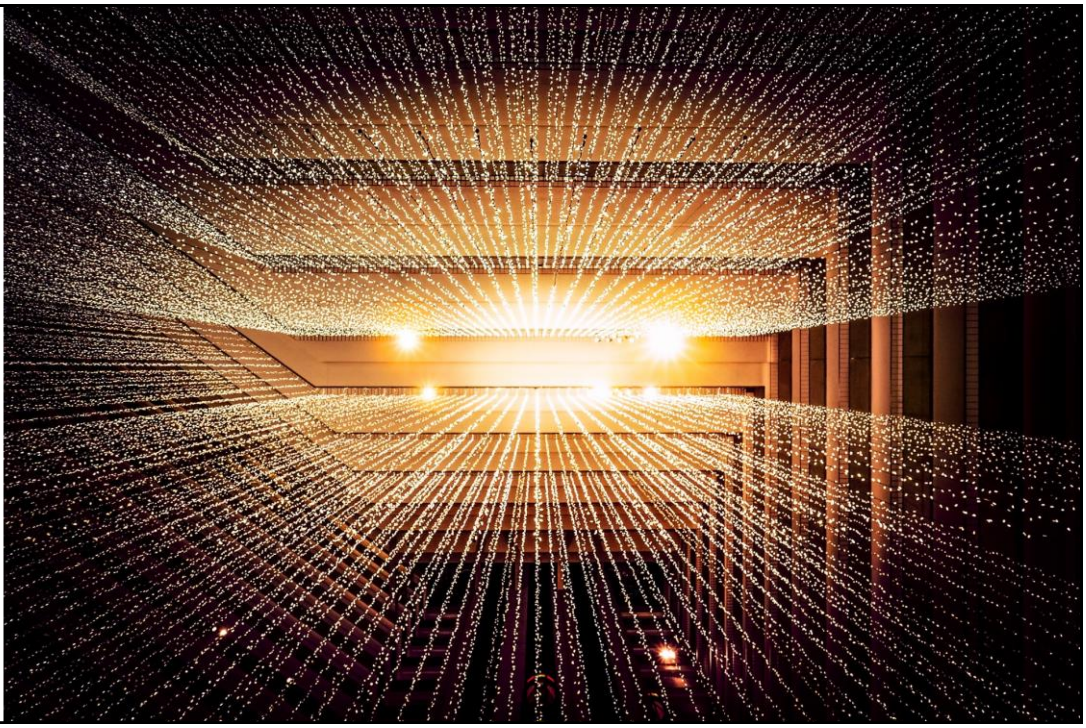
33

34

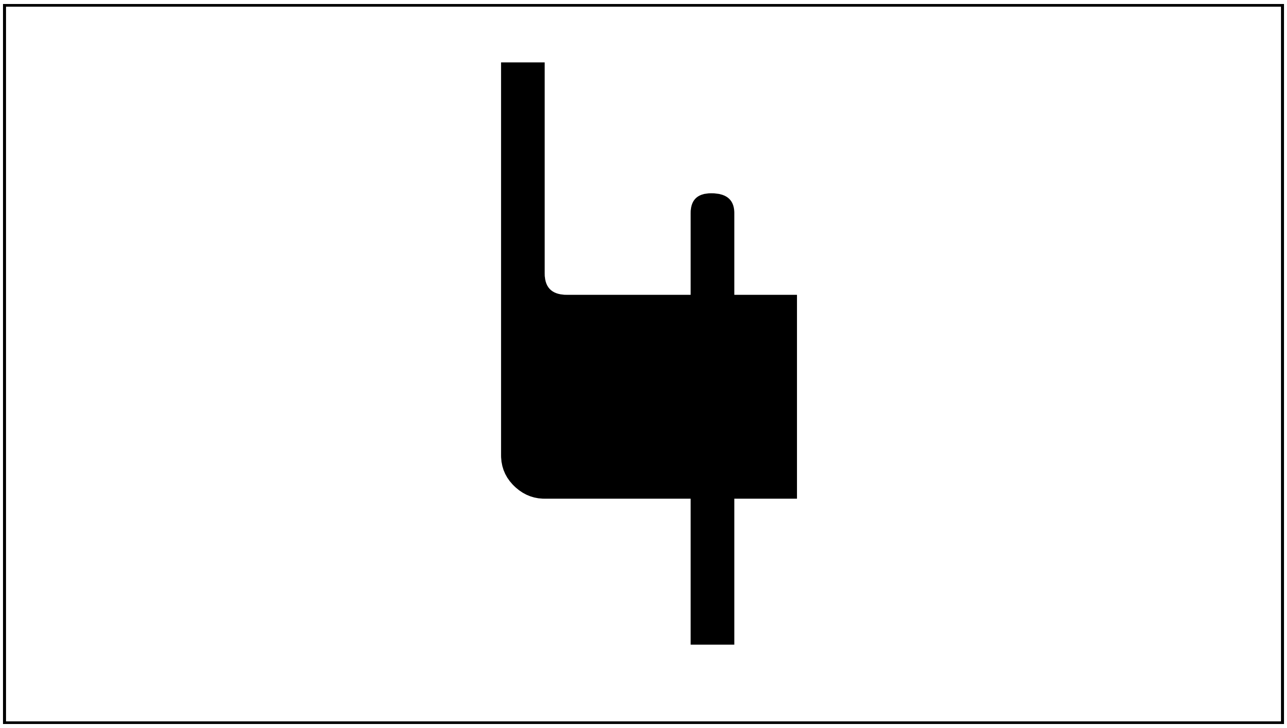


35

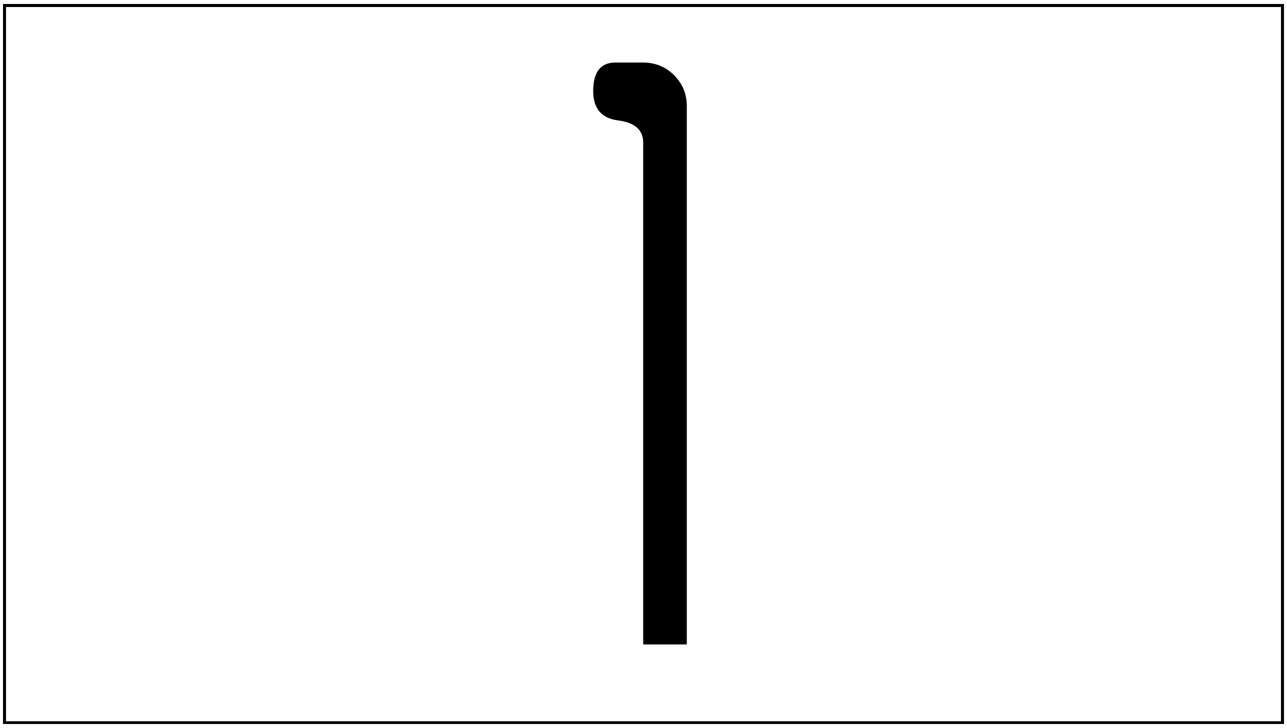
Data Sources



36



37



38

CALDERA

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.

Copyright(C) Caldera International Inc. 2001-2002. All rights reserved.

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

* UNIX is a registered trademark of The Open Group in the US and other countries.

39

Bell Laboratories

subject: Study of UNIX

date: September 14, 1972

from: T. R. Bashkow

Messrs. W. S. Bartlett
D. P. Clayton
D. H. Copp
Mmes. G. J. Hansen
J. Hintz
Mr. L. J. Kelly
Miss R. L. Klein

Messrs. J. J. Ludwig
J. F. Maranzano
Mrs. G. Pettit
Messrs. J. E. Ritacco
B. A. Tague
D. W. Vogel
Mrs. L. S. Wright

On Tuesday, September 19, at 9:30 a.m. in Room 2A-418 at Murray Hill, I will give a talk on my study of the UNIX operating system. The emphasis will be on the structure, functional components, and internal operation of the system.

MH-8234-TRB-mbh

T. R. Bashkow
T R Bashkow

Copy to
Mr. G. L. Baldwin

40

```
/ initialize inodes for special files (inodes 1 to 40.)

        mov     $40.,r1 / set r1=i-node-number 40.
1:
        jsr     r0,iget / read i-node 'r1' from disk into inode area of
                        / core and write modified inode out (if any)
        mov     $100017,i.flgs / set flags in core image of inode to indi-
                               / cate allocated, read (owner, non-owner),
                               / write (owner, non-owner)
        movb    $1,i.nlks / set no. of links = 1
        movb    $1,i.uid / set user id of owner = 1
        jsr     r0,setimod / set imod=1 to indicate i-node modified, also
                           / stuff time of modification into i-node
        dec     r1 / next i-node no. = present i-node no.-1
        bgt     1b / has i-node 1 been initialized; no, branch

/ initialize i-nodes r1.,...,47. and write the root device, binary, etc.,
/ directories onto fixed head disk.  user temporary, initialization prog.
```

Issue D   Date   3/17/72   ID IMO.1-1   Section E.0   Page 4

41

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 wo

/  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
```

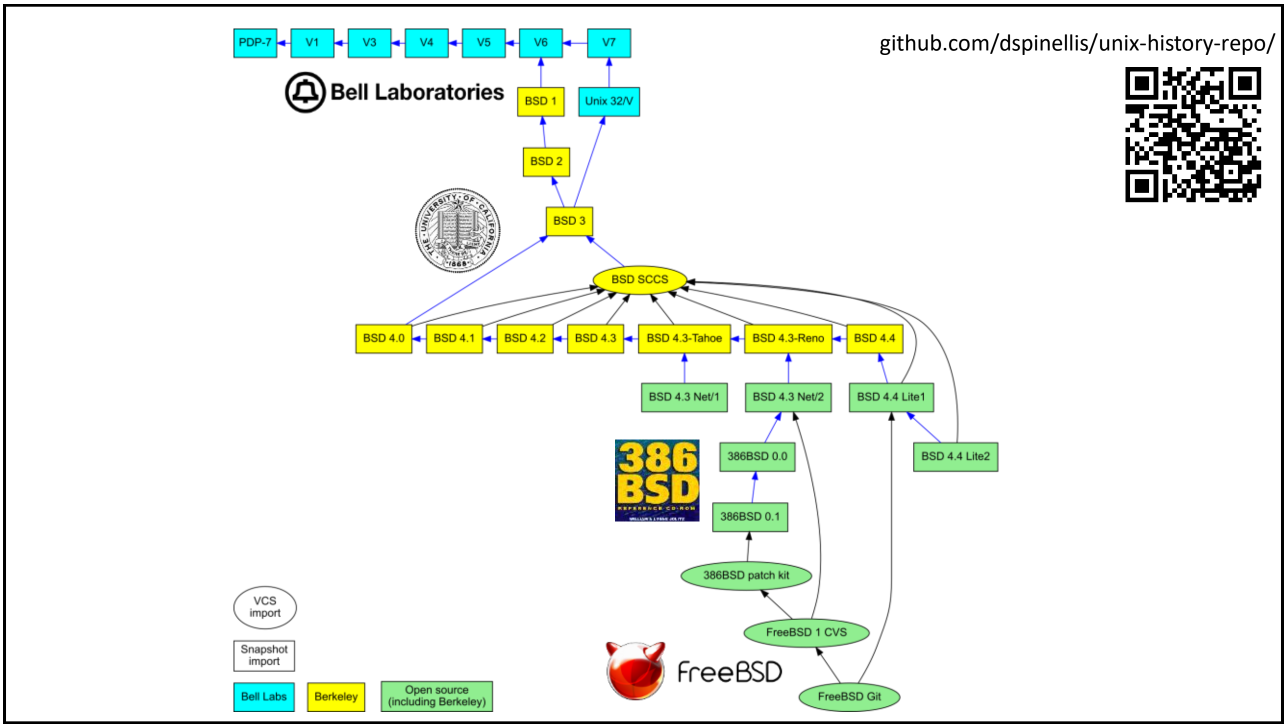
Issue D Date 3/17/72   ID IMO.1-1   Section E.0

UNIX IMPLEMENTATION

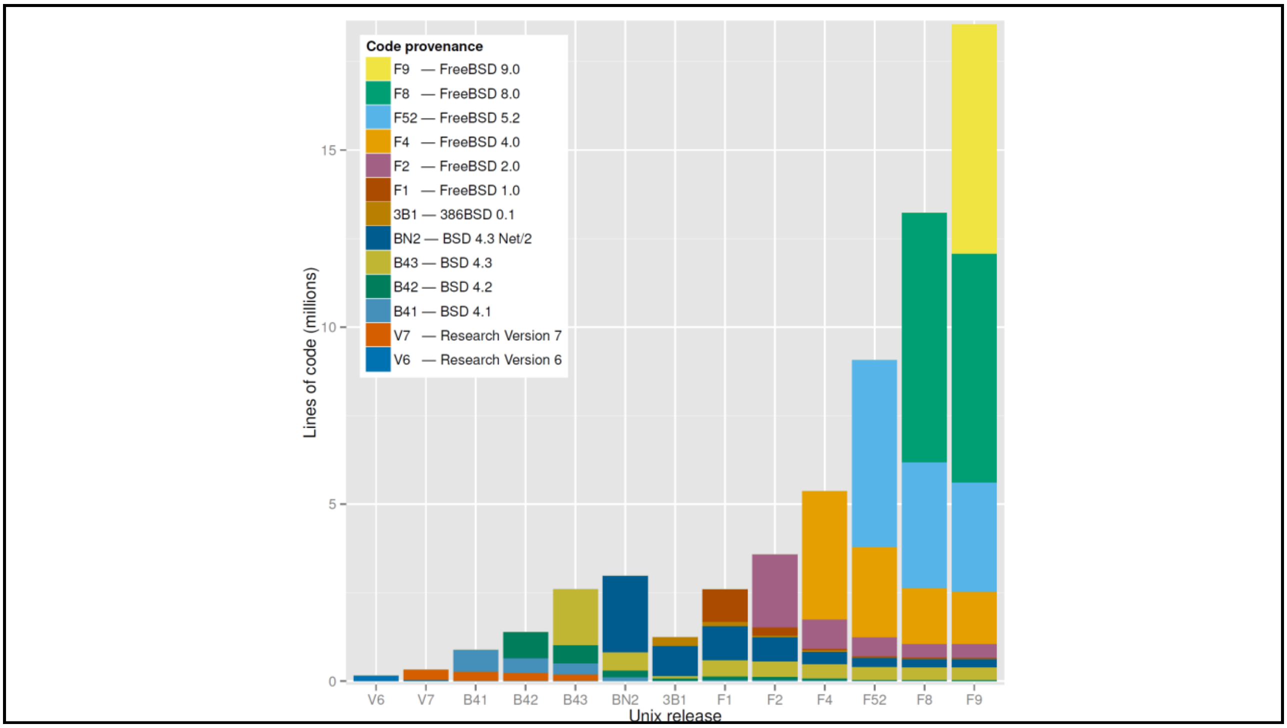
```
        mov     $idata,r0 / r0=base addr. of assembled directories.
        mov     $u.off,u.fofp / pointer to u.off in u.fofp (holds file
                              / offset)
1:
        mov     (r0)+,r1/r1=41.,...,47; "0" in the assembled directory
                        / header signals last
        beq     1f      / assembled directory has been written onto drum
        jsr     r0,imap / locate the inode map bit for i-node 'r1'
        bisb    mq,(r2) / set the bit to indicate the i-node is not
                        / available
        jsr     r0,iget / read inode 'r1' from disk into inode area of
                        / core and write modified i-node on drum (if any)
        mov     (r0)+,i.flgs / set flags in core image of inode from
                             / assembled directories header
        movb    (r0)+.i.nlks / set no. of links from header
        movb    (r0)+,i.uid / set user id of owner from header
        jsr     r0,setimod / set imod=1 to indicate inode modified: also,
                           / stuff time of modification into i-node
        mov     (r0)+,u.count / set byte count for write call equal to
                              / size of directory
        mov     r0,u.base / set buffer address for write to top of directory
        clr     u.off / clear file offset used in 'seek' and 'tell'
        add     u.count,r0 / r0 points to the header of the next directory
        jsr     r0,writei / write the directory and i-node onto drum
        br      1b / do next directory
        .endif

/ next 2 instructions not executed during cold boot.
        bis     $2000,sb0 / sb0 I/O queue entry for superblock on drum;
                          / set bit 10 to 1
        jsr     r0,ppoke / read drum superblock
1:
        tstb    sb0+1 / has I/O request been honored (for drum)?
        bne     1b / no, continue to idle.

