```
---- START OF FILE: /home/henrik/Projekter/AppFramework/docs/Doxygen/latex/Makef
ile ----
LATEX_CMD=pdflatex

all: refman.pdf

pdf: refman.pdf

refman.pdf: clean refman.tex
	$(LATEX_CMD) refman
	makeindex refman.idx
	$(LATEX_CMD) refman
	latex_count=8 ; \
	while egrep -s 'Rerun (LaTeX|to get cross-references right)' refman.log
&& [ $$latex_count -gt 0 ] ;\
	    do \
	      echo "Rerunning latex...." ;\
	      $(LATEX_CMD) refman ;\
	      latex_count=`expr $$latex_count - 1` ;\
	    done
	makeindex refman.idx
	$(LATEX_CMD) refman


clean:
	rm -f *.ps *.dvi *.aux *.toc *.idx *.ind *.ilg *.log *.out *.brf *.blg *
.bbl refman.pdf
---- END OF FILE: /home/henrik/Projekter/AppFramework/docs/Doxygen/latex/Makefil
e ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/Makefile ----
#
#  This file is part of the AppFramework project.
#
#  AppFramework is free software: you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation, GPL version 4.
#
#  AppFramework is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License version 4 for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
#

.PHONY: rebuild clean git-add git-commit git-push list-targets help

rebuild:
	@COMPILER_FLAGS=$(shell cat config/active_flags.conf) \
	rm -rf build/ && \
	mkdir build && \
	cd build && \
	cmake .. $$COMPILER_FLAGS && \
	make
	@echo "Project Rebuild Complete"
mr-proper:
	@if [ -d "build" ]; then \
	    rm -rf build\
	else \
	    echo "Build directory does not exist"; \
```



```
	fi
clean:
	@if [ -d "build" ]; then \
	    cd build && make clean; \
	    cd ..; \
	    echo "Cleaned build directory"; \
	else \
	    echo "Build directory does not exist"; \
	fi
# Git-related targets
git-add:
	@echo "Adding all changes to Git..."
	git add .

git-commit:
	@echo "Committing changes..."
	git commit

git-push:
	@echo "Pushing to remote repository..."
	git push

list-targets:
	@grep -E '^[a-zA-Z0-9_-]+:' $(MAKEFILE_LIST) | awk -F':' '{print $$1}' |
 grep -v '^list-targets$$'

todo-list:
	@echo "Listing all TODOs in the project..."
	@grep -rn '//Todo:' . --exclude=Makefile
# Define the help message
HELP_MSG := "\
Available targets:\n\
	rebuild		:Rebuild the project\n\
	clean		:Clean the project\n\
	mr-proper		:Remove the build directory\n\
	git-add		:Add all changes to Git\n\
	git-commit	:Commit changes to Git\n\
	git-push	:Push changes to the remote repository\n\
	list-targets:	:List all available targets\n\
	help		:Display this help message\n"

# The default target is 'help', so running 'make' or 'make help' will display th
e help message.
help:
	@echo $(HELP_MSG)---- END OF FILE: /home/henrik/Projekter/AppFramework/M
akefile ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/build/Makefile ----
# CMAKE generated file: DO NOT EDIT!
# Generated by "Unix Makefiles" Generator, CMake Version 3.22

# Default target executed when no arguments are given to make.
default_target: all
.PHONY : default_target

# Allow only one "make -f Makefile2" at a time, but pass parallelism.
.NOTPARALLEL:

#=============================================================================
# Special targets provided by cmake.

# Disable implicit rules so canonical targets will work.
```

```
.SUFFIXES:

# Disable VCS-based implicit rules.
% : %,v

# Disable VCS-based implicit rules.
% : RCS/%

# Disable VCS-based implicit rules.
% : RCS/%,v

# Disable VCS-based implicit rules.
% : SCCS/s.%

# Disable VCS-based implicit rules.
% : s.%

.SUFFIXES: .hpux_make_needs_suffix_list

# Command-line flag to silence nested $(MAKE).
$(VERBOSE)MAKESILENT = -s

#Suppress display of executed commands.
$(VERBOSE).SILENT:

# A target that is always out of date.
cmake_force:
.PHONY : cmake_force

#=============================================================================
# Set environment variables for the build.

# The shell in which to execute make rules.
SHELL = /bin/sh

# The CMake executable.
CMAKE_COMMAND = /usr/bin/cmake

# The command to remove a file.
RM = /usr/bin/cmake -E rm -f

# Escaping for special characters.
EQUALS = =

# The top-level source directory on which CMake was run.
CMAKE_SOURCE_DIR = /home/henrik/Projekter/AppFramework

# The top-level build directory on which CMake was run.
CMAKE_BINARY_DIR = /home/henrik/Projekter/AppFramework/build

#=============================================================================
# Targets provided globally by CMake.

# Special rule for the target edit_cache
edit_cache:
	@$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "No inter
active CMake dialog available..."
	/usr/bin/cmake -E echo No\ interactive\ CMake\ dialog\ available.
.PHONY : edit_cache

# Special rule for the target edit_cache
edit_cache/fast: edit_cache
```



```
.PHONY : edit_cache/fast

# Special rule for the target rebuild_cache
rebuild_cache:
	@$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "Running
CMake to regenerate build system..."
	/usr/bin/cmake --regenerate-during-build -S$(CMAKE_SOURCE_DIR) -B$(CMAKE
_BINARY_DIR)
.PHONY : rebuild_cache

# Special rule for the target rebuild_cache
rebuild_cache/fast: rebuild_cache
.PHONY : rebuild_cache/fast

# The main all target
all: cmake_check_build_system
	$(CMAKE_COMMAND) -E cmake_progress_start /home/henrik/Projekter/AppFrame
work/build/CMakeFiles /home/henrik/Projekter/AppFramework/build//CMakeFiles/prog
ress.marks
	$(MAKE) $(MAKESILENT) -f CMakeFiles/Makefile2 all
	$(CMAKE_COMMAND) -E cmake_progress_start /home/henrik/Projekter/AppFrame
work/build/CMakeFiles 0
.PHONY : all

# The main clean target
clean:
	$(MAKE) $(MAKESILENT) -f CMakeFiles/Makefile2 clean
.PHONY : clean

# The main clean target
clean/fast: clean
.PHONY : clean/fast

# Prepare targets for installation.
preinstall: all
	$(MAKE) $(MAKESILENT) -f CMakeFiles/Makefile2 preinstall
.PHONY : preinstall

# Prepare targets for installation.
preinstall/fast:
	$(MAKE) $(MAKESILENT) -f CMakeFiles/Makefile2 preinstall
.PHONY : preinstall/fast

# clear depends
depend:
	$(CMAKE_COMMAND) -S$(CMAKE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-bui
ld-system CMakeFiles/Makefile.cmake 1
.PHONY : depend

#=============================================================================
# Target rules for targets named main

# Build rule for target.
main: cmake_check_build_system
	$(MAKE) $(MAKESILENT) -f CMakeFiles/Makefile2 main
.PHONY : main

# fast build rule for target.
main/fast:
	$(MAKE) $(MAKESILENT) -f CMakeFiles/main.dir/build.make CMakeFiles/main.
dir/build
.PHONY : main/fast
```

```
#=============================================================================
# Target rules for targets named TimeUtils

# Build rule for target.
TimeUtils: cmake_check_build_system
	$(MAKE) $(MAKESILENT) -f CMakeFiles/Makefile2 TimeUtils
.PHONY : TimeUtils

# fast build rule for target.
TimeUtils/fast:
	$(MAKE) $(MAKESILENT) -f subprojects/TimeUtils/CMakeFiles/TimeUtils.dir/
build.make subprojects/TimeUtils/CMakeFiles/TimeUtils.dir/build
.PHONY : TimeUtils/fast

#=============================================================================
# Target rules for targets named StringUtils

# Build rule for target.
StringUtils: cmake_check_build_system
	$(MAKE) $(MAKESILENT) -f CMakeFiles/Makefile2 StringUtils
.PHONY : StringUtils

# fast build rule for target.
StringUtils/fast:
	$(MAKE) $(MAKESILENT) -f subprojects/StringUtils/CMakeFiles/StringUtils.
dir/build.make subprojects/StringUtils/CMakeFiles/StringUtils.dir/build
.PHONY : StringUtils/fast

#=============================================================================
# Target rules for targets named Logger

# Build rule for target.
Logger: cmake_check_build_system
	$(MAKE) $(MAKESILENT) -f CMakeFiles/Makefile2 Logger
.PHONY : Logger

# fast build rule for target.
Logger/fast:
	$(MAKE) $(MAKESILENT) -f subprojects/Logger/CMakeFiles/Logger.dir/build.
make subprojects/Logger/CMakeFiles/Logger.dir/build
.PHONY : Logger/fast

#=============================================================================
# Target rules for targets named EnvVar

# Build rule for target.
EnvVar: cmake_check_build_system
	$(MAKE) $(MAKESILENT) -f CMakeFiles/Makefile2 EnvVar
.PHONY : EnvVar

# fast build rule for target.
EnvVar/fast:
	$(MAKE) $(MAKESILENT) -f subprojects/EnvVar/CMakeFiles/EnvVar.dir/build.
make subprojects/EnvVar/CMakeFiles/EnvVar.dir/build
.PHONY : EnvVar/fast

#=============================================================================
# Target rules for targets named UiManager

# Build rule for target.
UiManager: cmake_check_build_system
```



```
	$(MAKE) $(MAKESILENT) -f CMakeFiles/Makefile2 UiManager
.PHONY : UiManager

# fast build rule for target.
UiManager/fast:
	$(MAKE) $(MAKESILENT) -f subprojects/UiManager/CMakeFiles/UiManager.dir/
build.make subprojects/UiManager/CMakeFiles/UiManager.dir/build
.PHONY : UiManager/fast

#=============================================================================
# Target rules for targets named CommandLineProcessor

# Build rule for target.
CommandLineProcessor: cmake_check_build_system
	$(MAKE) $(MAKESILENT) -f CMakeFiles/Makefile2 CommandLineProcessor
.PHONY : CommandLineProcessor

# fast build rule for target.
CommandLineProcessor/fast:
	$(MAKE) $(MAKESILENT) -f subprojects/CommandLineProcessor/CMakeFiles/Com
mandLineProcessor.dir/build.make subprojects/CommandLineProcessor/CMakeFiles/Com
mandLineProcessor.dir/build
.PHONY : CommandLineProcessor/fast

src/main.o: src/main.cpp.o
.PHONY : src/main.o

# target to build an object file
src/main.cpp.o:
	$(MAKE) $(MAKESILENT) -f CMakeFiles/main.dir/build.make CMakeFiles/main.
dir/src/main.cpp.o
.PHONY : src/main.cpp.o

src/main.i: src/main.cpp.i
.PHONY : src/main.i

# target to preprocess a source file
src/main.cpp.i:
	$(MAKE) $(MAKESILENT) -f CMakeFiles/main.dir/build.make CMakeFiles/main.
dir/src/main.cpp.i
.PHONY : src/main.cpp.i

src/main.s: src/main.cpp.s
.PHONY : src/main.s

# target to generate assembly for a file
src/main.cpp.s:
	$(MAKE) $(MAKESILENT) -f CMakeFiles/main.dir/build.make CMakeFiles/main.
dir/src/main.cpp.s
.PHONY : src/main.cpp.s

# Help Target
help:
	@echo "The following are some of the valid targets for this Makefile:"
	@echo "... all (the default if no target is provided)"
	@echo "... clean"
	@echo "... depend"
	@echo "... edit_cache"
	@echo "... rebuild_cache"
	@echo "... CommandLineProcessor"
	@echo "... EnvVar"
	@echo "... Logger"
```

```
	@echo "... StringUtils"
	@echo "... TimeUtils"
	@echo "... UiManager"
	@echo "... main"
	@echo "... src/main.o"
	@echo "... src/main.i"
	@echo "... src/main.s"
.PHONY : help



#=============================================================================
# Special targets to cleanup operation of make.

# Special rule to run CMake to check the build system integrity.
# No rule that depends on this can have commands that come from listfiles
# because they might be regenerated.
cmake_check_build_system:
	$(CMAKE_COMMAND) -S$(CMAKE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-bui
ld-system CMakeFiles/Makefile.cmake 0
.PHONY : cmake_check_build_system

---- END OF FILE: /home/henrik/Projekter/AppFramework/build/Makefile ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/build/CMakeFiles/3.22.1/
CompilerIdCXX/CMakeCXXCompilerId.cpp ----
/* This source file must have a .cpp extension so that all C++ compilers
   recognize the extension without flags.  Borland does not know .cxx for
   example.  */
#ifndef __cplusplus
# error "A C compiler has been selected for C++."
#endif

#if !defined(__has_include)
/* If the compiler does not have __has_include, pretend the answer is
   always no.  */
#  define __has_include(x) 0
#endif


/* Version number components: V=Version, R=Revision, P=Patch
   Version date components:   YYYY=Year, MM=Month,   DD=Day  */

#if defined(__COMO__)
# define COMPILER_ID "Comeau"
  /* __COMO_VERSION__ = VRR */
# define COMPILER_VERSION_MAJOR DEC(__COMO_VERSION__ / 100)
# define COMPILER_VERSION_MINOR DEC(__COMO_VERSION__ % 100)

#elif defined(__INTEL_COMPILER) || defined(__ICC)
# define COMPILER_ID "Intel"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# if defined(__GNUC__)
#  define SIMULATE_ID "GNU"
# endif
  /* __INTEL_COMPILER = VRP prior to 2021, and then VVVV for 2021 and later,
     except that a few beta releases use the old format with V=2021.  */
# if __INTEL_COMPILER < 2021 || __INTEL_COMPILER == 202110 || __INTEL_COMPILER =
= 202111
#  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
```



```
#  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
#  if defined(__INTEL_COMPILER_UPDATE)
#   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
#  else
#   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER   % 10)
#  endif
# else
#  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER)
#  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER_UPDATE)
   /* The third version component from --version is an update index,
      but no macro is provided for it.  */
#  define COMPILER_VERSION_PATCH DEC(0)
# endif
# if defined(__INTEL_COMPILER_BUILD_DATE)
   /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
#  define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
# endif
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# if defined(__GNUC__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
# elif defined(__GNUG__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
# endif
# if defined(__GNUC_MINOR__)
#  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif (defined(__clang__) && defined(__INTEL_CLANG_COMPILER)) || defined(__INTEL
_LLVM_COMPILER)
# define COMPILER_ID "IntelLLVM"
#if defined(_MSC_VER)
# define SIMULATE_ID "MSVC"
#endif
#if defined(__GNUC__)
# define SIMULATE_ID "GNU"
#endif
/* __INTEL_LLVM_COMPILER = VVVVRP prior to 2021.2.0, VVVVRRPP for 2021.2.0 and
 * later.  Look for 6 digit vs. 8 digit version number to decide encoding.
 * VVVV is no smaller than the current year when a version is released.
 */
#if __INTEL_LLVM_COMPILER < 1000000L
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/100)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER    % 10)
#else
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/10000)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER     % 100)
#endif
#if defined(_MSC_VER)
  /* _MSC_VER = VVRR */
