```
**********************************************************
   10610 Fri Oct 11 17:19:04 2013
new/usr/src/pkg/manifests/developer-build-onbld.mf
4108 remove ON_CRYPTO_BINS from tools
**********************************************************
```

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

#
# Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2010, Richard Lowe
# Copyright 2012, Piotr Jasiukajtis
#

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

#
# This package should not be incorporated.  This allows the tools
# to be upgraded without upgrading the entire system.
#
set name=org.opensolaris.noincorp value=true
set name=variant.arch value=$(ARCH)
dir path=opt group=sys
dir path=opt/onbld
dir path=opt/onbld/bin
dir path=opt/onbld/bin/$(ARCH)
dir path=opt/onbld/env
dir path=opt/onbld/etc
dir path=opt/onbld/etc/exception_lists
dir path=opt/onbld/gk
dir path=opt/onbld/lib
dir path=opt/onbld/lib/$(ARCH)
dir path=opt/onbld/lib/perl
dir path=opt/onbld/lib/python2.6
dir path=opt/onbld/lib/python2.6/onbld
dir path=opt/onbld/lib/python2.6/onbld/Checks
dir path=opt/onbld/lib/python2.6/onbld/Scm
dir path=opt/onbld/lib/python2.6/onbld/hgext
dir path=opt/onbld/man
dir path=opt/onbld/man/man1
$(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/aw mode=0555
$(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/chk4ubin mode=0555
file path=opt/onbld/bin/$(ARCH)/codereview mode=0555
file path=opt/onbld/bin/$(ARCH)/cscope-fast mode=0555
```



```
file path=opt/onbld/bin/$(ARCH)/ctfconvert mode=0555
file path=opt/onbld/bin/$(ARCH)/ctfdump mode=0555
file path=opt/onbld/bin/$(ARCH)/ctfmerge mode=0555
file path=opt/onbld/bin/$(ARCH)/ctfstabs mode=0555
file path=opt/onbld/bin/$(ARCH)/ctfstrip mode=0555
file path=opt/onbld/bin/$(ARCH)/cw mode=0555
$(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/elfextract mode=0555
file path=opt/onbld/bin/$(ARCH)/findunref mode=0555
$(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/forth mode=0555
$(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/forth_preload.so.1 mode=0555
file path=opt/onbld/bin/$(ARCH)/install mode=0555
file path=opt/onbld/bin/$(ARCH)/lintdump mode=0555
$(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/mbh_patch mode=0555
file path=opt/onbld/bin/$(ARCH)/ndrgen mode=0555
file path=opt/onbld/bin/$(ARCH)/ndrgen1 mode=0555
file path=opt/onbld/bin/$(ARCH)/pmodes mode=0555
file path=opt/onbld/bin/$(ARCH)/protocmp mode=0555
file path=opt/onbld/bin/$(ARCH)/protolist mode=0555
$(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/stabs mode=0555
$(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/tokenize mode=0555
$(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/tokenize.exe mode=0555
file path=opt/onbld/bin/Install mode=0555
file path=opt/onbld/bin/bindrop mode=0555
file path=opt/onbld/bin/bldenv mode=0555
file path=opt/onbld/bin/bringovercheck mode=0555
file path=opt/onbld/bin/build_cscope mode=0555
file path=opt/onbld/bin/cddlchk mode=0555
file path=opt/onbld/bin/check_rtime mode=0555
file path=opt/onbld/bin/checkpaths mode=0555
file path=opt/onbld/bin/checkproto mode=0555
file path=opt/onbld/bin/copyrightchk mode=0555
file path=opt/onbld/bin/cryptodrop mode=0555
file path=opt/onbld/bin/cstyle mode=0555
file path=opt/onbld/bin/ctfcvtptbl mode=0555
file path=opt/onbld/bin/ctffindmod mode=0555
file path=opt/onbld/bin/elfcmp mode=0555
file path=opt/onbld/bin/elfsigncmp mode=0555
file path=opt/onbld/bin/find_elf mode=0555
file path=opt/onbld/bin/findcrypto mode=0555
file path=opt/onbld/bin/flg.flp mode=0555
file path=opt/onbld/bin/genoffsets mode=0555
file path=opt/onbld/bin/get_depend_info mode=0555
file path=opt/onbld/bin/git-pbchk mode=0555
file path=opt/onbld/bin/hdrchk mode=0555
file path=opt/onbld/bin/hg-active mode=0555
file path=opt/onbld/bin/hgsetup mode=0555
file path=opt/onbld/bin/interface_check mode=0555
file path=opt/onbld/bin/interface_cmp mode=0555
file path=opt/onbld/bin/jstyle mode=0555
file path=opt/onbld/bin/make_pkg_db mode=0555
file path=opt/onbld/bin/mapfilechk mode=0555
file path=opt/onbld/bin/mkreadme_osol mode=0555
file path=opt/onbld/bin/mktpl mode=0555
file path=opt/onbld/bin/nightly mode=0555
file path=opt/onbld/bin/onu mode=0555
file path=opt/onbld/bin/protocmp.terse mode=0555
file path=opt/onbld/bin/sccscheck mode=0555
file path=opt/onbld/bin/signit mode=0555
file path=opt/onbld/bin/signproto mode=0555
file path=opt/onbld/bin/validate_flg mode=0555
file path=opt/onbld/bin/validate_paths mode=0555
file path=opt/onbld/bin/validate_pkg mode=0555
file path=opt/onbld/bin/wdiff mode=0555
file path=opt/onbld/bin/webrev mode=0555
file path=opt/onbld/bin/which_scm mode=0555
file path=opt/onbld/bin/ws mode=0555
```

```
124 file path=opt/onbld/bin/wsdiff mode=0555
125 file path=opt/onbld/bin/xref mode=0555
126 file path=opt/onbld/bin/xref.mk
127 file path=opt/onbld/env/developer
128 file path=opt/onbld/env/gatekeeper
129 file path=opt/onbld/env/illumos
130 file path=opt/onbld/etc/SampleLinks
131 file path=opt/onbld/etc/SamplePkgLinks
132 file path=opt/onbld/etc/exception_lists/check_rtime
133 file path=opt/onbld/etc/exception_lists/interface_check
134 file path=opt/onbld/etc/exception_lists/interface_cmp
135 file path=opt/onbld/etc/hgstyle
136 file path=opt/onbld/etc/its.conf
137 file path=opt/onbld/etc/its.reg
138 file path=opt/onbld/gk/.cshrc
139 file path=opt/onbld/gk/.login
140 file path=opt/onbld/gk/gen_make.machines mode=0755
141 file path=opt/onbld/lib/$(ARCH)/libdwarf.so.1
142 file path=opt/onbld/lib/perl/onbld_elfmod.pm
143 file path=opt/onbld/lib/perl/onbld_elfmod_vertype.pm
144 file path=opt/onbld/lib/python2.6/onbld/Checks/CStyle.py mode=0444
145 file path=opt/onbld/lib/python2.6/onbld/Checks/CStyle.pyc mode=0444
146 file path=opt/onbld/lib/python2.6/onbld/Checks/Cddl.py mode=0444
147 file path=opt/onbld/lib/python2.6/onbld/Checks/Cddl.pyc mode=0444
148 file path=opt/onbld/lib/python2.6/onbld/Checks/CmtBlk.py mode=0444
149 file path=opt/onbld/lib/python2.6/onbld/Checks/CmtBlk.pyc mode=0444
150 file path=opt/onbld/lib/python2.6/onbld/Checks/Comments.py mode=0444
151 file path=opt/onbld/lib/python2.6/onbld/Checks/Comments.pyc mode=0444
152 file path=opt/onbld/lib/python2.6/onbld/Checks/Copyright.py mode=0444
153 file path=opt/onbld/lib/python2.6/onbld/Checks/Copyright.pyc mode=0444
154 file path=opt/onbld/lib/python2.6/onbld/Checks/DbLookups.py mode=0444
155 file path=opt/onbld/lib/python2.6/onbld/Checks/DbLookups.pyc mode=0444
156 file path=opt/onbld/lib/python2.6/onbld/Checks/HdrChk.py mode=0444
157 file path=opt/onbld/lib/python2.6/onbld/Checks/HdrChk.pyc mode=0444
158 file path=opt/onbld/lib/python2.6/onbld/Checks/JStyle.py mode=0444
159 file path=opt/onbld/lib/python2.6/onbld/Checks/JStyle.pyc mode=0444
160 file path=opt/onbld/lib/python2.6/onbld/Checks/Keywords.py mode=0444
161 file path=opt/onbld/lib/python2.6/onbld/Checks/Keywords.pyc mode=0444
162 file path=opt/onbld/lib/python2.6/onbld/Checks/Mapfile.py mode=0444
163 file path=opt/onbld/lib/python2.6/onbld/Checks/Mapfile.pyc mode=0444
164 file path=opt/onbld/lib/python2.6/onbld/Checks/ProcessCheck.py mode=0444
165 file path=opt/onbld/lib/python2.6/onbld/Checks/ProcessCheck.pyc mode=0444
166 file path=opt/onbld/lib/python2.6/onbld/Checks/__init__.py mode=0444
167 file path=opt/onbld/lib/python2.6/onbld/Checks/__init__.pyc mode=0444
168 file path=opt/onbld/lib/python2.6/onbld/Scm/Backup.py mode=0444
169 file path=opt/onbld/lib/python2.6/onbld/Scm/Backup.pyc mode=0444
170 file path=opt/onbld/lib/python2.6/onbld/Scm/Version.py mode=0444
171 file path=opt/onbld/lib/python2.6/onbld/Scm/Version.pyc mode=0444
172 file path=opt/onbld/lib/python2.6/onbld/Scm/WorkSpace.py mode=0444
173 file path=opt/onbld/lib/python2.6/onbld/Scm/WorkSpace.pyc mode=0444
174 file path=opt/onbld/lib/python2.6/onbld/Scm/__init__.py mode=0444
175 file path=opt/onbld/lib/python2.6/onbld/Scm/__init__.pyc mode=0444
176 file path=opt/onbld/lib/python2.6/onbld/__init__.py mode=0444
177 file path=opt/onbld/lib/python2.6/onbld/__init__.pyc mode=0444
178 file path=opt/onbld/lib/python2.6/onbld/hgext/__init__.py mode=0444
179 file path=opt/onbld/lib/python2.6/onbld/hgext/__init__.pyc mode=0444
180 file path=opt/onbld/lib/python2.6/onbld/hgext/cdm.py mode=0444
181 file path=opt/onbld/man/man1/Install.1
182 file path=opt/onbld/man/man1/bldenv.1
183 file path=opt/onbld/man/man1/bringovercheck.1
184 file path=opt/onbld/man/man1/cddlchk.1
185 file path=opt/onbld/man/man1/check_rtime.1
186 file path=opt/onbld/man/man1/checkpaths.1
187 file path=opt/onbld/man/man1/codereview.1
188 file path=opt/onbld/man/man1/cstyle.1
189 file path=opt/onbld/man/man1/cw.1
```



```
190 file path=opt/onbld/man/man1/find_elf.1
191 file path=opt/onbld/man/man1/findunref.1
192 file path=opt/onbld/man/man1/flg.flp.1
193 file path=opt/onbld/man/man1/get_depend_info.1
194 file path=opt/onbld/man/man1/git-pbchk.1
195 file path=opt/onbld/man/man1/hdrchk.1
196 file path=opt/onbld/man/man1/hgsetup.1
197 file path=opt/onbld/man/man1/interface_check.1
198 file path=opt/onbld/man/man1/interface_cmp.1
199 file path=opt/onbld/man/man1/jstyle.1
200 file path=opt/onbld/man/man1/lintdump.1
201 file path=opt/onbld/man/man1/make_pkg_db.1
202 file path=opt/onbld/man/man1/mapfilechk.1
203 file path=opt/onbld/man/man1/ndrgen.1
204 file path=opt/onbld/man/man1/nightly.1
205 file path=opt/onbld/man/man1/onu.1
206 file path=opt/onbld/man/man1/sccscheck.1
211 file path=opt/onbld/man/man1/signit.1
212 file path=opt/onbld/man/man1/signproto.1
207 file path=opt/onbld/man/man1/webrev.1
208 file path=opt/onbld/man/man1/which_scm.1
209 file path=opt/onbld/man/man1/ws.1
210 file path=opt/onbld/man/man1/wsdiff.1
211 file path=opt/onbld/man/man1/xref.1
212 hardlink path=opt/onbld/bin/$(ARCH)/install.bin target=./install
213 legacy pkg=SUNWonbld desc="tools used to build the OS-Net consolidation" \
214     name="OS-Net Build Tools" version=11.11,REV=2009.10.22
215 license cr_Sun license=cr_Sun
216 license lic_CDDL license=lic_CDDL
217 license usr/src/tools/ctf/dwarf/THIRDPARTYLICENSE \
218     license=usr/src/tools/ctf/dwarf/THIRDPARTYLICENSE
219 license usr/src/tools/onbld/THIRDPARTYLICENSE \
220     license=usr/src/tools/onbld/THIRDPARTYLICENSE
221 link path=opt/onbld/bin/git-nits target=git-pbchk
222 link path=opt/onbld/lib/python target=python2.6
223 link path=opt/onbld/man/man1/git-nits.1 target=git-pbchk.1
224 # webrev(1) requires ps2pdf
225 depend fmri=print/filter/ghostscript type=require
226 # hgsetup(1) uses check-hostname(1) and nightly sendmail(1M)
227 depend fmri=service/network/smtp/sendmail type=require
228 # nightly(1) uses wget
229 depend fmri=web/wget type=require
```

