```
**********************************************************
   15834 Fri Jan 31 20:00:52 2014
new/usr/src/tools/scripts/webrev.1
3810 remove support for teamware from webrev
**********************************************************
```

```
.\"
.\" CDDL HEADER START
.\"
.\" The contents of this file are subject to the terms of the
.\" Common Development and Distribution License (the "License").
.\" You may not use this file except in compliance with the License.
.\"
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
.\" or http://www.opensolaris.org/os/licensing.
.\" See the License for the specific language governing permissions
.\" and limitations under the License.
.\"
.\" When distributing Covered Code, include this CDDL HEADER in each
.\" file and include the License file at usr/src/OPENSOLARIS.LICENSE.
.\" If applicable, add the following below this CDDL HEADER, with the
.\" fields enclosed by brackets "[]" replaced with your own identifying
.\" information: Portions Copyright [yyyy] [name of copyright owner]
.\"
.\" CDDL HEADER END
.\"
.\" Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
.\" Use is subject to license terms.
.\"
.\"
.TH webrev 1 "6 Dec 2010"
.SH NAME
webrev \- Generate HTML codereview materials
.SH SYNOPSIS
.B webrev
[
.I common-options
]

.B webrev
[
.I common-options
]
.I file-list-file
|
.I -

.B webrev
[
.I common-options
]
.B -w
.I wx-file

.B webrev
[
.I common-options
]
.B -l
.I arguments to 'putback'

See OPTIONS for common-options.
Note that the -l option is only applicable to TeamWare workspaces.

.SH DESCRIPTION
.B webrev
builds a set of HTML files suitable for performing code review of
```



```
source changes in a web browser.
It supports Mercurial, Git and Subversion repositories.
It supports Mercurial, Git, Subversion and Teamware repositories.
At its most basic, usage is:
.nf
        $ webrev
.fi

In which case \fBwebrev\fR attempts to figure out the list of files
for review.  If that fails, or if more control
for review (note that when using Teamware \fBputback\fR, this may take
a long time; see the -l option).  If that fails, or if more control
over the set of files is needed, a \fIfile list\fR may be specified.
\fBwebrev\fR also attempts to deduce a
.I basis for comparison
(interchangeably called the \fIparent\fR, but see SCM INTERACTIONS below).
A basis for comparison is needed in order to determine the differences
introduced by the code changes under review.

By default, \fBwebrev\fR creates a \fIwebrev\fR directory in the
workspace directory that contains the generated HTML files, a generated
PDF review, and a patch representing the changes.  It also places a
copy of the file list in that directory, and of both the old and new
raw files in the \fB$webrev_root/raw_files\fR directory.
To output the webrev somewhere other than the default location, use the
\fI-o <outdir>\fR option, or set the \fBWDIR\fR environment variable.
For example:
.nf
        $ webrev -o ~/public_html/myreview/
.fi
.PP
In the index file, each file is listed on a line with a link to the
relevant review materials.  Comments for each change will be included
automatically.  Cross references to bug (or other information) tracking
databases in the comments will become hyperlinks in the associated web
interface, according to the rules in CROSS REFERENCING below.

As a review aid, content may be added to the \fIindex\fR file in two ways.
First, the author may manually edit the file (for example by including
text that explains the changes in front of the links for each file).
Note that if webrev is run again, manual edits will be lost.  Second,
if a file named \fIwebrev-info\fR is present at the root of the workspace,
it will be automatically included in the \fIindex\fR file.  To include a
different file, see the \fI-i\fR option.

For each file in the file list, \fBwebrev\fR compares the file with the
version in the basis for comparison (i.e. the parent workspace) and
generates a variety of HTML renderings of the differences between
the two files; which of these renderings to use is largely a matter
of personal preference.  Additional, webrev emits a patch, the old
and new versions of the file, and a "raw" copy of the file which is
suitable for download.  For files which express differences, source
is formatted according to the following color coding:
.IP
.nf
     unchanged : black
       removed : brown
       changed : blue
           new : bold blue
.fi

.SH SCM INTERACTIONS
.PP
.B webrev
attempts to interact with the source code management system currently in use.
.B webrev
```

```
needs to be able locate the code under review (i.e. the workspace) and
the basis for comparison (i.e. the parent).  The method for doing so
depends upon the SCM in use, which
.B webrev
will also attempt to auto-discover.  In all cases,
.B webrev
must either discover the list of files which have changed, or else this list
must be manually specified, either in "webrev file list" format or in "wx"
format.
See FILE LIST for more details.
.PP
In all cases, if the user has activated the workspace with the
.BR ws (1)
or
.BR bldenv (1)
commands, \fBwebrev\fR will use the \fBCODEMGR_PARENT\fR and
\fBCODEMGR_WS\fR environment variables to identify parent and child
workspaces respectively.
To manually specify the basis for comparison, use the -p option or
specify the \fBCODEMGR_PARENT\fR variable in either the file list or
the environment.

.SS Discovering the SCM in use.
.B webrev
makes use of
.BR which_scm (1)
to determine the SCM in use for a given workspace.

.SS TeamWare
In the case of TeamWare \fBwebrev\fR will use the output of "workspace
name" to discover the workspace root, if not otherwise specified.
.PP
\fBwebrev\fR will attempt to use a
.BR wx (1)
active list in
\fBCODEMGR_WS/wx/active\fR.
To direct \fBwebrev\fR to determine the file list from the output of
.BR putback "(1),"
use the -l option.  (Note that \fBwebrev\fR may also elect to use
\fBputback\fR if it cannot determine the file list from
.BR wx "(1))."
The -l option indicates that subsequent arguments should be
treated as arguments to
.BR putback "(1)."
This can be used to prune the set of files which putback examines,
or to reference a teamware flp (file list program).

.SS Mercurial
In the case of Mercurial \fBwebrev\fR will attempt to use the output
from the
.BR hg (1)
"hg root" command to identify the workspace root, and the
"hg path default" command to identify the parent workspace.

.SS Git
In the case of Git \fBwebrev\fR will attempt to use the output from the
.BR git (1)
"git rev-parse --git-dir" command to identify the workspace root, and will
attempt to use the remote branch which the current branch is tracking as the
parent, if none is specified 'origin/master' will be used.

The parent specified when using git is, in all cases, a git 'tree-ish' and
never an actual git repository, remote or otherwise.  Anything specifiable to
git as a tree-ish should, similarly, be specifiable as a parent for webrev.
This includes branches, explicit revisions, reflog entries, etc. See
.BR git-rev-parse (1)
```



```
.SS Subversion
In the case of Subversion \fBwebrev\fR will attempt to use the output
from the
.BR svn (1)
"svn info" to find the workspace root and subversion repository URL.
.PP
The file list will be created from the output of the "svn status" command.

.SH CROSS REFERENCING
.PP
After extracting comments (see FILE LIST below),
.B webrev
will translate cross references into hyperlinks.  By default, information
about available information tracking systems can be found in
/opt/onbld/etc/its.reg, and the specification of a local domain and
selection and prioritization of systems
in /opt/onbld/etc/its.conf.  These file formats are self documenting.  Also
see the -I and -C options below.
.SH OPTIONS
.TP 10
.BI "-C " priority-file
In addition to the system default and an optional user-supplied ~/.its.conf,
use the specified file to specify a local domain list and prioritize the list
of information tracking systems to be searched automatically when resolving cros
references.
.TP 10
.BI "-D"
Delete remote webrev via SFTP. Default remote host is \fIcr.opensolaris.org\fR,
default remote directory for removal is the same as workspace/repository
basename. Remote target can be overriden using -t option. If combined with
-U the deletion will be performed first. Also, if used together with -U
and the removal fails, no upload is done. Without -U option no webrev will
be generated, just like if -n option was used. The deletion is done by
moving the webrev to special directory in user's home directory. It is
expected that the remote host periodically runs a script which deletes
the contents of this directory. See the ENVIRONMENT VARIABLES section for
more details about this directory.
.TP 10
.BI "-I " information-file
Use the specified file to seed the list of information tracking systems.
.TP 10
.BI "-i " include-file
Include the specified file into the index.html file which is generated
as part of the webrev.  This allows a snippet of XHTML to be added by
the webrev author. User content is contained by a <div> tag and
the markup should validate as XHTML 1.0 Transitional.
.TP 10
.BI "-l " putback-args
Extract the file list from the output of
.I putback -n.
Any arguments supplied will be passed to
.BR putback "(1)."
See SCM INTERACTIONS.  For more information about file
lists, see FILE LIST.  This argument should appear last.
.TP 10
.BI "-N"
Suppress all comments from all output forms html, txt and pdf.
.TP 10
.BI "-n"
Do not generate webrev. Useful whenever only upload is needed.
.TP 10
.B -O
Enable \fIOpenSolaris\fR mode: information tracking system hyperlinks
are generated using the EXTERNAL_URL field from the specified its.reg entry,
instead of the default INTERNAL_URL_domain field, and sources which appear in
```

```
220 \fIusr/closed\fR are automatically elided from the review.
221 .TP 10
222 .BI "-o " output-dir
223 Place output from running the script in the directory specified.  If
224 specified, this option takes precedence over the WDIR environment variable.
225 .TP 10
226 .BI "-p " basis-of-comparison
227 Specify a basis of comparison meaningful for the SCM currently in use.
228 See SCM INTERACTIONS and INCREMENTAL REVIEWS.
229 .TP 10
230 .BI "-t " target
231 Upload target. Specified in form of URI identifier. For SCP/SFTP it is
232 \fIssh://user@remote_host:remote_dir\fR and for rsync it is
233 \fIrsync://user@remote_host:remote_dir\fR. This option can override the
234 -o option if the URI is fully specified. The target is relative to
235 the top level directory of the default sftp/rsync directory tree.
236 .TP 10
237 .BI "-U"
238 Upload the webrev. Default remote host is \fIcr.opensolaris.org\fR.
239 Default transport is rsync. If it fails, fallback to SCP/SFTP transport
240 is done.
241 .TP 10
242 .BI "-w " wx-file
243 Extract the file list from the wx "active" file specified.  'wx' uses
244 this mode when invoking webrev.  The list is assumed to be in the
245 format expected by the \fIwx\fR package.  See FILE LIST, below.

247 .SH FILE LIST
248 .PP
249 .B Webrev
250 needs to be told or to discover which files have changed in a
251 given workspace.  By default,
252 .B webrev
253 will attempt to autodetect the
254 list of changed files by first consulting
255 .BR wx "(1)."
256 If this information is not available, webrev tries to consult the SCM (Source
257 Code Manager) currently in use.  If that fails, the user must intervene by
258 specifying either a file list or additional options specific to the SCM in use.

260 .SS Webrev Format
261 A webrev formatted file list contains a list of all the files to
262 be included in the review with paths relative to the workspace
263 directory, e.g.
264 .IP
265 .nf
266 \f(CWusr/src/uts/common/fs/nfs/nfs_subr.c
267 usr/src/uts/common/fs/nfs/nfs_export.c
268 usr/src/cmd/fs.d/nfs/mountd/mountd.c
269 .fi
270 .PP
271 Include the paths of any files added, deleted, or modified.
272 You can keep this list of files in the webrev directory
273 that webrev creates in the workspace directory
274 (CODEMGR_WS).

276 If CODEMGR_WS is not set, it may be specified as an environment variable
277 within the file list, e.g.
278 .IP
279 .nf
280 \f(CWCODEMGR_WS=/home/brent/myws
281 usr/src/uts/common/fs/nfs/nfs_subr.c
282 usr/src/uts/common/fs/nfs/nfs_export.c
283 usr/src/cmd/fs.d/nfs/mountd/mountd.c
284 .fi
285 .PP
```



```
286 To compare the workspace against one other than the parent (see also
287 the -p option), include a CODEMGR_PARENT line in the file list, like:
288 .IP
289 .nf
290 \f(CWCODEMGR_WS=/home/brent/myws
291 CODEMGR_PARENT=/ws/onnv-gate
292 usr/src/uts/common/fs/nfs/nfs_subr.c
293 usr/src/uts/common/fs/nfs/nfs_export.c
294 usr/src/cmd/fs.d/nfs/mountd/mountd.c
295 .fi
296 .PP
297 Finally, run webrev with the name of the file containing the file list as an
298 argument, e.g.
299 .nf
300         $ webrev file.list
301 .fi
302 .PP
303 If "-" is supplied as the name of the file, then stdin will be used.

305 .SS wx Format
306 If the \fI-w\fR flag is specified then \fBwebrev\fR
307 will assume the file list is in the format expected by the "wx" package:
308 pathname lines alternating with SCCS comment lines separated by blank
309 lines, e.g.
310 .IP
311 .nf
312 \f(CWusr/src/uts/common/fs/nfs/nfs_subr.c

314 1206578 Fix spelling error in comment

316 usr/src/uts/common/fs/nfs/nfs_export.c

318 4039272 cstyle fixes

320 usr/src/cmd/fs.d/nfs/mountd/mountd.c

322 1927634 mountd daemon doesn't handle expletives
323 .fi

325 .SH INCREMENTAL REVIEWS
326 When conducting multiple rounds of code review, it may be desirable to
327 generate a webrev which represents the delta between reviews.  In this
328 case, set the parent workspace to the path to the old webrev:

330 .IP
331 .nf
332 \f(CW$ webrev -o ~/public_html/myreview-rd2/ \\
333         -p ~/public_html/myreview/
334 .fi

336 .SH ENVIRONMENT VARIABLES
337 The following environment variables allow for customization of \fBwebrev\fR:

339 .PP
340 \fBCDIFFCMD\fR and \fBUDIFFCMD\fR are used when generating Cdiffs and Udiffs
341 respectively; their default values are "diff -b -C 5" and "diff -b -U
342 5".  To generate diffs with more (or less) than 5 lines of context or
343 with more (or less) strict whitespace handling, set one or both of
344 these variables in the user environment accordingly.

346 \fBWDIR\fR sets the output directory.  It is functionally equivalent to
347 the \fI-o\fR option.

349 \fBWDIFF\fR specifies the command used to generate Wdiffs. Wdiff generates a
350 full unified context listing with line numbers where unchanged
351 sections of code may be expanded and collapsed.  It also provides a
```

"split" feature that shows the same file in two HTML frames one above the other. The default path for this script is /ws/onnv-gate/public/bin/wdiff but WDIFF may be set to customize this to use a more convenient location.

\fBWEBREV_TRASH_DIR\fR specifies alternative location of trash directory for remote webrev deletion using the \fI-D\fR option. The directory is relative to the top level directory of the default sftp/rsync directory tree. The default value of this directory is ".trash".

