```
***********************************************************
   10814 Thu Sep 13 22:58:40 2018
new/usr/src/tools/scripts/bldenv.sh
9831 bldenv should adapt to nightly debug settings
***********************************************************
```

```
#!/usr/bin/ksh93
#
# 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 2011 Nexenta Systems, Inc.  All rights reserved.
# Copyright 2014 Garrett D'Amore <garrett@damore.org>
# Copyright 2018 Joyent, Inc.
#
# Uses supplied "env" file, based on /opt/onbld/etc/env, to set shell variables
# before spawning a shell for doing a release-style builds interactively
# and incrementally.
#

function fatal_error
{
        print -u2 "${progname}: $*"
        exit 1
}
```

*unchanged_portion_omitted_*

```
typeset -r USAGE=$'+
[-?\n@(#)\$Id: bldenv (OS/Net) 2008-04-06 \$\n]
[-author?OS/Net community <tools-discuss@opensolaris.org>]
[+NAME?bldenv - spawn shell for interactive incremental OS-Net
    consolidation builds]
[+DESCRIPTION?bldenv is a useful companion to the nightly(1) script for
    doing interactive and incremental builds in a workspace
    already built with nightly(1). bldenv spawns a shell set up
    with the same environment variables taken from an env_file,
    as prepared for use with nightly(1).]
[+?In addition to running a shell for interactive use, bldenv
    can optionally run a single command in the given environment,
    in the vein of sh -c or su -c. This is useful for
    scripting, when an interactive shell would not be. If the
    command is composed of multiple shell words or contains
    other shell metacharacters, it must be quoted appropriately.]
[+?bldenv is particularly useful for testing Makefile targets
    like clobber, install and _msg, which otherwise require digging
    through large build logs to figure out what is being
    done.]
[+?By default, bldenv will invoke the shell specified in
```



```
    $SHELL. If $SHELL is not set or is invalid, csh will be
    used.]
[c?force the use of csh, regardless of the  value  of $SHELL.]
[f?invoke csh with the -f (fast-start) option. This option is valid
    only if $SHELL is unset or if it points to csh.]
[d?set up environment for doing DEBUG builds. The default is non-DEBUG,
    unless the -F flag is specified in the nightly file.]
[d?set up environment for doing DEBUG builds (default is non-DEBUG)]
[t?set up environment to use the tools in usr/src/tools (this is the
    default, use +t to use the tools from /opt/onbld)]

<env_file> [command]

[+EXAMPLES]{
    [+?Example 1: Interactive use]{
        [+?Use bldenv to spawn a shell to perform  a  DEBUG  build  and
            testing of the  Makefile  targets  clobber and install for
            usr/src/cmd/true.]
        [+\n% rlogin wopr-2 -l gk
{root::wopr-2::49} bldenv -d /export0/jg/on10-se.env
Build type   is  DEBUG
RELEASE      is  5.10
VERSION      is  wopr-2::on10-se::11/01/2001
RELEASE_DATE is  May 2004
The top-level 'setup\' target is available to build headers
and tools.
Using /usr/bin/tcsh as shell.
{root::wopr-2::49}
{root::wopr-2::49} cd $SRC/cmd/true
{root::wopr-2::50} make
{root::wopr-2::51} make clobber
/usr/bin/rm -f true true.po
{root::wopr-2::52} make
/usr/bin/rm -f true
cat true.sh > true
chmod +x true
{root::wopr-2::53} make install
install -s -m 0555 -u root -g bin -f /export0/jg/on10-se/proto/root_sparc/usr/bi
'install\' is up to date.]
    }
    [+?Example 2: Non-interactive use]{
        [+?Invoke bldenv to create SUNWonbld with a single command:]
        [+\nexample% bldenv onnv_06 \'cd $SRC/tools && make pkg\']
        }
}
[+SEE ALSO?\bnightly\b(1)]
'

# main
builtin basename

# boolean flags (true/false)
typeset flags=(
        typeset c=false
        typeset f=false
        typeset d=false
        typeset O=false
        typeset o=false
        typeset t=true
        typeset s=(
                typeset e=false
                typeset h=false
                typeset d=false
                typeset o=false
        )
        typeset d_set=false
```

```
133         typeset DF_build=false
134 )

136 typeset progname="$(basename -- "${0}")"

138 OPTIND=1
135 SUFFIX="-nd"

