Modern Software Engineering and Research

A professional development workshop

Mark Galassi

Space Science and Applications group
Los Alamos National Laboratory
and
Institute for Computing in Research

2020-05-16, 2021-01-20, 2023-01-24

Last built 2023-02-13T19:07:22

LA-UR-20-24695

Outline

Goals and path

Goals

Goals and path

Goals

- Have a broad view of University curriculum, successes and limitations, state of industry.

Goals and path

Goals

- Have a broad view of University curriculum, successes and limitations, state of industry.
- Awareness of grand challenges in software engineering.

Goals and path

Goals

- Have a broad view of University curriculum, successes and limitations, state of industry.
- Awareness of grand challenges in software engineering.
- Awareness of current approaches to address those challenges.

Goals and path

Goals

- Have a broad view of University curriculum, successes and limitations, state of industry.
- Awareness of grand challenges in software engineering.
- Awareness of current approaches to address those challenges.
- Largely historical: my personal inclination to use history for metaphor and perspective.

Goals and path

Goals

- Have a broad view of University curriculum, successes and limitations, state of industry.
- Awareness of grand challenges in software engineering.
- Awareness of current approaches to address those challenges.
- Largely historical: my personal inclination to use history for metaphor and perspective.

The meandering path

Goals and path

Goals

- Have a broad view of University curriculum, successes and limitations, state of industry.
- Awareness of grand challenges in software engineering.
- Awareness of current approaches to address those challenges.
- Largely historical: my personal inclination to use history for metaphor and perspective.

The meandering path

- Curriculum

Goals and path

Goals

- Have a broad view of University curriculum, successes and limitations, state of industry.
- Awareness of grand challenges in software engineering.
- Awareness of current approaches to address those challenges.
- Largely historical: my personal inclination to use history for metaphor and perspective.

The meandering path

- Curriculum
- Programming languages

Goals and path

Goals

- Have a broad view of University curriculum, successes and limitations, state of industry.
- Awareness of grand challenges in software engineering.
- Awareness of current approaches to address those challenges.
- Largely historical: my personal inclination to use history for metaphor and perspective.

The meandering path

- Curriculum
- Programming languages
- Operating systems

Goals and path

Goals

- Have a broad view of University curriculum, successes and limitations, state of industry.
- Awareness of grand challenges in software engineering.
- Awareness of current approaches to address those challenges.
- Largely historical: my personal inclination to use history for metaphor and perspective.

The meandering path

- Curriculum
- Programming languages
- Operating systems
- Tools and methodologies

Goals and path

Goals

- Have a broad view of University curriculum, successes and limitations, state of industry.
- Awareness of grand challenges in software engineering.
- Awareness of current approaches to address those challenges.
- Largely historical: my personal inclination to use history for metaphor and perspective.

The meandering path

- Curriculum
- Programming languages
- Operating systems
- Tools and methodologies
- Case studies

Goals and path

Goals

- Have a broad view of University curriculum, successes and limitations, state of industry.
- Awareness of grand challenges in software engineering.
- Awareness of current approaches to address those challenges.
- Largely historical: my personal inclination to use history for metaphor and perspective.

The meandering path

- Curriculum
- Programming languages
- Operating systems
- Tools and methodologies
- Case studies

Style

- Slides are placeholders for me to then tell stories.

Goals and path

Goals

- Have a broad view of University curriculum, successes and limitations, state of industry.
- Awareness of grand challenges in software engineering.
- Awareness of current approaches to address those challenges.
- Largely historical: my personal inclination to use history for metaphor and perspective.

The meandering path

- Curriculum
- Programming languages
- Operating systems
- Tools and methodologies
- Case studies

Style

- Slides are placeholders for me to then tell stories.
  Please interrupt: I hope you will talk and tell stories too.

Goals and path

Goals

- Have a broad view of University curriculum, successes and limitations, state of industry.
- Awareness of grand challenges in software engineering.
- Awareness of current approaches to address those challenges.
- Largely historical: my personal inclination to use history for metaphor and perspective.

The meandering path

- Curriculum
- Programming languages
- Operating systems
- Tools and methodologies
- Case studies

Style

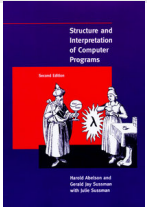
- Slides are placeholders for me to then tell stories.
  Please interrupt: I hope you will talk and tell stories too.
- Note: I am part of the secret cabal that seeks to give a seminar made entirely of xkcd slides.

Part I – Curriculum

The Computer Science Curriculum

From https://teachyourselfcs.com/

Computer programming

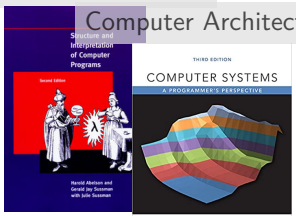


The Computer Science Curriculum

From https://teachyourselfcs.com/

Computer programming

Computer Architecture



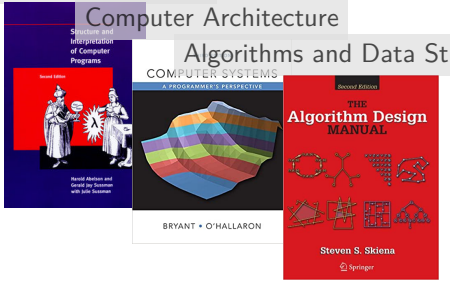
The Computer Science Curriculum

From https://teachyourselfcs.com/

Computer programming

Computer Architecture

Algorithms and Data Structures



The Computer Science Curriculum

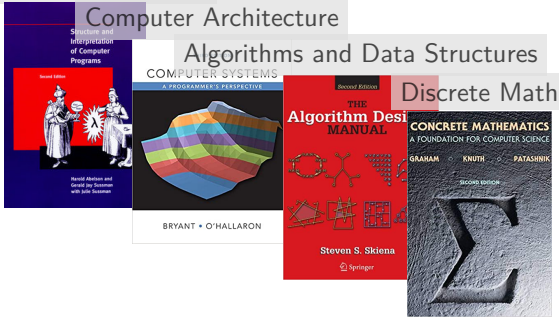
From https://teachyourselfcs.com/

Computer programming

Computer Architecture

Algorithms and Data Structures

Discrete Math



The Computer Science Curriculum

From https://teachyourselfcs.com/

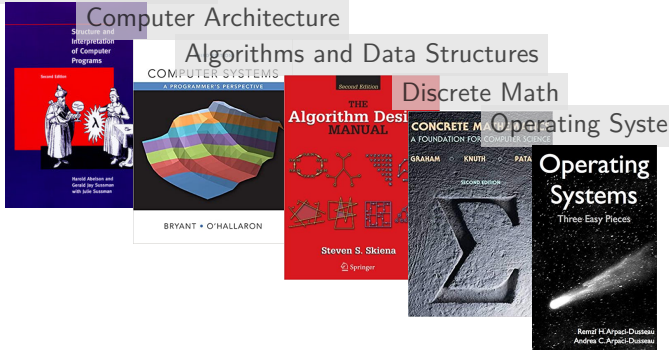
Computer programming

Computer Architecture

Algorithms and Data Structures

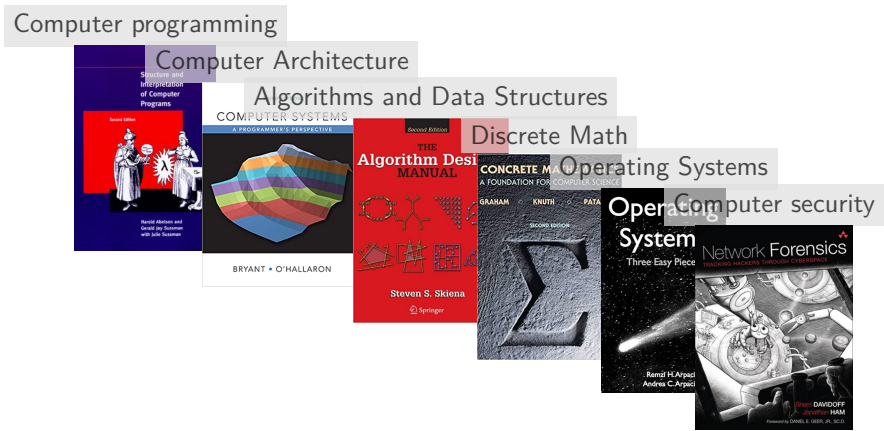
Discrete Math

Operating Systems



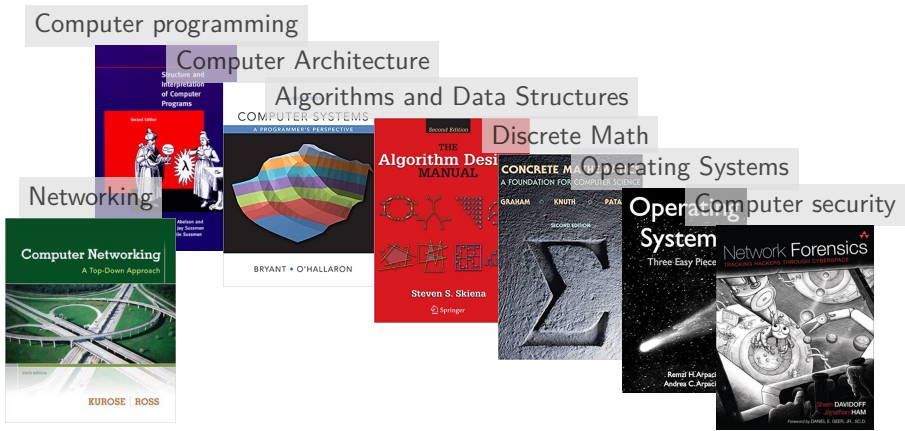
The Computer Science Curriculum

From https://teachyourselfcs.com/



The Computer Science Curriculum

From https://teachyourselfcs.com/

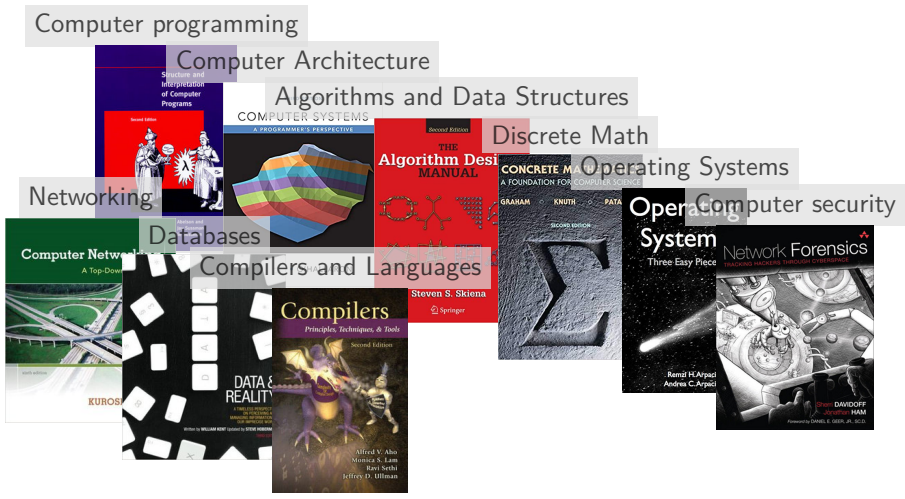


The Computer Science Curriculum

From https://teachyourselfcs.com/

The Computer Science Curriculum

From https://teachyourselfcs.com/

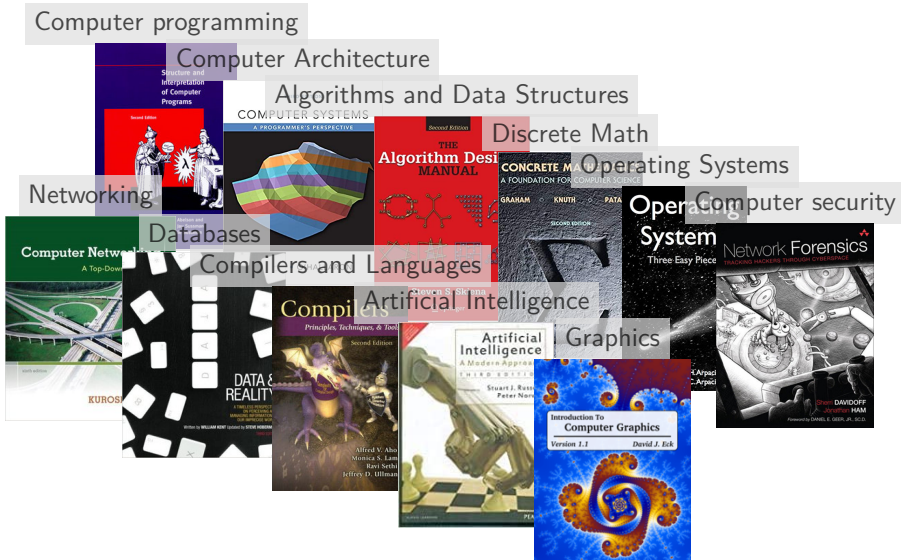


The Computer Science Curriculum

From https://teachyourselfcs.com/

The Computer Science Curriculum

From https://teachyourselfcs.com/



The Software Engineering curriculum

Margaret Hamilton, who led the MIT team that wrote the Apollo on-board software in the 1960s, is one of the coiners of the term "software engineering".

The Software Engineering curriculum

Margaret Hamilton, who led the MIT team that wrote the Apollo on-board software in the 1960s, is one of the coiners of the term "software engineering".

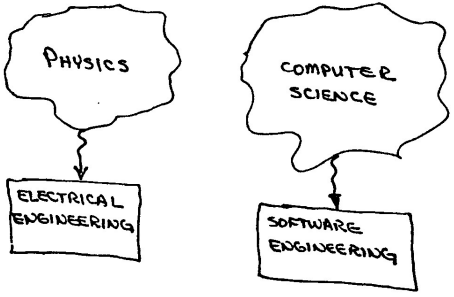


Figure 3.
Software-Electrical Engineering Analog

The Software Engineering curriculum

Margaret Hamilton, who led the MIT team that wrote the Apollo on-board software in the 1960s, is one of the coiners of the term "software engineering".

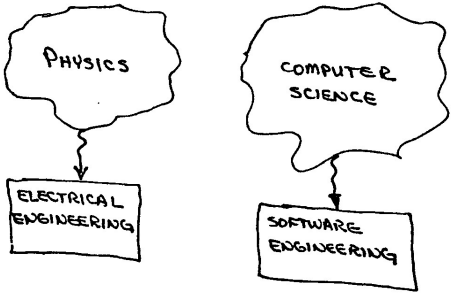


Figure 3.
Software-Electrical Engineering Analog

Most of the computer science department courses.

The Software Engineering curriculum

Margaret Hamilton, who led the MIT team that wrote the Apollo on-board software in the 1960s, is one of the coiners of the term "software engineering".

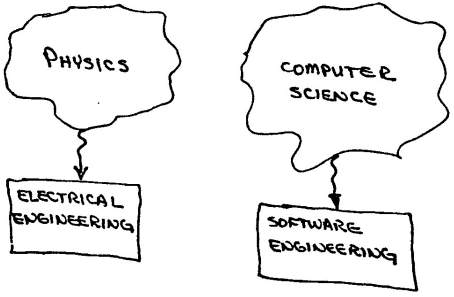


Figure 3.
Software-Electrical Engineering Analog

Most of the computer science department courses.
Less math.

The Software Engineering curriculum

Margaret Hamilton, who led the MIT team that wrote the Apollo on-board software in the 1960s, is one of the coiners of the term "software engineering".

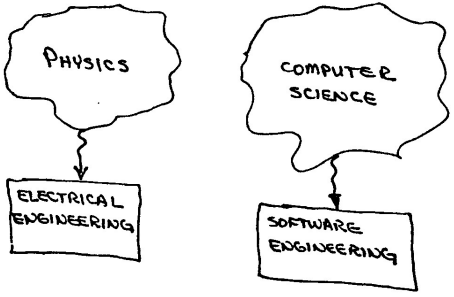


Figure 3.
Software-Electrical Engineering Analog

Most of the computer science department courses.
Less math.
Process and management classes.

The Software Engineering curriculum

Margaret Hamilton, who led the MIT team that wrote the Apollo on-board software in the 1960s, is one of the coiners of the term "software engineering".

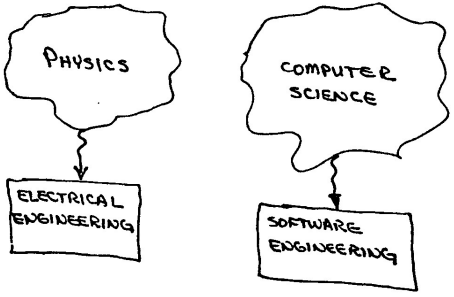


Figure 3.
Software-Electrical Engineering Analog

Most of the computer science department courses.
Less math.
Process and management classes.
ISO's "Software Engineering Body of Knowledge" (SWEBOK).

Part II – Programming languages

Grand challenges for programming language design

Terminology

Attitude toward terminology Suspend one's uncertainty.

Interpreter Slow and flexible.

Compiler Fast: compiles to machine code. And what is that machine code, with its fabled ones and zeros? See ▸ Machine language – 6502

Controlling complexity of large programs

Cutoff at about 100 thousand lines of code.

Performance

Language features are related to how well you can optimize.

Memory safety

Avoiding memory corruption while keeping high performance.

