```
 # Functions #
 #############

 # Returns the latest version of a given product into LATEST
 # Arg1 is the product to looks for
 getLatest() {
  LATEST=`ls -d $1* 2>/dev/null | tail -1 `
 }

 # Add path to BUILD_PATH if product exist
 # To reset BUILD_PATH before calling it
 # Arg1 is the product to look for
 # Arg2 is the subdirectory
 buildPath() {
  product=$1
  subdirectory=$2
  getLatest ${product}
  if [ -n "${LATEST}" ]; then
   if [ -n "${subdirectory}" ]; then
    subdirectory="/${subdirectory}"
   fi
   if [ -n "${BUILD_PATH}" ]; then
    BUILD_PATH=${BUILD_PATH}:${LATEST}${subdirectory}
   else
    BUILD_PATH=${LATEST}${subdirectory}
   fi
  fi
 }


# Set the title of the windows
function setTitle() {
 title="${LOGNAME}@${HOSTNAME}"
 echo -ne "\033]0; $title \007"
}

function sudo() {
# Trick to restore correct title after exiting a sudo
 /usr/bin/sudo $*
 setTitle
}

function ssh() {
# Trick to restore correct title after exiting a ssh
 /usr/bin/ssh $*
 setTitle
}


########
# Main #
########

# System environment
export PATH=/bin:/usr/bin:/usr/sbin:/sbin:/usr/ccs/bin:/usr/local/bin:/usr/ucb

OS_VERSION_MAJOR=`lsb_release -r | cut  -f2 | cut -d"." -f1`
OS_KERNEL_NAME=`uname -s`
```

```
OS_VER=`uname -r | cut -d "." -f 1,2 | sed 's/\./0/g'`
OS_BITS=`getconf LONG_BIT`

if [ ${OS_BITS} -eq 32 ]; then
 export LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib
else
 export LD_LIBRARY_PATH=/lib${OS_BITS}:/usr/lib${OS_BITS}:/usr/local/lib${OS_BITS}
fi

export MANPATH=/usr/share/man
export PS1='\[\e[0;36m\]\u@\h \w> \[\e[m\]'
export HISTSIZE=2000
export HISTTIMEFORMAT="%F %T "


# Production environment
if [ -d /files/ctt_far/linux ]; then
 export PRODUCTS_DIR=/files/ctt_far/linux
 tools_dir=${PRODUCTS_DIR}/tools
elif [ -d /export/products ]; then
 export PRODUCTS_DIR=/export/products
 tools_dir=${PRODUCTS_DIR}
elif [ -d ${HOME}/versions ]; then
 export PRODUCTS_DIR=${HOME}/versions
 tools_dir=${PRODUCTS_DIR}
fi


# Development environment
if [ -d /export/home/${LOGNAME} ]; then
 export HOMELOCAL=/export/home/${LOGNAME}
else
 export HOMELOCAL=${HOME}
fi

if [ -d ${HOMELOCAL}/main ]; then
 export MAIN_DIR=${HOMELOCAL}/main
fi

GCC=4.8.2
GCC_VERSION=`echo gcc-${GCC} | sed 's/\./0/g'`
DEFAULT_PLATFORM="${OS_KERNEL_NAME}AS${OS_VERSION_MAJOR}-${OS_VER}_${GCC_VERSION}_${OS_BITS}"

if [ -f ${PRODUCTS_DIR}/toolchain/${DEFAULT_PLATFORM}/env-gcc.sh ]; then
 source ${PRODUCTS_DIR}/toolchain/${DEFAULT_PLATFORM}/env-gcc.sh
 toolchain_dir=${PRODUCTS_DIR}/toolchain/${DEFAULT_PLATFORM}
else
 toolchain_dir=${PRODUCTS_DIR}
fi

export COMPIL_MODE="release"
export CVSROOT=:pserver:${LOGNAME}@cvsprosrc01.fr.world.socgen:/front


# Tools
BUILD_PATH=.
buildPath ${tools_dir}/netbeans bin
buildPath ${tools_dir}/jdk bin
buildPath ${toolchain_dir}/gdb bin
buildPath ${toolchain_dir}/valgrind bin
buildPath ${toolchain_dir}/python bin
buildPath ${toolchain_dir}/gnuplot bin
buildPath ${toolchain_dir}/openssl bin
export PATH=${BUILD_PATH}:${PATH}
```

```
# Intel C++ Studio XE
if [ -f /opt/intel/vtune_amplifier_xe_2013/amplxe-vars.sh ]; then
 for conf in /opt/intel/*xe/*-vars.sh
 do
  source ${conf}
 done
 source /opt/intel/bin/compilervars.sh intel64
fi


# Core dump management
ulimit -c unlimited


# Set the title of the windows
setTitle


# Alias
alias ls='ls --color=tty'
alias ll='ls -lh'
alias grep='grep --color=always'
alias less='less -R'
alias vi='vim'
alias cdm='cd $MAIN_DIR'
alias pp='ps -fp $(pgrep -f "^bin/"; echo 32767)'
alias gw='egrep --color=always " E | W " *`date +%Y%m%d`*'
alias t='tail -10f `ls -t | head -1`'
alias cgrep='grep -R --include="*.cpp" --include="*.h" --include="*.hpp"'

function g() {
 grep $* `ls -t | head -1`
}


# Loading personal bashrc
if [ -f ${HOME}/.bashrc.${LOGNAME} ]; then
 source ${HOME}/.bashrc.${LOGNAME}
fi
```

```
forceYes="false"
while getopts "y" flag
do
  case $flag in
   y) forceYes="true";;
  esac
