Modern Software Engineering and Research

A pandemic-adapted professional development workshop

Mark Galassi

Space Science and Applications group
Los Alamos National Laboratory

2020-05-16, 2021-01-20

Last built 2021-01-27T13:19:44

Outline

Goals

Outline

Goals

Goals and path

In the educational industrial complex we are required to state our goals before we start.
It might even be a good idea.

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.

The meandering path

- My path is largely historical because of my personal inclination to use history to give perspective.
- We need perspective so we are not tossed about by the short-term interest of industry.

Style

- Slides are placeholders for me to then tell stories.
  I hope you will talk and tell stories too.
- But also: I join the modern quest to give a seminar made entirely of xkcd slides.

Outline

Curriculum

Programming Languages

- Stories of programming languages
- Tour of languages
- Insights on languages

The Computer Science Curriculum

From https://teachyourselfcs.com/

Computer programming

Computer Architecture

Algorithms and Data Structures

Discrete Math

Operating Systems

Computer security

Networking

Databases

Compilers and Languages

Artificial Intelligence

Graphics

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".

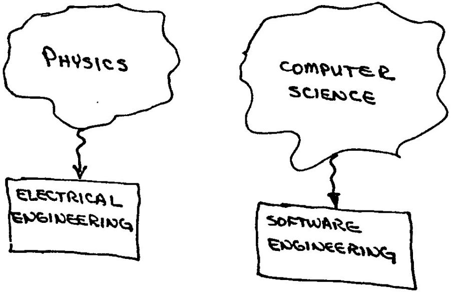


Figure 3.
Software-Electrical Engineering Analog

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

Outline

Curriculum

Programming Languages

- Stories of programming languages
- Tour of languages
- Insights on 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 tounsand lines of code.

Performance

Language features are related to how well you can optimize.

Memory safety

Avoiding memory corruption while keeping high performance.

Outline

Curriculum

Programming Languages

- Stories of programming languages
- Tour of languages
- Insights on languages

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

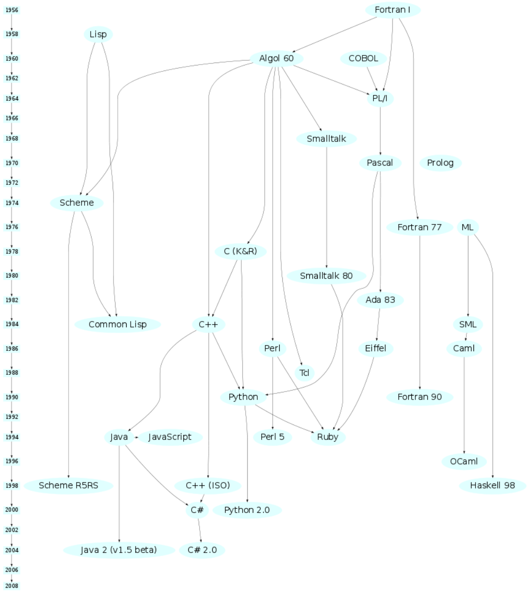
The 2010s

2010	Julia
2012	Kotlin
2017	WebAssembly

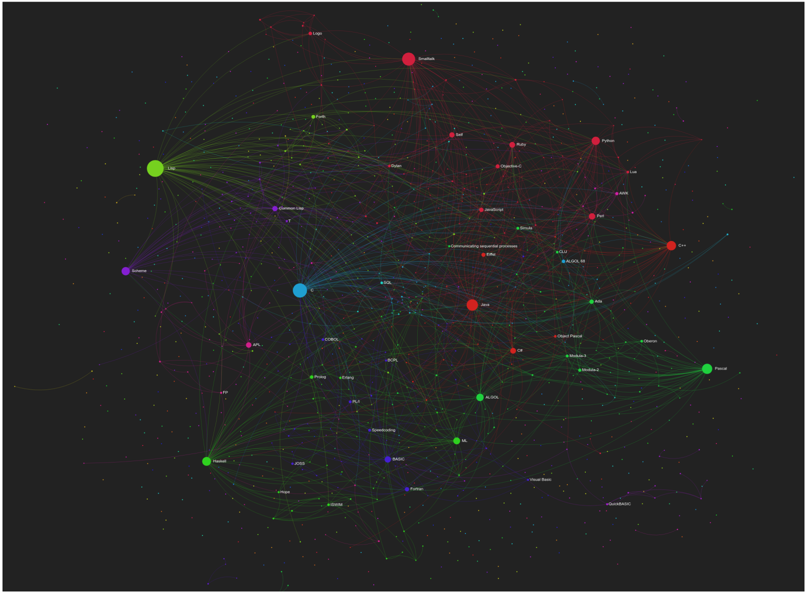
The future (created by Santa Fe youngsters)

