Lessons from Unix History

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

𝕏 @CoolSWEng

@CoolSWEng@mastodon.acm.org

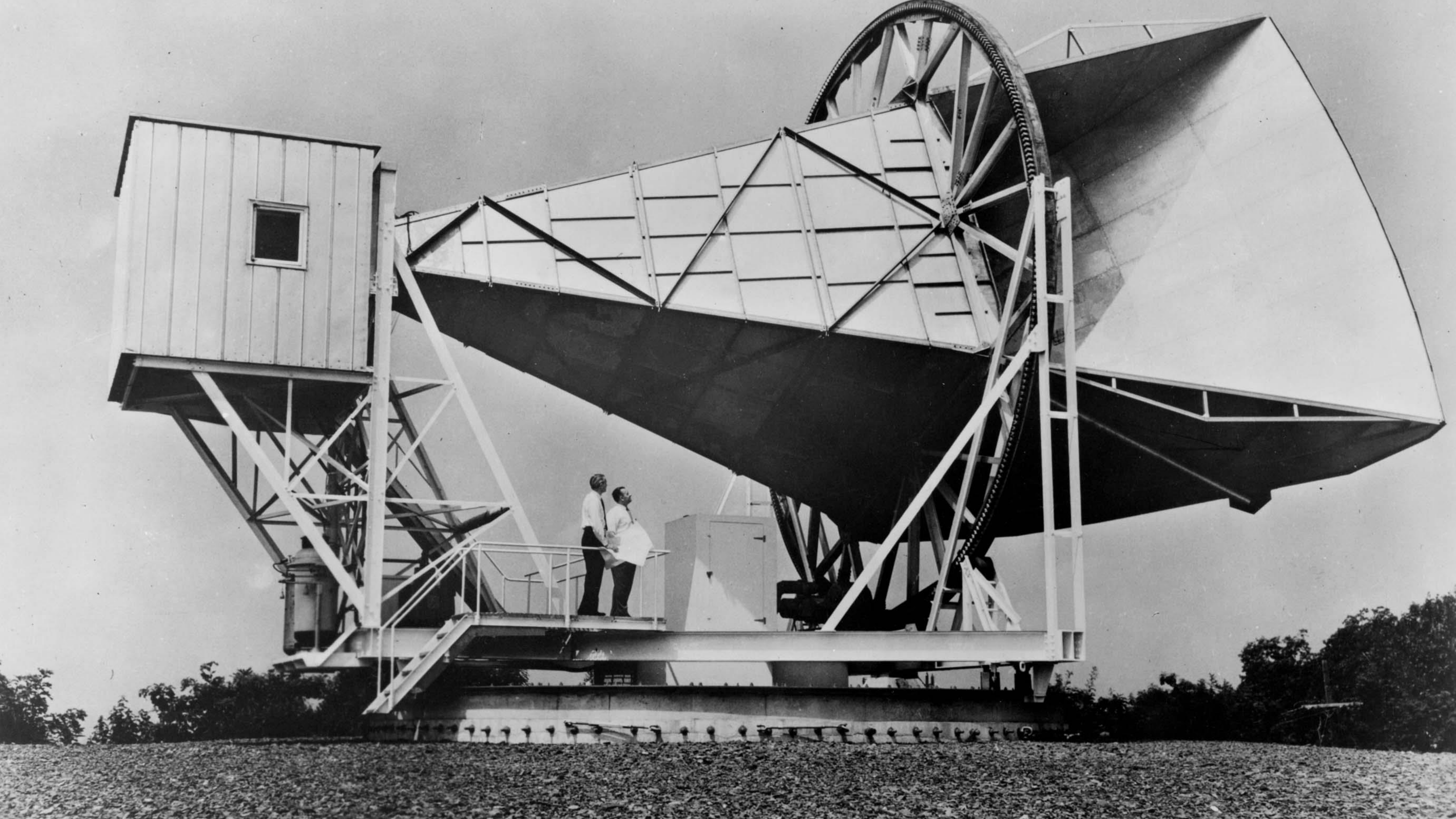
Unix and Linux are powering modern IT

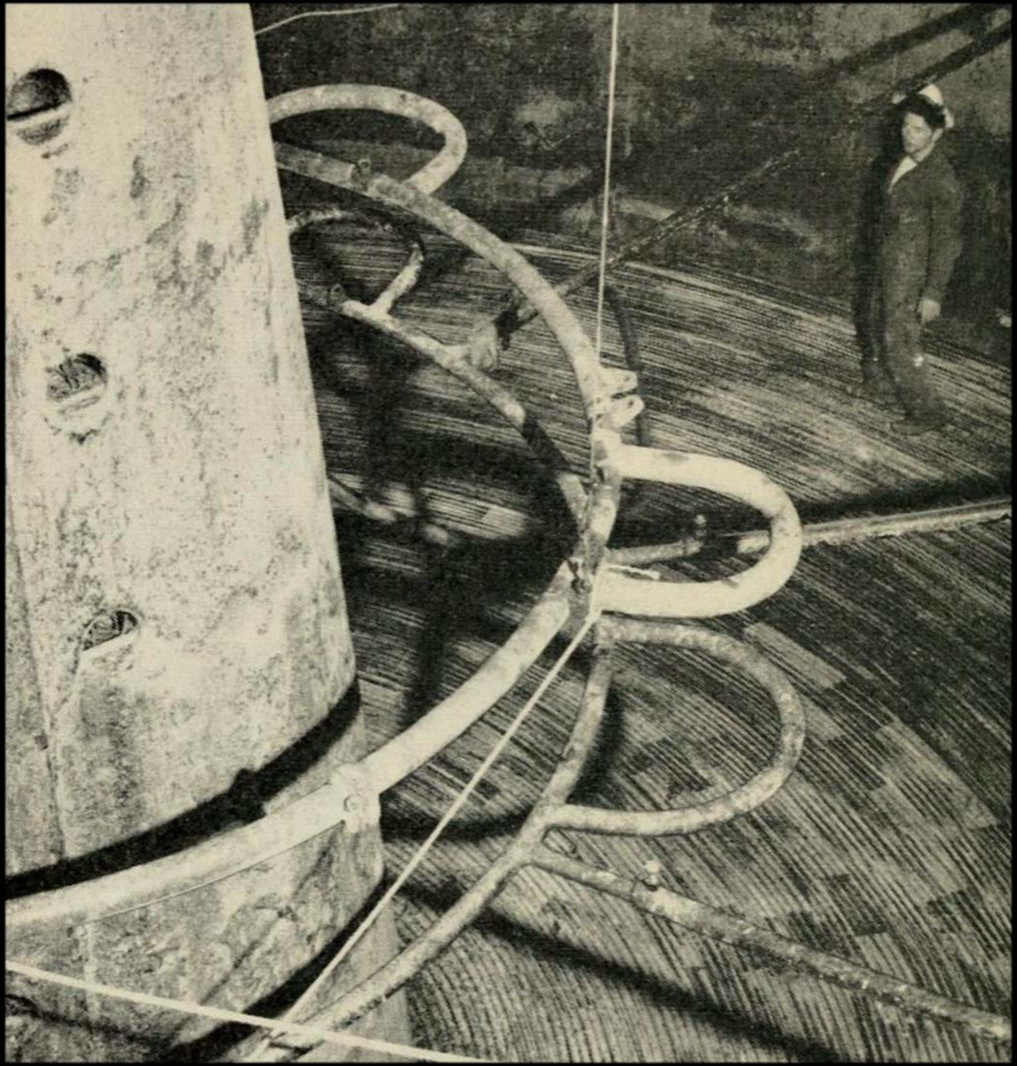
- Android (1 bn shipments / year)
- Google, Facebook, Amazon, X technology stacks
- macOS
- 60% of the top one million web servers,
- 75% of major cloud providers' instances,
- 97% of embedded systems,
- All of the world's top-500 supercomputers

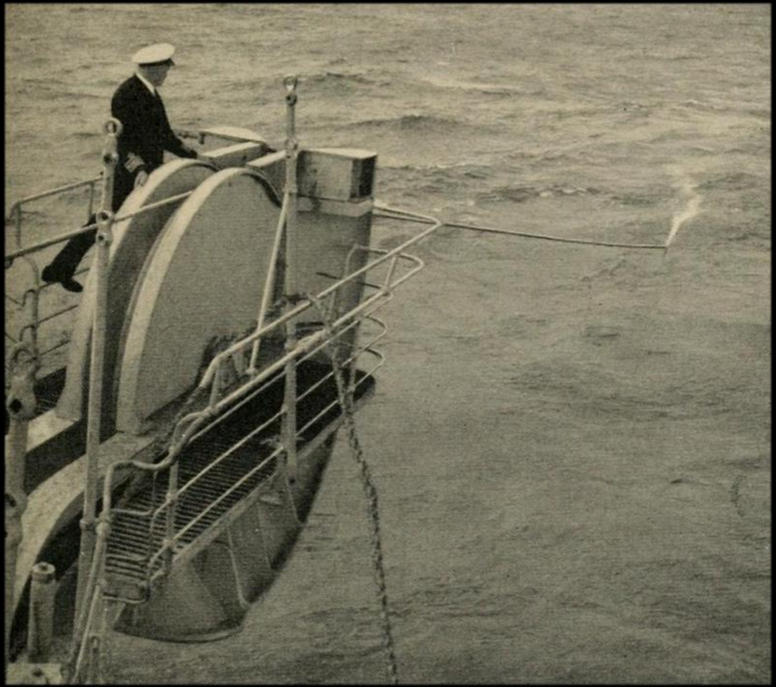
History

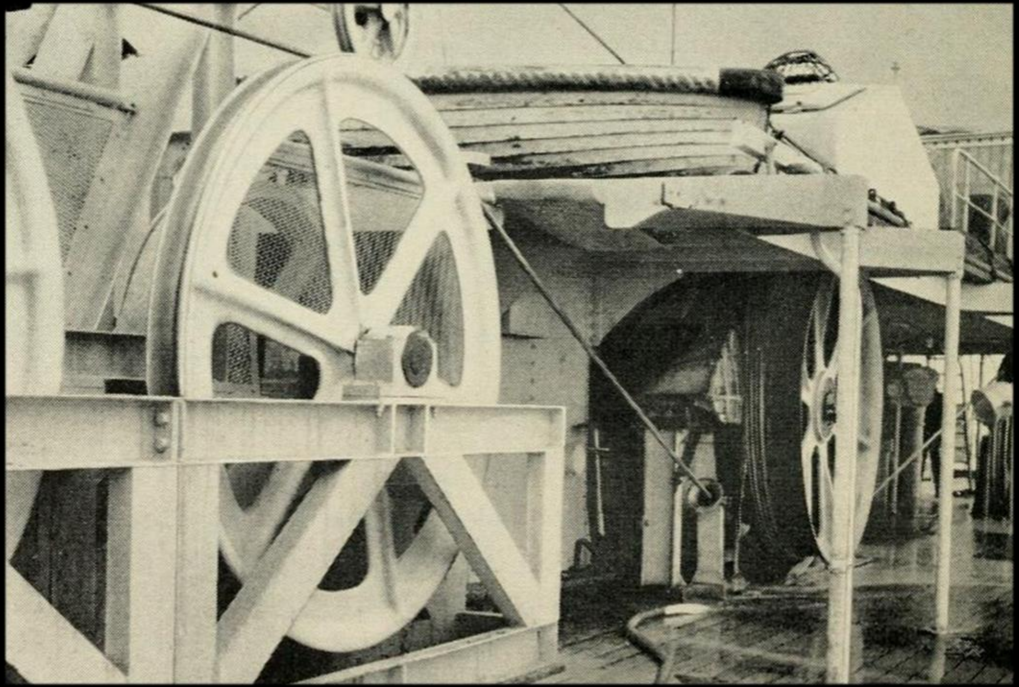
ALFR·
NOBEL
NAT·
MDCCC
XXXIII
OB·
MDCCC
XCVI

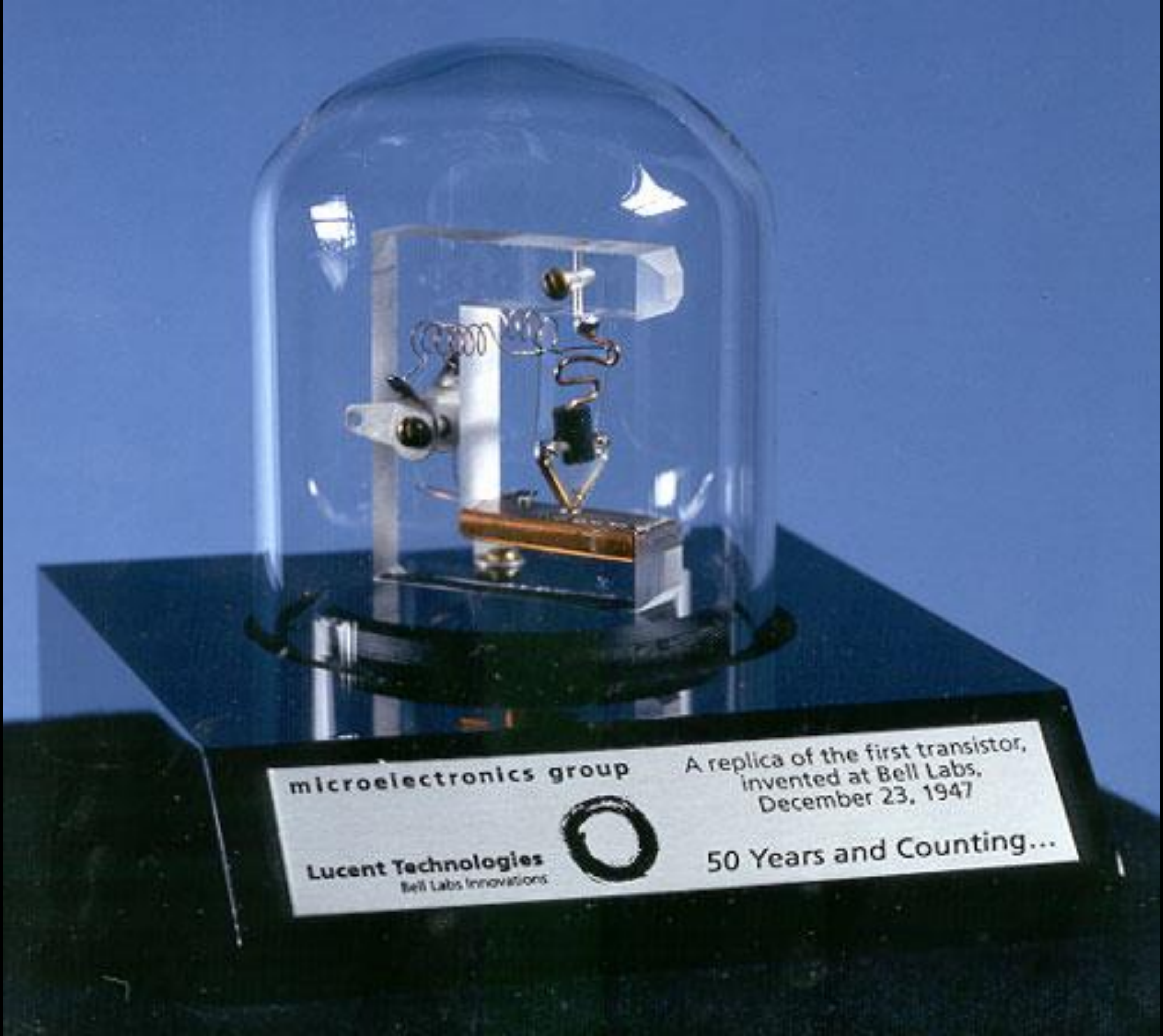
ACM
A.M. TURING AWARD

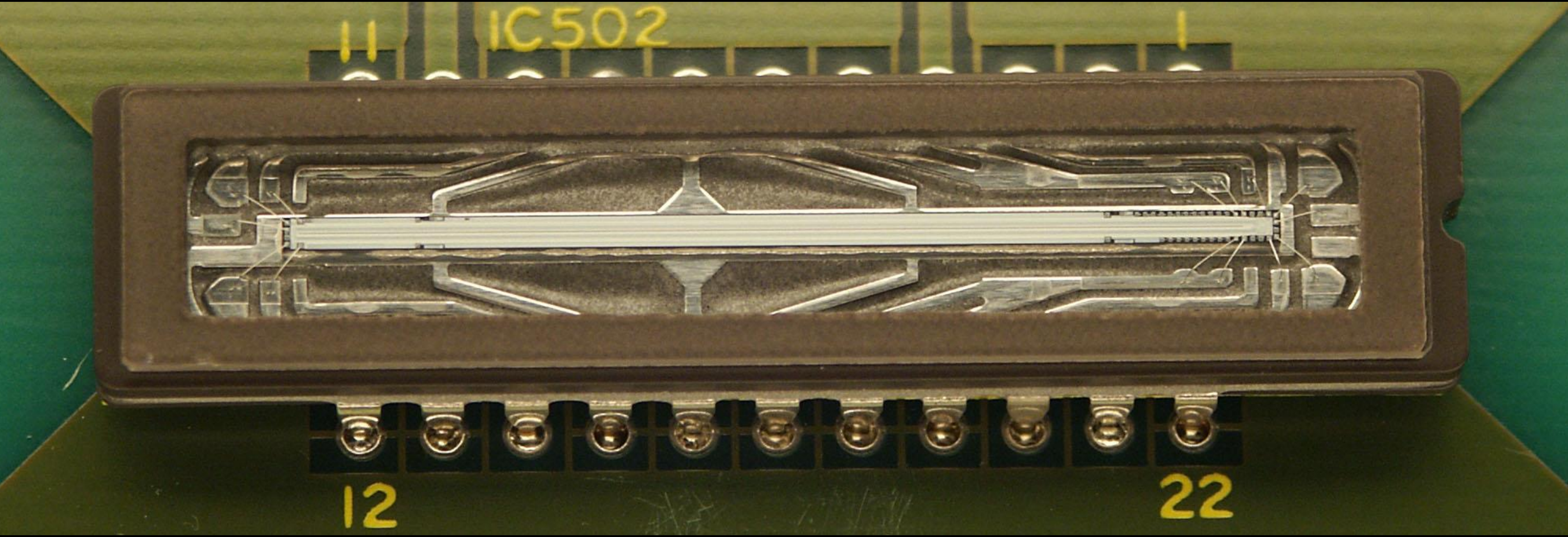









microelectronics group
A replica of the first transistor,
invented at Bell Labs,
December 23, 1947
50 Years and Counting...
Lucent Technologies
Bell Labs Innovations


11
IC502
1
12
22

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.

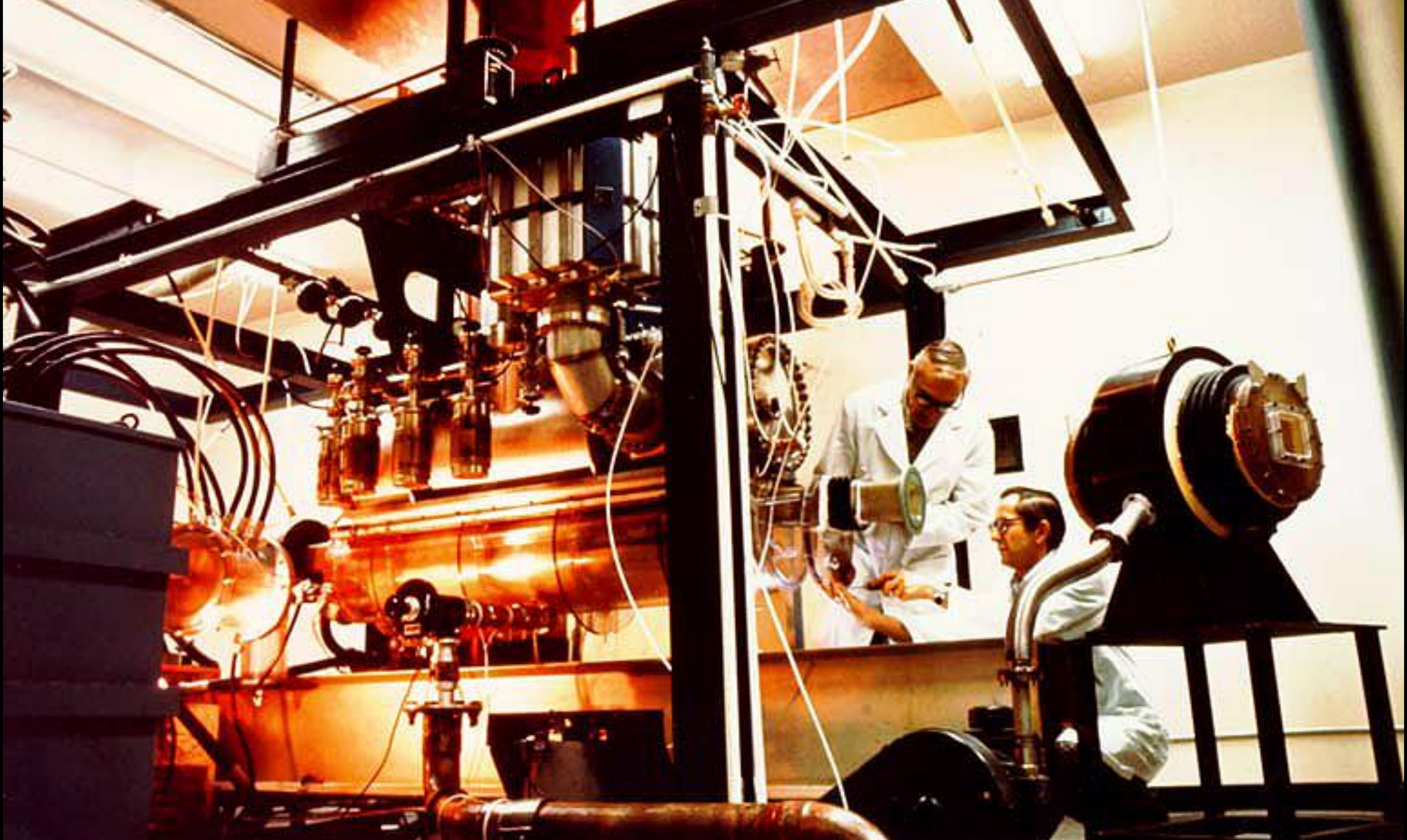
3. It is mathematically more suitable. Many of the limiting operations are simple in terms of the logarithm but would require clumsy restatement in terms of the number of possibilities.

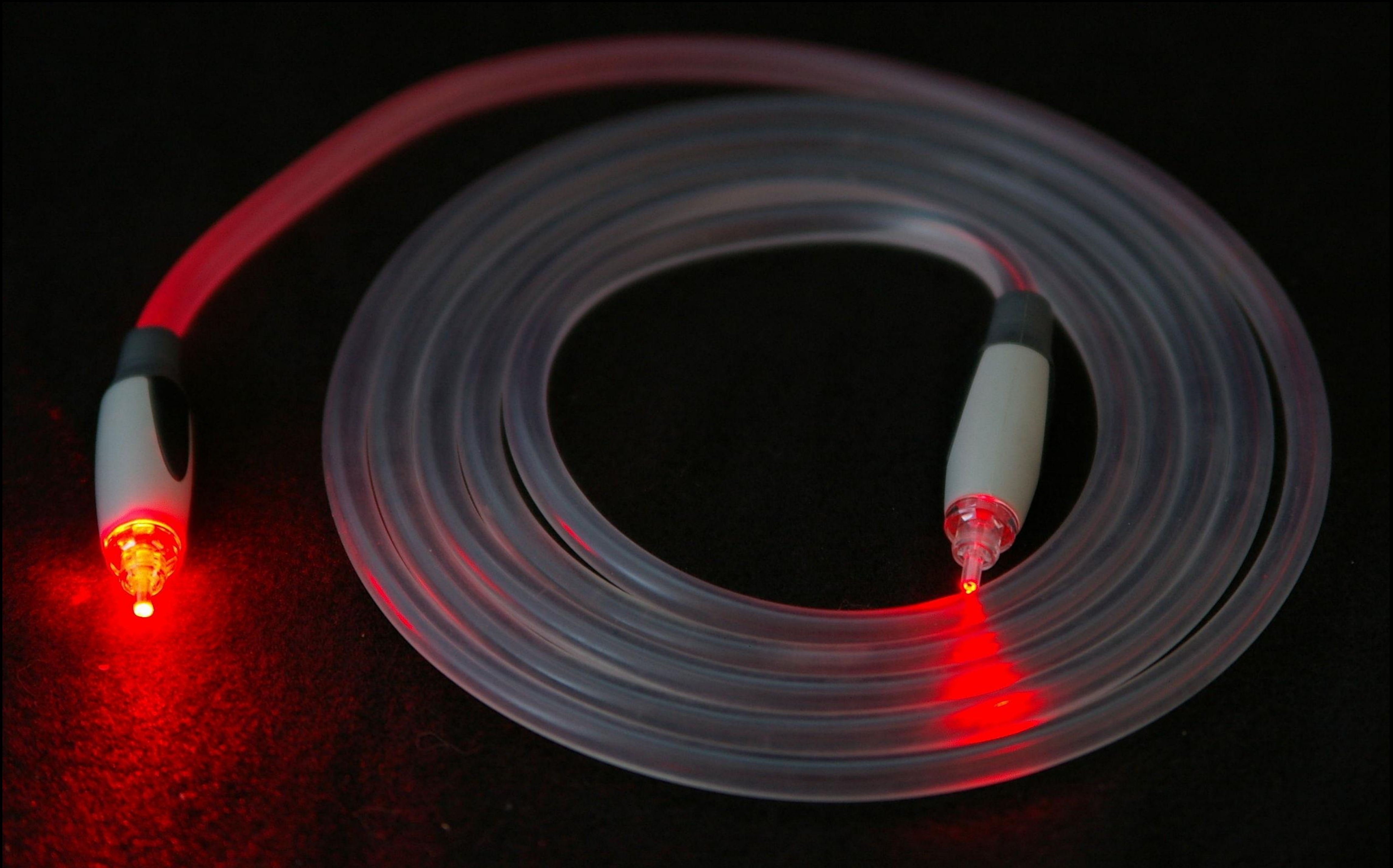
The choice of a logarithmic base corresponds to the choice of a unit for measuring information. If the base 2 is used the resulting units may be called binary digits, or more briefly *bits,* a word suggested by J. W. Tukey. A device with two stable positions, such as a relay or a flip-flop circuit, can 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

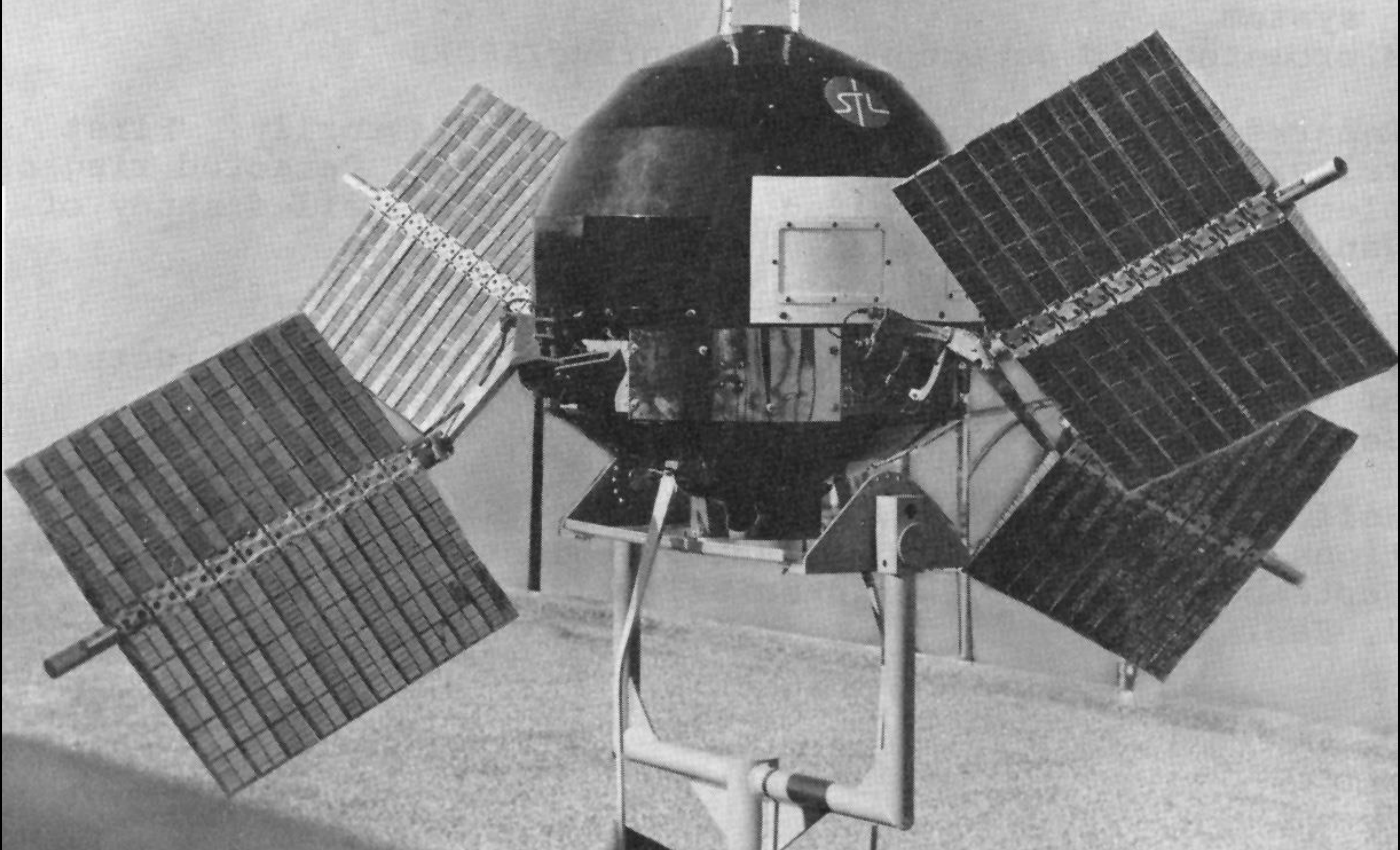
$$\begin{aligned}\log_2 M &= \log_{10} M / \log_{10} 2 \\ &= 3.32 \log_{10} M,\end{aligned}$$

[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.

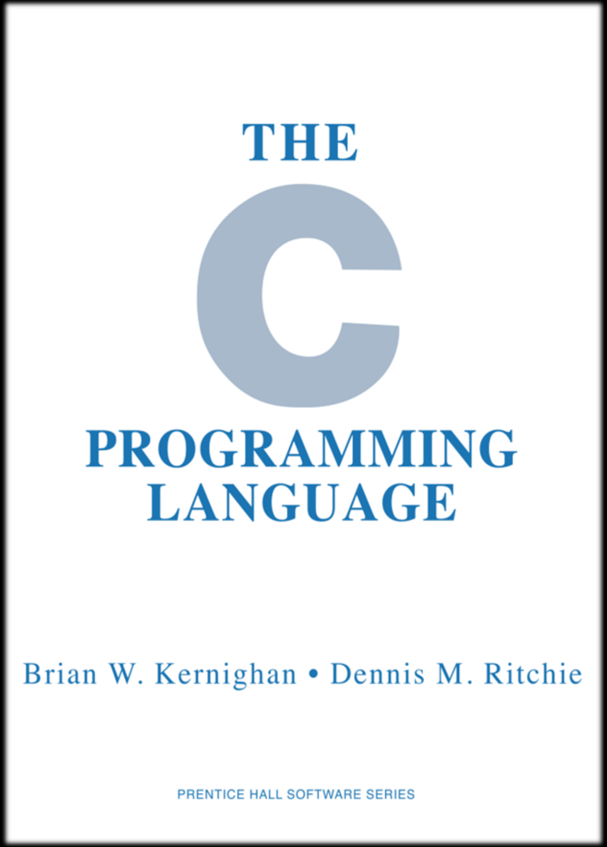
[2] Hartley, R. V. L., "Transmission of Information," *Bell System Technical Journal,* July 1928, p. 535.