```
**********************************************************
    2836 Fri Oct 11 17:19:06 2013
new/usr/src/tools/Makefile
4108 remove ON_CRYPTO_BINS from tools
**********************************************************
```

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

#
# Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
#

include ../Makefile.master

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

BOOT_SUBDIRS= \
        cw

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

#
#  special versions of commands for use only in build
#
UNSHIPPED_SUBDIRS = \
        elfsign

sparc_SUBDIRS= \
        chk4ubin \
        stabs \
        tokenize
```



```
i386_SUBDIRS=           \
        aw              \
        elfextract      \
        mbh_patch

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

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

include Makefile.tools

ROOTDIRS= \
        $(ROOTOPT) \
        $(ROOTONBLD) \
        $(ROOTONBLD)/bin \
        $(ROOTONBLD)/bin/$(MACH) \
        $(ROOTONBLD)/lib \
        $(ROOTONBLD)/lib/$(MACH) \
        $(ROOTONBLD)/lib/perl \
        $(ROOTONBLD)/lib/python2.6 \
        $(ROOTONBLD)/lib/python2.6/onbld \
        $(ROOTONBLD)/lib/python2.6/onbld/Checks \
        $(ROOTONBLD)/lib/python2.6/onbld/hgext \
        $(ROOTONBLD)/lib/python2.6/onbld/Scm \
        $(ROOTONBLD)/env \
        $(ROOTONBLD)/etc \
        $(ROOTONBLD)/etc/exception_lists \
        $(ROOTONBLD)/gk \
        $(ROOTONBLD)/man \
        $(ROOTONBLD)/man/man1

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

.KEEP_STATE:

#
# Only create directories in the tools proto area when doing an actual
# build, not a clean or clobber.
#
DOROOTDIRS= $(ROOTDIRS)
clobber:= DOROOTDIRS=
clean:= DOROOTDIRS=

all install: $(SUBDIRS)

clean: $(SUBDIRS)

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

```
128 lint: $(LINTSUBDIRS)

130 _msg: $(MSGSUBDIRS)

132 .PARALLEL: $(SUBDIRS) $(CLOSED_SUBDIRS)

134 $(SUBDIRS) $(CLOSED_SUBDIRS): $(BOOT_SUBDIRS)

136 $(BOOT_SUBDIRS) $(SUBDIRS): $$(DOROOTDIRS) $(ROOTONBLDLIBPY) FRC
137         @cd $@; pwd; $(MAKE) $(TARGET)

139 $(ROOTDIRS):
140         $(INS.dir)

142 $(ROOTONBLDLIBPY): $(ROOTDIRS)
143         $(RM) -r $@; $(SYMLINK) python2.6 $@

145 FRC:
```

```
**********************************************************
   11681 Fri Oct 11 17:19:07 2013
new/usr/src/tools/README.tools
4108 remove ON_CRYPTO_BINS from tools
**********************************************************
```

```
   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 (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.


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

  36 Layout of /opt/onbld
  37 --------------------

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

  43 /opt/onbld/gk
  44         gk account's home directory.

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

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

  52 /opt/onbld/env
  53         build environment files.

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

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

  61 /opt/onbld/lib/python<version>/onbld/hgext
```



```
  62         Mercurial extensions.

  64 /opt/onbld/lib/python/
  65         symlink to the modules directory of the currently preferred
  66         python version.  This exists to retain compatibility both for
  67         tools expecting only one supported version of python, and for
  68         user .hgrc files that expect to find cdm.py in
  69         /opt/onbld/lib/python/onbld/hgext.

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


  75 Tool Summary
  76 ------------

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

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

  91 cdm
  92         A Mercurial extension providing various commands useful for ON
  93         development

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

 104 checkproto
 105         Runs protocmp and protolist on a workspace (or uses the environment
 106         variable CODEMGR_WS to determine the workspace). Checks the proto area
 107         against the packages.

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

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

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

 117 cscope-fast
```

```
118         The fast version of cscope that we use internally. Seems to work,
119         but may need more testing before it's placed in the gate. The source
120         just really needs to be here.
121
122 cstyle
123         checks C source for compliance with OS/Net guidelines.

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

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

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

135 depcheck
136         A tool to try an assess the dependencies of executables.  This tool
137         is not a definitive dependency check, but it does use "strings" and
138         "ldd" to gather as much information as it can.  The dependency check
139         tool can handle filenames and pkgnames.  Before using the dependency
140         checker you must build a database which reflects the properties and
141         files in your system.

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

150 elfsign
151         Built from the same sources as the shipped elfsign(1), this
152         version is used in nightly -t builds to assure that the signing
153         process and format is the same as will be used on the target
154         system.

156 elfsigncmp
157         This script can be used in lieu of elfsign during a build.
158         It uses elfsign to sign a copy of the object and elfcmp -S to
159         verify that the signing caused no damage before updating
160         the object to be signed.
161
162 find_elf
163         Search a directory tree for ELF objects, and produce one line of
164         output per object. Used by check_rtime and interface_check to locate
165         the objects to examine.

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

179         $ findunref $INTELSRC $INTELSRC/tools/findunref/exception_list | \
180           sort > ~/unref-i386.out
181         $ findunref $SPARCSRC $SPARCSRC/tools/findunref/exception_list | \
182           sort > ~/unref-sparc.out
183         $ comm -12 ~/unref-i386.out ~/unref-sparc.out > ~/unref.out
```



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

189 hgsetup
190         creates a basic Mercurial configuration for the user.

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

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

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

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

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

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

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

231 pmodes
232         enforces proper file ownership and permissions in pkgmap and package
233         prototype* files.  converts files if necessary

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

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

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


247 ws
248         creates a shell with the environment set up to build in the given
249         workspace. Used mostly for non-full-build workspaces, so it sets up
```

```
250         to pull headers and libraries from the proto area of the parent if
251         they aren't in the childs proto area.

253 tokenize
254         Used to build the sun4u boot block.

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

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

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


274 How to do a full build
275 ----------------------

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

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

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

300 Files you have to update to add a tool
301 -------------------------------------

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

```
***********************************************************
    8085 Fri Oct 11 17:19:10 2013
new/usr/src/tools/env/developer.sh
4108 remove ON_CRYPTO_BINS from tools
***********************************************************
```

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

#
# Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
#

#       Configuration variables for the runtime environment of the nightly
# build script and other tools for construction and packaging of releases.
# This script is sourced by 'nightly' and 'bldenv' to set up the environment
# for the build. This example is suitable for building a developers workspace,
# which will contain the resulting packages and archives. It is based off
# the onnv release. It sets NIGHTLY_OPTIONS to make nightly do:
#       check ELF ABI/versioning (-A)
#       runs 'make check' (-C)
#       DEBUG and non-DEBUG builds (-D)
#       runs lint in usr/src (-l plus the LINTDIRS variable)
#       sends mail on completion (-m and the MAILTO variable)
#       creates packages for PIT/RE (-p)
#       checks for changes in ELF runpaths (-r)
#
NIGHTLY_OPTIONS="-ACDlmpr";             export NIGHTLY_OPTIONS

# This is a variable for the rest of the script - GATE doesn't matter to
# nightly itself
GATE=onnv-bugfixes;                     export GATE

# CODEMGR_WS - where is your workspace at (or what should nightly name it)
CODEMGR_WS="/builds/$GATE";                     export CODEMGR_WS

# PARENT_WS is used to determine the parent of this workspace. This is
# for the options that deal with the parent workspace (such as where the
# proto area will go).
#
# If you use this, it must be local (or nfs): nightly cannot copy
# over ssh or http.
PARENT_WS="/ws/onnv-gate";                      export PARENT_WS

# CLONE_WS is the workspace nightly should do a bringover from.
CLONE_WS="ssh://anonhg@onnv.sfbay.sun.com//export/onnv-clone";  export CLONE_WS

# CLOSED_CLONE_WS is the workspace from which nightly should acquire
# the usr/closed tree.
```



```
CLOSED_CLONE_WS="${CLONE_WS}/usr/closed";       export CLOSED_CLONE_WS

# This flag controls whether to build the closed source.  If
# undefined, nightly(1) and bldenv(1) will set it according to whether
# the closed source tree is present.  CLOSED_IS_PRESENT="no" means not
# building the closed sources.
# CLOSED_IS_PRESENT="yes";              export CLOSED_IS_PRESENT

# The bringover, if any, is done as STAFFER.
# Set STAFFER to your own login as gatekeeper or developer
# The point is to use group "staff" and avoid referencing the parent
# workspace as root.
# Some scripts optionally send mail messages to MAILTO.
#
STAFFER=nobody;                         export STAFFER
MAILTO=$STAFFER;                        export MAILTO

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

# You should not need to change the next four lines
LOCKNAME="`basename $CODEMGR_WS`_nightly.lock"; export LOCKNAME
ATLOG="$CODEMGR_WS/log";                        export ATLOG
LOGFILE="$ATLOG/nightly.log";                   export LOGFILE
MACH=`uname -p`;                                export MACH

# When the -A flag is specified, and ELF_DATA_BASELINE_DIR is defined,
# the ELF interface description file resulting from the build is compared
# to that from the specified directory. This ensures that our object
# versioning evolves in a backward compatible manner.
#
# You should not need to change this unless you wish to use locally cached
# baseline files. If you use this, it must be local (or nfs): nightly cannot
# copy over ssh or http.
#
ELF_DATA_BASELINE_DIR="/ws/onnv-gate/usr/src/ELF-data-baseline.$MACH";  export E

# This is usually just needed if the closed tree is missing, or when
# building a project gate with the -O (cap oh) flag.
# ON_CRYPTO_BINS="$PARENT_WS/packages/$MACH/on-crypto.$MACH.tar.bz2"
# export ON_CRYPTO_BINS

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

#
#       build environment variables, including version info for mcs, motd,
# motd, uname and boot messages. Mostly you shouldn't change this except
# when the release slips (nah) or you move an environment file to a new
# release
#
ROOT="$CODEMGR_WS/proto/root_${MACH}";  export ROOT
SRC="$CODEMGR_WS/usr/src";              export SRC
VERSION="$GATE";                        export VERSION

#
# the RELEASE and RELEASE_DATE variables are set in Makefile.master;
# there might be special reasons to override them here, but that
# should not be the case in general
#
# RELEASE="5.10.1";                     export RELEASE
# RELEASE_DATE="October 2007";          export RELEASE_DATE
```

```
124 # proto area in parent for optionally depositing a copy of headers and
125 # libraries corresponding to the protolibs target
126 # not applicable given the NIGHTLY_OPTIONS
127 #
128 PARENT_ROOT=$PARENT_WS/proto/root_$MACH; export PARENT_ROOT
129 PARENT_TOOLS_ROOT=$PARENT_WS/usr/src/tools/proto/root_$MACH-nd; export PARENT_TO

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

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

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

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

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

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

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

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

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



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

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

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

