
PROGRAM NAME: **deploy.pl**

AUTHOR: **Josep F. Abril** jabril@imim.es

LICENSE: **GNU General Public License (GNU-GPL)**

LAST UPDATE: **October 5, 2001**

DESCRIPTION: This perl script creates all the files we need to start a project report or to describe a program implementation under the NOWEB Literate Programming tool. It will produce the main NOWEB file, from which we can tangle the sources or weave the L^AT_EX documentation.

Genome Informatics Research Lab

Grup de Recerca en Infomàtica Biomèdica
Institut Municipal d'Investigació Mèdica
Universitat Pompeu Fabra

Contents

1	Introduction	1
1.1	Description	1
1.2	To Do	1
2	Implementation	2
2.1	Program outline	2
3	Template definitions	5
3.1	Perl scripts	5
3.2	Perl packages	6
3.3	Reports	10
4	Appendixes	17
A	empty appendix section	23
A.1	empty appendix subsection	23
B	Common code blocks	24
B.1	PERL scripts	24
B.1.1	Timing our scripts	24
B.1.2	Printing complex Data Structures	24
B.1.3	Common functions	25
B.1.4	Common functions for reporting program processes	25
B.2	AWK scripts	26
B.3	BASH scripts	26
B.4	Version control tags	26
B.5	GNU General Public License	26
C	Extracting code blocks from this document	27
C.1	Extracts Script code chunks from the NOWEB file	27
C.2	Extracting different Config Files	27
C.3	Extracting documentation and L ^A T _E X'ing it	27
C.4	Defining working shell variables for the current project	28

List of Tables

List of Figures

1	Flow diagram for deploy.pl.	1
---	-------------------------------------	---

1 Introduction

TO DO

- Flow diagram (Serial/Filter mode) for figure 1.



Figure 1: Flow diagram for deploy.pl showing interactions of external processes, output files and documents.

1.1 Description

1.2 To Do

- ▷ [Section 1, page 1]
Flow diagram (Serial/Filter mode) for figure 1.
- ▷ [Section 2.1, page 3]
Using GetOPTS. New options:
 - r -rcs : project is under RCS, so checkin.
 - c -cvs : project is under RCS, so checkin if logged in.
 - p -perl: add .pl to project name (and PERL shebang).
 - g -gawk: add .awk to project name (and GAWK shebang).
 - b -bash: add .sh to project name (and BASH shebang).
- ▷ [Section 2.1, page 4]
A switch for those working with RCS/CVS or none of both.
- ▷ [Section 3, page 5]
Template for C programs (including Header/Module/Makefile sections).
- ▷ [Section 3, page 5]
Re-structure templates section.
- ▷ [Section 3, page 5]
Finish templates to be used by PDFflatex and LaTeX2HTML.

2 Implementation

```

2a  <Program Info 2a>≡
    my $PROGRAM = 'deploy.pl';
    my $VERSION = '1.0_alpha';

2b  <Program Description 2b>≡
    # #-----#
    # #                                deploy.pl                                #
    # #-----#
    #
    #      deploy.pl [options] project_name template_file
    #
    #      Preparing all files we need to work with noweb on a project.
    #      Creates basic file set to work with noweb literate programming tool.
    #
    #      Copyright (C) 2001 - Josep Francesc ABRIL FERRANDO

```

2.1 Program outline

```

2c  <DEPLOY 2c>≡
    <PERL shebang 24a>
    #
    # MODULES
    #
    <Use Modules 2d>
    #
    # VARIABLES
    #
    <Global Vars 2e>
    #
    # MAIN LOOP
    #
    <Main Loop 3a>
    #
    # FUNCTIONS
    #
    <Functions 3b>

2d  <Use Modules 2d>≡

2e  <Global Vars 2e>≡
    my $USAGE = "\nUSAGE:\n\tdeploy.pl <projectname> <template>\n".
                "(It assumes that you are in the right directory)\n\n";
    my @working_dirs = qw(
                                bin bin/param
                                data
                                docs docs/psfigures docs/tables docs/html
                                tests
                                );
    my ($PROJECT, $TEMPLATE);
    # my $HOME = $ENV{HOME};
    my $CWD = `pwd`;
    chomp($CWD);
    my $PATH = $CWD;
    # $PATH =~ s%^$HOME/%%o;

```

```

3a  <Main Loop 3a>≡
    &parse_args();

    print STDERR "###\n### RUNNING $PROGRAM.....\n###\n".
        "### User: $USER\n".
        "### Date: $DATE\n###\n".
        "### Current Working Directory: $PWD\n".
        "### Setting PATH to: $PATH\n".
        "### Project NAME: $PROJECT\n###\n";

    &make_dirs();
    &new_noweb_doc();
    &extract_files();

    print STDERR "###\n### RUNNING deploy.pl..... DONE\n###\n";

    exit(0);

```

```

3b  <Functions 3b>≡
    sub parse_args() {
        scalar(@ARGV) == 2 || do {
            print STDERR $USAGE;
            exit(1);
        };
        $PROJECT = shift @ARGV;
        $TEMPLATE = shift @ARGV;
    } #

```

TO DO

- Using GetOPTS. New options:
 - r -rcs : project is under RCS, so checkin.
 - c -cvs : project is under RCS, so checkin if logged in.
 - p -perl: add .pl to project name (and PERL shebang).
 - g -gawk: add .awk to project name (and GAWK shebang).
 - b -bash: add .sh to project name (and BASH shebang).

```

3c  <Functions 3b>+≡
    sub make_dirs() {
        print STDERR "###\n### Creating Project Subdirectories...\n###\n";
        foreach my $d (@working_dirs) {
            print STDERR "### ... $d\n";
            system("mkdir $d") unless (-e $d && -d _);
        };
        print STDERR "###\n### Project Subdirectories..... DONE\n###\n";
    } # make_dirs

```

```

3d  <Functions 3b>+≡
    sub new_noweb_doc() {
        my $file = "$PROJECT.nw";
        (-e $file && -f _) && do {
            print STDERR "###\n### Project file \"$file\" does exist...\n".
                "### EXITING PROGRAM !!!\n";
            exit(1);
        };
        print STDERR "###\n### Writing Project NOWEB file: $file\n###\n";
        open(NOWEB, "> $file");
        open(DATA, "< $TEMPLATE") ||

