****
   11225 Fri Feb 2 23:18:32 2018
new/usr/src/pkg/manifests/developer-build-onbld.mf
9001 cdm is useless, remove it
9002 webrev should know how to get the git user name
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
****

```
#
# 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) 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2015, Richard Lowe
# Copyright 2012, Piotr Jasiukajtis
# Copyright 2014 Garrett D'Amore <garrett@damore.org>
# Copyright (c) 2014, Joyent, Inc.
# Copyright 2016 Toomas Soome <tsoome@me.com>
#

set name=pkg.fmri value=pkg:/developer/build/onbld@$(PKGVERS)
set name=pkg.description value="tools used to build the OS-Net consolidation"
set name=pkg.summary value="OS-Net Build Tools"
set name=info.classification \
    value="org.opensolaris.category.2008:Development/Distribution Tools"

#
# This package should not be incorporated.  This allows the tools
# to be upgraded without upgrading the entire system.
#
set name=org.opensolaris.noincorp value=true
set name=variant.arch value=$(ARCH)
dir path=opt group=sys
dir path=opt/onbld
dir path=opt/onbld/bin
dir path=opt/onbld/bin/$(ARCH)
dir path=opt/onbld/env
dir path=opt/onbld/etc
dir path=opt/onbld/etc/exception_lists
dir path=opt/onbld/lib
dir path=opt/onbld/lib/$(ARCH)
dir path=opt/onbld/lib/$(ARCH)/64
dir path=opt/onbld/lib/perl
dir path=opt/onbld/lib/python$(PYTHON_VERSION)
dir path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld
dir path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks
dir path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm
dir path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/hgext
```

```
dir path=opt/onbld/man
dir path=opt/onbld/man/man1onbld
dir path=opt/onbld/share
$(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/aw mode=0555
$(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/btxld mode=0555
$(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/chk4ubin mode=0555
file path=opt/onbld/bin/$(ARCH)/codereview mode=0555
file path=opt/onbld/bin/$(ARCH)/cscope-fast mode=0555
file path=opt/onbld/bin/$(ARCH)/ctfconvert mode=0555
file path=opt/onbld/bin/$(ARCH)/ctfdump mode=0555
file path=opt/onbld/bin/$(ARCH)/ctfmerge mode=0555
file path=opt/onbld/bin/$(ARCH)/ctfstabs mode=0555
file path=opt/onbld/bin/$(ARCH)/ctfstrip mode=0555
file path=opt/onbld/bin/$(ARCH)/cw mode=0555
$(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/elfextract mode=0555
file path=opt/onbld/bin/$(ARCH)/findunref mode=0555
$(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/forth mode=0555
$(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/forth_preload.so.1 mode=0555
file path=opt/onbld/bin/$(ARCH)/install mode=0555
file path=opt/onbld/bin/$(ARCH)/lintdump mode=0555
file path=opt/onbld/bin/$(ARCH)/localedef mode=0555
file path=opt/onbld/bin/$(ARCH)/make mode=0555
$(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/mbh_patch mode=0555
file path=opt/onbld/bin/$(ARCH)/ndrgen mode=0555
file path=opt/onbld/bin/$(ARCH)/ndrgen1 mode=0555
file path=opt/onbld/bin/$(ARCH)/protocmp mode=0555
file path=opt/onbld/bin/$(ARCH)/protolist mode=0555
$(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/stabs mode=0555
file path=opt/onbld/bin/$(ARCH)/tic mode=0555
$(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/tokenize mode=0555
$(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/tokenize.exe mode=0555
file path=opt/onbld/bin/$(ARCH)/zic mode=0555
file path=opt/onbld/bin/Install mode=0555
file path=opt/onbld/bin/bldenv mode=0555
file path=opt/onbld/bin/bringovercheck mode=0555
file path=opt/onbld/bin/build_cscope mode=0555
file path=opt/onbld/bin/cddlchk mode=0555
file path=opt/onbld/bin/check_rtime mode=0555
file path=opt/onbld/bin/checkpaths mode=0555
file path=opt/onbld/bin/copyrightchk mode=0555
file path=opt/onbld/bin/cstyle mode=0555
file path=opt/onbld/bin/elfcmp mode=0555
file path=opt/onbld/bin/find_elf mode=0555
file path=opt/onbld/bin/findcrypto mode=0555
file path=opt/onbld/bin/flg.flp mode=0555
file path=opt/onbld/bin/genoffsets mode=0555
file path=opt/onbld/bin/git-pbchk mode=0555
file path=opt/onbld/bin/hdrchk mode=0555
file path=opt/onbld/bin/hg-active mode=0555
file path=opt/onbld/bin/hgsetup mode=0555
file path=opt/onbld/bin/interface_check mode=0555
file path=opt/onbld/bin/interface_cmp mode=0555
file path=opt/onbld/bin/jstyle mode=0555
file path=opt/onbld/bin/mapfilechk mode=0555
file path=opt/onbld/bin/nightly mode=0555
file path=opt/onbld/bin/onu mode=0555
file path=opt/onbld/bin/protocmp.terse mode=0555
file path=opt/onbld/bin/sccscheck mode=0555
file path=opt/onbld/bin/signit mode=0555
file path=opt/onbld/bin/signproto mode=0555
file path=opt/onbld/bin/validate_flg mode=0555
file path=opt/onbld/bin/validate_paths mode=0555
file path=opt/onbld/bin/validate_pkg mode=0555
file path=opt/onbld/bin/wdiff mode=0555
file path=opt/onbld/bin/webrev mode=0555
file path=opt/onbld/bin/which_scm mode=0555
file path=opt/onbld/bin/ws mode=0555
file path=opt/onbld/bin/wsdiff mode=0555
file path=opt/onbld/bin/xref mode=0555
file path=opt/onbld/bin/xref.mk
file path=opt/onbld/env/illumos
file path=opt/onbld/etc/exception_lists/check_rtime
file path=opt/onbld/etc/exception_lists/interface_check
file path=opt/onbld/etc/exception_lists/interface_cmp
file path=opt/onbld/etc/hgstyle
file path=opt/onbld/etc/its.conf
file path=opt/onbld/etc/its.reg
file path=opt/onbld/lib/$(ARCH)/64/libmakestate.so.1
file path=opt/onbld/lib/$(ARCH)/libdwarf.so.1
file path=opt/onbld/lib/$(ARCH)/libmakestate.so.1
file path=opt/onbld/lib/perl/onbld_elfmod.pm
file path=opt/onbld/lib/perl/onbld_elfmod_vertype.pm
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/CStyle.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/CStyle.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Cddl.py mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Cddl.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/CmtBlk.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/CmtBlk.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Comments.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Comments.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Copyright.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Copyright.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/DbLookups.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/DbLookups.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/HdrChk.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/HdrChk.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/JStyle.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/JStyle.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Keywords.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Keywords.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/ManLint.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/ManLint.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Mapfile.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Mapfile.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/ProcessCheck.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/ProcessCheck.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/SpellCheck.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/SpellCheck.pyc \
```

```
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/__init__.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/__init__.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/Backup.py mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/Backup.pyc mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/Ignore.py mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/Ignore.pyc mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/Version.py mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/Version.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/WorkSpace.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/WorkSpace.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/__init__.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/__init__.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/__init__.py mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/__init__.pyc mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/hgext/__init__.py \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/hgext/__init__.pyc \
    mode=0444
file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/hgext/cdm.py mode=0444
file path=opt/onbld/man/man1onbld/Install.1onbld
file path=opt/onbld/man/man1onbld/bldenv.1onbld
file path=opt/onbld/man/man1onbld/bringovercheck.1onbld
file path=opt/onbld/man/man1onbld/btxld.1onbld
file path=opt/onbld/man/man1onbld/cddlchk.1onbld
file path=opt/onbld/man/man1onbld/check_rtime.1onbld
file path=opt/onbld/man/man1onbld/checkpaths.1onbld
file path=opt/onbld/man/man1onbld/codereview.1onbld
file path=opt/onbld/man/man1onbld/cstyle.1onbld
file path=opt/onbld/man/man1onbld/cw.1onbld
file path=opt/onbld/man/man1onbld/find_elf.1onbld
file path=opt/onbld/man/man1onbld/findunref.1onbld
file path=opt/onbld/man/man1onbld/flg.flp.1onbld
file path=opt/onbld/man/man1onbld/git-pbchk.1onbld
file path=opt/onbld/man/man1onbld/hdrchk.1onbld
file path=opt/onbld/man/man1onbld/hgsetup.1onbld
file path=opt/onbld/man/man1onbld/interface_check.1onbld
file path=opt/onbld/man/man1onbld/interface_cmp.1onbld
file path=opt/onbld/man/man1onbld/jstyle.1onbld
file path=opt/onbld/man/man1onbld/lintdump.1onbld
file path=opt/onbld/man/man1onbld/mapfilechk.1onbld
file path=opt/onbld/man/man1onbld/ndrgen.1onbld
file path=opt/onbld/man/man1onbld/nightly.1onbld
file path=opt/onbld/man/man1onbld/onu.1onbld
file path=opt/onbld/man/man1onbld/sccscheck.1onbld
file path=opt/onbld/man/man1onbld/signit.1onbld
file path=opt/onbld/man/man1onbld/signproto.1onbld
file path=opt/onbld/man/man1onbld/webrev.1onbld
file path=opt/onbld/man/man1onbld/which_scm.1onbld
file path=opt/onbld/man/man1onbld/ws.1onbld
file path=opt/onbld/man/man1onbld/wsdiff.1onbld
file path=opt/onbld/man/man1onbld/xref.1onbld
file path=opt/onbld/share/make.rules mode=0444
hardlink path=opt/onbld/bin/$(ARCH)/install.bin target=./install
legacy pkg=SUNWonbld desc="tools used to build the OS-Net consolidation" \
    name="OS-Net Build Tools" version=11.11,REV=2009.10.22
license cr_Sun license=cr_Sun
license lic_CDDL license=lic_CDDL
license usr/src/tools/ctf/dwarf/THIRDPARTYLICENSE \
    license=usr/src/tools/ctf/dwarf/THIRDPARTYLICENSE
license usr/src/tools/onbld/THIRDPARTYLICENSE \
    license=usr/src/tools/onbld/THIRDPARTYLICENSE
link path=opt/onbld/bin/$(ARCH)/dmake target=make
link path=opt/onbld/bin/git-nits target=git-pbchk
link path=opt/onbld/lib/python target=python$(PYTHON_VERSION)
link path=opt/onbld/man/man1onbld/git-nits.1onbld target=git-pbchk.1onbld
# webrev(1) requires ps2pdf
depend fmri=print/filter/ghostscript type=require
# nightly(1) uses wget
depend fmri=web/wget type=require
```

```
************************************************************
    3201 Fri Feb  2 23:18:33 2018
new/usr/src/tools/Makefile
9001 cdm is useless, remove it
9002 webrev should know how to get the git user name
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
************************************************************
#
# 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) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2014 Garrett D'Amore <garrett@damore.org>
# Copyright 2016 Toomas Soome <tsoome@me.com>
# Copyright (c) 2016, Chris Fraire <cfraire@me.com>.
#

include ../Makefile.master

# Bootstrap problem --
# 'cw' must be built before anything else can be built.

BOOT_SUBDIRS= \
        cw

COMMON_SUBDIRS= \
        codereview \
        codesign \
        cscope-fast \
        ctf \
        env \
        findunref \
        install.bin \
        lintdump \
        make \
        ndrgen \
        onbld \
        protocmp \
        protolist \
        scripts

#
#  special versions of commands for use only in build
#
UNSHIPPED_SUBDIRS = \
        localedef \
        mandoc \
```

```
        tic \
        vtfontcvt \
        zic

sparc_SUBDIRS= \
        chk4ubin \
        stabs \
        tokenize

i386_SUBDIRS=           \
        aw              \
        elfextract      \
        mbh_patch       \
        btxld

LINTSUBDIRS= \
        codereview \
        ctf \
        cw \
        findunref \
        lintdump \
        ndrgen \
        protocmp \
        protolist

SUBDIRS= \
        $($(MACH)_SUBDIRS) \
        $(COMMON_SUBDIRS) \
        $(UNSHIPPED_SUBDIRS)

include Makefile.tools

ROOTDIRS= \
        $(ROOTOPT) \
        $(ROOTONBLD) \
