

new/usr/src/tools/scripts/which\_scm.1

1

```
*****
2347 Sat May 9 16:08:43 2009
new/usr/src/tools/scripts/which_scm.1
Add git support to which_scm utility
*****
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 .\"
21 .\" Copyright 2008 Sun Microsystems, Inc. All rights reserved.
22 .\" Use is subject to license terms.
23 .\"
24 .\" ident      "%Z%M% %I%    %E% SMI"
25 .\"
26 .TH which_scm 1 "11 April 2008"
27 .SH NAME
28 which_scm \- Report Source Code Management system
29 .SH SYNOPSIS
30 .B which_scm
31
32 .SH DESCRIPTION
33 .B which_scm
34 detects the Source Code Management (SCM) system in use, and prints a
35 single line of text containing a one-word name for the SCM, followed
36 by a space, and then the top-level directory path for the workspace.
37
38 The typical usage within a ksh script is:
39 .nf
40     which_scm | read SCM_TYPE CODEMGR_WS || exit 1
41 .fi
42
43 If CODEMGR_WS is set in the environment, then \fBwhich_scm\fR
44 checks only that directory, assuming that it is the top of the tree.
45
46 If CODEMGR_WS is not set, then \fBwhich_scm\fR searches upwards to
47 find the containing workspace path, and reports that path.
48
49 .B which_scm
50 can detect the following types of SCM systems (these are the keywords
51 used in the output format):
52 .nf
53     cvs
54     git
55     mercurial
56     rcs
57     sccs
58     subversion
59     teamware
60 .fi
```

new/usr/src/tools/scripts/which\_scm.1

2

```
62 If the type of SCM in use is not known, then the string "unknown" is
63 printed, and the path returned is $CODEMGR_WS (if set) or the current
64 working directory. The command may exit with an error if the SCM
65 could be detected, but is unusable.
66
67 .SH ENVIRONMENT VARIABLES
68 The following environment variable is used by \fBwhich_scm\fR:
69
70 .PP
71 \fBCODEMGR_WS\fR path to current workspace.
72
73 .SH SEE ALSO
74 .IR cvs(1) ,
75 .IR hg(1) ,
76 .IR git(1) ,
77 .IR sccs(1)
78 .IR svn(1) ,
79 .IR workspace(1)
```

new/usr/src/tools/scripts/which\_scm.sh

1

```
*****
3935 Sat May 9 16:08:43 2009
new/usr/src/tools/scripts/which_scm.sh
Add git support to which_scm utility
*****
1 #!/usr/bin/ksh -p
2 #
3 # CDDL HEADER START
4 #
5 # The contents of this file are subject to the terms of the
6 # Common Development and Distribution License (the "License").
7 # You may not use this file except in compliance with the License.
8 #
9 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
10 # or http://www.opensolaris.org/os/licensing.
11 # See the License for the specific language governing permissions
12 # and limitations under the License.
13 #
14 # When distributing Covered Code, include this CDDL HEADER in each
15 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
16 # If applicable, add the following below this CDDL HEADER, with the
17 # fields enclosed by brackets "[]" replaced with your own identifying
18 # information: Portions Copyright [yyyy] [name of copyright owner]
19 #
20 # CDDL HEADER END
21 #
22 #
23 #
24 # Copyright 2008 Sun Microsystems, Inc. All rights reserved.
25 # Use is subject to license terms.
26 #
27 # ident "%Z%M% %I% %E% SMI"
28 #
29 #
30 # which_scm outputs two strings: one identifying the SCM in use, and
31 # the second giving the root directory for the SCM, if known, or just
32 # the current working directory if not known.
33 #
34 # There are three distinct types of SCM systems we can detect. The first
35 # type have a control directory per directory (RCS and SCCS), with no other
36 # structure. The second type have a control directory in each subdirectory
37 # within a tree (CVS and SVN). The last type have a single control
38 # directory at the top of the tree (Teamware and Mercurial).
39 #
40 # If the common CODEMGR_WS variable is set, then we look there for the
41 # SCM type and bail out if we can't determine it.
42 #
43 # If that variable is not set, then we start in the current directory
44 # and work our way upwards until we find the top of the tree or we
45 # encounter an error.
46 #
47 # We do handle nested SCM types, and report the innermost one, but if
48 # you nest one of the "second type" systems within another instance of
49 # itself, we'll keep going upwards and report the top of the nested
50 # set of trees.
51 #
52 #
53 # Check for well-known tree-type source code management (SCM) systems.
54 function primary_type
55 {
56     typeset scmid
57
58     [ -d "$1/Codemgr_wsdata" ] && scmid="$scmid teamware"
59     [ -d "$1/.hg" ] && scmid="$scmid mercurial"
60     [ -d "$1/CVS" ] && scmid="$scmid cvs"
61     [ -d "$1/.svn" ] && scmid="$scmid subversion"

```

new/usr/src/tools/scripts/which\_scm.sh

2

```
62 [ -d "$1/.git" ] && scmid="$scmid git"
63     echo $scmid
64 }
65
66 if [[ -n "$CODEMGR_WS" ]]; then
67     if [[ ! -d "$CODEMGR_WS" ]]; then
68         print -u2 "which_scm: $CODEMGR_WS is not a directory."
69         exit 1
70     fi
71     set -- $(primary_type "$CODEMGR_WS")
72     if [[ $# != 1 ]]; then
73         set -- unknown
74     fi
75     echo "$1 $CODEMGR_WS"
76     exit 0
77 fi
78
79 ORIG_CWD=$(pwd)
80
81 if [[ -d RCS ]]; then
82     echo "rcs $ORIG_CWD"
83     exit 0
84 fi
85
86 # If it's not Teamware, it could just be local SCCS.
87 LOCAL_TYPE=
88 [[ -d SCCS ]] && LOCAL_TYPE="sccs"
89
90 # Scan upwards looking for top of tree.
91 DIR=$ORIG_CWD
92 CWD_TYPE=$(primary_type "$DIR")
93 SCM_TYPE=
94 while [[ "$DIR" != / ]]; do
95     set -- $(primary_type "$DIR")
96     if [[ $# > 1 ]]; then
97         echo "unknown $ORIG_CWD"
98         exit 0
99     fi
100     SCM_TYPE="$1"
101     # We're done searching if we hit either a change in type or the top
102     # of a "third type" control system.
103     if [[ "$SCM_TYPE" != "$CWD_TYPE" || "$SCM_TYPE" == git || \
104         "$SCM_TYPE" == mercurial || "$SCM_TYPE" == teamware ]]; then
105         if [[ "$SCM_TYPE" != "$CWD_TYPE" || "$SCM_TYPE" == mercurial || \
106             "$SCM_TYPE" == teamware ]]; then
107             break
108         fi
109     PREVDIR="$DIR"
110     DIR=$(dirname "$DIR")
111 done
112
113 # We assume here that the system root directory isn't the root of the SCM.
114
115 # Check for the "second type" of repository. In all cases, we started
116 # out in the tree and stepped out on the last iteration, so we want
117 # $PREVDIR.
118 if [[ "$CWD_TYPE" == cvs || "$CWD_TYPE" == subversion ]]; then
119     echo "$CWD_TYPE $PREVDIR"
120     exit 0
121 fi
122
123 # If we still don't know what it is, then check for a local type in the
124 # original directory. If none, then we don't know what it is.
125 if [[ -z "$SCM_TYPE" ]]; then
126     if [[ -z "$LOCAL_TYPE" ]]; then
127         SCM_TYPE=unknown

```

**new/usr/src/tools/scripts/which\_scm.sh**

**3**

```
126         else
127             SCM_TYPE=$LOCAL_TYPE
128             DIR=$ORIG_CWD
129         fi
130 fi

132 echo "$SCM_TYPE $DIR"
133 exit 0
```

new/usr/src/tools/scripts/Makefile

1

```
*****
3010 Sat May 9 16:08:44 2009
new/usr/src/tools/scripts/Makefile
Add git-active utility
*****
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 #
21 #
22 # Copyright 2009 Sun Microsystems, Inc. All rights reserved.
23 # Use is subject to license terms.
24 #

26 SHELL=/usr/bin/ksh93

28 SHFILES= \
29     Install \
30     acr \
31     bfu \
32     bfudrop \
33     bindrop \
34     bldenv \
35     build_cscope \
36     bringovercheck \
37     checkpaths \
38     checkproto \
39     cstyle \
40     elfcmp \
41     flg.flp \
42     genoffsets \
43     hgsetup \
44     keywords \
45     makebfu \
46     mkacr \
47     mkbfu \
48     mkclosed \
49     nightly \
50     onblldrop \
51     protocmp.terse \
52     sccscheck \
53     sccscp \
54     sccshist \
55     sccsmv \
56     sccsrn \
57     sdrop \
58     webrev \
59     which_scm \
60     ws \
61     wx \
```

new/usr/src/tools/scripts/Makefile

2

```
62     wx2hg \
63     xref

65 PERLFILES= \
66     check_rtime \
67     jstyle \
68     mkreadme_osol \
69     mktpl \
70     validate_flg \
71     validate_paths \
72     wdiff

74 PYFILES= \
75     cddlchk \
76     copyrightchk \
77     git-active \
78     hdrchk \
79     hg-active \
80     mapfilechk \
81     rtick \
82     wsdiff

84 MANFILES= \
85     Install.1 \
86     acr.1 \
87     bldenv.1 \
88     bringovercheck.1 \
89     cddlchk.1 \
90     checkpaths.1 \
91     check_rtime.1 \
92     cstyle.1 \
93     flg.flp.1 \
94     hdrchk.1 \
95     hgsetup.1 \
96     jstyle.1 \
97     mapfilechk.1 \
98     mkacr.1 \
99     nightly.1 \
100    sccscheck.1 \
101    sccscp.1 \
102    sccsmv.1 \
103    sccsrn.1 \
104    webrev.1 \
105    which_scm.1 \
106    ws.1 \
107    wsdiff.1 \
108    wx.1 \
109    wx2hg.1 \
110    xref.1

112 MAKEFILES= \
113     xref.mk

115 ETCFILES= \
116     hgstyle \
117     its.conf \
118     its.reg

120 CLEANFILES = $(SHFILES) $(PERLFILES) $(PYFILES) bldenv.1

122 include ../Makefile.tools

124 OWNER=      root
125 GROUP=      bin

127 $(ROOTONBLDETCFILES) := FILEMODE= 644
```

