

new/usr/src/tools/env/developer.sh

1

```
*****
7815 Fri May 22 11:27:12 2015
new/usr/src/tools/env/developer.sh
nightly: don't check dmake version, now it's ours
*****
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 (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
24 #
25 #
26 # Configuration variables for the runtime environment of the nightly
27 # build script and other tools for construction and packaging of releases.
28 # This script is sourced by 'nightly' and 'bldenv' to set up the environment
29 # for the build. This example is suitable for building a developers workspace,
30 # which will contain the resulting packages and archives. It is based off
31 # the onnv release. It sets NIGHTLY_OPTIONS to make nightly do:
32 # check ELF ABI/versioning (-A)
33 # runs 'make check' (-C)
34 # DEBUG and non-DEBUG builds (-D)
35 # runs lint in usr/src (-l plus the LINTDIRS variable)
36 # sends mail on completion (-m and the MAILTO variable)
37 # creates packages for PIT/RE (-p)
38 # checks for changes in ELF runpaths (-r)
39 #
40 NIGHTLY_OPTIONS="-ACDlmp"; export NIGHTLY_OPTIONS
41 #
42 # This is a variable for the rest of the script - GATE doesn't matter to
43 # nightly itself
44 GATE=onnv-bugfixes; export GATE
45 #
46 # CODEMGR_WS - where is your workspace at (or what should nightly name it)
47 CODEMGR_WS="/builds/$GATE"; export CODEMGR_WS
48 #
49 # PARENT_WS is used to determine the parent of this workspace. This is
50 # for the options that deal with the parent workspace (such as where the
51 # proto area will go).
52 #
53 # If you use this, it must be local (or nfs): nightly cannot copy
54 # over ssh or http.
55 PARENT_WS="/ws/onnv-gate"; export PARENT_WS
56 #
57 # CLONE_WS is the workspace nightly should do a bringover from.
58 CLONE_WS="ssh://anonhg@onnv.sfbay.sun.com/~/export/onnv-clone"; export CLONE_WS
59 #
60 # CLOSED_CLONE_WS is the workspace from which nightly should acquire
61 # the usr/closed tree.
```

new/usr/src/tools/env/developer.sh

2

```
62 CLOSED_CLONE_WS="${CLONE_WS}/usr/closed"; export CLOSED_CLONE_WS
63 #
64 # The bringover, if any, is done as STAFFER.
65 # Set STAFFER to your own login as gatekeeper or developer
66 # The point is to use group "staff" and avoid referencing the parent
67 # workspace as root.
68 # Some scripts optionally send mail messages to MAILTO.
69 #
70 STAFFER=nobody; export STAFFER
71 MAILTO=$STAFFER; export MAILTO
72 #
73 # The project (see project(4)) under which to run this build. If not
74 # specified, the build is simply run in a new task in the current project.
75 BUILD_PROJECT=; export BUILD_PROJECT
76 #
77 # You should not need to change the next three lines
78 ATLOG="$CODEMGR_WS/log"; export ATLOG
79 LOGFILE="$ATLOG/nightly.log"; export LOGFILE
80 MACH=`uname -p`; export MACH
81 #
82 # When the -A flag is specified, and ELF_DATA_BASELINE_DIR is defined,
83 # the ELF interface description file resulting from the build is compared
84 # to that from the specified directory. This ensures that our object
85 # versioning evolves in a backward compatible manner.
86 #
87 # You should not need to change this unless you wish to use locally cached
88 # baseline files. If you use this, it must be local (or nfs): nightly cannot
89 # copy over ssh or http.
90 #
91 ELF_DATA_BASELINE_DIR="/ws/onnv-gate/usr/src/ELF-data-baseline.$MACH"; export E
92 #
93 # This is usually just needed if the closed tree is missing, or when
94 # building a project gate with the -O (cap oh) flag.
95 # ON_CRYPT_BIN="$PARENT_WS/packages/$MACH/on-crypto.$MACH.tar.bz2"
96 # export ON_CRYPT_BIN
97 #
98 # REF_PROTO_LIST - for comparing the list of stuff in your proto area
99 # with. Generally this should be left alone, since you want to see differences
100 # from your parent (the gate).
101 #
102 REF_PROTO_LIST=$PARENT_WS/usr/src/proto_list_${MACH}; export REF_PROTO_LIST
103 #
104 #
105 # build environment variables, including version info for mcs, motd,
106 # motd, uname and boot messages. Mostly you shouldn't change this except
107 # when the release slips (nah) or you move an environment file to a new
108 # release
109 #
110 ROOT="$CODEMGR_WS/proto/root_${MACH}"; export ROOT
111 SRC="$CODEMGR_WS/usr/src"; export SRC
112 VERSION="$GATE"; export VERSION
113 #
114 #
115 # the RELEASE and RELEASE_DATE variables are set in Makefile.master;
116 # there might be special reasons to override them here, but that
117 # should not be the case in general
118 #
119 # RELEASE="5.10.1"; export RELEASE
120 # RELEASE_DATE="October 2007"; export RELEASE_DATE
121 #
122 # proto area in parent for optionally depositing a copy of headers and
123 # libraries corresponding to the protolibs target
124 # not applicable given the NIGHTLY_OPTIONS
125 #
126 PARENT_ROOT=$PARENT_WS/proto/root_${MACH}; export PARENT_ROOT
127 PARENT_TOOLS_ROOT=$PARENT_WS/usr/src/tools/proto/root_${MACH}-nd; export PARENT_TO
```

```

129 #
130 # Package creation variables. You probably shouldn't change these,
131 # either.
132 #
133 # PKGARCHIVE determines where repositories will be created.
134 #
135 # PKGPUBLISHER* control the publisher settings for those repositories.
136 #
137 PKGARCHIVE="${CODEMGR_WS}/packages/${MACH}/nightly"; export PKGARCHIVE
138 # PKGPUBLISHER_REDIST="on-redist"; export PKGPUBLISHER_REDI
139 # PKGPUBLISHER_NONREDIST="on-extra"; export PKGPUBLISHER_NONR

141 # we want make to do as much as it can, just in case there's more than
142 # one problem.
143 MAKEFLAGS=k; export MAKEFLAGS

145 # Magic variable to prevent the devpro compilers/teamware from sending
146 # mail back to devpro on every use.
147 UT_NO_USAGE_TRACKING="1"; export UT_NO_USAGE_TRACKING

149 # Build tools - don't set these unless you know what you're doing. These
150 # variables allows you to get the compilers and onbld files locally or
151 # through cacheofs. Set BUILD_TOOLS to pull everything from one location.
152 # Alternately, you can set ONBLD_TOOLS to where you keep the contents of
153 # SUNWonbld and SPRO_ROOT to where you keep the compilers.
154 #
155 #BUILD_TOOLS=/opt; export BUILD_TOOLS
156 #ONBLD_TOOLS=/opt/onbld; export ONBLD_TOOLS
157 #SPRO_ROOT=/opt/SUNWspro; export SPRO_ROOT

159 # This goes along with lint - it is a series of the form "A [y|n]" which
160 # means "go to directory A and run 'make lint'" Then mail me (y) the
161 # difference in the lint output. 'y' should only be used if the area you're
162 # linting is actually lint clean or you'll get lots of mail.
163 # You shouldn't need to change this though.
164 #LINTDIRS="$SRC y"; export LINTDIRS

166 #
167 # Reference to IA32 IHV workspace, proto area and packages
168 #
169 #IA32_IHV_WS=/ws/${GATE}-ihv; export IA32_IHV_WS
170 #IA32_IHV_ROOT=${IA32_IHV_WS}/proto/root_i386; export IA32_IHV_ROOT
171 #IA32_IHV_PKGS=${IA32_IHV_WS}/packages/i386/nightly; export IA32_IHV_PKGS

173 #
174 # Reference to binary-only IA32 IHV packages
175 #
176 #IA32_IHV_BINARY_PKGS=/ws/${GATE}-ihv-bin
177 #export IA32_IHV_BINARY_PKGS

179 # Set this flag to 'n' to disable the automatic validation of the dmake
180 # version in use. The default is to check it.
181 #CHECK_DMAKE=y

179 # Set this flag to 'n' to disable the use of 'checkpaths'. The default,
180 # if the 'N' option is not specified, is to run this test.
181 #CHECK_PATHS=y

183 # Set this flag to 'y' to enable the use of elfsigncmp to validate the
184 # output of elfsign. Doing so requires that 't' be set in NIGHTLY_OPTIONS.
185 # The default is to not verify them.
186 #VERIFY_ELFSIGN=n

188 # BRINGOVER_FILES is the list of files nightly passes to bringover.
189 # If not set the default is "usr", but it can be used for bringing