1:
        decb    sysflg / mormally sysflag=0, indicates executing in system
        sys     exec; 2f; 1f / generates trap interrupt; trap vector =
                             / sysent; 0
        br      panic / execute file/etc/init

1:
        2f;0            this is file #47 listed on E0,9 (on cold boot) see E0,10
2:
        </etc/init\0> / UNIX looks for strings term, noted by nul\0

panic:
        clr     ps
1:
        dec     $0
        bne     1b
        dec     $5
        bne     1b
        jmp     *$173700 / rom loader address
```

Issue D Date 3/17/72   ID IMO.1-1   Section E.0 Page 5

42


PDP-7
V1
V3
V4
V5
V6
V7
Bell Laboratories
BSD 1
Unix 32/V
BSD 2
BSD 3
BSD SCCS
BSD 4.0
BSD 4.1
BSD 4.2
BSD 4.3
BSD 4.3-Tahoe
BSD 4.3-Reno
BSD 4.4
BSD 4.3 Net/1
BSD 4.3 Net/2
BSD 4.4 Lite1
BSD 4.4 Lite2
386BSD 0.0
386BSD 0.1
386BSD patch kit
FreeBSD 1 CVS
FreeBSD Git
FreeBSD
VCS import
Snapshot import
Bell Labs
Berkeley
Open source (including Berkeley)
github.com/dspinellis/unix-history-repo/

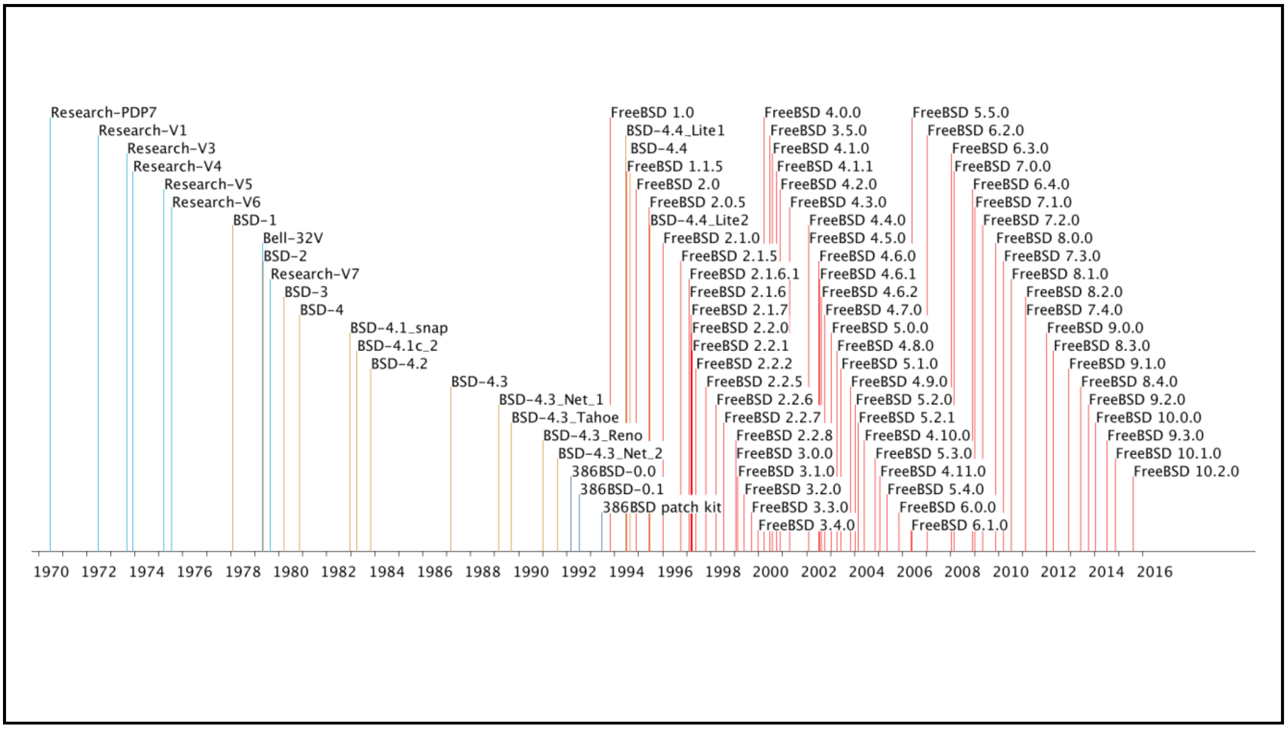
43


Code provenance
F9 — FreeBSD 9.0
F8 — FreeBSD 8.0
F52 — FreeBSD 5.2
F4 — FreeBSD 4.0
F2 — FreeBSD 2.0
F1 — FreeBSD 1.0
3B1 — 386BSD 0.1
BN2 — BSD 4.3 Net/2
B43 — BSD 4.3
B42 — BSD 4.2
B41 — BSD 4.1
V7 — Research Version 7
V6 — Research Version 6
Lines of code (millions)
0
5
10
15
V6 V7 B41 B42 B43 BN2 3B1 F1 F2 F4 F52 F8 F9
Unix release

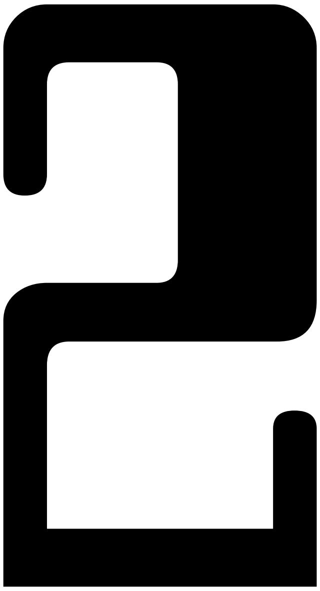
44

```
$ 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) }
```

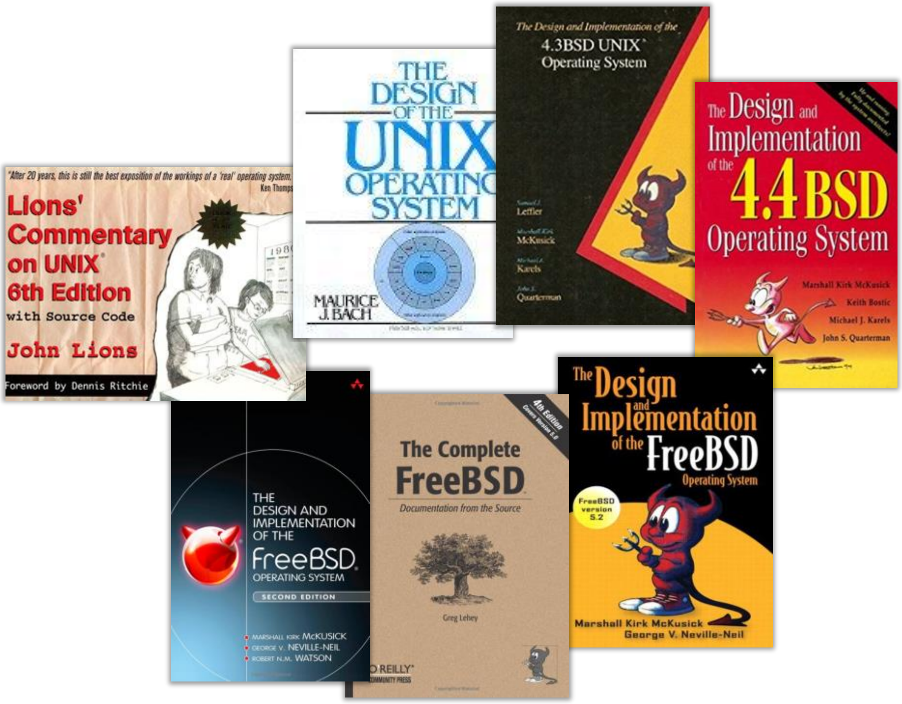
45



46


2

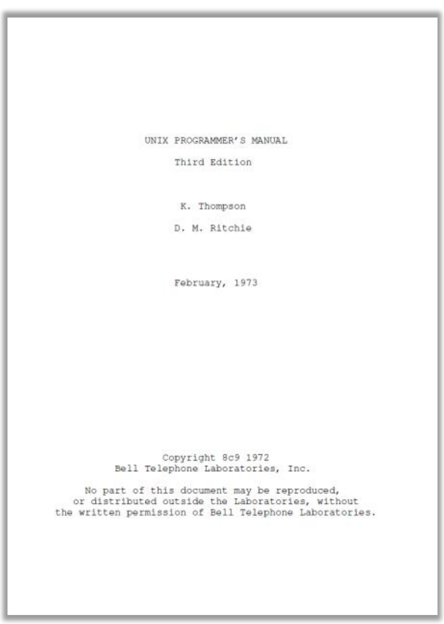
47


"After 20 years, this is still the best exposition of the workings of a 'real' operating system.
Lions' Commentary on UNIX 6th Edition
with Source Code
John Lions
Foreword by Dennis Ritchie
THE DESIGN OF THE UNIX OPERATING SYSTEM
MAURICE J. BACH
The Design and Implementation of the 4.3BSD UNIX Operating System
Leffler
McKusick
Karels
Quarterman
The Design and Implementation of the 4.4BSD Operating System
Marshall Kirk McKusick
Keith Bostic
Michael J. Karels
John S. Quarterman
THE DESIGN AND IMPLEMENTATION OF THE FreeBSD OPERATING SYSTEM
SECOND EDITION
MARSHALL KIRK McKUSICK
GEORGE V. NEVILLE-NEIL
ROBERT N.M. WATSON
The Complete FreeBSD
Documentation from the Source
Greg Lehey
4th Edition
The Design and Implementation of the FreeBSD Operating System
FreeBSD version 5.2
Marshall Kirk McKusick
George V. Neville-Neil

48

3

49


UNIX PROGRAMMER'S MANUAL

Third Edition

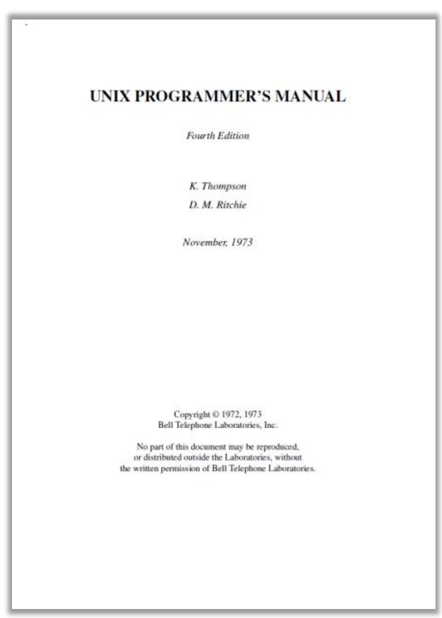
K. Thompson

D. M. Ritchie

February, 1973

Copyright 8c9 1972
Bell Telephone Laboratories, Inc.

No part of this document may be reproduced,
or distributed outside the Laboratories, without
the written permission of Bell Telephone Laboratories.


UNIX PROGRAMMER'S MANUAL

Fourth Edition

K. Thompson

D. M. Ritchie

November, 1973

Copyright © 1972, 1973
Bell Telephone Laboratories, Inc.

No part of this document may be reproduced,
or distributed outside the Laboratories, without
the written permission of Bell Telephone Laboratories.

50

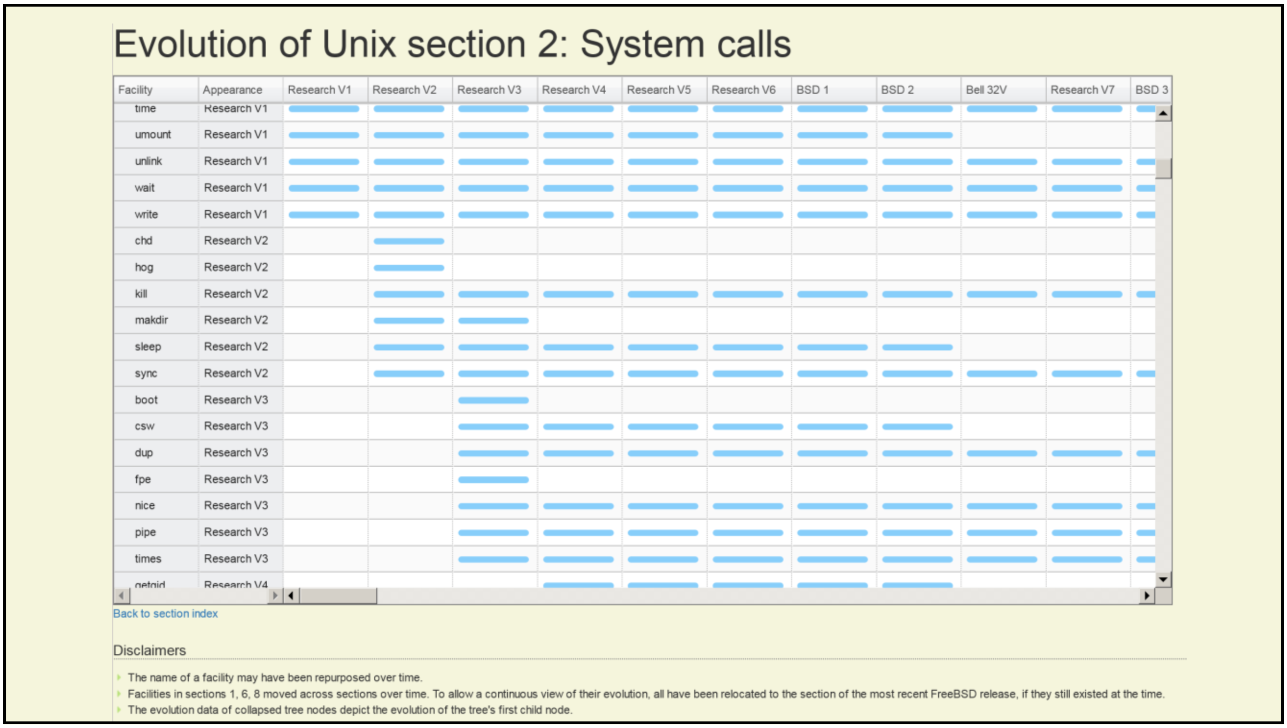
51

dspinellis.github.io/unix-history-man

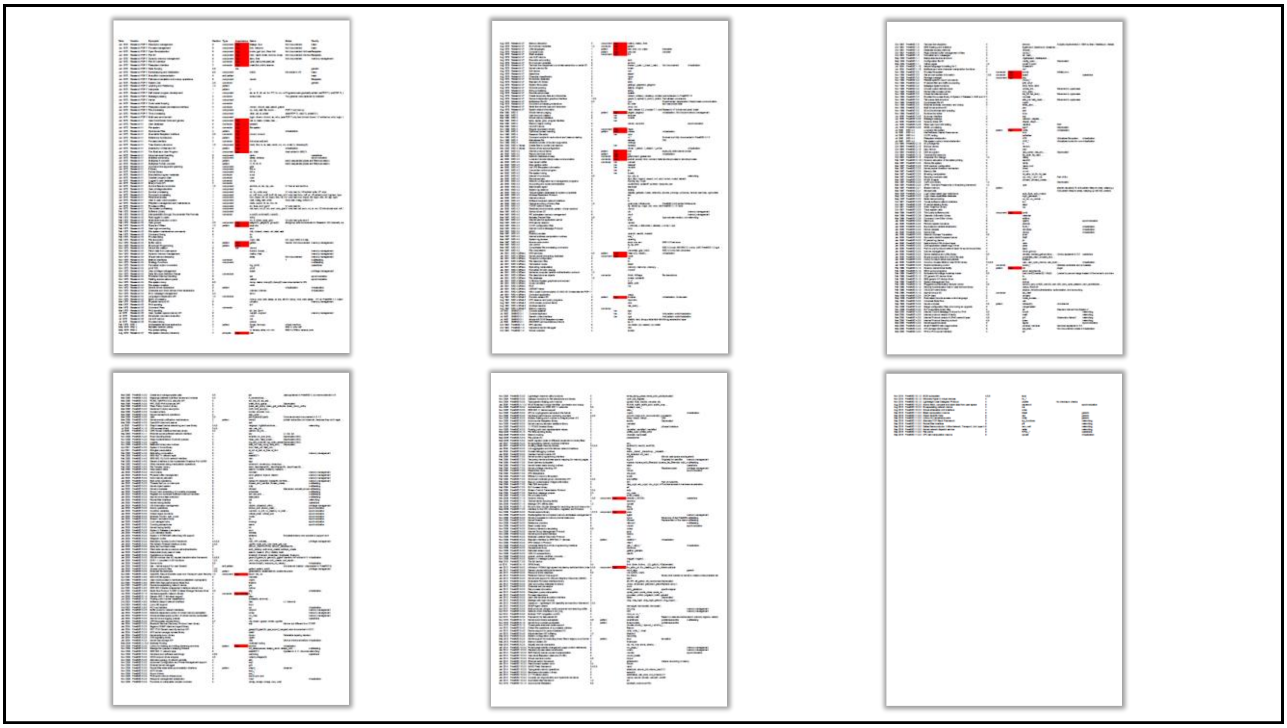
Evolution of Unix Facilities

1. User commands
2. System calls
3. C library functions
4. Devices and special files
5. File formats and conventions
6. Games et. al.
7. Miscellanea
8. System maintenance procedures and commands
9. System kernel interfaces

52

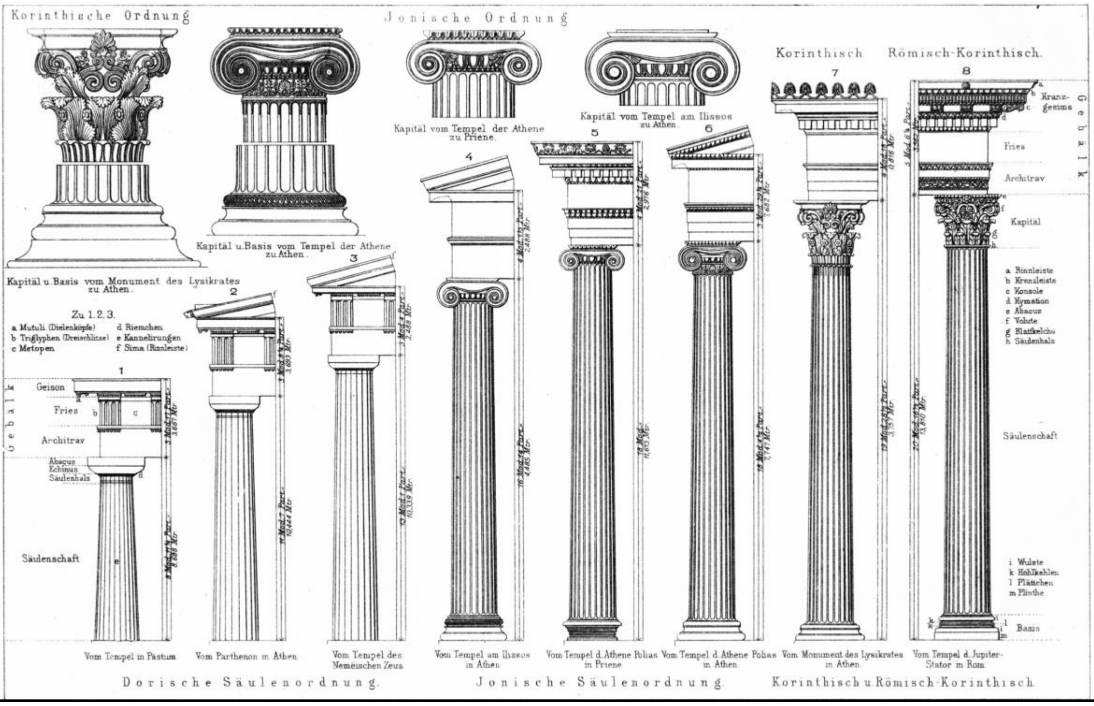

Evolution of Unix section 2: System calls
Facility
Appearance
Research V1
Research V2
Research V3
Research V4
Research V5
Research V6
BSD 1
BSD 2
Bell 32V
Research V7
BSD 3
time Research V1
umount Research V1
unlink Research V1
wait Research V1
write Research V1
chd Research V2
hog Research V2
kill Research V2
makdir Research V2
sleep Research V2
sync Research V2
boot Research V3
csw Research V3
dup Research V3
fpe Research V3
nice Research V3
pipe Research V3
times Research V3
Back to section index
Disclaimers
The name of a facility may have been repurposed over time.
Facilities in sections 1, 6, 8 moved across sections over time. To allow a continuous view of their evolution, all have been relocated to the section of the most recent FreeBSD release, if they still existed at the time.
The evolution data of collapsed tree nodes depict the evolution of the tree's first child node.

53



54

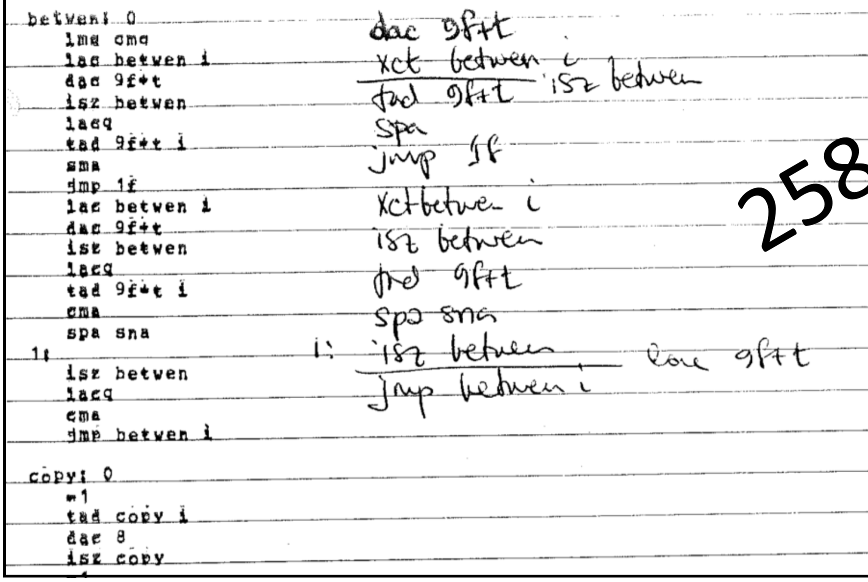
Architectural Design Decisions



55

56

PDP-7 [Unix] (1970)