**new/usr/src/tools/scripts/Makefile**

3

```
128 $(ROOTONBLDMAKEFILES) := FILEMODE= 644
129 $(ROOTONBLDMANIFILES) := FILEMODE= 644

131 .KEEP_STATE:

133 all: $(SHFILES) $(PERLFILES) $(PYFILES) $(MANIFILES) $(MAKEFILES)

135 install: all .WAIT $(ROOTONBLDSHFILES) $(ROOTONBLDPERLFILES) \
136 $(ROOTONBLDPYFILES) $(ROOTONBLDMANIFILES) \
137 $(ROOTONBLDMAKEFILES) $(ROOTONBLDETFILES)

139 clean:
140 $(RM) $(CLEANFILES)

142 bldenv: bldenv.sh stdenv.sh
143 $(RM) "$@"
144 sed -e '/# STDENV_START/ r stdenv.sh' bldenv.sh > "$@"
145 # Check for shell lint and fail if we hit warings
146 shlintout=$( /usr/bin/ksh93 -n "$@" 2>&1 ) ; \
147 [ "${shlintout}" != "" ] && \
148 { print -r -- "${shlintout}" ; false ; } || true
149 $(CHMOD) +x "$@"

151 bldenv.l: bldenv
152 $(RM) "$@"
153 (set +o errexit ; ksh93 $? --nroff ; true) 2>&1 | \
154 sed 's/\.DS/.nf/g;s/\.DE/.fi/' > "$@"

156 nightly: nightly.sh stdenv.sh
157 $(RM) "$@"
158 sed -e '/# STDENV_START/ r stdenv.sh' nightly.sh > nightly
159 $(CHMOD) +x "$@"

161 include ../Makefile.targ
```

new/usr/src/tools/scripts/webrev.sh

1

```
*****
89222 Sat May 9 16:08:44 2009
new/usr/src/tools/scripts/webrev.sh
Let webrev recognize git
*****
_____unchanged_portion_omitted_____

913 #
914 # fix_postscript
915 #

1780 #
1781 # Call git-active to get the active list output in the wx active list format
1782 #
1783 function git_active_wxfile
1784 {
1785     typeset child=$1
1786     typeset parent=$2

1788     TMPFLIST=/tmp/$.active
1789     $GIT_ACTIVE -w $child -p $parent -o $TMPFLIST
1790     wxfile=$TMPFLIST
1791 }

1793 #
1794 # flist_from_git
1795 # Call git-active to get a wx-style active list, and hand it off to
1796 # flist_from_wx
1797 #
1798 function flist_from_git
1799 {
1800     typeset child=$1
1801     typeset parent=$2

1803     print " File list from: git-active -p $parent ...`c"

1805     if [[ ! -x $GIT_ACTIVE ]]; then
1806         print          # Blank line for the `c above
1807         print -u2 "Error: git-active tool not found. Exiting"
1808         exit 1
1809     fi
1810     git_active_wxfile $child $parent
1811
1812     # flist_from_wx prints the Done, so we don't have to.
1813     flist_from_wx $TMPFLIST
1814 }

1816 #
1817 # flist_from_subversion
1818 #
1819 # Generate the file list by extracting file names from svn status.
1820 #
1821 function flist_from_subversion
1822 {
1823     CWS=$1
1824     OLDPWD=$2

1826     cd $CWS
1827     print -u2 " File list from: svn status ... `c"
1828     svn status | $AWK '/^[ACDMR]/ { print $NF }' > $FLIST
1829     print -u2 " Done."
1830     cd $OLDPWD
1831 }
_____unchanged_portion_omitted_____
```

new/usr/src/tools/scripts/webrev.sh

2

```
2036 function build_old_new_git
2037 {
2038     typeset olddir="$1"
2039     typeset newdir="$2"
2040     typeset old_mode=
2041     typeset new_mode=
2042     typeset old_object=
2043     typeset new_object=
2044     typeset file
2045     typeset type

2047     #
2048     # Get old file and its mode from the git object tree
2049     #
2050     if [[ "$PDIR" == "." ]]; then
2051         file="$PF"
2052     else
2053         file="$PDIR/$PF"
2054     fi
2055     git ls-tree $GIT_PARENT $file | read old_mode type old_object junk
2056     git cat-file $type $old_object > $olddir/$file 2>/dev/null

2058     if (( $? != 0 )); then
2059         rm -f $olddir/$PDIR/$PF
2060     else
2061         if [[ -n $old_mode ]]; then
2062             chmod $old_mode $olddir/$PDIR/$PF
2063         else
2064             # should never happen
2065             print -u2 "ERROR: set mode of $olddir/$PDIR/$PF"
2066         fi
2067     fi

2069     #
2070     # new version of the file.
2071     #
2072     rm -rf $newdir/$DIR/$F
2073     if [[ -e $CWS/./$DIR/$F ]]; then
2074         cp $CWS/./$DIR/$F $newdir/$DIR/$F
2075         # Temporary new_node = old_mode
2076         new_mode=$old_mode
2077         if [[ -n $new_mode ]]; then
2078             chmod $new_mode $newdir/$DIR/$F
2079         else
2080             # should never happen
2081             print -u2 "ERROR: set mode of $newdir/$DIR/$F"
2082         fi
2083     fi

2085 }

2087 function build_old_new_subversion
2088 {
2089     typeset olddir="$1"
2090     typeset newdir="$2"

2092     # Snag new version of file.
2093     rm -f $newdir/$DIR/$F
2094     [[ -e $CWS/$DIR/$F ]]&& cp $CWS/$DIR/$F $newdir/$DIR/$F

2096     if [[ -n $PWS && -e $PWS/$PDIR/$PF ]]; then
2097         cp $PWS/$PDIR/$PF $olddir/$PDIR/$PF
2098     else
2099         # Get the parent's version of the file.
2100         svn status $CWS/$DIR/$F | read stat file
2101         if [[ $stat != "A" ]]; then
```

new/usr/src/tools/scripts/webrev.sh

3

```
2102          svn cat -r BASE $CWS/$DIR/$F > $olddir/$PDIR/$PF
2103          fi
2104          fi
2105 }
_____ unchanged_portion_omitted _____
2127 function build_old_new
2128 {
2129     typeset WDIR=$1
2130     typeset PWS=$2
2131     typeset PDIR=$3
2132     typeset PF=$4
2133     typeset CWS=$5
2134     typeset DIR=$6
2135     typeset F=$7
2137     typeset olddir="$WDIR/raw_files/old"
2138     typeset newdir="$WDIR/raw_files/new"
2140     mkdir -p $olddir/$PDIR
2141     mkdir -p $newdir/$DIR
2143     if [[ $SCM_MODE == "teamware" ]]; then
2144         build_old_new_teamware "$olddir" "$newdir"
2145     elif [[ $SCM_MODE == "mercurial" ]]; then
2146         build_old_new_mercurial "$olddir" "$newdir"
2147     elif [[ $SCM_MODE == "git" ]]; then
2148         build_old_new_git "$olddir" "$newdir"
2149     elif [[ $SCM_MODE == "subversion" ]]; then
2150         build_old_new_subversion "$olddir" "$newdir"
2151     elif [[ $SCM_MODE == "unknown" ]]; then
2152         build_old_new_unknown "$olddir" "$newdir"
2153     fi
2155     if [[ ! -f $olddir/$PDIR/$PF && ! -f $newdir/$DIR/$F ]]; then
2156         print "**** Error: file not in parent or child"
2157         return 1
2158     fi
2159     return 0
2160 }
_____ unchanged_portion_omitted _____
2200 #
2201 #
2202 # Main program starts here
2203 #
2204 #
2206 trap "rm -f /tmp/$$.* ; exit" 0 1 2 3 15
2208 set +o noclobber
2210 PATH=$(dirname $(whence $0)):PATH
2212 [[ -z $WDIFF ]] && WDIFF='look_for_prog wdiff'
2213 [[ -z $WX ]] && WX='look_for_prog wx'
2214 [[ -z $HG_ACTIVE ]] && HG_ACTIVE='look_for_prog hg-active'
2215 [[ -z $GIT_ACTIVE ]] && GIT_ACTIVE='look_for_prog git-active'
2216 [[ -z $WHICH_SCM ]] && WHICH_SCM='look_for_prog which_scm'
2217 [[ -z $CODEREVIEW ]] && CODEREVIEW='look_for_prog codereview'
2218 [[ -z $PS2PDF ]] && PS2PDF='look_for_prog ps2pdf'
2219 [[ -z $PERL ]] && PERL='look_for_prog perl'
2220 [[ -z $RSYNC ]] && RSYNC='look_for_prog rsync'
2221 [[ -z $SCCS ]] && SCCS='look_for_prog sccs'
2222 [[ -z $AWK ]] && AWK='look_for_prog awk'
2223 [[ -z $AWK ]] && AWK='look_for_prog gawk'
```

new/usr/src/tools/scripts/webrev.sh

4

```
2224 [[ -z $AWK ]] && AWK='look_for_prog awk'
2225 [[ -z $SCP ]] && SCP='look_for_prog scp'
2226 [[ -z $SED ]] && SED='look_for_prog sed'
2227 [[ -z $SFTP ]] && SFTP='look_for_prog sftp'
2228 [[ -z $MKTEMP ]] && MKTEMP='look_for_prog mktemp'
2229 [[ -z $GREP ]] && GREP='look_for_prog grep'
2230 [[ -z $FIND ]] && FIND='look_for_prog find'
2232 # set name of trash directory for remote webrev deletion
2233 TRASH_DIR=".trash"
2234 [[ -n $WEBREV_TRASH_DIR ]] && TRASH_DIR=$WEBREV_TRASH_DIR
2236 if [[ ! -x $PERL ]]; then
2237     print -u2 "Error: No perl interpreter found. Exiting."
2238     exit 1
2239 fi
2241 if [[ ! -x $WHICH_SCM ]]; then
2242     print -u2 "Error: Could not find which_scm. Exiting."
2243     exit 1
2244 fi
2246 #
2247 # These aren't fatal, but we want to note them to the user.
2248 # We don't warn on the absence of 'wx' until later when we've
2249 # determined that we actually need to try to invoke it.
2250 #
2251 [[ ! -x $CODEREVIEW ]] && print -u2 "WARNING: codereview(1) not found."
2252 [[ ! -x $PS2PDF ]] && print -u2 "WARNING: ps2pdf(1) not found."
2253 [[ ! -x $WDIFF ]] && print -u2 "WARNING: wdiff not found."
2255 # Declare global total counters.
2256 integer TOTL TINS TDEL TMOD TUNC
2258 # default remote host for upload/delete
2259 typeset -r DEFAULT_REMOTE_HOST="cr.opensolaris.org"
2260 # prefixes for upload targets
2261 typeset -r rsync_prefix="rsync://"
2262 typeset -r ssh_prefix="ssh://"
2264 Cflag=
2265 Dflag=
2266 flist_mode=
2267 flist_file=
2268 iflag=
2269 lflag=
2270 lflag=
2271 nflag=
2272 nflag=
2273 oflag=
2274 oflag=
2275 pflag=
2276 tflag=
2277 uflag=
2278 Uflag=
2279 wflag=
2280 remote_target=
2282 #
2283 # NOTE: when adding/removing options it is necessary to sync the list
2284 # with usr/src/tools/onbld/hgext/cdm.py
2285 #
2286 while getopts "C:Di:I:lnNo:Op:t:Uw" opt
2287 do
2288     case $opt in
2289         C) Cflag=1
```

new/usr/src/tools/scripts/webrev.sh

5

```
2290             ITSCONF=$OPTARG;;
2292     D)         Dflag=1;;
2294     i)         iflag=1
2295             INCLUDE_FILE=$OPTARG;;
2297     I)         Iflag=1
2298             ITSREG=$OPTARG;;
2300     #
2301     # If -l has been specified, we need to abort further options
2302     # processing, because subsequent arguments are going to be
2303     # arguments to 'putback -n'.
2304     #
2305     l)         lflag=1
2306             break;;
2308     N)         Nflag=1;;
2310     n)         nflag=1;;
2312     O)         Oflag=1;;
2314     o)         oflag=1
2315             WDIR=$OPTARG;;
2317     p)         pflag=1
2318             codemgr_parent=$OPTARG;;
2320     t)         tflag=1
2321             remote_target=$OPTARG;;
2323     U)         Uflag=1;;
2325     w)         wflag=1;;
2327     ?)         usage;;
2328     esac
2329 done
2331 FLIST=/tmp/$$.flist
2333 if [[ -n $wflag && -n $lflag ]]; then
2334     usage
2335 fi
2337 # more sanity checking
2338 if [[ -n $nflag && -z $Uflag ]]; then
2339     print "it does not make sense to skip webrev generation" \
2340         "without -U"
2341     exit 1
2342 fi
2344 if [[ -n $tflag && -z $Uflag && -z $Dflag ]]; then
2345     echo "remote target has to be used only for upload or delete"
2346     exit 1
2347 fi
2349 #
2350 # For the invocation "webrev -n -U" with no other options, webrev will assume
2351 # that the webrev exists in ${CWS}/webrev, but will upload it using the name
2352 # $(basename ${CWS}). So we need to get CWS set before we skip any remaining
2353 # logic.
2354 #
2355 $WHICH_SCM | read SCM_MODE junk || exit 1
```