done
shift $((OPTIND-1))

if [ $# -ne 1 ]; then
 echo "Usage: `basename $0` [-y] x.y.z"
 echo " to build the tools with gcc-x.y.z"
 exit 1
fi

# Version of GCC to use
GCC_VER=$1
GCC=gcc-${GCC_VER}

PRODUCTS=/files/ctt_far/linux
BUILD_ROOT=/export/home/${LOGNAME}/build
INSTALL_DIR=${PRODUCTS}/INSTALL

# Compute the platform for this toolchain
OS_VERSION_MAJOR=`lsb_release -r | cut  -f2 | cut -d"." -f1`
OS_KERNEL_NAME=`uname -s`
OS_VER=`uname -r | cut -d "." -f 1,2 | sed 's/\./0/g'`
OS_BITS=`getconf LONG_BIT`
GCC_VERSION=`echo ${GCC} | sed 's/\./0/g'`
PLATFORM="${OS_KERNEL_NAME}AS${OS_VERSION_MAJOR}-${OS_VER}_${GCC_VERSION}_${OS_BITS}"

# Build and destination directory
BUILD_DIR=${BUILD_ROOT}/${PLATFORM}
TOOLCHAIN=${PRODUCTS}/toolchain/${PLATFORM}
ENV=${TOOLCHAIN}/env-gcc.sh

# Check if previous install exists
if [ "${forceYes}" = "false" -a -d "${TOOLCHAIN}" ]; then
 while true; do
  read -p "Destination directory [${TOOLCHAIN}] already exist. Do you want to [r]emove it, [k]eep it, or
    [c]ancel the operation ? " choice
  case ${choice} in
   [r]* ) rm -rf ${TOOLCHAIN};;
   [k]* ) break;;
   [c]* ) exit;;
   * ) echo "Please answer [r]emove, [k]eep, or [c]ancel. ";;
  esac
 done
fi

mkdir -p ${BUILD_DIR}
mkdir -p ${TOOLCHAIN}

# Tools to build
GDB=gdb-7.7
VALGRIND=valgrind-3.9.0
```

```
# Libraries to build
BOOST=boost-1.55.0
BOOST_TAR=boost_1_55_0
XERCES=xerces-c-3.1.1
OPENSSL_0=openssl-0.9.8y
OPENSSL_1=openssl-1.0.1g
KRB5=krb5-1.12.1
QUANTLIB=quantLib-1.4
QUANTLIB_TAR=QuantLib-1.4
QT=qt-4.6.3
QT_TAR=qt-everywhere-opensource-src-4.6.3
QWT=qwt-6.0.2
GNUPLOT=gnuplot-4.6.5
XLIB64=xlib64 # To symbolic link issue
LAPACK=lapack-3.5.0
PROTOBUF=protobuf-2.5.0
GMOCK=gmock-1.7.0
ZIPIOS=zipios-0.1.5
GSOAP=gsoap-2.8

# Dependencies from gcc for gdb
GMP_TAR=gmp-6.0.0a
GMP=gmp-6.0.0
MPFR=mpfr-3.1.2
MPC=mpc-1.0.2
LIBELF=libelf-0.8.13
PPL=ppl-1.1
ISL=isl-0.12.2
CLOOG=cloog-0.18.1

# Dependencies for gdb
EXPAT=expat-2.1.0
PYTHON=python-2.7.6
PYTHON_TAR=Python-2.7.6
PRETTY_PRINTER=gdb_prettyprinter-20121210

# Build options
TAR=tar
UNZIP=unzip
CONFIGURE="./configure"
MAKE="make -j3"
MAKE_CHECK="make check"
MAKE_INSTALL="make install"
ON_FAILURE="exit 1"

gcc_major=`echo ${GCC} | sed 's/\.[^.]*$//'`

# Compilo to use
. ${ENV}

# Optimization flags
export CFLAGS="-ggdb3 -DNDEBUG -O3 -march=core2 -mmmx -msse2 -mssse3 -fPIC"
export CXXFLAGS=${CFLAGS}
export LDFLAGS="-O3 -fwhole-program -fPIC"

# python
if [ ! -d "${TOOLCHAIN}/${PYTHON}" ]; then
 rm -rf ${BUILD_DIR}/${PYTHON_TAR}
 echo "Building ${PYTHON} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} zxvf ${INSTALL_DIR}/${PYTHON_TAR}.tar.gz
 cd ${PYTHON_TAR}
 ${CONFIGURE} --prefix=${TOOLCHAIN}/${PYTHON}
 ${MAKE} && ${MAKE_INSTALL} || ${ON_FAILURE}
 echo "Building ${PYTHON} end on `date`"
else
```

```
 echo "Skipping ${PYTHON}. Already built"
fi

# gdb
if [ ! -d "${TOOLCHAIN}/${GDB}" ]; then
 rm -rf ${BUILD_DIR}/${GDB}
 echo "Building ${GDB} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} jxvf ${INSTALL_DIR}/${GDB}.tar.bz2
 cd ${GDB}
 ${TAR} jxvf ${INSTALL_DIR}/${GMP_TAR}.tar.bz2
 mv ${GMP} gmp
 ${TAR} jxvf ${INSTALL_DIR}/${MPFR}.tar.bz2
 mv ${MPFR} mpfr
 ${TAR} zxvf ${INSTALL_DIR}/${MPC}.tar.gz
 mv ${MPC} mpc
 ${TAR} jxvf ${INSTALL_DIR}/${PPL}.tar.bz2
 mv ${PPL} ppl
 ${TAR} jxvf ${INSTALL_DIR}/${ISL}.tar.bz2
 mv ${ISL} isl
 ${TAR} zxvf ${INSTALL_DIR}/${CLOOG}.tar.gz
 mv ${CLOOG} cloog
 ${TAR} zxvf ${INSTALL_DIR}/${EXPAT}.tar.gz
 mv ${EXPAT} expat
 ${CONFIGURE} --prefix=${TOOLCHAIN}/${GDB} --build=x86_64-redhat-linux --enable-plugins --enable-gold=
    default --enable-ld=yes --enable-lto --enable-cloog-backend=isl --with-python=${TOOLCHAIN}/${PYTHON}
    --with-system-gdbinit=${TOOLCHAIN}/${GDB}/.gdbinit
 # Need DejaGNU to run make check
 ${MAKE} && ${MAKE_INSTALL} || ${ON_FAILURE}
 cd ${TOOLCHAIN}/${GDB}
 tar xvf ${INSTALL_DIR}/${PRETTY_PRINTER}.tar
