第 6 讲：The Programming Languages of OS

第二节：The Evolution of C Programming Practices: A Study of the Unix Operating System 1973–2015

陈渝

清华大学计算机系

yuchen@tsinghua.edu.cn

2020 年 3 月 22 日

History of C

- 1969: B created, based on BCPL, to replace PDP-7 assembler as the system programming language for Unix, remained a typeless language like BCPL
- 1971: NB (”new B”) created when porting B to PDP-11, types (int, char, arrays and pointers), array-to-pointer conversion, compilation to machine code
- 1972: Language renamed to C, structs, preprocessor, portable I/O
- 1973: Unix re-written in C
- 1978: The C Programming Language, 1st edition

The Development of the C Language, Dennis M. Ritchie,1993

Why was C be used to develop UNIX?

- Thompson decided that Unix possibly needed a system programming language, he created a language B(from BCPL). B can be thought of as C without types;
- BCPL, B, and C all fit firmly in the traditional procedural family typified by Fortran and Algol 60. They are **'close to the machine' abstractions**.
- BCPL, B have a single data type, the 'word,' or 'cell,' a fixed-length bit pattern. Memory in these languages consists of a linear array of such cells. B generated 'threaded code'.
- The C extended the B by adding types and also rewrote its compiler to generate PDP-11 machine instructions.
- The C compiler is capable of producing programs fast and small enough to compete with assembly language.
- This DEC VAX 11/780 machine became much more popular.

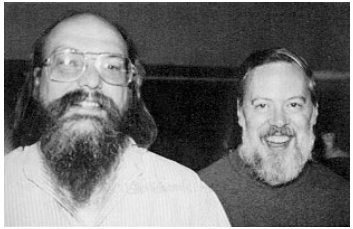
Why was C be used to develop UNIX?
C was a traditional procedural family language

- SPEED CLOSE TO ASSEMBLY
- 'CLOSE TO MACHINE' ABSTRACTION
- TYPE SAFETY
- PORTABILITY
- SIMPLE/SMALL LANG & BIG LIBRARY

Whence Success?

- The success of Unix itself was the most important factor
- Remains a simple and small language
- At the same time the language is sufficiently abstracted from machine details that program portability can be achieved
- C and its central library support always remained in touch with a real environment
- The actual C language as seen by millions of users using many different compilers has remained remarkably stable and unified compared to those of similarly widespread currency,for example Pascal and Fortran.

THE C PROGRAMMING LANGUAGE



objective

The objective of this work is to study the long term evo- lution of C programming in the context of the Unix oper- ating system development.

Formulate seven hypotheses associated with the long term evolution of C programming in the Unix operating system

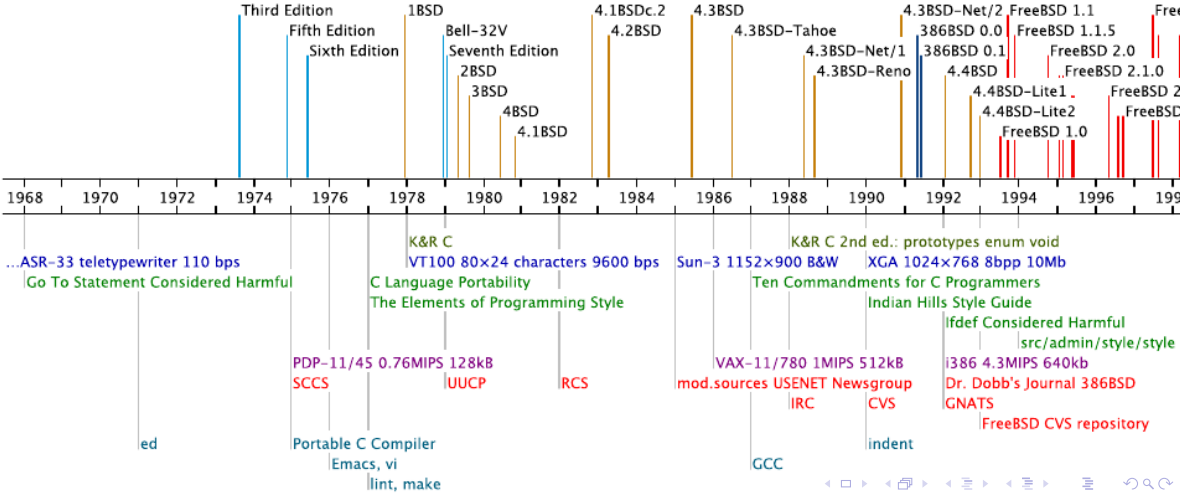
A Study of the Unix Operating System 1973–2015

Seven hypotheses associated with the long term evolution of C programming in the Unix operating system