```
**********************************************************
    8698 Fri Oct 11 17:19:14 2013
new/usr/src/tools/env/gatekeeper.sh
4108 remove ON_CRYPTO_BINS from tools
**********************************************************
```

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

#
# Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
#

#       Configuration variables for the runtime environment of the nightly
# build script and other tools for construction and packaging of releases.
# This script is sourced by 'nightly' and 'bldenv' to set up the environment
# for the build. This example is suitable for building a gate,
# which will contain the resulting packages and archives (builds of the gate
# are done in children and then the resulting archives, packages, and proto
# area are put into the parent for everyone to use). It is based off
# the onnv release. It sets NIGHTLY_OPTIONS to make nightly do:
#       DEBUG and non-DEBUG builds (-D)
#       creates packages for PIT/RE (-p)
#       checks for new interfaces in libraries (-A)
#       runs 'make check' (-C)
#       runs lint in usr/src (-l plus the LINTDIRS variable)
#       sends mail on completion (-m and the MAILTO variable)
#       updates the protolist in the parent for children to compare with (-u)
#       updates the proto area in the parent when done (-U)
#       checks for changes in ELF runpaths (-r)
#       checks for changes in unreferenced files (-f)
#
NIGHTLY_OPTIONS="-ADClmpuUrf";          export NIGHTLY_OPTIONS

# This is a variable for the rest of the script - GATE doesn't matter to
# nightly itself
GATE=onnv-gate;                                 export GATE

# CODEMGR_WS - where is your workspace at (or what should nightly name it)
# there is only one definition here, which assumes all the gate build machines
# (sparc and x86) are set up the same. But remember, this is a script, so
# you _could_ look at $MACH or 'uname -n' and set these variables differently.
CODEMGR_WS="/builds/$GATE";                     export CODEMGR_WS

# PARENT_WS is used to determine the parent of this workspace. This is
# for the options that deal with the parent workspace (such as where the
# proto area will go).
#
# If you use this, it must be local (or nfs): nightly cannot copy
```



```
# over ssh or http.
PARENT_WS="/ws/$GATE";                          export PARENT_WS

# CLONE_WS is the workspace nightly should do a bringover from.
CLONE_WS="ssh://anonhg@onnv.sfbay.sun.com//export/onnv-clone";  export CLONE_WS

# CLOSED_CLONE_WS is the workspace from which nightly will acquire the
# usr/closed tree.
CLOSED_CLONE_WS="${CLONE_WS}/usr/closed"
export CLOSED_CLONE_WS

# This flag controls whether to build the closed source.  If
# undefined, nightly(1) and bldenv(1) will set it according to whether
# the closed source tree is present.  CLOSED_IS_PRESENT="no" means not
# building the closed sources.
# CLOSED_IS_PRESENT="yes";              export CLOSED_IS_PRESENT

# The bringover, if any, is done as STAFFER.
# Set STAFFER to your own login as gatekeeper or integration engineer.
# The point is to use group "staff" and avoid referencing the parent
# workspace as root.
# Some scripts optionally send mail messages to MAILTO.
#
STAFFER=nobody;                         export STAFFER
MAILTO=$STAFFER;                        export MAILTO

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

# You should not need to change the next four lines
LOCKNAME="`basename $CODEMGR_WS`_nightly.lock"; export LOCKNAME
ATLOG="$CODEMGR_WS/log";                        export ATLOG
LOGFILE="$ATLOG/nightly.log";                   export LOGFILE
MACH=`uname -p`;                                export MACH

# When the -A flag is specified, and ELF_DATA_BASELINE_DIR is defined,
# the ELF interface description file resulting from the build is compared
# to that from the specified directory. This ensures that our object
# versioning evolves in a backward compatible manner.
#
# You should not need to change this unless you wish to use locally cached
# baseline files. If you use this, it must be local (or nfs): nightly cannot
# copy over ssh or http.
#
ELF_DATA_BASELINE_DIR="/ws/onnv-gate/usr/src/ELF-data-baseline.$MACH";  export E

# This is usually just needed if the closed tree is missing, or when
# building a project gate with the -O (cap oh) flag.
# ON_CRYPTO_BINS="$PARENT_WS/packages/$MACH/on-crypto.$MACH.tar.bz2"
# export ON_CRYPTO_BINS

# REF_PROTO_LIST - for comparing the list of stuff in your proto area
# with. Generally this should be left alone, since you want to see differences
# between todays build and yesterdays.
#
REF_PROTO_LIST=$PARENT_WS/usr/src/proto_list_${MACH}; export REF_PROTO_LIST

#
#       build environment variables, including version info for mcs, motd,
# motd, uname and boot messages. Mostly you shouldn't change this except
# when the release slips (nah) or when starting a new release.
#
ROOT="$CODEMGR_WS/proto/root_${MACH}";  export ROOT
SRC="$CODEMGR_WS/usr/src";              export SRC
VERSION="$GATE";                        export VERSION
```

```
124 #
125 # the RELEASE and RELEASE_DATE variables are set in Makefile.master;
126 # there might be special reasons to override them here, but that
127 # should not be the case in general
128 #
129 # RELEASE="5.10.1";                     export RELEASE
130 # RELEASE_DATE="October 2007";          export RELEASE_DATE

132 # proto area in parent for optionally depositing a copy of headers and
133 # libraries corresponding to the protolibs target
134 #
135 PARENT_ROOT=$PARENT_WS/proto/root_$MACH; export PARENT_ROOT
136 PARENT_TOOLS_ROOT=$PARENT_WS/usr/src/tools/proto/root_$MACH-nd; export PARENT_TO

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


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

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

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

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

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

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



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

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

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

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

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

```
**********************************************************
    8296 Fri Oct 11 17:19:17 2013
new/usr/src/tools/env/illumos.sh
4108 remove ON_CRYPTO_BINS from tools
**********************************************************
______unchanged_portion_omitted_

  99 maxjobs DMAKE_MAX_JOBS # "DMAKE_MAX_JOBS" passed as ksh(1) name reference
 100 export DMAKE_MAX_JOBS

 102 # path to onbld tool binaries
 103 ONBLD_BIN='/opt/onbld/bin'

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

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

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

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

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

 130 # You should not need to change the next four lines
 131 export LOCKNAME="$(basename -- "$CODEMGR_WS")_nightly.lock"
 132 export ATLOG="$CODEMGR_WS/log"
 133 export LOGFILE="$ATLOG/nightly.log"
 134 export MACH="$(uname -p)"

 136 #
 137 #  The following macro is the closed binaries.  Once
 138 #  Illumos has totally freed itself, we can remove this reference.
 137 #  The following two macros are the closed/crypto binaries.  Once
 138 #  Illumos has totally freed itself, we can remove these references.
 139 #
 140 # Location of encumbered binaries.
 141 export ON_CLOSED_BINS="$CODEMGR_WS/closed"
 142 # Location of signed cryptographic binaries.
 143 export ON_CRYPTO_BINS="$CODEMGR_WS/on-crypto.$MACH.tar.bz2"

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


 150 export ROOT="$CODEMGR_WS/proto/root_${MACH}"
 151 export SRC="$CODEMGR_WS/usr/src"
 152 export MULTI_PROTO="no"
```



```
 154 #
 155 #       build environment variables, including version info for mcs, motd,
 156 # motd, uname and boot messages. Mostly you shouldn't change this except
 157 # when the release slips (nah) or you move an environment file to a new
 158 # release
 159 #
 160 export VERSION="$GATE"

 162 #
 163 # the RELEASE and RELEASE_DATE variables are set in Makefile.master;
 164 # there might be special reasons to override them here, but that
 165 # should not be the case in general
 166 #
 167 # export RELEASE='5.11'
 168 # export RELEASE_DATE='October 2007'

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

 177 # Package creation variables.  You probably shouldn't change these,
 178 # either.
 179 #
 180 # PKGARCHIVE determines where the repository will be created.
 181 #
 182 # PKGPUBLISHER_REDIST controls the publisher setting for the repository.
 183 #
 184 export PKGARCHIVE="${CODEMGR_WS}/packages/${MACH}/nightly"
 185 # export PKGPUBLISHER_REDIST='on-redist'

 187 # Package manifest format version.
 188 export PKGFMT_OUTPUT='v1'

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

 194 # Magic variable to prevent the devpro compilers/teamware from sending
 195 # mail back to devpro on every use.
 196 export UT_NO_USAGE_TRACKING='1'

