```
**********************************************************
   70592 Thu Feb 13 20:11:25 2014
new/usr/src/tools/scripts/nightly.sh
Nightly can use a different tempdir.
**********************************************************
______unchanged_portion_omitted_


 774 MACH=‘uname -p‘

 776 if [ "$OPTHOME" = "" ]; then
 777         OPTHOME=/opt
 778         export OPTHOME
 779 fi
 780 if [ "$TEAMWARE" = "" ]; then
 781         TEAMWARE=$OPTHOME/teamware
 782         export TEAMWARE
 783 fi

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

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

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

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

 801         -A      check for ABI differences in .so files
 802         -C      check for cstyle/hdrchk errors
 803         -D      do a build with DEBUG on
 804         -F      do _not_ do a non-DEBUG build
 805         -G      gate keeper default group of options (-au)
 806         -I      integration engineer default group of options (-ampu)
 807         -M      do not run pmodes (safe file permission checker)
 808         -N      do not run protocmp
 809         -R      default group of options for building a release (-mp)
 810         -U      update proto area in the parent
 811         -V VERS set the build version string to VERS
 812         -X      copy x86 IHV proto area
 813         -f      find unreferenced files
 814         -i      do an incremental build (no "make clobber")
 815         -l      do "make lint" in $LINTDIRS (default: $SRC y)
 816         -m      send mail to $MAILTO at end of build
 817         -n      do not do a bringover
 818         -o      build using root privileges to set OWNER/GROUP (old style)
 819         -p      create packages
 820         -r      check ELF runtime attributes in the proto area
 821         -t      build and use the tools in $SRC/tools (default setting)
 822         +t      Use the build tools in $ONBLD_TOOLS/bin
 823         -u      update proto_list_$MACH and friends in the parent workspace;
 824                 when used with -f, also build an unrefmaster.out in the parent
 825         -w      report on differences between previous and current proto areas
 826         -z      compress cpio archives with gzip
 827         -W      Do not report warnings (freeware gate ONLY)
 828 ’
 829 #
 830 #       A log file will be generated under the name $LOGFILE
 831 #       for partially completed build and log.‘date ’+%F’‘
```



```
 832 #       in the same directory for fully completed builds.
 833 #

 835 # default values for low-level FLAGS; G I R are group FLAGS
 836 A_FLAG=n
 837 C_FLAG=n
 838 D_FLAG=n
 839 F_FLAG=n
 840 f_FLAG=n
 841 i_FLAG=n; i_CMD_LINE_FLAG=n
 842 l_FLAG=n
 843 M_FLAG=n
 844 m_FLAG=n
 845 N_FLAG=n
 846 n_FLAG=n
 847 o_FLAG=n
 848 P_FLAG=n
 849 p_FLAG=n
 850 r_FLAG=n
 851 T_FLAG=n
 852 t_FLAG=y
 853 U_FLAG=n
 854 u_FLAG=n
 855 V_FLAG=n
 856 W_FLAG=n
 857 w_FLAG=n
 858 X_FLAG=n
 859 #
 860 XMOD_OPT=
 861 #
 862 build_ok=y

 864 #
 865 # examine arguments
 866 #

 868 OPTIND=1
 869 while getopts +intV: FLAG
 870 do
 871         case $FLAG in
 872           i )   i_FLAG=y; i_CMD_LINE_FLAG=y
 873                 ;;
 874           n )   n_FLAG=y
 875                 ;;
 876          +t )   t_FLAG=n
 877                 ;;
 878           V )   V_FLAG=y
 879                 V_ARG="$OPTARG"
 880                 ;;
 881          \? )   echo "$USAGE"
 882                 exit 1
 883                 ;;
 884         esac
 885 done

 887 # correct argument count after options
 888 shift ‘expr $OPTIND - 1‘

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

 896 # check if user is running nightly as root
 897 # ISUSER is set non-zero if an ordinary user runs nightly, or is zero
```

```
 898 # when root invokes nightly.