STL

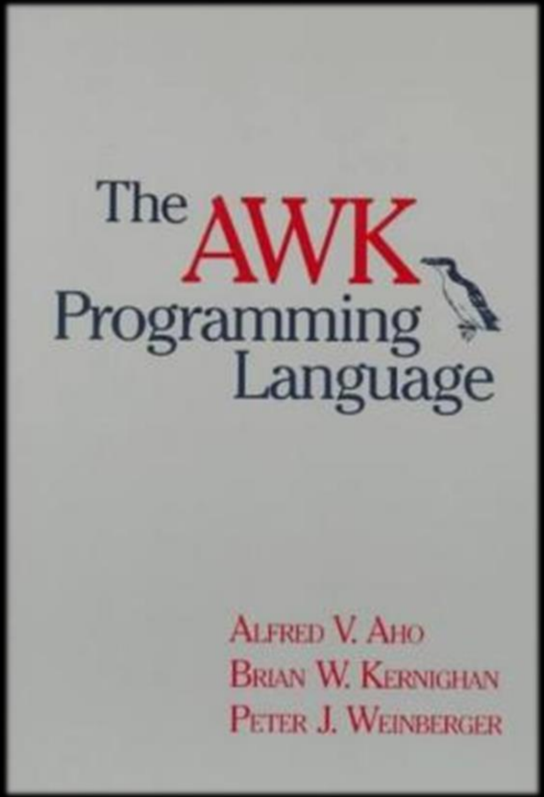
MOTOROLA
NOKIA
NOKIA
22:18
08-01-2006
NOKIA
Insert SIM
ERICSSON
Insert SIM
SAMSUNG



THE C PROGRAMMING LANGUAGE

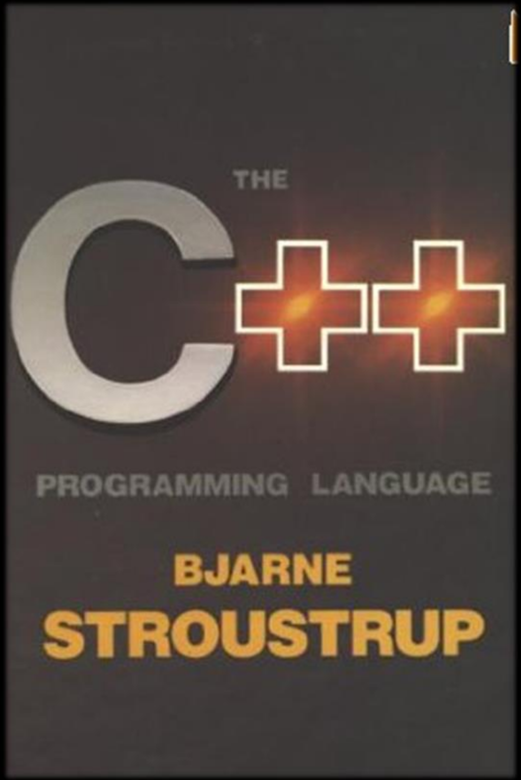
Brian W. Kernighan • Dennis M. Ritchie

PRENTICE HALL SOFTWARE SERIES



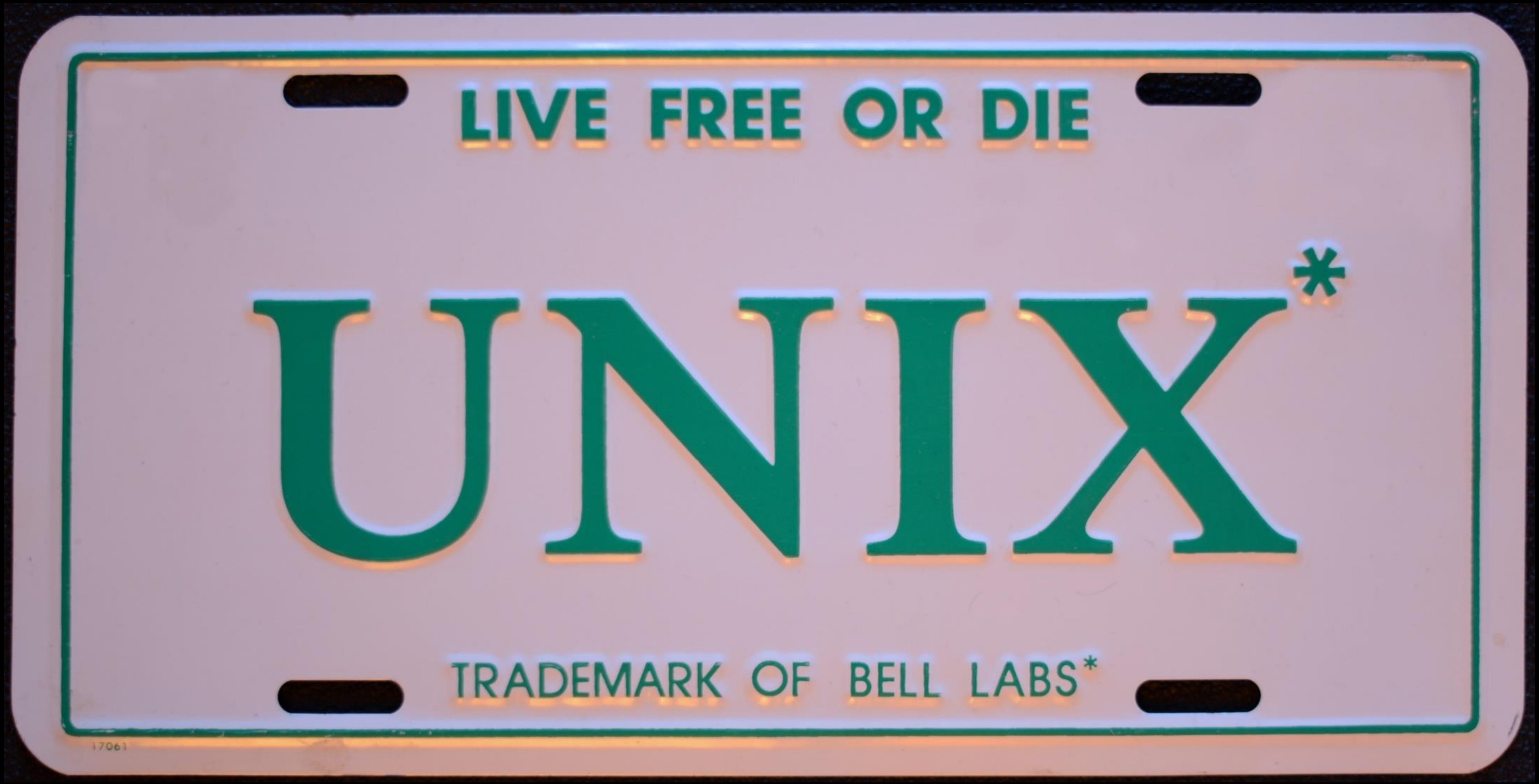
The AWK Programming Language

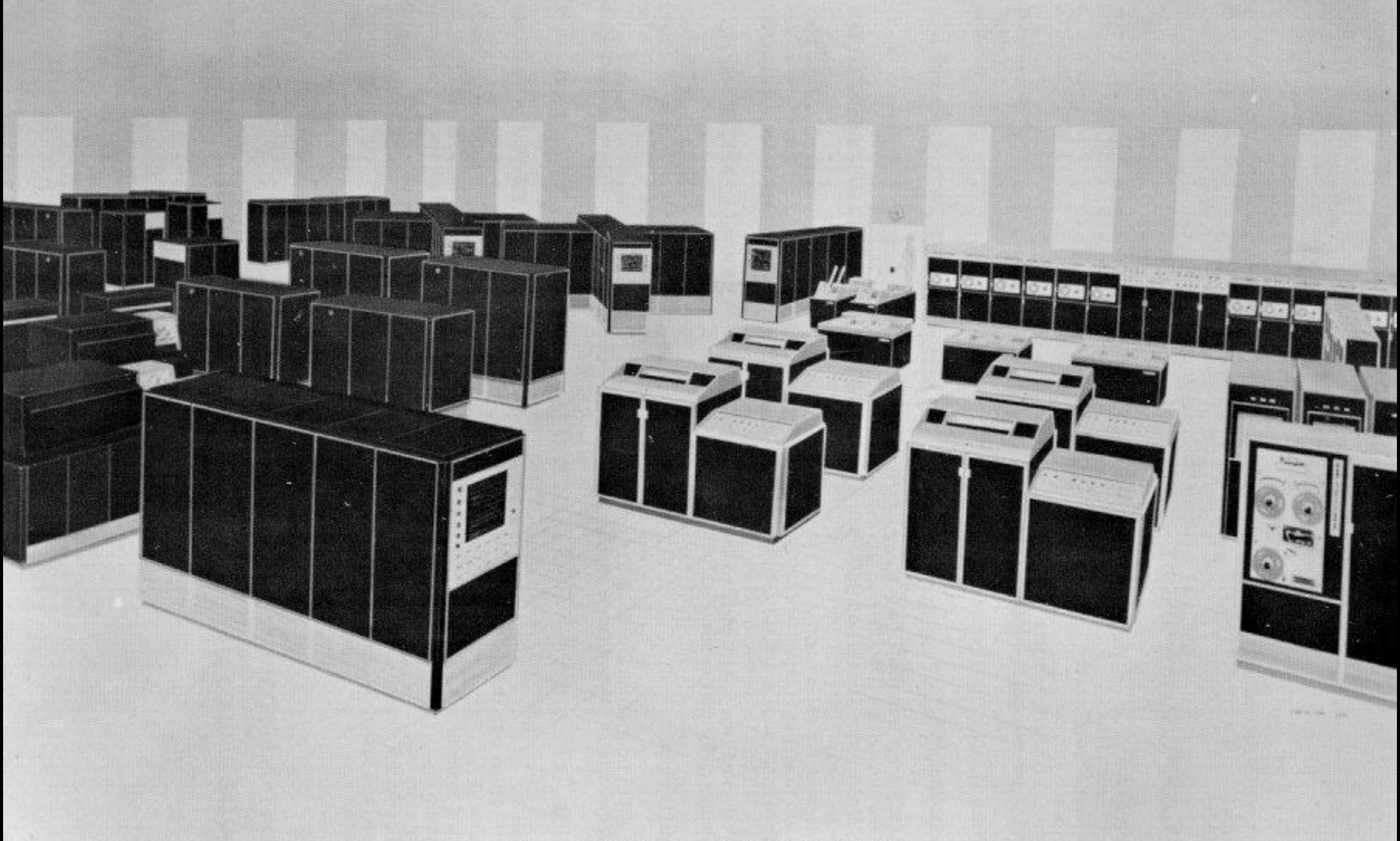
Alfred V. Aho
Brian W. Kernighan
Peter J. Weinberger



THE C++ PROGRAMMING LANGUAGE

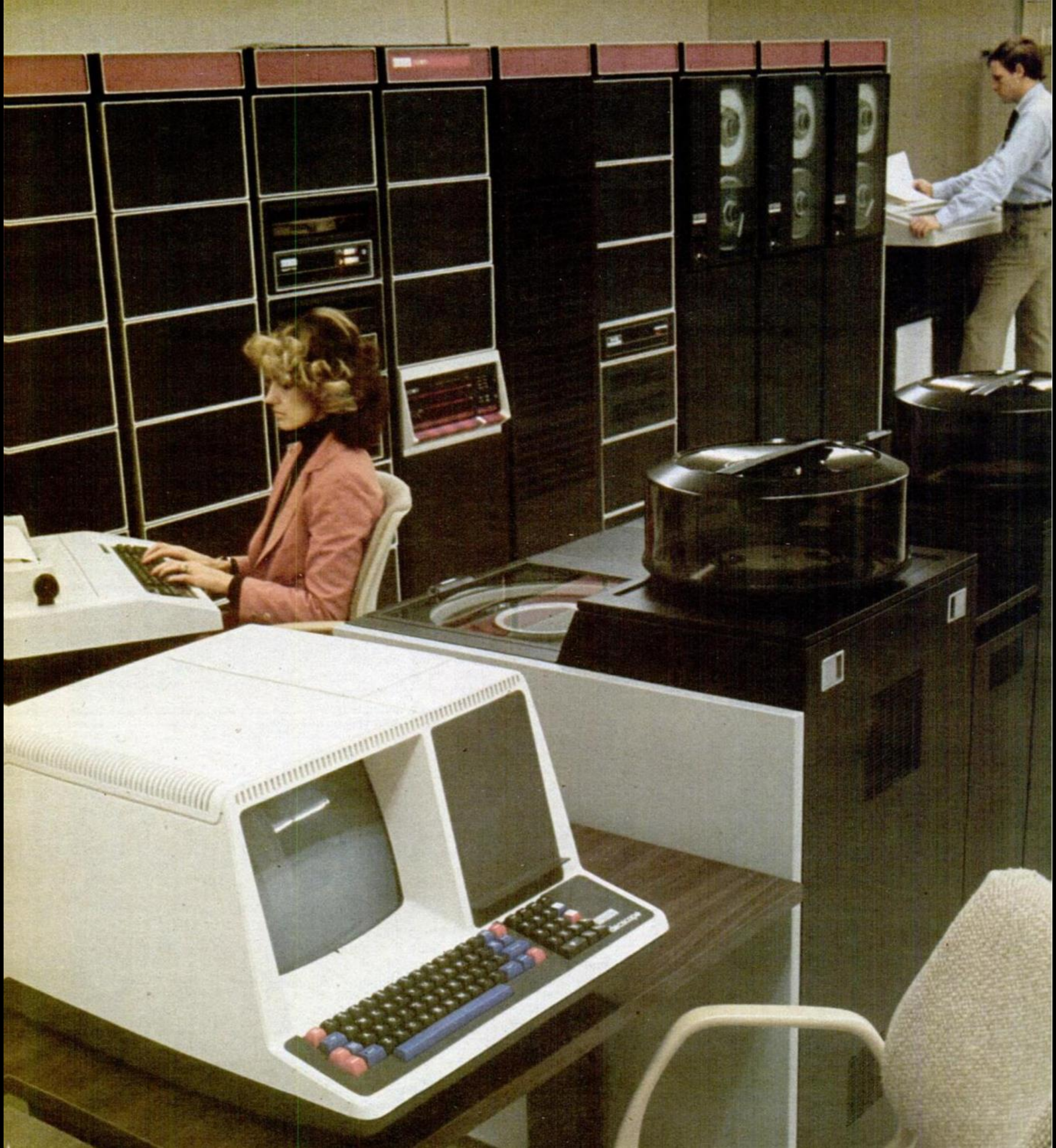
BJARNE STROUSTRUP


LIVE FREE OR DIE
UNIX*
TRADEMARK OF BELL LABS*
17061



GE-645 SYSTEM

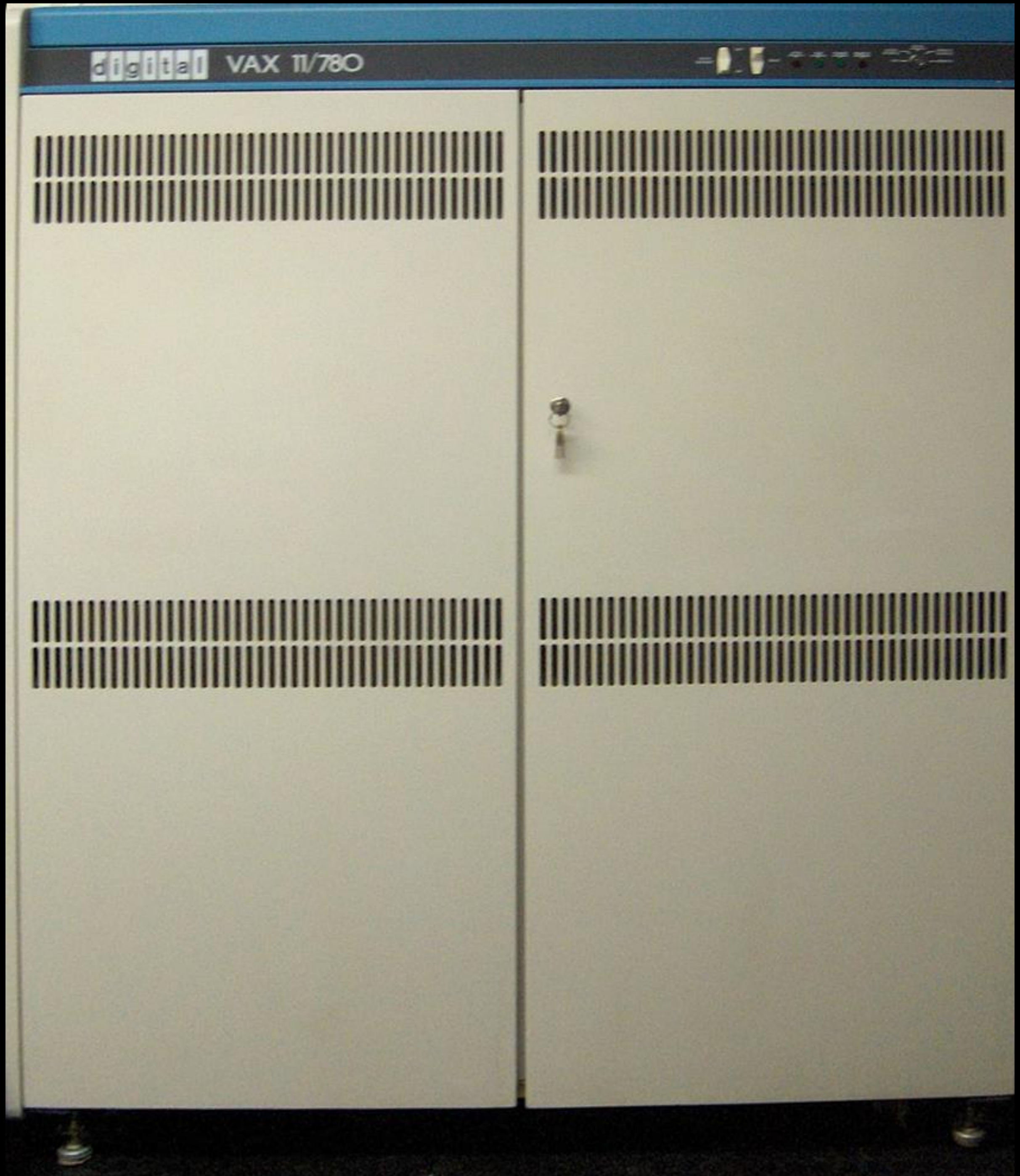
JANUS 71



digital

Faces of Open Source / Peter Adams

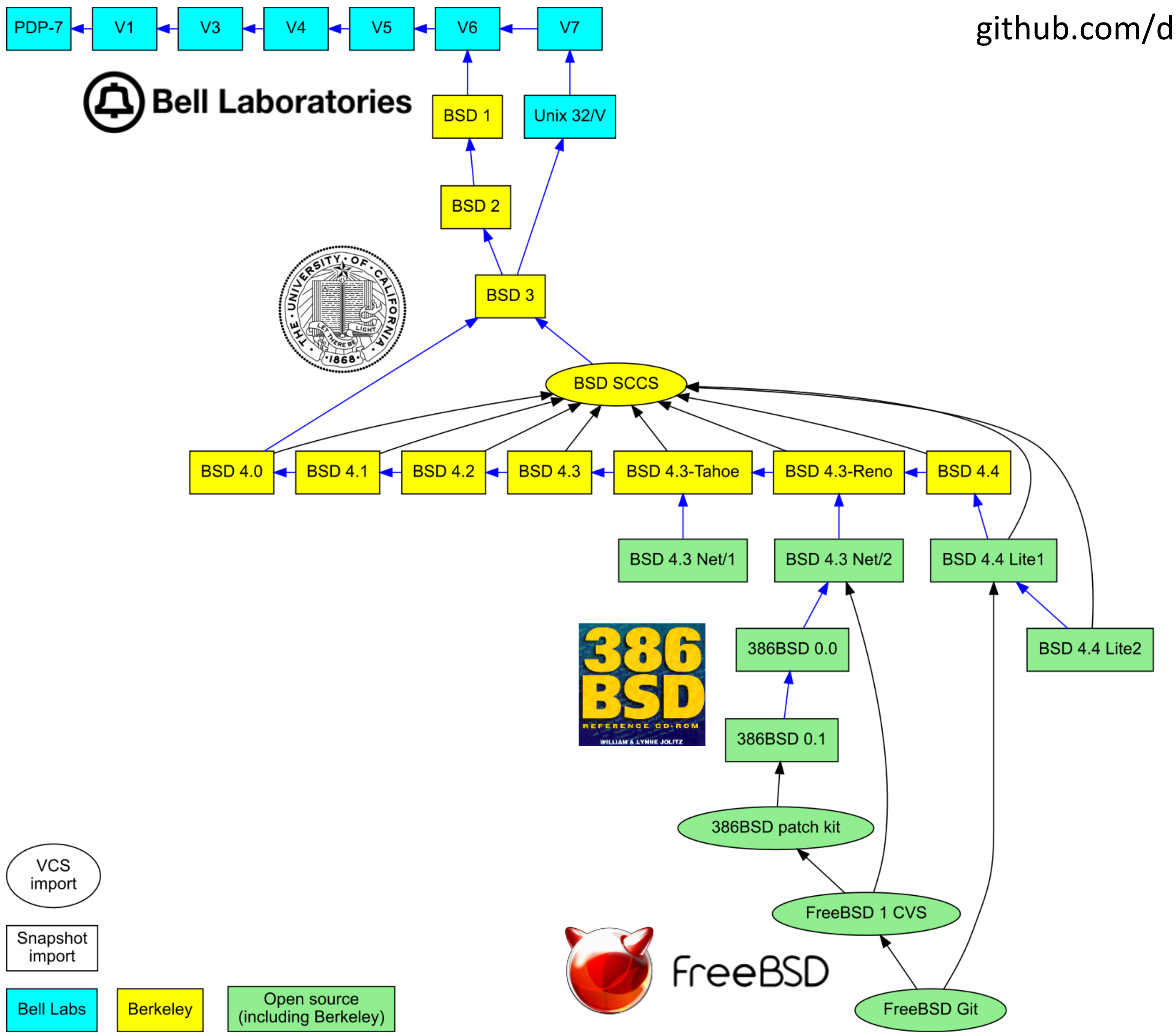
VT100


digital
VAX 11/780

MLH
R

Data Sources

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



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

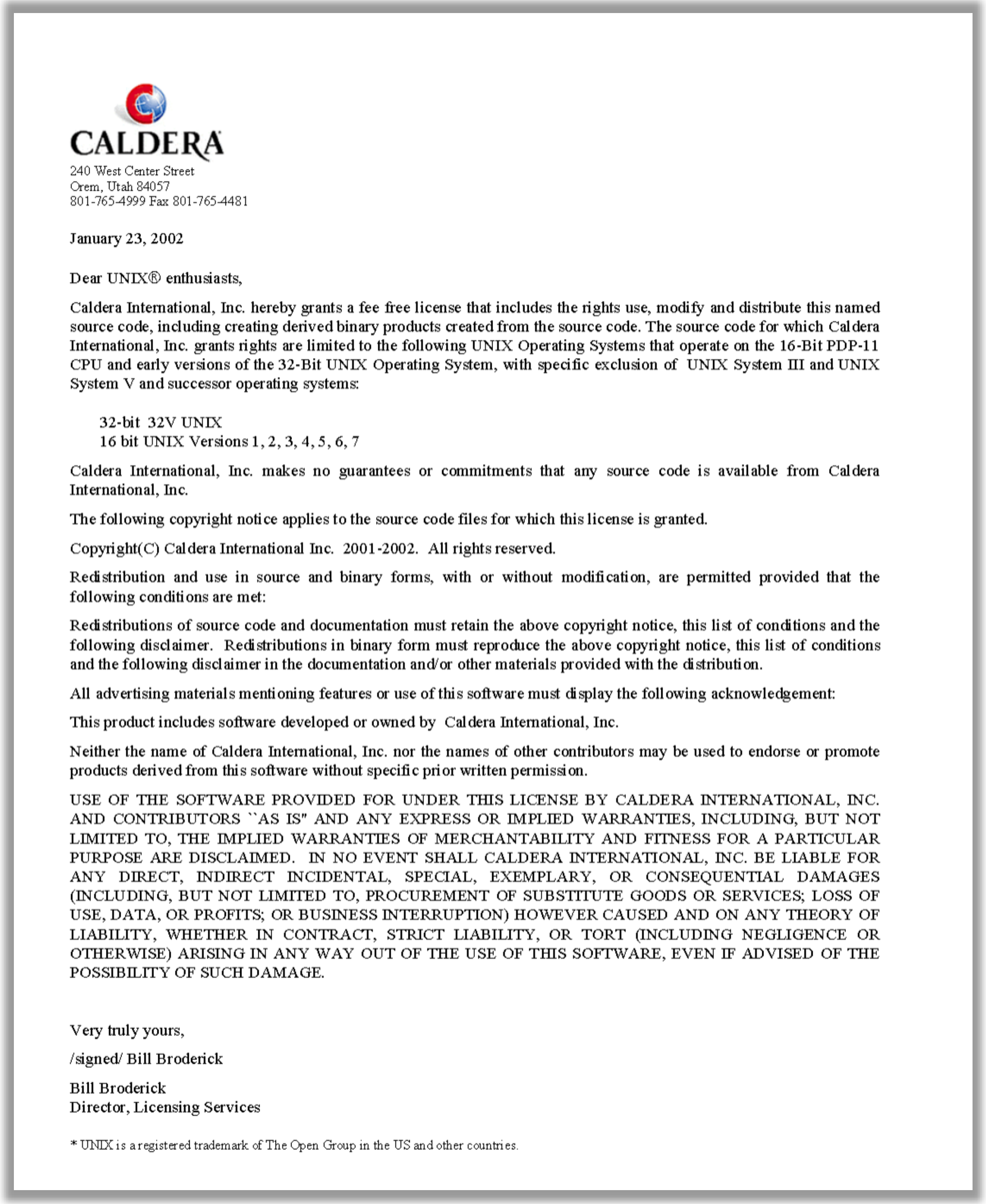
VCS import

Snapshot import

Bell Labs

Berkeley

Open source (including Berkeley)



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.

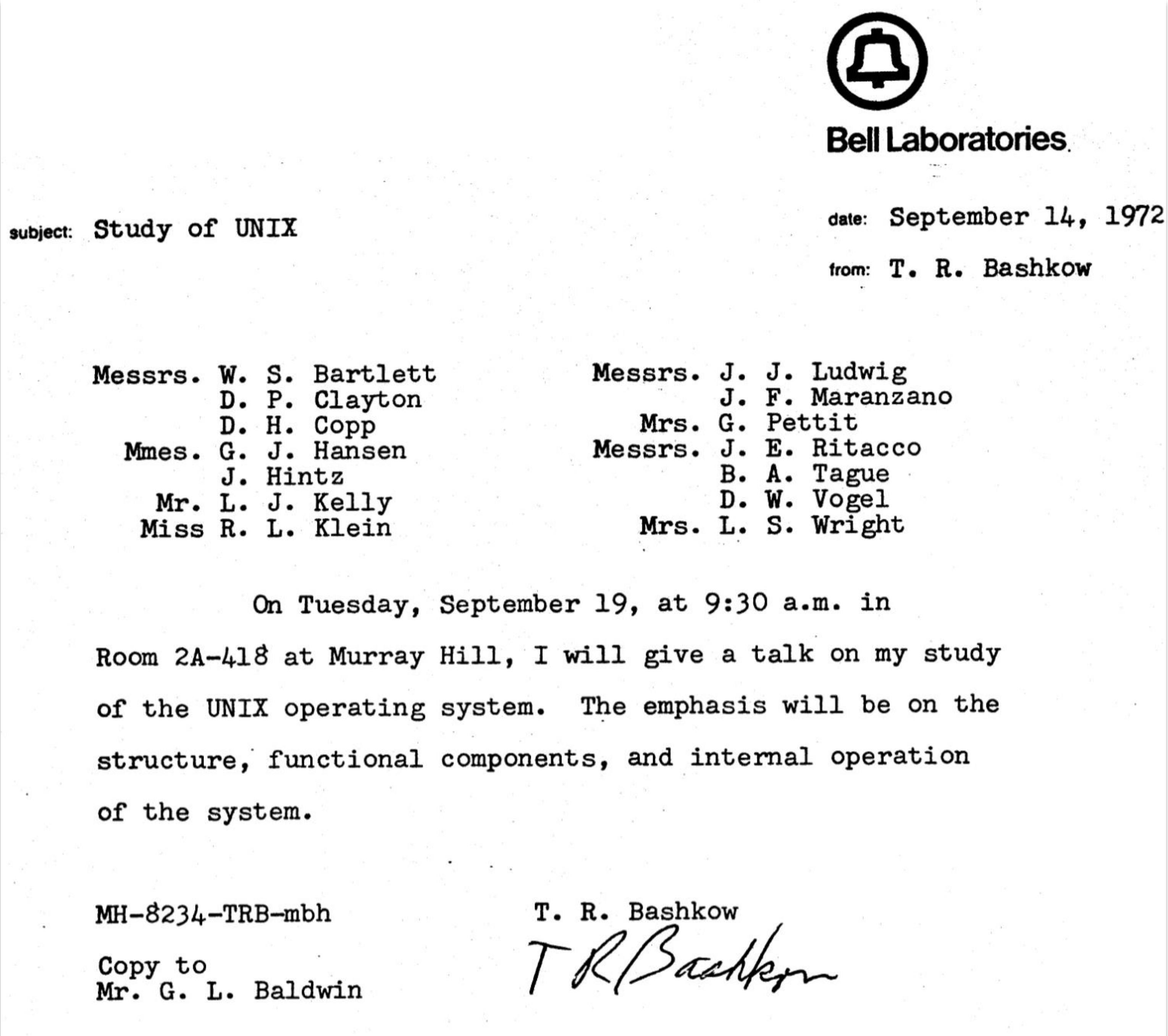
The following copyright notice applies to the source code files for which this license is granted.

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.