        $(ROOTONBLD)/bin \
        $(ROOTONBLD)/bin/$(MACH) \
        $(ROOTONBLD)/lib \
        $(ROOTONBLD)/lib/$(MACH) \
        $(ROOTONBLD)/lib/$(MACH)/64 \
        $(ROOTONBLD)/lib/perl \
        $(ROOTONBLD)/lib/python$(PYTHON_VERSION) \
        $(ROOTONBLD)/lib/python$(PYTHON_VERSION)/onbld \
        $(ROOTONBLD)/lib/python$(PYTHON_VERSION)/onbld/Checks \
        $(ROOTONBLD)/lib/python$(PYTHON_VERSION)/onbld/hgext \
        $(ROOTONBLD)/lib/python$(PYTHON_VERSION)/onbld/Scm \
        $(ROOTONBLD)/env \
        $(ROOTONBLD)/etc \
        $(ROOTONBLD)/etc/exception_lists \
        $(ROOTONBLD)/share \
        $(ROOTONBLD)/man \
        $(ROOTONBLD)/man/man1onbld

all :=          TARGET= install
install :=      TARGET= install
clean :=        TARGET= clean
clobber :=      TARGET= clobber
lint :=         TARGET= lint
_msg :=         TARGET= _msg

.KEEP_STATE:

#
# Only create directories in the tools proto area when doing an actual
# build, not a clean or clobber.
#
```

```
DOROOTDIRS= $(ROOTDIRS)
clobber:= DOROOTDIRS=
clean:= DOROOTDIRS=

DOROOTONBLDLIBPY= $(ROOTONBLDLIBPY)
clobber:= DOROOTONBLDLIBPY=
clean:= DOROOTONBLDLIBPY=

all install: $(SUBDIRS)

clean: $(SUBDIRS)

clobber: $(SUBDIRS)
        $(RM) -rf $(TOOLS_PROTO)

lint: $(LINTSUBDIRS)

_msg: $(MSGSUBDIRS)

.PARALLEL: $(SUBDIRS) $(CLOSED_SUBDIRS)

$(SUBDIRS) $(CLOSED_SUBDIRS): $(BOOT_SUBDIRS)

$(BOOT_SUBDIRS) $(SUBDIRS): $$(DOROOTDIRS) $$(DOROOTONBLDLIBPY) FRC
        @cd $@; pwd; $(MAKE) $(TARGET)

$(ROOTDIRS):
        $(INS.dir)

$(ROOTONBLDLIBPY): $(ROOTDIRS)
        $(RM) -r $@; $(SYMLINK) python$(PYTHON_VERSION) $@

make: ctf

FRC:
```

```
***********************************************************
    9944 Fri Feb  2 23:18:33 2018
new/usr/src/tools/README.tools
9001 cdm is useless, remove it
9002 webrev should know how to get the git user name
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
***********************************************************
#
# 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) 1999, 2010, Oracle and/or its affiliates. All rights reserved.


This directory contains the tools used to do a full build of the
OS/Net workspace.  They usually live in the /opt/onbld directory on build
machines. From here, 'make install' will build and install the tools
in $ROOT/opt/onbld.

Layout of /opt/onbld
--------------------

/opt/onbld/etc/abi
        contains Solaris ABI database (ABI_*.db) and exceptions
        for ABI Auditing tool (interface_check, interface_cmp).

/opt/onbld/bin
        basic bin directory - contains scripts.

/opt/onbld/bin/${MACH}
        architecture-specific bin directory for binaries.

/opt/onbld/env
        build environment files.

/opt/onbld/lib
        libraries used by the build tools.

/opt/onbld/lib/python<version>/
        python modules used by the build tools.

/opt/onbld/lib/python<version>/onbld/hgext
        Mercurial extensions.

/opt/onbld/lib/python/
        symlink to the modules directory of the currently preferred
        python version.
        python version.  This exists to retain compatibility both for
```

```
        tools expecting only one supported version of python, and for
        user .hgrc files that expect to find cdm.py in
        /opt/onbld/lib/python/onbld/hgext.

/opt/onbld/man
        rudimentary man pages for some of the tools.


Tool Summary
------------

bldenv
        companion to 'nightly.' Takes the same environment file you
        used with 'nightly,' and starts a shell with the environment
        set up the same way as 'nightly' set it up. This is useful
        if you're trying to quickly rebuild portions of a workspace
        built by 'nightly'. 'ws' should not be used for this since it
        sets the environment up differently and may cause everything
        to rebuild (because of different -I or -L paths).

build_cscope
        builds cscope databases in the uts, the platform subdirectories
        of uts, and in usr/src. Uses cscope-fast.

cdm
        A Mercurial extension providing various commands useful for ON
        development

check_rtime
        checks ELF attributes used by ELF dynamic objects in the proto area.
        Used by 'nightly's -r option, to check a number of ELF runtime
        attributes for consistency with common build rules.  nightly uses
        the -o option to simplify the output for diffing with previous
        build results.  It also uses the -i option to obtain NEEDED and RUNPATH
        entries, which help detect changes in software dependencies and makes
        sure objects don't have any strange runpaths like /opt/SUNWspro/lib.

codereview
        Given two filenames, creates a postscript file with the file
        differences highlighted.

codesign
        Tools for signing cryptographic modules using the official
        Sun release keys stored on a remote signing server. This
        directory contains signit, a client program for signing
        files with the signing server; signproto, a shell script
        that finds crypto modules in $ROOT and signs them using
        signit; and codesign_server.pl, the code that runs on the
        server. The codesign_server code is not used on an ON
        build machine but is kept here for source control purposes.

copyrightchk
        Checks that files have appropriate SMI copyright notices.
        Primarily used by wx

cscope-fast
        The fast version of cscope that we use internally. Seems to work,
        but may need more testing before it's placed in the gate. The source
        just really needs to be here.

cstyle
        checks C source for compliance with OS/Net guidelines.

ctfconvert
        Convert symbolic debugging information in an object file to the Compact
        ANSI-C Type Format (CTF).
```

```
ctfdump
        Decode and display CTF data stored in a raw file or in an ELF file.

ctfmerge
        Merge the CTF data from one or more object files.

elfcmp
        Compares two ELF modules (e.g. .o files, executables) section by
        section.  Useful for determining whether "trivial" changes -
        cstyle, lint, etc - actually changed the code.  The -S option
        is used to test whether two binaries are the same except for
        the elfsign signature.

find_elf
        Search a directory tree for ELF objects, and produce one line of
        output per object. Used by check_rtime and interface_check to locate
        the objects to examine.

findunref
        Finds all files in a source tree that have access times older than a
        certain time and are not in a specified list of exceptions.  Since
        'nightly' timestamps the start of the build, and findunref uses its
        timestamp (by default), this can be used to find all files that were
        unreferenced during a nightly build).  Since some files are only used
        during a SPARC or Intel build, 'findunref' needs to be run on
        workspaces from both architectures and the results need to be merged.
        For instance, if $INTELSRC and $SPARCSRC are set to the usr/src
        directories of your Intel and SPARC nightly workspaces, then you
        can merge the results like so:

        $ findunref $INTELSRC $INTELSRC/tools/findunref/exception_list | \
          sort > ~/unref-i386.out
        $ findunref $SPARCSRC $SPARCSRC/tools/findunref/exception_list | \
          sort > ~/unref-sparc.out
        $ comm -12 ~/unref-i386.out ~/unref-sparc.out > ~/unref.out

hdrchk
        checks headers for compliance with OS/Net standards (form, includes,
        C++ guards).

hgsetup
        creates a basic Mercurial configuration for the user.

hg-active
        helper used by webrev to generate file lists for Mercurial
        workspaces.

install.bin
        binary version of /usr/sbin/install. Used to be vastly faster
        (since /usr/sbin/install is a shell script), but may only be a bit
        faster now. One speedup includes avoiding the name service for the
        well-known, never-changing password entries like 'root' and 'sys.'

interface_check
        detects and reports invalid versioning in ELF objects.
        Optionally generates an interface description file for
        the workspace.

interface_cmp
        Compares two interface description files, as produced by
        interface_check, and flags invalid deviations in ELF object
        versioning between them. interface_cmp can be used between Solaris
        gates to ensure that older releases remain compatible with the
        development gate. It can also be used to validate new changes to
        the development gate before they are integrated.
```

```
lintdump
        dumps the contents of one or more lint libraries; see lintdump(1)

ndrgen
        Network Data Language (NDL) RPC protocol compiler to support DCE
        RPC/MSRPC and SMB/CIFS.  ndrgen takes an input protocol definition
        file (say, proto.ndl) and generates an output C source file
        (proto_ndr.c) containing the Network Data Representation (NDR)
        marshalling routines to implement the RPC protocol.

nightly
        nightly build script. Takes an environment (or 'env') file describing
        such things as the workspace, the parent, and what to build. See
        env/developer and env/gatekeeper for sample, hopefully well-commented
        env files.

protocmp
        compares proto lists and the package definitions. Used by nightly
        to determine if the proto area matches the packages, and to detect
        differences between a childs proto area and a parents.

protocmp.terse
        transforms the output of protocmp into something a bit more friendly

protolist
        create a list of what's in the proto area, to feed to protocmp.


ws
        creates a shell with the environment set up to build in the given
        workspace. Used mostly for non-full-build workspaces, so it sets up
        to pull headers and libraries from the proto area of the parent if
        they aren't in the childs proto area.

tokenize
        Used to build the sun4u boot block.

webrev
        Generates a set of HTML pages that show side-by-side diffs of
        changes in your workspace, for easy communication of code
        review materials.  Can automagically find edited files or use a
        manually-generated list; knows how to use wx's active file for
        lists of checked-out files and proposed SCCS comments.

which_scm
        Reports the current Source Code Management (SCM) system in use
        and the top-level directory of the workspace.

wsdiff
        Detect object differences between two ON proto areas. Used by
        nightly(1) to determine what changed between two builds. Handy
        for identifying the set of built objects impacted by a given
        source change. This information is needed for patch construction.


How to do a full build
----------------------

1. Find an environment file that might do what you want to do. If you're just
   a developer wanting to do a full build in a child of the gate, copy the
   'developer' environment file to a new name (private to you and/or the
   work being done in this workspace, to avoid collisions with others). Then
   edit the file and tailor it to your workspace. Remember that this file
   is a shell script, so it can do more than set environment variables.
```

```
2. Run 'nightly' and give it your environment file as an
   option. 'nightly' will first look for your environment file in
   /opt/onbld/env, and if it's not there then it will look for it as an
   absolute or relative path. Some people put their environment files in
   their workspace to keep them close.

3. When 'nightly' is complete, it will send a summary of what happened to
   $MAILTO. Usually, the less info in the mail the better. If you have failures,
   you can go look at the full log of what happened, generally in
   $CODEMGR_WS/log/log.<date>/nightly.log (the mail_msg it sent and the proto
   list are there too). You can also find the individual build logs, like
   'make clobber' and 'make install' output in $SRC, under names like
   clobber-${MACH}.out and install-${MACH}.out (for a DEBUG build). These
   will be smaller than nightly.log, and maybe more searchable.

Files you have to update to add a tool
--------------------------------------

1.  Add the tool in its appropriate place.
2.  Update the Makefile as required.
3.  Update usr/src/pkg/manifests/developer-build-onbld.mf
4.  Update usr/src/tools/README.tools (this file).
5.  Repeat 1-4 for any man pages.
```

