
PROGRAM NAME: **coords2gff**

AUTHOR: **The Author/s** author@imim.es

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

LAST UPDATE: **September 4, 2001**

DESCRIPTION: Short description of your program here !!!

Genome Informatics Research Lab
Grup de Recerca en Infomàtica Biomèdica
Institut Municipal d'Investigació Mèdica
Universitat Pompeu Fabra

Contents

| | | |
|----------|---|----------|
| 1 | Introduction | 1 |
| 1.1 | Program description | 1 |
| 1.2 | Input | 1 |
| 1.3 | Output | 1 |
| 1.4 | To Do | 1 |
| 2 | Implementation | 2 |
| 2.1 | Program outline | 2 |
| A | empty appendix section | 3 |
| A.1 | empty appendix subsection | 3 |
| B | Common code blocks | 4 |
| B.1 | PERL scripts | 4 |
| B.1.1 | Timing our scripts | 4 |
| B.1.2 | Printing complex Data Structures | 4 |
| B.1.3 | Common functions | 4 |
| B.1.4 | Common functions for reporting program processes | 5 |
| B.2 | BASH scripts | 6 |
| B.3 | Version control tags | 6 |
| B.4 | GNU General Public License | 6 |
| C | Extracting code blocks from this document | 7 |
| C.1 | Extracts Script code chunks from the NOWEB file | 7 |
| C.2 | Extracting different Config Files | 7 |
| C.3 | Extracting documentation and L ^A T _E X'ing it | 7 |
| C.4 | Defining working shell variables for the current project | 8 |

List of Tables

List of Figures

1 Introduction

1.1 Program description

1.2 Input

1.3 Output

1.4 To Do

- This is a first draft of the coords2gff. [Section 2.1, page 2]

2 Implementation

2.1 Program outline

2a $\langle coords2gff\ 2a \rangle \equiv$
 $\langle PERL\ shebang\ 4a \rangle$
 #
 # MODULES
 #
 $\langle Use\ Modules\ 2b \rangle$
 #
 # VARIABLES
 #
 $\langle Global\ Vars\ 2c \rangle$
 #
 # MAIN LOOP
 #
 $\langle Main\ Loop\ 2d \rangle$
 #
 # FUNCTIONS
 #
 $\langle Functions\ 2e \rangle$

2b $\langle Use\ Modules\ 2b \rangle \equiv$

2c $\langle Global\ Vars\ 2c \rangle \equiv$

2d $\langle Main\ Loop\ 2d \rangle \equiv$

 `exit(0);`

2e $\langle Functions\ 2e \rangle \equiv$
 `sub {`
 `} #`

TO DO

- This is a first draft of the coords2gff.

A empty appendix section

A.1 empty appendix subsection

B Common code blocks

B.1 PERL scripts

4a *<PERL shebang 4a>*≡

```
#!/usr/bin/perl -w
# This is perl, version 5.005_03 built for i386-linux
<GNU License 6d>
<Version Control Id Tag 6c>
#
use strict;
```

4b *<Global Constants - Boolean 4b>*≡

```
my ($T,$F) = (1,0); # for 'T'rue and 'F'alse
```


 We also set here the date when the script is running and who is the user running it.

4c *<Global Vars - User and Date 4c>*≡

```
my $DATE = localtime;
my $USER = $ENV{USER};
```

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.

4d *<Use Modules - Benchmark 4d>*≡

```
use Benchmark;
<Timer ON 4e>
```

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

4e *<Timer ON 4e>*≡

```
my @Timer = (new Benchmark);
```

4f *<Common PERL subs - Benchmark 4f>*≡

```
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.

4g *<Use Modules - Dumper 4g>*≡

```
use Data::Dumper;
local $Data::Dumper::Purity = 0;
local $Data::Dumper::Deepcopy = 1;
```

B.1.3 Common functions

4h *<Skip comments and empty records 4h>*≡

```
next if /\^#\o;
next if /\^s*\$/o;
chomp;
```

5a *<Common PERL subs - Min Max 5a>≡*

```
#
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
```

5b *<Common PERL subs - Text fill 5b>≡*

```
#
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).