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

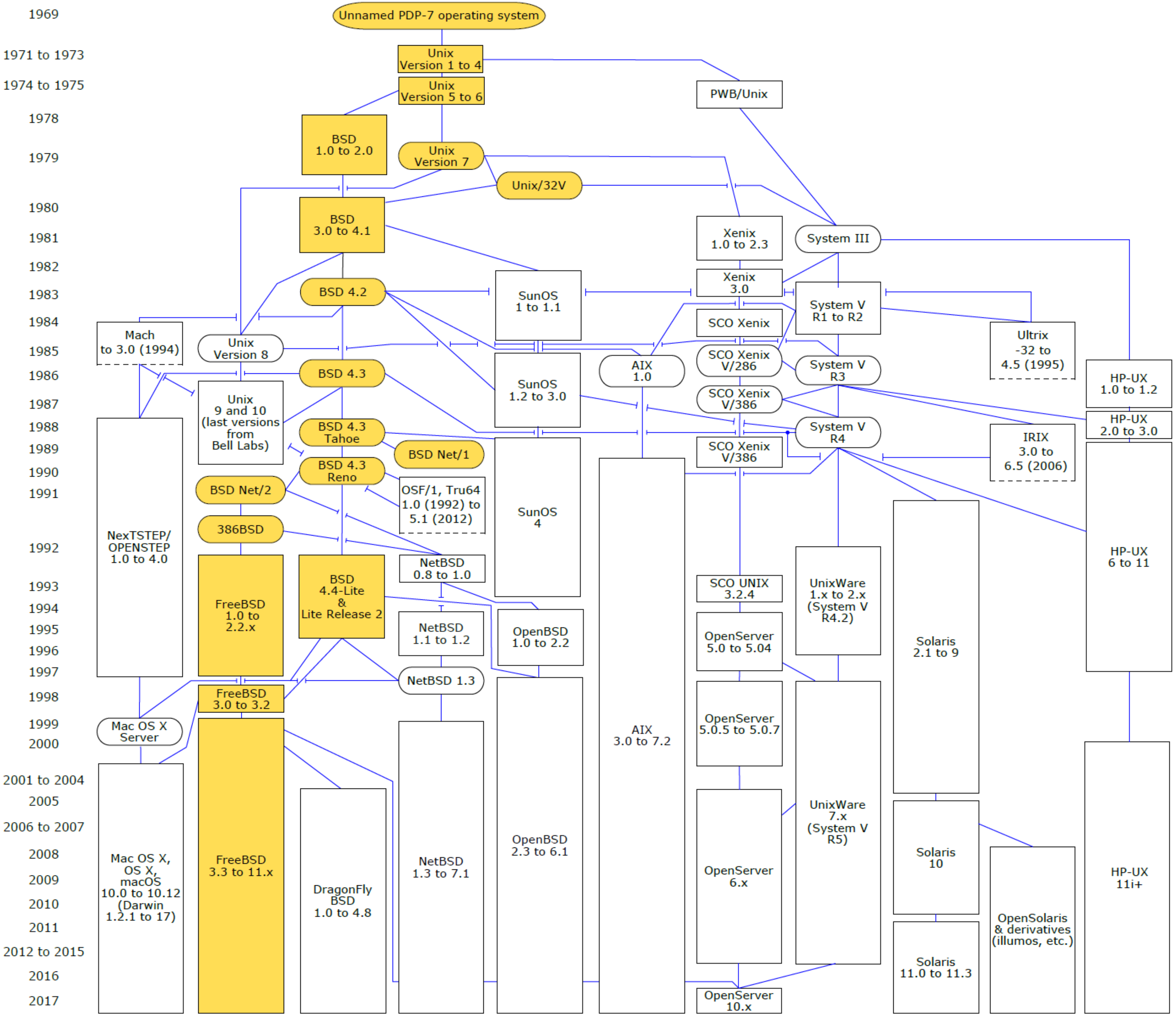
T. R. Bashkow

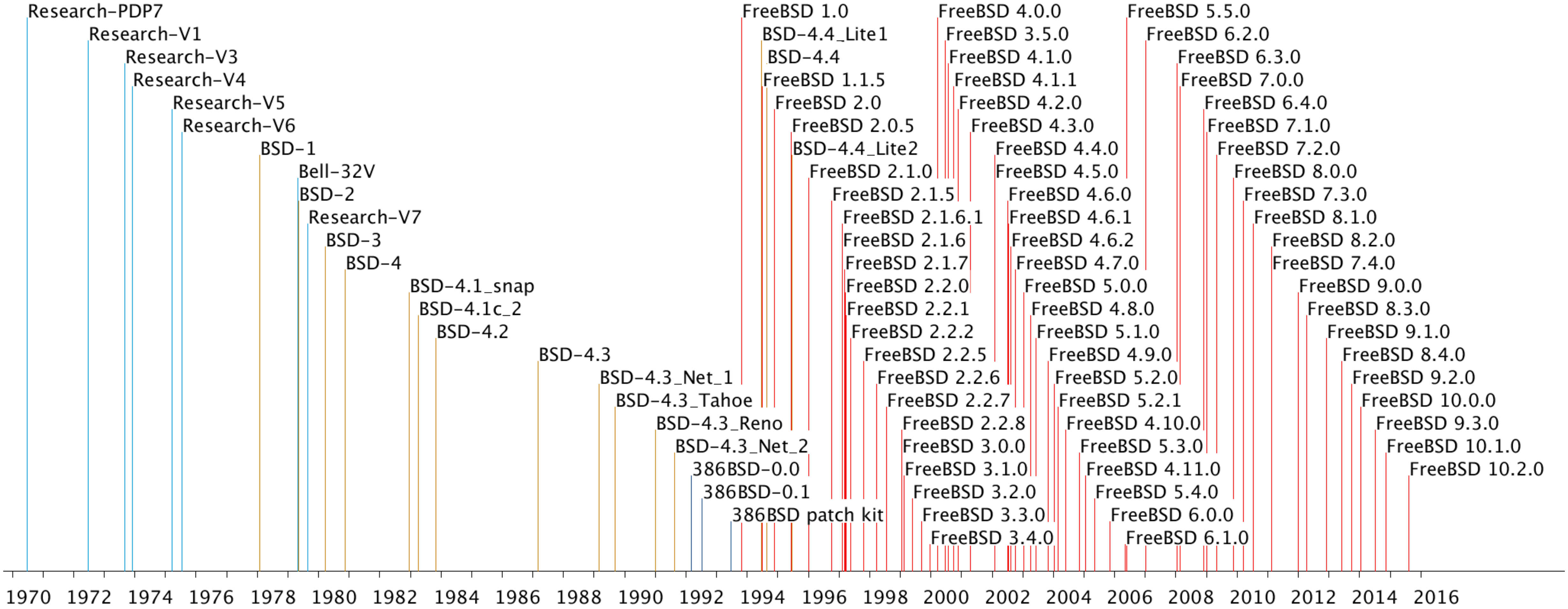
T R Bashkow

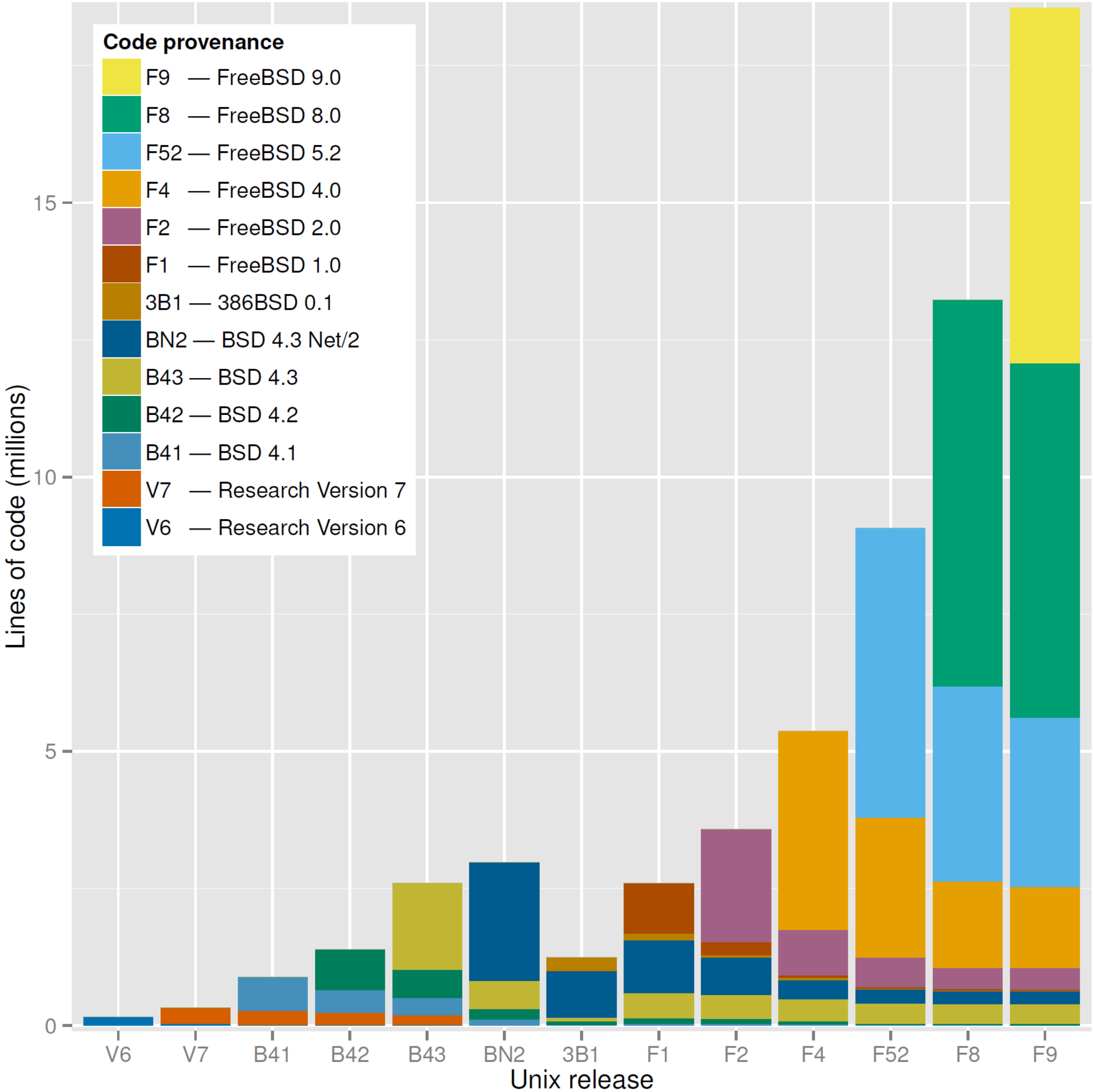
Copy to
Mr. G. L. Baldwin

```
/ 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.
/ 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
```

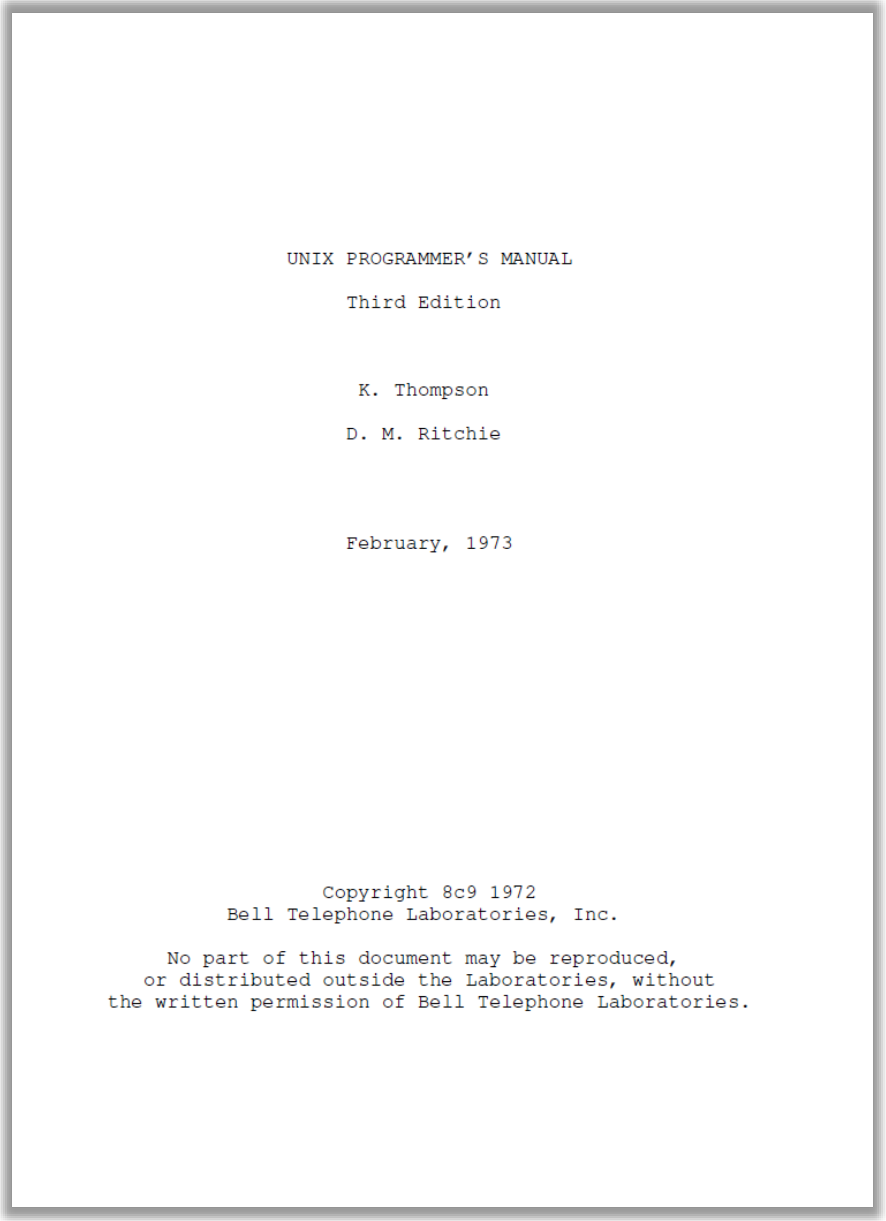
```
        mov     $idata,r0 / r0=base addr. of assembled directories.
        mov     $u.off,u.fofp / pointer to u.off in u.fofp (holds file
                              / offset)
1:
        mov     (r0)+,r1/r1=41.,...,47; "0" in the assembled directory
                          / header signals last
        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:                                                 (on cold boot)
        2f;0         —this is file #47 listed on E0,9  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
```


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
HP-UX 1.0 to 1.2
BSD 4.3
SunOS 1.2 to 3.0
Unix 9 and 10 (last versions from Bell Labs)
SCO Xenix V/386
HP-UX 2.0 to 3.0
BSD 4.3 Tahoe
System V R4
IRIX 3.0 to 6.5 (2006)
BSD Net/1
SCO Xenix V/386
BSD 4.3 Reno
BSD Net/2
OSF/1, Tru64 1.0 (1992) to 5.1 (2012)
SunOS 4
NexTSTEP/ OPENSTEP 1.0 to 4.0
386BSD
NetBSD 0.8 to 1.0
HP-UX 6 to 11
FreeBSD 1.0 to 2.2.x
BSD 4.4-Lite & Lite Release 2
SCO UNIX 3.2.4
UnixWare 1.x to 2.x (System V R4.2)
Solaris 2.1 to 9
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
NetBSD 1.3
FreeBSD 3.0 to 3.2
AIX 3.0 to 7.2
OpenServer 5.0.5 to 5.0.7
Mac OS X Server
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
OpenServer 6.x
UnixWare 7.x (System V R5)
Solaris 10
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
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016


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

```
$ 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) }
```



UNIX PROGRAMMER'S MANUAL

Third Edition

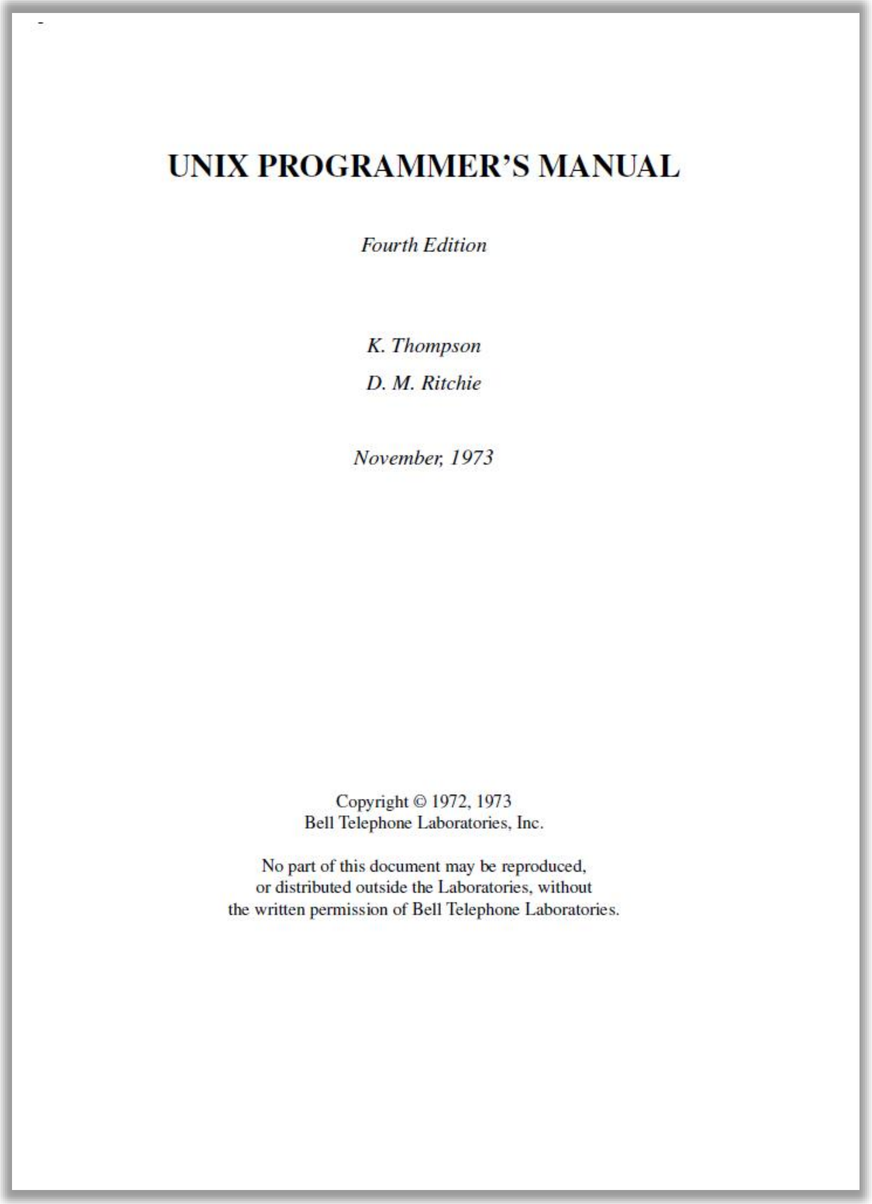
K. Thompson

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

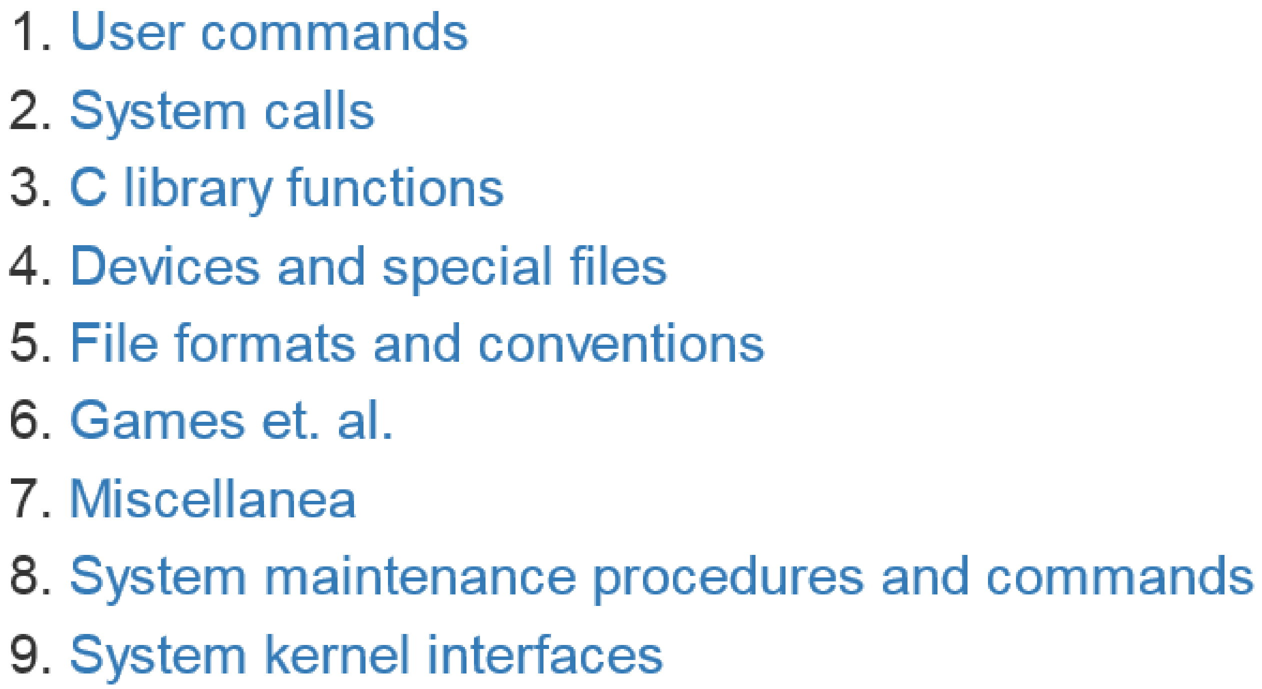
November, 1973

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.

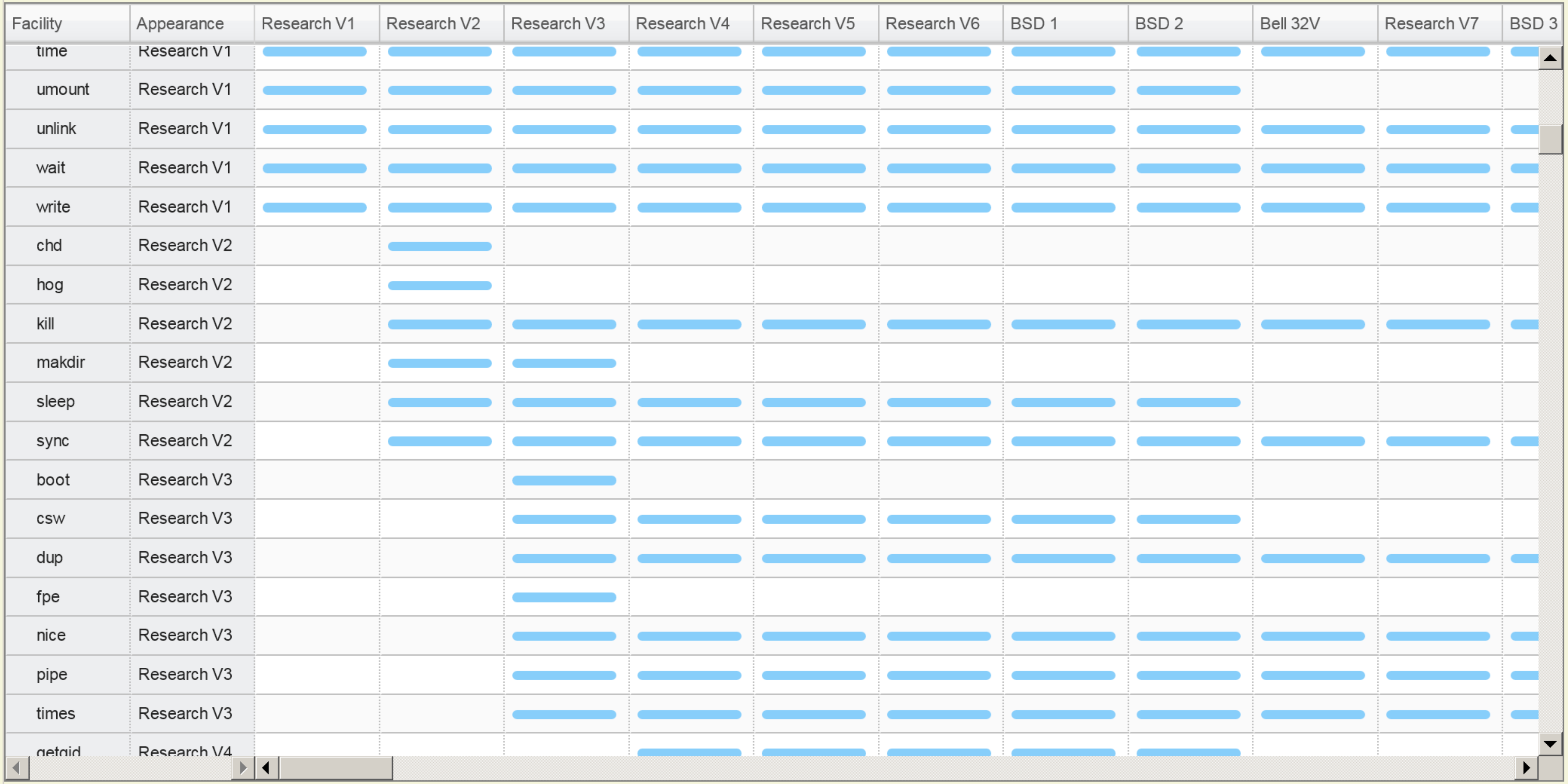
dspinellis.github.io/unix-history-man

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

Evolution of Unix section 2: System calls

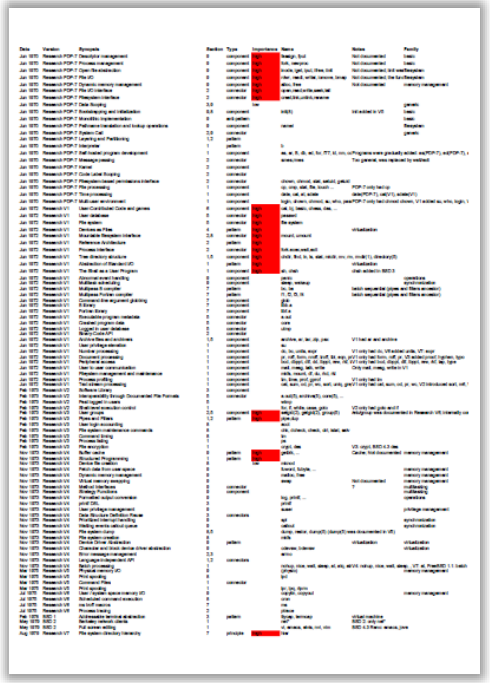
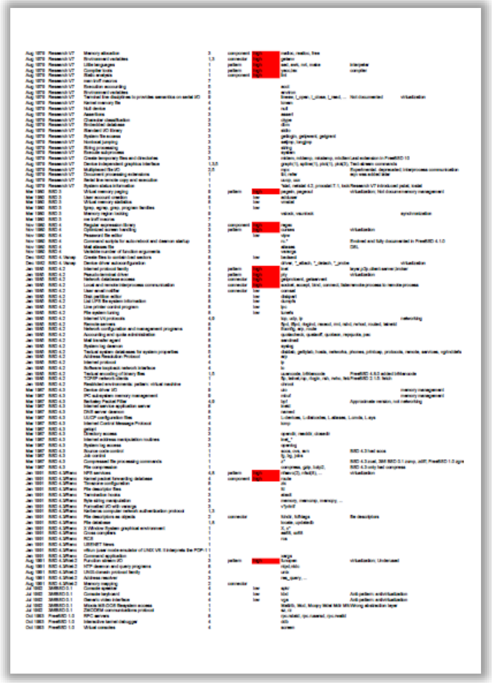
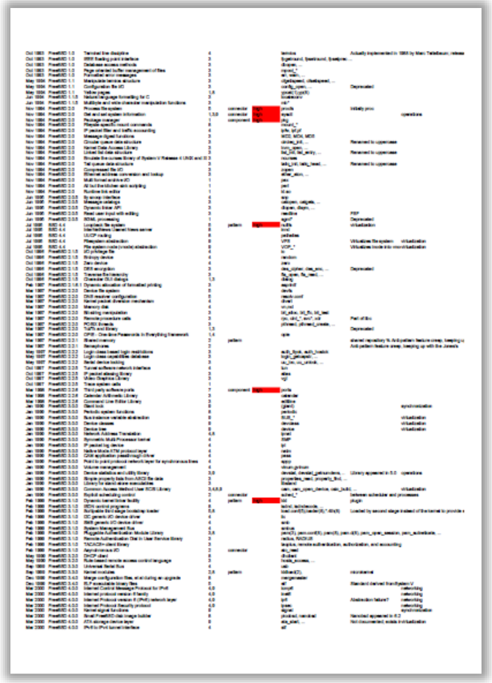
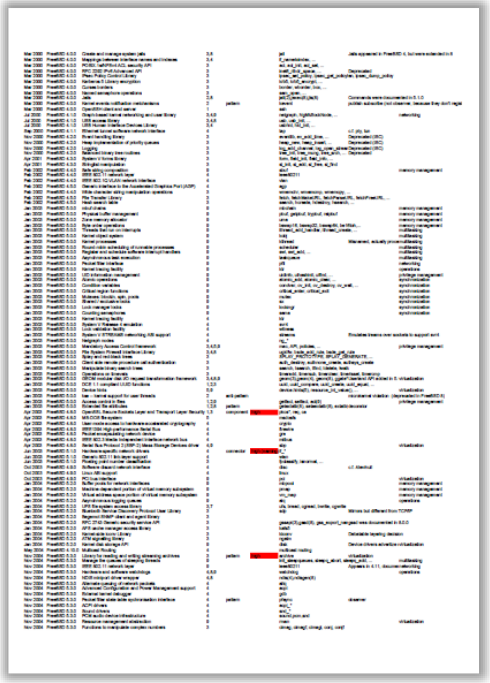
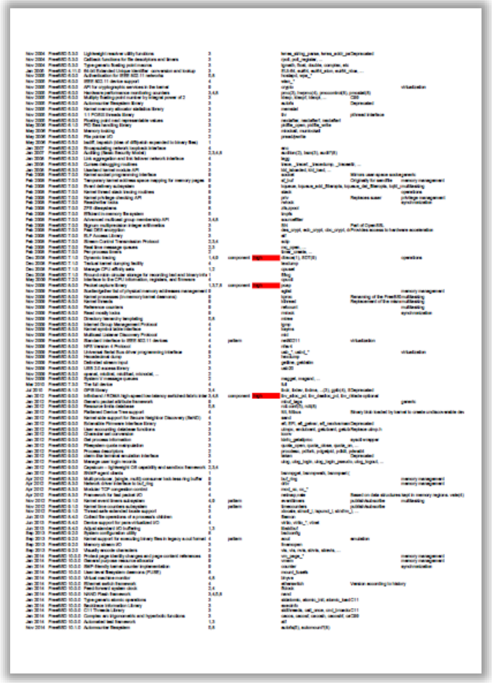
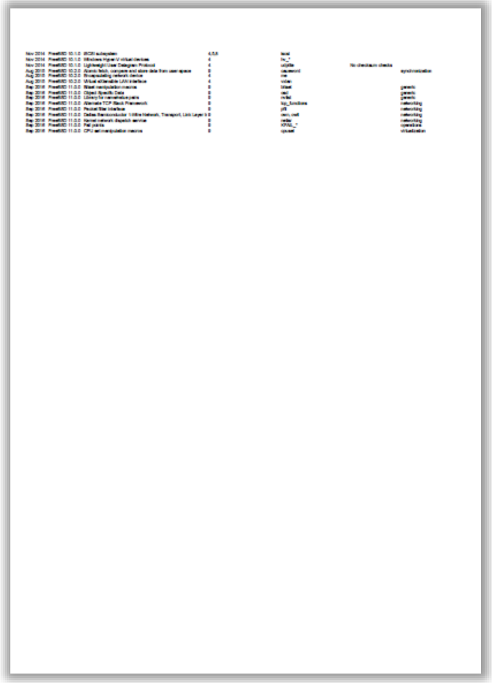


Facility	Appearance	Research V1	Research V2	Research V3	Research V4	Research V5	Research V6	BSD 1	BSD 2	Bell 32V	Research V7	BSD 3
time	Research V1	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬
umount	Research V1	▬	▬	▬	▬	▬	▬	▬	▬			
unlink	Research V1	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬
wait	Research V1	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬
write	Research 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			▬	▬	▬	▬	▬	▬	▬	▬	▬
getgid	Research V4				▬	▬	▬	▬	▬			

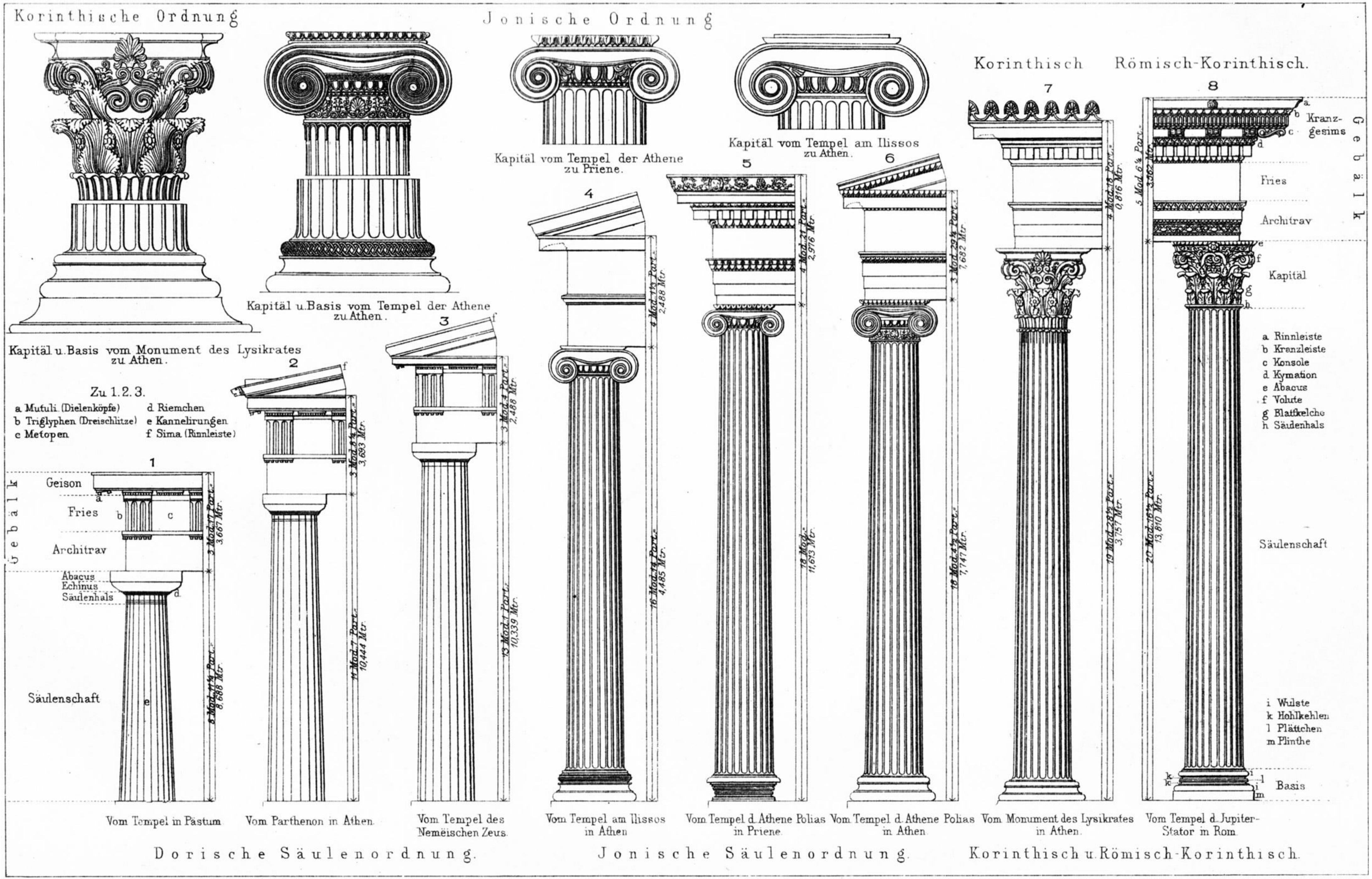
Back to section index

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.