.SH UPLOADING WEBREVS
A webrev can be uploaded to remote site using the -U option. To simply generate new webrev and upload it to the default remote host use the following command:
.IP
.nf
\f(CW$ webrev -U
.fi
.PP
This will generate the webrev to local directory named ’webrev’ and upload it to remote host with remote directory name equal to local workspace/repository name. To change both local and remote directory name, -U can be combined with -o option. The following command will store the webrev to local directory named "foo.onnv" and upload it to the remote host with the same directory name:
.IP
.nf
\f(CW$ webrev -U -o $CODEMGR_WS/foo.onnv
.fi
.PP
If there is a need for manual change of the webrev before uploading, -U can be combined with -n option so that first command will just generate the webrev and the second command will upload it without generating it again:
.IP
.nf
\f(CW$ webrev
\f(CW$ webrev -n -U
.fi
.PP
For custom remote targets, -t option allows to specify all components:
.IP
.nf
\f(CW$ webrev -U -t \\
        ssh://user@cr.opensolaris.org:foo/bar/bugfix.onnv
.fi
.PP
If the remote path is specified as absolute, \fBwebrev\fR will assume all the directories are already created. If the path is relative, \fBwebrev\fR will try to create all needed directories. This only works with SCP/SFTP transport.
.PP
By default, rsync transport will use SSH for transferring the data to remote site. To specify custom username, use entry in SSH client configuration file, for example:
.IP
.nf
\f(CWHost cr.opensolaris.org
  Hostname cr.opensolaris.org
  User vkotal
.fi

.SH DELETING WEBREVS
When deleting a webrev directory on remote site which has a different name than the basename of local repository it is necessary to specify the output option:
.IP
.nf
\f(CW$ webrev -Do webrev-foo.onnv

.fi
.PP
Otherwise \fBwebrev\fR will attempt to remove remote directory with the same name as basename of the local repository.
.PP
For the nested directory case it is necessary to specify the full target:
.IP
.nf
\f(CW$ webrev -D -t \\
        ssh://user@cr.opensolaris.org:foo/bar/bugfix.onnv
.fi
.PP
This will remove just the \fIbugfix.onnv\fR directory.

.SH SEE ALSO
.BR putback "(1),"
.BR workspace "(1),"
.BR hg "(1),"
.BR git "(1),"
.BR ssh_config "(4),"
.BR svn "(1),"
.BR which_scm "(1)"

.SH ACKNOWLEDGEMENTS
Acknowledgements to Rob Thurlow, Mike Eisler, Lin Ling, Rod Evans, Mike Kupfer, Greg Onufer, Glenn Skinner, Oleg Larin, David Robinson, Matthew Cross, David L. Paktor, Neal Gafter, John Beck, Darren Moffat, Norm Shulman, Bill Watson, Pedro Rubio and Bill Shannon for valuable feedback and insight in building webrev.

Have fun!
.br
Brent Callaghan 11/28/96

```
**********************************************************
   83920 Fri Jan 31 20:00:52 2014
new/usr/src/tools/scripts/webrev.sh
3810 remove support for teamware from webrev
**********************************************************
```

```
#!/usr/bin/ksh93 -p
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2002, 2010, Oracle and/or its affiliates. All rights reserved.
#

# Copyright 2008, 2010, Richard Lowe
# Copyright 2012 Marcel Telka <marcel@telka.sk>
# Copyright 2014 Bart Coddens <bart.coddens@gmail.com>
#endif /* ! codereview */

#
# This script takes a file list and a workspace and builds a set of html files
# suitable for doing a code review of source changes via a web page.
# Documentation is available via the manual page, webrev.1, or just
# type 'webrev -h'.
#
# Acknowledgements to contributors to webrev are listed in the webrev(1)
# man page.
#

REMOVED_COLOR=brown
CHANGED_COLOR=blue
NEW_COLOR=blue

HTML='<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">\n'

FRAMEHTML='<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">\n'

STDHEAD='<meta http-equiv="cache-control" content="no-cache"></meta>
<meta http-equiv="Pragma" content="no-cache"></meta>
<meta http-equiv="Expires" content="-1"></meta>
<!--
   Note to customizers: the body of the webrev is IDed as SUNWwebrev
   to allow easy overriding by users of webrev via the userContent.css
```



```
   mechanism available in some browsers.

   For example, to have all "removed" information be red instead of
   brown, set a rule in your userContent.css file like:

       body#SUNWwebrev span.removed { color: red ! important; }
-->
<style type="text/css" media="screen">
body {
    background-color: #eeeeee;
}
hr {
    border: none 0;
    border-top: 1px solid #aaa;
    height: 1px;
}
div.summary {
    font-size: .8em;
    border-bottom: 1px solid #aaa;
    padding-left: 1em;
    padding-right: 1em;
}
div.summary h2 {
    margin-bottom: 0.3em;
}
div.summary table th {
    text-align: right;
    vertical-align: top;
    white-space: nowrap;
}
span.lineschanged {
    font-size: 0.7em;
}
span.oldmarker {
    color: red;
    font-size: large;
    font-weight: bold;
}
span.newmarker {
    color: green;
    font-size: large;
    font-weight: bold;
}
span.removed {
    color: brown;
}
span.changed {
    color: blue;
}
span.new {
    color: blue;
    font-weight: bold;
}
span.chmod {
    font-size: 0.7em;
    color: #db7800;
}
a.print { font-size: x-small; }
a:hover { background-color: #ffcc99; }
</style>

<style type="text/css" media="print">
pre { font-size: 0.8em; font-family: courier, monospace; }
span.removed { color: #444; font-style: italic }
span.changed { font-weight: bold; }
span.new { font-weight: bold; }
```

```
span.newmarker { font-size: 1.2em; font-weight: bold; }
span.oldmarker { font-size: 1.2em; font-weight: bold; }
a.print {display: none}
hr { border: none 0; border-top: 1px solid #aaa; height: 1px; }
</style>
'

#
# UDiffs need a slightly different CSS rule for 'new' items (we don't
# want them to be bolded as we do in cdiffs or sdiffs).
#
UDIFFCSS='
<style type="text/css" media="screen">
span.new {
    color: blue;
    font-weight: normal;
}
</style>
'

#
# Display remote target with prefix and trailing slash.
#
function print_upload_header
{
        typeset -r prefix=$1
        typeset display_target

        if [[ -z $tflag ]]; then
                display_target=${prefix}${remote_target}
        else
                display_target=${remote_target}
        fi

        if [[ ${display_target} != */ ]]; then
                display_target=${display_target}/
        fi

        print "      Upload to: ${display_target}\n" \
            "     Uploading: \c"
}

#
# Upload the webrev via rsync. Return 0 on success, 1 on error.
#
function rsync_upload
{
        if (( $# != 2 )); then
                print "\nERROR: rsync_upload: wrong usage ($#)"
                exit 1
        fi

        typeset -r dst=$1
        integer -r print_err_msg=$2

        print_upload_header ${rsync_prefix}
        print "rsync ... \c"
        typeset -r err_msg=$( $MKTEMP /tmp/rsync_err.XXXXXX )
        if [[ -z $err_msg ]]; then
                print "\nERROR: rsync_upload: cannot create temporary file"
                return 1
        fi
        #
        # The source directory must end with a slash in order to copy just
        # directory contents, not the whole directory.
        #
```



```
        typeset src_dir=$WDIR
        if [[ ${src_dir} != */ ]]; then
                src_dir=${src_dir}/
        fi
        $RSYNC -r -q ${src_dir} $dst 2>$err_msg
        if (( $? != 0 )); then
                if (( ${print_err_msg} > 0 )); then
                        print "Failed.\nERROR: rsync failed"
                        print "src dir: '${src_dir}'\ndst dir: '$dst'"
                        print "error messages:"
                        $SED 's/^/> /' $err_msg
                        rm -f $err_msg
                fi
                return 1
        fi

        rm -f $err_msg
        print "Done."
        return 0
}

#
# Create directories on remote host using SFTP. Return 0 on success,
# 1 on failure.
#
function remote_mkdirs
{
        typeset -r dir_spec=$1
        typeset -r host_spec=$2

        #
        # If the supplied path is absolute we assume all directories are
        # created, otherwise try to create all directories in the path
        # except the last one which will be created by scp.
        #
        if [[ "${dir_spec}" == */* && "${dir_spec}" != /* ]]; then
                print "mkdirs \c"
                #
                # Remove the last directory from directory specification.
                #
                typeset -r dirs_mk=${dir_spec%/*}
                typeset -r batch_file_mkdir=$( $MKTEMP \
                    /tmp/webrev_mkdir.XXXXXX )
                if [[ -z $batch_file_mkdir ]]; then
                        print "\nERROR: remote_mkdirs:" \
                            "cannot create temporary file for batch file"
                        return 1
                fi
                OLDIFS=$IFS
                IFS=/
                typeset dir
                for dir in ${dirs_mk}; do
                        #
                        # Use the '-' prefix to ignore mkdir errors in order
                        # to avoid an error in case the directory already
                        # exists. We check the directory with chdir to be sure
                        # there is one.
                        #
                        print -- "-mkdir ${dir}" >> ${batch_file_mkdir}
                        print "chdir ${dir}" >> ${batch_file_mkdir}
                done
                IFS=$OLDIFS
                typeset -r sftp_err_msg=$( $MKTEMP /tmp/webrev_scp_err.XXXXXX )
                if [[ -z ${sftp_err_msg} ]]; then
                        print "\nERROR: remote_mkdirs:" \
                            "cannot create temporary file for error messages"
```

```
260                         return 1
261                 fi
262                 $SFTP -b ${batch_file_mkdir} ${host_spec} 2>${sftp_err_msg} 1>&2
263                 if (( $? != 0 )); then
264                         print "\nERROR: failed to create remote directories"
265                         print "error messages:"
266                         $SED 's/^/> /' ${sftp_err_msg}
267                         rm -f ${sftp_err_msg} ${batch_file_mkdir}
268                         return 1
269                 fi
270                 rm -f ${sftp_err_msg} ${batch_file_mkdir}
271         fi

273         return 0
274 }

276 #
277 # Upload the webrev via SSH. Return 0 on success, 1 on error.
278 #
279 function ssh_upload
280 {
281         if (( $# != 1 )); then
282                 print "\nERROR: ssh_upload: wrong number of arguments"
283                 exit 1
284         fi

286         typeset dst=$1
287         typeset -r host_spec=${dst%%:*}
288         typeset -r dir_spec=${dst#*:}

290         #
291         # Display the upload information before calling delete_webrev
292         # because it will also print its progress.
293         #
294         print_upload_header ${ssh_prefix}

296         #
297         # If the deletion was explicitly requested there is no need
298         # to perform it again.
299         #
300         if [[ -z $Dflag ]]; then
301                 #
302                 # We do not care about return value because this might be
303                 # the first time this directory is uploaded.
304                 #
305                 delete_webrev 0
306         fi

308         #
309         # Create remote directories. Any error reporting will be done
310         # in remote_mkdirs function.
311         #
312         remote_mkdirs ${dir_spec} ${host_spec}
313         if (( $? != 0 )); then
314                 return 1
315         fi

317         print "upload ... \c"
318         typeset -r scp_err_msg=$( $MKTEMP /tmp/scp_err.XXXXXX )
319         if [[ -z ${scp_err_msg} ]]; then
320                 print "\nERROR: ssh_upload:" \
321                     "cannot create temporary file for error messages"
322                 return 1
323         fi
324         $SCP -q -C -B -o PreferredAuthentications=publickey -r \
325                 $WDIR $dst 2>${scp_err_msg}
```



```
326         if (( $? != 0 )); then
327                 print "Failed.\nERROR: scp failed"
328                 print "src dir: '$WDIR'\ndst dir: '$dst'"
329                 print "error messages:"
330                 $SED 's/^/> /' ${scp_err_msg}
331                 rm -f ${scp_err_msg}
332                 return 1
333         fi

335         rm -f ${scp_err_msg}
336         print "Done."
337         return 0
338 }

340 #
341 # Delete webrev at remote site. Return 0 on success, 1 or exit code from sftp
342 # on failure. If first argument is 1 then perform the check of sftp return
343 # value otherwise ignore it. If second argument is present it means this run
344 # only performs deletion.
345 #
346 function delete_webrev
347 {
348         if (( $# < 1 )); then
349                 print "delete_webrev: wrong number of arguments"
350                 exit 1
351         fi

353         integer -r check=$1
354         integer delete_only=0
355         if (( $# == 2 )); then
356                 delete_only=1
357         fi

359         #
360         # Strip the transport specification part of remote target first.
361         #
362         typeset -r stripped_target=${remote_target##*://}
363         typeset -r host_spec=${stripped_target%%:*}
364         typeset -r dir_spec=${stripped_target#*:}
365         typeset dir_rm

367         #
368         # Do not accept an absolute path.
369         #
370         if [[ ${dir_spec} == /* ]]; then
371                 return 1
372         fi

374         #
375         # Strip the ending slash.
376         #
377         if [[ ${dir_spec} == */ ]]; then
378                 dir_rm=${dir_spec%%/}
379         else
380                 dir_rm=${dir_spec}
381         fi

383         if (( ${delete_only} > 0 )); then
384                 print "       Removing: \c"
385         else
386                 print "rmdir \c"
387         fi
388         if [[ -z "$dir_rm" ]]; then
389                 print "\nERROR: empty directory for removal"
390                 return 1
391         fi
```

```
393         #
394         # Prepare batch file.
395         #
396         typeset -r batch_file_rm=$( $MKTEMP /tmp/webrev_remove.XXXXXX )
397         if [[ -z $batch_file_rm ]]; then
398                 print "\nERROR: delete_webrev: cannot create temporary file"
399                 return 1
400         fi
401         print "rename $dir_rm $TRASH_DIR/removed.$$" > $batch_file_rm

403         #
404         # Perform remote deletion and remove the batch file.
405         #
406         typeset -r sftp_err_msg=$( $MKTEMP /tmp/webrev_scp_err.XXXXXX )
407         if [[ -z ${sftp_err_msg} ]]; then
408                 print "\nERROR: delete_webrev:" \
409                     "cannot create temporary file for error messages"
410                 return 1
411         fi
412         $SFTP -b $batch_file_rm $host_spec 2>${sftp_err_msg} 1>&2
413         integer -r ret=$?
414         rm -f $batch_file_rm
415         if (( $ret != 0 && $check > 0 )); then
416                 print "Failed.\nERROR: failed to remove remote directories"
417                 print "error messages:"
418                 $SED 's/^/> /' ${sftp_err_msg}
419                 rm -f ${sftp_err_msg}
420                 return $ret
421         fi
422         rm -f ${sftp_err_msg}
423         if (( ${delete_only} > 0 )); then
424                 print "Done."
425         fi

427         return 0
428 }

430 #
431 # Upload webrev to remote site
432 #
433 function upload_webrev
434 {
435         integer ret

437         if [[ ! -d "$WDIR" ]]; then
438                 print "\nERROR: webrev directory '$WDIR' does not exist"
439                 return 1
440         fi

442         #
443         # Perform a late check to make sure we do not upload closed source
444         # to remote target when -n is used. If the user used custom remote
445         # target he probably knows what he is doing.
446         #
447         if [[ -n $nflag && -z $tflag ]]; then
448                 $FIND $WDIR -type d -name closed \
449                         | $GREP closed >/dev/null
450                 if (( $? == 0 )); then
451                         print "\nERROR: directory '$WDIR' contains" \
452                             "\"closed\" directory"
453                         return 1
454                 fi
455         fi
```



```
458         #
459         # We have the URI for remote destination now so let's start the upload.