new/usr/src/tools/scripts/webrev.sh

6

```
2356 if [[ $SCM_MODE == "teamware" ]]; then
2357     #
2358     # Teamware priorities:
2359     # 1. CODEMGR_WS from the environment
2360     # 2. workspace name
2361     #
2362     [[ -z $codemgr_ws && -n $CODEMGR_WS ]] && codemgr_ws=$CODEMGR_WS
2363     if [[ -n $codemgr_ws && ! -d $codemgr_ws ]]; then
2364         print -u2 "$codemgr_ws: no such workspace"
2365         exit 1
2366     fi
2367     [[ -z $codemgr_ws ]] && codemgr_ws=$(workspace name)
2368     codemgr_ws=$(cd $codemgr_ws;print $PWD)
2369     CODEMGR_WS=$codemgr_ws
2370     CWS=$codemgr_ws
2371 elif [[ $SCM_MODE == "mercurial" ]]; then
2372     #
2373     # Mercurial priorities:
2374     # 1. hg root from CODEMGR_WS environment variable
2375     # 2. hg root from directory of invocation
2376     #
2377     [[ -z $codemgr_ws && -n $CODEMGR_WS ]] && \
2378         codemgr_ws=$(hg root -R $CODEMGR_WS 2>/dev/null)
2379     [[ -z $codemgr_ws ]] && codemgr_ws=$(hg root 2>/dev/null)
2380     CWS=$codemgr_ws
2381 elif [[ $SCM_MODE == "git" ]]; then
2382     #
2383     # Git priorities:
2384     # 1. git rev-parse --git-dir from CODEMGR_WS environment variable
2385     # 2. git rev-parse --git-dir from directory of invocation
2386     #
2387     [[ -z $codemgr_ws && -n $CODEMGR_WS ]] && \
2388         codemgr_ws=$(git --git-dir=$CODEMGR_WS rev-parse --git-dir \
2389             2>/dev/null)
2390     [[ -z $codemgr_ws ]] && \
2391         codemgr_ws=$(git rev-parse --git-dir 2>/dev/null)
2392     CWS=$codemgr_ws
2393 elif [[ $SCM_MODE == "subversion" ]]; then
2394     #
2395     # Subversion priorities:
2396     # 1. CODEMGR_WS from environment
2397     # 2. Relative path from current directory to SVN repository root
2398     #
2399     if [[ -n $CODEMGR_WS && -d $CODEMGR_WS/.svn ]]; then
2400         CWS=$CODEMGR_WS
2401     else
2402         svn info | while read line; do
2403             if [[ $line == "URL: *" ]]; then
2404                 url=${line#URL: }
2405                 elif [[ $line == "Repository Root: *" ]]; then
2406                     repo=${line#Repository Root: }
2407             fi
2408         done
2410         rel=${url#$repo}
2411         CWS=${PWD%$rel}
2412     fi
2413 fi
2415 #
2416 # If no SCM has been determined, take either the environment setting
2417 # setting for CODEMGR_WS, or the current directory if that wasn't set.
2418 #
2419 if [[ -z ${CWS} ]]; then
2420     CWS=${CODEMGR_WS:-.}
2421 fi
```

```

2425 #
2426 # If the command line options indicate no webrev generation, either
2427 # explicitly (-n) or implicitly (-D but not -U), then there's a whole
2428 # ton of logic we can skip.
2429 #
2430 # Instead of increasing indentation, we intentionally leave this loop
2431 # body open here, and exit via break from multiple points within.
2432 # Search for DO_EVERYTHING below to find the break points and closure.
2433 #
2434 for do_everything in 1; do

2436 # DO_EVERYTHING: break point
2437 if [[ -n $nflag || ( -z $Uflag && -n $Dflag ) ]]; then
2438     break
2439 fi

2441 #
2442 # If this manually set as the parent, and it appears to be an earlier webrev,
2443 # then note that fact and set the parent to the raw_files/new subdirectory.
2444 #
2445 if [[ -n $pflag && -d $codemgr_parent/raw_files/new ]]; then
2446     parent_webrev="$codemgr_parent"
2447     codemgr_parent="$codemgr_parent/raw_files/new"
2448 fi

2450 if [[ -z $wflag && -z $lflag ]]; then
2451     shift $((SOPTIND - 1))

2453     if [[ $1 == "-" ]]; then
2454         cat > $FLIST
2455         flist_mode="stdin"
2456         flist_done=1
2457         shift
2458     elif [[ -n $1 ]]; then
2459         if [[ ! -r $1 ]]; then
2460             print -u2 "$1: no such file or not readable"
2461             usage
2462         fi
2463         cat $1 > $FLIST
2464         flist_mode="file"
2465         flist_file=$1
2466         flist_done=1
2467         shift
2468     else
2469         flist_mode="auto"
2470     fi
2471 fi

2473 #
2474 # Before we go on to further consider -l and -w, work out which SCM we think
2475 # is in use.
2476 #
2477 case "$SCM_MODE" in
2478     teamware|mercurial|git|subversion)
2479     ;;
2480     unknown)
2481         if [[ $flist_mode == "auto" ]]; then
2482             print -u2 "Unable to determine SCM in use and file list not spec
2483             print -u2 "See which_scm(1) for SCM detection information."
2484             exit 1
2485         fi
2486         ;;

```

```

2487 *)
2488     if [[ $flist_mode == "auto" ]]; then
2489         print -u2 "Unsupported SCM in use ($SCM_MODE) and file list not
2490         exit 1
2491     fi
2492     ;;
2493 esac

2495 print -u2 "    SCM detected: $SCM_MODE"

2497 if [[ -n $lflag ]]; then
2498     #
2499     # If the -l flag is given instead of the name of a file list,
2500     # then generate the file list by extracting file names from a
2501     # putback -n.
2502     #
2503     shift $((SOPTIND - 1))
2504     if [[ $SCM_MODE == "teamware" ]]; then
2505         flist_from_teamware "$*"
2506     else
2507         print -u2 -- "Error: -l option only applies to TeamWare"
2508         exit 1
2509     fi
2510     flist_done=1
2511     shift $#
2512 elif [[ -n $wflag ]]; then
2513     #
2514     # If the -w is given then assume the file list is in Bonwick's "wx"
2515     # command format, i.e. pathname lines alternating with SCCS comment
2516     # lines with blank lines as separators. Use the SCCS comments later
2517     # in building the index.html file.
2518     #
2519     shift $((SOPTIND - 1))
2520     wxfile=$1
2521     if [[ -z $wxfile && -n $CODEMGR_WS ]]; then
2522         if [[ -r $CODEMGR_WS/wx/active ]]; then
2523             wxfile=$CODEMGR_WS/wx/active
2524         fi
2525     fi

2527     [[ -z $wxfile ]] && print -u2 "wx file not specified, and could not " \
2528     "be auto-detected (check \$CODEMGR_WS)" && exit 1

2530     if [[ ! -r $wxfile ]]; then
2531         print -u2 "$wxfile: no such file or not readable"
2532         usage
2533     fi

2535     print -u2 " File list from: wx 'active' file '$wxfile' ... \c"
2536     flist_from_wx $wxfile
2537     flist_done=1
2538     if [[ -n "$*" ]]; then
2539         shift
2540     fi
2541 elif [[ $flist_mode == "stdin" ]]; then
2542     print -u2 " File list from: standard input"
2543 elif [[ $flist_mode == "file" ]]; then
2544     print -u2 " File list from: $flist_file"
2545 fi

2547 if [[ $# -gt 0 ]]; then
2548     print -u2 "WARNING: unused arguments: $"
2549 fi

2551 #
2552 # Before we entered the DO_EVERYTHING loop, we should have already set CWS

```

```

2553 # and CODEMGR_WS as needed. Here, we set the parent workspace.
2554 #
2555
2556 if [[ $SCM_MODE == "teamware" ]]; then
2557
2558     #
2559     # Teamware priorities:
2560     #
2561     #     1) via -p command line option
2562     #     2) in the user environment
2563     #     3) in the flist
2564     #     4) automatically based on the workspace
2565     #
2566
2567     #
2568     # For 1, codemgr_parent will already be set. Here's 2:
2569     #
2570     [[ -z $codemgr_parent && -n $CODEMGR_PARENT ]] && \
2571     codemgr_parent=$CODEMGR_PARENT
2572     if [[ -n $codemgr_parent && ! -d $codemgr_parent ]]; then
2573         print -u2 "$codemgr_parent: no such directory"
2574         exit 1
2575     fi
2576
2577     #
2578     # If we're in auto-detect mode and we haven't already gotten the file
2579     # list, then see if we can get it by probing for wx.
2580     #
2581     if [[ -z $flist_done && $flist_mode == "auto" && -n $codemgr_ws ]]; then
2582         if [[ ! -x $WX ]]; then
2583             print -u2 "WARNING: wx not found!"
2584         fi
2585
2586         #
2587         # We need to use wx list -w so that we get renamed files, etc.
2588         # but only if a wx active file exists-- otherwise wx will
2589         # hang asking us to initialize our wx information.
2590         #
2591         if [[ -x $WX && -f $codemgr_ws/wx/active ]]; then
2592             print -u2 " File list from: 'wx list -w' ... \c"
2593             $WX list -w > $FLIST
2594             $WX comments > /tmp/$$.wx_comments
2595             wxfile=/tmp/$$.wx_comments
2596             print -u2 "done"
2597             flist_done=1
2598         fi
2599     fi
2600
2601     #
2602     # If by hook or by crook we've gotten a file list by now (perhaps
2603     # from the command line), eval it to extract environment variables from
2604     # it: This is method 3 for finding the parent.
2605     #
2606     if [[ -z $flist_done ]]; then
2607         flist_from_teamware
2608     fi
2609     env_from_flist
2610
2611     #
2612     # (4) If we still don't have a value for codemgr_parent, get it
2613     # from workspace.
2614     #
2615     [[ -z $codemgr_parent ]] && codemgr_parent='workspace parent'
2616     if [[ ! -d $codemgr_parent ]]; then
2617         print -u2 "$CODEMGR_PARENT: no such parent workspace"
2618         exit 1