```
betwen: 0
	lmq cmq
	lac betwen i
	dac 9f+t
	isz betwen
	lacq
	tad 9f+t i
	sma
	jmp 1f
	lac betwen i
	dac 9f+t
	isz betwen
	lacq
	tad 9f+t i
	cma
	spa sna
1:
	isz betwen
	lacq
	cma
	jmp betwen i

copy: 0
	-1
	tad copy i
	dac 8
	isz copy
```

```
	dac 9f+t
	xct betwen i
	isz betwen
	tad 9f+t
	spa
	jmp 1f
	xct betwen i
	isz betwen
	tad 9f+t
	spa sna
1:	isz betwen
	law 9f+t
	jmp betwen i
```

2584 lines

57

… and I once heard an old-timer growl at a young programmer:

“I've written boot loaders that were shorter than your variable names!”

— Stephen C. Johnson

58

Layering and Partitioning

```
adm.s   cat.s    dskio.s   init.s    s6.s
ald.s   check.s  dskres.s  lcase.b   s7.s
apr.s   chmod.s  dsksav.s  maksys.s  s8.s
as.s    chown.s  ds.s      s1.s      s9.s
bc.s    chrm.s   dsw.s     s2.s      scope.v
bi.s    cp.s     ed1.s     s3.s      sop.s
bl.s    db.s     ed2.s     s4.s      trysys.s
cas.s   dmabs.s  ind.b     s5.s
```

59

Process Management (fork)

```
.fork:
   jms lookfor; 0 " not-used
      skp
      jms error
   dac 9f+t
   isz uniqpid
   lac uniqpid
   dac u.ac
   law sysexit
   dac u.swapret
   lac o200000
   tad u.ulistp i
   dac u.ulistp i
   jms dskswap; 07000
   lac 9f+t
   dac u.ulistp
   lac o100000
   xor u.ulistp i
   dac u.ulistp i
   lac u.pid
```

60

Descriptor Management

```
fget: 0
   jms betwen; d0; d9
      jmp fget i
   cll; mul; 9
[redacted]
   tad ofilesp
   dac 9f+t
   dac .+2
   jms copy; ..; fnode; 3
   isz fget
   jmp fget i

fput: 0
   lac 9f+t
   dac .+3
   jms copy; fnode; ..; 3
   jmp fput i
t = t+1
```

61

Separation of File Metadata from File Naming

```
inode:
   i.flags: .=.+1
   i.dskps: .=.+7
   i.uid: .=.+1
   i.nlks: .=.+1
   i.size: .=.+1
   i.uniq: .=.+1
      . = inode+12
```

```
namei: 0
   jms iget
   -1
   tad namei i
   dac 9f+t+1
   isz namei
   lac i.flags
   and o20
   sna
   jmp namei i
   -8
   tad i.size
   cma
   lrss 3
   dac 9f+t
   sna
   jmp namei i
   dzm di
```

62

Devices as Files

```
ttyin:
    <tt>;<yi>;<n 040;040040
ttyout:
    <tt>;<yo>;<ut>; 040040
keybd:
    <ke>;<yb>;<oa>;<rd>
displ:
    <di>;<sp>;<la>;<y 040
sh:
    <sh>; 040040;040040;040040
system:
    <sy>;<st>;<em>; 040040
```

63

File I/O API

- open
- read
- write
- seek
- tell
- close

64

File System API

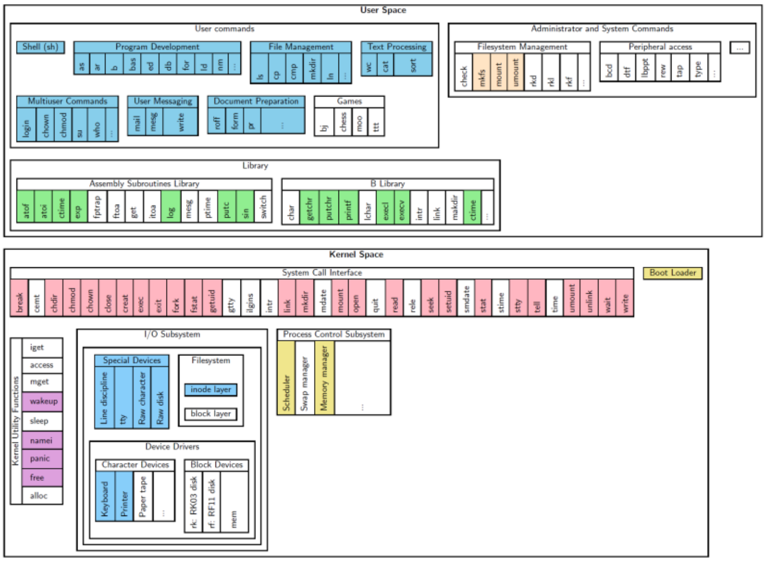
- creat
- rename
- link
- unlink

Interpreter

```
main $(                                              ind.6
   extrn read, write;
   auto i, c, state, line 100;

loop:                                                /* ind */
   state = i = 0;
loop1:
   c = read();
   if(c==4) return;
   if(c=='!' & state==0) state = 2;
   if((c<'0' ^ c>'9'&c<'a' ^ c>'z') & state==0) state = 1;
   line[i] = c;
   i = i+1;
   if(c!=012) goto loop1;
   if(state==2 ^ i==1) goto noi;
   write('  ');
   write(' ');
noi:
   i = 0;
loop3:
   c = line[i];
   write(c);
   i = i+1;
   if(c!=012) goto loop3;
   goto loop;
$)
```

67

First Research Edition (Nov 1971)



68

First Edition Unix 1972 FreeBSD 11.1 2018

```
sysrele / 0
sysexit / 1
sysfork / 2
sysread / 3
syswrite / 4
sysopen / 5
sysclose / 6
syswait / 7
syscreat / 8
syslink / 9
sysunlink / 10
```