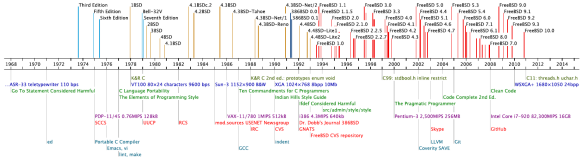
- Programming practices reflect technology affordances
- Modularity increases with code size
- New language features are increasingly used to saturation point
- Programmers trust the compiler for register allocation
- Code formatting practices converge to a common standard
- Software complexity evolution follows self correction feedback mechanisms
- Code readability increases

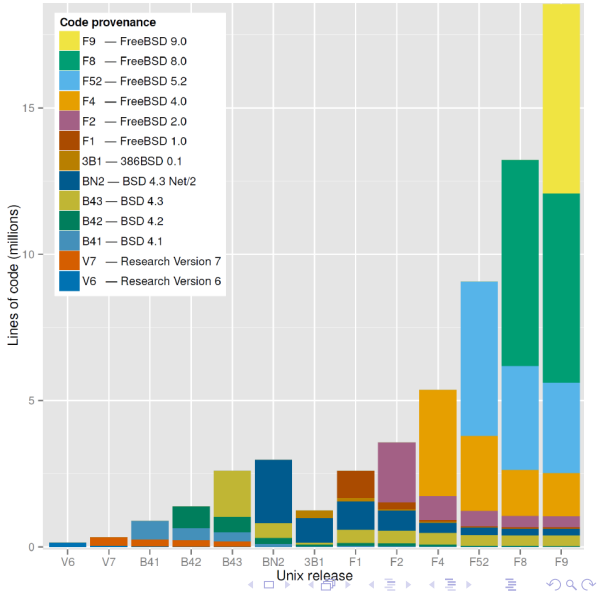
A Study of the Unix Operating System 1973–2015

Timeline of indicative analyzed revisions and milestones in (from top to bottom): C language evolution, developer interfaces, programming guidelines, processing capacity, collaboration mechanisms, and tools.



A Study of the Unix Operating System 1973–2015



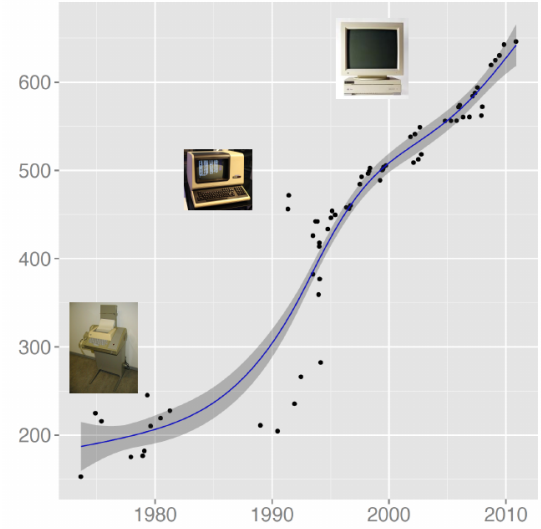


A Study of the Unix Operating System 1973–2015

H1: Programming practices reflect technology affordances

Increase in mean file length

(lines / file)

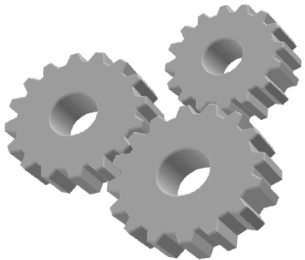


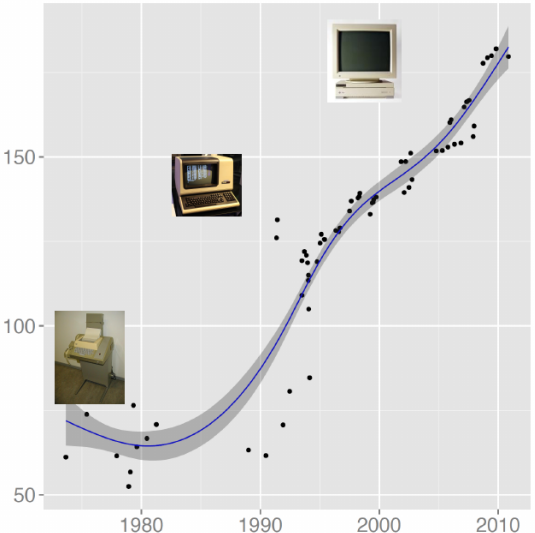
A Study of the Unix Operating System 1973–2015

H1: Programming practices reflect technology affordances

Increase in mean file functionality

(statements / file)



