The svn-multi.pl Script

Martin Scharrer
martin@scharrer-online.de
http://latex.scharrer-online.de/svn-multi
CTAN: http://tug.ctan.org/pkg/svn-multi

Version 0.1a
July 26, 2010

Note: This document is work in progress.

1 Usage

See the section in the `svn-multi` package manual or the `usage` function below.

2 Implementation

2.1 Script Head

Loading of required Modules.

```
use strict;
use warnings;
use File::Basename;
```

Declaration of constants VERSION, REV and DATE for script info output:

```
my $VERSION = "0.2";
my ($REV,$DATE) =
  (split ' ','$Id$')[2,3];
```

Declaration of other global constants and variables:

```
my $dollar  = '$';
my @PATH;
my %EXCLUDE = map { $_ => 1 } qw(sty tex aux log out toc fff ttt svn svx);
```

Subfunction prototypes:

```
sub create_svxfile ($@);
sub usage;
```

Check if help was requested:

```
if (!@ARGV or grep { $_ eq '--help' or $_ eq '-h' } @ARGV) {
  usage();
}
```

Print identification line like other TeX related programs:

```
print STDOUT "This is svn-multi.pl, Version $VERSION-$REV, $DATE\n";
```

The first argument is taken as the jobname. To be more userfriendly the name can include some standard extension and a path which are stripped.

```
my ($jobname, $dir, $suffix) = fileparse(shift @ARGV, qr/\.(tex|ltx|dtx|svn)$/);
```

If a directory was specified this program is changing into it because the file paths in the `.aux` file are relative to it.

```
if ($dir && $dir ne './') {
  printf STDOUT "Main directory is '$dir'.\n";
  chdir($dir);
}

if ($jobname =~ /^-/) {
  usage();
}
my $outfile = "$jobname.svx";

my %external;

```

Regular expressions to read the `.aux` file:

svnexternalpath: The format is:
`\@svnexternalpath{{patha}{pathb}{...}{pathd}}`

```
my $resvnexternalpath = qr/
        ^                   # at begin of line
   \s*                      # allow for spaces
      \\\@svnexternalpath   # the macro name
   \s*
        {                   # begin token group
   \s*
        (?:                 # paths:
            {               # { of first path
              (.*)          # everything else, e.g: 'patha}{pathb}{pathc'
            }               # } of last path
            |               # or nothing
        )
   \s*
        }                   # end token group
   \s*
        $                   # end of line
   /x;
```

svnexternal: The format is:
`\@svnexternal[group name]{{filea}{fileb}{...}{filed}}`

```
my $resvnexternal = qr/
        ^                   # at begin of line
```

```
  \s*                       # allow for spaces
     \\\@svnexternal        # the macro name
  \s*
       (?:                  # optional:
           \[               # opening [
               ([^\]]*)     # group name (everything until ])
           \]               # closing ]
       )?
  \s*
       {                    # begin token group
           ([^}]+)          # file name (everything until })
       }                    # end token group
  \s*
       {                    # begin token group
  \s*
       (?:                  # paths:
           {                # { of first file
             (.*)           # everything else, e.g: 'filea}{fileb}{filec'
           }                # } of last file
           |                # or nothing
       )
  \s*
       }                    # end token group
  \s*
       $                    # end of line
  /x;
if (-e "$jobname.aux" and open( my $svnfh, '<', "$jobname.aux")) {
  print STDOUT "Reading '$jobname.aux'.\n";
  while (<$svnfh>) {
    chomp;
    if  (/$resvnexternalpath/) {
      push @PATH, ( split /}\s*{/, $1 );
    }
    elsif (/$resvnexternal/) {
      my ($group,$file,$list) = ($1||"",$2,$3||"");
      $file =~ s/^\.\///;
      push @{$external{$file}{$group} ||= []}, ( split /}\s*{/, $list );
    }
  }
  close ($svnfh);
}
else {
  warn "No .aux file found for '$jobname'!\n";
}

# Add TEXINPUTS to path
push @PATH, map { $_ =~ s/(?<!\/)$/\//; $_ } grep { $_ }
        split(':', $ENV{'TEXINPUTS'}||"");

