The Production of Software

- Technology
- Economics
- Sociology

Basics

- What is software?
- Descriptions of what a computer should do
- Plans for handling data and events
- Written down
- Converted into something the computer can use

How to write it down

- Flip switches
- Punch holes in paper
- Assembler
- C
- C++
- Python
- Visual programming

How much do I have to write?

- “lines of code”
- A simple filter algorithm: 1-10
- Small plugin: 100-10,000
- Big plugin (Halion,Kontakt): 10,000 – 50,000
- JACK (audio I/O, routing): 20,000
- Ardour: 90,000
- Gimp (photoshop): 650,000
- Blender (art,3d,animation): 1M
- Linux kernel: about 5-10M lines
- OpenOffice (word, etc): about 10M
- Windows NT (2003): 60-70M
- A GNU/Linux distribution: 200M

```c
#include <stdio.h>

int main(void) {

   printf("Hello World");
   return 0;
}
```

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. HELLOWORLD.
000300
000400*
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100100
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400     DISPLAY " " LINE 1 POSITION
1 ERASE EOS.
100500     DISPLAY "Hello world!" LINE
15 POSITION 10.
100600     STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800     EXIT.
```

How to convert it into machine code

- Mechanical (Ye Olden Days)
- Assemble it (almost 1:1 mapping)
- Compile it (difficult)
- Interpret it (difficult)

Compilation

Convert this:

```
if (src_data.end_of_input && src_data.output_frames_gen == 0)
                                    return 0;

        src_data.data_in += src_data.input_frames_used *
                            source->channels();
      src_data.input_frames -= src_data.input_frames_used ;

