Shazaib Dawood
CS 211
09/23/2024

- Provide a basic description of how valgrind can be used to help in debugging a program. Describe some of the flag options. Include example(s) of how to run valgrind.
    - Uses
        - can be used to detect memory errors, leaks, initialized memory usage, and invalid memory access in C/C++ programs
        - identifies issues with dynamic memory allocation and deallocation
        - can detect small memory management errors that can cause the program to have unpredictable behavior later
    - flag options
        - - -leak-check = full
            - gives detailed information on memory leaks
        - - -track-origins = yes
            - tracks the origin of uninitialized memory
        - - -tool = memcheck
            - default tool for memory checking
    - example
        - valgrind --leak-check=full ./programName
        - runs a full memory leak check on programName
- Provide a basic description of how gdb can be used to help in debugging a program. Describe some of the various commands used within the gdb environment. Include example(s) of how to run gdb.
    - Uses
        - a command line tool that helps debug C/C++ programs by allowing step-by-step execution, variable inspection, and control over program flow
        - used to analyze core dumps, step through code, and catch runtime errors
        - allows for controlled program execution to make tracing the source of runtime errors easier
    - commands
        - run
            - starts program
        - break [line number / function name]
            - sets a breakpoint at a specific line or function
        - next
            - steps to the next line of code
        - step
            - steps into function calls

        - print [variable name]
            - prints the current value of a variable
        - continue
            - resumes program after a breakpoint
    - examples
        - gdb ./programName
        - (gdb) run
        - (gdb) break main
        - (gdb) print variableName
        - runs gdb on programName, sets a breakpoint in main, prints current value of variableName
- Provide a basic description of how a GUI Debugger (such as CLion, XCode, VScode) can be used to help in debugging a program. Describe some of the various strategies and/or tools within the GUI to make it an effective debugger.
    - Uses
        - a graphical user interface based debugger that is integrated into IDEs
        - has visual tools for setting breakpoints, stepping through code, and inspecting variables
        - has a user friendly interface for managing debugging tasks without having to memorize commands
        - visual tools help map program flow for larger programs
    - tools/strategies
        - breakpoints
            - set breakpoints on specific line numbers to pause program execution
        - watch windows
            - track specific variables as they change during execution
        - call stack
            - visualize the function calls
        - variable inspector
            - edit variables during runtime
    - examples
        - open the program in the IDE
        - set breakpoint on specific line numbers
        - run the program in debug mode
        - step through the program code using navigation tools
        - use watch windows to observe the values of the variable during execution
- How are the different debugging tools (valgrind, gdb, CLion/XCode/VScode) similar? How are they different? How can they be used to complement each other?
    - Differences

        - Valgrind
            - focuses on detecting memory related errors
            - doesn’t have step by step code execution
        - GDB
            - command line based
            - precise control over code execution
            - Variable inspection
        - GUI debuggers
            - IDE integrated
            - visual interface for debugging
            - Better for larger programs
    - similarities
        - all allow for detecting segmentation faults and memory leaks
        - all allow for tracking down bugs