 899 /usr/bin/id | grep '^uid=0(' >/dev/null 2>&1
 900 ISUSER=$?;      export ISUSER

 902 #
 903 # force locale to C
 904 LC_COLLATE=C;   export LC_COLLATE
 905 LC_CTYPE=C;     export LC_CTYPE
 906 LC_MESSAGES=C;  export LC_MESSAGES
 907 LC_MONETARY=C;  export LC_MONETARY
 908 LC_NUMERIC=C;   export LC_NUMERIC
 909 LC_TIME=C;      export LC_TIME

 911 # clear environment variables we know to be bad for the build
 912 unset LD_OPTIONS
 913 unset LD_AUDIT          LD_AUDIT_32             LD_AUDIT_64
 914 unset LD_BIND_NOW       LD_BIND_NOW_32          LD_BIND_NOW_64
 915 unset LD_BREADTH        LD_BREADTH_32           LD_BREADTH_64
 916 unset LD_CONFIG         LD_CONFIG_32            LD_CONFIG_64
 917 unset LD_DEBUG          LD_DEBUG_32             LD_DEBUG_64
 918 unset LD_DEMANGLE       LD_DEMANGLE_32          LD_DEMANGLE_64
 919 unset LD_FLAGS          LD_FLAGS_32             LD_FLAGS_64
 920 unset LD_LIBRARY_PATH   LD_LIBRARY_PATH_32      LD_LIBRARY_PATH_64
 921 unset LD_LOADFLTR       LD_LOADFLTR_32          LD_LOADFLTR_64
 922 unset LD_NOAUDIT        LD_NOAUDIT_32           LD_NOAUDIT_64
 923 unset LD_NOAUXFLTR      LD_NOAUXFLTR_32         LD_NOAUXFLTR_64
 924 unset LD_NOCONFIG       LD_NOCONFIG_32          LD_NOCONFIG_64
 925 unset LD_NODIRCONFIG    LD_NODIRCONFIG_32       LD_NODIRCONFIG_64
 926 unset LD_NODIRECT       LD_NODIRECT_32          LD_NODIRECT_64
 927 unset LD_NOLAZYLOAD     LD_NOLAZYLOAD_32        LD_NOLAZYLOAD_64
 928 unset LD_NOOBJALTER     LD_NOOBJALTER_32        LD_NOOBJALTER_64
 929 unset LD_NOVERSION      LD_NOVERSION_32         LD_NOVERSION_64
 930 unset LD_ORIGIN         LD_ORIGIN_32            LD_ORIGIN_64
 931 unset LD_PRELOAD        LD_PRELOAD_32           LD_PRELOAD_64
 932 unset LD_PROFILE        LD_PROFILE_32           LD_PROFILE_64

 934 unset CONFIG
 935 unset GROUP
 936 unset OWNER
 937 unset REMOTE
 938 unset ENV
 939 unset ARCH
 940 unset CLASSPATH
 941 unset NAME

 943 #
 944 # To get ONBLD_TOOLS from the environment, it must come from the env file.
 945 # If it comes interactively, it is generally TOOLS_PROTO, which will be
 946 # clobbered before the compiler version checks, which will therefore fail.
 947 #
 948 unset ONBLD_TOOLS

 950 #
 951 #       Setup environmental variables
 952 #
 953 if [ -f /etc/nightly.conf ]; then
 954         . /etc/nightly.conf
 955 fi

 957 if [ -f $1 ]; then
 958         if [[ $1 = */* ]]; then
 959                 . $1
 960         else
 961                 . ./$1
 962         fi
 963 else
```



```
 964         if [ -f $OPTHOME/onbld/env/$1 ]; then
 965                 . $OPTHOME/onbld/env/$1
 966         else
 967                 echo "Cannot find env file as either $1 or $OPTHOME/onbld/env/$1
 968                 exit 1
 969         fi
 970 fi

 972 # contents of stdenv.sh inserted after next line:
 973 # STDENV_START
 974 # STDENV_END

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

 986 #
 987 # place ourselves in a new task, respecting BUILD_PROJECT if set.
 988 #
 989 if [ -z "$BUILD_PROJECT" ]; then
 990         /usr/bin/newtask -c $$
 991 else
 992         /usr/bin/newtask -c $$ -p $BUILD_PROJECT
 993 fi

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

 998 #
 999 # See if NIGHTLY_OPTIONS is set
1000 #
1001 if [ "$NIGHTLY_OPTIONS" = "" ]; then
1002         NIGHTLY_OPTIONS="-aBm"
1003 fi

1005 #
1006 # If BRINGOVER_WS was not specified, let it default to CLONE_WS
1007 #
1008 if [ "$BRINGOVER_WS" = "" ]; then
1009         BRINGOVER_WS=$CLONE_WS
1010 fi

1012 #
1013 # If CLOSED_BRINGOVER_WS was not specified, let it default to CLOSED_CLONE_WS
1014 #
1015 if [ "$CLOSED_BRINGOVER_WS" = "" ]; then
1016         CLOSED_BRINGOVER_WS=$CLOSED_CLONE_WS
1017 fi

1019 #
1020 # If BRINGOVER_FILES was not specified, default to usr
1021 #
1022 if [ "$BRINGOVER_FILES" = "" ]; then
1023         BRINGOVER_FILES="usr"
1024 fi

1026 check_closed_tree

1028 #
1029 # 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 +ABCDdFfGIilMmNnoPpRrTtUuWwXxz FLAG $NIGHTLY_OPTIONS
do
        case $FLAG in
          A )   A_FLAG=y
                ;;
          B )   D_FLAG=y
                ;; # old version of D
          C )   C_FLAG=y
                ;;
          D )   D_FLAG=y
                ;;
          F )   F_FLAG=y
                ;;
          f )   f_FLAG=y
                ;;
          G )   u_FLAG=y
                ;;
          I )   m_FLAG=y
                p_FLAG=y
                u_FLAG=y
                ;;
          i )   i_FLAG=y
                ;;
          l )   l_FLAG=y
                ;;
          M )   M_FLAG=y
                ;;
          m )   m_FLAG=y
                ;;
          N )   N_FLAG=y
                ;;
          n )   n_FLAG=y
                ;;
          o )   o_FLAG=y
                ;;
          P )   P_FLAG=y
                ;; # obsolete
          p )   p_FLAG=y
                ;;
          R )   m_FLAG=y
                p_FLAG=y
                ;;
          r )   r_FLAG=y
                ;;
          T )   T_FLAG=y
                ;; # obsolete
         +t )   t_FLAG=n
                ;;
          U )   if [ -z "${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
                ;;