The story of programming languages

From https://www.scriptol.com/programming/chronology.php

The story of programming languages

From https://www.scriptol.com/programming/chronology.php

Prehistory

1840 Analytical Engine (Charles Babbage and Ada Lovelace)

1943 ENIAC coding system

1947-1949 Assembly language

1955 FLOW-MATIC (Grace Hopper)

The story of programming languages

From https://www.scriptol.com/programming/chronology.php

Prehistory

1840 Analytical Engine (Charles Babbage and Ada Lovelace)

1943 ENIAC coding system

1947-1949 Assembly language

1955 FLOW-MATIC (Grace Hopper)

The 1950s

1957 FORTRAN (John Backus)

1958 LISP (John McCarthy)

1959 COBOL (CODASYL group)

The story of programming languages

From https://www.scriptol.com/programming/chronology.php

Prehistory

1840 Analytical Engine (Charles Babbage and Ada Lovelace)

1943 ENIAC coding system

1947-1949 Assembly language

1955 FLOW-MATIC (Grace Hopper)

The 1950s

1957 FORTRAN (John Backus)

1958 LISP (John McCarthy)

1959 COBOL (CODASYL group)

The 1960s

1960 ALGOL 60

1962 APL

1964 BASIC

1964 Simula

1969 PL/1, B

The story of programming languages

From https://www.scriptol.com/programming/chronology.php

Prehistory

1840 Analytical Engine (Charles Babbage and Ada Lovelace)

1943 ENIAC coding system

1947-1949 Assembly language

1955 FLOW-MATIC (Grace Hopper)

The 1950s

1957 FORTRAN (John Backus)

1958 LISP (John McCarthy)

1959 COBOL (CODASYL group)

The 1960s

1960 ALGOL 60

1962 APL

1964 BASIC

1964 Simula

1969 PL/1, B

The 1970s

1970 Pascal

1972 C

1973 FORTH, ML

1975 Scheme

1977 Bourne shell

The story of programming languages

From https://www.scriptol.com/programming/chronology.php

Prehistory

1840 Analytical Engine (Charles Babbage and Ada Lovelace)

1943 ENIAC coding system

1947-1949 Assembly language

1955 FLOW-MATIC (Grace Hopper)

The 1950s

1957 FORTRAN (John Backus)

1958 LISP (John McCarthy)

1959 COBOL (CODASYL group)

The 1960s

1960 ALGOL 60

1962 APL

1964 BASIC

1964 Simula

1969 PL/1, B

The 1970s

1970 Pascal

1972 C

1973 FORTH, ML

1975 Scheme

1977 Bourne shell

The 1980s

1980 Smalltalk

1983 Ada

1985 Postscript, C++

1987 Perl

1988 Tcl

The story of programming languages

From https://www.scriptol.com/programming/chronology.php

Prehistory

1840 Analytical Engine (Charles Babbage and Ada Lovelace)

1943 ENIAC coding system

1947-1949 Assembly language

1955 FLOW-MATIC (Grace Hopper)

The 1950s

1957 FORTRAN (John Backus)

1958 LISP (John McCarthy)

1959 COBOL (CODASYL group)

The 1960s

1960 ALGOL 60

1962 APL

1964 BASIC

1964 Simula

1969 PL/1, B

The 1970s

1970 Pascal

1972 C

1973 FORTH, ML

1975 Scheme

1977 Bourne shell

The 1980s

1980 Smalltalk

1983 Ada

1985 Postscript, C++

1987 Perl

1988 Tcl

The 1990s

1990 Haskell

1991 Python

1995 Java, javascript, Ruby, PHP

The story of programming languages

From https://www.scriptol.com/programming/chronology.php

Prehistory

1840 Analytical Engine (Charles Babbage and Ada Lovelace)
1943 ENIAC coding system
1947-1949 Assembly language
1955 FLOW-MATIC (Grace Hopper)

The 1950s

1957 FORTRAN (John Backus)
1958 LISP (John McCarthy)
1959 COBOL (CODASYL group)

The 1960s

1960 ALGOL 60
1962 APL
1964 BASIC
1964 Simula
1969 PL/1, B

The 1970s

1970 Pascal
1972 C
1973 FORTH, ML
1975 Scheme
1977 Bourne shell

The 1980s

1980 Smalltalk
1983 Ada
1985 Postscript, C++
1987 Perl
1988 Tcl

The 1990s

1990 Haskell
1991 Python
1995 Java, javascript, Ruby, PHP

The "aughts"

2000 C#
2004 Scala
2006 Rust
2007 Scratch
2009 Go

The story of programming languages

From https://www.scriptol.com/programming/chronology.php

Prehistory

1840 Analytical Engine (Charles Babbage and Ada Lovelace)
1943 ENIAC coding system
1947-1949 Assembly language
1955 FLOW-MATIC (Grace Hopper)

The 1950s

1957 FORTRAN (John Backus)
1958 LISP (John McCarthy)
1959 COBOL (CODASYL group)

The 1960s

1960 ALGOL 60
1962 APL
1964 BASIC
1964 Simula
1969 PL/1, B

The 1970s

1970 Pascal
1972 C
1973 FORTH, ML
1975 Scheme
1977 Bourne shell

The 1980s

1980 Smalltalk
1983 Ada
1985 Postscript, C++
1987 Perl
1988 Tcl

The 1990s

1990 Haskell
1991 Python
1995 Java, javascript, Ruby, PHP

The "aughts"

2000 C#
2004 Scala
2006 Rust
2007 Scratch
2009 Go

The 2010s

2010 Julia
2012 Kotlin
2017 WebAssembly

The story of programming languages

From https://www.scriptol.com/programming/chronology.php

Prehistory

1840 Analytical Engine (Charles Babbage and Ada Lovelace)
1943 ENIAC coding system
1947-1949 Assembly language
1955 FLOW-MATIC (Grace Hopper)

The 1950s

1957 FORTRAN (John Backus)
1958 LISP (John McCarthy)
1959 COBOL (CODASYL group)

The 1960s

1960 ALGOL 60
1962 APL
1964 BASIC
1964 Simula
1969 PL/1, B

The 1970s

1970 Pascal
1972 C
1973 FORTH, ML
1975 Scheme
1977 Bourne shell

The 1980s

1980 Smalltalk
1983 Ada
1985 Postscript, C++
1987 Perl
1988 Tcl

The 1990s

1990 Haskell
1991 Python
1995 Java, javascript, Ruby, PHP

The "aughts"

2000 C#
2004 Scala
2006 Rust
2007 Scratch
2009 Go

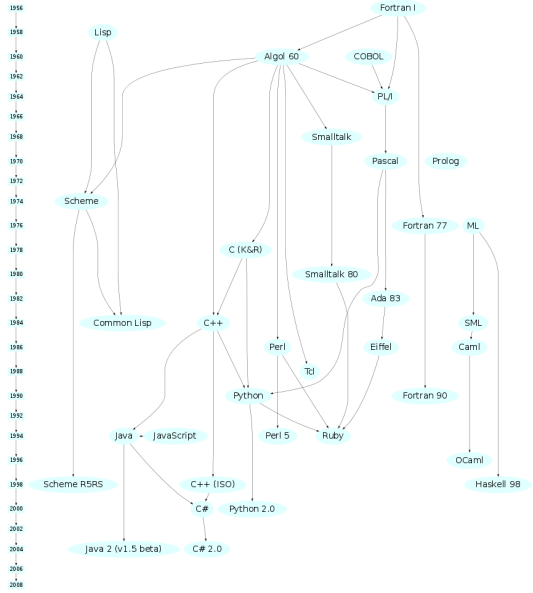
The 2010s

2010 Julia
2012 Kotlin
2017 WebAssembly

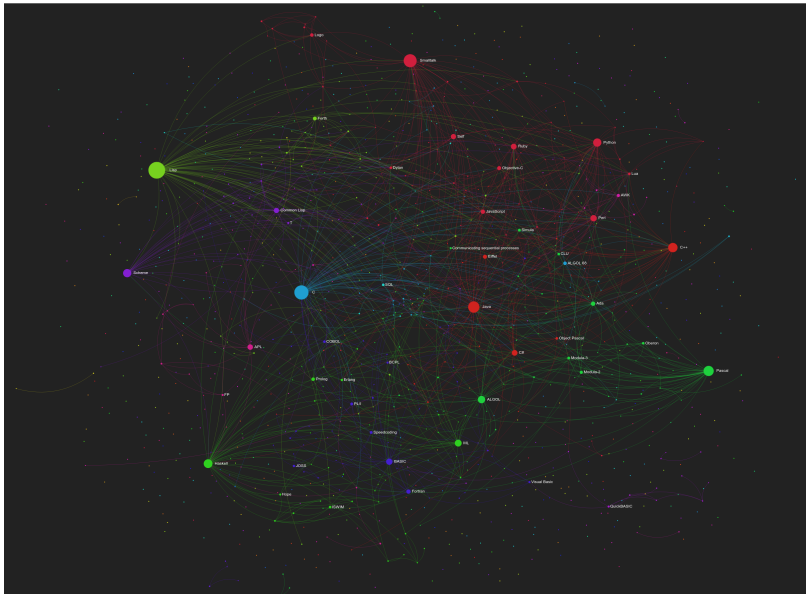
The future (created by Santa Fe youngsters)

2027 greenchile
2030 joemama
2032 updog

The story of programming languages – timeline



The story of programming languages – influence



What do these languages look like?

From http://rosettacode.org/wiki/Loops/For

Rosetta code

What we will investigate

We will write the "stars" program which prints first one, then two, three, four and five stars on separate lines, so we can discuss the following about each language: (a) Motivation and history, (b) Syntax peculiarities and "feel"

archetypes FORTRAN, LISP, COBOL

wide diversity FORTH, Smalltalk, Pascal, Haskell

currently relevant C, C++, Go, Rust, Python, R, javascript, sh

Machine language – 6502

Hexadecimal opcodes for a program that calculates $2 + 5$
From `https://www.atariarchives.org/mlb/chapter2.php`

Hex:

1000 A9 02 69 05 8D A0 0F 60

Binary:

1000000000000 10101001 00000010 01101001 00000101 10001101 10100000 00001111 01100000

And yes, that's what they mean when they say "it's all ones and zeros."

Assembly language – 6502

```
1000 A9 02    LDA #$02
1002 69 05    ADC #$05
1004 8D A0 0F STA $0FA0
1007 60       RTS
```

FORTRAN

```
CC compile with "gfortran stars.for -o stars_fortran"
CC run with "./stars_fortran"
      PROGRAM FORLOOP
        INTEGER I, J

        DO 20 I = 1, 5
          DO 10 J = 1, I
C           Print the asterisk.
            WRITE (*,5001) '*'
   10     CONTINUE
C         Print a newline.
          WRITE (*,5000) ''
   20   CONTINUE

        STOP

 5000   FORMAT (A)

 5001   FORMAT (A, $)

C5001   FORMAT (A, ADVANCE='NO')
      END
```

LISP

```
;; recursive approach; you can run this with "gcl < stars.lisp"
(defun print-stars (number)
  "Print a given number of stars, using recursion"
  (if (= number 0)
      (progn
        (write-char #\*)
        (terpri))
    (progn
      (write-char #\*)
      (print-stars (1- number)))))

(defun print-triangle (n-rows)
  (if (= n-rows 0)
      (print-stars n-rows)
    (progn
      (print-stars n-rows)
      (print-triangle (1- n-rows)))))

(print-triangle 5)
```

COBOL

```
       IDENTIFICATION DIVISION.
      * compile with "cobc stars.cob  -o stars_cobol"
       PROGRAM-ID. Display-Triangle.

       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  Outer-Counter PIC 9.
       01  Inner-Counter PIC 9.

       PROCEDURE DIVISION.
       PERFORM VARYING Outer-Counter FROM 1 BY 1 UNTIL 5 < Outer-Counter

           PERFORM VARYING Inner-Counter FROM 1 BY 1
                   UNTIL Outer-Counter < Inner-Counter
               DISPLAY "*" NO ADVANCING
           END-PERFORM

           DISPLAY "" *> Output a newline
       END-PERFORM

       GOBACK
       .
```

C

```c
/* compile with "gcc stars.c -o stars_c" and run with "./stars_c" */
#include <stdio.h>

int main()
{
  int i, j;
  for (i = 1; i <= 5; i++) {
    for (j = 1; j <= i; j++) {
      putchar('*');
    }
    putchar('\n');
  }
}
```

Interlude: obfuscated C

From https://www.ioccc.org/years.html#1987

```
#define iv 4
#define v ;(void
#define XI(xi)int xi[iv*'V'];
#define L(c,I,i)c(){d(I);m(i);}
#include <stdio.h>
int*cc,c,i,ix='\t',exit(),X='\n'*'\d';XI(VI)XI(xi)extern(*vi[])(),(*
signal())();char*V,cm,D['x'],M='\n',I,*gets();L(MV,V,(c+='d',ix))m(x){v)
signal(X/'I',vi[x]);}d(x)char*x;{v)write(i,x,i);}L(MC,V,M+I)xv(){c>=i?m(
c/M/M+M):(d(&M),m(cm));}L(mi,V+cm,M)L(md,V,M)MM(){c=c*M%X;V-=cm;m(ix);}
LXX(){gets(D)||(vi[iv])();c=atoi(D);while(c>=X){c-=X;d("m");}V="ivxlcdm"
+iv;m(ix);}LV(){c-=c;while((i=cc[*D=getchar()])>-I)i?(c?(c<i&&I(-c-c,
"%d"),I(i,"+%d")):I(i,"(%d")):(c&&I(M,")"),I(*D,"%c")),c=i;c&&I(X,")"),I
(-i,"%c");m(iv-!(i&I));}L(ml,V,'\f')li(){m(cm+!isatty(i=I));}ii(){m(c=cm
= ++I)v)pipe(VI);cc=xi+cm++;for(V="jWYmDEnX";*V;V++)xi[*V^' ']=c,xi[*V++]
=c,c*=M,xi[*V^' ']=xi[*V]=c>>I;cc[-I]-=ix v)close(*VI);cc[M]-=M;}main(){
(*vi)();for(;v)write(VI[I],V,M));}I(xl,lx)char*lx;{v)printf(lx,xl)v)
fflush(stdout);}L(xx,V+I,(c-=X/cm,ix))int(*vi[])()={ii,li,LXX,LV,exit,I,
d,I,d,xv,MM,md,MC,ml,MV,xx,xx,xx,xx,MV,mi};
```

Forth

```
( run this with "gforth < stars.forth"
: triangle ( n -- )
  1+ 1 do
    cr i 0 do [char] * emit loop
  loop ;
5 triangle
```

Smalltalk

```
"run with gst stars.st"
1 to: 5 do: [ :aNumber |
  aNumber timesRepeat: [ '*' display ].
  Character nl display.
]
```

Pascal

```
(* compile with "fpc stars.p -ostars_pascal", run with "./stars_pascal" *)
program stars(output);

var
  i, j: integer;

begin
  for i := 1 to 5 do
    begin
      for j := 1 to i do
        write('*');
      writeln
    end
end.
```

Haskell

```haskell
-- | compile with "ghc stars.hs -o stars_haskell" and run with "./stars_haskell"
import Control.Monad

main = do
  forM_ [1..5] $ \i -> do
    forM_ [1..i] $ \j -> do
      putChar '*'
    putChar '\n'
```

Javascript

```
// run with "node < stars.js", or change console.log(s) to print(s)
// and you can run with "rhino < stars.js"
var i, j;
for (i = 1; i <= 5; i += 1) {
  s = '';
  for (j = 0; j < i; j += 1)
    s += '*';
  console.log(s);
}
```

Python

```
#! /usr/bin/env python3

# run this with "python3 stars.py"

for i in range(5):
    for j in range(i+1):
        print('*', end="")
    print()
```

R

```
// run with "R -f stars.R"
for(i in 0:4) {
  s <- ""
  for(j in 0:i) {
    s <- paste(s, "*", sep="")
  }
  print(s)
}
```

Java

```
// compile with "javac stars.java" and run with "java stars"
public class stars {
    public static void main(String[] args) {
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Rust

```rust
// compile with "rustc stars.rs -o stars_rust", run with "./stars_rust"
fn main() {
    for i in 0..5 {
        for _ in 0..=i {
            print!("*");
        }

        println!();
    }
}
```

Go

```
// compile with "gccgo stars.go -o stars_go", run with "./stars_go"
package main

import "fmt"

func main() {
    for i := 1; i <= 5; i++ {
        for j := 1; j <= i; j++ {
            fmt.Printf("*")
        }
        fmt.Printf("\n")
    }
}
```

sh