```

```

    die ("#### ERROR #### Template File does not exists: $TEMPLATE . $!\n");
(my $FIXPROJ = $PROJECT) =~ s{([\W_])}{\\$1}og;
while (<DATA>) {
    my ($FINDPATH,$FINDPROJECT,$FINDQUOTED) =
        ('@@@PATH@@@','@@@PROJECT@@@','@@@QUOTED@@@');
    my $l = $_;
    CURRENT: {
        $l =~ /$FINDPATH/o && do {
            $l =~ s/$FINDPATH/$PATH/o;
            last CURRENT;
        };
        $l =~ /$FINDPROJECT/o && do {
            $l =~ s/$FINDPROJECT/$PROJECT/o;
            last CURRENT;
        };
        $l =~ /$FINDQUOTED/o && do {
            $l =~ s/$FINDQUOTED/$FIXPROJ/o;
        };
    }; # CURRENT
    print NOWEB $l;
};
close(DATA);
close(NOWEB);
print STDERR "###\n### NOWEB file..... DONE\n###\n";
} # new_noweb_doc

```

```

4  <Functions 3b>+≡
    sub extract_files() {
        print STDERR "###\n### Extracting Files from NOWEB file...\n###\n";
        # my $WORK = '$HOME/.'.$PATH;
        my $WORK = $PATH;
        my $nwfile = "$PROJECT.nw";
        system « "+++EOS+++";
        notangle -R\'BASH Environment Variables\' $WORK/$nwfile > $WORK/.bash_VARS ;
        notangle -Rweaving $WORK/$nwfile > $WORK/nw2tex ;
        notangle -RLaTeXing $WORK/$nwfile > $WORK/ltx ;
        chmod a+x $WORK/nw2tex ;
        chmod a+x $WORK/ltx ;
        # ci -l -i0.1 -t-\'\\t\\t$nwfile: NOWEB file for $PROJECT\' \\
        # -m\'BASIC TEMPLATE for THIS PROJECT\' $nwfile ;
        emacs $nwfile \&
        $WORK/nw2tex ;
        $WORK/ltx ;
        /usr/X11R6/bin/ghostview -color -title -magstep -1 \\
                                -portrait -a4 $WORK/docs/$PROJECT.ps \&
        +++EOS+++
        print STDERR "###\n### File Extraction..... DONE\n###\n";
    } # extract_files

```

TO DO

- A switch for those working with RCS/CVS or none of both.

3 Template definitions

TO DO

- Template for C programs (including Header/Module/Makefile sections).
- Re-structure templates section.
- Finish templates to be used by PDFLatex and LaTeX2HTML.

3.1 Perl scripts

This template may contain one or more scripts, but it was thought to describe a single program implementation in perl language (it may require few adjustments to work with other languages because NOWEB is not language dependent).

```

5a  <PROJECT SCRIPT Template 5a>≡
    <EMACS header 10b>
    <LaTeX header 10c>
    <LaTeX packages 10d>
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
    \begin{document}
    %
    <LaTeX basic definitions 11>
    <PROGRAM LaTeX title 5b>
    %
    <LaTeX Frontmatter 13b>
    <LaTeX SCRIPT Mainmatter 14b>
    <LaTeX Backmatter 16b>
    %
    <LaTeX common code appendix - perl 17>
    <LaTeX common code appendix - bash 19b>
    <LaTeX common code appendix - version 20a>
    <LaTeX common code appendix - license 20b>
    <LaTeX common code appendix - noweb 20c>
    %
    \end{document}
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

5b  <PROGRAM LaTeX title 5b>≡
    \thispagestyle{empty}

    \begin{titlepage}

    \ \vfill
    \begin{center}
    \begin{bfseries}
    \begin{large}
    \newlength{\lttbl}\setlength{\lttbl}{0.25\linewidth}
    \newlength{\rttbl}\setlength{\rttbl}{0.70\linewidth}
    %\fbox{
    %\vskip 2ex
    \begin{tabular}{>\scshape}r@{\quad}l}
    \rule{\lttbl}{0pt} & \rule{\rttbl}{0pt} \\[2ex]
    \multicolumn{2}{c}{\shortstack{\rule[0ex]{0.95\linewidth}{2pt}\\[0ex]
    \rule[1ex]{0.95\linewidth}{2pt}}}\[2ex]
    Program Name: & {\Huge\progname} \\[3ex]
  
```

```

\multicolumn{2}{c}{\rule[0.5ex]{0.95\linewidth}{2pt}}\\[2ex]
  Author: & {\Large
            \begin{minipage}[t]{0.95\rttbl}
            \authorslist
            \end{minipage}}
            \|[2ex]
  License: & {\license}
            \|[2ex]
  Last Update: & {\today}
            \|[2ex]
  Description: & {\large\mdseries
                 \begin{minipage}[t]{0.95\rttbl}
                 \description
                 \end{minipage}}
                 \|[2ex]
\\
\multicolumn{2}{c}{\shortstack{\rule[0ex]{0.95\linewidth}{2pt}}\|[0ex]
                        \rule[1ex]{0.95\linewidth}{2pt}}\|[2ex]

\end{tabular}
}% % fbox
\end{large}
\end{bfseries}
\end{center}

\vfill

\begin{raggedleft}
\showaffiliation
\end{raggedleft}

\end{titlepage} %'

```

3.2 Perl packages

Perl packages (or also called modules) must rely on three basic files to be portable: the perl module itself ('your_script.pm'), the tester ('test.pl') and the makefile ('Makefile.PL'). There are three secondary files within the distribution:

```

6 <PROJECT PERL PACKAGE Template 6>≡
  <EMACS header 10b>
  <LaTeX header 10c>
  <LaTeX packages 10d>
  %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %
  \begin{document}
  %
  <LaTeX basic definitions 11>
  <PROGRAM LaTeX title 5b>
  %
  <LaTeX Frontmatter 13b>
  <LaTeX PACKAGES Mainmatter 14a>
  <LaTeX Backmatter 16b>
  %
  <LaTeX common code appendix - perl 17>
  <LaTeX common code appendix - bash 19b>
  <LaTeX common code appendix - version 20a>
  <LaTeX common code appendix - license 20b>
  <LaTeX common code appendix - noweb 20c>
  %
  \end{document}
  %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```



```

7  <PERL PACKAGE files 7>≡
    \sctn{Module outline}

    «MODULE»=
package Sample;

use 5.006;
use strict;
use warnings;

require Exporter;
use AutoLoader qw(AUTOLOAD);

our @ISA = qw(Exporter);

# Items to export into callers namespace by default. Note: do not export
# names by default without a very good reason. Use EXPORT_OK instead.
# Do not simply export all your public functions/methods/constants.