 sed -i -e "s#GDB_HOME#${TOOLCHAIN}/${GDB}#g" -e "s#GCC_HOME#${TOOLCHAIN}/${GCC}#g" -e "s#GCC_VER#$
    {GCC_VER}#g" .gdbinit
 echo "Building ${GDB} end on `date`"
else
 echo "Skipping ${GDB}. Already built"
fi

# valgrind
if [ ! -d "${TOOLCHAIN}/${VALGRIND}" ]; then
 rm -rf ${BUILD_DIR}/${VALGRIND}
 echo "Building ${VALGRIND} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} jxvf ${INSTALL_DIR}/${VALGRIND}.tar.bz2
 cd ${VALGRIND}
 # valgrind do not support fPIC and NDEBUG
 CFLAGS_OLD=${CFLAGS}
 LDFLAGS_OLD=${LDFLAGS}
 export CFLAGS="-ggdb3 -O3 -march=core2 -mmmx -msse2 -mssse3"
 export CXXFLAGS=${CFLAGS}
 export LDFLAGS="-O3 -fwhole-program"
 ${CONFIGURE} --prefix=${TOOLCHAIN}/${VALGRIND} --build=x86_64-redhat-linux
 ${MAKE} && ${MAKE_CHECK} && ${MAKE_INSTALL} || ${ON_FAILURE}
 export CFLAGS=${CFLAGS_OLD}
 export CXXFLAGS=${CFLAGS}
 export LDFLAGS=${LDFLAGS_OLD}
 echo "Building ${VALGRIND} end on `date`"
else
 echo "Skipping ${VALGRIND}. Already built"
fi

# boost
if [ ! -d "${TOOLCHAIN}/${BOOST}" ]; then
 rm -rf ${BUILD_DIR}/${BOOST_TAR}
 echo "Building ${BOOST} begin on `date`"
 cd ${BUILD_DIR}
```

```
 ${TAR} jxvf ${INSTALL_DIR}/${BOOST_TAR}.tar.bz2
 cd ${BOOST_TAR}
 ./bootstrap.sh --prefix=${TOOLCHAIN}/${BOOST}
 ./b2 install --disable-icu
 echo "Building ${BOOST} end on `date`"
else
 echo "Skipping ${BOOST}. Already built"
fi

# xerces
if [ ! -d "${TOOLCHAIN}/${XERCES}" ]; then
 rm -rf ${BUILD_DIR}/${XERCES}
 echo "Building ${XERCES} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} zxvf ${INSTALL_DIR}/${XERCES}.tar.gz
 cd ${XERCES}
 ${CONFIGURE} --prefix=${TOOLCHAIN}/${XERCES} --disable-transcoder-icu
 ${MAKE} && ${MAKE_CHECK} && ${MAKE_INSTALL} || ${ON_FAILURE}
 echo "Building ${XERCES} end on `date`"
else
 echo "Skipping ${XERCES}. Already built"
fi

# openssl for RH5 (because of RTMA)
if [ ! -d "${TOOLCHAIN}/${OPENSSL_0}" ]; then
 rm -rf ${BUILD_DIR}/${OPENSSL_0}
 echo "Building ${OPENSSL_0} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} zxvf ${INSTALL_DIR}/${OPENSSL_0}.tar.gz
 cd ${OPENSSL_0}
 ./config --prefix=${TOOLCHAIN}/${OPENSSL_0}
 # Does not support make -j3
 make && ${MAKE_INSTALL} || ${ON_FAILURE}
 echo "Building ${OPENSSL_0} end on `date`"
else
 echo "Skipping ${OPENSSL_0}. Already built"
fi

# openssl for RH6 (because of RTMA)
if [ ! -d "${TOOLCHAIN}/${OPENSSL_1}" ]; then
 rm -rf ${BUILD_DIR}/${OPENSSL_1}
 echo "Building ${OPENSSL_1} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} zxvf ${INSTALL_DIR}/${OPENSSL_1}.tar.gz
 cd ${OPENSSL_1}
 ./config --prefix=${TOOLCHAIN}/${OPENSSL_1}
 # Does not support make -j3
 make && ${MAKE_INSTALL} || ${ON_FAILURE}
 echo "Building ${OPENSSL_1} end on `date`"
else
 echo "Skipping ${OPENSSL_1}. Already built"
fi

# krb5
if [ ! -d "${TOOLCHAIN}/${KRB5}" ]; then
 rm -rf ${BUILD_DIR}/${KRB5}
 echo "Building ${KRB5} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} zxvf ${INSTALL_DIR}/${KRB5}.tar.gz
 cd ${KRB5}/src
 WARN_CFLAGS_OLD=${WARN_CFLAGS}
 export WARN_CFLAGS="-Wall -Wno-error=uninitialized"
 # make check required installation to be done first. Do not run it
 ${CONFIGURE} --prefix=${TOOLCHAIN}/${KRB5}
 ${MAKE} && ${MAKE_INSTALL} || ${ON_FAILURE}
 export WARN_CFLAGS=${WARN_CFLAGS_OLD}
```

```
 echo "Building ${KRB5} end on `date`"
else
 echo "Skipping ${KRB5}. Already built"
fi

# quantlib
if [ ! -d "${TOOLCHAIN}/${QUANTLIB}_${BOOST}" ]; then
 rm -rf ${BUILD_DIR}/${QUANTLIB_TAR}
 echo "Building ${QUANTLIB} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} zxvf ${INSTALL_DIR}/${QUANTLIB_TAR}.tar.gz
 cd ${QUANTLIB_TAR}
 LD_LIBRARY_PATH_OLD=${LD_LIBRARY_PATH}
 export LD_LIBRARY_PATH=${TOOLCHAIN}/${BOOST}/lib:${LD_LIBRARY_PATH}
 ${CONFIGURE} --prefix=${TOOLCHAIN}/${QUANTLIB}_${BOOST} --with-boost-include=${TOOLCHAIN}/${BOOST}/
    include --with-boost-lib=${TOOLCHAIN}/${BOOST}/lib
 ${MAKE} && ${MAKE_CHECK} && ${MAKE_INSTALL} || ${ON_FAILURE}
 export LD_LIBRARY_PATH=${LD_LIBRARY_PATH_OLD}