```

```

190 # over deleted_files or other nifty directories.
191 #BRINGOVER_FILES="usr deleted_files"

193 # POST_NIGHTLY can be any command to be run at the end of nightly. See
194 # nightly(1) for interactions between environment variables and this command.
195 #POST_NIGHTLY=

```

new/usr/src/tools/env/gatekeeper.sh

1

```
*****
8428 Fri May 22 11:27:13 2015
new/usr/src/tools/env/gatekeeper.sh
nightly: don't check dmake version, now it's ours
*****
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 (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
24 #
25 #
26 # Configuration variables for the runtime environment of the nightly
27 # build script and other tools for construction and packaging of releases.
28 # This script is sourced by 'nightly' and 'blendv' to set up the environment
29 # for the build. This example is suitable for building a gate,
30 # which will contain the resulting packages and archives (builds of the gate
31 # are done in children and then the resulting archives, packages, and proto
32 # area are put into the parent for everyone to use). It is based off
33 # the onnv release. It sets NIGHTLY_OPTIONS to make nightly do:
34 # DEBUG and non-DEBUG builds (-D)
35 # creates packages for PIT/RE (-p)
36 # checks for new interfaces in libraries (-A)
37 # runs 'make check' (-C)
38 # runs lint in usr/src (-l plus the LINTDIRS variable)
39 # sends mail on completion (-m and the MAILTO variable)
40 # updates the protolist in the parent for children to compare with (-u)
41 # updates the proto area in the parent when done (-U)
42 # checks for changes in ELF runpaths (-r)
43 # checks for changes in unreferenced files (-f)
44 #
45 NIGHTLY_OPTIONS="-ADclmpuUrf"; export NIGHTLY_OPTIONS
46 #
47 # This is a variable for the rest of the script - GATE doesn't matter to
48 # nightly itself
49 GATE=onnv-gate; export GATE
50 #
51 # CODEMGR_WS - where is your workspace at (or what should nightly name it)
52 # there is only one definition here, which assumes all the gate build machines
53 # (sparc and x86) are set up the same. But remember, this is a script, so
54 # you could look at $MACH or `uname -n` and set these variables differently.
55 CODEMGR_WS="/builds/$GATE"; export CODEMGR_WS
56 #
57 # PARENT_WS is used to determine the parent of this workspace. This is
58 # for the options that deal with the parent workspace (such as where the
59 # proto area will go).
60 #
61 # If you use this, it must be local (or nfs): nightly cannot copy
```

