

# The `svn-multi.pl` Script

Martin Scharrer

[martin@scharrer-online.de](mailto: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.

```
1 use strict;
2 use warnings;
3 use File::Basename;
```

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

```
4 my $VERSION = "0.2";
5 my ($REV,$DATE) =
6   (split ' ', 'Id: svn-multi-pl.dtx 1873 2010-07-26 15:31:45Z martin $')[2,3];
```

Declaration of other global constants and variables:

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

Subfunction prototypes:

```
10 sub create_svxfile ($0);
11 sub usage;
```

Check if help was requested:

```
12 if (!@ARGV or grep { $_ eq '--help' or $_ eq '-h' } @ARGV) {
13   usage();
14 }
```

Print identification line like other TeX related programs:

```
15 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.

```
16 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.

```
17 if ($dir && $dir ne './') {
18   printf STDOUT "Main directory is '$dir'.\n";
19   chdir($dir);
20 }

21
22 if ($jobname =~ /\~/) {
23   usage();
24 }
25 my $outfile = "$jobname.svx";
26
27 my %external;
28
```

Regular expressions to read the .aux file:

**svnexternalpath:** The format is:

`\@svnexternalpath{{patha}{pathb}{...}{pathd}}`

```
29 my $resvnexternalpath = qr/
30   ^                                # at begin of line
31   \s*                              # allow for spaces
32   \\@svnexternalpath              # the macro name
33   \s*
34   {                                # begin token group
35   \s*
36   (? :                             # paths:
37     {                               # { of first path
38       (.*?)                         # everything else, e.g: 'patha}{pathb}{pathc'
39     }                               # } of last path
40     |                               # or nothing
41   )
42   \s*
43   }                                # end token group
44   \s*
45   $                                # end of line
46   /x;
```

**svnexternal:** The format is:

`\@svnexternal[group name]{{filea}{fileb}{...}{filed}}`

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

```

49 \s*          # allow for spaces
50   \\@svnexternal # the macro name
51 \s*
52   (?:        # optional:
53     \[        # opening [
54     ([^\]]*)  # group name (everything until ])
55     \]        # closing ]
56   )?
57 \s*
58   {          # begin token group
59     ([^}]+)  # file name (everything until })
60   }          # end token group
61 \s*
62   {          # begin token group
63 \s*
64   (?:        # paths:
65     {        # { of first file
66     (.*?)    # everything else, e.g: 'filea}{fileb}{filec'
67     }        # } of last file
68     |        # or nothing
69   )
70 \s*
71   }          # end token group
72 \s*
73   $          # end of line
74 /x;

75 if (-e "$jobname.aux" and open( my $svnfh, '<', "$jobname.aux")) {
76   print STDOUT "Reading '$jobname.aux'.\n";
77   while (<$svnfh>) {
78     chomp;
79     if (/ $resvnexternalpath/) {
80       push @PATH, ( split /\s*{/ , $1 );
81     }
82     elsif (/ $resvnexternal/) {
83       my ($group,$file,$list) = ($1||"",$2,$3||"");
84       $file =~ s/^\.\.//;
85       push @{$external{$file}{$group}} ||= [], ( split /\s*{/ , $list );
86     }
87   }
88   close ($svnfh);
89 }
90 else {
91   warn "No .aux file found for '$jobname'!\n";
92 }
93
94 # Add TEXINPUTS to path
95 push @PATH, map { $_ =~ s/(?!\/)\/\./; $_ } grep { $_ }
96   split(':', $ENV{'TEXINPUTS'}||"");
97

```

```

98 my @mainfilepairs;
99 my $maintex = "$jobname.tex";
100 if (exists $external{$maintex}) {
101   while ( my ($group,$list) = each %{$external{$maintex}} ) {
102     push @mainfilepairs, [ $group, [ @$list ] ];
103   }
104   delete $external{$maintex};
105 }
106
107 push @mainfilepairs, parse_args(@ARGV);
108 create_svxfile("$jobname.svx", @mainfilepairs )
109   if @mainfilepairs;
110
111 foreach my $file (keys %external) {
112   my @pairs;
113   my $svxfile = $file;
114   $svxfile =~ s/\.(tex|ltx)$/.svx/;
115   while ( my ($group,$list) = each %{$external{$file}} ) {
116     push @pairs, [ $group, [ @$list ] ];
117   }
118   create_svxfile($svxfile, @pairs);
119 }

```

## 2.2 Functions

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

```

120 sub parse_args {
121   my @args = @_;
122   my $group = '';
123   my @files;
124   my $readfg;
125   my @pairs;
126
127   foreach my $arg (@args) {
128     if ($readfg) {
129       $readfg = 0;
130       $group = $arg;
131       $group =~ s/^[\'"]|[\']$//; # '
132     }
133     elsif ($arg =~ /^--group|^--fg/) {
134       push @pairs, [ $group, [ @files ] ];
135       @files = ();
136       if ($arg =~ /^--group=(.*)/) {
137         $group = $1;
138         $group =~ s/^[\'"]|[\']$//; # '
139       }
140     }
141     else {
142       $readfg = 1;

```

```

143     }
144     elsif ($arg =~ /^--fls/) {
145         push @files, read_fls("$jobname.flc");
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 one 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 }

```

**create 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 $lastgroup) ) {
185             print "\nsvngroup{$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.

```
229 sub svnidlone {
230   my $href = shift;
231   return <<"EOT";
232 }
233 {${dollar}HeadURL: $href->{URL} \${}
234 {${dollar}LastChangedDate: $href->{'Last Changed Date'} \${}
235 {${dollar}LastChangedRevision: $href->{'Last Changed Rev'} \${}
236 {${dollar}LastChangedBy: $href->{'Last Changed Author'} \${}
237 EOT
238 }
```

**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.

```
239 sub read_fls {
240   my $fls = shift;
241   my %stack;
242   open (my $fh, '<', $fls) or return;
243   while (<$fh>) {
244     chomp;
245     if (/^INPUT ([^\./].*)$/) {
246       my $file = $1;
247       my $ext = substr($file, rindex($file, '.')+1);
248       $stack{$1} = 1 if not exists $EXCLUDE{$ext};
249     }
250   }
251   close($fh);
252   return keys %stack;
253 }
```

**usage** Prints usage information.

```
254 sub usage {
255   print STDOUT <<'EOT';
256 Usage:
257 svn-multi.pl jobname[.tex] [--fls] [--group|-g <group name>] [input_files] ...
258 ... [--group|-g <group name>] [input_files] ...
259
260 Description:
261 This LaTeX helper script collects Subversion keywords from non-(La)TeX files
262 and provides it to the 'svn-multi' package using '.svx' files. It will first
263 scan the file '<jobname>.aux' for files declared by the '\svnextern' macro but
264 also allows to provide additional files including the corresponding groups. The
265 keywords for the additional files will be written in the file '<jobname>.svx'.
266
267 Options:
268 jobname[.tex] : The LaTeX 'jobname', i.e. the basename of your main LaTeX file.
269 --group <GN>  : Use given group name <GN> for all following files,
270 or -g <GN>    including the one read by a '--fls' option, until the next
271                group is specified.
```

```

272 --fls : Read list of (additional) files from the file '<jobname>.fls'. This
273         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__

```