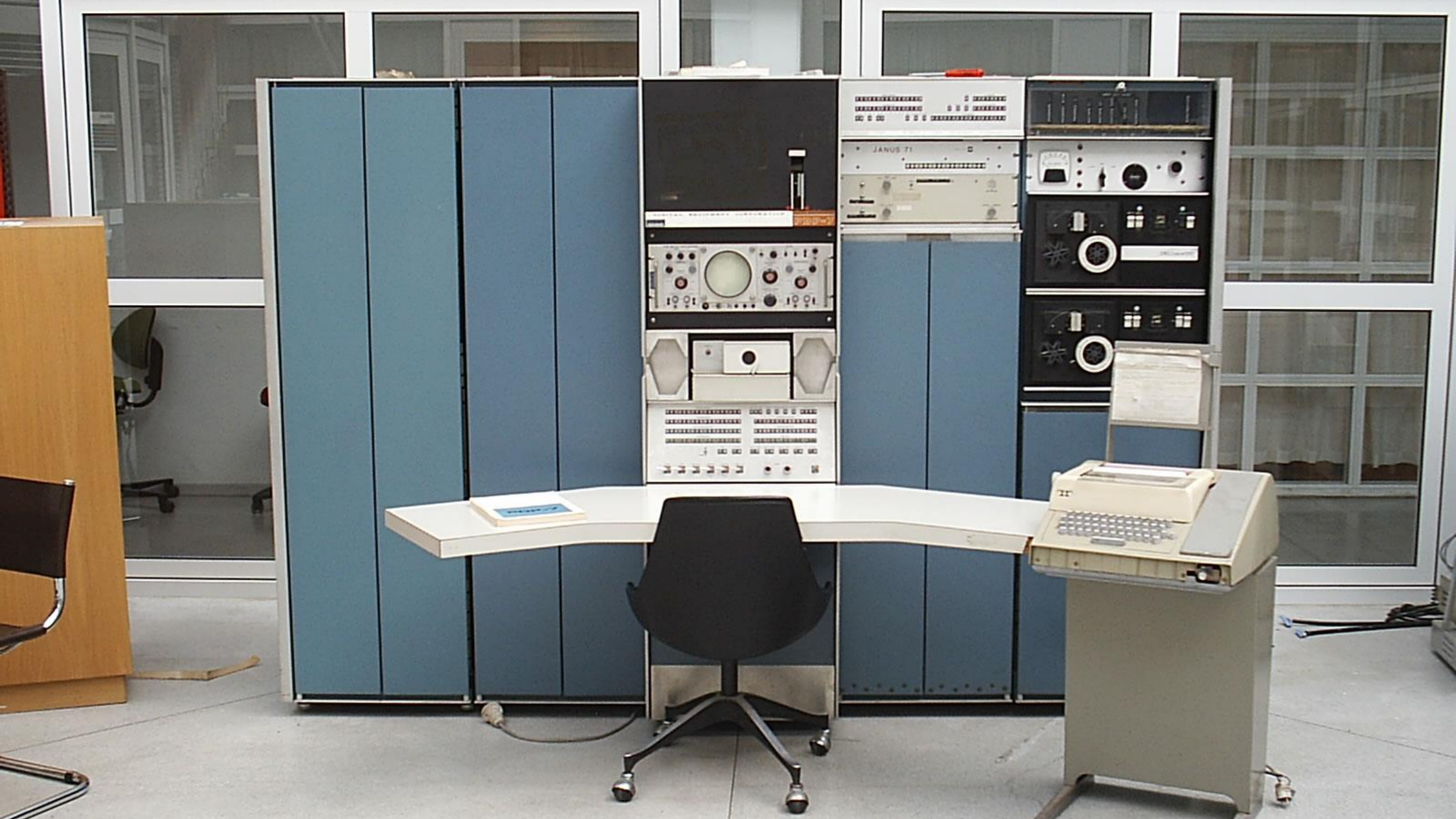







Unix Architecture Evolution



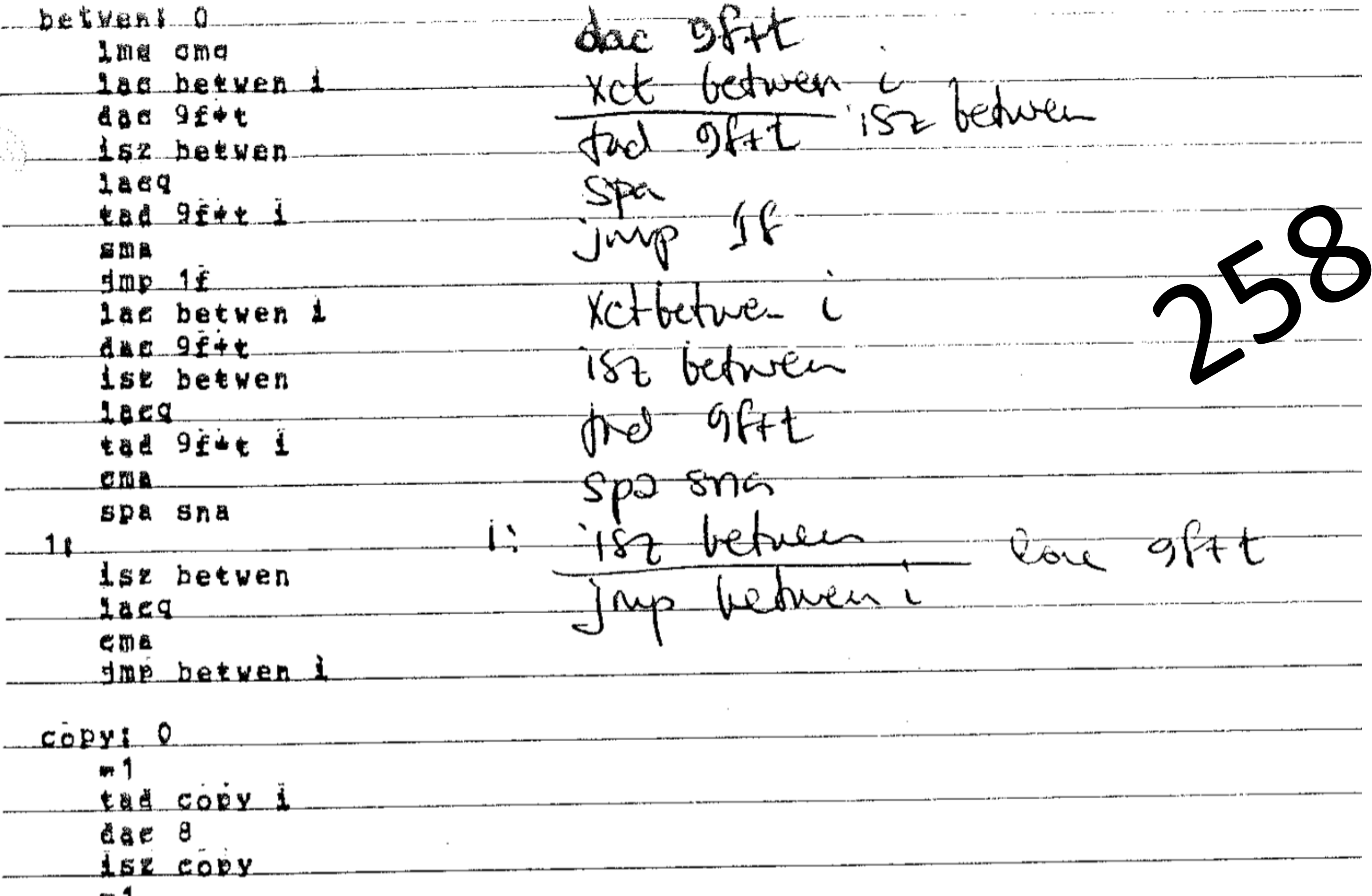
Lessons from the Unix Architecture Evolution

Write small programs.


JANUS 71
PDP-7

PDP-7 [Unix] (1970)

2584 lines



```
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
   -1
```

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

… 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

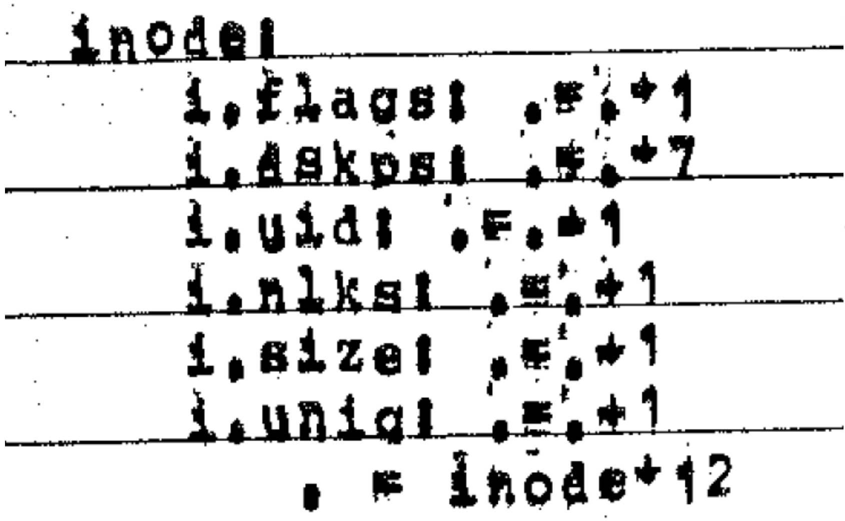
Build modular code through partitioning, composition, and layering.

Layering and Partitioning

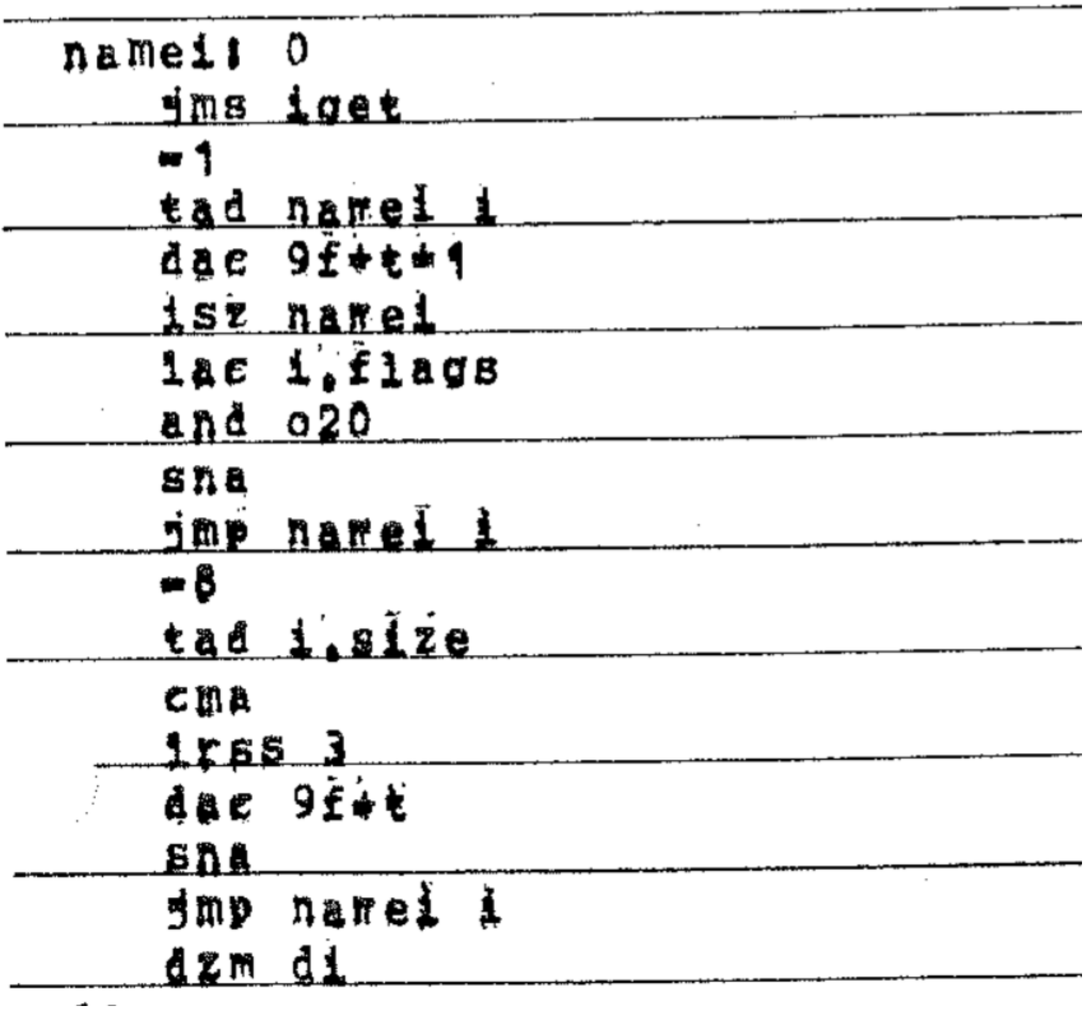
```
adm.s  cat.s    dskio.s   init.s    s6.s
ald.s  check.s  dskres.s  lcase.b   s7.s
apr.s  chmod.s  dsksav.s  maksys.s  s8.s
as.s   chown.s  ds.s      s1.s      s9.s
bc.s   chrm.s   dsw.s     s2.s      scope.v
bi.s   cp.s     ed1.s     s3.s      sop.s
bl.s   db.s     ed2.s     s4.s      trysys.s
cas.s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

Value developer time over machine time.

Separation of File Metadata from File Naming

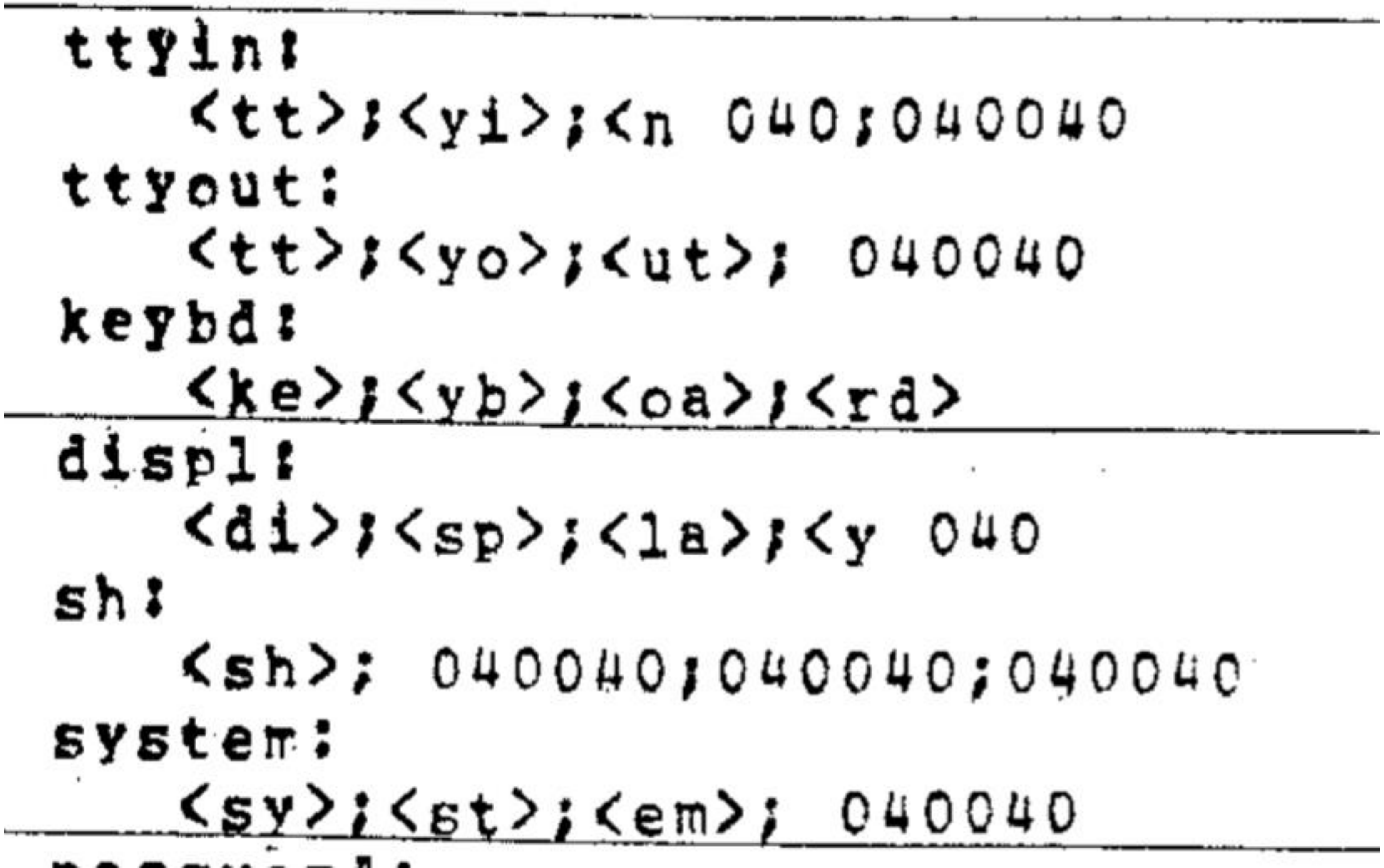


```
inode:
   i.flags: .=.+1
   i.dskps: .=.+7
   i.uid: .=.+1
   i.nlks: .=.+1
   i.size: .=.+1
   i.uniq: .=.+1
      . = inode+12
```



```
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
```

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
```

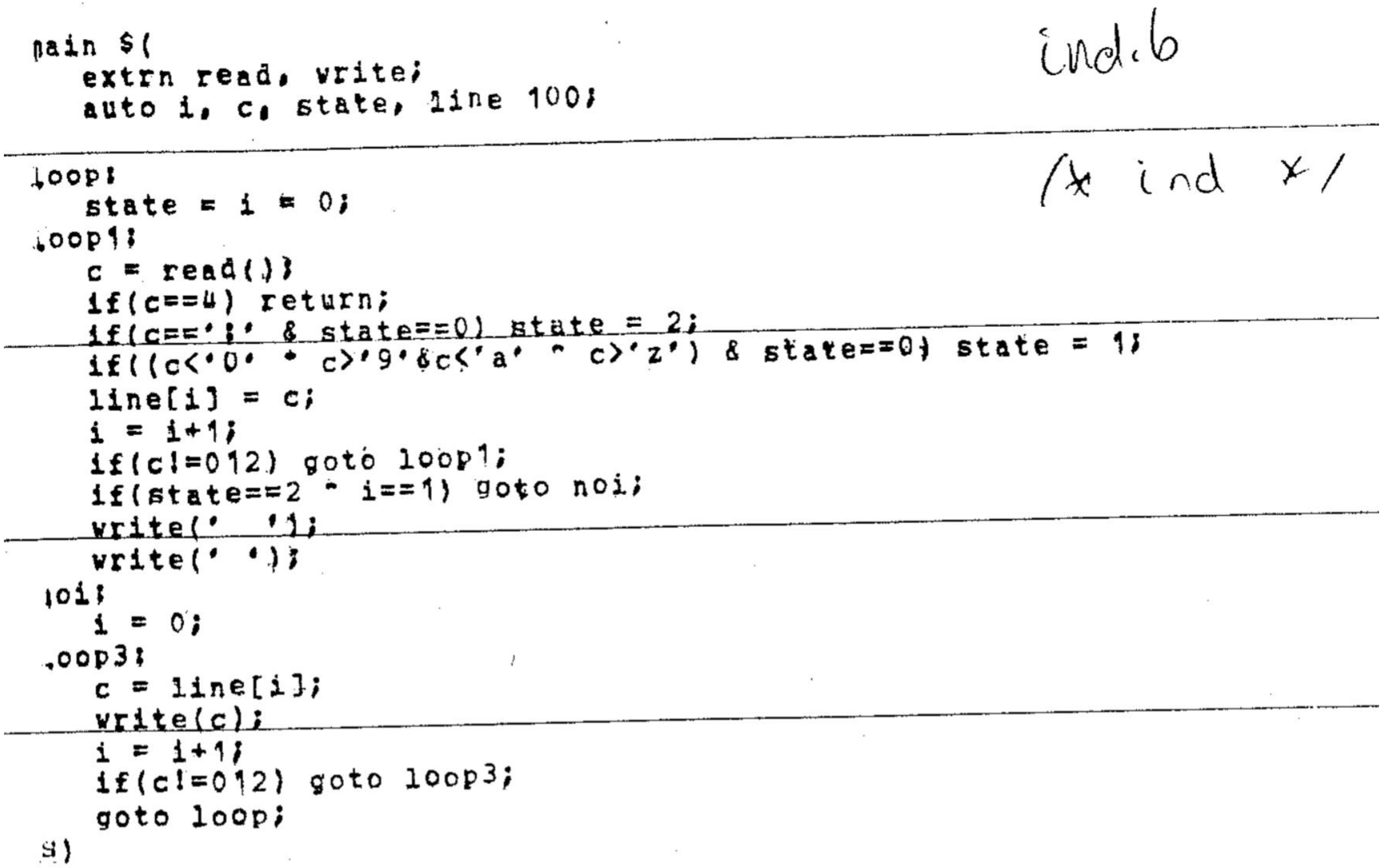
File I/O API

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

File System API

- creat
- rename
- link
- unlink

Interpreter

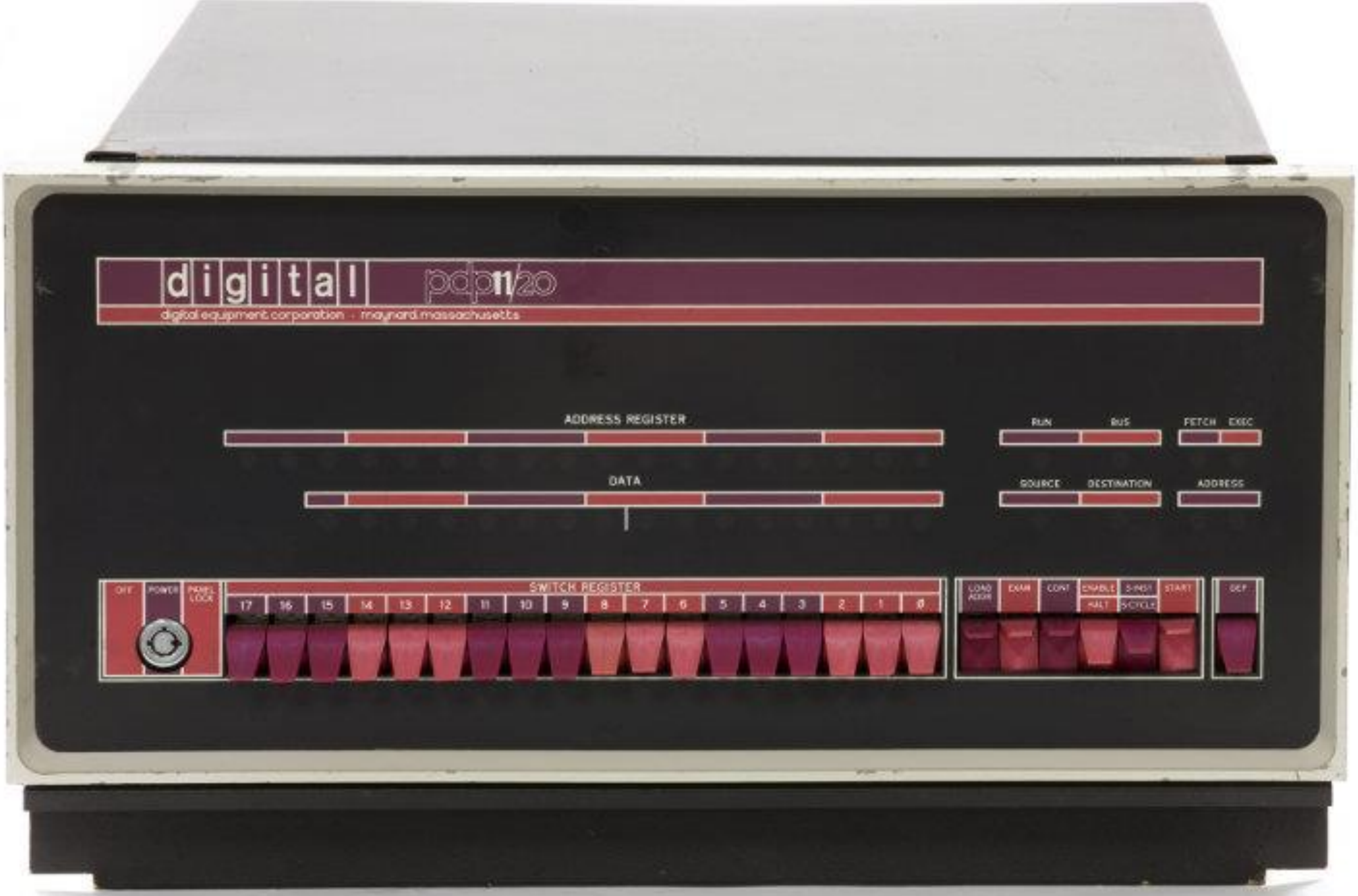


ind.6

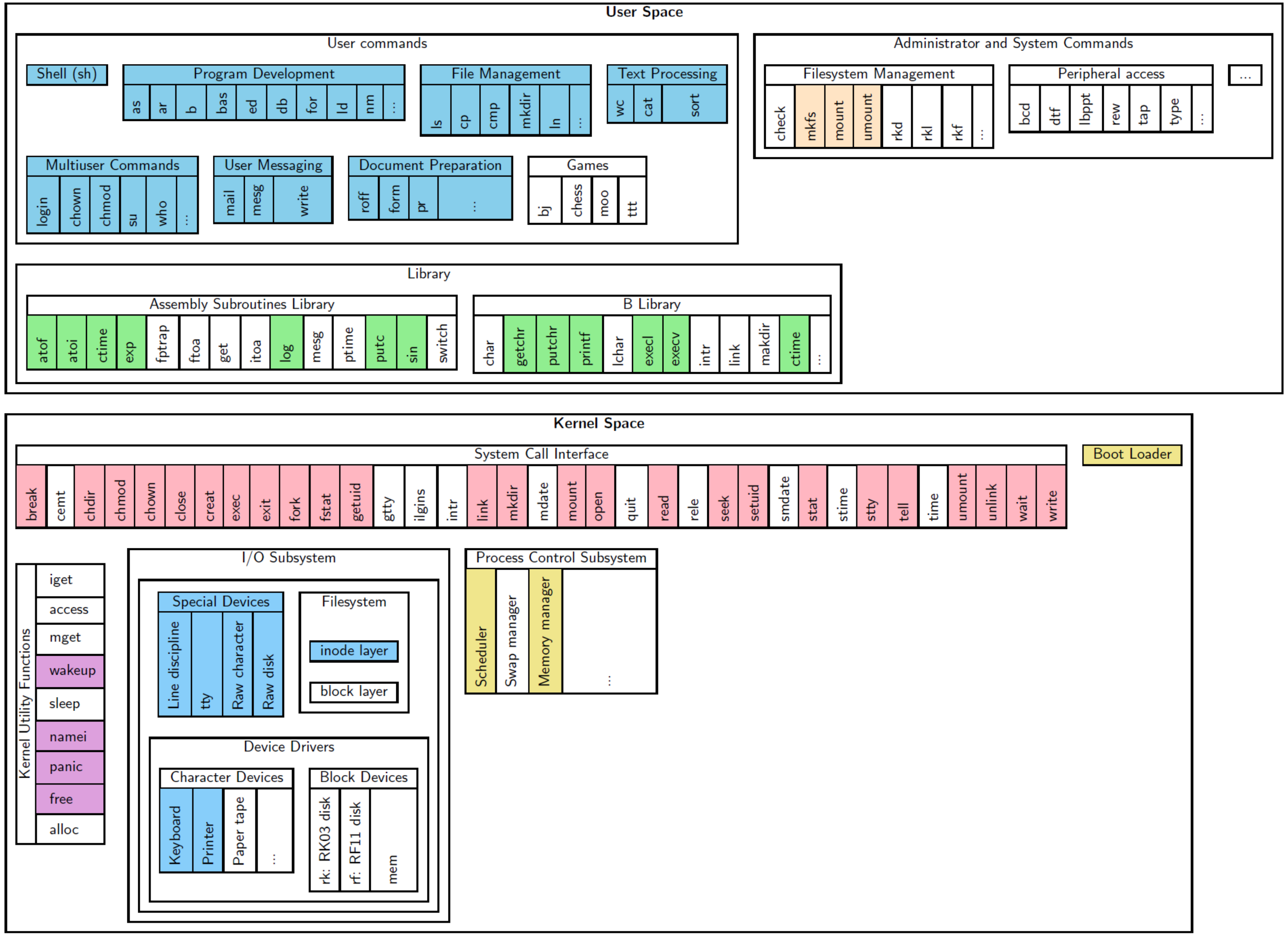
/* ind */

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

loop:
   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;
$)
```

Prototype software before polishing it.


digital
pdp11/20
digital equipment corporation · maynard, massachusetts
ADDRESS REGISTER
DATA
RUN
BUS
FETCH
EXEC
SOURCE
DESTINATION
ADDRESS
OFF
POWER
PANEL LOCK
SWITCH REGISTER
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
Ø
LOAD ADDR
EXAM
CONT
ENABLE
HALT
S-INST
S-CYCLE
START
DEP

First Research Edition (Nov 1971)



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

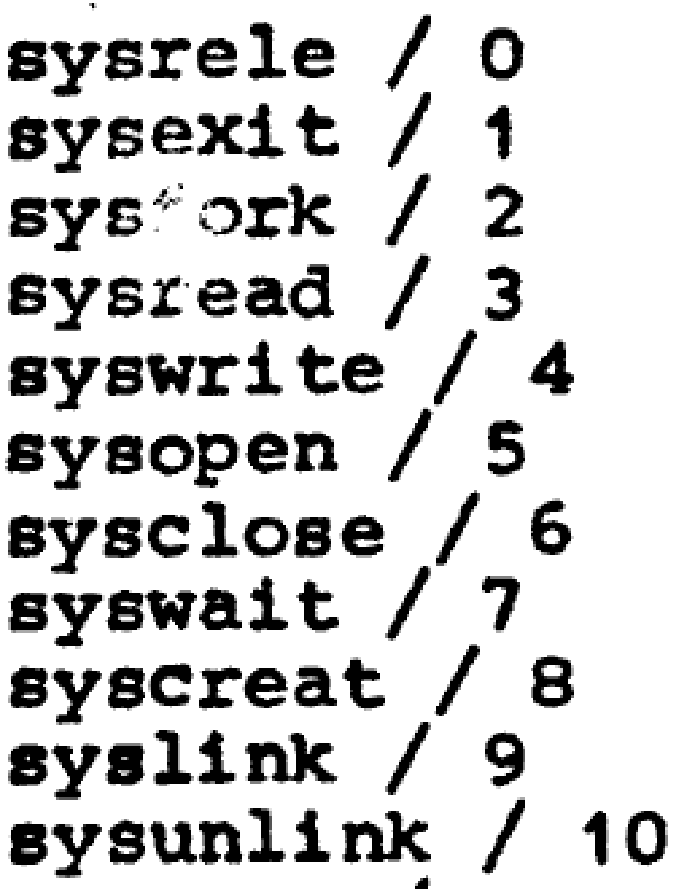
Kernel Utility Functions: iget, access, mget, wakeup, sleep, namei, panic, free, alloc

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

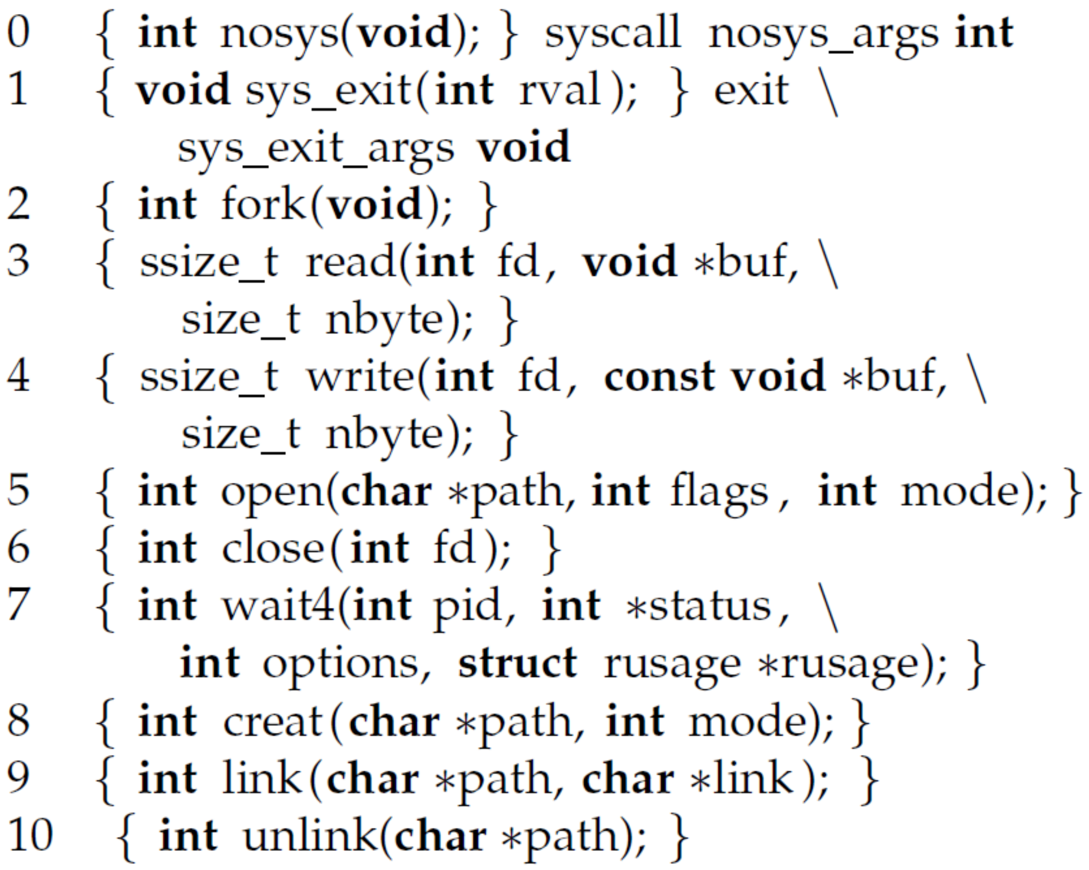
Process Control Subsystem: Scheduler, Swap manager, Memory manager, ⋮

First Edition — 1972



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

FreeBSD 11.1 — 2018

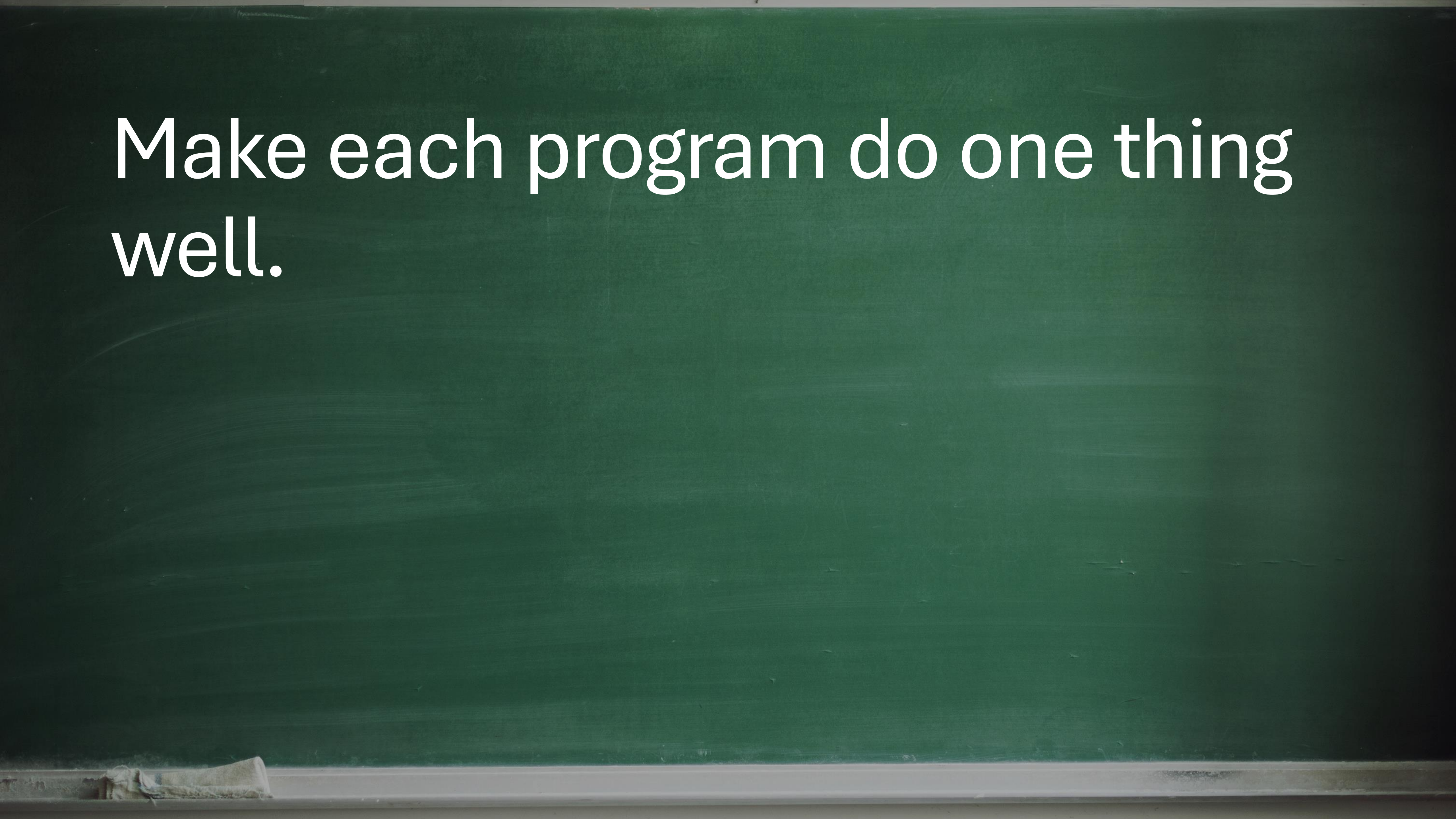


