Why
the Unix Philosophy
still matters

markus schnalke <meillo@marmaro.de>

Goals of this talk

- Introduce the Unix Phil
- Show that most modern software is crap
- Explain why the Unix Phil leads to good/better software
- Convince you that good software is of matter

- Make you think

Roadmap

- Background
- What is the Unix Phil?
- The Unix Phil after Gancarz
- Real world examples
- Final thoughts

Background

How I met the Unix Phil

First contact through dwm (suckless project)

“cat -v Considered Harmful”

“The Unix and the Echo”

“The Unix Programming Environment”

Better understanding through digging in the past

Historical background

The late 60s and early 70s

Operating systems are complex

Brooks’ “The Mythical Man-Month”

MULTICS had just failed

A lot of different hardware

Limited computing power

Textual input and output (line printers)

Everything is a file

Is the(?) basic concept in Unix (and even more in Plan9)

Made simple operating systems possible

It is not covered by the Unix Phil
The Unix Phil is on a different level

What is the Unix Phil?

Unix is mainly two things:

- An operating system (system calls)
- A toolchest (coreutils)

What is the Unix Phil *itself*?

“The Unix philosophy is a set of cultural norms and philosophical approaches to developing software based on the experience of leading developers of the Unix operating system.” (wikipedia)

How the inventors of Unix write software.

Common things of classic Unix tools.

Difficult to define

Unix Phil vs. SW dev processes

The Unix Phil

- much: *what* to program
- few: *how* to program

Software developments processes:

- few: *what* to program
- much: *how* to program

Extreme Programming is like the Unix Phil but with more *how* than *what*, and with formalisms

What is the Unix Phil?

- Doug McIlroy (1978)
- Mike Gancarz: “The Unix Philosophy” (1994)
- Eric S. Raymond: “The Art of Unix Programming” (2003)

- Richard Gabriel: “Worse is Better” (1989)

Doug McIlroy

This is the Unix philosophy:

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface.

Mike Gancarz: “The Unix Philosophy”

- Small is beautiful.
- Make each program do one thing well.
- Build a prototype as soon as possible.
- Choose portability over efficiency.
- Store data in flat text files.
- Use software leverage to your advantage.
- Use shell scripts to increase leverage and portability.
- Avoid captive user interfaces.
- Make every program a filter.

Plus ten lesser tenets

The Unix Phil after Gancarz

Small is beautiful

Small software is easier to understand, write, maintain

Less lines of code contain less bugs

Monsters are large

Make each program do one thing well

Programs with many functions are large

One thing is easier to understand

Often straight forward to implement

Toolchests

Reusable

Build a prototype as soon as possible

Shows the quality of the design

Shows the problems of the software

The best way to shape a software

Users find bugs

Incremental development

Choose portability over efficiency

(Originates in a lot of incompatible hardware in history)

Use is most important

Availability

Only needs to be fast enough

Store data in flat text files

(originally: “Store numerical data in flat ASCII files”)

Binary data is machine-dependent

Human readable data is:

- As generic as possible
- Is very likely supported
- Many tools work on it (Unix toolchest)
- Directly editable by humans

Processing needs only to be fast enough

Use software leverage to your advantage

What do we have computers for?

Make best use of computing power

Reduce development effort

Toolchests and a powerful shell

Use shell scripts to increase leverage and portability

- Was very important in history
- High level languages
- Prototyping
- Quick hacks
- Users are “programmers”

Avoid captive user interfaces

- Don’t assume the user to be human
- Exclude the user whenever possible
- Automate
- How does it scale?
- Bloat

Make every program a filter

- Programs transform data
- Combine programs
- Have one common interface
- Toolchests

real world examples

find -printf

How to reformat the output of find(1) to have “FILENAME PATH” instead of “PATH/FILENAME”?

The “easy” way: `find /dir -printf "%P %h\n"`

The “good” way:
`find /dir | sed 's,\(.*\)/\(.*\),\2 \1,'`

The difference shows off when one wants, for instance, the path to be manipulated further.

Source (in German):
`http://debianforum.de/forum/viewtopic.php?t=117683`

Various

Who uses `grep -R` ?

`cat -v`

Pagers are taken for granted

What about the readline?

MH / nmh

A Mail User Agent (MUA)

Is a toolchest

Work with mails like with generic files

The only(?) MUA that follows the Unix Phil

Has a very special feeling

uzbl

A web browser that adheres to the Unix Phil

A young project (about 1 year)

Central question:
What is the one task a web browser covers?

Makes very visible use of software leverage

Suffers hard from our broken web

Final thoughts

Say no

In today's computer world, following the Unix Phil means often asceticism

One needs to abjure a lot of "nice" features

Actually, it is abjuring the *bad* features

Leads to a valuable attitude, IMO

Transfer it to your everyday life

Avoid complexity

Avoid complexity first and foremost

Complexity is the "boss enemy", software developers fight against

Strive for simplicity, clarity, generality

Transfer it to your everyday life

Good solutions

We don't need just solutions, we need good ones

Today, we can make almost everything possible, but we still cannot make it good

Transfer it to your everyday life

Live it

The Unix Phil is more than just a few guidelines

You cannot follow only some of the tenets

To understand the Unix Phil, you need to engage with it

It's a philosophy – live it!

References

Literature

- "**The Unix Philosophy**" by Mike Gancarz
  Go and get it!
- "**The Unix Programming Environment**" by Kernighan and Pike
  A Bible for Unix-lovers.
- "**The Mythical Man-Month**" and "**No Silver Bullet**" by Fred Brooks
  About complexity in software development.
- "**The Practice of Programming**" by Kernighan and Pike
  How good code should look like.
- "**cat -v Considered Harmful**" by Pike and Kernighan
  `http://harmful.cat-v.org/cat-v/`
  A must-read.

This talk was prepared using tools of the Heirloom project:
`http://heirloom.sf.net`

The slides macros are based on
`http://repo.cat-v.org/troff-slider/`

The slides are available on my website
`http://marmaro.de/docs` and on
`http://ulm.ccc.de/ChaosSeminar/`

See my paper on the topic, too.

2010-03-08