 echo "Building ${QUANTLIB} end on `date`"
else
 echo "Skipping ${QUANTLIB}. Already built"
fi

# qt
if [ ! -d "${TOOLCHAIN}/${QT}" ]; then
 rm -rf ${BUILD_DIR}/${QT_TAR}
 echo "Building ${QT} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} zxvf ${INSTALL_DIR}/${QT_TAR}.tar.gz
 cd ${QT_TAR}
 CFLAGS_OLD=${CFLAGS}
 export CFLAGS="${CFLAGS} -fpermissive"
 export CXXFLAGS=${CFLAGS}
 ${CONFIGURE} --prefix=${TOOLCHAIN}/${QT} --opensource -confirm-license -no-pch
 ${MAKE} && ${MAKE_CHECK} && ${MAKE_INSTALL} || ${ON_FAILURE}
 export CFLAGS=${CFLAGS_OLD}
 export CXXFLAGS=${CFLAGS}
 echo "Building ${QT} end on `date`"
else
 echo "Skipping ${QT}. Already built"
fi

# qwt
if [ ! -d "${TOOLCHAIN}/${QWT}_${QT}" ]; then
 rm -rf ${BUILD_DIR}/${QWT}
 echo "Building ${QWT} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} jxvf ${INSTALL_DIR}/${QWT}.tar.bz2
 cd ${QWT}
 sed -i "s#QWT_INSTALL_PREFIX[  ]*=[    ]*/usr/local/.*#QWT_INSTALL_PREFIX=${TOOLCHAIN}/${QWT}_${QT}#"
    qwtconfig.pri
 ${TOOLCHAIN}/${QT}/bin/qmake qwt.pro
 ${MAKE} && ${MAKE_CHECK} && ${MAKE_INSTALL} || ${ON_FAILURE}
 echo "Building ${QWT} end on `date`"
else
 echo "Skipping ${QWT}. Already built"
fi

# xlib64 (only symbolic link to fix)
if [ ! -d "${TOOLCHAIN}/${XLIB64}" ]; then
 echo "Building ${XLIB64} begin on `date`"
 cd ${TOOLCHAIN}
 tar xvf ${INSTALL_DIR}/${XLIB64}.tar
 echo "Building $${XLIB64} end on `date`"
else
 echo "Skipping ${XLIB64}. Already built"
```

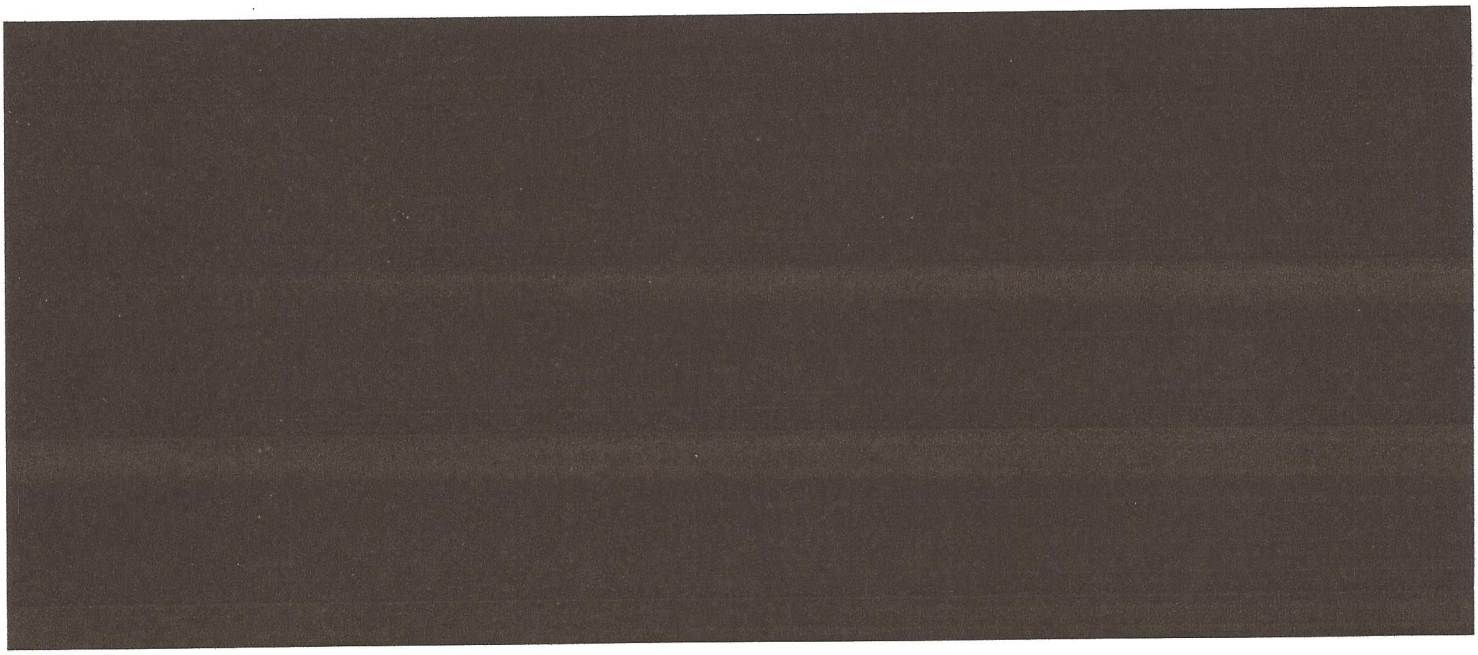
```
fi

# gnuplot
if [ ! -d "${TOOLCHAIN}/${GNUPLOT}" ]; then
 rm -rf ${BUILD_DIR}/${GNUPLOT}
 echo "Building ${GNUPLOT} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} zxvf ${INSTALL_DIR}/${GNUPLOT}.tar.gz
 cd ${GNUPLOT}
 LDFLAGS_OLD=${LDFLAGS}
 export LDFLAGS="${LDFLAGS} -L${TOOLCHAIN}/${XLIB64}/lib"
 ${CONFIGURE} --prefix=${TOOLCHAIN}/${GNUPLOT}
 ${MAKE} && ${MAKE_CHECK} && ${MAKE_INSTALL} || ${ON_FAILURE}
 export LDFLAGS=${LDFLAGS_OLD}
 echo "Building ${GNUPLOT} end on `date`"
else
 echo "Skipping ${GNUPLOT}. Already built"
fi

# lapack (and BLAS)
if [ ! -d "${TOOLCHAIN}/${LAPACK}" ]; then
 rm -rf ${BUILD_DIR}/${LAPACK}
 echo "Building ${LAPACK} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} zxvf ${INSTALL_DIR}/${LAPACK}.tgz
 cd ${LAPACK}
 cp -p make.inc.example make.inc
 ${MAKE} blaslib
 ${MAKE} all || ${ON_FAILURE}
 mkdir -p ${TOOLCHAIN}/${LAPACK}/include ${TOOLCHAIN}/${LAPACK}/lib
 cp -p *.a ${TOOLCHAIN}/${LAPACK}/lib
 tar c lapacke/include | tar x -C ${TOOLCHAIN}/${LAPACK}/include
 echo "Building ${LAPACK} end on `date`"
else
 echo "Skipping ${LAPACK}. Already built"
fi

# protobuf
if [ ! -d "${TOOLCHAIN}/${PROTOBUF}" ]; then
 rm -rf ${BUILD_DIR}/${PROTOBUF}
 echo "Building ${PROTOBUF} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} xvfj ${INSTALL_DIR}/${PROTOBUF}.tar.bz2
 cd ${PROTOBUF}
 ${CONFIGURE} --prefix=${TOOLCHAIN}/${PROTOBUF}
 ${MAKE} install || ${ON_FAILURE}
 echo "Building ${PROTOBUF} end on `date`"
else
 echo "Skipping ${PROTOBUF}. Already built"
fi

# gmock/gtest
if [ ! -d "${TOOLCHAIN}/${GMOCK}" ]; then
 rm -rf ${BUILD_DIR}/${GMOCK}
 echo "Building ${GMOCK} begin on `date`"
 cd ${BUILD_DIR}
 ${UNZIP} ${INSTALL_DIR}/${GMOCK}.zip
 cd ${GMOCK}
 ${CONFIGURE} # make install not available see README
 ${MAKE} && (mv ${BUILD_DIR}/${GMOCK} ${TOOLCHAIN}) || ${ON_FAILURE}
 echo "Building ${GMOCK} end on `date`"
else
 echo "Skipping ${GMOCK}. Already built"
fi

# zipios++
```

```
if [ ! -d "${TOOLCHAIN}/${ZIPIOS}" ]; then
 rm -rf ${BUILD_DIR}/${ZIPIOS}
 echo "Building ${ZIPIOS} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} xvf ${INSTALL_DIR}/${ZIPIOS}.tar.gz
 cd ${ZIPIOS}
 ${CONFIGURE}  --prefix=${TOOLCHAIN}/${ZIPIOS}
 ${MAKE} && ${MAKE_INSTALL} || ${ON_FAILURE}
 echo "Building ${ZIPIOS} end on `date`"
else
 echo "Skipping ${ZIPIOS}. Already built"
fi

# gsoap
if [ ! -d "${TOOLCHAIN}/${GSOAP}" ]; then
 rm -rf ${BUILD_DIR}/${GSOAP}