```
# to run it just paste it into the shell or type "/bin/sh stars.sh" or
# make it executable with "chmod +x stars.sh" and then run it with
# "./stars.sh"
for i in `seq 1 5`
do
    for j in `seq 1 $i`
    do
        echo -n "*"
    done
    echo
done
```

Distilling insight from the tour

Distilling insight from the tour

Compiled versus interpreted

. . . (discussion) . . .

Distilling insight from the tour

Compiled versus interpreted

. . . (discussion) . . .

Broad classes of language **syntax** styles

. . . (discussion) . . .

Distilling insight from the tour

Compiled versus interpreted

. . . (discussion) . . .

Broad classes of language **syntax** styles

. . . (discussion) . . .

Broad classes of language **semantic** styles

. . . (discussion) . . .

Distilling insight from the tour

Compiled versus interpreted

. . . (discussion) . . .

Broad classes of language **syntax** styles

. . . (discussion) . . .

Broad classes of language **semantic** styles

. . . (discussion) . . .

Evolution

Who influences whom? (Frame 11)

Distilling insight from the tour

Compiled versus interpreted

. . . (discussion) . . .

Broad classes of language **syntax** styles

. . . (discussion) . . .

Broad classes of language **semantic** styles

. . . (discussion) . . .

Evolution

Who influences whom? (Frame 11)

Bearing upon the grand challenge problems

. . . (discussion) . . . more in the discussion of methodologies

Fear and loathing of programming languages – indifference

Brian Kernighan: Why Pascal is Not My Favorite Programming Language
From `http://www.lysator.liu.se/c/bwk-on-pascal.html`

Early comment

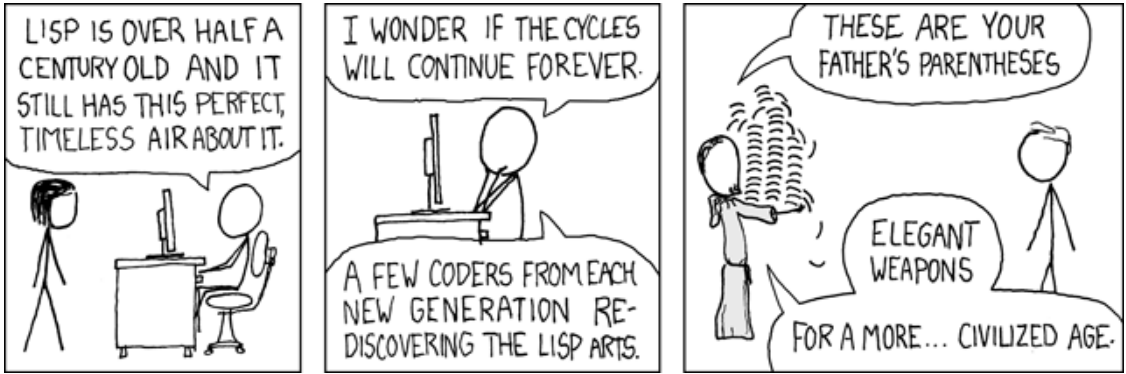
Comparing C and Pascal is rather like comparing a Learjet to a Piper Cub - one is meant for getting something done while the other is meant for learning - so such comparisons tend to be somewhat farfetched. . . .

Conclusion, stated in intro

. . . To state my conclusions at the outset: Pascal may be an admirable language for teaching beginners how to program; I have no first-hand experience with that. It was a considerable achievement for 1968. It has certainly influenced the design of recent languages, of which Ada is likely to be the most important. But in its standard form (both current and proposed), Pascal is not adequate for writing real programs. It is suitable only for small, self-contained programs that have only trivial interactions with their environment and that make no use of any software written by anyone else. . . .

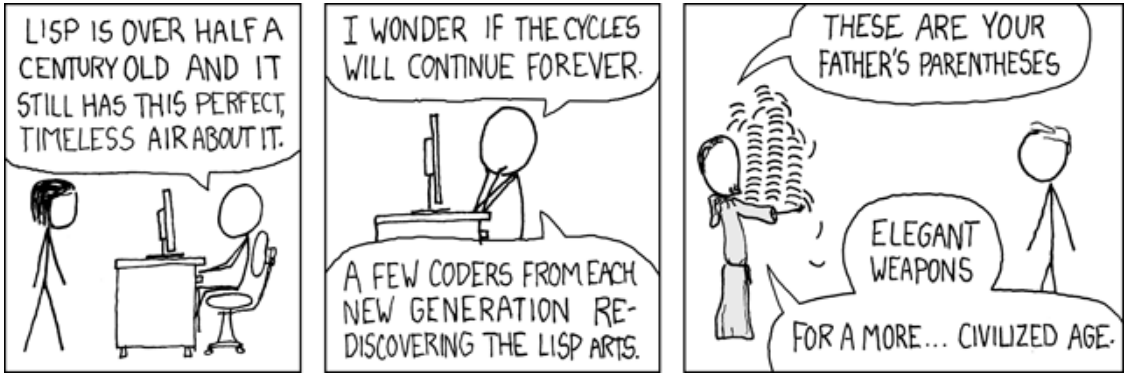
Fear and loathing of programming languages – admiration

Naturalmente . . . xkcd: https://xkcd.com/297/



Fear and loathing of programming languages – admiration

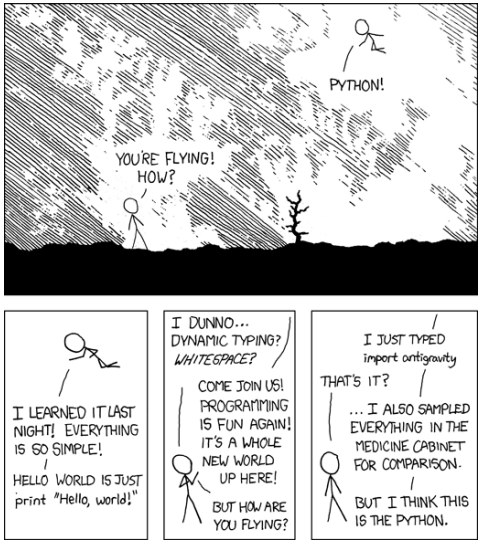
Naturalmente . . . xkcd: https://xkcd.com/297/



I've just received word that the Emperor has dissolved the MIT computer science program permanently.

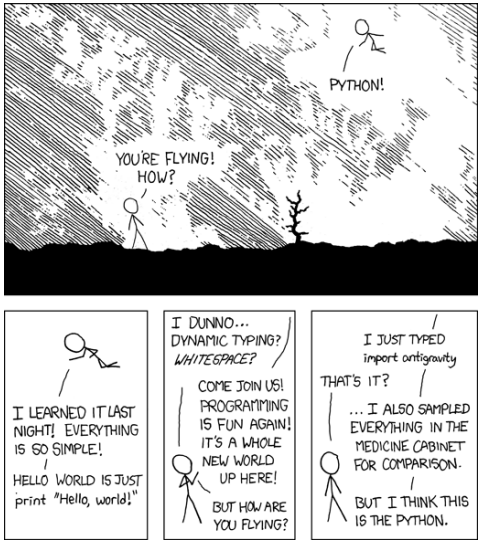
Fear and loathing in programming languages – love

Naturalmente . . . xkcd: https://xkcd.com/353/



Fear and loathing in programming languages – love

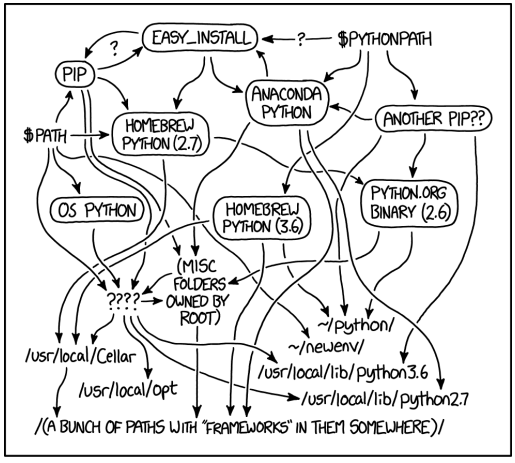
Naturalmente . . . xkcd: `https://xkcd.com/353/`



I wrote 20 short programs in Python yesterday. It was wonderful. Perl, I'm leaving you.

Fear and loathing of programming languages – disillusionment

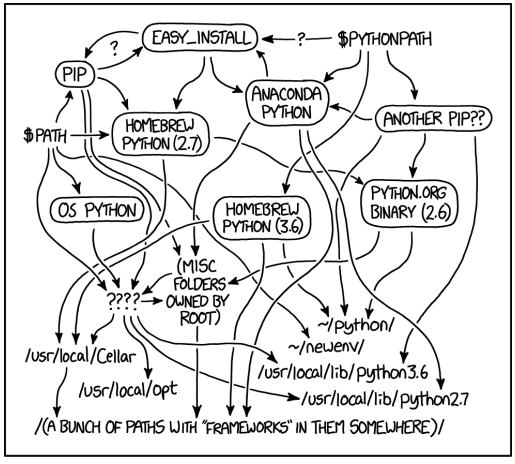
Naturalmente . . . xkcd: https://xkcd.com/1987/



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Fear and loathing of programming languages – disillusionment

Naturalmente . . . xkcd: https://xkcd.com/1987/



The Python environmental protection agency wants to seal it in a cement chamber, with pictorial messages to future civilizations warning them about the danger of using sudo to install random Python packages.

Links - the story of programming languages – visualizations

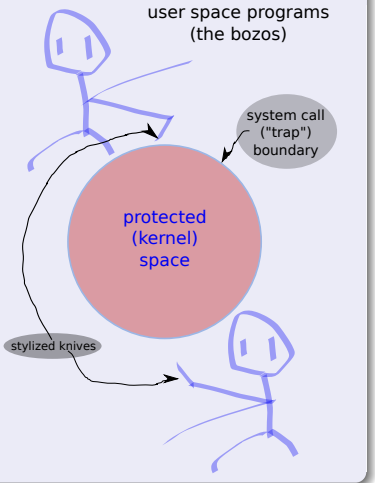
https://github.com/stereobooster/programming-languages-genealogical-tree
http://svalver.github.io/Proglang/
http://svalver.github.io/Proglang/paradigms.html
https://www.youtube.com/watch?v=ZkP4sv3H6g8
https://www.youtube.com/watch?v=Og847HVwRSI
https://vole.wtf/coder-serial-killer-quiz/
The “extra slides” area has two of those videos embedded.

Part III – Operating systems

Part III – Operating systems

Operating system: what is it?

Keith Packard, 1986



Hardware abstraction layer

Imagine in your head the fine-grained operations involved in reading a file from a disk. Compare it to getting milk from the refrigerator.

Protection

This started out as avoiding stomping on memory and device read/write. Today also relevant to cybersecurity.

Grand challenges for operating system design

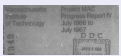
Loading huge programs

FORTRAN multi-pass compiler on one tape: FMS.

Proto-time-sharing

SHARE and SHARE-OS, BBNMON.

Project MAC: McCarthy's discussion of *interrupts versus polling* and hardware support.



Grand challenges for operating system design

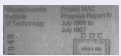
Loading huge programs

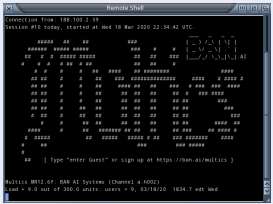
FORTRAN multi-pass compiler on one tape: FMS.

Proto-time-sharing

SHARE and SHARE-OS, BBNMON.

Project MAC: McCarthy's discussion of *interrupts versus polling* and hardware support.





- Hierarchical filesystem, virtual memory, symmetric multiprocessing, multiple languages, a *ton* more.
- Security by design (B2) - mandatory access control, no buffer overflows (PL/I), runtime call sanity checks.
- *Spiral* model of s/w development.
- One of the obvious things that went wrong with Multics as a commercial success was just that it was sort of over-engineered in a sense. There was just too much in it.
  Dennis Ritchie

Grand challenges for operating system design

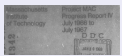
Loading huge programs

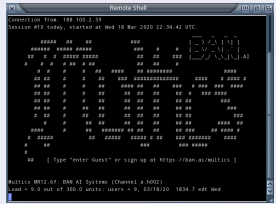
FORTRAN multi-pass compiler on one tape: FMS.

Proto-time-sharing

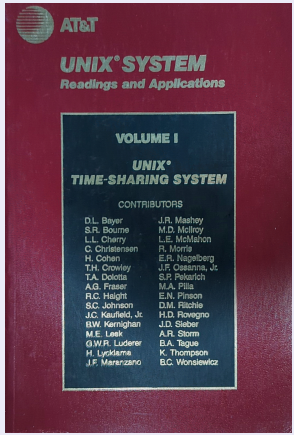
SHARE and SHARE-OS, BBNMON.

Project MAC: McCarthy's discussion of *interrupts versus polling* and hardware support.





- Hierarchical filesystem, virtual memory, symmetric multiprocessing, multiple languages, a *ton* more.
- Security by design (B2) - mandatory access control, no buffer overflows (PL/I), runtime call sanity checks.
- *Spiral* model of s/w development.
- One of the obvious things that went wrong with Multics as a commercial success was just that it was sort of over-engineered in a sense. There was just too much in it.
  Dennis Ritchie



Original article by Ken Thompson and Dennis Ritchie.

Key innovations

Early days

Key innovations

Early days

- Grace Hopper's linker with tape operations.

Key innovations

Early days

- Grace Hopper's linker with tape operations.

- Time sharing: CTSS and ITS $\rightarrow$ Multics $\rightarrow$ UNIX
  - slice from 140ms to [0.75,6]ms.

Key innovations

Early days

- Grace Hopper's linker with tape operations.

- Time sharing: CTSS and ITS $\rightarrow$ Multics $\rightarrow$ UNIX
  - slice from 140ms to [0.75,6]ms.
- OS/360 portable OS.

Key innovations

Early days

- Grace Hopper's linker with tape operations.

- Time sharing: CTSS and ITS $\rightarrow$ Multics $\rightarrow$ UNIX - slice from 140ms to [0.75,6]ms.
- OS/360 portable OS.
- Multics and UNIX innovations - abstraction.

Key innovations

Early days

- Grace Hopper's linker with tape operations.

- Time sharing: CTSS and ITS → Multics → UNIX - slice from 140ms to [0.75,6]ms.
- OS/360 portable OS.
- Multics and UNIX innovations - abstraction.
- Sockets, multiple sources of input, and `select()`.

A 4.2bsd Interprocess Communication Primer
DRAFT of July 27, 1983

Samuel J. Leffler

Robert S. Fabry

William N. Joy

Computer Systems Research Group
Department of Electrical Engineering and Computer Science
University of California, Berkeley

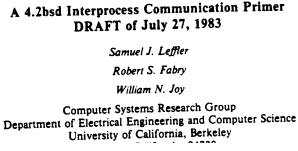
21st century

Key innovations

Early days

- Grace Hopper's linker with tape operations.

- Time sharing: CTSS and ITS → Multics → UNIX - slice from 140ms to [0.75,6]ms.
- OS/360 portable OS.
- Multics and UNIX innovations - abstraction.
- Sockets, multiple sources of input, and `select()`.



A 4.2bsd Interprocess Communication Primer
DRAFT of July 27, 1983

Samuel J. Leffler

Robert S. Fabry

William N. Joy

Computer Systems Research Group
Department of Electrical Engineering and Computer Science
University of California, Berkeley

21st century

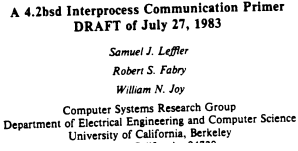
- Advanced file systems.

Key innovations

Early days

- Grace Hopper's linker with tape operations.

- Time sharing: CTSS and ITS → Multics → UNIX - slice from 140ms to [0.75,6]ms.
- OS/360 portable OS.
- Multics and UNIX innovations - abstraction.
- Sockets, multiple sources of input, and `select()`.



A 4.2bsd Interprocess Communication Primer
DRAFT of July 27, 1983

Samuel J. Leffler
Robert S. Fabry
William N. Joy

Computer Systems Research Group
Department of Electrical Engineering and Computer Science
University of California, Berkeley

21st century

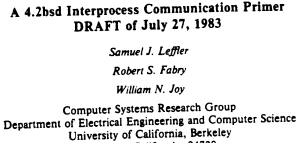
- Advanced file systems.
- Modern security issues.

Key innovations

Early days

- Grace Hopper's linker with tape operations.

- Time sharing: CTSS and ITS → Multics → UNIX - slice from 140ms to [0.75,6]ms.
- OS/360 portable OS.
- Multics and UNIX innovations - abstraction.
- Sockets, multiple sources of input, and `select()`.



A 4.2bsd Interprocess Communication Primer
DRAFT of July 27, 1983

Samuel J. Leffler
Robert S. Fabry
William N. Joy

Computer Systems Research Group
Department of Electrical Engineering and Computer Science
University of California, Berkeley

21st century

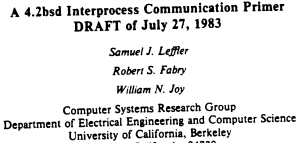
- Advanced file systems.
- Modern security issues.
- Multicore and energy savings.

Key innovations

Early days

- Grace Hopper's linker with tape operations.

- Time sharing: CTSS and ITS → Multics → UNIX - slice from 140ms to [0.75,6]ms.
- OS/360 portable OS.
- Multics and UNIX innovations - abstraction.
- Sockets, multiple sources of input, and `select()`.



A 4.2bsd Interprocess Communication Primer
DRAFT of July 27, 1983

Samuel J. Leffler
Robert S. Fabry
William N. Joy

Computer Systems Research Group
Department of Electrical Engineering and Computer Science
University of California, Berkeley

21st century

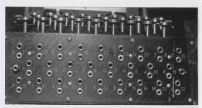
- Advanced file systems.
- Modern security issues.
- Multicore and energy savings.
- Distributed computing (reprised).
- ...

The story of operating systems - part 1

The story of operating systems - part 1

Prehistory

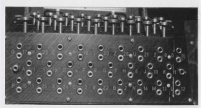
pre-1955 Plugboards (no OS)



The story of operating systems - part 1

Prehistory

pre-1955 Plugboards (no OS)



1955-1960

Single batch - perforated paper paper tape, IBM, and punchcards.

Single batch - tape Unisys and magnetic tape, 128 char/inch, eventually 800 char/inch.

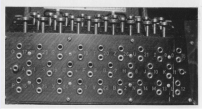
1956 General Motors GM-NAA for IBM 701 and then 704.

1957 Start of Compatible TimeSharing System (CTSS) development at MIT.

The story of operating systems - part 1

Prehistory

pre-1955 Plugboards (no OS)



1955-1960

Single batch - perforated paper paper tape, IBM, and punchcards.

Single batch - tape Unisys and magnetic tape, 128 char/inch, eventually 800 char/inch.

1956 General Motors GM-NAA for IBM 701 and then 704.

1957 Start of Compatible TimeSharing System (CTSS) development at MIT.

The 1960s

1960 SHARE Operating System (SOS), later renamed IBSYS.

1960 FORTRAN Monitor System (FMS): OS and FORTRAN compiler on same tape!

1960 Bell Monitor (BELLMON or BESYS), University of Michgan Executive System (UMES).

1961 CTSS demonstrated at MIT on IBM 709.

1963 SHARE OS taken by IBM and renamed IBSYS.

1963 DTSS (Dartmouth).

1965-2023 IBM OS/360.

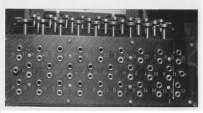
1967 Incompatible Timesharing System (ITS) at MIT, PDP-6, PDP-10.

1967 Multics (Multiplexed Information and Computer Services) (Bell Labs, GE, MIT), GE 645.

The story of operating systems - part 1

Prehistory

pre-1955 Plugboards (no OS)



1955-1960

Single batch - perforated paper paper tape, IBM, and punchcards.

Single batch - tape Unisys and magnetic tape, 128 char/inch, eventually 800 char/inch.

1956 General Motors GM-NAA for IBM 701 and then 704.

1957 Start of Compatible TimeSharing System (CTSS) development at MIT.

The 1960s

1960 SHARE Operating System (SOS), later renamed IBSYS.

1960 FORTRAN Monitor System (FMS): OS and FORTRAN compiler on same tape!

1960 Bell Monitor (BELLMON or BESYS), University of Michgan Executive System (UMES).

1961 CTSS demonstrated at MIT on IBM 709.

1963 SHARE OS taken by IBM and renamed IBSYS.

1963 DTSS (Dartmouth).

1965-2023 IBM OS/360.

1967 Incompatible Timesharing System (ITS) at MIT, PDP-6, PDP-10.

1967 Multics (Multiplexed Information and Computer Services) (Bell Labs, GE, MIT), GE 645.

The 1970s

1970 Unix, PDP-7.

1970 TOPS-10, PDP-7 and PDP-11 - DECWAR and MUD!!

1972 RSTS, PDP-11.

1972 RT-11, PDP-11 - real-time!

1972 RSX-11, PDP-11.

1972 VM/370 (also VM/CMS).

1973 Unix v.4, rewritten in C, PDP-11.

1974 CP/M, Motorola 8080.

1975 UNIX v.6.

1975 Ken Thompson sabbatical at Berkeley.

1977 1BSD UNIX released, 30 copies sent out.

1978 Unix ported to the Interdata 8/32.

1978 VMS, VAX, virtual memory and virtual machine support.

1978 UCSD p-System, PDP-11, IBM PC, Apple II, III, Lisa.

1978 Apple DOS, 6502.