```
0   { int nosys(void); } syscall nosys_args int
1   { void sys_exit(int rval); } exit \
        sys_exit_args void
2   { int fork(void); }
3   { ssize_t read(int fd, void *buf, \
        size_t nbyte); }
4   { ssize_t write(int fd, const void *buf, \
        size_t nbyte); }
5   { int open(char *path, 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); }
```

Linux 6.10 — 2024

0	i386	restart_syscall
1	i386	exit
2	i386	fork
3	i386	read
4	i386	write
5	i386	open
6	i386	close
7	i386	waitpid
8	i386	creat
9	i386	link
10	i386	unlink

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



Make each program do one thing well.

1st edition

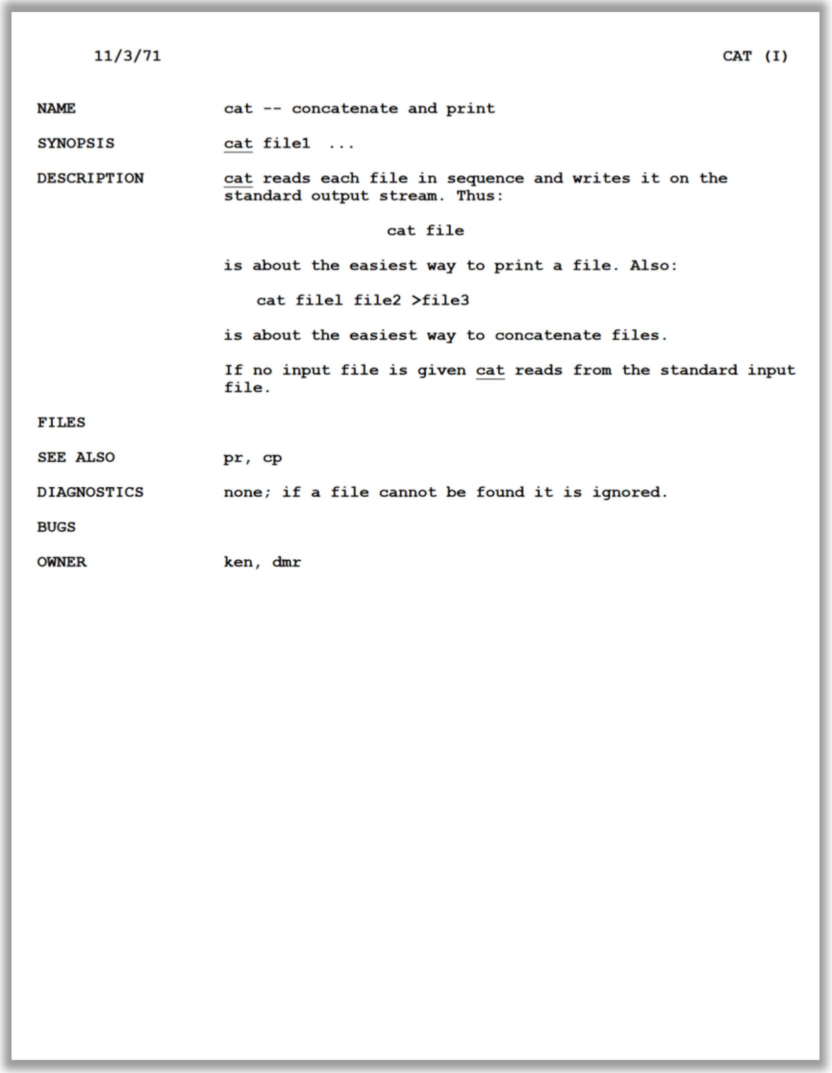
TABLE OF CONTENTS

Nov 3 1971

I. COMMANDS

II. SYSTEM CALLS

III. SUBROUTINES



11/3/71 CAT (I)

NAME cat -- concatenate and print

SYNOPSIS cat file1 ...

DESCRIPTION cat reads each file in sequence and writes it on the standard output stream. Thus:

```
cat file
```

is about the easiest way to print a file. Also:

```
cat file1 file2 >file3
```

is about the easiest way to concatenate files.

If no input file is given cat reads from the standard input file.

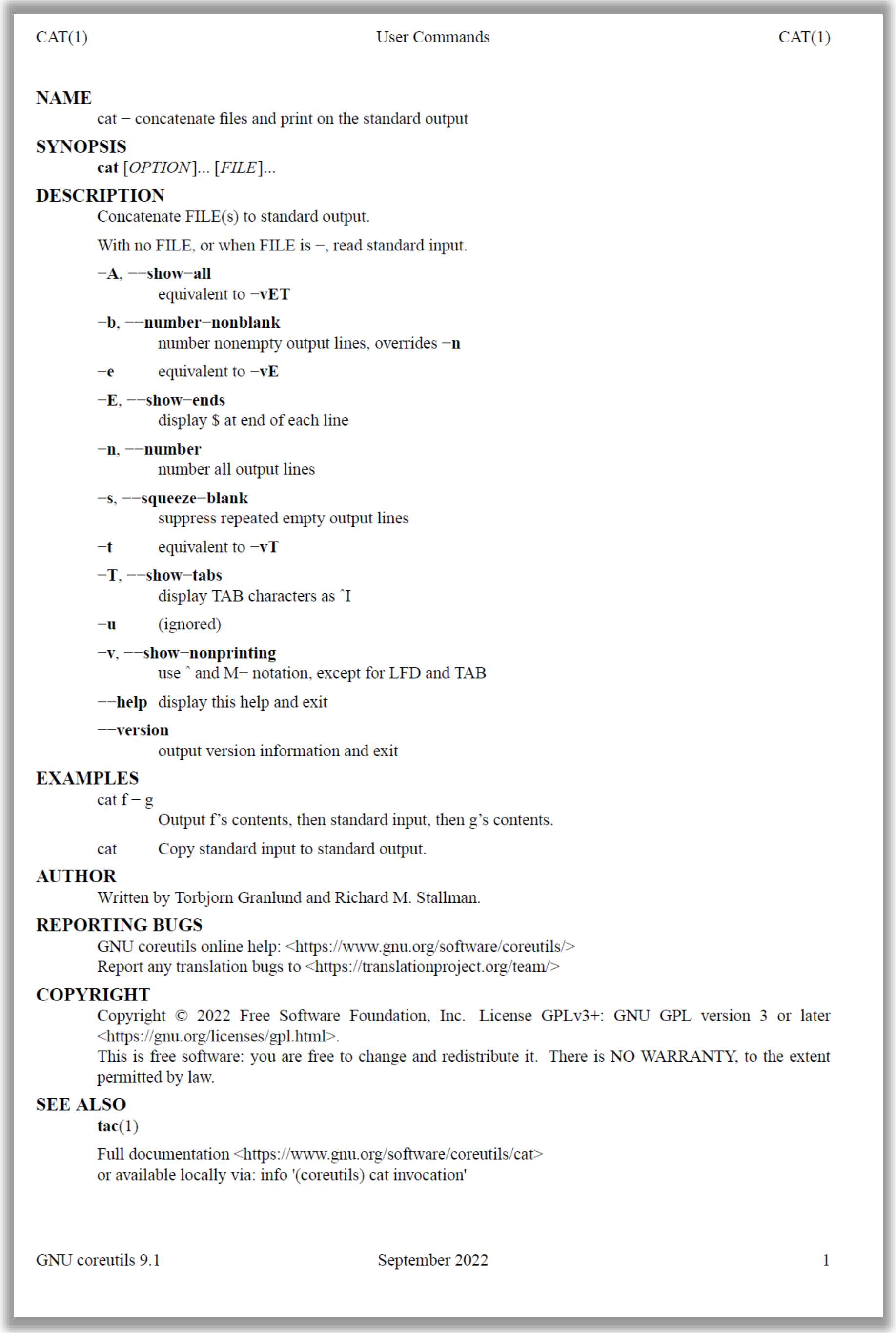
FILES

SEE ALSO pr, cp

DIAGNOSTICS none; if a file cannot be found it is ignored.

BUGS

OWNER ken, dmr



CAT(1) User Commands CAT(1)

NAME

cat − concatenate files and print on the standard output

SYNOPSIS

cat [*OPTION*]... [*FILE*]...

DESCRIPTION

Concatenate FILE(s) to standard output.

With no FILE, or when FILE is −, read standard input.

−A, **−−show−all**
equivalent to **−vET**

−b, **−−number−nonblank**
number nonempty output lines, overrides **−n**

−e equivalent to **−vE**

−E, **−−show−ends**
display $ at end of each line

−n, **−−number**
number all output lines

−s, **−−squeeze−blank**
suppress repeated empty output lines

−t equivalent to **−vT**

−T, **−−show−tabs**
display TAB characters as ^I

−u (ignored)

−v, **−−show−nonprinting**
use ^ and M− notation, except for LFD and TAB

−−help display this help and exit

−−version
output version information and exit

EXAMPLES

cat f − g
Output f's contents, then standard input, then g's contents.

cat Copy standard input to standard output.

AUTHOR

Written by Torbjorn Granlund and Richard M. Stallman.

REPORTING BUGS

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Report any translation bugs to <https://translationproject.org/team/>

COPYRIGHT

Copyright © 2022 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

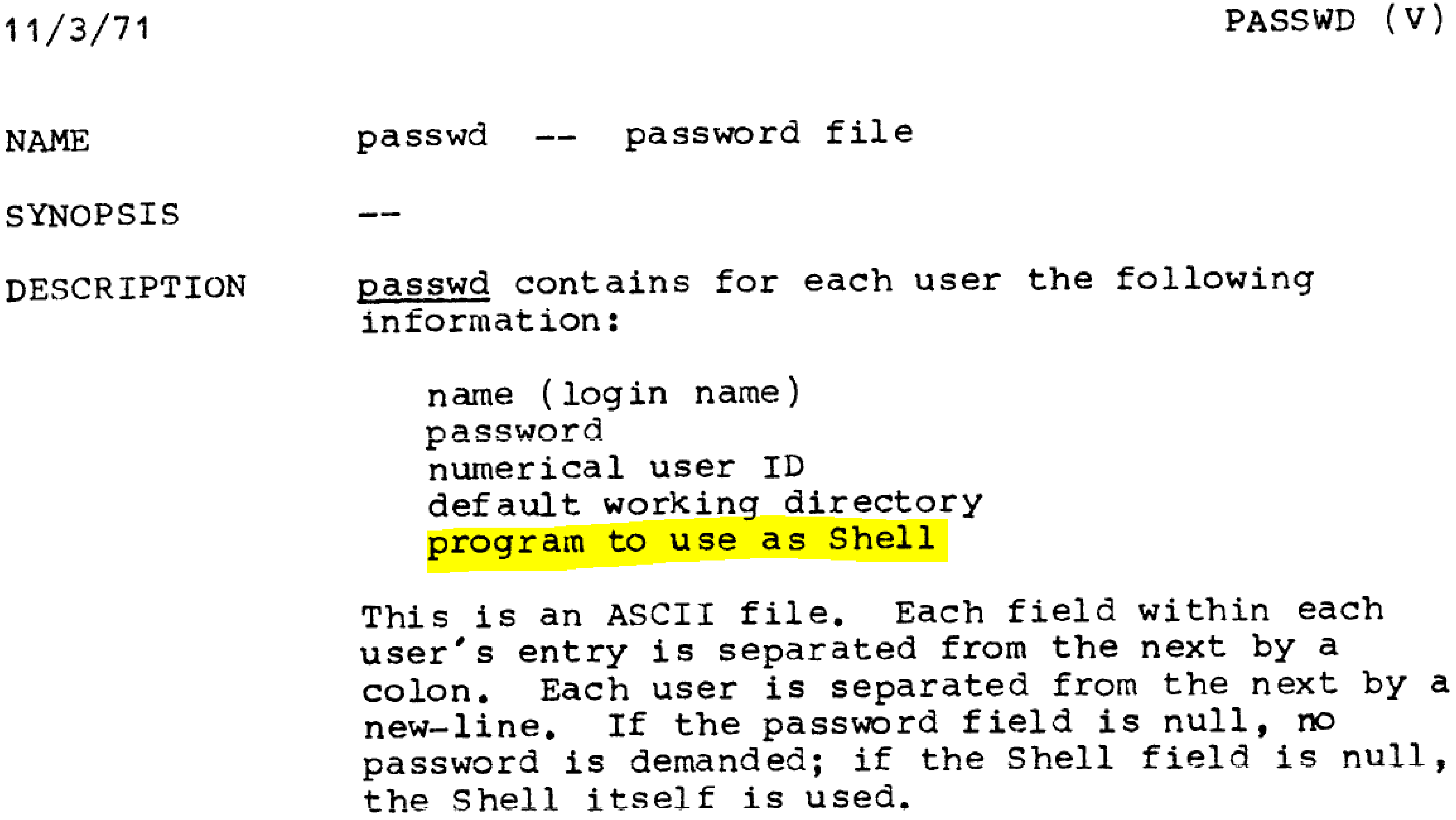
tac(1)

Full documentation <https://www.gnu.org/software/coreutils/cat>
or available locally via: info '(coreutils) cat invocation'

GNU coreutils 9.1 September 2022 1

Avoid captive interfaces.

The Shell as a User Program



11/3/71 PASSWD (V)

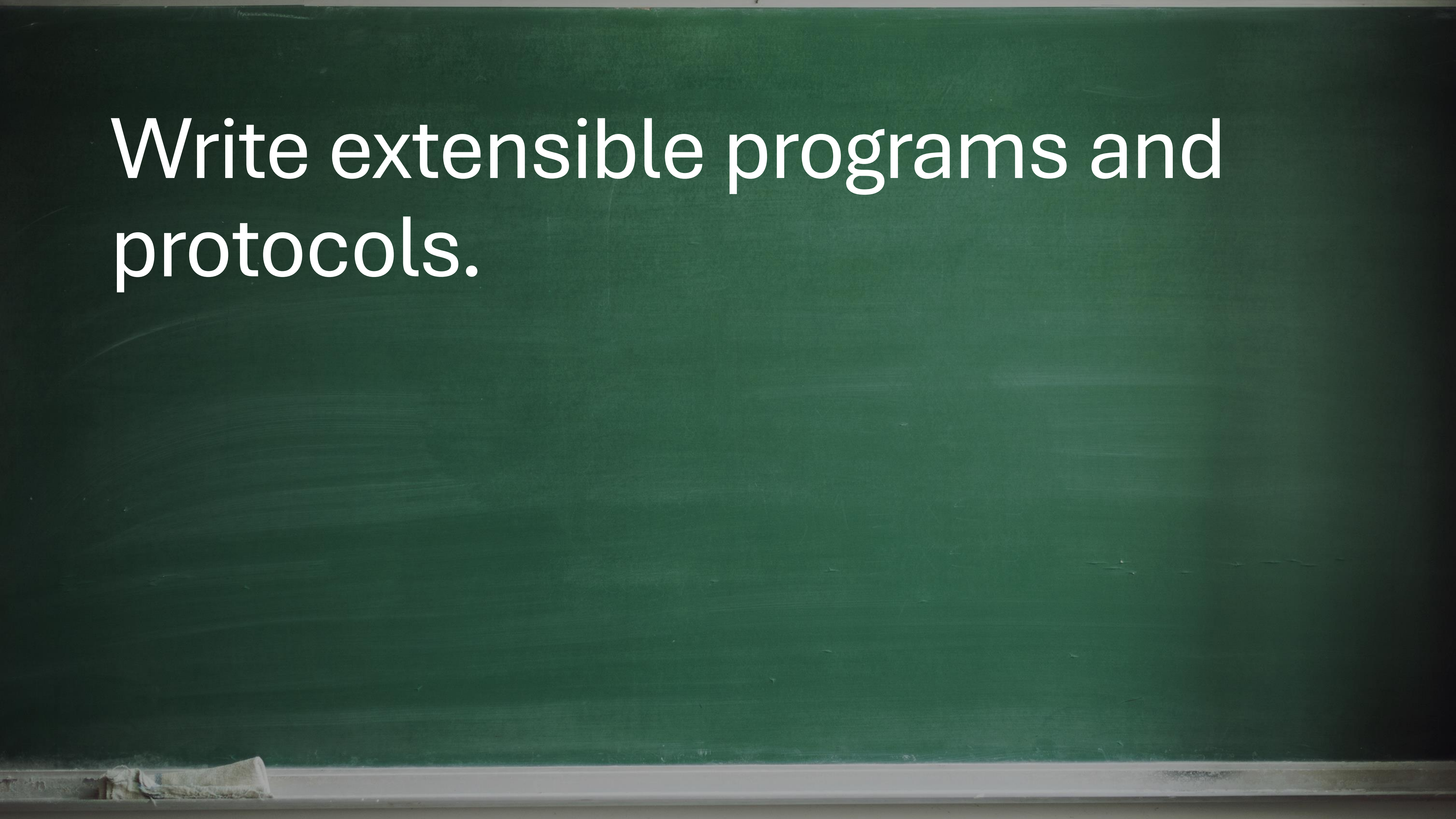
NAME passwd -- password file

SYNOPSIS --

DESCRIPTION passwd contains for each user the following information:

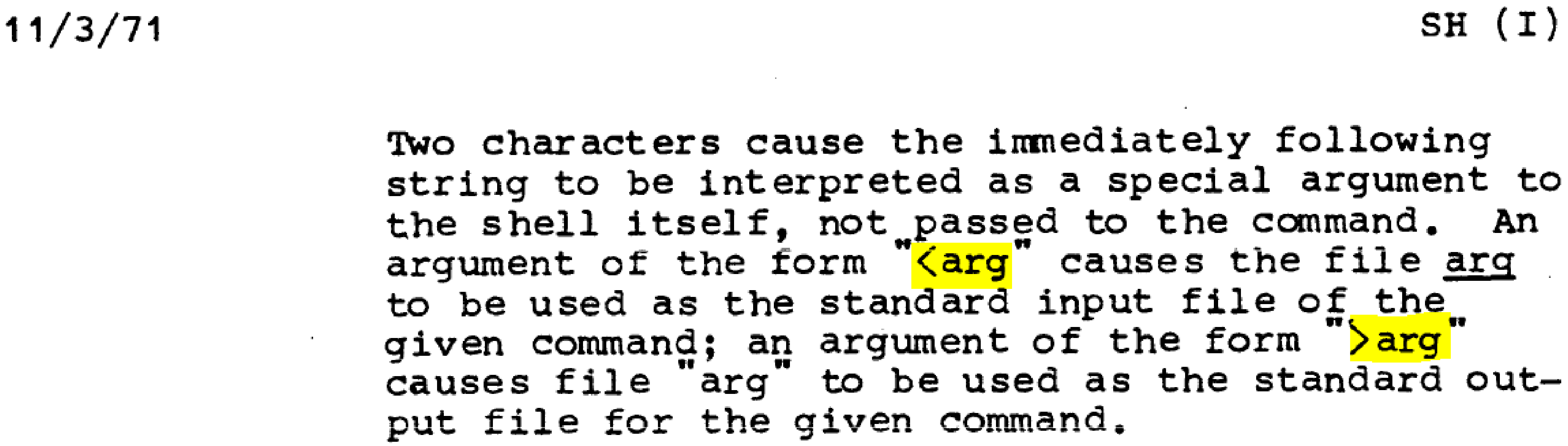
name (login name)
password
numerical user ID
default working directory
program to use as Shell

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.



Write extensible programs and protocols.

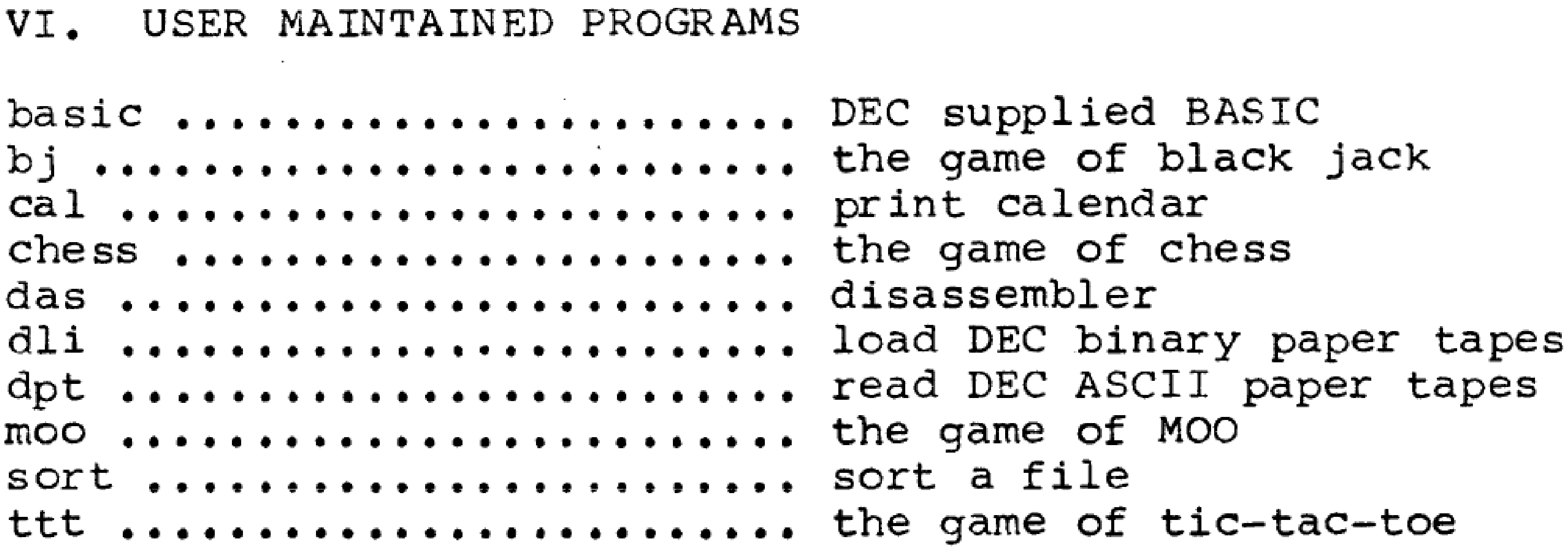
Abstraction of Standard I/O



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.

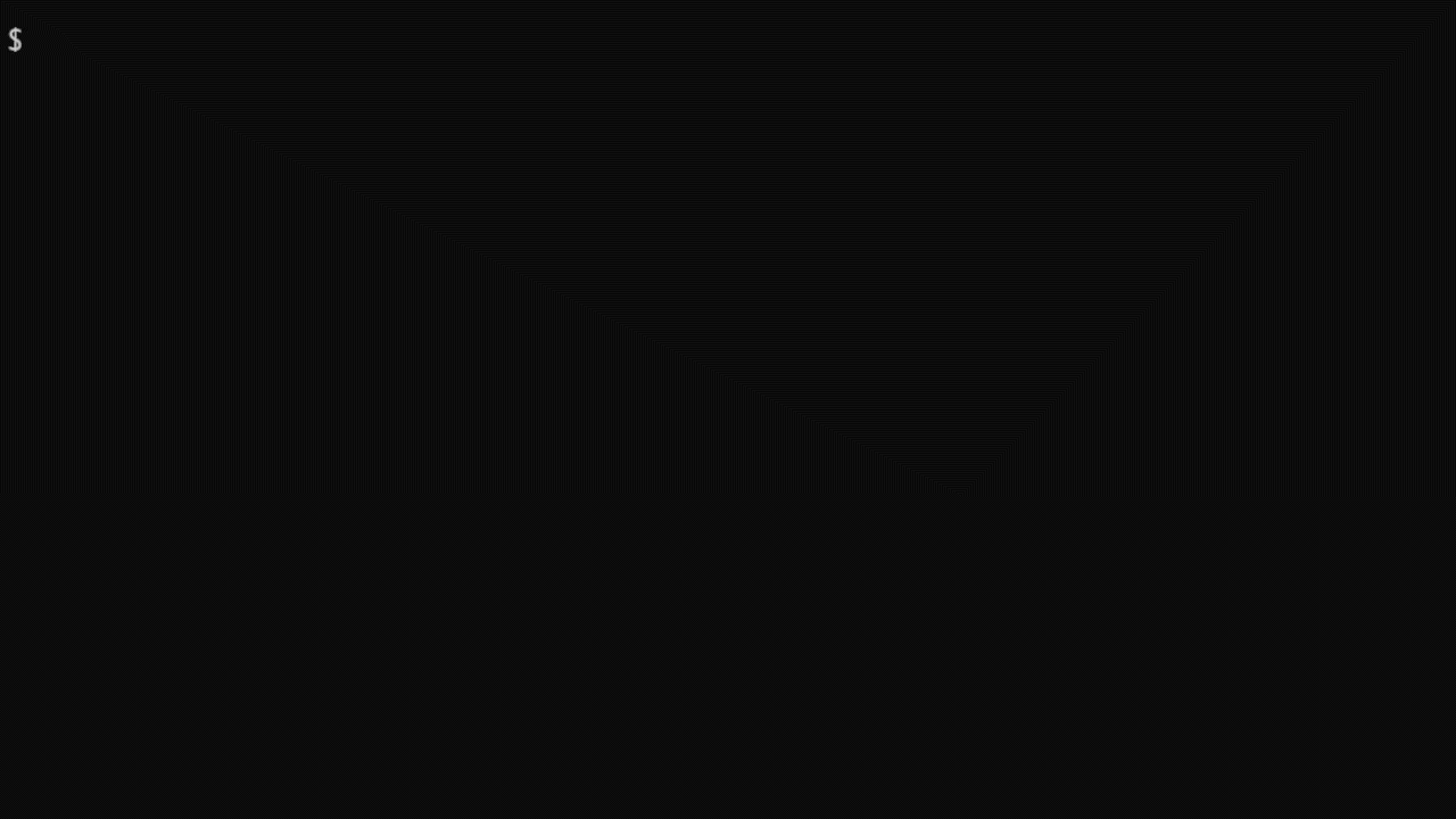
User-Contributed Tools and Games



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

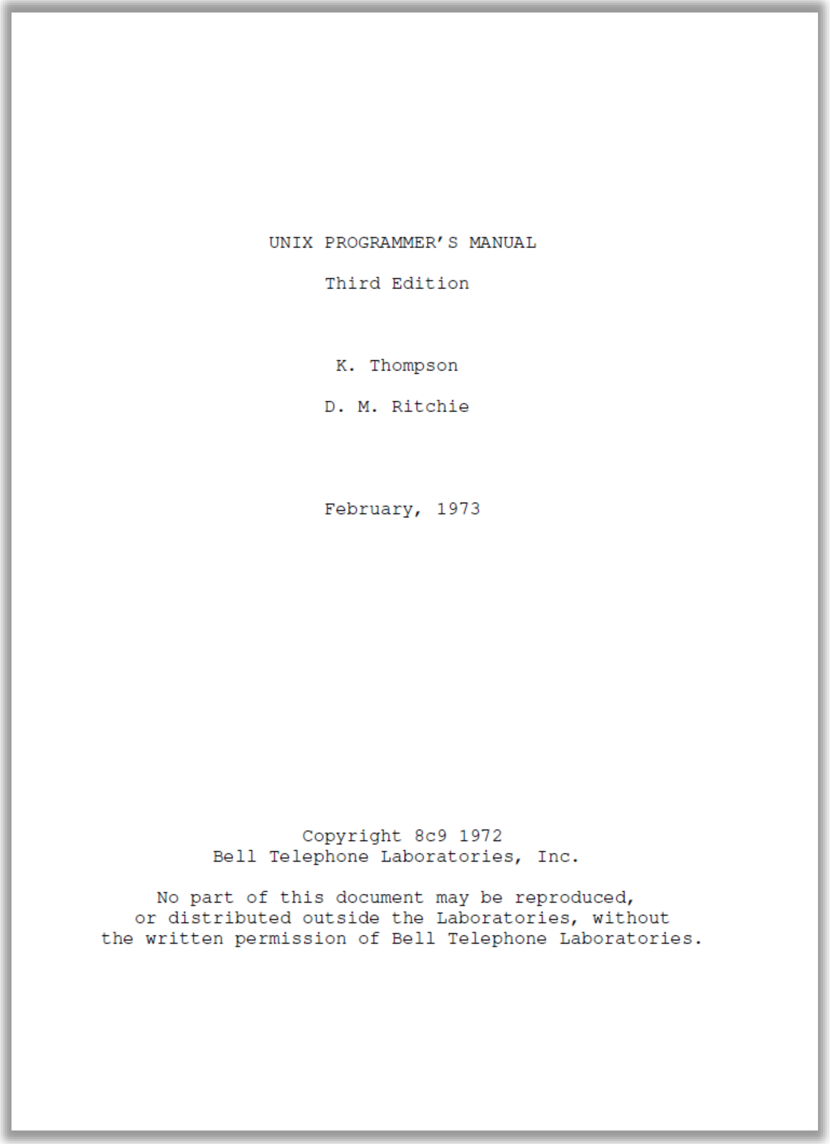
ai-cli-lib



