# The right language when you really needed it
(Integration between C language and high-level languages)

- Software evaluation parameters
- Performance vs productivity
- Productivity vs quality
- Hardware compatibility vs performance

# The right language when you really needed it
(Integration between C language and high-level languages)

## Software evaluation parameters

### Productivity
Illusions and smog-sellers

### Performance
Wasteland

### Hardware requirements
TCO

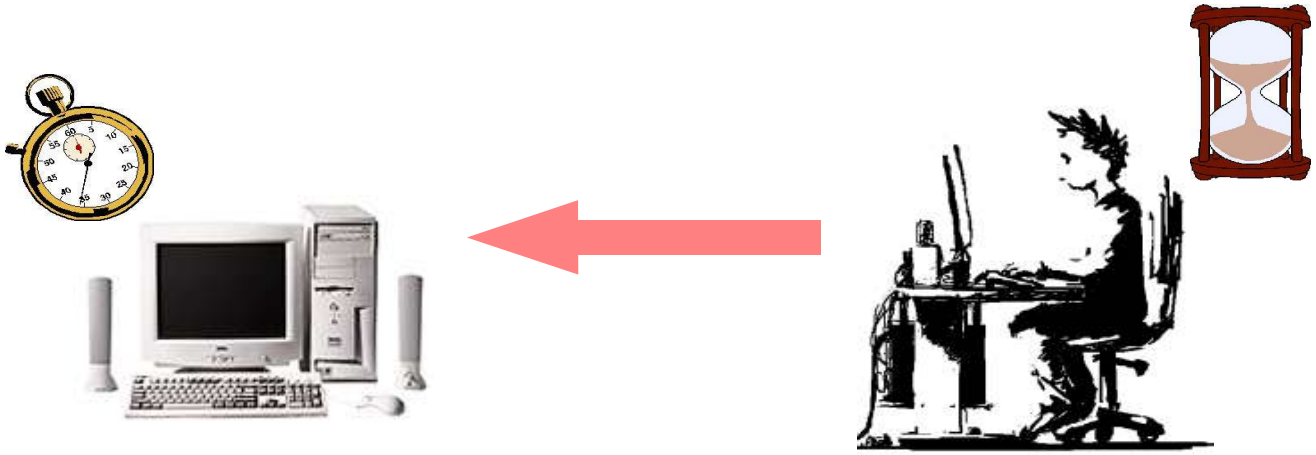
### Hardaware compatibility
Hardware Abstract Layer
System Libraries and language libraries

### Quality
Documentation taboo
The fast run to chaos

### Control process degree
The possibility to have fast reactions

# The right language when you relaly needed it
(Integration between C language and high-level languages)

Performance vs productivity



Power only where needed

# The right language when you really needed it
(Integration between C language and high-level languages)

## Productivity vs quality

How can I obtain more productivity?

Using obscure tools to improve productivity deprives the possibility to provide a complete support