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

Outline

Curriculum

Programming Languages

- Stories of programming languages
- Tour of languages
- Insights on languages

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`

```c
#define iv 4
#define v ;(void
#define XI(xi)int xi[iv*'V'];
#define L(c,l,i)c(){d(l);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&&l(-c-c,
"%d"),l(i,"+%d")):l(i,"(%d")):(c&&l(M,")"),l(*D,"%c")),c=i;c&&l(X,")"),l
(-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));}l(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,l,
d,l,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

```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

```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

```
// 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

```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
```

Outline

Curriculum

Programming Languages

- Stories of programming languages
- Tour of languages
- Insights on languages

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 13)

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

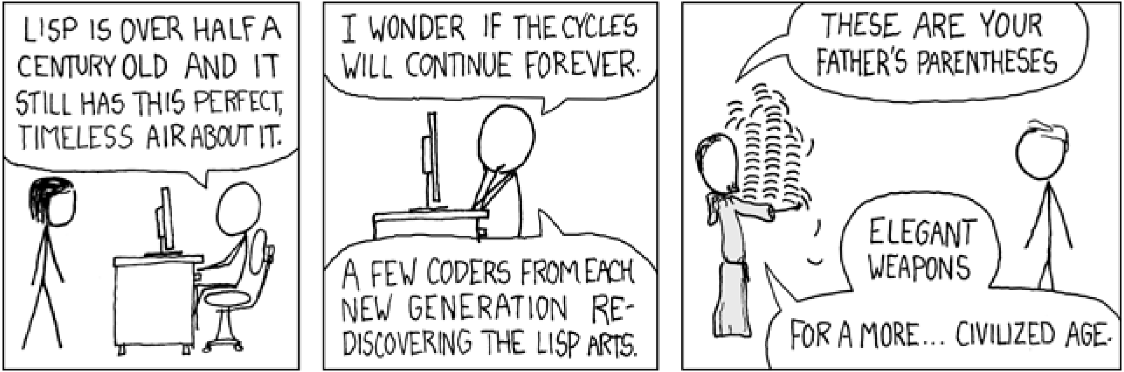
Comparing C and Pascal is rather like comparing a Learjet to a Piper Cub - one is meant for getting something done while the other is meant for learning - so such comparisons tend to be somewhat farfetched. . . .

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

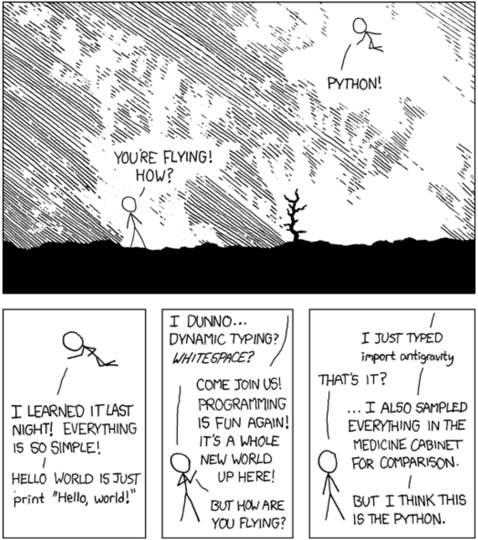
Naturalmente . . . xkcd: https://xkcd.com/297/


LISP IS OVER HALF A CENTURY OLD AND IT STILL HAS THIS PERFECT, TIMELESS AIR ABOUT IT.
I WONDER IF THE CYCLES WILL CONTINUE FOREVER.
A FEW CODERS FROM EACH NEW GENERATION RE-DISCOVERING THE LISP ARTS.
THESE ARE YOUR FATHER'S PARENTHESES
ELEGANT WEAPONS
FOR A MORE... CIVILIZED AGE.