# This allows declaration      use Sample ':all';
# If you do not need this, moving things directly into @EXPORT or @EXPORT_OK
# will save memory.
our %EXPORT_TAGS = ( 'all' => [ qw(

) ] );

our @EXPORT_OK = ( @{ $EXPORT_TAGS{'all'} } );

our @EXPORT = qw(

);
our $VERSION = '0.01';

# Preloaded methods go here.

# Autoload methods go after =cut, and are processed by the autosplit program.

1;
__END__
# Below is stub documentation for your module. You better edit it!

=head1 NAME

@@@PROJECT@@@ - Perl extension for blah blah blah

=head1 SYNOPSIS

    use @@@PROJECT@@@;
    blah blah blah

=head1 DESCRIPTION

Stub documentation for @@@PROJECT@@@, created by "deploy.pl".
It looks like the author of the extension was negligent
enough to leave the stub unedited.

Blah blah blah.

=head2 EXPORT

```

None by default.

=head1 AUTHOR

A. U. Thor, E<lt>a.u.thor@a.galaxy.far.far.awayE<gt>

=head1 SEE ALSO

L<perl>.

=cut

@

8a <PERL PACKAGE files 7>+≡
 \sctn{Makefile outline}

```
«MAKEFILE»=
use ExtUtils::MakeMaker;
# See lib/ExtUtils/MakeMaker.pm for details of how to influence
# the contents of the Makefile that is written.
WriteMakefile(
    'NAME'          => '***PROJECT***',
    'VERSION_FROM'  => '***PROJECT***.pm', # finds $VERSION
    'PREREQ_PM'     => {}, # e.g., Module::Name => 1.1
    ($] >= 5.005 ?   ## Add these new keywords supported since 5.005
        (ABSTRACT_FROM => '***PROJECT***.pm', # retrieve abstract from module
         AUTHOR       => 'A. U. Thor <a.u.thor@a.galaxy.far.far.away>') : ()),
);
@
```

8b <PERL PACKAGE files 7>+≡
 \sctn{Tester file outline}

```
«TESTER»=
# Before 'make install' is performed this script should be runnable with
# 'make test'. After 'make install' it should work as 'perl test.pl'

#####

# change 'tests => 1' to 'tests => last_test_to_print';

use Test;
BEGIN { plan tests => 1 };
use Sample;
ok(1); # If we made it this far, we're ok.

#####

# Insert your test code below, the Test module is use()ed here so read
# its man page ( perldoc Test ) for help writing this test script.

@
```

8c <PERL PACKAGE files 7>+≡
 \sctn{README file main chunk}

```
«README»=
***PROJECT*** version 0.01
```

=====

The README is used to introduce the module and provide instructions on how to install the module, any machine dependencies it may have (for example C compilers and installed libraries) and any other information that should be provided before the module is installed.

A README file is required for CPAN modules since CPAN extracts the README file from a module distribution so that people browsing the archive can use it get an idea of the modules uses. It is usually a good idea to provide version information here so that people can decide whether fixes for the module are worth downloading.

INSTALLATION

To install this module type the following:

```
perl Makefile.PL
make
make test
make install
```

DEPENDENCIES

This module requires these other modules and libraries:

blah blah blah

COPYRIGHT AND LICENCE

Put the correct copyright and licence information here.

Copyright (C) 2001 A. U. Thor blah blah blah
@

9a \langle PERL PACKAGE files 7 \rangle +≡
 \sctn{Manifest file main chunk}

```
«MANIFEST»=
Changes
Makefile.PL
MANIFEST
README
Sample.pm
test.pl
@
```

9b \langle PERL PACKAGE files 7 \rangle +≡
 \sctn{Changes log file main chunk}

```
«CHANGES»=
Revision history for Perl extension @@@PROJECT@@@.

0.01 Mon Aug 6 22:01:46 2001
    - original version; created by deploy.pl on
      a template from h2xs 1.21 with options -Xn.
@
```

3.3 Reports

```

10a  <PROJECT REPORT Template 10a>≡
      <EMACS header 10b>
      <LaTeX header 10c>
      <LaTeX packages 10d>
      %
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %
      \begin{document}
      %
      <LaTeX basic definitions 11>
      <REPORT LaTeX title 13a>
      %
      <LaTeX Frontmatter 13b>
      <LaTeX REPORT Mainmatter 16a>
      <LaTeX Backmatter 16b>
      %
      <LaTeX common code appendix - perl 17>
      <LaTeX common code appendix - awk 19a>
      <LaTeX common code appendix - bash 19b>
      <LaTeX common code appendix - version 20a>
      <LaTeX common code appendix - noweb 20c>
      %
      \end{document}
      %
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

Your NOWEB documents (suffix them with '.nw' instead of '.tex' to remember yourself that they are special \LaTeX documents if you like) must start with something like the following code:

```

10b  <EMACS header 10b>≡
      % -*- mode: Noweb; noweb-code-mode: perl-mode; tab-width: 4 -*-

```

This line tells emacs to highlight code chunks with perl language highlight syntax (other emacs modes can be 'awk-mode', 'sh-mode', 'c-mode', and so on).

```

10c  <LaTeX header 10c>≡
      \documentclass[11pt]{article}
      %
      %234567890123456789012345678901234567890123456789012345678901234567890
      %      1          2          3          4          5          6          7          8
      %
      % <Version Control Id Tag 26d>
      %