# The right language when you really needed it
(Integration between C language and high-level languages)

HW compatibility vs performance

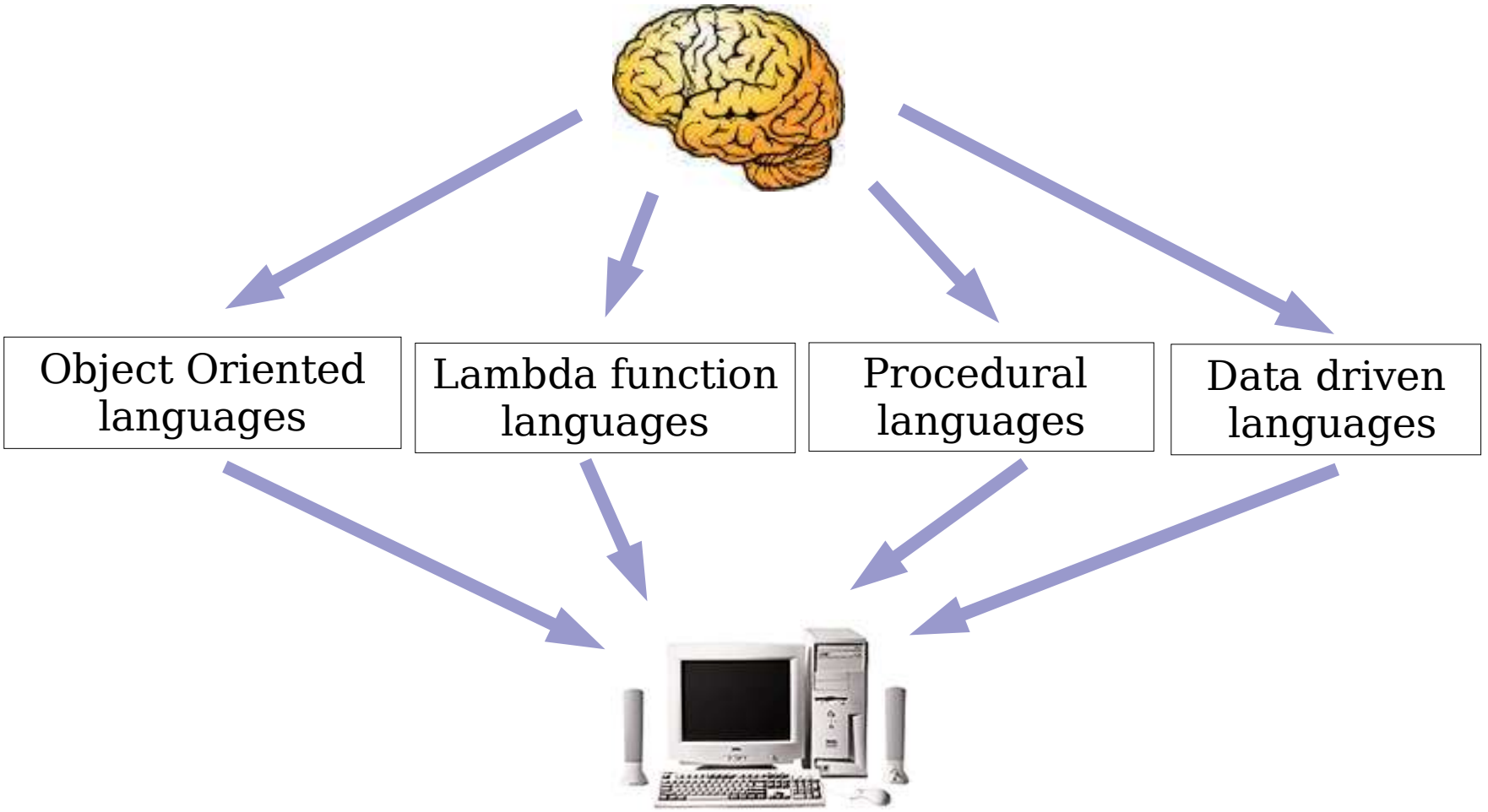
Why should I use system libraies?
Is the software integrated with the system? (WxWidget)

# The right language when you really needed it
(Integration between C language and high-level languages)

- Different languages for different philosophies
- Tcl/Tk
- Perl
- Perl/XS/C vs Tcl/C

# The right language when you really needed it
(Integration between C language and high-level languages)

Different languages for different philosophies



# The right language when you really needed it
(Integration between C language and high-level languages)

## Tcl/Tk

Introduction to Tcl/Tk:
- Tcl programming examples:
- Multyplatform structure (file join/file split ...)
- Safe slave enviroments (interp)
- Basic Tk overview

Tcl weakness:
- Coherency (list commands)
- Multitasking is performed just by thread

Tcl strengths:
- Possibility to enhance the Tcl structures using Tcl
- Tcl is easy and powerful
- Tcl is available for many different platforms

