Dustin Landry dal4235
Gregory Ledet Jr gxl7120

OS Report for Project 1

TASK 1: - dustin

Report

As part of your report, answer the following questions about Task 1:

1. Why is the ability to check input so important?

- Whenever you run a program in the terminal of an operating system often times you need to add arguments or some type of input to the program itself. Whenever that program is executed it must check if the user entered the appropriate input or not for the program to run correctly.

2. Other than simply providing the wrong type of input, what other ways can you think of for bad input to cause an error? Consider situations other than typing input when prompted.

- If the program doesn't correctly check the size of the input, you can cause a buffer overflow which can either lead to a crash of the program or potentially even malicious user-inputted code to be executed.

TASK 2: - dustin

Report

As part of your report on this project, answer the following questions about Task 2:

1. Remove the busy waiting loop used whenever a thread shouts and run the task with 5 shouters and 5 shouts per shouter. Then have each thread yield once after shouting and run another test with the same parameters. Note your results and explain your observations. Undo any changes made to accommodate this question before submitting your assignment.

- When I take out the busy waiting loop and run it with 5 threads and 5 shouts per thread, there is no pseudo-randomness to it. Each thread runs 5 times in a row and outputs the same shout every time. When I add only one yield at the end of each shout each thread takes a turn shouting a pseudo-random phrase each loop.

2. Temporarily disable your input validation, run a minimum of 5 tests with garbage input, and note the results. How would an end user react to this? Undo any changes made to accommodate this question before submitting your assignment.

- After i disabled the validation function and ran the program with bad input, nachos did not crash, however nothing was outputted. The main thread died gracefully after the forked threads could not output any data.

Dustin Landry dal4235
Gregory Ledet Jr gxl7120

TASK 3: - greg
Report
As part of your report, answer the following question about Task 3:
1. What other solutions can you think of to improper input on the command line?

- I could possibly edit the condition statement to have a loop such that if the user enters an improper input instead of it closing the thread it would state that it is an invalid input and ask to enter a correct form of input. It would show the valid input options including an option to close and then let the user enter the character input associated with the functions the thread will fork to. The valid input would stop the loop and continue normally.

TASK 4: - dustin and greg
Summary
You must turn in a report on this assignment along with your code. In addition to the questions listed under each task, the report should answer the following:

1. In your own words, explain how you implemented each task. Did you encounter any bugs? If so, how did you fix them? If you failed to complete any tasks, list them here and briefly explain why.

- For task 1, I split up the task into a few different functions. promptCheck a function that prompts the user for input and returns the input of type char*. identification accepts the user input as a parameter and returns the type of input of type char. finally display accepts the user input and the type as parameters and displays the input by which type it is. Obviously the identification function is the main one in this task, basically I use a few counting variables that goes through each character one-by-one, the function then determines the type by what values each count variable has stored after going through the char string. It should all work.
- For task 2, I again split up the task into more functions. promptShout is a function that prompts the user for two integer values. validation accepts the two values and validates if they are both integer values. Finally createThreads creates the appropriate amounts of threads equal to the amount entered by the user and then forks the threads to a function called ShoutThread that randomly shouts the appropriate number of shouts to each thread. It should all work.
- (Greg)For task 3, I created two global variables, a character for the numeric input and a boolean to be used in threadtest.cc that would state if the input was for the assignment tasks. I copied how the "-rs" if statement was done and edited the format to include the "-A" in an if statement. Back in threadtest.cc I made an if statement for the global boolen variable such that if the variable was true, it would run what was in the statement. In the statement itself, the global character variable is used in an if statement to determine which function the current thread will fork to. If the character variable doesn't match the available options, it will go through the "if" options till it reaches the "else" statement, of which it will prompt the "Invalid input" statement.

Dustin Landry dal4235
Gregory Ledet Jr gxl7120

2. What did you learn from working on this assignment?
    - I always knew a little about threads through Java programming, but know i have a more solid grasp of how to use them. This assignment also taught me the importance of input validation and helped me dust off the more lower level programming concepts like dealing with pointers that I felt rusty on after I found the wonderful world of Java.
    - (Greg)I knew a bit about threads from a previous class, but I mostly worked on creating the functions for the threads to fork to. Being able to work in the system.cc let me understand a little about how the OS will act based on the input.

3. What sort of data structures and algorithms did you use for each task?
    - My own greedy algorithms mixed with a few very basic data-structures like pointers to character arrays. I also used a thread array to create and fork multiple threads based on the users input.