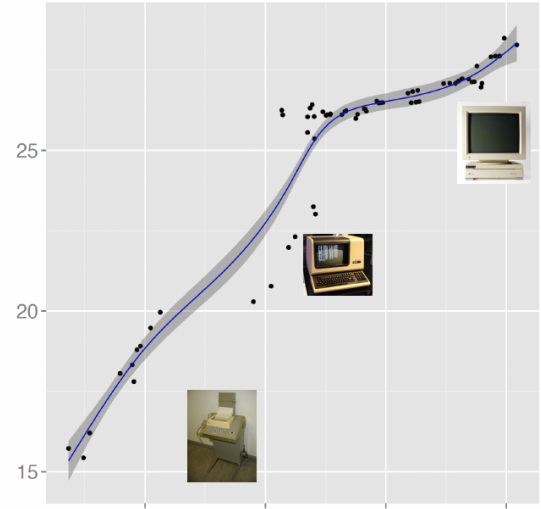


A Study of the Unix Operating System 1973–2015

H1: Programming practices reflect technology affordances

Increase in mean line length

(characters / line)



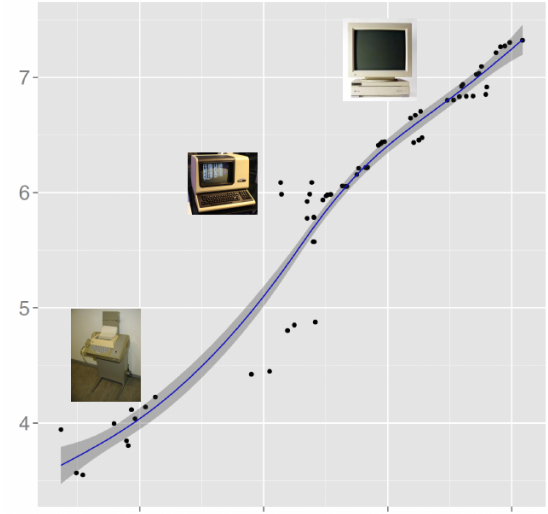
A Study of the Unix Operating System 1973–2015

H1: Programming practices reflect technology affordances

Increase in mean identifier length

(characters / line)

int creat();



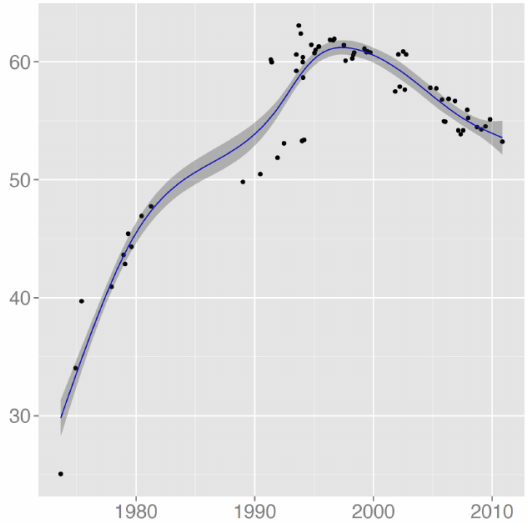
A Study of the Unix Operating System 1973–2015

H1: Programming practices reflect technology affordances

Increase in mean function length (lines / function)

{

}

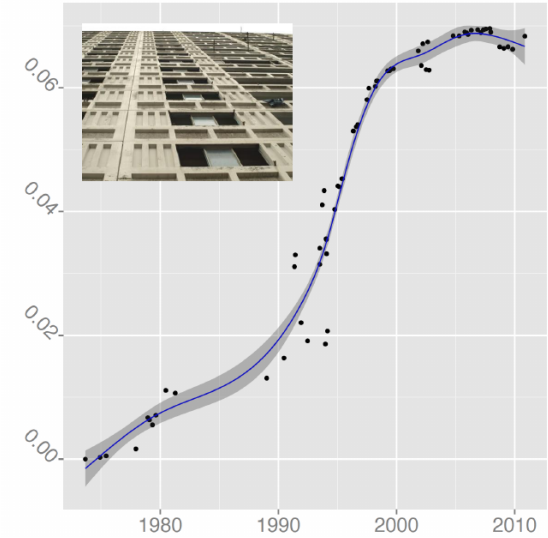


Half Century of Unix: History, Preservation, and Lessons Learned,Diomidis Spinellis, keynote of OW2 Consortium, 2017

H2: Modularity increases with code size

Increase in number of **static** declarations / statement

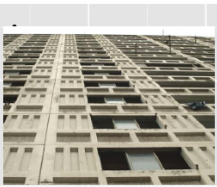
static short splice;



H2: Modularity increases with code size

Increase in number of **#include** directives / line

#include "if_uba.h"