Tcl and C:
- Tcl_CreateCommand

# The right language when you really needed it
(Integration between C language and high-level languages)

## Tcl and C

```
#include <tcl.h>
#include <stdio.h>

int cProc (ClientData cd, Tcl_Interp *interp, int argc, CONST char* argv[]) {
        //
        // Questa funzione esegue un addizione fra "a" e "b"
        //
        int out=0, a=0, b=0, e=0;
        char outStr[20];

        if (argc != 3) {
                interp->result = "Error! Usage myAdd <int> <int>";
                e=1;
        } else {
                if (Tcl_GetInt (interp, argv[1], &a) == TCL_OK && Tcl_GetInt (interp, argv[2], &b) == TCL_OK) {
                        out=a+b;
                } else {
                        interp->result = "Error!The parameters must be integer numbers";
                        e=1;
                }
        }

        if (e == 1) {
                return TCL_ERROR;
        } else {
                sprintf (outStr, "%d", out);
                Tcl_SetResult (interp, outStr, TCL_VOLATILE);
                return TCL_OK;
        }
}


int Myadd_Init (Tcl_Interp *interp) {
        //
        // Questa funzione registra la procedura scritta in C
        //
        Tcl_CreateCommand (interp, "myAdd", cProc, (ClientData)NULL, (Tcl_CmdDeleteProc *)NULL);
        if (Tcl_PkgProvide (interp, "myAddPkg", "1.0") != TCL_OK) {
                interp->result = "Error while I am registering the library";
                return TCL_ERROR;
        }
        return TCL_OK;
}
```

Tcl library

Check for parameters type

C code

Tcl function name

my C function

interface

destroyer

Tcl pckg name

# The right language when you really needed it
(Integration between C language and high-level languages)

## Perl

Introduction to Perl
- Perl programming examples

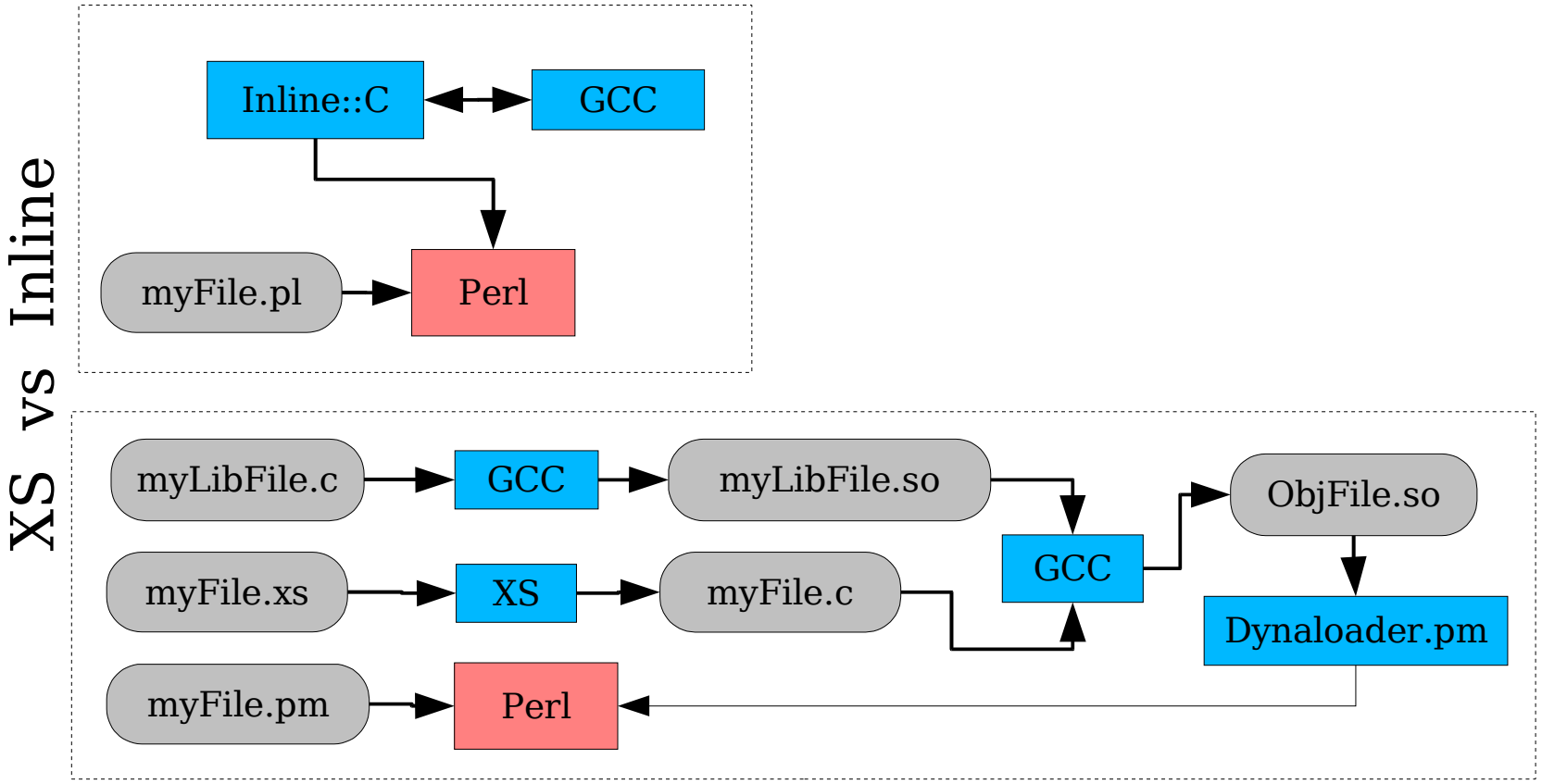
Perl weakness:
- The possibility to write a unreadable code
- The trust contract with the user