new/usr/src/tools/env/gatekeeper.sh

2

```
62 # over ssh or http.
63 PARENT_WS="/ws/$GATE"; export PARENT_WS
64 #
65 # CLONE_WS is the workspace nightly should do a bringover from.
66 CLONE_WS="ssh://anonhg@onnv.sfbay.sun.com//export/onnv-clone"; export CLONE_WS
67 #
68 # CLOSED_CLONE_WS is the workspace from which nightly will acquire the
69 # usr/closed tree.
70 CLOSED_CLONE_WS="${CLONE_WS}/usr/closed"
71 export CLOSED_CLONE_WS
72 #
73 # The bringover, if any, is done as STAFFER.
74 # Set STAFFER to your own login as gatekeeper or integration engineer.
75 # The point is to use group "staff" and avoid referencing the parent
76 # workspace as root.
77 # Some scripts optionally send mail messages to MAILTO.
78 #
79 STAFFER=nobody; export STAFFER
80 MAILTO=$STAFFER; export MAILTO
81 #
82 # The project (see project(4)) under which to run this build. If not
83 # specified, the build is simply run in a new task in the current project.
84 BUILD_PROJECT=; export BUILD_PROJECT
85 #
86 # You should not need to change the next three lines
87 ATLOG="$CODEMGR_WS/log"; export ATLOG
88 LOGFILE="$ATLOG/nightly.log"; export LOGFILE
89 MACH=`uname -p`; export MACH
90 #
91 # When the -A flag is specified, and ELF_DATA_BASELINE_DIR is defined,
92 # the ELF interface description file resulting from the build is compared
93 # to that from the specified directory. This ensures that our object
94 # versioning evolves in a backward compatible manner.
95 #
96 # You should not need to change this unless you wish to use locally cached
97 # baseline files. If you use this, it must be local (or nfs): nightly cannot
98 # copy over ssh or http.
99 #
100 ELF_DATA_BASELINE_DIR="/ws/onnv-gate/usr/src/ELF-data-baseline.$MACH"; export E
101 #
102 # This is usually just needed if the closed tree is missing, or when
103 # building a project gate with the -O (cap oh) flag.
104 # ON_CRYPTOBINS="$PARENT_WS/packages/$MACH/on-crypto.$MACH.tar.bz2"
105 # export ON_CRYPTOBINS
106 #
107 # REF_PROTO_LIST - for comparing the list of stuff in your proto area
108 # with. Generally this should be left alone, since you want to see differences
109 # between today's build and yesterdays.
110 #
111 REF_PROTO_LIST=$PARENT_WS/usr/src/proto_list_{$MACH}; export REF_PROTO_LIST
112 #
113 #
114 # build environment variables, including version info for mcs, motd,
115 # motd, uname and boot messages. Mostly you shouldn't change this except
116 # when the release slips (nah) or when starting a new release.
117 #
118 ROOT="$CODEMGR_WS/proto/root_{$MACH}"; export ROOT
119 SRC="$CODEMGR_WS/usr/src"; export SRC
120 VERSION="$GATE"; export VERSION
121 #
122 #
123 # the RELEASE and RELEASE_DATE variables are set in Makefile.master;
124 # there might be special reasons to override them here, but that
125 # should not be the case in general
126 #
127 # RELEASE="5.10.1"; export RELEASE
```

new/usr/src/tools/env/gatekeeper.sh

3

```
128 # RELEASE_DATE="October 2007";           export RELEASE_DATE
130 # proto area in parent for optionally depositing a copy of headers and
131 # libraries corresponding to the protolibs target
132 #
133 PARENT_ROOT=$PARENT_WS/proto/root_${MACH}; export PARENT_ROOT
134 PARENT_TOOLS_ROOT=$PARENT_WS/usr/src/tools/proto/root_${MACH}-nd; export PARENT_TO

136 #
137 # Package creation variables. You probably shouldn't change these,
138 # either.
139 #
140 # PKGARCHIVE determines where repositories will be created.
141 #
142 # PKGPUBLISHER* control the publisher settings for those repositories.
143 #
144 PKGARCHIVE="${PARENT_WS}/packages/${MACH}/nightly";   export PKGARCHIVE
145 # PKGPUBLISHER_REDIST="on-nightly";                 export PKGPUBLISHER_REDIST
146 # PKGPUBLISHER_NONREDIST="on-extra";                 export PKGPUBLISHER_NONREDIST

149 # we want make to do as much as it can, just in case there's more than
150 # one problem. This is especially important with the gate, since multiple
151 # unrelated broken things can be integrated.
152 MAKEFLAGS=k;   export MAKEFLAGS

154 # Magic variable to prevent the devpro compilers/teamware from sending
155 # mail back to devpro on every use.
156 UT_NO_USAGE_TRACKING="1"; export UT_NO_USAGE_TRACKING

158 # Build tools - don't set these unless you know what you're doing. These
159 # variables allows you to get the compilers and onbld files locally or
160 # through cacheofs. Set BUILD_TOOLS to pull everything from one location.
161 # Alternately, you can set ONBLD_TOOLS to where you keep the contents of
162 # SUNWonbld and SPRO_ROOT to where you keep the compilers.
163 #
164 #BUILD_TOOLS=/opt;           export BUILD_TOOLS
165 #ONBLD_TOOLS=/opt/onbld;     export ONBLD_TOOLS
166 #SPRO_ROOT=/opt/SUNspro;     export SPRO_ROOT

168 # This goes along with lint - it is a series of the form "A [y|n]" which
169 # means "go to directory A and run 'make lint'" Then mail me (y) the
170 # difference in the lint output. 'y' should only be used if the area you're
171 # linting is actually lint clean or you'll get lots of mail.
172 # You shouldn't need to change this though.
173 #LINTDIRS="$SRC y";   export LINTDIRS

175 #
176 # Reference to IA32 IHV workspace, proto area and packages
177 #
178 #IA32_IHV_WS=/ws/${GATE}-ihv;           export IA32_IHV_WS
179 #IA32_IHV_ROOT=$IA32_IHV_WS/proto/root_i386;   export IA32_IHV_ROOT
180 #IA32_IHV_PKGS=$IA32_IHV_WS/packages/i386/nightly;   export IA32_IHV_PKGS

182 #
183 # Reference to binary-only IA32 IHV packages
184 #
185 #IA32_IHV_BINARY_PKGS=/ws/${GATE}-ihv-bin
186 #export IA32_IHV_BINARY_PKGS

188 # Set this flag to 'n' to disable the automatic validation of the dmake
189 # version in use. The default is to check it.
190 #CHECK_DMAKE=y

188 # Set this flag to 'n' to disable the use of 'checkpaths'. The default,
189 # if the 'N' option is not specified, is to run this test.
```