 echo "Building ${GSOAP} begin on `date`"
 cd ${BUILD_DIR}
 ${UNZIP} ${INSTALL_DIR}/${GSOAP}.zip
 cd ${GSOAP}
 # link error with -fwhole-program
 LDFLAGS_OLD=${LDFLAGS}
 export LDFLAGS="-O3 -fPIC"
 ${CONFIGURE}  --prefix=${TOOLCHAIN}/${GSOAP}
 make && ${MAKE_INSTALL} || ${ON_FAILURE}
 export LDFLAGS=${LDFLAGS_OLD}
 echo "Building ${GSOAP} end on `date`"
else
 echo "Skipping ${GSOAP}. Already built"
fi


# Cleanup
rm -rf ${BUILD_DIR}
```

```
# Options parsing
forceYes="false"
while getopts "y" flag
do
  case $flag in
   y) forceYes="true";;
  esac
done
shift $((OPTIND-1))


if [ $# -ne 1 ]; then
 echo "Usage: `basename $0` [-y] x.y.z"
 echo " to build gcc-x.y.z"
 exit 1
fi

# Version of GCC to build
GCC=gcc-$1

PRODUCTS=/files/ctt_far/linux
BUILD_ROOT=/export/home/${LOGNAME}/build
INSTALL_DIR=${PRODUCTS}/INSTALL

# Compute the platform for this toolchain
OS_VERSION_MAJOR=`lsb_release -r | cut  -f2 | cut -d"." -f1`
OS_KERNEL_NAME=`uname -s`
OS_VER=`uname -r | cut -d "." -f 1,2 | sed 's/\./0/g'`
OS_BITS=`getconf LONG_BIT`
GCC_VERSION=`echo ${GCC} | sed 's/\./0/g'`
PLATFORM="${OS_KERNEL_NAME}AS${OS_VERSION_MAJOR}-${OS_VER}_${GCC_VERSION}_${OS_BITS}"

# Build and destination directory
BUILD_DIR=${BUILD_ROOT}/${PLATFORM}
TOOLCHAIN=${PRODUCTS}/toolchain/${PLATFORM}
DESCRIPTION=${TOOLCHAIN}/build-gcc.txt
ENV=${TOOLCHAIN}/env-gcc.sh

# Check if previous install exists
if [ "${forceYes}" = "false" -a -d "${TOOLCHAIN}" ]; then
 while true; do
  read -p "Destination directory [${TOOLCHAIN}] already exist. Do you want to [r]emove it, [k]eep it, or ↙
    [c]ancel the operation ? " choice
  case ${choice} in
   [r]* ) rm -rf ${TOOLCHAIN};;
   [k]* ) break;;
   [c]* ) exit;;
   * ) echo "Please answer [r]emove, [k]eep, or [c]ancel. ";;
  esac
 done
fi

mkdir -p ${BUILD_DIR}
mkdir -p ${TOOLCHAIN}

# Dependencies
GMP_TAR=gmp-6.0.0a
GMP=gmp-6.0.0
```

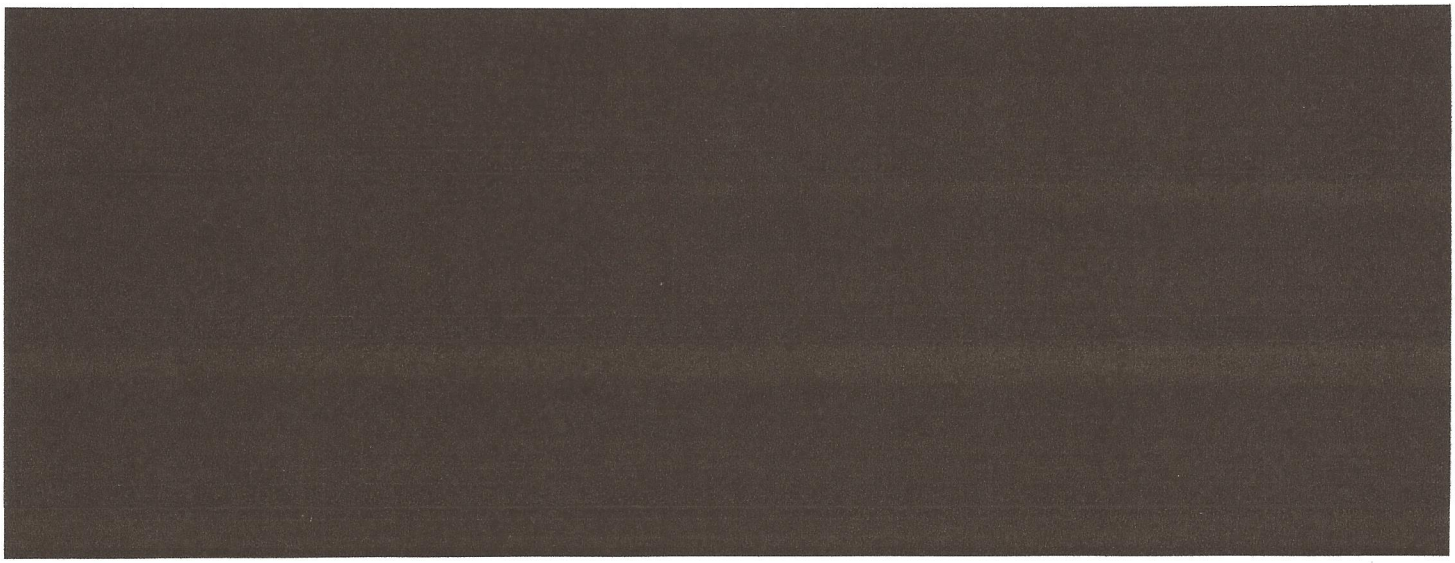
```
MPFR=mpfr-3.1.2
MPC=mpc-1.0.2
LIBELF=libelf-0.8.13
PPL=ppl-1.1
ISL=isl-0.12.2
CLOOG=cloog-0.18.1
BINUTILS=binutils-2.24


# Build options
TAR=tar
CONFIGURE="./configure"
MAKE="make -j3"
MAKE_CHECK="make check"
MAKE_INSTALL="make install"
ON_FAILURE="exit 1"

# Optimization flags
export CFLAGS="-ggdb3 -DNDEBUG -O3 -march=core2 -mmmx -msse2 -mssse3 -fPIC"
export CXXFLAGS=${CFLAGS}
export LDFLAGS="-O3 -fwhole-program -fPIC"

# binutils
if [ ! -d "${TOOLCHAIN}/${BINUTILS}" ]; then
 rm -rf ${BUILD_DIR}/${BINUTILS}
 echo "Building ${BINUTILS} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} jxvf ${INSTALL_DIR}/${BINUTILS}.tar.bz2
 cd ${BINUTILS}
 ${TAR} jxvf ${INSTALL_DIR}/${GMP_TAR}.tar.bz2
 mv ${GMP} gmp
 ${TAR} jxvf ${INSTALL_DIR}/${MPFR}.tar.bz2
 mv ${MPFR} mpfr
 ${TAR} zxvf ${INSTALL_DIR}/${MPC}.tar.gz
 mv ${MPC} mpc
 ${TAR} jxvf ${INSTALL_DIR}/${PPL}.tar.bz2
 mv ${PPL} ppl
 ${TAR} jxvf ${INSTALL_DIR}/${ISL}.tar.bz2
 mv ${ISL} isl
 ${TAR} zxvf ${INSTALL_DIR}/${CLOOG}.tar.gz
 mv ${CLOOG} cloog
 ${CONFIGURE} --prefix=${TOOLCHAIN}/${BINUTILS} --build=x86_64-redhat-linux --enable-plugins --enable-
    gold=default --enable-ld=yes --enable-lto --enable-cloog-backend=isl
 ${MAKE} && ${MAKE_INSTALL} || ${ON_FAILURE}
 echo "Building ${BINUTILS} end on `date`"
else
 echo "Skipping ${BINUTILS}. Already built"
fi

# gcc
if [ ! -d "${TOOLCHAIN}/${GCC}" ]; then
 rm -rf ${BUILD_DIR}/${GCC}
 rm -rf ${BUILD_DIR}/${GCC}-obj
 echo "Building ${GCC} begin on `date`"
 cd ${BUILD_DIR}
 ${TAR} jxvf ${INSTALL_DIR}/${GCC}.tar.bz2
 cd ${GCC}
 ${TAR} jxvf ${INSTALL_DIR}/${GMP_TAR}.tar.bz2
 mv ${GMP} gmp
 ${TAR} jxvf ${INSTALL_DIR}/${MPFR}.tar.bz2
 mv ${MPFR} mpfr
 ${TAR} zxvf ${INSTALL_DIR}/${MPC}.tar.gz
 mv ${MPC} mpc
 ${TAR} jxvf ${INSTALL_DIR}/${PPL}.tar.bz2
 mv ${PPL} ppl
 ${TAR} jxvf ${INSTALL_DIR}/${ISL}.tar.bz2
```

```
 mv ${ISL} isl
 ${TAR} zxvf ${INSTALL_DIR}/${CLOOG}.tar.gz
 mv ${CLOOG} cloog
 ${TAR} zxvf ${INSTALL_DIR}/${LIBELF}.tar.gz
 mv ${LIBELF} libelf
 cd ..