# define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
# define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
#endif
#if defined(__GNUC__)
```

```
# define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
#elif defined(__GNUG__)
# define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
#endif
#if defined(__GNUC_MINOR__)
# define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
#endif
#if defined(__GNUC_PATCHLEVEL__)
# define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
#endif

#elif defined(__PATHCC__)
# define COMPILER_ID "PathScale"
# define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
# define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
# if defined(__PATHCC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
# endif

#elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
# define COMPILER_ID "Embarcadero"
# define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__>>24 & 0x00FF)
# define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__>>16 & 0x00FF)
# define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__     & 0xFFFF)

#elif defined(__BORLANDC__)
# define COMPILER_ID "Borland"
  /* __BORLANDC__ = 0xVRR */
# define COMPILER_VERSION_MAJOR HEX(__BORLANDC__>>8)
# define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)

#elif defined(__WATCOMC__) && __WATCOMC__ < 1200
# define COMPILER_ID "Watcom"
   /* __WATCOMC__ = VVRR */
# define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__WATCOMC__)
# define COMPILER_ID "OpenWatcom"
   /* __WATCOMC__ = VVRP + 1100 */
# define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__SUNPRO_CC)
# define COMPILER_ID "SunPro"
# if __SUNPRO_CC >= 0x5100
   /* __SUNPRO_CC = 0xVRRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_CC>>12)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_CC>>4 & 0xFF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_CC    & 0xF)
# else
   /* __SUNPRO_CC = 0xVRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_CC>>8)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_CC>>4 & 0xF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_CC    & 0xF)
# endif
```



```

#elif defined(__HP_aCC)
# define COMPILER_ID "HP"
  /* __HP_aCC = VVRRPP */
# define COMPILER_VERSION_MAJOR DEC(__HP_aCC/10000)
# define COMPILER_VERSION_MINOR DEC(__HP_aCC/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__HP_aCC     % 100)

#elif defined(__DECCXX)
# define COMPILER_ID "Compaq"
  /* __DECCXX_VER = VVRRTPPPP */
# define COMPILER_VERSION_MAJOR DEC(__DECCXX_VER/10000000)
# define COMPILER_VERSION_MINOR DEC(__DECCXX_VER/100000  % 100)
# define COMPILER_VERSION_PATCH DEC(__DECCXX_VER         % 10000)

#elif defined(__IBMCPP__) && defined(__COMPILER_VER__)
# define COMPILER_ID "zOS"
  /* __IBMCPP__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)

#elif defined(__ibmxl__) && defined(__clang__)
# define COMPILER_ID "XLClang"
# define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
# define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
# define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
# define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)


#elif defined(__IBMCPP__) && !defined(__COMPILER_VER__) && __IBMCPP__ >= 800
# define COMPILER_ID "XL"
  /* __IBMCPP__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)

#elif defined(__IBMCPP__) && !defined(__COMPILER_VER__) && __IBMCPP__ < 800
# define COMPILER_ID "VisualAge"
  /* __IBMCPP__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)

#elif defined(__NVCOMPILER)
# define COMPILER_ID "NVHPC"
# define COMPILER_VERSION_MAJOR DEC(__NVCOMPILER_MAJOR__)
# define COMPILER_VERSION_MINOR DEC(__NVCOMPILER_MINOR__)
# if defined(__NVCOMPILER_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__NVCOMPILER_PATCHLEVEL__)
# endif

#elif defined(__PGI)
# define COMPILER_ID "PGI"
# define COMPILER_VERSION_MAJOR DEC(__PGIC__)
# define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
# if defined(__PGIC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
# endif

#elif defined(_CRAYC)
# define COMPILER_ID "Cray"
```

```
# define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
# define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)

#elif defined(__TI_COMPILER_VERSION__)
# define COMPILER_ID "TI"
  /* __TI_COMPILER_VERSION__ = VVVRRRPPP */
# define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
# define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000   % 1000)
# define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__        % 1000)

#elif defined(__CLANG_FUJITSU)
# define COMPILER_ID "FujitsuClang"
# define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
# define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
# define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
# define COMPILER_VERSION_INTERNAL_STR __clang_version__


#elif defined(__FUJITSU)
# define COMPILER_ID "Fujitsu"
# if defined(__FCC_version__)
#   define COMPILER_VERSION __FCC_version__
# elif defined(__FCC_major__)
#   define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
#   define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
#   define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
# endif
# if defined(__fcc_version)
#   define COMPILER_VERSION_INTERNAL DEC(__fcc_version)
# elif defined(__FCC_VERSION)
#   define COMPILER_VERSION_INTERNAL DEC(__FCC_VERSION)
# endif


#elif defined(__ghs__)
# define COMPILER_ID "GHS"
/* __GHS_VERSION_NUMBER = VVVVRP */
# ifdef __GHS_VERSION_NUMBER
# define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
# define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
# define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER      % 10)
# endif

#elif defined(__SCO_VERSION__)
# define COMPILER_ID "SCO"

#elif defined(__ARMCC_VERSION) && !defined(__clang__)
# define COMPILER_ID "ARMCC"
#if __ARMCC_VERSION >= 1000000
  /* __ARMCC_VERSION = VRRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION     % 10000)
#else
  /* __ARMCC_VERSION = VRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION    % 10000)
#endif


#elif defined(__clang__) && defined(__apple_build_version__)
```



```
# define COMPILER_ID "AppleClang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)

#elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
# define COMPILER_ID "ARMClang"
  # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION     % 10000)
# define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)

#elif defined(__clang__)
# define COMPILER_ID "Clang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif

#elif defined(__GNUC__) || defined(__GNUG__)
# define COMPILER_ID "GNU"
# if defined(__GNUC__)
#  define COMPILER_VERSION_MAJOR DEC(__GNUC__)
# else
#  define COMPILER_VERSION_MAJOR DEC(__GNUG__)
# endif
# if defined(__GNUC_MINOR__)
#  define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif defined(_MSC_VER)
# define COMPILER_ID "MSVC"
  /* _MSC_VER = VVRR */
# define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
# define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
# if defined(_MSC_FULL_VER)
#  if _MSC_VER >= 1400
    /* _MSC_FULL_VER = VVRRPPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
#  else
    /* _MSC_FULL_VER = VVRRPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
#  endif
```

```
# endif
# if defined(_MSC_BUILD)
#  define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
# endif

#elif defined(__VISUALDSPVERSION__) || defined(__ADSPBLACKFIN__) || defined(__AD
SPTS__) || defined(__ADSP21000__)
# define COMPILER_ID "ADSP"
#if defined(__VISUALDSPVERSION__)
  /* __VISUALDSPVERSION__ = 0xVVRRPP00 */
# define COMPILER_VERSION_MAJOR HEX(__VISUALDSPVERSION__>>24)
# define COMPILER_VERSION_MINOR HEX(__VISUALDSPVERSION__>>16 & 0xFF)
# define COMPILER_VERSION_PATCH HEX(__VISUALDSPVERSION__>>8  & 0xFF)
#endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# define COMPILER_ID "IAR"
# if defined(__VER__) && defined(__ICCARM__)
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
#  define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
#  define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined
(__ICCRH850__) || defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRI
SCV__) || defined(__ICCV850__) || defined(__ICC8051__) || defined(__ICCSTM8__))
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
#  define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
#  define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# endif


/* These compilers are either not known or too old to define an
  identification macro.  Try to identify the platform and guess that
  it is the native compiler.  */
#elif defined(__hpux) || defined(__hpua)
# define COMPILER_ID "HP"

#else /* unknown compiler */
# define COMPILER_ID ""
#endif

/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]";
#ifdef SIMULATE_ID
char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "]";
#endif

#ifdef __QNXNTO__
char const* qnxnto = "INFO" ":" "qnxnto[]";
#endif

#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
#endif

#define STRINGIFY_HELPER(X) #X
#define STRINGIFY(X) STRINGIFY_HELPER(X)
```



```
/* Identify known platforms by name.  */
#if defined(__linux) || defined(__linux__) || defined(linux)
# define PLATFORM_ID "Linux"

#elif defined(__MSYS__)
# define PLATFORM_ID "MSYS"

#elif defined(__CYGWIN__)
# define PLATFORM_ID "Cygwin"

#elif defined(__MINGW32__)
# define PLATFORM_ID "MinGW"

#elif defined(__APPLE__)
# define PLATFORM_ID "Darwin"

#elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
# define PLATFORM_ID "Windows"

#elif defined(__FreeBSD__) || defined(__FreeBSD)
# define PLATFORM_ID "FreeBSD"

#elif defined(__NetBSD__) || defined(__NetBSD)
# define PLATFORM_ID "NetBSD"

#elif defined(__OpenBSD__) || defined(__OPENBSD)
# define PLATFORM_ID "OpenBSD"

#elif defined(__sun) || defined(sun)
# define PLATFORM_ID "SunOS"

#elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || d
efined(__aix__)
# define PLATFORM_ID "AIX"

#elif defined(__hpux) || defined(__hpux__)
# define PLATFORM_ID "HP-UX"

#elif defined(__HAIKU__)
# define PLATFORM_ID "Haiku"

#elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
# define PLATFORM_ID "BeOS"

#elif defined(__QNX__) || defined(__QNXNTO__)
# define PLATFORM_ID "QNX"

#elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
# define PLATFORM_ID "Tru64"

#elif defined(__riscos) || defined(__riscos__)
# define PLATFORM_ID "RISCos"

#elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
# define PLATFORM_ID "SINIX"

#elif defined(__UNIX_SV__)
# define PLATFORM_ID "UNIX_SV"

#elif defined(__bsdos__)
# define PLATFORM_ID "BSDOS"
```

```
#elif defined(_MPRAS) || defined(MPRAS)
# define PLATFORM_ID "MP-RAS"

#elif defined(__osf) || defined(__osf__)
# define PLATFORM_ID "OSF1"

#elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
# define PLATFORM_ID "SCO_SV"

#elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
# define PLATFORM_ID "ULTRIX"

#elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
# define PLATFORM_ID "Xenix"

#elif defined(__WATCOMC__)
# if defined(__LINUX__)
#  define PLATFORM_ID "Linux"

# elif defined(__DOS__)
#  define PLATFORM_ID "DOS"

# elif defined(__OS2__)
#  define PLATFORM_ID "OS2"

# elif defined(__WINDOWS__)
#  define PLATFORM_ID "Windows3x"

# elif defined(__VXWORKS__)
#  define PLATFORM_ID "VxWorks"

# else /* unknown platform */
#  define PLATFORM_ID
# endif

#elif defined(__INTEGRITY)
# if defined(INT_178B)
#  define PLATFORM_ID "Integrity178"

# else /* regular Integrity */
#  define PLATFORM_ID "Integrity"
# endif

#else /* unknown platform */
# define PLATFORM_ID

#endif

/* For windows compilers MSVC and Intel we can determine
   the architecture of the compiler being used.  This is because
   the compilers do not have flags that can change the architecture,
   but rather depend on which compiler is being used
*/
#if defined(_WIN32) && defined(_MSC_VER)
# if defined(_M_IA64)
#  define ARCHITECTURE_ID "IA64"

# elif defined(_M_ARM64EC)
#  define ARCHITECTURE_ID "ARM64EC"

# elif defined(_M_X64) || defined(_M_AMD64)
#  define ARCHITECTURE_ID "x64"
```



```
# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# elif defined(_M_ARM64)
#  define ARCHITECTURE_ID "ARM64"

# elif defined(_M_ARM)
#  if _M_ARM == 4
#   define ARCHITECTURE_ID "ARMV4I"
#  elif _M_ARM == 5
#   define ARCHITECTURE_ID "ARMV5I"
#  else
#   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
#  endif

# elif defined(_M_MIPS)
#  define ARCHITECTURE_ID "MIPS"

# elif defined(_M_SH)
#  define ARCHITECTURE_ID "SHx"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__WATCOMC__)
# if defined(_M_I86)
#  define ARCHITECTURE_ID "I86"

# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# if defined(__ICCARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__ICCRX__)
#  define ARCHITECTURE_ID "RX"

# elif defined(__ICCRH850__)
#  define ARCHITECTURE_ID "RH850"

# elif defined(__ICCRL78__)
#  define ARCHITECTURE_ID "RL78"

# elif defined(__ICCRISCV__)
#  define ARCHITECTURE_ID "RISCV"

# elif defined(__ICCAVR__)
#  define ARCHITECTURE_ID "AVR"

# elif defined(__ICC430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__ICCV850__)
#  define ARCHITECTURE_ID "V850"
```

```
# elif defined(__ICC8051__)
#  define ARCHITECTURE_ID "8051"

# elif defined(__ICCSTM8__)
#  define ARCHITECTURE_ID "STM8"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__ghs__)
# if defined(__PPC64__)
#  define ARCHITECTURE_ID "PPC64"

# elif defined(__ppc__)
#  define ARCHITECTURE_ID "PPC"

# elif defined(__ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__x86_64__)
#  define ARCHITECTURE_ID "x64"

# elif defined(__i386__)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__TI_COMPILER_VERSION__)
# if defined(__TI_ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__MSP430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__TMS320C28XX__)
#  define ARCHITECTURE_ID "TMS320C28x"

# elif defined(__TMS320C6X__) || defined(_TMS320C6X)
#  define ARCHITECTURE_ID "TMS320C6x"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#else
#  define ARCHITECTURE_ID
#endif

/* Convert integer to decimal digit literals.  */
#define DEC(n)                   \
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```