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

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

 216 # Set this flag to 'n' to disable the automatic validation of the dmake
 217 # version in use.  The default is to check it.
 218 #CHECK_DMAKE='y'
```

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

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

228 # Uncomment this to disable support for SMB printing.
229 # export ENABLE_SMB_PRINTING='#'
```

```
**********************************************************
    9488 Fri Oct 11 17:19:21 2013
new/usr/src/tools/scripts/Install.1
4108 remove ON_CRYPTO_BINS from tools
**********************************************************
_____unchanged_portion_omitted_
```

```
  56 ]
  57 .RB [ " \-d "
  58 .IR "work dir" " ]"
  59 .br
  60 .RB [ " \-D "
  61 .IR "library dir" " ]"
  62 .RB [ " \-G "
  63 .IB glomname " ]"
  64 .RI [ " module ... " ]
  65 .LP
  66 or
  67 .LP
  68 .BR "Install \-R " "[ options ]"
  69 .SH DESCRIPTION
  70 .LP
  71 .B Install
  72 is a utility which simplifies the process of installing a 5.0 system.
  73 .B Install
  74 goes into a built ON workspace (or any kernel source tree),
  75 looks at the Makefiles,
  76 and figures out how to construct the /kernel and /usr/kernel directories.
  77 It then creates a tarfile
  78 .RB "(see " tar "(1))"
  79 containing /kernel, /usr/kernel, and a few related /etc files.  If a
  80 .I target ([user@]machine:/dir)
  81 is specified, the tarfile is either copied to
  82 .IR machine:/dir " (-T) or untarred on " "machine" " in " "/dir" " (-t),"
  83 using the remote user id
  84 .IR user ,
  85 if specified.
  86 With no options,
  87 .B Install
  88 creates a sun4c system from files in the current workspace (as indicated
  89 by $SRC) and places the tarfile in /tmp/Install.username/Install.sun4c.tar.

  91 .SH OPTIONS
  92 .TP 20n
  93 .BI "-w" " ws"
  94 Install the system built in the ON workspace
  95 .I ws.  ws
  96 must be a built ON workspace \(em
  97 .B Install
  98 will not automatically invoke
  99 .BR make "(1).  If " \-w " is not specified, " Install " uses the current
 100 workspace (as indicated by $CODEMGR_WS).  If there is no current workspace,
 101 .B Install
 102 checks to see if you are in an appropriate source directory, e.g. uts/sun4c;
 103 if so,
 104 .B Install
 105 takes files from there.  Otherwise,
 106 .B Install
 107 looks for files under $SRC/uts.
 108 .TP
 109 .BI "-s" " source directory"
 110 where to look for files [default: $SRC/uts].
 111 .TP
 112 .BI "-k" " kernel arch"
 113 the type of kernel to install.  The default is sun4c; however, if you invoke
 114 .B Install
 115 from $SRC/uts/sun4z,
```



```
 116 .B Install
 117 assumes you want a sun4z kernel.
 118 .TP
 119 .B "-n"
 120 No target; just create the tarfile in
 121 /tmp/Install.username/Install.sun4c.tar [default].
 122 .BR "-n" " implies " "-p" .
 123 .TP
 124 .BI "-t" " target"
 125 Install the system on
 126 .I target ([user@]machine:/dir).
 127 This means that kernel/unix is copied to
 128 .I machine:/dir/kernel/unix,
 129 etc.
 130 .IR /dir " is typically either " / " or " /mnt.
 131 .BR "-t" " implies " "-c" .
 132 The default remote user id is the same as the local one ($LOGNAME).
 133 .TP
 134 .BI "-T" " target"
 135 Copy the tarfile to
 136 .I target ([user@]machine:/dir).
 137 This creates the file
 138 .I /dir/Install.tar
 139 on
 140 .I machine.
 141 To finish the install, log on to
 142 .I machine
 143 as root, and type
 144 .RB ‘‘ "cd /; tar xvf /dir/Install.tar" "’’."
 145 .BR "-T" " implies " "-c" .
 146 .TP
 147 .B "-u"
 148 Install unix only.
 149 .TP
 150 .B "-m"
 151 Install modules only.
 152 .TP
 153 .B "-a"
 154 Install unix and all modules [default].
 155 .TP
 156 .B "-v"
 157 Verbose mode.
 158 .TP
 159 .B "-V"
 160 REALLY verbose mode.  Useful mainly for debugging.
 161 .TP
 162 .B "-q"
 163 Quiet mode [default].  Only fatal messages are printed.
 164 .TP
 165 .B "-c"
 166 Clean up.  After a successful install, delete the files created in
 167 /tmp/Install.username.  This is the default behavior if a
 168 .I target
 169 is specified with
 170 .BR "-t" " or " "-T" .
 171 .TP
 172 .B "-p"
 173 Preserve temp files.  This is the default behavior when no
 174 .I target
 175 is specified
 176 .RB ( "-n" ).
 177 .TP
 178 .B "-R"
 179 Recover from a failed
 180 .BR Install .
 181 This is not required, it’s just faster than restarting.
```

```
182 A typical scenario is for
183 .B Install
184 to run smoothly right up to the very end, but then die with
185 "Permission denied" when it tries to rsh/rcp to the target machine.
186 At this point, you log on to the target machine, diddle the permissions,
187 log off, and type
188 .RB ‘‘ "Install -R" "’’."
189 .B Install
190 will only have to retry the rsh/rcp,
191 rather than rebuild the tarfile from scratch.
192 .TP
193 .BI "-d" " temp directory"
194 specifies where
195 .B Install
196 should create its temp files [default: /tmp/Install.username].  This is
197 useful if you have limited space in /tmp (\fBInstall\fR can take as
198 much as 100MB).
199 The suffix "Install.username" is always appended.
200 .TP
201 .B "-L"
202 add a system to your library.  This allows you to build a personal
203 collection of installable systems from various environments and for
204 various architectures.  When you type
205 .RB ‘‘ "Install -w /ws/ws_name -k arch -L" "’’, " Install
206 creates a tarfile called
207 .I ws_name.arch.tar
208 in your library directory (~/LibInstall by default).
209 .BR "-L" " implies " "-c" .
210 .TP
211 .BI "-l" " library file"
212 Installs the system contained in
213 .I library file.
214 You may omit the ‘‘.tar’’ suffix.  For example,
215 .RB ‘‘ "Install -l my_ws.sun4c -t machine:/" ’’
216 installs a system you previously built with
217 .B "-L"
218 (from sun4c files in my_ws) on
219 .IR machine:/ .
220 This is equivalent to typing
221 .RB ‘‘ "rsh machine ’(cd /; tar xvf -)’ <~/LibInstall/my_ws.sun4c.tar" ’’,
222 but it’s easier to remember.
223 .TP
224 .BI "-D" " lib directory"
225 specifies the library directory [default: $HOME/LibInstall].
226 .TP
227 .BI "-G " glomname
228 gloms /kernel and /usr/kernel together into a single /kernel directory.
229 Useful for development work, e.g. use "Install -G good [...]" to create a
230 "/kernel.good".
231 .TP
232 .BR "-o " "{ \fBobj\fP | \fBdebug\fP }"
233 object directory. The default is "debug".
234 .TP
235 .B \-3
236 32-bit modules only
237 .TP
238 .B \-6
239 64-bit modules only
240 .TP
241 .B \-K
242 Do not include kmdb misc module or dmods
243 .TP
244 .B "-h"
245 Help.  Prints a brief summary of
246 .BR Install "’s"
247 options.
```



```
248 .LP
249 If you are in a directory like $SRC/uts/sun4z when you invoke
250 .BR Install ,
251 it will infer that you want to install a sun4z system
252 from the current workspace.
253 .LP
254 If you supply a list of modules, it overrides any of the
255 .B "-uma"
256 options.  You only need to specify the basename of the
257 module(s), e.g. ‘‘\fBInstall ufs nfs le\fR’’.
258 ‘‘\fBInstall unix\fR’’ is equivalent to ‘‘\fBInstall -u\fR’’, and
259 ‘‘\fBInstall modules\fR’’ is equivalent to ‘‘\fBInstall -m\fR’’.
260 .LP
261 You can customize
262 .B Install
263 by creating a .Installrc file in your home directory.  .Installrc
264 should consist of a list of command-line-style options, e.g:
265 .LP
266 .B
267         -w /ws/foo
268 .br
269 .B
270         -t labmachine:/mnt -pv
271 .LP
272 .B Install
273 processes default options first, then .Installrc
274 options, then command-line options.  In the case of
275 conflicting options (e.g. \fB-uma\fR), the last one wins.
276 .LP
277 In order to use the most convenient form of
278 .BR Install " (‘‘" "Install -t machine:/" "’’),"
279 you will need to do the following on the target machine:
280 .LP
281 .br
282         (1) add your machine name to the /etc/hosts.equiv file
283 .br
284         (2) add your username to the /etc/{passwd,shadow} files
285 .br
286         (3) chown -R yourself /kernel /usr/kernel
287 .br
288         (4) chmod -R u+w /kernel /usr/kernel
289 .SH "ENVIRONMENT"
290 .LP
291 You can set the following variables in your environment:
292 .LP
293 ON_CRYPTO_BINS
294 .IP
295 file containing signed cryptographic binaries.  This is only needed if
296 you are not building the closed-source tree.
297 .LP
293 INSTALL_RC [default: $HOME/.Installrc]
294 .IP
295 file containing default options for \fBInstall\fR
296 .LP
297 INSTALL_STATE [default: $HOME/.Install.state]
298 .IP
299 where \fBInstall\fR keeps its state information
300 .LP
301 INSTALL_DIR [default: /tmp/Install.username]
302 .IP
303 where \fBInstall\fR does its work.  This can be overridden on
304 the command line with \fB\-d\fR.
305 .LP
306 INSTALL_LIB [default: $HOME/LibInstall]
307 .IP
308 where \fBInstall\fR gets/puts library files.  This can be overridden on
```

```
 309 the command line with \fB\-D\fR.
 310 .LP
 311 INSTALL_CP [default: cp -p]
 312 .IP
 313 the command to copy files locally
 314 .LP
 315 INSTALL_RCP [default: rcp -p]
 316 .IP
 317 the command to copy files remotely
 318 .bp
 319 .SH "EXAMPLES"
 320 .LP
 321 .B
 322 Install -w /ws/blort -t machine:/
 323 .IP
 324 .RI "installs the system built in workspace " /ws/blort " on " machine:/
 325 .LP
 326 .B
 327 Install -w /ws/blort -T machine:/tmp
 328 .br
 329 .B
 330 rsh machine -l root "cd /; tar xvf /tmp/Install.tar"
 331 .IP
 332 is an equivalent way to do the previous example
 333 .LP
 334 .B Install
 335 .IP
 336 makes a tarfile containing a sun4c kernel,
 337 and places it in /tmp/Install.username/Install.sun4c.tar.  However, if you
 338 are in one of the arch directories (e.g. $SRC/uts/sun4m) when you invoke
 339 .BR Install ,
 340 you will get a tarfile for that architecture instead.
 341 .LP
 342 .B
 343 Install -k sun4m -w /ws/on493 -t mpbox:/ ufs
 344 .IP
 345 installs a new sun4m ufs module from workspace /ws/on493 on mpbox:/
 346 .SH "FILES"
 347 $HOME/.Installrc, $HOME/.Install.state
 348 .SH "SEE ALSO"
 349 .BR tar "(1), " rsh "(1), " rcp "(1)"
 350 .SH "BUGS"
 351 .BR tar "(1) and " rsh "(1)"
 352 do not have particularly useful exit codes.  To compensate,
 353 .B Install
 354 feeds stderr through grep -v and throws away error messages which it
 355 considers harmless.  If there's anything left,
 356 .B Install
 357 assumes it is fatal.  It's a hack, but it works.
```

```
**********************************************************
   22743 Fri Oct 11 17:19:25 2013
new/usr/src/tools/scripts/Install.sh
4108 remove ON_CRYPTO_BINS from tools
**********************************************************
______unchanged_portion_omitted_

338 #
339 # Unpack the crypto tarball into the given tree, then massage the
340 # tree so that the binaries are all in objNN or debugNN directories.
341 #
342 function unpack_crypto {
343         typeset tarfile=$1
344         typeset ctop=$2
345         [ -d "$ctop" ] || fail "Can't create tree for crypto modules."

347         [ "$VERBOSE" = "V" ] && echo "unpacking crypto tarball into $ctop..."
348         bzcat "$tarfile" | (cd "$ctop"; tar xf -)

350         typeset root="$ctop/proto/root_$MACH"
351         [ $OBJD = obj ] && root="$ctop/proto/root_$MACH-nd"
352         [ -d "$root" ] || fail "Can't unpack crypto tarball."

354         (cd "$root"; for d in platform kernel usr/kernel; do
355                 [ ! -d $d ] && continue
356                 find $d -type f -print
357         done) | while read file; do
358                 typeset dir=$(dirname "$file")
359                 typeset base=$(basename "$file")
360                 typeset type=$(basename "$dir")
361                 if [ "$type" = amd64 ]; then
362                         newdir="$dir/${OBJD}64"
363                 elif [ "$type" = sparcv9 ]; then
364                         newdir="$dir/${OBJD}64"
365                 else
366                         newdir="$dir/${OBJD}32"
367                 fi
368                 mkdir -p "$root/$newdir"
369                 [ "$VERBOSE" = "V" ] && echo "mv $file $newdir"
370                 mv "$root/$file" "$root/$newdir"
371         done
372 }

374 #
375 # usage: fixcrypto listfile ctop
376 # Massage entries in listfile for crypto modules, so that they point
377 # into ctop.
378 #
379 function fixcrypto {
380         typeset listfile=$1
381         typeset ctop=$2

383         typeset ccontents=/tmp/crypto-toc$$
384         find "$ctop" -type f -print > $ccontents
385         typeset root=root_$MACH
386         [ "$OBJD" = obj ] && root=root_$MACH-nd

388         grep -v ^MOD $listfile > $listfile.no-mod
389         grep ^MOD $listfile | while read tag srcdir module targdir size impl; do
390                 #
391                 # We don't just grep for ${OBJD}$size/$module because
392                 # there can be generic and platform-dependent versions
393                 # of a module.
394                 #
395                 newsrcfile=$(grep -w $root/$targdir/${OBJD}$size/$module $cconte
396                 if [ -n "$newsrcfile" ]; then
```



```
397                         # srcdir doesn't include final objNN or debugNN
398                         echo $tag $module $targdir $size $impl \
399                             $(dirname $(dirname "$newsrcfile"))
400                 else
401                         echo $tag $module $targdir $size $impl $srcdir
402                 fi
403         done > $listfile.mod
404         cat $listfile.mod $listfile.no-mod > $listfile

406         rm -f $listfile.mod
407         rm -f $listfile.no-mod
408         rm -f $ccontents
409 }

411 #
339 # Copy a module, or create a link, as needed.
340 #

342 function copymod {
343         case $1 in
344         MOD)
345                 targdir=$INSTALL_FILES/$4
346                 tstmkdir $targdir
347                 target=$targdir/$3
348                 verbose "$INSTALL_CP $2/${OBJD}$5/$3 $target"
349                 $INSTALL_CP $2/${OBJD}$5/$3 $target || \
350                     fail "can't create $target"
351                 ;;
352         SYMLINK)
353                 targdir=$INSTALL_FILES/$4
354                 tstmkdir $targdir
355                 target=$targdir/$5
356                 rm -f $target
357                 verbose "ln -s $3 $target"
358                 ln -s $3 $target || fail "can't create $target"
359                 ;;
360         LINK)
361                 targdir=$INSTALL_FILES/$5
362                 tstmkdir $targdir
363                 target=$targdir/$6
364                 rm -f $target
365                 verbose "ln $INSTALL_FILES/$3/$4 $target"
366                 ln $INSTALL_FILES/$3/$4 $target || fail "can't create $target"
367                 ;;
368         CONF)
369                 target=$INSTALL_FILES/$3
370                 tstmkdir `dirname $target`
371                 conffile=`basename $3`
372                 verbose "$INSTALL_CP $4/$conffile $target"
373                 $INSTALL_CP $4/$conffile $target
374                 ;;
375         *)
376                 fail "unrecognized modlist entry: $*"
377                 ;;
378         esac
379 }
______unchanged_portion_omitted_

405 #
406 # Copy kernel modules to $INSTALL_DIR
407 #

409 function copy_kernel {

411         case $KARCH in
412                 sun4*)          ISA=sparc;      MACH=sparc      ;;
```

```
413                 i86*)           ISA=intel;      MACH=i386       ;;
414                 *)              fail "${KARCH}: invalid kernel architecture";;
415         esac
416         export MACH

418         if [ "$GLOM" = "no" ]; then
419                 verbose "Source = $UTS, ISA = $ISA, kernel = $KARCH"
420         else
421                 verbose "Source = $UTS, ISA = $ISA, kernel = $KARCH, impl = $IMP
422         fi

424         test -d $KARCH || fail "${KARCH}: invalid kernel architecture"
425         test -d $ISA || fail "${ISA}: invalid instruction set architecture"

427         tstmkdir $INSTALL_FILES
428         rm -rf $modstatedir
429         tstmkdir $modstatedir
430         export MODSTATE=$modstatedir/state

432         #
433         # Figure out which "make" to use.  dmake is faster than serial
434         # make, but dmake 7.3 has a bug that causes it to lose log
435         # output, which means the modlist might be incomplete.
436         #
437         make=dmake
438         dmvers=‘$make -version‘
439         if [ $? -ne 0 ]; then
440                 make=/usr/ccs/bin/make
441         elif [[ $dmvers = *Distributed?Make?7.3* ]]; then
442                 unset make
443                 searchpath="/ws/onnv-tools/SUNWspro/SOS10/bin
444                         /opt/SUNWspro/SOS10/bin
445                         /opt/SUNWspro/bin"
446                 for dmpath in $searchpath; do
447                         verbose "Trying $dmpath/dmake"
448                         if [ -x $dmpath/dmake ]; then
449                                 dmvers=‘$dmpath/dmake -version‘
450                                 if [[ $dmvers != *Distributed?Make?7.3* ]]; then
451                                         make="$dmpath/dmake"
452                                         break;
453                                 fi
454                         fi
455                 done
456                 if [ -z $make ]; then
457                         make=/usr/ccs/bin/make
458                         echo "Warning: dmake 7.3 doesn’t work with Install;" \
459                                 "using $make"
460                 fi
461         fi

463         #
464         # Get a list of all modules, configuration files, and links
465         # that we might want to install.
466         #
467         verbose "Building module list..."
468         (cd $KARCH; MAKEFLAGS=e $make -K $MODSTATE modlist.karch) | \
469             egrep "^MOD|^CONF|^LINK|^SYMLINK" > $modlist
470         [ "$VERBOSE" = "V" ] && cat $modlist
471         check_modlist $modlist
545         if [ -n "$ON_CRYPTO_BINS" ]; then
546                 cryptotar="$ON_CRYPTO_BINS"
547                 if [ "$OBJD" = obj ]; then
548                         isa=$(uname -p)
549                         cryptotar=$(echo "$ON_CRYPTO_BINS" |
550                             sed -e s/.$isa.tar.bz2/-nd.$isa.tar.bz2/)
551                 fi
```



```
552                 [ -f "$cryptotar" ] || fail "crypto ($cryptotar) doesn’t exist"
553                 cryptotree=$(mktemp -d /tmp/crypto.XXXXXX)
554                 [ -n "$cryptotree" ] || fail "can’t create tree for crypto"
555                 unpack_crypto "$cryptotar" "$cryptotree"
556                 #
557                 # fixcrypto must come before fixglom, because
558                 # fixcrypto uses the unglommed path to find things in
559                 # the unpacked crypto.
560                 #
561                 fixcrypto $modlist "$cryptotree"
562         fi
472         if [ "$GLOM" = "yes" ]; then
473                 fixglom $modlist $GLOMNAME
474                 filtimpl $modlist $IMPL
475         fi
476         if [[ -n "$files" && "$files" != All ]]; then
477                 filtmod $modlist "$files"
478         fi

480         #
481         # Copy modules and create links.  For architectures with both
482         # 32- and 64-bit modules, we’ll likely have duplicate
483         # configuration files, so do those after filtering out the
484         # duplicates.
485         #
486         verbose "Copying files to ${INSTALL_FILES}..."

488         #
489         # The IFS is reset to the newline character so we can buffer the
490         # output of grep without piping it directly to copymod, otherwise
491         # if fail() is called, then it will deadlock in fail()’s wait call
492         #
493         OIFS="$IFS"
494         IFS="
495         "
496         set -- ‘grep -v "^CONF" $modlist‘;
497         IFS="$OIFS"
498         for onemod in "$@"; do
499                 copymod $onemod
500         done
501
502         OIFS="$IFS"
503         IFS="
504         "
505         set -- ‘grep "^CONF" $modlist | sort | uniq‘;
506         IFS="$OIFS"
507         for onemod in "$@"; do
508                 copymod $onemod
509         done

511         #
512         # Add the glommed kernel name to the root archive
513         #
514         if [[ $GLOM == "yes" ]];
515         then
516                 filelist="$INSTALL_FILES/etc/boot/solaris/filelist.ramdisk"
517                 mkdir -p ‘dirname $filelist‘
518                 echo "platform/$KARCH/$GLOMNAME" >$filelist
519         fi

521         STATE=1 # all kernel modules copied correctly
522         save_state
523 }
```