```

```
my @mainfilepairs;
my $maintex = "$jobname.tex";
if (exists $external{$maintex}) {
  while ( my ($group,$list) = each %{$external{$maintex}} ) {
  push @mainfilepairs, [ $group, [ @$list ] ];
  }
  delete $external{$maintex};
}

push @mainfilepairs, parse_args(@ARGV);
create_svxfile("$jobname.svx", @mainfilepairs )
  if @mainfilepairs;

foreach my $file (keys %external) {
  my @pairs;
  my $svxfile = $file;
  $svxfile =~ s/\.(tex|ltx)$/.svx/;
  while ( my ($group,$list) = each %{$external{$file}} ) {
  push @pairs, [ $group, [ @$list ] ];
  }
  create_svxfile($svxfile, @pairs);
}
```

2.2 Functions

`parse args` Parses the arguments and builds a list of (group,(files)) pairs.

```
sub parse_args {
  my @args = @_;
  my $group = '';
  my @files;
  my $readfg;
  my @pairs;

  foreach my $arg (@args) {
    if ($readfg) {
      $readfg = 0;
      $group = $arg;
      $group =~ s/^["']|["']$//; # '
    }
    elsif ($arg =~ /^--group|^-?-fg/) {
      push @pairs, [ $group, [ @files ] ];
      @files = ();
      if ($arg =~ /^--group=(.*)/) {
        $group = $1;
        $group =~ s/^["']|["']$//; # '
      }
      else {
        $readfg = 1;
      }
```

```
143     }
144     elsif ($arg =~ /^--fls/) {
145       push @files, read_fls("$jobname.fls");
146     }
147     else {
148       push @files, $arg;
149     }
150   }
151   push @pairs, [ $group, [ @files ] ] if @files;
152   return @pairs;
153 }
```

`path search` Search all directories in `PATH` to find the given file and return the first own found.

```
154 sub path_search {
155   my $file = shift;
156   $file =~ s/##/#/g;
157   return $file if not $file or -e $file or not @PATH;
158
159   foreach my $dir (@PATH) {
160     if (-e "$dir$file") {
161       return "$dir$file";
162     }
163   }
164
165   return $file;
166 }
```

`ceate svxfile` Creates the `.svx` file named by the first argument. The second argument is a list of (group name/files) pairs.

```
167 sub create_svxfile ($@) {
168   my ($svxfile, @fgpair) = @_;
169   my $lastgroup;
170   my $fgused = 0;
171   my %seen;
172   return if not @fgpair or not $svxfile;
173
174   open(my $svxfh, '>', $svxfile) or do {
175     warn "ERROR: Could not create SVX file '$svxfile'!\n";
176     return;
177   };
178   print STDOUT "Generating .svx file '$svxfile'.\n";
179   select $svxfh;
180   print "% Generated by svn-multi.pl v$VERSION\n\n";
181
182   while ( my ($group, $files) = @{shift @fgpair||[]}) {
183     no warnings 'uninitialized';
184     if ( (not defined $lastgroup and $group) or ($group ne $lastgroup) ) {
185       print "\\svngroup{$group}\n";
186     }
187     use warnings;
```

```
188     if ($group) {
189       $fgused = 1;
190     }
191
192     foreach my $file (@$files) {
193       $file = path_search($file);
194
195       # Only print the file once per group and .svx file
196       next if $seen{$group}{$file};
197       $seen{$group}{$file} = 1;
198
199       open(my $infoh, '-|', "svn info '$file' 2>/dev/null") or next;
200       my %info = map { chomp; split /\s*:\s*/, $_, 2 } <$infoh>;
201       close($infoh);
202       if (not keys %info) {
203         print "% Could not receive keywords for '$file'!\n\n";
204         next;
205       }
206       print "% Keywords for '$file'\n";
207       print svnidlong(\%info);
208       print "\\svnexternalfile";
209       print "[$group]" if $group;
210       print "{$file}\n";
211       print "\n"
212     }
213
214     $lastgroup = $group;
215   }
216   print "\n";
217   close ($svxfh);
218 }
```

