Introduction to the UNIX Environment: Computing as Praxis

Claude Rubinson, Ph.D.
University of Houston—Downtown
rubinsonc@uhd.edu
cjr@grundrisse.org

ACM UHD
Houston, Texas
October 29, 2019
Why does a 50 year-old technology dominate modern computing?
Resources


• Thomas Scoville, “UNIX as Literature” (1998, Miller Freeman)

The UNIX Philosophy

- Assume that users are intelligent and know what they’re doing.
  - “UNIX gives you just enough rope to hang yourself—and then a couple of more feet, just to be sure.”
  - “UNIX was not designed to stop you from doing stupid things, because that would also stop you from doing clever things.”

- Complexity is dangerous: privilege simplicity above correctness, consistency, and completeness.

- Nobody can predict the future; be humble and remain flexible.
The UNIX Environment

- UNIX is not merely a platform upon which to run applications; rather, the commandline is your primary interface.

- A properly designed UNIX program, shell script, or shell function will integrate into your environment; can think of UNIX programs as “extensions” or “plugins” to the environment.

- Over the long-term, you customize your environment to your needs, preferences, and style.
Gancarz’s Tenets of UNIX

1. Small is beautiful
2. Make each program do one thing well.
3. Build a prototype as soon as possible.
4. Choose portability over efficiency
5. Store [numerical] data in flat ASCII files.
6. Use software leverage to your advantage.
7. Use shell scripts to increase leverage and portability.
8. Avoid captive user interfaces.
9. Make every program a filter.
Small is beautiful; Make each program do one thing well

Small programs:

- are easy to understand.
- are easy to maintain.
- consume fewer system resources.
- are easier to combine with other tools.
Make every program a filter; Avoid captive user interfaces

• All computer programs are filters; UNIX makes this explicit and puts the user in control.
  – cf., GUI applications and racist machine learning algorithms

• CUIs trap data and assume that their user is human, which means that they don’t interface well with other programs.
Build a prototype as soon as possible

- Shell scripting encourages rapid, iterative test/rewrite development
  - cf., Manifesto for Agile Software Development (2001)
- “Worse is better”
  - Seek the 90% solution
  - “When in doubt, use brute force.”
Choose portability over efficiency; Store [numerical] data in flat ASCII files

• The UNIX environment generally assumes plain text and, therefore, provides a rich set of line-oriented text manipulation tools (e.g., editors and version control) and languages (e.g., grep/sed/awk, sort|uniq -c, calc/bc/dc).

• stdin, stdout, and stderr don’t distinguish between plain text and binary streams.
Use software leverage to your advantage; Use shell scripts to increase leverage and portability

- Shell scripts allow you to build up your analysis incrementally by piping from stdout to stdin.
- Other programs are treated as black boxes.
- Is often easier to call other programs from the shell than to shell out.
- Shell pipelines are often highly performant
- But: largely unidirectional (FIFOs and temp files can help here).
“UNIX is user-friendly. It’s just picky about who its friends are.”