Tcl strengths:
- Coherency (oop)
- A lot of existent libraries

Perl and C:
- Inline::C
- XS

# The right language when you really needed it
(Integration between C language and high-level languages)

## Perl and C (XS vs Inline)

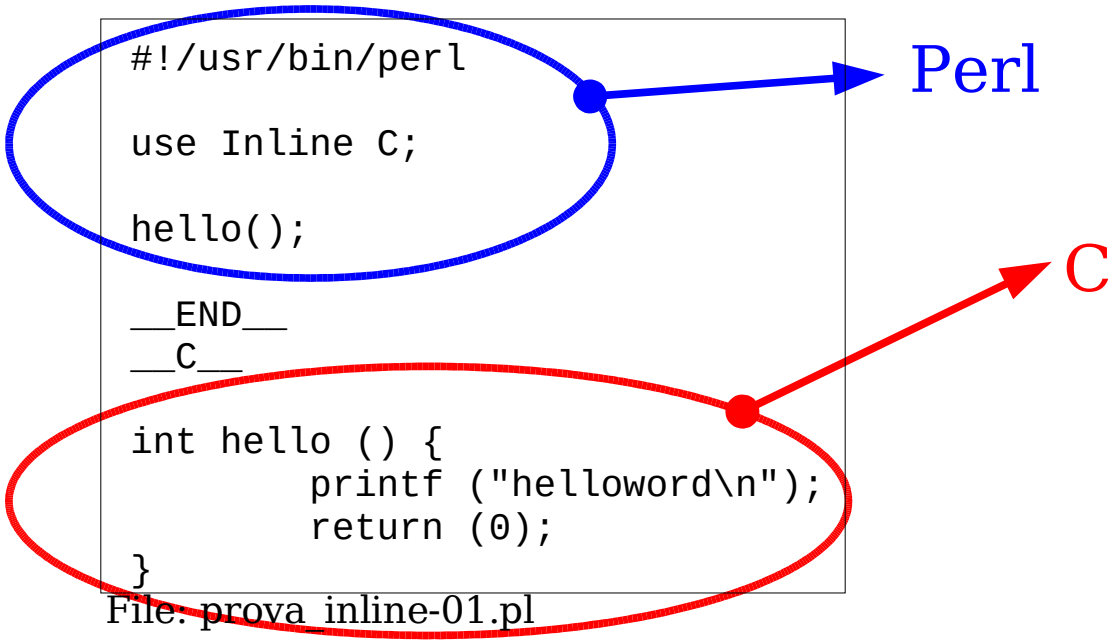


# The right language when you really needed it
(Integration between C language and high-level languages)

## Perl Inline::C

```
#!/usr/bin/perl

use Inline C;

hello();

__END__
__C__

int hello () {
        printf ("helloword\n");
        return (0);
}
```