```
$
```

Write programs that work together as filters that process text streams.

Third Research Edition (Feb 1973)



UNIX PROGRAMMER'S MANUAL

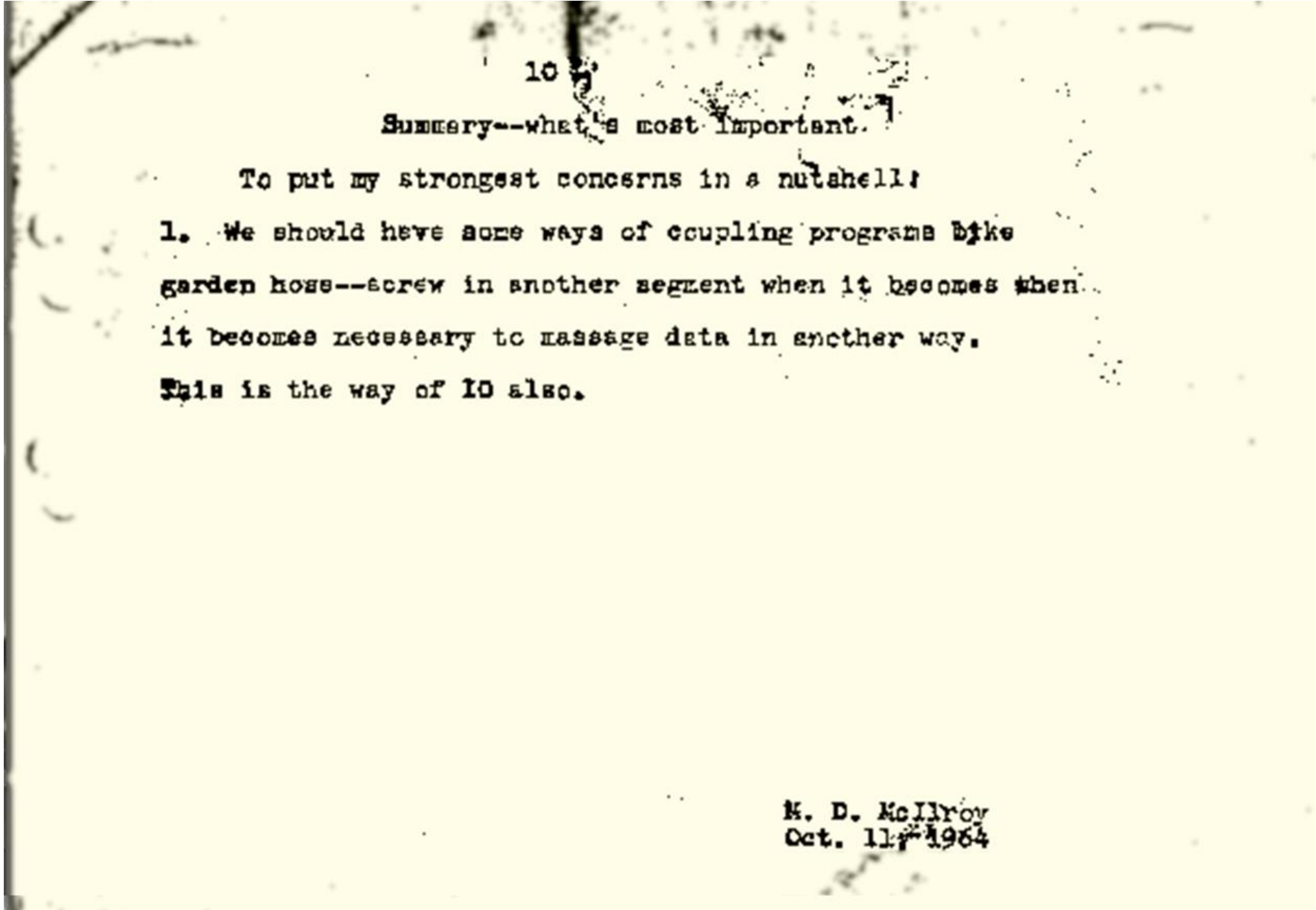
Third Edition

K. Thompson

D. M. Ritchie

February, 1973

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.

M. D. McIlroy
Oct. 11, 1964

NAME

pipe -- create a pipe

SYNOPSIS

```
sys pipe          / pipe = 42.; not in assembler
(file descriptor in r0)
```

DESCRIPTION

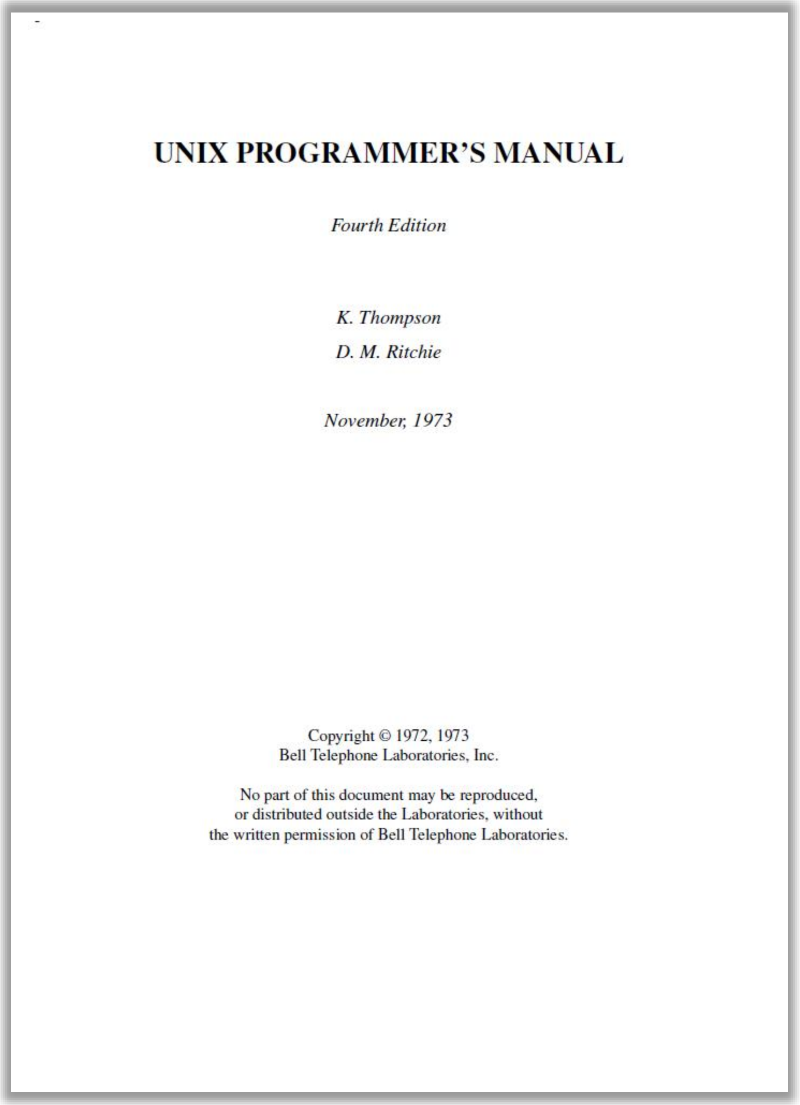
The *pipe* system call creates an I/O mechanism called a pipe. The file descriptor returned can be used in both read and write operations. When the pipe is written, the data is buffered up to 504 bytes at which time the writing process is suspended. A read on the pipe will pick up the buffered data.

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.

Write maintainable programs.

Fourth Research Edition (Nov 1973)



UNIX PROGRAMMER'S MANUAL

Fourth Edition

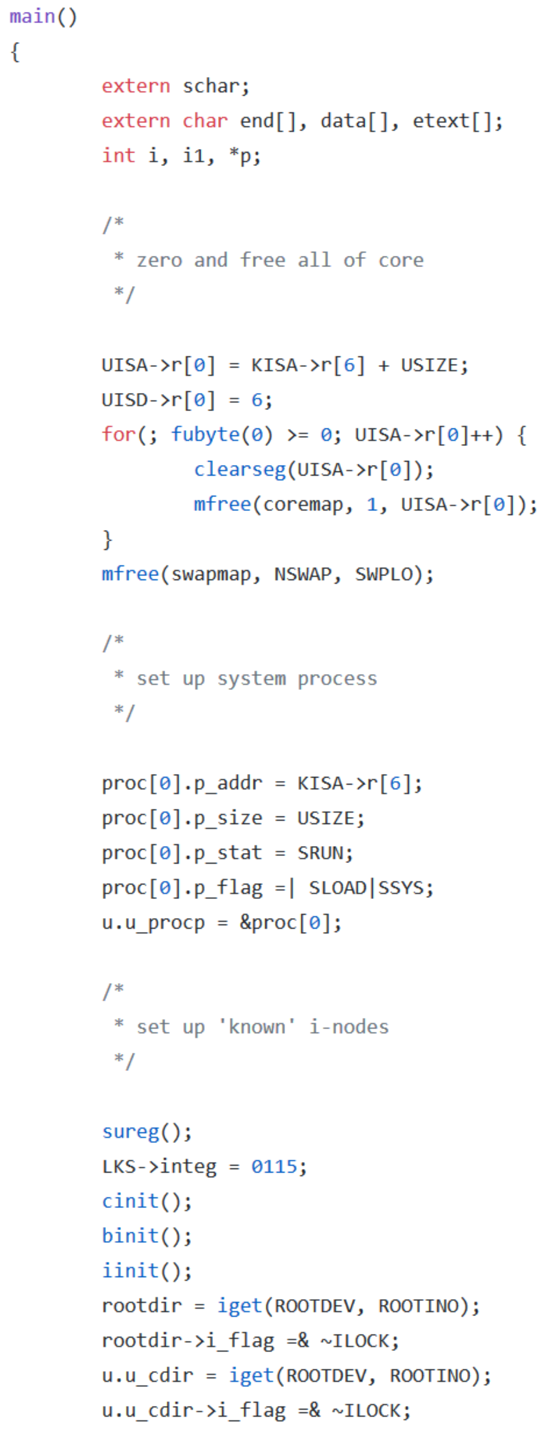
K. Thompson

D. M. Ritchie

November, 1973

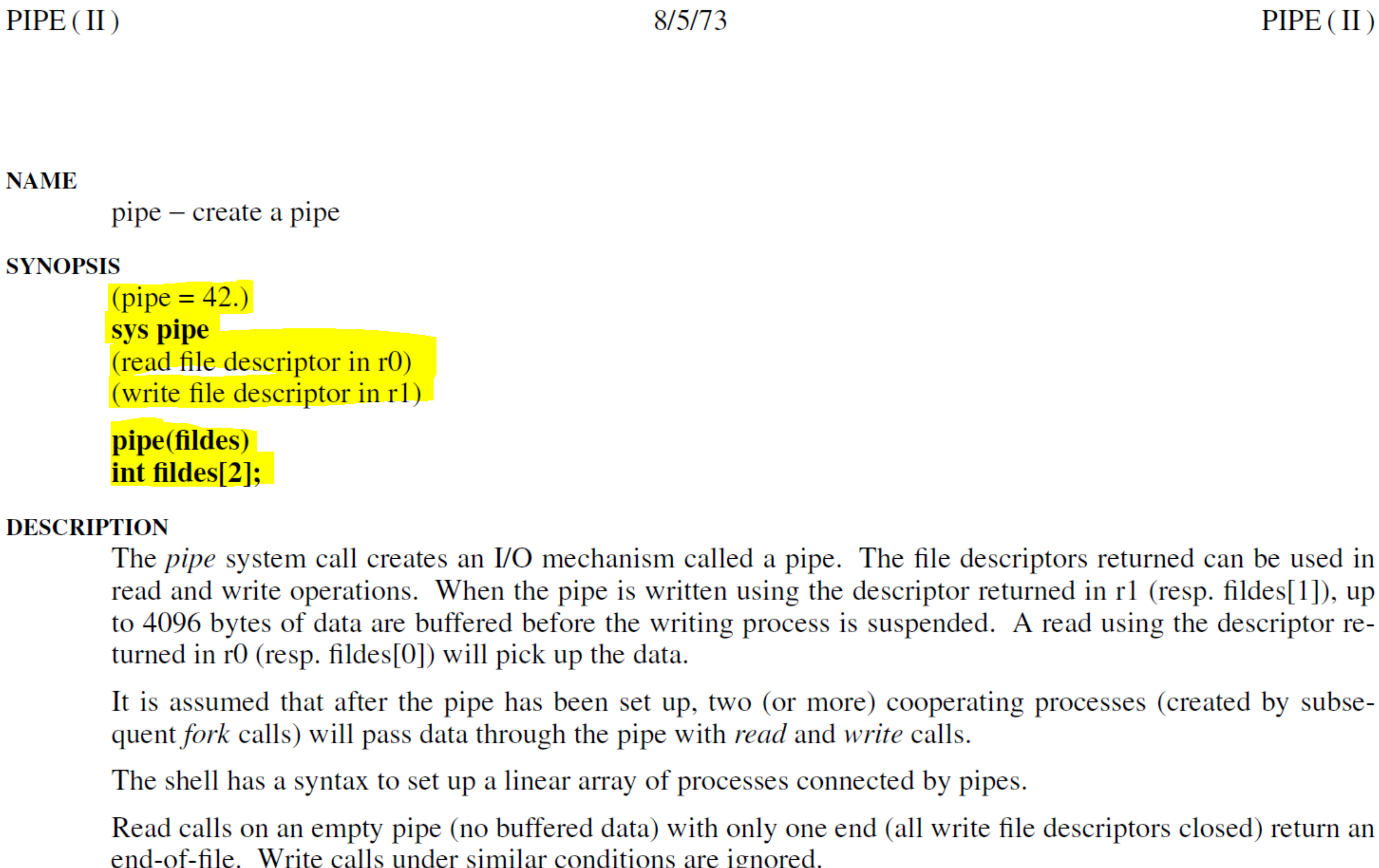
Structured Programming

- Kernel implemented in “New B”
  - 6373 lines New B
  - 768 lines PDP-11 assembly
- Improvement:
  - First Ed.: 248 global symbols
  - Fourth Ed.: 105 functions, 50 assembly symbols



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

Avoid unnecessary output and make failures easy to diagnose.

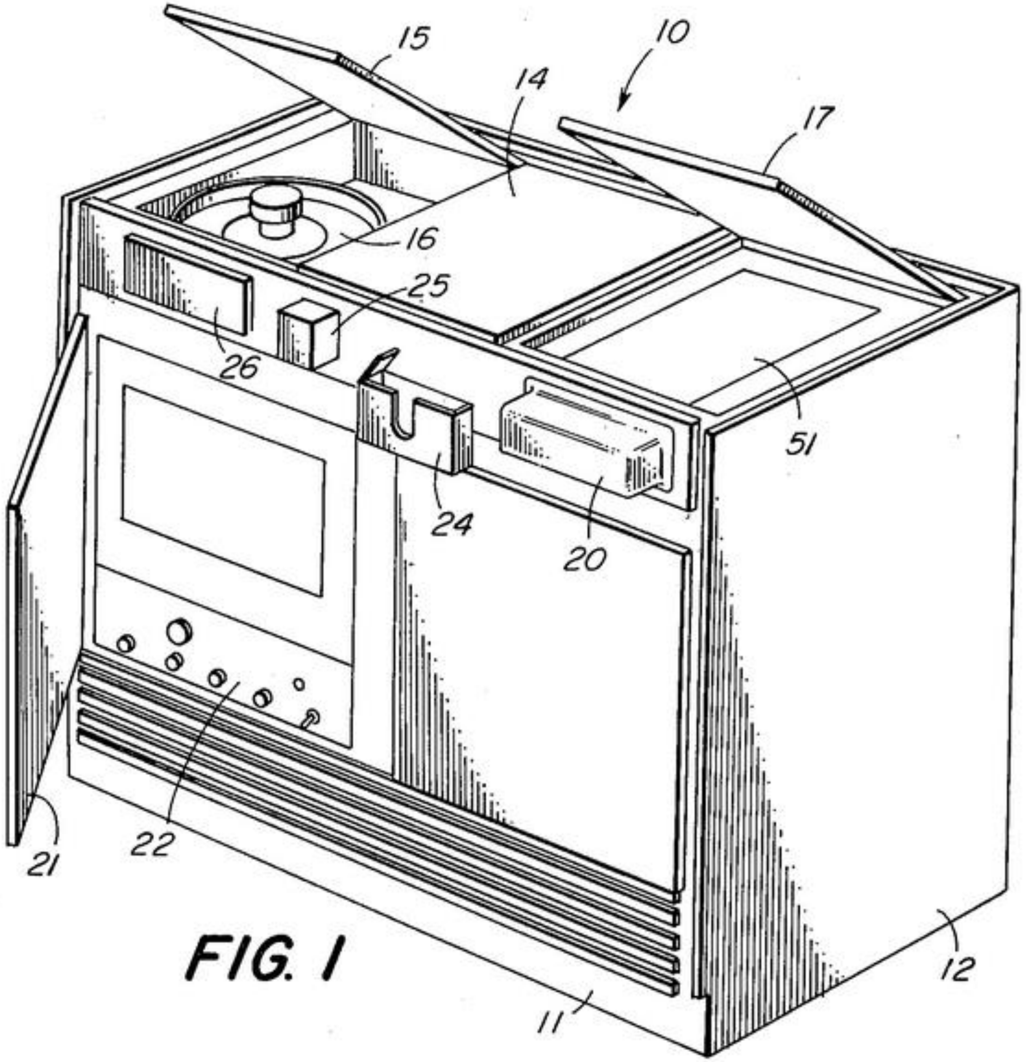


FIG. 1

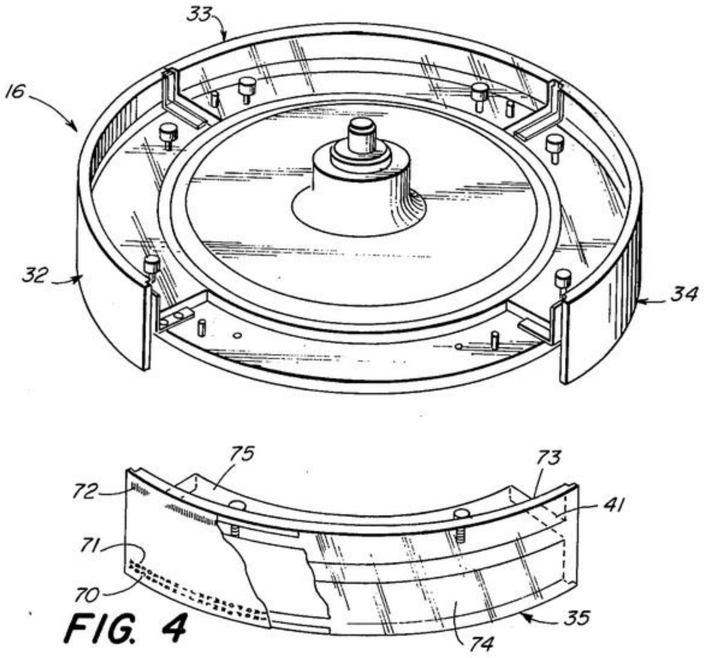


FIG. 4

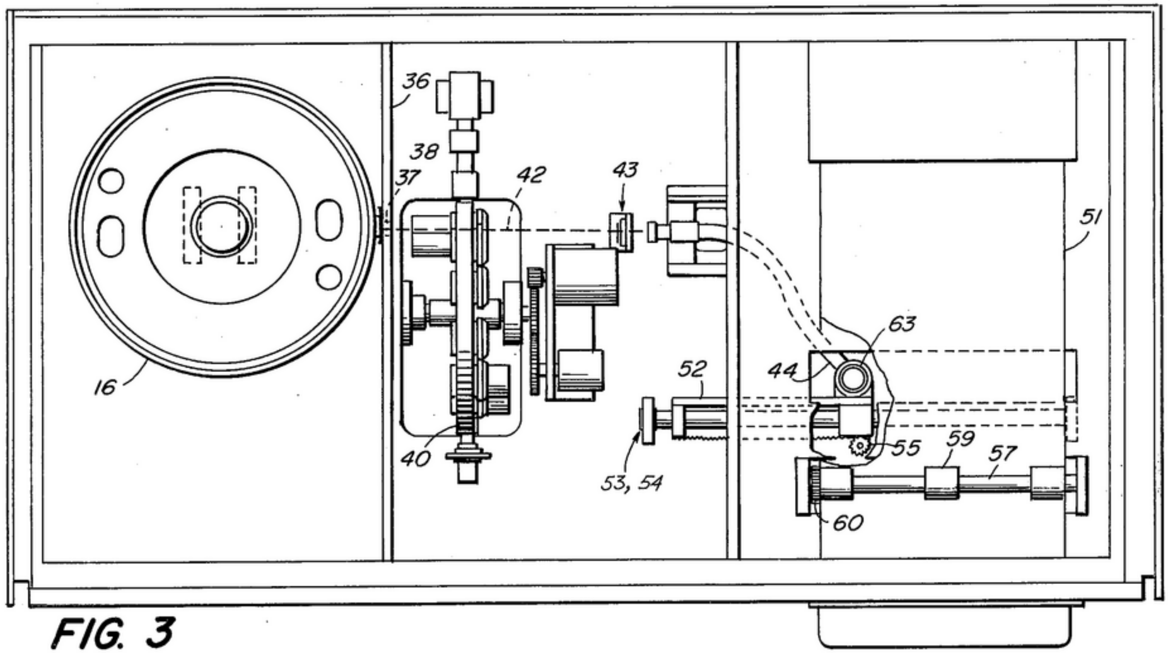
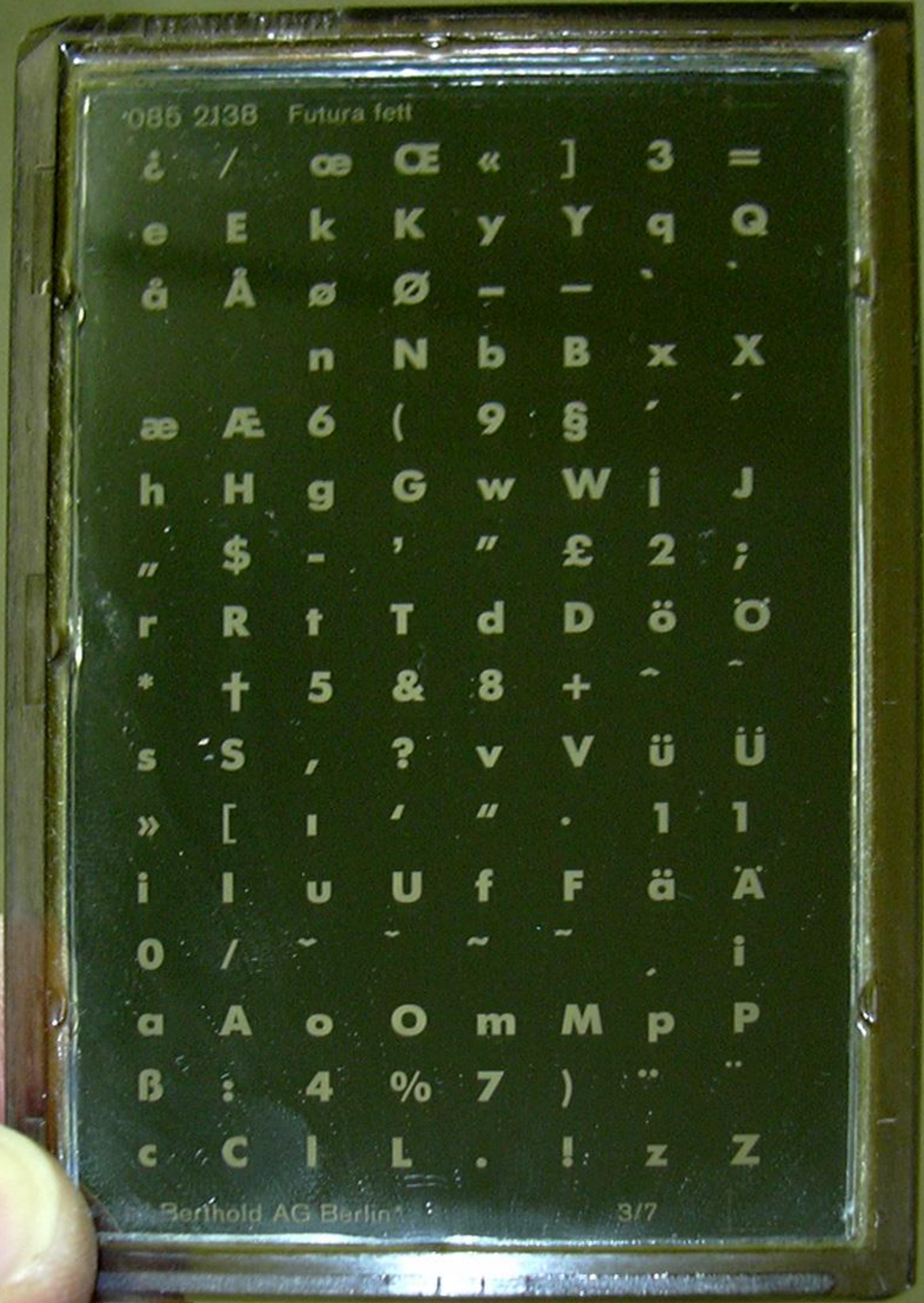


FIG. 3



085 2138 Futura fett

¿	/	œ	Œ	«	]	3	=
e	E	k	K	y	Y	q	Q
å	Å	ø	Ø	–	—	`	˙
		n	N	b	B	x	X
æ	Æ	6	(	9	§	´	´
h	H	g	G	w	W	¡	J
„	$	-	’	”	£	2	;
r	R	t	T	d	D	ö	Ö
*	†	5	&	8	+	ˆ	˜
s	S	,	?	v	V	ü	Ü
»	[	ı	‘	“	·	1	1
i	I	u	U	f	F	ä	Ä
0	/	˘	˘	~	˜	¸	i
a	A	o	O	m	M	p	P
ß	:	4	%	7	)	¨	¨
c	C	l	L	.	!	z	Z

Berthold AG Berlin 3/7

NAME

cat – phototypesetter interface

DESCRIPTION

Cat provides the interface to a Graphic Systems C/A/T phototypesetter. Bytes written on the file specify font, size, and other control information as well as the characters to be flashed. The coding will not be described here.

Only one process may have this file open at a time. It is write-only.

FILES

/dev/cat

SEE ALSO

troff (I), Graphic Systems specification (available on request)

BUGS

Faces of Open Source / Peter Adams

“After phototypesetting, you had to take a long wide strip of paper and feed it carefully into a smelly, icky machine which eventually (several minutes later) spat out the paper with the printing visible.”

“One afternoon several of us had the same experience — typesetting something, feeding the paper through the developer, only to find a single, beautifully typeset line: "cannot open file foobar" The grumbles were loud enough and in the presence of the right people, and a couple of days later the standard error file was born...”

— Stephen C. Johnson

Use shell scripts to increase leverage and portability.

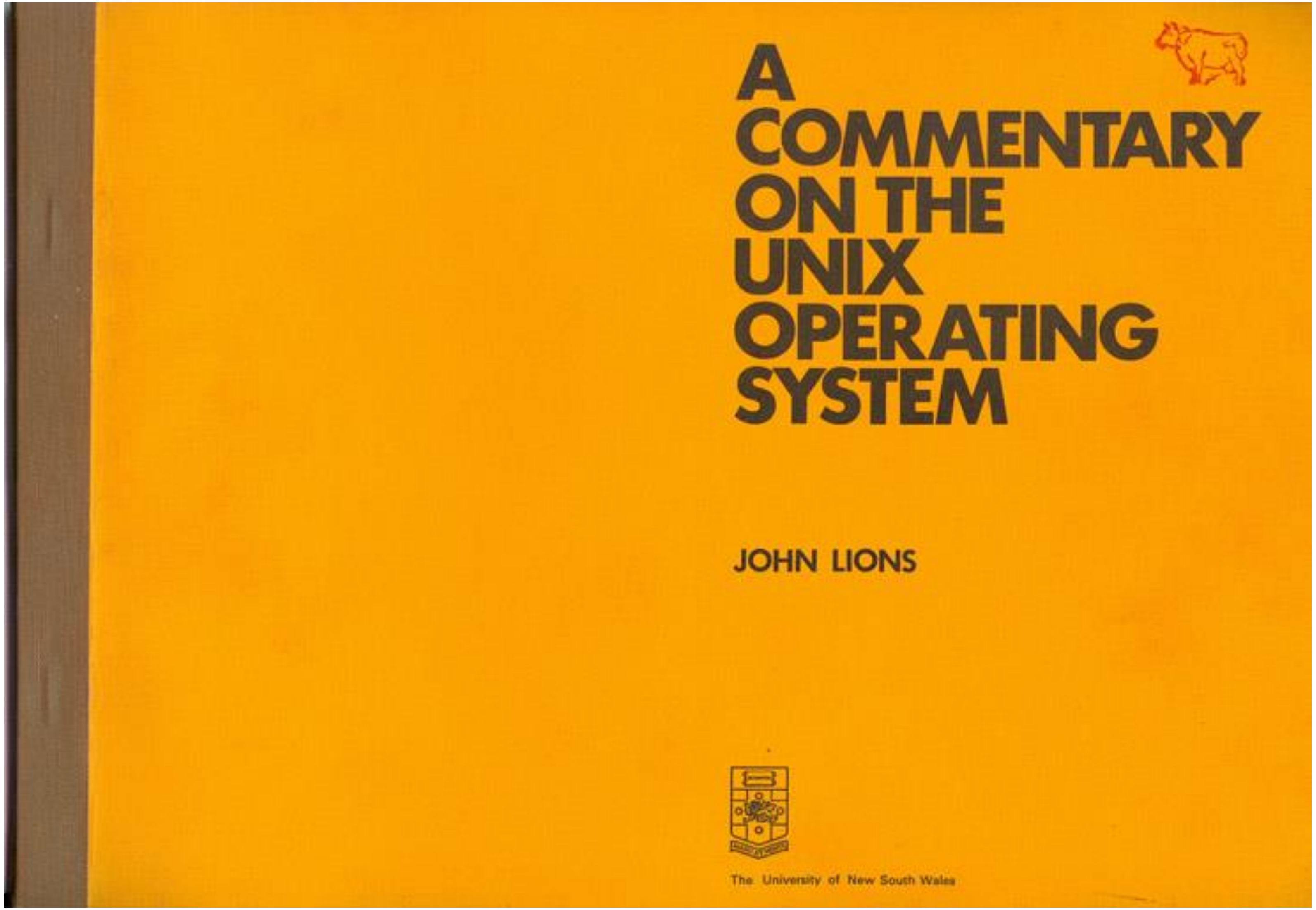
Fifth Research Edition (Jun 1974)

- 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
```

Choose appropriately powerful abstractions.

