

7 Hábitos de um Bom Editor

Lisbon Perl Mongers
Perl Tech Meeting 10 de Julho 2007



Bram Moolenaar
Main author of Vim
Works at Google

Inspirado na apresentação “Seven habits of effective text editing”

por Bram Moolenaar

Disponível on-line um video google da apresentação em:

“7 habits For Effective Text Editing 2.0”

<http://video.google.com/videoplay?docid=2538831956647446078>

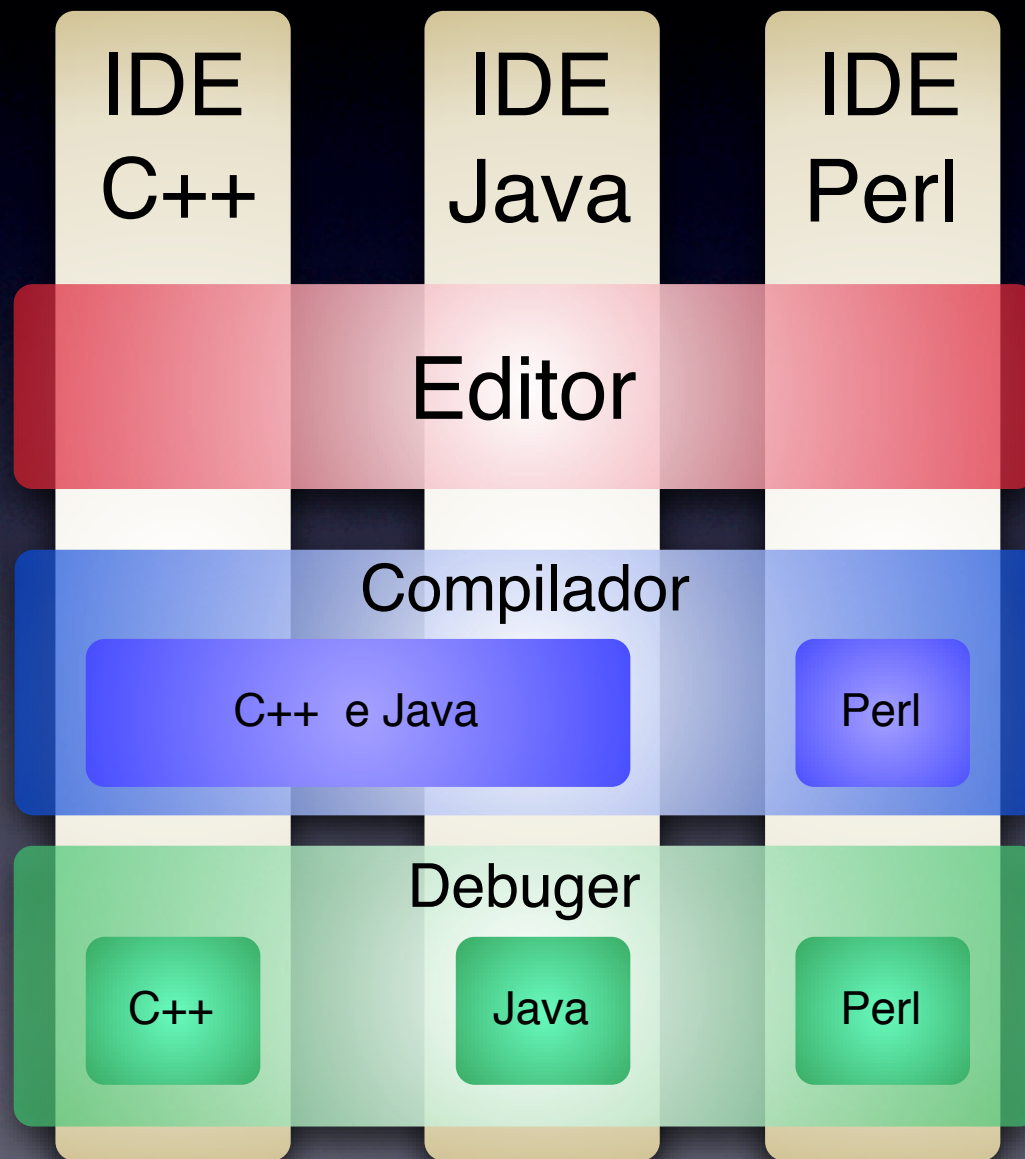
E o powerpoint disponível em:

<http://www.moolenaar.net/vim.html>

Agenda

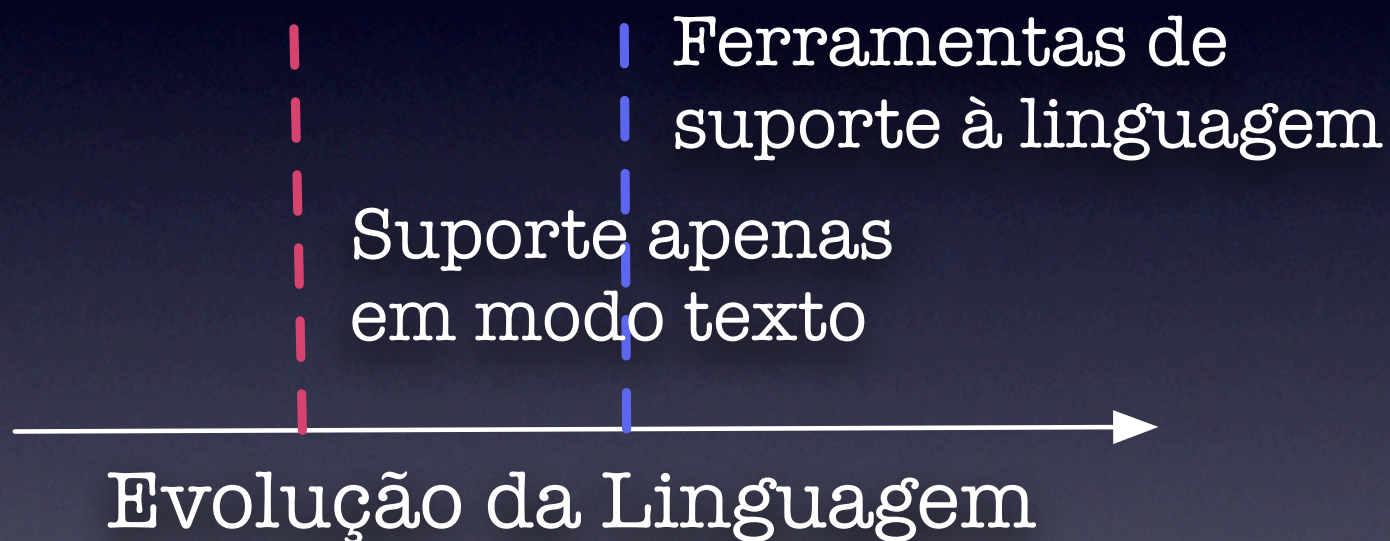
- IDE vs Editores
- 7 Hábitos para uma edição mais eficaz

IDE vs Editores



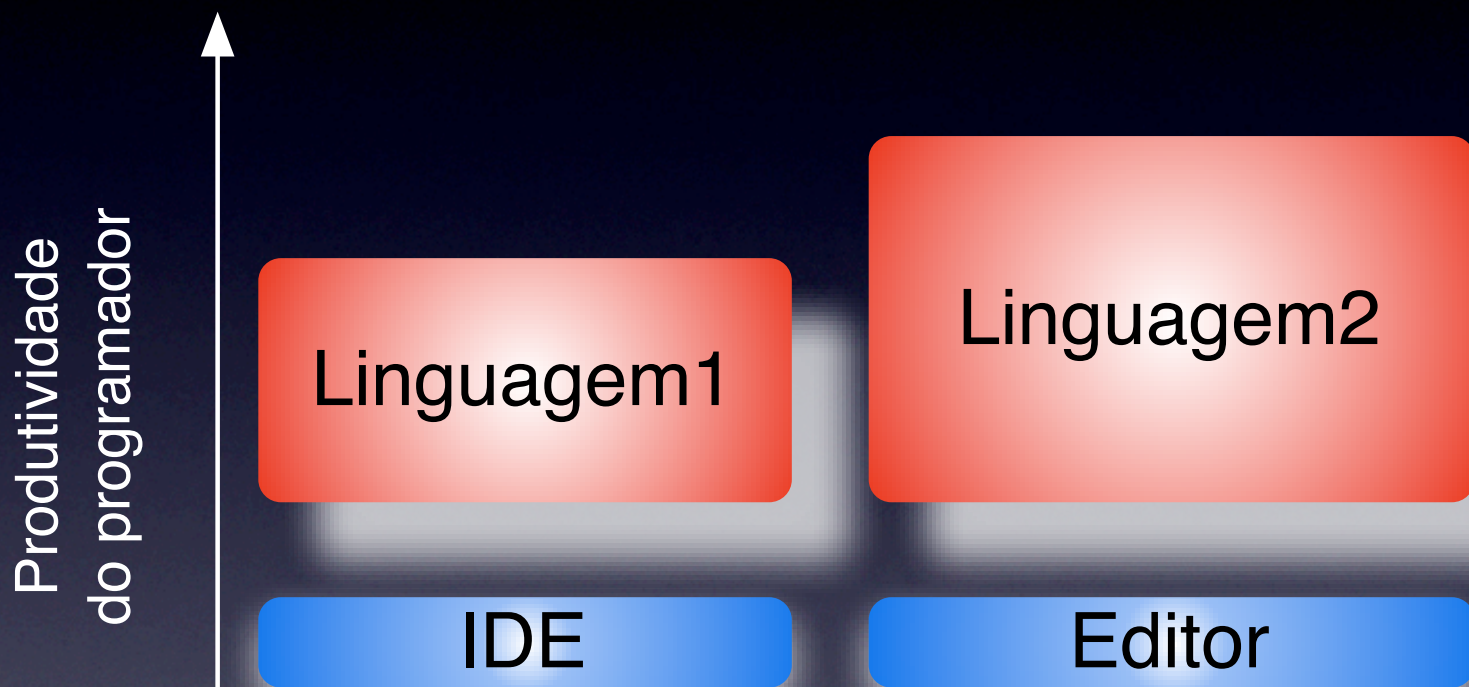
A visão dos IDEs (Integrated Development environment)