new/usr/src/tools/env/gatekeeper.sh

4

```
190 #CHECK_PATHS=y

192 # Set this flag to 'y' to enable the use of elfsigncmp to validate the
193 # output of elfsign. Doing so requires that 't' be set in NIGHTLY_OPTIONS.
194 # The default is to not verify them.
195 #VERIFY_ELFSIGN=n

197 # BRINGOVER_FILES is the list of files nightly passes to bringover.
198 # If not set the default is "usr", but it can be used for bringing
199 # over deleted_files or other nifty directories.
200 #BRINGOVER_FILES="usr deleted_files"

202 # POST_NIGHTLY can be any command to be run at the end of nightly. See
203 # nightly(1) for interactions between environment variables and this command.
204 #POST_NIGHTLY=
```

new/usr/src/tools/env/illumos.sh

1

```
*****
8903 Fri May 22 11:27:13 2015
new/usr/src/tools/env/illumos.sh
nightly: don't check dmake version, now it's ours
*****
_____unchanged_portion_omitted_____

101 maxjobs DMAKE_MAX_JOBS # "DMAKE_MAX_JOBS" passed as ksh(1) name reference
102 export DMAKE_MAX_JOBS

104 # path to onbld tool binaries
105 ONBLD_BIN="/opt/onbld/bin"

107 # PARENT_WS is used to determine the parent of this workspace. This is
108 # for the options that deal with the parent workspace (such as where the
109 # proto area will go).
110 export PARENT_WS=""

112 # CLONE_WS is the workspace nightly should do a bringover from.
113 export CLONE_WS="ssh://anonhg@hg.illumos.org/illumos-gate"

115 # The bringover, if any, is done as STAFFER.
116 # Set STAFFER to your own login as gatekeeper or developer
117 # The point is to use group "staff" and avoid referencing the parent
118 # workspace as root.
119 # Some scripts optionally send mail messages to MAILTO.
120 #
121 export STAFFER="$LOGNAME"
122 export MAILTO="$STAFFER"

124 # If you wish the mail messages to be From: an arbitrary address, export
125 # MAILFROM.
126 #export MAILFROM="user@example.com"

128 # The project (see project(4)) under which to run this build. If not
129 # specified, the build is simply run in a new task in the current project.
130 export BUILD_PROJECT=""

132 # You should not need to change the next three lines
133 export ATLOG="$CODEMGR_WS/log"
134 export LOGFILE="$ATLOG/nightly.log"
135 export MACH="$(uname -p)"

137 #
138 # The following two macros are the closed/crypto binaries. Once
139 # illumos has totally freed itself, we can remove these references.
140 #
141 # Location of encumbered binaries.
142 export ON_CLOSED_BINS="$CODEMGR_WS/closed"
143 # Location of signed cryptographic binaries.
144 export ON_CRYPTO_BINS="$CODEMGR_WS/on-crypto.$MACH.tar.bz2"

146 # REF_PROTO_LIST - for comparing the list of stuff in your proto area
147 # with. Generally this should be left alone, since you want to see differences
148 # from your parent (the gate).
149 #
150 export REF_PROTO_LIST="$PARENT_WS/usr/src/proto_list_${MACH}"

153 export ROOT="$CODEMGR_WS/proto/root_${MACH}"
154 export SRC="$CODEMGR_WS/usr/src"
155 export MULTI_PROTO="no"

157 #
158 # build environment variables, including version info for mcs, motd,
159 # motd, uname and boot messages. Mostly you shouldn't change this except
```