Sixth Research Edition (May 1975)



A COMMENTARY ON THE UNIX OPERATING SYSTEM

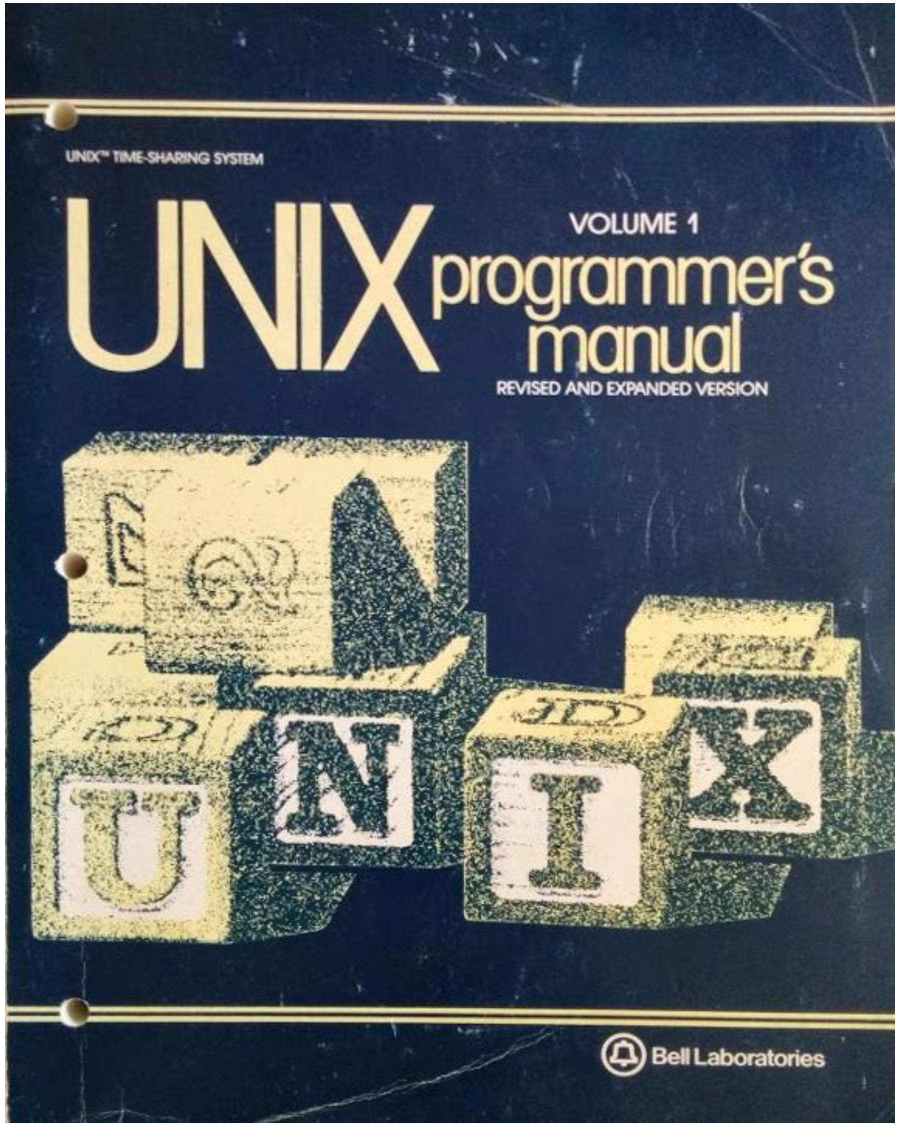
JOHN LIONS

The University of New South Wales

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
```

Seventh Research Edition (Jan 1979)



UNIX™ TIME-SHARING SYSTEM

VOLUME 1

UNIX programmer's manual

REVISED AND EXPANDED VERSION

Bell Laboratories

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”

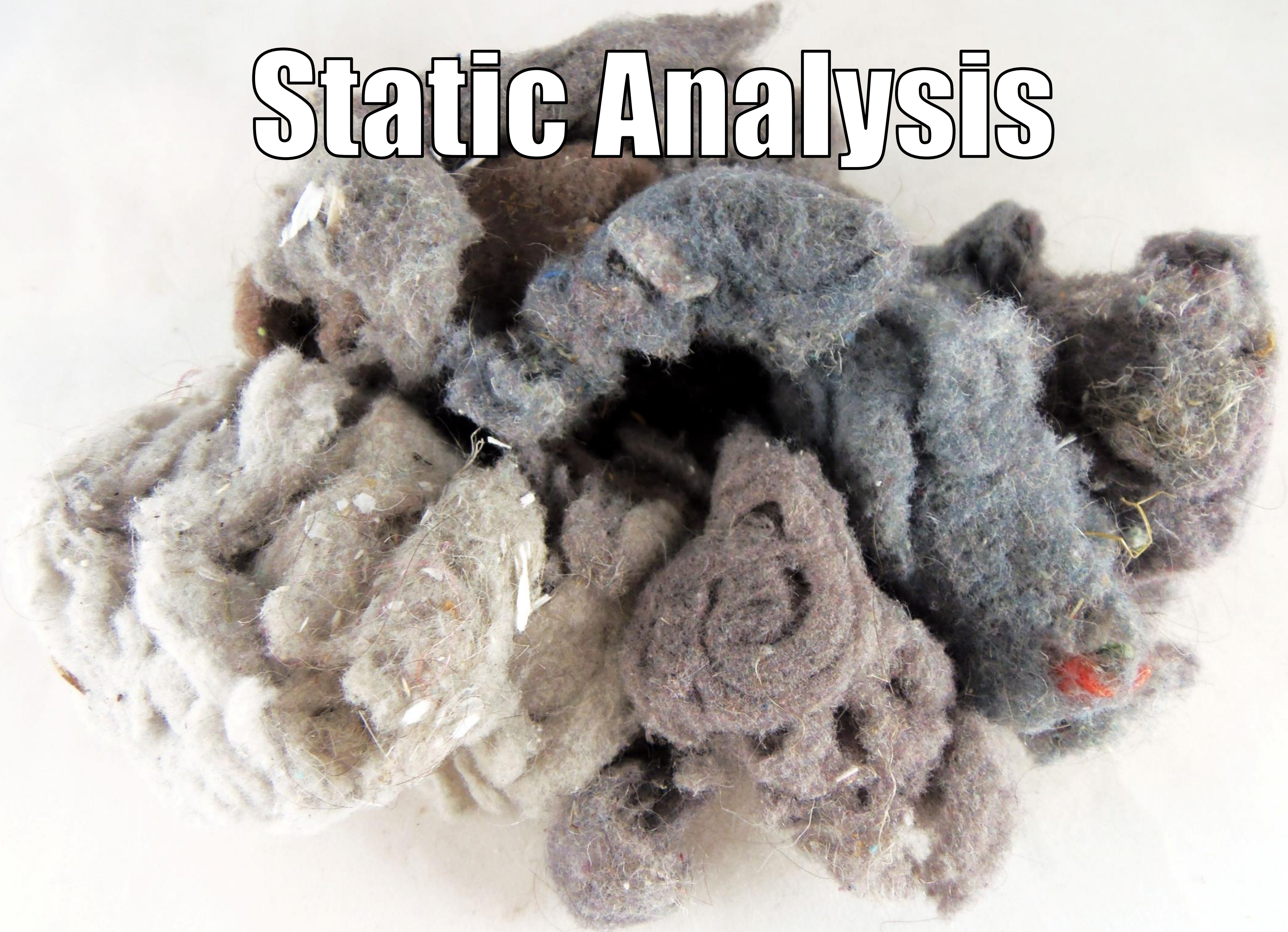
— Steve Johnson

Separate mechanisms from policy.

Dynamic User Memory Allocation

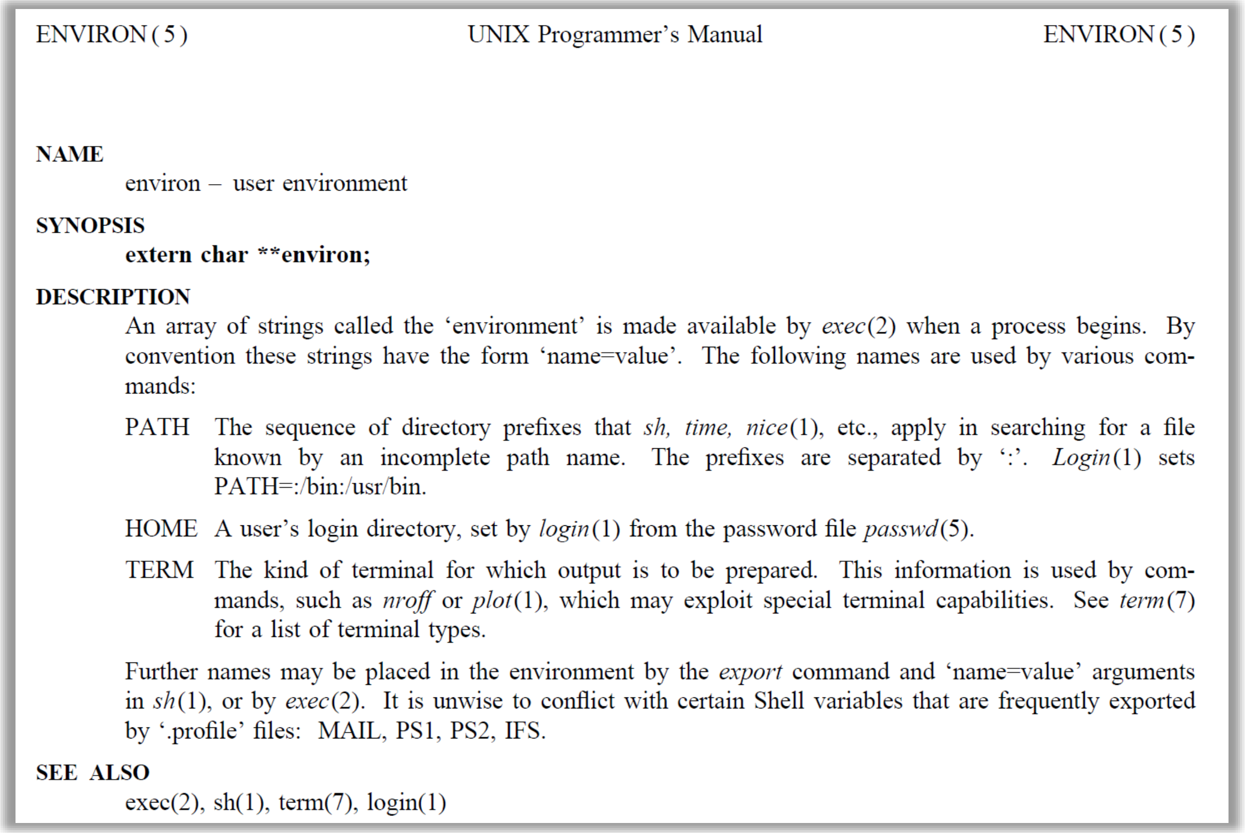
- malloc(3), free(3)
  - Used by 26 programs: awk cc col cron dc dcheck diff ed eqn expr graph icheck learn ls m4 neqn nm quot ratfor spline struct tar tsort uucp xsend quiz
- stdio(3), mp(3)

Static Analysis



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).

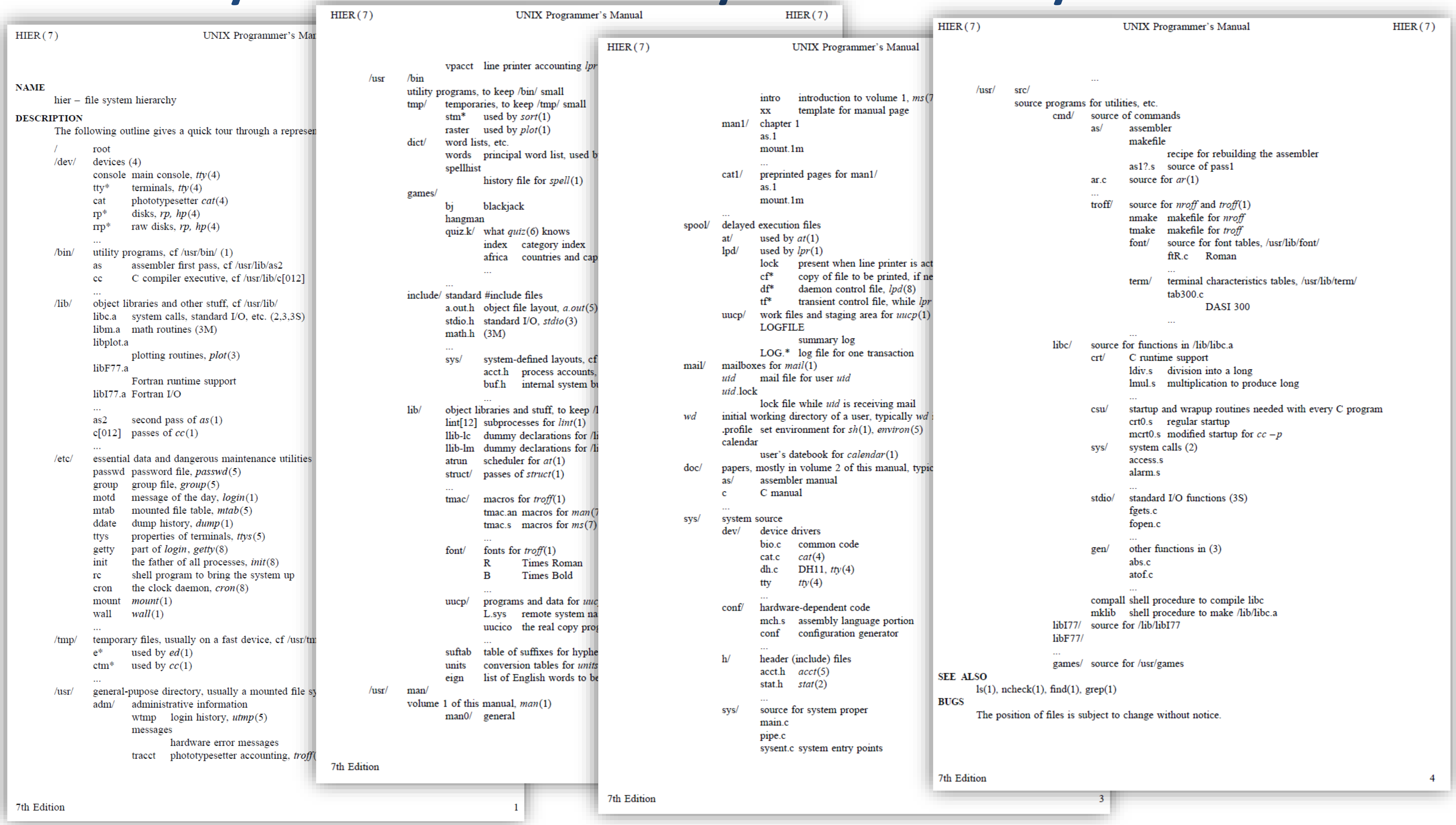
TERM The kind of terminal for which output is to be prepared. This information is used by commands, such as *nroff* or *plot*(1), which may exploit special terminal capabilities. See *term*(7) for a list of terminal types.

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)

Filesystem Directory Hierarchy

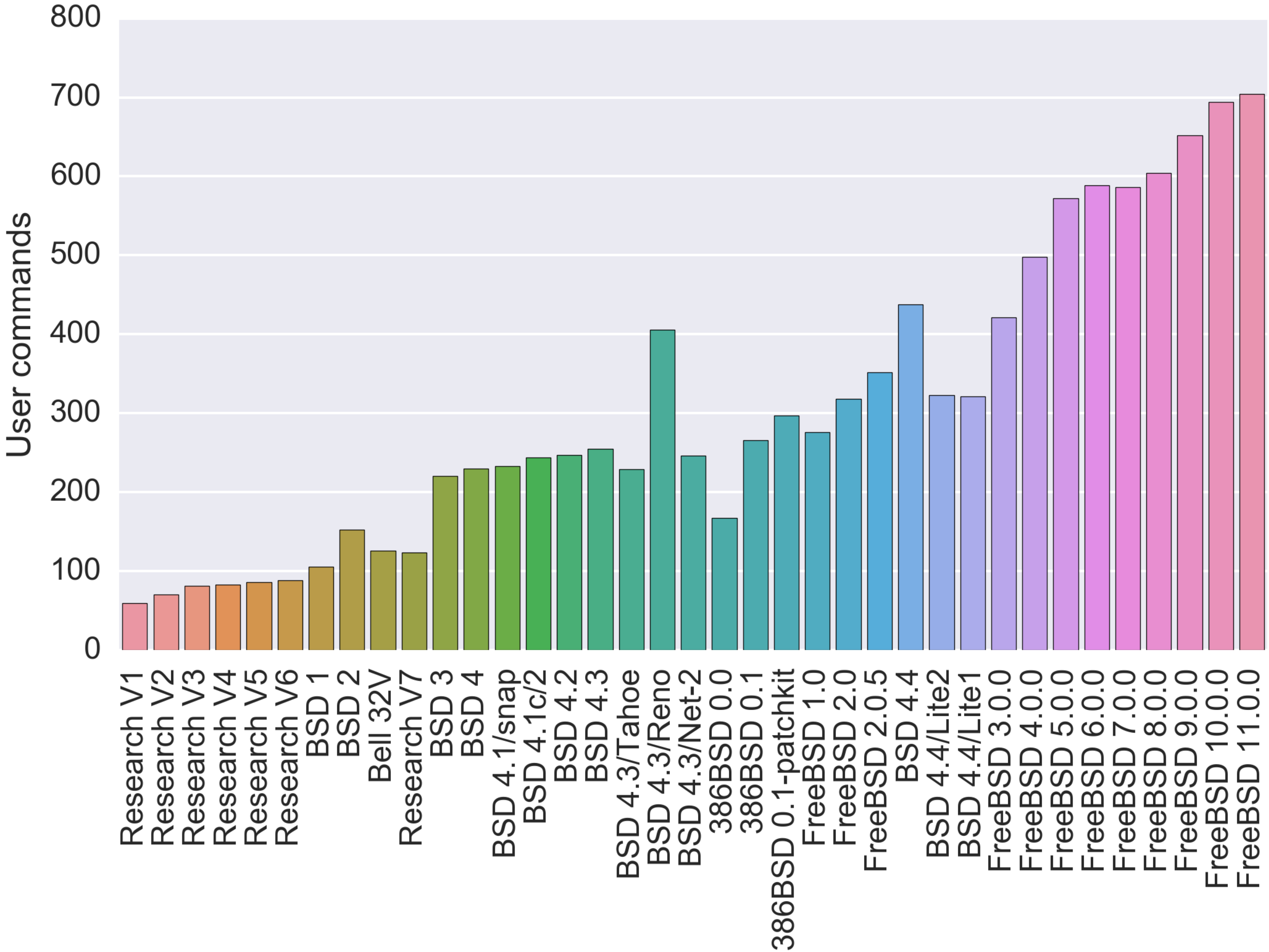


```
HIER ( 7 )                                        UNIX Programmer's Mar

NAME
        hier – file system hierarchy
DESCRIPTION
        The following outline gives a quick tour through a represen
        /         root
        /dev/     devices (4)
                  console  main console, tty(4)
                  tty*     terminals, tty(4)
                  cat      phototypesetter cat(4)
                  rp*      disks, rp, hp(4)
                  rrp*     raw disks, rp, hp(4)
                  ...
        /bin/     utility programs, cf /usr/bin/ (1)
                  as       assembler first pass, cf /usr/lib/as2
                  cc       C compiler executive, cf /usr/lib/c[012]
                  ...
        /lib/     object libraries and other stuff, cf /usr/lib/
                  libc.a   system calls, standard I/O, etc. (2,3,3S)
                  libm.a   math routines (3M)
                  libplot.a
                           plotting routines, plot(3)
                  libF77.a
                           Fortran runtime support
                  libI77.a Fortran I/O
                  ...
                  as2      second pass of as(1)
                  c[012]   passes of cc(1)
                  ...
        /etc/     essential data and dangerous maintenance utilities
                  passwd   password file, passwd(5)
                  group    group file, group(5)
                  motd     message of the day, login(1)
                  mtab     mounted file table, mtab(5)
                  ddate    dump history, dump(1)
                  ttys     properties of terminals, ttys(5)
                  getty    part of login, getty(8)
                  init     the father of all processes, init(8)
                  rc       shell program to bring the system up
                  cron     the clock daemon, cron(8)
                  mount    mount(1)
                  wall     wall(1)
                  ...
        /tmp/     temporary files, usually on a fast device, cf /usr/tm
                  e*       used by ed(1)
                  ctm*     used by cc(1)
                  ...
        /usr/     general-pupose directory, usually a mounted file sy
                  adm/     administrative information
                           wtmp     login history, utmp(5)
                           messages
                                    hardware error messages
                           tracct   phototypesetter accounting, troff(

7th Edition
```

```
HIER ( 7 )                         UNIX Programmer's Manual                    HIER ( 7 )

                  vpacct   line printer accounting lpr
        /usr      /bin
                  utility programs, to keep /bin/ small
                  tmp/     temporaries, to keep /tmp/ small
                           stm*     used by sort(1)
                           raster   used by plot(1)
                  dict/    word lists, etc.
                           words    principal word list, used b
                           spellhist
                                    history file for spell(1)
                  games/
                           bj       blackjack
                           hangman
                           quiz.k/  what quiz(6) knows
                                    index    category index
                                    africa   countries and cap
                                    ...
                           ...
                  include/ standard #include files
                           a.out.h  object file layout, a.out(5)
                           stdio.h  standard I/O, stdio(3)
                           math.h   (3M)
                           ...
                           sys/     system-defined layouts, cf
                                    acct.h   process accounts,
                                    buf.h    internal system bu
                                    ...
                  lib/     object libraries and stuff, to keep /l
                           lint[12] subprocesses for lint(1)
                           llib-lc  dummy declarations for /li
                           llib-lm  dummy declarations for /li
                           atrun    scheduler for at(1)
                           struct/  passes of struct(1)
                           ...
                           tmac/    macros for troff(1)
                                    tmac.an macros for man(7
                                    tmac.s   macros for ms(7)
                                    ...
                           font/    fonts for troff(1)
                                    R        Times Roman
                                    B        Times Bold
                                    ...
                           uucp/    programs and data for uuc
                                    L.sys    remote system na
                                    uucico   the real copy prog
                                    ...
                           suftab   table of suffixes for hyphe
                           units    conversion tables for units
                           eign     list of English words to be
        /usr/     man/
                  volume 1 of this manual, man(1)
                           man0/    general

7th Edition                                                 1
```

```
HIER ( 7 )                                   UNIX Programmer's Manual

                           intro    introduction to volume 1, ms(7
                           xx       template for manual page
                  man1/    chapter 1
                           as.1
                           mount.1m
                           ...
                  cat1/    preprinted pages for man1/
                           as.1
                           mount.1m
                  ...
        spool/    delayed execution files
                  at/      used by at(1)
                  lpd/     used by lpr(1)
                           lock     present when line printer is act
                           cf*      copy of file to be printed, if ne
                           df*      daemon control file, lpd(8)
                           tf*      transient control file, while lpr
                  uucp/    work files and staging area for uucp(1)
                           LOGFILE
                                    summary log
                           LOG.*    log file for one transaction
        mail/     mailboxes for mail(1)
                  uid      mail file for user uid
                  uid.lock
                           lock file while uid is receiving mail
        wd        initial working directory of a user, typically wd i
                  .profile set environment for sh(1), environ(5)
                  calendar
                           user's datebook for calendar(1)
        doc/      papers, mostly in volume 2 of this manual, typic
                  as/      assembler manual
                  c        C manual
                  ...
        sys/      system source
                  dev/     device drivers
                           bio.c    common code
                           cat.c    cat(4)
                           dh.c     DH11, tty(4)
                           tty      tty(4)
                           ...
                  conf/    hardware-dependent code
                           mch.s    assembly language portion
                           conf     configuration generator
                           ...
                  h/       header (include) files
                           acct.h   acct(5)
                           stat.h   stat(2)
                           ...
                  sys/     source for system proper
                           main.c
                           pipe.c
                           sysent.c system entry points

7th Edition                                                     3
```

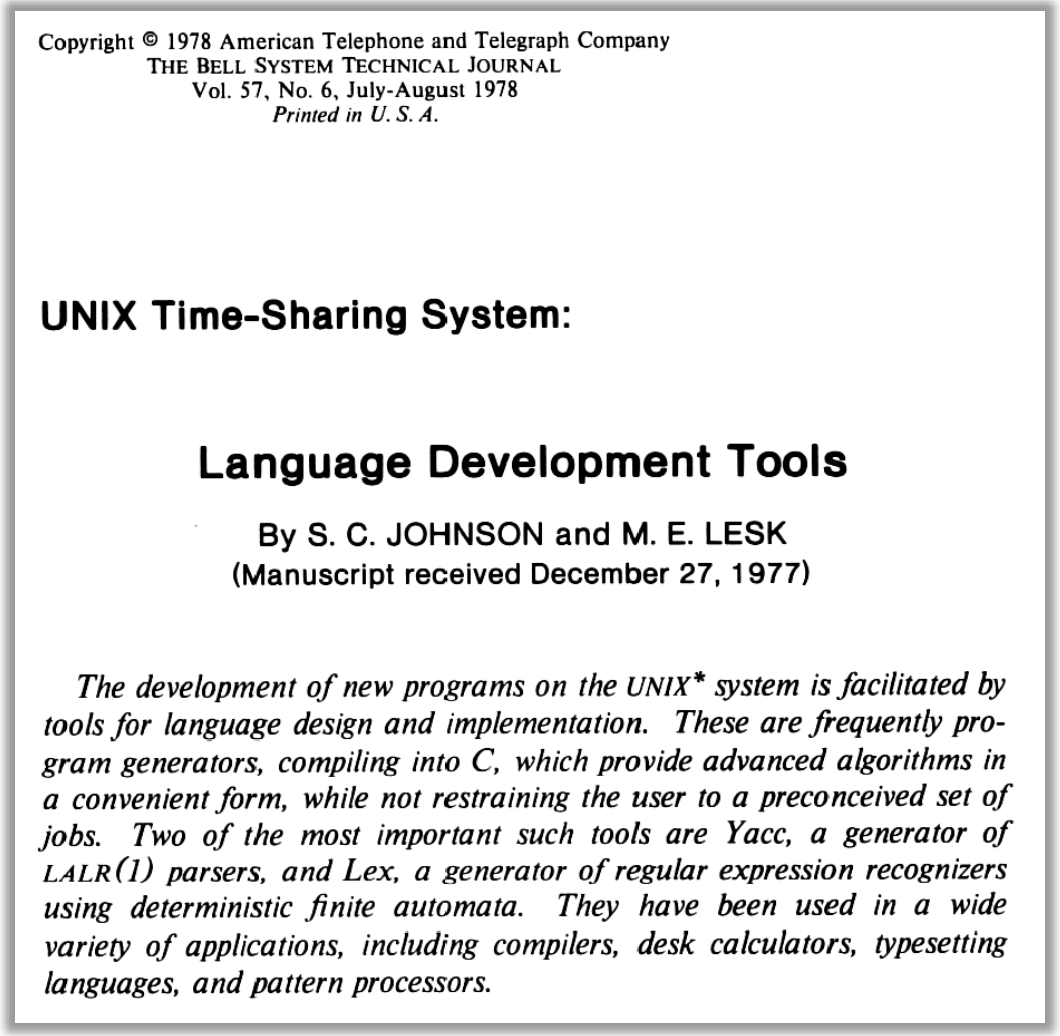
```
HIER ( 7 )                        UNIX Programmer's Manual                      HIER ( 7 )

                                    ...
        /usr/     src/
                  source programs for utilities, etc.
                           cmd/     source of commands
                                    as/      assembler
                                             makefile
                                                      recipe for rebuilding the assembler
                                             as1?.s   source of pass1
                                    ar.c     source for ar(1)
                                    ...
                                    troff/   source for nroff and troff(1)
                                             nmake    makefile for nroff
                                             tmake    makefile for troff
                                             font/    source for font tables, /usr/lib/font/
                                                      ftR.c    Roman
                                                      ...
                                             term/    terminal characteristics tables, /usr/lib/term/
                                                      tab300.c
                                                               DASI 300
                                                      ...
                                             ...
                           libc/    source for functions in /lib/libc.a
                                    crt/     C runtime support
                                             ldiv.s   division into a long
                                             lmul.s   multiplication to produce long
                                             ...
                                    csu/     startup and wrapup routines needed with every C program
                                             crt0.s   regular startup
                                             mcrt0.s  modified startup for cc –p
                                    sys/     system calls (2)
                                             access.s
                                             alarm.s
                                             ...
                                    stdio/   standard I/O functions (3S)
                                             fgets.c
                                             fopen.c
                                             ...
                                    gen/     other functions in (3)
                                             abs.c
                                             atof.c
                                             ...
                                    compall  shell procedure to compile libc
                                    mklib    shell procedure to make /lib/libc.a
                           libI77/  source for /lib/libI77
                           libF77/
                           ...
                           games/   source for /usr/games
SEE ALSO
        ls(1), ncheck(1), find(1), grep(1)
BUGS
        The position of files is subject to change without notice.

