****
7815 Fri May 22 11:27:12 2015
new/usr/src/tools/env/developer.sh
nightly: don’t check dmake version, now it’s ours
****

```
#
# 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

# 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 three lines
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

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

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

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

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

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

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

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

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

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

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

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

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

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

# POST_NIGHTLY can be any command to be run at the end of nightly.  See
# nightly(1) for interactions between environment variables and this command.
#POST_NIGHTLY=
**********************************************************
    8428 Fri May 22 11:27:13 2015
new/usr/src/tools/env/gatekeeper.sh
nightly: don't check dmake version, now it's ours
**********************************************************
#
# 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

# 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 three lines
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

#
# 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

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

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


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

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

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

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

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

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

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

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

```
#CHECK_PATHS=y

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

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

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

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

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

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

# 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).
export PARENT_WS=''

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

# 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.
#
export STAFFER="$LOGNAME"
export MAILTO="$STAFFER"

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

# 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.
export BUILD_PROJECT=''

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

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

# 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).
#
export REF_PROTO_LIST="$PARENT_WS/usr/src/proto_list_${MACH}"


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

#
#       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
#
export VERSION="$GATE"

#
# 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
#
# export RELEASE='5.11'
# export RELEASE_DATE='October 2007'

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
MACH=`uname -p`

if [ "$OPTHOME" = "" ]; then
        OPTHOME=/opt
        export OPTHOME
fi

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

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

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

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

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

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

```
f_FLAG=n
i_FLAG=n; i_CMD_LINE_FLAG=n
l_FLAG=n
M_FLAG=n
m_FLAG=n
N_FLAG=n
n_FLAG=n
p_FLAG=n
r_FLAG=n
t_FLAG=y
U_FLAG=n
u_FLAG=n
V_FLAG=n
w_FLAG=n
W_FLAG=n
#
build_ok=y
build_extras_ok=y

#
# examine arguments
#

OPTIND=1
while getopts +intV:W FLAG
do
        case $FLAG in
          i )   i_FLAG=y; i_CMD_LINE_FLAG=y
                ;;
          n )   n_FLAG=y
                ;;
         +t )   t_FLAG=n
                ;;
          V )   V_FLAG=y
                V_ARG="$OPTARG"
                ;;
          W )   W_FLAG=y
                ;;
         \? )   echo "$USAGE"
                exit 1
                ;;
        esac
done

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

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

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

#
# force locale to C
LANG=C;         export LANG
LC_ALL=C;       export LC_ALL
LC_COLLATE=C;   export LC_COLLATE
LC_CTYPE=C;     export LC_CTYPE
LC_MESSAGES=C;  export LC_MESSAGES
```

```
LC_MONETARY=C;  export LC_MONETARY
LC_NUMERIC=C;   export LC_NUMERIC
LC_TIME=C;      export LC_TIME

# clear environment variables we know to be bad for the build
unset LD_OPTIONS
unset LD_AUDIT          LD_AUDIT_32             LD_AUDIT_64
unset LD_BIND_NOW       LD_BIND_NOW_32          LD_BIND_NOW_64
unset LD_BREADTH        LD_BREADTH_32           LD_BREADTH_64
unset LD_CONFIG         LD_CONFIG_32            LD_CONFIG_64
unset LD_DEBUG          LD_DEBUG_32             LD_DEBUG_64
unset LD_DEMANGLE       LD_DEMANGLE_32          LD_DEMANGLE_64
unset LD_FLAGS          LD_FLAGS_32             LD_FLAGS_64
unset LD_LIBRARY_PATH   LD_LIBRARY_PATH_32      LD_LIBRARY_PATH_64
unset LD_LOADFLTR       LD_LOADFLTR_32          LD_LOADFLTR_64
unset LD_NOAUDIT        LD_NOAUDIT_32           LD_NOAUDIT_64
unset LD_NOAUXFLTR      LD_NOAUXFLTR_32         LD_NOAUXFLTR_64
unset LD_NOCONFIG       LD_NOCONFIG_32          LD_NOCONFIG_64
unset LD_NODIRCONFIG    LD_NODIRCONFIG_32       LD_NODIRCONFIG_64
unset LD_NODIRECT       LD_NODIRECT_32          LD_NODIRECT_64
unset LD_NOLAZYLOAD     LD_NOLAZYLOAD_32        LD_NOLAZYLOAD_64
unset LD_NOOBJALTER     LD_NOOBJALTER_32        LD_NOOBJALTER_64
unset LD_NOVERSION      LD_NOVERSION_32         LD_NOVERSION_64
unset LD_ORIGIN         LD_ORIGIN_32            LD_ORIGIN_64
unset LD_PRELOAD        LD_PRELOAD_32           LD_PRELOAD_64
unset LD_PROFILE        LD_PROFILE_32           LD_PROFILE_64

unset CONFIG
unset GROUP
unset OWNER
unset REMOTE
unset ENV
unset ARCH
unset CLASSPATH
unset NAME

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

#
#       Setup environmental variables
#
if [ -f /etc/nightly.conf ]; then
        . /etc/nightly.conf
fi

if [ -f $1 ]; then
        if [[ $1 = */* ]]; then
                . $1
        else
                . ./$1
        fi
else
        if [ -f $OPTHOME/onbld/env/$1 ]; then
                . $OPTHOME/onbld/env/$1
        else
                echo "Cannot find env file as either $1 or $OPTHOME/onbld/env/$1
                exit 1
        fi
fi

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