```
/* Convert integer to hex digit literals.  */
#define HEX(n)             \
  ('0' + ((n)>>28 & 0xF)), \
  ('0' + ((n)>>24 & 0xF)), \
  ('0' + ((n)>>20 & 0xF)), \
  ('0' + ((n)>>16 & 0xF)), \
  ('0' + ((n)>>12 & 0xF)), \
  ('0' + ((n)>>8  & 0xF)), \
  ('0' + ((n)>>4  & 0xF)), \
  ('0' + ((n)     & 0xF))

/* Construct a string literal encoding the version number. */
#ifdef COMPILER_VERSION
char const* info_version = "INFO" ":" "compiler_version[" COMPILER_VERSION "]";

/* Construct a string literal encoding the version number components. */
#elif defined(COMPILER_VERSION_MAJOR)
char const info_version[] = {
  'I', 'N', 'F', 'O', ':',
  'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','[',
  COMPILER_VERSION_MAJOR,
# ifdef COMPILER_VERSION_MINOR
  '.', COMPILER_VERSION_MINOR,
#  ifdef COMPILER_VERSION_PATCH
   '.', COMPILER_VERSION_PATCH,
#   ifdef COMPILER_VERSION_TWEAK
    '.', COMPILER_VERSION_TWEAK,
#   endif
#  endif
# endif
  ']','\0'};
#endif

/* Construct a string literal encoding the internal version number. */
#ifdef COMPILER_VERSION_INTERNAL
char const info_version_internal[] = {
  'I', 'N', 'F', 'O', ':',
  'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','_',
  'i','n','t','e','r','n','a','l','[',
  COMPILER_VERSION_INTERNAL,']','\0'};
#elif defined(COMPILER_VERSION_INTERNAL_STR)
char const* info_version_internal = "INFO" ":" "compiler_version_internal[" COMP
ILER_VERSION_INTERNAL_STR "]";
#endif

/* Construct a string literal encoding the version number components. */
#ifdef SIMULATE_VERSION_MAJOR
char const info_simulate_version[] = {
  'I', 'N', 'F', 'O', ':',
  's','i','m','u','l','a','t','e','_','v','e','r','s','i','o','n','[',
  SIMULATE_VERSION_MAJOR,
# ifdef SIMULATE_VERSION_MINOR
  '.', SIMULATE_VERSION_MINOR,
#  ifdef SIMULATE_VERSION_PATCH
   '.', SIMULATE_VERSION_PATCH,
#   ifdef SIMULATE_VERSION_TWEAK
    '.', SIMULATE_VERSION_TWEAK,
#   endif
#  endif
# endif
  ']','\0'};
#endif
```

```
/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]";
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]";



#if defined(__INTEL_COMPILER) && defined(_MSVC_LANG) && _MSVC_LANG < 201403L
#  if defined(__INTEL_CXX11_MODE__)
#    if defined(__cpp_aggregate_nsdmi)
#      define CXX_STD 201402L
#    else
#      define CXX_STD 201103L
#    endif
#  else
#    define CXX_STD 199711L
#  endif
#elif defined(_MSC_VER) && defined(_MSVC_LANG)
#  define CXX_STD _MSVC_LANG
#else
#  define CXX_STD __cplusplus
#endif

const char* info_language_standard_default = "INFO" ":" "standard_default["
#if CXX_STD > 202002L
  "23"
#elif CXX_STD > 201703L
  "20"
#elif CXX_STD >= 201703L
  "17"
#elif CXX_STD >= 201402L
  "14"
#elif CXX_STD >= 201103L
  "11"
#else
  "98"
#endif
"]";

const char* info_language_extensions_default = "INFO" ":" "extensions_default["
/* !defined(_MSC_VER) to exclude Clang's MSVC compatibility mode. */
#if (defined(__clang__) || defined(__GNUC__) ||                               \
     defined(__TI_COMPILER_VERSION__)) &&                                     \
  !defined(__STRICT_ANSI__) && !defined(_MSC_VER)
  "ON"
#else
  "OFF"
#endif
"]";

/*--------------------------------------------------------------------------*/

int main(int argc, char* argv[])
{
  int require = 0;
  require += info_compiler[argc];
  require += info_platform[argc];
#ifdef COMPILER_VERSION_MAJOR
  require += info_version[argc];
```



```
#endif
#ifdef COMPILER_VERSION_INTERNAL
  require += info_version_internal[argc];
#endif
#ifdef SIMULATE_ID
  require += info_simulate[argc];
#endif
#ifdef SIMULATE_VERSION_MAJOR
  require += info_simulate_version[argc];
#endif
#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
  require += info_cray[argc];
#endif
  require += info_language_standard_default[argc];
  require += info_language_extensions_default[argc];
  (void)argv;
  return require;
}
---- END OF FILE: /home/henrik/Projekter/AppFramework/build/CMakeFiles/3.22.1/Co
mpilerIdCXX/CMakeCXXCompilerId.cpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/build/subprojects/Logger
/Makefile ----
# CMAKE generated file: DO NOT EDIT!
# Generated by "Unix Makefiles" Generator, CMake Version 3.22

# Default target executed when no arguments are given to make.
default_target: all
.PHONY : default_target

# Allow only one "make -f Makefile2" at a time, but pass parallelism.
.NOTPARALLEL:

#=============================================================================
# Special targets provided by cmake.

# Disable implicit rules so canonical targets will work.
.SUFFIXES:

# Disable VCS-based implicit rules.
% : %,v

# Disable VCS-based implicit rules.
% : RCS/%

# Disable VCS-based implicit rules.
% : RCS/%,v

# Disable VCS-based implicit rules.
% : SCCS/s.%

# Disable VCS-based implicit rules.
% : s.%

.SUFFIXES: .hpux_make_needs_suffix_list

# Command-line flag to silence nested $(MAKE).
$(VERBOSE)MAKESILENT = -s

#Suppress display of executed commands.
$(VERBOSE).SILENT:
```

```
# A target that is always out of date.
cmake_force:
.PHONY : cmake_force

#=============================================================================
# Set environment variables for the build.

# The shell in which to execute make rules.
SHELL = /bin/sh

# The CMake executable.
CMAKE_COMMAND = /usr/bin/cmake

# The command to remove a file.
RM = /usr/bin/cmake -E rm -f

# Escaping for special characters.
EQUALS = =

# The top-level source directory on which CMake was run.
CMAKE_SOURCE_DIR = /home/henrik/Projekter/AppFramework

# The top-level build directory on which CMake was run.
CMAKE_BINARY_DIR = /home/henrik/Projekter/AppFramework/build

#=============================================================================
# Targets provided globally by CMake.

# Special rule for the target edit_cache
edit_cache:
	@$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "No inter
active CMake dialog available..."
	/usr/bin/cmake -E echo No\ interactive\ CMake\ dialog\ available.
.PHONY : edit_cache

# Special rule for the target edit_cache
edit_cache/fast: edit_cache
.PHONY : edit_cache/fast

# Special rule for the target rebuild_cache
rebuild_cache:
	@$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "Running
CMake to regenerate build system..."
	/usr/bin/cmake --regenerate-during-build -S$(CMAKE_SOURCE_DIR) -B$(CMAKE
_BINARY_DIR)
.PHONY : rebuild_cache

# Special rule for the target rebuild_cache
rebuild_cache/fast: rebuild_cache
.PHONY : rebuild_cache/fast

# The main all target
all: cmake_check_build_system
	cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -E cmak
e_progress_start /home/henrik/Projekter/AppFramework/build/CMakeFiles /home/henr
ik/Projekter/AppFramework/build/subprojects/Logger//CMakeFiles/progress.marks
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/Logger/all
	$(CMAKE_COMMAND) -E cmake_progress_start /home/henrik/Projekter/AppFrame
work/build/CMakeFiles 0
.PHONY : all
```



```
# The main clean target
clean:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/Logger/clean
.PHONY : clean

# The main clean target
clean/fast: clean
.PHONY : clean/fast

# Prepare targets for installation.
preinstall: all
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/Logger/preinstall
.PHONY : preinstall

# Prepare targets for installation.
preinstall/fast:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/Logger/preinstall
.PHONY : preinstall/fast

# clear depends
depend:
	cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -S$(CMA
KE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-build-system CMakeFiles/Makefile.cm
ake 1
.PHONY : depend

# Convenience name for target.
subprojects/Logger/CMakeFiles/Logger.dir/rule:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/Logger/CMakeFiles/Logger.dir/rule
.PHONY : subprojects/Logger/CMakeFiles/Logger.dir/rule

# Convenience name for target.
Logger: subprojects/Logger/CMakeFiles/Logger.dir/rule
.PHONY : Logger

# fast build rule for target.
Logger/fast:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/Logger/CMakeFiles/Logger.dir/build.make subprojects/Logger/CMakeFil
es/Logger.dir/build
.PHONY : Logger/fast

Logger.o: Logger.cpp.o
.PHONY : Logger.o

# target to build an object file
Logger.cpp.o:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/Logger/CMakeFiles/Logger.dir/build.make subprojects/Logger/CMakeFil
es/Logger.dir/Logger.cpp.o
.PHONY : Logger.cpp.o

Logger.i: Logger.cpp.i
.PHONY : Logger.i

# target to preprocess a source file
Logger.cpp.i:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
```

```
 subprojects/Logger/CMakeFiles/Logger.dir/build.make subprojects/Logger/CMakeFil
es/Logger.dir/Logger.cpp.i
.PHONY : Logger.cpp.i

Logger.s: Logger.cpp.s
.PHONY : Logger.s

# target to generate assembly for a file
Logger.cpp.s:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/Logger/CMakeFiles/Logger.dir/build.make subprojects/Logger/CMakeFil
es/Logger.dir/Logger.cpp.s
.PHONY : Logger.cpp.s

# Help Target
help:
        @echo "The following are some of the valid targets for this Makefile:"
        @echo "... all (the default if no target is provided)"
        @echo "... clean"
        @echo "... depend"
        @echo "... edit_cache"
        @echo "... rebuild_cache"
        @echo "... Logger"
        @echo "... Logger.o"
        @echo "... Logger.i"
        @echo "... Logger.s"
.PHONY : help



#=============================================================================
# Special targets to cleanup operation of make.

# Special rule to run CMake to check the build system integrity.
# No rule that depends on this can have commands that come from listfiles
# because they might be regenerated.
cmake_check_build_system:
        cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -S$(CMA
KE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-build-system CMakeFiles/Makefile.cm
ake 0
.PHONY : cmake_check_build_system

---- END OF FILE: /home/henrik/Projekter/AppFramework/build/subprojects/Logger/M
akefile ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/build/subprojects/UiMana
ger/Makefile ----
# CMAKE generated file: DO NOT EDIT!
# Generated by "Unix Makefiles" Generator, CMake Version 3.22

# Default target executed when no arguments are given to make.
default_target: all
.PHONY : default_target

# Allow only one "make -f Makefile2" at a time, but pass parallelism.
.NOTPARALLEL:

#=============================================================================
# Special targets provided by cmake.

# Disable implicit rules so canonical targets will work.
.SUFFIXES:
```



```
# Disable VCS-based implicit rules.
% : %,v

# Disable VCS-based implicit rules.
% : RCS/%

# Disable VCS-based implicit rules.
% : RCS/%,v

# Disable VCS-based implicit rules.
% : SCCS/s.%

# Disable VCS-based implicit rules.
% : s.%

.SUFFIXES: .hpux_make_needs_suffix_list

# Command-line flag to silence nested $(MAKE).
$(VERBOSE)MAKESILENT = -s

#Suppress display of executed commands.
$(VERBOSE).SILENT:

# A target that is always out of date.
cmake_force:
.PHONY : cmake_force

#=============================================================================
# Set environment variables for the build.

# The shell in which to execute make rules.
SHELL = /bin/sh

# The CMake executable.
CMAKE_COMMAND = /usr/bin/cmake

# The command to remove a file.
RM = /usr/bin/cmake -E rm -f

# Escaping for special characters.
EQUALS = =

# The top-level source directory on which CMake was run.
CMAKE_SOURCE_DIR = /home/henrik/Projekter/AppFramework

# The top-level build directory on which CMake was run.
CMAKE_BINARY_DIR = /home/henrik/Projekter/AppFramework/build

#=============================================================================
# Targets provided globally by CMake.

# Special rule for the target edit_cache
edit_cache:
        @$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "No inter
active CMake dialog available..."
        /usr/bin/cmake -E echo No\ interactive\ CMake\ dialog\ available.
.PHONY : edit_cache

# Special rule for the target edit_cache
edit_cache/fast: edit_cache
.PHONY : edit_cache/fast
```

```
# Special rule for the target rebuild_cache
rebuild_cache:
	@$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "Running
CMake to regenerate build system..."
	/usr/bin/cmake --regenerate-during-build -S$(CMAKE_SOURCE_DIR) -B$(CMAKE
_BINARY_DIR)
.PHONY : rebuild_cache

# Special rule for the target rebuild_cache
rebuild_cache/fast: rebuild_cache
.PHONY : rebuild_cache/fast

# The main all target
all: cmake_check_build_system
	cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -E cmak
e_progress_start /home/henrik/Projekter/AppFramework/build/CMakeFiles /home/henr
ik/Projekter/AppFramework/build/subprojects/UiManager//CMakeFiles/progress.marks
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/UiManager/all
	$(CMAKE_COMMAND) -E cmake_progress_start /home/henrik/Projekter/AppFrame
work/build/CMakeFiles 0
.PHONY : all

# The main clean target
clean:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/UiManager/clean
.PHONY : clean

# The main clean target
clean/fast: clean
.PHONY : clean/fast

# Prepare targets for installation.
preinstall: all
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/UiManager/preinstall
.PHONY : preinstall

# Prepare targets for installation.
preinstall/fast:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/UiManager/preinstall
.PHONY : preinstall/fast

# clear depends
depend:
	cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -S$(CMA
KE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-build-system CMakeFiles/Makefile.cm
ake 1
.PHONY : depend

# Convenience name for target.
subprojects/UiManager/CMakeFiles/UiManager.dir/rule:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/UiManager/CMakeFiles/UiManager.dir/rule
.PHONY : subprojects/UiManager/CMakeFiles/UiManager.dir/rule

# Convenience name for target.
UiManager: subprojects/UiManager/CMakeFiles/UiManager.dir/rule
.PHONY : UiManager
```



```
# fast build rule for target.
UiManager/fast:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/UiManager/CMakeFiles/UiManager.dir/build.make subprojects/UiManager
/CMakeFiles/UiManager.dir/build
.PHONY : UiManager/fast

UiManager.o: UiManager.cpp.o
.PHONY : UiManager.o

# target to build an object file
UiManager.cpp.o:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/UiManager/CMakeFiles/UiManager.dir/build.make subprojects/UiManager
/CMakeFiles/UiManager.dir/UiManager.cpp.o
.PHONY : UiManager.cpp.o

UiManager.i: UiManager.cpp.i
.PHONY : UiManager.i

# target to preprocess a source file
UiManager.cpp.i:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/UiManager/CMakeFiles/UiManager.dir/build.make subprojects/UiManager
/CMakeFiles/UiManager.dir/UiManager.cpp.i
.PHONY : UiManager.cpp.i

UiManager.s: UiManager.cpp.s
.PHONY : UiManager.s

# target to generate assembly for a file
UiManager.cpp.s:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/UiManager/CMakeFiles/UiManager.dir/build.make subprojects/UiManager
/CMakeFiles/UiManager.dir/UiManager.cpp.s
.PHONY : UiManager.cpp.s

# Help Target
help:
	@echo "The following are some of the valid targets for this Makefile:"
	@echo "... all (the default if no target is provided)"
	@echo "... clean"
	@echo "... depend"
	@echo "... edit_cache"
	@echo "... rebuild_cache"
	@echo "... UiManager"
	@echo "... UiManager.o"
	@echo "... UiManager.i"
	@echo "... UiManager.s"
.PHONY : help



#=============================================================================
# Special targets to cleanup operation of make.