460         #
461         if [[ -n $tflag ]]; then
462                 if [[ "${remote_target}" == ${rsync_prefix}?* ]]; then
463                         rsync_upload ${remote_target##$rsync_prefix} 1
464                         ret=$?
465                         return $ret
466                 elif [[ "${remote_target}" == ${ssh_prefix}?* ]]; then
467                         ssh_upload ${remote_target##$ssh_prefix}
468                         ret=$?
469                         return $ret
470                 fi
471         else
472                 #
473                 # Try rsync first and fallback to SSH in case it fails.
474                 #
475                 rsync_upload ${remote_target} 0
476                 ret=$?
477                 if (( $ret != 0 )); then
478                         print "Failed. (falling back to SSH)"
479                         ssh_upload ${remote_target}
480                         ret=$?
481                 fi
482                 return $ret
483         fi
484 }

486 #
487 # input_cmd | url_encode | output_cmd
488 #
489 # URL-encode (percent-encode) reserved characters as defined in RFC 3986.
490 #
491 # Reserved characters are: :/?#[]@!$&'()*+,;=
492 #
493 # While not a reserved character itself, percent '%' is reserved by definition
494 # so encode it first to avoid recursive transformation, and skip '/' which is
495 # a path delimiter.
496 #
497 # The quotation character is deliberately not escaped in order to make
498 # the substitution work with GNU sed.
499 #
500 function url_encode
501 {
502         $SED -e "s|%|%25|g" -e "s|:|%3A|g" -e "s|\&|%26|g" \
503             -e "s|?|%3F|g" -e "s|#|%23|g" -e "s|\[|%5B|g" \
504             -e "s|*|%2A|g" -e "s|@|%40|g" -e "s|\!|%21|g" \
505             -e "s|=|%3D|g" -e "s|;|%3B|g" -e "s|\]|%5D|g" \
506             -e "s|(|%28|g" -e "s|)|%29|g" -e "s|'|%27|g" \
507             -e "s|+|%2B|g" -e "s|\,|%2C|g" -e "s|\\\$|%24|g"
508 }

510 #
511 # input_cmd | html_quote | output_cmd
512 # or
513 # html_quote filename | output_cmd
514 #
515 # Make a piece of source code safe for display in an HTML <pre> block.
516 #
517 html_quote()
518 {
519         $SED -e "s/&/\&amp;/g" -e "s/</\&lt;/g" -e "s/>/\&gt;/g" "$@" | expand
520 }

522 #
523 # Trim a digest-style revision to a conventionally readable yet useful length
```

```
524 #
525 trim_digest()
526 {
527         typeset digest=$1
528
529         echo $digest | $SED -e 's/\([0-9a-f]\{12\}\).*/\1/'
530 }
531
532 #
533 # input_cmd | its2url | output_cmd
534 #
535 # Scan for information tracking system references and insert <a> links to the
536 # relevant databases.
537 #
538 its2url()
539 {
540         $SED -f ${its_sed_script}
541 }
542
543 #
544 # strip_unchanged <infile> | output_cmd
545 #
546 # Removes chunks of sdiff documents that have not changed. This makes it
547 # easier for a code reviewer to find the bits that have changed.
548 #
549 # Deleted lines of text are replaced by a horizontal rule. Some
550 # identical lines are retained before and after the changed lines to
551 # provide some context.  The number of these lines is controlled by the
552 # variable C in the $AWK script below.
553 #
554 # The script detects changed lines as any line that has a "<span class="
555 # string embedded (unchanged lines have no particular class and are not
556 # part of a <span>).  Blank lines (without a sequence number) are also
557 # detected since they flag lines that have been inserted or deleted.
558 #
559 strip_unchanged()
560 {
561         $AWK '
562         BEGIN   { C = c = 20 }
563         NF == 0 || /<span class="/ {
564                 if (c > C) {
565                         c -= C
566                         inx = 0
567                         if (c > C) {
568                                 print "\n</pre><hr></hr><pre>"
569                                 inx = c % C
570                                 c = C
571                         }
572
573                         for (i = 0; i < c; i++)
574                                 print ln[(inx + i) % C]
575                 }
576                 c = 0;
577                 print
578                 next
579         }
580         {       if (c >= C) {
581                         ln[c % C] = $0
582                         c++;
583                         next;
584                 }
585                 c++;
586                 print
587         }
588         END     { if (c > (C * 2)) print "\n</pre><hr></hr>" }
```



```
590         ' $1
591 }
592
593 #
594 # sdiff_to_html
595 #
596 # This function takes two files as arguments, obtains their diff, and
597 # processes the diff output to present the files as an HTML document with
598 # the files displayed side-by-side, differences shown in color.  It also
599 # takes a delta comment, rendered as an HTML snippet, as the third
600 # argument.  The function takes two files as arguments, then the name of
601 # file, the path, and the comment.  The HTML will be delivered on stdout,
602 # e.g.
603 #
604 #   $ sdiff_to_html old/usr/src/tools/scripts/webrev.sh \
605 #         new/usr/src/tools/scripts/webrev.sh \
606 #         webrev.sh usr/src/tools/scripts \
607 #         '<a href="http://monaco.sfbay.sun.com/detail.jsp?cr=1234567">
608 #          1234567</a> my bugid' > <file>.html
609 #
610 # framed_sdiff() is then called which creates $2.frames.html
611 # in the webrev tree.
612 #
613 # FYI: This function is rather unusual in its use of awk.  The initial
614 # diff run produces conventional diff output showing changed lines mixed
615 # with editing codes.  The changed lines are ignored - we're interested in
616 # the editing codes, e.g.
617 #
618 #      8c8
619 #      57a61
620 #      63c66,76
621 #      68,93d80
622 #      106d90
623 #      108,110d91
624 #
625 #  These editing codes are parsed by the awk script and used to generate
626 #  another awk script that generates HTML, e.g the above lines would turn
627 #  into something like this:
628 #
629 #      BEGIN { printf "<pre>\n" }
630 #      function sp(n) {for (i=0;i<n;i++)printf "\n"}
631 #      function wl(n) {printf "<font color=%s>%4d %s </font>\n", n, NR, $0}
632 #      NR==8           {wl("#7A7ADD");next}
633 #      NR==54          {wl("#7A7ADD");sp(3);next}
634 #      NR==56          {wl("#7A7ADD");next}
635 #      NR==57          {wl("black");printf "\n"; next}
636 #        :               :
637 #
638 #  This script is then run on the original source file to generate the
639 #  HTML that corresponds to the source file.
640 #
641 #  The two HTML files are then combined into a single piece of HTML that
642 #  uses an HTML table construct to present the files side by side.  You'll
643 #  notice that the changes are color-coded:
644 #
645 #   black     - unchanged lines
646 #   blue      - changed lines
647 #   bold blue - new lines
648 #   brown     - deleted lines
649 #
650 #  Blank lines are inserted in each file to keep unchanged lines in sync
651 #  (side-by-side).  This format is familiar to users of sdiff(1) or
652 #  Teamware's filemerge tool.
653 #
654 sdiff_to_html()
655 {
```

```
656         diff -b $1 $2 > /tmp/$$.diffs

658         TNAME=$3
659         TPATH=$4
660         COMMENT=$5

662         #
663         #  Now we have the diffs, generate the HTML for the old file.
664         #
665         $AWK '
666         BEGIN   {
667                 printf "function sp(n) {for (i=0;i<n;i++)printf \"\\n\"}\n"
668                 printf "function removed() "
669                 printf "{printf \"<span class=\\\"removed\\\">%%4d %%s</span>\\n
670                 printf "function changed() "
671                 printf "{printf \"<span class=\\\"changed\\\">%%4d %%s</span>\\n
672                 printf "function bl() {printf \"%%4d %%s\\n\", NR, $0}\n"
673 }
674         /^</    {next}
675         /^>/    {next}
676         /^---/  {next}

678         {
679         split($1, a, /[cad]/) ;
680         if (index($1, "a")) {
681                 if (a[1] == 0) {
682                         n = split(a[2], r, /,/);
683                         if (n == 1)
684                                 printf "BEGIN\t\t{sp(1)}\n"
685                         else
686                                 printf "BEGIN\t\t{sp(%d)}\n",\
687                                 (r[2] - r[1]) + 1
688                         next
689                 }

691                 printf "NR==%s\t\t{", a[1]
692                 n = split(a[2], r, /,/);
693                 s = r[1];
694                 if (n == 1)
695                         printf "bl();printf \"\\n\"; next}\n"
696                 else {
697                         n = r[2] - r[1]
698                         printf "bl();sp(%d);next}\n",\
699                         (r[2] - r[1]) + 1
700                 }
701                 next
702         }
703         if (index($1, "d")) {
704                 n = split(a[1], r, /,/);
705                 n1 = r[1]
706                 n2 = r[2]
707                 if (n == 1)
708                         printf "NR==%s\t\t{removed(); next}\n" , n1
709                 else
710                         printf "NR==%s,NR==%s\t{removed(); next}\n" , n1, n2
711                 next
712         }
713         if (index($1, "c")) {
714                 n = split(a[1], r, /,/);
715                 n1 = r[1]
716                 n2 = r[2]
717                 final = n2
718                 d1 = 0
719                 if (n == 1)
720                         printf "NR==%s\t\t{changed();" , n1
721                 else {
```



```
722                         d1 = n2 - n1
723                         printf "NR==%s,NR==%s\t{changed();" , n1, n2
724                 }
725                 m = split(a[2], r, /,/);
726                 n1 = r[1]
727                 n2 = r[2]
728                 if (m > 1) {
729                         d2  = n2 - n1
730                         if (d2 > d1) {
731                                 if (n > 1) printf "if (NR==%d)", final
732                                 printf "sp(%d);", d2 - d1
733                         }
734                 }
735                 printf "next}\n" ;

737                 next
738         }
739         }

741         END     { printf "{printf \"%%4d %%s\\n\", NR, $0 }\n" }
742         ' /tmp/$$.diffs > /tmp/$$.file1

744         #
745         #  Now generate the HTML for the new file
746         #
747         $AWK '
748         BEGIN   {
749                 printf "function sp(n) {for (i=0;i<n;i++)printf \"\\n\"}\n"
750                 printf "function new() "
751                 printf "{printf \"<span class=\\\"new\\\">%%4d %%s</span>\\n\",
752                 printf "function changed() "
753                 printf "{printf \"<span class=\\\"changed\\\">%%4d %%s</span>\\n
754                 printf "function bl() {printf \"%%4d %%s\\n\", NR, $0}\n"
755         }

757         /^</    {next}
758         /^>/    {next}
759         /^---/  {next}

761         {
762         split($1, a, /[cad]/) ;
763         if (index($1, "d")) {
764                 if (a[2] == 0) {
765                         n = split(a[1], r, /,/);
766                         if (n == 1)
767                                 printf "BEGIN\t\t{sp(1)}\n"
768                         else
769                                 printf "BEGIN\t\t{sp(%d)}\n",\
770                                 (r[2] - r[1]) + 1
771                         next
772                 }

774                 printf "NR==%s\t\t{", a[2]
775                 n = split(a[1], r, /,/);
776                 s = r[1];
777                 if (n == 1)
778                         printf "bl();printf \"\\n\"; next}\n"
779                 else {
780                         n = r[2] - r[1]
781                         printf "bl();sp(%d);next}\n",\
782                         (r[2] - r[1]) + 1
783                 }
784                 next
785         }
786         if (index($1, "a")) {
787                 n = split(a[2], r, /,/);
```

```
788                 n1 = r[1]
789                 n2 = r[2]
790                 if (n == 1)
791                         printf "NR==%s\t\t{new() ; next}\n" , n1
792                 else
793                         printf "NR==%s,NR==%s\t{new() ; next}\n" , n1, n2
794                 next
795         }
796         if (index($1, "c")) {
797                 n = split(a[2], r, /,/);
798                 n1 = r[1]
799                 n2 = r[2]
800                 final = n2
801                 d2 = 0;
802                 if (n == 1) {
803                         final = n1
804                         printf "NR==%s\t\t{changed();" , n1
805                 } else {
806                         d2 = n2 - n1
807                         printf "NR==%s,NR==%s\t{changed();" , n1, n2
808                 }
809                 m = split(a[1], r, /,/);
810                 n1 = r[1]
811                 n2 = r[2]
812                 if (m > 1) {
813                         d1  = n2 - n1
814                         if (d1 > d2) {
815                                 if (n > 1) printf "if (NR==%d)", final
816                                 printf "sp(%d);", d1 - d2
817                         }
818                 }
819                 printf "next}\n" ;
820                 next
821         }
822         }
823         END     { printf "{printf \"%%4d %%s\\n\", NR, $0 }\n" }
824         ' /tmp/$$.diffs > /tmp/$$.file2

826         #
827         # Post-process the HTML files by running them back through $AWK
828         #
829         html_quote < $1 | $AWK -f /tmp/$$.file1 > /tmp/$$.file1.html

831         html_quote < $2 | $AWK -f /tmp/$$.file2 > /tmp/$$.file2.html

833         #
834         # Now combine into a valid HTML file and side-by-side into a table
835         #
836         print "$HTML<head>$STDHEAD"
837         print "<title>$WNAME Sdiff $TPATH/$TNAME</title>"
838         print "</head><body id=\"SUNWwebrev\">"
839         print "<a class=\"print\" href=\"javascript:print()\">Print this page</a
840         print "<pre>$COMMENT</pre>\n"
841         print "<table><tr valign=\"top\">"
842         print "<td><pre>"

844         strip_unchanged /tmp/$$.file1.html

846         print "</pre></td><td><pre>"

848         strip_unchanged /tmp/$$.file2.html

850         print "</pre></td>"
851         print "</tr></table>"
852         print "</body></html>"
```



```
854         framed_sdiff $TNAME $TPATH /tmp/$$.file1.html /tmp/$$.file2.html \
855             "$COMMENT"
856 }


859 #
860 # framed_sdiff <filename> <filepath> <lhsfile> <rhsfile> <comment>
861 #
862 # Expects lefthand and righthand side html files created by sdiff_to_html.
863 # We use insert_anchors() to augment those with HTML navigation anchors,
864 # and then emit the main frame.  Content is placed into:
865 #
866 #    $WDIR/DIR/$TNAME.lhs.html
867 #    $WDIR/DIR/$TNAME.rhs.html
868 #    $WDIR/DIR/$TNAME.frames.html
869 #
870 # NOTE: We rely on standard usage of $WDIR and $DIR.
871 #
872 function framed_sdiff
873 {
874         typeset TNAME=$1
875         typeset TPATH=$2
876         typeset lhsfile=$3
877         typeset rhsfile=$4
878         typeset comments=$5
879         typeset RTOP

881         # Enable html files to access WDIR via a relative path.
882         RTOP=$(relative_dir $TPATH $WDIR)

884         # Make the rhs/lhs files and output the frameset file.