new/usr/src/tools/env/illumos.sh

2

```
160 # when the release slips (nah) or you move an environment file to a new
161 # release
162 #
163 export VERSION="$GATE"

165 #
166 # the RELEASE and RELEASE_DATE variables are set in Makefile.master;
167 # there might be special reasons to override them here, but that
168 # should not be the case in general
169 #
170 # export RELEASE='5.11'
171 # export RELEASE_DATE='October 2007'

173 # proto area in parent for optionally depositing a copy of headers and
174 # libraries corresponding to the protolib target
175 # not applicable given the NIGHTLY_OPTIONS
176 #
177 export PARENT_ROOT="$PARENT_WS/proto/root_${MACH}"
178 export PARENT_TOOLS_ROOT="$PARENT_WS/usr/src/tools/proto/root_${MACH}-nd"

180 # Package creation variables. You probably shouldn't change these,
181 # either.
182 #
183 # PKGARCHIVE determines where the repository will be created.
184 #
185 # PKGPUBLISHER_REDIST controls the publisher setting for the repository.
186 #
187 export PKGARCHIVE="{CODEMGR_WS}/packages/{MACH}/nightly"
188 # export PKGPUBLISHER_REDIST='on-redist'

190 # Package manifest format version.
191 export PKGFMT_OUTPUT='v1'

193 # we want make to do as much as it can, just in case there's more than
194 # one problem.
195 export MAKEFLAGS='k'

197 # Magic variable to prevent the devpro compilers/teamware from sending
198 # mail back to devpro on every use.
199 export UT_NO_USAGE_TRACKING='1'

201 # Build tools - don't change these unless you know what you're doing. These
202 # variables allows you to get the compilers and onbld files locally or
203 # through cacheofs. Set BUILD_TOOLS to pull everything from one location.
204 # Alternately, you can set ONBLD_TOOLS to where you keep the contents of
205 # SUNWonbld and SPRO_ROOT to where you keep the compilers. SPRO_VROOT
206 # exists to make it easier to test new versions of the compiler.
207 export BUILD_TOOLS="/opt"
208 #export ONBLD_TOOLS="/opt/onbld"
209 export SPRO_ROOT="/opt/SUNWspr"
210 export SPRO_VROOT="$SPRO_ROOT"

212 # This goes along with lint - it is a series of the form "A [y|n]" which
213 # means "go to directory A and run 'make lint'". Then mail me (y) the
214 # difference in the lint output. 'y' should only be used if the area you're
215 # linting is actually lint clean or you'll get lots of mail.
216 # You shouldn't need to change this though.
217 #export LINTDIRS="$SRC y"

219 # Set this flag to 'n' to disable the automatic validation of the dmake
220 # version in use. The default is to check it.
221 #CHECK_DMAKE='y'

219 # Set this flag to 'n' to disable the use of 'checkpaths'. The default,
220 # if the 'N' option is not specified, is to run this test.
221 #CHECK_PATHS='y'
```

```
223 # POST_NIGHTLY can be any command to be run at the end of nightly. See
224 # nightly(1) for interactions between environment variables and this command.
225 #POST_NIGHTLY=

227 # Comment this out to disable support for IPP printing, i.e. if you
228 # don't want to bother providing the Apache headers this needs.
229 export ENABLE_IPP_PRINTING=

231 # Comment this out to disable support for SMB printing, i.e. if you
232 # don't want to bother providing the CUPS headers this needs.
233 export ENABLE_SMB_PRINTING=

235 # If your distro uses certain versions of Perl, make sure either Makefile.master
236 # contains your new defaults OR your .env file sets them.
237 # These are how you would override for building on OmniOS r151012, for example.
238 #export PERL_VERSION=5.16.1
239 #export PERL_ARCH=i86pc-solaris-thread-multi-64int
240 #export PERL_PKGVERS=-5161
```

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

1

```
*****
59152 Fri May 22 11:27:14 2015
new/usr/src/tools/scripts/nightly.sh
nightly: don't check dmake version, now it's ours
*****
_____unchanged_portion_omitted_____

614 MACH=`uname -p`

616 if [ "$OPTHOME" = "" ]; then
617     OPTHOME=/opt
618     export OPTHOME
619 fi

621 USAGE='Usage: nightly [-in] [+t] [-V VERS ] <env_file>'

623 Where:
624     -i         Fast incremental options (no clobber, lint, check)
625     -n         Do not do a bringover
626     +t         Use the build tools in $ONBLD_TOOLS/bin
627     -V VERS   set the build version string to VERS

629     <env_file> file in Bourne shell syntax that sets and exports
630     variables that configure the operation of this script and many of
631     the scripts this one calls. If <env_file> does not exist,
632     it will be looked for in $OPTHOME/onbld/env.

634 non-DEBUG is the default build type. Build options can be set in the
635 NIGHTLY_OPTIONS variable in the <env_file> as follows:

637     -A         check for ABI differences in .so files
638     -C         check for cstyle/hdrchk errors
639     -D         do a build with DEBUG on
640     -F         do _not_ do a non-DEBUG build
641     -G         gate keeper default group of options (-au)
642     -I         integration engineer default group of options (-ampu)
643     -M         do not run pmodes (safe file permission checker)
644     -N         do not run protocmp
645     -R         default group of options for building a release (-mp)
646     -U         update proto area in the parent
647     -V VERS   set the build version string to VERS
648     -f         find unreferenced files
649     -i         do an incremental build (no "make clobber")
650     -l         do "make lint" in $LINTDIRS (default: $SRC y)
651     -m         send mail to $MAILTO at end of build
652     -n         do not do a bringover
653     -p         create packages
654     -r         check ELF runtime attributes in the proto area
655     -t         build and use the tools in $SRC/tools (default setting)
656     +t         Use the build tools in $ONBLD_TOOLS/bin
657     -u         update proto_list_$MACH and friends in the parent workspace;
658     when used with -f, also build an unrefmaster.out in the parent
659     -w         report on differences between previous and current proto areas
660 '
661 #
662 #     A log file will be generated under the name $LOGFILE
663 #     for partially completed build and log.`date +%F`
664 #     in the same directory for fully completed builds.
665 #

667 # default values for low-level FLAGS; G I R are group FLAGS
668 A_FLAG=n
669 C_FLAG=n
670 D_FLAG=n
671 F_FLAG=n
```