File: prova_inline-01.pl



# The right language when you really needed it
(Integration between C language and high-level languages)

## XS (h2xs standard model)

```
[]$ h2xs -n example -A
[]$ ll example

-rw-r--r--  1 warlock    149 2006-09-29 15:13 Changes
-rw-r--r--  1 warlock     83 2006-09-29 15:13 MANIFEST
-rw-r--r--  1 warlock    832 2006-09-29 15:13 Makefile.PL
-rw-r--r--  1 warlock   1166 2006-09-29 15:13 README
-rw-r--r--  1 warlock    218 2006-09-29 15:30 example.xs
drwxr-xr-x  2 warlock   4096 2006-09-29 15:23 lib/
-rw-r--r--  1 warlock 117049 2006-09-29 15:13 ppport.h
drwxr-xr-x  2 warlock   4096 2006-09-29 15:21 t/
```

Created by h2xs files

```
[]$ cat example/example.xs

#include <stdio.h>
#include "EXTERN.h"
#include "perl.h"
#include "XSUB.h"

#include "ppport.h"


MODULE = example        PACKAGE = example

int
helloword ()
    CODE:
        printf ("Helloword\n");
        RETVAL=0;
    OUTPUT:
        RETVAL
```

Our XS file

```
[]# ln -l <example path>/example/blib/arch/../example.so <perl lib path>/.
[]$ perl -I<example path>/example/lib -e 'use example; &example::helloword()'
helloword
```

# The right language when you really needed it
(Integration between C language and high-level languages)

Perl/XS/C vs Tcl/C

# The right language when you really needed it
(the package manager)

- What should a software package manager do?
- Do you know the installation target?
- Wpkg Framework
- Wpkg package architecture
- Examples

# The right language when you really needed it
(the package manager)

What should a software package manager do?

## Installing
- Checking for dependences
- Installing files in their right positions
- Update the system (es: manpage indexes, ldconfig...)

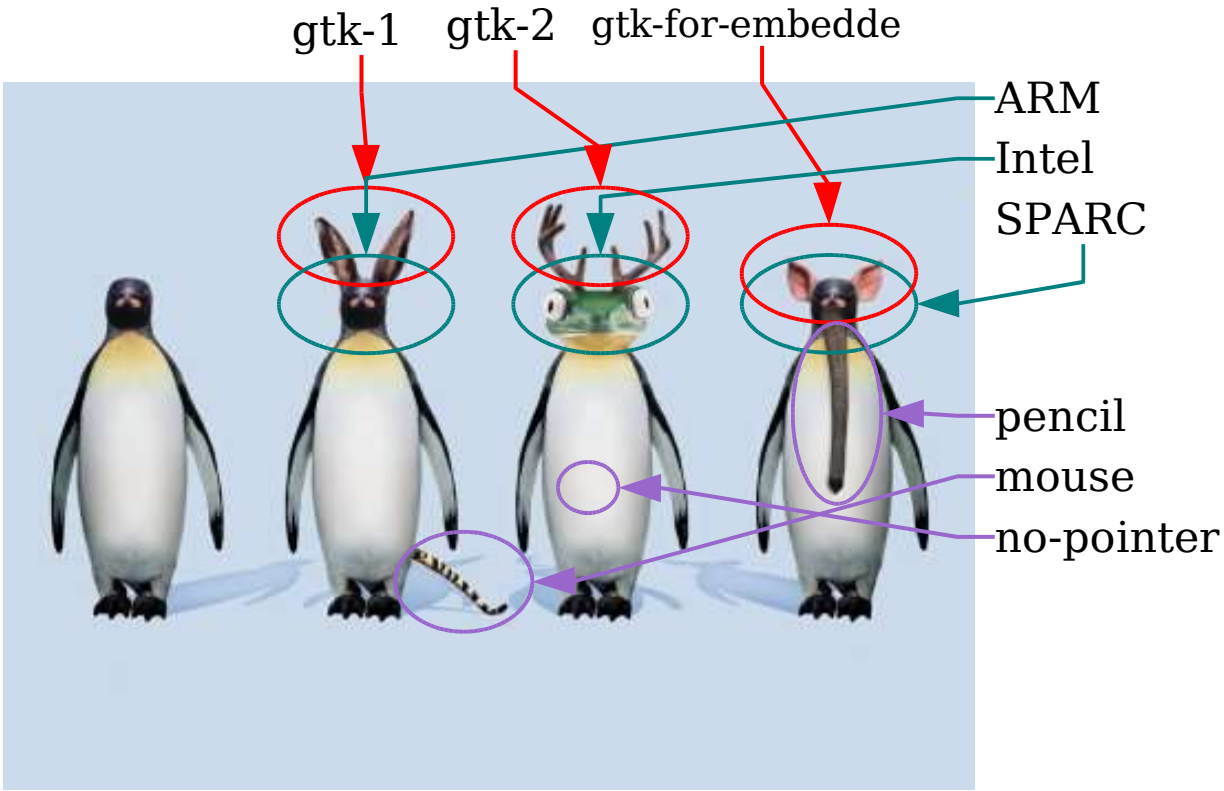
## Uninstalling
- Checking for broken dependence
- Removing the previous installed files
- Update the system