140 while getopts -a "${progname}" "${USAGE}" OPT ; do
141     case ${OPT} in
142           c)    flags.c=true  ;;
143           +c)   flags.c=false ;;
144           f)    flags.f=true  ;;
145           +f)   flags.f=false ;;
146           d)    flags.d=true ; flags.d_set=true ;;
147           +d)   flags.d=false ; flags.d_set=true ;;
143           d)    flags.d=true  SUFFIX=""    ;;
144           +d)   flags.d=false SUFFIX="-nd" ;;
148           t)    flags.t=true  ;;
149           +t)   flags.t=false ;;
150           \?)   usage ;;
151     esac
152 done
153 shift $((OPTIND-1))

155 # test that the path to the environment-setting file was given
156 if (( $# < 1 )) ; then
157         usage
158 fi

160 # force locale to C
161 export \
162         LANG=C \
163         LC_ALL=C \
164         LC_COLLATE=C \
165         LC_CTYPE=C \
166         LC_MESSAGES=C \
167         LC_MONETARY=C \
168         LC_NUMERIC=C \
169         LC_TIME=C

171 # clear environment variables we know to be bad for the build
172 unset \
173         LD_OPTIONS \
174         LD_LIBRARY_PATH \
175         LD_AUDIT \
176         LD_BIND_NOW \
177         LD_BREADTH \
178         LD_CONFIG \
179         LD_DEBUG \
180         LD_FLAGS \
181         LD_LIBRARY_PATH_64 \
182         LD_NOVERSION \
183         LD_ORIGIN \
184         LD_LOADFLTR \
185         LD_NOAUXFLTR \
186         LD_NOCONFIG \
187         LD_NODIRCONFIG \
188         LD_NOOBJALTER \
189         LD_PRELOAD \
190         LD_PROFILE \
191         CONFIG \
192         GROUP \
193         OWNER \
194         REMOTE \
195         ENV \
```



```
196         ARCH \
197         CLASSPATH

199 #
200 # Setup environment variables
201 #
202 if [[ -f /etc/nightly.conf ]]; then
203         source /etc/nightly.conf
204 fi

206 if [[ -f "$1" ]]; then
207         if [[ "$1" == */* ]]; then
208                 source "$1"
209         else
210                 source "./$1"
211         fi
212 else
213         if [[ -f "/opt/onbld/env/$1" ]]; then
214                 source "/opt/onbld/env/$1"
215         else
216                 printf \
217                     'Cannot find env file as either %s or /opt/onbld/env/%s\n' \
218                     "$1" "$1"
219                 exit 1
220         fi
221 fi
222 shift

224 # contents of stdenv.sh inserted after next line:
225 # STDENV_START
226 # STDENV_END

228 # Check if we have sufficient data to continue...
229 [[ -v CODEMGR_WS ]] || fatal_error "Error: Variable CODEMGR_WS not set."
230 [[ -d "${CODEMGR_WS}" ]] || fatal_error "Error: ${CODEMGR_WS} is not a directory
231 [[ -f "${CODEMGR_WS}/usr/src/Makefile" ]] || fatal_error "Error: ${CODEMGR_WS}/u

233 # must match the getopts in nightly.sh
234 OPTIND=1
235 NIGHTLY_OPTIONS="-${NIGHTLY_OPTIONS#-}"
236 while getopts '+0ABCDdFfGIilMmNnpRrtUuwW' FLAG $NIGHTLY_OPTIONS
237 do
238         case "$FLAG" in
239           t)    flags.t=true  ;;
240           +t)   flags.t=false ;;
241           F)    flags.DF_build=true ;;
242           *)    ;;
243         esac
244 done

246 # DEBUG is a little bit complicated.  First, bldenv -d/+d over-rides
247 # the env file.  Otherwise, we'll default to DEBUG iff we are *not*
248 # building non-DEBUG bits at all.
249 if [ "${flags.d_set}" != "true" ] && "${flags.DF_build}"; then
250         flags.d=true
251 fi

253 POUND_SIGN="#"
254 # have we set RELEASE_DATE in our env file?
255 if [ -z "$RELEASE_DATE" ]; then
256         RELEASE_DATE=$(LC_ALL=C date +"%B %Y")
257 fi
258 BUILD_DATE=$(LC_ALL=C date +%Y-%b-%d)
259 BASEWSDIR=$(basename -- "${CODEMGR_WS}")
260 DEV_CM="\"@(#)SunOS Internal Development: $LOGNAME $BUILD_DATE [$BASEWSDIR]\""
261 export DEV_CM RELEASE_DATE POUND_SIGN
```

```
263 print 'Build type   is  \c'
264 if ${flags.d} ; then
265         print 'DEBUG'
266         SUFFIX=""
267         unset RELEASE_BUILD
268         unset EXTRA_OPTIONS
269         unset EXTRA_CFLAGS
270 else
271         # default is a non-DEBUG build
272         print 'non-DEBUG'
273         SUFFIX="-nd"
274         export RELEASE_BUILD=
275         unset EXTRA_OPTIONS
276         unset EXTRA_CFLAGS
277 fi

279 # update build-type variables