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

Fear and loathing in programming languages – love

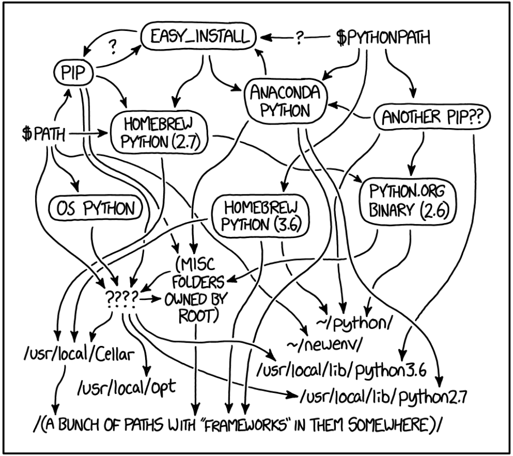
Naturalmente . . . xkcd: `https://xkcd.com/353/`


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

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.

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=0g847HVwRSI
https://vole.wtf/coder-serial-killer-quiz/
The “extra slides” area has two of those videos embedded.

Outline

Pedagogically speaking

Workflow and tools
- Automation and efficiency
- Editors

Version control

Debugging

The bones of the world

Case study: Beowulf clusters

Licensing, software freedom, open source

My view of what a student should know – IT

Just my view – colored by hiring students in Los Alamos

A cool IT job

- Immediate employment.
- Campus jobs!!
- Sysadmin vs. click jockey.

Some required knowledge

- GNU/Linux hobbyist.
- Broad view of industry - thirst for knowledge.
- Networking with OpenWRT.

WE TOOK THE HOSTAGES, SECURED THE BUILDING, AND CUT THE COMMUNICATION LINES LIKE YOU SAID.
EXCELLENT.
BUT THEN THIS GUY CLIMBED UP THE VENTILATION DUCTS AND WALKED ACROSS BROKEN GLASS, KILLING ANYONE WE SENT TO STOP HIM.
AND HE RESCUED THE HOSTAGES?
NO, HE IGNORED THEM. HE JUST RECONNECTED THE CABLES WE CUT, MUTTERING SOMETHING ABOUT "UPTIME".
SHIT, WE'RE DEALING WITH A SYSADMIN.

The weird sense of duty really good sysadmins have can border on the sociopathic, but it's nice to know that it stands between the forces of darkness and your cat blog's servers.

My view of what a student should know – programming

Just my view – colored by hiring students in Los Alamos

A programmer who loves it

- Research-level knowledge of the industry.-
- Old-time hacker ethic.
- Spend time developing a good "carpenter's workshop".
- Don't be scared of theory, but be very practical.

Some required knowledge

- Python, then C, then the hypermoderns.
- Hardware and OS internals.
- Service to science and engineering.
- Breathes the gluing together of UNIX tools.

The poll of Oxford and Stanford professors

Informally: could you distill ways in which you want “me” to prepare your undergraduates?

Laura F, professor of evolutionary anthropology

- Master a text editor.
- Learn to touch type.

Jeremy G-F, professor of public health

- Obviously programming.
- But also intuiting how often to come to me: every day is too often, every two weeks is not enough.

David G-G, professor of physics

- Obviously programming and math.
- But I find it really makes a difference if they have some experience with hardware working on their car, or some other approach to hardware.

Internship opportunities

What’s there and how to prepare

Opportunities in nothern NM

- Always aim “one up”.
- Los Alamos (11th and up).
- Institute for Computing in Research (10th and up).
- Santa Fe Institute - largely pulled out of HS.
- Local industry: OpenEye s/w, Descartes labs.
- Google summer of code (undergrade and up).
- Outreachy (undergrad and up).