****
*   1394 Fri Feb 2 23:18:34 2018*
new/usr/src/tools/onbld/Makefile
9001 cdm is useless, remove it
9002 webrev should know how to get the git user name
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
****

```
  1 #
  2 # CDDL HEADER START
  3 #
  4 # The contents of this file are subject to the terms of the
  5 # Common Development and Distribution License (the "License").
  6 # You may not use this file except in compliance with the License.
  7 #
  8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
  9 # or http://www.opensolaris.org/os/licensing.
 10 # See the License for the specific language governing permissions
 11 # and limitations under the License.
 12 #
 13 # When distributing Covered Code, include this CDDL HEADER in each
 14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 15 # If applicable, add the following below this CDDL HEADER, with the
 16 # fields enclosed by brackets "[]" replaced with your own identifying
 17 # information: Portions Copyright [yyyy] [name of copyright owner]
 18 #
 19 # CDDL HEADER END
 20 #

 22 #
 23 # Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
 24 #

 26 include $(SRC)/Makefile.master
 27 include ../Makefile.tools

 29 SUBDIRS= \
 30         Checks  \
 31         hgext   \
 31         Scm

 33 PYSRCS = \
 34         __init__.py

 36 PYOBJS =        $(PYSRCS:%.py=%.pyc)
 37 PYTOPDIR =      $(ROOTONBLDLIB)
 38 PYMODDIR =      onbld

 40 all     := TARGET = all
 41 install := TARGET = install
 42 clean   := TARGET = clean
 43 clobber := TARGET = clobber

 45 include ../Makefile.python

 47 .KEEP_STATE:

 49 all: $(PYVERSOBJS) $(SUBDIRS)

 51 install: all $(ROOTPYFILES) $(SUBDIRS)

 53 clean: $(SUBDIRS)

 55 clobber: clean pyclobber $(SUBDIRS)

 57 $(SUBDIRS): FRC
```

```
 58         @cd $@; pwd; $(MAKE) $(TARGET)

 60 FRC:
```

```
**********************************************************
    1210 Fri Feb  2 23:18:35 2018
new/usr/src/tools/onbld/Scm/Makefile
9001 cdm is useless, remove it
9002 webrev should know how to get the git user name
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
#
# 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) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2014, Joyent, Inc.
#

include $(SRC)/Makefile.master
include ../../Makefile.tools

PYSRCS = \
        __init__.py     \
        Ignore.py
        Backup.py       \
        Ignore.py       \
        Version.py      \
        WorkSpace.py

PYOBJS =        $(PYSRCS:%.py=%.pyc)
PYTOPDIR =      $(ROOTONBLDLIB)
PYMODDIR =      onbld/Scm

include ../../Makefile.python

all: $(PYVERSOBJS)

install: all $(ROOTPYFILES)

clean:

clobber: clean pyclobber
```

```
************************************************************
    3904 Fri Feb  2 23:18:38 2018
new/usr/src/tools/scripts/Makefile
9001 cdm is useless, remove it
9002 webrev should know how to get the git user name
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
************************************************************
#
# 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) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
#
# Copyright 2010, Richard Lowe

SHELL=/usr/bin/ksh93

SHFILES= \
        Install \
        bldenv \
        build_cscope \
        bringovercheck \
        checkpaths \
        cstyle \
        elfcmp \
        flg.flp \
        genoffsets \
        hgsetup \
        nightly \
        onu \
        protocmp.terse \
        sccscheck \
        webrev \
        which_scm \
        ws \
        xref

PERLFILES= \
        check_rtime \
        find_elf \
        interface_check \
        interface_cmp \
        jstyle \
        validate_flg \
        validate_paths \
        wdiff

PERLMODULES= \
```

```
        onbld_elfmod.pm \
        onbld_elfmod_vertype.pm


PYFILES= \
        cddlchk \
        copyrightchk \
        git-pbchk \
        hdrchk \
        hg-active \
        mapfilechk \
        validate_pkg \
        wsdiff

SCRIPTLINKS= \
        git-nits

MAN1ONBLDFILES= \
        Install.1onbld \
        bldenv.1onbld \
        bringovercheck.1onbld \
        cddlchk.1onbld \
        checkpaths.1onbld \
        check_rtime.1onbld \
        cstyle.1onbld \
        find_elf.1onbld \
        flg.flp.1onbld \
        git-pbchk.1onbld \
        hdrchk.1onbld \
        interface_check.1onbld \
        interface_cmp.1onbld \
        hgsetup.1onbld \
        jstyle.1onbld \
        mapfilechk.1onbld \
        nightly.1onbld \
        onu.1onbld \
        sccscheck.1onbld \
        webrev.1onbld \
        which_scm.1onbld \
        ws.1onbld \
        wsdiff.1onbld \
        xref.1onbld

MAN1ONBLDLINKS= \
        git-nits.1onbld

MAKEFILES= \
        xref.mk

ETCFILES= \
        hgstyle \
        its.conf \
        its.reg

EXCEPTFILES= \
        check_rtime \
        interface_check \
        interface_cmp

CLEANFILES = $(SHFILES) $(PERLFILES) $(PYFILES) bldenv.1onbld onu.sh

onu.sh: onu.sh.in
        $(SED) -e "s:@PYTHON_VERSION@:$(PYTHON_VERSION):g" < onu.sh.in > $@

include ../Makefile.tools
```

```
ROOTONBLDSCRIPTLINKS = $(SCRIPTLINKS:%=$(ROOTONBLDBIN)/%)
ROOTONBLDMAN1ONBLDLINKS = $(MAN1ONBLDLINKS:%=$(ROOTONBLDMAN1ONBLD)/%)

$(ROOTONBLDETCFILES)            := FILEMODE=    644
$(ROOTONBLDEXCEPTFILES)         := FILEMODE=    644
$(ROOTONBLDPERLMODULES)         := FILEMODE=    644
$(ROOTONBLDMAKEFILES)           := FILEMODE=    644
$(ROOTONBLDMAN1ONBLDFILES)      := FILEMODE=    644

.KEEP_STATE:

all:    $(SHFILES) $(PERLFILES) $(PERLMODULES) $(PYFILES) \
        $(MAN1ONBLDFILES) $(MAKEFILES)

$(ROOTONBLDBIN)/git-nits:
        $(RM) $(ROOTONBLDBIN)/git-nits
        $(SYMLINK) git-pbchk $(ROOTONBLDBIN)/git-nits

$(ROOTONBLDMAN1ONBLD)/git-nits.1onbld:
        $(RM) $(ROOTONBLDMAN1ONBLD)/git-nits.1onbld
        $(SYMLINK) git-pbchk.1onbld $(ROOTONBLDMAN1ONBLD)/git-nits.1onbld

install: all .WAIT $(ROOTONBLDSHFILES) $(ROOTONBLDPERLFILES)            \
                $(ROOTONBLDPERLMODULES) $(ROOTONBLDPYFILES)             \
                $(ROOTONBLDSCRIPTLINKS) $(ROOTONBLDMAN1ONBLDFILES)      \
                $(ROOTONBLDMAKEFILES) $(ROOTONBLDETCFILES)              \
                $(ROOTONBLDEXCEPTFILES) $(ROOTONBLDMAN1ONBLDLINKS)

clean:
        $(RM) $(CLEANFILES)

bldenv: bldenv.sh stdenv.sh
        $(RM) "$@"
        sed -e '/# STDENV_START/ r stdenv.sh' bldenv.sh > "$@"
        # Check for shell lint and fail if we hit warnings
        shlintout="$$( /usr/bin/ksh93 -n "$@" 2>&1 )" ; \
                [[ "$${shlintout}" != "" ]] && \
                { print -r -- "$${shlintout}" ; false ; } || true
        $(CHMOD) +x "$@"

bldenv.1onbld: bldenv
        $(RM) "$@"
        (set +o errexit ; ksh93 $? --nroff ; true) 2>&1 | \
        sed -e 's/\.DS/.nf/g;s/\.DE/.fi/' \
        -e 's/\.TH BLDENV 1/.TH BLDENV 1ONBLD/' \
        -e 's/(1)/(1ONBLD)/' > "$@"

nightly: nightly.sh stdenv.sh
        $(RM) "$@"
        sed -e '/# STDENV_START/ r stdenv.sh' nightly.sh > nightly
        $(CHMOD) +x "$@"

include ../Makefile.targ
```

```
**********************************************************
   15791 Fri Feb  2 23:18:40 2018
new/usr/src/tools/scripts/webrev.1onbld
9001 cdm is useless, remove it
9002 webrev should know how to get the git user name
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
.\"
.\" 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 1ONBLD "Mar 27, 2016"
.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

.SH DESCRIPTION
.B webrev
builds a set of HTML files suitable for performing code review of
source changes in a web browser.
It supports Git and Subversion repositories.
It supports Mercurial, Git and Subversion 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
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 (1ONBLD)
or
.BR bldenv (1ONBLD)
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 (1ONBLD)
to determine the SCM in use for a given workspace.

.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 " revision
Generate webrev for single commit specified by \fIrevision\fR (git only).
.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 "-h " head-revision
Specify the explicit head to generate webrev from (git only).
.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 "-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
\fIusr/closed\fR are automatically elided from the review.
.TP 10
.BI "-o " output-dir
Place output from running the script in the directory specified.  If
specified, this option takes precedence over the WDIR environment variable.
.TP 10
.BI "-p " basis-of-comparison
Specify a basis of comparison meaningful for the SCM currently in use.
See SCM INTERACTIONS and INCREMENTAL REVIEWS.
.TP 10
.BI "-t " target
Upload target. Specified in form of URI identifier. For SCP/SFTP it is
\fIssh://user@remote_host:remote_dir\fR and for rsync it is
\fIrsync://user@remote_host:remote_dir\fR. This option can override the
-o option if the URI is fully specified. The target is relative to
the top level directory of the default sftp/rsync directory tree.
.TP 10
.BI "-U"
Upload the webrev. Default remote host is \fIcr.opensolaris.org\fR.
Default transport is rsync. If it fails, fallback to SCP/SFTP transport
is done.
.TP 10
.BI "-w " wx-file
Extract the file list from the wx "active" file specified.  'wx' uses
this mode when invoking webrev.  The list is assumed to be in the
format expected by the \fIwx\fR package.  See FILE LIST, below.

.SH FILE LIST
.PP
.B Webrev
```

```
needs to be told or to discover which files have changed in a
given workspace.  By default,
.B webrev
will attempt to autodetect the
list of changed files by first consulting
.BR wx "(1)."
If this information is not available, webrev tries to consult the SCM (Source
Code Manager) currently in use.  If that fails, the user must intervene by
specifying either a file list or additional options specific to the SCM in use.

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

If CODEMGR_WS is not set, it may be specified as an environment variable
within the file list, e.g.
.IP
.nf
\f(CWCODEMGR_WS=/home/brent/myws
usr/src/uts/common/fs/nfs/nfs_subr.c
usr/src/uts/common/fs/nfs/nfs_export.c
usr/src/cmd/fs.d/nfs/mountd/mountd.c
.fi
.PP
To compare the workspace against one other than the parent (see also
the -p option), include a CODEMGR_PARENT line in the file list, like:
.IP
.nf
\f(CWCODEMGR_WS=/home/brent/myws
CODEMGR_PARENT=/ws/onnv-gate
usr/src/uts/common/fs/nfs/nfs_subr.c
usr/src/uts/common/fs/nfs/nfs_export.c
usr/src/cmd/fs.d/nfs/mountd/mountd.c
.fi
.PP
Finally, run webrev with the name of the file containing the file list as an
argument, e.g.
.nf
        $ webrev file.list
.fi
.PP
If "-" is supplied as the name of the file, then stdin will be used.

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

1206578 Fix spelling error in comment
```