5c *<Common PERL subs - Counter 5c>≡*

```
#
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
```

5d *<Global Vars - Counter 5d>≡*

```
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.

5e *<Common PERL subs - STDERR 5e>≡*

```
sub report() { print STDERR sprintf($MessageList{ shift @_ },@_) }
```

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

5f *<Common PERL subs - STDERR 5e>+≡*

```
sub warn() { print STDERR sprintf($ErrorList{ shift @_ }, @_) }
```

B.2 BASH scripts

```

6a  <BASH shebang 6a>≡
    #!/usr/bin/bash
    # GNU bash, version 2.03.6(1)-release (i386-redhat-linux-gnu)
    <Version Control Id Tag 6c>
    #
    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;

6b  <BASH script end 6b>≡
    { echo "###"; echo "### Execution time for [$0] : $SECONDS secs";
      echo "$L$L$L$L";
      echo ""; } 1>&2;
    #
    exit 0

```

B.3 Version control tags

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

```

6c  <Version Control Id Tag 6c>≡
    # $Id: deploy.nw,v 1.7 2001/09/03 18:23:46 jabril Exp $

```

B.4 GNU General Public License

```

6d  <GNU License 6d>≡
    # #-----#
    # #                                coords2gff                                #
    # #-----#
    #
    # Remember to put a short description of your script here...
    #
    # Copyright (C) 2001 - Josep Francesc ABRIL FERRANDO
    #
    # 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.

```
7a <tangling 7a>≡
    # showing line numbering comments in program
    notangle -L -R"coords2gff" $WORK/$nwfile.nw | \
        perl -ne '$.>1 && print' | cpif $BIN/coords2gff ;
    chmod a+x $BIN/coords2gff ;

7b <tangling 7a>+≡
    # reformatting program with perltidy
    notangle -R"coords2gff" $WORK/$nwfile.nw | \
        perltidy - | cpif $BIN/coords2gff ;
    # html pretty-printing program with perltidy
    notangle -R"coords2gff" $WORK/$nwfile.nw | \
        perltidy -html - | cpif $DOCS/html/coords2gff.html ;
    #
```

C.2 Extracting different Config Files

```
7c <tangling 7a>+≡
    notangle -R"root" $WORK/$nwfile.nw | \
        cpif $DATA/root_config ;
```

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

```
7d <tangling 7a>+≡
    notangle -Rweaving $WORK/$nwfile.nw | cpif $WORK/nw2tex ;
    notangle -RLaTeXing $WORK/$nwfile.nw | cpif $WORK/ltx ;
    chmod a+x $WORK/nw2tex $WORK/ltx;

7e <tangling complementary LaTeX files 7e>≡
    notangle -R"HIDE: LaTeX new definitions" $WORK/$nwfile.nw | cpif $DOCS/defs.tex ;
    notangle -R"HIDE: TODO" $WORK/$nwfile.nw | cpif $DOCS/todo.tex ;

7f <weaving 7f>≡
    <BASH shebang 6a>
    # weaving and LaTeXing
    <BASH Environment Variables 8b>
    <tangling complementary LaTeX files 7e>
    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 6b>
```

```

8a  <LaTeXing 8a>≡
    <BASH shebang 6a>
    # only LaTeXing
    <BASH Environment Variables 8b>
    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 6b>

```

C.4 Defining working shell variables for the current project

```

8b  <BASH Environment Variables 8b>≡
    #
    # Setting Global Variables
    WORK="/home/ug/jabril/development/softjabril/gfftools/coords2gff" ;
    BIN="$WORK/bin" ;
    PARAM="$BIN/param" ;
    DOCS="$WORK/docs" ;
    DATA="$WORK/data" ;
    nwfile="coords2gff" ;
    export WORK BIN PARAM DOCS DATA nwfile ;
    #

8c  <tangling 7a>+≡
    #
    # BASH Environment Variables
    notangle -R'BASH Environment Variables' $WORK/$nwfile.nw | \
        cpif $WORK/.bash_VARS ;
    source $WORK/.bash_VARS ;
    #

```