A Filosofia do Unix

Rubens Queiroz de Almeida
rubens.queiroz@gmail.com
www.Dicas-L.com.br

Um Pequeno Histórico

Ken Thompson (sentado) e Dennis Ritchie (de pé) em um PDP-11 em 1972.

Um Pequeno Histórico

Que os céus protejam aquele que deixar cair a sua caixa de cartões perfurados

- Baseado no sistema MULTICS (*Multiplexed Operating and Computing System*)
- Projeto conjunto GE, MIT, Bell Labs
- Objetivo: demonstrar que um sistema operacional de propósito geral, multiusuário, com compartilhamento de tempo era viável
- 1969: a parceria de desenvolvimento do MULTICS se defaz

Um Pequeno Histórico

- UNICS - *Uniplexed Operating and Computing System*
- 1969: é escrita, por Ken Thompson, a primeira versão do Unix
- 1971: o sistema é portado para um PDP-11, com 16 kbytes de memória, 8 kbytes para os programas dos usuários e um disco de 512 kbytes
  - *O kernel continha 11.000 linhas de código, das quais 10.000 foram reescritas mais tarde na linguagem C.*

Um Pequeno Histórico

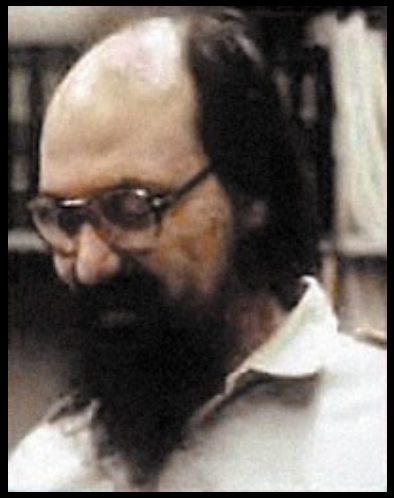
- 1978: primeira versão BSD do Unix
- 1984: a Universidade da Califórnia em Berkeley, libera a versão 4.2BSD, que incluía uma implementação completa dos protocolos TCP/IP.
- 1991: Aparece o Linux
- 2010 - Microsoft vai à falência

O Kernel do Unix

O kernel do Unix é o único código que não pode ser substituído pelo usuário segundo seu gosto. Por esta razão, o kernel deve tomar o mínimo possível de decisões reais.

Ken Thompson,
Unix Implementation
The Bell System Technical Journal, vol 57, Nº 6, Julho/Agosto 1978 pg. 1931

Ainda o Kernel do Unix



Ken Thompson

Eu diria que a maior realização intelectual embutida no Unix é o sucesso que Ken Thompson e Dennis Ritchie tiveram em compreender quanto pode ser deixado fora de um sistema operacional sem diminuir sua capacidade.

Dennis Ritchie

The Bugs

The UNIX manuals are sometimes derided for the BUGS section.

This is the place where the author(s) of a program list its design limitations. One UNIX critic said of this policy: *If they know about the bugs, why don't they fix them?* The point is that the early UNIX authors established the beneficial habit of documenting limits to the program, rather than always letting the end user find them.

Dennis Ritchie comments:

Every other manual has bugs sections; they just aren't published. Many of the BUGS sections were intended as pointers for further development of the programs, rather than as warnings to the user.

Ritchie adds:

Our habit of trying to document bugs and limitations visibly was enormously useful to the system. As we put out each edition, the presence of these sections shamed us into fixing innumerable things rather than exhibiting them in public. I remember clearly adding or editing many of these sections,then saying to myself "*I can't write this*," and fixing the code instead.

Ritchie, personal correspondence

Postulados Básicos

- Pequeno é bonito
- Faça cada programa desempenhar apenas uma tarefa e bem
- Crie um protótipo o mais rápido possível
- Prefira a portabilidade à eficiência
- Armazene dados numéricos em arquivos ASCII puros
- Use o software para alavancar seus benefícios
- Use shell scripts para aumentar e portabilidade
- Evite interfaces para o usuário cativas
- Faça de cada programa um filtro

Postulado 1

Pequeno é bonito

- Programas pequenos são:
  - fáceis de se entender
  - fáceis de se manter
  - consomem menos recursos do sistema
  - são mais fáceis de combinar com outras ferramentas