          X )   # now that we no longer need realmode builds, just
                # copy IHV packages.  only meaningful on x86.
                if [ "$MACH" = "i386" ]; then
                        X_FLAG=y
                fi
                ;;
          x )   XMOD_OPT="-x"
                ;;
         \? )   echo "$USAGE"
                exit 1
                ;;
        esac
done

if [ $ISUSER -ne 0 ]; then
        if [ "$o_FLAG" = "y" ]; then
                echo "Old-style build requires root permission."
                exit 1
        fi

        # 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:$OPTHOME/SUNWspro/bin:$TEAMWARE/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"

unset CH
if [ "$o_FLAG" = "y" ]; then
# root invoked old-style build -- make sure it works as it always has
# by exporting 'CH'.  The current Makefile.master doesn't use this, but
# the old ones still do.
        PROTOCMPTERSE="protocmp.terse"
        CH=
        export CH
else
        PROTOCMPTERSE="protocmp.terse -gu"
fi
POUND_SIGN="#"
# have we set RELEASE_DATE in our env file?
if [ -z "$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
```

```
1162 # by avoiding repeated shell invocations to evaluate Makefile.master definitions
1163 # we export o_FLAG and X_FLAG for use by makebfu, and by usr/src/pkg/Makefile
1164 export o_FLAG X_FLAG POUND_SIGN RELEASE_DATE DEV_CM

1166 maketype="distributed"
1167 if [[ -z "$MAKE" ]]; then
1168         MAKE=dmake
1169 elif [[ ! -x "$MAKE" ]]; then
1170         echo "\$MAKE is set to garbage in the environment"
1171         exit 1
1172 fi
1173 # get the dmake version string alone
1174 DMAKE_VERSION=$( $MAKE -v )
1175 DMAKE_VERSION=${DMAKE_VERSION#*: }
1176 # focus in on just the dotted version number alone
1177 DMAKE_MAJOR=$( echo $DMAKE_VERSION | \
1178         sed -e 's/.*\<\([^.]*\.[^   ]*\).*$/\1/' )
1179 # extract the second (or final) integer
1180 DMAKE_MINOR=${DMAKE_MAJOR#*.}
1181 DMAKE_MINOR=${DMAKE_MINOR%%.*}
1182 # extract the first integer
1183 DMAKE_MAJOR=${DMAKE_MAJOR%%.*}
1184 CHECK_DMAKE=${CHECK_DMAKE:-y}
1185 # x86 was built on the 12th, sparc on the 13th.
1186 if [ "$CHECK_DMAKE" = "y" -a \
1187      "$DMAKE_VERSION" != "Sun Distributed Make 7.3 2003/03/12" -a \
1188      "$DMAKE_VERSION" != "Sun Distributed Make 7.3 2003/03/13" -a \( \
1189      "$DMAKE_MAJOR" -lt 7 -o \
1190      "$DMAKE_MAJOR" -eq 7 -a "$DMAKE_MINOR" -lt 4 \) ]; then
1191         if [ -z "$DMAKE_VERSION" ]; then
1192                 echo "$MAKE is missing."