```
]RUN
HELLO, APPLE II HERE
```

1979 2BSD UNIX released.

1979 3BSD UNIX (also called VMUNIX) released, VAX.

The story of operating systems - part deux

The 1980s

1980 Xenix, x86.
1981 MS-DOS 1.x
1981 XINU.
1982 Commodore DOS, 6502.
1983 Project GNU announced.
1983 Berkeley 4.2BSD UNIX: introduces sockets.
1983 AT&T System V UNIX.
1983 SunOS 1.0, 4.2BSD, 68010.
1984 Macos 1.0, 68000.
1985 SunOS 2.0, 4.2BSD, introduces NFS, YP, RPC, ...
1985 AmigaOS, AtariOS, MS-Windows 1.0.
1986 AIX, HP-UX, SunOS 3: UNIX wars begin!!
1987 IRIX, MIPS.
1987 OS/2.
1989 RiscOS, MIPS.

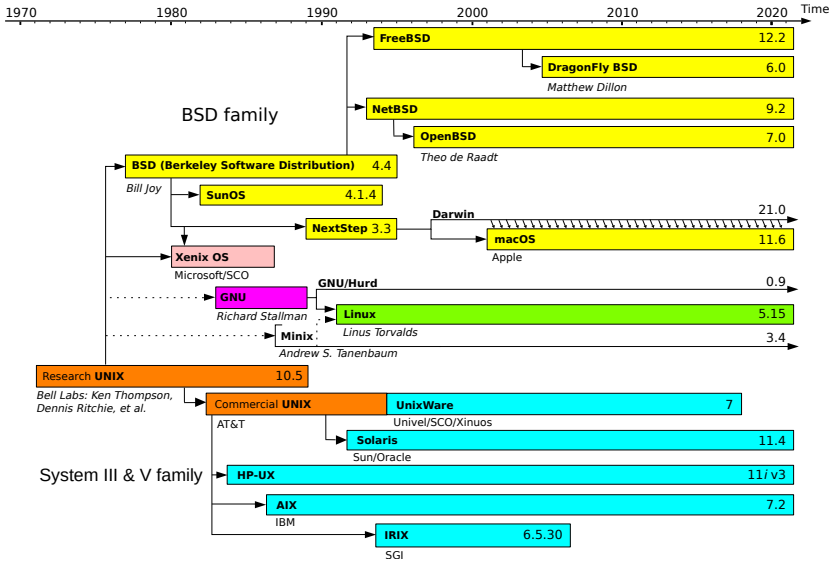
The 1990s

1991 Linux 0.1, x86.
1992 Linux 0.12, x86 - kernel now under the GPL!
1992 SunOS 5.x (Solaris), SPARC.
1992 MS-Windows 3.1.
1992 Novell NetWare 3, has TCP/IP.
1992 Plan 9.
1993 Slackware 1.0 (first linux *distribution*).
1993 MS-Windows NT.
1994 Red Hat.
1994 NetBSD 1.0, first of the FOSS BSDs.
1995 MS-Windows 95.
1995 Debian 1.1.
1996 PalmOS.
1998 eCos.

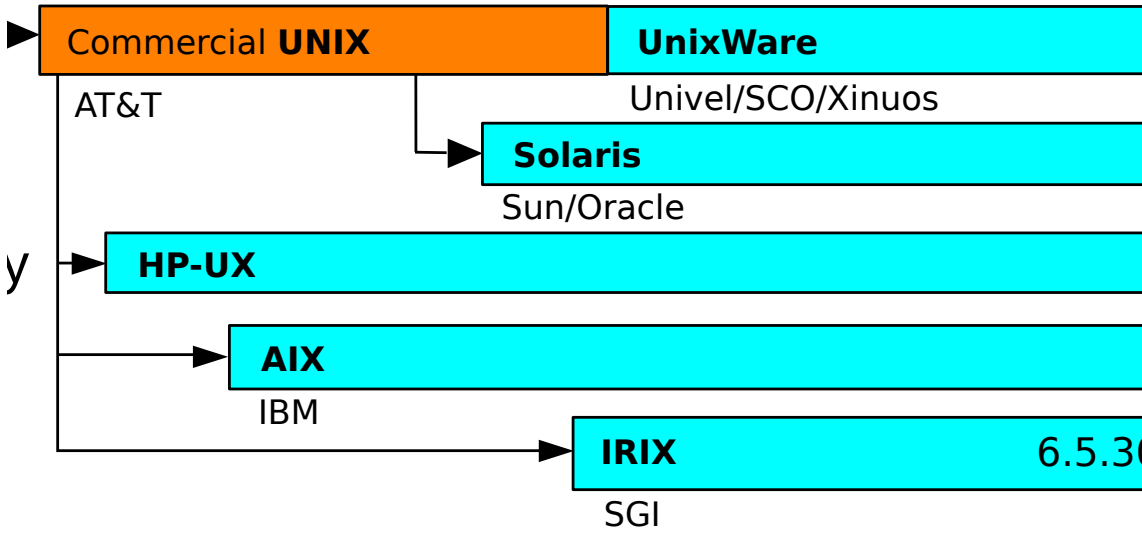
The 2000s

2001 MS-Windows XP.
2001 MacOS X 10.0, PowerPC.
2003 Fedora Core 1.
2004 Ubuntu 4.10..
2004 Reactos.
2007 iOS, ARM.
2008 Android, ARM.

The story of operating systems – UNIX timeline



The story of operating systems – UNIX timeline (zoom on UNIX wars)



UNIX wars

From Keith Packard “A Political History of X”

talk at LinuxConf Australia, 2020

A Political History of X

or

How I Stopped Worrying and Learned to Love the GPL

Keith Packard
SiFive
keithp@keithp.com

UNIX wars

From Keith Packard “A Political History of X”

talk at LinuxConf Australia, 2020

A Political History of X

or

How I Stopped Worrying and Learned to Love the GPL

Keith Packard
SiFive
keithp@keithp.com

X and UNIX show parallel dysfunction

Sun, HP, Digital, Apollo, Tektronix, IBM, MIPS, Silicon Graphics: all had their own UNIX and their own X.

Collapse of Unix

- App market failed to thrive
  - So many Unix versions
  - So many UI wants
  - So much gratuitous re-engineering
- Windows happened
  - Stupid cheap hardware
  - Completely standard ABI
  - “good enough is good enough”

UNIX wars

From Keith Packard “A Political History of X”

talk at LinuxConf Australia, 2020

A Political History of X

or

How I Stopped Worrying and Learned to Love the GPL

Keith Packard
SiFive
keithp@keithp.com

X and UNIX show parallel dysfunction

Sun, HP, Digital, Apollo, Tektronix, IBM, MIPS, Silicon Graphics: all had their own UNIX and their own X.

Collapse of Unix

- App market failed to thrive
  - So many Unix versions
  - So many UI wants
  - So much gratuitous re-engineering
- Windows happened
  - Stupid cheap hardware
  - Completely standard ABI
  - “good enough is good enough”

how it played out - and lessons

First: Windows eats the lunch of the UNIX distributions where it can.

UNIX wars

From Keith Packard “A Political History of X”

talk at LinuxConf Australia, 2020

A Political History of X

or

How I Stopped Worrying and Learned to Love the GPL

Keith Packard
SiFive
keithp@keithp.com

X and UNIX show parallel dysfunction

Sun, HP, Digital, Apollo, Tektronix, IBM, MIPS, Silicon Graphics: all had their own UNIX and their own X.

Collapse of Unix

- App market failed to thrive
  - So many Unix versions
  - So many UI wants
  - So much gratuitous re-engineering
- Windows happened
  - Stupid cheap hardware
  - Completely standard ABI
  - “good enough is good enough”

how it played out - and lessons

First: Windows eats the lunch of the UNIX distributions where it can.
Then: Linux took away the rest of UNIX’s market share, and X.org of X’s.

UNIX wars

From Keith Packard “A Political History of X”

talk at LinuxConf Australia, 2020

A Political History of X

or

How I Stopped Worrying and Learned to Love the GPL

Keith Packard
SiFive
keithp@keithp.com

X and UNIX show parallel dysfunction

Sun, HP, Digital, Apollo, Tektronix, IBM, MIPS, Silicon Graphics: all had their own UNIX and their own X.

Collapse of Unix

- App market failed to thrive
  - So many Unix versions
  - So many UI wants
  - So much gratuitous re-engineering
- Windows happened
  - Stupid cheap hardware
  - Completely standard ABI
  - “good enough is good enough”

how it played out - and lessons

First: Windows eats the lunch of the UNIX distributions where it can.
Then: Linux took away the rest of UNIX’s market share, and X.org of X’s. Anecdote of Stallman and the X project.

Distilling insight from the tour

Distilling insight from the tour

Devices and abstraction

. . . (discussion) . . .

Distilling insight from the tour

Devices and abstraction

. . . (discussion) . . .

Lockstep with hardware generations

. . . (discussion) . . .

Distilling insight from the tour

Devices and abstraction

. . . (discussion) . . .

Lockstep with hardware generations

. . . (discussion) . . .

Corporate control and licensing

. . . (discussion) . . .

Distilling insight from the tour

Devices and abstraction

. . . (discussion) . . .

Lockstep with hardware generations

. . . (discussion) . . .

Corporate control and licensing

. . . (discussion) . . .

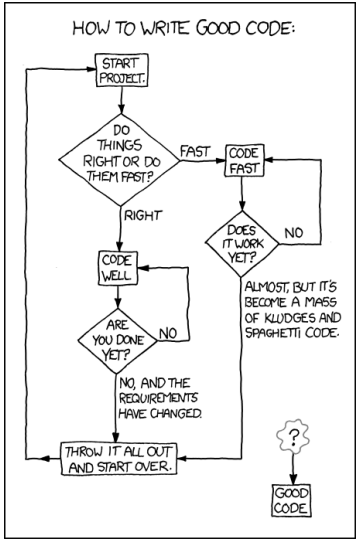
Evolution

Who influences whom?

Part IV – Methodologies

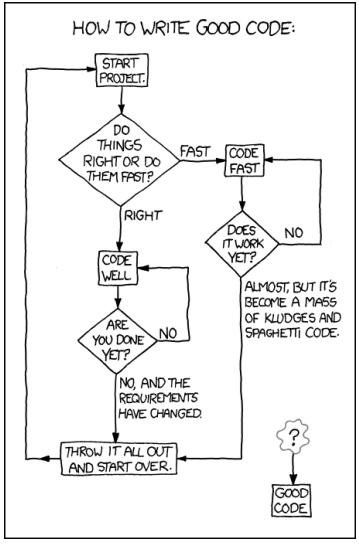
Is there a path to good code?

Naturalmente . . . xkcd: https://xkcd.com/1597/



Is there a path to good code?

Naturalmente . . . xkcd: `https://xkcd.com/1597/`



You can either hang out in the Android Loop or the HURD loop.

So, then, uhmm, what is the path to good code?

After all we have discussed:

Could it be...

- Is it a good editor?
- Is it version control?
- Is it continuous integration?
- Is it ninja debugging?
- Is it principled testing?
- Is it the Silicon Valley "ABC principle"?
- Is it recognition that one good programmer is worth 10 average programmers?
- Good management? No management?
- A big team? A small team? Just one hacker?
- Is it a good collaboration server?
- Is it another software engineering fad?
- Is it all or most of these, plus some other undiscovered ones?

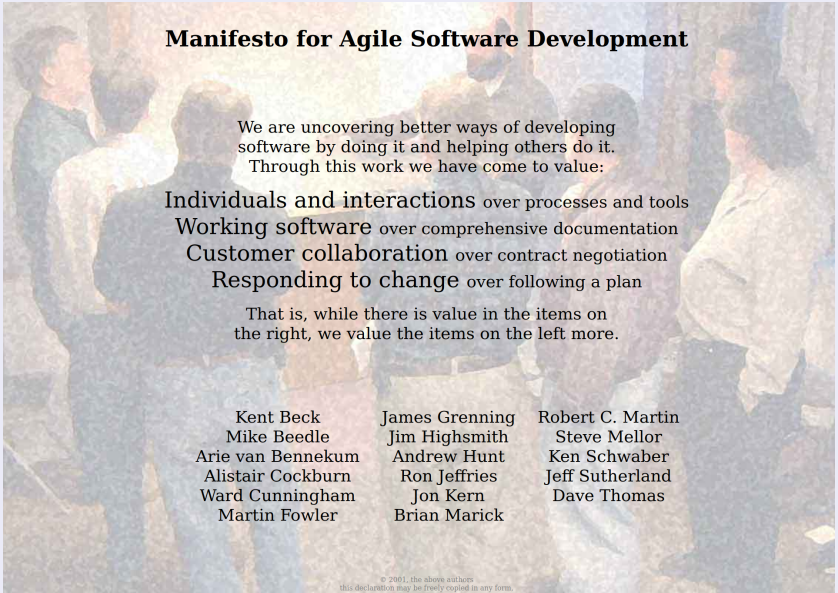
The rhythm

The rhythm

who'se to blame?

Maybe it's the movies, maybe it's the books
Maybe it's the government and all the other crooks
Maybe it's the drugs, maybe it's the parents
Maybe it's the gangs, or the colors that we're wearin'
Maybe it's the high schools, maybe it's the teachers
Tattoos, pipe bombs underneath the bleachers
Maybe it's the music, maybe it's the crack

Modern buzzword-rich methodologies: Agile



Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

The Agile “12 principles”

Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity–the art of maximizing the amount of work not done–is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

But OOP is still so. . .

OOP is still cool

OOP Is Still Cool in 2023

2023-01-05 by Tomas Tulka

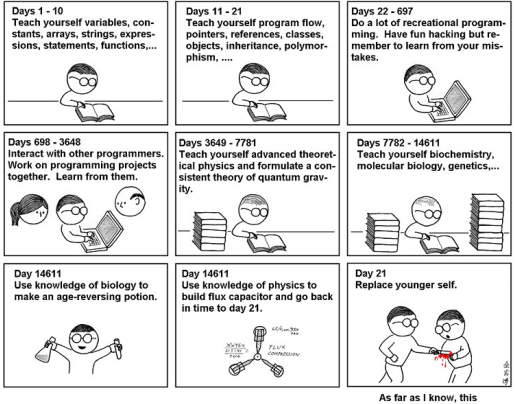
Object-oriented programming might not be perfect but it is still the best we have.

Learning C++

Abstruse Goose

«« First « Previous | Random | Next » Current »»

How to Teach Yourself Programming



As far as I know, this is the easiest way to "Teach Yourself C++ in 21 Days".

Seriously, why is everyone in **such a rush**?

Nuance in OOP from Mathew Heaney

From http://www.adapower.com/adapower1/articles/popularity.html

> Of course we know now that dynamic binding is nearly as efficient as static binding. The Smalltalk legacy lives on, however, and reuse via inheritance came to be seen as the Measure Of All Good Things.

> But there is a dark side to this, called the "fragile base class" problem. Deep inheritance hierarchies create a lot of coupling between abstractions, creating a tension between reuse and information hiding. An abstraction is basically exposing its representation by announcing that it inherits from another abstraction, and we should all know the kind of maintenance headaches you have when you don't practice information hiding.

> Thankfully, the tide seems to be turning, and people are beginning to realize that type extension is not so great after all, and that "mere" aggregation is often preferable. Deep inheritance hierarchies as a re-use mechanism may be fine for by-reference languages like Smalltalk and Eiffel, but leaf-classes in a by-value language like Ada95 or C++ become VERY SENSITIVE to the representation of the ancestor classes, which means massive re-compilations are often required any time you touch a base class. (This is the sort of problem we had for other reasons in Ada83, which motivated the inclusion of child packages in Ada95.)

Some of my opinions on project management

People matter.

Some of my opinions on project management

People matter. But why?

Some of my opinions on project management

People matter. But why?

- Cargo cult programming.
- Lack of nuance.
- Order of magnitude faster implementation.
- The silicon valley ABC principle.

Some of my opinions on project management

People matter. But why?

- Cargo cult programming.
- Lack of nuance.
- Order of magnitude faster implementation.
- The silicon valley ABC principle.

Write a lot - Michael Connelly's "murder book".

- Design documents.
- Comments in code (but not as a replacement for clean thinking).
- Emails.
- Lessons learned.
- Case studies.

Modern approaches to well-tooled chat interaction are enormously better than traditional meetings.

Management has to exist – you need both Oppenheimer and Groves, as Jeff Bloch would say – but they have to be deeply self-aware and industry-aware.

Manager should have the attitude of "the coach has to do the full warm-up run with the team." Technically they should always know when they have understood things and when they have not.

This means you cannot have a manager who is managing because they hit a technical ceiling.

Part IV – Workflow and tools

Automation and efficiency

Dave Barry, 1994-02-06

Automation and efficiency

Dave Barry, 1994-02-06

[. . .] How am I able to produce columns with such a high degree of accuracy, day in and day out, 54 weeks per year?

Automation and efficiency

Dave Barry, 1994-02-06

[. . .] How am I able to produce columns with such a high degree of accuracy, day in and day out, 54 weeks per year?

The answer is: I use a computer. This enables me to be highly efficient. Suppose, for example, that I need to fill up column space by writing BOOGER BOOGER BOOGER BOOGER BOOGER. To accomplish this in the old precomputer days, I would have had to type "BOOGER" five times manually. But now all I have to do is type it once, then simply hold the left-hand "mouse" button down while "dragging" the "mouse" so that the "cursor" moves over the text that I wish to "select"; then release the left-hand "mouse" [. . .]

Automation and efficiency .. 2

[. . .] button and position the “cursor” over the “Edit” heading on the “menu bar”; then click the left-hand “mouse” button to reveal the “edit menu”; then position the “cursor” over the “Copy” command; then click the left-hand “mouse” button; then move the “cursor” to the point where I wish to insert the “selected” text, then click the left-hand “mouse” button; then position the “cursor” over the “Edit” heading on the “menu bar” again; then click the left-hand “mouse” button to reveal the “edit menu”; then position the “cursor” over the “Paste” command; then click the left-hand “mouse” button four times; and then, as the French say, “voila!” (Literally,“My hand hurts!”)

Automation and efficiency - what is the purpose of computers?

My take on the purpose of computers

Automation and efficiency - what is the purpose of computers?

My take on the purpose of computers

The purpose of computers is to automate repetitive tasks.

Automation and efficiency - what is the purpose of computers?

My take on the purpose of computers

The purpose of computers is to automate repetitive tasks.

The purpose of computers is to automate repetitive tasks.

Automation and efficiency - what is the purpose of computers?

My take on the purpose of computers

The purpose of computers is to automate repetitive tasks.

The purpose of computers is to automate repetitive tasks.

The purpose of computers is to automate repetitive tasks.

Automation and efficiency - what is the purpose of computers?

My take on the purpose of computers

The purpose of computers is to automate repetitive tasks.

The purpose of computers is to automate repetitive tasks.

The purpose of computers is to automate repetitive tasks.

The ballad of Jack Thompson

- The magna charta.
- The Idaho retreat.
- Workshop skills.

Automation and efficiency - what is the purpose of computers?

My take on the purpose of computers

The purpose of computers is to automate repetitive tasks.

The purpose of computers is to automate repetitive tasks.

The purpose of computers is to automate repetitive tasks.

The ballad of Jack Thompson

- The magna charta.
- The Idaho retreat.
- Workshop skills.

The maxim, and how to apply it

- Maxim: You should have a running thread in your mind that is always saying “dude, should you be automating that?”

Automation and efficiency - what is the purpose of computers?

My take on the purpose of computers

The purpose of computers is to automate repetitive tasks.

The purpose of computers is to automate repetitive tasks.

The purpose of computers is to automate repetitive tasks.

The ballad of Jack Thompson

- The magna charta.
- The Idaho retreat.
- Workshop skills.

The maxim, and how to apply it

- Maxim: You should have a running thread in your mind that is always saying "dude, should you be automating that?"
- When that bell goes off, have your hacker friend on speed dial. The way to "make their day" is to ask their help in automating a task.

Automation and efficiency - what is the purpose of computers?

My take on the purpose of computers

The purpose of computers is to automate repetitive tasks.

The purpose of computers is to automate repetitive tasks.

The purpose of computers is to automate repetitive tasks.

The ballad of Jack Thompson

- The magna charta.
- The Idaho retreat.
- Workshop skills.

The maxim, and how to apply it

- Maxim: You should have a running thread in your mind that is always saying "dude, should you be automating that?"
- When that bell goes off, have your hacker friend on speed dial. The way to "make their day" is to ask their help in automating a task.
- Little by little you become the hacker on other people's speed dial, then you have a running thread in your mind saying "why do I feel so good at this validation of people asking me for help? Kind of embarrassing..."

The UNIX way – an example

The !Kung of the Kalahari desert

Getting data

Download the Howell file with data from the bushmen:

```
wget https://raw.githubusercontent.com/rmcelreath/rethinking/master/data/Howell1.csv
```

the top of the file looks like:

```
$ head Howell1.csv
"height";"weight";"age";"male"
151.765;47.8256065;63;1
139.7;36.4858065;63;0
136.525;31.864838;65;0
156.845;53.0419145;41;1
145.415;41.276872;51;0
163.83;62.992589;35;1
149.225;38.2434755;32;0
168.91;55.4799715;27;1
147.955;34.869885;19;0
```

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

```
cat Howell1.csv | wc -l
```

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

```
cat Howell1.csv | wc -l
```

How many people?

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

```
cat Howell1.csv | wc -l
```

How many people?

```
cat Howell1.csv | grep -v height | wc -l
```

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

```
cat Howell1.csv | wc -l
```

How many people?

```
cat Howell1.csv | grep -v height | wc -l
```

Who are the tallest 5 people?

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

```
cat Howell1.csv | wc -l
```

How many people?

```
cat Howell1.csv | grep -v height | wc -l
```

Who are the tallest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 1 | tail -5
```

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

```
cat Howell1.csv | wc -l
```

How many people?

```
cat Howell1.csv | grep -v height | wc -l
```

Who are the tallest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 1 | tail -5
```

Who are the oldest 5 people?

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

```
cat Howell1.csv | wc -l
```

How many people?

```
cat Howell1.csv | grep -v height | wc -l
```

Who are the tallest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 1 | tail -5
```

Who are the oldest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 3 | tail -5
```

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

```
cat Howell1.csv | wc -l
```

How many people?

```
cat Howell1.csv | grep -v height | wc -l
```

Who are the tallest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 1 | tail -5
```

Who are the oldest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 3 | tail -5
```

How many men?

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

```
cat Howell1.csv | wc -l
```

How many people?

```
cat Howell1.csv | grep -v height | wc -l
```

Who are the tallest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 1 | tail -5
```

Who are the oldest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 3 | tail -5
```

How many men?

```
cat Howell1.csv | grep '1$' | wc -l
```

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

```
cat Howell1.csv | wc -l
```

How many people?

```
cat Howell1.csv | grep -v height | wc -l
```

Who are the tallest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 1 | tail -5
```

Who are the oldest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 3 | tail -5
```

How many men?

```
cat Howell1.csv | grep '1$' | wc -l
```

How many women?

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

```
cat Howell1.csv | wc -l
```

How many people?

```
cat Howell1.csv | grep -v height | wc -l
```

Who are the tallest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 1 | tail -5
```

Who are the oldest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 3 | tail -5
```

How many men?

```
cat Howell1.csv | grep '1$' | wc -l
```

How many women?

```
cat Howell1.csv | grep '0$' | wc -l
```

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

```
cat Howell1.csv | wc -l
```

How many people?

```
cat Howell1.csv | grep -v height | wc -l
```

Who are the tallest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 1 | tail -5
```

Who are the oldest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 3 | tail -5
```

How many men?

```
cat Howell1.csv | grep '1$' | wc -l
```

How many women?

```
cat Howell1.csv | grep '0$' | wc -l
```

What is the average age?

The UNIX way – asking questions about a text file

Can I look at that file a bit better?

```
cat Howell1.csv | sed 's/;/    /g'
cat Howell1.csv | sed 's/;/    /g' | less
```

How many lines?

```
cat Howell1.csv | wc -l
```

How many people?