280 PKGARCHIVE="${PKGARCHIVE}${SUFFIX}"

282 #       Set PATH for a build
283 PATH="/opt/onbld/bin:/opt/onbld/bin/${MACH}:/opt/SUNWspro/bin:/usr/ccs/bin:/usr/
284 if [[ "${SUNWSPRO}" != "" ]]; then
285         export PATH="${SUNWSPRO}/bin:$PATH"
286 fi

288 if [[ -n "${MAKE}" ]]; then
289         if [[ -x "${MAKE}" ]]; then
290                 export PATH="$(dirname -- "${MAKE}"):$PATH"
291         else
292                 print "\$MAKE (${MAKE}) is not a valid executible"
293                 exit 1
294         fi
295 fi

297 TOOLS="${SRC}/tools"
298 TOOLS_PROTO="${TOOLS}/proto/root_${MACH}-nd" ; export TOOLS_PROTO

300 if "${flags.t}" ; then
301         export ONBLD_TOOLS="${ONBLD_TOOLS:=${TOOLS_PROTO}/opt/onbld}"

303         export STABS="${TOOLS_PROTO}/opt/onbld/bin/${MACH}/stabs"
304         export CTFSTABS="${TOOLS_PROTO}/opt/onbld/bin/${MACH}/ctfstabs"
305         export GENOFFSETS="${TOOLS_PROTO}/opt/onbld/bin/genoffsets"

307         export CTFCONVERT="${TOOLS_PROTO}/opt/onbld/bin/${MACH}/ctfconvert"
308         export CTFMERGE="${TOOLS_PROTO}/opt/onbld/bin/${MACH}/ctfmerge"
309         export NDRGEN="${TOOLS_PROTO}/opt/onbld/bin/${MACH}/ndrgen"

311         PATH="${TOOLS_PROTO}/opt/onbld/bin/${MACH}:${PATH}"
312         PATH="${TOOLS_PROTO}/opt/onbld/bin:${PATH}"
313         export PATH
314 fi

316 export DMAKE_MODE=${DMAKE_MODE:-parallel}

318 #
319 # Work around folks who have historically used GCC_ROOT and convert it to
320 # GNUC_ROOT. We leave GCC_ROOT in the environment for now (though this could
321 # mess up the case where multiple different gcc versions are being used to
322 # shadow).
323 #
324 if [[ -n "${GCC_ROOT}" ]]; then
325         export GNUC_ROOT=${GCC_ROOT}
326 fi
```



```
328 DEF_STRIPFLAG="-s"

330 TMPDIR="/tmp"

332 export \
333         PATH TMPDIR \
334         POUND_SIGN \
335         DEF_STRIPFLAG \
336         RELEASE_DATE
337 unset \
338         CFLAGS \
339         LD_LIBRARY_PATH

341 # a la ws
342 ENVLDLIBS1=
343 ENVLDLIBS2=
344 ENVLDLIBS3=
345 ENVCPPFLAGS1=
346 ENVCPPFLAGS2=
347 ENVCPPFLAGS3=
348 ENVCPPFLAGS4=
349 PARENT_ROOT=
350 PARENT_TOOLS_ROOT=

352 if [[ "$MULTI_PROTO" != "yes" && "$MULTI_PROTO" != "no" ]]; then
353         printf \
354             'WARNING: invalid value for MULTI_PROTO (%s); setting to "no".\n' \
355             "$MULTI_PROTO"
356         export MULTI_PROTO="no"
357 fi

359 [[ "$MULTI_PROTO" == "yes" ]] && export ROOT="${ROOT}${SUFFIX}"

361 ENVLDLIBS1="-L$ROOT/lib -L$ROOT/usr/lib"
362 ENVCPPFLAGS1="-I$ROOT/usr/include"
363 MAKEFLAGS=e

365 export \
366         ENVLDLIBS1 \
367         ENVLDLIBS2 \
368         ENVLDLIBS3 \
369         ENVCPPFLAGS1 \
370         ENVCPPFLAGS2 \
371         ENVCPPFLAGS3 \
372         ENVCPPFLAGS4 \
373         MAKEFLAGS \
374         PARENT_ROOT \
375         PARENT_TOOLS_ROOT

377 printf 'RELEASE      is %s\n'   "$RELEASE"
378 printf 'VERSION      is %s\n'   "$VERSION"
379 printf 'RELEASE_DATE is %s\n\n' "$RELEASE_DATE"

381 if [[ -f "$SRC/Makefile" ]] && egrep -s '^setup:' "$SRC/Makefile" ; then
382         print "The top-level 'setup' target is available \c"
383         print "to build headers and tools."
384         print ""

386 elif "${flags.t}" ; then
387         printf \
388             'The tools can be (re)built with the install target in %s.\n\n' \
389             "${TOOLS}"
390 fi

392 #
393 # place ourselves in a new task, respecting BUILD_PROJECT if set.
```

```
 394 #
 395 /usr/bin/newtask -c $$ ${BUILD_PROJECT:+-p$BUILD_PROJECT}

 397 if [[ "${flags.c}" == "false" && -x "$SHELL" && \
 398     "$(basename -- "${SHELL}")" != "csh" ]]; then
 399         # $SHELL is set, and it's not csh.

 401         if "${flags.f}" ; then
 402                 print 'WARNING: -f is ignored when $SHELL is not csh'
 403         fi

 405         printf 'Using %s as shell.\n' "$SHELL"
 406         exec "$SHELL" ${@:+-c "$@"}

 408 elif "${flags.f}" ; then
 409         print 'Using csh -f as shell.'
 410         exec csh -f ${@:+-c "$@"}

 412 else
 413         print 'Using csh as shell.'
 414         exec csh ${@:+-c "$@"}
 415 fi

 417 # not reached
```