885         print "$HTML<head>$STDHEAD" > $WDIR/$DIR/$TNAME.lhs.html

887         cat >> $WDIR/$DIR/$TNAME.lhs.html <<-EOF
888             <script type="text/javascript" src="${RTOP}ancnav.js"></script>
889             </head>
890             <body id="SUNWwebrev" onkeypress="keypress(event);">
891             <a name="0"></a>
892             <pre>$comments</pre><hr></hr>
893         EOF

895         cp $WDIR/$DIR/$TNAME.lhs.html $WDIR/$DIR/$TNAME.rhs.html

897         insert_anchors $lhsfile >> $WDIR/$DIR/$TNAME.lhs.html
898         insert_anchors $rhsfile >> $WDIR/$DIR/$TNAME.rhs.html

900         close='</body></html>'

902         print $close >> $WDIR/$DIR/$TNAME.lhs.html
903         print $close >> $WDIR/$DIR/$TNAME.rhs.html

905         print "$FRAMEHTML<head>$STDHEAD" > $WDIR/$DIR/$TNAME.frames.html
906         print "<title>$WNAME Framed-Sdiff " \
907             "$TPATH/$TNAME</title> </head>" >> $WDIR/$DIR/$TNAME.frames.html
908         cat >> $WDIR/$DIR/$TNAME.frames.html <<-EOF
909           <frameset rows="*,60">
910             <frameset cols="50%,50%">
911               <frame src="$TNAME.lhs.html" scrolling="auto" name="lhs"></frame>
912               <frame src="$TNAME.rhs.html" scrolling="auto" name="rhs"></frame>
913             </frameset>
914           <frame src="${RTOP}ancnav.html" scrolling="no" marginwidth="0"
915            marginheight="0" name="nav"></frame>
916           <noframes>
917             <body id="SUNWwebrev">
918               Alas 'frames' webrev requires that your browser supports frames
919               and has the feature enabled.
```

```
            </body>
          </noframes>
          </frameset>
        </html>
        EOF
}


#
# fix_postscript
#
#       - removing all extraneous headers/trailers
#       - making the page numbers right
#       - removing pages devoid of contents which confuse some
#         postscript readers.
#
# From Casper.
#
function fix_postscript
{
        infile=$1

        cat > /tmp/$$.crmerge.pl << \EOF

        print scalar(<>);               # %!PS-Adobe---
        print "%%Orientation: Landscape\n";

        $pno = 0;
        $doprint = 1;

        $page = "";

        while (<>) {
                next if (/^%%Pages:\s*\d+/);

                if (/^%%Page:/) {
                        if ($pno == 0 || $page =~ /\)S/) {
                                # Header or single page containing text
                                print "%%Page: ? $pno\n" if ($pno > 0);
                                print $page;
                                $pno++;
                        } else {
                                # Empty page, skip it.
                        }
                        $page = "";
                        $doprint = 1;
                        next;
                }

                # Skip from %%Trailer of one document to Endprolog
                # %%Page of the next
                $doprint = 0 if (/^%%Trailer/);
                $page .= $_ if ($doprint);
        }

        if ($page =~ /\)S/) {
                print "%%Page: ? $pno\n";
                print $page;
        } else {
                $pno--;
        }
        print "%%Trailer\n%%Pages: $pno\n";
EOF

        $PERL /tmp/$$.crmerge.pl < $infile
}
```

*_____unchanged_portion_omitted_*


```
#
# comments_from_teamware {text|html} parent-file child-file
#
# Find the first delta in the child that's not in the parent.  Get the
# newest delta from the parent, get all deltas from the child starting
# with that delta, and then get all info starting with the second oldest
# delta in that list (the first delta unique to the child).
#
# This code adapted from Bill Shannon's "spc" script
#
comments_from_teamware()
{
        fmt=$1
        pfile=$PWS/$2
        cfile=$CWS/$3

        if [[ ! -f $PWS/${2%/*}/SCCS/s.${2##*/} && -n $RWS ]]; then
                pfile=$RWS/$2
        fi

        if [[ -f $pfile ]]; then
                psid=$($SCCS prs -d:I: $pfile 2>/dev/null)
        else
                psid=1.1
        fi

        set -A sids $($SCCS prs -l -r$psid -d:I: $cfile 2>/dev/null)
        N=${#sids[@]}

        nawkprg='
                /^COMMENTS:/    {p=1; continue}
                /^D [0-9]+\.[0-9]+/ {printf "--- %s ---\n", $2; p=0; }
                NF == 0u        { continue }
                {if (p==0) continue; print $0 }'

        if [[ $N -ge 2 ]]; then
                sid1=${sids[$((N-2))]}  # Gets 2nd to last sid

                if [[ $fmt == "text" ]]; then
                        $SCCS prs -l -r$sid1 $cfile  2>/dev/null | \
                            $AWK "$nawkprg"
                        return
                fi

                $SCCS prs -l -r$sid1 $cfile  2>/dev/null | \
                    html_quote | its2url | $AWK "$nawkprg"
        fi
}

#
# comments_from_wx {text|html} filepath
#
# Given the pathname of a file, find its location in a "wx" active
# file list and print the following comment.  Output is either text or
# HTML; if the latter, embedded bugids (sequence of 5 or more digits)
# are turned into URLs.
#
# This is also used with Mercurial and the file list provided by hg-active.
#
comments_from_wx()
{
        typeset fmt=$1
        typeset p=$2

        comm='$AWK '
```

```
1451         $1 == "'$p'" {
1452                 do getline ; while (NF > 0)
1453                 getline
1454                 while (NF > 0) { print ; getline }
1455                 exit
1456         }' < $wxfile`

1458         if [[ -z $comm ]]; then
1459                 comm="*** NO COMMENTS ***"
1460         fi

1462         if [[ $fmt == "text" ]]; then
1463                 print -- "$comm"
1464                 return
1465         fi

1467         print -- "$comm" | html_quote | its2url

1469 }

1471 #
1472 # getcomments {text|html} filepath parentpath
1473 #
1474 # Fetch the comments depending on what SCM mode we're in.
1475 #
1476 getcomments()
1477 {
1478         typeset fmt=$1
1479         typeset p=$2
1480         typeset pp=$3

1482         if [[ -n $Nflag ]]; then
1483                 return
1484         fi
1485         #
1486         # Mercurial support uses a file list in wx format, so this
1487         # will be used there, too
1488         #
1489         if [[ -n $wxfile ]]; then
1490                 comments_from_wx $fmt $p
 638         else
 639                 if [[ $SCM_MODE == "teamware" ]]; then
 640                         comments_from_teamware $fmt $pp $p
 641                 fi
1491         fi
1492 }
______unchanged_portion_omitted_

1668 #
 820 # flist_from_teamware [ <args-to-putback-n> ]
 821 #
 822 # Generate the file list by extracting file names from a putback -n.  Some
 823 # names may come from the "update/create" messages and others from the
 824 # "currently checked out" warning.  Renames are detected here too.  Extract
 825 # values for CODEMGR_WS and CODEMGR_PARENT from the output of the putback
 826 # -n as well, but remove them if they are already defined.
 827 #
 828 function flist_from_teamware
 829 {
 830         if [[ -n $codemgr_parent && -z $parent_webrev ]]; then
 831                 if [[ ! -d $codemgr_parent/Codemgr_wsdata ]]; then
 832                         print -u2 "parent $codemgr_parent doesn't look like a" \
 833                             "valid teamware workspace"
 834                         exit 1
 835                 fi
 836                 parent_args="-p $codemgr_parent"
```



```
 837         fi

 839         print " File list from: 'putback -n $parent_args $*' ... \c"

 841         putback -n $parent_args $* 2>&1 |
 842             $AWK '
 843                 /^update:|^create:/     {print $2}
 844                 /^Parent workspace:/    {printf("CODEMGR_PARENT=%s\n",$3)}
 845                 /^Child workspace:/     {printf("CODEMGR_WS=%s\n",$3)}
 846                 /^The following files are currently checked out/ {p = 1; continu
 847                 NF == 0                 {p=0 ; continue}
 848                 /^rename/               {old=$3}
 849                 $1 == "to:"             {print $2, old}
 850                 /^"/                    {continue}
 851                 p == 1                  {print $1}' |
 852             sort -r -k 1,1 -u | sort > $FLIST

 854         print " Done."
 855 }

 857 #
1669 # Call hg-active to get the active list output in the wx active list format
1670 #
1671 function hg_active_wxfile
1672 {
1673         typeset child=$1
1674         typeset parent=$2

1676         TMPFLIST=/tmp/$$.active
1677         $HG_ACTIVE -w $child -p $parent -o $TMPFLIST
1678         wxfile=$TMPFLIST
1679 }
______unchanged_portion_omitted_

1821 function look_for_prog
1822 {
1823         typeset path
1824         typeset ppath
1825         typeset progname=$1

1827         ppath=$PATH
1828         ppath=$ppath:/usr/sfw/bin:/usr/bin:/usr/sbin
1829         ppath=$ppath:/opt/onbld/bin
1018         ppath=$ppath:/opt/teamware/bin:/opt/onbld/bin
1830         ppath=$ppath:/opt/onbld/bin/`uname -p`

1832         PATH=$ppath prog=`whence $progname`
1833         if [[ -n $prog ]]; then
1834                 print $prog
1835         fi
1836 }
______unchanged_portion_omitted_

1040 function build_old_new_teamware
1041 {
1042         typeset olddir="$1"
1043         typeset newdir="$2"

1045         # If the child's version doesn't exist then
1046         # get a readonly copy.

1048         if [[ ! -f $CWS/$DIR/$F && -f $CWS/$DIR/SCCS/s.$F ]]; then
1049                 $SCCS get -s -p $CWS/$DIR/$F > $CWS/$DIR/$F
1050         fi

1052         # The following two sections propagate file permissions the
```

```
1053         # same way SCCS does.  If the file is already under version
1054         # control, always use permissions from the SCCS/s.file.  If
1055         # the file is not under SCCS control, use permissions from the
1056         # working copy.  In all cases, the file copied to the webrev
1057         # is set to read only, and group/other permissions are set to
1058         # match those of the file owner.  This way, even if the file
1059         # is currently checked out, the webrev will display the final
1060         # permissions that would result after check in.

1062         #
1063         # Snag new version of file.
1064         #
1065         rm -f $newdir/$DIR/$F
1066         cp $CWS/$DIR/$F $newdir/$DIR/$F
1067         if [[ -f $CWS/$DIR/SCCS/s.$F ]]; then
1068                 chmod 'get_file_mode $CWS/$DIR/SCCS/s.$F' \
1069                     $newdir/$DIR/$F
1070         fi
1071         chmod u-w,go=u $newdir/$DIR/$F

1073         #
1074         # Get the parent's version of the file. First see whether the
1075         # child's version is checked out and get the parent's version
1076         # with keywords expanded or unexpanded as appropriate.
1077         #
1078         if [[ -f $PWS/$PDIR/$PF && ! -f $PWS/$PDIR/SCCS/s.$PF && \
1079             ! -f $PWS/$PDIR/SCCS/p.$PF ]]; then
1080                 # Parent is not a real workspace, but just a raw
1081                 # directory tree - use the file that's there as
1082                 # the old file.

1084                 rm -f $olddir/$PDIR/$PF
1085                 cp $PWS/$PDIR/$PF $olddir/$PDIR/$PF
1086         else
1087                 if [[ -f $PWS/$PDIR/SCCS/s.$PF ]]; then
1088                         real_parent=$PWS
1089                 else
1090                         real_parent=$RWS
1091                 fi

1093                 rm -f $olddir/$PDIR/$PF

1095                 if [[ -f $real_parent/$PDIR/$PF ]]; then
1096                         if [ -f $CWS/$DIR/SCCS/p.$F ]; then
1097                                 $SCCS get -s -p -k $real_parent/$PDIR/$PF > \
1098                                     $olddir/$PDIR/$PF
1099                         else
1100                                 $SCCS get -s -p    $real_parent/$PDIR/$PF > \
1101                                     $olddir/$PDIR/$PF
1102                         fi
1103                         chmod 'get_file_mode $real_parent/$PDIR/SCCS/s.$PF' \
1104                             $olddir/$PDIR/$PF
1105                 fi
1106         fi
1107         if [[ -f $olddir/$PDIR/$PF ]]; then
1108                 chmod u-w,go=u $olddir/$PDIR/$PF
1109         fi
1110 }

1851 function build_old_new_mercurial
1852 {
1853         typeset olddir="$1"
1854         typeset newdir="$2"
1855         typeset old_mode=
1856         typeset new_mode=
1857         typeset file
```



```
1859         #
1860         # Get old file mode, from the parent revision manifest entry.
1861         # Mercurial only stores a "file is executable" flag, but the
1862         # manifest will display an octal mode "644" or "755".
1863         #
1864         if [[ "$PDIR" == "." ]]; then
1865                 file="$PF"
1866         else
1867                 file="$PDIR/$PF"
1868         fi
1869         file='echo $file | $SED 's#/#\\\/#g''
1870         # match the exact filename, and return only the permission digits
1871         old_mode='$SED -n -e "/^\\(...\\) . ${file}$/s//\\1/p" \
1872             < $HG_PARENT_MANIFEST'

1874         #
1875         # Get new file mode, directly from the filesystem.
1876         # Normalize the mode to match Mercurial's behavior.
1877         #
1878         new_mode='get_file_mode $CWS/$DIR/$F'
1879         if [[ -n "$new_mode" ]]; then
1880                 if [[ "$new_mode" = *[1357]* ]]; then
1881                         new_mode=755
1882                 else
1883                         new_mode=644
1884                 fi
1885         fi

1887         #
1888         # new version of the file.
1889         #
1890         rm -rf $newdir/$DIR/$F
1891         if [[ -e $CWS/$DIR/$F ]]; then
1892                 cp $CWS/$DIR/$F $newdir/$DIR/$F
1893                 if [[ -n $new_mode ]]; then
1894                         chmod $new_mode $newdir/$DIR/$F
1895                 else
1896                         # should never happen
1897                         print -u2 "ERROR: set mode of $newdir/$DIR/$F"
1898                 fi
1899         fi

1901         #
1902         # parent's version of the file
1903         #
1904         # Note that we get this from the last version common to both
1905         # ourselves and the parent.  References are via $CWS since we have no
1906         # guarantee that the parent workspace is reachable via the filesystem.