 mkdir ${GCC}-obj
 cd ${GCC}-obj
 export CFLAGS_OLD=${CFLAGS}
 export CFLAGS="${CFLAGS} -Wno-error=unused-variable"
 export CXXFLAGS=${CFLAGS}
 ../${GCC}/${CONFIGURE} --prefix=${TOOLCHAIN}/${GCC} --disable-shared --build=x86_64-redhat-linux --
    enable-languages=c,c++,fortran --enable-libgomp --enable-threads=posix --enable-lto --enable-ld --
    enable-cloog-backend=isl --with-fpmath=sse
`${MAKE} && ${MAKE_INSTALL} || ${ON_FAILURE}
 export CFLAGS=${CFLAGS_OLD}
 export CXXFLAGS=${CFLAGS}
 echo "Building ${GCC} end on `date`"
else
 echo "Skipping ${GCC}. Already built"
fi

# Environment file
echo '#!/bin/sh' > ${ENV}
echo "# Generated on `date`" >> ${ENV}
echo >> ${ENV}
echo "TOOLCHAIN=${TOOLCHAIN}" >> ${ENV}
echo >> ${ENV}
echo export PATH='${TOOLCHAIN}'/${GCC}/bin:'${TOOLCHAIN}'/${BINUTILS}/bin:'${PATH}' >> ${ENV}
echo export CC='${TOOLCHAIN}'/${GCC}/bin/gcc >> ${ENV}
echo export CXX='${TOOLCHAIN}'/${GCC}/bin/g++ >> ${ENV}

# Description
echo "Build: ${GCC}" > ${DESCRIPTION}
echo "Platform: ${PLATFORM}" >> ${DESCRIPTION}
echo "Host: `uname -a`" >> ${DESCRIPTION}
echo "Author: ${LOGNAME}" >> ${DESCRIPTION}
echo "Date: `date '+%Y/%m/%d %H:%M:%S'`" >> ${DESCRIPTION}
echo 'BuildScript: $Id: build-devenv-gcc.sh,v 1.28 2014/04/24 16:29:17 jguigui Exp $' >> ${DESCRIPTION}
echo >> ${DESCRIPTION}
echo "Dependencies: " >> ${DESCRIPTION}
echo ${GMP} >> ${DESCRIPTION}
echo ${MPFR} >> ${DESCRIPTION}
echo ${MPC} >> ${DESCRIPTION}
echo ${LIBELF} >> ${DESCRIPTION}
echo ${PPL} >> ${DESCRIPTION}
echo ${ISL} >> ${DESCRIPTION}
echo ${CLOOG} >> ${DESCRIPTION}
echo ${BINUTILS} >> ${DESCRIPTION}
echo >> ${DESCRIPTION}
echo "Environment:" >> ${DESCRIPTION}
echo "CFLAGS: ${CFLAGS}" >> ${DESCRIPTION}
echo "LDFLAGS: ${LDFLAGS}" >> ${DESCRIPTION}
echo "PATH: ${PATH}" >> ${DESCRIPTION}
echo "LD_LIBRARY_PATH: ${LD_LIBRARY_PATH}" >> ${DESCRIPTION}
echo >> ${DESCRIPTION}
echo "Build options:" >> ${DESCRIPTION}
${TOOLCHAIN}/${GCC}/bin/gcc -v 2>> ${DESCRIPTION}

# Cleanup
rm -rf ${BUILD_DIR}
```



```
#
# Variable expansions
#
CXXFLAGS += $(addprefix -I, $(INCDIR) $(INCLUDES) $(INCLUDES_TEST))

OBJS = $(addprefix $(OBJDIR)/,$(SRCS:.cpp=.o))
OBJS += $(addprefix $(OBJDIR)/,$(patsubst %.cpp,%.o,$(patsubst %.cc,%.o,$(GEN_SRCS))))

OBJS_TEST = $(addprefix $(OBJDIR)/,$(SRCS_TEST:.cpp=.o))
OBJS_TEST += $(addprefix $(OBJDIR)/,$(patsubst %.cpp,%.o,$(patsubst %.cc,%.o,$(GEN_SRCS_TEST))))

LD_PATH = $(addprefix -L,$(LINK_PATH))
LD_PATH_TEST = $(addprefix -L,$(LINK_PATH_TEST))

noop=
space = $(noop) $(noop)
LD_LIBRARY_PATH_TEST = $(subst $(space),:,$(LINK_PATH)):$(subst $(space),:,$(LINK_PATH_TEST))

LD_LIBS = $(addprefix -l,$(LINK_LIBS))
LD_LIBS_TEST = $(addprefix -l,$(LINK_LIBS_TEST))


#
# Targets
#

$(OBJS): | $(GEN_HEADERS)
$(OBJS_TEST): | $(GEN_HEADERS) $(GEN_HEADERS_TEST)

$(LIB): $(OBJS)
    @echo "### Linking static library $@ in $(COMPIL_MODE) mode ###"
    @[ -d $(LIBDIR) ] || mkdir -p $(LIBDIR)
    $(VERBOSITY_FLAG)$(AR) -crsv $@ $?