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

2

```
672 f_FLAG=n
673 i_FLAG=n; i_CMD_LINE_FLAG=n
674 l_FLAG=n
675 M_FLAG=n
676 m_FLAG=n
677 N_FLAG=n
678 n_FLAG=n
679 p_FLAG=n
680 r_FLAG=n
681 t_FLAG=y
682 U_FLAG=n
683 u_FLAG=n
684 V_FLAG=n
685 w_FLAG=n
686 W_FLAG=n
687 #
688 build_ok=y
689 build_extras_ok=y

691 #
692 # examine arguments
693 #

695 OPTIND=1
696 while getopts +intV:W FLAG
697 do
698     case $FLAG in
699         i ) i_FLAG=y; i_CMD_LINE_FLAG=y
700             ;;
701         n ) n_FLAG=y
702             ;;
703         +t ) t_FLAG=n
704             ;;
705         V ) V_FLAG=y
706             V_ARG="$OPTARG"
707             ;;
708         W ) W_FLAG=y
709             ;;
710         \? ) echo "$USAGE"
711             exit 1
712             ;;
713         esac
714 done

716 # correct argument count after options
717 shift `expr $OPTIND - 1`

719 # test that the path to the environment-setting file was given
720 if [ $# -ne 1 ]; then
721     echo "$USAGE"
722     exit 1
723 fi

725 # check if user is running nightly as root
726 # ISUSER is set non-zero if an ordinary user runs nightly, or is zero
727 # when root invokes nightly.
728 /usr/bin/id | grep '^uid=0(' >/dev/null 2>&1
729 ISUSER=$?; export ISUSER

731 #
732 # force locale to C
733 LANG=C; export LANG
734 LC_ALL=C; export LC_ALL
735 LC_COLLATE=C; export LC_COLLATE
736 LC_CTYPE=C; export LC_CTYPE
737 LC_MESSAGES=C; export LC_MESSAGES
```

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

```

738 LC_MONETARY=C; export LC_MONETARY
739 LC_NUMERIC=C; export LC_NUMERIC
740 LC_TIME=C; export LC_TIME

742 # clear environment variables we know to be bad for the build
743 unset LD_OPTIONS
744 unset LD_AUDIT          LD_AUDIT_32          LD_AUDIT_64
745 unset LD_BIND_NOW      LD_BIND_NOW_32       LD_BIND_NOW_64
746 unset LD_BREADTH      LD_BREADTH_32       LD_BREADTH_64
747 unset LD_CONFIG       LD_CONFIG_32        LD_CONFIG_64
748 unset LD_DEBUG        LD_DEBUG_32         LD_DEBUG_64
749 unset LD_DEMANGLE     LD_DEMANGLE_32      LD_DEMANGLE_64
750 unset LD_FLAGS        LD_FLAGS_32         LD_FLAGS_64
751 unset LD_LIBRARY_PATH  LD_LIBRARY_PATH_32  LD_LIBRARY_PATH_64
752 unset LD_LOADFLTR     LD_LOADFLTR_32      LD_LOADFLTR_64
753 unset LD_NOAUDIT      LD_NOAUDIT_32       LD_NOAUDIT_64
754 unset LD_NOAUXFLTR    LD_NOAUXFLTR_32     LD_NOAUXFLTR_64
755 unset LD_NOCONFIG     LD_NOCONFIG_32      LD_NOCONFIG_64
756 unset LD_NODIRCONFIG  LD_NODIRCONFIG_32   LD_NODIRCONFIG_64
757 unset LD_NODIRECT     LD_NODIRECT_32      LD_NODIRECT_64
758 unset LD_NOLAZYLOAD   LD_NOLAZYLOAD_32  LD_NOLAZYLOAD_64
759 unset LD_NOOBJALTER   LD_NOOBJALTER_32   LD_NOOBJALTER_64
760 unset LD_NOVERSION    LD_NOVERSION_32     LD_NOVERSION_64
761 unset LD_ORIGIN       LD_ORIGIN_32        LD_ORIGIN_64
762 unset LD_PRELOAD      LD_PRELOAD_32       LD_PRELOAD_64
763 unset LD_PROFILE      LD_PROFILE_32       LD_PROFILE_64

765 unset CONFIG
766 unset GROUP
767 unset OWNER
768 unset REMOTE
769 unset ENV
770 unset ARCH
771 unset CLASSPATH
772 unset NAME

774 #
775 # To get ONBLD_TOOLS from the environment, it must come from the env file.
776 # If it comes interactively, it is generally TOOLS_PROTO, which will be
777 # clobbered before the compiler version checks, which will therefore fail.
778 #
779 unset ONBLD_TOOLS

781 #
782 # Setup environmental variables
783 #
784 if [ -f /etc/nightly.conf ]; then
785     . /etc/nightly.conf
786 fi

788 if [ -f $1 ]; then
789     if [[ $1 = */* ]]; then
790         . $1
791     else
792         . ./$1
793     fi
794 else
795     if [ -f $OPTHOME/onbld/env/$1 ]; then
796         . $OPTHOME/onbld/env/$1
797     else
798         echo "Cannot find env file as either $1 or $OPTHOME/onbld/env/$1
799         exit 1
800     fi
801 fi

803 # contents of stdenv.sh inserted after next line:

```

3

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

```

804 # STDENV_START
805 # STDENV_END

807 # Check if we have sufficient data to continue...
808 [[ -v CODEMGR_WS ]] || fatal_error "Error: Variable CODEMGR_WS not set."
809 if [[ "${NIGHTLY_OPTIONS}" == ~(F)n ]]; then
810     # Check if the gate data are valid if we don't do a "bringover" below
811     [[ -d "${CODEMGR_WS}" ]] || \
812         fatal_error "Error: ${CODEMGR_WS} is not a directory."
813     [[ -f "${CODEMGR_WS}/usr/src/Makefile" ]] || \
814         fatal_error "Error: ${CODEMGR_WS}/usr/src/Makefile not found."
815 fi

817 #
818 # place ourselves in a new task, respecting BUILD_PROJECT if set.
819 #
820 if [ -z "$BUILD_PROJECT" ]; then
821     /usr/bin/newtask -c $$
822 else
823     /usr/bin/newtask -c $$ -p $BUILD_PROJECT
824 fi

826 ps -o taskid= -p $$ | read build_taskid
827 ps -o project= -p $$ | read build_project

829 #
830 # See if NIGHTLY_OPTIONS is set
831 #
832 if [ "$NIGHTLY_OPTIONS" = "" ]; then
833     NIGHTLY_OPTIONS="-aBm"
834 fi

836 #
837 # If BRINGOVER_WS was not specified, let it default to CLONE_WS
838 #
839 if [ "$BRINGOVER_WS" = "" ]; then
840     BRINGOVER_WS=$CLONE_WS
841 fi

843 #
844 # If BRINGOVER_FILES was not specified, default to usr
845 #
846 if [ "$BRINGOVER_FILES" = "" ]; then
847     BRINGOVER_FILES="usr"
848 fi

850 check_closed_bins

852 #
853 # Note: changes to the option letters here should also be applied to the
854 # bldenv script. 'd' is listed for backward compatibility.
855 #
856 NIGHTLY_OPTIONS=-${NIGHTLY_OPTIONS}-
857 OPTIND=1
858 while getopts +ABCdFfGIlMmNnpRrtUuwW FLAG $NIGHTLY_OPTIONS
859 do
860     case $FLAG in
861         A ) A_FLAG=y
862             ;;
863         B ) D_FLAG=y
864             ;; # old version of D
865         C ) C_FLAG=y
866             ;;
867         D ) D_FLAG=y
868             ;;
869         F ) F_FLAG=y

```

4

```

870      ;;
871      f )  f_FLAG=y
872          ;;
873      G )  u_FLAG=y
874          ;;
875      I )  m_FLAG=y
876          p_FLAG=y
877          u_FLAG=y
878          ;;
879      i )  i_FLAG=y
880          ;;
881      l )  l_FLAG=y
882          ;;
883      M )  M_FLAG=y
884          ;;
885      m )  m_FLAG=y
886          ;;
887      N )  N_FLAG=y
888          ;;
889      n )  n_FLAG=y
890          ;;
891      p )  p_FLAG=y
892          ;;
893      R )  m_FLAG=y
894          p_FLAG=y
895          ;;
896      r )  r_FLAG=y
897          ;;
898      +t ) t_FLAG=n
899          ;;
900      U ) if [ -z "${PARENT_ROOT}" ]; then
901          echo "PARENT_ROOT must be set if the U flag is" \
902              "present in NIGHTLY_OPTIONS."
903          exit 1
904      fi
905      NIGHTLY_PARENT_ROOT=$PARENT_ROOT
906      if [ -n "${PARENT_TOOLS_ROOT}" ]; then
907          NIGHTLY_PARENT_TOOLS_ROOT=$PARENT_TOOLS_ROOT
908      fi
909      U_FLAG=y
910          ;;
911      u )  u_FLAG=y
912          ;;
913      w )  w_FLAG=y
914          ;;
915      W )  W_FLAG=y
916          ;;
917      \? ) echo "$USAGE"
918          exit 1
919          ;;
920      esac
921 done

923 if [ $ISUSER -ne 0 ]; then
924     # Set default value for STAFFER, if needed.
925     if [ -z "$STAFFER" -o "$STAFFER" = "nobody" ]; then
926         STAFFER="/usr/xpg4/bin/id -un"
927         export STAFFER
928     fi
929 fi

931 if [ -z "$MAILTO" -o "$MAILTO" = "nobody" ]; then
932     MAILTO=$STAFFER
933     export MAILTO
934 fi