1907         #
1908         if [[ -n $parent_webrev && -e $PWS/$PDIR/$PF ]]; then
1909                 cp $PWS/$PDIR/$PF $olddir/$PDIR/$PF
1910         elif [[ -n $HG_PARENT ]]; then
1911                 hg cat -R $CWS -r $HG_PARENT $CWS/$PDIR/$PF > \
1912                     $olddir/$PDIR/$PF 2>/dev/null

1914                 if (( $? != 0 )); then
1915                         rm -f $olddir/$PDIR/$PF
1916                 else
1917                         if [[ -n $old_mode ]]; then
1918                                 chmod $old_mode $olddir/$PDIR/$PF
1919                         else
1920                                 # should never happen
1921                                 print -u2 "ERROR: set mode of $olddir/$PDIR/$PF"
1922                         fi
1923                 fi
```

```
1924         fi
1925 }
_____unchanged_portion_omitted_

2025 function build_old_new
2026 {
2027         typeset WDIR=$1
2028         typeset PWS=$2
2029         typeset PDIR=$3
2030         typeset PF=$4
2031         typeset CWS=$5
2032         typeset DIR=$6
2033         typeset F=$7

2035         typeset olddir="$WDIR/raw_files/old"
2036         typeset newdir="$WDIR/raw_files/new"

2038         mkdir -p $olddir/$PDIR
2039         mkdir -p $newdir/$DIR

2041         if [[ $SCM_MODE == "mercurial" ]]; then
1302         if [[ $SCM_MODE == "teamware" ]]; then
1303                 build_old_new_teamware "$olddir" "$newdir"
1304         elif [[ $SCM_MODE == "mercurial" ]]; then
2042                 build_old_new_mercurial "$olddir" "$newdir"
2043         elif [[ $SCM_MODE == "git" ]]; then
2044                 build_old_new_git "$olddir" "$newdir"
2045         elif [[ $SCM_MODE == "subversion" ]]; then
2046                 build_old_new_subversion "$olddir" "$newdir"
2047         elif [[ $SCM_MODE == "unknown" ]]; then
2048                 build_old_new_unknown "$olddir" "$newdir"
2049         fi

2051         if [[ ! -f $olddir/$PDIR/$PF && ! -f $newdir/$DIR/$F ]]; then
2052                 print "*** Error: file not in parent or child"
2053                 return 1
2054         fi
2055         return 0
2056 }


2059 #
2060 # Usage message.
2061 #
2062 function usage
2063 {
2064         print 'Usage:\twebrev [common-options]
2065         webrev [common-options] ( <file> | - )
2066         webrev [common-options] -w <wx file>

2068 Options:
2069         -C <filename>: Use <filename> for the information tracking configuration
2070         -D: delete remote webrev
2071         -i <filename>: Include <filename> in the index.html file.
2072         -I <filename>: Use <filename> for the information tracking registry.
2073         -n: do not generate the webrev (useful with -U)
2074         -O: Print bugids/arc cases suitable for OpenSolaris.
2075         -o <outdir>: Output webrev to specified directory.
2076         -p <compare-against>: Use specified parent wkspc or basis for comparison
2077         -t <remote_target>: Specify remote destination for webrev upload
2078         -U: upload the webrev to remote destination
2079         -w <wxfile>: Use specified wx active file.

2081 Environment:
2082         WDIR: Control the output directory.
2083         WEBREV_TRASH_DIR: Set directory for webrev delete.
```



```
1348 SCM Specific Options:
1349         TeamWare: webrev [common-options] -l [arguments to 'putback']

2085 SCM Environment:
2086         CODEMGR_WS: Workspace location.
2087         CODEMGR_PARENT: Parent workspace location.
2088 '

2090         exit 2
2091 }

2093 #
2094 #
2095 # Main program starts here
2096 #
2097 #

2099 trap "rm -f /tmp/$$.* ; exit" 0 1 2 3 15

2101 set +o noclobber

2103 PATH=$(/bin/dirname "$(whence $0)"):$PATH

2105 [[ -z $WDIFF ]] && WDIFF=`look_for_prog wdiff`
2106 [[ -z $WX ]] && WX=`look_for_prog wx`
2107 [[ -z $HG_ACTIVE ]] && HG_ACTIVE=`look_for_prog hg-active`
2108 [[ -z $GIT ]] && GIT=`look_for_prog git`
2109 [[ -z $WHICH_SCM ]] && WHICH_SCM=`look_for_prog which_scm`
2110 [[ -z $CODEREVIEW ]] && CODEREVIEW=`look_for_prog codereview`
2111 [[ -z $PS2PDF ]] && PS2PDF=`look_for_prog ps2pdf`
2112 [[ -z $PERL ]] && PERL=`look_for_prog perl`
2113 [[ -z $RSYNC ]] && RSYNC=`look_for_prog rsync`
2114 [[ -z $SCCS ]] && SCCS=`look_for_prog sccs`
2115 [[ -z $AWK ]] && AWK=`look_for_prog nawk`
2116 [[ -z $AWK ]] && AWK=`look_for_prog gawk`
2117 [[ -z $AWK ]] && AWK=`look_for_prog awk`
2118 [[ -z $SCP ]] && SCP=`look_for_prog scp`
2119 [[ -z $SED ]] && SED=`look_for_prog sed`
2120 [[ -z $SFTP ]] && SFTP=`look_for_prog sftp`
2121 [[ -z $SORT ]] && SORT=`look_for_prog sort`
2122 [[ -z $MKTEMP ]] && MKTEMP=`look_for_prog mktemp`
2123 [[ -z $GREP ]] && GREP=`look_for_prog grep`
2124 [[ -z $FIND ]] && FIND=`look_for_prog find`

2126 # set name of trash directory for remote webrev deletion
2127 TRASH_DIR=".trash"
2128 [[ -n $WEBREV_TRASH_DIR ]] && TRASH_DIR=$WEBREV_TRASH_DIR

2130 if [[ ! -x $PERL ]]; then
2131         print -u2 "Error: No perl interpreter found.  Exiting."
2132         exit 1
2133 fi

2135 if [[ ! -x $WHICH_SCM ]]; then
2136         print -u2 "Error: Could not find which_scm.  Exiting."
2137         exit 1
2138 fi

2140 #
2141 # These aren't fatal, but we want to note them to the user.
2142 # We don't warn on the absence of 'wx' until later when we've
2143 # determined that we actually need to try to invoke it.
2144 #
2145 [[ ! -x $CODEREVIEW ]] && print -u2 "WARNING: codereview(1) not found."
2146 [[ ! -x $PS2PDF ]] && print -u2 "WARNING: ps2pdf(1) not found."
```

```
2147 [[ ! -x $WDIFF ]] && print -u2 "WARNING: wdiff not found."

2149 # Declare global total counters.
2150 integer TOTL TINS TDEL TMOD TUNC

2152 # default remote host for upload/delete
2153 typeset -r DEFAULT_REMOTE_HOST="cr.opensolaris.org"
2154 # prefixes for upload targets
2155 typeset -r rsync_prefix="rsync://"
2156 typeset -r ssh_prefix="ssh://"

2158 Cflag=
2159 Dflag=
2160 flist_mode=
2161 flist_file=
2162 iflag=
2163 Iflag=
2164 lflag=
2165 Nflag=
2166 nflag=
2167 Oflag=
2168 oflag=
2169 pflag=
2170 tflag=
2171 uflag=
2172 Uflag=
2173 wflag=
2174 remote_target=

2176 #
2177 # NOTE: when adding/removing options it is necessary to sync the list
2178 #       with usr/src/tools/onbld/hgext/cdm.py
2179 #
2180 while getopts "C:Di:I:lnNo:Op:t:Uw" opt
2181 do
2182         case $opt in
2183         C)      Cflag=1
2184                 ITSCONF=$OPTARG;;

2186         D)      Dflag=1;;

2188         i)      iflag=1
2189                 INCLUDE_FILE=$OPTARG;;

2191         I)      Iflag=1
2192                 ITSREG=$OPTARG;;

2194         #
2195         # If -l has been specified, we need to abort further options
2196         # processing, because subsequent arguments are going to be
2197         # arguments to 'putback -n'.
2198         #
2199         l)      lflag=1
2200                 break;;

2202         N)      Nflag=1;;

2204         n)      nflag=1;;

2206         O)      Oflag=1;;

2208         o)      oflag=1
2209                 # Strip the trailing slash to correctly form remote target.
2210                 WDIR=${OPTARG%/};;

2212         p)      pflag=1
```



```
2213                 codemgr_parent=$OPTARG;;

2215         t)      tflag=1
2216                 remote_target=$OPTARG;;

2218         U)      Uflag=1;;

2220         w)      wflag=1;;

2222         ?)      usage;;
2223         esac
2224 done

2226 FLIST=/tmp/$$.flist

2228 if [[ -n $wflag && -n $lflag ]]; then
2229         usage
2230 fi

2232 # more sanity checking
2233 if [[ -n $nflag && -z $Uflag ]]; then
2234         print "it does not make sense to skip webrev generation" \
2235             "without -U"
2236         exit 1
2237 fi

2239 if [[ -n $tflag && -z $Uflag && -z $Dflag ]]; then
2240         echo "remote target has to be used only for upload or delete"
2241         exit 1
2242 fi

2244 #
2245 # For the invocation "webrev -n -U" with no other options, webrev will assume
2246 # that the webrev exists in ${CWS}/webrev, but will upload it using the name
2247 # $(basename ${CWS}).  So we need to get CWS set before we skip any remaining
2248 # logic.
2249 #
2250 $WHICH_SCM | read SCM_MODE junk || exit 1
2251 if [[ $SCM_MODE == "mercurial" ]]; then
1517 if [[ $SCM_MODE == "teamware" ]]; then
1518         #
1519         # Teamware priorities:
1520         # 1. CODEMGR_WS from the environment
1521         # 2. workspace name
1522         #
1523         [[ -z $codemgr_ws && -n $CODEMGR_WS ]] && codemgr_ws=$CODEMGR_WS
1524         if [[ -n $codemgr_ws && ! -d $codemgr_ws ]]; then
1525                 print -u2 "$codemgr_ws: no such workspace"
1526                 exit 1
1527         fi
1528         [[ -z $codemgr_ws ]] && codemgr_ws=$(workspace name)
1529         codemgr_ws=$(cd $codemgr_ws;print $PWD)
1530         CODEMGR_WS=$codemgr_ws
1531         CWS=$codemgr_ws
1532 elif [[ $SCM_MODE == "mercurial" ]]; then
2252         #
2253         # Mercurial priorities:
2254         # 1. hg root from CODEMGR_WS environment variable
2255         # 1a. hg root from CODEMGR_WS/usr/closed if we're somewhere under
2256         #     usr/closed when we run webrev
2257         # 2. hg root from directory of invocation
2258         #
2259         if [[ ${PWD} =~ "usr/closed" ]]; then
2260                 testparent=${CODEMGR_WS}/usr/closed
2261                 # If we're in OpenSolaris mode, we enforce a minor policy:
2262                 # help to make sure the reviewer doesn't accidentally publish
```

```
2263                 # source which is under usr/closed
2264                 if [[ -n "$Oflag" ]]; then
2265                         print -u2 "OpenSolaris output not permitted with" \
2266                             "usr/closed changes"
2267                         exit 1
2268                 fi
2269         else
2270                 testparent=${CODEMGR_WS}
2271         fi
2272         [[ -z $codemgr_ws && -n $testparent ]] && \
2273             codemgr_ws=$(hg root -R $testparent 2>/dev/null)
2274         [[ -z $codemgr_ws ]] && codemgr_ws=$(hg root 2>/dev/null)
2275         CWS=$codemgr_ws
2276 elif [[ $SCM_MODE == "git" ]]; then
2277         #
2278         # Git priorities:
2279         # 1. git rev-parse --git-dir from CODEMGR_WS environment variable
2280         # 2. git rev-parse --git-dir from directory of invocation
2281         #
2282         [[ -z $codemgr_ws && -n $CODEMGR_WS ]] && \
2283             codemgr_ws=$($GIT --git-dir=$CODEMGR_WS/.git rev-parse --git-dir \
2284                 2>/dev/null)
2285         [[ -z $codemgr_ws ]] && \
2286             codemgr_ws=$($GIT rev-parse --git-dir 2>/dev/null)

2288         if [[ "$codemgr_ws" == ".git" ]]; then
2289                 codemgr_ws="${PWD}/${codemgr_ws}"
2290         fi

2292         codemgr_ws=$(dirname $codemgr_ws) # Lose the '/.git'
2293         CWS="$codemgr_ws"
2294 elif [[ $SCM_MODE == "subversion" ]]; then
2295         #
2296         # Subversion priorities:
2297         # 1. CODEMGR_WS from environment
2298         # 2. Relative path from current directory to SVN repository root
2299         #
2300         if [[ -n $CODEMGR_WS && -d $CODEMGR_WS/.svn ]]; then
2301                 CWS=$CODEMGR_WS
2302         else
2303                 svn info | while read line; do
2304                         if [[ $line == "URL: "* ]]; then
2305                                 url=${line#URL: }
2306                         elif [[ $line == "Repository Root: "* ]]; then
2307                                 repo=${line#Repository Root: }
2308                         fi
2309                 done

2311                 rel=${url#$repo}
2312                 CWS=${PWD%$rel}
2313         fi
2314 fi

2316 #
2317 # If no SCM has been determined, take either the environment setting
2318 # setting for CODEMGR_WS, or the current directory if that wasn't set.
2319 #
2320 if [[ -z ${CWS} ]]; then
2321         CWS=${CODEMGR_WS:-.}
2322 fi

2324 #
2325 # If the command line options indicate no webrev generation, either
2326 # explicitly (-n) or implicitly (-D but not -U), then there's a whole
2327 # ton of logic we can skip.
2328 #
```



```
2329 # Instead of increasing indentation, we intentionally leave this loop
2330 # body open here, and exit via break from multiple points within.
2331 # Search for DO_EVERYTHING below to find the break points and closure.
2332 #
2333 for do_everything in 1; do

2335 # DO_EVERYTHING: break point
2336 if [[ -n $nflag || ( -z $Uflag && -n $Dflag ) ]]; then
2337         break
2338 fi

2340 #
2341 # If this manually set as the parent, and it appears to be an earlier webrev,
2342 # then note that fact and set the parent to the raw_files/new subdirectory.
2343 #
2344 if [[ -n $pflag && -d $codemgr_parent/raw_files/new ]]; then
2345         parent_webrev=$(readlink -f "$codemgr_parent")
2346         codemgr_parent=$(readlink -f "$codemgr_parent/raw_files/new")
2347 fi

2349 if [[ -z $wflag && -z $lflag ]]; then
2350         shift $(($OPTIND - 1))

2352         if [[ $1 == "-" ]]; then
2353                 cat > $FLIST
2354                 flist_mode="stdin"
2355                 flist_done=1
2356                 shift
2357         elif [[ -n $1 ]]; then
2358                 if [[ ! -r $1 ]]; then
2359                         print -u2 "$1: no such file or not readable"
2360                         usage
2361                 fi
2362                 cat $1 > $FLIST
2363                 flist_mode="file"
2364                 flist_file=$1
2365                 flist_done=1
2366                 shift
2367         else
2368                 flist_mode="auto"
2369         fi
2370 fi

2372 #
2373 # Before we go on to further consider -l and -w, work out which SCM we think
2374 # is in use.
2375 #
2376 case "$SCM_MODE" in
2377 mercurial|git|subversion)
1658 teamware|mercurial|git|subversion)
2378         ;;
2379 unknown)
2380         if [[ $flist_mode == "auto" ]]; then
2381                 print -u2 "Unable to determine SCM in use and file list not spec
2382                 print -u2 "See which_scm(1) for SCM detection information."
