Patrick Charles clid : prc9219

Ryan Mazerole clid : ram3274

NachOS Project Stage 1

CMPS 455 - SP 16

Dr Kumar

**Task 1 Input Identification**

Why is the ability to check input so important?

The wrong type of input can cause compiler errors and crash the program. A program is coded to accept a specific type of input to give you the desired output

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.

**Task 2: Shouting Threads**

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.

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

**Task 3: Command Line**

What other solutions can you think of to improper input on the command line?

**Task 4: Report**

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.

Task 1:

The first step was to create a function and initialize Boolean flags for each type and a character counter, then prompt user for data and store in a char array. Then loop through each value of the array and flag the types as T or F depending on the character using if statements inside the loop. Next, we implemented a truth table using if statements to determine what type of data was input and output the result.

Task 2:

Create a Task 2 function for the input and shout function. Prompt user for number of threads and shouts and perform input validation for each after scanning data in using while loops and the getchar() function to check that it is an integer, if not, it prompts the user for valid input until received.

Then using a for loop set to the # of threads, we create a new thread each time and fork it to the shout function using “ t->Fork(Shout, i ) “ where I is the loop index and thread number.

The shout function uses an char array initialized with random shouts and a double for loops. The first loop loops through however many times the function is told to shout and generates a random number between 2-5 and prints the name of the shouter (thread) and it’s random shout. Then the second for loop uses the random number to yield 2-5 times.

What did you learn from working on this assignment?

What sort of data structures and algorithms did you use for each task?

For loops, while loops, pointers, arrays, functions