

# Unix Philosophy

**Author:** Daniele Pizzolli  
**Contact:** [ors@tovel.it](mailto:ors@tovel.it)  
**Copyright:** 2010 Some Rights Reserved <http://creativecommons.org/licenses/by-sa/3.0/>  
**Location:** Villa Sant'Ignazio — Trento  
**Version:** Draft  
**Note:** Please send your feedback.

Basically it is a summary from the [TAoUP].

I think that the quotations are allowed as *fair use*.

## Unix Philosophy Condensed

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

Doug McIlroy (quoted in “A quarter Century of Unix” [Salus])

## Does Unix Philosophy still matter?

Now there are...

- GUI
- Touch Screen
- Multi Touch Screen
- HTML, XML
- Multimedia: images, audio, video...
- Embedded Interaction(?)
- ...

## Let see Unix Ideas with more details

From [TAoUP], starting from page 13; misleading examples are mine.

### Rule of Modularity (1)

Write simple parts connected by clean interfaces.

```
$ echo Modularity | tee rules.txt
```

### Rule of Clarity (2)

Rule of Clarity: Clarity is better than cleverness.

```
$ for i in 1 2 3; do echo $i; done
```

## Rule of Composition (3)

Design programs to be connected to other programs.

```
$ find -type f -iname '*txt' -print0 | xargs -0r wc -l | sort -n
```

## Rule of Separation (4)

- Separate policy from mechanism
- Separate interfaces from engines

```
$ sudo /etc/init.d/ssh restart
$ sudo tcpdump -s 0 -ni eth0 -w dump; wireshark dump
$ sudo ettercap -Tzq
```

## Rule of Separation (4)[p.2]

TODO

Code:

```
#!/bin/sh
#
# A simple example of dependency between packages
# TODO: trap for TMPFILE deletion

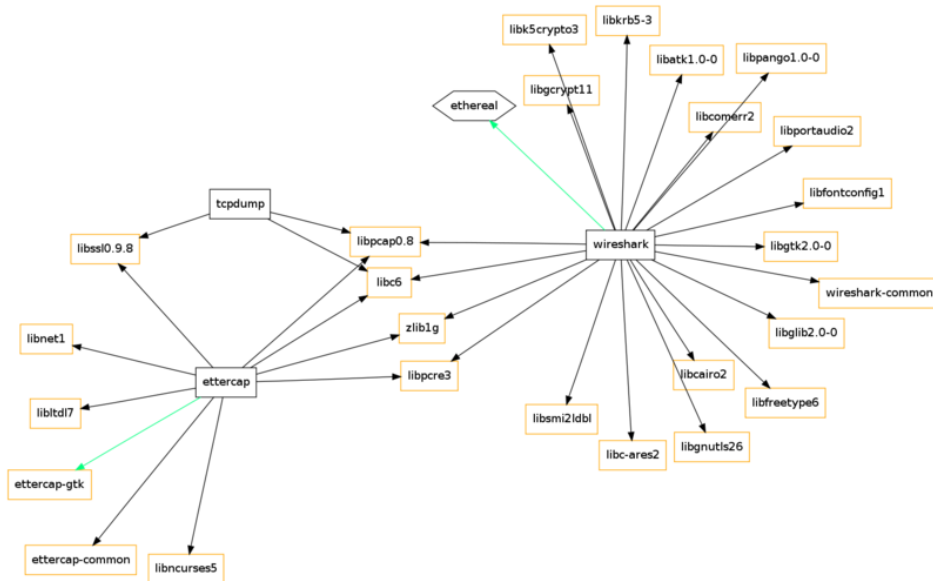
set -e

TMPFILE="$(mktemp -t tmp-XXXX)"
OUTPUT_DIR="output"
OUTPUT_FILE="dep_graph.png"

apt-cache -o APT::Cache::GivenOnly=1 dotty tcpdump wireshark ettercap > "${TMPFILE}"
fdp -T png -o "${OUTPUT_DIR}/${OUTPUT_FILE}" "${TMPFILE}"
mogrify -geometry 800x600 "${OUTPUT_DIR}/${OUTPUT_FILE}"
rm "${TMPFILE}"

# NEOF
```

Result:



Package Dependencies

## Rule of Simplicity (5)

- Design for simplicity;

```
$ halt
```

- add complexity only where you must.

```
$ gcc -s -O2 -W -Wall -ansi -Wstrict-prototypes -Wpointer-arith -pedantic hello.c
```

## Rule of Parsimony (6)

- Write big program only when it is clear by demonstration that nothing else will do.

```
$ dpkg-query -W -f='${Installed-Size} ${Package}\n' 'emacs*' 'vim*' | \
  grep -v '^ ' | sort -n
```

## Rule of Transparency (7)

- Design for visibility to make inspection and debugging easier.

```
$ set -x; echo test; ls > /dev/null; set +x
```

## Rule of Robustness (8)

- Robustness is the child of transparency and simplicity.

```
$ # Note: output discarded
$ dd bs=100 count=100 if=/dev/urandom | tee rand.bin | sort > sort.bin
```

```
$ diff -a rand.bin sort.bin > diff.bin
$ cp rand.bin rand1.bin
$ patch rand1.bin < diff.bin
$ md5sum rand.bin rand1.bin sort.bin
$ # Note: output non discarded
$cmp rand.bin rand1.bin
cmp: EOF on rand.bin
```

*Note: almost correct...*

## Rule of Representation (9)

- Fold knowledge into data, so program logic can be solid and robust.

```
$ head -n 1 /etc/passwd /etc/group
```

## Rule of Least Surprise (10)

- In interface design, always do the least surprising thing.

```
$ rm -i file
```

## Rule of Silence (11)

- When a program has nothing surprising to say, it should say nothing.

```
$ clear
$ reset
```

## Rule of Repair (12)

- Repair what you can -- but when you must fail, fail noisily as soon as possible.

```
$ shopt -s cdspell
```

## Rule of Economy (13)

- Programmer time is expensive; conserve it in preference to machine time.

```
$
```

## Rule of Generation (14)

- Avoid hand-hacking; write program to write programs when you can.

```
$ less /usr/share/shorewall-perl/Shorewall/Compiler.pm
```

*You must be really a master of two or more world to do this.*

## Rule of Optimization (15)

- Prototype before polishing. Get it working before optimize it.

```
$ usercount() { ps -A -o ruser=RealUser | cut -d ' ' -f 1 | sort | uniq }
$ usercount() { who | cut -d ' ' -f 1 | sort | uniq }
```

*This is a quite poor example*

## Rule of Diversity (16)

- Distrust all claims for “one true way”.

```
$ head
$ sed -e 10q
$ awk '(FNR <= 10)'
```

## Rule of Extensibility (17)

- Design for the future, because it will be here sooner than you think.

```
$ date --date='@0'
$ date --date='@987654321'
$ date --date='@9876543210'
```

*but keep an eye on the past too:*

```
$ date +%s --date='Jan 1 00:00:01 CET 1969'
$ date -62135599735
$ date +%s --date='Jan 1 00:01:01 CET 0001'
```

## K.I.S.S.

- Keep
- It
- Simple
- Stupid!

The KISS principle