```
0  { int nosys(void); } syscall nosys_args int
1  { void sys_exit(int rval); } exit \
       sys_exit_args void
2  { int fork(void); }
3  { ssize_t read(int fd, void *buf, \
       size_t nbyte); }
4  { ssize_t write(int fd, const void *buf, \
       size_t nbyte); }
5  { int open(char *path, int flags, int mode); }
6  { int close(int fd); }
7  { int wait4(int pid, int *status, \
       int options, struct rusage *rusage); }
8  { int creat(char *path, int mode); }
9  { int link(char *path, char *link); }
10 { int unlink(char *path); }
```

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

69

The Shell as a User Program

11/3/71 PASSWD (V)

NAME passwd -- password file

SYNOPSIS --

DESCRIPTION passwd contains for each user the following information:

name (login name)
password
numerical user ID
default working directory
program to use as Shell

This is an ASCII file. Each field within each user's entry is separated from the next by a colon. Each user is separated from the next by a new-line. If the password field is null, no password is demanded; if the Shell field is null, the Shell itself is used.

70

Abstraction of Standard I/O

11/3/71 SH (I)

Two characters cause the immediately following string to be interpreted as a special argument to the shell itself, not passed to the command. An argument of the form "<arg" causes the file arg to be used as the standard input file of the given command; an argument of the form ">arg" causes file "arg" to be used as the standard output file for the given command.

71

Interoperability through Documented File Formats

V. FILE FORMATS

a.out	assembler and loader output
archive	archive file
bppt	binary paper tape format
core	core image file
directory	directory format
file system	file system format
passwd	password file
uids	map names to user ID's
utmp	logged-in user information

72

Format	Description	Clients
a.out	Assembler and linker output	as, ld, strip, nm, un
Archive	Object code libraries	ar, ld
Core	Crashed program image	Kernel, db
Directory	File system directories	du, find, ls, ln, mkdir, rmdir
File system	File system format	check, dump,* mkfs, restor*
Ident	GECOD ident card format	opr
Password	User accounts and passwords	chown, find, getpw,* login,* ls, passwd*
Tape*	DECtape file format	mt,* tap*
Uid	User identifier to name map	chown
utmp	Logged in users	init, login,* who,* write*
wtmp*	Users login history	acct, date, init, login, tacct, who

73

User-Contributed Tools and Games

VI. USER MAINTAINED PROGRAMS

basic .... DEC supplied BASIC
bj .... the game of black jack
cal .... print calendar
chess .... the game of chess
das .... disassembler
dli .... load DEC binary paper tapes
dpt .... read DEC ASCII paper tapes
moo .... the game of MOO
sort .... sort a file
ttt .... the game of tic-tac-toe

74

Tree Directory Structure

- mkdir(II)
- chdir(II)

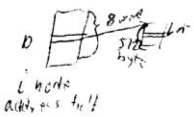
--

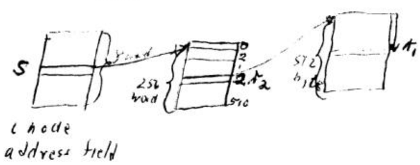
$\therefore$ max file $= 8 \times 256 \times 512 = 2^3 \times 2^8 \times 2^9 = 2^{20}$ bytes
but actually limited by offset in file table $= 2^{16}$ bytes

small $\frac{n}{512} = 0 \le b < 8$ rem r

large $\frac{n}{512} = b$ rem $0 \le r_1 < 512$ $r_1 =$ byte #

$\frac{b}{256} = 0 \le s < 8$ rem $0 \le r_2 < 256$ $2 \cdot r_2 =$ ptr





- chdir(I)
- find(I)
- ln(I)
- ls(I)
- stat(I)
- mkdir(I)
- mv(I)
- rm(I)
- rmdir(I)

75

Mountable File System Interface

- mount(II)
- umount(II)

- mount(I)
- umount(I)

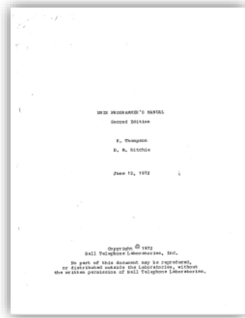
76

Second Research Edition (Jun 1972)

- Software Library

III. SUBROUTINES

atan	arctangent
atof	convert ASCII to floating
atoi	convert ASCII to integer
const	floating-point constants
ctime	convert time to ASCII
exp	exponential function
fptrap	floating-point simulator
ftoa	convert floating to ASCII
gerts	communicate with GCOS
getc	get character
hypot	compute hypotenuse
itoa	convert integer to ASCII
log	logarithm base e
mesg	print string on typewriter
nlist	read name list
ptime	print time
putc	write character or word
qsort	quicker sort
salloc	storage allocator
sin	sine, cosine
sqrt	square root
switch	transfer depending on value


UNIX PROGRAMMER'S MANUAL
Second Edition

K. Thompson
D. M. Ritchie

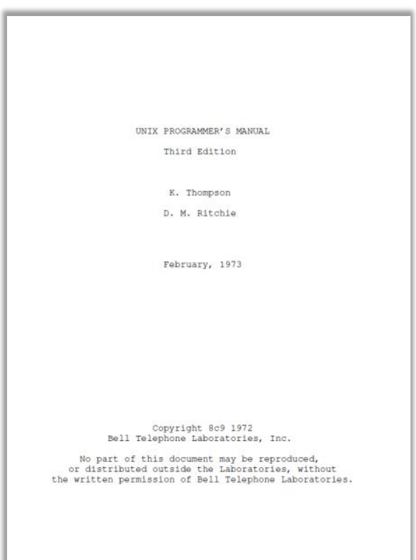
June 12, 1972

Copyright © 1972
Bell Telephone Laboratories, Inc.

No part of this document may be reproduced,
or distributed outside the Laboratories, without
the written permission of Bell Telephone Laboratories.

77

Third Research Edition (Feb 1973)


UNIX PROGRAMMER'S MANUAL
Third Edition

K. Thompson
D. M. Ritchie

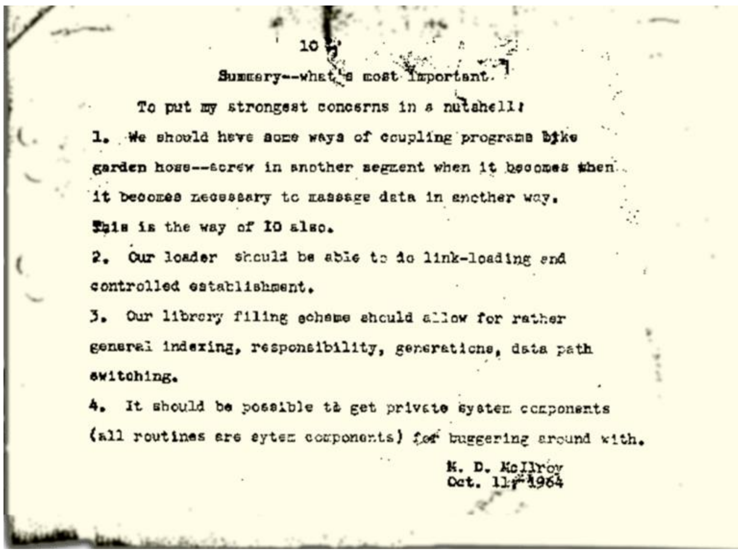
February, 1973

Copyright 8c9 1972
Bell Telephone Laboratories, Inc.

No part of this document may be reproduced,
or distributed outside the Laboratories, without
the written permission of Bell Telephone Laboratories.

78

Pipes and Filters



10

Summary--what's most important.

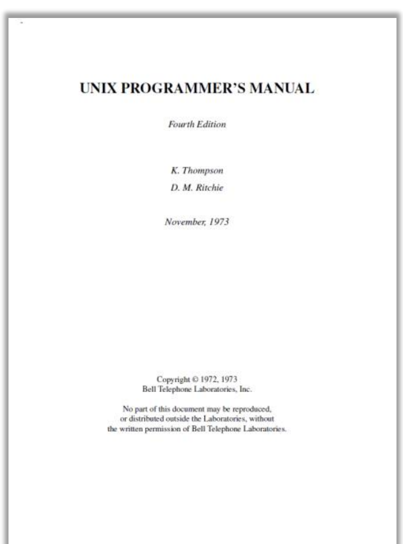
To put my strongest concerns in a nutshell:

1. We should have some ways of coupling programs like garden hose--screw in another segment when it becomes when it becomes necessary to massage data in another way. This is the way of IO also.
2. Our loader should be able to do link-loading and controlled establishment.
3. Our library filing scheme should allow for rather general indexing, responsibility, generations, data path switching.
4. It should be possible to get private system components (all routines are system components) for buggering around with.

M. D. McIlroy
Oct. 11, 1964

79

Fourth Research Edition (Nov 1973)



UNIX PROGRAMMER'S MANUAL

Fourth Edition

K. Thompson
D. M. Ritchie

November, 1973

Copyright © 1972, 1973
Bell Telephone Laboratories, Inc.

No part of this document may be reproduced,
or distributed outside the Laboratories, without
the written permission of Bell Telephone Laboratories.

80

Structured Programming

- Kernel implemented in “New B”
- 6373 lines New B
- 768 lines PDP-11 assembly
- 105 functions + 50 assembly symbols
- vs 248 global symbols in the First Edition

```
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;
```

81

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.

82

Data Structure Definition & Reuse

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

83

Dynamic Resource Management

```
int     coremap[CMAPSIZ];
int     swapmap[SMAPSIZ];

struct map {
        char *m_size;
        char *m_addr;
};

malloc(mp, size)
struct map *mp;
{
 …
```

84

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

85

Driver Interface

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

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

86

Buffer Cache

Fourth Edition

```
#define B_READ   01
#define B_DONE   02
#define B_ERROR  04
#define B_BUSY   010
#define B_XMEM   060
#define B_WANTED 0100
#define B_RELOC  0200
#define B_ASYNC  0400
#define B_DELWRI 01000
```

FreeBSD 11.1

```
#define B_ASYNC 0x00000004
 /* Start I/O, do not wait.
   */
[…]
#define B_DELWRI 0x00000080
/* Delay I/O until buffer
   reused. */
#define B_DONE 0x00000200
/* I/O completed. */
```

87

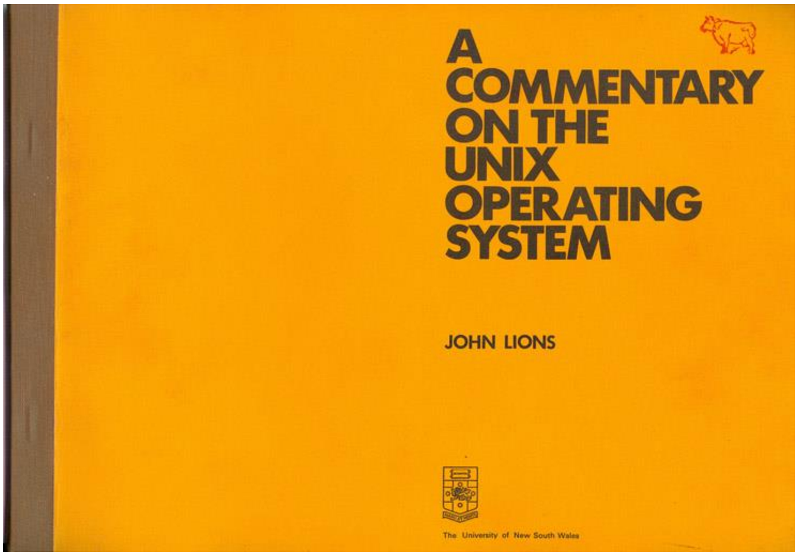
Fifth Research Edition (Jun 1974)

- Command Files

```
chdir /usr/source/s3
cc -c ctime.c
ar r /lib/liba.a ctime.o
rm ctime.o
chdir /usr/source/s1
cc -s -n date.c
cp a.out /bin/date
cc -s -n dump.c
cp a.out /bin/dump
cc -s -n ls.c
cp a.out /bin/ls
rm a.out
```

88

Sixth Research Edition (May 1975)



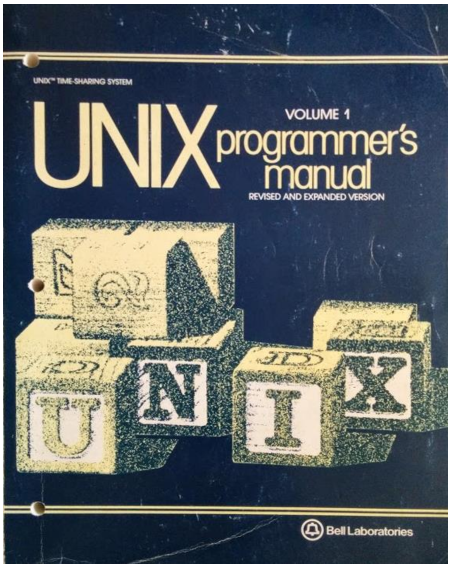
89

Portable C Library

```
alloc.c    clenf.c    makbuf.c   scan1.c
calloc.c   copen.c    maktab.c   scan2.c
cclose.c   cputc.c    nexch.c    scan3.c
ceof.c     cwrd.c     nodig.c    system.c
cerror.c   dummy.s    printf.c   tmpnam.c
cexit.c    ftoa.c     putch.c    unget.c
cflush.c   getch.c    puts.c     unprnt.s
cfree.c    gets.c     relvec.c   wdleng.c
cgetc.c    getvec.c   revput.c
ciodec.c   iehzap.c   run
```

90

Seventh Research Edition (Jan 1979)



91

Unix as a Virtual Machine

Also, about this time [1973] I had a fateful discussion with Dennis, in which he said:

"I think it may be easier to port Unix to a new piece of hardware than to port a complex application from Unix to a new OS"

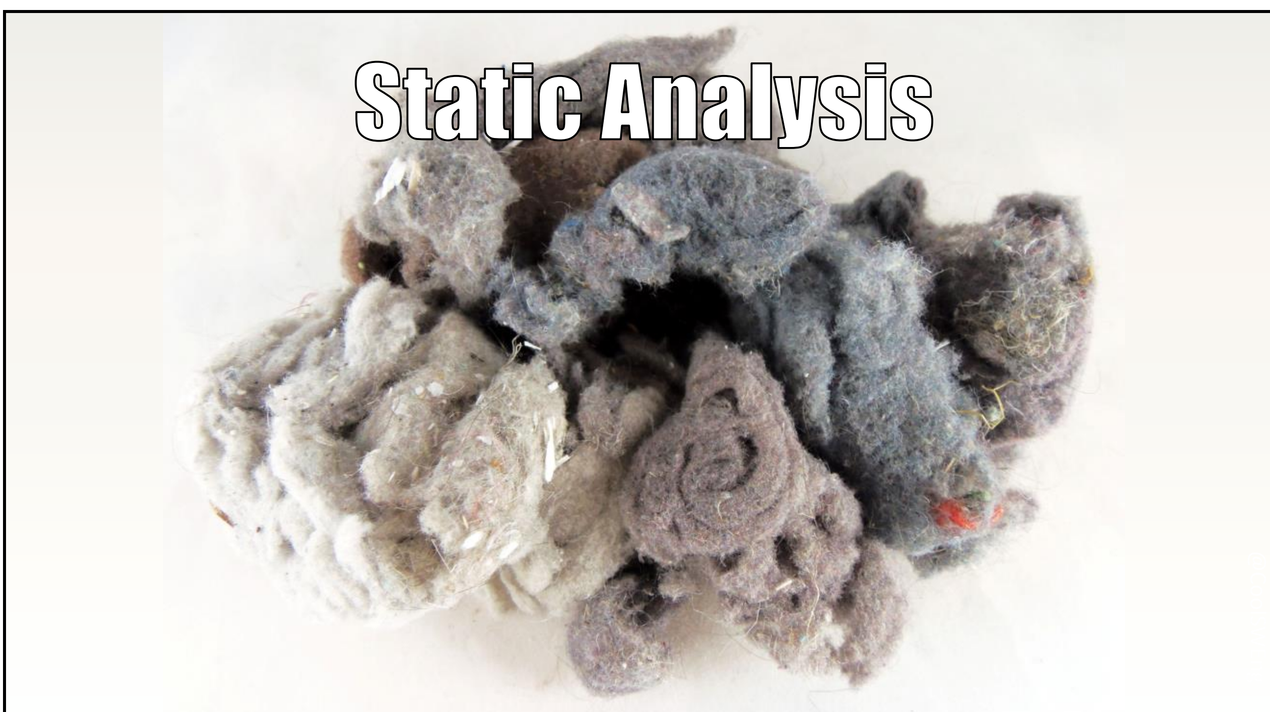
— Steve Johnson

92

Dynamic User Memory Allocation

- malloc(3), free(3)
  - Used by 26 programs: awk cc col cron dc dcheck diff ed eqn expr graph icheck learn ls m4 neqn nm quot ratfor spline struct tar tsort uucp xsend quiz
- stdio(3), mp(3)

93



94

Environment Variables

- KEY=value
- Kernel
- Shell
- C Library

ENVIRON (5) UNIX Programmer's Manual ENVIRON (5)

NAME

environ – user environment

SYNOPSIS

extern char **environ;

DESCRIPTION

An array of strings called the 'environment' is made available by *exec*(2) when a process begins. By convention these strings have the form 'name=value'. The following names are used by various commands:

PATH The sequence of directory prefixes that *sh, time, nice*(1), etc., apply in searching for a file known by an incomplete path name. The prefixes are separated by ':'. *Login*(1) sets PATH=:/bin:/usr/bin.

HOME A user's login directory, set by *login*(1) from the password file *passwd*(5).

TERM The kind of terminal for which output is to be prepared. This information is used by commands, such as *nroff* or *plot*(1), which may exploit special terminal capabilities. See *term*(7) for a list of terminal types.

Further names may be placed in the environment by the *export* command and 'name=value' arguments in *sh*(1), or by *exec*(2). It is unwise to conflict with certain Shell variables that are frequently exported by '.profile' files: MAIL, PS1, PS2, IFS.

SEE ALSO

exec(2), sh(1), term(7), login(1)

95

Language Development Tools

- lex(1)
- yacc(1)
- 12 clients: awk bc cpp egrep eqn lex m4 make pcc neqn struct
-

Copyright © 1978 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 57, No. 6, July-August 1978
Printed in U. S. A.

UNIX Time-Sharing System:

Language Development Tools

By S. C. JOHNSON and M. E. LESK
(Manuscript received December 27, 1977)

The development of new programs on the UNIX system is facilitated by tools for language design and implementation. These are frequently program generators, compiling into C, which provide advanced algorithms in a convenient form, while not restraining the user to a preconceived set of jobs. Two of the most important such tools are Yacc, a generator of LALR(1) parsers, and Lex, a generator of regular expression recognizers using deterministic finite automata. They have been used in a wide variety of applications, including compilers, desk calculators, typesetting languages, and pattern processors.*

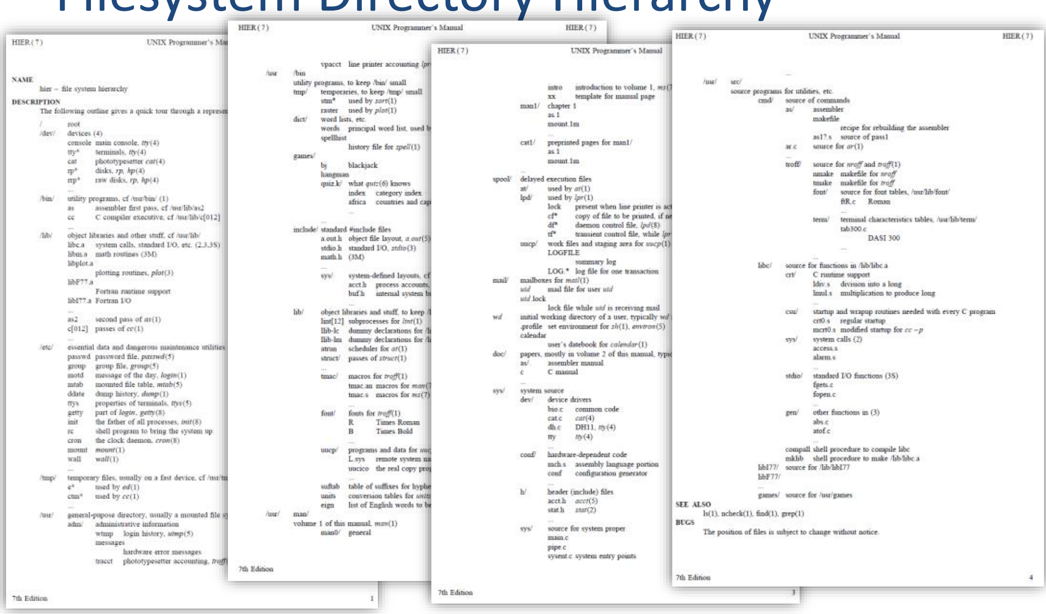
96

Domain-Specific Languages

- sh
- awk
- sed
- find
- expr
- egrep
- m4
- make

97

Filesystem Directory Hierarchy



98

99

First and Second Berkeley Software Distributions (1978)

- Software Packages
  - csh
  - ex
  - Mail
  - Pascal
  - termlib

100

3BSD (1979)

- Virtual Memory Paging
  - vm_*.c
  - 2808 out of 16039 C source code
  - 17% of kernel source code
  - vread(2), vwrite(2), vfork(2)

101

4BSD (Oct 1980)

102

Regular Expression Library: regex(3)

– 5 implementations: awk, sed, ed, grep, expr
– 1 client: more(1)
– 2 more by 4.3: dbx(1), rdist(1)
– 4 replacements in FreeBSD: ed, grep, sed, expr

103

Optimized Screen Handling

– curses(3)
– termcap(5)

104

4.2BSD (Sep 1983)

- Internet Protocol Family
  - ARP, IP, TCP, UDP, ICMP
- Local and Remote Interprocess Communication
  - socket(2), etc.
- Network and User Database Access
  - getfsent(3x), getgrent(3), gethostent(3n), getnetent(3n), getprotoent(3n), getpwent(3), getservent(3n)
- Pseudo-Terminal Driver
  - pty(4)

105

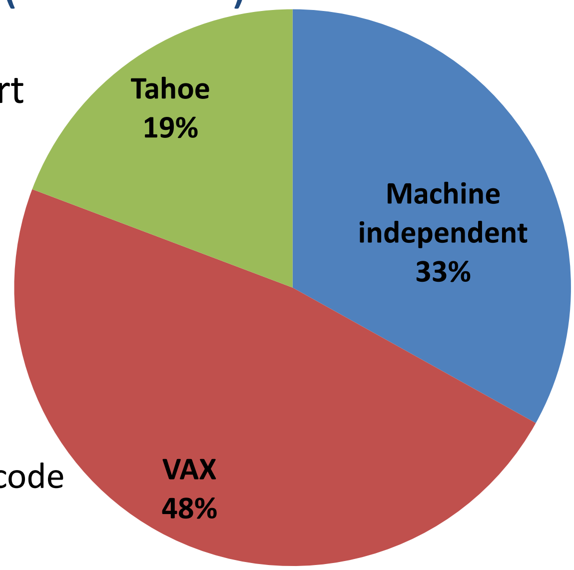
- 2 library functions
  - rcmd(3x)
  - rexec(3x)
- 11 system daemons
  - comsat(8c)
  - ftpd(8c)
  - gettable(8c)
  - implogd(8c)
  - rexecd(8c)
  - rlogind(8c)
  - af(8c)
  - rshd(8c)
  - rwhod(8c)
  - telnetd(8c)
  - tftpd(8c)
- 8 user-mode programs
  - ftp(1c)
  - rlogin(1c)
  - rsh(1c)
  - talk(1c)
  - telnet(1c)
  - tftp(1c)
  - whois(1c)
  - sendmail(1c)

System call	Uses
bind	23
connect	15
accept	13
select	12
listen	11
sendto	10
shutdown	9
recvfrom	8
getsockname	6
recv	2
send	2
sendmsg	1
getsockopt	0
recvmsg	0
socketpair	0

106

4.3BSD Tahoe (Jun 1988)

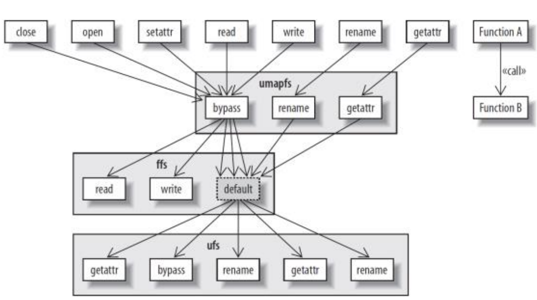
- Multiple CPU Architecture Support
  - VAX
  - CCI Power 6/32 (Tahoe)
- Timezone Handling
  - Community contribution
  - Separation of timezone rules from code



107

4.3BSD Reno (Jun 1990)

- Kernel Packet Forwarding Database
  - route(4)
  - routed(8), XNSrouted(8)
- Virtual Filesystem Interface