2383                 exit 1
2384         fi
2385         ;;
2386 *)
2387         if [[ $flist_mode == "auto" ]]; then
2388                 print -u2 "Unsupported SCM in use ($SCM_MODE) and file list not
2389                 exit 1
2390         fi
2391         ;;
2392 esac
```

```
2394 print -u2 "   SCM detected: $SCM_MODE"

2396 if [[ -n $wflag ]]; then
1677 if [[ -n $lflag ]]; then
1678         #
1679         # If the -l flag is given instead of the name of a file list,
1680         # then generate the file list by extracting file names from a
1681         # putback -n.
1682         #
1683         shift $(($OPTIND - 1))
1684         if [[ $SCM_MODE == "teamware" ]]; then
1685                 flist_from_teamware "$*"
1686         else
1687                 print -u2 -- "Error: -l option only applies to TeamWare"
1688                 exit 1
1689         fi
1690         flist_done=1
1691         shift $#
1692 elif [[ -n $wflag ]]; then
2397         #
2398         # If the -w is given then assume the file list is in Bonwick's "wx"
2399         # command format, i.e.  pathname lines alternating with SCCS comment
2400         # lines with blank lines as separators.  Use the SCCS comments later
2401         # in building the index.html file.
2402         #
2403         shift $(($OPTIND - 1))
2404         wxfile=$1
2405         if [[ -z $wxfile && -n $CODEMGR_WS ]]; then
2406                 if [[ -r $CODEMGR_WS/wx/active ]]; then
2407                         wxfile=$CODEMGR_WS/wx/active
2408                 fi
2409         fi

2411         [[ -z $wxfile ]] && print -u2 "wx file not specified, and could not " \
2412             "be auto-detected (check \$CODEMGR_WS)" && exit 1

2414         if [[ ! -r $wxfile ]]; then
2415                 print -u2 "$wxfile: no such file or not readable"
2416                 usage
2417         fi

2419         print -u2 " File list from: wx 'active' file '$wxfile' ... \c"
2420         flist_from_wx $wxfile
2421         flist_done=1
2422         if [[ -n "$*" ]]; then
2423                 shift
2424         fi
2425 elif [[ $flist_mode == "stdin" ]]; then
2426         print -u2 " File list from: standard input"
2427 elif [[ $flist_mode == "file" ]]; then
2428         print -u2 " File list from: $flist_file"
2429 fi

2431 if [[ $# -gt 0 ]]; then
2432         print -u2 "WARNING: unused arguments: $*"
2433 fi

2435 #
2436 # Before we entered the DO_EVERYTHING loop, we should have already set CWS
2437 # and CODEMGR_WS as needed.  Here, we set the parent workspace.
2438 #
2439 if [[ $SCM_MODE == "mercurial" ]]; then

1736 if [[ $SCM_MODE == "teamware" ]]; then

1738         #
```



```
1739         # Teamware priorities:
1740         #
1741         #       1) via -p command line option
1742         #       2) in the user environment
1743         #       3) in the flist
1744         #       4) automatically based on the workspace
1745         #

1747         #
1748         # For 1, codemgr_parent will already be set.  Here's 2:
1749         #
1750         [[ -z $codemgr_parent && -n $CODEMGR_PARENT ]] && \
1751             codemgr_parent=$CODEMGR_PARENT
1752         if [[ -n $codemgr_parent && ! -d $codemgr_parent ]]; then
1753                 print -u2 "$codemgr_parent: no such directory"
1754                 exit 1
1755         fi

1757         #
1758         # If we're in auto-detect mode and we haven't already gotten the file
1759         # list, then see if we can get it by probing for wx.
1760         #
1761         if [[ -z $flist_done && $flist_mode == "auto" && -n $codemgr_ws ]]; then
1762                 if [[ ! -x $WX ]]; then
1763                         print -u2 "WARNING: wx not found!"
1764                 fi

1766                 #
1767                 # We need to use wx list -w so that we get renamed files, etc.
1768                 # but only if a wx active file exists-- otherwise wx will
1769                 # hang asking us to initialize our wx information.
1770                 #
1771                 if [[ -x $WX && -f $codemgr_ws/wx/active ]]; then
1772                         print -u2 " File list from: 'wx list -w' ... \c"
1773                         $WX list -w > $FLIST
1774                         $WX comments > /tmp/$$.wx_comments
1775                         wxfile=/tmp/$$.wx_comments
1776                         print -u2 "done"
1777                         flist_done=1
1778                 fi
1779         fi

1781         #
1782         # If by hook or by crook we've gotten a file list by now (perhaps
1783         # from the command line), eval it to extract environment variables from
1784         # it: This is method 3 for finding the parent.
1785         #
1786         if [[ -z $flist_done ]]; then
1787                 flist_from_teamware
1788         fi
1789         env_from_flist

1791         #
1792         # (4) If we still don't have a value for codemgr_parent, get it
1793         # from workspace.
1794         #
1795         [[ -z $codemgr_parent ]] && codemgr_parent=`workspace parent`
1796         if [[ ! -d $codemgr_parent ]]; then
1797                 print -u2 "$CODEMGR_PARENT: no such parent workspace"
1798                 exit 1
1799         fi

1801         PWS=$codemgr_parent

1803         [[ -n $parent_webrev ]] && RWS=$(workspace parent $CWS)
```

```
1805 elif [[ $SCM_MODE == "mercurial" ]]; then
2440         #
2441         # Parent can either be specified with -p
2442         # Specified with CODEMGR_PARENT in the environment
2443         # or taken from hg's default path.
2444         #

2446         if [[ -z $codemgr_parent && -n $CODEMGR_PARENT ]]; then
2447                 codemgr_parent=$CODEMGR_PARENT
2448         fi

2450         if [[ -z $codemgr_parent ]]; then
2451                 codemgr_parent=`hg path -R $codemgr_ws default 2>/dev/null`
2452         fi

2454         PWS=$codemgr_parent

2456         #
2457         # If the parent is a webrev, we want to do some things against
2458         # the natural workspace parent (file list, comments, etc)
2459         #
2460         if [[ -n $parent_webrev ]]; then
2461                 real_parent=$(hg path -R $codemgr_ws default 2>/dev/null)
2462         else
2463                 real_parent=$PWS
2464         fi

2466         #
2467         # If hg-active exists, then we run it.  In the case of no explicit
2468         # flist given, we'll use it for our comments.  In the case of an
2469         # explicit flist given we'll try to use it for comments for any
2470         # files mentioned in the flist.
2471         #
2472         if [[ -z $flist_done ]]; then
2473                 flist_from_mercurial $CWS $real_parent
2474                 flist_done=1
2475         fi

2477         #
2478         # If we have a file list now, pull out any variables set
2479         # therein.  We do this now (rather than when we possibly use
2480         # hg-active to find comments) to avoid stomping specifications
2481         # in the user-specified flist.
2482         #
2483         if [[ -n $flist_done ]]; then
2484                 env_from_flist
2485         fi

2487         #
2488         # Only call hg-active if we don't have a wx formatted file already
2489         #
2490         if [[ -x $HG_ACTIVE && -z $wxfile ]]; then
2491                 print "  Comments from: hg-active -p $real_parent ...\c"
2492                 hg_active_wxfile $CWS $real_parent
2493                 print " Done."
2494         fi

2496         #
2497         # At this point we must have a wx flist either from hg-active,
2498         # or in general.  Use it to try and find our parent revision,
2499         # if we don't have one.
2500         #
2501         if [[ -z $HG_PARENT ]]; then
2502                 eval `$SED -e "s/#.*$//" $wxfile | $GREP HG_PARENT=`
2503         fi
```



```
2505         #
2506         # If we still don't have a parent, we must have been given a
2507         # wx-style active list with no HG_PARENT specification, run
2508         # hg-active and pull an HG_PARENT out of it, ignore the rest.
2509         #
2510         if [[ -z $HG_PARENT && -x $HG_ACTIVE ]]; then
2511                 $HG_ACTIVE -w $codemgr_ws -p $real_parent | \
2512                     eval `$SED -e "s/#.*$//" | $GREP HG_PARENT=`
2513         elif [[ -z $HG_PARENT ]]; then
2514                 print -u2 "Error: Cannot discover parent revision"
2515                 exit 1
2516         fi

2518         pnode=$(trim_digest $HG_PARENT)
2519         PRETTY_PWS="${PWS} (at ${pnode})"
2520         cnode=$(hg parent -R $codemgr_ws --template '{node|short}' \
2521             2>/dev/null)
2522         PRETTY_CWS="${CWS} (at ${cnode})"}
2523 elif [[ $SCM_MODE == "git" ]]; then
2524         #
2525         # Parent can either be specified with -p, or specified with
2526         # CODEMGR_PARENT in the environment.
2527         #

2529         if [[ -z $codemgr_parent && -n $CODEMGR_PARENT ]]; then
2530                 codemgr_parent=$CODEMGR_PARENT
2531         fi

2533         # Try to figure out the parent based on the branch the current
2534         # branch is tracking, if we fail, use origin/master
2535         this_branch=$($GIT branch | nawk '$1 == "*" { print $2 }')
2536         par_branch="origin/master"

2538         # If we're not on a branch there's nothing we can do
2539         if [[ $this_branch != "(no branch)" ]]; then
2540                 $GIT for-each-ref
2541                     --format='%(refname:short) %(upstream:short)' refs/heads/ |
2542                     while read local remote; do
2543                         [[ "$local" == "$this_branch" ]] && par_branch="$remote"
2544                     done
2545         fi

2547         if [[ -z $codemgr_parent ]]; then
2548                 codemgr_parent=$par_branch
2549         fi
2550         PWS=$codemgr_parent

2552         #
2553         # If the parent is a webrev, we want to do some things against
2554         # the natural workspace parent (file list, comments, etc)
2555         #
2556         if [[ -n $parent_webrev ]]; then
2557                 real_parent=$par_branch
2558         else
2559                 real_parent=$PWS
2560         fi

2562         if [[ -z $flist_done ]]; then
2563                 flist_from_git "$CWS" "$real_parent"
2564                 flist_done=1
2565         fi

2567         #
2568         # If we have a file list now, pull out any variables set
2569         # therein.
2570         #
```

```
2571         if [[ -n $flist_done ]]; then
2572                 env_from_flist
2573         fi

2575         #
2576         # If we don't have a wx-format file list, build one we can pull change
2577         # comments from.
2578         #
2579         if [[ -z $wxfile ]]; then
2580                 print "  Comments from: git...\c"
2581                 git_wxfile "$CWS" "$real_parent"
2582                 print " Done."
2583         fi

2585         if [[ -z $GIT_PARENT ]]; then
2586                 GIT_PARENT=$($GIT merge-base "$real_parent" HEAD)
2587         fi
2588         if [[ -z $GIT_PARENT ]]; then
2589                 print -u2 "Error: Cannot discover parent revision"
2590                 exit 1
2591         fi

2593         pnode=$(trim_digest $GIT_PARENT)

2595         if [[ $real_parent == */* ]]; then
2596                 origin=$(echo $real_parent | cut -d/ -f1)
2597                 origin=$($GIT remote -v | \
2598                     $AWK '$1 == "'$origin'" { print $2; exit }')
2599                 PRETTY_PWS="${PWS} (${origin} at ${pnode})"
2600         else
2601                 PRETTY_PWS="${PWS} (at ${pnode})"
2602         fi

2604         cnode=$($GIT --git-dir=${codemgr_ws}/.git rev-parse --short=12 HEAD \
2605             2>/dev/null)
2606         PRETTY_CWS="${CWS} (at ${cnode})"
2607 elif [[ $SCM_MODE == "subversion" ]]; then

2609         #
2610         # We only will have a real parent workspace in the case one
2611         # was specified (be it an older webrev, or another checkout).
2612         #
2613         [[ -n $codemgr_parent ]] && PWS=$codemgr_parent

2615         if [[ -z $flist_done && $flist_mode == "auto" ]]; then
2616                 flist_from_subversion $CWS $OLDPWD
2617         fi
2618 else
2619     if [[ $SCM_MODE == "unknown" ]]; then
2620         print -u2 "    Unknown type of SCM in use"
2621     else
2622         print -u2 "    Unsupported SCM in use: $SCM_MODE"
2623     fi

2625     env_from_flist

2627     if [[ -z $CODEMGR_WS ]]; then
2628         print -u2 "SCM not detected/supported and CODEMGR_WS not specified"
2629         exit 1
2630     fi

2632     if [[ -z $CODEMGR_PARENT ]]; then
2633         print -u2 "SCM not detected/supported and CODEMGR_PARENT not specified"
2634         exit 1
2635     fi
```



```
2637     CWS=$CODEMGR_WS
2638     PWS=$CODEMGR_PARENT
2639 fi

2641 #
2642 # If the user didn't specify a -i option, check to see if there is a
2643 # webrev-info file in the workspace directory.
2644 #
2645 if [[ -z $iflag && -r "$CWS/webrev-info" ]]; then
2646         iflag=1
2647         INCLUDE_FILE="$CWS/webrev-info"
2648 fi

2650 if [[ -n $iflag ]]; then
2651         if [[ ! -r $INCLUDE_FILE ]]; then
2652                 print -u2 "include file '$INCLUDE_FILE' does not exist or is" \
2653                     "not readable."
2654                 exit 1
2655         else
2656                 #
2657                 # $INCLUDE_FILE may be a relative path, and the script alters
2658                 # PWD, so we just stash a copy in /tmp.
2659                 #
2660                 cp $INCLUDE_FILE /tmp/$$.include
2661         fi
2662 fi

2664 # DO_EVERYTHING: break point
2665 if [[ -n $Nflag ]]; then
2666         break
2667 fi

2669 typeset -A itsinfo
2670 typeset -r its_sed_script=/tmp/$$.its_sed
2671 valid_prefixes=
2672 if [[ -z $nflag ]]; then
2673         DEFREGFILE="$(/bin/dirname "$(whence $0)")/../etc/its.reg"
2674         if [[ -n $Iflag ]]; then
2675                 REGFILE=$ITSREG
2676         elif [[ -r $HOME/.its.reg ]]; then
2677                 REGFILE=$HOME/.its.reg
2678         else
2679                 REGFILE=$DEFREGFILE
2680         fi
2681         if [[ ! -r $REGFILE ]]; then
2682                 print "ERROR: Unable to read database registry file $REGFILE"
2683                 exit 1
2684         elif [[ $REGFILE != $DEFREGFILE ]]; then
2685                 print "   its.reg from: $REGFILE"
2686         fi

2688         $SED -e '/^#/d' -e '/^[         ]*$/d' $REGFILE | while read LINE; do

2690                 name=${LINE%%=*}
2691                 value="${LINE#*=}"

2693                 if [[ $name == PREFIX ]]; then
2694                         p=${value}
2695                         valid_prefixes="${p} ${valid_prefixes}"
2696                 else
2697                         itsinfo["${p}_${name}"]="${value}"
2698                 fi
2699         done


2702         DEFCONFFILE="$(/bin/dirname "$(whence $0)")/../etc/its.conf"
```

```
2703         CONFFILES=$DEFCONFFILE
2704         if [[ -r $HOME/.its.conf ]]; then
2705                 CONFFILES="${CONFFILES} $HOME/.its.conf"
2706         fi
2707         if [[ -n $Cflag ]]; then
2708                 CONFFILES="${CONFFILES} ${ITSCONF}"
2709         fi
2710         its_domain=
2711         its_priority=
2712         for cf in ${CONFFILES}; do
2713                 if [[ ! -r $cf ]]; then
2714                         print "ERROR: Unable to read database configuration file
2715                         exit 1
2716                 elif [[ $cf != $DEFCONFFILE ]]; then
2717                         print "       its.conf: reading $cf"
2718                 fi
2719                 $SED -e '/^#/d' -e '/^[         ]*$/d' $cf | while read LINE; do
2720                     eval "${LINE}"
2721                 done
2722         done

2724         #
2725         # If an information tracking system is explicitly identified by prefix,
2726         # we want to disregard the specified priorities and resolve it according
2727         #
2728         # To that end, we'll build a sed script to do each valid prefix in turn.