```
cat Howell1.csv | grep -v height | wc -l
```

Who are the tallest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 1 | tail -5
```

Who are the oldest 5 people?

```
cat Howell1.csv | grep -v height | sed 's/;/    /g' | sort -n -k 3 | tail -5
```

How many men?

```
cat Howell1.csv | grep '1$' | wc -l
```

How many women?

```
cat Howell1.csv | grep '0$' | wc -l
```

What is the average age?

```
cat Howell1.csv | grep -v height | sed 's/;/ /g' | awk '{sum+=$3} END {print "AVG =",sum/NR}'
```

The UNIX way – example of web scraping

Build up a web scraping filter

Listing 1: Anatomy of a web scraping pipeline

```
wget -O - https://www.verywellfamily.com/top-1000-baby-girl-names-2757832 | lynx -stdin --dump
wget -O - https://www.verywellfamily.com/top-1000-baby-girl-names-2757832 | lynx -stdin --dump | less
wget -O - https://www.verywellfamily.com/top-1000-baby-girl-names-2757832 | lynx -stdin --dump | grep '. ' |
      grep '^ *[0-9]'
wget -O - https://www.verywellfamily.com/top-1000-baby-girl-names-2757832 | lynx -stdin --dump | grep '. ' |
      grep '^ *[0-9]' | grep -v http
wget -O - https://www.verywellfamily.com/top-1000-baby-girl-names-2757832 | lynx -stdin --dump | grep '. ' |
      grep '^ *[0-9]' | grep -v http | grep -v file: | grep -v about:
wget -O - https://www.verywellfamily.com/top-1000-baby-girl-names-2757832 | lynx -stdin --dump | grep '. ' |
      grep '^ *[0-9]' | grep -v http | grep -v file: | grep -v about: | grep '[0-9]\.'
NAMES=`wget -O - https://www.verywellfamily.com/top-1000-baby-girl-names-2757832 | lynx -stdin --dump |
       grep '. ' | grep '^ *[0-9]' | grep -v http | grep -v file: | grep -v about: | grep '[0-9]\.'`
echo $NAMES
## now you can go to town on this
```

The UNIX way – what is it?

Philosophy at the user level

- Your rightful place is in the command line.
- Redirection (< and >) and pipes (|) are wonderful.
- Use many small programs which interact together to form **pipelines**.
- People breezily say "just use sed and awk" – thanks to the examples above we now know what they mean.
- Use graphical and integrated utilities with suspicion.
- grep, sed, awk, wc, wget, youtube-dl, ...

The UNIX way – what is it?

Philosophy at the user level

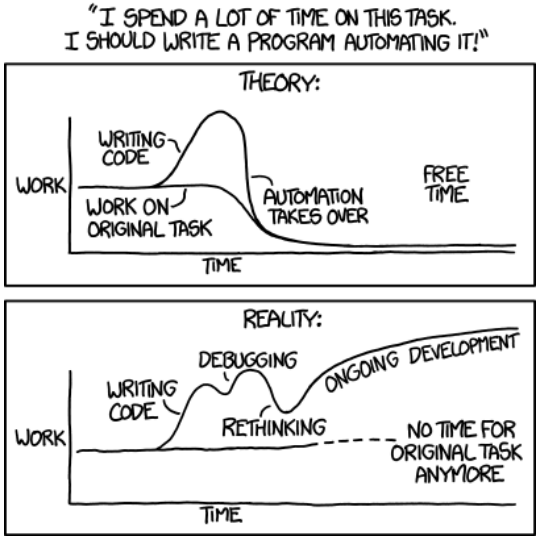
- Your rightful place is in the command line.
- Redirection (< and >) and pipes (|) are wonderful.
- Use many small programs which interact together to form **pipelines**.
- People breezily say "just use sed and awk" – thanks to the examples above we now know what they mean.
- Use graphical and integrated utilities with suspicion.
- grep, sed, awk, wc, wget, youtube-dl, ...

Philosophy at the programmer level

- Don't write huge programs: write small programs that can be put together as a pipeline.
- Use scripting languages like Python to glue together compiled programs.

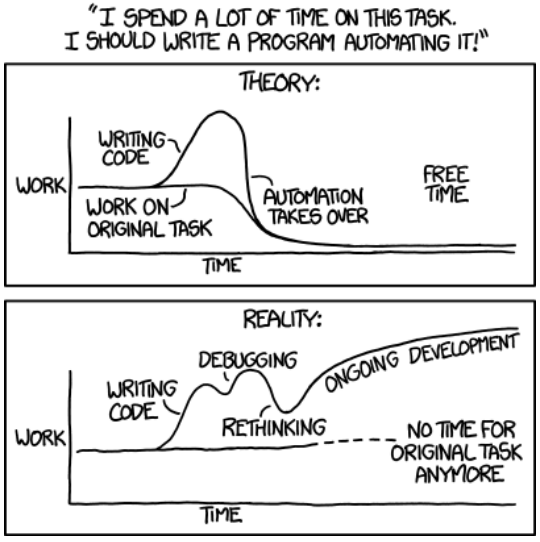
Perils of a nerd automating a task

Naturalmente . . . xkcd: https://xkcd.com/1319/



Perils of a nerd automating a task

Naturalmente . . . xkcd: https://xkcd.com/1319/



'Automating' comes from the roots 'auto-' meaning 'self-', and '******, meaning ********.

Command line diversions

Partially from https://www.binarytides.com/linux-fun-commands/

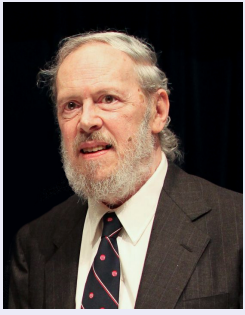
One-line ascii art

```
echo an example of figlet | figlet
banner "have a nice day"
cowsay hey dude
cowsay -f dragon "Run for cover, I feel a sneeze coming on."
cowsay -l
cowsay -f ghostbusters Who you Gonna Call
sl
fortune
factor 12103  # factoring numbers? can we use this to search for Mersenne primes?
factor `echo "2^7-1" | bc` ; factor `echo "2^11-1" | bc` ; factor `echo "2^13-1" | bc`
pi 50
espeak "Hello Linux, where are the penguins"
telnet towel.blinkenlights.nl
```

jpeg to ascii

```
wget https://upload.wikimedia.org/wikipedia/commons/2/23/Dennis_Ritchie_2011.jpg
## make your terminal very big and try
jp2a -f Dennis_Ritchie_2011.jpg
jp2a -f --color Dennis_Ritchie_2011.jpg
```

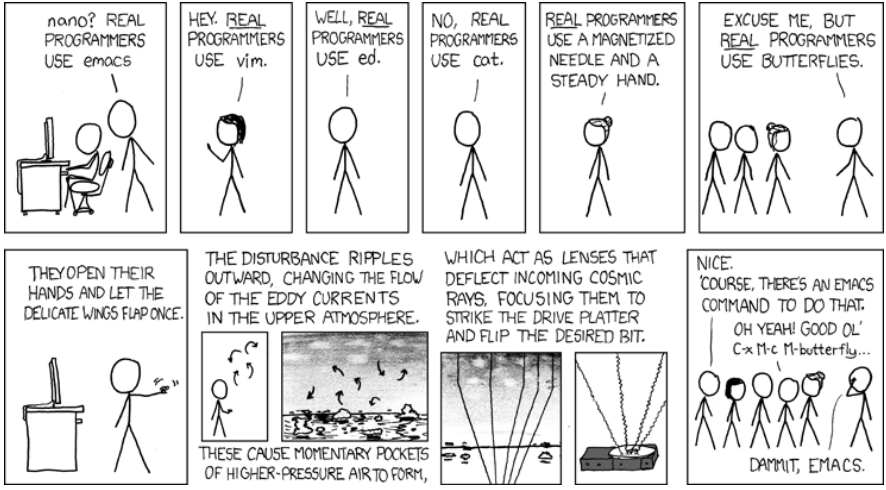
Dennis Ritchie



Dennis Ritchie who created the C programming language and co-created UNIX. Let's make ascii art of him.

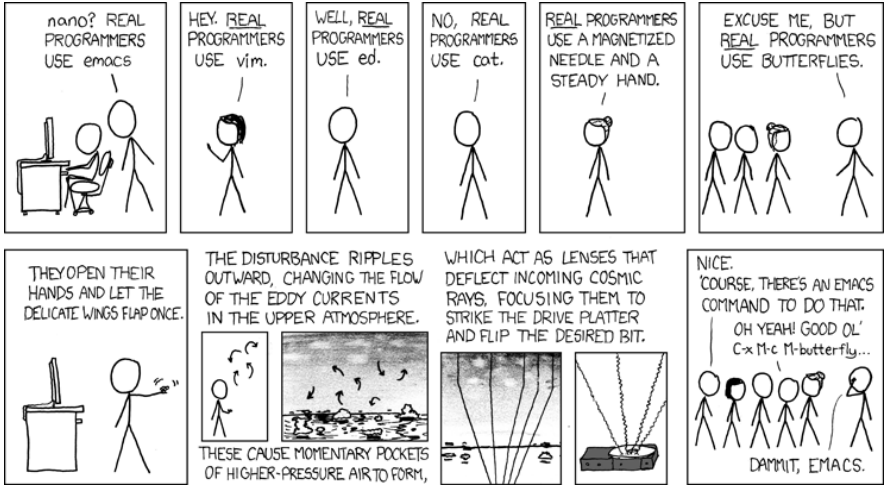
Editor wars

Naturalmente . . . xkcd: https://xkcd.com/378/



Editor wars

Naturalmente . . . xkcd: https://xkcd.com/378/



Real programmers set the universal constants at the start such that the universe evolves to contain the disk with the data they want.

Emacs vs. vi

Two guys are sitting in a bar, and get talking.
"What's you IQ?" one asks.
"169" is the reply.
"Wow, amazing — my IQ's 172. What're your ideas on Hawking's latest work on black hole evaporation?"
And the two get chatting and become lifelong friends.

Emacs vs. vi

Two guys are sitting in a bar, and get talking.
"What's you IQ?" one asks.
"169" is the reply.
"Wow, amazing — my IQ's 172. What're your ideas on Hawking's latest work on black hole evaporation?"
And the two get chatting and become lifelong friends.

Further down the bar, two other guys are comparing IQs.
"Mine's 104"
"Gosh, mine's 102. What do you think about the latest Cubs game?" And the two become lifelong friends.

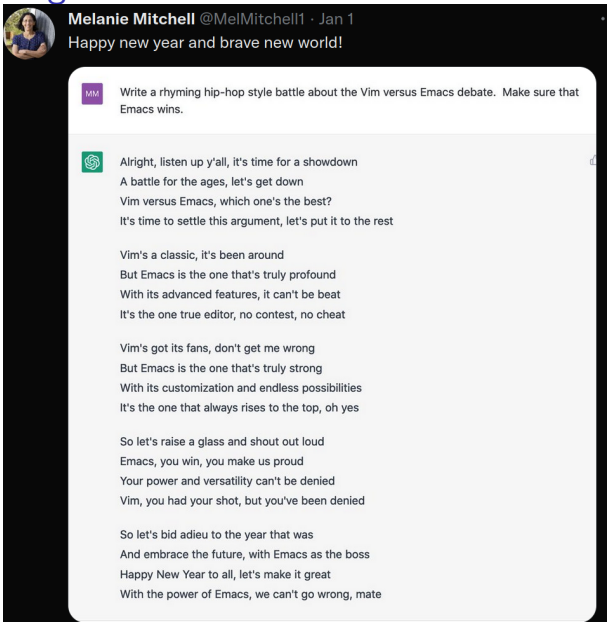
Emacs vs. vi

Two guys are sitting in a bar, and get talking.
"What's you IQ?" one asks.
"169" is the reply.
"Wow, amazing — my IQ's 172. What're your ideas on Hawking's latest work on black hole evaporation?"
And the two get chatting and become lifelong friends.

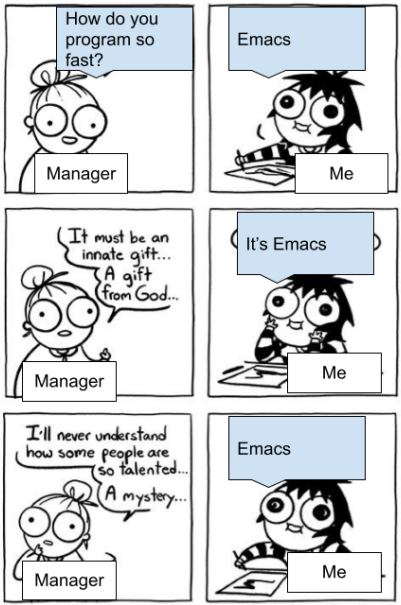
Further down the bar, two other guys are comparing IQs.
"Mine's 104"
"Gosh, mine's 102. What do you think about the latest Cubs game?" And the two become lifelong friends.

Even further down the bar, two other guys are also comparing IQs.
"Mine's 53."
"Wow! Mine's 54. Do you use emacs or vi?"

emacs vs. vi in the age of ChatGPT



emacs vs. vi – productivity



Tour of programming editors

. . . and integrated development environments (IDEs)

Editors

- Personal preference: emacs
- Personal preference: vi for quickies.
- Truth is: vim is full-featured and even getting slow!

IDEs

- Eclipse.
- VSCode.
- Smaller ones, like kdevelop.

Version control (VC) – generalities

Motivation and history

Version control (VC) – generalities

Motivation and history

Importance of tracking changes and reproducing old versions.

Version control (VC) – generalities

Motivation and history

Importance of tracking changes and reproducing old versions.

Technical coordination of contributions from other programmers.

Version control (VC) – generalities

Motivation and history

Importance of tracking changes and reproducing old versions.

Technical coordination of contributions from other programmers.

1972 SCCS, Marc Rochkind, Bell Labs
1982 RCS
1990 CVS
2000 Subversion
2005 mercurial
2005 git, Linus Torvalds

Version control (VC) – generalities

Motivation and history

Importance of tracking changes and reproducing old versions.

Technical coordination of contributions from other programmers.

1972 SCCS, Marc Rochkind, Bell Labs
1982 RCS
1990 CVS
2000 Subversion
2005 mercurial
2005 git, Linus Torvalds

"Social networking" VC sites

Web sites that add wikis, bug tracking, other collaboration features.

Version control (VC) – generalities

Motivation and history

Importance of tracking changes and reproducing old versions.

Technical coordination of contributions from other programmers.

1972 SCCS, Marc Rochkind, Bell Labs
1982 RCS
1990 CVS
2000 Subversion
2005 mercurial
2005 git, Linus Torvalds

"Social networking" VC sites

Web sites that add wikis, bug tracking, other collaboration features.

1998 sourceware.cygnus.com
1999 sourceforge.net
2008 github
2011 gitlab
2012 bitbucket

Version control – git workflow

From https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html

One time setup

```
git config --global user.name "FirstName LastName"
git config --global user.email "user@domain.tld"
git config --global --list
```

One time per project

If you are creating a new project:

```
git init .
```

If you are cloning an existing project from somwhere:

```
git clone git@someplace.tld:/path/to/master/reponame
cd reponame
```

One time when you add new files

```
echo "int main() { return 0; }" > trivial.c
git add trivial.c
```

Version control – git workflow (continued)

From https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html

Daily work flow

1. git pull ## pulls in what other people have been doing
2. Edit code and save.
3. git commit -a
4. git push ## synchronize out to other people's code

Taking stock

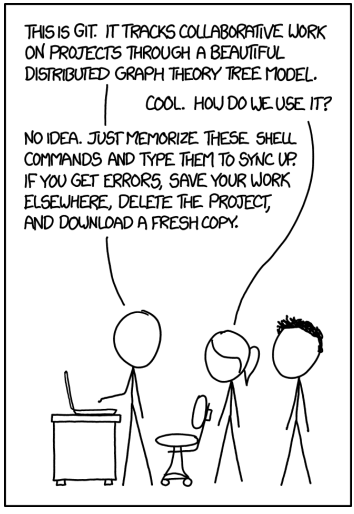
1. git log ## detailed information on what's been happening
2. git tag release-1.5 ## reproducibly define a release

But...

Git was created by Linus Torvalds to develop the Linux kernel. It is poorly suited to most people's work flows. Still, it is the most used.
Only (and very good) alternative: Mercurial, but much less used.

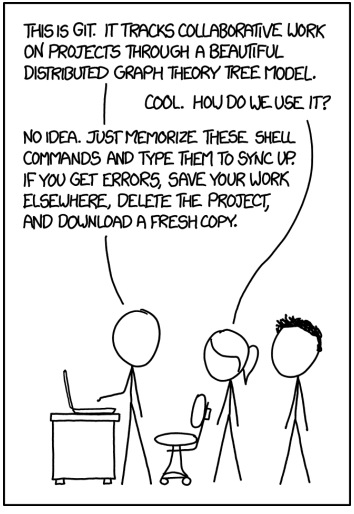
Version control – git - be skeptical

Naturalmente . . . xkcd: https://xkcd.com/1597/



Version control – git - be skeptical

Naturalmente . . . xkcd: https://xkcd.com/1597/



If that doesn't fix it, git.txt contains the phone number of a friend of mine who understands git. Just wait through a few minutes of 'It's really pretty simple, just think of branches as...' and eventually you'll learn the commands that will fix everything.

The need for debugging

A memory error

Worked example of simple program that blows past the limits on an array.

```
/* compile with "gcc -g -fno-stack-protector mem-trash.c -o mem-trash", run with "./mem-trash" */
#include <stdio.h>
#include <string.h>

int main()
{
  char my_string[9];
  int important_array[8];
  int crucial_value;
  int i;

  crucial_value = 42;
  printf("just set crucial_value to: %d\n", crucial_value);
  strcpy(my_string, "this is a string that is longer than what I have allocated for it");
  printf("Just set my_string to be <%s>\n", my_string);
  printf("After setting my_string, crucial_value is: %d\n", crucial_value);
  for (i = 0; i < 8; ++i) {
    important_array[i] = i*i; /* fill this important array with the squares of numbers */
  }
  printf("After setting the array, my_string is <%s>\n", my_string);
}
```

The need for debugging

A memory error

Worked example of simple program that blows past the limits on an array.

```c
/* compile with "gcc -g -fno-stack-protector mem-trash.c -o mem-trash", run with "./mem-trash" */
#include <stdio.h>
#include <string.h>

int main()
{
  char my_string[9];
  int important_array[8];
  int crucial_value;
  int i;

  crucial_value = 42;
  printf("just set crucial_value to: %d\n", crucial_value);
  strcpy(my_string, "this is a string that is longer than what I have allocated for it");
  printf("Just set my_string to be <%s>\n", my_string);
  printf("After setting my_string, crucial_value is: %d\n", crucial_value);
  for (i = 0; i < 8; ++i) {
    important_array[i] = i*i; /* fill this important array with the squares of numbers */
  }
  printf("After setting the array, my_string is <%s>\n", my_string);
}
```

Output

```
$ ./mem-trash
just set crucial_value to: 42
Just set my_string to be <this is a string that is longer than what I have allocated for it>
After setting my_string, crucial_value is: 1920234272
After setting the array, my_string is <this is a st>
Segmentation fault (core dumped)
```

Source level debugging

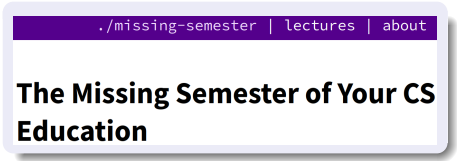
gdb for C

Share a terminal session to run this gdb example:

```
$ gdb mem-trash
(gdb) break main
Breakpoint 1 at 0x652: file mem-trash.c, line 12.
(gdb) run
Starting program: /home/markgalassi/repo/talks/2020-05-sfps-professional-development/mem-trash
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at mem-trash.c:12
12      crucial_value = 42;
(gdb) next
13      printf("just set crucial_value to: %d\n", crucial_value);
(gdb) next
just set crucial_value to: 42
14      strcpy(my_string, "this is a string that is longer than what I have allocated for it");
(gdb) next
15      printf("Just set my_string to be <%s>\n", my_string);
(gdb) next
Just set my_string to be <this is a string that is longer than what I have allocated for it>
16      printf("After setting my_string, crucial_value is: %d\n", crucial_value);
(gdb) print crucial_value
$1 = 1920234272
(gdb) next
After setting my_string, crucial_value is: 1920234272
17      for (i = 0; i < 8; ++i) {
## and so forth
```

The MIT “missing semester”



The MIT “missing semester”

./missing-semester | lectures | about

The Missing Semester of Your CS Education

Schedule

1/13/20: Course overview + the shell
1/14/20: Shell Tools and Scripting
1/15/20: Editors (Vim)
1/16/20: Data Wrangling
1/21/20: Command-line Environment
1/22/20: Version Control (Git)
1/23/20: Debugging and Profiling
1/27/20: Metaprogramming
1/28/20: Security and Cryptography
1/29/20: Potpourri
1/30/20: Q&A

The MIT "missing semester"

./missing-semester | lectures | about

The Missing Semester of Your CS Education

Schedule

1/13/20: Course overview + the shell
1/14/20: Shell Tools and Scripting
1/15/20: Editors (Vim)
1/16/20: Data Wrangling
1/21/20: Command-line Environment
1/22/20: Version Control (Git)
1/23/20: Debugging and Profiling
1/27/20: Metaprogramming
1/28/20: Security and Cryptography
1/29/20: Potpourri
1/30/20: Q&A

Classes teach [...] but there's one critical subject that's rarely covered [...]: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Computers were built to automate manual tasks, yet students often perform repetitive tasks by hand or fail to take full advantage of powerful tools such as version control and text editors. In the best case, this results in inefficiencies and wasted time; in the worst case, it results in issues like data loss or inability to complete certain tasks.