```

```

2619     fi
2620
2621     PWS=$codemgr_parent
2622 \
2623
2624     [[ -n $parent_webrev ]] && RWS=$(workspace parent $CWS)
2625
2626 elif [[ $SCM_MODE == "mercurial" ]]; then
2627     #
2628     # Parent can either be specified with -p
2629     # Specified with CODEMGR_PARENT in the environment
2630     # or taken from hg's default path.
2631     #
2632
2633     if [[ -z $codemgr_parent && -n $CODEMGR_PARENT ]]; then
2634         codemgr_parent=$CODEMGR_PARENT
2635     fi
2636
2637     if [[ -z $codemgr_parent ]]; then
2638         codemgr_parent='hg path -R $codemgr_ws default 2>/dev/null'
2639     fi
2640
2641     CWS_REV='hg parent -R $codemgr_ws --template '{node|short}' 2>/dev/null'
2642     PWS=$codemgr_parent
2643
2644     #
2645     # If the parent is a webrev, we want to do some things against
2646     # the natural workspace parent (file list, comments, etc)
2647     #
2648     if [[ -n $parent_webrev ]]; then
2649         real_parent=$(hg path -R $codemgr_ws default 2>/dev/null)
2650     else
2651         real_parent=$PWS
2652     fi
2653
2654     #
2655     # If hg-active exists, then we run it. In the case of no explicit
2656     # flist given, we'll use it for our comments. In the case of an
2657     # explicit flist given we'll try to use it for comments for any
2658     # files mentioned in the flist.
2659     #
2660     if [[ -z $flist_done ]]; then
2661         flist_from_mercurial $CWS $real_parent
2662         flist_done=1
2663     fi
2664
2665     #
2666     # If we have a file list now, pull out any variables set
2667     # therein. We do this now (rather than when we possibly use
2668     # hg-active to find comments) to avoid stomping specifications
2669     # in the user-specified flist.
2670     #
2671     if [[ -n $flist_done ]]; then
2672         env_from_flist
2673     fi
2674
2675     #
2676     # Only call hg-active if we don't have a wx formatted file already
2677     #
2678     if [[ -x $HG_ACTIVE && -z $wxfile ]]; then
2679         print " Comments from: hg-active -p $real_parent ...\c"
2680         hg_active_wxfile $CWS $real_parent
2681         print " Done."
2682     fi
2683
2684     #

```

```

2685 # At this point we must have a wx flist either from hg-active,
2686 # or in general. Use it to try and find our parent revision,
2687 # if we don't have one.
2688 #
2689 if [[ -z $HG_PARENT ]]; then
2690     eval `\$SED -e "s/#.*$//" $wxfile | $GREP HG_PARENT=`
2691 fi

2693 #
2694 # If we still don't have a parent, we must have been given a
2695 # wx-style active list with no HG_PARENT specification, run
2696 # hg-active and pull an HG_PARENT out of it, ignore the rest.
2697 #
2698 if [[ -z $HG_PARENT && -x $HG_ACTIVE ]]; then
2699     SHG_ACTIVE -w $codemgr_ws -p $real_parent | \
2700     eval `\$SED -e "s/#.*$//" | $GREP HG_PARENT=`
2701 elif [[ -z $HG_PARENT ]]; then
2702     print -u2 "Error: Cannot discover parent revision"
2703     exit 1
2704 fi
2705 elif [[ $SCM_MODE == "git" ]]; then
2706 #
2707 # Parent can either be specified with -p
2708 # Specified with CODEMGR_PARENT in the environment
2709 # or taken from git config.
2710 #
2712 if [[ -z $codemgr_parent && -n $CODEMGR_PARENT ]]; then
2713     codemgr_parent=$CODEMGR_PARENT
2714 fi

2716 if [[ -z $codemgr_parent ]]; then
2717     codemgr_parent=$(git --git-dir=$codemgr_ws config \
2718         remote.origin.url 2>/dev/null)
2719 fi

2721 CWS_REV=$(git --git-dir=$codemgr_ws rev-parse HEAD 2>/dev/null)
2722 PWS=$codemgr_parent

2724 #
2725 # If the parent is a webrev, we want to do some things against
2726 # the natural workspace parent (file list, comments, etc)
2727 #
2728 if [[ -n $parent_webrev ]]; then
2729     real_parent=$(git --git-dir $codemgr_ws config \
2730         remote.origin.url 2>/dev/null)
2731 else
2732     real_parent=$PWS
2733 fi

2735 #
2736 # If git-active exists, then we run it. In the case of no explicit
2737 # flist given, we'll use it for our comments. In the case of an
2738 # explicit flist given we'll try to use it for comments for any
2739 # files mentioned in the flist.
2740 #
2741 if [[ -z $flist_done ]]; then
2742     flist_from_git $CWS $real_parent
2743     flist_done=1
2744 fi

2746 #
2747 # If we have a file list now, pull out any variables set
2748 # therein. We do this now (rather than when we possibly use
2749 # git-active to find comments) to avoid stomping specifications
2750 # in the user-specified flist.

```

```

2751 #
2752 if [[ -n $flist_done ]]; then
2753     env_from_flist
2754 fi

2756 #
2757 # Only call git-active if we don't have a wx formatted file already
2758 #
2759 if [[ -x $GIT_ACTIVE && -z $wxfile ]]; then
2760     print " Comments from: git-active -p $real_parent ...c"
2761     git_active_wxfile $CWS $real_parent
2762     print " Done."
2763 fi

2764 #
2765 # At this point we must have a wx flist either from git-active,
2766 # or in general. Use it to try and find our parent revision,
2767 # if we don't have one.
2768 #
2769 if [[ -z $GIT_PARENT ]]; then
2770     eval `\$SED -e "s/#.*$//" $wxfile | $GREP GIT_PARENT=`
2771 fi

2774 #
2775 # If we still don't have a parent, we must have been given a
2776 # wx-style active list with no GIT_PARENT specification, run
2777 # git-active and pull an GIT_PARENT out of it, ignore the rest.
2778 #
2779 if [[ -z $GIT_PARENT && -x $GIT_ACTIVE ]]; then
2780     $GIT_ACTIVE -w $codemgr_ws -p $real_parent | \
2781     eval `\$SED -e "s/#.*$//" | $GREP GIT_PARENT=`
2782 elif [[ -z $GIT_PARENT ]]; then
2783     print -u2 "Error: Cannot discover parent revision"
2784     exit 1
2785 fi
2786 WDIR=${WDIR:-$CWS/./webrev}
2787 elif [[ $SCM_MODE == "subversion" ]]; then

2789 #
2790 # We only will have a real parent workspace in the case one
2791 # was specified (be it an older webrev, or another checkout).
2792 #
2793 [[ -n $codemgr_parent ]] && PWS=$codemgr_parent

2795 if [[ -z $flist_done && $flist_mode == "auto" ]]; then
2796     flist_from_subversion $CWS $OLDPWD
2797 fi

2798 else
2799     if [[ $SCM_MODE == "unknown" ]]; then
2800         print -u2 " Unknown type of SCM in use"
2801     else
2802         print -u2 " Unsupported SCM in use: $SCM_MODE"
2803     fi

2805     env_from_flist

2807 if [[ -z $CODEMGR_WS ]]; then
2808     print -u2 "SCM not detected/supported and CODEMGR_WS not specified"
2809     exit 1
2810 fi

2812 if [[ -z $CODEMGR_PARENT ]]; then
2813     print -u2 "SCM not detected/supported and CODEMGR_PARENT not specified"
2814     exit 1
2815 fi

```

```

2817     CWS=$CODEMGR_WS
2818     PWS=$CODEMGR_PARENT
2819 fi

2821 #
2822 # If the user didn't specify a -i option, check to see if there is a
2823 # webrev-info file in the workspace directory.
2824 #
2825 if [[ -z $iflag && -r "$CWS/webrev-info" ]]; then
2826     iflag=1
2827     INCLUDE_FILE="$CWS/webrev-info"
2828 fi

2830 if [[ -n $iflag ]]; then
2831     if [[ ! -r $INCLUDE_FILE ]]; then
2832         print -u2 "include file '$INCLUDE_FILE' does not exist or is" \
2833             "not readable."
2834         exit 1
2835     else
2836         #
2837         # $INCLUDE_FILE may be a relative path, and the script alters
2838         # PWD, so we just stash a copy in /tmp.
2839         #
2840         cp $INCLUDE_FILE /tmp/.$$include
2841     fi
2842 fi

2844 # DO EVERYTHING: break point
2845 if [[ -n $Nflag ]]; then
2846     break
2847 fi

2849 typeset -A itsinfo
2850 typeset -r its_sed_script=/tmp/.$$its_sed
2851 valid_prefixes=
2852 if [[ -z $nflag ]]; then
2853     DEFREGFILE="$(dirname $(whence $0))/../etc/its.reg"
2854     if [[ -n $iflag ]]; then
2855         REGFILE=$ITSREG
2856     elif [[ -r $HOME/.its.reg ]]; then
2857         REGFILE=$HOME/.its.reg
2858     else
2859         REGFILE=$DEFREGFILE
2860     fi
2861     if [[ ! -r $REGFILE ]]; then
2862         print "ERROR: Unable to read database registry file $REGFILE"
2863         exit 1
2864     elif [[ $REGFILE != $DEFREGFILE ]]; then
2865         print "    its.reg from: $REGFILE"
2866     fi

2868     $SED -e '/^#/d' -e '/^[      ]*$/d' $REGFILE | while read LINE; do
2869         name=${LINE%%=*}
2870         value=${LINE#*=}
2871
2873         if [[ $name == PREFIX ]]; then
2874             p=${value}
2875             valid_prefixes="$p ${valid_prefixes}"
2876         else
2877             itsinfo["${p}_${name}"]="${value}"
2878         fi
2879     done

2882     DEFCONFFILE="$(dirname $(whence $0))/../etc/its.conf"

```

```

2883     CONFFILES=$DEFCONFFILE
2884     if [[ -r $HOME/.its.conf ]]; then
2885         CONFFILES="${CONFFILES} $HOME/.its.conf"
2886     fi
2887     if [[ -n $Cflag ]]; then
2888         CONFFILES="${CONFFILES} ${ITSCONF}"
2889     fi
2890     its_domain=
2891     its_priority=
2892     for cf in ${CONFFILES}; do
2893         if [[ ! -r $cf ]]; then
2894             print "ERROR: Unable to read database configuration file"
2895             exit 1
2896         elif [[ $cf != $DEFCONFFILE ]]; then
2897             print "    its.conf: reading $cf"
2898         fi
2899         $SED -e '/^#/d' -e '/^[      ]*$/d' $cf | while read LINE; do
2900             eval "${LINE}"
2901         done
2902     done

2904     #
2905     # If an information tracking system is explicitly identified by prefix,
2906     # we want to disregard the specified priorities and resolve it according
2907     #
2908     # To that end, we'll build a sed script to do each valid prefix in turn.
2909     #
2910     for p in ${valid_prefixes}; do
2911         #
2912         # When an informational URL was provided, translate it to a
2913         # hyperlink.  When omitted, simply use the prefix text.
2914         #
2915         if [[ -z ${itsinfo["${p}_INFO"]} ]]; then
2916             itsinfo["${p}_INFO"]=${p}
2917         else
2918             itsinfo["${p}_INFO"]="

```

```

2949         }" >> ${its_sed_script}
2950     done

2952     #
2953     # The previous loop took care of explicit specification. Now use
2954     # the configured priorities to attempt implicit translations.
2955     #
2956     for p in ${its_priority}; do
2957         print "/^${itsinfo[${p}_REGEX]}{          }/ {
2958             s:${itsinfo[${p}_REGEX]};${itsinfo[${p}_URL]};g
2959         }" >> ${its_sed_script}
2960     done
2961 fi

2963 #
2964 # Search for DO_EVERYTHING above for matching "for" statement
2965 # and explanation of this terminator.
2966 #
2967 done

2969 #
2970 # Output directory.
2971 #
2972 WDIR=${WDIR:-$CWS/webrev}

2974 #
2975 # Name of the webrev, derived from the workspace name or output directory;
2976 # in the future this could potentially be an option.
2977 #
2978 if [[ -n $oflag ]]; then
2979     WNAME=${WDIR##*/}
2980 else
2981     WNAME=${CWS##*/}
2982 fi

2984 # Make sure remote target is well formed for remote upload/delete.
2985 if [[ -n $Dflag || -n $Uflag ]]; then
2986     #
2987     # If remote target is not specified, build it from scratch using
2988     # the default values.
2989     #
2990     if [[ -z $tflag ]]; then
2991         remote_target=${DEFAULT_REMOTE_HOST}:${WNAME}
2992     else
2993         #
2994         # Check upload target prefix first.
2995         #
2996         if [[ "${remote_target}" != ${rsync_prefix}* &&
2997             "${remote_target}" != ${ssh_prefix}* ]]; then
2998             print "ERROR: invalid prefix of upload URI" \
2999                 "(${remote_target})"
3000             exit 1
3001         fi
3002         #
3003         # If destination specification is not in the form of
3004         # host_spec:remote_dir then assume it is just remote hostname
3005         # and append a colon and destination directory formed from
3006         # local webrev directory name.
3007         #
3008         typeset target_no_prefix=${remote_target##*/}
3009         if [[ ${target_no_prefix} == /* ]]; then
3010             if [[ "${remote_target}" == /* ]]; then
3011                 remote_target=${remote_target}${WNAME}
3012             fi
3013         else
3014             if [[ ${target_no_prefix} == /* ]]; then