```
/*
 * Operations on vnodes.
 */
struct vnodeops {
        int     (*vn_lookup)(           /* ndp */ );
        int     (*vn_create)(           /* ndp, fflags, vap, cred */ );
        int     (*vn_mknod)(            /* ndp, vap, cred */ );
        int     (*vn_open)(             /* vp, fflags, cred */ );
        int     (*vn_close)(            /* vp, fflags, cred */ );
        int     (*vn_access)(           /* vp, fflags, cred */ );
        int     (*vn_getattr)(          /* vp, vap, cred */ );
        int     (*vn_setattr)(          /* vp, vap, cred */ );

        int     (*vn_read)(             /* vp, uiop, offp, ioflag, cred */ );
        int     (*vn_write)(            /* vp, uiop, offp, ioflag, cred */ );
        int     (*vn_ioctl)(            /* vp, com, data, fflag, cred */ );
        int     (*vn_select)(           /* vp, which, cred */ );
        int     (*vn_mmap)(             /* vp, ..., cred */ );
        int     (*vn_fsync)(            /* vp, fflags, cred */ );
        int     (*vn_seek)(             /* vp, (old)offp, off, whence */ );
….
```

108

4.3BSD Net/2 (Jun 1991)

- Stream I/O Functions
  - funopen(3)
  - GNU funopencookie(3) added in FreeBSD 11

109

4.4BSD (Jun 1994)

- Stackable Filesystems
  - mount_null(8)
  - mount_union(8)
- Generic System Control Interface (MIB)
  - sysctl(1)
  - sysctl(3)
  - sysctl(9)

110

111

386BSD Patch Kit (1992-1993)

- Organized Community Contributions
  - From open source software …
  - … to an open source **project**
- Patch metadata
  - title
  - author
  - description
  - prerequisites

112

FreeBSD

113

FreeBSD 1.1 (May 1994)

- Package Manager
  - Patch
  - Compile
  - Install
  - Uninstall
  - Handling of dependencies

114

FreeBSD 2.0 (Nov 1994)

- Process Filesystem
  - procfs(5)
- Dynamically Loadable Kernel Modules
  - lkm(4), then kld(4)
  - device drivers
  - file systems
  - emulators
  - system calls
  - **992** modules in FreeBSD 11.1

115

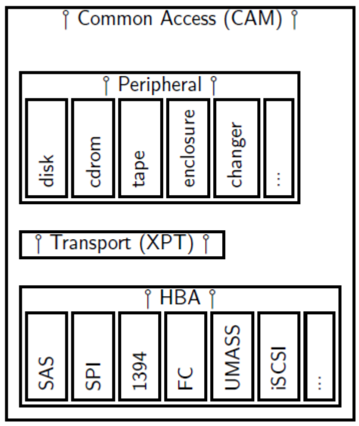
FreeBSD 2.1 (Nov 1995)

- Linux Emulation
- Packet Capture Library

116

FreeBSD 3.0 (Jan 1999)

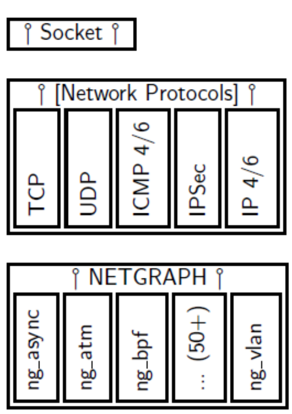
- Common Access Method I/O Subsystem (CAM)



117

FreeBSD 3.4 (Dec 1999)

- Graph-based Kernel Networking and User Library (NETGRAPH)



118

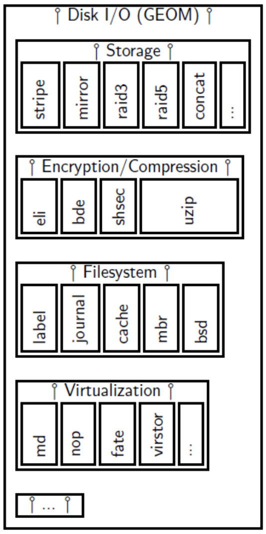
FreeBSD 4.0 (Mar 2000)

- OpenSSL Secure Sockets Layer and Transport Layer Security framework
  - Version 0.9.4
  - 1127 files, 227118 lines
  - libssl(3), libcrypto(3), openssl(1)
- Jail: Isolate a process and its descendants
- Access Control Lists

119

FreeBSD 5.0 (May 2006)

- Symmetric Multiprocessing
- Modular Disk I/O Request Transformation Framework (GEOM)
- Mandatory Access Control
- Pluggable Authentication Module



120

FreeBSD 5.3 (Nov 2004)

- Streaming Archive Access Library
- Miniport Driver Wrapper

121

FreeBSD 6.2 (Jan 2007)

- Basic Security Module Auditing

122

FreeBSD 7 (Feb 2008)

- ZFS File System and Storage Pools
- Dynamic Tracing

123

FreeBSD 9.0 (Jan 2012)

- Para-virtualized I/O
- Infiniband Support
- Application Compartmentalization

124

FreeBSD 10.0 (Jan 2014)

- Virtual Machine Monitor (bhyve)
- Fast User Space Raw Packet Processing

125

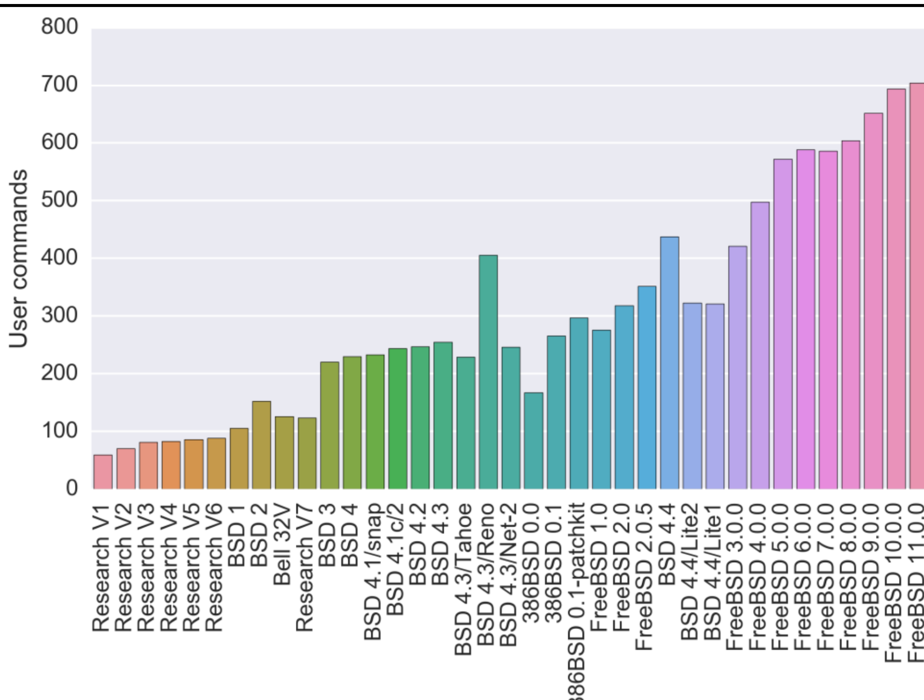
FreeBSD 11.0 (Sep 2016)

- Network Blacklisting

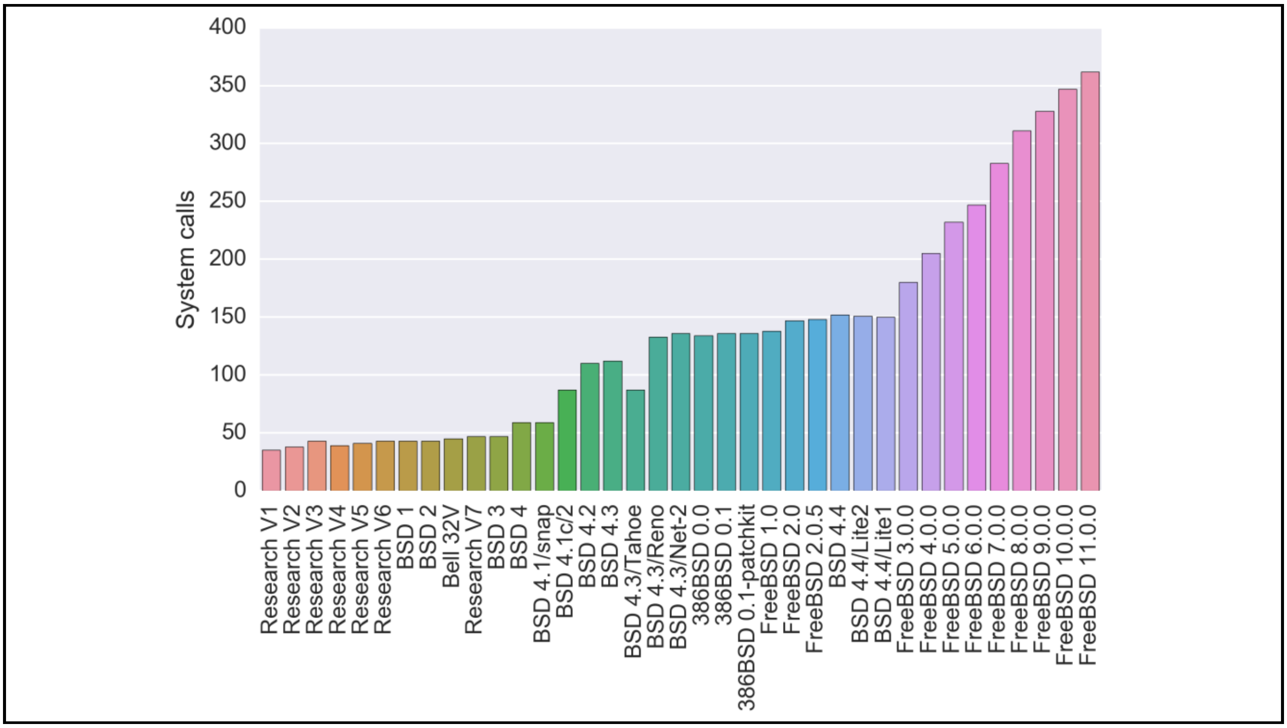
126

Quantitative Results

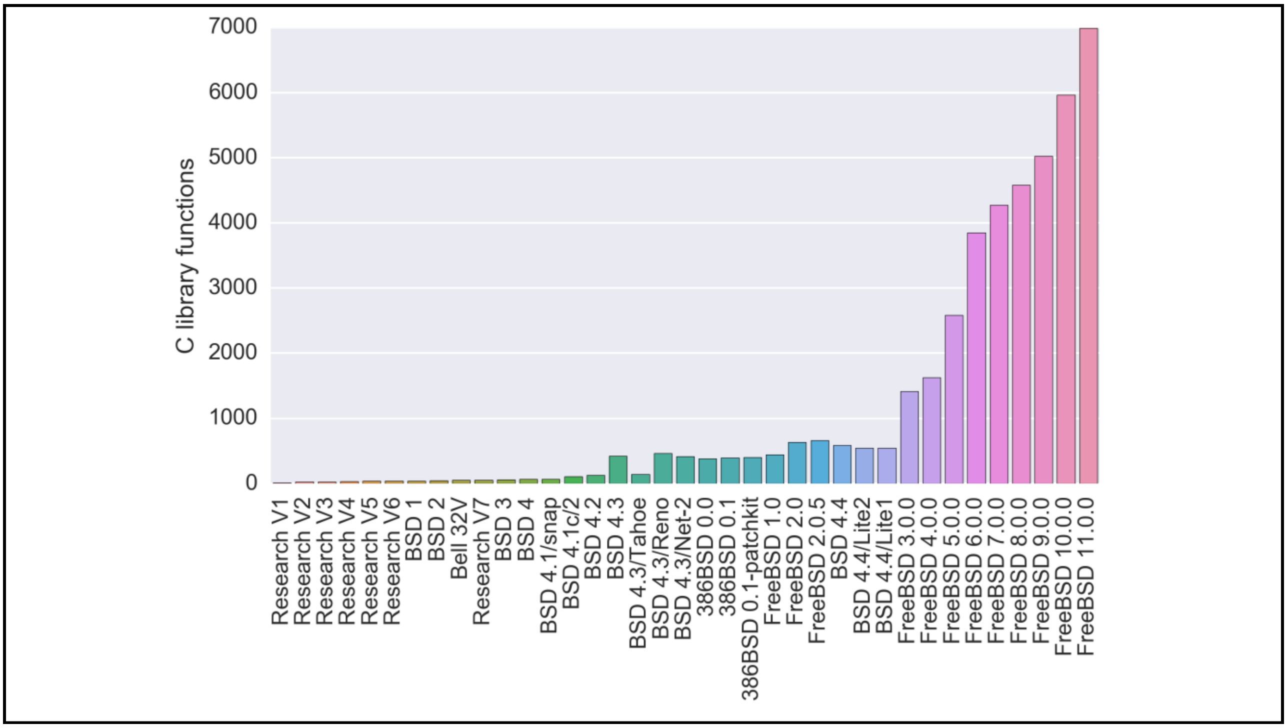
127



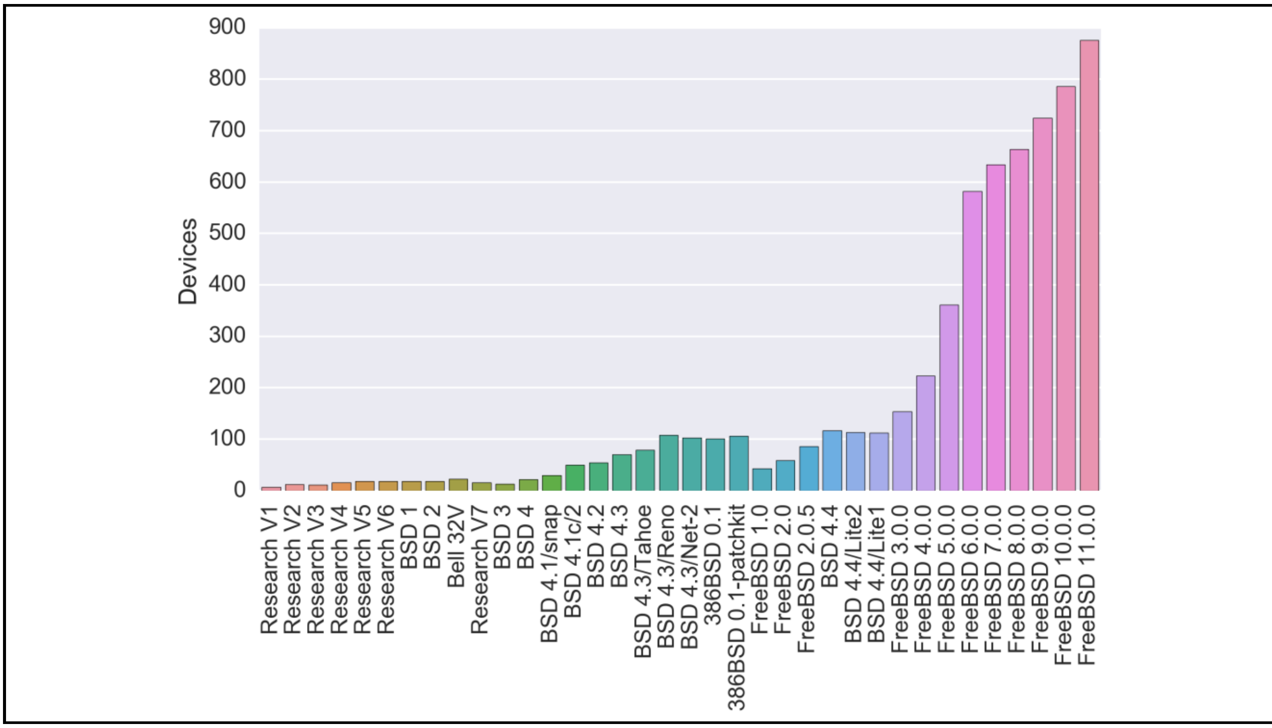