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

4039272 cstyle fixes

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

1927634 mountd daemon doesn’t handle expletives
.fi

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

.IP
.nf
\f(CW$ webrev -o ~/public_html/myreview-rd2/ \\
         -p ~/public_html/myreview/
.fi

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

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

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

\fBWDIFF\fR specifies the command used to generate Wdiffs. Wdiff generates a
full unified context listing with line numbers where unchanged
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 hg "(1),"
.BR git "(1),"
.BR ssh_config "(4),"
.BR svn "(1),"
.BR which_scm "(1ONBLD)"

.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
.nf
                Brent Callaghan  11/28/96
.fi
```

```
**********************************************************
   83609 Fri Feb  2 23:18:42 2018
new/usr/src/tools/scripts/webrev.sh
9001 cdm is useless, remove it
9002 webrev should know how to get the git user name
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
______unchanged_portion_omitted_


#
# fix_postscript
#

#
# 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 '
        $1 == "'$p'" {
                do getline ; while (NF > 0)
                getline
                while (NF > 0) { print ; getline }
                exit
        }' < $wxfile`

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

        if [[ $fmt == "text" ]]; then
                print -- "$comm"
                return
        fi

        print -- "$comm" | html_quote | its2url

}

#
# getcomments {text|html} filepath parentpath
#
# Fetch the comments depending on what SCM mode we're in.
#
getcomments()
{
        typeset fmt=$1
        typeset p=$2
        typeset pp=$3

        if [[ -n $Nflag ]]; then
                return
        fi

```

```
        #
        # Mercurial support uses a file list in wx format, so this
        # will be used there, too
        #
        if [[ -n $wxfile ]]; then
                comments_from_wx $fmt $p
        fi
}
______unchanged_portion_omitted_

#
# Call hg-active to get the active list output in the wx active list format
#
function hg_active_wxfile
{
        typeset child=$1
        typeset parent=$2

        TMPFLIST=/tmp/$$.active
        $HG_ACTIVE -w $child -p $parent -o $TMPFLIST
        wxfile=$TMPFLIST
}

#
# flist_from_mercurial
# Call hg-active to get a wx-style active list, and hand it off to
# flist_from_wx
#
function flist_from_mercurial
{
        typeset child=$1
        typeset parent=$2

        print " File list from: hg-active -p $parent ...\c"
        if [[ ! -x $HG_ACTIVE ]]; then
                print           # Blank line for the \c above
                print -u2 "Error: hg-active tool not found.  Exiting"
                exit 1
        fi
        hg_active_wxfile $child $parent

        # flist_from_wx prints the Done, so we don't have to.
        flist_from_wx $TMPFLIST
}

#
# Transform a specified 'git log' output format into a wx-like active list.
#
function git_wxfile
{
        typeset child="$1"
        typeset parent="$2"

        TMPFLIST=/tmp/$$.active
        $PERL -e 'my (%files, %realfiles, $msg);
        my $parent = $ARGV[0];
        my $child = $ARGV[1];

        open(F, "git diff -M --name-status $parent..$child |");
        while (<F>) {
            chomp;
            if (/^R(\d+)\s+([^ ]+)\s+([^ ]+)/) { # rename
                if ($1 >= 75) {          # Probably worth treating as a rename
                    $realfiles{$3} = $2;
                } else {
                    $realfiles{$3} = $3;
                    $realfiles{$2} = $2;
                }
            } else {
                my $f = (split /\s+/, $_)[1];
                $realfiles{$f} = $f;
            }
        }
        close(F);

        my $state = 1;              # 0|comments, 1|files
        open(F, "git whatchanged --pretty=format:%B $parent..$child |");
        while (<F>) {
            chomp;
            if (/^:[0-9]{6}/) {
                my ($unused, $fname, $fname2) = split(/\t/, $_);
                $fname = $fname2 if defined($fname2);
                next if !defined($realfiles{$fname}); # No real change
                $state = 1;
                chomp $msg;
                $files{$fname} .= $msg;
            } else {
                if ($state == 1) {
                    $state = 0;
                    $msg = /^\n/ ? "" : "\n";
                }
                $msg .= "$_\n" if ($_);
            }
        }
        close(F);

        for (sort keys %files) {
            if ($realfiles{$_} ne $_) {
                print "$_ $realfiles{$_}\n$files{$_}\n\n";
            } else {
                print "$_\n$files{$_}\n\n"
            }
        }’ ${parent} ${child} > $TMPFLIST

        wxfile=$TMPFLIST
}
```

______***unchanged_portion_omitted_***

```
function build_old_new_mercurial
{
        typeset olddir="$1"
        typeset newdir="$2"
        typeset old_mode=
        typeset new_mode=
        typeset file

        #
        # Get old file mode, from the parent revision manifest entry.
        # Mercurial only stores a "file is executable" flag, but the
        # manifest will display an octal mode "644" or "755".
        #
        if [[ "$PDIR" == "." ]]; then
                file="$PF"
        else
                file="$PDIR/$PF"
        fi
        file=‘echo $file | $SED ’s#/#\\\/#g’‘
        # match the exact filename, and return only the permission digits
        old_mode=‘$SED -n -e "/^\\(...\\) . ${file}$/s//\\1/p" \
            < $HG_PARENT_MANIFEST‘

        #
```

```
        # Get new file mode, directly from the filesystem.
        # Normalize the mode to match Mercurial’s behavior.
        #
        new_mode=‘get_file_mode $CWS/$DIR/$F‘
        if [[ -n "$new_mode" ]]; then
                if [[ "$new_mode" = *[1357]* ]]; then
                        new_mode=755
                else
                        new_mode=644
                fi
        fi

        #
        # new version of the file.
        #
        rm -rf $newdir/$DIR/$F
        if [[ -e $CWS/$DIR/$F ]]; then
                cp $CWS/$DIR/$F $newdir/$DIR/$F
                if [[ -n $new_mode ]]; then
                        chmod $new_mode $newdir/$DIR/$F
                else
                        # should never happen
                        print -u2 "ERROR: set mode of $newdir/$DIR/$F"
                fi
        fi