unchanged_portion_omitted_

```
599 function copy_kmdb {
```

```
600         typeset kmdbtgtdir=$INSTALL_FILES/platform/$KARCH/$GLOMNAME/misc
601         typeset bitdirs=
602         typeset isadir=
603         typeset b64srcdir=
604         typeset b64tgtdir=
605         typeset b32srcdir=
606         typeset b32tgtdir=
607         typeset machdir=
608         typeset platdir=

610         if [[ $KMDB = "no" || ! -d $SRC/cmd/mdb ]] ; then
611                 # The kmdb copy was suppressed or the workspace doesn't contain
612                 # the mdb subtree.  Either way, there's nothing to do.
613                 STATE=2
614                 save_state
615                 return
616         fi

618         if [[ $(mach) = "i386" ]] ; then
619                 isadir="intel"
620                 b64srcdir="amd64"
621                 b64tgtdir="amd64"
622                 b32srcdir="ia32"
623                 b32tgtdir="."
624         else
625                 isadir="sparc"
626                 b64srcdir="v9"
627                 b64tgtdir="sparcv9"
628                 b32srcdir="v7"
629                 b32tgtdir="."
630         fi

632         typeset foundkmdb=no
633         typeset kmdbpath=
634         typeset destdir=

636         platdir=$INSTALL_FILES/platform/$KARCH/$GLOMNAME
637         if [[ $GLOM = "yes" ]] ; then
638                 machdir=$platdir
639         else
640                 machdir=$INSTALL_FILES/kernel
641         fi

643         srctrees=$SRC
644         if [[ -d $SRC/../closed && "$CLOSED_IS_PRESENT" != no ]]; then
645                 srctrees="$srctrees $SRC/../closed"
737         else
738                 if [ -z "$ON_CRYPTO_BINS" ]; then
739                         echo "Warning: ON_CRYPTO_BINS not set; pre-signed" \
740                             "crypto not provided."
646         fi
742         fi
647         if [[ $WANT64 = "yes" ]] ; then
648                 # kmdbmod for sparc and x86 are built and installed
649                 # in different places
650                 if [[ $(mach) = "i386" ]] ; then
651                         kmdbpath=$SRC/cmd/mdb/$isadir/$b64srcdir/kmdb/kmdbmod
652                         destdir=$machdir/misc/$b64tgtdir
653                 else
654                         kmdbpath=$SRC/cmd/mdb/$KARCH/$b64srcdir/kmdb/kmdbmod
655                         destdir=$platdir/misc/$b64tgtdir
656                 fi

658                 if kmdb_copy_kmdbmod $kmdbpath $destdir ; then
659                         foundkmdb="yes"
```



```
661                         for tree in $srctrees; do
662                                 kmdb_copy_machkmods \
663                                     $tree/cmd/mdb/$isadir/$b64srcdir \
664                                     $machdir/kmdb/$b64tgtdir
665                                 kmdb_copy_karchkmods $tree/cmd/mdb/$KARCH \
666                                     $platdir/kmdb/$b64tgtdir $b64srcdir
667                         done
668                 fi
669         fi

671         if [[ $WANT32 = "yes" ]] ; then
672                 kmdbpath=$SRC/cmd/mdb/$isadir/$b32srcdir/kmdb/kmdbmod
673                 destdir=$machdir/misc/$b32tgtdir

675                 if kmdb_copy_kmdbmod $kmdbpath $destdir ; then
676                         foundkmdb="yes"

678                         for tree in $srctrees; do
679                                 kmdb_copy_machkmods \
680                                     $tree/cmd/mdb/$isadir/$b32srcdir \
681                                     $machdir/kmdb/$b32tgtdir
682                                 kmdb_copy_karchkmods $tree/cmd/mdb/$KARCH \
683                                     $platdir/kmdb/$b32tgtdir $b32srcdir
684                         done
685                 fi
686         fi

688         # A kmdb-less workspace isn't fatal, but it is potentially problematic,
689         # as the changes made to uts may have altered something upon which kmdb
690         # depends.  We will therefore remind the user that they haven't built it
691         # yet.
692         if [[ $foundkmdb != "yes" ]] ; then
693                 echo "WARNING: kmdb isn't built, and won't be included"
694         fi

696         STATE=2
697         save_state
698         return
699 }
______unchanged_portion_omitted_

773 function okexit {
774         cd /tmp
775         test "$CLEANUP" = c && remove_dir $INSTALL_DIR
776         save_state
777         rm -rf $modstatedir
778         rm -f $modlist
875         [ -n "$cryptotree" ] && rm -rf "$cryptotree"
779         verbose "Install complete"
780         exit 0
781 }
______unchanged_portion_omitted_
```

```
**********************************************************
    3621 Fri Oct 11 17:19:29 2013
new/usr/src/tools/scripts/Makefile
4108 remove ON_CRYPTO_BINS from tools
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
#
# Copyright 2010, Richard Lowe

SHELL=/usr/bin/ksh93

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

PERLFILES= \
        check_rtime \
        find_elf \
        interface_check \
        interface_cmp \
        jstyle \
        mkreadme_osol \
        mktpl \
        validate_flg \
        validate_paths \
        wdiff
```



```
PERLMODULES= \
        onbld_elfmod.pm \
        onbld_elfmod_vertype.pm


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

SCRIPTLINKS= \
        git-nits

MAN1FILES= \
        Install.1 \
        bldenv.1 \
        bringovercheck.1 \
        cddlchk.1 \
        checkpaths.1 \
        check_rtime.1 \
        cstyle.1 \
        find_elf.1 \
        flg.flp.1 \
        git-pbchk.1 \
        hdrchk.1 \
        interface_check.1 \
        interface_cmp.1 \
        hgsetup.1 \
        jstyle.1 \
        mapfilechk.1 \
        nightly.1 \
        onu.1 \
        sccscheck.1 \
        webrev.1 \
        which_scm.1 \
        ws.1 \
        wsdiff.1 \
        xref.1

MAN1LINKS= \
        git-nits.1

MAKEFILES= \
        xref.mk

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

EXCEPTFILES= \
        check_rtime \
        interface_check \
        interface_cmp

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

include ../Makefile.tools

ROOTONBLDSCRIPTLINKS = $(SCRIPTLINKS:%=$(ROOTONBLDBIN)/%)
```

```
127 ROOTONBLDMAN1LINKS = $(MAN1LINKS:%=$(ROOTONBLDMAN1)/%)

129 $(ROOTONBLDETCFILES)    := FILEMODE=    644
130 $(ROOTONBLDEXCEPTFILES) := FILEMODE=    644
131 $(ROOTONBLDPERLMODULES) := FILEMODE=    644
132 $(ROOTONBLDMAKEFILES)   := FILEMODE=    644
133 $(ROOTONBLDMAN1FILES)   := FILEMODE=    644

135 .KEEP_STATE:

137 all:    $(SHFILES) $(PERLFILES) $(PERLMODULES) $(PYFILES) \
138         $(MAN1FILES) $(MAKEFILES)

140 $(ROOTONBLDBIN)/git-nits:
141         $(RM) $(ROOTONBLDBIN)/git-nits
142         $(SYMLINK) git-pbchk $(ROOTONBLDBIN)/git-nits

144 $(ROOTONBLDMAN1)/git-nits.1:
145         $(RM) $(ROOTONBLDMAN1)/git-nits.1
146         $(SYMLINK) git-pbchk.1 $(ROOTONBLDMAN1)/git-nits.1

148 install: all .WAIT $(ROOTONBLDSHFILES) $(ROOTONBLDPERLFILES)    \
149                 $(ROOTONBLDPERLMODULES) $(ROOTONBLDPYFILES)     \
150                 $(ROOTONBLDSCRIPTLINKS) $(ROOTONBLDMAN1FILES)   \
151                 $(ROOTONBLDMAKEFILES) $(ROOTONBLDETCFILES)      \
152                 $(ROOTONBLDEXCEPTFILES) $(ROOTONBLDMAN1LINKS)

154 clean:
155         $(RM) $(CLEANFILES)

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

166 bldenv.1: bldenv
167         $(RM) "$@"
168         (set +o errexit ; ksh93 $? --nroff ; true) 2>&1 | \
169         sed 's/\.DS/.nf/g;s/\.DE/.fi/' > "$@"

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

176 include ../Makefile.targ
```

```
**********************************************************
    7129 Fri Oct 11 17:19:32 2013
new/usr/src/tools/scripts/bindrop.sh
4108 remove ON_CRYPTO_BINS from tools
**********************************************************
```

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

#
# Copyright (c) 2007, 2010, Oracle and/or its affiliates. All rights reserved.
#
# Create an encumbered binaries tarball from a full build proto area,
# less the contents of an OpenSolaris proto area.
# less the contents of an OpenSolaris proto area.  Special handling
# for crypto binaries that need to be signed by Sun Release
# Engineering.
#

usage="bindrop [-n] basename"

isa=‘uname -p‘

PATH="$PATH:/usr/bin:/usr/sfw/bin"

function fail {
        print -u2 "bindrop: $@"
        exit 1
}
```

*unchanged_portion_omitted_*

```
nondebug=n
while getopts n flag; do
        case $flag in
        n)
                nondebug=y
                ;;
        ?)
                print -u2 "usage: $usage"
                exit 1
                ;;
        esac
done
shift $(($OPTIND - 1))

if [[ $# -ne 1 ]]; then
        print -u2 "usage: $usage"
        exit 1
fi

tarfile="$CODEMGR_WS/$1.$isa.tar"

rootdir="root_$isa"
suffix=""
if [[ "$nondebug" = y ]]; then
        rootdir="root_$isa-nd"
        suffix="-nd"
fi

[[ -d "${ROOT}${suffix}-closed" ]] || fail "can’t find closed root proto area."

tmpdir=$(mktemp -dt bindropXXXXX)
[[ -n "$tmpdir" ]] || fail "can’t create temporary directory."
mkdir -p "$tmpdir/closed/$rootdir" || exit 1

#
# This will hold a temp list of directories that must be kept, even if
# empty.
#
needdirs=$(mktemp -t needdirsXXXXX)
[[ -n "$needdirs" ]] || fail "can’t create temporary directory list file."