128


System calls
0
50
100
150
200
250
300
350
400
Research V1
Research V2
Research V3
Research V4
Research V5
Research V6
BSD 1
BSD 2
Bell 32V
Research V7
BSD 3
BSD 4
BSD 4.1/snap
BSD 4.1c/2
BSD 4.2
BSD 4.3
BSD 4.3/Tahoe
BSD 4.3/Reno
BSD 4.3/Net-2
386BSD 0.0
386BSD 0.1
386BSD 0.1-patchkit
FreeBSD 1.0
FreeBSD 2.0
FreeBSD 2.0.5
BSD 4.4
BSD 4.4/Lite2
BSD 4.4/Lite1
FreeBSD 3.0.0
FreeBSD 4.0.0
FreeBSD 5.0.0
FreeBSD 6.0.0
FreeBSD 7.0.0
FreeBSD 8.0.0
FreeBSD 9.0.0
FreeBSD 10.0.0
FreeBSD 11.0.0

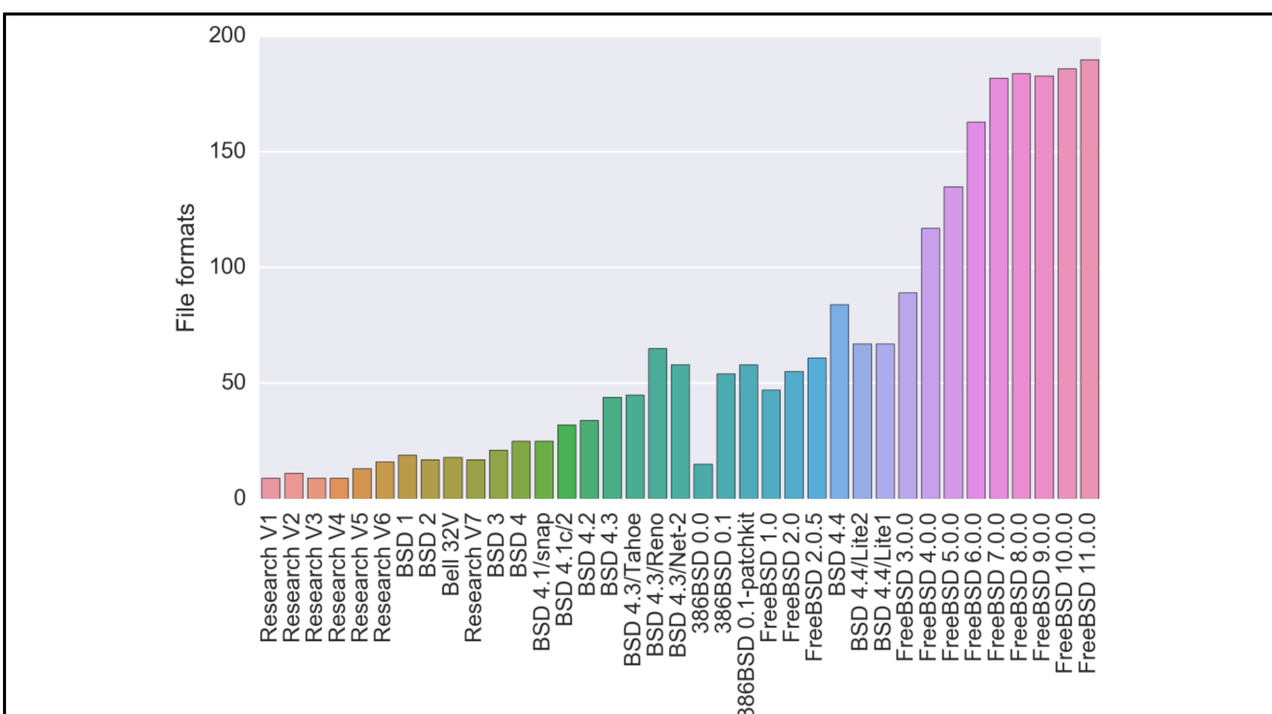
129


C library functions
0
1000
2000
3000
4000
5000
6000
7000
Research V1
Research V2
Research V3
Research V4
Research V5
Research V6
BSD 1
BSD 2
Bell 32V
Research V7
BSD 3
BSD 4
BSD 4.1/snap
BSD 4.1c/2
BSD 4.2
BSD 4.3
BSD 4.3/Tahoe
BSD 4.3/Reno
BSD 4.3/Net-2
386BSD 0.0
386BSD 0.1
386BSD 0.1-patchkit
FreeBSD 1.0
FreeBSD 2.0
FreeBSD 2.0.5
BSD 4.4
BSD 4.4/Lite2
BSD 4.4/Lite1
FreeBSD 3.0.0
FreeBSD 4.0.0
FreeBSD 5.0.0
FreeBSD 6.0.0
FreeBSD 7.0.0
FreeBSD 8.0.0
FreeBSD 9.0.0
FreeBSD 10.0.0
FreeBSD 11.0.0

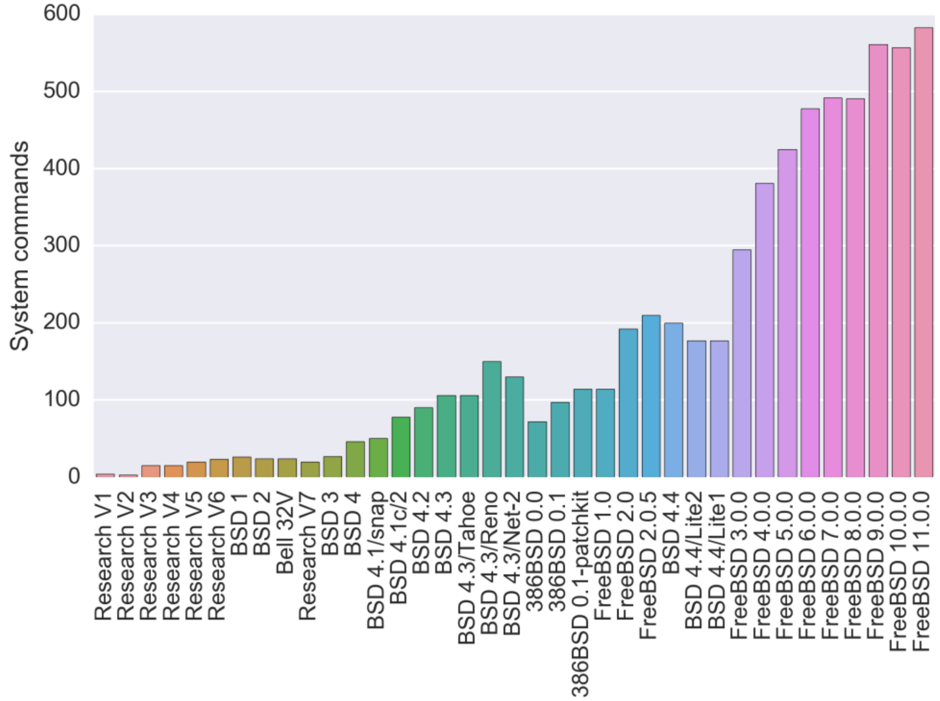
130


Devices
0
100
200
300
400
500
600
700
800
900
Research V1
Research V2
Research V3
Research V4
Research V5
Research V6
BSD 1
BSD 2
Bell 32V
Research V7
BSD 3
BSD 4
BSD 4.1/snap
BSD 4.1c/2
BSD 4.2
BSD 4.3
BSD 4.3/Tahoe
BSD 4.3/Reno
BSD 4.3/Net-2
386BSD 0.1
386BSD 0.1-patchkit
FreeBSD 1.0
FreeBSD 2.0
FreeBSD 2.0.5
BSD 4.4
BSD 4.4/Lite2
BSD 4.4/Lite1
FreeBSD 3.0.0
FreeBSD 4.0.0
FreeBSD 5.0.0
FreeBSD 6.0.0
FreeBSD 7.0.0
FreeBSD 8.0.0
FreeBSD 9.0.0
FreeBSD 10.0.0
FreeBSD 11.0.0

131

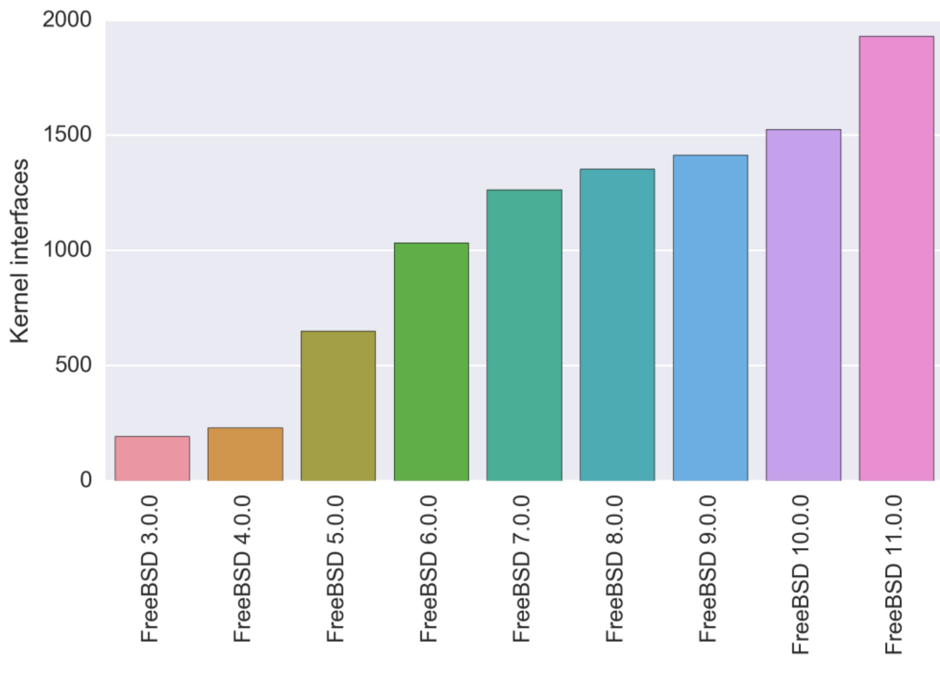

File formats
0
50
100
150
200
Research V1
Research V2
Research V3
Research V4
Research V5
Research V6
BSD 1
BSD 2
Bell 32V
Research V7
BSD 3
BSD 4
BSD 4.1/snap
BSD 4.1c/2
BSD 4.2
BSD 4.3
BSD 4.3/Tahoe
BSD 4.3/Reno
BSD 4.3/Net-2
386BSD 0.0
386BSD 0.1
386BSD 0.1-patchkit
FreeBSD 1.0
FreeBSD 2.0
FreeBSD 2.0.5
BSD 4.4
BSD 4.4/Lite2
BSD 4.4/Lite1
FreeBSD 3.0.0
FreeBSD 4.0.0
FreeBSD 5.0.0
FreeBSD 6.0.0
FreeBSD 7.0.0
FreeBSD 8.0.0
FreeBSD 9.0.0
FreeBSD 10.0.0
FreeBSD 11.0.0

132



System commands
0
100
200
300
400
500
600
Research V1
Research V2
Research V3
Research V4
Research V5
Research V6
BSD 1
BSD 2
Bell 32V
Research V7
BSD 3
BSD 4
BSD 4.1/snap
BSD 4.1c/2
BSD 4.2
BSD 4.3
BSD 4.3/Tahoe
BSD 4.3/Reno
BSD 4.3/Net-2
386BSD 0.0
386BSD 0.1
386BSD 0.1-patchkit
FreeBSD 1.0
FreeBSD 2.0
FreeBSD 2.0.5
BSD 4.4
BSD 4.4/Lite2
BSD 4.4/Lite1
FreeBSD 3.0.0
FreeBSD 4.0.0
FreeBSD 5.0.0
FreeBSD 6.0.0
FreeBSD 7.0.0
FreeBSD 8.0.0
FreeBSD 9.0.0
FreeBSD 10.0.0
FreeBSD 11.0.0

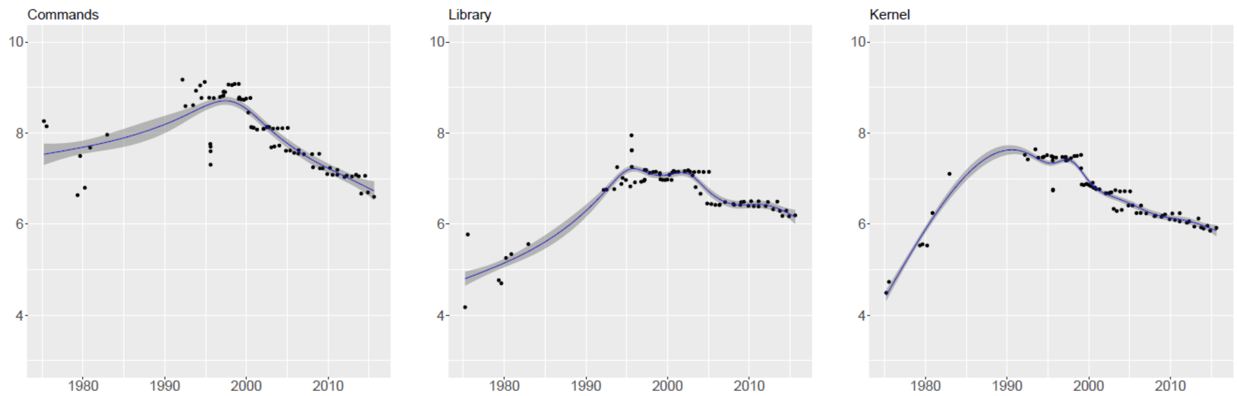
133



Kernel interfaces
0
500
1000
1500
2000
FreeBSD 3.0.0
FreeBSD 4.0.0
FreeBSD 5.0.0
FreeBSD 6.0.0
FreeBSD 7.0.0
FreeBSD 8.0.0
FreeBSD 9.0.0
FreeBSD 10.0.0
FreeBSD 11.0.0

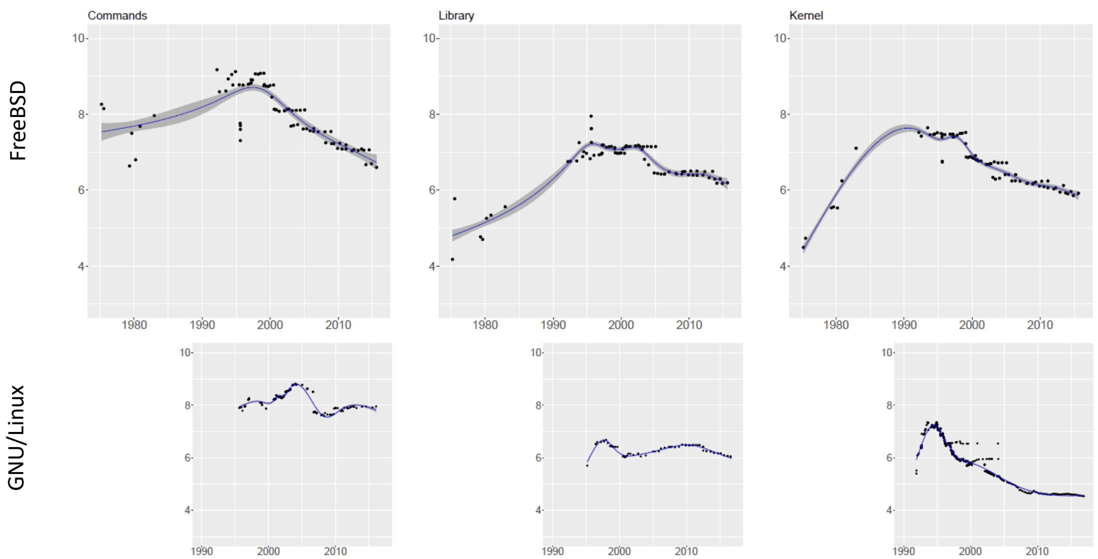
134

Mean Cyclomatic Complexity



135

Mean Cyclomatic Complexity



136

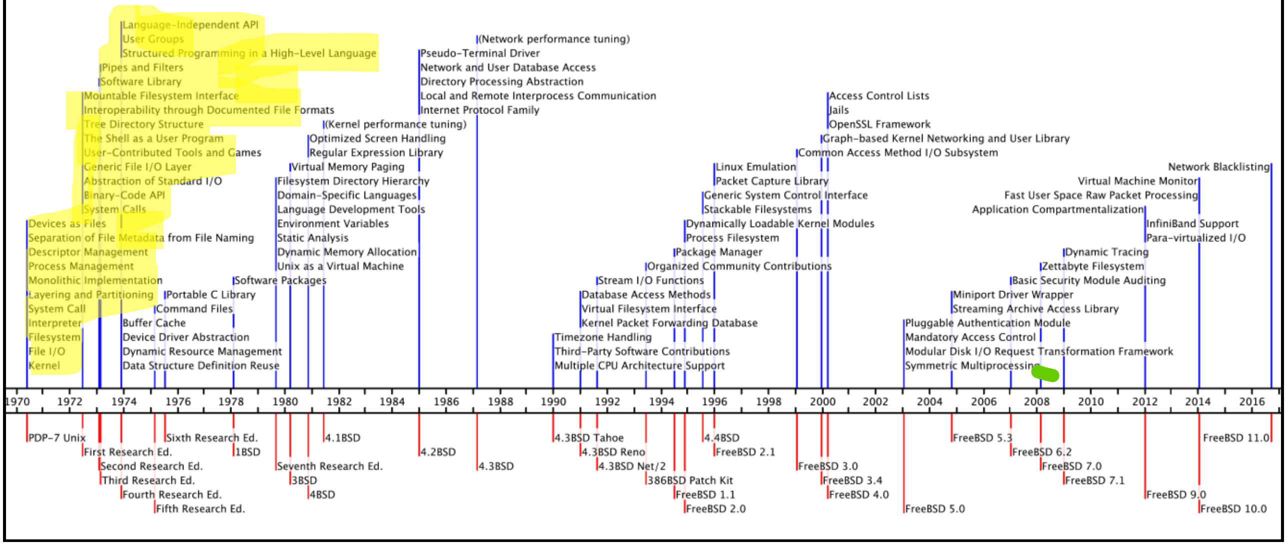
Theory of OS Architectural Evolution

137

Form and Pace of Architectural Evolution

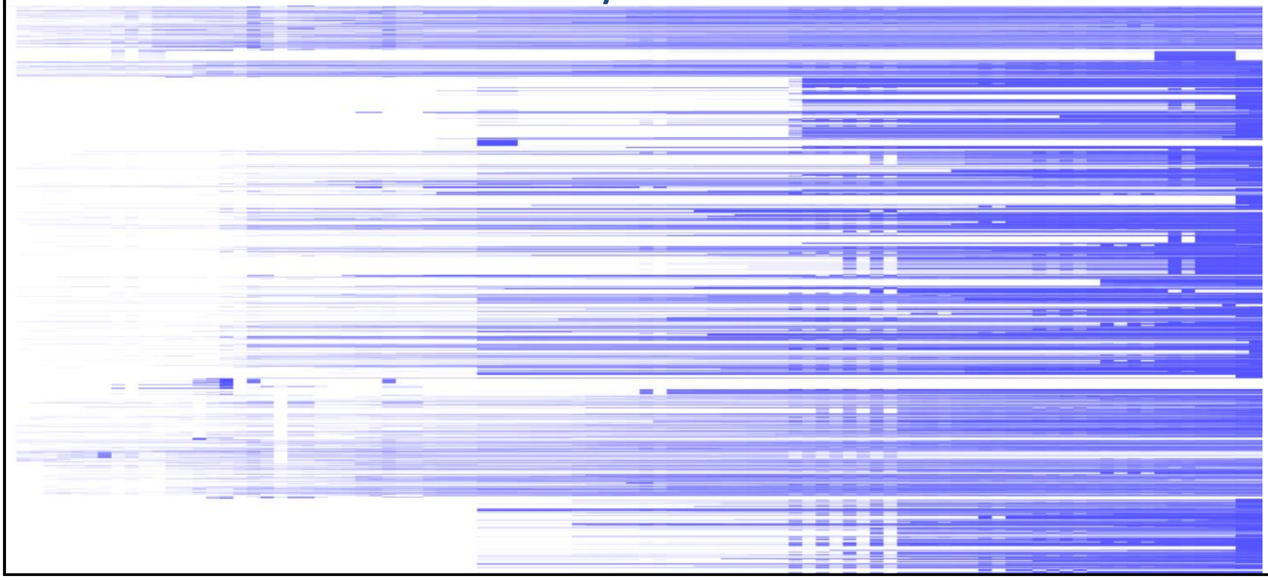
138

Many core architecture decisions are taken at the beginning of the system's lifetime

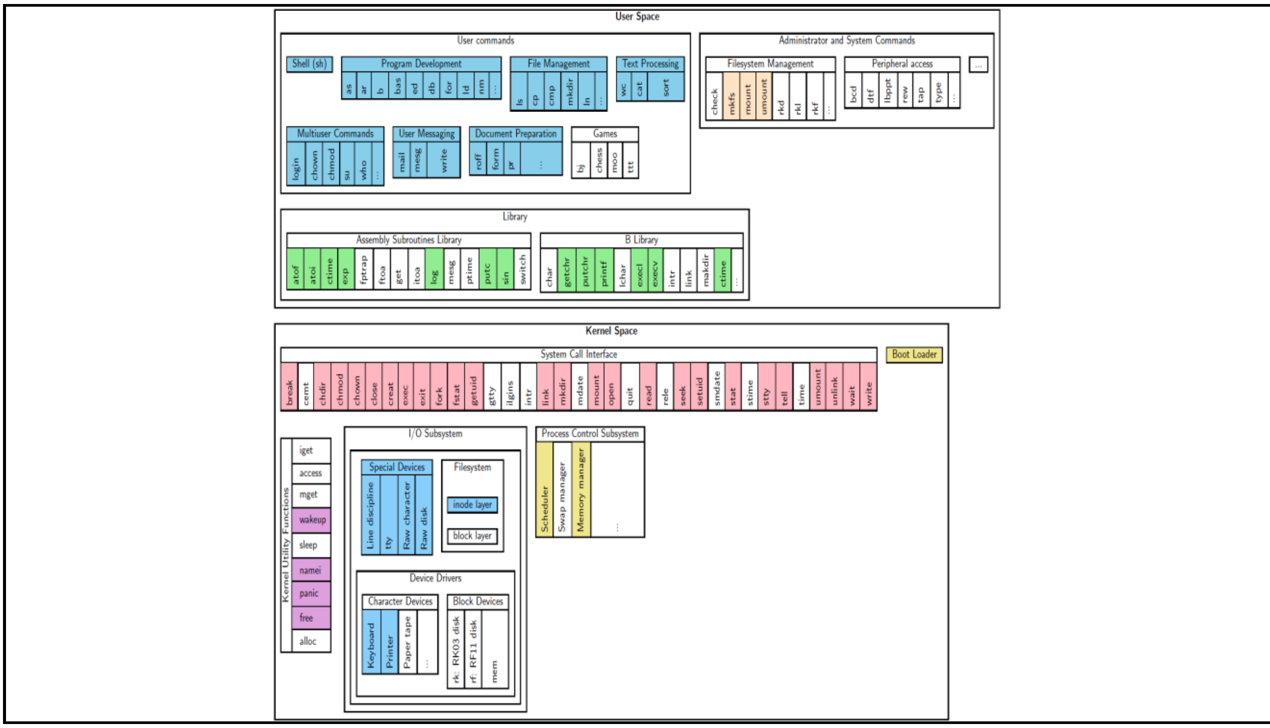


139

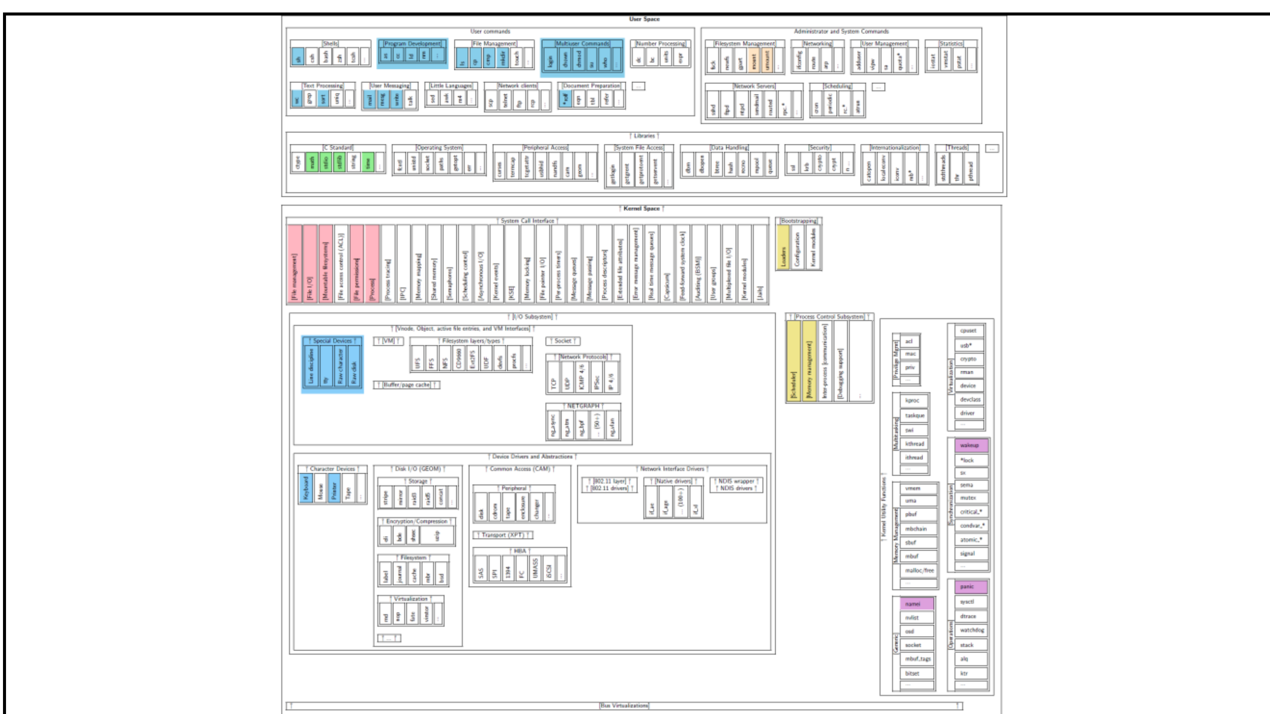
Most architecture decisions survive over the system lifetime



140

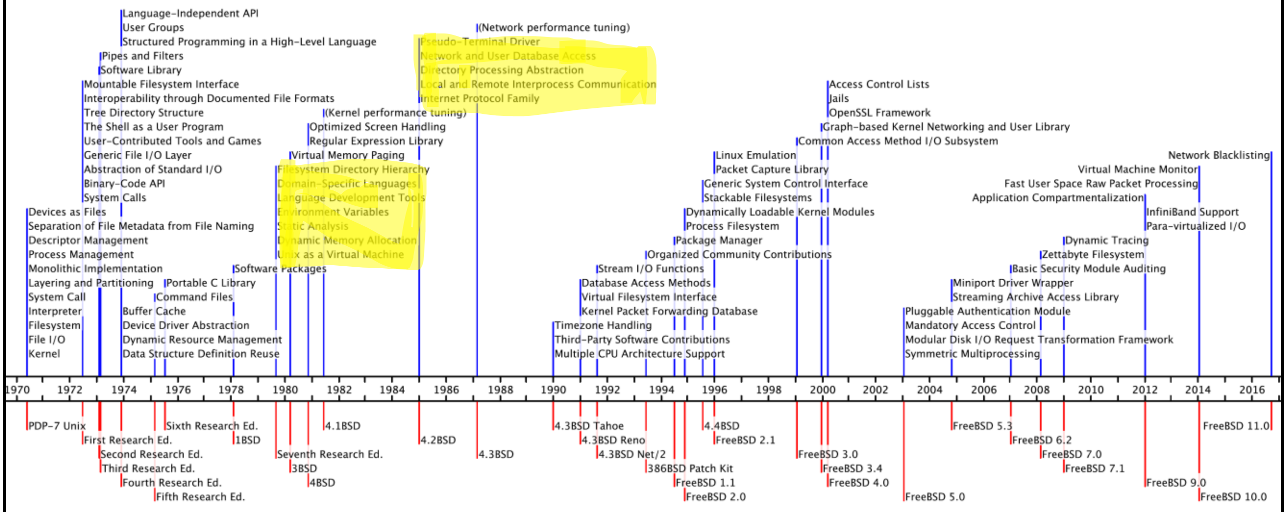

User Space
User commands
Shell (sh)
Program Development
as
ar
b
bas
ed