# Special rule to run CMake to check the build system integrity.
# No rule that depends on this can have commands that come from listfiles
# because they might be regenerated.
cmake_check_build_system:
	cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -S$(CMA
```

```
KE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-build-system CMakeFiles/Makefile.cm
ake 0
.PHONY : cmake_check_build_system

---- END OF FILE: /home/henrik/Projekter/AppFramework/build/subprojects/UiManage
r/Makefile ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/build/subprojects/EnvVar
/Makefile ----
# CMAKE generated file: DO NOT EDIT!
# Generated by "Unix Makefiles" Generator, CMake Version 3.22

# Default target executed when no arguments are given to make.
default_target: all
.PHONY : default_target

# Allow only one "make -f Makefile2" at a time, but pass parallelism.
.NOTPARALLEL:

#=============================================================================
# Special targets provided by cmake.

# Disable implicit rules so canonical targets will work.
.SUFFIXES:

# Disable VCS-based implicit rules.
% : %,v

# Disable VCS-based implicit rules.
% : RCS/%

# Disable VCS-based implicit rules.
% : RCS/%,v

# Disable VCS-based implicit rules.
% : SCCS/s.%

# Disable VCS-based implicit rules.
% : s.%

.SUFFIXES: .hpux_make_needs_suffix_list

# Command-line flag to silence nested $(MAKE).
$(VERBOSE)MAKESILENT = -s

#Suppress display of executed commands.
$(VERBOSE).SILENT:

# A target that is always out of date.
cmake_force:
.PHONY : cmake_force

#=============================================================================
# Set environment variables for the build.

# The shell in which to execute make rules.
SHELL = /bin/sh

# The CMake executable.
CMAKE_COMMAND = /usr/bin/cmake

# The command to remove a file.
```



```
RM = /usr/bin/cmake -E rm -f

# Escaping for special characters.
EQUALS = =

# The top-level source directory on which CMake was run.
CMAKE_SOURCE_DIR = /home/henrik/Projekter/AppFramework

# The top-level build directory on which CMake was run.
CMAKE_BINARY_DIR = /home/henrik/Projekter/AppFramework/build

#=============================================================================
# Targets provided globally by CMake.

# Special rule for the target edit_cache
edit_cache:
        @$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "No inter
active CMake dialog available..."
        /usr/bin/cmake -E echo No\ interactive\ CMake\ dialog\ available.
.PHONY : edit_cache

# Special rule for the target edit_cache
edit_cache/fast: edit_cache
.PHONY : edit_cache/fast

# Special rule for the target rebuild_cache
rebuild_cache:
        @$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "Running
CMake to regenerate build system..."
        /usr/bin/cmake --regenerate-during-build -S$(CMAKE_SOURCE_DIR) -B$(CMAKE
_BINARY_DIR)
.PHONY : rebuild_cache

# Special rule for the target rebuild_cache
rebuild_cache/fast: rebuild_cache
.PHONY : rebuild_cache/fast

# The main all target
all: cmake_check_build_system
        cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -E cmak
e_progress_start /home/henrik/Projekter/AppFramework/build/CMakeFiles /home/henr
ik/Projekter/AppFramework/build/subprojects/EnvVar//CMakeFiles/progress.marks
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/EnvVar/all
        $(CMAKE_COMMAND) -E cmake_progress_start /home/henrik/Projekter/AppFrame
work/build/CMakeFiles 0
.PHONY : all

# The main clean target
clean:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/EnvVar/clean
.PHONY : clean

# The main clean target
clean/fast: clean
.PHONY : clean/fast

# Prepare targets for installation.
preinstall: all
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/EnvVar/preinstall
```

```
.PHONY : preinstall

# Prepare targets for installation.
preinstall/fast:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/EnvVar/preinstall
.PHONY : preinstall/fast

# clear depends
depend:
	cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -S$(CMA
KE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-build-system CMakeFiles/Makefile.cm
ake 1
.PHONY : depend

# Convenience name for target.
subprojects/EnvVar/CMakeFiles/EnvVar.dir/rule:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/EnvVar/CMakeFiles/EnvVar.dir/rule
.PHONY : subprojects/EnvVar/CMakeFiles/EnvVar.dir/rule

# Convenience name for target.
EnvVar: subprojects/EnvVar/CMakeFiles/EnvVar.dir/rule
.PHONY : EnvVar

# fast build rule for target.
EnvVar/fast:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/EnvVar/CMakeFiles/EnvVar.dir/build.make subprojects/EnvVar/CMakeFil
es/EnvVar.dir/build
.PHONY : EnvVar/fast

EnvVar.o: EnvVar.cpp.o
.PHONY : EnvVar.o

# target to build an object file
EnvVar.cpp.o:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/EnvVar/CMakeFiles/EnvVar.dir/build.make subprojects/EnvVar/CMakeFil
es/EnvVar.dir/EnvVar.cpp.o
.PHONY : EnvVar.cpp.o

EnvVar.i: EnvVar.cpp.i
.PHONY : EnvVar.i

# target to preprocess a source file
EnvVar.cpp.i:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/EnvVar/CMakeFiles/EnvVar.dir/build.make subprojects/EnvVar/CMakeFil
es/EnvVar.dir/EnvVar.cpp.i
.PHONY : EnvVar.cpp.i

EnvVar.s: EnvVar.cpp.s
.PHONY : EnvVar.s

# target to generate assembly for a file
EnvVar.cpp.s:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/EnvVar/CMakeFiles/EnvVar.dir/build.make subprojects/EnvVar/CMakeFil
es/EnvVar.dir/EnvVar.cpp.s
.PHONY : EnvVar.cpp.s
```



```
# Help Target
help:
	@echo "The following are some of the valid targets for this Makefile:"
	@echo "... all (the default if no target is provided)"
	@echo "... clean"
	@echo "... depend"
	@echo "... edit_cache"
	@echo "... rebuild_cache"
	@echo "... EnvVar"
	@echo "... EnvVar.o"
	@echo "... EnvVar.i"
	@echo "... EnvVar.s"
.PHONY : help



#=============================================================================
# Special targets to cleanup operation of make.

# Special rule to run CMake to check the build system integrity.
# No rule that depends on this can have commands that come from listfiles
# because they might be regenerated.
cmake_check_build_system:
	cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -S$(CMA
KE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-build-system CMakeFiles/Makefile.cm
ake 0
.PHONY : cmake_check_build_system

---- END OF FILE: /home/henrik/Projekter/AppFramework/build/subprojects/EnvVar/M
akefile ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/build/subprojects/TimeUt
ils/Makefile ----
# CMAKE generated file: DO NOT EDIT!
# Generated by "Unix Makefiles" Generator, CMake Version 3.22

# Default target executed when no arguments are given to make.
default_target: all
.PHONY : default_target

# Allow only one "make -f Makefile2" at a time, but pass parallelism.
.NOTPARALLEL:

#=============================================================================
# Special targets provided by cmake.

# Disable implicit rules so canonical targets will work.
.SUFFIXES:

# Disable VCS-based implicit rules.
% : %,v

# Disable VCS-based implicit rules.
% : RCS/%

# Disable VCS-based implicit rules.
% : RCS/%,v

# Disable VCS-based implicit rules.
% : SCCS/s.%

# Disable VCS-based implicit rules.
```

```
% : s.%

.SUFFIXES: .hpux_make_needs_suffix_list

# Command-line flag to silence nested $(MAKE).
$(VERBOSE)MAKESILENT = -s

#Suppress display of executed commands.
$(VERBOSE).SILENT:

# A target that is always out of date.
cmake_force:
.PHONY : cmake_force

#=============================================================================
# Set environment variables for the build.

# The shell in which to execute make rules.
SHELL = /bin/sh

# The CMake executable.
CMAKE_COMMAND = /usr/bin/cmake

# The command to remove a file.
RM = /usr/bin/cmake -E rm -f

# Escaping for special characters.
EQUALS = =

# The top-level source directory on which CMake was run.
CMAKE_SOURCE_DIR = /home/henrik/Projekter/AppFramework

# The top-level build directory on which CMake was run.
CMAKE_BINARY_DIR = /home/henrik/Projekter/AppFramework/build

#=============================================================================
# Targets provided globally by CMake.

# Special rule for the target edit_cache
edit_cache:
        @$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "No inter
active CMake dialog available..."
        /usr/bin/cmake -E echo No\ interactive\ CMake\ dialog\ available.
.PHONY : edit_cache

# Special rule for the target edit_cache
edit_cache/fast: edit_cache
.PHONY : edit_cache/fast

# Special rule for the target rebuild_cache
rebuild_cache:
        @$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "Running
CMake to regenerate build system..."
        /usr/bin/cmake --regenerate-during-build -S$(CMAKE_SOURCE_DIR) -B$(CMAKE
_BINARY_DIR)
.PHONY : rebuild_cache

# Special rule for the target rebuild_cache
rebuild_cache/fast: rebuild_cache
.PHONY : rebuild_cache/fast

# The main all target
```



```
all: cmake_check_build_system
        cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -E cmak
e_progress_start /home/henrik/Projekter/AppFramework/build/CMakeFiles /home/henr
ik/Projekter/AppFramework/build/subprojects/TimeUtils//CMakeFiles/progress.marks
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/TimeUtils/all
        $(CMAKE_COMMAND) -E cmake_progress_start /home/henrik/Projekter/AppFrame
work/build/CMakeFiles 0
.PHONY : all

# The main clean target
clean:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/TimeUtils/clean
.PHONY : clean

# The main clean target
clean/fast: clean
.PHONY : clean/fast

# Prepare targets for installation.
preinstall: all
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/TimeUtils/preinstall
.PHONY : preinstall

# Prepare targets for installation.
preinstall/fast:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/TimeUtils/preinstall
.PHONY : preinstall/fast

# clear depends
depend:
        cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -S$(CMA
KE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-build-system CMakeFiles/Makefile.cm
ake 1
.PHONY : depend

# Convenience name for target.
subprojects/TimeUtils/CMakeFiles/TimeUtils.dir/rule:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/TimeUtils/CMakeFiles/TimeUtils.dir/rule
.PHONY : subprojects/TimeUtils/CMakeFiles/TimeUtils.dir/rule

# Convenience name for target.
TimeUtils: subprojects/TimeUtils/CMakeFiles/TimeUtils.dir/rule
.PHONY : TimeUtils

# fast build rule for target.
TimeUtils/fast:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/TimeUtils/CMakeFiles/TimeUtils.dir/build.make subprojects/TimeUtils
/CMakeFiles/TimeUtils.dir/build
.PHONY : TimeUtils/fast

TimeUtils.o: TimeUtils.cpp.o
.PHONY : TimeUtils.o

# target to build an object file
TimeUtils.cpp.o:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
```

```
 subprojects/TimeUtils/CMakeFiles/TimeUtils.dir/build.make subprojects/TimeUtils
/CMakeFiles/TimeUtils.dir/TimeUtils.cpp.o
.PHONY : TimeUtils.cpp.o

TimeUtils.i: TimeUtils.cpp.i
.PHONY : TimeUtils.i

# target to preprocess a source file
TimeUtils.cpp.i:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/TimeUtils/CMakeFiles/TimeUtils.dir/build.make subprojects/TimeUtils
/CMakeFiles/TimeUtils.dir/TimeUtils.cpp.i
.PHONY : TimeUtils.cpp.i

TimeUtils.s: TimeUtils.cpp.s
.PHONY : TimeUtils.s

# target to generate assembly for a file
TimeUtils.cpp.s:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/TimeUtils/CMakeFiles/TimeUtils.dir/build.make subprojects/TimeUtils
/CMakeFiles/TimeUtils.dir/TimeUtils.cpp.s
.PHONY : TimeUtils.cpp.s

# Help Target
help:
        @echo "The following are some of the valid targets for this Makefile:"
        @echo "... all (the default if no target is provided)"
        @echo "... clean"
        @echo "... depend"
        @echo "... edit_cache"
        @echo "... rebuild_cache"
        @echo "... TimeUtils"
        @echo "... TimeUtils.o"
        @echo "... TimeUtils.i"
        @echo "... TimeUtils.s"
.PHONY : help



#=============================================================================
# Special targets to cleanup operation of make.

# Special rule to run CMake to check the build system integrity.
# No rule that depends on this can have commands that come from listfiles
# because they might be regenerated.
cmake_check_build_system:
        cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -S$(CMA
KE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-build-system CMakeFiles/Makefile.cm
ake 0
.PHONY : cmake_check_build_system

---- END OF FILE: /home/henrik/Projekter/AppFramework/build/subprojects/TimeUtil
s/Makefile ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/build/subprojects/String
Utils/Makefile ----
# CMAKE generated file: DO NOT EDIT!
# Generated by "Unix Makefiles" Generator, CMake Version 3.22

# Default target executed when no arguments are given to make.
default_target: all
```



```
.PHONY : default_target

# Allow only one "make -f Makefile2" at a time, but pass parallelism.
.NOTPARALLEL:

#=============================================================================
# Special targets provided by cmake.

# Disable implicit rules so canonical targets will work.
.SUFFIXES:

# Disable VCS-based implicit rules.
% : %,v

# Disable VCS-based implicit rules.
% : RCS/%

# Disable VCS-based implicit rules.
% : RCS/%,v

# Disable VCS-based implicit rules.
% : SCCS/s.%

# Disable VCS-based implicit rules.
% : s.%

.SUFFIXES: .hpux_make_needs_suffix_list

# Command-line flag to silence nested $(MAKE).
$(VERBOSE)MAKESILENT = -s

#Suppress display of executed commands.
$(VERBOSE).SILENT:

# A target that is always out of date.
cmake_force:
.PHONY : cmake_force

#=============================================================================
# Set environment variables for the build.