```
# STDENV_START
# STDENV_END

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

#
# place ourselves in a new task, respecting BUILD_PROJECT if set.
#
if [ -z "$BUILD_PROJECT" ]; then
        /usr/bin/newtask -c $$
else
        /usr/bin/newtask -c $$ -p $BUILD_PROJECT
fi

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

#
# See if NIGHTLY_OPTIONS is set
#
if [ "$NIGHTLY_OPTIONS" = "" ]; then
        NIGHTLY_OPTIONS="-aBm"
fi

#
# If BRINGOVER_WS was not specified, let it default to CLONE_WS
#
if [ "$BRINGOVER_WS" = "" ]; then
        BRINGOVER_WS=$CLONE_WS
fi

#
# If BRINGOVER_FILES was not specified, default to usr
#
if [ "$BRINGOVER_FILES" = "" ]; then
        BRINGOVER_FILES="usr"
fi

check_closed_bins

#
# Note: changes to the option letters here should also be applied to the
#       bldenv script.  `d' is listed for backward compatibility.
#
NIGHTLY_OPTIONS=-${NIGHTLY_OPTIONS#-}
OPTIND=1
while getopts +ABCDdFfGIilMmNnpRrtUuwW FLAG $NIGHTLY_OPTIONS
do
        case $FLAG in
          A )   A_FLAG=y
                ;;
          B )   D_FLAG=y
                ;; # old version of D
          C )   C_FLAG=y
                ;;
          D )   D_FLAG=y
                ;;
          F )   F_FLAG=y
```

```
                ;;
          f )   f_FLAG=y
                ;;
          G )   u_FLAG=y
                ;;
          I )   m_FLAG=y
                p_FLAG=y
                u_FLAG=y
                ;;
          i )   i_FLAG=y
                ;;
          l )   l_FLAG=y
                ;;
          M )   M_FLAG=y
                ;;
          m )   m_FLAG=y
                ;;
          N )   N_FLAG=y
                ;;
          n )   n_FLAG=y
                ;;
          p )   p_FLAG=y
                ;;
          R )   m_FLAG=y
                p_FLAG=y
                ;;
          r )   r_FLAG=y
                ;;
         +t )   t_FLAG=n
                ;;
          U )   if [ -z "${PARENT_ROOT}" ]; then
                        echo "PARENT_ROOT must be set if the U flag is" \
                            "present in NIGHTLY_OPTIONS."
                        exit 1
                fi
                NIGHTLY_PARENT_ROOT=$PARENT_ROOT
                if [ -n "${PARENT_TOOLS_ROOT}" ]; then
                        NIGHTLY_PARENT_TOOLS_ROOT=$PARENT_TOOLS_ROOT
                fi
                U_FLAG=y
                ;;
          u )   u_FLAG=y
                ;;
          w )   w_FLAG=y
                ;;
          W )   W_FLAG=y
                ;;
         \? )   echo "$USAGE"
                exit 1
                ;;
        esac
done

