Emacs e eev, ou: como automatizar quase tudo

Eduardo Ochs - http://angg.twu.net/

13 de agosto de 2005

O que se diz por aí

Emacs é um editor de texto

- enorme
- complicado
- ocupa muita memória
- usa muitas teclas

Mas na verdade o Emacs é o editor de texto *extensível* mais simples possível.

Outra definição pro Emacs

Emacs é um *ambiente Lisp*
(criado na década de 1970)
e Lisp é uma linguagem de programação
(criada na década de 1950).

Isso não é meio velho?
Geometria, números inteiros e frações têm milhares de anos.
A noção de número que a gente tem hoje em dia é de 1858.[1]

O que é um computador?
Como fazer uma máquina que executa operações em seqüência?

- 1822: Difference Engine (Charles Babbage; mecânica)
- 1936: Turing Machine (Alan Turing; teórica)
- 1940+: computadores eletrônicos

[1] definida por "cortes de Dedekind":
$0.999999999\ldots = 1.0000000000\ldots = 1$

O que é um computador

= Memória + processador.

O processador lê alguns bytes da memória e
"executa uma instrução" —
a memória (o "estado" do computador) se modifica.

Um trecho da memória corresponde aos pontos da tela.
Algumas posições da memória correspondem ao teclado.

Um computador assim é parecido com uma Máquina de Turing.
Ele é fácil de construir mas difícil de programar.

O que é um programa?

(Estamos na décadas de 1950).
Como expressar um procedimento como uma série de passos?
Como fazer uma máquina que executa estes passos para nós?

Considere: *as pessoas que estão inventando computadores e linguagens de programação em 1950 estão a par de 2500 anos de discussões.*

1950: Lisp. É mais natural (para seres humanos) expressar algoritmos em Lisp do que em "linguagem de máquina".

Problema $\xrightarrow{\text{programador}}$ linguagem de máquina

Problema $\xrightarrow{\text{programador}}$ programa em Lisp

Objetos em Lisp

Objetos em Lisp podem ser dos seguintes tipos:

- Números: 0, 1, 2, -20, 3.14
- Strings: "", "AbC", " Foo bar", "(+ 1 2)"
- Símbolos: a, b, nil, t, square, +, *
- Listas: (), (2 "dois"),
  (* (+ 1 2) (+ 3 4)),
  (concat "abc" "def")

Truque: TODAS as coisas mais complicadas são implementadas usando listas, e listas são construídas usando 'cons'es.

Como um ambiente Lisp tradicional funciona

```
LISP> (+ 1 2)
3
LISP> (* (+ 1 2) (+ 3 4))
21
LISP> (setq a 5)
5
LISP> a
5
LISP> (* a a)
25
```

$$\text{"}(+\ 1\ 2)\text{"} \xrightarrow{\text{read}} (+\ 1\ 2) \xrightarrow{\text{eval}} 3 \xrightarrow{\text{print}} \text{"}3\text{"} \qquad \text{(na tela)}$$

Como o eval funciona

$$(* \quad 3 \quad 7) \longmapsto 21$$
$$(* \quad (+\ 1\ 2) \quad (+\ 3\ 4)) \longmapsto 21$$

$$(\text{setq} \quad a \quad 22) \longmapsto 22$$
$$a \longmapsto 22$$

$$22 \longmapsto 22$$
$$\text{"abc"} \longmapsto \text{"abc"}$$
$$t \longmapsto t$$
$$nil \longmapsto nil$$
$$() \longmapsto nil$$

$$(<\ 1\ 2) \longmapsto t \quad \text{(verdadeiro)}$$
$$(<\ 2\ 0) \longmapsto nil \quad \text{(falso)}$$

$$(\text{if}\ (<\ 1\ 2)\ \text{"sim"}\ \text{"nao"}) \longmapsto \text{"sim"}$$

Tudo num sistema Lisp são objetos em Lisp

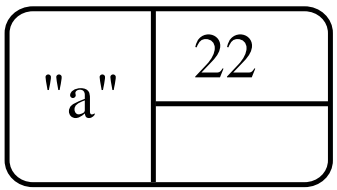
Funções (programas):