```

```

3015         print "ERROR: badly formed upload URI" \
3016             "(${remote_target})"
3017         exit 1
3018     else
3019         remote_target=${remote_target}:${WNAME}
3020     fi
3021 fi
3022 fi

3024 #
3025 # Strip trailing slash. Each upload method will deal with directory
3026 # specification separately.
3027 #
3028 remote_target=${remote_target%/}
3029 fi

3031 #
3032 # Option -D by itself (option -U not present) implies no webrev generation.
3033 #
3034 if [[ -z $Uflag && -n $Dflag ]]; then
3035     delete_webrev 1 1
3036     exit $?
3037 fi

3039 #
3040 # Do not generate the webrev, just upload it or delete it.
3041 #
3042 if [[ -n $nflag ]]; then
3043     if [[ -n $Dflag ]]; then
3044         delete_webrev 1 1
3045         (( $? == 0 )) || exit $?
3046     fi
3047     if [[ -n $Uflag ]]; then
3048         upload_webrev
3049         exit $?
3050     fi
3051 fi

3053 if [ "${WDIR%/*}" ]; then
3054     WDIR=$PWD/$WDIR
3055 fi

3057 if [[ ! -d $WDIR ]]; then
3058     mkdir -p $WDIR
3059     (( $? != 0 )) && exit 1
3060 fi

3062 #
3063 # Summarize what we're going to do.
3064 #
3065 if [[ -n $CWS_REV ]]; then
3066     print "    Workspace: $CWS (at $CWS_REV)"
3067 else
3068     print "    Workspace: $CWS"
3069 fi
3070 if [[ -n $parent_webrev ]]; then
3071     print "Compare against: webrev at $parent_webrev"
3072 else
3073     if [[ -n $HG_PARENT ]]; then
3074         hg_parent_short='echo $HG_PARENT \
3075             | $SED -e 's/\([0-9a-f]\{12\}\).*\/1/'
3076         print "Compare against: $PWS (at $hg_parent_short)"
3077     else
3078         print "Compare against: $PWS"
3079     fi
3080 fi

```



```

3213 build_old_new "$WDIR" "$PWS" "$PDIR" "$PF" "$CWS" "$DIR" "$F" || \
3214 continue

3216 #
3217 # Keep the old PWD around, so we can safely switch back after
3218 # diff generation, such that build_old_new runs in a
3219 # consistent environment.
3220 #
3221 OWD=$PWD
3222 cd $WDIR/raw_files
3223 ofile=old/$PDIR/$PF
3224 nfile=new/$DIR/$F

3226 mv_but_nodiff=
3227 cmp $ofile $nfile > /dev/null 2>&1
3228 if [[ $? == 0 && $rename == 1 ]]; then
3229     mv_but_nodiff=1
3230 fi

3232 #
3233 # If we have old and new versions of the file then run the appropriate
3234 # diffs. This is complicated by a couple of factors:
3235 #
3236 # - renames must be handled specially: we emit a 'remove'
3237 #   diff and an 'add' diff
3238 # - new files and deleted files must be handled specially
3239 # - Solaris patch(lm) can't cope with file creation
3240 #   (and hence renames) as of this writing.
3241 # - To make matters worse, gnu patch doesn't interpret the
3242 #   output of Solaris diff properly when it comes to
3243 #   adds and deletes. We need to do some "cleansing"
3244 #   transformations:
3245 #       [to add a file] @@ -1,0 +X,Y @@ --> @@ -0,0 +X,Y @@
3246 #       [to del a file] @@ -X,Y +1,0 @@ --> @@ -X,Y +0,0 @@
3247 #
3248 cleanse_rmfile="$SED 's/^\(@@ [0-9+,-]*\) [0-9+,-]* @@$/\1 +0,0 @@/'"
3249 cleanse_newfile="$SED 's/^@@ [0-9+,-]* \([0-9+,-]* @@\)/@@ -0,0 \1/'"

3251 rm -f $WDIR/$DIR/$F.patch
3252 if [[ -z $rename ]]; then
3253     if [ ! -f "$ofile" ]; then
3254         diff -u /dev/null $nfile | sh -c "$cleanse_newfile" \
3255             > $WDIR/$DIR/$F.patch
3256     elif [ ! -f "$nfile" ]; then
3257         diff -u $ofile /dev/null | sh -c "$cleanse_rmfile" \
3258             > $WDIR/$DIR/$F.patch
3259     else
3260         diff -u $ofile $nfile > $WDIR/$DIR/$F.patch
3261     fi
3262 else
3263     diff -u $ofile /dev/null | sh -c "$cleanse_rmfile" \
3264         > $WDIR/$DIR/$F.patch

3266     diff -u /dev/null $nfile | sh -c "$cleanse_newfile" \
3267         >> $WDIR/$DIR/$F.patch

3269 fi

3271 #
3272 # Tack the patch we just made onto the accumulated patch for the
3273 # whole wad.
3274 #
3275 cat $WDIR/$DIR/$F.patch >> $WDIR/$WNAME.patch

3277 print " patch\c"

```

```

3279 if [[ -f $ofile && -f $nfile && -z $mv_but_nodiff ]]; then

3281     ${CDIFFCMD:-diff -bt -C 5} $ofile $nfile > $WDIR/$DIR/$F.cdifff
3282     diff_to_html $F $DIR/$F "C" "$COMM" < $WDIR/$DIR/$F.cdifff \
3283         > $WDIR/$DIR/$F.cdifff.html
3284     print " cdifff\c"

3286     ${UDIFFCMD:-diff -bt -U 5} $ofile $nfile > $WDIR/$DIR/$F.udifff
3287     diff_to_html $F $DIR/$F "U" "$COMM" < $WDIR/$DIR/$F.udifff \
3288         > $WDIR/$DIR/$F.udifff.html

3290     print " udifff\c"

3292     if [[ -x $WDIFF ]]; then
3293         $WDIFF -c "$COMM" \
3294             -t "$WNAME wdifff $DIR/$F" $ofile $nfile > \
3295             $WDIR/$DIR/$F.wdifff.html 2>/dev/null
3296         if [[ $? -eq 0 ]]; then
3297             print " wdifff\c"
3298         else
3299             print " wdiffs[fail]\c"
3300         fi
3301     fi

3303     sdifff_to_html $ofile $nfile $F $DIR "$COMM" \
3304         > $WDIR/$DIR/$F.sdifff.html
3305     print " sdifff\c"

3307     print " frames\c"

3309     rm -f $WDIR/$DIR/$F.cdifff $WDIR/$DIR/$F.udifff

3311     difflines $ofile $nfile > $WDIR/$DIR/$F.count

3313 elif [[ -f $ofile && -f $nfile && -n $mv_but_nodiff ]]; then
3314     # renamed file: may also have differences
3315     difflines $ofile $nfile > $WDIR/$DIR/$F.count
3316 elif [[ -f $nfile ]]; then
3317     # new file: count added lines
3318     difflines /dev/null $nfile > $WDIR/$DIR/$F.count
3319 elif [[ -f $ofile ]]; then
3320     # old file: count deleted lines
3321     difflines $ofile /dev/null > $WDIR/$DIR/$F.count
3322 fi

3324 #
3325 # Now we generate the postscript for this file. We generate diffs
3326 # only in the event that there is delta, or the file is new (it seems
3327 # tree-killing to print out the contents of deleted files).
3328 #
3329 if [[ -f $nfile ]]; then
3330     ocr=$ofile
3331     [[ ! -f $ofile ]] && ocr=/dev/null

3333     if [[ -z $mv_but_nodiff ]]; then
3334         textcomm='getcomments text $P $PP'
3335         if [[ -x $CODEREVIEW ]]; then
3336             $CODEREVIEW -y "$textcomm" \
3337                 -e $ocr $nfile \
3338                 > /tmp/$$.psfile 2>/dev/null &&
3339                 cat /tmp/$$.psfile >> $WDIR/$WNAME.ps
3340         if [[ $? -eq 0 ]]; then
3341             print " ps\c"
3342         else
3343             print " ps[fail]\c"
3344         fi

```

```

3345         fi
3346     fi
3347 fi
3349 if [[ -f $ofile ]]; then
3350     source_to_html Old $PP < $ofile > $WDIR/$DIR/$F-.html
3351     print " old\c"
3352 fi
3354 if [[ -f $nfile ]]; then
3355     source_to_html New $P < $nfile > $WDIR/$DIR/$F.html
3356     print " new\c"
3357 fi
3359 cd $OWD
3361 print
3362 done
3364 frame_nav_js > $WDIR/ancnav.js
3365 frame_navigation > $WDIR/ancnav.html
3367 if [[ ! -f $WDIR/$WNAME.ps ]]; then
3368     print " Generating PDF: Skipped: no output available"
3369 elif [[ -x $CODEREVIEW && -x $PS2PDF ]]; then
3370     print " Generating PDF: \c"
3371     fix_postscript $WDIR/$WNAME.ps | $PS2PDF - > $WDIR/$WNAME.pdf
3372     print "Done."
3373 else
3374     print " Generating PDF: Skipped: missing 'ps2pdf' or 'codereview'"
3375 fi
3377 # If we're in OpenSolaris mode and there's a closed dir under $WDIR,
3378 # delete it - prevent accidental publishing of closed source
3380 if [[ -n "$Oflag" ]]; then
3381     $FIND $WDIR -type d -name closed -exec /bin/rm -rf {} \;
3382 fi
3384 # Now build the index.html file that contains
3385 # links to the source files and their diffs.
3387 cd $CWS
3389 # Save total changed lines for Code Inspection.
3390 print "$TOTL" > $WDIR/TotalChangedLines
3392 print "    index.html: \c"
3393 INDEXFILE=$WDIR/index.html
3394 exec 3<&1          # duplicate stdout to FD3.
3395 exec 1<&-         # Close stdout.
3396 exec > $INDEXFILE # Open stdout to index file.
3398 print "$HTML<head>$STDHEAD"
3399 print "<title>$WNAME</title>"
3400 print "</head>"
3401 print "<body id=\"SUNWwebrev\">"
3402 print "<div class=\"summary\">"
3403 print "<h2>Code Review for $WNAME</h2>"
3405 print "<table>"
3407 #
3408 # Get the preparer's name:
3409 #
3410 # If the SCM detected is Mercurial, and the configuration property