Aproximação à produtividade

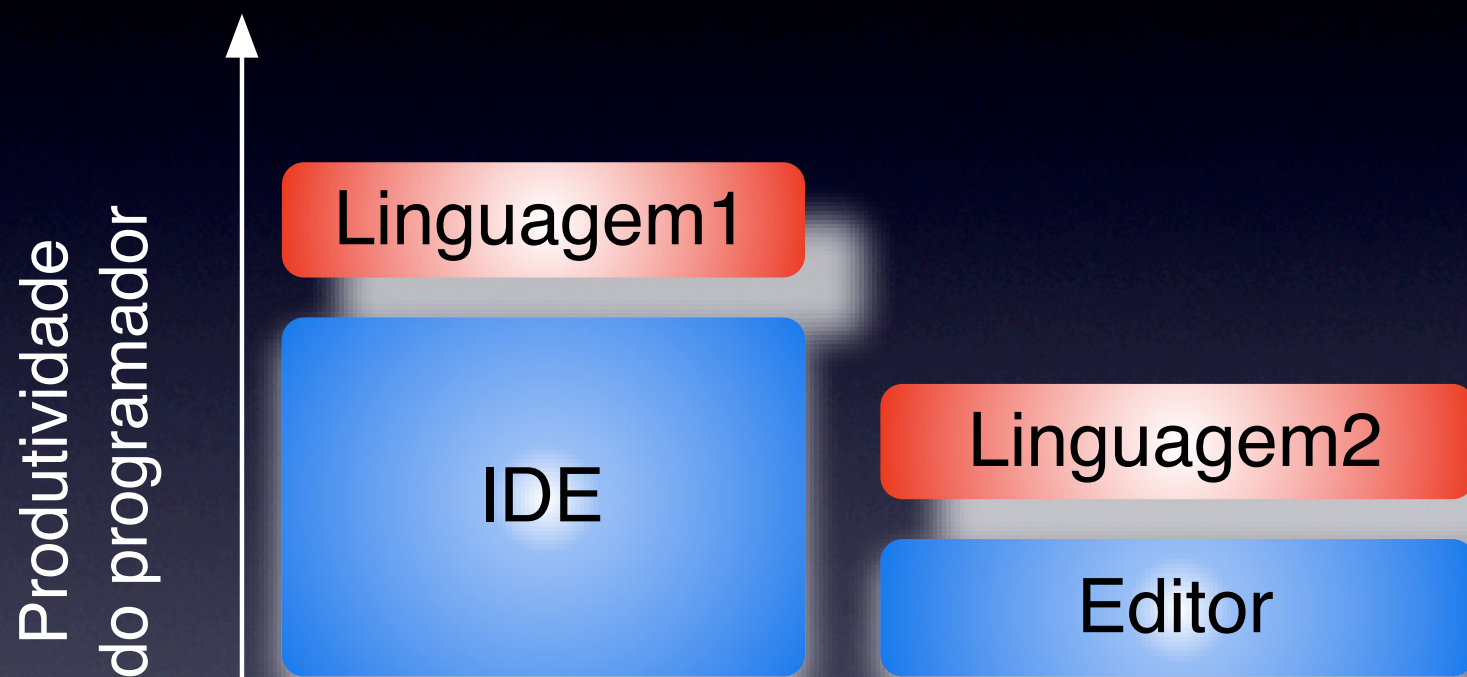


Fonte: <http://osteele.com/archives/2004/11/ides>

Quando uma linguagem nova aparece (ou uma versão nova de uma linguagem) o suporte da ferramentas a essa linguagem começa pelos editores e só mais tarde começam a aparecer ferramentas (IDEs) realmente sofisticados.



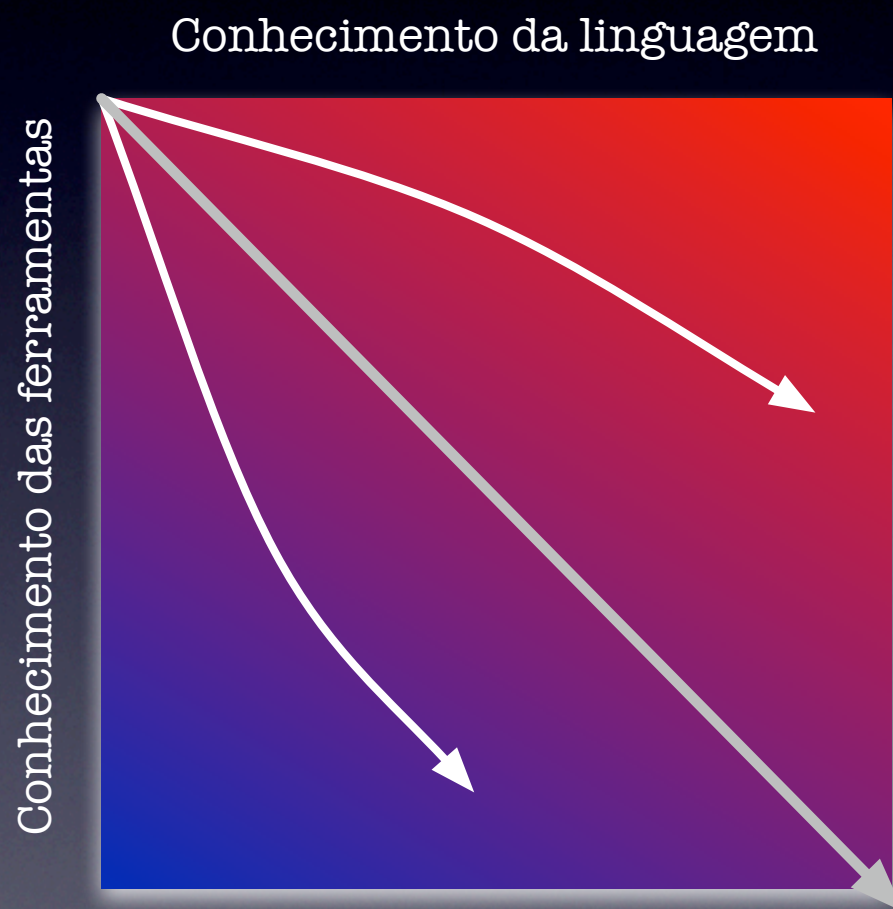
Quando se tem uma perspectiva mais focada nas linguagens as ferramentas acabam por fazer uma diferença menor. Na verdade é a linguagem que eu escolho para fazer o desenvolvimento que me vai tornar mais ou menos produtivo. As pessoas da comunidade Perl, têm tipicamente esta visão do mundo.



Quando se é uma pessoa mais focada nas ferramentas a minha produtividade é mais alterada pelo tipo, sofisticação e conhecimento que tenho da ferramenta em causa. A linguagem torna-se menos importante, e foca-mo-nos mais no conhecimento profundo da ferramentas.

A comunidade .Net e Visual Studio, tem tipicamente mais esta visão do mundo.

Ao fim de algum tempo chega-se ao mesmo nível de produtividade mas o percurso é diferente



Na verdade em termos de produtividade poderá chegar-se (ou não) ao mesmo nível. O percurso é que é diferente.