# The shell in which to execute make rules.
SHELL = /bin/sh

# The CMake executable.
CMAKE_COMMAND = /usr/bin/cmake

# The command to remove a file.
RM = /usr/bin/cmake -E rm -f

# Escaping for special characters.
EQUALS = =

# The top-level source directory on which CMake was run.
CMAKE_SOURCE_DIR = /home/henrik/Projekter/AppFramework

# The top-level build directory on which CMake was run.
CMAKE_BINARY_DIR = /home/henrik/Projekter/AppFramework/build

#=============================================================================
# Targets provided globally by CMake.
```

```
# Special rule for the target edit_cache
edit_cache:
	@$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "No inter
active CMake dialog available..."
	/usr/bin/cmake -E echo No\ interactive\ CMake\ dialog\ available.
.PHONY : edit_cache

# Special rule for the target edit_cache
edit_cache/fast: edit_cache
.PHONY : edit_cache/fast

# Special rule for the target rebuild_cache
rebuild_cache:
	@$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "Running
CMake to regenerate build system..."
	/usr/bin/cmake --regenerate-during-build -S$(CMAKE_SOURCE_DIR) -B$(CMAKE
_BINARY_DIR)
.PHONY : rebuild_cache

# Special rule for the target rebuild_cache
rebuild_cache/fast: rebuild_cache
.PHONY : rebuild_cache/fast

# The main all target
all: cmake_check_build_system
	cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -E cmak
e_progress_start /home/henrik/Projekter/AppFramework/build/CMakeFiles /home/henr
ik/Projekter/AppFramework/build/subprojects/StringUtils//CMakeFiles/progress.mar
ks
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/StringUtils/all
	$(CMAKE_COMMAND) -E cmake_progress_start /home/henrik/Projekter/AppFrame
work/build/CMakeFiles 0
.PHONY : all

# The main clean target
clean:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/StringUtils/clean
.PHONY : clean

# The main clean target
clean/fast: clean
.PHONY : clean/fast

# Prepare targets for installation.
preinstall: all
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/StringUtils/preinstall
.PHONY : preinstall

# Prepare targets for installation.
preinstall/fast:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/StringUtils/preinstall
.PHONY : preinstall/fast

# clear depends
depend:
	cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -S$(CMA
KE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-build-system CMakeFiles/Makefile.cm
ake 1
```



```
.PHONY : depend

# Convenience name for target.
subprojects/StringUtils/CMakeFiles/StringUtils.dir/rule:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/StringUtils/CMakeFiles/StringUtils.dir/rule
.PHONY : subprojects/StringUtils/CMakeFiles/StringUtils.dir/rule

# Convenience name for target.
StringUtils: subprojects/StringUtils/CMakeFiles/StringUtils.dir/rule
.PHONY : StringUtils

# fast build rule for target.
StringUtils/fast:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/StringUtils/CMakeFiles/StringUtils.dir/build.make subprojects/Strin
gUtils/CMakeFiles/StringUtils.dir/build
.PHONY : StringUtils/fast

StringUtils.o: StringUtils.cpp.o
.PHONY : StringUtils.o

# target to build an object file
StringUtils.cpp.o:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/StringUtils/CMakeFiles/StringUtils.dir/build.make subprojects/Strin
gUtils/CMakeFiles/StringUtils.dir/StringUtils.cpp.o
.PHONY : StringUtils.cpp.o

StringUtils.i: StringUtils.cpp.i
.PHONY : StringUtils.i

# target to preprocess a source file
StringUtils.cpp.i:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/StringUtils/CMakeFiles/StringUtils.dir/build.make subprojects/Strin
gUtils/CMakeFiles/StringUtils.dir/StringUtils.cpp.i
.PHONY : StringUtils.cpp.i

StringUtils.s: StringUtils.cpp.s
.PHONY : StringUtils.s

# target to generate assembly for a file
StringUtils.cpp.s:
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/StringUtils/CMakeFiles/StringUtils.dir/build.make subprojects/Strin
gUtils/CMakeFiles/StringUtils.dir/StringUtils.cpp.s
.PHONY : StringUtils.cpp.s

# Help Target
help:
	@echo "The following are some of the valid targets for this Makefile:"
	@echo "... all (the default if no target is provided)"
	@echo "... clean"
	@echo "... depend"
	@echo "... edit_cache"
	@echo "... rebuild_cache"
	@echo "... StringUtils"
	@echo "... StringUtils.o"
	@echo "... StringUtils.i"
	@echo "... StringUtils.s"
.PHONY : help
```

```
#=============================================================================
# Special targets to cleanup operation of make.

# Special rule to run CMake to check the build system integrity.
# No rule that depends on this can have commands that come from listfiles
# because they might be regenerated.
cmake_check_build_system:
	cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -S$(CMA
KE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-build-system CMakeFiles/Makefile.cm
ake 0
.PHONY : cmake_check_build_system

---- END OF FILE: /home/henrik/Projekter/AppFramework/build/subprojects/StringUt
ils/Makefile ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/build/subprojects/Comman
dLineProcessor/Makefile ----
# CMAKE generated file: DO NOT EDIT!
# Generated by "Unix Makefiles" Generator, CMake Version 3.22

# Default target executed when no arguments are given to make.
default_target: all
.PHONY : default_target

# Allow only one "make -f Makefile2" at a time, but pass parallelism.
.NOTPARALLEL:

#=============================================================================
# Special targets provided by cmake.

# Disable implicit rules so canonical targets will work.
.SUFFIXES:

# Disable VCS-based implicit rules.
% : %,v

# Disable VCS-based implicit rules.
% : RCS/%

# Disable VCS-based implicit rules.
% : RCS/%,v

# Disable VCS-based implicit rules.
% : SCCS/s.%

# Disable VCS-based implicit rules.
% : s.%

.SUFFIXES: .hpux_make_needs_suffix_list

# Command-line flag to silence nested $(MAKE).
$(VERBOSE)MAKESILENT = -s

#Suppress display of executed commands.
$(VERBOSE).SILENT:

# A target that is always out of date.
cmake_force:
.PHONY : cmake_force
```



```
#=============================================================================
# Set environment variables for the build.

# The shell in which to execute make rules.
SHELL = /bin/sh

# The CMake executable.
CMAKE_COMMAND = /usr/bin/cmake

# The command to remove a file.
RM = /usr/bin/cmake -E rm -f

# Escaping for special characters.
EQUALS = =

# The top-level source directory on which CMake was run.
CMAKE_SOURCE_DIR = /home/henrik/Projekter/AppFramework

# The top-level build directory on which CMake was run.
CMAKE_BINARY_DIR = /home/henrik/Projekter/AppFramework/build

#=============================================================================
# Targets provided globally by CMake.

# Special rule for the target edit_cache
edit_cache:
	@$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "No inter
active CMake dialog available..."
	/usr/bin/cmake -E echo No\ interactive\ CMake\ dialog\ available.
.PHONY : edit_cache

# Special rule for the target edit_cache
edit_cache/fast: edit_cache
.PHONY : edit_cache/fast

# Special rule for the target rebuild_cache
rebuild_cache:
	@$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "Running
CMake to regenerate build system..."
	/usr/bin/cmake --regenerate-during-build -S$(CMAKE_SOURCE_DIR) -B$(CMAKE
_BINARY_DIR)
.PHONY : rebuild_cache

# Special rule for the target rebuild_cache
rebuild_cache/fast: rebuild_cache
.PHONY : rebuild_cache/fast

# The main all target
all: cmake_check_build_system
	cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -E cmak
e_progress_start /home/henrik/Projekter/AppFramework/build/CMakeFiles /home/henr
ik/Projekter/AppFramework/build/subprojects/CommandLineProcessor//CMakeFiles/pro
gress.marks
	cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/CommandLineProcessor/all
	$(CMAKE_COMMAND) -E cmake_progress_start /home/henrik/Projekter/AppFrame
work/build/CMakeFiles 0
.PHONY : all

# The main clean target
clean:
```

```
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/CommandLineProcessor/clean
.PHONY : clean

# The main clean target
clean/fast: clean
.PHONY : clean/fast

# Prepare targets for installation.
preinstall: all
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/CommandLineProcessor/preinstall
.PHONY : preinstall

# Prepare targets for installation.
preinstall/fast:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/CommandLineProcessor/preinstall
.PHONY : preinstall/fast

# clear depends
depend:
        cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -S$(CMA
KE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-build-system CMakeFiles/Makefile.cm
ake 1
.PHONY : depend

# Convenience name for target.
subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/rule:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 CMakeFiles/Makefile2 subprojects/CommandLineProcessor/CMakeFiles/CommandLinePro
cessor.dir/rule
.PHONY : subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/ru
le

# Convenience name for target.
CommandLineProcessor: subprojects/CommandLineProcessor/CMakeFiles/CommandLinePro
cessor.dir/rule
.PHONY : CommandLineProcessor

# fast build rule for target.
CommandLineProcessor/fast:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build
.PHONY : CommandLineProcessor/fast

Argument.o: Argument.cpp.o
.PHONY : Argument.o

# target to build an object file
Argument.cpp.o:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/Argument.c
pp.o
.PHONY : Argument.cpp.o

Argument.i: Argument.cpp.i
.PHONY : Argument.i

# target to preprocess a source file
```



```
Argument.cpp.i:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/Argument.c
pp.i
.PHONY : Argument.cpp.i

Argument.s: Argument.cpp.s
.PHONY : Argument.s

# target to generate assembly for a file
Argument.cpp.s:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/Argument.c
pp.s
.PHONY : Argument.cpp.s

ArgumentConfig.o: ArgumentConfig.cpp.o
.PHONY : ArgumentConfig.o

# target to build an object file
ArgumentConfig.cpp.o:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/ArgumentCo
nfig.cpp.o
.PHONY : ArgumentConfig.cpp.o

ArgumentConfig.i: ArgumentConfig.cpp.i
.PHONY : ArgumentConfig.i

# target to preprocess a source file
ArgumentConfig.cpp.i:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/ArgumentCo
nfig.cpp.i
.PHONY : ArgumentConfig.cpp.i

ArgumentConfig.s: ArgumentConfig.cpp.s
.PHONY : ArgumentConfig.s

# target to generate assembly for a file
ArgumentConfig.cpp.s:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/ArgumentCo
nfig.cpp.s
.PHONY : ArgumentConfig.cpp.s

Arguments.o: Arguments.cpp.o
.PHONY : Arguments.o

# target to build an object file
Arguments.cpp.o:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/Arguments.
cpp.o
.PHONY : Arguments.cpp.o
```

```
Arguments.i: Arguments.cpp.i
.PHONY : Arguments.i

# target to preprocess a source file
Arguments.cpp.i:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/Arguments.
cpp.i
.PHONY : Arguments.cpp.i

Arguments.s: Arguments.cpp.s
.PHONY : Arguments.s

# target to generate assembly for a file
Arguments.cpp.s:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/Arguments.
cpp.s
.PHONY : Arguments.cpp.s

CommandLineProcessor.o: CommandLineProcessor.cpp.o
.PHONY : CommandLineProcessor.o

# target to build an object file
CommandLineProcessor.cpp.o:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/CommandLin
eProcessor.cpp.o
.PHONY : CommandLineProcessor.cpp.o

CommandLineProcessor.i: CommandLineProcessor.cpp.i
.PHONY : CommandLineProcessor.i

# target to preprocess a source file
CommandLineProcessor.cpp.i:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/CommandLin
eProcessor.cpp.i
.PHONY : CommandLineProcessor.cpp.i

CommandLineProcessor.s: CommandLineProcessor.cpp.s
.PHONY : CommandLineProcessor.s

# target to generate assembly for a file
CommandLineProcessor.cpp.s:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/CommandLin
eProcessor.cpp.s
.PHONY : CommandLineProcessor.cpp.s

ConfigManager.o: ConfigManager.cpp.o
.PHONY : ConfigManager.o

# target to build an object file
ConfigManager.cpp.o:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
```



```
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/ConfigMana
ger.cpp.o
.PHONY : ConfigManager.cpp.o

ConfigManager.i: ConfigManager.cpp.i
.PHONY : ConfigManager.i

# target to preprocess a source file
ConfigManager.cpp.i:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/ConfigMana
ger.cpp.i
.PHONY : ConfigManager.cpp.i

ConfigManager.s: ConfigManager.cpp.s
.PHONY : ConfigManager.s

# target to generate assembly for a file
ConfigManager.cpp.s:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/ConfigMana
ger.cpp.s
.PHONY : ConfigManager.cpp.s

DefaultSettings.o: DefaultSettings.cpp.o
.PHONY : DefaultSettings.o

# target to build an object file
DefaultSettings.cpp.o:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/DefaultSet
tings.cpp.o
.PHONY : DefaultSettings.cpp.o

DefaultSettings.i: DefaultSettings.cpp.i
.PHONY : DefaultSettings.i

# target to preprocess a source file
DefaultSettings.cpp.i:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/DefaultSet
tings.cpp.i
.PHONY : DefaultSettings.cpp.i

DefaultSettings.s: DefaultSettings.cpp.s
.PHONY : DefaultSettings.s

# target to generate assembly for a file
DefaultSettings.cpp.s:
        cd /home/henrik/Projekter/AppFramework/build && $(MAKE) $(MAKESILENT) -f
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/build.make
 subprojects/CommandLineProcessor/CMakeFiles/CommandLineProcessor.dir/DefaultSet
tings.cpp.s
.PHONY : DefaultSettings.cpp.s

# Help Target
help:
        @echo "The following are some of the valid targets for this Makefile:"
```

```
	@echo "... all (the default if no target is provided)"
	@echo "... clean"
	@echo "... depend"
	@echo "... edit_cache"
	@echo "... rebuild_cache"
	@echo "... CommandLineProcessor"
	@echo "... Argument.o"
	@echo "... Argument.i"
	@echo "... Argument.s"
	@echo "... ArgumentConfig.o"
	@echo "... ArgumentConfig.i"
	@echo "... ArgumentConfig.s"
	@echo "... Arguments.o"
	@echo "... Arguments.i"
	@echo "... Arguments.s"
	@echo "... CommandLineProcessor.o"
	@echo "... CommandLineProcessor.i"
	@echo "... CommandLineProcessor.s"
	@echo "... ConfigManager.o"
	@echo "... ConfigManager.i"
	@echo "... ConfigManager.s"
	@echo "... DefaultSettings.o"
	@echo "... DefaultSettings.i"
	@echo "... DefaultSettings.s"
.PHONY : help



#=============================================================================
# Special targets to cleanup operation of make.