#
# Copy the closed root parallel tree into a temp directory.
#
(cd "${ROOT}${suffix}-closed"; tar cf - .) | (cd "$tmpdir/closed/$rootdir"; tar

#
# Remove internal ON crypto signing certs
#
delete="
        etc/certs/SUNWosnetSE
        etc/certs/SUNWosnetSolaris
        etc/crypto/certs/SUNWosnet
        etc/crypto/certs/SUNWosnetLimited
        etc/crypto/certs/SUNWosnetCF
        etc/crypto/certs/SUNWosnetCFLimited
        "

#
# Remove miscellaneous files that we don’t want to ship.
#

# SUNWsvvs (SVVS test drivers).
delete="
delete="$delete
        usr/include/sys/svvslo.h
        usr/include/sys/tidg.h
        usr/include/sys/tivc.h
        usr/include/sys/tmux.h
        usr/kernel/drv/amd64/svvslo
        usr/kernel/drv/amd64/tidg
        usr/kernel/drv/amd64/tivc
        usr/kernel/drv/amd64/tmux
        usr/kernel/drv/sparcv9/svvslo
        usr/kernel/drv/sparcv9/tidg
        usr/kernel/drv/sparcv9/tivc
        usr/kernel/drv/sparcv9/tmux
        usr/kernel/drv/svvslo
        usr/kernel/drv/svvslo.conf
        usr/kernel/drv/tidg
        usr/kernel/drv/tidg.conf
        usr/kernel/drv/tivc
        usr/kernel/drv/tivc.conf
```

```
134         usr/kernel/drv/tmux
135         usr/kernel/drv/tmux.conf
136         usr/kernel/strmod/amd64/lmodb
137         usr/kernel/strmod/amd64/lmode
138         usr/kernel/strmod/amd64/lmodr
139         usr/kernel/strmod/amd64/lmodt
140         usr/kernel/strmod/lmodb
141         usr/kernel/strmod/lmode
142         usr/kernel/strmod/lmodr
143         usr/kernel/strmod/lmodt
144         usr/kernel/strmod/sparcv9/lmodb
145         usr/kernel/strmod/sparcv9/lmode
146         usr/kernel/strmod/sparcv9/lmodr
147         usr/kernel/strmod/sparcv9/lmodt
148 "
149 # encumbered binaries and associated files
150 delete="$delete
151         kernel/drv/amd64/bmc
152         kernel/drv/bmc
153         kernel/drv/bmc.conf
154         kernel/drv/ifp.conf
155         kernel/drv/sparcv9/ifp
156         kernel/drv/sparcv9/isp
157         kernel/drv/sparcv9/qus
158         kernel/drv/spwr
159         kernel/kmdb/sparcv9/isp
160         usr/has/bin/ksh
161         usr/has/bin/pfksh
162         usr/has/bin/rksh
163         usr/include/sys/scsi/adapters/ifpcmd.h
164         usr/include/sys/scsi/adapters/ifpio.h
165         usr/include/sys/scsi/adapters/ifpmail.h
166         usr/include/sys/scsi/adapters/ifpreg.h
167         usr/include/sys/scsi/adapters/ifpvar.h
168         usr/include/sys/scsi/adapters/ispcmd.h
169         usr/include/sys/scsi/adapters/ispmail.h
170         usr/include/sys/scsi/adapters/ispreg.h
171         usr/include/sys/scsi/adapters/ispvar.h
172         usr/lib/mdb/kvm/sparcv9/isp.so
173         usr/platform/sun4u/include/sys/memtestio.h
174         usr/platform/sun4u/include/sys/memtestio_ch.h
175         usr/platform/sun4u/include/sys/memtestio_chp.h
176         usr/platform/sun4u/include/sys/memtestio_ja.h
177         usr/platform/sun4u/include/sys/memtestio_jg.h
178         usr/platform/sun4u/include/sys/memtestio_sf.h
179         usr/platform/sun4u/include/sys/memtestio_sr.h
180         usr/platform/sun4u/include/sys/memtestio_u.h
181         usr/platform/sun4u/include/sys/memtestio_pn.h
182         usr/platform/sun4v/include/sys/memtestio.h
183         usr/platform/sun4v/include/sys/memtestio_ni.h
184         usr/platform/sun4v/include/sys/memtestio_v.h
185 "
186 # memory fault injector test framework
187 delete="$delete
188         usr/bin/mtst
189         platform/sun4u/kernel/drv/sparcv9/memtest
190         platform/sun4u/kernel/drv/memtest.conf
191         platform/sun4v/kernel/drv/sparcv9/memtest
192         platform/sun4v/kernel/drv/memtest.conf
193         kernel/drv/memtest.conf
194         kernel/drv/memtest
195         kernel/drv/amd64/memtest
196         usr/platform/i86pc/lib/mtst/mtst_AuthenticAMD_15.so
197         usr/platform/i86pc/lib/mtst/mtst_AuthenticAMD.so
198         usr/platform/i86pc/lib/mtst/mtst_generic.so
199         usr/platform/i86pc/lib/mtst/mtst_GenuineIntel.so
```



```
200 "
201 for f in $delete; do
202         rm -rf "$tmpdir/closed/$rootdir/$f"
203 done

205 #
206 # Remove any header files.  If they're in the closed tree, they're
207 # probably not freely redistributable.
208 #
209 (cd "$tmpdir/closed/$rootdir"; find . -name \*.h \
210         -a \! -name lc_core.h \
211         -a \! -name localedef.h \
212         -exec rm -f {} \;)

214 #
215 # Remove empty directories that the open tree doesn't need.
216 #
217 # Step 1: walk the redistributable manifests to find out which directories
218 # are specified in the open packages; save that list to a temporary
219 # file $needdirs.
220 #
221 MACH=$(uname -p)
222 (cd "$SRC/pkg/packages.$MACH"; \
223         nawk '/^dir/ {sub(/.+ path=/, ""); print $1;}' *.metadata.*.redist | \
224         sort -u > "$needdirs")

226 #
227 # Step 2: go to our closed directory, and find all the subdirectories,
228 # filtering out the ones needed by the open packages (saved in that
229 # temporary file).  Sort in reverse order, so that parent directories
230 # come after any subdirectories, and pipe that to rmdir.  If there are
231 # still any lingering files, rmdir will complain.  That's fine--we
232 # only want to delete empty directories--so redirect the complaints to
233 # /dev/null.
234 #
235 (cd "$tmpdir/closed/$rootdir"; \
236         find * -type d -print | /usr/xpg4/bin/grep -xv -f $needdirs | \
237         sort -r | xargs -l rmdir 2>/dev/null )

239 rm "$needdirs"

241 #
256 # Exclude signed crypto binaries; they are delivered in their
257 # own tarball.
258 #
259 ROOT="$tmpdir/closed/$rootdir" findcrypto "$SRC/tools/codesign/creds" |
260     awk '{ print $2 }' | (cd "$tmpdir/closed/$rootdir"; xargs rm -f)

262 #
242 # Add binary license files.
243 #

245 cp -p "$SRC/tools/opensolaris/BINARYLICENSE.txt" "$tmpdir/closed" || \
246     fail "can't add BINARYLICENSE.txt"
247 mkreadme "$tmpdir/closed"
248 if [ -f "$CODEMGR_WS/THIRDPARTYLICENSE.ON-BINARIES" ]; then
249     cp -p "$CODEMGR_WS/THIRDPARTYLICENSE.ON-BINARIES" "$tmpdir/closed"
250 fi

252 (cd "$tmpdir"; tar cf "$tarfile" closed) || fail "can't create $tarfile."
253 bzip2 -f "$tarfile" || fail "can't compress $tarfile".

255 rm -rf "$tmpdir"

257 exit 0
```

```
**********************************************************
   18242 Fri Oct 11 17:19:35 2013
new/usr/src/tools/scripts/nightly.1
4108 remove ON_CRYPTO_BINS from tools
**********************************************************
```

```
.\" "
.\" " The contents of this file are subject to the terms of the
.\" " Common Development and Distribution License (the "License").
.\" " You may not use this file except in compliance with the License.
.\" "
.\" " You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
.\" " or http://www.opensolaris.org/os/licensing.
.\" " See the License for the specific language governing permissions
.\" " and limitations under the License.
.\" "
.\" " When distributing Covered Code, include this CDDL HEADER in each
.\" " file and include the License file at usr/src/OPENSOLARIS.LICENSE.
.\" " If applicable, add the following below this CDDL HEADER, with the
.\" " fields enclosed by brackets "[]" replaced with your own identifying
.\" " information: Portions Copyright [yyyy] [name of copyright owner]
.\" "
.\" " CDDL HEADER END
.\" "
.\" "Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved
.\" "Copyright 2012 Joshua M. Clulow <josh@sysmgr.org>
.\" "
.TH nightly 1 "6 July 2010"
.SH NAME
.I nightly
\- build an OS-Net consolidation overnight
.SH SYNOPSIS
\fBnightly [-in] [-V VERS] <env_file>\fP
.LP
.SH DESCRIPTION
.IX "OS-Net build tools" "nightly" "" "\fBnightly\fP"
.LP
.I nightly,
the mother of all build scripts,
can bringover, build, archive, package, error check, and
generally do everything it takes to
turn OS/Net consolidation source code into useful stuff.
It is customizable to permit you to run anything from a
simple build to all of the cross-checking a gatekeeper
needs.  The advantage to using
.I nightly
is that you build things correctly, consistently and
automatically, with the best practices; building with
.I nightly
can mean never having to say you’re sorry to your
gatekeeper.
.LP
More
specifically,
.I nightly
performs the following tasks, in order, if
all these things are desired:
.LP
.RS
.TP
\(bu
perform a "make clobber" to clean up old binaries
.TP
\(bu
bringover from the identified parent gate/clone
.TP
\(bu
```



```
perform non-DEBUG and DEBUG builds
.TP
\(bu
list proto area files and compare with previous list
.TP
\(bu
copy updated proto area to parent
.TP
\(bu
list shared lib interface and compare with previous list
.TP
\(bu
perform a "make lint" of the kernel and report errors
.TP
\(bu
perform a "make check" to report hdrchk/cstyle errors
.TP
\(bu
report the presence of any core files
.TP
\(bu
check the ELF runtime attributes of all dynamic objects
.TP
\(bu
check for unreferenced files
.TP
\(bu
report on which proto area objects have changed (since the last build)
.TP
\(bu
report the total build time
.TP
\(bu
save a detailed log file for reference
.TP
\(bu
mail the user a summary of the completed build
.RE
.LP
The actions of the script are almost completely determined by
the environment variables in the
.I env
file, the only necessary argument.  Ths only thing you really
need to use
.I nightly
is an
.I env
file that does what you want.
.LP
Like most of the other build tools in usr/src/tools, this script tends
to change on a fairly regular basis; do not expect to be able to build
OS/Net with a version of nightly significantly older than your source
tree.  It has what is effectively a Consolidation Private relationship
to other build tools and with many parts of the OS/Net makefiles,
although it may also be used to build other consolidations.
.LP
.SH NIGHTLY_OPTIONS
The environment variable NIGHTLY_OPTIONS controls the actions
.I nightly
will take as it proceeds.
The -i, -n, +t and -V options may also be used from the command
line to control the actions without editing your environment file.
The -i and -n options complete the build more quickly by bypassing
some actions. If NIGHTLY_OPTIONS is not set, then "-Bmt" build
options will be used.
```

```
.B Basic action options
.TP 10
.B \-D
Do a build with DEBUG on (non-DEBUG is built by default)
.TP
.B \-F
Do _not_ do a non-DEBUG build (use with -D to get just a DEBUG build)
.TP
.B \-M
Do not run pmodes (safe file permission checker)
.TP
.B \-i
Do an incremental build, suppressing the "make clobber" that by
default removes all existing binaries and derived files.  From the
command line, -i also suppresses the lint pass and the cstyle/hdrchk
pass
.TP
.B \-n
Suppress the bringover so that the build will start immediately with
current source code
.TP
.B \-o
Do an "old style" (pre-S10) build using root privileges to set OWNER
and GROUP from the Makefiles.
.TP
.B \-p
Create packages for regular install
.TP
.B \-U
Update proto area in the parent workspace
.TP
.B \-u
Update the parent workspace with files generated by the build, as follows.
.RS
.TP
\(bu
Copy proto_list_${MACH} and friends to usr/src in the parent.
.TP
\(bu
When used with -f, build a usr/src/unrefmaster.out in
the parent by merging all the usr/src/unref-${MACH}.out files in the
parent.
.TP
\(bu
When used with -A or -r, copy the contents of the resulting
ELF-data.${MACH} directory to usr/src/ELF-data.${MACH} in the parent
workspace.
.RE
.TP
.B \-m
Send mail to $MAILTO at end of build
.TP
.B \-t
Build and use the tools in $SRC/tools (default setting).
.TP
.B \+t
Use the build tools in "$ONBLD_TOOLS/bin".

.LP
.B Code checking options
.TP 10
.B \-A
Check for ABI discrepancies in .so files.
It is only required for shared object developers when there is an
addition, deletion or change of interface in the .so files.
.TP
```



```
.B \-C
Check for cstyle/hdrchk errors
.TP
.B \-f
Check for unreferenced files.  Since the full workspace must be built
in order to accurately identify unreferenced files, -f is ignored for
incremental (-i) builds, or builds that do not include -l, and -p.
.TP
.B \-r
Check the ELF runtime attributes of all dynamic objects
.TP
.B \-l
Do "make lint" in $LINTDIRS (default: $SRC n)
.TP
.B \-N
Do not run protocmp or checkpaths (note: this option is not
recommended, especially in conjunction with the \-p option)
.TP
.B \-W
Do not report warnings (for freeware gate ONLY)
.TP
.B \-w
Report which proto area objects differ between this and the last build.
See wsdiff(1) for details. Note that the proto areas used for comparison
are the last ones constructed as part of the build. As an example, if both
a non-debug and debug build are performed (in that order), then the debug
proto area will be used for comparison (which might not be what you want).
.LP
.B Groups of options
.TP 10
.B \-G
Gate keeper default group of options (-u)
.TP
.B \-I
Integration engineer default group of options (-mpu)
.TP
.B \-R
Default group of options for building a release (-mp)

.LP
.B Source Build options
.TP 10
.B \-S E | D | H
Build the Export, Domestic, or Hybrid source product. Only Export and
Domestic are truly buildable at this time.
.TP 10
.B \-S O
Simulate an OpenSolaris build on a full tree.  This can be used by
internal developers to ensure that they haven't broken the build for
external developers.
.LP
Source build options only make sense for a full internal tree (open
and closed source).  Only one source build option can be specified at
a time.