## Quering
- What about the package?
- Which file by the pachage?
- Which package have originally instelled the file?
- Which softwares I have installed?

## Verifing
- Installed files status

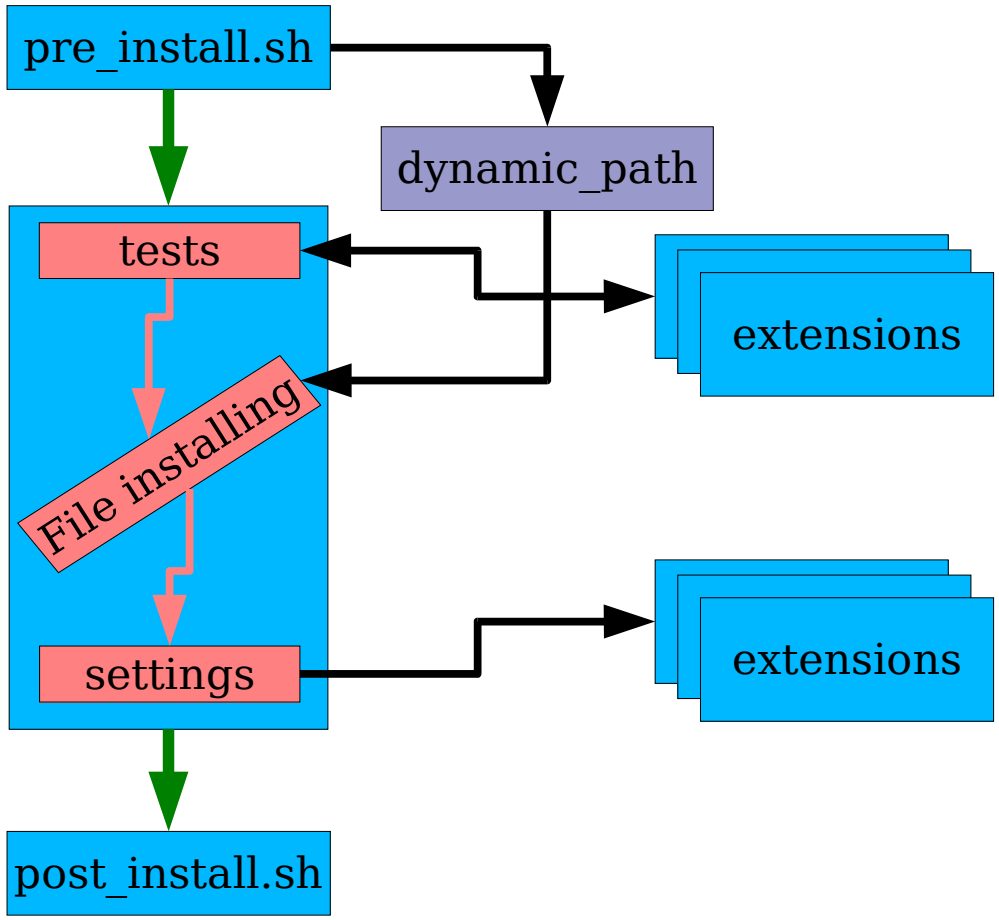
# The right language when you really needed it
(the package manager)

Do you know the installation target?