db
for
ld
nm
...
File Management
ls
cp
cmp
mkdir
ln
...
Text Processing
wc
cat
sort
Multiuser Commands
login
chown
chmod
su
who
...
User Messaging
mail
mesg
write
Document Preparation
roff
form
pr
...
Games
bj
chess
moo
ttt
Administrator and System Commands
Filesystem Management
check
mkfs
mount
umount
rkd
rkl
rkf
...
Peripheral access
bcd
dtf
lbppt
rew
tap
type
...
...
Library
Assembly Subroutines Library
atof
atoi
ctime
exp
fptrap
ftoa
get
itoa
log
mesg
ptime
putc
sin
switch
B Library
char
getchr
putchr
printf
lchar
execl
execv
intr
link
makdir
ctime
...
Kernel Space
System Call Interface
break
cemt
chdir
chmod
chown
close
creat
exec
exit
fork
fstat
getuid
gtty
ilgins
intr
link
mkdir
mdate
mount
open
quit
read
rele
seek
setuid
smdate
stat
stime
stty
tell
time
umount
unlink
wait
write
Boot Loader
Kernel Utility Functions
iget
access
mget
wakeup
sleep
namei
panic
free
alloc
I/O Subsystem
Special Devices
Line discipline
tty
Raw character
Raw disk
Filesystem
inode layer
block layer
Device Drivers
Character Devices
Keyboard
Printer
Paper tape
...
Block Devices
rk: RK03 disk
rf: RF11 disk
mem
Process Control Subsystem
Scheduler
Swap manager
Memory manager
...

141


User Space
User commands
Administrator and System Commands
Kernel Space
System Call Interface
I/O Subsystem
Process Control Subsystem
Device Drivers and Abstractions
Kernel Utility Functions

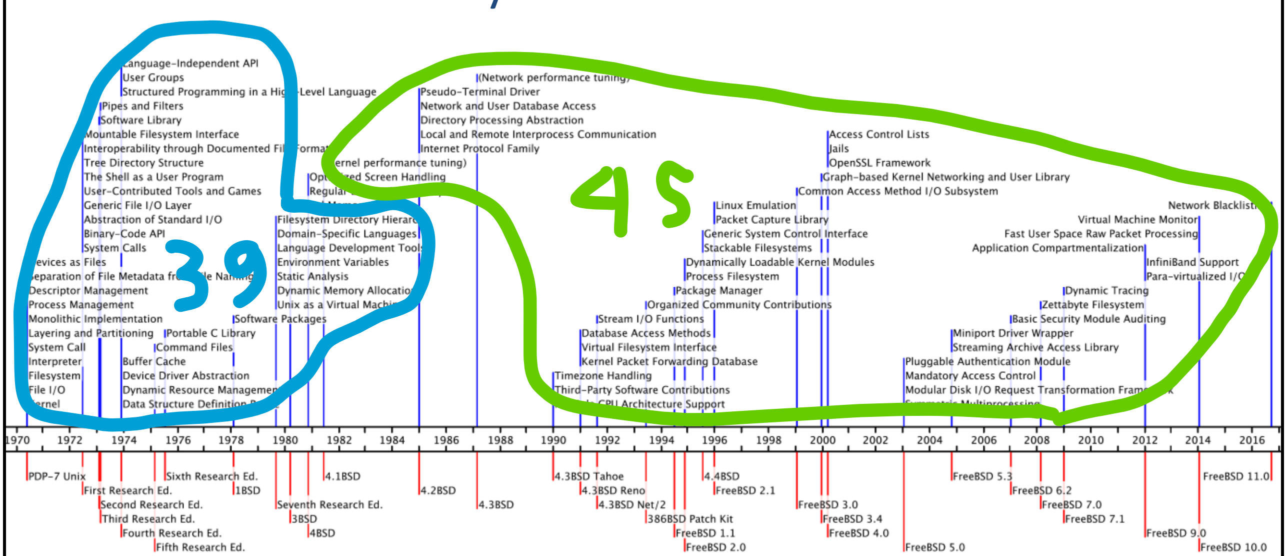
142

New architecture decisions are continuously made, further fueling architecture evolution



143

The rate of architecture decisions declines over the system's lifetime



144

Accumulation of Architectural Technical Debt

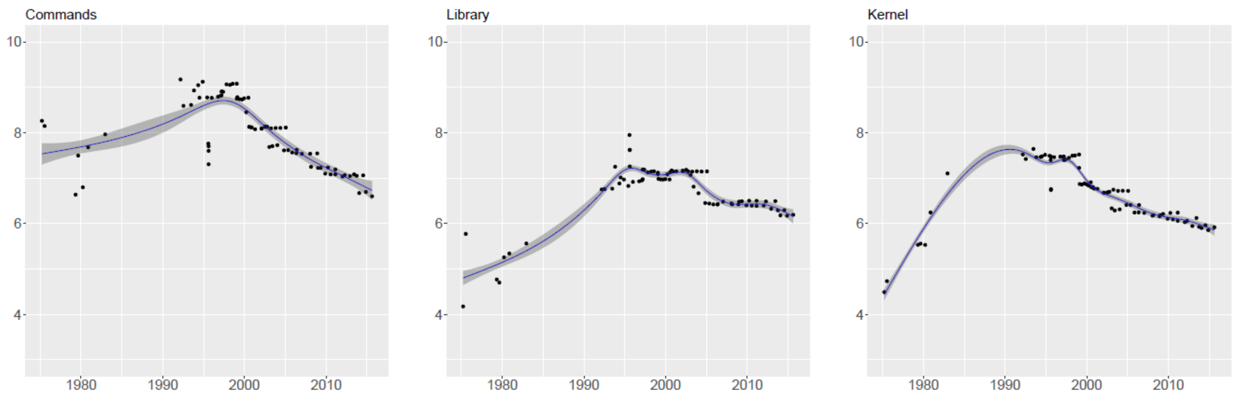
145

Architecture decisions offering features that are either similar to existing ones or remain under-used

- read, pread, readv, preadv, recv, recvfro, recvmsg
- /var/log, syslogd, acct, auditd
- UGO permissions, ACL, MAC
- Threads and processes for multitasking

146

Debt systematically paid back despite increasing system size and complexity



147

Forces of Architectural Evolution

148

Conventions instead of enforcement facilitate evolution by reducing effort and offering flexibility

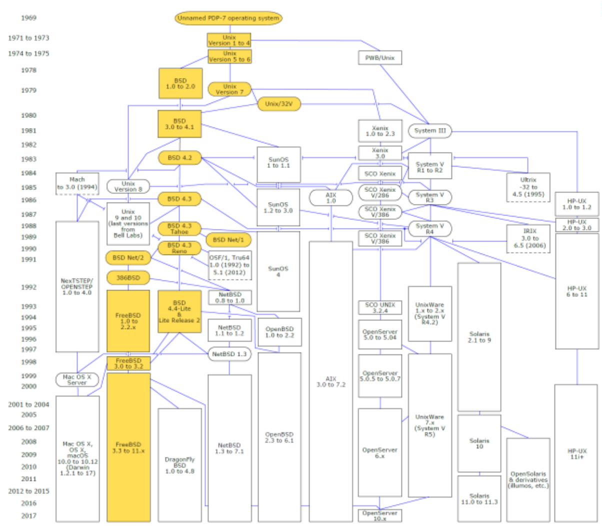
- Text files
- Directory hierarchy
- Documented file formats
- Environment variables

149

Portability, due to its inherent complexity, is a key driver of architectural evolution

150

An ecosystem of other OSs and third parties constantly shapes the architecture evolution



Major Third-Party Subsystems in FreeBSD 11.1

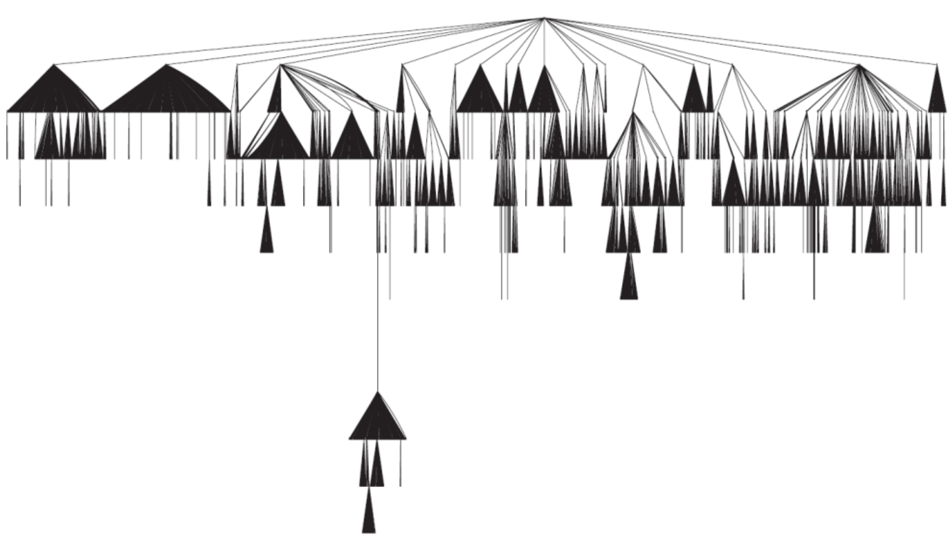
Subsystem	kLoC	LoC %
llvm	3413	10.81
gcc	1576	4.99
binutils	1111	3.52
ntp	873	2.76
heimdal	756	2.39
openssl	661	2.09
subversion	558	1.77
gdb	488	1.54
groff	438	1.39
ofed	404	1.28
libstdc++	394	1.25
wpa	380	1.20
libarchive	310	0.98
sqlite3	281	0.89
ncurses	242	0.77
netbsd-tests	239	0.76
zfs	230	0.73
dtrace	205	0.65
sendmail	205	0.65
unbound	189	0.60
gcclibs	187	0.59
openssh	179	0.57
byacc	150	0.48
libc++	142	0.45
tcpdump	123	0.39
compiler-rt	121	0.38
ldns	115	0.36
tcsh	109	0.34
openbsm	102	0.32
elftoolchain	101	0.32

Major FreeBSD Third-Party Influences

Source	Commits	LoC
TrustedBSD Project	1 215	413 339
NetBSD	1 166	2 665 223
OpenBSD	726	113 195
KAME	451	163 874
Semihalf sp.	330	214 289
DragonflyBSD	179	675 906
Linux	151	109 600
Qualcomm Atheros, Inc.	139	46 608
ABT Systems Ltd	133	8 704
Juniper Networks, Inc.	125	66 971
NetApp, Inc.	120	8 044