Postulado 2

Faça cada programa desempenhar apenas uma tarefa e bem

- A Mosca do Jacques Cousteau
- O programa deve fazer o que tem que fazer e sair do caminho
- Princípio K.I.S.S
- Precisa mesmo escrever o programa?
- Programas unifuncionais tendem a ser pequenos
- Programas pequenos tendem a ser unifuncionais
- Se você não consegue fazer uma coisa bem, então você não entendeu o problema

Postulado 3

Crie um protótipo o mais rápido possível

- Ninguém sabe tudo ou pode prever o futuro
- Ninguém sabe exatamente o que quer
- Ninguém consegue comunicar exatamente o que quer
- Mesmo os mestres sabem que mudanças são inevitáveis
- Softwares nunca são terminados, apenas liberados
- Protótipos feitos mais cedo reduzem os riscos
- Os três sistemas do homem

Postulado 4

Prefira a portabilidade à eficiência

- O hardware do ano que vem vai ser mais rápido
- Não gaste muito tempo tornando um programa mais rápido
- A maneira mais eficiente raramente é portável
- Software portável reduz despesas com treinamento
- Programas bons nunca morrem: são portados para outras plataformas
- Atari 2600: Atari VCS

Postulado 5

Armazene dados numéricos em arquivos ASCII puros

- ASCII é um formato consagrado para troca de dados
- Fácil de ler e editar
- Simplificam o uso de ferramentas Unix de manipulação de texto
- awk, cut, diff, expand, expr, fmt, grep, head, lex, more, paste, roff, sed, sort, tail, test, tr, wc
- Maior portabilidade supera a falta de velocidade
- A próxima geração de computadores irá resolver os problemas de performance

Postulado 6

Use o software para obter uma alavancagem de seus benefícios

- Programadores bons escrevem bom código, programadores geniais tomam emprestado código bom
- Evite a síndrome do Não Foi Inventado Aqui (NIH)

Postulado 6 (...)

- Permita que outras pessoas usem o seu código para alavancar seu próprio trabalho
- Automatize o máximo possível
  - Não perca tempo fazendo manualmente o que o seu computador pode fazer

Uma idéia compartilhada vale duas na cabeça

Postulado 7

Use shell scripts para aumentar a alavancagem e a portabilidade

```
echo `who|awk '{print $1}'|sort|uniq`|sed 's/ /,/g'
```

```
  echo                   177
  who                    755
  awk                  3.412
  sort                 2.614
  uniq                   302
  sed                  2.093
  ======================
  Total                9.353
```

Postulado 7 (...)

- Shell scripts propiciam uma grande alavancagem
- Também economizam o seu tempo
- São mais portáveis que código C
- Resista à tentação de reescrever shell scripts em C

Postulado 8

Evite interfaces para o usuário cativas

- (CUI - Captive User Interfaces)
- CUIs pressupõem que o usuário é humano
- CUIs tendem a adotar o enfoque Grande é Bonito
- Programas com CUIs são mais difíceis de combinar com outros programas
- CUIs não escalam bem
  - Ex.: Criação de usuários
- Não se aproveitam da alavancagem do software

Postulado 9

Faça de cada programa um filtro

- Todos os programas são filtros
- Programas não criam dados, as pessoas sim
- Computadores convertem dados de um formato para outro

Programas como Filtros

- stdin
- stdout
- stderr

Postulados Secundários

- Permita que o usuário adapte seu ambiente
- Faça o kernel do sistema operacional pequeno e leve
- Use letras minúsculas e não complique
- Salve as árvores
- O silêncio é de ouro
- Pense em paralelo
- A soma das partes é maior do que o todo
- Procure a solução 90%
- Pior é melhor
- Pense hierarquicamente

Bibliografia

- *The Unix Philosophy*, Mike Gankarz, Digital Press
- *Free as in Freedom*, O'Reilly and Associates
- *A Quarter Century of Unix*, Peter Salus, Addison-Wesley
- *The Art of Unix Programming, Eric Raymond*
  - *http://www.faqs.org/docs/artu/*
- *Bell Labs Computing Science Technical Reports* (CSTR)
  - *http://www.cs.bell-labs.com/cm/cs/cstr.html*

Referências Adicionais

- Dicas-L: software livre, artigos, milhares de dicas de informática, cursos
  - http://www.Dicas-L.com.br
- Aprendendo Inglês
  - *http://www.aprendendoingles.com.br*
- Contando Histórias
  - *http://www.contandohistorias.com.br*
- Sistema Rau-Tu de Perguntas e Respostas
  - *http://www.rau-tu.unicamp.br*


Fim
村松誠【MAKOTO】