How to prepare

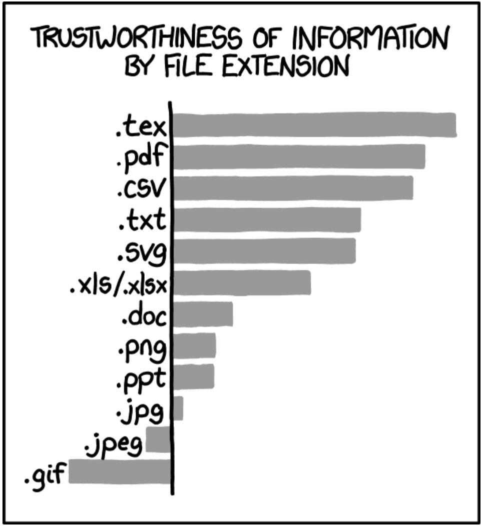
- Teachers should write them a good letter!
- Effort into cover and resume. Templates at `https://computinginresearch.org/research-internships/apply/`
- Strategize on the application, make cold calls, use soft skills.
- The importance of writing! “clear thinking <-> clear writing”
- Release source code, even if for small programs.

Examples of praise for resumes

- I immediately noticed the one with Computer Modern fonts (LaTeX).
- (Her) resume is adorable, especially the pie chart of "a day in the life." …right about their letters of recommendation; very poorly written. It isn’t even signed.

Documentation formats, again

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



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

Underrepresented groups

A deadly serious problem

Stats and causes

- I won't even bother with stats: you know them better than I do.
- The opportunity gap.
- The writing gap.
- Economic needs of the family.
- Stereotype threat: kills many approaches.
- Impostor syndrome.
- The ballad of Melanie Mitchell: electronics.

Approaches

- Geographical segregation of some minorities: miles on your tires. Excellent results from Monte del Sol and Capital.
- The "retail politics" of outreach to young women. Then make it safe!
- Do not lower the standard or stereotype threat will appear.
- Pay a stipend.

Underrepresented groups – results

In our pipeline:

Serious programming

A 10hr weekend workshop.

- Most workshops are now majority-minority.
- One has been majority female, most are close.
- Not making special individual effort => no young women.
- Organizing the “women in computer science” documentary and panel.

Programming mini-courses

Drop-in 1.5 hours every fortnight for advanced students.

- Most evenings are majority female.
- Minorities well represented.
- Conclusion: we at the advanced level we see no underrepresentation.

Underrepresented groups – results (continued)

In our pipeline:

Working groups

Math and physics working groups for students whose schools move too slowly.

- Intent: prepare students for internships where they need more advanced coursework. Offset Santa Fe difficulty in access to high level classes.
- Somewhat more participants in math working group are female.
- Most participants in physics working group are young men. Gender gap is present.

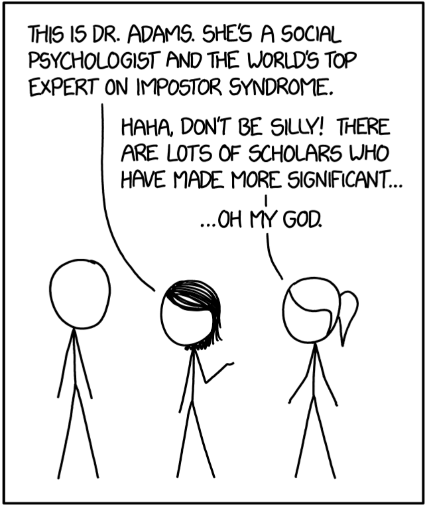
Institute for Computing in Research

Pays students to spend part of summer doing research.

- Mentors are trained to look at resumes and renormalize for different access to resume preparation help.
- Summer 2019 pilot program: 100% from underrepresented groups.
- Summer 2020: majority minority and 44% female.

Impostor Syndrome

Naturalmente . . . xkcd: https://xkcd.com/1954/


THIS IS DR. ADAMS. SHE'S A SOCIAL PSYCHOLOGIST AND THE WORLD'S TOP EXPERT ON IMPOSTOR SYNDROME.
HAHA, DON'T BE SILLY! THERE ARE LOTS OF SCHOLARS WHO HAVE MADE MORE SIGNIFICANT...
...OH MY GOD.

It's actually worst in people who study the Dunning-Kruger effect. We tried to organize a conference on it, but the only people who would agree to give the keynote were random undergrads.