Penso que a longo médio prazo fará mais sentido um conhecimento da linguagem. Contudo algum conhecimento das ferramentas também é importante, pelo que é nisso que foco o resto da minha apresentação.

Porque não usar um IDE

(Bem é o Randal que diz :-)

"If you can't program careful enough to not need a debugger, then either slow down your rate of coding, or pick a different profession. Please."

Randal L. Schwartz

Como escolher um bom editor

- Ver o que é que o craques usam e porquê?
- Se trabalho em várias plataformas o meu editor deve ser suportado nessas várias plataformas
- Sinto-me confortável com o meu editor
- Ver se o meu editor faz o mínimo do que vamos apresentar de seguida

10

Ver o que os outros programadores que eu admiro usam, ver se me podem ajudar a escolher um editor caso ainda o não tenha feito. É bom discutir com os outros que editores usam e porquê (as razões da escolha são muito importantes, pois podem não ser as que se adaptam ao meu trabalho)

Quem usa: Textmate, Emacs, Vim, outro (qual?)?

Se uso várias plataformas, é boa ideia que o meu editor seja suportado nessas plataformas. É bom investir num editor que já está consolidado no mercado (não vou gastar tempo a aprender uma coisa que corre riscos de desaparecer).

Se gosto do editor que uso, então não vale a pena mudar.

Não vale a pena entrar numa discussão de qual é o melhor editor, pois o melhor editor para mim pode não ser o melhor para ti.

Por vezes pode ser necessário utilizar editores específicos para fazer tarefas específicas, por exemplo ver logs de 1 GB (por exemplo uso joe em unix e ultraedit em windows pois usam memory mapped files o que os torna mais rápidos a abrir ficheiros gigantes)

Randal Schwartz usa emacs e o
Damian Conway usa vim

"Given that I use the kitchen sink of programming languages (Perl), it only makes sense that I use the kitchen sink of editors (Emacs)."

Randal L. Schwartz

○ Larry não tenho ideia....

O problema

Editamos montes de texto:

- Source code
- Documentação
- email

Gastamos 40% do nosso tempo a editar. Será que conseguimos melhorar um bocadinho o desempenho e ao mesmo tempo tornar a tarefa um pouco mais agradável?
Editar significa interagir com texto, por vezes pode ser ler código.

3 passos para melhorar

- Detectar ineficiência
- Descobrir uma maneira mais rápida
- Tornar um hábito

13

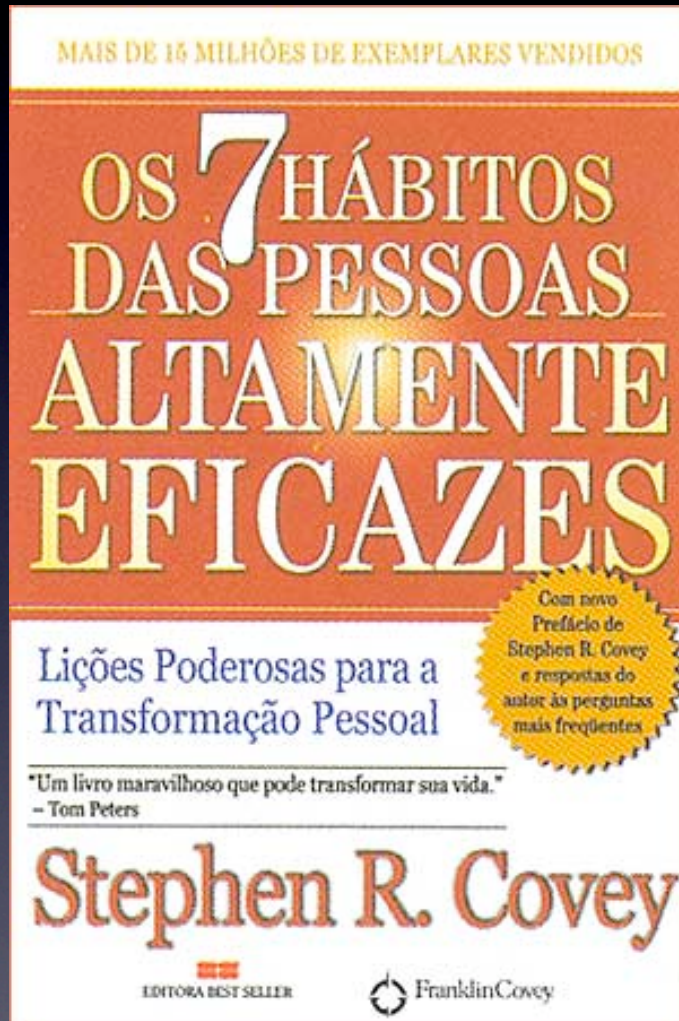
Perceber o que demoro a fazer, faço muitas vezes, e que se torna repetitivo.
Descobrir padrões no meu trabalho. Por exemplo na próxima hora de trabalho descobrir onde eu gasto o meu tempo, o que me atrapalha e atrasa?