Illumos	97	56 618
OpenSolaris	95	125 503
Wheel Systems, Inc.	81	3 552
Yandex LLC	64	3 630
Apple, Inc.	58	13 378

151

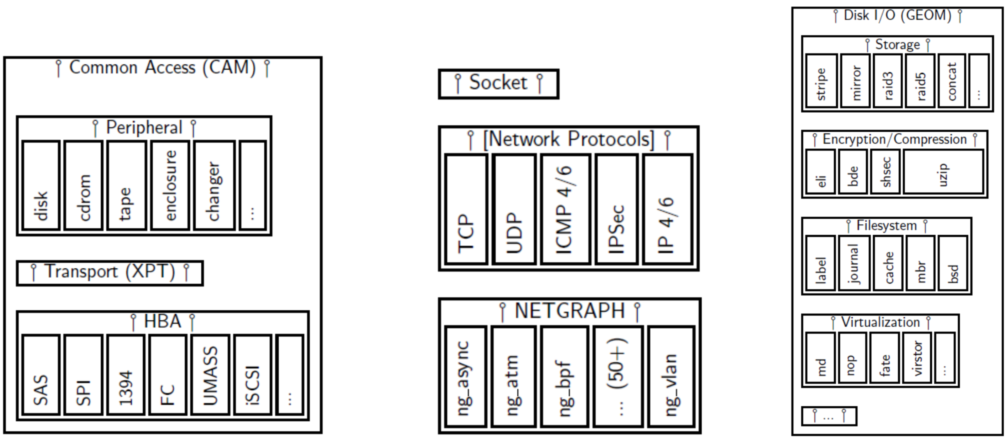
Third-party subsystems facilitate evolution through code reuse, but add technical debt



152

Large subsystems form their own independent architecture

- Netgraph, SSL, MAC, PAM, GEOM, BSM, ZFS, DTrace



153

Thank you!

Diomidis Spinellis and Paris Avgeriou. Evolution of the Unix system architecture: An exploratory case study. *IEEE Transactions on Software Engineering*, 2019. doi:10.1109/TSE.2019.2892149

www.spinellis.gr
@CoolSWEng

154

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?sbcars2020

edX TU Delft

155

Image Credits

- Faces of Open Source / Peter Adams
- Data: Joshua Sortino
- Hackers at Junction 2015: Vmuru
- ASR-33 Teletype: Rama & Musée Bolo
- VT100: Jason Scott
- PDP 11/20: Image courtesy of Computer History Museum

(Creative commons licenses)

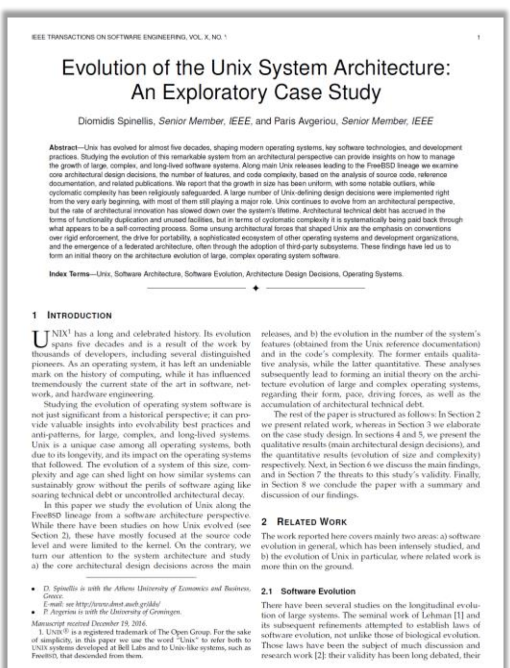
- PDP11/40: Stefan_Kögl, CC BY-SA 3.0
- Digital VAX 11/780: Emiliano Russo, PD
- Numbers: Nick Hillier
- Building construction: chuttersnap
- Technical Debt: Jacob Duck Die Pfändung
- Twisted skyscraper: Mitya Ivanov
- SPARCstation, Mike Chapman, PD

156

Funding Credit

The research described has been partially carried out as part of the CROSSMINER Project, which has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 732223.

157



IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. X, NO. 1 1

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.

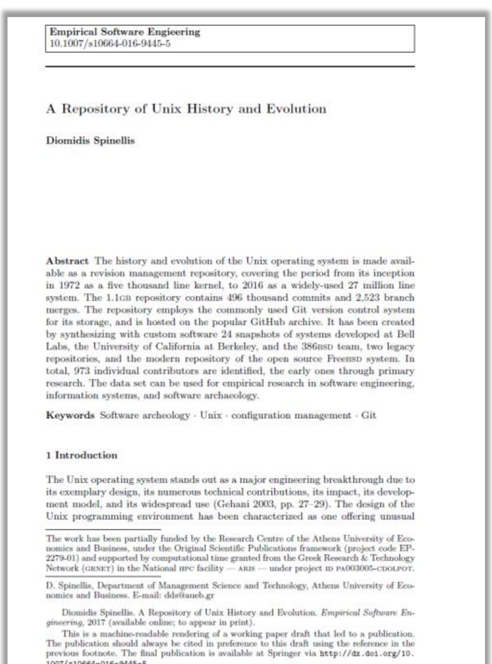
Diomidis Spinellis and Paris Avgeriou. Evolution of the Unix system architecture: An exploratory case study. *IEEE Transactions on Software Engineering*, 2019. doi:10.1109/TSE.2019.2892149

158

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
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/
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

159



Empirical Software Engieering
10.1007/s10664-016-9445-5

A Repository of Unix History and Evolution

Diomidis Spinellis

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 environment has been characterized as one offering unusual

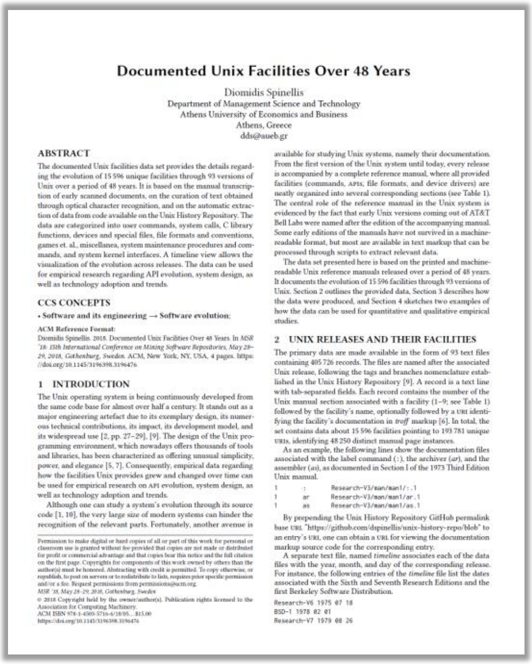
The work has been partially funded by the Research Centre of the Athens University of Economics and Business, under the Original Scientific Publications framework (project code EP-2279-01) and supported by computational time granted from the Greek Research & Technology Network (GRNET) in the National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

D. Spinellis, Department of Management Science and Technology, Athens University of Economics and Business. E-mail: dds@aueb.gr

Diomidis Spinellis. A Repository of Unix History and Evolution. *Empirical Software Engineering*, 2017 (available online; to appear in print).

This is a machine-readable rendering of a working paper draft that led to a publication. The publication should always be cited in preference to this draft using the reference in the previous footnote. The final publication is available at Springer via http://dx.doi.org/10.1007/s10664-016-9445-5.

160



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
```

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MSR '18, May 28–29, 2018, Gothenburg, Sweden
© 2018 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-5716-6/18/05...$15.00
https://doi.org/10.1145/3196398.3196476

161