    @touch $(TOP)
    @echo ""

$(BIN): $(OBJS)
    @echo "### Linking binary $@ in $(COMPIL_MODE) mode ###"
    @[ -d $(BINDIR) ] || mkdir -p $(BINDIR)
    $(VERBOSITY_FLAG)$(CXX) $(LDFLAGS) -o $@ $(OBJS) $(LD_PATH) $(LD_LIBS)
    @echo ""

$(BIN_TEST): $(OBJS_TEST) $(LIB)
    @echo "### Linking test binary $@ in $(COMPIL_MODE) mode ###"
```

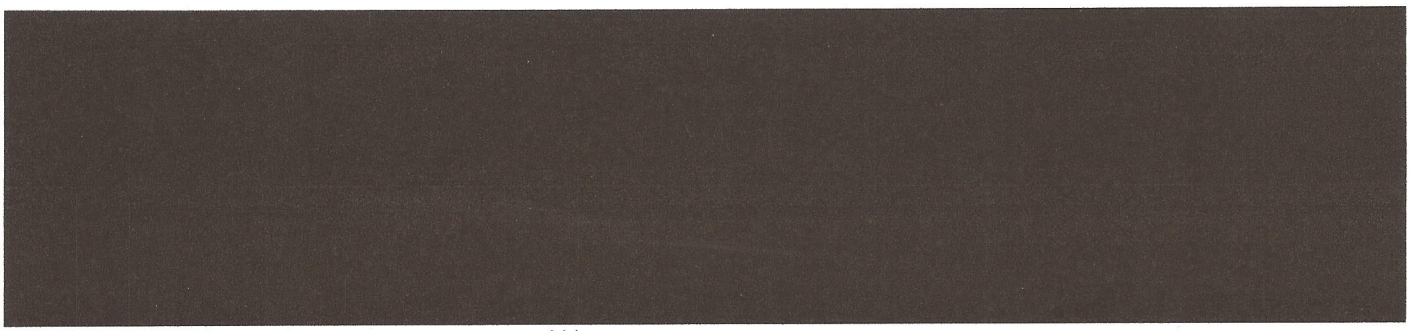
```
    @[ -d $(BINDIR) ] || mkdir -p $(BINDIR)
ifdef LIB
    $(VERBOSITY_FLAG)$(CXX) $(LDFLAGS) -o $@ $(OBJS_TEST) -L$(LIBDIR) $(LD_PATH) $(LD_PATH_TEST) -l$
    (patsubst lib%.a,%,$(notdir $(LIB))) $(LD_LIBS) $(LD_LIBS_TEST)
else
    $(VERBOSITY_FLAG)$(CXX) $(LDFLAGS) -o $@ $(OBJS_TEST) -L$(LIBDIR) $(LD_PATH) $(LD_PATH_TEST) $
    (LD_LIBS) $(LD_LIBS_TEST)
endif
    @echo ""

check: $(BIN_TEST)
    @echo "### Running test binary $< in $(COMPIL_MODE) mode ###"
    #LD_LIBRARY_PATH=$(LD_LIBRARY_PATH_TEST) valgrind --leak-check=full $(BIN_TEST) $(CHECK_ARGS)
    LD_LIBRARY_PATH=$(LD_LIBRARY_PATH_TEST) $(BIN_TEST) $(CHECK_ARGS)

check_valgrind: $(BIN_TEST)
    @echo "### Running test binary $< in $(COMPIL_MODE) mode with valgrind/memcheck ###"
    LD_LIBRARY_PATH=$(LD_LIBRARY_PATH_TEST) valgrind --leak-check=full --num-callers=40 --xml=yes --xml-
    file=memcheck.xml $(BIN_TEST) $(CHECK_ARGS)


.PHONY: clean
clean:
    @echo "### Cleaning $(COMPIL_MODE) build ###"
    $(VERBOSITY_FLAG)-rm -rf $(BUILDDIRS) $(GENDIR)
    @echo ""


# Dependencies management
# The -MD option of gcc has generated the dependencies makefiles .d which are included here
DEPENDS = $(addprefix $(OBJDIR)/,$(SRCS:.cpp=.d))
DEPENDS += $(addprefix $(OBJDIR)/,$(patsubst %.cpp,%.d,$(patsubst %.cc,%.d,$(GEN_SRCS))))
-include $(DEPENDS)

DEPENDS_TEST = $(addprefix $(OBJDIR)/,$(SRCS_TEST:.cpp=.d))
DEPENDS_TEST += $(addprefix $(OBJDIR)/,$(patsubst %.cpp,%.d,$(patsubst %.cc,%.d,$(GEN_SRCS_TEST))))
-include $(DEPENDS_TEST)


#
# Rules
#
$(OBJDIR)/%.o: $(SRCDIR)/%.cpp
    @echo "### Building $< in $(COMPIL_MODE) mode ###"
    @[ -d $(dir $@) ] || mkdir -p $(dir $@)
    $(VERBOSITY_FLAG)$(CXX) -c $(CXXFLAGS) $(CPPFLAGS) $< -o $@
    @echo ""

$(OBJDIR)/%.pb.o: $(GENDIR)/%.pb.cc
    @echo "### Building $< in $(COMPIL_MODE) mode ###"
    @[ -d $(dir $@) ] || mkdir -p $(dir $@)
    $(VERBOSITY_FLAG)$(CXX) -c $(CXXFLAGS) $(CPPFLAGS) $< -o $@
    @echo ""

$(OBJDIR)/%.o: $(GENDIR)/%.cpp
    @echo "### Building $< in $(COMPIL_MODE) mode ###"
    @[ -d $(dir $@) ] || mkdir -p $(dir $@)
    $(VERBOSITY_FLAG)$(CXX) -c $(CXXFLAGS) $(CPPFLAGS) $< -o $@
    @echo ""

$(GENDIR)/%.pb.h $(GENDIR)/%.pb.cc: $(PROTODIR)/%.proto
    @echo "### Generating $< ###"
    @[ -d $(dir $@) ] || mkdir -p $(dir $@)
    $(VERBOSITY_FLAG)$(PROTO_DIR)/bin/protoc --proto_path=$(PROTODIR) --cpp_out=$(GENDIR) $<
    @echo ""
```



```
.SUFFIXES:
.SUFFIXES: .cpp .h .hpp .o .a .d

# Initialize environment variables
OS := $(shell uname -s | tr [:upper:] [:lower:])
OS_VERSION_MAJOR := $(shell lsb_release -r | cut  -f2 | cut -d"." -f1)
OS_KERNEL_NAME := $(shell uname -s)
OS_VER := $(shell uname -r | cut -d "." -f 1,2 | sed 's/\./0/g')
OS_BITS := $(shell getconf LONG_BIT)
COMPILO := $(shell echo $(COMPILER) | cut -d"-" -f1)
COMPILO_VER := $(shell echo $(COMPILER) | cut -d"-" -f2 | sed 's/\./0/g')
PLATFORM := $(OS_KERNEL_NAME)AS$(OS_VERSION_MAJOR)-$(OS_VER)_$(COMPILO)-$(COMPILO_VER)_$(OS_BITS)
PRODUCTS_LIB := $(PRODUCTS_DIR)/lib
PRODUCTS_TOOLCHAIN := $(PRODUCTS_DIR)/toolchain/$(PLATFORM)

# Setup compiler
export LD_LIBRARY_PATH := $(shell . $(PRODUCTS_TOOLCHAIN)/env-gcc.sh && echo $$LD_LIBRARY_PATH )
export PATH :=  $(shell . $(PRODUCTS_TOOLCHAIN)/env-gcc.sh && echo $$PATH )
CC := $(PRODUCTS_TOOLCHAIN)/$(COMPILER)/bin/gcc
CXX := $(PRODUCTS_TOOLCHAIN)/$(COMPILER)/bin/g++

# Rtma version
RTMA_PLATFORM = $(shell ls $(RTMA_DIR)/common/lib | grep "$(OS_KERNEL_NAME)AS$(OS_VERSION_MAJOR)-$
    (OS_VER)_$(COMPILO)-.*_$(OS_BITS)" | tail -1 )

#
# Default CXX flags
#
# -MD is used to generate a dependency output file
# Paranoiac mode: -Weffc++ -Wold-style-cast
#WARN_CFLAGS ?= -Wall -Wno-deprecated -Wno-delete-non-virtual-dtor
WARN_CFLAGS ?= -Wall -Wno-deprecated
CXXFLAGS += -MD $(WARN_CFLAGS) -DOS_VER=$(OS_VER) -DCOMPILO=$(COMPILO) -DCOMPILO_VER=$(COMPILO_VER) -
    DPLATFORM=$(PLATFORM) -DCOMPIL_MODE=$(COMPIL_MODE)