Exemplo: Se ao escrever tenho dificuldade em encontrar sinónimos, ou correcções para o texto, então sou ineficiente.

Sumário:
1. Identificar os problemas
2. Descobrir as soluções
3. Utilizar

Parece simples. E é!

Porque os sete hábitos?



Para estar de acordo com o título da apresentação e do Livro em que a apresentação foi inspirada.

“Os 7 hábitos de pessoas altamente eficazes”.

O número 7 é o número da perfeição.

Os sete hábitos do Covey são:

Seja Proactivo

Comece com o fim em mente

Dê prioridade ao que é prioritário

Pense Ganhar-Ganhar

Procure primeiro compreender para depois ser compreendido

Crie Sinergias

Afine as suas ferramentas

Hábito I: Mover-se rapidamente

Passo I: Identificar a ineficiência

Quando procuramos uma variável num ficheiro
(ou directoria)

Utilizamos:

Ctrl-S

escrevemos a variável

Ctrl-S,

15

Se conseguirmos ser mais rápidos a encontrar o que procuramos, vamos ganhar muito tempo. Muitas vezes andamos à procura de coisas em ficheiros.

Por exemplo estamos a tentar descobrir onde se utiliza uma variável. É uma variável que é utilizada em muitos sítios.

No emacs fazemos Ctrl-s e escrevemos o nome da variável. Depois vamos fazendo Ctrl-S até encontramos o que pretendemos. Isso pode demorar um bocado. Há várias maneiras de acelerar o processo.

Hábito 1: Mover-se rapidamente

Passo 2: Descobrir uma maneira mais rápida

```
require Exporter;

our @ISA = qw(Exporter);
our @EXPORT = qw(valid_nif);

sub valid_nif {

    my $nif = shift or return undef;
    $nif =~ /\^d{9}$/ or return undef;

    my $control = chop $nif;
    my $sum;

    for (2..9) {
        $sum += $_ * chop $nif;
    }

    my $expected = 11 - $sum % 11;
```

- Ctrl-S, Ctrl-W e então Ctrl-S e ligar o highlighting.
- O comando list-matching-lines
- Folding

16

Quando estamos no modo de busca o Ctrl-W permite ir copiando para a zona onde se descreve a busca aquilo que está no buffer que estamos a editar (a partir do ponto onde está o cursor). Além disso a busca no emacs é incremental para no primeiro ponto onde encontrou qualquer coisa, mas se acrescentar letras, pode saltar para a frente. O facto de fazer highlighting a expressão que se está a procurar também ajuda a ver outros pontos onde está o que pretendemos encontrar. A possibilidade de procurar incrementalmente e utilizando expressões regulares (search-regexp) também é muito interessante (pode ser acedido por Esc-Ctrl-S).

O equivalente em vim é o * e o hlsearch

Uma ideia que o Bram dá, é o da utilização de folding o emacs suporta isso com o modo outline, mas eu não gosto particularmente (provavelmente é uma questão de hábito).

O comando list-matching-lines faz uma espécie de grep sobre o ficheiro onde estamos. Também é muito útil.

O comando list-matching-lines (assim como muitas coisas no emacs) usam expressões regulares que não sendo exactamente iguais às dos perl, são muito parecidas.

Hábito 1: Mover-se rapidamente

Passo 3: Tornar um hábito

Alterar (criar) .emacs:

Garantir que o highlighting está ligado

Associar uma tecla aos comandos mais utilizados (global-set-key)

Activar o modo de outline (caso se goste de utilizar o folding)

O equivalente em vim é o * e o hlsearch

Colocar uma global-set-key para o list-matching-lines pode ser muito interessante, pois é algo que se pode utilizar com muita regularidade.