2729         #
2730         for p in ${valid_prefixes}; do
2731                 #
2732                 # When an informational URL was provided, translate it to a
2733                 # hyperlink.  When omitted, simply use the prefix text.
2734                 #
2735                 if [[ -z ${itsinfo["${p}_INFO"]} ]]; then
2736                         itsinfo["${p}_INFO"]=${p}
2737                 else
2738                         itsinfo["${p}_INFO"]="<a href=\\\"${itsinfo["${p}_INFO"]
2739                 fi

2741                 #
2742                 # Assume that, for this invocation of webrev, all references
2743                 # to this information tracking system should resolve through
2744                 # the same URL.
2745                 #
2746                 # If the caller specified -O, then always use EXTERNAL_URL.
2747                 #
2748                 # Otherwise, look in the list of domains for a matching
2749                 # INTERNAL_URL.
2750                 #
2751                 [[ -z $Oflag ]] && for d in ${its_domain}; do
2752                         if [[ -n ${itsinfo["${p}_INTERNAL_URL_${d}"]} ]]; then
2753                                 itsinfo["${p}_URL"]="${itsinfo[${p}_INTERNAL_URL
2754                                 break
2755                         fi
2756                 done
2757                 if [[ -z ${itsinfo["${p}_URL"]} ]]; then
2758                         itsinfo["${p}_URL"]="${itsinfo[${p}_EXTERNAL_URL]}"
2759                 fi

2761                 #
2762                 # Turn the destination URL into a hyperlink
2763                 #
2764                 itsinfo["${p}_URL"]="<a href=\\\"${itsinfo[${p}_URL]}\\\">&</a>"

2766                 # The character class below contains a literal tab
2767                 print "/^${p}[:         ]/ {
2768                                 s;${itsinfo[${p}_REGEX]};${itsinfo[${p}_URL]};g
```



```
2769                                 s;^${p};${itsinfo[${p}_INFO]};
2770                         }" >> ${its_sed_script}
2771         done

2773         #
2774         # The previous loop took care of explicit specification.  Now use
2775         # the configured priorities to attempt implicit translations.
2776         #
2777         for p in ${its_priority}; do
2778                 print "/^${itsinfo[${p}_REGEX]}[        ]/ {
2779                                 s;^${itsinfo[${p}_REGEX]};${itsinfo[${p}_URL]};g
2780                         }" >> ${its_sed_script}
2781         done
2782 fi

2784 #
2785 # Search for DO_EVERYTHING above for matching "for" statement
2786 # and explanation of this terminator.
2787 #
2788 done

2790 #
2791 # Output directory.
2792 #
2793 WDIR=${WDIR:-$CWS/webrev}

2795 #
2796 # Name of the webrev, derived from the workspace name or output directory;
2797 # in the future this could potentially be an option.
2798 #
2799 if [[ -n $oflag ]]; then
2800         WNAME=${WDIR##*/}
2801 else
2802         WNAME=${CWS##*/}
2803 fi

2805 # Make sure remote target is well formed for remote upload/delete.
2806 if [[ -n $Dflag || -n $Uflag ]]; then
2807         #
2808         # If remote target is not specified, build it from scratch using
2809         # the default values.
2810         #
2811         if [[ -z $tflag ]]; then
2812                 remote_target=${DEFAULT_REMOTE_HOST}:${WNAME}
2813         else
2814                 #
2815                 # Check upload target prefix first.
2816                 #
2817                 if [[ "${remote_target}" != ${rsync_prefix}* &&
2818                     "${remote_target}" != ${ssh_prefix}* ]]; then
2819                         print "ERROR: invalid prefix of upload URI" \
2820                             "($remote_target)"
2821                         exit 1
2822                 fi
2823                 #
2824                 # If destination specification is not in the form of
2825                 # host_spec:remote_dir then assume it is just remote hostname
2826                 # and append a colon and destination directory formed from
2827                 # local webrev directory name.
2828                 #
2829                 typeset target_no_prefix=${remote_target##*://}
2830                 if [[ ${target_no_prefix} == *:* ]]; then
2831                         if [[ "${remote_target}" == *: ]]; then
2832                                 remote_target=${remote_target}${WNAME}
2833                         fi
2834                 else
```

```
2835                        if [[ ${target_no_prefix} == */* ]]; then
2836                                print "ERROR: badly formed upload URI" \
2837                                    "($remote_target)"
2838                                exit 1
2839                        else
2840                                remote_target=${remote_target}:${WNAME}
2841                        fi
2842                fi
2843        fi

2845        #
2846        # Strip trailing slash. Each upload method will deal with directory
2847        # specification separately.
2848        #
2849        remote_target=${remote_target%/}
2850 fi

2852 #
2853 # Option -D by itself (option -U not present) implies no webrev generation.
2854 #
2855 if [[ -z $Uflag && -n $Dflag ]]; then
2856         delete_webrev 1 1
2857         exit $?
2858 fi

2860 #
2861 # Do not generate the webrev, just upload it or delete it.
2862 #
2863 if [[ -n $nflag ]]; then
2864         if [[ -n $Dflag ]]; then
2865                 delete_webrev 1 1
2866                 (( $? == 0 )) || exit $?
2867         fi
2868         if [[ -n $Uflag ]]; then
2869                 upload_webrev
2870                 exit $?
2871         fi
2872 fi

2874 if [ "${WDIR%%/*}" ]; then
2875         WDIR=$PWD/$WDIR
2876 fi

2878 if [[ ! -d $WDIR ]]; then
2879         mkdir -p $WDIR
2880         (( $? != 0 )) && exit 1
2881 fi

2883 #
2884 # Summarize what we're going to do.
2885 #
2886 print "      Workspace: ${PRETTY_CWS:-$CWS}"
2887 if [[ -n $parent_webrev ]]; then
2888         print "Compare against: webrev at $parent_webrev"
2889 else
2890         print "Compare against: ${PRETTY_PWS:-$PWS}"
2891 fi

2893 [[ -n $INCLUDE_FILE ]] && print "      Including: $INCLUDE_FILE"
2894 print "      Output to: $WDIR"

2896 #
2897 # Save the file list in the webrev dir
2898 #
2899 [[ ! $FLIST -ef $WDIR/file.list ]] && cp $FLIST $WDIR/file.list
```



```
2901 rm -f $WDIR/$WNAME.patch
2902 rm -f $WDIR/$WNAME.ps
2903 rm -f $WDIR/$WNAME.pdf

2905 touch $WDIR/$WNAME.patch

2907 print "   Output Files:"

2909 #
2910 # Clean up the file list: Remove comments, blank lines and env variables.
2911 #
2912 $SED -e "s/#.*$//" -e "/=/d" -e "/^[   ]*$/d" $FLIST > /tmp/$$.flist.clean
2913 FLIST=/tmp/$$.flist.clean

2915 #
2916 # For Mercurial, create a cache of manifest entries.
2917 #
2918 if [[ $SCM_MODE == "mercurial" ]]; then
2919         #
2920         # Transform the FLIST into a temporary sed script that matches
2921         # relevant entries in the Mercurial manifest as follows:
2922         # 1) The script will be used against the parent revision manifest,
2923         #    so for FLIST lines that have two filenames (a renamed file)
2924         #    keep only the old name.
2925         # 2) Escape all forward slashes the filename.
2926         # 3) Change the filename into another sed command that matches
2927         #    that file in "hg manifest -v" output:  start of line, three
2928         #    octal digits for file permissions, space, a file type flag
2929         #    character, space, the filename, end of line.
2930         # 4) Eliminate any duplicate entries.  (This can occur if a
2931         #    file has been used as the source of an hg cp and it's
2932         #    also been modified in the same changeset.)
2933         #
2934         SEDFILE=/tmp/$$.manifest.sed
2935         $SED '
2936                 s#^[^ ]* ##
2937                 s#/#\\/#g
2938                 s#^.*$#/^... . &$/p#
2939         ' < $FLIST | $SORT -u > $SEDFILE

2941         #
2942         # Apply the generated script to the output of "hg manifest -v"
2943         # to get the relevant subset for this webrev.
2944         #
2945         HG_PARENT_MANIFEST=/tmp/$$.manifest
2946         hg -R $CWS manifest -v -r $HG_PARENT |
2947             $SED -n -f $SEDFILE > $HG_PARENT_MANIFEST
2948 fi

2950 #
2951 # First pass through the files: generate the per-file webrev HTML-files.
2952 #
2953 cat $FLIST | while read LINE
2954 do
2955         set - $LINE
2956         P=$1

2958         #
2959         # Normally, each line in the file list is just a pathname of a
2960         # file that has been modified or created in the child.  A file
2961         # that is renamed in the child workspace has two names on the
2962         # line: new name followed by the old name.
2963         #
2964         oldname=""
2965         oldpath=""
2966         rename=
```

```
2967         if [[ $# -eq 2 ]]; then
2968                 PP=$2                   # old filename
2969                 if [[ -f $PP ]]; then
2970                         oldname=" (copied from $PP)"
2971                 else
2972                         oldname=" (renamed from $PP)"
2973                 fi
2974                 oldpath="$PP"
2975                 rename=1
2976                 PDIR=${PP%/*}
2977                 if [[ $PDIR == $PP ]]; then
2978                         PDIR="."   # File at root of workspace
2979                 fi

2981                 PF=${PP##*/}

2983                 DIR=${P%/*}
2984                 if [[ $DIR == $P ]]; then
2985                         DIR="."   # File at root of workspace
2986                 fi

2988                 F=${P##*/}

2990         else
2991                 DIR=${P%/*}
2992                 if [[ "$DIR" == "$P" ]]; then
2993                         DIR="."   # File at root of workspace
2994                 fi

2996                 F=${P##*/}

2998                 PP=$P
2999                 PDIR=$DIR
3000                 PF=$F
3001         fi

3003         COMM=‘getcomments html $P $PP‘

3005         print "\t$P$oldname\n\t\t\c"

3007         # Make the webrev mirror directory if necessary
3008         mkdir -p $WDIR/$DIR

3010         #
3011         # We stash old and new files into parallel directories in $WDIR
3012         # and do our diffs there.  This makes it possible to generate
3013         # clean looking diffs which don’t have absolute paths present.
3014         #

3016         build_old_new "$WDIR" "$PWS" "$PDIR" "$PF" "$CWS" "$DIR" "$F" || \
3017             continue

3019         #
3020         # Keep the old PWD around, so we can safely switch back after
3021         # diff generation, such that build_old_new runs in a
3022         # consistent environment.
3023         #
3024         OWD=$PWD
3025         cd $WDIR/raw_files
3026         ofile=old/$PDIR/$PF
3027         nfile=new/$DIR/$F

3029         mv_but_nodiff=
3030         cmp $ofile $nfile > /dev/null 2>&1
3031         if [[ $? == 0 && $rename == 1 ]]; then
3032                 mv_but_nodiff=1
```



```
3033         fi

3035         #
3036         # If we have old and new versions of the file then run the appropriate
3037         # diffs.  This is complicated by a couple of factors:
3038         #
3039         #       - renames must be handled specially: we emit a ’remove’
3040         #         diff and an ’add’ diff
3041         #       - new files and deleted files must be handled specially
3042         #       - Solaris patch(1m) can’t cope with file creation
3043         #         (and hence renames) as of this writing.
3044         #       - To make matters worse, gnu patch doesn’t interpret the
3045         #         output of Solaris diff properly when it comes to
3046         #         adds and deletes.  We need to do some "cleansing"
3047         #         transformations:
3048         #           [to add a file] @@ -1,0 +X,Y @@  -->  @@ -0,0 +X,Y @@
3049         #           [to del a file] @@ -X,Y +1,0 @@  -->  @@ -X,Y +0,0 @@
3050         #
3051         cleanse_rmfile="$SED ’s/^\(@@ [0-9+,-]*\) [0-9+,-]* @@$/\1 +0,0 @@/’"
3052         cleanse_newfile="$SED ’s/^@@ [0-9+,-]* \([0-9+,-]* @@\)$/@@ -0,0 \1/’"

3054         rm -f $WDIR/$DIR/$F.patch
3055         if [[ -z $rename ]]; then
3056                 if [ ! -f "$ofile" ]; then
3057                         diff -u /dev/null $nfile | sh -c "$cleanse_newfile" \
3058                             > $WDIR/$DIR/$F.patch
3059                 elif [ ! -f "$nfile" ]; then
3060                         diff -u $ofile /dev/null | sh -c "$cleanse_rmfile" \
3061                             > $WDIR/$DIR/$F.patch
3062                 else
3063                         diff -u $ofile $nfile > $WDIR/$DIR/$F.patch
3064                 fi
3065         else
3066                 diff -u $ofile /dev/null | sh -c "$cleanse_rmfile" \
3067                     > $WDIR/$DIR/$F.patch

3069                 diff -u /dev/null $nfile | sh -c "$cleanse_newfile" \
3070                     >> $WDIR/$DIR/$F.patch
3071         fi

3073         #
3074         # Tack the patch we just made onto the accumulated patch for the
3075         # whole wad.