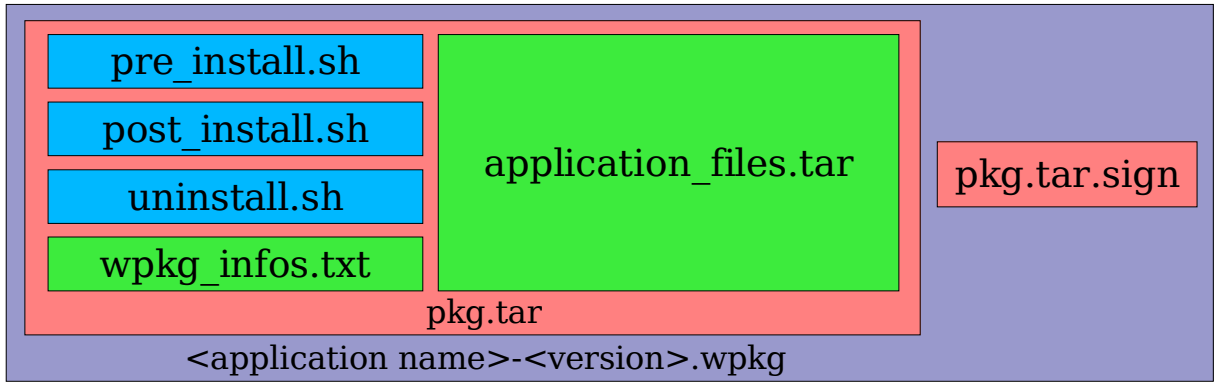
# The right language when you really needed it
(the package manager)

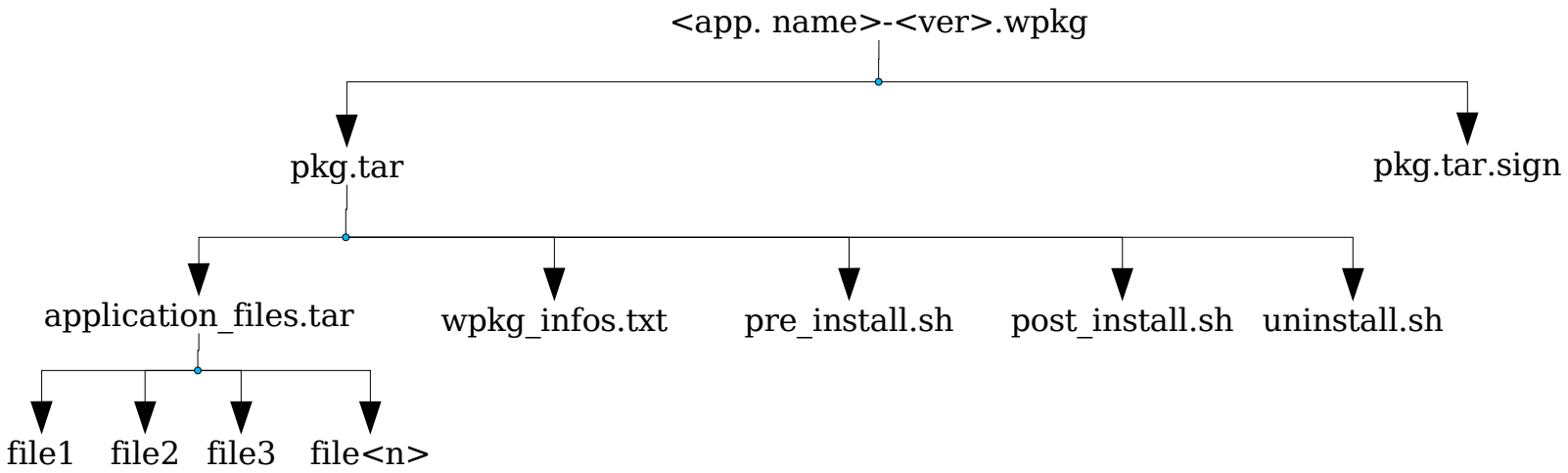
## Wpkg Framework



# The right language when you really needed it
(the package manager)

## Wpkg package architecture



# Licence



Silvano Catinella
catinella@yahoo.com
+39 348 5631681