Hábito 2: Não escrever duas vezes

Passo 1: Identificar a ineficiência

É difícil escrever :

`XmpCreatePixmapFromData()`

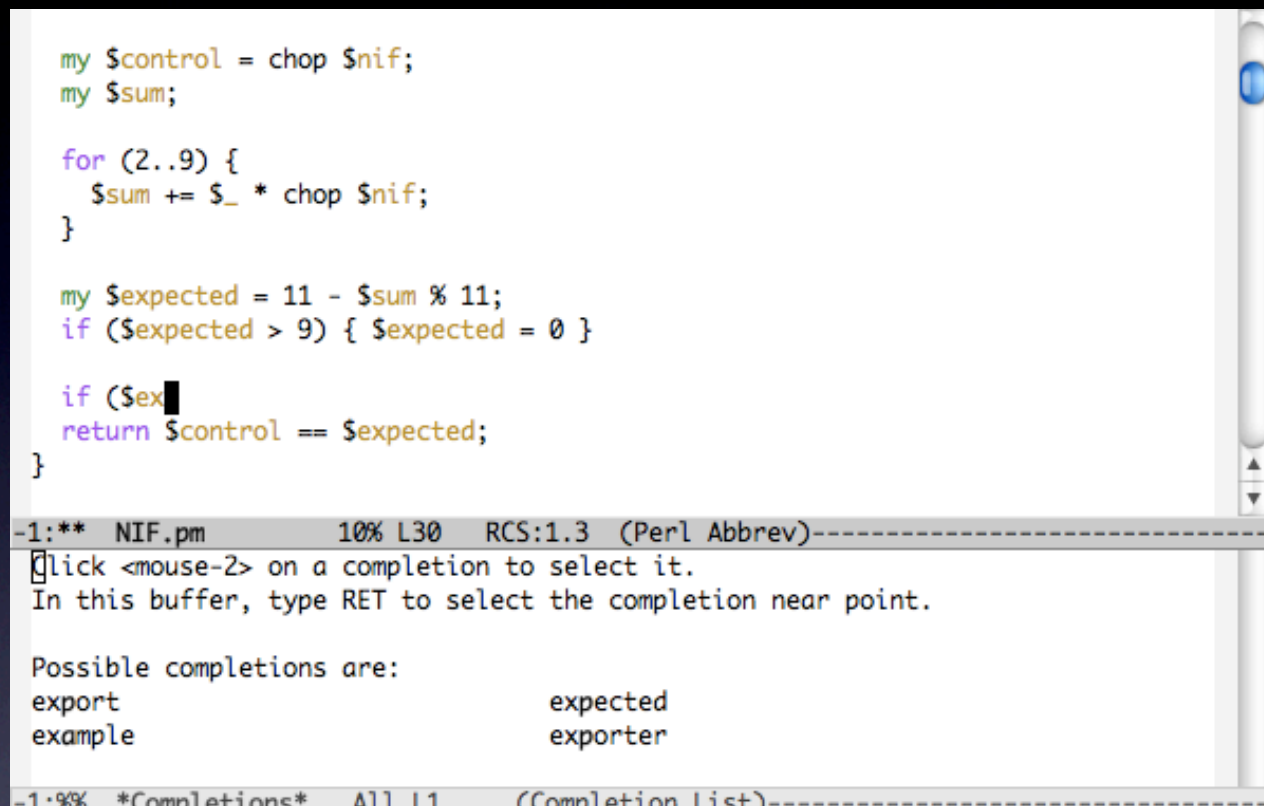
ou

`john@alfarrabio.di.uminho.pt`

e muitas vezes escrevemos mal

Hábito 2: Não escrever duas vezes

Passo 2: Descobrir uma maneira mais rápida



The screenshot shows an Emacs editor window with a Perl script. The script defines a function that calculates a value based on a loop and a modulo operation. The cursor is positioned at the end of the line `if ($ex`. Below the code, a completion list is displayed, showing possible completions for the current line. The list includes `export`, `example`, `expected`, and `exporter`. The status bar at the bottom indicates the current buffer is `NIF.pm` and the completion list is active.

```
my $control = chop $nif;
my $sum;

for (2..9) {
    $sum += $_ * chop $nif;
}

my $expected = 11 - $sum % 11;
if ($expected > 9) { $expected = 0 }

if ($ex
return $control == $expected;
}
```

-1:** NIF.pm 10% L30 RCS:1.3 (Perl Abbrev)-----
Click <mouse-2> on a completion to select it.
In this buffer, type RET to select the completion near point.