3076         #
3077         cat $WDIR/$DIR/$F.patch >> $WDIR/$WNAME.patch

3079         print " patch\c"

3081         if [[ -f $ofile && -f $nfile && -z $mv_but_nodiff ]]; then

3083                 ${CDIFFCMD:-diff -bt -C 5} $ofile $nfile > $WDIR/$DIR/$F.cdiff
3084                 diff_to_html $F $DIR/$F "C" "$COMM" < $WDIR/$DIR/$F.cdiff \
3085                     > $WDIR/$DIR/$F.cdiff.html
3086                 print " cdiffs\c"

3088                 ${UDIFFCMD:-diff -bt -U 5} $ofile $nfile > $WDIR/$DIR/$F.udiff
3089                 diff_to_html $F $DIR/$F "U" "$COMM" < $WDIR/$DIR/$F.udiff \
3090                     > $WDIR/$DIR/$F.udiff.html

3092                 print " udiffs\c"

3094                 if [[ -x $WDIFF ]]; then
3095                         $WDIFF -c "$COMM" \
3096                             -t "$WNAME Wdiff $DIR/$F" $ofile $nfile > \
3097                             $WDIR/$DIR/$F.wdiff.html 2>/dev/null
3098                         if [[ $? -eq 0 ]]; then
```

```
3099                                    print " wdiffs\c"
3100                            else
3101                                    print " wdiffs[fail]\c"
3102                            fi
3103                    fi

3105                    sdiff_to_html $ofile $nfile $F $DIR "$COMM" \
3106                        > $WDIR/$DIR/$F.sdiff.html
3107                    print " sdiffs\c"

3109                    print " frames\c"

3111                    rm -f $WDIR/$DIR/$F.cdiff $WDIR/$DIR/$F.udiff

3113                    difflines $ofile $nfile > $WDIR/$DIR/$F.count

3115            elif [[ -f $ofile && -f $nfile && -n $mv_but_nodiff ]]; then
3116                    # renamed file: may also have differences
3117                    difflines $ofile $nfile > $WDIR/$DIR/$F.count
3118            elif [[ -f $nfile ]]; then
3119                    # new file: count added lines
3120                    difflines /dev/null $nfile > $WDIR/$DIR/$F.count
3121            elif [[ -f $ofile ]]; then
3122                    # old file: count deleted lines
3123                    difflines $ofile /dev/null > $WDIR/$DIR/$F.count
3124            fi

3126            #
3127            # Now we generate the postscript for this file.  We generate diffs
3128            # only in the event that there is delta, or the file is new (it seems
3129            # tree-killing to print out the contents of deleted files).
3130            #
3131            if [[ -f $nfile ]]; then
3132                    ocr=$ofile
3133                    [[ ! -f $ofile ]] && ocr=/dev/null

3135                    if [[ -z $mv_but_nodiff ]]; then
3136                            textcomm=`getcomments text $P $PP`
3137                            if [[ -x $CODEREVIEW ]]; then
3138                                    $CODEREVIEW -y "$textcomm" \
3139                                        -e $ocr $nfile \
3140                                        > /tmp/$$.psfile 2>/dev/null &&
3141                                        cat /tmp/$$.psfile >> $WDIR/$WNAME.ps
3142                                    if [[ $? -eq 0 ]]; then
3143                                            print " ps\c"
3144                                    else
3145                                            print " ps[fail]\c"
3146                                    fi
3147                            fi
3148                    fi
3149            fi

3151            if [[ -f $ofile ]]; then
3152                    source_to_html Old $PP < $ofile > $WDIR/$DIR/$F-.html
3153                    print " old\c"
3154            fi

3156            if [[ -f $nfile ]]; then
3157                    source_to_html New $P < $nfile > $WDIR/$DIR/$F.html
3158                    print " new\c"
3159            fi

3161            cd $OWD

3163            print
3164    done
```



```
3166    frame_nav_js > $WDIR/ancnav.js
3167    frame_navigation > $WDIR/ancnav.html

3169    if [[ ! -f $WDIR/$WNAME.ps ]]; then
3170            print " Generating PDF: Skipped: no output available"
3171    elif [[ -x $CODEREVIEW && -x $PS2PDF ]]; then
3172            print " Generating PDF: \c"
3173            fix_postscript $WDIR/$WNAME.ps | $PS2PDF - > $WDIR/$WNAME.pdf
3174            print "Done."
3175    else
3176            print " Generating PDF: Skipped: missing 'ps2pdf' or 'codereview'"
3177    fi

3179    # If we're in OpenSolaris mode and there's a closed dir under $WDIR,
3180    # delete it - prevent accidental publishing of closed source

3182    if [[ -n "$Oflag" ]]; then
3183            $FIND $WDIR -type d -name closed -exec /bin/rm -rf {} \;
3184    fi

3186    # Now build the index.html file that contains
3187    # links to the source files and their diffs.

3189    cd $CWS

3191    # Save total changed lines for Code Inspection.
3192    print "$TOTL" > $WDIR/TotalChangedLines

3194    print "     index.html: \c"
3195    INDEXFILE=$WDIR/index.html
3196    exec 3<&1                       # duplicate stdout to FD3.
3197    exec 1<&-                       # Close stdout.
3198    exec > $INDEXFILE               # Open stdout to index file.

3200    print "$HTML<head>$STDHEAD"
3201    print "<title>$WNAME</title>"
3202    print "</head>"
3203    print "<body id=\"SUNWwebrev\">"
3204    print "<div class=\"summary\">"
3205    print "<h2>Code Review for $WNAME</h2>"

3207    print "<table>"

3209    #
3210    # Get the preparer's name:
3211    #
3212    # If the SCM detected is Mercurial, and the configuration property
3213    # ui.username is available, use that, but be careful to properly escape
3214    # angle brackets (HTML syntax characters) in the email address.
3215    #
3216    # Otherwise, use the current userid in the form "John Doe (jdoe)", but
3217    # to maintain compatibility with passwd(4), we must support '&' substitutions.
3218    #
3219    preparer=
3220    if [[ "$SCM_MODE" == mercurial ]]; then
3221            preparer=`hg showconfig ui.username 2>/dev/null`
3222            if [[ -n "$preparer" ]]; then
3223                    preparer="$(echo "$preparer" | html_quote)"
3224            fi
3225    fi
3226    if [[ -z "$preparer" ]]; then
3227            preparer=$(
3228                $PERL -e '
3229                    ($login, $pw, $uid, $gid, $quota, $cmt, $gcos) = getpwuid($<);
3230                    if ($login) {
```

```
3231                    $gcos =~ s/\&/ucfirst($login)/e;
3232                    printf "%s (%s)\n", $gcos, $login;
3233                } else {
3234                    printf "(unknown)\n";
3235                }
3236        ')
3237 fi

3239 PREPDATE=$(LC_ALL=C /usr/bin/date +%Y-%b-%d\ %R\ %z\ %Z)
3240 print "<tr><th>Prepared by:</th><td>$preparer on $PREPDATE</td></tr>"
3241 print "<tr><th>Workspace:</th><td>${PRETTY_CWS:-$CWS}"
3242 print "</td></tr>"
3243 print "<tr><th>Compare against:</th><td>"
3244 if [[ -n $parent_webrev ]]; then
3245         print "webrev at $parent_webrev"
3246 else
3247         print "${PRETTY_PWS:-$PWS}"
3248 fi
3249 print "</td></tr>"
3250 print "<tr><th>Summary of changes:</th><td>"
3251 printCI $TOTL $TINS $TDEL $TMOD $TUNC
3252 print "</td></tr>"

3254 if [[ -f $WDIR/$WNAME.patch ]]; then
3255         wpatch_url="$(print $WNAME.patch | url_encode)"
3256         print "<tr><th>Patch of changes:</th><td>"
3257         print "<a href=\"$wpatch_url\">$WNAME.patch</a></td></tr>"
3258 fi
3259 if [[ -f $WDIR/$WNAME.pdf ]]; then
3260         wpdf_url="$(print $WNAME.pdf | url_encode)"
3261         print "<tr><th>Printable review:</th><td>"
3262         print "<a href=\"$wpdf_url\">$WNAME.pdf</a></td></tr>"
3263 fi

3265 if [[ -n "$iflag" ]]; then
3266         print "<tr><th>Author comments:</th><td><div>"
3267         cat /tmp/$$.include
3268         print "</div></td></tr>"
3269 fi
3270 print "</table>"
3271 print "</div>"

3273 #
3274 # Second pass through the files: generate the rest of the index file
3275 #
3276 cat $FLIST | while read LINE
3277 do
3278         set - $LINE
3279         P=$1

3281         if [[ $# == 2 ]]; then
3282                 PP=$2
3283                 oldname="$PP"
3284         else
3285                 PP=$P
3286                 oldname=""
3287         fi

3289         mv_but_nodiff=
3290         cmp $WDIR/raw_files/old/$PP $WDIR/raw_files/new/$P > /dev/null 2>&1
3291         if [[ $? == 0 && -n "$oldname" ]]; then
3292                 mv_but_nodiff=1
3293         fi

3295         DIR=${P%/*}
3296         if [[ $DIR == $P ]]; then
```



```
3297                 DIR="."   # File at root of workspace
3298         fi

3300         # Avoid processing the same file twice.
3301         # It's possible for renamed files to
3302         # appear twice in the file list

3304         F=$WDIR/$P

3306         print "<p>"

3308         # If there's a diffs file, make diffs links

3310         if [[ -f $F.cdiff.html ]]; then
3311                 cdiff_url="$(print $P.cdiff.html | url_encode)"
3312                 udiff_url="$(print $P.udiff.html | url_encode)"
3313                 print "<a href=\"$cdiff_url\">Cdiffs</a>"
3314                 print "<a href=\"$udiff_url\">Udiffs</a>"

3316                 if [[ -f $F.wdiff.html && -x $WDIFF ]]; then
3317                         wdiff_url="$(print $P.wdiff.html | url_encode)"
3318                         print "<a href=\"$wdiff_url\">Wdiffs</a>"
3319                 fi

3321                 sdiff_url="$(print $P.sdiff.html | url_encode)"
3322                 print "<a href=\"$sdiff_url\">Sdiffs</a>"

3324                 frames_url="$(print $P.frames.html | url_encode)"
3325                 print "<a href=\"$frames_url\">Frames</a>"
3326         else
3327                 print " ------ ------ ------"

3329                 if [[ -x $WDIFF ]]; then
3330                         print " ------"
3331                 fi

3333                 print " ------"
3334         fi

3336         # If there's an old file, make the link

3338         if [[ -f $F-.html ]]; then
3339                 oldfile_url="$(print $P-.html | url_encode)"
3340                 print "<a href=\"$oldfile_url\">Old</a>"
3341         else
3342                 print " ---"
3343         fi

3345         # If there's an new file, make the link

3347         if [[ -f $F.html ]]; then
3348                 newfile_url="$(print $P.html | url_encode)"
3349                 print "<a href=\"$newfile_url\">New</a>"
3350         else
3351                 print " ---"
3352         fi

3354         if [[ -f $F.patch ]]; then
3355                 patch_url="$(print $P.patch | url_encode)"
3356                 print "<a href=\"$patch_url\">Patch</a>"
3357         else
3358                 print " -----"
3359         fi

3361         if [[ -f $WDIR/raw_files/new/$P ]]; then
3362                 rawfiles_url="$(print raw_files/new/$P | url_encode)"
```

```
3363                 print "<a href=\"$rawfiles_url\">Raw</a>"
3364         else
3365                 print " ---"
3366         fi

3368         print "<b>$P</b>"

3370         # For renamed files, clearly state whether or not they are modified
3371         if [[ -f "$oldname" ]]; then
3372                 if [[ -n "$mv_but_nodiff" ]]; then
3373                         print "<i>(copied from $oldname)</i>"
3374                 else
3375                         print "<i>(copied and modified from $oldname)</i>"
3376                 fi
3377         elif [[ -n "$oldname" ]]; then
3378                 if [[ -n "$mv_but_nodiff" ]]; then
3379                         print "<i>(renamed from $oldname)</i>"
3380                 else
3381                         print "<i>(renamed and modified from $oldname)</i>"
3382                 fi
3383         fi

3385         # If there's an old file, but no new file, the file was deleted
3386         if [[ -f $F-.html && ! -f $F.html ]]; then
3387                 print " <i>(deleted)</i>"
3388         fi

3390         #
3391         # Check for usr/closed and deleted_files/usr/closed
3392         #
3393         if [ ! -z "$Oflag" ]; then
3394                 if [[ $P == usr/closed/* || \
3395                     $P == deleted_files/usr/closed/* ]]; then
3396                         print "  <i>Closed source: omitted from" \
3397                             "this review</i>"
3398                 fi
3399         fi

3401         print "</p>"
3402         # Insert delta comments

3404         print "<blockquote><pre>"
3405         getcomments html $P $PP
3406         print "</pre>"

3408         # Add additional comments comment

3410         print "<!-- Add comments to explain changes in $P here -->"

3412         # Add count of changes.

3414         if [[ -f $F.count ]]; then
3415             cat $F.count
3416             rm $F.count
3417         fi

3419         if [[ $SCM_MODE == "mercurial" ||
2785         if [[ $SCM_MODE == "teamware" ||
2786             $SCM_MODE == "mercurial" ||
3420             $SCM_MODE == "unknown" ]]; then

3422                 # Include warnings for important file mode situations:
3423                 # 1) New executable files
3424                 # 2) Permission changes of any kind
3425                 # 3) Existing executable files
```



```
3427                 old_mode=
3428                 if [[ -f $WDIR/raw_files/old/$PP ]]; then
3429                         old_mode=‘get_file_mode $WDIR/raw_files/old/$PP‘
3430                 fi

3432                 new_mode=
3433                 if [[ -f $WDIR/raw_files/new/$P ]]; then
3434                         new_mode=‘get_file_mode $WDIR/raw_files/new/$P‘
3435                 fi

3437                 if [[ -z "$old_mode" && "$new_mode" = *[1357]* ]]; then
3438                         print "<span class=\"chmod\">"
3439                         print "<p>new executable file: mode $new_mode</p>"
3440                         print "</span>"
3441                 elif [[ -n "$old_mode" && -n "$new_mode" &&
3442                     "$old_mode" != "$new_mode" ]]; then
3443                         print "<span class=\"chmod\">"
3444                         print "<p>mode change: $old_mode to $new_mode</p>"
3445                         print "</span>"
3446                 elif [[ "$new_mode" = *[1357]* ]]; then
3447                         print "<span class=\"chmod\">"
3448                         print "<p>executable file: mode $new_mode</p>"
3449                         print "</span>"
3450                 fi
3451         fi

3453         print "</blockquote>"
3454 done

3456 print
3457 print
3458 print "<hr></hr>"
3459 print "<p style=\"font-size: small\">"
3460 print "This code review page was prepared using <b>$0</b>."
3461 print "Webrev is maintained by the <a href=\"http://www.illumos.org\">"
3462 print "illumos</a> project.  The latest version may be obtained"
3463 print "<a href=\"http://src.illumos.org/source/xref/illumos-gate/usr/src/tools/s
3464 print "</body>"
3465 print "</html>"

3467 exec 1<&-                       # Close FD 1.
3468 exec 1<&3                       # dup FD 3 to restore stdout.
3469 exec 3<&-                       # close FD 3.

3471 print "Done."

3473 #
3474 # If remote deletion was specified and fails do not continue.
3475 #
3476 if [[ -n $Dflag ]]; then
3477         delete_webrev 1 1
3478         (( $? == 0 )) || exit $?
3479 fi

3481 if [[ -n $Uflag ]]; then
3482         upload_webrev
3483         exit $?
3484 fi
```