Outline

Pedagogically speaking

Workflow and tools

- Automation and efficiency
- Editors

Version control

Debugging

The bones of the world

Case study: Beowulf clusters

Licensing, software freedom, open source

Outline

Pedagogically speaking

Workflow and tools

- Automation and efficiency
- Editors

Version control

Debugging

The bones of the world

Case study: Beowulf clusters

Licensing, software freedom, open source

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

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?

```
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

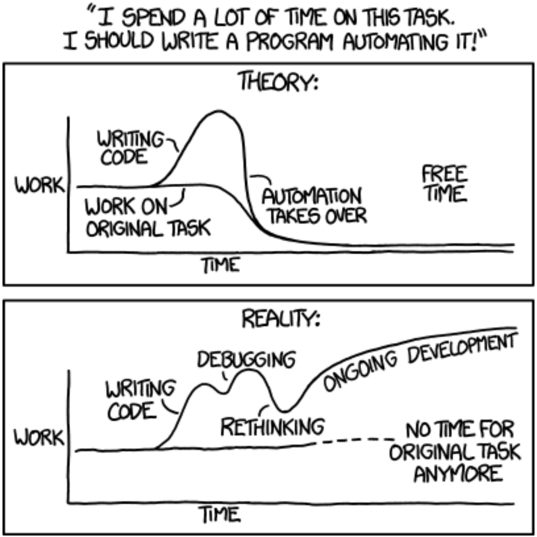
- Your rightful place is in the command line.
- Redirection (< and >) and pipes (|) are wonderful.
- Use many small programs which interact together to form **pipelines**.
- People breezily say “just use sed and awk” – thanks to the examples above we now know what they mean.
- Use graphical and integrated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

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/



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

Command line diversions

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

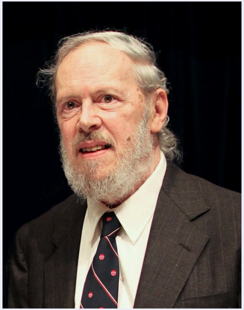
One-line ascii art

```
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



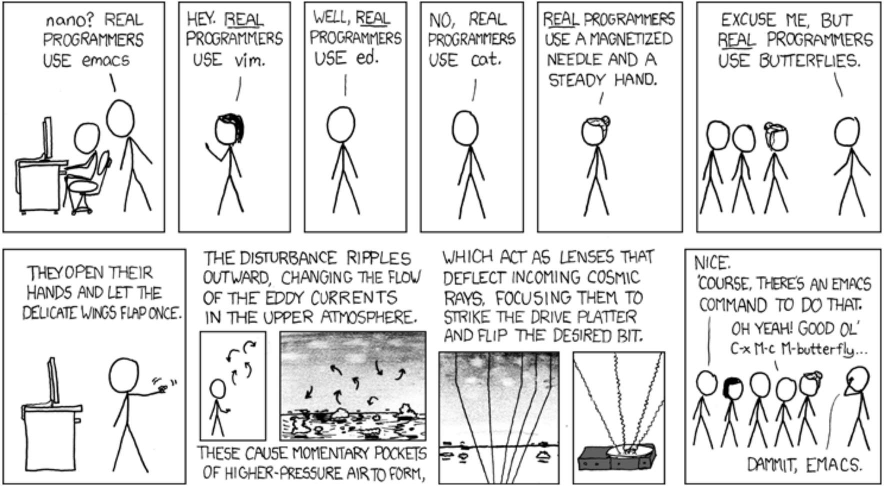
Dennis Ritchie who created the C programming language and co-created UNIX. Let's make ascii art of him.

Outline

- Pedagogically speaking
- Workflow and tools
  - Automation and efficiency
  - Editors
- Version control
- Debugging
- The bones of the world
- Case study: Beowulf clusters
- Licensing, software freedom, open source