```

```

10d  <LaTeX packages 10d>≡
      \usepackage{noweb}
      \usepackage[a4paper,offset={0pt,0pt},hmargin={2cm,2cm},vmargin={1cm,1cm}]{geometry}
      \usepackage{graphics}
      \usepackage[dvips]{graphicx}
      %% pstricks
      \usepackage[dvips]{pstricks}
      \usepackage{pstricks}
      %\usepackage{pst-node}
      %\usepackage{pst-char}
      %\usepackage{pst-grad}
      %% bibliography
      \usepackage{natbib}
      %% latex2html
      \usepackage{url}
      \usepackage{html}

```

```

\usepackage{htmllist}
%% tables
\usepackage{dcolumn}
%\usepackage{colortbl}
%\usepackage{multirow}
%\usepackage{hhline}
%\usepackage{tabularx}
%% seminar
%\usepackage{semcolor,semlayer,semrot,semhelv,sem-page,slidesec}
%% draft watermark
%\usepackage[all,dvips]{draftcopy}
%\draftcopySetGrey{0.9}
%\draftcopyName{CONFIDENTIAL}{100}
%% layout
\usepackage{fancyhdr} % Do not use \usepackage{fancybox} -> TOCs disappear
%\usepackage{lscape}
%\usepackage{rotating}
%\usepackage{multicol}
%% fonts
\usepackage{times}\fontfamily{ptm}\selectfont
\usepackage{tlenc}

% noweb options
\noweboptions{smallcode}
\def\nwendcode{\endtrivlist \endgroup} % relax page breaking scheme
\let\nwdocspar=\par %

\input defs.tex % from <LaTeX new definitions> chunk

```

```

11 <LaTeX basic definitions 11>≡
«HIDE: LaTeX new definitions»=
%% Colors for gff2ps
\input ColorDefs.tex

%% New Commands are defined here
\newcommand{\sctn}[1]{\section{#1}}
\newcommand{\subscn}[1]{\subsection{#1}}
\newcommand{\subsubscn}[1]{\subsubsection{#1}}
\newcommand{\desc}[1]{\item[#1] \ \ \}
\newcommand{\todo}[1]{
  \vskip 3ex
  \hspace{-0.75cm}
  \psframebox[framearc=0.2,linecolor=darkred,linewidth=1pt,
    fillstyle=solid,fillcolor=verylightyellow,framesep=2ex]{
    \begin{minipage}[t]{16cm}
    \vskip -4.75ex
    \hspace{-1.25cm}
    \psframebox[framearc=1,linecolor=darkred,linewidth=1.25pt,
      fillstyle=solid,fillcolor=verylightorange,framesep=5pt]{
        \textcolor{darkred}{\textbf{\hspace{2ex}TO DO\hspace{2ex}}}}
    } % psframebox
    \begin{itemize}\setlength{\itemsep}{-0.5ex} #1 \end{itemize}
  \end{minipage}
} % psframebox
\vskip 1.5ex
} % newcommand todo
\newcommand{\\todoitem}[2]{
  \item[$\triangleright$] [\textit{Section}~\ref{#2},
    \textit{page}~\pageref{#2}]\ \ {#1}

```

```

} % newcommand todoitem
«HIDE: new LaTeX commands»

##### PSTricks definitions
\pslongbox{ExFrame}{\psframebox}
\newcommand{\cfn}[1]{\fcolorbox{black}{#1}{\textcolor{#1}{\rule[-.3ex]{1cm}{1ex}}}}
\newpsobject{showgrid}{psgrid}{subgriddiv=0,griddots=1,gridlabels=6pt}
% \pscharpath[fillstyle=solid, fillcolor=verydarkcyan, linecolor=black, linewidth=1pt]{\s
«HIDE: new LaTeX pstricks»

##### global urls
% \newcommand{\getpsf}[1]{\html{\htmladdnormallink{Get PostScript file}{./Psfiles/#1}}}}
«HIDE: new LaTeX urls»

##### defs
\def\noweb{\textsc{noweb}}
\def\ps{\textsc{PostScript}}
«HIDE: new LaTeX definitions»

##### TODO defs
«HIDE: new defs TODO»

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
\def\genomelab{\textbf{Genome Informatics Research Lab}}
\def\progrname{@@@QUOTED@@@}
\def\tit{\textsc{\progrname}}
%
\def\mtauthor{
  \htmladdnormallink{\texttt{author@imim.es}}
    {MAILTO:author@imim.es?subject=[@@@PROJECT@@@]}
} % def mtauthor
%
\def\authorslist{
  The Author/s {\mdseries\small\dotfill \mtauthor } \\\
  % Other authors here...\
} % def authorslist
\def\authorshort{
  Authors list here % Other authors here...
} % def authorshort
%
\def\license{GNU General Public License (GNU-GPL)}
%
\def\progdesc{
  Short description of your program here !!!
} % def progdesc
%
\def\showaffiliation{
  \scalebox{0.9 1}{\Large\textsl{\genomelab}}\\\
  Grup de Recerca en Infom\`atica Biom\`edica\\
  Institut Municipal d'Investigaci\`o M\`edica\\
  Universitat Pompeu Fabra\\[2ex]
} % def showaffiliation
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

##### Setting text for footers and headers
\fancyhead{} % clear all fields
\fancyfoot{} % clear all fields
\fancyhead[RO,LE]{\thepage}

```

```

\fancyhead[LO,RE]{\tit\quad\rightmark}
\fancyfoot[LO,LE]{\small\textbf{\genomelab}}
\fancyfoot[CO,CE]{\small\textsl{\authorshort}}
\fancyfoot[RO,RE]{\small\textbf{\today}}
\renewcommand{\headrulewidth}{1pt}
\renewcommand{\footrulewidth}{1pt}
%
@

```

```

«HIDE: new LaTeX commands»=
@
«HIDE: new LaTeX pstricks»=
@
«HIDE: new LaTeX urls»=
@
«HIDE: new LaTeX definitions»=
@
«HIDE: new defs TODO»=
@