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

                if (( $? != 0 )); then
                        rm -f $olddir/$PDIR/$PF
                else
                        if [[ -n $old_mode ]]; then
                                chmod $old_mode $olddir/$PDIR/$PF
                        else
                                # should never happen
                                print -u2 "ERROR: set mode of $olddir/$PDIR/$PF"
                        fi
                fi
        fi
}

function build_old_new_git
{
        typeset olddir="$1"
        typeset newdir="$2"
        typeset o_mode=
        typeset n_mode=
        typeset o_object=
        typeset n_object=
        typeset OWD=$PWD
        typeset file
        typeset type

        cd $CWS
        #
        # Get old file and its mode from the git object tree
        #
        if [[ "$PDIR" == "." ]]; then
                file="$PF"
        else
                file="$PDIR/$PF"
        fi

        if [[ -n $parent_webrev && -e $PWS/$PDIR/$PF ]]; then
                cp $PWS/$PDIR/$PF $olddir/$PDIR/$PF
        else
                $GIT ls-tree $GIT_PARENT $file | read o_mode type o_object junk
                $GIT cat-file $type $o_object > $olddir/$file 2>/dev/null

                if (( $? != 0 )); then
                        rm -f $olddir/$file
                elif [[ -n $o_mode ]]; then
                        # Strip the first 3 digits, to get a regular octal mode
                        o_mode=${o_mode/???/}
                        chmod $o_mode $olddir/$file
                else
                        # should never happen
                        print -u2 "ERROR: set mode of $olddir/$file"
                fi
        fi

        #
        # new version of the file.
        #
        if [[ "$DIR" == "." ]]; then
                file="$F"
        else
                file="$DIR/$F"
        fi
        rm -rf $newdir/$file

        if [[ -e $CWS/$DIR/$F ]]; then
                cp $CWS/$DIR/$F $newdir/$DIR/$F
                chmod $(get_file_mode $CWS/$DIR/$F) $newdir/$DIR/$F
        fi
        cd $OWD
}
```
______***unchanged_portion_omitted_***

```
function build_old_new
{
        typeset WDIR=$1
        typeset PWS=$2
        typeset PDIR=$3
        typeset PF=$4
        typeset CWS=$5
        typeset DIR=$6
        typeset F=$7

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

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

        if [[ $SCM_MODE == "git" ]]; then
        if [[ $SCM_MODE == "mercurial" ]]; then
                build_old_new_mercurial "$olddir" "$newdir"
        elif [[ $SCM_MODE == "git" ]]; then
                build_old_new_git "$olddir" "$newdir"
```

```
        elif [[ $SCM_MODE == "subversion" ]]; then
                build_old_new_subversion "$olddir" "$newdir"
        elif [[ $SCM_MODE == "unknown" ]]; then
                build_old_new_unknown "$olddir" "$newdir"
        fi

        if [[ ! -f $olddir/$PDIR/$PF && ! -f $newdir/$DIR/$F ]]; then
                print "*** Error: file not in parent or child"
                return 1
        fi
        return 0
}
```
______***unchanged_portion_omitted_***

```
#
#
# Main program starts here
#
#

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

set +o noclobber

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

[[ -z $WDIFF ]] && WDIFF=‘look_for_prog wdiff‘
[[ -z $WX ]] && WX=‘look_for_prog wx‘
[[ -z $HG_ACTIVE ]] && HG_ACTIVE=‘look_for_prog hg-active‘
[[ -z $GIT ]] && GIT=‘look_for_prog git‘
[[ -z $WHICH_SCM ]] && WHICH_SCM=‘look_for_prog which_scm‘
[[ -z $CODEREVIEW ]] && CODEREVIEW=‘look_for_prog codereview‘
[[ -z $PS2PDF ]] && PS2PDF=‘look_for_prog ps2pdf‘
[[ -z $PERL ]] && PERL=‘look_for_prog perl‘
[[ -z $RSYNC ]] && RSYNC=‘look_for_prog rsync‘
[[ -z $SCCS ]] && SCCS=‘look_for_prog sccs‘
[[ -z $AWK ]] && AWK=‘look_for_prog nawk‘
[[ -z $AWK ]] && AWK=‘look_for_prog gawk‘
[[ -z $AWK ]] && AWK=‘look_for_prog awk‘
[[ -z $SCP ]] && SCP=‘look_for_prog scp‘
[[ -z $SED ]] && SED=‘look_for_prog sed‘
[[ -z $SFTP ]] && SFTP=‘look_for_prog sftp‘
[[ -z $SORT ]] && SORT=‘look_for_prog sort‘
[[ -z $MKTEMP ]] && MKTEMP=‘look_for_prog mktemp‘
[[ -z $GREP ]] && GREP=‘look_for_prog grep‘
[[ -z $FIND ]] && FIND=‘look_for_prog find‘
[[ -z $MANDOC ]] && MANDOC=‘look_for_prog mandoc‘
[[ -z $COL ]] && COL=‘look_for_prog col‘

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

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

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

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

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

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

cflag=
Cflag=
Dflag=
flist_mode=
flist_file=
hflag=
iflag=
Iflag=
lflag=
Nflag=
nflag=
Oflag=
oflag=
pflag=
tflag=
uflag=
Uflag=
wflag=
remote_target=

#
# NOTE: when adding/removing options it is necessary to sync the list
#       with usr/src/tools/onbld/hgext/cdm.py
#
while getopts "c:C:Dh:i:I:lnNo:Op:t:Uw" opt
do
        case $opt in
        c)      cflag=1
                codemgr_head=$OPTARG
                codemgr_parent=$OPTARG~1;;

        C)      Cflag=1
                ITSCONF=$OPTARG;;

        D)      Dflag=1;;

        h)      hflag=1
                codemgr_head=$OPTARG;;

        i)      iflag=1
                INCLUDE_FILE=$OPTARG;;

        I)      Iflag=1
                ITSREG=$OPTARG;;

        N)      Nflag=1;;

        n)      nflag=1;;

        O)      Oflag=1;;
```

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

        p)      pflag=1
                codemgr_parent=$OPTARG;;

        t)      tflag=1
                remote_target=$OPTARG;;

        U)      Uflag=1;;

        w)      wflag=1;;

        ?)      usage;;
        esac
done

FLIST=/tmp/$$.flist

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

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

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

#
# For the invocation "webrev -n -U" with no other options, webrev will assume
# that the webrev exists in ${CWS}/webrev, but will upload it using the name
# $(basename ${CWS}).  So we need to get CWS set before we skip any remaining
# logic.
#
$WHICH_SCM | read SCM_MODE junk || exit 1

if [[ $SCM_MODE == "git" ]]; then
if [[ $SCM_MODE == "mercurial" ]]; then
        #
        # Mercurial priorities:
        # 1. hg root from CODEMGR_WS environment variable
        # 1a. hg root from CODEMGR_WS/usr/closed if we're somewhere under
        #    usr/closed when we run webrev
        # 2. hg root from directory of invocation
        #
        if [[ ${PWD} =~ "usr/closed" ]]; then
                testparent=${CODEMGR_WS}/usr/closed
                # If we're in OpenSolaris mode, we enforce a minor policy:
                # help to make sure the reviewer doesn't accidentally publish
                # source which is under usr/closed
                if [[ -n "$Oflag" ]]; then
                        print -u2 "OpenSolaris output not permitted with" \
                            "usr/closed changes"
                        exit 1
                fi
        else
                testparent=${CODEMGR_WS}
        fi
        [[ -z $codemgr_ws && -n $testparent ]] && \
            codemgr_ws=$(hg root -R $testparent 2>/dev/null)
        [[ -z $codemgr_ws ]] && codemgr_ws=$(hg root 2>/dev/null)
        CWS=$codemgr_ws
elif [[ $SCM_MODE == "git" ]]; then
        #
        # Git priorities:
        # 1. git rev-parse --git-dir from CODEMGR_WS environment variable
        # 2. git rev-parse --git-dir from directory of invocation
        #
        [[ -z $codemgr_ws && -n $CODEMGR_WS ]] && \
            codemgr_ws=$($GIT --git-dir=$CODEMGR_WS/.git rev-parse --git-dir \
                2>/dev/null)
        [[ -z $codemgr_ws ]] && \
            codemgr_ws=$($GIT rev-parse --git-dir 2>/dev/null)

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

        if [[ "$codemgr_ws" = *"/.git" ]]; then
                codemgr_ws=$(dirname $codemgr_ws) # Lose the '/.git'
        fi
        CWS="$codemgr_ws"
elif [[ $SCM_MODE == "subversion" ]]; then
        #
        # Subversion priorities:
        # 1. CODEMGR_WS from environment
        # 2. Relative path from current directory to SVN repository root
        #
        if [[ -n $CODEMGR_WS && -d $CODEMGR_WS/.svn ]]; then
                CWS=$CODEMGR_WS
        else
                svn info | while read line; do
                        if [[ $line == "URL: "* ]]; then
                                url=${line#URL: }
                        elif [[ $line == "Repository Root: "* ]]; then
                                repo=${line#Repository Root: }
                        fi
                done

                rel=${url#$repo}
                CWS=${PWD%$rel}
        fi
fi

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

#
# If the command line options indicate no webrev generation, either
# explicitly (-n) or implicitly (-D but not -U), then there's a whole
# ton of logic we can skip.
#
# Instead of increasing indentation, we intentionally leave this loop
# body open here, and exit via break from multiple points within.
# Search for DO_EVERYTHING below to find the break points and closure.
#
for do_everything in 1; do

# DO_EVERYTHING: break point
```

```
if [[ -n $nflag || ( -z $Uflag && -n $Dflag ) ]]; then
        break
fi

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

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

        if [[ $1 == "-" ]]; then
                cat > $FLIST
                flist_mode="stdin"
                flist_done=1
                shift
        elif [[ -n $1 ]]; then
                if [[ ! -r $1 ]]; then
                        print -u2 "$1: no such file or not readable"
                        usage
                fi
                cat $1 > $FLIST
                flist_mode="file"
                flist_file=$1
                flist_done=1
                shift
        else
                flist_mode="auto"
        fi
fi

#
# Before we go on to further consider -l and -w, work out which SCM we think
# is in use.
#
case "$SCM_MODE" in
git|subversion)
mercurial|git|subversion)
        ;;
unknown)
        if [[ $flist_mode == "auto" ]]; then
                print -u2 "Unable to determine SCM in use and file list not spec
                print -u2 "See which_scm(1) for SCM detection information."
                exit 1
        fi
        ;;
*)
        if [[ $flist_mode == "auto" ]]; then
                print -u2 "Unsupported SCM in use ($SCM_MODE) and file list not
                exit 1
        fi
        ;;
esac

print -u2 "   SCM detected: $SCM_MODE"

if [[ -n $wflag ]]; then
        #
        # If the -w is given then assume the file list is in Bonwick's "wx"
        # command format, i.e.  pathname lines alternating with SCCS comment
        # lines with blank lines as separators.  Use the SCCS comments later
        # in building the index.html file.
        #
        shift $(($OPTIND - 1))
        wxfile=$1
        if [[ -z $wxfile && -n $CODEMGR_WS ]]; then
                if [[ -r $CODEMGR_WS/wx/active ]]; then
                        wxfile=$CODEMGR_WS/wx/active
                fi
        fi

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

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

        print -u2 " File list from: wx ’active’ file ’$wxfile’ ... \c"
        flist_from_wx $wxfile
        flist_done=1
        if [[ -n "$*" ]]; then
                shift
        fi
elif [[ $flist_mode == "stdin" ]]; then
        print -u2 " File list from: standard input"
elif [[ $flist_mode == "file" ]]; then
        print -u2 " File list from: $flist_file"
fi

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

#
# Before we entered the DO_EVERYTHING loop, we should have already set CWS
# and CODEMGR_WS as needed.  Here, we set the parent workspace.
#
if [[ $SCM_MODE == "mercurial" ]]; then
        #
        # Parent can either be specified with -p
        # Specified with CODEMGR_PARENT in the environment
        # or taken from hg’s default path.
        #

if [[ $SCM_MODE == "git" ]]; then
        if [[ -z $codemgr_parent && -n $CODEMGR_PARENT ]]; then
                codemgr_parent=$CODEMGR_PARENT
        fi

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

        PWS=$codemgr_parent

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