```

```

3411 # ui.username is available, use that, but be careful to properly escape
3412 # angle brackets (HTML syntax characters) in the email address.
3413 #
3414 # Otherwise, use the current userid in the form "John Doe (jdoe)", but
3415 # to maintain compatibility with passwd(4), we must support '&' substitutions.
3416 #
3417 preparer=
3418 if [[ "$SCM_MODE" == mercurial ]]; then
3419     preparer='hg showconfig ui.username 2>/dev/null'
3420     if [[ -n "$preparer" ]]; then
3421         preparer="$(echo "$preparer" | html_quote)"
3422     fi
3423 fi
3424 if [[ -z "$preparer" ]]; then
3425     preparer=$(
3426         $PERL -e '
3427             ($login, $pw, $uid, $gid, $quota, $cmt, $gcos) = getpwuid($<);
3428             if ($login) {
3429                 $gcos =~ s/\&/ucfirst($login)/e;
3430                 printf "%s (%s)\n", $gcos, $login;
3431             } else {
3432                 printf "(unknown)\n";
3433             }
3434         '
3435 fi
3437 print "<tr><th>Prepared by:</th><td>$preparer on 'date'</td></tr>"
3438 print "<tr><th>Workspace:</th><td>$CWS"
3439 if [[ -n $CWS_REV ]]; then
3440     print "(at $CWS_REV)"
3441 fi
3442 print "</td></tr>"
3443 print "<tr><th>Compare against:</th><td>"
3444 if [[ -n $parent_webrev ]]; then
3445     print "webrev at $parent_webrev"
3446 else
3447     print "$PWS"
3448     if [[ -n $hg_parent_short ]]; then
3449         print "(at $hg_parent_short)"
3450     fi
3451 fi
3452 print "</td></tr>"
3453 print "<tr><th>Summary of changes:</th><td>"
3454 printCI $TOTL $TINS $TDEL $TMOD $TUNC
3455 print "</td></tr>"
3457 if [[ -f $WDIR/$WNAME.patch ]]; then
3458     wpatch_url="$(print $WNAME.patch | url_encode)"
3459     print "<tr><th>Patch of changes:</th><td>"
3460     print "<a href=\"\$wpatch_url\">$WNAME.patch</a></td></tr>"
3461 fi
3462 if [[ -f $WDIR/$WNAME.pdf ]]; then
3463     wpdf_url="$(print $WNAME.pdf | url_encode)"
3464     print "<tr><th>Printable review:</th><td>"
3465     print "<a href=\"\$wpdf_url\">$WNAME.pdf</a></td></tr>"
3466 fi
3468 if [[ -n "$iflag" ]]; then
3469     print "<tr><th>Author comments:</th><td><div>"
3470     cat /tmp/$$.include
3471     print "</div></td></tr>"
3472 fi
3473 print "</table>"
3474 print "</div>"
3476 #

```

```

3477 # Second pass through the files: generate the rest of the index file
3478 #
3479 cat $FLIST | while read LINE
3480 do
3481     set - $LINE
3482     P=$1
3483
3484     if [[ $# == 2 ]]; then
3485         PP=$2
3486         oldname="$PP"
3487     else
3488         PP=$P
3489         oldname=""
3490     fi
3491
3492     mv_but_nodiff=
3493     cmp $WDIR/raw_files/old/$PP $WDIR/raw_files/new/$P > /dev/null 2>&1
3494     if [[ $? == 0 && -n "$oldname" ]]; then
3495         mv_but_nodiff=1
3496     fi
3497
3498     DIR=${P%/*}
3499     if [[ $DIR == $P ]]; then
3500         DIR="." # File at root of workspace
3501     fi
3502
3503     # Avoid processing the same file twice.
3504     # It's possible for renamed files to
3505     # appear twice in the file list
3506
3507     F=$WDIR/$P
3508
3509     print "<p>"
3510
3511     # If there's a diffs file, make diffs links
3512
3513     if [[ -f $F.cdifff.html ]]; then
3514         cdifff_url="$(print $P.cdifff.html | url_encode)"
3515         udiffff_url="$(print $P.udifff.html | url_encode)"
3516         print "<a href=\"$cdifff_url\">Cdifff</a>"
3517         print "<a href=\"$udiffff_url\">Udiffff</a>"
3518
3519         if [[ -f $F.wdifff.html && -x $WDIFF ]]; then
3520             wdiffff_url="$(print $P.wdifff.html | url_encode)"
3521             print "<a href=\"$wdiffff_url\">Wdifff</a>"
3522         fi
3523
3524         sdifff_url="$(print $P.sdifff.html | url_encode)"
3525         print "<a href=\"$sdifff_url\">Sdifff</a>"
3526
3527         frames_url="$(print $P.frames.html | url_encode)"
3528         print "<a href=\"$frames_url\">Frames</a>"
3529     else
3530         print "-----"
3531
3532         if [[ -x $WDIFF ]]; then
3533             print "-----"
3534         fi
3535
3536         print "-----"
3537     fi
3538
3539     # If there's an old file, make the link
3540
3541     if [[ -f $F-.html ]]; then
3542         oldfile_url="$(print $F-.html | url_encode)"

```

```

3543         print "<a href=\"$oldfile_url\">Old</a>"
3544     else
3545         print "----"
3546     fi
3547
3548     # If there's a new file, make the link
3549
3550     if [[ -f $F.html ]]; then
3551         newfile_url="$(print $F.html | url_encode)"
3552         print "<a href=\"$newfile_url\">New</a>"
3553     else
3554         print "----"
3555     fi
3556
3557     if [[ -f $F.patch ]]; then
3558         patch_url="$(print $F.patch | url_encode)"
3559         print "<a href=\"$patch_url\">Patch</a>"
3560     else
3561         print "-----"
3562     fi
3563
3564     if [[ -f $WDIR/raw_files/new/$P ]]; then
3565         rawfiles_url="$(print raw_files/new/$P | url_encode)"
3566         print "<a href=\"$rawfiles_url\">Raw</a>"
3567     else
3568         print "----"
3569     fi
3570
3571     print "<b>$P</b>"
3572
3573     # For renamed files, clearly state whether or not they are modified
3574     if [[ -n "$oldname" ]]; then
3575         if [[ -n "$mv_but_nodiff" ]]; then
3576             print "<i>(renamed only, was $oldname)</i>"
3577         else
3578             print "<i>(modified and renamed, was $oldname)</i>"
3579         fi
3580     fi
3581
3582     # If there's an old file, but no new file, the file was deleted
3583     if [[ -f $F-.html && ! -f $F.html ]]; then
3584         print "<i>(deleted)</i>"
3585     fi
3586
3587     #
3588     # Check for usr/closed and deleted_files/usr/closed
3589     #
3590     if [ ! -z "$Oflag" ]; then
3591         if [[ $P == usr/closed/* || \
3592             $P == deleted_files/usr/closed/* ]]; then
3593             print "&nbsp;&nbsp;&nbsp;<i>Closed source: omitted from \
3594                 \"this review</i>"
3595         fi
3596     fi
3597
3598     print "</p>"
3599     # Insert delta comments
3600
3601     print "<blockquote><pre>"
3602     getcomments html $P $PP
3603     print "</pre>"
3604
3605     # Add additional comments comment
3606
3607     print "<!-- Add comments to explain changes in $P here -->"

```

```

3609      # Add count of changes.
3611      if [[ -f $F.count ]]; then
3612          cat $F.count
3613          rm $F.count
3614      fi
3616      if [[ $SCM_MODE == "teamware" ||
3617           $SCM_MODE == "mercurial" ||
3618           $SCM_MODE == "unknown" ]]; then
3620          # Include warnings for important file mode situations:
3621          # 1) New executable files
3622          # 2) Permission changes of any kind
3623          # 3) Existing executable files
3625          old_mode=
3626          if [[ -f $WDIR/raw_files/old/$PP ]]; then
3627              old_mode=`get_file_mode $WDIR/raw_files/old/$PP`
3628          fi
3630          new_mode=
3631          if [[ -f $WDIR/raw_files/new/$P ]]; then
3632              new_mode=`get_file_mode $WDIR/raw_files/new/$P`
3633          fi
3635          if [[ -z "$old_mode" && "$new_mode" = *[1357]* ]]; then
3636              print "<span class=\"chmod\">"
3637              print "<p>new executable file: mode $new_mode</p>"
3638              print "</span>"
3639          elif [[ -n "$old_mode" && -n "$new_mode" &&
3640                "$old_mode" != "$new_mode" ]]; then
3641              print "<span class=\"chmod\">"
3642              print "<p>mode change: $old_mode to $new_mode</p>"
3643              print "</span>"
3644          elif [[ "$new_mode" = *[1357]* ]]; then
3645              print "<span class=\"chmod\">"
3646              print "<p>executable file: mode $new_mode</p>"
3647              print "</span>"
3648          fi
3649      fi
3651      print "</blockquote>"
3652  done
3654  print
3655  print
3656  print "<hr></hr>"
3657  print "<p style=\"font-size: small\">"
3658  print "This code review page was prepared using <b>$0</b>."
3659  print "Webrev is maintained by the <a href=\"http://www.opensolaris.org\">"
3660  print "OpenSolaris</a> project. The latest version may be obtained"
3661  print "<a href=\"http://src.opensolaris.org/source/xref/onnv/onnv-gate/usr/src/t"
3662  print "</body>"
3663  print "</html>"
3665  exec 1<&-          # Close FD 1.
3666  exec 1<&3         # dup FD 3 to restore stdout.
3667  exec 3<&-        # close FD 3.
3669  print "Done."
3671  #
3672  # If remote deletion was specified and fails do not continue.
3673  #
3674  if [[ -n $Dflag ]]; then

```

```

3675          delete_webrev 1 1
3676          (( $? == 0 )) || exit $?
3677      fi
3679  if [[ -n $Uflag ]]; then
3680      upload_webrev
3681      exit $?
3682  fi

```

new/usr/src/tools/findunref/exception\_list.git

1

\*\*\*\*\*

1155 Sat May 9 16:08:44 2009

new/usr/src/tools/findunref/exception\_list.git

Add exception\_list for git

\*\*\*\*\*

```
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 #
21 #
22 #
23 # Copyright 2008 Sun Microsystems, Inc. All rights reserved.
24 # Use is subject to license terms.
25 #
26 #
27 #
28 # Git-specific exception list
29 #
30 # See README.exception_lists for details
31 #
32 #
33 #
34 # Without nested repositories, this list could be empty, because ON
35 # checks for unref relative to usr, and the git files are all in the
36 # root of the repository.
37 #
38 #
39 */.git
```

```

*****
3406 Sat May 9 16:08:44 2009
new/usr/src/tools/scripts/git-active.py
Add git-active utility
*****
1 #!/usr/bin/python
2 #
3 # This program is free software; you can redistribute it and/or modify
4 # it under the terms of the GNU General Public License version 2
5 # as published by the Free Software Foundation.
6 #
7 # This program is distributed in the hope that it will be useful,
8 # but WITHOUT ANY WARRANTY; without even the implied warranty of
9 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
10 # GNU General Public License for more details.
11 #
12 # You should have received a copy of the GNU General Public License
13 # along with this program; if not, write to the Free Software
14 # Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
15 #
17 #
18 # Copyright 2009 Grigale, Inc. All rights reserved.
19 # Use is subject to license terms.
20 #
22 '''
23 Create a wx-style active list on stdout based on a Git
24 workspace in support of webrev's Git support.
25 '''
27 #
28 # NB: This assumes the normal onbld directory structure
29 #
30 import os
31 import sys
32 import optparse
33 import subprocess
35 def execCmd(cmd):
36     '''Executes external command'''
38     p = subprocess.Popen(cmd, stdin=subprocess.PIPE, stdout=subprocess.PIPE,
39                          stderr=subprocess.PIPE)
40     (out, err) = p.communicate()
42     if out == None:
43         out = ""
45     return (p.returncode, out.splitlines(), err.splitlines())
47 def usage():
48     sys.stderr.write("usage: %s [-p parent] -w workspace\n" %
49                    os.path.basename(__file__))
50     sys.exit(2)
52 def main(argv):
54     parser = optparse.OptionParser(version='prog 1.0')
55     parser.add_option("-p", dest="parentpath", default="", help="parent repo")
56     parser.add_option("-w", dest="wspath", default="", help="workspace")
57     parser.add_option("-o", dest="outputfile", help="output file")
58     parser.disable_interspersed_args()
60     (options, args) = parser.parse_args()