```

13a \langle REPORT LaTeX title 13a $\rangle \equiv$

```

\thispagestyle{empty}

\begin{titlepage}

\ \vfill
\begin{center}
\textbf{\Huge \progrname}\\[5ex]

% \textbf{\Large Authors List Here}\\[1ex]
\textbf{\Large Authors List Here}\\[5ex]
% \raisebox{0.85ex}{\footnotesize$\,\,\dag$}\\[0.5ex]

\textbf{\large -- \today --}\\[10ex]

\begin{abstract}
\begin{center}
\parbox{0.75\linewidth}{
\progdesc
} % parbox
\end{center}
\end{abstract}

\vfill

\begin{raggedleft}
\showaffiliation
\raisebox{0.85ex}{\footnotesize$\dag$, $\}$\{\large e-mail: \mtauthor}\}
\end{raggedleft}
\end{center}

\end{titlepage} %'

```

13b \langle LaTeX Frontmatter 13b $\rangle \equiv$

```

%%%%%%%%%% FRONTMATTER

\newpage %%%%%%%%%%%
\pagenumbering{roman}

```

```

\setcounter{page}{1}
\pagestyle{fancy}
% Marks redefinition must go here because pagestyle
% resets the values to the default ones.
\renewcommand{\sectionmark}[1]{\markboth{}{\thesection.\ #1}}
\renewcommand{\subsectionmark}[1]{\markboth{}{\thesubsection.\ \textsl{#1}}}

\tableofcontents
\listoftables
\listoffigures

\vfill
\begin{center}
{\small\$\$ \verb$Id: deploy.nw,v 1.11 2001/10/05 16:19:45 jabril Exp $$>$ }
\end{center}

```

14a *⟨LaTeX PACKAGES Mainmatter 14a⟩*≡

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MAINMATTER

\newpage %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\pagenumbering{arabic}
\setcounter{page}{1}

\sctn{Introduction}

\subscn{Program description}

% \subscn{Input}

% \subscn{Output}

% \subscn{Comments}

\subscn{To Do}

⟨PERL PACKAGE files 7⟩

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\begin{comment}
\end{comment}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

14b *⟨LaTeX SCRIPT Mainmatter 14b⟩*≡

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MAINMATTER

\newpage %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\pagenumbering{arabic}
\setcounter{page}{1}

\sctn{Introduction}

\subscn{Program description}

\subscn{Input}

\subscn{Output}

% \subscn{Comments}

```



```

\subsectn{To Do}

\begin{itemize}
\input todo.tex
\end{itemize}

\newpage %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\sctn{Implementation}

«Program Info»=
my $PROGRAM = '@@@PROJECT@@@';
my $VERSION = '0.1_alpha';
@
«Prog USAGE»=
$PROGRAM [options] < input_files > output_files
@
«Prog DESC»=
A short description of what your program does.
@

«Program Description»=
# #-----#
# #                                     @@@PROJECT@@@                               #
# #-----#
#
# $PROGRAM [options] < input_files > output_files
#
# Remember to put a short description of what your program does here...
#
# Copyright (C) 2001 - Josep Francesc ABRIL FERRANDO
@

\subsectn{Program outline}

«@@@PROJECT@@@»=
«PERL shebang»
#
# MODULES
#
«Use Modules»
#
# VARIABLES
#
«Global Vars»
#
# MAIN LOOP
#
«Main Loop»
#
# FUNCTIONS
#
«Functions»
@

«Use Modules»=
@

```

```

«Global Vars»=
@

«Main Loop»=

exit(0);
@

«Functions»=
sub {
} #
@

\label{todo:AAA}
«HIDE: new defs TODO»=
\def\todoAAA{This is a first draft of the {\programe}.} % todoAAA
@
«HIDE: TODO»=
\todoitem{\todoAAA}{todo:AAA}
@
\todo{ \item \todoAAA } % todo

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\begin{comment}
\end{comment}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

16a \langle LaTeX REPORT Mainmatter 16a $\rangle \equiv$

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MAINMATTER

\newpage %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\pagenumbering{arabic}
\setcounter{page}{1}

\sctn{Introduction}

\sctn{Working on data}

\subsctn{Command-line}

«BASH commands»=
#
#
@

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\begin{comment}
\end{comment}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

16b \langle LaTeX Backmatter 16b $\rangle \equiv$

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% BACKMATTER

% \newpage %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% \bibliographystyle{apalike}
% \bibliography{/home1/rguigo/docs/biblio/References}

```

```

\newpage %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\appendix

\sctn{empty appendix section}

\subsectn{empty appendix subsection}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\begin{comment}
\end{comment}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

4 Appendixes

17 \langle LaTeX common code appendix - perl 17 $\rangle \equiv$

```

\newpage %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\sctn{Common code blocks}

\subsectn{PERL scripts}

«PERL shebang»=
#!/usr/bin/perl -w
# This is perl, version 5.005_03 built for i386-linux
#
«Program Description»
#
«GNU License»
#
«Version Control Id Tag»
#
use strict;
#
«Program Info»
my $DATE = localtime;
my $USER = defined($ENV{USER}) ? $ENV{USER} : 'Child Process';
my $host = 'hostname';
chomp($host);
#
@

«Global Constants - Boolean»=
my ($T,$F) = (1,0); # for 'T' rue and 'F'alse
@ %def $T $F

\subsubsectn{Timing our scripts}

The '[[Benchmark]]' module encapsulates a number of routines to help to figure out how long
to execute a piece of code and the whole script.

«Use Modules - Benchmark»=
use Benchmark;
    «Timer ON»
@

See '[[man Benchmark]]' for further info about this package.