```
        #
        # If hg-active exists, then we run it.  In the case of no explicit
        # flist given, we’ll use it for our comments.  In the case of an
        # explicit flist given we’ll try to use it for comments for any
        # files mentioned in the flist.
        #
        if [[ -z $flist_done ]]; then
                flist_from_mercurial $CWS $real_parent
                flist_done=1
        fi

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

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

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

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

        pnode=$(trim_digest $HG_PARENT)
        PRETTY_PWS="${PWS} (at ${pnode})"
        cnode=$(hg parent -R $codemgr_ws --template ’{node|short}’ \
            2>/dev/null)
        PRETTY_CWS="${CWS} (at ${cnode})"}
elif [[ $SCM_MODE == "git" ]]; then
        # Check that "head" revision specified with -c or -h is sane
        if [[ -n $cflag || -n $hflag ]]; then
                head_rev=$($GIT rev-parse --verify --quiet "$codemgr_head")
                if [[ -z $head_rev ]]; then
                        print -u2 "Error: bad revision ${codemgr_head}"
                        exit 1
                fi
        fi
```

```
	if [[ -z $codemgr_head ]]; then
		codemgr_head="HEAD";
	fi

	# Parent can either be specified with -p, or specified with
	# CODEMGR_PARENT in the environment.
	if [[ -z $codemgr_parent && -n $CODEMGR_PARENT ]]; then
		codemgr_parent=$CODEMGR_PARENT
	fi

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

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

	if [[ -z $codemgr_parent ]]; then
		codemgr_parent=$par_branch
	fi
	PWS=$codemgr_parent

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

	if [[ -z $flist_done ]]; then
		flist_from_git "$codemgr_head" "$real_parent"
		flist_done=1
	fi

	#
	# If we have a file list now, pull out any variables set
	# therein.
	#
	if [[ -n $flist_done ]]; then
		env_from_flist
	fi

	#
	# If we don't have a wx-format file list, build one we can pull change
	# comments from.
	#
	if [[ -z $wxfile ]]; then
		print "  Comments from: git...\c"
		git_wxfile "$codemgr_head" "$real_parent"
		print " Done."
	fi
```

```
	if [[ -z $GIT_PARENT ]]; then
		GIT_PARENT=$($GIT merge-base "$real_parent" "$codemgr_head")
	fi
	if [[ -z $GIT_PARENT ]]; then
		print -u2 "Error: Cannot discover parent revision"
		exit 1
	fi

	pnode=$(trim_digest $GIT_PARENT)

	if [[ -n $cflag ]]; then
		PRETTY_PWS="previous revision (at ${pnode})"
	elif [[ $real_parent == */* ]]; then
		origin=$(echo $real_parent | cut -d/ -f1)
		origin=$($GIT remote -v | \
		    $AWK '$1 == "'$origin'" { print $2; exit }')
		PRETTY_PWS="${PWS} (${origin} at ${pnode})"
	elif [[ -n $pflag && -z $parent_webrev ]]; then
		PRETTY_PWS="${CWS} (explicit revision ${pnode})"
	else
		PRETTY_PWS="${PWS} (at ${pnode})"
	fi

	cnode=$($GIT --git-dir=${codemgr_ws}/.git rev-parse --short=12 \
	    ${codemgr_head} 2>/dev/null)

	if [[ -n $cflag || -n $hflag ]]; then
		PRETTY_CWS="${CWS} (explicit head at ${cnode})"
	else
		PRETTY_CWS="${CWS} (at ${cnode})"
	fi
elif [[ $SCM_MODE == "subversion" ]]; then

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

	if [[ -z $flist_done && $flist_mode == "auto" ]]; then
		flist_from_subversion $CWS $OLDPWD
	fi
else
	if [[ $SCM_MODE == "unknown" ]]; then
		print -u2 "    Unknown type of SCM in use"
	else
		print -u2 "    Unsupported SCM in use: $SCM_MODE"
	fi

	env_from_flist

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

	if [[ -z $CODEMGR_PARENT ]]; then
		print -u2 "SCM not detected/supported and " \
		    "CODEMGR_PARENT not specified"
		exit 1
	fi

	CWS=$CODEMGR_WS
	PWS=$CODEMGR_PARENT
fi
```

```
#
# If the user didn’t specify a -i option, check to see if there is a
# webrev-info file in the workspace directory.
#
if [[ -z $iflag && -r "$CWS/webrev-info" ]]; then
        iflag=1
        INCLUDE_FILE="$CWS/webrev-info"
fi

if [[ -n $iflag ]]; then
        if [[ ! -r $INCLUDE_FILE ]]; then
                print -u2 "include file ’$INCLUDE_FILE’ does not exist or is" \
                    "not readable."
                exit 1
        else
                #
                # $INCLUDE_FILE may be a relative path, and the script alters
                # PWD, so we just stash a copy in /tmp.
                #
                cp $INCLUDE_FILE /tmp/$$.include
        fi
fi

# DO_EVERYTHING: break point
if [[ -n $Nflag ]]; then
        break
fi

typeset -A itsinfo
typeset -r its_sed_script=/tmp/$$.its_sed
valid_prefixes=
if [[ -z $nflag ]]; then
        DEFREGFILE="$(/bin/dirname "$(whence $0)")/../etc/its.reg"
        if [[ -n $Iflag ]]; then
                REGFILE=$ITSREG
        elif [[ -r $HOME/.its.reg ]]; then
                REGFILE=$HOME/.its.reg
        else
                REGFILE=$DEFREGFILE
        fi
        if [[ ! -r $REGFILE ]]; then
                print "ERROR: Unable to read database registry file $REGFILE"
                exit 1
        elif [[ $REGFILE != $DEFREGFILE ]]; then
                print "   its.reg from: $REGFILE"
        fi

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

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

                if [[ $name == PREFIX ]]; then
                        p=${value}
                        valid_prefixes="${p} ${valid_prefixes}"
                else
                        itsinfo["${p}_${name}"]="${value}"
                fi
        done


        DEFCONFFILE="$(/bin/dirname "$(whence $0)")/../etc/its.conf"
        CONFFILES=$DEFCONFFILE
        if [[ -r $HOME/.its.conf ]]; then
                CONFFILES="${CONFFILES} $HOME/.its.conf"
```

```
        fi
        if [[ -n $Cflag ]]; then
                CONFFILES="${CONFFILES} ${ITSCONF}"
        fi
        its_domain=
        its_priority=
        for cf in ${CONFFILES}; do
                if [[ ! -r $cf ]]; then
                        print "ERROR: Unable to read database configuration file
                        exit 1
                elif [[ $cf != $DEFCONFFILE ]]; then
                        print "       its.conf: reading $cf"
                fi
                $SED -e ’/^#/d’ -e ’/^[         ]*$/d’ $cf | while read LINE; do
                    eval "${LINE}"
                done
        done

        #
        # If an information tracking system is explicitly identified by prefix,
        # we want to disregard the specified priorities and resolve it according
        #
        # To that end, we’ll build a sed script to do each valid prefix in turn.
        #
        for p in ${valid_prefixes}; do
                #
                # When an informational URL was provided, translate it to a
                # hyperlink.  When omitted, simply use the prefix text.
                #
                if [[ -z ${itsinfo["${p}_INFO"]} ]]; then
                        itsinfo["${p}_INFO"]=${p}
                else
                        itsinfo["${p}_INFO"]="<a href=\\\"${itsinfo["${p}_INFO"]
                fi

                #
                # Assume that, for this invocation of webrev, all references
                # to this information tracking system should resolve through
                # the same URL.
                #
                # If the caller specified -O, then always use EXTERNAL_URL.
                #
                # Otherwise, look in the list of domains for a matching
                # INTERNAL_URL.
                #
                [[ -z $Oflag ]] && for d in ${its_domain}; do
                        if [[ -n ${itsinfo["${p}_INTERNAL_URL_${d}"]} ]]; then
                                itsinfo["${p}_URL"]="${itsinfo[${p}_INTERNAL_URL
                                break
                        fi
                done
                if [[ -z ${itsinfo["${p}_URL"]} ]]; then
                        itsinfo["${p}_URL"]="${itsinfo[${p}_EXTERNAL_URL]}"
                fi