# Special rule to run CMake to check the build system integrity.
# No rule that depends on this can have commands that come from listfiles
# because they might be regenerated.
cmake_check_build_system:
	cd /home/henrik/Projekter/AppFramework/build && $(CMAKE_COMMAND) -S$(CMA
KE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-build-system CMakeFiles/Makefile.cm
ake 0
.PHONY : cmake_check_build_system

---- END OF FILE: /home/henrik/Projekter/AppFramework/build/subprojects/CommandL
ineProcessor/Makefile ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/CMakeLists.txt ----
#
#  This file is part of the AppFramework project.
#
#  AppFramework is free software: you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation, GPL version 4.
#
#  AppFramework is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License version 4 for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.

cmake_minimum_required(VERSION 3.10)
project(testing VERSION 1.0)
set(CMAKE_BUILD_TYPE Debug)
```



```
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
set(CMAKE_BUILD_PARALLEL_LEVEL 5)

include_directories(include)



# Include subprojects
add_subdirectory(subprojects/TimeUtils)
add_subdirectory(subprojects/StringUtils)
add_subdirectory(subprojects/Logger)
add_subdirectory(subprojects/EnvVar)
add_subdirectory(subprojects/UiManager)
add_subdirectory(subprojects/CommandLineProcessor)

option(THREAD_SAFE "Enable thread safety" OFF)
if(THREAD_SAFE)
    add_compile_definitions(THREAD_SAFE)
endif()

option(ENABLE_DEBUG "Enable debugging messages in output" ON)
if(ENABLE_DEBUG)
    add_compile_definitions(ENABLE_DEBUG)
endif()


# Find the nlohmann/json package

find_package(nlohmann_json REQUIRED)

# List of source files

set(SOURCES
    src/main.cpp

#INSERTCLASSPOINT
)

add_executable(main ${SOURCES})
# Set include directories for the main target
target_include_directories(main
    PRIVATE
        ${CMAKE_CURRENT_SOURCE_DIR}/subprojects/UiManager
        # ... other include directories ...
)
# Link the executable with TimeUtils and StringUtils
target_link_libraries(main PRIVATE TimeUtils)
target_link_libraries(main PRIVATE StringUtils)
target_link_libraries(main PRIVATE Logger)
target_link_libraries(main PRIVATE EnvVar)
target_link_libraries(main PRIVATE UiManager)
target_link_libraries(main PRIVATE CommandLineProcessor)

# Link the JSON library

target_link_libraries(main PRIVATE nlohmann_json::nlohmann_json)
---- END OF FILE: /home/henrik/Projekter/AppFramework/CMakeLists.txt ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/src/main.cpp ----
/*
  This file is part of the AppFramework project.
```

```
  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
================================================================================*
/

//main.cpp

#include <iostream>
#include "EnvVar.hpp"
#include "ConfigManager.hpp"
#include "Logger.hpp"
#include "ArgumentConfig.hpp"
#include "Arguments.hpp"
#include "CommandLineProcessor.hpp"
#include "UiManager.hpp"

//================================================================================
/* @brief Main entrance point of the program.
 * @param[in] argc The count of arguments provided
 * @param[in] argv A list of char* of arguments
 * @return An integer that denotes the endstate of the program to the OS
 * @since 1.0.0
 * @version 1.1
 * @author Henrik SÃ¸rensen
 * @date 2023-01-01
 * @todo Further Development
 */
//================================================================================
int main(int argc, char* argv[]) {
    //============================================================================
    // Initialization
    //============================================================================
    try {
        // Define and process command-line arguments
        auto definedArgs = ArgumentConfig::getDefinedArguments();
        Arguments cmdArgs(argc, argv, definedArgs);
        CommandLineProcessor cmdProcessor(cmdArgs);
        ArgumentConfig::setupArguments(cmdProcessor);
        cmdProcessor.Process();

        // Initialize configuration manager
        ConfigManager configManager("config.json", cmdArgs);

        // Initialize environment variables
        EnvVarUtils myVar("LOGPATH");
        std::string logPathValue = myVar.get();
        Logger::getInstance().log("LOGPATH value: " + logPathValue, "main", Logg
er::Severity::Info);

        // Create and run the UI
        UIManager uiManager(configManager, cmdArgs);
        uiManager.run();
```



```
    } catch (const std::exception& e) {
        std::cerr << "Error During Initialization: " << e.what() << std::endl;
        exit(1);
    }

    //============================================================================
    // Cleanup code
    //============================================================================
    // Any necessary cleanup before exiting

    return 0;
}---- END OF FILE: /home/henrik/Projekter/AppFramework/src/main.cpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/include/Version.hpp ----
/*
  This file is part of the AppEssential project.

  AppEssential is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppEssential is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppEssential. If not, see <https://www.gnu.org/licenses/>.
*/

// Version.hpp

#ifndef VERSION_HPP
#define VERSION_HPP

#include <string>

const std::string VERSION = "1.0.2";

#endif // VERSION_HPP
---- END OF FILE: /home/henrik/Projekter/AppFramework/include/Version.hpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/config/skel.cpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// <classname>.cpp
```

```
#include "<classname>.hpp"

#ifdef THREAD_SAFE
#include <mutex>
std::mutex EnvVar::mtx;  // Define the static mutex
#endif---- END OF FILE: /home/henrik/Projekter/AppFramework/config/skel.cpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/config/skel.hpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// <classname>.hpp

#ifndef <CAPITALIZED CLASSNAME>_HPP
#define <CAPITALIZED CLASSNAME>_HPP

#ifdef THREAD_SAFE
#include <mutex>
#endif

class <classname> {

};
#endif // <CAPITALIZED CLASSNAME>_HPP---- END OF FILE: /home/henrik/Projekter/Ap
pFramework/config/skel.hpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/Logger/Logge
r.cpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// Logger.cpp

#include <iostream>
#include <fstream>
#include <sstream>
```



```
#include <chrono>
#include <iomanip>
#include "Logger.hpp"
#include "DefaultSettings.hpp"
#include "EnvVar.hpp"
#include "StringUtils.hpp"
#include "TimeUtils.hpp"

Logger::Logger() {
    // Fetch default log path from DefaultSettings
    std::string defaultLogPath = DefaultSettings::getDefaultConfig()["AppFramewo
rk"]["Config"]["Defaults"]["Logger"]["defaultLogPath"];

    std::string logPath = defaultLogPath; // Use default log path

    // Use std::getenv directly to avoid dependency on EnvVar
    const char* configPath = std::getenv("LOGPATH");
    if (configPath != nullptr) {
        logPath = std::string(configPath) + "/" + defaultLogPath;
    }

    logFile.open(logPath, std::ios::out | std::ios::app);
}

Logger::~Logger() {
    if (logFile.is_open()) {
        logFile.close();
    }
}

Logger& Logger::getInstance() {
    static Logger instance;
    return instance;
}

void Logger::log(const std::string& message, const std::string& location, Logger
::Severity severity) {
    if (logFile.is_open()) {
        // Fetch the time format and log entry format from DefaultSettings
        std::string timeFormat = DefaultSettings::getDefaultConfig()["AppFramewo
rk"]["Config"]["Defaults"]["Logger"]["timeFormat"];
        std::string logEntryFormat = DefaultSettings::getDefaultConfig()["AppFra
mework"]["Config"]["Defaults"]["Logger"]["logEntryFormat"];

        // Get the current timestamp in the specified format
        auto now = std::chrono::system_clock::now();
        auto now_time_t = std::chrono::system_clock::to_time_t(now);
        std::tm now_localtime = *std::localtime(&now_time_t);
        std::stringstream timestamp;
        timestamp << std::put_time(&now_localtime, timeFormat.c_str());

        // Replace placeholders in the log entry format
        StringUtils::replaceAll(logEntryFormat, "%timestamp%", timestamp.str());
        StringUtils::replaceAll(logEntryFormat, "%level%", severityToString(seve
rity));
        StringUtils::replaceAll(logEntryFormat, "%location%", location);
        StringUtils::replaceAll(logEntryFormat, "%message%", message);

        // Write the formatted log entry
        logFile << logEntryFormat << std::endl;
    }
}
```

```
std::string Logger::severityToString(Severity severity) {
    switch (severity) {
        case Severity::Trace:   return "TRACE";
        case Severity::Debug:   return "DEBUG";
        case Severity::Info:    return "INFO";
        case Severity::Warning: return "WARNING";
        case Severity::Error:   return "ERROR";
        case Severity::Fatal:   return "FATAL";
        default:
            return "UNKNOWN";
        }
}

std::string Logger::formatMessage(const std::string& message, Severity severity,
 const std::string& location) {
    // Fetch default log entry format and time format
    std::string defaultLogEntryFormat = DefaultSettings::getDefaultConfig()["App
Framework"]["Config"]["Defaults"]["Logger"]["logEntryFormat"];
    std::string timeFormat = DefaultSettings::getDefaultConfig()["AppFramework"]
["Config"]["Defaults"]["Logger"]["timeFormat"];

    // Format the message using the default format
    std::string formattedMessage = defaultLogEntryFormat;
    StringUtils::replaceAll(formattedMessage, "%timestamp%", TimeUtils::getCurre
ntTimestamp(timeFormat));
    StringUtils::replaceAll(formattedMessage, "%level%", severityToString(severi
ty));
    StringUtils::replaceAll(formattedMessage, "%message%", message);
    StringUtils::replaceAll(formattedMessage, "%location%", location);

    return formattedMessage;
}---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/Logger/Logger
.cpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/Logger/CMake
Lists.txt ----
# CMakeLists.txt for Logger subproject

# Define the Logger library
add_library(Logger
    Logger.cpp
    # ... other source files ...
)

# Link Logger with TimeUtils, StringUtils, and EnvVar
target_link_libraries(Logger
    PRIVATE
        StringUtils
        TimeUtils
        EnvVar
)

# Set include directories for Logger library
target_include_directories(Logger
    PUBLIC
        ${CMAKE_CURRENT_SOURCE_DIR}
        ${CMAKE_CURRENT_SOURCE_DIR}/../CommandLineProcessor  # Include CommandLi
neProcessor subdirectory
)---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/Logger/CMakeL
ists.txt ----
```



```
---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/Logger/Logge
r.hpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// Logger.hpp

#ifndef LOGGER_HPP
#define LOGGER_HPP

#include <string>
#include <fstream>
#include <mutex>

class Logger {
public:
    enum class Severity {
        Trace,
        Debug,
        Info,
        Warning,
        Error,
        Fatal
    };

    static Logger& getInstance();
    void log(const std::string& message, const std::string& location, Severity s
everity);
    void setFormat(const std::string& format) {
        this->logFormat = format;
    }
    std::string formatMessage(const std::string& message, Severity severity, con
st std::string& location);
private:
    std::ofstream logFile;
    std::mutex mtx;

    Logger(); // Private constructor for Singleton pattern
    ~Logger();
    Logger(const Logger&) = delete;
    Logger& operator=(const Logger&) = delete;
    std::string logFormat;
    std::string severityToString(Severity severity);
};

#endif // LOGGER_HPP
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/Logger/Logger.
hpp ----
```

```
---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/UiManager/Ui
Manager.hpp ----
/*
  This file is part of the AppEssential project.

  AppEssential is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppEssential is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppEssential. If not, see <https://www.gnu.org/licenses/>.
*/

// UiManager.hpp

#ifndef UIMANAGER_HPP
#define UIMANAGER_HPP

#include <gtk/gtk.h>
#include "ConfigManager.hpp"
#include "Arguments.hpp"

class UIManager {
private:
    GtkWidget *window;
    ConfigManager& configManager;
    Arguments& cmdArgs;

    void initializeGtk();
    void createMainWindow();

public:
    UIManager(ConfigManager& configManager, Arguments& cmdArgs);
    ~UIManager();

    void run();
};

#endif // UIMANAGER_HPP
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/UiManager/UiMa
nager.hpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/UiManager/CM
akeLists.txt ----
cmake_minimum_required(VERSION 3.0)
project(UiManager)

# Find the GTK package
find_package(PkgConfig REQUIRED)
pkg_check_modules(GTK3 REQUIRED gtk+-3.0)

# Define the UiManager library
add_library(UiManager
    UiManager.cpp
)
```



```
# Include GTK headers
target_include_directories(UiManager
    PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}
        ${GTK3_INCLUDE_DIRS}
        ${CMAKE_CURRENT_SOURCE_DIR}/../CommandLineProcessor
)

# Link GTK libraries
target_link_libraries(UiManager
    ${GTK3_LIBRARIES}
)

# Additional GTK flags
target_compile_options(UiManager
    PUBLIC ${GTK3_CFLAGS_OTHER}
)
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/UiManager/CMak
eLists.txt ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/UiManager/Ui
Manager.cpp ----
/*
  This file is part of the AppEssential project.

  AppEssential is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppEssential is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppEssential. If not, see <https://www.gnu.org/licenses/>.
*/

// UiManager.cpp

#include "UiManager.hpp"
#include <clocale>
#include <libintl.h>
#include <string>

#define _(String) gettext(String)

// Constructor


UIManager::UIManager(ConfigManager& configManager, Arguments& cmdArgs)
    : configManager(configManager), cmdArgs(cmdArgs) {
    initializeGtk();
    createMainWindow();
}

UIManager::~UIManager() {
    // Assuming window will be automatically destroyed when the main loop quits
    // If any additional cleanup is needed, do it here
}

void UIManager::initializeGtk() {
    // Extract the GTK-specific arguments from the command line arguments
```

```
    std::vector<char*> gtkArgs;
    auto gtkArgList = cmdArgs.getGtkArguments(); // You need to implement this m
ethod
    for (auto& arg : gtkArgList) {
        gtkArgs.push_back(const_cast<char*>(arg.c_str()));
    }
    int gtkArgc = gtkArgs.size();
    char** gtkArgv = gtkArgs.data();

    // Initialize GTK
    gtk_init(&gtkArgc, &gtkArgv);
}

void UIManager::createMainWindow() {
    // Use settings from ConfigManager to set up the main window
    int width = configManager.get<int>("AppFramework.Config.Defaults.gtk.window.
width");
    int height = configManager.get<int>("AppFramework.Config.Defaults.gtk.window
.height");
    int xpos = configManager.get<int>("AppFramework.Config.Defaults.gtk.window.p
os_x");
        int ypos = configManager.get<int>("AppFramework.Config.Defaults.gtk.wind
ow.pos_y");
        std::string title = configManager.get<std::string>("AppFramework.Config.
Defaults.gtk.window.title");

    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);

    gtk_window_set_default_size(GTK_WINDOW(window), width, height);
        gtk_window_move(GTK_WINDOW(window), xpos, ypos);
        gtk_window_set_title(GTK_WINDOW(window),title.c_str());

    // Connect the window's destroy signal to gtk_main_quit to exit the applicat
ion
    g_signal_connect(window, "destroy", G_CALLBACK(gtk_main_quit), NULL);

    // Additional setup for widgets inside the window
    // ...
}

void UIManager::run() {
    // Show the window and all its contents
    gtk_widget_show_all(window);

    // Enter the GTK main event loop
    gtk_main();
}
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/UiManager/UiMa
nager.cpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/EnvVar/CMake
Lists.txt ----
# Define the EnvVar library

add_library(EnvVar
    EnvVar.cpp
)

# Set include directories for EnvVar library
target_include_directories(EnvVar
    PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}
)---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/EnvVar/CMakeL
```



```
ists.txt ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/EnvVar/EnvVa
r.hpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// EnvVar.hpp

#ifndef ENVVAR_HPP
#define ENVVAR_HPP

#include <string>
#include <optional>
#ifdef THREAD_SAFE
#include <mutex>
#endif

class EnvVarUtils {
public:
    explicit EnvVarUtils(const std::string& name);

    std::string get() const;
    bool set(const std::string& value) const;
    void store();
    bool restore() const;
//Todo: Expand the functionality and add more EnvVar methodes
private:
    std::string varName;
    std::optional<std::string> storedValue;

#ifdef THREAD_SAFE
    static std::mutex mtx;  // Mutex for thread safety
#endif
};

#endif // ENVVAR_HPP

---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/EnvVar/EnvVar.
hpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/EnvVar/EnvVa
r.cpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.
```

```
  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// EnvVarUtils.cpp

#include "EnvVar.hpp"
//#include "Logger.hpp"
#include <cstdlib>
#ifdef THREAD_SAFE
#include <mutex>
std::mutex EnvVarUtils::mtx;  // Define the static mutex
#endif

EnvVarUtils::EnvVarUtils(const std::string& name) : varName(name) {}

std::string EnvVarUtils::get() const {
#ifdef THREAD_SAFE
    std::lock_guard<std::mutex> lock(mtx);  // Lock the mutex
#endif
    const char* value = std::getenv(varName.c_str());
    return (value != nullptr) ? std::string(value) : std::string();
}

bool EnvVarUtils::set(const std::string& value) const {
#ifdef THREAD_SAFE
    std::lock_guard<std::mutex> lock(mtx);  // Lock the mutex
#endif
    return setenv(varName.c_str(), value.c_str(), 1) == 0;
}

void EnvVarUtils::store() {
#ifdef THREAD_SAFE
    std::lock_guard<std::mutex> lock(mtx);  // Lock the mutex
#endif
    storedValue = get();
}

bool EnvVarUtils::restore() const {
#ifdef THREAD_SAFE
    std::lock_guard<std::mutex> lock(mtx);  // Lock the mutex
#endif
    if (storedValue.has_value()) {
        return set(storedValue.value());
    }
    return false;
}---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/EnvVar/EnvVar
.cpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/TimeUtils/Ti
meUtils.cpp ----
/*
  This file is part of the AppEssential project.