```

We set an array to keep record of timing for each section.

```
«Timer ON»=
my @Timer = (new Benchmark);
@

«Common PERL subs - Benchmark»=
sub timing() {
    push @Timer, (new Benchmark);
    # partial time
    $_[0] ||
        (return timestr(timediff($Timer[$#Timer], $Timer[( $#Timer - 1 ] ))));
    # total time
    return timestr(timediff($Timer[$#Timer], $Timer[0]));
} # timing
@
```

\subsubscn{Printing complex Data Structures}

With '[[Data::Dumper]]' we are able to pretty print complex data structures for debugging

```
«Use Modules - Dumper»=
use Data::Dumper;
local $Data::Dumper::Purity = 0;
local $Data::Dumper::Deepcopy = 1;
@
```

\subsubscn{Common functions}

```
«Skip comments and empty records»=
next if /^#\s/;
next if /^s*$/;
chomp;
@
```

```
«Common PERL subs - Min Max»=
#
sub max() {
    my $z = shift @_;
    foreach my $l (@_) { $z = $l if $l > $z };
    return $z;
} # max
sub min() {
    my $z = shift @_;
    foreach my $l (@_) { $z = $l if $l < $z };
    return $z;
} # min
@
```

```
«Common PERL subs - Text fill»=
#
sub fill_right() { $_[0].($_[2] x ($_[1] - length($_[0]))) }
sub fill_left() { ($_[2] x ($_[1] - length($_[0]))).$_[0] }
sub fill_mid() {
    my $l = length($_[0]);
    my $k = int(($_[1] - $l)/2);
    ($_[2] x $k).$_[0].($_[2] x ($_[1] - ($l+$k)));
}
```

```
} # fill_mid
@
```

These functions are used to report to STDERR a single char for each record processed (useful for reporting parsed records).

```
«Common PERL subs - Counter»=
#
sub counter { # $_[0]~current_pos++ $_[1]~char
    print STDERR "$_[1]";
    (( $_[0] % 50 ) == 0 ) && (print STDERR "[".&fill_left($_[0],6,"0")."]\n");
} # counter
#
sub counter_end { # $_[0]~current_pos $_[1]~char
    (( $_[0] % 50 ) != 0 ) && (print STDERR "[".&fill_left($_[0],6,"0")."]\n");
} # counter_end
@

«Global Vars - Counter»=
my ($n,$c); # counter and char (for &counter function)
@ %def $n $c
```

```
\subsubscn{Common functions for reporting program processes}
\label{sec:messengerpt}
```

Function '[[report]]' requires that a hash variable '[[%MessageList]]' has been set, such as contains the strings for each report message we will need. The first parameter for '[[report]]' is the order to retrieve the message string, the other parameters passed are processed by the '[[sprintf]]' function on that string.

```
«Common PERL subs - STDERR»=
sub report() { print STDERR sprintf($MessageList{ shift @_ },@_ ) }
@
```

The same happens to '[[warn]]' function which also requires a hash variable '[[%ErrorList]]' containing the error messages.

```
«Common PERL subs - STDERR»=
sub warn() { print STDERR sprintf($ErrorList{ shift @_ }, @_ ) }
@
```

19a *<LaTeX common code appendix - awk 19a>≡*

```
\subscn{AWK scripts}
```

```
«GAWK shebang»=
#!/usr/bin/gawk -f
# GNU Awk 3.0.4
«Version Control Id Tag»
@
```

19b *<LaTeX common code appendix - bash 19b>≡*

```
\subscn{BASH scripts}
```

```
«BASH shebang»=
#!/usr/bin/bash
# GNU bash, version 2.03.6(1)-release (i386-redhat-linux-gnu)
«Version Control Id Tag»
```

```

#
SECONDS=0 # Reset Timing
# Which script are we running...
L="#####"
{ echo "$L$L$L$L";
  echo "### RUNNING [$0]";
  echo "### Current date:`date`";
  echo "###"; } 1>&2;
@

«BASH script end»=
{ echo "###"; echo "### Execution time for [$0] : $SECONDS secs";
  echo "$L$L$L$L";
  echo ""; } 1>&2;
#
exit 0
@

```

20a *<LaTeX common code appendix - version 20a>*≡
`\subscn{Version control tags}`

This document is under Revision Control System (RCS). The version you are currently reading is the following:

```

«Version Control Id Tag»=
# $Id: deploy.nw,v 1.11 2001/10/05 16:19:45 jabril Exp $
@

```

20b *<LaTeX common code appendix - license 20b>*≡
`\subscn{GNU General Public License}`

```

«GNU License»=
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
# #-----#
@

```

20c *<LaTeX common code appendix - noweb 20c>*≡
`\newpage %%`

`\scn{Extracting code blocks from this document}`

From this file we can obtain both the code and the documentation. The following instructions are needed:

```
\subscn{Extracts Script code chunks from the {\noweb} file} % \[-0.5ex]
```

Remember when tangling that '-L' option allows you to include program line-numbering relative to original {\noweb} file. Then the first line of the executable files is a comment bang, and must be removed to make scripts runnable.

```
«tangling»=
# showing line numbering comments in program
notangle -L -R"@@@PROJECT@@@" $WORK/$nwfile.nw | \
    perl -ne '$.>1 && print' | cpif $BIN/@@@PROJECT@@@ ;
chmod a+x $BIN/@@@PROJECT@@@ ;
@

«tangling»=
# reformatting program with perltidy
notangle -R"@@@PROJECT@@@" $WORK/$nwfile.nw | \
    perltidy - | cpif $BIN/@@@PROJECT@@@ ;
# html pretty-printing program with perltidy
notangle -R"@@@PROJECT@@@" $WORK/$nwfile.nw | \
    perltidy -html - | cpif $DOCS/html/@@@PROJECT@@@.html ;
#
@
```

```
\subscn{Extracting different Config Files} % \[-0.5ex]
```

```
«tangling»=
notangle -R"root" $WORK/$nwfile.nw | \
    cpif $DATA/root_config ;
@ %$
```

```
\subscn{Extracting documentation and \LaTeX{}'ing it} % \[-0.5ex] %'
```

```
«tangling»=
notangle -Rweaving $WORK/$nwfile.nw | cpif $WORK/nw2tex ;
notangle -RLaTeXing $WORK/$nwfile.nw | cpif $WORK/ltx ;
chmod a+x $WORK/nw2tex $WORK/ltx;
@
```

```
«tangling complementary LaTeX files»=
notangle -R"HIDE: LaTeX new definitions" $WORK/$nwfile.nw | cpif $DOCS/defs.tex ;
notangle -R"HIDE: TODO" $WORK/$nwfile.nw | cpif $DOCS/todo.tex ;
@
```

```
«weaving»=
«BASH shebang»
# weaving and LaTeXing
«BASH Environment Variables»
«tangling complementary LaTeX files»
noweave -v -t4 -delay -x -filter 'elide "HIDE: *"' \
    $WORK/$nwfile.nw | cpif $DOCS/$nwfile.tex ;
# noweave -t4 -delay -index $WORK/$nwfile.nw > $DOCS/$nwfile.tex
pushd $DOCS/ ;
#
latex $nwfile.tex ;
dvips $nwfile.dvi -o $nwfile.ps -t a4 ;
#
popd;
«BASH script end»
@
```

```

«LaTeXing»=
«BASH shebang»
# only LaTeXing
«BASH Environment Variables»
pushd $DOCS/ ;
#
echo "### RUNNING LaTeX on $nwfile.tex" 1>&2 ;
latex $nwfile.tex ;
latex $nwfile.tex ;
latex $nwfile.tex ;
dvips $nwfile.dvi -o $nwfile.ps -t a4 ;
#
# pdflatex $nwfile.tex ;
echo "### CONVERTING PS to PDF: $nwfile" 1>&2 ;
ps2pdf $nwfile.ps $nwfile.pdf ;
#
popd ;
«BASH script end»
@ %$