CXXFLAGS_RELEASE = -ggdb3 -DNDEBUG -O3 -march=core2 -mmmx -msse2 -mssse3 -fPIC
LDFLAGS_RELEASE = -O3 -fwhole-program -fPIC

CXXFLAGS_DEBUG = -ggdb3 -O0 -march=core2 -mmmx -msse2 -mssse3 -fPIC
LDFLAGS_DEBUG = -fPIC

# Mode release, debug or profiling
ifeq ($(COMPIL_MODE),release)
    CXXFLAGS += $(CXXFLAGS_RELEASE)
    LDFLAGS += $(LDFLAGS_RELEASE)
    LIB_COMPIL_MODE = release
else ifeq ($(COMPIL_MODE),debug)
    CXXFLAGS += $(CXXFLAGS_DEBUG)
```

```
    LDFLAGS += $(LDFLAGS_DEBUG)
    LIB_COMPIL_MODE = debug
else ifeq ($(COMPIL_MODE),profiling)
    CXXFLAGS += $(CXXFLAGS_RELEASE) -pg
    LDFLAGS += $(LDFLAGS_RELEASE) -pg
    LIB_COMPIL_MODE = release
else ifeq ($(COMPIL_MODE),insure)
    CC=insure /export/products/gcc-4.2.4/bin/gcc
    CXX=insure /export/products/gcc-4.2.4/bin/g++
    CXXFLAGS = -DOS=$(OS) -DOS_VER=$(OS_VER) -DCOMPILO=$(COMPILO) -DCOMPILO_VER=40204 -g -DMEM_LEAK_TOOL
    LDFLAGS =
    LIB_COMPIL_MODE = debug
else ifeq ($(COMPIL_MODE),advice)
    CXXFLAGS += $(CXXFLAGS_RELEASE) -D_GLIBCXX_PROFILE
    LDFLAGS += $(LDFLAGS_RELEASE)
    LIB_COMPIL_MODE = release
else
 $(error COMPIL_MODE should be set to "release", "debug", "profiling", "insure" or "advice")
endif

# Log verbosity
VERBOSITY = verbose
ifeq ($(VERBOSITY),verbose)
    VERBOSITY_FLAG =
else ifeq ($(VERBOSITY),quiet)
    VERBOSITY_FLAG = @
else
 $(error VERBOSITY should be set to "verbose" or "quiet")
endif

# OS
ifeq ($(OS),windows)
    CXXFLAGS += -DWINDOWS
else ifeq ($(OS),linux)
    CXXFLAGS += -DUNX
else
 $(error OS should be set to "windows" or "linux")
endif

ifndef OS_VER
 $(error OS_VER must de defined. For instance 206 for Linux 2.6.18 or 501 for WinXP)
endif

#
# Directories
#
SRCDIR = $(TOP)
INCDIR = $(TOP)
OBJDIR = $(TOP)obj/$(PLATFORM)/$(COMPIL_MODE)
LIBDIR = $(TOP)lib/$(PLATFORM)/$(COMPIL_MODE)
BINDIR = $(TOP)bin/$(PLATFORM)/$(COMPIL_MODE)
BUILDDIRS = $(OBJDIR) $(LIBDIR) $(BINDIR)
```

```
COMPILER=gcc-4.8.1
include $(TOP)/../../../../Common/DevTools/Makefile.header

BOOST_DIR=$(PRODUCTS_TOOLCHAIN)/boost-1.54.0
QUANTLIB_DIR=$(PRODUCTS_TOOLCHAIN)/quantLib-1.2.1_boost-1.54.0
RTMA_DIR=$(PRODUCTS_LIB)/rtmaLib-1.18.0.2

APP_VERSION=40200

CXXFLAGS += -std=c++11 -DPS_VERSION=$(APP_VERSION) -DBOOST_NO_CXX11_EXPLICIT_CONVERSION_OPERATORS
WARN_CFLAGS = -Wall -Wno-deprecated -Wno-delete-non-virtual-dtor -Wno-unused-local-typedefs -Wno-
    overloaded-virtual -Wno-unused-variable

LDFLAGS += -rdynamic
CXXFLAGS += -DERROR_TRACING

INCLUDES += \
    $(TOP)../../CPPTools/include \
    $(BOOST_DIR)/include
```



```
TOP := $(dir $(lastword $(MAKEFILE_LIST)))

include $(TOP)../Makefile.project


INCLUDES  += \
    $(TOP)../SABRPricerLib \
    $(TOP)../RTDNSrvLight \
    $(TOP)../RTDNLightClient \
    $(TOP)../Parameters \
    $(QUANTLIB_DIR)/include \
    $(RTMA_DIR)/common/include \
    $(RTMA_DIR)/rtdn/libs/include \
    $(RTMA_DIR)/rtdn/lightfeed/include \
    $(RTMA_DIR)/RTGen/common/include

SRCS = \
    DicoLoader.cpp \
    Maturity.cpp \
    PriceServer.cpp

LINK_PATH = \
    $(TOP)../SABRPricerLib/lib/$(PLATFORM)/$(COMPIL_MODE) \
    $(TOP)../RTDNSrvLight/lib/$(PLATFORM)/$(COMPIL_MODE) \
    $(TOP)../RTDNLightClient/lib/$(PLATFORM)/$(COMPIL_MODE) \
    $(TOP)../Parameters/lib/$(PLATFORM)/$(COMPIL_MODE) \
    $(TOP)../../CPPTools/lib/$(PLATFORM)/$(COMPIL_MODE) \
    $(QUANTLIB_DIR)/lib \
    $(BOOST_DIR)/lib \
    $(RTMA_DIR)/common/lib/$(RTMA_PLATFORM)/$(LIB_COMPIL_MODE) \
    $(RTMA_DIR)/rtdn/libs/lib/$(RTMA_PLATFORM)/$(LIB_COMPIL_MODE) \
    $(RTMA_DIR)/rtdn/lightfeed/lib/$(RTMA_PLATFORM)/$(LIB_COMPIL_MODE) \
    $(RTMA_DIR)/RTGen/common/lib/$(RTMA_PLATFORM)/$(LIB_COMPIL_MODE)

LINK_LIBS = \
    SABRPricer \
    RTDNSrvLight \
    RTDNLightClient \
    Parameters \
    lightfeed \
    CPPTools \
    DistribTCPUDP \
    DistribTCP \
    Distrib \
    RTThread \
    RTSocket \
    RTGenCommon \
    common \
    RTType \
    QuantLib \
    boost_thread \
    boost_date_time \
    boost_system \
    dl \
    rt \
    pthread
```

```
BIN = $(BINDIR)/PriceServer

$(BIN) : $(TOP)../../CPPTools  $(TOP)../SABRPricerLib $(TOP)../RTDNSrvLight $(TOP)../RTDNLightClient $
    (TOP)../Parameters

.PHONY: all
all: $(BIN)

include $(TOP)/../../../../Common/DevTools/Makefile.footer
```