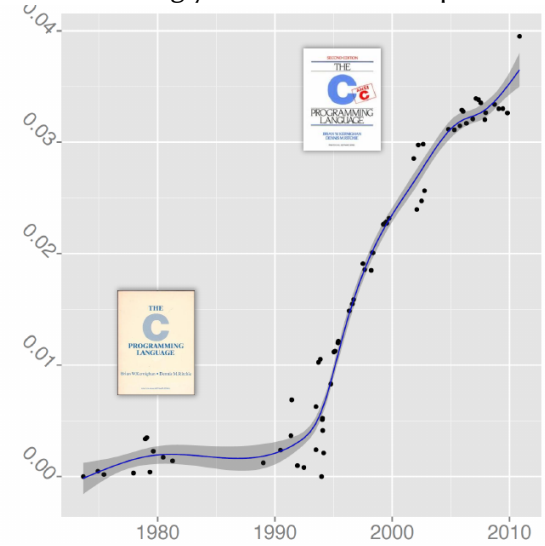


A Study of the Unix Operating System 1973–2015

H3: New language features are increasingly used to saturation point

Increase in number of **const** declarations / statement

const char *panicstr;

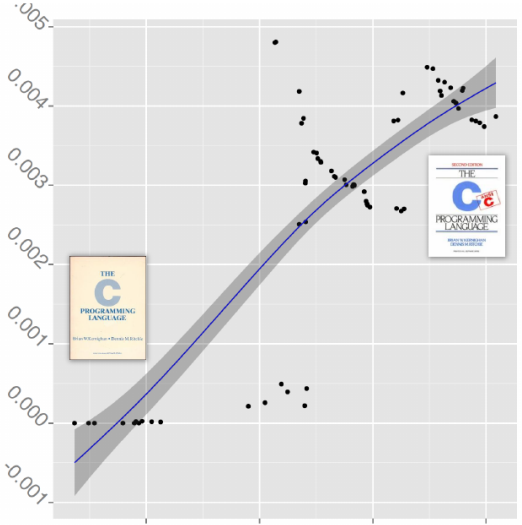


A Study of the Unix Operating System 1973–2015

H3: New language features are increasingly used to saturation point

Increase in number of **enum** declarations / statement

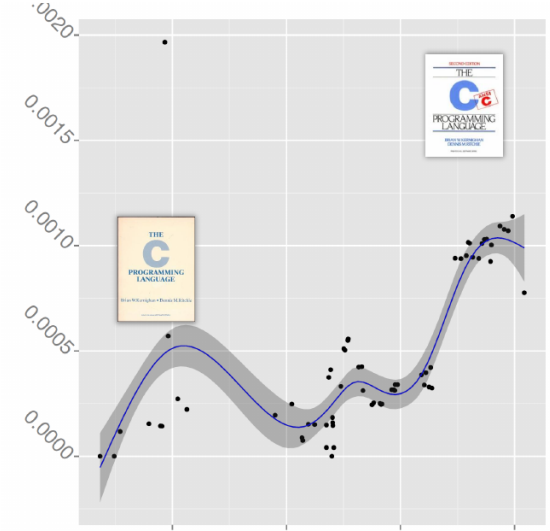
enum uio_rw rw;



H3: New language features are increasingly used to saturation point

Increase in number of **inline** declarations / statement

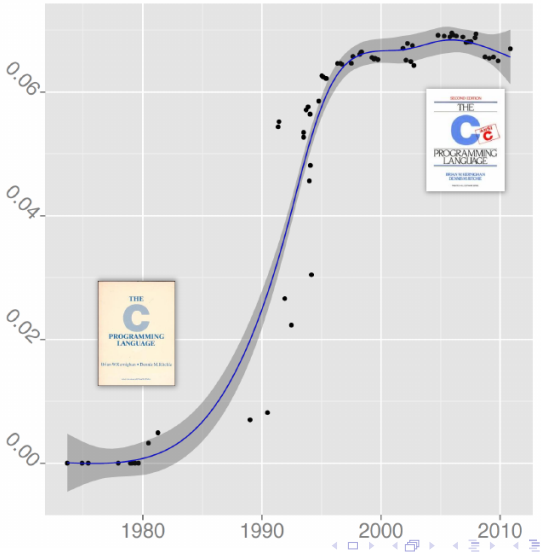
inline uchar get_byte ();