```

\subscn{Defining working shell variables for the current project} % \[-0.5ex]

```

«BASH Environment Variables»=
#
# Setting Global Variables
WORK="@@@PATH@@@" ;
BIN="$WORK/bin" ;
PARAM="$BIN/param" ;
SRC="$WORK/src" ; # where to put the distributable files
DOCS="$WORK/docs" ;
DATA="$WORK/data" ;
TEST="$WORK/tests" ;
nwfile="@@@PROJECT@@@" ;
export WORK BIN PARAM DOCS DATA nwfile ;
#
@

```

```

«tangling»=
#
# BASH Environment Variables
notangle -R'BASH Environment Variables' $WORK/$nwfile.nw | \
    cpif $WORK/.bash_VARS ;
source $WORK/.bash_VARS ;
#
@

```


A empty appendix section

A.1 empty appendix subsection

B Common code blocks

B.1 PERL scripts

```

24a  <PERL shebang 24a>≡
      #!/usr/bin/perl -w
      # This is perl, version 5.005_03 built for i386-linux
      #
      <Program Description 2b>
      #
      <GNU License 26e>
      #
      <Version Control Id Tag 26d>
      #
      use strict;
      #
      <Program Info 2a>
      my $DATE = localtime;
      my $USER = defined($ENV{USER}) ? $ENV{USER} : 'Child Process';
      my $host = 'hostname';
      chomp($host);
      #

24b  <Global Constants - Boolean 24b>≡
      my ($T,$F) = (1,0); # for 'T'rue and 'F'alse

```

B.1.1 Timing our scripts

The 'Benchmark' module encapsulates a number of routines to help to figure out how long it takes to execute a piece of code and the whole script.

```

24c  <Use Modules - Benchmark 24c>≡
      use Benchmark;
      <Timer ON 24d>

      See 'man Benchmark' for further info about this package. We set an array to keep record of timing for
      each section.

24d  <Timer ON 24d>≡
      my @Timer = (new Benchmark);

24e  <Common PERL subs - Benchmark 24e>≡
      sub timing() {
          push @Timer, (new Benchmark);
          # partial time
          $_[0] ||
              (return timestr(timediff($Timer[$#Timer],$Timer[( $#Timer - 1)])));
          # total time
          return timestr(timediff($Timer[$#Timer],$Timer[0]));
      } # timing

```

B.1.2 Printing complex Data Structures

With 'Data::Dumper' we are able to pretty print complex data structures for debugging them.

```

24f  <Use Modules - Dumper 24f>≡
      use Data::Dumper;
      local $Data::Dumper::Purity = 0;
      local $Data::Dumper::Deepcopy = 1;

```

B.1.3 Common functions

```

25a  <Skip comments and empty records 25a>≡
      next if /^#\s/;
      next if /^s*$/;
      chomp;

25b  <Common PERL subs - Min Max 25b>≡
      #
      sub max() {
          my $z = shift @_;
          foreach my $l (@_) { $z = $l if $l > $z };
          return $z;
      } # max
      sub min() {
          my $z = shift @_;
          foreach my $l (@_) { $z = $l if $l < $z };
          return $z;
      } # min

25c  <Common PERL subs - Text fill 25c>≡
      #
      sub fill_right() { $_[0].($_[2] x ($_[1] - length($_[0]))) }
      sub fill_left()  { ($_[2] x ($_[1] - length($_[0]))).$_[0] }
      sub fill_mid()   {
          my $l = length($_[0]);
          my $k = int(($_[1] - $l)/2);
          ($_[2] x $k).$_[0].($_[2] x ($_[1] - ($l+$k)));
      } # fill_mid

```

These functions are used to report to STDERR a single char for each record processed (useful for reporting parsed records).

```

25d  <Common PERL subs - Counter 25d>≡
      #
      sub counter { # $_[0]~current_pos++ $_[1]~char
          print STDERR "$_[1]";
          (($_[0] % 50) == 0) && (print STDERR "[".&fill_left($_[0],6,"0")."]\n");
      } # counter
      #
      sub counter_end { # $_[0]~current_pos $_[1]~char
          (($_[0] % 50) != 0) && (print STDERR "[".&fill_left($_[0],6,"0")."]\n");
      } # counter_end

25e  <Global Vars - Counter 25e>≡
      my ($n,$c); # counter and char (for &counter function)

```

B.1.4 Common functions for reporting program processes

Function 'report' requires that a hash variable '%MessageList' has been set, such hash contains the strings for each report message we will need. The first parameter for 'report' is a key for that hash, in order to retrieve the message string, the other parameters passed are processed by the sprintf function on that string.

```

25f  <Common PERL subs - STDERR 25f>≡
      sub report() { print STDERR sprintf($MessageList{ shift @_ },@_) }

```

The same happens to 'warn' function which also requires a hash variable '%ErrorList' containing the error messages.

```

25g  <Common PERL subs - STDERR 25f>+≡
      sub warn() { print STDERR sprintf($ErrorList{ shift @_ }, @_ ) }

```

B.2 AWK scripts

26a `<GAWK shebang 26a>≡`
`#!/usr/bin/gawk -f`
`# GNU Awk 3.0.4`
`<Version Control Id Tag 26d>`