```

```

62     if not options.wspath:
63         usage()
65     fh = None
66     if options.outputfile:
67         try:
68             fh = open(options.outputfile, 'w')
69         except EnvironmentError, e:
70             sys.stderr.write("could not open output file: %s\n" % e)
71             sys.exit(1)
72     else:
73         fh = sys.stdout
75     cmd = ["git", "--git-dir=%s" % options.wspath, "log", "--name-only",
76           "--parents", "--reverse", "--pretty=short", "master.."]
77     (rc, out, err) = execCmd(cmd)
79     #print rc, out, err
81     if "commit" in out[0]:
82         parent = out[0].split()[2]
84     files = {}
85     comment = None
86     for i in out:
87         if "commit" in i:
88             comment = None
89             continue
90         if i == "" or i.startswith("Author"):
91             continue
92         if i.startswith(" "):
93             comment = i.strip()
94             continue
95         if comment:
96             if i not in files:
97                 files[i] = []
98             files[i].append(comment)
100     fh.write("GIT_PARENT=%s\n" % parent)
102     for file in files:
103         #if entry.is_renamed():
104             # fh.write("%s %s\n" % (entry.name, entry.parentname))
105         #else:
106             # fh.write("%s\n" % entry.name)
107             fh.write("%s\n\n" % file)
108             fh.write("%s\n\n" % '\n'.join(files[file]))
110 #try:
111 #     Version.check_version()
112 #except Version.VersionMismatch, e:
113 #     sys.stderr.write("Error: %s\n" % e)
114 #     sys.exit(1)
116 if __name__ == '__main__':
117     try:
118         main(sys.argv[1:])
119     except KeyboardInterrupt:
120         sys.exit(1)
121 #     except util.Abort, msg:
122 #         sys.stderr.write("Abort: %s\n" % msg)
123 #         sys.exit(1)

```

new/usr/src/tools/SUNWonbld/prototype\_com

1

\*\*\*\*\*

9602 Sat May 9 16:08:45 2009

new/usr/src/tools/SUNWonbld/prototype\_com

Add git-active utility

\*\*\*\*\*

```
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 #
21 #
22 # Copyright 2009 Sun Microsystems, Inc. All rights reserved.
23 # Use is subject to license terms.
24 #
25 #
26 #
27 # This required package information file contains a list of package contents.
28 # The 'pkgmk' command uses this file to identify the contents of a package
29 # and their location on the development machine when building the package.
30 # Can be created via a text editor or through use of the 'pkgproto' command.
31 #
32 #
33 #!search <pathname pathname ...> # where to find pkg objects
34 #!include <filename> # include another 'prototype' file
35 #!default <mode> <owner> <group> # default used if not specified on entry
36 #!<param>=<value> # puts parameter in pkg environment
37 #
38 # packaging files
39 i depend
40 i pkginfo
41 i copyright
42 i postinstall
43 i preremove
44 #
45 # source locations relative to the prototype file
46 #
47 # SUNWonbld
48 #
49 d none opt 755 root sys
50 d none opt/onbld 755 root bin
51 d none opt/onbld/bin 755 root bin
52 f none opt/onbld/bin/Install 555 root bin
53 f none opt/onbld/bin/acr 555 root bin
54 f none opt/onbld/bin/bfu 555 root bin
55 f none opt/onbld/bin/bfudrop 555 root bin
56 f none opt/onbld/bin/bldenv 555 root bin
57 f none opt/onbld/bin/bringovercheck 555 root bin
58 f none opt/onbld/bin/build_scope 555 root bin
59 f none opt/onbld/bin/cddlchk 555 root bin
60 f none opt/onbld/bin/check_rtime 555 root bin
61 f none opt/onbld/bin/checkpaths 555 root bin
```

new/usr/src/tools/SUNWonbld/prototype\_com

2

```
62 f none opt/onbld/bin/checkproto 555 root bin
63 f none opt/onbld/bin/copyrightchk 555 root bin
64 f none opt/onbld/bin/cstyle 555 root bin
65 f none opt/onbld/bin/ctffindmod 555 root bin
66 f none opt/onbld/bin/ctfcvtptbl 555 root bin
67 f none opt/onbld/bin/bindrop 555 root bin
68 f none opt/onbld/bin/elfcmp 555 root bin
69 f none opt/onbld/bin/elfsigncmp 555 root bin
70 f none opt/onbld/bin/flg.flp 555 root bin
71 f none opt/onbld/bin/genoffsets 555 root bin
72 f none opt/onbld/bin/get_depend_info 555 root bin
73 f none opt/onbld/bin/git-active 555 root bin
74 f none opt/onbld/bin/hdrchk 555 root bin
75 f none opt/onbld/bin/hg-active 555 root bin
76 f none opt/onbld/bin/hgsetup 555 root bin
77 f none opt/onbld/bin/intf_check 555 root bin
78 f none opt/onbld/bin/jstyle 555 root bin
79 f none opt/onbld/bin/keywords 555 root bin
80 f none opt/onbld/bin/makebfu 555 root bin
81 f none opt/onbld/bin/make_pkg_db 555 root bin
82 f none opt/onbld/bin/mapfilechk 555 root bin
83 f none opt/onbld/bin/mkacr 555 root bin
84 f none opt/onbld/bin/mkbfu 555 root bin
85 f none opt/onbld/bin/mkclosed 555 root bin
86 f none opt/onbld/bin/mkreadme_osol 555 root bin
87 f none opt/onbld/bin/mktpl 555 root bin
88 f none opt/onbld/bin/nightly 555 root bin
89 f none opt/onbld/bin/onblddrop 555 root bin
90 f none opt/onbld/bin/protocmp terse 555 root bin
91 f none opt/onbld/bin/rtichk 555 root bin
92 f none opt/onbld/bin/scscscheck 555 root bin
93 f none opt/onbld/bin/scscsccp 555 root bin
94 f none opt/onbld/bin/scscshist 555 root bin
95 f none opt/onbld/bin/scscsmv 555 root bin
96 f none opt/onbld/bin/scscsrm 555 root bin
97 f none opt/onbld/bin/sdrop 555 root bin
98 f none opt/onbld/bin/signit 555 root bin
99 f none opt/onbld/bin/signproto 555 root bin
100 f none opt/onbld/bin/validate_flg 555 root bin
101 f none opt/onbld/bin/validate_paths 555 root bin
102 f none opt/onbld/bin/wdiff 555 root bin
103 f none opt/onbld/bin/webrev 555 root bin
104 f none opt/onbld/bin/which_scm 555 root bin
105 f none opt/onbld/bin/ws 555 root bin
106 f none opt/onbld/bin/wsdiff 555 root bin
107 f none opt/onbld/bin/wx 555 root bin
108 f none opt/onbld/bin/wx2hg 555 root bin
109 f none opt/onbld/bin/xref 555 root bin
110 f none opt/onbld/bin/xref.mk 644 root bin
111 d none opt/onbld/lib 755 root bin
112 d none opt/onbld/lib/python 755 root bin
113 d none opt/onbld/lib/python/onbld 755 root bin
114 f none opt/onbld/lib/python/onbld/__init__.py 444 root bin
115 f none opt/onbld/lib/python/onbld/__init__.pyc 444 root bin
116 d none opt/onbld/lib/python/onbld/Checks 755 root bin
117 f none opt/onbld/lib/python/onbld/Checks/__init__.py 444 root bin
118 f none opt/onbld/lib/python/onbld/Checks/__init__.pyc 444 root bin
119 f none opt/onbld/lib/python/onbld/Checks/CStyle.py 444 root bin
120 f none opt/onbld/lib/python/onbld/Checks/CStyle.pyc 444 root bin
121 f none opt/onbld/lib/python/onbld/Checks/Cddl.py 444 root bin
122 f none opt/onbld/lib/python/onbld/Checks/Cddl.pyc 444 root bin
123 f none opt/onbld/lib/python/onbld/Checks/CmtBlk.py 444 root bin
124 f none opt/onbld/lib/python/onbld/Checks/CmtBlk.pyc 444 root bin
125 f none opt/onbld/lib/python/onbld/Checks/Comments.py 444 root bin
126 f none opt/onbld/lib/python/onbld/Checks/Comments.pyc 444 root bin
127 f none opt/onbld/lib/python/onbld/Checks/Copyright.py 444 root bin
```

```

128 f none opt/onbld/lib/python/onbld/Checks/Copyright.pyc 444 root bin
129 f none opt/onbld/lib/python/onbld/Checks/DbLookups.py 444 root bin
130 f none opt/onbld/lib/python/onbld/Checks/DbLookups.pyc 444 root bin
131 f none opt/onbld/lib/python/onbld/Checks/HdrChk.py 444 root bin
132 f none opt/onbld/lib/python/onbld/Checks/HdrChk.pyc 444 root bin
133 f none opt/onbld/lib/python/onbld/Checks/JStyle.py 444 root bin
134 f none opt/onbld/lib/python/onbld/Checks/JStyle.pyc 444 root bin
135 f none opt/onbld/lib/python/onbld/Checks/Keywords.py 444 root bin
136 f none opt/onbld/lib/python/onbld/Checks/Keywords.pyc 444 root bin
137 f none opt/onbld/lib/python/onbld/Checks/Mapfile.py 444 root bin
138 f none opt/onbld/lib/python/onbld/Checks/Mapfile.pyc 444 root bin
139 f none opt/onbld/lib/python/onbld/Checks/ProcessCheck.py 444 root bin
140 f none opt/onbld/lib/python/onbld/Checks/ProcessCheck.pyc 444 root bin
141 f none opt/onbld/lib/python/onbld/Checks/Rti.py 444 root bin
142 f none opt/onbld/lib/python/onbld/Checks/Rti.pyc 444 root bin
143 d none opt/onbld/lib/python/onbld/hgext 755 root bin
144 f none opt/onbld/lib/python/onbld/hgext/___init___py 444 root bin
145 f none opt/onbld/lib/python/onbld/hgext/___init___pyc 444 root bin
146 f none opt/onbld/lib/python/onbld/hgext/cdm.py 444 root bin
147 f none opt/onbld/lib/python/onbld/hgext/cdm.pyc 444 root bin
148 d none opt/onbld/lib/python/onbld/Scm 755 root bin
149 f none opt/onbld/lib/python/onbld/Scm/___init___py 444 root bin
150 f none opt/onbld/lib/python/onbld/Scm/___init___pyc 444 root bin
151 f none opt/onbld/lib/python/onbld/Scm/Backup.py 444 root bin
152 f none opt/onbld/lib/python/onbld/Scm/Backup.pyc 444 root bin
153 f none opt/onbld/lib/python/onbld/Scm/Version.py 444 root bin
154 f none opt/onbld/lib/python/onbld/Scm/Version.pyc 444 root bin
155 f none opt/onbld/lib/python/onbld/Scm/WorkSpace.py 444 root bin
156 f none opt/onbld/lib/python/onbld/Scm/WorkSpace.pyc 444 root bin
157 d none opt/onbld/env 755 root bin
158 f none opt/onbld/env/developer 644 root bin
159 f none opt/onbld/env/gatekeeper 644 root bin
160 f none opt/onbld/env/opensolaris 644 root bin
161 d none opt/onbld/etc 755 root bin
162 f none opt/onbld/etc/SampleLinks 644 root bin
163 f none opt/onbld/etc/SamplePkgLinks 644 root bin
164 d none opt/onbld/etc/abi 755 root bin
165 f none opt/onbld/etc/abi/exceptions 444 root bin
166 f none opt/onbld/etc/abi/ABI_i386.db 444 root bin
167 f none opt/onbld/etc/abi/ABI_sparc.db 444 root bin
168 f none opt/onbld/etc/hgstyle 644 root bin
169 f none opt/onbld/etc/its.conf 644 root bin
170 f none opt/onbld/etc/its.reg 644 root bin
171 d none opt/onbld/gk 755 root bin
172 f none opt/onbld/gk/.cshrc 644 root bin
173 f none opt/onbld/gk/.login 644 root bin
174 d none opt/onbld/man 755 root bin
175 d none opt/onbld/man/man1 755 root bin
176 f none opt/onbld/man/man1/Install.1 644 root bin
177 f none opt/onbld/man/man1/acr.1 644 root bin
178 f none opt/onbld/man/man1/bldenv.1 644 root bin
179 f none opt/onbld/man/man1/bringovercheck.1 644 root bin
180 f none opt/onbld/man/man1/cddlchk.1 644 root bin
181 f none opt/onbld/man/man1/check_rtime.1 644 root bin
182 f none opt/onbld/man/man1/checkpaths.1 644 root bin
183 f none opt/onbld/man/man1/codereview.1 644 root bin
184 f none opt/onbld/man/man1/cstyle.1 644 root bin
185 f none opt/onbld/man/man1/cw.1 644 root bin
186 f none opt/onbld/man/man1/flg.flp.1 644 root bin
187 f none opt/onbld/man/man1/get_depend_info.1 644 root bin
188 f none opt/onbld/man/man1/intf_check.1 644 root bin
189 f none opt/onbld/man/man1/hdrchk.1 644 root bin
190 f none opt/onbld/man/man1/hgsetup.1 644 root bin
191 f none opt/onbld/man/man1/jstyle.1 644 root bin
192 f none opt/onbld/man/man1/lintdump.1 644 root bin
193 f none opt/onbld/man/man1/make_pkg_db.1 644 root bin