H3: New language features are increasingly used to saturation point

Increase in number of **void** declarations / statement

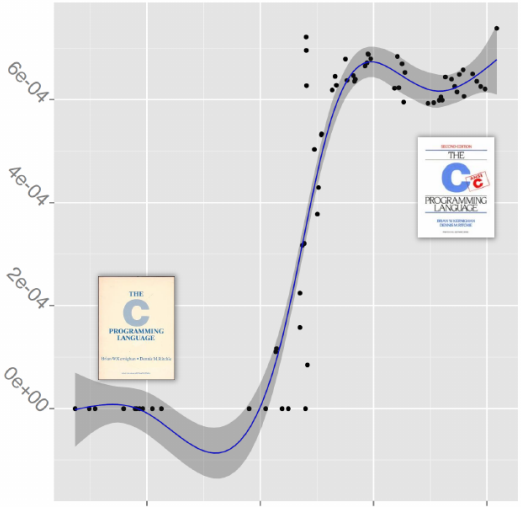
sc_max_unit(**void**)



H3: New language features are increasingly used to saturation point

Increase in number of **volatile** declarations / statement

volatile struct proc *p, *pp;

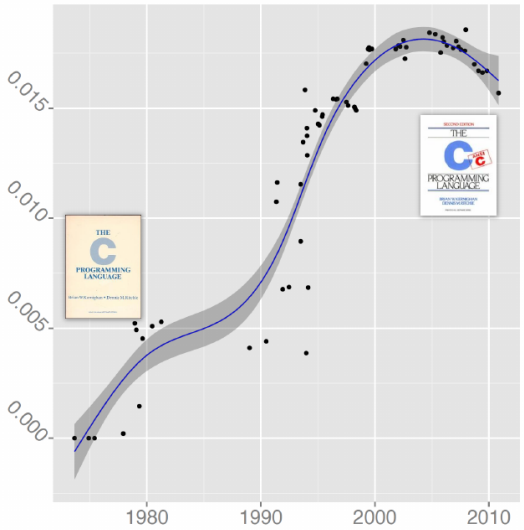


A Study of the Unix Operating System 1973–2015

H3: New language features are increasingly used to saturation point

Increase in number of **unsigned** declarations / statement

unsigned c[BMAX + 1];

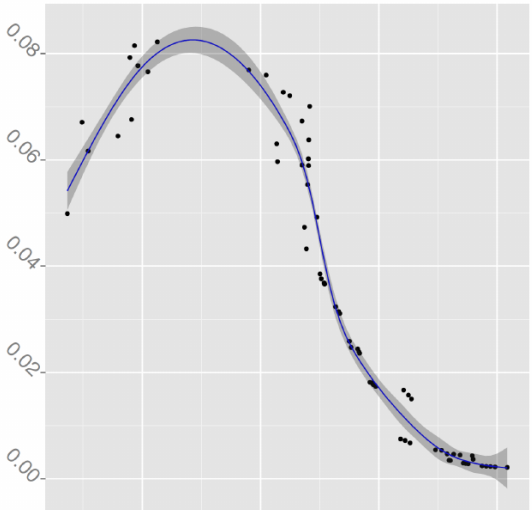


A Study of the Unix Operating System 1973–2015

H4: Programmers trust the compiler for register allocation

Decreasing number of **register** declarations / statement

register struct ifnet *ifp;



H5: Code formatting practices converge to a common standard

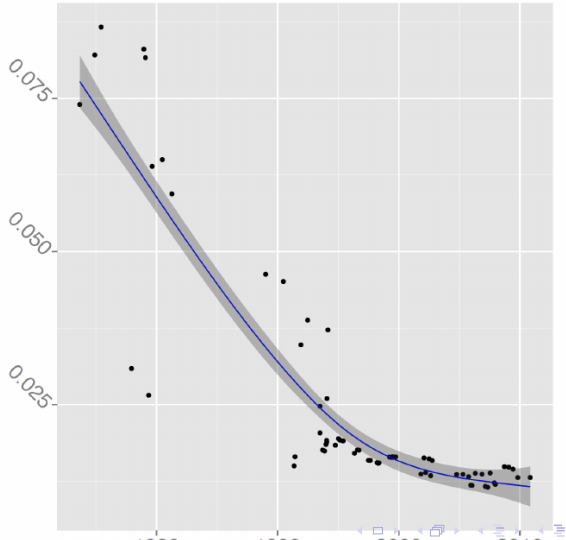
```
LOOP    tdystak(0);
        stakchk(); /* may reduce sbrk */
        exitset();
        IF (flags&prompt) ANDF standin->fstak==0 ANDF !eof
        THEN    IF mailnod.namval
                    ANDF stat(mailnod.namval,&statb)>=0 ANDF statb.st_size
                    ANDF (statb.st_mtime != mailtime)
                    ANDF mailtime
                THEN    prs(mailmsg)
                FI
                mailtime=statb.st_mtime;
                prs(ps1nod.namval); alarm(TIMEOUT); flags |= waiting;
        FI

        trapnote=0; peekc=readc();
        IF eof
        THEN    return;
        FI
        alarm(0); flags &= ~waiting;
        execute(cmd(NL,MTFLG),0);
        eof |= (flags&oneflg);
POOL
```