Editor wars

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


nano? REAL PROGRAMMERS USE emacs
HEY. REAL PROGRAMMERS USE vim.
WELL, REAL PROGRAMMERS USE ed.
NO, REAL PROGRAMMERS USE cat.
REAL PROGRAMMERS USE A MAGNETIZED NEEDLE AND A STEADY HAND.
EXCUSE ME, BUT REAL PROGRAMMERS USE BUTTERFLIES.
THEY OPEN THEIR HANDS AND LET THE DELICATE WINGS FLAP ONCE.
THE DISTURBANCE RIPPLES OUTWARD, CHANGING THE FLOW OF THE EDDY CURRENTS IN THE UPPER ATMOSPHERE.
THESE CAUSE MOMENTARY POCKETS OF HIGHER-PRESSURE AIR TO FORM,
WHICH ACT AS LENSES THAT DEFLECT INCOMING COSMIC RAYS, FOCUSING THEM TO STRIKE THE DRIVE PLATTER AND FLIP THE DESIRED BIT.
NICE.
'COURSE, THERE'S AN EMACS COMMAND TO DO THAT.
OH YEAH! GOOD OL' C-x M-c M-butterfly...
DAMMIT, EMACS.

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.

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?”

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.

Outline

- Pedagogically speaking
- Workflow and tools
  - Automation and efficiency
  - Editors
- Version control
- Debugging
- The bones of the world
- Case study: Beowulf clusters
- Licensing, software freedom, open source

Version control (VC) – generalities

Motivation and history

Importance of tracking changes and reproducing old versions.

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

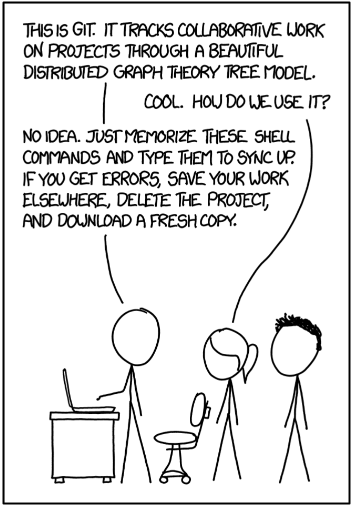
1. git log ## detailed information on what's been happening
2. git tag release-1.5 ## reproducibly define a release

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/


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

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.

