

La filosofia Unix/Linux nella progettazione e implementazione del software

**Laboratorio di Sistemi Operativi I
Anno Accademico 2005-2006**

Francesco Pedullà
(Tecnologie Informatiche)

Massimo Verola
(Informatica)

Copyright © 2005 Francesco Pedullà, Massimo Verola
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license can be found at: <http://www.gnu.org/licenses/fdl.html#TOC1>

Basi della filosofia Unix

- Il sistema operativo Unix e le sue varianti hanno il loro punto di forza in una filosofia di base nell'approcciare l'analisi ed il progetto del software
- La filosofia Unix non si basa su un metodo formale di progettazione ma piuttosto si è evoluta con un metodo bottom-up, in cui la prassi e l'esperienza hanno portato alla identificazioni di alcune “regole generali” che è bene tenere sempre a mente
- Tali regole sono utili a qualsiasi programmatore in qualsiasi ambiente di sistema operativo
- Tali regole sono esposte e spiegate nel classico “The Art of UNIX Programming” di Eric S. Raymond, uno degli sviluppatori di Unix dal 1982

La regole per la progettazione ed implementazione del SW

1. Rule of Modularity: Write simple parts connected by clean interfaces.
2. Rule of Clarity: Clarity is better than cleverness.
3. Rule of Composition: Design programs to be connected to other programs.
4. Rule of Separation: Separate policy from mechanism; separate interfaces from engines.
5. Rule of Simplicity: Design for simplicity; add complexity only where you must.
6. Rule of Parsimony: Write a big program only when it is clear by demonstration that nothing else will do.

La regole per la progettazione ed implementazione del SW

7. Rule of Transparency: Design for visibility to make inspection and debugging easier.
8. Rule of Robustness: Robustness is the child of transparency and simplicity.
9. Rule of Representation: Fold knowledge into data so program logic can be stupid and robust.
10. Rule of Least Surprise: In interface design, always do the least surprising thing.
11. Rule of Silence: When a program has nothing surprising to say, it should say nothing.
12. Rule of Repair: When you must fail, fail noisily and as soon as possible.

Regole per la progettazione ed implementazione del SW

13. Rule of Economy: Programmer time is expensive; conserve it in preference to machine time.
14. Rule of Generation: Avoid hand-hacking; write programs to write programs when you can.
15. Rule of Optimization: Prototype before polishing. Get it working before you optimize it.
16. Rule of Diversity: Distrust all claims for "one true way".
17. Rule of Extensibility: Design for the future, because it will be here sooner than you think.