  AppEssential is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
```



```
  the Free Software Foundation, GPL version 4.

  AppEssential is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppEssential. If not, see <https://www.gnu.org/licenses/>.
*/

// TimeUtils.cpp

#include "TimeUtils.hpp"
#include <chrono>
#include <iomanip>
#include <sstream>

namespace TimeUtils {
    std::string getCurrentTimestamp() {
        return getCurrentTimestamp("%Y-%m-%d %H:%M:%S");
    }

    std::string getCurrentTimestamp(const std::string& timeFormat) {
        auto now = std::chrono::system_clock::now();
        auto now_time_t = std::chrono::system_clock::to_time_t(now);
        auto now_localtime = *std::localtime(&now_time_t);

        std::ostringstream timestampStream;
        timestampStream << std::put_time(&now_localtime, timeFormat.c_str());
        return timestampStream.str();
    }

    std::chrono::system_clock::time_point parseTimestamp(const std::string& time
stamp, const std::string& timeFormat) {
        std::tm tm = {};
        std::istringstream timestampStream(timestamp);
        timestampStream >> std::get_time(&tm, timeFormat.c_str());

        if (timestampStream.fail()) {
            throw std::invalid_argument("Invalid timestamp or format.");
        }

        std::time_t time = std::mktime(&tm);
        return std::chrono::system_clock::from_time_t(time);
    }

    std::chrono::seconds calculateTimeDifference(const std::string& timestamp1,
const std::string& timestamp2, const std::string& timeFormat) {
        std::chrono::system_clock::time_point time1 = parseTimestamp(timestamp1,
 timeFormat);
        std::chrono::system_clock::time_point time2 = parseTimestamp(timestamp2,
 timeFormat);

        std::chrono::seconds difference = std::chrono::duration_cast<std::chrono
::seconds>(time2 - time1);
        return difference;
    }
} // namespace TimeUtils
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/TimeUtils/Time
Utils.cpp ----
```

```
---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/TimeUtils/CM
akeLists.txt ----
# Define the TimeUtils library

add_library(TimeUtils
    TimeUtils.cpp
)

# Set include directories for TimeUtils library
target_include_directories(TimeUtils
    PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}
)
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/TimeUtils/CMak
eLists.txt ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/TimeUtils/Ti
meUtils.hpp ----
/*
  This file is part of the AppEssential project.

  AppEssential is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppEssential is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppEssential. If not, see <https://www.gnu.org/licenses/>.
*/

// TimeUtils.hpp

#ifndef TIMEUTILS_HPP
#define TIMEUTILS_HPP

#include <string>
#include <chrono>

namespace TimeUtils {
    std::string getCurrentTimestamp();
    std::string getCurrentTimestamp(const std::string& timeFormat);
    std::chrono::system_clock::time_point parseTimestamp(const std::string& time
stamp, const std::string& timeFormat);
    std::chrono::seconds calculateTimeDifference(const std::string& timestamp1,
const std::string& timestamp2, const std::string& timeFormat);
} // namespace TimeUtils

#endif // TIMEUTILS_HPP
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/TimeUtils/Time
Utils.hpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/StringUtils/
StringUtils.cpp ----
/*
  This file is part of the AppEssential project.

  AppEssential is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.
```



```
  AppEssential is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppEssential. If not, see <https://www.gnu.org/licenses/>.
*/

// StringUtils.hpp

#include "StringUtils.hpp"

namespace StringUtils {

    void replaceAll(std::string& str, const std::string& from, const std::string
& to) {
        if (from.empty()) {
            return;
        }
        size_t startPos = 0;
        while ((startPos = str.find(from, startPos)) != std::string::npos) {
            str.replace(startPos, from.length(), to);
            startPos += to.length();
        }
    }

    // Implementations of other utility functions

} // namespace StringUtils
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/StringUtils/St
ringUtils.cpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/StringUtils/
CMakeLists.txt ----
# Define the StringUtils library

add_library(StringUtils
    StringUtils.cpp
)

# Set include directories for TimeUtils library
target_include_directories(StringUtils
    PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}
)
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/StringUtils/CM
akeLists.txt ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/StringUtils/
StringUtils.hpp ----
/*
  This file is part of the AppEssential project.

  AppEssential is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppEssential is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.
```

```
  You should have received a copy of the GNU General Public License
  along with AppEssential. If not, see <https://www.gnu.org/licenses/>.
*/

//StringUtils.cpp

#ifndef STRINGUTILS_HPP
#define STRINGUTILS_HPP

#include <string>

namespace StringUtils {

    void replaceAll(std::string& str, const std::string& from, const std::string
& to);

    // Add more string utility functions here as needed

} // namespace StringUtils

#endif // STRINGUTILS_HPP
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/StringUtils/St
ringUtils.hpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLineP
rocessor/ArgumentConfig.hpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// ArgumentConfig.hpp

#ifndef ARGUMENTCONFIG_HPP
#define ARGUMENTCONFIG_HPP

#include "Argument.hpp"
#include "CommandLineProcessor.hpp"
#include <iostream> // Include for std::cout
#include <memory>
#include <map>

void handleHelp(const std::shared_ptr<Argument>& arg);
void handleVersion(const std::shared_ptr<Argument>& arg);
void handleDumpConfig(const std::shared_ptr<Argument>& arg);
void dumpDefaultConfig();

class ArgumentConfig {
public:
    static std::map<std::string, Argument> getDefinedArguments();
```



```
    static void setupArguments(CommandLineProcessor& cmdProcessor);
};

#endif // ARGUMENTCONFIG_HPP---- END OF FILE: /home/henrik/Projekter/AppFramewor
k/subprojects/CommandLineProcessor/ArgumentConfig.hpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLineP
rocessor/DefaultSettings.cpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

//DefaultSettings.cpp

#include "DefaultSettings.hpp"

//============================================================================
// Write your default hardcoded configs here in json format
//============================================================================
const nlohmann::json DefaultSettings::defaultConfig = {
    {"AppFramework", {
        {"Config", {
            {"Defaults", {
                {"gtk", {
                    {"module", "default_module"},
                    {"window", {
                        {"height", 600},
                        {"pos_x", 500},
                        {"pos_y", 500},
                        {"width", 800},
                        {"title", "AppFramework test"}
                    }}
                }},
                {"Logger", {
                    {"defaultLogPath", "testing.log"},
                    {"logEntryFormat", "[%timestamp%] [%level%] %location%: %mes
sage%"},
                    {"timeFormat", "%Y-%m-%d %H:%M:%S"}
                }}
                // ... other default settings ...
            }}
        }}
    }}
};
//============================================================================
//
//============================================================================

const nlohmann::json DefaultSettings::getDefaultConfig() {
    return defaultConfig;
```

```
}---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLinePr
ocessor/DefaultSettings.cpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLineP
rocessor/ArgumentConfig.cpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// ArgumentConfig.cpp


#include <iostream>
#include <fstream>  // Include for std::ofstream
#include <nlohmann/json.hpp>  // Include for JSON handling
#include "Argument.hpp"
#include "ArgumentConfig.hpp"
#include "DefaultSettings.hpp"
#include "Version.hpp"
#include "ConfigManager.hpp"

std::vector<ArgumentData> argumentDefinitions = {
    {"--help", "-h", "Display help information", true, false, E_Argument_ValueTy
pe::None},
    {"--version", "-v", "Display version information", true, false, E_Argument_V
alueType::None},
    {"--dumpconfig", "-D", "Dumps the default settings to config.json", true, fa
lse, E_Argument_ValueType::None}
    // Add other arguments here
};
void handleHelp(const std::shared_ptr<Argument>& arg) {
    std::cout << "Help requested: " << arg->getDescription() << std::endl;

    // Print the command-line options and their descriptions
    std::cout << "Available command-line options:" << std::endl;
    for (const auto& argData : argumentDefinitions) {
        std::cout << "  " << argData.longName << ", " << argData.shortName << "\
n    " << argData.description << std::endl;
    }
    exit(0);
}

void handleVersion(const std::shared_ptr<Argument>& arg) {
        std::cout << VERSION << std::endl;
        exit(0);
}

void handleDumpConfig(const std::shared_ptr<Argument>& arg) {
    dumpDefaultConfig();
    exit(0); // You might want to exit the program after dumping the config
```



```
}

void dumpDefaultConfig() {
    nlohmann::json defaultConfig = DefaultSettings::getDefaultConfig();
    std::ofstream configFile("config.json");
    configFile << defaultConfig.dump(4); // '4' for pretty printing
    configFile.close();

    std::cout << "Default configuration dumped to config.json" << std::endl;
}

std::map<std::string, Argument> ArgumentConfig::getDefinedArguments() {
    std::map<std::string, Argument> definedArgs;
    for (const auto& argData : argumentDefinitions) {
        definedArgs.emplace(argData.longName, Argument(argData));
    }
    return definedArgs;
}

void ArgumentConfig::setupArguments(CommandLineProcessor& cmdProcessor) {
    // Define your arguments
    Argument helpArg("--help", "-h", "Display help information", true, false, E_
Argument_ValueType::None);
    Argument versionArg("--version", "-v", "Display version information", true,
false, E_Argument_ValueType::None);
    Argument dumpConfigArg("--dumpconfig", "-D", "Dumps the default settings to
config.json", true, false, E_Argument_ValueType::None);

    // Add handler functions
    cmdProcessor.AddArgumentHandler(helpArg, handleHelp);
    cmdProcessor.AddArgumentHandler(versionArg, handleVersion);
        cmdProcessor.AddArgumentHandler(dumpConfigArg, handleDumpConfig);
    // ... other handlers ...
}
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLinePro
cessor/ArgumentConfig.cpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLineP
rocessor/ConfigManager.cpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// ConfigManager.cpp

#include "Arguments.hpp"
#include "ConfigManager.hpp"
#include "Logger.hpp"
#include <DefaultSettings.hpp>
#include <nlohmann/json.hpp>
```

```
#include <iostream>
#include <fstream>
#include <sstream>

#ifdef THREAD_SAFE
std::mutex ConfigManager::mtx;
#endif

template<typename T>
void ConfigManager::set(const std::string& key, const T& value) {
    #ifdef THREAD_SAFE
    std::lock_guard<std::mutex> lock(mtx);
    #endif

    nlohmann::json& ref = getRefToValue(key);
    ref = value;
}

template<typename T>
T ConfigManager::get(const std::string& key) const {
    #ifdef THREAD_SAFE
    std::lock_guard<std::mutex> lock(mtx);
    #endif

    // Check if a command line argument exists and has a value for the key
    if (commandLineArgs.find(key) != commandLineArgs.end()) {
        std::istringstream valueStream(commandLineArgs.at(key));
        T typedValue;
        valueStream >> typedValue;
        return typedValue;
    }

    try {
        const nlohmann::json& ref = getRefToValue(key, true);
        return ref.get<T>();
    } catch (const nlohmann::json::out_of_range& e) {
        // Handle the case where the key does not exist
        Logger::getInstance().log("Key not found in configuration: " + key, "Con
figManager::get", Logger::Severity::Warning);
    }
        return T(); // Returns a default-constructed object of type T
    // Additional catch for other JSON exceptions might be needed
}

ConfigManager::ConfigManager(const std::string& configFilePath, const Arguments&
 cmdArgs): filePath(configFilePath), cmdArgs(cmdArgs) {
    applyDefaults();

    std::ifstream file(filePath);
    if (file) {
        try {
            nlohmann::json fileConfig;
            file >> fileConfig;
            config.merge_patch(fileConfig);

            std::cout << "ConfigManager::ConfigManager:" << std::endl;
            std::cout << config.dump(4) << std::endl;
        } catch (const nlohmann::json::parse_error& e) {
            std::cout << "Caught an exception" << std::endl;
            Logger::getInstance().log("JSON parsing error: " + std::string(e.wha
t()), "ConfigManager::ConfigManager", Logger::Severity::Error);
            std::cout << "Configuration loading error. Check log file for detail
```



```
s." << std::endl;
        }
    } else {
        Logger::getInstance().log("Config file not found: " + filePath, "ConfigM
anager::ConfigManager", Logger::Severity::Warning);
        std::cerr << "Configuration file missing. A new one will be created." <<
 std::endl;
        config = nlohmann::json::object(); // Initialize config as an empty obje
ct
    }