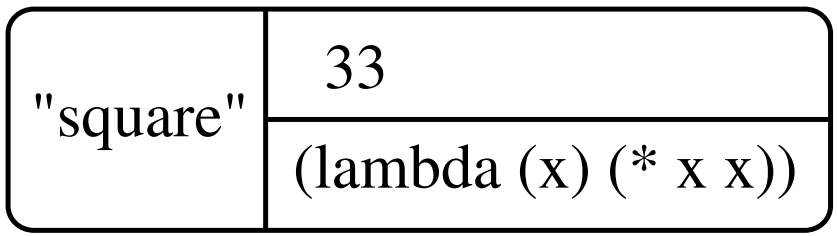
$5 \xrightarrow{\text{square}} 25$

$x \longmapsto (* \; x \; x)$

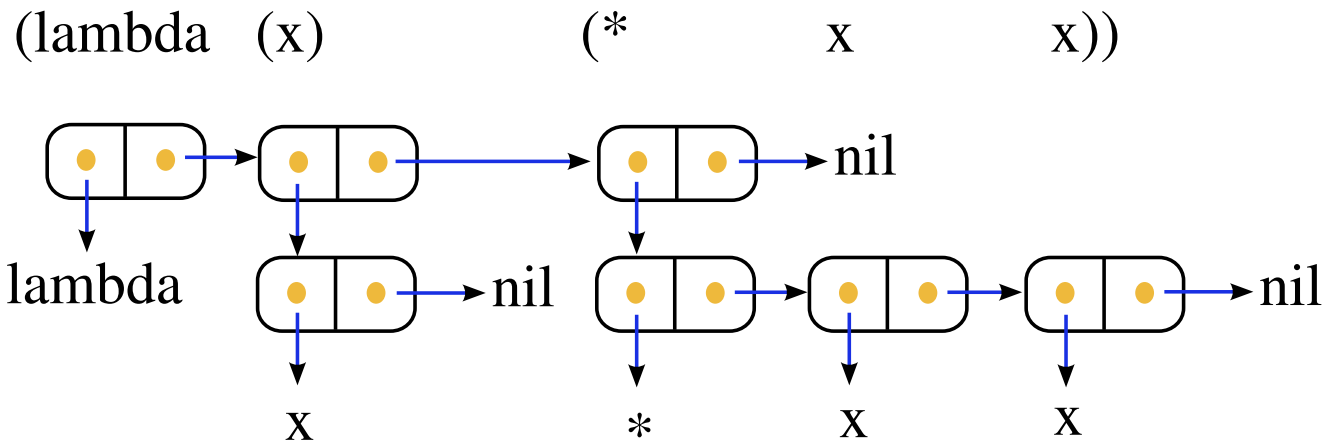
square = (lambda (x) (* x x))

(setq a 22)
(setq square 33)
(defun square (x) (* x x))





Como listas são construídas usando "cons"



O estado do sistema é determinado pelo(s) valor(es) dos símbolos.

É possível explicar o Lisp inteiro, com *todos* os detalhes, em uma hora ou menos — a parte difícil é o eval.
Nós não temos esse tempo, mas não precisamos dos detalhes do Lisp agora — é melhor vermos o Emacs primeiro.

Editores de texto extensíveis

O núcleo do Emacs é o editor de texto extensível
o mais simples possível.

Início da década de 70: TECO (editor programável)
Linguagem de progração ruim, extensões incompatíveis entre si.
Como fazer o melhor editor programável possível?

Idéia: pegue a linguagem de programação mais flexível de todas —
Lisp — e acrescente funções para transformá-la num editor.

Outra idéia importante: o editor seria desenvolvido pela
comunidade.