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

                # The character class below contains a literal tab
                print "/^${p}[:         ]/ {
                                s;${itsinfo[${p}_REGEX]};${itsinfo[${p}_URL]};g
                                s;^${p};${itsinfo[${p}_INFO]};
                        }" >> ${its_sed_script}
        done
        #
        # The previous loop took care of explicit specification.  Now use
        # the configured priorities to attempt implicit translations.
        #
        for p in ${its_priority}; do
                print "/^${itsinfo[${p}_REGEX]}[        ]/ {
                                s;^${itsinfo[${p}_REGEX]};${itsinfo[${p}_URL]};g
                        }" >> ${its_sed_script}
        done
fi

#
# Search for DO_EVERYTHING above for matching "for" statement
# and explanation of this terminator.
#
done

#
# Output directory.
#
WDIR=${WDIR:-$CWS/webrev}

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

# Make sure remote target is well formed for remote upload/delete.
if [[ -n $Dflag || -n $Uflag ]]; then
        #
        # If remote target is not specified, build it from scratch using
        # the default values.
        #
        if [[ -z $tflag ]]; then
                remote_target=${DEFAULT_REMOTE_HOST}:${WNAME}
        else
                #
                # Check upload target prefix first.
                #
                if [[ "${remote_target}" != ${rsync_prefix}* &&
                    "${remote_target}" != ${ssh_prefix}* ]]; then
                        print "ERROR: invalid prefix of upload URI" \
                            "($remote_target)"
                        exit 1
                fi
                #
                # If destination specification is not in the form of
                # host_spec:remote_dir then assume it is just remote hostname
                # and append a colon and destination directory formed from
                # local webrev directory name.
                #
                typeset target_no_prefix=${remote_target##*://}
                if [[ ${target_no_prefix} == *:* ]]; then
                        if [[ "${remote_target}" == *: ]]; then
                                remote_target=${remote_target}${WNAME}
                        fi
                else
                        if [[ ${target_no_prefix} == */* ]]; then
                                print "ERROR: badly formed upload URI" \
                                        "($remote_target)"
```

```
                                exit 1
                        else
                                remote_target=${remote_target}:${WNAME}
                        fi
                fi
        fi

        #
        # Strip trailing slash. Each upload method will deal with directory
        # specification separately.
        #
        remote_target=${remote_target%/}
fi

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

#
# Do not generate the webrev, just upload it or delete it.
#
if [[ -n $nflag ]]; then
        if [[ -n $Dflag ]]; then
                delete_webrev 1 1
                (( $? == 0 )) || exit $?
        fi
        if [[ -n $Uflag ]]; then
                upload_webrev
                exit $?
        fi
fi

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

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

#
# Summarize what we’re going to do.
#
print "      Workspace: ${PRETTY_CWS:-$CWS}"
if [[ -n $parent_webrev ]]; then
        print "Compare against: webrev at $parent_webrev"
else
        print "Compare against: ${PRETTY_PWS:-$PWS}"
fi