```

```

194 f none opt/onbld/man/man1/mapfilechk.1 644 root bin
195 f none opt/onbld/man/man1/mkacr.1 644 root bin
196 f none opt/onbld/man/man1/ndrgen.1 644 root bin
197 f none opt/onbld/man/man1/nightly.1 644 root bin
198 f none opt/onbld/man/man1/sccscheck.1 644 root bin
199 f none opt/onbld/man/man1/sccscp.1 644 root bin
200 f none opt/onbld/man/man1/sccsmv.1 644 root bin
201 f none opt/onbld/man/man1/sccsrm.1 644 root bin
202 f none opt/onbld/man/man1/signit.1 644 root bin
203 f none opt/onbld/man/man1/signproto.1 644 root bin
204 f none opt/onbld/man/man1/webrev.1 644 root bin
205 f none opt/onbld/man/man1/which_scm.1 644 root bin
206 f none opt/onbld/man/man1/ws.1 644 root bin
207 f none opt/onbld/man/man1/wsdiff.1 644 root bin
208 f none opt/onbld/man/man1/wx.1 644 root bin
209 f none opt/onbld/man/man1/wx2hg.1 644 root bin
210 f none opt/onbld/man/man1/xref.1 644 root bin

```

\*\*\*\*\*

12866 Sat May 9 16:08:45 2009

new/usr/src/tools/README.tools

Add git-active utility

\*\*\*\*\*

```

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 #
21 #
22 # Copyright 2008 Sun Microsystems, Inc. All rights reserved.
23 # Use is subject to license terms.
24 #
25 #ident "%Z%M% %I% %E% SMI"

27 This directory contains the tools used to do a full build of the
28 OS/Net workspace. They usually live in the /opt/onbld directory on build
29 machines. From here, 'make install' will build and install the tools
30 in $ROOT/opt/onbld. If you like, 'make pkg' will build the SUNWonbld
31 package in $(PKGARCHIVE). Installing that package will populate the
32 /opt/onbld directory, and create a root account for building called 'gk',
33 which uses csh and has a home directory of /opt/onbld/gk. You can
34 use this account to do full builds with 'nightly'. You don't have to,
35 but the 'gk' account has the path setup properly, has a .make.machines
36 file for dmake, and has a .login that sets up for dmake.

38 Layout of /opt/onbld
39 -----

41 /opt/onbld/etc/abi
42 contains Solaris ABI database (ABI_*.db) and exceptions
43 for ABI Auditing tool (intf_check).

45 /opt/onbld/gk
46 gk account's home directory.

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

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

54 /opt/onbld/env
55 build environment files.

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

60 /opt/onbld/lib/python
61 python modules used by the build tools.

```

```

63 /opt/onbld/lib/python/onbld/hgext
64 Mercurial extensions.

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

70 Tool Summary
71 -----

73 bfu
74 bonwick/faulkner upgrade. Loads a set of cpio archives created
75 by 'mkbfu' onto a machine, either live or on alternate root
76 and /usr filesystems. Attempts to preserve important files,
77 but may require manual intervention before reboot to resolve
78 changes to preserved files.

80 bfuld
81 Used by bfu to survive getting a new runtime linker when extracting
82 new cpio archives onto a live system. Patches binaries to use
83 a saved runtime linker in /tmp during the bfu process.
84 Not run by anything but bfu.

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

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

99 cdm
100 A Mercurial extension providing various commands useful for ON
101 development

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

112 checkproto
113 Runs protocmp and protolist on a workspace (or uses the environment
114 variable CODEMGR_WS to determine the workspace). Checks the proto area
115 against the packages.

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

121 codesign
122 Tools for signing cryptographic modules using the official
123 Sun release keys stored on a remote signing server. This
124 directory contains signit, a client program for signing
125 files with the signing server; signproto, a shell script
126 that finds crypto modules in $ROOT and signs them using
127 signit; and codesign_server.pl, the code that runs on the

```

```

128     server. The codesign_server code is not used on an ON
129     build machine but is kept here for source control purposes.

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

135 cscope-fast
136 The fast version of cscope that we use internally. Seems to work,
137 but may need more testing before it's placed in the gate. The source
138 just really needs to be here.
139
140 cstyle
141 checks C source for compliance with OS/Net guidelines.

143 ctfcconvert
144 Convert symbolic debugging information in an object file to the Compact
145 ANSI-C Type Format (CTF).

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

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

153 depcheck
154 A tool to try an assess the dependencies of executables. This tool
155 is not a definitive dependency check, but it does use "strings" and
156 "ldd" to gather as much information as it can. The dependency check
157 tool can handle filenames and pkgnames. Before using the dependency
158 checker you must build a database which reflects the properties and
159 files in your system.

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

168 elfsign
169 Built from the same sources as the shipped elfsign(1), this
170 version is used in nightly -t builds to assure that the signing
171 process and format is the same as will be used on the target
172 system.

174 elfsigncmp
175 This script can be used in lieu of elfsign during a build.
176 It uses elfsign to sign a copy of the object and elfcmp -S to
177 verify that the signing caused no damage before updating
178 the object to be signed.
179
180 findunref
181 Finds all files in a source tree that have access times older than a
182 certain time and are not in a specified list of exceptions. Since
183 'nightly' timestamps the start of the build, and findunref uses its
184 timestamp (by default), this can be used to find all files that were
185 unreferenced during a nightly build). Since some files are only used
186 during a SPARC or Intel build, 'findunref' needs to be run on
187 workspaces from both architectures and the results need to be merged.
188 For instance, if $INTELSRC and $SPARCSRC are set to the usr/src
189 directories of your Intel and SPARC nightly workspaces, then you
190 can merge the results like so:

192 $ findunref $INTELSRC $INTELSRC/tools/findunref/exception_list | \
193 sort > ~/unref-i386.out

```

```

194 $ findunref $SPARCSRC $SPARCSRC/tools/findunref/exception_list | \
195 sort > ~/unref-sparc.out
196 $ comm -12 ~/unref-i386.out ~/unref-sparc.out > ~/unref.out

198 git-active
199 helper used by webrev to generate file lists for Git workspaces.

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

205 hgsetup
206 creates a basic Mercurial configuration for the user.

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

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

218 intf_check
219 detects and reports ABI versioning and stability problems.

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

224 keywords
225 checks files for proper SCCS keywords.

227 makebfu
228 simple wrapper around 'mkbfu' for use outside nightly (when in a build
229 shell from 'ws' or 'bldenv').

231 mkbfu
232 makes cpio archives out of the proto area suitable for bfu'ing.
233 Used by 'nightly' and 'makebfu'.

235 ndrngen
236 Network Data Language (NDL) RPC protocol compiler to support DCE
237 RPC/MSRPC and SMB/CIFS. ndrngen takes an input protocol definition
238 file (say, proto.ndl) and generates an output C source file
239 (proto_nldr.c) containing the Network Data Representation (NDR)
240 marshalling routines to implement the RPC protocol.

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

248 pmodes
249 enforces proper file ownership and permissions in pkgmap and package
250 prototype* files. converts files if necessary

252 protocmp
253 compares proto lists and the package definitions. Used by nightly
254 to determine if the proto area matches the packages, and to detect
255 differences between a child's proto area and a parent's.

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

```

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

263 rtichk  
 264 checks that a set of CRs have approved RTIs. Primarily used  
 265 by wx

267 sccscp  
 268 copy a file under SCCS control to another location in a workspace.  
 269 also updates teamware's nametable.

271 sccshist  
 272 Display the history, comments and diffs, of a file under SCCS  
 273 control.

275 sccsmv  
 276 rename a file under SCCS control to another location in a workspace.  
 277 also updates teamware's nametable.

279 sccsrm  
 280 delete a file under SCCS control workspace. also updates teamware's  
 281 nametable. Actually renames it to .del-<file>-'date' so that others  
 282 will see it move when it is brought over (in case they were working  
 283 on it).

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

291 wx  
 292 A great workspace tool by bonwick. See wx.README for information  
 293 and warnings.

295 wx2hg  
 296 Converts a TeamWare workspace under the control of wx to a  
 297 Mercurial workspace, discarding intermediate deltas.

299 tokenize  
 300 Used to build the sun4u boot block.

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

309 which\_scm  
 310 Reports the current Source Code Management (SCM) system in use  
 311 and the top-level directory of the workspace.

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

320 How to do a full build  
 321 -----

323 1. Find an environment file that might do what you want to do. If you're just  
 324 a developer wanting to do a full build in a child of the gate, copy the  
 325 'developer' environment file to a new name (private to you and/or the

326 work being done in this workspace, to avoid collisions with others). Then  
 327 edit the file and tailor it to your workspace. Remember that this file  
 328 is a shell script, so it can do more than set environment variables.

330 2. Login as 'gk' (or root, but your PATH and .make.machines for dmake will  
 331 not be right). Run 'nightly' and give it your environment file as an  
 332 option. 'nightly' will first look for your environment file in  
 333 /opt/onbld/env, and if it's not there then it will look for it as an  
 334 absolute or relative path. Some people put their environment files in  
 335 their workspace to keep them close.

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

346 Files you have to update to add a tool  
 347 -----

349 1. Add the tool in its appropriate place.  
 350 2. Update the Makefile as required.  
 351 3. Update usr/src/tools/SUNWonbld/prototype\_\*.  
 352 4. Update usr/src/tools/README.tools (this file).  
 353 5. Repeat 1-4 for any man pages.