Outline

Pedagogically speaking

Workflow and tools

- Automation and efficiency
- Editors

Version control

Debugging

The bones of the world

Case study: Beowulf clusters

Licensing, software freedom, open source

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
```

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

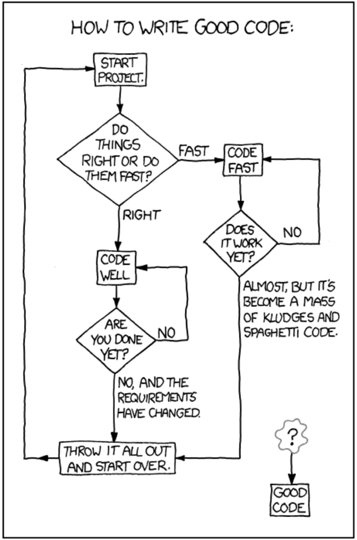
After all we have discussed:

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 like?
- Is it another software engineering fad?
- Is it all or most of these, plus some other undiscovered ones?

Is there a path to good code?

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


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
RIGHT
DOES IT WORK YET?
NO
CODE WELL
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
?
THROW IT ALL OUT AND START OVER.
GOOD CODE

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

Outline

- Pedagogically speaking
- Workflow and tools
  - Automation and efficiency
  - Editors
- Version control
- Debugging
- The bones of the world
- Case study: Beowulf clusters
- Licensing, software freedom, open source

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

Outline

Pedagogically speaking

Workflow and tools

- Automation and efficiency
- Editors

Version control

Debugging

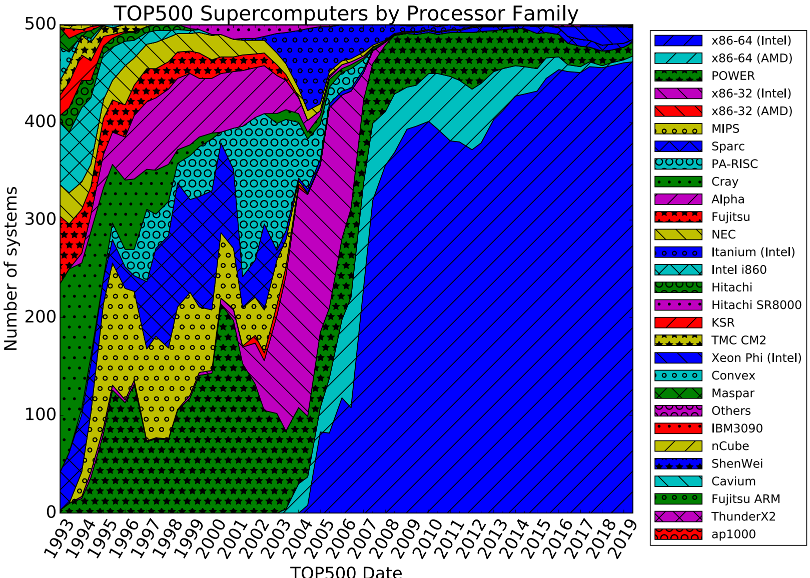
The bones of the world

Case study: Beowulf clusters

Licensing, software freedom, open source

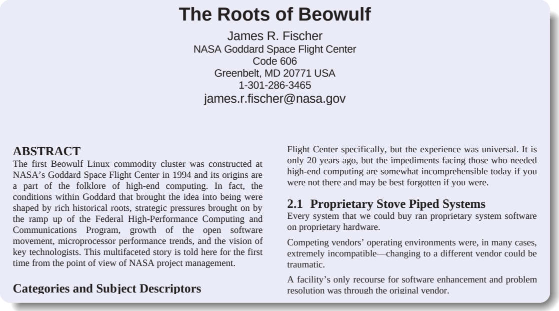
Supercomputer hardware type evolution

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



The Roots of Beowulf

James R. Fischer
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.

Categories and Subject Descriptors

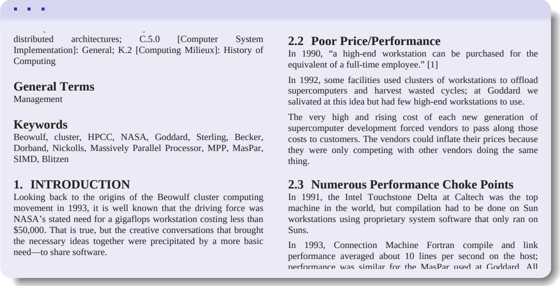
Flight Center specifically, but the experience was universal. It is only 20 years ago, but the impediments facing those who needed high-end computing are somewhat incomprehensible today if you were not there and may be best forgotten if you were.

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.

A facility's only recourse for software enhancement and problem resolution was through the original vendor.



. . .