Half Century of Unix: History, Preservation, and Lessons Learned,Diomidis Spinellis, keynote of OW2 Consortium, 2017

H5: Code formatting practices converge to a common standard

Decrease in code **inconsistency**

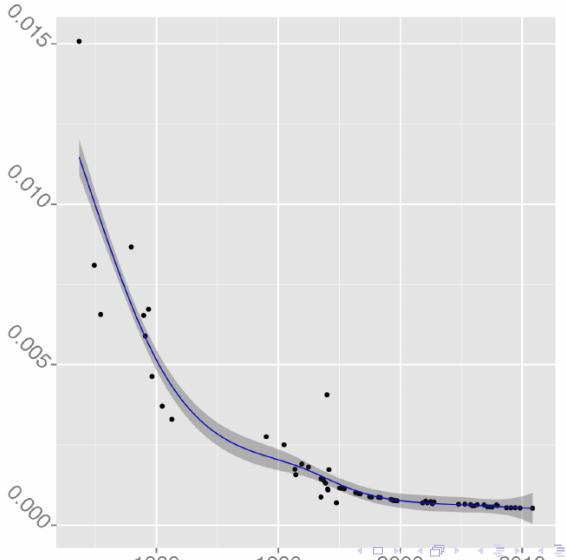
```
if (q()) {          if( q() )
                        {
```



H5: Code formatting practices converge to a common standard

Decrease in
indentation spaces
standard deviation

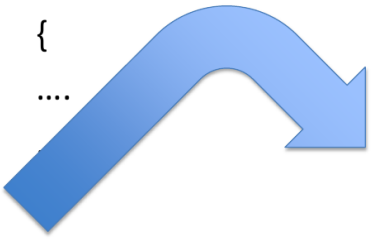
```
if (a)
  while (b)
        for (;;)
```

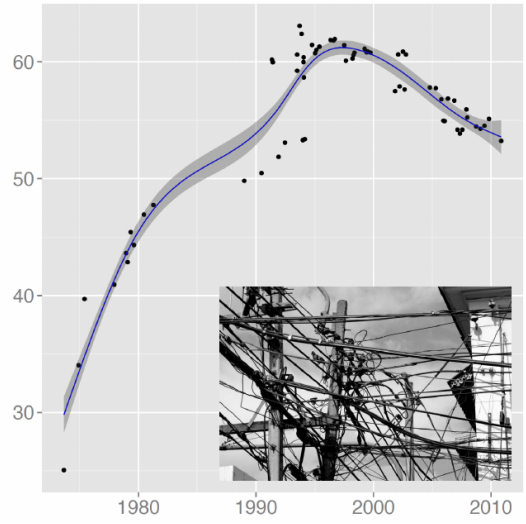


A Study of the Unix Operating System 1973–2015

H6: Software complexity evolution follows self correction

Mean lines / function

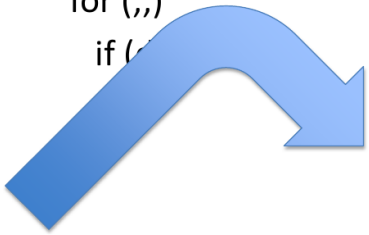


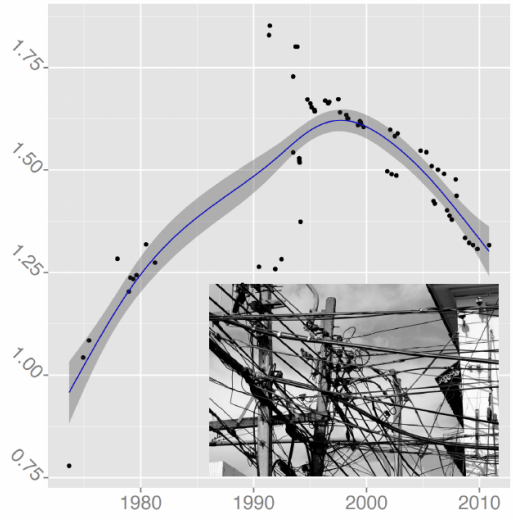


H6: Software complexity evolution follows self correction

Mean statement nesting

```
if (a)
  while (b)
    for (;;)
      if (
```





Half Century of Unix: History, Preservation, and Lessons Learned,Diomidis Spinellis, keynote of OW2 Consortium, 2017

A Study of the Unix Operating System 1973–2015

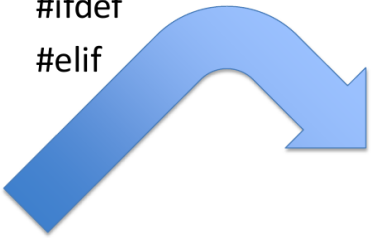
H6: Software complexity evolution follows self correction