```

```

936 PATH="$OPTHOME/onbld/bin:$OPTHOME/onbld/bin/${MACH}:/usr/ccs/bin"
937 PATH="$PATH:$OPTHOME/SUNWspro/bin:/usr/bin:/usr/sbin:/usr/ucb"
938 PATH="$PATH:/usr/openwin/bin:/usr/sfw/bin:/opt/sfw/bin:."
939 export PATH

941 # roots of source trees, both relative to $SRC and absolute.
942 relsrcdirs="."
943 absrsrcdirs="$SRC"

945 PROTOCMPTERSE="protocmp.terse -gu"
946 POUND_SIGN="#"
947 # have we set RELEASE_DATE in our env file?
948 if [ -z "$RELEASE_DATE" ]; then
949     RELEASE_DATE=$(LC_ALL=C date +"%B %Y")
950 fi
951 BUILD_DATE=$(LC_ALL=C date +%Y-%b-%d)
952 BASEWSDIR=$(basename $CODEMGR_WS)
953 DEV_CM="\@(#)SunOS Internal Development: $LOGNAME $BUILD_DATE [$BASEWSDIR]\\"

955 # we export POUND_SIGN, RELEASE_DATE and DEV_CM to speed up the build process
956 # by avoiding repeated shell invocations to evaluate Makefile.master
957 # definitions.
958 export POUND_SIGN RELEASE_DATE DEV_CM

960 maketype="distributed"
961 if [[ -z "$MAKE" ]]; then
962     MAKE=dmake
963 elif [[ ! -x "$MAKE" ]]; then
964     echo "\$MAKE is set to garbage in the environment"
965     exit 1
966 fi
967 # get the dmake version string alone
968 DMAKE_VERSION=$( $MAKE -v )
969 DMAKE_VERSION=${DMAKE_VERSION#*: }
970 # focus in on just the dotted version number alone
971 DMAKE_MAJOR=$( echo $DMAKE_VERSION | \
972     sed -e 's/.*\<([^\.]*)\.[^\ ]*\)/.*$/\1/' )
973 # extract the second (or final) integer
974 DMAKE_MINOR=${DMAKE_MAJOR#*.}
975 DMAKE_MINOR=${DMAKE_MINOR%*.}
976 # extract the first integer
977 DMAKE_MAJOR=${DMAKE_MAJOR%*.}
978 CHECK_DMAKE=${CHECK_DMAKE:-y}
979 # x86 was built on the 12th, sparc on the 13th.
980 if [ "$CHECK_DMAKE" = "y" -a \
981     "$DMAKE_VERSION" != "Sun Distributed Make 7.3 2003/03/12" -a \
982     "$DMAKE_VERSION" != "Sun Distributed Make 7.3 2003/03/13" -a \
983     "$DMAKE_MAJOR" -lt 7 -o \
984     "$DMAKE_MAJOR" -eq 7 -a "$DMAKE_MINOR" -lt 4 \ ] ]; then
985     if [ -z "$DMAKE_VERSION" ]; then
986         echo "$MAKE is missing."
987         exit 1
988     fi
989     echo 'whence $MAKE\' version is:"
990     echo " ${DMAKE_VERSION}"
991     cat <<EOF

993 This version may not be safe for use, if you really want to use this version
994 anyway add the following to your environment to disable this check:

996 CHECK_DMAKE=n
997 EOF
998     exit 1
999 fi
1000 export PATH
1001 export MAKE

```

```

970 if [ "${SUNWSPRO}" != "" ]; then
971     PATH="${SUNWSPRO}/bin:$PATH"
972     export PATH
973 fi

975 hostname=$(uname -n)
976 if [[ $DMAKE_MAX_JOBS != +([0-9]) || $DMAKE_MAX_JOBS -eq 0 ]]
977 then
978     maxjobs=
979     if [[ -f $HOME/.make.machines ]]
980     then
981         # Note: there is a hard tab and space character in the []s
982         # below.
983         egrep -i "^[ \t]*$hostname[ \t\." \
984             $HOME/.make.machines | read host jobs
985         maxjobs=${jobs##*=}
986     fi

988     if [[ $maxjobs != +([0-9]) || $maxjobs -eq 0 ]]
989     then
990         # default
991         maxjobs=4
992     fi

994     export DMAKE_MAX_JOBS=$maxjobs
995 fi

997 DMAKE_MODE=parallel;
998 export DMAKE_MODE

1000 if [ -z "${ROOT}" ]; then
1001     echo "ROOT must be set."
1002     exit 1
1003 fi

1005 #
1006 # if -V flag was given, reset VERSION to V_ARG
1007 #
1008 if [ "$V_FLAG" = "y" ]; then
1009     VERSION=$V_ARG
1010 fi

1012 TMPDIR="/tmp/nightly.tmpdir.$$"
1013 export TMPDIR
1014 rm -rf ${TMPDIR}
1015 mkdir -p $TMPDIR || exit 1
1016 chmod 777 $TMPDIR

1018 #
1019 # Keep elfsign's use of pkcs11_softtoken from looking in the user home
1020 # directory, which doesn't always work. Needed until all build machines
1021 # have the fix for 6271754
1022 #
1023 SOFTTOKEN_DIR=$TMPDIR
1024 export SOFTTOKEN_DIR

1026 #
1027 # Tools should only be built non-DEBUG. Keep track of the tools proto
1028 # area path relative to $TOOLS, because the latter changes in an
1029 # export build.
1030 #
1031 # TOOLS_PROTO is included below for builds other than usr/src/tools
1032 # that look for this location. For usr/src/tools, this will be
1033 # overridden on the $MAKE command line in build_tools().
1034 #

```

```

1035 TOOLS=${SRC}/tools
1036 TOOLS_PROTO_REL=proto/root_${MACH}-nd
1037 TOOLS_PROTO=${TOOLS}/${TOOLS_PROTO_REL}; export TOOLS_PROTO

1039 unset CFLAGS LD_LIBRARY_PATH LDFLAGS

1041 # create directories that are automatically removed if the nightly script
1042 # fails to start correctly
1043 function newdir {
1044     dir=$1
1045     toadd=
1046     while [ ! -d $dir ]; do
1047         toadd="$dir $toadd"
1048         dir='dirname $dir'
1049     done
1050     torm=
1051     newlist=
1052     for dir in $toadd; do
1053         if staffer mkdir $dir; then
1054             newlist="$ISUSER $dir $newlist"
1055             torm="$dir $torm"
1056         else
1057             [ -z "$storm" ] || staffer rmdir $storm
1058             return 1
1059         fi
1060     done
1061     newdirlist="$newlist $newdirlist"
1062     return 0
1063 }

```

unchanged portion omitted