Theory of Science & Free Thinking

Ole Peter Smith, IME, UFG, ole@mat.ufg.br

EMSL '10
15-16/10/2010
Uberlândia-GO

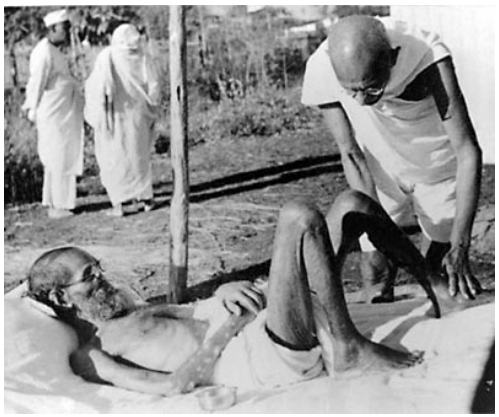
The Road to Knowledge? Simple:
Err, and Err, and Err again
But Less, and Less, and Less

Piet Hein

The Father of Nonviolent Resistance: Mahatma Gandhi

I really do like Your Christ - but dislike Your Christians
Why can'tYyour Christians be more like Your Christ?

LORD WILLINGDON'S DILEMMA



Live as if you'll die tomorrow
Learn as if you'll live forever

Me & IT

- 1980: Draw Bicycle Whell
  Amiga/Basic
- 1990: Turbo Pascal 5.0
  Conjugating Spanish Verbs...
- 1992: PhD in Optimization
  Auto LISP/CAD, C/C++, Shells
- 1996: Split, join (regex) in C/C++...
  $\rightsquigarrow$ Failure
- 1996: Systems Administrator, MAT, DTU
- 1996, Perl: System - Web - Tk
- 2002: Brasil
- 2004: SAdE, Sistema Adm. Escolar » No DBs!
- 2009: IME, UFG
- PHP

UFG

What does the Free in 'Free Software' Mean?

- No problem may be solved
  Using the same parameters that lead to it
  Einstein
- I have no Rules is a Rule...
- Axioms - Paradigmas - Dogmas
- Question, Always!
- Free Thinking:
  My Rights Starts Where Yours End
- Isonomy
- Tolerances
- Free Knowledge - Science

Using Perl we are Free to do:

- Linux user/group adm
- Quota adm
- Backup adm
- Tk Window App
- CGI programming
- Lowlevel networking
- Perl & Expect: Controle command-line programs
- ...
- Exercise 1:
  Rename the php executable on your Linux & Restart
  What Happened?
- Exercise 2:
  Rename perl executable on your Linux & Restart
  What Happened?

Perl: use strict

Perl

```
use strict;

my $var=0;
print $ver;
```

Compile time error!

PHP

```
my $var=0;
print $ver;
```

'No' Problem

Liberty is to be strict, if and when we want...

`perl -w`: Stricter!
Perl also used for critical operations
Taint

Variables

Type	Perl	PHP
Scalar	$	$
List	@	$
Hash	%	$

```
my $scalar="Oi";
my @list=();
$list[3]=$scalar;
my %list=();
$list{ "key" }=$scalar;
my $list=\@list; #Ref to list
print $list->[3];
my $list=\%list; #Ref to hash
print $list->{ "key" };
```

Functions

- One Argument

```
sub MyFunction
{
    my $arg1=$_;
}
```

- List of Arguments

```
sub MyFunction
{
    my @args=@_;
}
```

Lists

- ```
  my @list1=("Ole","Fernando");
  my @list2=qw/Ole Peter Smith/;
  my @list=grep { /e$/ } @list2;
  push(@list,"da","Silva",@list2);
  ```


  ```

Return two Values: Perl

```
sub MyFunc
{
    ...
    return ($res1,$res2);
}

sub MyOtherFunc
{
    ...
    my ($res1,$res2)=MyFunc();
}
```

Return two Values: PHP

```
sub MyFunc
{
   ...
   return array($res1,$res2);
}

sub MyOtherFunc
{
   ...
   $list=MyFunc();

   $res1=$list[0];
   $res2=$list[1];
}
```

Named Arguments

```
sub MyFunc
{
   my %args=@_;
   if ($args{ "BoldFirst" }) ...
   unless ($args{ "TableHead" }) ...
}
sub MyOtherFunc
{
   ...
   MyFunc
   (
      "BoldFirst" => 1,
      "TableHead" => 0,
   );
}
```