distributed architectures; C.5.0 [Computer System Implementation]: General; K.2 [Computing Milieux]: History of 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 the necessary ideas together were precipitated by a more basic need—to share software.

2.2 Poor Price/Performance

In 1990, "a high-end workstation can be purchased for the 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.

In 1993, Connection Machine Fortran compile and link performance averaged about 10 lines per second on the host; performance was similar for the MasPar used at Goddard. All

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 85)
- Taking a long view, should you wait for prices to reach a commodity level, or should you work with the earlier more expensive generations?

Outline

Pedagogically speaking

Workflow and tools

- Automation and efficiency
- Editors

Version control

Debugging

The bones of the world

Case study: Beowulf clusters

Licensing, software freedom, open source

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

- 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).

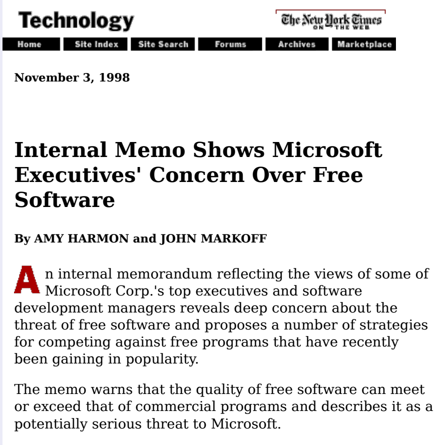
The GNU/Linux operating system

- 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”



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.

...

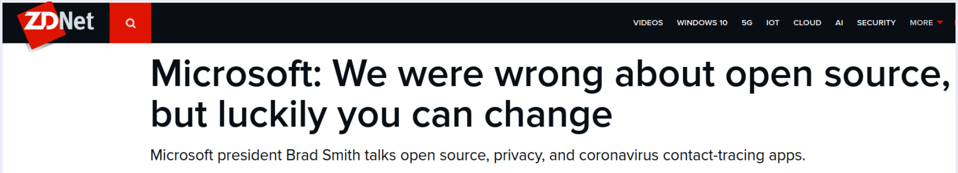
Consequently, OSS poses a direct, short-term revenue and platform threat to Microsoft – particularly in server space. Additionally, the intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer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



Microsoft: We were wrong about open source, but luckily you can change

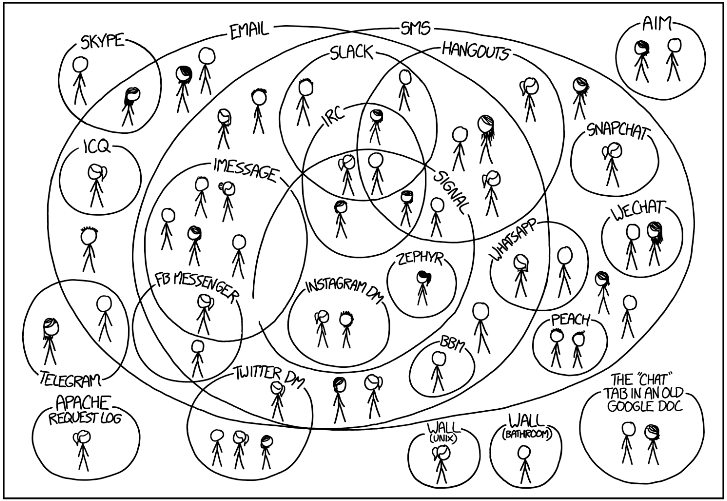
Microsoft president Brad Smith talks open source, privacy, and coronavirus contact-tracing apps.

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/

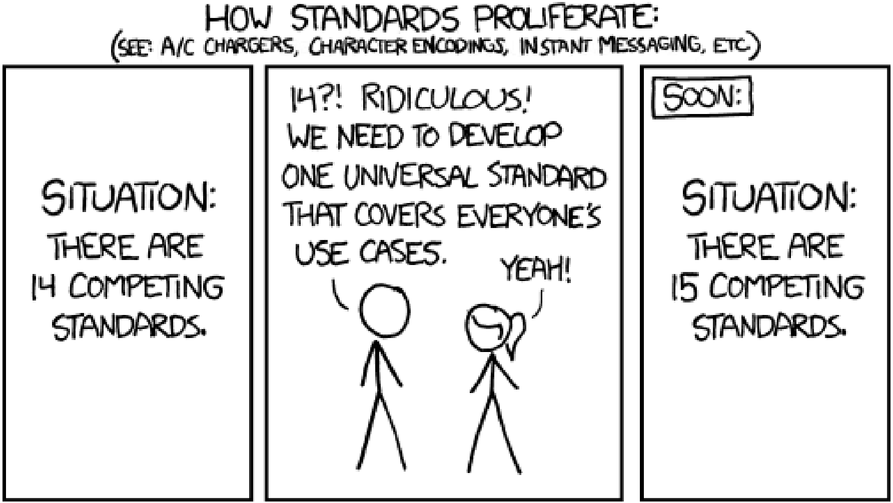


I HAVE A HARD TIME KEEPING TRACK OF WHICH CONTACTS USE WHICH CHAT SYSTEMS.

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

Example: messaging “standards”

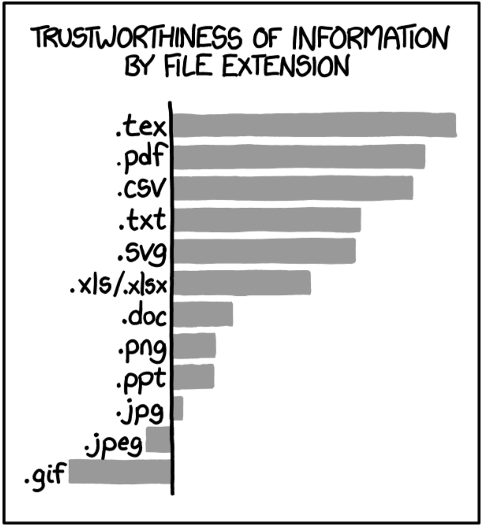
Naturalmente . . . xkcd: https://xkcd.com/1810/


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

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/



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