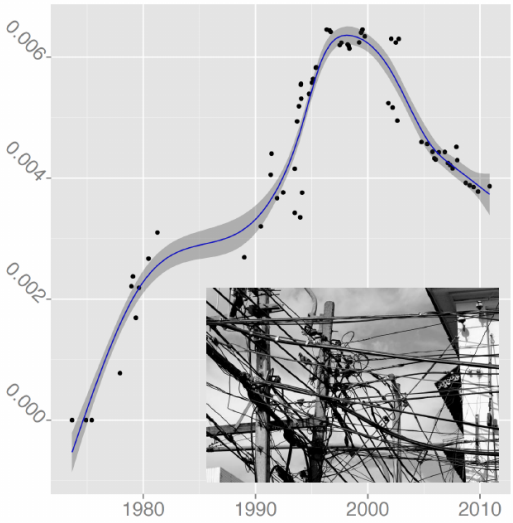
Density of C preprocessor conditionals

#if

#ifdef

#elif



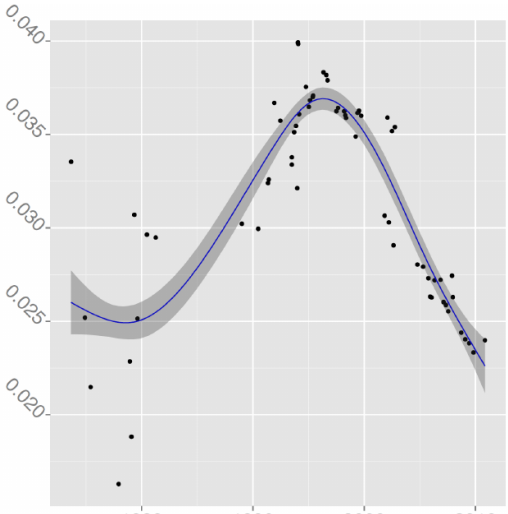


H6: Software complexity evolution follows self correction

Density of C preprocessor non-include directives

#define

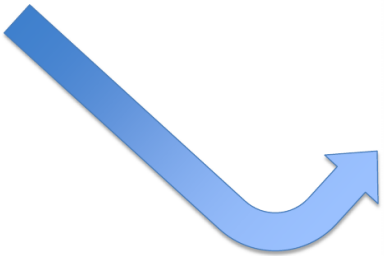
#if

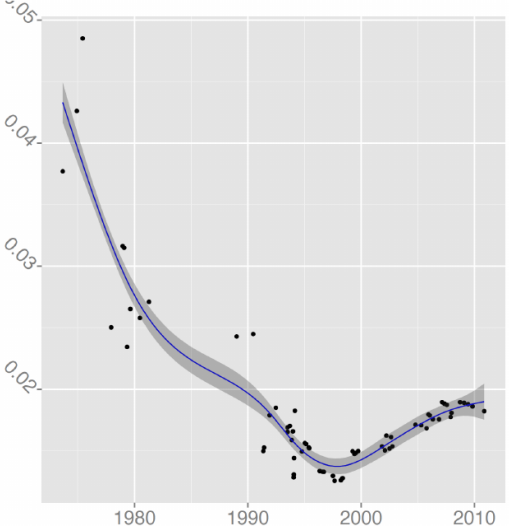


A Study of the Unix Operating System 1973–2015

H6: Software complexity evolution follows self correction

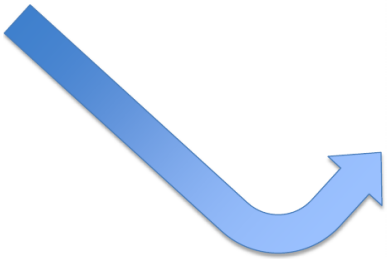
goto keyword density

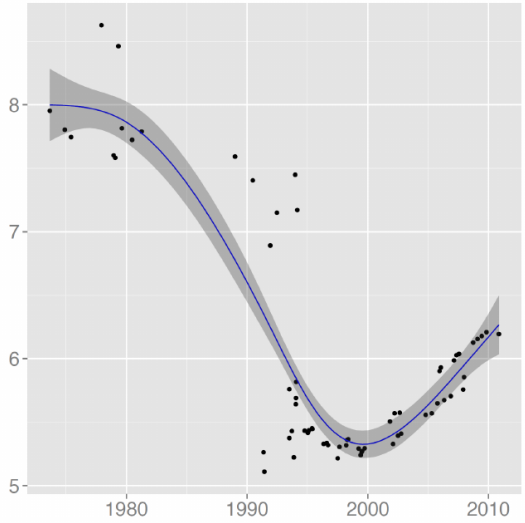




H7: Code readability increases

Mean indentation spaces converge around 6



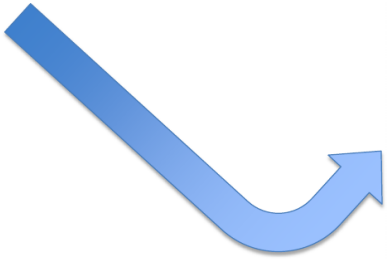


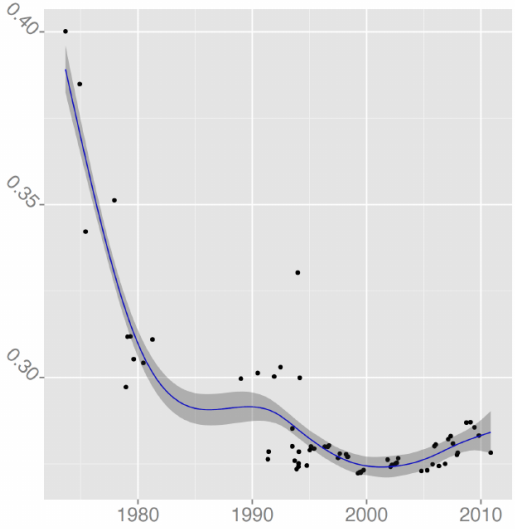
Half Century of Unix: History, Preservation, and Lessons Learned,Diomidis Spinellis, keynote of OW2 Consortium, 2017

H7: Code readability increases

Statements / line decrease