Emacs = Lisp + umas poucas coisas

Interface Lisp convencional:
LISP> (+ 1 2)
3
"(+ 1 2)" $\overset{\text{read}}{\longmapsto}$ (+ 1 2) $\overset{\text{eval}}{\longmapsto}$ 3 $\overset{\text{print}}{\longmapsto}$ "3" (na tela)

Interface do Emacs:
C-x C-f foo RET $\longmapsto$ "\C-x\C-ffoo\r"
$\longmapsto$ (find-file "foo") $\overset{\text{eval}}{\longmapsto}$ nil

Além disso:

- Buffers ($\leftrightarrow$) arquivos
- Cursor ("point")
- "Windows" (cada uma mostra um buffer)
- "Frames" (cada uma mostra uma ou mais windows; 1992)
- Última linha da tela: "echo area" (quando mostra mensagens), "minibuffer" (durante comandos complicados)

Emacs = Lisp + umas poucas coisas (2)

(Continuação...)

- M-x eval-last-sexp (ou: C-x C-e)
- eval-region, load-library
- Arquivo de inicialização: .emacs
- Modos:
  - "Fundamental" (básico, para edição)
  - "Dired" ("directory editor")
  - "Info"
  - "Lisp Interaction"
- Suporte a "faces" (cores e fontes) e imagens (1992)

...e em cima disso montaram um sistema modular...

Exemplo: hippie-expand (hippie-exp.el)
(Prefixos: hippie-, he-, try-)

Como aprender?

- Como eu faço X?
  - Perguntar pra pessoa do seu lado no laboratório
  - Ler a documentação
- Como Y funciona (ou: porque Y funciona assim e não como eu esperava)?
  - Perguntar pro autor do Y
  - Perguntar pra mais alguém
  - Ler o código-fonte
  - Fazer experimentos (e hipóteses) (*)
  - Rodar o código de Y passo-a-passo

Obs: em programas comerciais fechados a única possibilidade é (*).

Acesso à documentação

- Arquivos de texto
- Arquivos HTML (formato recente: 1992?)
- Manpages
- Código-fonte (indentação, syntax highlighting, tags)
- Manuais em formato Info
- Programas externos:
  - Web browsers
  - Visualizadores de arquivos ps, pdf, dvi, etc

Emacs = comunidade (1)

De <http://www.gnu.org/software/emacs/emacs-paper.html> (um artigo do Stallman de 1979):

The programmable editor is an outstanding opportunity to learn to program! A beginner can see the effect of his simple program on the text he is editing; this feedback is fast and in an easily understood form. Educators have found display programming to be very suited for children experimenting with programming, for just this reason (see LOGO).

Programming editor commands has the additional advantage that a program need not be very large to be tangibly useful in editing. A first project can be very simple. One can thus slide very smoothly from using the editor to edit into learning to program with it.

(continua)

Emacs = comunidade (2)

(Continuação:)

When large numbers of nontechnical workers are using a programmable editor, they will he tempted constantly to begin programming in the course of their day-to-day lives. This should contribute greatly to computer literacy, especially because many of the people thus exposed will be secretaries taught by society that they are incapable of doing mathematics, and unable to imagine for a moment that they can learn to program. But that won't stop them from learning it if they don't know that it is programming that they are learning! According to Bernard Greenberg, this is already happening with Multics EMACS.

Um pouco de etologia

- Macho alfa - competição, liderança, visibilidade
- Machos beta - cooperação, trabalho de grupo, educação dos filhotes
- Machos ômega - ou muito inferiores na hierarquia ou solitários (vivem fora da matilha)

Todo mundo presta atenção a cada movimento do macho alfa.
A cultura atual (noticiários, propagandas, filmes) fala principalmente de alfas.
As fêmeas preferem os alfas.

O alfa não sabe lavar pratos.

O método polonês para quase acabar com os ratos

Pegue dez ratos.
Ponha-os no tanque da área de serviço.
Dê um jeito deles não conseguirem sair.
Deixe-os com água e sem comida.
Eles vão começar a se comer uns aos outros.
Depois de um tempo (uma semana?) vai sobrar um único rato.
Solte-o na sua casa.
Os outros ratos da casa percebem o que ele passou e o que ele sabe fazer e se mudam pra outro lugar.

Esse rato é o mais forte (característica de alfa) mas os outros não querem tê-lo como líder.

O programador alfa vende produtos acabados

No mundo do software grupos de programadores competem para fazer o melhor produto.
O eev não é “o melhor programa” nem “a melhor solução”.