What does the world do with computers

The picture is different from what you might think

By number of units: embedded

The data in 2000 CE was that less than 1% of computing power was on desktop/laptop computers.

By computing power: server farms

Numbers hard to get: cagey cloud computing vendors.
Amazon, Microsoft, Google "bet the farm" on cloud platform - 90% of Microsoft R&D was for its cloud.

The bones of the world

What do computers actually do in the world? – possible categorization

- agricultural machinery
- civil infrastructure
- personal desktop
- factory automation
- vehicle control
- personal laptop
- office desktop
- home automation
- very small embedded
- web server
- supercomputers
- personal mobile
- engineering workstation
- networking infrastructure

The bones of the world

What do computers actually do in the world? – **home user** awareness

agricultural machinery

civil infrastructure

personal desktop

factory automation

vehicle control

personal laptop

office desktop

home automation

very small embedded

web server

supercomputers

personal mobile

engineering workstation

networking infrastructure

The bones of the world

What do computers actually do in the world? – **mechanical engineer** awareness

agricultural machinery

civil infrastructure

personal desktop

factory automation

vehicle control

personal laptop

office desktop

home automation

very small embedded

web server

supercomputers

personal mobile

engineering workstation

networking infrastructure

The bones of the world

What do computers actually do in the world? – **electrical engineer** awareness

agricultural machinery

civil infrastructure

personal desktop

factory automation

vehicle control

personal laptop

office desktop

home automation

very small embedded

web server

supercomputers

personal mobile

engineering workstation

networking infrastructure

The bones of the world

What do computers actually do in the world? – **scientist** awareness

agricultural machinery

civil infrastructure

personal desktop

factory automation

vehicle control

personal laptop

office desktop

home automation

very small embedded

web server

supercomputers

personal mobile

engineering workstation

networking infrastructure

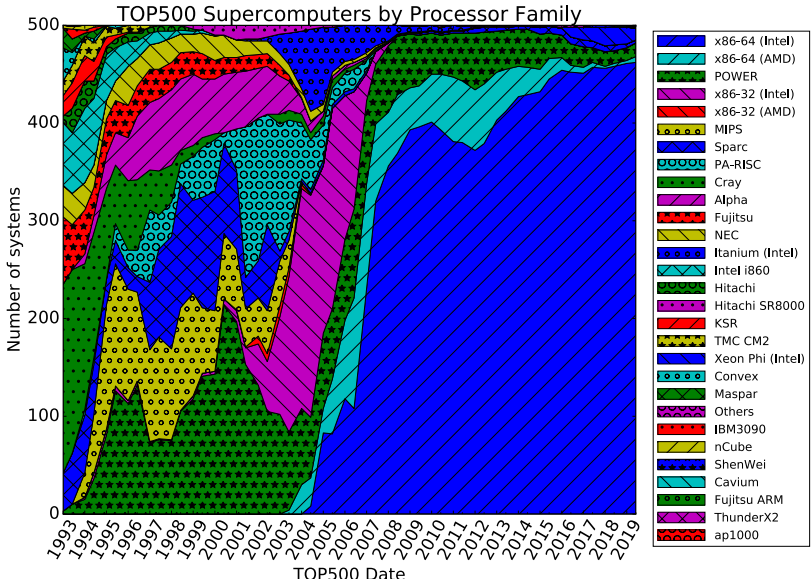
The bones of the world

What do computers actually do in the world? – **software engineer** awareness

- agricultural machinery
- civil infrastructure
- personal desktop
- factory automation
- vehicle control
- personal laptop
- office desktop
- home automation
- very small embedded
- web server
- supercomputers
- personal mobile
- engineering workstation
- networking infrastructure

Supercomputer hardware type evolution

From https://en.wikipedia.org/wiki/TOP500#/media/File:Processor_families_in_TOP500_supercomputers.svg



Case study: The Roots of Beowulf

From https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150001285.pdf

Case study: The Roots of Beowulf

From https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150001285.pdf

NASA Goddard Space Flight Center
Code 606
Greenbelt, MD 20771 USA
1-301-286-3465
james.r.fischer@nasa.gov

ABSTRACT

The first Beowulf Linux commodity cluster was constructed at NASA's Goddard Space Flight Center in 1994 and its origins are a part of the folklore of high-end computing. In fact, the conditions within Goddard that brought the idea into being were shaped by rich historical roots, strategic pressures brought on by the ramp up of the Federal High-Performance Computing and Communications Program, growth of the open software movement, microprocessor performance trends, and the vision of key technologists. This multifaceted story is told here for the first time from the point of view of NASA project management.

Flight Center specifically, but the experience was universal. It is only 20 years ago, but the impediments facing those who needed high-end computing are somewhat incomprehensible today if you were not there and may be best forgotten if you were.

2.1 Proprietary Stove Piped Systems

Every system that we could buy ran proprietary system software on proprietary hardware.

Competing vendors' operating environments were, in many cases, extremely incompatible—changing to a different vendor could be traumatic.

Case study: The Roots of Beowulf

From https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150001285.pdf

NASA Goddard Space Flight Center
Code 606
Greenbelt, MD 20771 USA
1-301-286-3465
james.r.fischer@nasa.gov

ABSTRACT

The first Beowulf Linux commodity cluster was constructed at NASA's Goddard Space Flight Center in 1994 and its origins are a part of the folklore of high-end computing. In fact, the conditions within Goddard that brought the idea into being were shaped by rich historical roots, strategic pressures brought on by the ramp up of the Federal High-Performance Computing and Communications Program, growth of the open software movement, microprocessor performance trends, and the vision of key technologists. This multifaceted story is told here for the first time from the point of view of NASA project management.

Flight Center specifically, but the experience was universal. It is only 20 years ago, but the impediments facing those who needed high-end computing are somewhat incomprehensible today if you were not there and may be best forgotten if you were.

2.1 Proprietary Stove Piped Systems

Every system that we could buy ran proprietary system software on proprietary hardware.

Competing vendors' operating environments were, in many cases, extremely incompatible—changing to a different vendor could be traumatic.

. . .

Computing

General Terms
Management

Keywords
Beowulf, cluster, HPCC, NASA, Goddard, Sterling, Becker, Dorband, Nickolls, Massively Parallel Processor, MPP, MasPar, SIMD, Blitzen

1. INTRODUCTION

Looking back to the origins of the Beowulf cluster computing movement in 1993, it is well known that the driving force was NASA's stated need for a gigaflops workstation costing less than $50,000. That is true, but the creative conversations that brought

equivalent of a full-time employee." [1]

In 1992, some facilities used clusters of workstations to offload supercomputers and harvest wasted cycles; at Goddard we salivated at this idea but had few high-end workstations to use.

The very high and rising cost of each new generation of supercomputer development forced vendors to pass along those costs to customers. The vendors could inflate their prices because they were only competing with other vendors doing the same thing.

2.3 Numerous Performance Choke Points

In 1991, the Intel Touchstone Delta at Caltech was the top machine in the world, but compilation had to be done on Sun workstations using proprietary system software that only ran on Suns.

Case study: The Roots of Beowulf

From https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150001285.pdf

NASA Goddard Space Flight Center
Code 606
Greenbelt, MD 20771 USA
1-301-286-3465
james.r.fischer@nasa.gov

ABSTRACT

The first Beowulf Linux commodity cluster was constructed at NASA's Goddard Space Flight Center in 1994 and its origins are a part of the folklore of high-end computing. In fact, the conditions within Goddard that brought the idea into being were shaped by rich historical roots, strategic pressures brought on by the ramp up of the Federal High-Performance Computing and Communications Program, growth of the open software movement, microprocessor performance trends, and the vision of key technologists. This multifaceted story is told here for the first time from the point of view of NASA project management.

Flight Center specifically, but the experience was universal. It is only 20 years ago, but the impediments facing those who needed high-end computing are somewhat incomprehensible today if you were not there and may be best forgotten if you were.

2.1 Proprietary Stove Piped Systems

Every system that we could buy ran proprietary system software on proprietary hardware.

Competing vendors' operating environments were, in many cases, extremely incompatible—changing to a different vendor could be traumatic.

. . .

Computing

General Terms
Management

Keywords
Beowulf, cluster, HPCC, NASA, Goddard, Sterling, Becker, Dorband, Nickolls, Massively Parallel Processor, MPP, MasPar, SIMD, Blitzen

1. INTRODUCTION

Looking back to the origins of the Beowulf cluster computing movement in 1993, it is well known that the driving force was NASA's stated need for a gigaflops workstation costing less than $50,000. That is true, but the creative conversations that brought

equivalent of a full-time employee." [1]

In 1992, some facilities used clusters of workstations to offload supercomputers and harvest wasted cycles; at Goddard we salivated at this idea but had few high-end workstations to use.

The very high and rising cost of each new generation of supercomputer development forced vendors to pass along those costs to customers. The vendors could inflate their prices because they were only competing with other vendors doing the same thing.

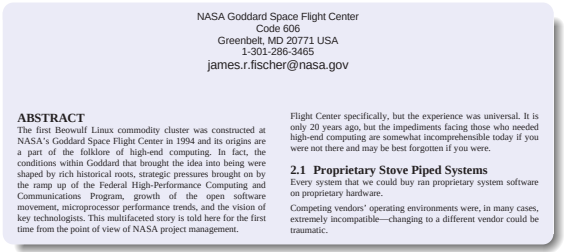
2.3 Numerous Performance Choke Points

In 1991, the Intel Touchstone Delta at Caltech was the top machine in the world, but compilation had to be done on Sun workstations using proprietary system software that only ran on Suns.

Discussion points

Case study: The Roots of Beowulf

From https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150001285.pdf



NASA Goddard Space Flight Center
Code 606
Greenbelt, MD 20771 USA
1-301-286-3465
james.r.fischer@nasa.gov

ABSTRACT

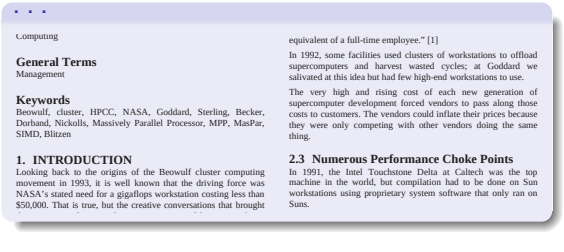
The first Beowulf Linux commodity cluster was constructed at NASA's Goddard Space Flight Center in 1994 and its origins are a part of the folklore of high-end computing. In fact, the conditions within Goddard that brought the idea into being were shaped by rich historical roots, strategic pressures brought on by the ramp up of the Federal High-Performance Computing and Communications Program, growth of the open software movement, microprocessor performance trends, and the vision of key technologists. This multifaceted story is told here for the first time from the point of view of NASA project management.

Flight Center specifically, but the experience was universal. It is only 20 years ago, but the impediments facing those who needed high-end computing are somewhat incomprehensible today if you were not there and may be best forgotten if you were.

2.1 Proprietary Stove Piped Systems

Every system that we could buy ran proprietary system software on proprietary hardware.

Competing vendors' operating environments were, in many cases, extremely incompatible—changing to a different vendor could be traumatic.



. . .

Computing

General Terms
Management

Keywords
Beowulf, cluster, HPCC, NASA, Goddard, Sterling, Becker, Dorband, Nickolls, Massively Parallel Processor, MPP, MasPar, SIMD, Blitzen

1. INTRODUCTION

Looking back to the origins of the Beowulf cluster computing movement in 1993, it is well known that the driving force was NASA's stated need for a gigaflops workstation costing less than $50,000. That is true, but the creative conversations that brought

equivalent of a full-time employee." [1]

In 1992, some facilities used clusters of workstations to offload supercomputers and harvest wasted cycles; at Goddard we salivated at this idea but had few high-end workstations to use.

The very high and rising cost of each new generation of supercomputer development forced vendors to pass along those costs to customers. The vendors could inflate their prices because they were only competing with other vendors doing the same thing.

2.3 Numerous Performance Choke Points

In 1991, the Intel Touchstone Delta at Caltech was the top machine in the world, but compilation had to be done on Sun workstations using proprietary system software that only ran on Suns.

Discussion points

- What were the problems in supercomputing, and how did Beowulf and Beowulf Blade clusters address them?

Case study: The Roots of Beowulf

From https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150001285.pdf

NASA Goddard Space Flight Center
Code 606
Greenbelt, MD 20771 USA
1-301-286-3465
james.r.fischer@nasa.gov

ABSTRACT

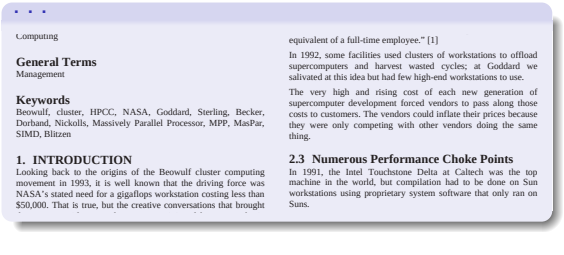
The first Beowulf Linux commodity cluster was constructed at NASA's Goddard Space Flight Center in 1994 and its origins are a part of the folklore of high-end computing. In fact, the conditions within Goddard that brought the idea into being were shaped by rich historical roots, strategic pressures brought on by the ramp up of the Federal High-Performance Computing and Communications Program, growth of the open software movement, microprocessor performance trends, and the vision of key technologists. This multifaceted story is told here for the first time from the point of view of NASA project management.

Flight Center specifically, but the experience was universal. It is only 20 years ago, but the impediments facing those who needed high-end computing are somewhat incomprehensible today if you were not there and may be best forgotten if you were.

2.1 Proprietary Stove Piped Systems

Every system that we could buy ran proprietary system software on proprietary hardware.

Competing vendors' operating environments were, in many cases, extremely incompatible—changing to a different vendor could be traumatic.



. . .

Computing

General Terms
Management

Keywords
Beowulf, cluster, HPCC, NASA, Goddard, Sterling, Becker, Dorband, Nickolls, Massively Parallel Processor, MPP, MasPar, SIMD, Blitzen

1. INTRODUCTION

Looking back to the origins of the Beowulf cluster computing movement in 1993, it is well known that the driving force was NASA's stated need for a gigaflops workstation costing less than $50,000. That is true, but the creative conversations that brought

equivalent of a full-time employee." [1]

In 1992, some facilities used clusters of workstations to offload supercomputers and harvest wasted cycles; at Goddard we salivated at this idea but had few high-end workstations to use.

The very high and rising cost of each new generation of supercomputer development forced vendors to pass along those costs to customers. The vendors could inflate their prices because they were only competing with other vendors doing the same thing.

2.3 Numerous Performance Choke Points

In 1991, the Intel Touchstone Delta at Caltech was the top machine in the world, but compilation had to be done on Sun workstations using proprietary system software that only ran on Suns.

Discussion points

- What were the problems in supercomputing, and how did Beowulf and Beowulf Blade clusters address them?
- How does Beowulf's arrival fit into the plot in (Frame 87)

Case study: The Roots of Beowulf

From https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150001285.pdf

NASA Goddard Space Flight Center
Code 606
Greenbelt, MD 20771 USA
1-301-286-3465
james.r.fischer@nasa.gov

ABSTRACT

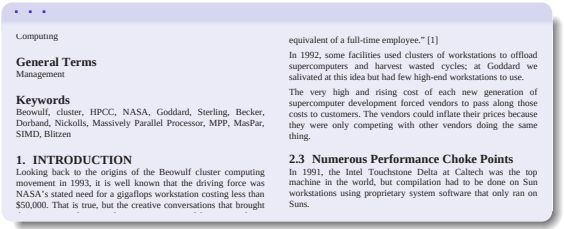
The first Beowulf Linux commodity cluster was constructed at NASA's Goddard Space Flight Center in 1994 and its origins are a part of the folklore of high-end computing. In fact, the conditions within Goddard that brought the idea into being were shaped by rich historical roots, strategic pressures brought on by the ramp up of the Federal High-Performance Computing and Communications Program, growth of the open software movement, microprocessor performance trends, and the vision of key technologists. This multifaceted story is told here for the first time from the point of view of NASA project management.

Flight Center specifically, but the experience was universal. It is only 20 years ago, but the impediments facing those who needed high-end computing are somewhat incomprehensible today if you were not there and may be best forgotten if you were.

2.1 Proprietary Stove Piped Systems

Every system that we could buy ran proprietary system software on proprietary hardware.

Competing vendors' operating environments were, in many cases, extremely incompatible—changing to a different vendor could be traumatic.



. . .

Computing

General Terms
Management

Keywords
Beowulf, cluster, HPCC, NASA, Goddard, Sterling, Becker, Dorband, Nickolls, Massively Parallel Processor, MPP, MasPar, SIMD, Blitzen

1. INTRODUCTION

Looking back to the origins of the Beowulf cluster computing movement in 1993, it is well known that the driving force was NASA's stated need for a gigaflops workstation costing less than $50,000. That is true, but the creative conversations that brought

equivalent of a full-time employee." [1]

In 1992, some facilities used clusters of workstations to offload supercomputers and harvest wasted cycles; at Goddard we salivated at this idea but had few high-end workstations to use.

The very high and rising cost of each new generation of supercomputer development forced vendors to pass along those costs to customers. The vendors could inflate their prices because they were only competing with other vendors doing the same thing.

2.3 Numerous Performance Choke Points

In 1991, the Intel Touchstone Delta at Caltech was the top machine in the world, but compilation had to be done on Sun workstations using proprietary system software that only ran on Suns.

Discussion points

- What were the problems in supercomputing, and how did Beowulf and Beowulf Blade clusters address them?
- How does Beowulf's arrival fit into the plot in (Frame 87)
- Taking a long view, should you wait for prices to reach a commodity level, or should you work with the earlier more expensive generations?

Software freedom

The crisis

- The arcadian state: SHARE (1950s), DECUS (1960s), it was obvious!
- The CMU printer driver, Symbolics, Lisp Machines Inc., and the raiding of the MIT AI lab.

GNU Manifesto, 1983

What's GNU? Gnu's Not UNIX!

GNU, which stands for Gnu's Not Unix, is the name for the complete Unix-compatible software system which I am writing so that I can give it away free to everyone who can use it. Several other volunteers are helping me. Contributions of time, money, programs and equipment are greatly needed.
So far we have an Emacs text editor with Lisp for writing editor commands, a source level debugger, a yacc-compatible parser generator, a linker, and around 35 utilities. A shell (command interpreter) is nearly completed. A new portable optimizing C compiler has compiled itself and may be released this year. An initial kernel exists but many more features are needed to emulate Unix. When the kernel and compiler are finished, it will be possible to distribute a GNU system suitable for program development. We will use TeX as our text formatter, ...

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.
- The four freedoms of free software.