a(); b++; d();



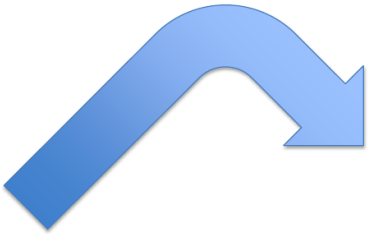


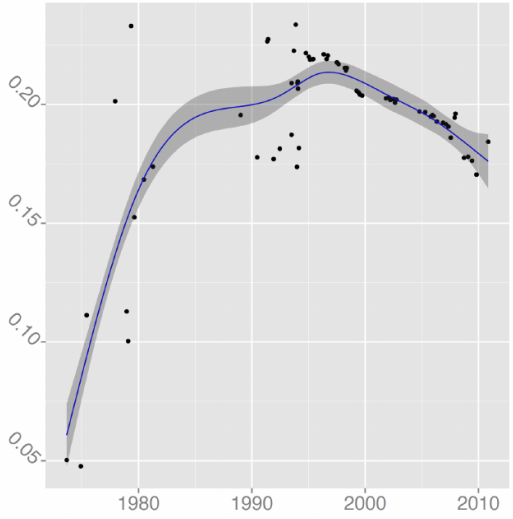
Half Century of Unix: History, Preservation, and Lessons Learned,Diomidis Spinellis, keynote of OW2 Consortium, 2017

A Study of the Unix Operating System 1973–2015

H7: Code readability increases

Comment character density

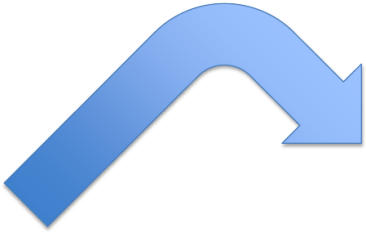


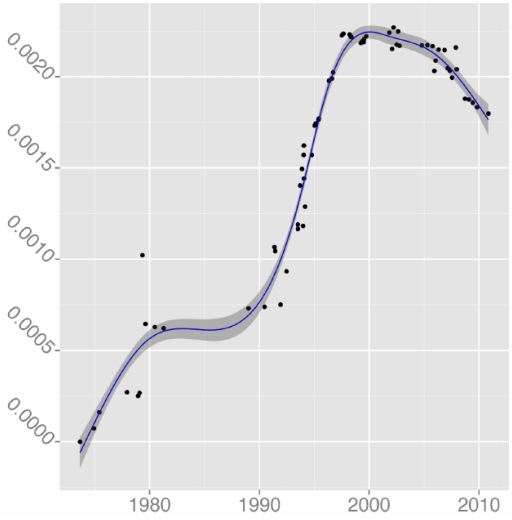


Half Century of Unix: History, Preservation, and Lessons Learned,Diomidis Spinellis, keynote of OW2 Consortium, 2017

H7: Code readability increases

Kludge word density





References

- The Development of the C Language, Dennis M. Ritchie,1993
- The Evolution of C Programming Practices: A Study of the Unix Operating System 1973–2015, Diomidis Spinellis, ICSE, 2016
- Half Century of Unix: History, Preservation, and Lessons Learned,Diomidis Spinellis, keynote of OW2 Consortium, 2017