7th Edition                                                                        4
```


User commands
0
100
200
300
400
500
600
700
800
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

Write abstract programs that generate code instead of writing code by hand.

Language Development Tools

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



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.*

Raise abstraction through DSLs.

Domain-Specific Languages

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

Architectural innovations are sticky and face increasing resistance.

BUILDERS OF BERKELEY

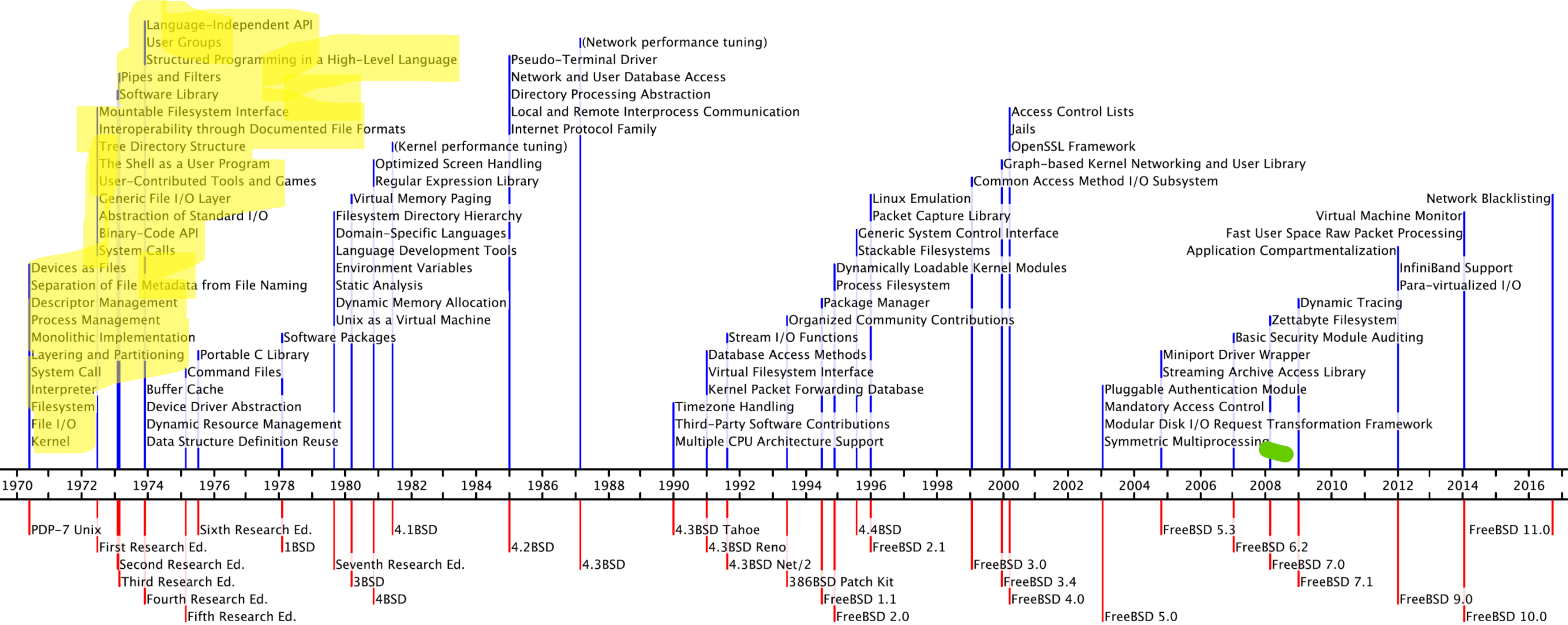
Regular Expression Library: regex(3)

- 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

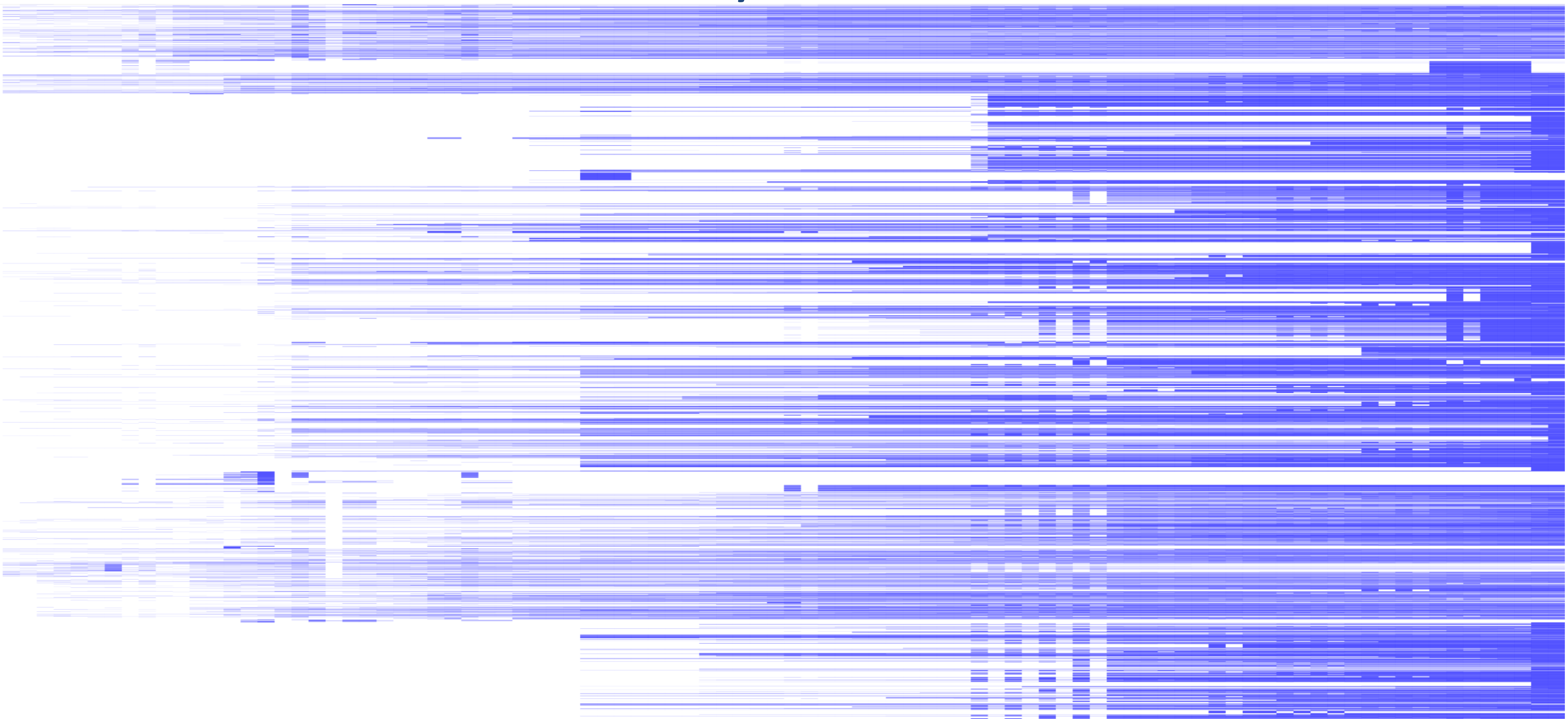
4.3BSD Net/2 (Jun 1991)

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

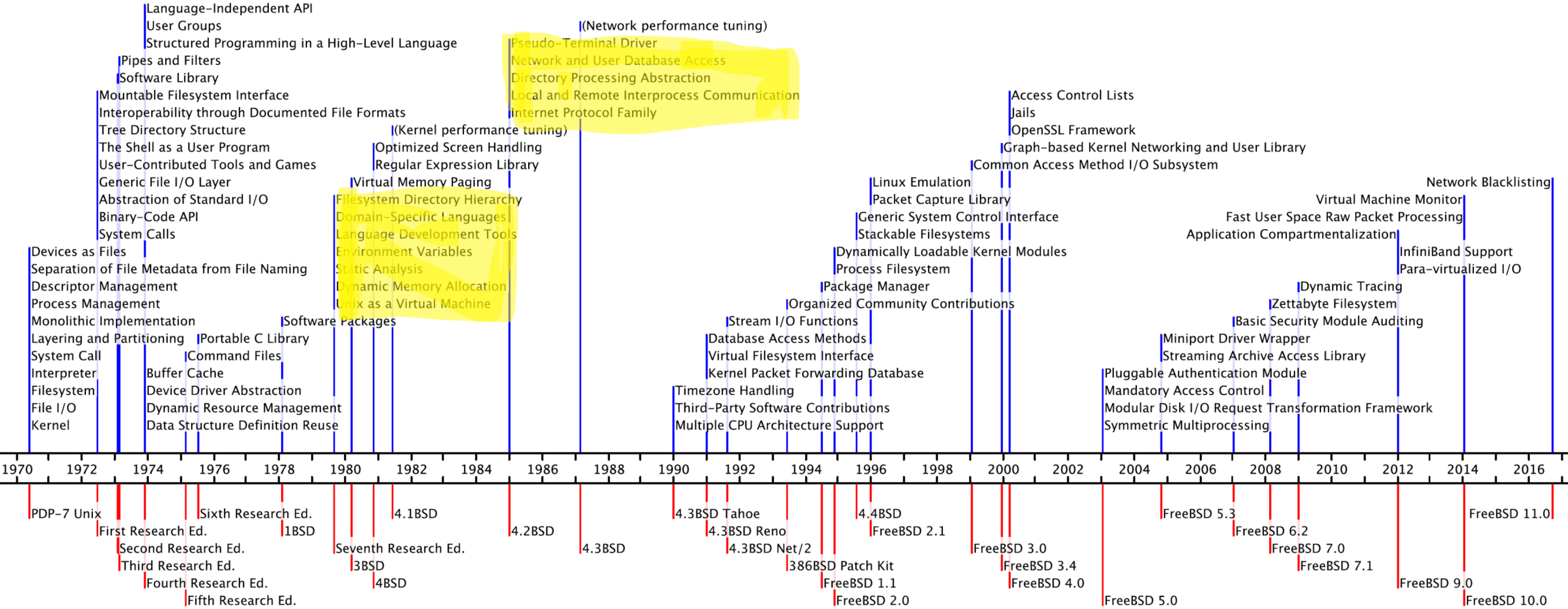
Many core architecture decisions are taken at the beginning of the system's lifetime



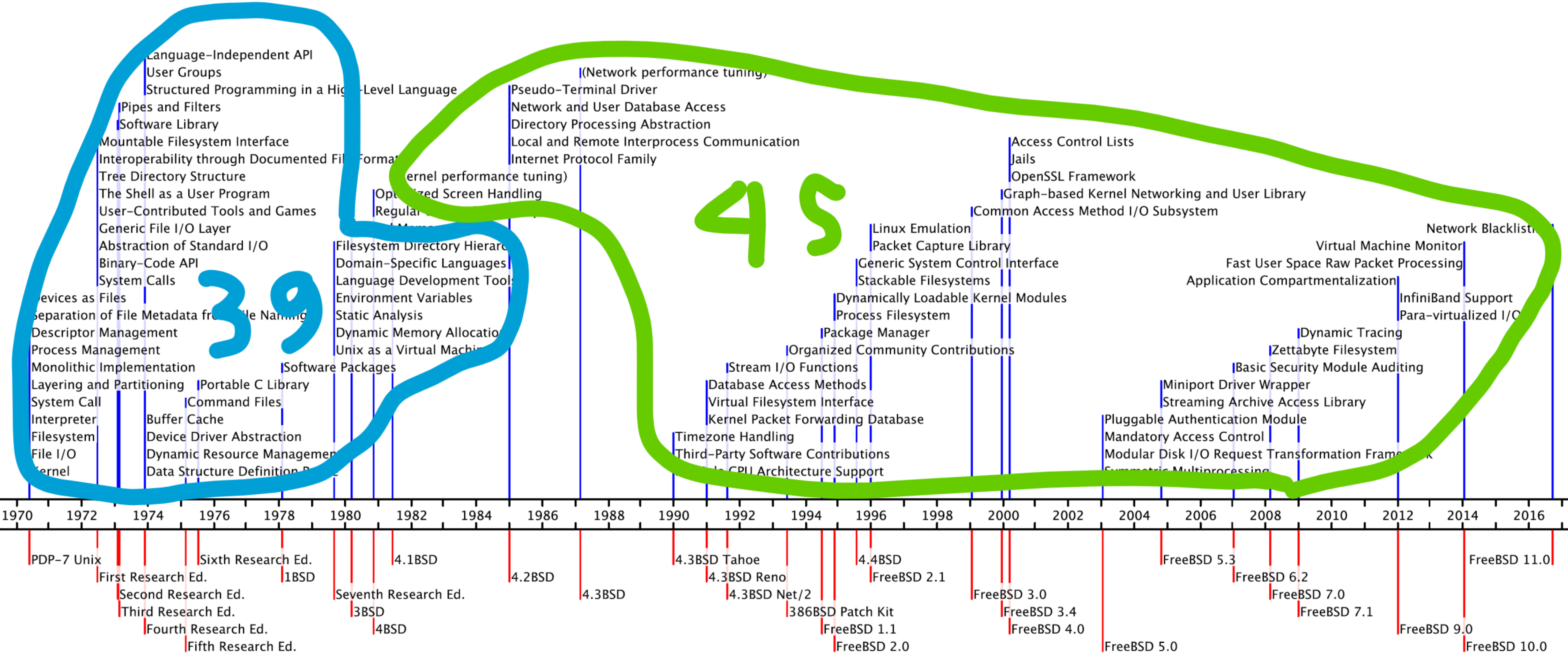
Most architecture decisions survive over the system lifetime

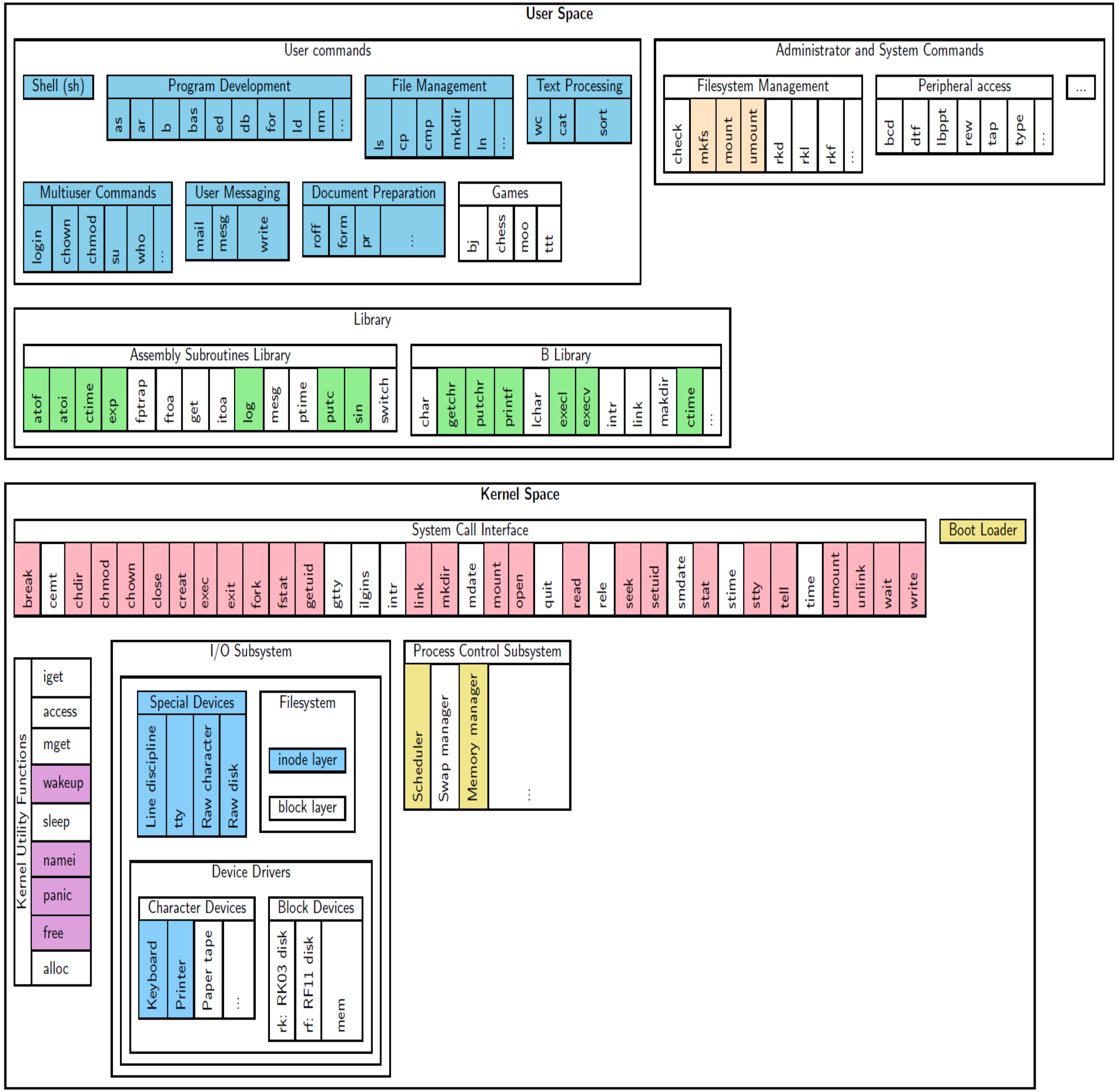


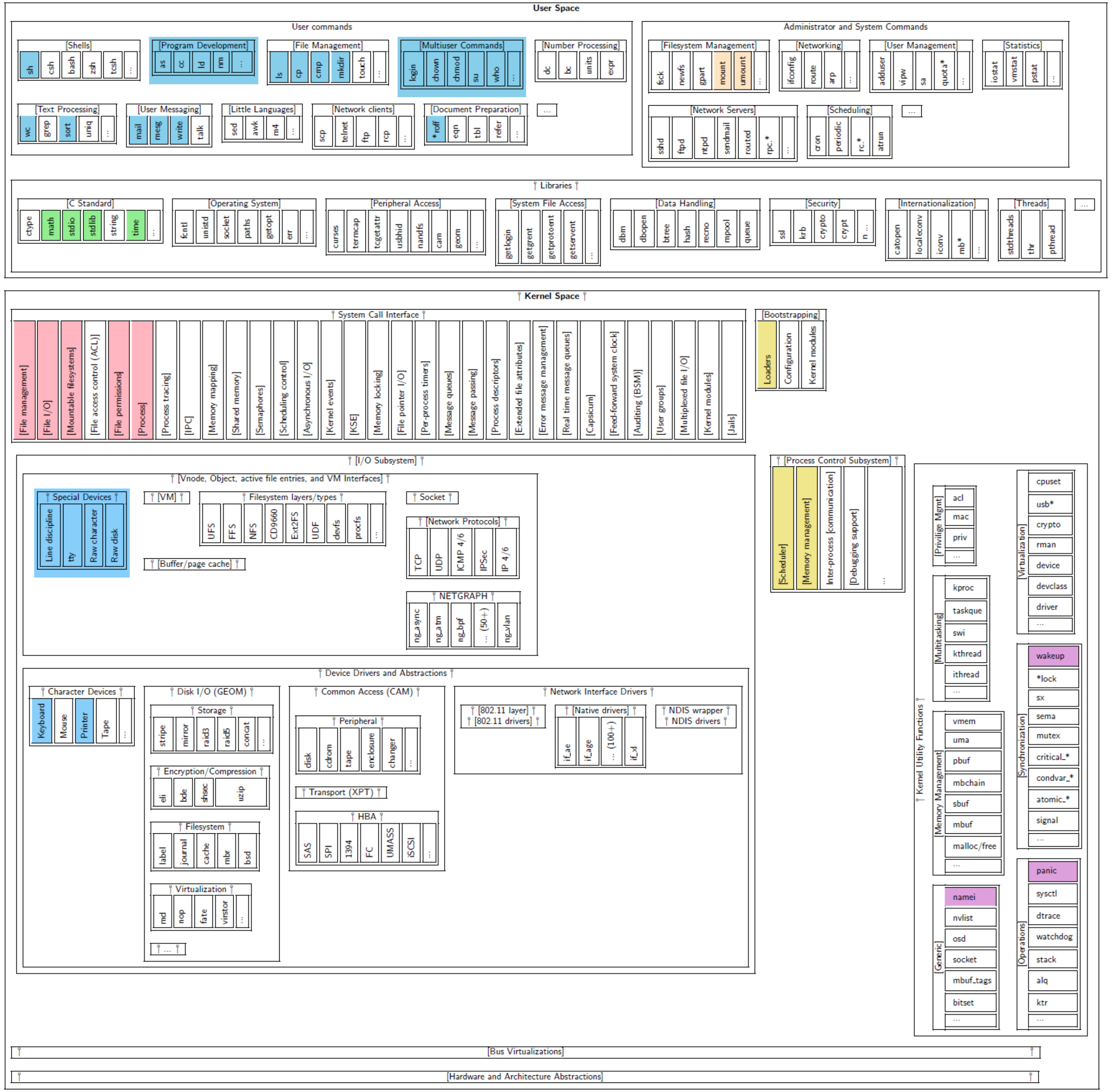
New architecture decisions are continuously made, further fueling architecture evolution



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




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
...


User Space
User commands
[Shells]
sh
csh
bash
zsh
tcsh
...
[Program Development]
as
cc
ld
nm
...
[File Management]
ls
cp
cmp
mkdir
touch
...
[Multiuser Commands]
login
chown
chmod
su
who
...
[Number Processing]
dc
bc
units
expr
[Text Processing]
wc
grep
sort
uniq
...
[User Messaging]
mail
mesg
write
talk
[Little Languages]
sed
awk
m4
...
[Network clients]
scp
telnet
ftp
rcp
...
[Document Preparation]
*roff
eqn
tbl
refer
...
...
Administrator and System Commands
[Filesystem Management]
fsck
newfs
gpart
mount
umount
...
[Networking]
ifconfig
route
arp
...
[User Management]
adduser
vipw
sa
quota*
...
[Statistics]
iostat
vmstat
pstat
...
[Network Servers]
sshd
ftpd
ntpd
sendmail
routed
rpc.*
...
[Scheduling]
cron
periodic
rc.*
atrun
...
Libraries
[C Standard]
ctype
math
stdio
stdlib
string
time
...
[Operating System]
fcntl
unistd
socket
paths
getopt
err
...
[Peripheral Access]
curses
termcap
tcgetattr
usbhid
nandfs
cam
geom
...
[System File Access]
getlogin
getgrent
getprotoent
getservent
...
[Data Handling]
dbm
dbopen
btree
hash
recno
mpool
queue
[Security]
ssl
krb
crypto
crypt
n...
[Internationalization]
catopen
localeconv
iconv
mb*
...
[Threads]
stdthreads
thr
pthread
...
Kernel Space
System Call Interface
[File management]
[File I/O]
[Mountable filesystems]
[File access control (ACL)]
[File permissions]
[Process]
[Process tracing]
[IPC]
[Memory mapping]
[Shared memory]
[Semaphores]
[Scheduling control]
[Asynchronous I/O]
[Kernel events]
[KSE]
[Memory locking]
[File pointer I/O]
[Per-process timers]
[Message queues]
[Message passing]
[Process descriptors]
[Extended file attributes]
[Error message management]
[Real time message queues]
[Capsicum]
[Feed-forward system clock]
[Auditing (BSM)]
[User groups]
[Multiplexed file I/O]
[Kernel modules]
[Jails]
[Bootstrapping]
Loaders
Configuration
Kernel modules
[I/O Subsystem]
[Vnode, Object, active file entries, and VM Interfaces]
Special Devices
Line discipline
tty
Raw character
Raw disk
[VM]
Filesystem layers/types
UFS
FFS
NFS
CD9660
Ext2FS
UDF
devfs
procfs
...
Socket
[Network Protocols]
TCP
UDP
ICMP 4/6
IPSec
IP 4/6
[Buffer/page cache]
NETGRAPH
ng_async
ng_atm
ng_bpf
... (50+)
ng_vlan
[Process Control Subsystem]
[Scheduler]
[Memory management]
Inter-process [communication]
[Debugging support]
...
Device Drivers and Abstractions
Character Devices
Keyboard
Mouse
Printer
Tape
...
Disk I/O (GEOM)
Storage
stripe
mirror
raid3
raid5
concat
...
Encryption/Compression
eli
bde
shsec
uzip
Filesystem
label
journal
cache
mbr
bsd
Virtualization
md
nop
gate
virstor
...
...
Common Access (CAM)
Peripheral
disk
cdrom
tape
enclosure
changer
...
Transport (XPT)
HBA
SAS
SPI
1394
FC
UMASS
iSCSI
...
Network Interface Drivers
[802.11 layer]
[802.11 drivers]
[Native drivers]
if_ae
if_age
... (100+)
if_xl
NDIS wrapper
NDIS drivers
Kernel Utility Functions
[Privilege Mgmt]
acl
mac
priv
...
[Multitasking]
kproc
taskque
swi
kthread
ithread
...
[Memory Management]
vmem
uma
pbuf
mbchain
sbuf
mbuf
malloc/free
...
[Generic]
namei
nvlist
osd
socket
mbuf_tags
bitset
...
[Virtualization]
cpuset
usb*
crypto
rman
device
devclass
driver
...
[Synchronization]
wakeup
*lock
sx
sema
mutex
critical_*
condvar_*
atomic_*
signal
...
[Operations]
panic
sysctl
dtrace
watchdog
stack
alq
ktr
...
[Bus Virtualizations]
[Hardware and Architecture Abstractions]

Package managers grow ecosystems and communities.

386
BSD
REFERENCE CD-ROM
Official 386BSD Release 1.0
William &
Lynne Jolitz
DR. DOBB'S CD-ROM LIBRARY

386BSD Patch Kit (1992-1993)

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

FreeBSD®

FreeBSD 1.1 (May 1994)

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

Package ecosystems

Operating system

- GNU/Linux: Debian, Fedora, Ubuntu, …
- FreeBSD, NetBSD, OpenBSD

Package repository / manager

- Maven / mvn, Gradle
- PyPI / pip
- NPM / npm, yarn
- CRAN
- RubyGems / gem

Activity

- Data science (Python Anaconda)

ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ

ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

TU Delft

Thank you!

🌐 www.spinellis.gr

𝕏 @CoolSWEng

@CoolSWEng@mastodon.acm.org

✉ dds@aueb.gr

Business
Analytics Lab

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

Image Credits

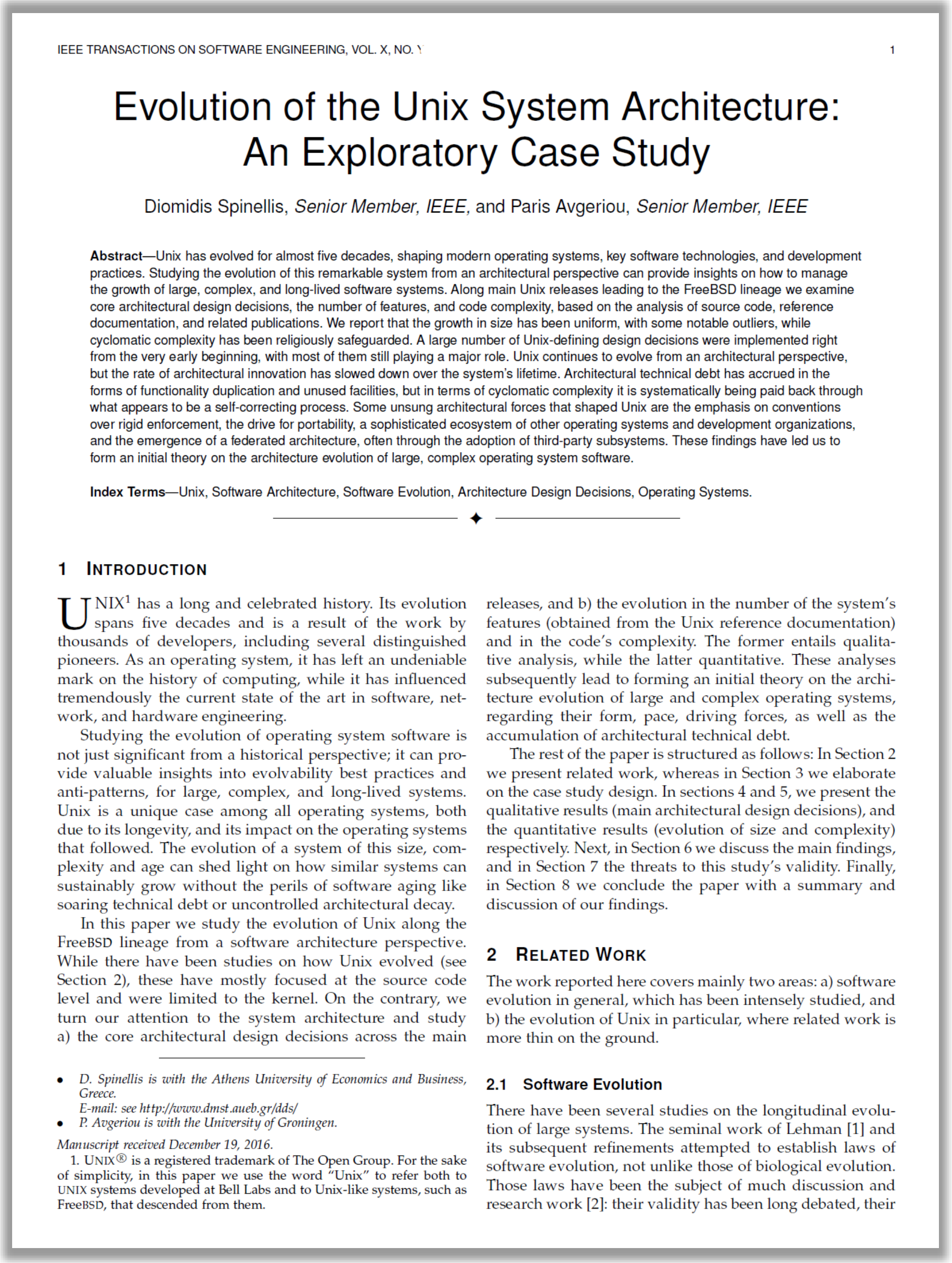
- Faces of Open Source / Peter Adams
- Data: Joshua Sortino
- Hackers at Junction 2015: Vmuru
- ASR-33 Teletype: Rama & Musée Bolo
- VT100: Jason Scott
- PDP 11/20: Image courtesy of Computer History Museum

(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

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.



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

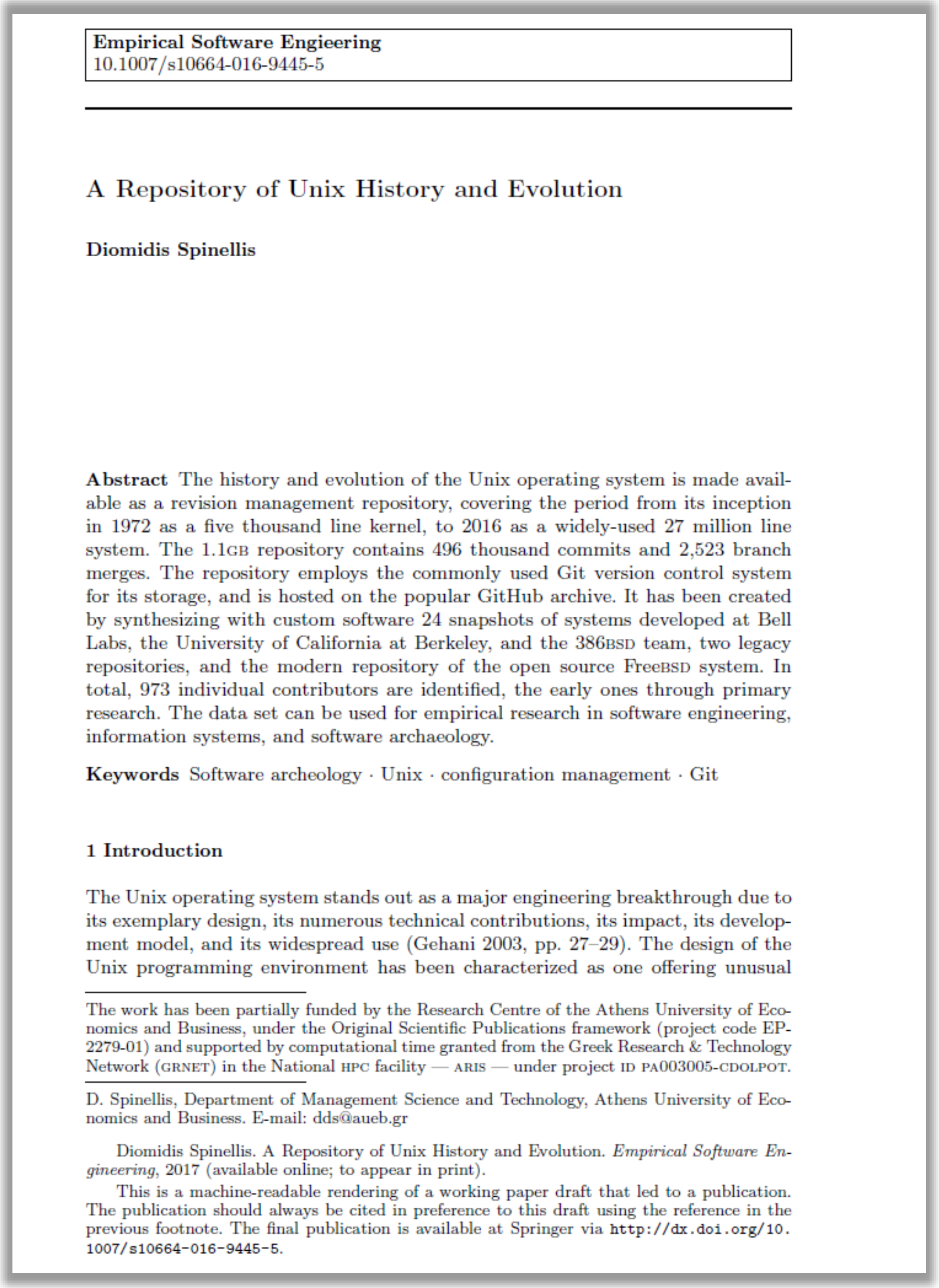
There have been several studies on the longitudinal evolution of large systems. The seminal work of Lehman [1] and its subsequent refinements attempted to establish laws of software evolution, not unlike those of biological evolution. Those laws have been the subject of much discussion and research work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



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

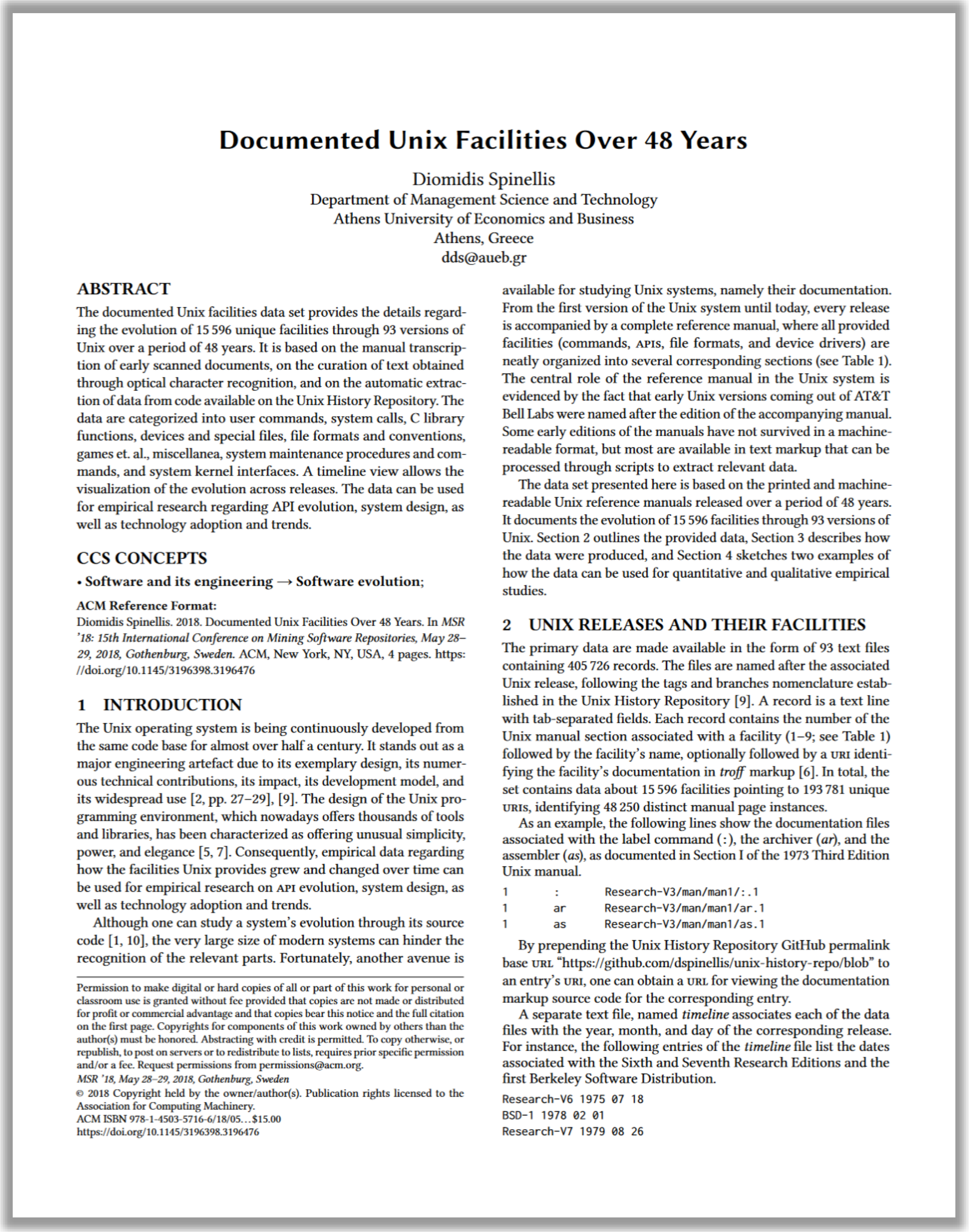
The work has been partially funded by the Research Centre of the Athens University of Economics and Business, under the Original Scientific Publications framework (project code EP-2279-01) and supported by computational time granted from the Greek Research & Technology Network (GRNET) in the National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

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.

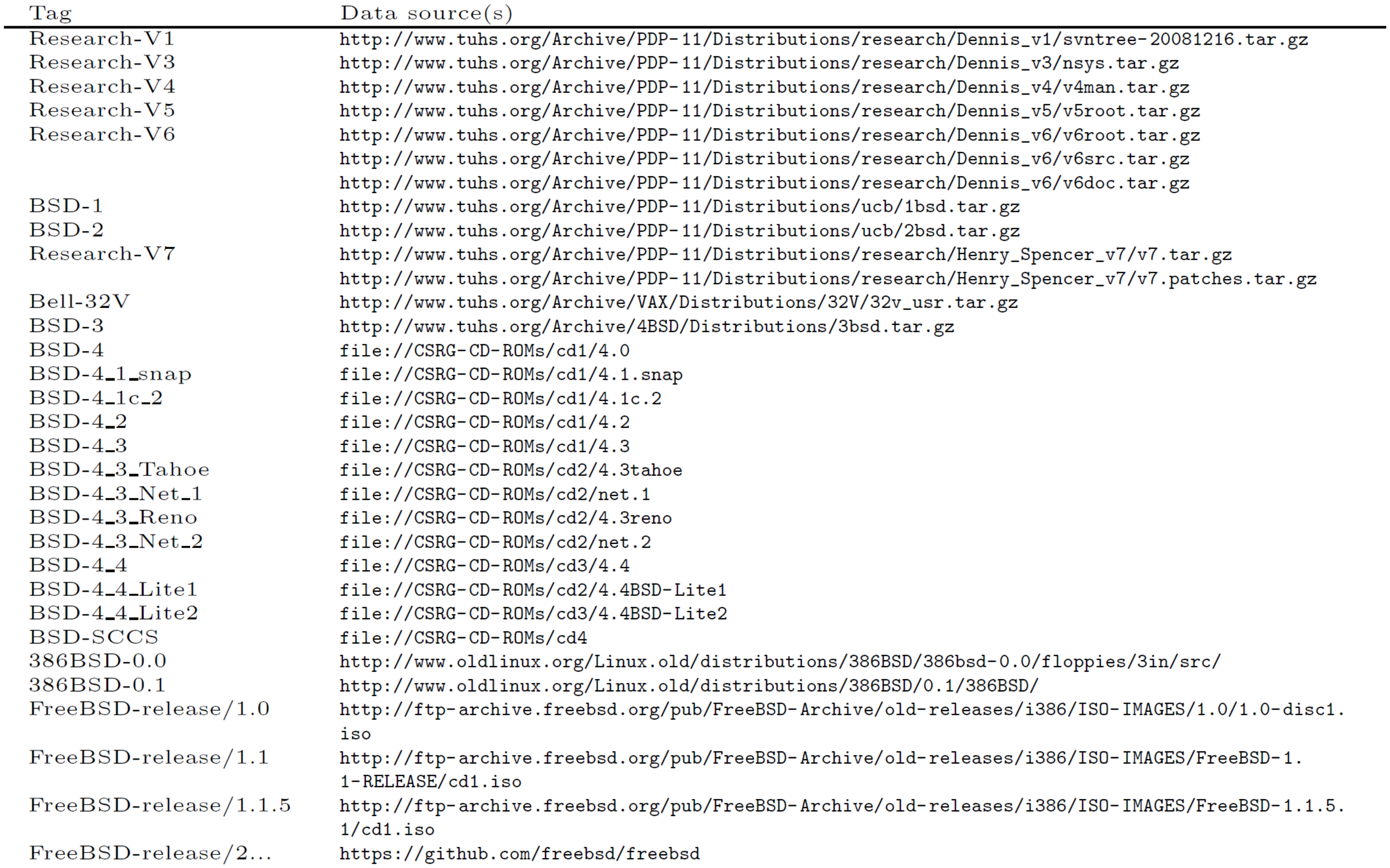
```
Research-V6 1975 07 18
BSD-1 1978 02 01
Research-V7 1979 08 26
```

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

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