1193                 exit 1
1194         fi
1195         echo 'whence $MAKE'" version is:"
1196         echo "  ${DMAKE_VERSION}"
1197         cat <<EOF

1199 This version may not be safe for use.  Either set TEAMWARE to a better
1200 path or (if you really want to use this version of dmake anyway), add
1201 the following to your environment to disable this check:

1203   CHECK_DMAKE=n
1204 EOF
1205         exit 1
1206 fi
1207 export PATH
1208 export MAKE

1210 if [ "${SUNWSPRO}" != "" ]; then
1211         PATH="${SUNWSPRO}/bin:$PATH"
1212         export PATH
1213 fi

1215 hostname=$(uname -n)
1216 if [[ $DMAKE_MAX_JOBS != +([0-9]) || $DMAKE_MAX_JOBS -eq 0 ]]
1217 then
1218         maxjobs=
1219         if [[ -f $HOME/.make.machines ]]
1220         then
1221                 # Note: there is a hard tab and space character in the []s
1222                 # below.
1223                 egrep -i "^[    ]*$hostname[    \.]" \
1224                         $HOME/.make.machines | read host jobs
1225                 maxjobs=${jobs##*=}
1226         fi
```



```
1228         if [[ $maxjobs != +([0-9]) || $maxjobs -eq 0 ]]
1229         then
1230                 # default
1231                 maxjobs=4
1232         fi

1234         export DMAKE_MAX_JOBS=$maxjobs
1235 fi

1237 DMAKE_MODE=parallel;
1238 export DMAKE_MODE

1240 if [ -z "${ROOT}" ]; then
1241         echo "ROOT must be set."
1242         exit 1
1243 fi

1245 #
1246 # if -V flag was given, reset VERSION to V_ARG
1247 #
1248 if [ "$V_FLAG" = "y" ]; then
1249         VERSION=$V_ARG
1250 fi

1252 #
1253 # Check for IHV root for copying ihv proto area
1254 #
1255 if [ "$X_FLAG" = "y" ]; then
1256         if [ "$IA32_IHV_ROOT" = "" ]; then
1257                 echo "IA32_IHV_ROOT: must be set for copying ihv proto"
1258                 args_ok=n
1259         fi
1260         if [ ! -d "$IA32_IHV_ROOT" ]; then
1261                 echo "$IA32_IHV_ROOT: not found"
1262                 args_ok=n
1263         fi
1264         if [ "$IA32_IHV_WS" = "" ]; then
1265                 echo "IA32_IHV_WS: must be set for copying ihv proto"
1266                 args_ok=n
1267         fi
1268         if [ ! -d "$IA32_IHV_WS" ]; then
1269                 echo "$IA32_IHV_WS: not found"
1270                 args_ok=n
1271         fi
1272 fi

1274 if [[ -z "$TMPDIR_ROOT" ]]; then
1275         TMPDIR_ROOT="/tmp"
1276 fi
1277 TMPDIR="${TMPDIR_ROOT}/nightly.tmpdir.$$"
1274 TMPDIR="/tmp/nightly.tmpdir.$$"
1278 export TMPDIR
1279 rm -rf ${TMPDIR}
1280 mkdir -p $TMPDIR || exit 1
1281 chmod 777 $TMPDIR

1283 #
1284 # Keep elfsign's use of pkcs11_softtoken from looking in the user home
1285 # directory, which doesn't always work.   Needed until all build machines
1286 # have the fix for 6271754
1287 #
1288 SOFTTOKEN_DIR=$TMPDIR
1289 export SOFTTOKEN_DIR

1291 #
1292 # Tools should only be built non-DEBUG.  Keep track of the tools proto
```

```
1293 # area path relative to $TOOLS, because the latter changes in an
1294 # export build.
1295 #
1296 # TOOLS_PROTO is included below for builds other than usr/src/tools
1297 # that look for this location.  For usr/src/tools, this will be
1298 # overridden on the $MAKE command line in build_tools().
1299 #
1300 TOOLS=${SRC}/tools
1301 TOOLS_PROTO_REL=proto/root_${MACH}-nd
1302 TOOLS_PROTO=${TOOLS}/${TOOLS_PROTO_REL};        export TOOLS_PROTO

1304 unset   CFLAGS LD_LIBRARY_PATH LDFLAGS

1306 # create directories that are automatically removed if the nightly script
1307 # fails to start correctly
1308 function newdir {
1309         dir=$1
1310         toadd=
1311         while [ ! -d $dir ]; do
1312                 toadd="$dir $toadd"
1313                 dir=‘dirname $dir‘
1314         done
1315         torm=
1316         newlist=
1317         for dir in $toadd; do
1318                 if staffer mkdir $dir; then
1319                         newlist="$ISUSER $dir $newlist"
1320                         torm="$dir $torm"
1321                 else
1322                         [ -z "$torm" ] || staffer rmdir $torm
1323                         return 1
1324                 fi
1325         done
1326         newdirlist="$newlist $newdirlist"
1327         return 0
1328 }
```

_____***unchanged_portion_omitted_***