.LP
.B Miscellaneous options
.TP 10
.B \-O
generate deliverables for OpenSolaris.  Tarballs containing binaries
of closed-source components are put in $CODEMGR_WS.
generate deliverables for OpenSolaris.  Tarballs containing signed
cryptographic binaries and binaries
of closed-source components are put in $CODEMGR_WS.  (The
cryptographic tarballs are copies of the
ones that are put in the parent directory of
```

```
258 $PKGARCHIVE.)
255 .TP 10
256 .B \-V VERS
257 set the build version string to VERS, overriding VERSION
258 .TP
259 .B \-X
260 Copies the proto area and packages from the IHV and IHV-bin gates into the
261 nightly proto and package areas.  This is only available on i386.  See
262 .B REALMODE ENVIRONMENT VARIABLES
263 and
264 .B BUILDING THE IHV WORKSPACE
265 below.

267 .LP
268 .SH ENVIRONMENT VARIABLES
269 .LP
270 Here is a list of prominent environment variables that
271 .I nightly
272 references and the meaning of each variable.
273 .LP
274 .RE
275 .B CODEMGR_WS
276 .RS 5
277 The root of your workspace, including whatever metadata is kept by
278 the source code management system.  This is the workspace in which the
279 build will be done.
280 .RE
281 .LP
282 .B PARENT_WS
283 .RS 5
284 The root of the workspace that is the parent of the
285 one being built.  This is particularly relevant for configurations
286 with a main
287 workspace and build workspaces underneath it; see the
288 \-u and \-U
289 options as well as the PKGARCHIVE environment variable, for more
290 information.
291 .RE
292 .LP
293 .B BRINGOVER_WS
294 .RS 5
295 This is the workspace from which
296 .I nightly
297 will fetch sources to either populate or update your workspace;
298 it defaults to $CLONE_WS.
299 .RE
300 .LP
301 .B CLOSED_BRINGOVER_WS
302 .RS 5
303 A full Mercurial workspace has two repositories: one for open source
304 and one for closed source.  If this variable is non-null,
305 .I nightly
306 will pull from the repository that it names to get the closed source.
307 It defaults to $CLOSED_CLONE_WS.
308 .LP
309 If $CODEMGR_WS already exists and contains only the open repository,
310 .I nightly
311 will ignore this variable; you'll need to pull the closed repository
312 by hand if you want it.
313 .RE
314 .LP
315 .B CLONE_WS
316 .RS 5
317 This is the workspace from which
318 .I nightly
319 will fetch sources by default.  This is
```



```
320 often distinct from the parent, particularly if the parent is a gate.
321 .RE
322 .LP
323 .B CLOSED_CLONE_WS
324 .RS 5
325 This is the default closed-source Mercurial repository that
326 .I nightly
327 might pull from (see
328 .B CLOSED_BRINGOVER_WS
329 for details).
330 .RE
331 .LP
332 .B SRC
333 .RS 5
334 Root of OS-Net source code, referenced by the Makefiles.  It is
335 the starting point of build activity.  It should be expressed
336 in terms of $CODEMGR_WS.
337 .RE
338 .LP
339 .B ROOT
340 .RS 5
341 Root of the proto area for the build.  The makefiles direct
342 installation of build products to this area and
343 direct references to these files by builds of commands and other
344 targets.  It should be expressed in terms of $CODEMGR_WS.
345 .LP
346 If $MULTI_PROTO is "no", $ROOT may contain a DEBUG or non-DEBUG
347 build.  If $MULTI_PROTO is "yes", $ROOT contains the DEBUG build and
348 $ROOT-nd contains the non-DEBUG build.
349 .LP
350 For OpenSolaris deliveries (\fB\-O\fR), $ROOT-closed contains a parallel
351 proto area containing the DEBUG build of just usr/closed components, and
352 $ROOT-nd-closed contains the non-DEBUG equivalent.
353 .RE
354 .LP
355 .B TOOLS_ROOT
356 .RS 5
357 Root of the tools proto area for the build.  The makefiles direct
358 installation of tools build products to this area.  Unless \fB+t\fR
359 is part of $NIGHTLY_OPTIONS, these tools will be used during the
360 build.
361 .LP
362 As built by nightly, this will always contain non-DEBUG objects.
363 Therefore, this will always have a -nd suffix, regardless of
364 $MULTI_PROTO.
365 .RE
366 .LP
367 .B MACH
368 .RS 5
369 The instruction set architecture of the build machine as given
370 by \fIuname -p\fP, e.g. sparc, i386.
371 .RE
372 .LP
373 .B LOCKNAME
374 .RS 5
375 The name of the file used to lock out multiple runs of
376 .IR nightly .
377 This should generally be left to the default setting.
378 .RE
379 .LP
380 .B ATLOG
381 .RS 5
382 The location of the log directory maintained by
383 .IR nightly .
384 This should generally be left to the default setting.
385 .RE
```

```
.LP
.B LOGFILE
.RS 5
The name of the log file in the $ATLOG directory maintained by
.IR nightly .
This should generally be left to the default setting.
.RE
.LP
.B STAFFER
.RS 5
The non-root account to use on the build machine for the
bringover from the clone or parent workspace.
This may not be the same identify used by the SCM.
.RE
.LP
.B MAILTO
.RS 5
The address to be used to send completion e-mail at the end of
the build (for the \-m option).
.RE
.LP
.B MAILFROM
.RS 5
The address to be used for From: in the completion e-mail at the
end of the build (for the \-m option).
.RE
.LP
.B REF_PROTO_LIST
.RS 5
Name of file used with protocmp to compare proto area contents.
.RE
.LP
.B PARENT_ROOT
.RS 5
The parent root, which is the destination for copying the proto
area(s) when using the \-U option.
.RE
.LP
.B PARENT_TOOLS_ROOT
.RS 5
The parent tools root, which is the destination for copying the tools
proto area when using the \-U option.
.RE
.LP
.B RELEASE
.RS 5
The release version number to be used; e.g., 5.10.1 (Note: this is set
in Makefile.master and should not normally be overridden).
.RE
.LP
.B VERSION
.RS 5
The version text string to be used; e.g., "onnv:‘date ’+%Y-%m-%d’’".
.RE
.LP
.B RELEASE_DATE
.RS 5
The release date text to be used; e.g., October 2009. If not set in
your environment file, then this text defaults to the output from
$(LC_ALL=C date +"%B %Y"); e.g., "October 2009".
.RE
.LP
.B INTERNAL_RELEASE_BUILD
.RS 5
See Makefile.master - but it mostly controls id strings. Generally,
let
```



```
.I nightly
set this for you.
.RE
.LP
.B RELEASE_BUILD
.RS 5
Define this to build a release with a non-DEBUG kernel.
Generally, let
.I nightly
set this for you based on its options.
.RE
.LP
.B PKGARCHIVE
.RS 5
The destination for packages.  This may be relative to
$CODEMGR_WS for private packages or relative to $PARENT_WS
if you have different workspaces for different architectures
but want one hierarchy of packages.
.RE
.LP
.B MAKEFLAGS
.RS 5
Set default flags to make; e.g., -k to build all targets regardless of errors.
.RE
.LP
.B UT_NO_USAGE_TRACKING
.RS 5
Disables usage reporting by listed Devpro tools. Otherwise it sends mail
to some Devpro machine every time the tools are used.
.RE
.LP
.B LINTDIRS
.RS 5
Directories to lint with the \-l option.
.RE
.LP
.B BUILD_TOOLS
.RS 5
BUILD_TOOLS is the root of all tools including the compilers; e.g.,
/ws/onnv-tools.  It is used by the makefile system, but not nightly.
.RE
.LP
.B ONBLD_TOOLS
.RS 5
ONBLD_TOOLS is the root of all the tools that are part of SUNWonbld; e.g.,
/ws/onnv-tools/onbld.  By default, it is derived from
.BR BUILD_TOOLS .
It is used by the makefile system, but not nightly.
.RE
.LP
.B SPRO_ROOT
.RS 5
The gate-defined default location for the Sun compilers, e.g.
/ws/onnv-tools/SUNWspro.  By default, it is derived from
.BR BUILD_TOOLS .
It is used by the makefile system, but not nightly.
.RE
.LP
.B JAVA_ROOT
.RS 5
The location for the java compilers for the build, generally /usr/java.
.RE
.LP
.B OPTHOME
.RS 5
The gate-defined default location of things formerly in /opt; e.g.,
```

```
518 /ws/onnv-tools.  This is used by nightly, but not the makefiles.
519 .RE
520 .LP
521 .B TEAMWARE
522 .RS 5
523 The gate-defined default location for the Teamware tools; e.g.,
524 /ws/onnv-tools/SUNWspro.  By default, it is derived from
525 .BR OPTHOME .
526 This is used by nightly, but not the makefiles.  There is no
527 corresponding variable for Mercurial or Subversion, which are assumed
528 to be installed in the default path.
529 .RE
530 .LP
531 .B OPEN_SRCDIR
532 .RS 5
533 The open source tree is copied to this directory when simulating an
534 OpenSolaris build (\fB\-S O\fR).  It defaults to $CODEMGR_WS/open_src.
535 .LP
536 .RE
537 .B ON_CLOSED_BINS
538 .RS 5
539 OpenSolaris builds do not contain the closed source tree.  Instead,
540 the developer downloads a closed binaries tree and unpacks it.
541 .B ON_CLOSED_BINS
542 tells nightly
543 where to find these closed binaries, so that it can add them into the
544 build.
545 .LP
546 .RE
551 .B ON_CRYPTO_BINS
552 .RS 5
553 This is the path to a compressed tarball that contains debug
554 cryptographic binaries that have been signed to allow execution
555 outside of Sun, e.g., $PARENT_WS/packages/$MACH/on-crypto.$MACH.bz2.
556 .I nightly
557 will automatically adjust the path for non-debug builds.  This tarball
558 is needed if the closed-source tree is not present.  Also, it is
559 usually needed when generating OpenSolaris deliverables from a project
560 workspace.  This is because most projects do not have access to the
561 necessary key and certificate that would let them sign their own
562 cryptographic binaries.
563 .LP
564 .RE
547 .B CHECK_PATHS
548 .RS 5
549 Normally, nightly runs the 'checkpaths' script to check for
550 discrepancies among the files that list paths to other files, such as
551 exception lists and req.flg.  Set this flag to 'n' to disable this
552 check, which appears in the nightly output as "Check lists of files."
553 .RE
554 .LP
555 .B CHECK_DMAKE
556 .RS 5
557 Nightly validates that the version of dmake encountered is known to be
558 safe to use.  Set this flag to 'n' to disable this test, allowing any
559 version of dmake to be used.
560 .RE
561 .LP
562 .B MULTI_PROTO
563 .RS 5
564 If "no" (the default),
565 .I nightly
566 will reuse $ROOT for both the DEBUG and non-DEBUG builds.  If "yes",
567 the DEBUG build will go in $ROOT and the non-DEBUG build will go in
568 $ROOT-nd.  Other values will be treated as "no".  Use of the
569 .B \-O
```



```
570 flag forces MULTI_PROTO to "yes".
571 .RE
572 .LP
573 .SH NIGHTLY HOOK ENVIRONMENT VARIABLES
574 .LP
575 Several optional environment variables may specify commands to run at
576 various points during the build.  Commands specified in the hook
577 variable will be run in a subshell; command output will be appended to
578 the mail message and log file.  If the hook exits with a non-zero
579 status, the build is aborted immediately.  Environment variables
580 defined in the environment file will be available.
581 .LP
582 .B SYS_PRE_NIGHTLY
583 .RS 5
584 Run just after the workspace lock is acquired.  This is reserved for
585 per-build-machine customizations and should be set only in /etc/nightly.conf
586 .RE
587 .LP
588 .B PRE_NIGHTLY
589 .RS 5
590 Run just after SYS_PRE_NIGHTLY.
591 .RE
592 .LP
593 .B PRE_BRINGOVER
594 .RS 5
595 Run just before bringover is started; not run if no bringover is done.
596 .RE
597 .LP
598 .B POST_BRINGOVER
599 .RS 5
600 Run just after bringover completes; not run if no bringover is done.
601 .RE
602 .LP
603 .B POST_NIGHTLY
604 .RS 5
605 Run after the build completes, with the return status of nightly - one
606 of "Completed", "Interrupted", or "Failed" - available in the
607 environment variable NIGHTLY_STATUS.
608 .RE
609 .LP
610 .B SYS_POST_NIGHTLY
611 .RS 5
612 This is reserved for per-build-machine customizations, and runs
613 immedately after POST_NIGHTLY.
614 .RE
615 .LP
616 .SH REALMODE ENVIRONMENT VARIABLES
617 .LP
618 The following environment variables referenced by
619 .I nightly
620 are only required when the -X option is used.
621 .LP
622 .RE
623 .B IA32_IHV_WS
624 .RS 5
625 Reference to the IHV workspace containing IHV driver binaries.
626 The IHV workspace must be fully built before starting the ON realmode build.
627 .LP
628 .RE
629 .B IA32_IHV_ROOT
630 .RS 5
631 Reference to the IHV workspace proto area.
632 The IHV workspace must be fully built before starting the ON realmode build.
633 .LP
634 .RE
635 .B IA32_IHV_PKGS
```