    return src_data.output_frames_gen * source->channels();
```

into this:

10101010100110101011101000101010101010101

Interpreters

- Fundamentally the same task as compiling
- Done when the program is run, not before
- The program is read by another program, which translates it into lower level (simpler) pieces
- The lowest level pieces are bits of code inside the interpreter that were compiled already
- Execute those bits of code in the right order
- Some interpreters actually do run-time compilation to machine code

Interpreters 2

- Examples: Python, Java, Perl, PHP, Ruby
- Tend to be “high level languages”
- Interpreters for C and other low level languages do exist but are generally slow
- **Major benefits:**
- Less (or different) bugs
- Easier to use libraries
- Higher level ideas to work with
- Reduced development time
- **Major drawbacks:**
- Slower
- Not realtime safe (generally)

The Development Process

- Compiled programs: Edit/Compile/Run-Debug
- Interpreted: Edit/Run-Debug
- Development methodologies
- Pair programming
- Users as debuggers

How Software Gets to Users

- OK, so somebody wrote a piece of software
  - How to get it to people who might use it?
    - Could give them the source code
  - Could give them the compiled version
    - A fundamental choice

Giving Users The Source

- User has to compile it
- User can modify it
- User can read it
- User (or designated body) can fix it
- The source code is portable to new machines
- The source code reveals ideas
- The source code inspires ideas

Giving Users the Executable

- Users get to just 'click-n-run'
- Less chance of errors during user-driven compile
- Much easier to support
- No (easy) way for users to see the ideas
- No (easy) way for users to fix issues or modify the program

Problems with distributing software

- Is the target platform predictable?
- Where to install?
- How to install?
- What else does the user need to install?
- Will it break anything?
- How to uninstall?
- OS monopolies make this easier...
- But not that much easier

Early Hacker Culture

- Started in the 1950's-1960's
- Computers as big as a room
- Shared physical resource => programs get shared
- Pride in a “good hack”
- Source code on paper

Hackers Spreading

- End of the 1960's, early 1970's saw first home or “hobbyist” computers
- Early hackers saw an opportunity for fun and also to provide software to others
- Community computing
- Source code on floppy disc
- Very little to no money involved

The Gates Letter, 1976

February 3, 1976

An Open Letter to Hobbyists

To me, the most critical thing in the hobby market right now is the lack of good software courses, books and software itself. Without good software and an owner who understands programming, a hobby computer is wasted. Will quality software be written for the hobby market?

Almost a year ago, Paul Allen and myself, expecting the hobby market to expand, hired Monte Davidoff and developed Altair BASIC. Though the initial work took only two months, the three of us have spent most of the last year documenting, improving and adding features to BASIC. Now we have 4K, 8K, EXTENDED, ROM and DISK BASIC. The value of the computer time we have used exceeds $40,000.

The feedback we have gotten from the hundreds of people who say they are using BASIC has all been positive. Two surprising things are apparent, however. 1) Most of these "users" never bought BASIC (less than 10% of all Altair owners have bought BASIC), and 2) The amount of royalties we have received from sales to hobbyists makes the time spent of Altair BASIC worth less than $2 an hour.

Why is this? As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if the people who worked on it get paid?

Is this fair? One thing you don't do by stealing software is get back at MITS for some problem you may have had. MITS doesn't make money selling software. The royalty paid to us, the manual, the tape and the overhead make it a break-even operation. One thing you do do is prevent good software from being written. Who can afford to do professional work for nothing? What hobbyist can put 3-man years into programming, finding all bugs, documenting his product and distribute for free? The fact is, no one besides us has invested a lot of money in hobby software. We have written 6800 BASIC, and are writing 8080 APL and 6800 APL, but there is very little incentive to make this software available to hobbyists. Most directly, the thing you do is theft.

What about the guys who re-sell Altair BASIC, aren't they making money on hobby software? Yes, but those who have been reported to us may lose in the end. They are the ones who give hobbyists a bad name, and should be kicked out of any club meeting they show up at.

I would appreciate letters from any one who wants to pay up, or has a suggestion or comment. Just write me at 1180 Alvarado SE, #114, Albuquerque, New Mexico, 87108. Nothing would please me more than being able to hire ten programmers and deluge the hobby market with good software.

Bill Gates

Bill Gates
General Partner, Micro-Soft

Homebrew Computer Club Newsletter - Volume 2, Issue 1, January 31, 1976

Stuff Breaks (1980)

- MIT networked printer upgrade
- Old model had been hacked to send a message to a user (often on a different floor)
- MIT denied access to the new printer's source code
- No hack ... inconvenient ... irritation ... inspiration
- 1985, Richard Stallman founds the GNU Project
- GNU's Not Unix
- Goal: to write an entire Unix-like operating system

Free Software

The modern definition has four points, which are numbered zero to three. Free software is identified by whether or not the recipient has the freedoms to:

run the program, for any purpose (freedom 0)

study how the program works, and adapt it to your needs (freedom 1)

redistribute copies so you can help your neighbor (freedom 2)

improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3)

Note that "Access to the source code is a precondition" for freedoms 1 and 3.

Free, Libre, Open

- Stallman was first to label and define
- “Free” is problematic in english, but not in French and many other languages
- “Free as in Beer, or Free as in Freedom
- Others had different ideas about why of giving users the source are important

Reasons to Like Open Source

- Freedom
- Cost
- Development issues
- Media buzzword
- Longevity

Support

- Its all the same, but different
  - Online support options
- The significance of word-of-mouth

Real World Economics 1

Harman International Industries is loud and clear. It makes high-end stereo and audio equipment for consumer and professional markets. Its consumer group makes loudspeakers, CD and DVD players, CD recorders, and amplifiers under brands Harman/Kardon, Infinity, Becker, Logic 7, JBL, Mark Levinson, and others. Harman's auto unit sells branded audio systems through several car makers, including Toyota/Lexus and General Motors. Its professional unit makes audio equipment, such as studio monitors, amplifiers, microphones, and mixing consoles for recording studios, cinemas, touring performers, and others. KKR & Co. and Goldman Sachs abandoned an $8 billion leveraged buyout of Harman in September 2007.

Real World Economics 2

Harman International
Key numbers for fiscal year ending June, 2008:
Sales: $4,112.5M
One year growth: 15.8%
Net income: $107.8M
Income growth: (65.7%)

Real World Economics 3

People whose job is to mix it up, turn to Digidesign. It makes computer-based digital audio production systems and software for the music, film, multimedia, and broadcasting industries. Its brand names include ProTools, VENUE, and ICON. The company's tools assist in tasks including audio creation, post production, collaboration, and distribution. Digidesign, a unit of Avid Technology since 1995, also markets computer audio peripherals through its M-Audio division (acquired in mid-2004). In 2006 Avid Technology acquired London-based Sibelius Software, a developer of music applications software products, and combined it Digidesign.

Key numbers for fiscal year ending December, 2007:
Sales: $167.4M

Real World Economics 4

This company doesn't produce records, but it does help people make plenty of music. Yamaha is the world's leading maker of musical instruments, including pianos, electronic keyboards, and synthesizers. It also manufactures and distributes a wide variety of wind and percussion instruments, as well as guitars, violins, and other string instruments. Yamaha also is a leading manufacturer of home audio and video equipment and makes a line of professional audio gear. In addition, the company makes semiconductors and other electronic components. Yamaha Corporation is motorcycle manufacturer Yamaha Motor Co.'s largest shareholder.

Key numbers for fiscal year ending March, 2007:
Sales: $4,667.1M
One year growth: 2.8%
Net income: $236.3M
Income growth: (1.2%)

What US$1M Gets You

- 10 really good programmers
- 5 really good programmers, a good manager, a secretary, an office, and a little profit
- 4 really good programmers, a good manager, a secretary, an office, and a marketing campaign

You Guessed This Part

US$1M annual revenue is HUGE in the audio technology world

On the other hand ...

- Ableton employs at least 70 people, has good managers, widespread marketing
  - The message is: it CAN work
- The corollary is that it NORMALLY doesn't

Lesson

- Make something Joe The Musician might buy 10 times ... Profit!
- Make something Joe the Audio Engineer might buy once... Struggle!
- Trade shows (MusikMesse, NAMM) dominated by ... guitars, monitors, mics, amps
- Musicians FTW!

A Real Free Software Story

- Yes, you guessed it
- 8 years
- 40 developers
- 5000 source downloads a month
- 5000 OS X downloads a month (?)

Myths

- If you build it they will come
- A thousand eyes make all bugs shallow
- Open source software is bad
- Proprietary software is bad
- Its (easier|harder) to get paid with open source
- Just sell service

Next Week

- Plugins #1
- Find a feature of a piece of music/audio software that is really good
- Find a feature of a piece of music/audio software that think could be done better
- Explain both things