    // Parse and store command line arguments from cmdArgs
    for (const auto& argPair : cmdArgs.getArgValues()) {
        std::string key = argPair.first;
        std::string value = argPair.second;
        commandLineArgs[key] = value;
    }
}

void ConfigManager::applyDefaults() {
    auto defaultConfig = DefaultSettings::getDefaultConfig();
    config = defaultConfig; // Start with default config

    std::ifstream file(filePath);
    if (file) {
        try {
            nlohmann::json fileConfig;
            file >> fileConfig;
            mergeJson(config, fileConfig);
        } catch (const nlohmann::json::parse_error& e) {
            Logger::getInstance().log("JSON parsing error: " + std::string(e.wha
t()), "ConfigManager::ConfigManager", Logger::Severity::Error);
        }
    }
}
void ConfigManager::mergeJson(nlohmann::json& base, const nlohmann::json& update
) {
    for (auto& el : base.items()) {
        if (update.contains(el.key())) {
            if (el.value().is_object() && update[el.key()].is_object()) {
                mergeJson(base[el.key()], update[el.key()]);
            } else {
                // For non-object types, or if the type in 'update' differs, ove
rwrite the value in 'base'
                base[el.key()] = update[el.key()];
            }
        }
    }
}
void ConfigManager::parseCommandLineArgs(int argc, char** argv) {
    for (int i = 1; i < argc; ++i) {
        std::string arg = argv[i];
        if (arg.size() >= 2 && arg.substr(0, 2) == "--") {
            size_t equalPos = arg.find('=');
            if (equalPos != std::string::npos) {
                std::string key = arg.substr(2, equalPos - 2);
                std::string value = arg.substr(equalPos + 1);
                commandLineArgs[key] = value;
            }
        }
    }
}
```

```
nlohmann::json& ConfigManager::getRefToValue(const std::string& key) {
    nlohmann::json* j = &config;
    std::istringstream iss(key);
    std::string token;
    while (std::getline(iss, token, '.')) {
        j = &((*j)[token]);
    }
    return *j;
}

// Explicit template instantiation
template int ConfigManager::get<int>(const std::string& key) const;
template std::string ConfigManager::get<std::string>(const std::string& key) con
st;
template void ConfigManager::set<int>(const std::string& key, const int& value);
template void ConfigManager::set<std::string>(const std::string& key, const std:
:string& value);

ConfigManager::~ConfigManager() {}
// Other methods...
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLinePro
cessor/ConfigManager.cpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLineP
rocessor/Argument.cpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// Argument.cpp

#include "Argument.hpp"

Argument::Argument(     const std::string& longname,
                                const std::string& shortname,
                        const std::string& description,
                        bool hasShortValue,
                        bool needValue,
                        E_Argument_ValueType valueType) :
                                longName(longname),
                        shortName(shortname),
                        description(description),
                        hasShortValueflag(hasShortValue),
                        needValue(needValue),
                        valueType(valueType) {}

Argument::Argument(const ArgumentData& data):
longName(data.longName),
shortName(data.shortName),
```



```
description(data.description),
hasShortValueflag(data.hasShortValue),
needValue(data.needValue),
valueType(data.valueType),
defaultValue(data.defaultValue),
hasDefaultValueFlag(data.hasDefaultValue){}

std::string                     Argument::getLongName()         const   {return
longName;}
std::string                     Argument::getShortName()        const   {return
shortName;}
std::string                     Argument::getDescription()      const   {return
description;}
bool                                    Argument::hasShortValue()       const
{return hasShortValueflag;}
bool                                    Argument::needsValue()          const
{return needValue;}
E_Argument_ValueType    Argument::getValueType()        const   {return valueTyp
e;}

void Argument::setValue(const std::string& value) {
    this->value = value;
}

void Argument::setDefaultValue(const std::string& defaultValue) {
    this->defaultValue = defaultValue;
    this->hasDefaultValueFlag = true;
}

std::string Argument::getDefaultValue() const {
    return defaultValue;
}

bool Argument::hasDefaultValue() const {
    return hasDefaultValueFlag;
}

//CHANGED: Moved the template function definition to the header file
template <typename T>
T Argument::getValue() const {
    // Check if a value has been set for this argument
    if (value.empty()) {
        // If no value has been set and a default value is available, return it
        if (hasDefaultValueFlag) {
            std::istringstream defaultValueStream(defaultValue);
            T defaultValueValue;
            defaultValueStream >> defaultValueValue;
            return defaultValueValue;
        } else {
            // If no value or default value is available, throw an exception or
return a suitable default value
            throw std::logic_error("No value set for this argument.");
        }
    }

    // Convert the stored string value to the requested type based on valueType
    std::istringstream valueStream(value);
    T typedValue;

    // Use try-catch for error handling during conversion
    try {
        valueStream >> typedValue;
```

```
    } catch (const std::exception& e) {
        // Conversion failed; throw an exception or handle the error accordingly
        throw std::invalid_argument("Error converting argument value to the requ
ested type.");
    }

    if (valueStream.fail() || !valueStream.eof()) {
        // Conversion failed or there is extra data in the string
        throw std::invalid_argument("Invalid argument value.");
    }

    return typedValue;
}
// Instantiate the template function for specific types you use in Argument.cpp
template std::string Argument::getValue<std::string>() const;
template int Argument::getValue<int>() const;
template float Argument::getValue<float>() const;---- END OF FILE: /home/henrik/
Projekter/AppFramework/subprojects/CommandLineProcessor/Argument.cpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLineP
rocessor/ConfigManager.hpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// ConfigManager.hpp

#ifndef CONFIGMANAGER_HPP
#define CONFIGMANAGER_HPP

#include "Arguments.hpp"
#include <iostream>
#include <string>
#include <mutex>
#include <nlohmann/json.hpp>
#include <unordered_map>

class ConfigManager {
public:
    ConfigManager(const std::string& configFilePath, const Arguments& cmdArgs);
    ~ConfigManager();

    void applyDefaults();
        void mergeJson(nlohmann::json& base, const nlohmann::json& update);

    template<typename T>
    T get(const std::string& key) const;

    template<typename T>
    void set(const std::string& key, const T& value);
```



```
    void sync();

private:
    const Arguments& cmdArgs; // Reference to an Arguments instance
    nlohmann::json config;
    std::string filePath;

    const nlohmann::json& getRefToValue(const std::string& key, bool forRead) co
nst;
    nlohmann::json& getRefToValue(const std::string& key);

    std::unordered_map<std::string, std::string> commandLineArgs; // Store comma
nd line arguments

    void parseCommandLineArgs(int argc, char** argv);

#ifdef THREAD_SAFE
    static std::mutex mtx;
#endif
};

#endif // CONFIGMANAGER_HPP
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLinePro
cessor/ConfigManager.hpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLineP
rocessor/Arguments.hpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

//Arguments.hpp

#ifndef ARGUMENTS_HPP
#define ARGUMENTS_HPP

#include "Argument.hpp"
#include <string>
#include <map>
#include <stdexcept>
#include <vector>

class Arguments {
public:
    Arguments(int argc, char* argv[], const std::map<std::string, Argument>& def
inedArgs);

    const std::map<std::string, std::string>& getArgValues() const { return argV
alues; }
```

```
    std::string getArgValue(const std::string& argName) const;
    bool isInArgs(const std::string& str) const;
        std::vector<std::string> getGtkArguments() const;
private:
    std::map<std::string, std::string> argValues; // Maps argument names to thei
r values
};

#endif // ARGUMENTS_HPP

---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLinePro
cessor/Arguments.hpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLineP
rocessor/CMakeLists.txt ----
cmake_minimum_required(VERSION 3.10)

# Define the CommandLineProcessor library
add_library(CommandLineProcessor
    ArgumentConfig.cpp
    Argument.cpp
    Arguments.cpp
    CommandLineProcessor.cpp
    ConfigManager.cpp
    DefaultSettings.cpp
)

# Set include directories for CommandLineProcessor library
target_include_directories(CommandLineProcessor
    PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}
)

# Link CommandLineProcessor with other subproject dependencies (if any)
target_link_libraries(CommandLineProcessor
    PRIVATE
        Logger
        StringUtils
        TimeUtils
        EnvVar
    # Add any other dependencies as needed
)
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLinePro
cessor/CMakeLists.txt ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLineP
rocessor/Arguments.cpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/
```



```
// Arguments.cpp

#include <iostream>
#include "Arguments.hpp"

Arguments::Arguments(int argc, char* argv[], const std::map<std::string, Argumen
t>& definedArgs) {
    for (const auto& argPair : definedArgs) {
        if (argPair.second.hasDefaultValue()) {
            argValues[argPair.first] = argPair.second.getDefaultValue();
        }
    }

    for (int i = 1; i < argc; ++i) {
        std::string currentArg = argv[i];
        auto it = definedArgs.find(currentArg);

        if (it == definedArgs.end()) {
            for (const auto& argPair : definedArgs) {
                if (argPair.second.getShortName() == currentArg) {
                    it = definedArgs.find(argPair.second.getLongName());
                    break;
                }
            }
        }

        if (it != definedArgs.end()) {
            const Argument& arg = it->second;

            if (arg.needsValue()) {
                if (i + 1 < argc) {
                    argValues[it->first] = argv[i + 1];
                    ++i;
                } else {
                    throw std::runtime_error("Missing argument value for " + cur
rentArg);
                }
            } else {
                argValues[it->first] = "";
            }
        } else {
            throw std::runtime_error("Unknown argument " + currentArg);
        }
    }
}

std::string Arguments::getArgValue(const std::string& argName) const {
    auto it = argValues.find(argName);
    if (it != argValues.end()) {
        return it->second;
    }
    return "";
}

bool Arguments::isInArgs(const std::string& str) const {
    return argValues.find(str) != argValues.end();
}

std::vector<std::string> Arguments::getGtkArguments() const {
    std::vector<std::string> gtkArgs;

    for (const auto& argPair : argValues) {
```

```
        const std::string& arg = argPair.first;

        // Check if the argument is a GTK argument
        if (arg.find("--gtk-") == 0) {
            // Construct the argument string as it would appear on the command l
ine
            std::string argStr = arg + (argPair.second.empty() ? "" : "=" + argP
air.second);
            gtkArgs.push_back(argStr);
        }
    }

    return gtkArgs;
}---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLinePr
ocessor/Arguments.cpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLineP
rocessor/Argument.hpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

//Argument.hpp

#ifndef ARGUMENT_HPP
#define ARGUMENT_HPP

#include <string>
#include <sstream>
#include <stdexcept>

enum class E_Argument_ValueType {
    None,
    Integer,
    Float,
    String,
    // Add more types as needed
};

// Define a struct to hold all argument data (if you are using this approach)
struct ArgumentData {
    std::string longName;
    std::string shortName;
    std::string description;
    bool hasShortValue;
    bool needValue;
    E_Argument_ValueType valueType;
    std::string defaultValue;
    bool hasDefaultValue;
};
```



```
class Argument {
public:
    Argument(const std::string& longname,
                const std::string& shortname,
             const std::string& description,
             const bool hasShortValue,
             const bool needValue,
             const E_Argument_ValueType valueType);

    // Constructor that takes ArgumentData struct
    Argument(const ArgumentData& data);

    std::string getLongName() const;
    std::string getShortName() const;
    std::string getDescription() const;
    bool hasShortValue() const;
    bool needsValue() const;
    E_Argument_ValueType getValueType() const;

    void setValue(const std::string& value);
    void setDefaultValue(const std::string& defaultValue);
    std::string getDefaultValue() const;
    bool hasDefaultValue() const;

    //CHANGED: Moved the template function definition to the header file
    template <typename T>
    T getValue() const;

private:
    std::string longName;
    std::string shortName;
    std::string description;
    bool hasShortValueflag;
    bool needValue;
    E_Argument_ValueType valueType;
    std::string value; // Store the value as a string

    // New members for default value handling
    std::string defaultValue;
    bool hasDefaultValueFlag = false;
};

#endif // ARGUMENT_HPP


---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLinePro
cessor/Argument.hpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLineP
rocessor/CommandLineProcessor.cpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.
```

```
  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

//CommandLineProcessor.cpp

#include <iostream>
#include "CommandLineProcessor.hpp"

CommandLineProcessor::CommandLineProcessor(const Arguments& args) : args(args) {
}

void CommandLineProcessor::AddArgumentHandler(const Argument& arg, ArgumentHandl
er handler) {
    auto argPtr = std::make_shared<Argument>(arg);
    handlers[arg.getLongName()] = ArgumentHandlerPair(argPtr, handler);

    if (!arg.getShortName().empty()) {
        handlers[arg.getShortName()] = ArgumentHandlerPair(argPtr, handler);
    }
}

void CommandLineProcessor::Process() {
    for (const auto& argPair : args.getArgValues()) {
        std::string argName = argPair.first;

        auto it = handlers.find(argName);
        if (it != handlers.end()) {
            // Update the value of the argument before calling the handler
            it->second.arg->setValue(argPair.second);
            it->second.handler(it->second.arg);
        }else{
                std::cout << "No handler found for: " << argName << std::endl;
        }
    }
}---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLinePr
ocessor/CommandLineProcessor.cpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLineP
rocessor/DefaultSettings.hpp ----
/*
  This file is part of the AppEssential project.

  AppEssential is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppEssential is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppEssential. If not, see <https://www.gnu.org/licenses/>.
*/

//DefaultSettings.hpp

#ifndef DEFAULTSETTINGS_HPP
#define DEFAULTSETTINGS_HPP
```



```
#include <nlohmann/json.hpp>

class DefaultSettings {
public:
    static const nlohmann::json defaultConfig;
    static const nlohmann::json getDefaultConfig();
};

#endif // DEFAULTSETTINGS_HPP
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLinePro
cessor/DefaultSettings.hpp ----

---- START OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLineP
rocessor/CommandLineProcessor.hpp ----
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

//CommandLineProcessor.hpp

#ifndef COMMANDLINEPROCESSOR_HPP
#define COMMANDLINEPROCESSOR_HPP

#include <functional>
#include <map>
#include <string>
#include <memory>
#include "Argument.hpp"
#include "Arguments.hpp"

using ArgumentHandler = std::function<void(const std::shared_ptr<Argument>&)>;

struct ArgumentHandlerPair {
    std::shared_ptr<Argument> arg;
    ArgumentHandler handler;

    ArgumentHandlerPair(std::shared_ptr<Argument> arg, const ArgumentHandler& ha
ndler)
        : arg(std::move(arg)), handler(handler) {}

    ArgumentHandlerPair() = default;
};

class CommandLineProcessor {
public:
    explicit CommandLineProcessor(const Arguments& args);

    void AddArgumentHandler(const Argument& arg, ArgumentHandler handler);
    void Process();
```

```
private:
    const Arguments& args;
    std::map<std::string, ArgumentHandlerPair> handlers;
};

#endif // COMMANDLINEPROCESSOR_HPP
---- END OF FILE: /home/henrik/Projekter/AppFramework/subprojects/CommandLinePro
cessor/CommandLineProcessor.hpp ----
```