```
 636 .RS 5
 637 Reference to the IHV workspace packages.  If this is empty or the directory
 638 is non-existent, then nightly will skip copying the packages.
 639 .LP
 640 .RE
 641 .B IA32_IHV_BINARY_PKGS
 642 .RS 5
 643 Reference to binary-only IHV packages.  If this is empty or the directory
 644 is non-existent, then nightly will skip copying the packages.
 645 .LP
 646 .RE
 647 .B SPARC_RM_PKGARCHIVE
 648 .RS 5
 649 Destination for sparc realmode package SUNWrmodu.
 650 Yes, this sparc package really is built on x86.
 651 .SH FILES
 652 .LP
 653 .RS 5
 654 /etc/nightly.conf
 655 .RE
 656 .LP
 657 If present, nightly executes this file just prior to executing the
 658 .I env
 659 file.
 660 .SH BUILDING THE IHV WORKSPACE
 661 .LP
 662 The IHV workspace can be built with
 663 .I nightly.
 664 The recommended options are:
 665 .LP
 666 .RS 5
 667 NIGHTLY_OPTIONS="-pmWN"
 668 .RE
 669 .LP
 670 None of the realmode environment variables needed for ON realmode builds
 671 are required to build the IHV workspace.
 672 .SH EXAMPLES
 673 .LP
 674 Start with the example file in usr/src/tools/env/developer.sh
 675 (or gatekeeper.sh), copy to myenv and make your changes.
 676 .LP
 677 .PD 0
 678 # grep NIGHTLY_OPTIONS myenv
 679 .LP
 680 NIGHTLY_OPTIONS="-ACrlapDm"
 681 .LP
 682 export NIGHTLY_OPTIONS
 683 .LP
 684 # /opt/onbld/bin/nightly -i myenv
 685 .PD
 686 .LP
 687 .SH SEE ALSO
 688 .BR bldenv (1)
```

```
**********************************************************
   85117 Fri Oct 11 17:19:37 2013
new/usr/src/tools/scripts/nightly.sh
4108 remove ON_CRYPTO_BINS from tools
**********************************************************
______unchanged_portion_omitted_

411 #
412 # Function to do the build, including package generation.
413 # usage: build LABEL SUFFIX ND MULTIPROTO
414 # - LABEL is used to tag build output.
415 # - SUFFIX is used to distinguish files (e.g., DEBUG vs non-DEBUG,
416 #   open-only vs full tree).
417 # - ND is "-nd" (non-DEBUG builds) or "" (DEBUG builds).
418 # - If MULTIPROTO is "yes", it means to name the proto area according to
419 #   SUFFIX.  Otherwise ("no"), (re)use the standard proto area.
420 #
421 function build {
422         LABEL=$1
423         SUFFIX=$2
424         ND=$3
425         MULTIPROTO=$4
426         INSTALLOG=install${SUFFIX}-${MACH}
427         NOISE=noise${SUFFIX}-${MACH}
428         PKGARCHIVE=${PKGARCHIVE_ORIG}${SUFFIX}

430         ORIGROOT=$ROOT
431         [ $MULTIPROTO = no ] || export ROOT=$ROOT$SUFFIX

433         if [[ "$O_FLAG" = y ]]; then
434                 echo "\nSetting CLOSEDROOT= ${ROOT}-closed\n" >> $LOGFILE
435                 export CLOSEDROOT=${ROOT}-closed
436         fi

438         export ENVLDLIBS1='myldlibs $ROOT'
439         export ENVCPPFLAGS1='myheaders $ROOT'

441         this_build_ok=y
442         #
443         #       Build OS-Networking source
444         #
445         echo "\n==== Building OS-Net source at 'date' ($LABEL) ====\n" \
446                 >> $LOGFILE

448         rm -f $SRC/${INSTALLOG}.out
449         cd $SRC
450         /bin/time $MAKE -e install 2>&1 | \
451             tee -a $SRC/${INSTALLOG}.out >> $LOGFILE

453         if [[ "$SCM_TYPE" = teamware ]]; then
454                 echo "\n==== SCCS Noise ($LABEL) ====\n" >> $mail_msg_file
455                 egrep 'sccs(check:| *get)' $SRC/${INSTALLOG}.out >> \
456                         $mail_msg_file
457         fi

459         echo "\n==== Build errors ($LABEL) ====\n" >> $mail_msg_file
460         egrep ":" $SRC/${INSTALLOG}.out |
461                 egrep -e "(^${MAKE}:|[  ]error[:        \n])" | \
462                 egrep -v "Ignoring unknown host" | \
463                 egrep -v "cc .* -o error " | \
464                 egrep -v "warning" >> $mail_msg_file
465         if [ "$?" = "0" ]; then
466                 build_ok=n
467                 this_build_ok=n
468         fi
469         grep "bootblock image is .* bytes too big" $SRC/${INSTALLOG}.out \
```



```
470                 >> $mail_msg_file
471         if [ "$?" = "0" ]; then
472                 build_ok=n
473                 this_build_ok=n
474         fi

476         if [ "$W_FLAG" = "n" ]; then
477                 echo "\n==== Build warnings ($LABEL) ====\n" >>$mail_msg_file
478                 egrep -i warning: $SRC/${INSTALLOG}.out \
479                         | egrep -v '^tic:' \
480                         | egrep -v "symbol (\'|')timezone' has differing types:"
481                         | egrep -v "parameter <PSTAMP> set to" \
482                         | egrep -v "Ignoring unknown host" \
483                         | egrep -v "redefining segment flags attribute for" \
484                         >> $mail_msg_file
485         fi

487         echo "\n==== Ended OS-Net source build at 'date' ($LABEL) ====\n" \
488                 >> $LOGFILE

490         echo "\n==== Elapsed build time ($LABEL) ====\n" >>$mail_msg_file
491         tail -3  $SRC/${INSTALLOG}.out >>$mail_msg_file

493         if [ "$i_FLAG" = "n" -a "$W_FLAG" = "n" ]; then
494                 rm -f $SRC/${NOISE}.ref
495                 if [ -f $SRC/${NOISE}.out ]; then
496                         mv $SRC/${NOISE}.out $SRC/${NOISE}.ref
497                 fi
498                 grep : $SRC/${INSTALLOG}.out \
499                         | egrep -v '^/' \
500                         | egrep -v '^(Start|Finish|real|user|sys|./bld_awk)' \
501                         | egrep -v '^tic:' \
502                         | egrep -v '^mcs' \
503                         | egrep -v '^LD_LIBRARY_PATH=' \
504                         | egrep -v 'ar: creating' \
505                         | egrep -v 'ar: writing' \
506                         | egrep -v 'conflicts:' \
507                         | egrep -v ':saved created' \
508                         | egrep -v '^stty.*c:' \
509                         | egrep -v '^mfgname.c:' \
510                         | egrep -v '^uname-i.c:' \
511                         | egrep -v '^volumes.c:' \
512                         | egrep -v '^lint library construction:' \
513                         | egrep -v 'tsort: INFORM:' \
514                         | egrep -v 'stripalign:' \
515                         | egrep -v 'chars, width' \
516                         | egrep -v "symbol (\'|')timezone' has differing types:"
517                         | egrep -v 'PSTAMP' \
518                         | egrep -v '|%WHOANDWHERE%|' \
519                         | egrep -v '^Manifying' \
520                         | egrep -v 'Ignoring unknown host' \
521                         | egrep -v 'Processing method:' \
522                         | egrep -v '^Writing' \
523                         | egrep -v 'spellin1:' \
524                         | egrep -v '^adding:' \
525                         | egrep -v "^echo 'msgid" \
526                         | egrep -v '^echo ' \
527                         | egrep -v '\.c:$' \
528                         | egrep -v '^Adding file:' \
529                         | egrep -v 'CLASSPATH=' \
530                         | egrep -v '\/var\/mail\/:saved' \
531                         | egrep -v -- '-DUTS_VERSION=' \
532                         | egrep -v '^Running Mkbootstrap' \
533                         | egrep -v '^Applet length read:' \
534                         | egrep -v 'bytes written:' \
535                         | egrep -v '^File:SolarisAuthApplet.bin' \
```

```
536                         | egrep -v -i 'jibversion' \
537                         | egrep -v '^Output size:' \
538                         | egrep -v '^Solo size statistics:' \
539                         | egrep -v '^Using ROM API Version' \
540                         | egrep -v '^Zero Signature length:' \
541                         | egrep -v '^Note \(probably harmless\):' \
542                         | egrep -v '::' \
543                         | egrep -v -- '-xcache' \
544                         | egrep -v '^\+' \
545                         | egrep -v '^cc1: note: -fwritable-strings' \
546                         | egrep -v 'svccfg-native -s svc:/' \
547                         | sort | uniq >$SRC/${NOISE}.out
548                 if [ ! -f $SRC/${NOISE}.ref ]; then
549                         cp $SRC/${NOISE}.out $SRC/${NOISE}.ref
550                 fi
551                 echo "\n==== Build noise differences ($LABEL) ====\n" \
552                         >>$mail_msg_file
553                 diff $SRC/${NOISE}.ref $SRC/${NOISE}.out >>$mail_msg_file
554         fi

556         #
557         #       Re-sign selected binaries using signing server
558         #       (gatekeeper builds only)
559         #
560         if [ -n "$CODESIGN_USER" -a "$this_build_ok" = "y" ]; then
561                 echo "\n==== Signing proto area at 'date' ====\n" >> $LOGFILE
562                 signing_file="${TMPDIR}/signing"
563                 rm -f ${signing_file}
564                 export CODESIGN_USER
565                 signproto $SRC/tools/codesign/creds 2>&1 | \
566                         tee -a ${signing_file} >> $LOGFILE
567                 echo "\n==== Finished signing proto area at 'date' ====\n" \
568                     >> $LOGFILE
569                 echo "\n==== Crypto module signing errors ($LABEL) ====\n" \
570                     >> $mail_msg_file
571                 egrep 'WARNING|ERROR' ${signing_file} >> $mail_msg_file
572                 if (( $? == 0 )) ; then
573                         build_ok=n
574                         this_build_ok=n
575                 fi
576         fi

578         #
557         #       Building Packages
558         #
559         if [ "$p_FLAG" = "y" -a "$this_build_ok" = "y" ]; then
560                 if [ -d $SRC/pkg -o -d $SRC/pkgdefs ]; then
561                         echo "\n==== Creating $LABEL packages at 'date' ====\n"
562                                 >> $LOGFILE
563                         echo "Clearing out $PKGARCHIVE ..." >> $LOGFILE
564                         rm -rf $PKGARCHIVE >> "$LOGFILE" 2>&1
565                         mkdir -p $PKGARCHIVE >> "$LOGFILE" 2>&1

567                         for d in pkg pkgdefs; do
568                                 if [ ! -f "$SRC/$d/Makefile" ]; then
569                                         continue
570                                 fi
571                                 rm -f $SRC/$d/${INSTALLOG}.out
572                                 cd $SRC/$d
573                                 /bin/time $MAKE -e install 2>&1 | \
574                                         tee -a $SRC/$d/${INSTALLOG}.out >> $LOGF
575                         done

577                         echo "\n==== package build errors ($LABEL) ====\n" \
578                                 >> $mail_msg_file
```



```
580                         for d in pkg pkgdefs; do
581                                 if [ ! -f "$SRC/$d/Makefile" ]; then
582                                         continue
583                                 fi

585                                 egrep "${MAKE}|ERROR|WARNING" $SRC/$d/${INSTALLO
586                                         grep ':' | \
587                                         grep -v PSTAMP | \
588                                         egrep -v "Ignoring unknown host" \
589                                         >> $mail_msg_file
590                         done
591                 else
592                         #
593                         # Handle it gracefully if -p was set but there are
594                         # neither pkg nor pkgdefs directories.
595                         #
596                         echo "\n==== No $LABEL packages to build ====\n" \
597                                 >> $LOGFILE
598                 fi
599         else
600                 echo "\n==== Not creating $LABEL packages ====\n" >> $LOGFILE
601         fi

603         ROOT=$ORIGROOT
604 }
```

_____***unchanged_portion_omitted_***