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.
- The four freedoms of free software.
- Licensing of derivative products.

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.
- The four freedoms of free software.
- Licensing of derivative products.
- Copyleft juxtaposed to permissive licensing.

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.
- The four freedoms of free software.
- Licensing of derivative products.
- Copyleft juxtaposed to permissive licensing.
- Historical background for US copyright and patent laws.

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.
- The four freedoms of free software.
- Licensing of derivative products.
- Copyleft juxtaposed to permissive licensing.
- Historical background for US copyright and patent laws.
- The correct juxtaposition: "free vs. proprietary", **not** "free vs. commercial": lots of people make very good money from free software.

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.
- The four freedoms of free software.
- Licensing of derivative products.
- Copyleft juxtaposed to permissive licensing.
- Historical background for US copyright and patent laws.
- The correct juxtaposition: "free vs. proprietary", **not** "free vs. commercial": lots of people make very good money from free software.

Classic licenses

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.
- The four freedoms of free software.
- Licensing of derivative products.
- Copyleft juxtaposed to permissive licensing.
- Historical background for US copyright and patent laws.
- The correct juxtaposition: "free vs. proprietary", **not** "free vs. commercial": lots of people make very good money from free software.

Classic licenses

- Public domain.

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.
- The four freedoms of free software.
- Licensing of derivative products.
- Copyleft juxtaposed to permissive licensing.
- Historical background for US copyright and patent laws.
- The correct juxtaposition: "free vs. proprietary", **not** "free vs. commercial": lots of people make very good money from free software.

Classic licenses

- Public domain.
- GNU General Public License (GPL).

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.
- The four freedoms of free software.
- Licensing of derivative products.
- Copyleft juxtaposed to permissive licensing.
- Historical background for US copyright and patent laws.
- The correct juxtaposition: "free vs. proprietary", **not** "free vs. commercial": lots of people make very good money from free software.

Classic licenses

- Public domain.
- GNU General Public License (GPL).
- GNU Lesser General Public License (LGPL).

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.
- The four freedoms of free software.
- Licensing of derivative products.
- Copyleft juxtaposed to permissive licensing.
- Historical background for US copyright and patent laws.
- The correct juxtaposition: "free vs. proprietary", **not** "free vs. commercial": lots of people make very good money from free software.

Classic licenses

- Public domain.
- GNU General Public License (GPL).
- GNU Lesser General Public License (LGPL).
- Berkeley Software Distribution (BSD) license.

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.
- The four freedoms of free software.
- Licensing of derivative products.
- Copyleft juxtaposed to permissive licensing.
- Historical background for US copyright and patent laws.
- The correct juxtaposition: "free vs. proprietary", **not** "free vs. commercial": lots of people make very good money from free software.

Classic licenses

- Public domain.
- GNU General Public License (GPL).
- GNU Lesser General Public License (LGPL).
- Berkeley Software Distribution (BSD) license.
- MIT X11 license.

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.
- The four freedoms of free software.
- Licensing of derivative products.
- Copyleft juxtaposed to permissive licensing.
- Historical background for US copyright and patent laws.
- The correct juxtaposition: "free vs. proprietary", **not** "free vs. commercial": lots of people make very good money from free software.

Classic licenses

- Public domain.
- GNU General Public License (GPL).
- GNU Lesser General Public License (LGPL).
- Berkeley Software Distribution (BSD) license.
- MIT X11 license.
- Pointless proliferation of licenses.

Licensing, copyright, copyleft, and MIT puns

Free as in freedom - cost is the minor issue here.

Licensing concepts

- Difference between copyright and license.
- What if you do not specify a license?
- Amazing the kind of people who get this wrong.
- The four freedoms of free software.
- Licensing of derivative products.
- Copyleft juxtaposed to permissive licensing.
- Historical background for US copyright and patent laws.
- The correct juxtaposition: "free vs. proprietary", **not** "free vs. commercial": lots of people make very good money from free software.

Classic licenses

- Public domain.
- GNU General Public License (GPL).
- GNU Lesser General Public License (LGPL).
- Berkeley Software Distribution (BSD) license.
- MIT X11 license.
- Pointless proliferation of licenses.
- The Creative Commons milieu.

Ethics, convenience, combativeness

The founding of the free software movement

The open source movement

Milestones

1987 Eric Raymond: "The Cathedral and the Bazaar" (TCatB).

Key phrase: "Given enough eyeballs, all bugs are shallow" (Linus's law).

1998-01 Frank Hecker internal Netscape whitepaper: make source code free. Cites TCatB.

1998-02-02 Christine Peterson coins term "open source". Goal: communicate advantages of free s/w to commercial s/w companies.

1998-02-05 Strategy group at Netscape adopts term open source.

1998-04-07 O'Reilly "Freeware Summit" becomes known as "Open Source Summit".

The movement

- Free Software and Open Source Software: same referent (body of software).
- Focus on usefulness rather than the ethical underpinnings.
- Gets around English language ambiguity of "free" (speech and beer).
- Ends up causing its own ambiguity due to conflation with various other uses of the word open.
- Composite terms: FOSS, FLOSS.
- Used almost universally by companies that release free software (i.e. all companies).

The GNU/Linux operating system

- Linus Torvalds announces a new kernel, 1991-09-17.
- Torvalds Torvalds places Linux under the GNU General Public License.
- Torvalds states “ Making Linux GPL’d was definitely the best thing I ever did.” 1997-09-30
- The Linux kernel brings the last key component to the GNU operating system. Terminology wars.

1898 – FUD around GNU/Linux

https://archive.nytimes.com/www.nytimes.com/library/tech/98/11/biztech/articles/03memo.html

The "Halloween Documents"

Technology

The New York Times on the Web

Home | Site Index | Site Search | Forums | Archives | Marketplace

November 3, 1998

Internal Memo Shows Microsoft Executives' Concern Over Free Software

By AMY HARMON and JOHN MARKOFF

An internal memorandum reflecting the views of some of Microsoft Corp.'s top executives and software development managers reveals deep concern about the threat of free software and proposes a number of strategies for competing against free programs that have recently been gaining in popularity.

The memo warns that the quality of free software can meet or exceed that of commercial programs and describes it as a potentially serious threat to Microsoft.

New York Times, 1998-11-03.

...

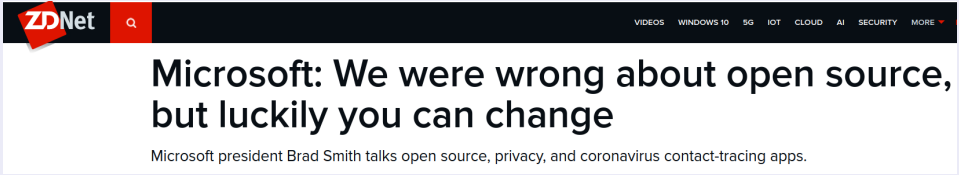
Consequently, OSS poses a direct, short-term revenue and platform threat to Microsoft – particularly in server space. Additionally, the intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mindshare threat.

...

[...] the memorandum calls the free software movement a "long-term credible" threat and warns that employing a traditional Microsoft marketing strategy known as "FUD," an acronym for "fear, uncertainty and doubt," will not succeed against the developers of free software.

The (software) pillars of the earth

Microsoft retreats from its position

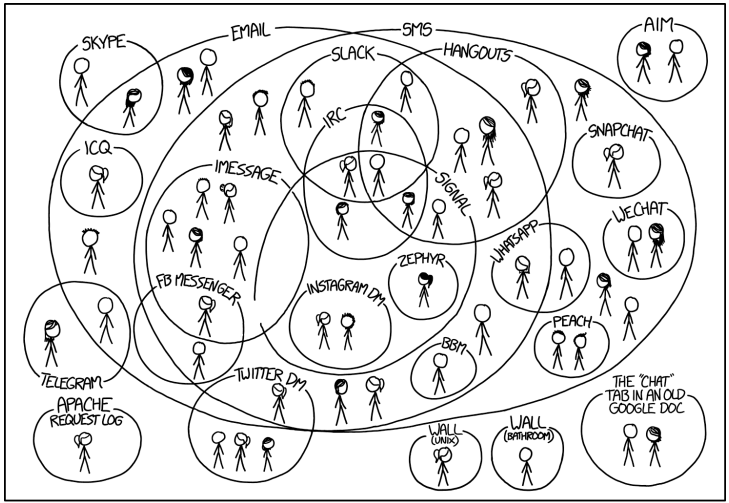


ZDNet, 2020-05-18, `https://www.theregister.co.uk/2020/05/15/microsoft_brad_smith_open_source/`

- Non-penetration: home computers (but soon...), some engineering CAD packages, some graphical front-ends for operating systems.
- Penetration: all web servers, all departmental serverse, all supercomputerse, all embedded systems, all phones.
- Two big pillars: gcc and linux kernel.
- Smaller pillars: apache, all version control systems, all programming languages, most web client-side frameworks.
- The GNU/Linux distributions: terminology.

Picking what will last – messaging systems

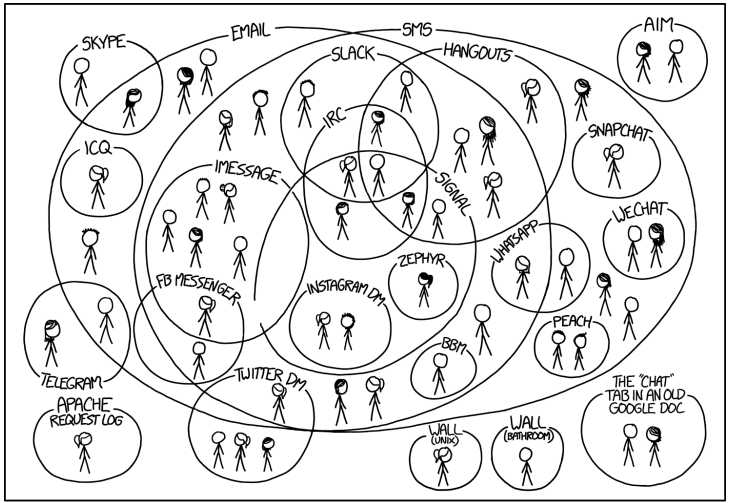
Naturalmente . . . xkcd: https://xkcd.com/1810/



I HAVE A HARD TIME KEEPING TRACK OF WHICH CONTACTS USE WHICH CHAT SYSTEMS.

Picking what will last – messaging systems

Naturalmente . . . xkcd: https://xkcd.com/1810/

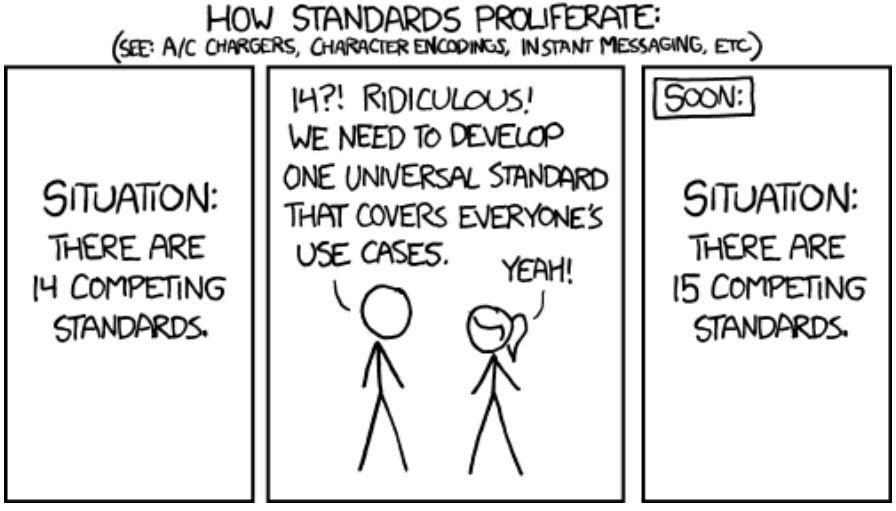


I HAVE A HARD TIME KEEPING TRACK OF WHICH CONTACTS USE WHICH CHAT SYSTEMS.

I'm one of the few Instagram users who connects solely through the Unix 'talk' gateway.

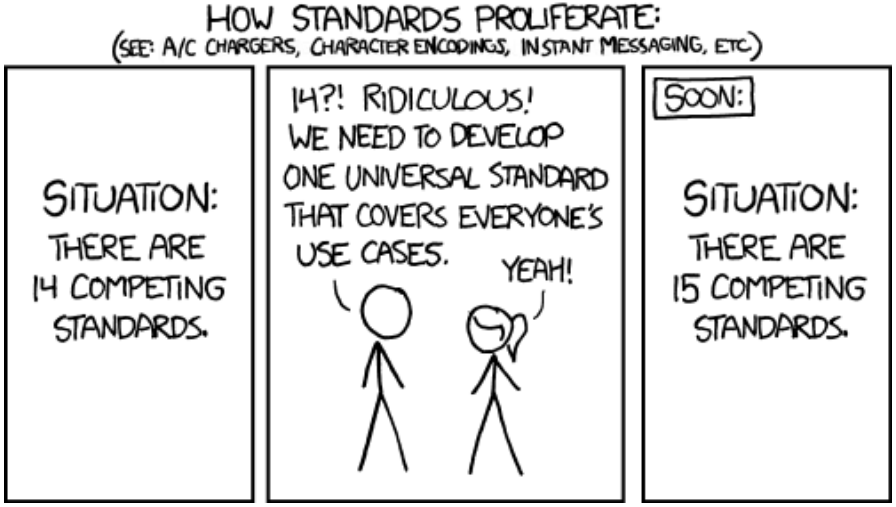
Example: messaging “standards”

Naturalmente . . . xkcd: https://xkcd.com/1810/



Example: messaging "standards"

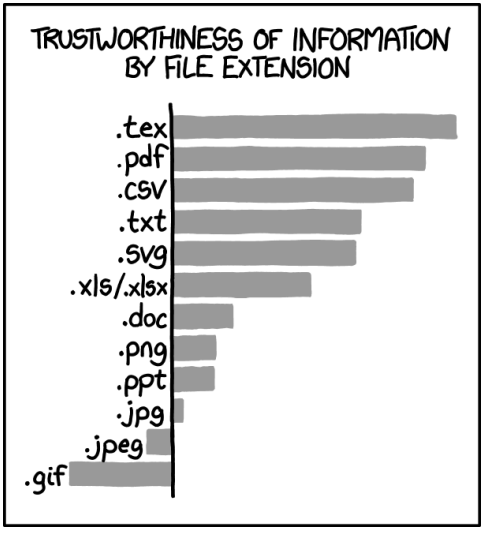
Naturalmente . . . xkcd: https://xkcd.com/1810/



Fortunately, the charging one has been solved now that we've all standardized on mini-USB. Or is it micro-USB? ****.

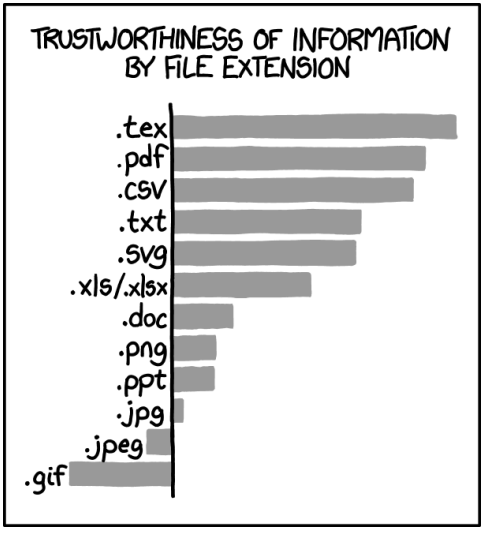
Documentation formats

Naturalmente . . . xkcd: https://xkcd.com/1301/



Documentation formats

Naturalmente . . . xkcd: https://xkcd.com/1301/



I have never been lied to by data in a .txt file which has been hand-aligned.

Part V – Case studies

Case studies - and what do we get out of them?

The awful

Not quite on the scale of Bhopal, but can't say much more than that.

The awful that got back on track

The example of IBM's OS/360.

Case studies - and what do we get out of them?

The awful

Not quite on the scale of Bhopal, but can't say much more than that.

The awful that got back on track

The example of IBM's OS/360.

Things that work

`ycombinator` discussion of sqlite:
https://news.ycombinator.com/item?id=34258858

An example that did not work

An example that did not work

Think of a project you are aware of that failed with a bang.

An example that did not work

Think of a project you are aware of that failed with a bang.
Think of it in a lovingly humorous manner, of course, since...

An example that did not work

Think of a project you are aware of that failed with a bang.
Think of it in a lovingly humorous manner, of course, since...
...we are here to get “lessons learned”,

An example that did not work

Think of a project you are aware of that failed with a bang.
Think of it in a lovingly humorous manner, of course, since...
...we are here to get “lessons learned”, not to be nasty. After all, it will never be as bad as:

An example that did not work

Think of a project you are aware of that failed with a bang.
Think of it in a lovingly humorous manner, of course, since...
...we are here to get “lessons learned”, not to be nasty. After all, it will never be as bad as:

An example that did not work

Think of a project you are aware of that failed with a bang.
Think of it in a lovingly humorous manner, of course, since...
...we are here to get "lessons learned", not to be nasty. After all, it will never be as bad as:

Deteriorating section of the Bhopal MIC (Methyl isocyanate) plant in 2008, decades after the gas leak. Initial death toll: 2259 people, total: 20000 people.

Case studies – failures – Apple Maps

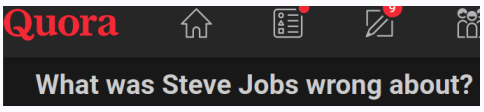
https://www.quora.com/What-was-Steve-Jobs-wrong-about

Steve Jobs's was celebrated for his business and product development skills after turning Apple in 1990s from "arguably one of the worst-managed companies in the industry" to a dominant tech company.

Case studies – failures – Apple Maps

https://www.quora.com/What-was-Steve-Jobs-wrong-about

Steve Jobs's was celebrated for his business and product development skills after turning Apple in 1990s from "arguably one of the worst-managed companies in the industry" to a dominant tech company.



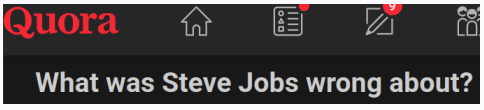
Mis-sizing Apple Maps

Maps is a difficult problem.

Case studies – failures – Apple Maps

https://www.quora.com/What-was-Steve-Jobs-wrong-about

Steve Jobs's was celebrated for his business and product development skills after turning Apple in 1990s from "arguably one of the worst-managed companies in the industry" to a dominant tech company.



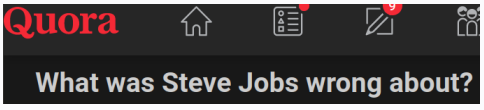
Mis-sizing Apple Maps

Maps is a difficult problem.At Google, at around the time Apple started working on Maps, I'd estimate Google had acquired several Maps-related companies at a cost of several hundred million dollars.

Case studies – failures – Apple Maps

https://www.quora.com/What-was-Steve-Jobs-wrong-about

Steve Jobs's was celebrated for his business and product development skills after turning Apple in 1990s from "arguably one of the worst-managed companies in the industry" to a dominant tech company.



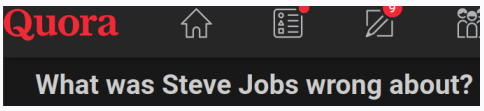
Mis-sizing Apple Maps

Maps is a difficult problem.At Google, at around the time Apple started working on Maps, I'd estimate Google had acquired several Maps-related companies at a cost of several hundred million dollars. Google had several hundreds of engineers working on Maps and Geography-related projects on a full-time basis.

Case studies – failures – Apple Maps

https://www.quora.com/What-was-Steve-Jobs-wrong-about

Steve Jobs's was celebrated for his business and product development skills after turning Apple in 1990s from "arguably one of the worst-managed companies in the industry" to a dominant tech company.