`svnid` Generates `\svnid` macro lines. Awaits a hash with the information received from `svn`. The `$` sign is masked to avoid keyword extension by Subversion inside this source file. Additional modules are needed to produce the date format used by `$Id$`.

```
219 sub svnid {
220   use Date::Parse;
221   use Date::Format;
222   my $href = shift;
223   return "" if (not defined $href->{Name});
224   my $date = time2str("%Y-%m-%d %XZ", str2time($href->{'Last Changed Date'}), 'Z');
225   return <<"EOT";
226 \\svnid{${dollar}Id: $href->{Name} $href->{'Last Changed Rev'} $date $href->{'Last Changed Aut
227 EOT
228 }
```

`svnidlong` Generates `\svnidlong` macro lines. Awaits a hash with the information received from `svn`. The `$` sign is masked to avoid keyword extension by Subversion inside this source file.

```
sub svnidlong {
  my $href = shift;
  return <<"EOT";
\\svnidlong
{${dollar}HeadURL: $href->{URL} \$}
{${dollar}LastChangedDate: $href->{'Last Changed Date'} \$}
{${dollar}LastChangedRevision: $href->{'Last Changed Rev'} \$}
{${dollar}LastChangedBy: $href->{'Last Changed Author'} \$}
EOT
}
```

`read fls` Reads the `.fls` file and looks for `INPUT relativedir/file` lines. The file is ignored if its extension is in the `EXCLUDE` list.

```
sub read_fls {
  my $fls = shift;
  my %stack;
  open (my $fh, '<', $fls) or return;
  while (<$fh>) {
    chomp;
    if (/^INPUT ([^\/].*)$/) {
      my $file = $1;
      my $ext = substr($file, rindex($file,'.')+1);
      $stack{$1} = 1 if not exists $EXCLUDE{$ext};
    }
  }
  close($fh);
  return keys %stack;
}
```

`usage` Prints usage information.

```
sub usage {
  print STDOUT <<'EOT';
Usage:
 svn-multi.pl jobname[.tex] [--fls] [--group|-g <group name>] [input_files] ...
 ... [--group|-g <group name>] [input_files] ...

Description:
 This LaTeX helper script collects Subversion keywords from non-(La)TeX files
 and provides it to the 'svn-multi' package using '.svx' files.  It will first
 scan the file '<jobname>.aux' for files declared by the '\svnextern' macro but
 also allows to provide additional files including the corresponding groups. The
 keywords for the additional files will be written in the file '<jobname>.svx'.

Options:
 jobname[.tex] : The LaTeX `jobname`, i.e. the basename of your main LaTeX file.
 --group <GN>  : Use given group name <GN> for all following files,
 or -g <GN>      including the one read by a '--fls' option, until the next
                 group is specified.
 --fls  : Read list of (additional) files from the file '<jobname>.fls'. This
          file is produced by LaTeX when run with the '--recorder' option and
```

```
274           contains a list of all input and output files used by the LaTeX main
275           file. Only input files with a relative path will be used.  A
276           previously selected group will be honoured.
277
278 Examples:
279 The main LaTeX file here is 'mymainlatexfile.tex'.
280
281  svn-multi.pl mymainlatexfile
282     Creates Subversion keywords for all files declared by '\svnextern' inside
283     the LaTeX code.
284
285  svn-multi.pl mymainlatexfile --group=FLS --fls
286     Creates Subversion keywords for all files declared by '\svnextern' inside
287     the LaTeX code. In addition it does the same for all relative input files
288     mentioned in the .fls file which are placed in the 'FLS' group.
289
290  svn-multi.pl mymainlatexfile a b c --group=B e d f
291     In addition to the '\svnextern' declared files the keywords for the files
292     'a', 'b' and 'c' will be added without a specific group, i.e. the last group
293     specified in the LaTeX file before the '\svnextern' macro will be used. The
294     keywords for 'e', 'd', 'f' will be part of group 'B'.
295
296  svn-multi.pl mymainlatexfile --group=A a --group=B b --group='' c
297     File 'a' is in group 'A', 'b' is in 'B' and 'c' is not in any group.
298
299 Further Information:
300 See the svn-multi package manual for more information about this script.
301 EOT
302   exit(0);
303 }
```

End of File

```
304 __END__
```