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

if [ -z "$MAILTO" -o "$MAILTO" = "nobody" ]; then
        MAILTO=$STAFFER
        export MAILTO
fi
```

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

# roots of source trees, both relative to $SRC and absolute.
relsrcdirs="."
abssrcdirs="$SRC"

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

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

maketype="distributed"
if [[ -z "$MAKE" ]]; then
        MAKE=dmake
elif [[ ! -x "$MAKE" ]]; then
        echo "\$MAKE is set to garbage in the environment"
        exit 1
fi
# get the dmake version string alone
DMAKE_VERSION=$( $MAKE -v )
DMAKE_VERSION=${DMAKE_VERSION#*: }
# focus in on just the dotted version number alone
DMAKE_MAJOR=$( echo $DMAKE_VERSION | \
        sed -e ’s/.*\<\([^.]*\.[^   ]*\).*$/\1/’ )
# extract the second (or final) integer
DMAKE_MINOR=${DMAKE_MAJOR#*.}
DMAKE_MINOR=${DMAKE_MINOR%%.*}
# extract the first integer
DMAKE_MAJOR=${DMAKE_MAJOR%%.*}
CHECK_DMAKE=${CHECK_DMAKE:-y}
# x86 was built on the 12th, sparc on the 13th.
if [ "$CHECK_DMAKE" = "y" -a \
     "$DMAKE_VERSION" != "Sun Distributed Make 7.3 2003/03/12" -a \
     "$DMAKE_VERSION" != "Sun Distributed Make 7.3 2003/03/13" -a \( \
     "$DMAKE_MAJOR" -lt 7 -o \
     "$DMAKE_MAJOR" -eq 7 -a "$DMAKE_MINOR" -lt 4 \) ]; then
        if [ -z "$DMAKE_VERSION" ]; then
                echo "$MAKE is missing."
                exit 1
        fi
        echo ‘whence $MAKE‘" version is:"
        echo "  ${DMAKE_VERSION}"
        cat <<EOF

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

  CHECK_DMAKE=n
EOF
        exit 1
fi
export PATH
export MAKE
```

```
if [ "${SUNWSPRO}" != "" ]; then
        PATH="${SUNWSPRO}/bin:$PATH"
        export PATH
fi

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

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

        export DMAKE_MAX_JOBS=$maxjobs
fi

DMAKE_MODE=parallel;
export DMAKE_MODE

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

#
# if -V flag was given, reset VERSION to V_ARG
#
if [ "$V_FLAG" = "y" ]; then
        VERSION=$V_ARG
fi

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

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

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

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

unset   CFLAGS LD_LIBRARY_PATH LDFLAGS

# create directories that are automatically removed if the nightly script
# fails to start correctly
function newdir {
        dir=$1
        toadd=
        while [ ! -d $dir ]; do
                toadd="$dir $toadd"
                dir=`dirname $dir`
        done
        torm=
        newlist=
        for dir in $toadd; do
                if staffer mkdir $dir; then
                        newlist="$ISUSER $dir $newlist"
                        torm="$dir $torm"
                else
                        [ -z "$torm" ] || staffer rmdir $torm
                        return 1
                fi
        done
        newdirlist="$newlist $newdirlist"
        return 0
}
```
______***unchanged_portion_omitted_***