import subprocess

import os

import logging

from helpers import run\_subprocess

logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s - %(message)s")

logger = logging.getLogger(\_\_name\_\_)

def compile\_code(toolchain\_path, source\_path, binary\_path, debug=False, extra\_flags=None):

"""

Compiles RISC-V source code into an executable.

Args:

toolchain\_path (str): Path to the RISC-V toolchain.

source\_path (str): Path to the source code file.

binary\_path (str): Output path for the compiled binary.

debug (bool): Enables debug logging if True.

extra\_flags (list): Additional compiler flags.

"""

riscv\_gcc = os.path.join(os.path.expanduser(toolchain\_path), 'bin', 'riscv32-unknown-elf-gcc')

# Enable debug mode dynamically

if os.getenv("DEBUG\_MODE") == "1" or debug:

logger.setLevel(logging.DEBUG)

# Validate paths

if not os.path.isdir(toolchain\_path):

logger.error(f"Toolchain path not found: {toolchain\_path}")

raise FileNotFoundError(f"Toolchain path not found: {toolchain\_path}")

if not os.path.isfile(riscv\_gcc):

logger.error(f"RISC-V GCC compiler not found at: {riscv\_gcc}")

raise FileNotFoundError(f"RISC-V GCC compiler not found at: {riscv\_gcc}")

if not os.path.isfile(source\_path):

logger.error(f"Source file not found at: {source\_path}")

raise FileNotFoundError(f"Source file not found at: {source\_path}")

logger.info(f"Using compiler: {riscv\_gcc}")

logger.info(f"Compiling: {source\_path} -> {binary\_path}")

# Construct compilation command

compile\_command = [riscv\_gcc, '-o', binary\_path, source\_path, '-march=rv32im', '-mabi=ilp32']

if extra\_flags:

compile\_command.extend(extra\_flags)

if debug:

logger.debug(f"Compile command: {' '.join(compile\_command)}")

# Run compilation

try:

run\_subprocess(compile\_command)

logger.info(f"Compilation successful: {binary\_path}")

except subprocess.CalledProcessError as e:

logger.error(f"Compilation failed with error: {e.stderr}")

raise

except Exception as e:

logger.error(f"Unexpected error during compilation: {e}")

raise