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

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

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

```
touch $WDIR/$WNAME.patch

print "   Output Files:"

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

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

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

#
# First pass through the files: generate the per-file webrev HTML-files.
#
cat $FLIST | while read LINE
do
        set - $LINE
        P=$1

        #
        # Normally, each line in the file list is just a pathname of a
        # file that has been modified or created in the child.  A file
        # that is renamed in the child workspace has two names on the
        # line: new name followed by the old name.
        #
        oldname=""
        oldpath=""
        rename=
        if [[ $# -eq 2 ]]; then
                PP=$2                   # old filename
                if [[ -f $PP ]]; then
```

```
                        oldname=" (copied from $PP)"
                else
                        oldname=" (renamed from $PP)"
                fi
                oldpath="$PP"
                rename=1
                PDIR=${PP%/*}
                if [[ $PDIR == $PP ]]; then
                        PDIR="."   # File at root of workspace
                fi

                PF=${PP##*/}

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

                F=${P##*/}

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

                F=${P##*/}

                PP=$P
                PDIR=$DIR
                PF=$F
        fi

        COMM=‘getcomments html $P $PP‘

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

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

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

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

        #
        # Keep the old PWD around, so we can safely switch back after
        # diff generation, such that build_old_new runs in a
        # consistent environment.
        #
        OWD=$PWD
        cd $WDIR/raw_files

        #
        # The "git apply" command does not tolerate the spurious
        # "./" that we otherwise insert; be careful not to include
        # it in the paths that we pass to diff(1).
        #
        if [[ $PDIR == "." ]]; then
                ofile=old/$PF
        else
                ofile=old/$PDIR/$PF
```

```
	fi
	if [[ $DIR == "." ]]; then
		nfile=new/$F
	else
		nfile=new/$DIR/$F
	fi

	mv_but_nodiff=
	cmp $ofile $nfile > /dev/null 2>&1
	if [[ $? == 0 && $rename == 1 ]]; then
		mv_but_nodiff=1
	fi

	#
	# If we have old and new versions of the file then run the appropriate
	# diffs.  This is complicated by a couple of factors:
	#
	#	- renames must be handled specially: we emit a 'remove'
	#	  diff and an 'add' diff
	#	- new files and deleted files must be handled specially
	#	- GNU patch doesn't interpret the output of illumos diff
	#	  properly when it comes to adds and deletes.  We need to
	#	  do some "cleansing" transformations:
	#	    [to add a file] @@ -1,0 +X,Y @@  -->  @@ -0,0 +X,Y @@
	#	    [to del a file] @@ -X,Y +1,0 @@  -->  @@ -X,Y +0,0 @@
	#
	cleanse_rmfile="$SED 's/^\(@@ [0-9+,-]*\) [0-9+,-]* @@$/\1 +0,0 @@/'"
	cleanse_newfile="$SED 's/^@@ [0-9+,-]* \([0-9+,-]* @@\)$/@@ -0,0 \1/'"

	rm -f $WDIR/$DIR/$F.patch
	if [[ -z $rename ]]; then
		if [ ! -f "$ofile" ]; then
			diff -u /dev/null $nfile | sh -c "$cleanse_newfile" \
			    > $WDIR/$DIR/$F.patch
		elif [ ! -f "$nfile" ]; then
			diff -u $ofile /dev/null | sh -c "$cleanse_rmfile" \
			    > $WDIR/$DIR/$F.patch
		else
			diff -u $ofile $nfile > $WDIR/$DIR/$F.patch
		fi
	else
		diff -u $ofile /dev/null | sh -c "$cleanse_rmfile" \
		    > $WDIR/$DIR/$F.patch

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

	#
	# Tack the patch we just made onto the accumulated patch for the
	# whole wad.
	#
	cat $WDIR/$DIR/$F.patch >> $WDIR/$WNAME.patch
	print " patch\c"

	if [[ -f $ofile && -f $nfile && -z $mv_but_nodiff ]]; then
		${CDIFFCMD:-diff -bt -C 5} $ofile $nfile > $WDIR/$DIR/$F.cdiff
		diff_to_html $F $DIR/$F "C" "$COMM" < $WDIR/$DIR/$F.cdiff \
		    > $WDIR/$DIR/$F.cdiff.html
		print " cdiffs\c"

		${UDIFFCMD:-diff -bt -U 5} $ofile $nfile > $WDIR/$DIR/$F.udiff
		diff_to_html $F $DIR/$F "U" "$COMM" < $WDIR/$DIR/$F.udiff \
		    > $WDIR/$DIR/$F.udiff.html
		print " udiffs\c"
```

```
		if [[ -x $WDIFF ]]; then
			$WDIFF -c "$COMM" \
			    -t "$WNAME Wdiff $DIR/$F" $ofile $nfile > \
			    $WDIR/$DIR/$F.wdiff.html 2>/dev/null
			if [[ $? -eq 0 ]]; then
				print " wdiffs\c"
			else
				print " wdiffs[fail]\c"
			fi
		fi

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

		rm -f $WDIR/$DIR/$F.cdiff $WDIR/$DIR/$F.udiff
		difflines $ofile $nfile > $WDIR/$DIR/$F.count
	elif [[ -f $ofile && -f $nfile && -n $mv_but_nodiff ]]; then
		# renamed file: may also have differences
		difflines $ofile $nfile > $WDIR/$DIR/$F.count
	elif [[ -f $nfile ]]; then
		# new file: count added lines
		difflines /dev/null $nfile > $WDIR/$DIR/$F.count
	elif [[ -f $ofile ]]; then
		# old file: count deleted lines
		difflines $ofile /dev/null > $WDIR/$DIR/$F.count
	fi

	#
	# Check if it's man page, and create plain text, html and raw (ascii)
	# output for the new version, as well as diffs against old version.
	#
	if [[ -f "$nfile" && "$nfile" = *.+([0-9])*([a-zA-Z]) && \
	    -x $MANDOC && -x $COL ]]; then
		$MANDOC -Tascii $nfile | $COL -b > $nfile.man.txt
		source_to_html txt < $nfile.man.txt > $nfile.man.txt.html
		print " man-txt\c"
		print "$MANCSS" > $WDIR/raw_files/new/$DIR/man.css
		$MANDOC -Thtml -Ostyle=man.css $nfile > $nfile.man.html
		print " man-html\c"
		$MANDOC -Tascii $nfile > $nfile.man.raw
		print " man-raw\c"
		if [[ -f "$ofile" && -z $mv_but_nodiff ]]; then
			$MANDOC -Tascii $ofile | $COL -b > $ofile.man.txt
			${CDIFFCMD:-diff -bt -C 5} $ofile.man.txt \
			    $nfile.man.txt > $WDIR/$DIR/$F.man.cdiff
			diff_to_html $F $DIR/$F "C" "$COMM" < \
			    $WDIR/$DIR/$F.man.cdiff > \
			    $WDIR/$DIR/$F.man.cdiff.html
			print " man-cdiffs\c"
			${UDIFFCMD:-diff -bt -U 5} $ofile.man.txt \
			    $nfile.man.txt > $WDIR/$DIR/$F.man.udiff
			diff_to_html $F $DIR/$F "U" "$COMM" < \
			    $WDIR/$DIR/$F.man.udiff > \
			    $WDIR/$DIR/$F.man.udiff.html
			print " man-udiffs\c"
			if [[ -x $WDIFF ]]; then
				$WDIFF -c "$COMM" -t "$WNAME Wdiff $DIR/$F" \
				    $ofile.man.txt $nfile.man.txt > \
				    $WDIR/$DIR/$F.man.wdiff.html 2>/dev/null
				if [[ $? -eq 0 ]]; then
					print " man-wdiffs\c"
				else
					print " man-wdiffs[fail]\c"
				fi
```

```
                        fi
                        sdiff_to_html $ofile.man.txt $nfile.man.txt $F.man $DIR
                            "$COMM" > $WDIR/$DIR/$F.man.sdiff.html
                        print " man-sdiffs\c"
                        print " man-frames\c"
                fi
                rm -f $ofile.man.txt $nfile.man.txt
                rm -f $WDIR/$DIR/$F.man.cdiff $WDIR/$DIR/$F.man.udiff
        fi

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

                if [[ -z $mv_but_nodiff ]]; then
                        textcomm=‘getcomments text $P $PP‘
                        if [[ -x $CODEREVIEW ]]; then
                                $CODEREVIEW -y "$textcomm" \
                                    -e $ocr $nfile \
                                    > /tmp/$$.psfile 2>/dev/null &&
                                    cat /tmp/$$.psfile >> $WDIR/$WNAME.ps
                                if [[ $? -eq 0 ]]; then
                                        print " ps\c"
                                else
                                        print " ps[fail]\c"
                                fi
                        fi
                fi
        fi

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

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

        cd $OWD

        print
done

frame_nav_js > $WDIR/ancnav.js
frame_navigation > $WDIR/ancnav.html

if [[ ! -f $WDIR/$WNAME.ps ]]; then
        print " Generating PDF: Skipped: no output available"
elif [[ -x $CODEREVIEW && -x $PS2PDF ]]; then
        print " Generating PDF: \c"
        fix_postscript $WDIR/$WNAME.ps | $PS2PDF - > $WDIR/$WNAME.pdf
        print "Done."
else
        print " Generating PDF: Skipped: missing ’ps2pdf’ or ’codereview’"
fi

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

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

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

cd $CWS

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

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

print "$HTML<head>$STDHEAD"
print "<title>$WNAME</title>"
print "</head>"
print "<body id=\"SUNWwebrev\">"
print "<div class=\"summary\">"
print "<h2>Code Review for $WNAME</h2>"

print "<table>"

#
# Get the preparer’s name:
#
# If the SCM detected is Git, and the configuration property user.name is
# available, use that, but be careful to properly escape angle brackets (HTML
# syntax characters) in the email address.
# If the SCM detected is Mercurial, and the configuration property
# ui.username is available, use that, but be careful to properly escape
# angle brackets (HTML syntax characters) in the email address.
#
# Otherwise, use the current userid in the form "John Doe (jdoe)", but
# to maintain compatibility with passwd(4), we must support ’&’ substitutions.
#
preparer=
if [[ "$SCM_MODE" == git ]]; then
        preparer=$(git config user.name 2>/dev/null)
if [[ "$SCM_MODE" == mercurial ]]; then
        preparer=‘hg showconfig ui.username 2>/dev/null‘
        if [[ -n "$preparer" ]]; then
                preparer="$(echo "$preparer" | html_quote)"
        fi
fi
if [[ -z "$preparer" ]]; then
        preparer=$(
            $PERL -e ’
                ($login, $pw, $uid, $gid, $quota, $cmt, $gcos) = getpwuid($<);
                if ($login) {
                    $gcos =~ s/\&/ucfirst($login)/e;
                    printf "%s (%s)\n", $gcos, $login;
                } else {
                    printf "(unknown)\n";
                }
        ’)
fi

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

if [[ -f $WDIR/$WNAME.patch ]]; then
        wpatch_url="$(print $WNAME.patch | url_encode)"
        print "<tr><th>Patch of changes:</th><td>"
        print "<a href=\"$wpatch_url\">$WNAME.patch</a></td></tr>"
fi
if [[ -f $WDIR/$WNAME.pdf ]]; then
        wpdf_url="$(print $WNAME.pdf | url_encode)"
        print "<tr><th>Printable review:</th><td>"
        print "<a href=\"$wpdf_url\">$WNAME.pdf</a></td></tr>"
fi

if [[ -n "$iflag" ]]; then
        print "<tr><th>Author comments:</th><td><div>"
        cat /tmp/$$.include
        print "</div></td></tr>"
fi
print "</table>"
print "</div>"

#
# Second pass through the files: generate the rest of the index file
#
cat $FLIST | while read LINE
do
        set - $LINE
        P=$1

        if [[ $# == 2 ]]; then
                PP=$2
                oldname="$PP"
        else
                PP=$P
                oldname=""
        fi

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

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

        # Avoid processing the same file twice.
        # It’s possible for renamed files to
        # appear twice in the file list

        F=$WDIR/$P

        print "<p>"

        # If there’s a diffs file, make diffs links
```

```

        if [[ -f $F.cdiff.html ]]; then
                cdiff_url="$(print $P.cdiff.html | url_encode)"
                udiff_url="$(print $P.udiff.html | url_encode)"
                sdiff_url="$(print $P.sdiff.html | url_encode)"
                frames_url="$(print $P.frames.html | url_encode)"
                print "<a href=\"$cdiff_url\">Cdiffs</a>"
                print "<a href=\"$udiff_url\">Udiffs</a>"
                if [[ -f $F.wdiff.html && -x $WDIFF ]]; then
                        wdiff_url="$(print $P.wdiff.html | url_encode)"
                        print "<a href=\"$wdiff_url\">Wdiffs</a>"
                fi
                print "<a href=\"$sdiff_url\">Sdiffs</a>"
                print "<a href=\"$frames_url\">Frames</a>"
        else
                print " ------ ------"
                if [[ -x $WDIFF ]]; then
                        print " ------"
                fi
                print " ------ ------"
        fi

        # If there’s an old file, make the link

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

        # If there’s an new file, make the link

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

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

        if [[ -f $WDIR/raw_files/new/$P ]]; then
                rawfiles_url="$(print raw_files/new/$P | url_encode)"
                print "<a href=\"$rawfiles_url\">Raw</a>"
        else
                print " ---"
        fi

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

        # For renamed files, clearly state whether or not they are modified
        if [[ -f "$oldname" ]]; then
                if [[ -n "$mv_but_nodiff" ]]; then
                        print "<i>(copied from $oldname)</i>"
                else
                        print "<i>(copied and modified from $oldname)</i>"
                fi
        elif [[ -n "$oldname" ]]; then
                if [[ -n "$mv_but_nodiff" ]]; then
                        print "<i>(renamed from $oldname)</i>"
                else
                        print "<i>(renamed and modified from $oldname)</i>"
                fi
        fi

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

        # Check for usr/closed and deleted_files/usr/closed
        if [ ! -z "$Oflag" ]; then
                if [[ $P == usr/closed/* || \
                    $P == deleted_files/usr/closed/* ]]; then
                        print "&nbsp;&nbsp;<i>Closed source: omitted from" \
                            "this review</i>"
                fi
        fi

        manpage=
        if [[ -f $F.man.cdiff.html || \
            -f $WDIR/raw_files/new/$P.man.txt.html ]]; then
                manpage=1
                print "<br/>man:"
        fi

        if [[ -f $F.man.cdiff.html ]]; then
                mancdiff_url="$(print $P.man.cdiff.html | url_encode)"
                manudiff_url="$(print $P.man.udiff.html | url_encode)"
                mansdiff_url="$(print $P.man.sdiff.html | url_encode)"
                manframes_url="$(print $P.man.frames.html | url_encode)"
                print "<a href=\"$mancdiff_url\">Cdiffs</a>"
                print "<a href=\"$manudiff_url\">Udiffs</a>"
                if [[ -f $F.man.wdiff.html && -x $WDIFF ]]; then
                        manwdiff_url="$(print $P.man.wdiff.html | url_encode)"
                        print "<a href=\"$manwdiff_url\">Wdiffs</a>"
                fi
                print "<a href=\"$mansdiff_url\">Sdiffs</a>"
                print "<a href=\"$manframes_url\">Frames</a>"
        elif [[ -n $manpage ]]; then
                print " ------ ------"
                if [[ -x $WDIFF ]]; then
                        print " ------"
                fi
                print " ------ ------"
        fi

        if [[ -f $WDIR/raw_files/new/$P.man.txt.html ]]; then
                mantxt_url="$(print raw_files/new/$P.man.txt.html | url_encode)"
                print "<a href=\"$mantxt_url\">TXT</a>"
                manhtml_url="$(print raw_files/new/$P.man.html | url_encode)"
                print "<a href=\"$manhtml_url\">HTML</a>"
                manraw_url="$(print raw_files/new/$P.man.raw | url_encode)"
                print "<a href=\"$manraw_url\">Raw</a>"
        elif [[ -n $manpage ]]; then
                print " --- ---- ---"
        fi

        print "</p>"

        # Insert delta comments
        print "<blockquote><pre>"
        getcomments html $P $PP
        print "</pre>"

        # Add additional comments comment
```

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

        # Add count of changes.
        if [[ -f $F.count ]]; then
            cat $F.count
            rm $F.count
        fi

        if [[ $SCM_MODE == "unknown" ]]; then
        if [[ $SCM_MODE == "mercurial" ||
            $SCM_MODE == "unknown" ]]; then
                # Include warnings for important file mode situations:
                # 1) New executable files
                # 2) Permission changes of any kind
                # 3) Existing executable files
                old_mode=
                if [[ -f $WDIR/raw_files/old/$PP ]]; then
                        old_mode=`get_file_mode $WDIR/raw_files/old/$PP`
                fi

                new_mode=
                if [[ -f $WDIR/raw_files/new/$P ]]; then
                        new_mode=`get_file_mode $WDIR/raw_files/new/$P`
                fi

                if [[ -z "$old_mode" && "$new_mode" = *[1357]* ]]; then
                        print "<span class=\"chmod\">"
                        print "<p>new executable file: mode $new_mode</p>"
                        print "</span>"
                elif [[ -n "$old_mode" && -n "$new_mode" &&
                    "$old_mode" != "$new_mode" ]]; then
                        print "<span class=\"chmod\">"
                        print "<p>mode change: $old_mode to $new_mode</p>"
                        print "</span>"
                elif [[ "$new_mode" = *[1357]* ]]; then
                        print "<span class=\"chmod\">"
                        print "<p>executable file: mode $new_mode</p>"
                        print "</span>"
                fi
        fi

        print "</blockquote>"
done

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

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

print "Done."

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

if [[ -n $Uflag ]]; then
        upload_webrev
        exit $?
fi
```