Possible completions are:
export expected
example exporter
-1:8% *Completions* All 11 (Completion List)-----

- O abbrev-mode (minor mode) pode utilizar-se também o dynamic abbrev
- Utilizar algum modo de code-completion por exemplo o SEPIA (Simple Emacs Perl Integration) ou o CEDET (Collection of Emacs Development Environment Tools)

Toda a gente usa o tab na shell, porque é que não se pode utilizar nos editores (há o tab nos editores pode ser utilizado para outras coisas). Mas será que não podemos utilizar um método equivalente?

Sim o abbreviation-mode ou abbreviation-minor-mode permite utilizar abreviaturas para palavras muito grandes.

O emacs também tem um modo que é o dynamic-abbreviation, em que se utiliza o buffer que se está a editar para tentar adivinhar como expandir a abreviatura.

esc-tab
esc-tab/

Hábito 2: Não escrever duas vezes

Passo 3: Tornar um hábito

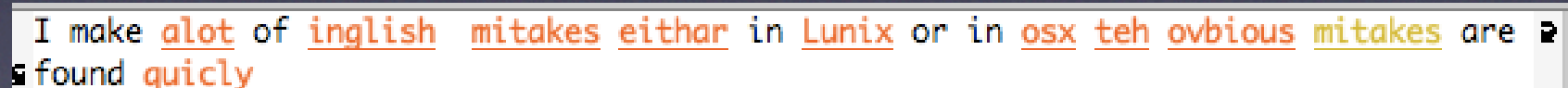
- gravar as abbreviations mais comuns
- afinar qual o tamanho ideal para abreviatura

Hábito 3: Corrigir o que está mal

Passo 1: Identificar a ineficiência

Escrevo Inglês/Português/Perl com muitos erros

Passo 2: Descobrir uma maneira mais rápida Utilizar o ispell(aspell) flyspeel-mode



I make alot of inglish mitakes eithar in Lunix or in osx teh ovbious mitakes are
found quicly

Hábito 3: Corrigir o que está mal

Passo 3: Tornar um hábito

- Actualizar o dicionário
- Incluir palavras novas no dicionário
- Ter vários dicionários e dialectos

Hábito 4: Um ficheiro nunca vem sozinho

Passo 1: Identificar a ineficiência

Quando “herdamos” código é muito difícil entendermo-nos no emaranhado de código (e ficheiros)

Passo 2: Descobrir uma maneira mais rápida

tags (etags)

grep

manter o desktop

23

Os tags files permitem ao editor descobrir em que ficheiro e linha está definida uma função (variável).

Tem que se ler o ficheiro de tags no emacs utilizando o comando visit-tags-file.

É bom porque o tags conhece uma data de linguagens:

ada , asm, c, c++, c*, cobol, erlang, fortran, java, lisp, makefile, pascal, perl, postscript, proc, prolog, python, scheme, tex, texinfo, yacc.

Atenção que para o vim utiliza-se o ctags.

Quem é que já tinha ouvido falar em tags?

Hábito 5: Vamos trabalhar em conjunto

Passo 1: Identificar a ineficiência

- Utilizo editores em montes de programas, mas não consigo que esses programas utilizem o meu editor

Passo 2: Descobrir uma maneira mais rápida

- É difícil depende muito do programa (nem sempre é possível ou “limpinho”)
- Por exemplo é possível utilizar o emacs como editor da MailApp mas não é fácil

Hábito 5: Vamos trabalhar em conjunto

Passo 2: Descobrir uma maneira mais rápida

Mas o emacs fala com:

- sistemas de controlo de versões
- compilador
- debugger

Hábito 6: Há muitos padrões que repetimos

Passo 1: Identificar a ineficiência

- Os headers das funções são sempre iguais
- A documentação tem sempre a mesma estrutura

Passo 2: Descobrir uma maneira mais rápida

- Utilizar skeletons com os padrões típicos
- Configurar os skeletons que interessam

Hábito 6: Há muitos padrões que repetimos

Passo 3:Tornar um hábito

- Configurar skeletons por linguagem

Hábito 7: Afiar as ferramentas

- Regularmente fazer uma avaliação das ferramentas que estamos a utilizar (e como as estamos a utilizar)
- Utilizar feedback - Aprender a partir dos problemas que encontrei (e que os outros encontraram)