Mis-sizing Apple Maps

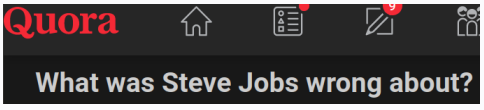
Maps is a difficult problem.At Google, at around the time Apple started working on Maps, I'd estimate Google had acquired several Maps-related companies at a cost of several hundred million dollars. Google had several hundreds of engineers working on Maps and Geography-related projects on a full-time basis. You could compare the level of effort in a project like this, as executed by Google, to a duck. A duck always appears calm on the surface of the water, but, under the water, the duck's feet are always paddling like hell.

Take-home

Case studies – failures – Apple Maps

https://www.quora.com/What-was-Steve-Jobs-wrong-about

Steve Jobs's was celebrated for his business and product development skills after turning Apple in 1990s from "arguably one of the worst-managed companies in the industry" to a dominant tech company.



Mis-sizing Apple Maps

Maps is a difficult problem.At Google, at around the time Apple started working on Maps, I'd estimate Google had acquired several Maps-related companies at a cost of several hundred million dollars. Google had several hundreds of engineers working on Maps and Geography-related projects on a full-time basis. You could compare the level of effort in a project like this, as executed by Google, to a duck. A duck always appears calm on the surface of the water, but, under the water, the duck's feet are always paddling like hell.

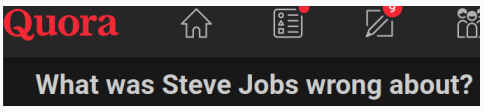
Take-home

Software as the main driver of complexity.

Case studies – failures – Apple Maps

https://www.quora.com/What-was-Steve-Jobs-wrong-about

Steve Jobs's was celebrated for his business and product development skills after turning Apple in 1990s from "arguably one of the worst-managed companies in the industry" to a dominant tech company.



Mis-sizing Apple Maps

Maps is a difficult problem.At Google, at around the time Apple started working on Maps, I'd estimate Google had acquired several Maps-related companies at a cost of several hundred million dollars. Google had several hundreds of engineers working on Maps and Geography-related projects on a full-time basis. You could compare the level of effort in a project like this, as executed by Google, to a duck. A duck always appears calm on the surface of the water, but, under the water, the duck's feet are always paddling like hell.

Take-home

Software as the main driver of complexity.
Ducks!

Case studies – failures – Virtual Case File

Report: FBI wasted millions on 'Virtual Case File'

Case studies – failures – Virtual Case File

Report: FBI wasted millions on 'Virtual Case File'

whether any portion of the $170 million Virtual Case File (VCF) software program can be salvaged.

He also promised to tell them at that time how much additional money would be needed to complete the project.

Mueller testified that if a current test shows the project has to be scrapped, he estimates the loss to taxpayers at $104 million.

Case studies – failures – Virtual Case File

Report: FBI wasted millions on 'Virtual Case File'

whether any portion of the $170 million Virtual Case File (VCF) software program can be salvaged.

Manage alerts | What is this?

He also promised to tell them at that time how much additional money would be needed to complete the project.

Mueller testified that if a current test shows the project has to be scrapped, he estimates the loss to taxpayers at $104 million.

Software project by the FBI to replace old "Automated Case Support" software system.

Case studies – failures – Virtual Case File

Report: FBI wasted millions on 'Virtual Case File'

whether any portion of the $170 million Virtual Case File (VCF) software program can be salvaged.

Manage alerts | What is this?

He also promised to tell them at that time how much additional money would be needed to complete the project.

Mueller testified that if a current test shows the project has to be scrapped, he estimates the loss to taxpayers at $104 million.

Software project by the FBI to replace old "Automated Case Support" software system.

[the Aerospace Corporation] said the SAIC software was incomplete, inadequate and so poorly designed that it would be essentially unusable under real-world conditions.

Case studies – failures – Virtual Case File

Report: FBI wasted millions on 'Virtual Case File'

whether any portion of the $170 million Virtual Case File (VCF) software program can be salvaged.

Manage alerts | What is this?

He also promised to tell them at that time how much additional money would be needed to complete the project.

Mueller testified that if a current test shows the project has to be scrapped, he estimates the loss to taxpayers at $104 million.

Software project by the FBI to replace old "Automated Case Support" software system.

[the Aerospace Corporation] said the SAIC software was incomplete, inadequate and so poorly designed that it would be essentially unusable under real-world conditions.

Even in rudimentary tests, the system did not comply with basic requirements, the report said.

`https://spectrum.ieee.org/computing/software/who-killed-the-virtual-case-file`

Virtual Case File failure – lessons learned

Virtual Case File failure – lessons learned

- Lack of a strong technical architecture (“blueprint”) from the outset led to poor architectural decisions.

Virtual Case File failure – lessons learned

- Lack of a strong technical architecture (“blueprint”) from the outset led to poor architectural decisions.
- Repeated changes in specification.

Virtual Case File failure – lessons learned

- Lack of a strong technical architecture ("blueprint") from the outset led to poor architectural decisions.
- Repeated changes in specification.
- Repeated turnover of management, which contributed to the specification problem.

Virtual Case File failure – lessons learned

- Lack of a strong technical architecture ("blueprint") from the outset led to poor architectural decisions.
- Repeated changes in specification.
- Repeated turnover of management, which contributed to the specification problem.
- Micromanagement of software developers.

Virtual Case File failure – lessons learned

- Lack of a strong technical architecture ("blueprint") from the outset led to poor architectural decisions.
- Repeated changes in specification.
- Repeated turnover of management, which contributed to the specification problem.
- Micromanagement of software developers.
- The inclusion of many FBI personnel who had little or no formal training in computer science as managers and even engineers on the project.

Virtual Case File failure – lessons learned

- Lack of a strong technical architecture (“blueprint”) from the outset led to poor architectural decisions.
- Repeated changes in specification.
- Repeated turnover of management, which contributed to the specification problem.
- Micromanagement of software developers.
- The inclusion of many FBI personnel who had little or no formal training in computer science as managers and even engineers on the project.
- Scope creep as requirements were continually added to the system even as it was falling behind schedule.

Virtual Case File failure – lessons learned

- Lack of a strong technical architecture ("blueprint") from the outset led to poor architectural decisions.
- Repeated changes in specification.
- Repeated turnover of management, which contributed to the specification problem.
- Micromanagement of software developers.
- The inclusion of many FBI personnel who had little or no formal training in computer science as managers and even engineers on the project.
- Scope creep as requirements were continually added to the system even as it was falling behind schedule.
- Code bloat due to changing specifications and scope creep—at one point it was estimated the software had over 700,000 lines of code.

Virtual Case File failure – lessons learned

- Lack of a strong technical architecture ("blueprint") from the outset led to poor architectural decisions.
- Repeated changes in specification.
- Repeated turnover of management, which contributed to the specification problem.
- Micromanagement of software developers.
- The inclusion of many FBI personnel who had little or no formal training in computer science as managers and even engineers on the project.
- Scope creep as requirements were continually added to the system even as it was falling behind schedule.
- Code bloat due to changing specifications and scope creep—at one point it was estimated the software had over 700,000 lines of code.
- Planned use of a flash cutover deployment made it difficult to adopt the system until it was perfected.

Virtual Case File failure – lessons learned

- Lack of a strong technical architecture ("blueprint") from the outset led to poor architectural decisions.
- Repeated changes in specification.
- Repeated turnover of management, which contributed to the specification problem.
- Micromanagement of software developers.
- The inclusion of many FBI personnel who had little or no formal training in computer science as managers and even engineers on the project.
- Scope creep as requirements were continually added to the system even as it was falling behind schedule.
- Code bloat due to changing specifications and scope creep—at one point it was estimated the software had over 700,000 lines of code.
- Planned use of a flash cutover deployment made it difficult to adopt the system until it was perfected.

Plenty information technology disasters


CLOUD & COMPUTING

Michigan Sues HP Over $49 Million IT Modernization Project

Several deadline extensions beyond 2010, Michigan's Business Application Modernization project remains unfinished.

Virtual Case File failure – lessons learned

- Lack of a strong technical architecture ("blueprint") from the outset led to poor architectural decisions.
- Repeated changes in specification.
- Repeated turnover of management, which contributed to the specification problem.
- Micromanagement of software developers.
- The inclusion of many FBI personnel who had little or no formal training in computer science as managers and even engineers on the project.
- Scope creep as requirements were continually added to the system even as it was falling behind schedule.
- Code bloat due to changing specifications and scope creep—at one point it was estimated the software had over 700,000 lines of code.
- Planned use of a flash cutover deployment made it difficult to adopt the system until it was perfected.

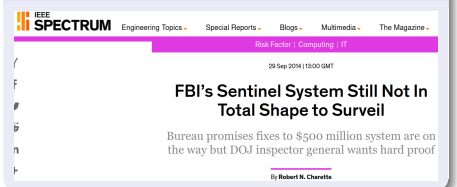
Plenty information technology disasters


CLOUD & COMPUTING

Michigan Sues HP Over $49 Million IT Modernization Project

Several deadline extensions beyond 2010, Michigan's Business Application Modernization project remains unfinished.

The FBI tries again


IEEE SPECTRUM Engineering Topics Special Reports Blogs Multimedia The Magazine

Risk Factor | Computing | IT

29 Sep 2014 | 13:00 GMT

FBI's Sentinel System Still Not In Total Shape to Surveil

Bureau promises fixes to $500 million system are on the way but DOJ inspector general wants hard proof

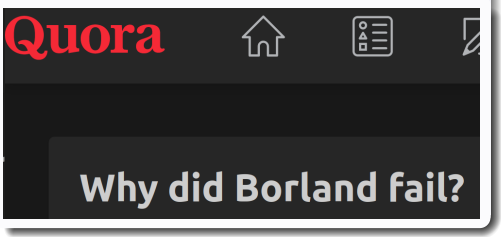
By Robert N. Charette

Case studies – Boreland compilers

Boreland had it all: Pascal and then C compilers were widely used and loved.

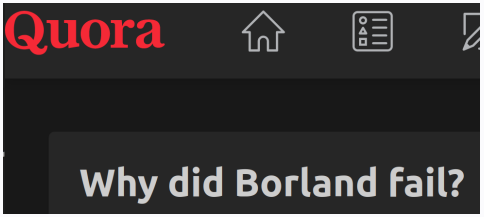
Case studies – Boreland compilers

Boreland had it all: Pascal and then C compilers were widely used and loved.



Case studies – Boreland compilers

Boreland had it all: Pascal and then C compilers were widely used and loved.

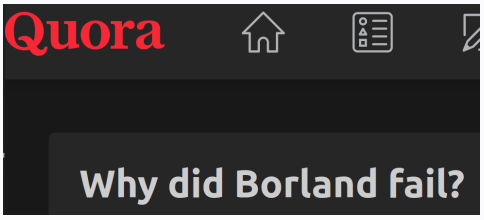


tech

After realising that the compiler didn't produce the same results compiling the same code twice and finding that the debugger was buggy I had to just drop Borland.

Case studies – Boreland compilers

Boreland had it all: Pascal and then C compilers were widely used and loved.



tech

After realising that the compiler didn't produce the same results compiling the same code twice and finding that the debugger was buggy I had to just drop Borland.

management

Borland lost its way when executive management decided to change the company to pursue a different market.
A few years after Borland went public, founder and CEO Philippe Kahn began to have increasing disagreements with the Borland board of directors. [...] board of directors wanted to shift gears and pursue the "enterprise" software market. I get the impression that this difference of opinion simmered for years. Ultimately the board fired Kahn and threw the company headlong into the pursuit of the enterprise market.
Borland's long slow death spiral began when it turned away from what it knew best to chase a unicorn it knew nothing about.

market direction

Overall, Borland competed too much with Microsoft. It became a war of "who wants to bear the mantle of responsiblity the most?" and Microsoft just kept pulling tricks and pouring resources into their developer tools, office apps, etc. Without legislation preventing an OS vendor from also supplying applications, the fight was futile as long as MS had the passion for the space.

Case studies – Python packaging

Case studies – Python packaging

[...] area of Python that many developers have problems with [...] many different solutions pop up over the years [...] wars, and attempts to solve it [...] packaging ecosystem and tools making their lives harder [...] confused about virtual environments [...] is the organization behind most of the packaging tools and standards part of the problem itself?

Join me on a journey through packaging in Python and elsewhere [...] classic packaging stack (involving setuptools and friends), the scientific stack (with conda), [...] modern/alternate [...] Pipenv, Poetry, Hatch, or PDM.

Case studies – Python packaging

[...] area of Python that many developers have problems with [...] many different solutions pop up over the years [...] wars, and attempts to solve it [...] packaging ecosystem and tools making their lives harder [...] confused about virtual environments [...] is the organization behind most of the packaging tools and standards part of the problem itself?

Join me on a journey through packaging in Python and elsewhere [...] classic packaging stack (involving setuptools and friends), the scientific stack (with conda), [...] modern/alternate [...] Pipenv, Poetry, Hatch, or PDM.

Contents

SQLite - it just works

SQLite - it just works

SQLite - it just works

▲ Why SQLite succeeded as a database (2016) (changelog.com)

128 points by Tomte 17 days ago | hide | past | favorite | 85 comments

SQLite - it just works

▲ Why SQLite succeeded as a database (2016) (changelog.com)

128 points by Tomte 17 days ago | hide | past | favorite | 85 comments

Richard Hipp

SQLite came from a Hipp's experience maintaining an INFORMIX database on a navy *battleship*, and the server kept crashing.

SQLite - it just works

▲ Why SQLite succeeded as a database (2016) (changelog.com)
128 points by Tomte 17 days ago | hide | past | favorite | 85 comments

Richard Hipp

SQLite came from a Hipp's experience maintaining an INFORMIX database on a navy *battleship*, and the server kept crashing.
"Richard, why don't you just write one?" "Okay, I'll give it a try."

SQLite - it just works

▲ Why SQLite succeeded as a database (2016) (changelog.com)
128 points by Tomte 17 days ago | hide | past | favorite | 85 comments

Richard Hipp

SQLite came from a Hipp's experience maintaining an INFORMIX database on a navy *battleship*, and the server kept crashing.
"Richard, why don't you just write one?" "Okay, I'll give it a try."

Newt Gingrich and Bill Clinton were having a fight of some sort, so all government contracts got shut down, so I was out of work for a few months, and I thought, "Well, I'll just write that database engine now."

SQLite - it just works

▲ Why SQLite succeeded as a database (2016) (changelog.com)
128 points by Tomte 17 days ago | hide | past | favorite | 85 comments

Richard Hipp

SQLite came from a Hipp's experience maintaining an INFORMIX database on a navy *battleship*, and the server kept crashing.
"Richard, why don't you just write one?" "Okay, I'll give it a try."

Newt Gingrich and Bill Clinton were having a fight of some sort, so all government contracts got shut down, so I was out of work for a few months, and I thought, "Well, I'll just write that database engine now."

We were going around boasting to everybody naively that SQLite didn't have any bugs in it, or no serious bugs, but Android definitely proved us wrong. Look, I used to think that I could write software with no bugs in it. It's amazing how many bugs will crop up when your software suddenly gets shipped on millions of devices.

SQLite - it just works

SQLite - it just works

. . . but I'd be in more trouble if SQLite disappeared. Richard's database is the most used database in the world, and by some counts, it's the most widely deployed software module of any type. If it disappeared, your web browser wouldn't work, your smartphone probably wouldn't start up, and probably your car wouldn't start up, as well.

Its impact on the world is massive, and there's plenty of places where it could have lost its way. The Consortium could have stifled progress or run out of money, or the full year it took to address all the android bugs could have easily burnt Richard out, but he prevailed and now he's in a great position to offer advice to others who want to create impactful open source software.

SQLite - it just works

. . . but I'd be in more trouble if SQLite disappeared. Richard's database is the most used database in the world, and by some counts, it's the most widely deployed software module of any type. If it disappeared, your web browser wouldn't work, your smartphone probably wouldn't start up, and probably your car wouldn't start up, as well.

Its impact on the world is massive, and there's plenty of places where it could have lost its way. The Consortium could have stifled progress or run out of money, or the full year it took to address all the android bugs could have easily burnt Richard out, but he prevailed and now he's in a great position to offer advice to others who want to create impactful open source software.

"probably one of the dark horse reasons for success is its "open source but not open contributions" model and strong personal leadership from Richard"

Case studies – GNU Scientific Library (GSL)

Case studies – GNU Scientific Library (GSL)

PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Opinion piece

Check for updates

Cite this article: Fortunato L, Galassi M. 2021 The case for free and open source software in research and scholarship. *Phil. Trans. R. Soc. A* **379**: 20200079.

The case for free and open source software in research and scholarship

Laura Fortunato[1,2] and Mark Galassi[3]

[1]Institute of Cognitive and Evolutionary Anthropology, University of Oxford, 64 Banbury Road, Oxford OX2 6PN, UK
[2]Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA
[3]Space Science and Applications Group, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

Article gives a general introduction to free/open-source software aimed at quantitative social scientists.

Uses the vagaries that brought to the GNU Scientific Library in the 1990s.

Case studies – GNU Scientific Library (GSL)

PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Opinion piece

Check for updates

Cite this article: Fortunato L, Galassi M. 2021 The case for free and open source software in research and scholarship. *Phil. Trans. R. Soc. A* **379**: 20200079.

The case for free and open source software in research and scholarship

Laura Fortunato[1,2] and Mark Galassi[3]

[1]Institute of Cognitive and Evolutionary Anthropology, University of Oxford, 64 Banbury Road, Oxford OX2 6PN, UK
[2]Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA
[3]Space Science and Applications Group, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

Article gives a general introduction to free/open-source software aimed at quantitative social scientists.

Uses the vagaries that brought to the GNU Scientific Library in the 1990s.

Numerical analysis landscape: 1980s and 1990s

Case studies – GNU Scientific Library (GSL)

PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Opinion piece

Check for updates

Cite this article: Fortunato L, Galassi M. 2021 The case for free and open source software in research and scholarship. *Phil. Trans. R. Soc. A* **379**: 20200079.

The case for free and open source software in research and scholarship

Laura Fortunato[1,2] and Mark Galassi[3]

[1]Institute of Cognitive and Evolutionary Anthropology, University of Oxford, 64 Banbury Road, Oxford OX2 6PN, UK
[2]Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA
[3]Space Science and Applications Group, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

Article gives a general introduction to free/open-source software aimed at quantitative social scientists.

Uses the vagaries that brought to the GNU Scientific Library in the 1990s.

Numerical analysis landscape: 1980s and 1990s

Grand old packages: SLATEC, netlib, CLAMS

Sandia, Los Alamos, Air Force Research Lab Technical Exchange Committee (SLATEC): high quality public domain math library - started in 1974, still shipping now.
netlib: set up in 1985 for network access to high quality libraries.
CLAMS: Common Los Alamos Math System - proprietary and now defunct.

Case studies – GNU Scientific Library (GSL)

PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Opinion piece

Check for updates

Cite this article: Fortunato L, Galassi M. 2021 The case for free and open source software in research and scholarship. *Phil. Trans. R. Soc. A* **379**: 20200079.

The case for free and open source software in research and scholarship

Laura Fortunato[1,2] and Mark Galassi[3]

[1]Institute of Cognitive and Evolutionary Anthropology, University of Oxford, 64 Banbury Road, Oxford OX2 6PN, UK
[2]Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA
[3]Space Science and Applications Group, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

Article gives a general introduction to free/open-source software aimed at quantitative social scientists.

Uses the vagaries that brought to the GNU Scientific Library in the 1990s.

Numerical analysis landscape: 1980s and 1990s

Grand old packages: SLATEC, netlib, CLAMS

Sandia, Los Alamos, Air Force Research Lab Technical Exchange Committee (SLATEC): high quality public domain math library - started in 1974, still shipping now.
netlib: set up in 1985 for network access to high quality libraries.
CLAMS: Common Los Alamos Math System - proprietary and now defunct.

But...

It's all in FORTRAN. 1990s are coming, physicists are programming in C.
Many relied on Numerical Recipes, nice pedagogical explainations but rubbish code.