B.3 BASH scripts

26b `<BASH shebang 26b>≡`
`#!/usr/bin/bash`
`# GNU bash, version 2.03.6(1)-release (i386-redhat-linux-gnu)`
`<Version Control Id Tag 26d>`
`#`
`SECONDS=0 # Reset Timing`
`# Which script are we running...`
`L="#####"`
`{ echo "LLLL$L";`
`echo "### RUNNING [$0]";`
`echo "### Current date:`date`";`
`echo "###"; } 1>&2;`

26c `<BASH script end 26c>≡`
`{ echo "###"; echo "### Execution time for [$0] : $SECONDS secs";`
`echo "LLLL$L";`
`echo "; } 1>&2;`
`#`
`exit 0`

B.4 Version control tags

This document is under Revision Control System (RCS). The version you are currently reading is the following:

26d `<Version Control Id Tag 26d>≡`
`# $Id: deploy.nw,v 1.11 2001/10/05 16:19:45 jabril Exp $`

B.5 GNU General Public License

26e `<GNU License 26e>≡`
`# This program is free software; you can redistribute it and/or modify`
`# it under the terms of the GNU General Public License as published by`
`# the Free Software Foundation; either version 2 of the License, or`
`# (at your option) any later version.`
`#`
`# This program is distributed in the hope that it will be useful,`
`# but WITHOUT ANY WARRANTY; without even the implied warranty of`
`# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the`
`# GNU General Public License for more details.`
`#`
`# You should have received a copy of the GNU General Public License`
`# along with this program; if not, write to the Free Software`
`# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.`
`#`
`# #-----#`

C Extracting code blocks from this document

From this file we can obtain both the code and the documentation. The following instructions are needed:

C.1 Extracts Script code chunks from the NOWEB file

Remember when tangling that '-L' option allows you to include program line-numbering relative to original NOWEB file. Then the first line of the executable files is a comment, not a shebang, and must be removed to make scripts runnable.

```
27a <tangling development version 27a>≡
# showing line numbering comments in program
notangle -L -R"DEPLOY" $WORK/$nwfile.nw | \
perl -ne '$.>1 && print' | cpif $BIN/deploy.pl ;
chmod a+x $BIN/deploy.pl ;
#
```

We use `perltidy`¹ to reformat final versions of perl scripts (without line numbering comments, correct indentations, etc...) and to pretty-print in html format.

```
27b <tangling 27b>≡
# reformatting program with perltidy
notangle -R"DEPLOY" $WORK/$nwfile.nw | \
perltidy - | cpif $BIN/deploy.pl ;
# pretty-printing program with perltidy
notangle -R"DEPLOY" $WORK/$nwfile.nw | \
perltidy -html - | cpif $DOCS/html/deploy.html ;
#
```

C.2 Extracting different Config Files

```
27c <tangling 27b>+≡
notangle -R"PROJECT SCRIPT Template" $WORK/$nwfile.nw | \
cpif $DATA/perlscript.nw ;
notangle -R"PROJECT PERL PACKAGE Template" $WORK/$nwfile.nw | \
cpif $DATA/perlpackage.nw ;
notangle -R"PROJECT REPORT Template" $WORK/$nwfile.nw | \
cpif $DATA/report.nw ;
```

C.3 Extracting documentation and L^AT_EX'ing it

```
27d <tangling 27b>+≡
notangle -Rweaving $WORK/$nwfile.nw | cpif $WORK/nw2tex ;
notangle -RLaTeXing $WORK/$nwfile.nw | cpif $WORK/ltx ;
chmod a+x $WORK/nw2tex $WORK/ltx;

27e <tangling complementary LaTeX files 27e>≡
notangle -R"HIDE: LaTeX deploy new definitions" \
$WORK/$nwfile.nw | cpif $DOCS/defs.tex ;
notangle -R"HIDE: TODO" $WORK/$nwfile.nw | cpif $DOCS/todo.tex ;
```

¹<http://perltidy.sourceforge.net/>

```

27f  <weaving 27f>≡
      <BASH shebang 26b>
      # weaving and LaTeXing
      <BASH Environment Variables 28b>
      <tangling complementary LaTeX files 27e>
      noweave -v -t4 -delay -x -filter 'elide "HIDE: *"' \
              $WORK/$nwfile.nw | cpif $DOCS/$nwfile.tex ;
      # noweave -t4 -delay -index $WORK/$nwfile.nw > $DOCS/$nwfile.tex
      pushd $DOCS/ ;
      #
      echo "### RUNNING LaTeX on $nwfile.tex" 1>&2 ;
      latex $nwfile.tex ;
      dvips $nwfile.dvi -o $nwfile.ps -t a4 ;
      #
      popd ;
      <BASH script end 26c>

28a  <LaTeXing 28a>≡
      <BASH shebang 26b>
      # only LaTeXing
      <BASH Environment Variables 28b>
      pushd $DOCS/ ;
      #
      echo "### RUNNING LaTeX on $nwfile.tex" 1>&2 ;
      latex $nwfile.tex ;
      latex $nwfile.tex ;
      latex $nwfile.tex ;
      dvips $nwfile.dvi -o $nwfile.ps -t a4 ;
      #
      # pdflatex $nwfile.tex ;
      echo "### CONVERTING PS to PDF: $nwfile" 1>&2 ;
      ps2pdf $nwfile.ps $nwfile.pdf ;
      #
      popd ;
      <BASH script end 26c>

```

C.4 Defining working shell variables for the current project

```

28b  <BASH Environment Variables 28b>≡
      #
      # Setting Global Variables
      WORK="$HOME/development/softjabril/deploy" ;
      BIN="$WORK/bin" ;
      PARAM="$BIN/param" ;
      DOCS="$WORK/docs" ;
      DATA="$WORK/data" ;
      TEST="$WORK/tests" ;
      nwfile="deploy" ;
      export WORK BIN PARAM DOCS DATA nwfile ;
      #

28c  <tangling 27b>+≡
      #
      # BASH Environment Variables
      notangle -R'BASH Environment Variables' $WORK/$nwfile.nw | \
              cpif $WORK/.bash_VARS ;
      source $WORK/.bash_VARS ;
      #

```