3410 Final Project Part 2
Group 15
Names: Shaina Stein, Victoria Hennemann, Zuxuan Chen

---

host.c Methods

main()

Main method is responsible for constructing the histograms by calling the construct_hist method, as well as constructing the records using the construct_record method. Next, we create pipes for communication between the parent and child processes. One pipe is called the signal_pipe, and is used by main_loop_cli to get data from main_loop_data. The other is called the data_pipe. After setting up the pipes, we use the fork method to create parent and child processes. The child calls the main_loop_cli method while the parent calls the main_loop_data method. Lastly, all of the file descriptors are closed.

main_loop_cli()

This method takes in a pointer to the record file and the histograms as arguments. Since we are only sending things through the signal pipe, and reading from the data pipe, we must close the read end of signal_pipe and the write end of data_pipe. Next, a buffer is created to hold the user input. We use the matches function and strcmp function to compare what the user typed to the existing commands such as “resume”, “pause”, “record”, and so on. If what they typed matches exactly to one of the existing commands, we write the command to the signal pipe using the write command. If the command is “pause”, it pauses the program, and “resume” will resume the program. If the command is “request”, it will print out the data packet of the last read bit of data. The exit command will exit out of the program, and the record command should print out all of the current records so far. The different “hist” commands will print out the corresponding histogram using the print_hist functions. Lastly, the help command will print out the different possible commands.

read_cmd()

This method reads the command from the buffer and stores the corresponding number into variable *cmd. (request is 2, pause is 3, resume is 4, and exit is 5.) Inside this method, there is an infinite loop. It will keep reading the command as long as the program is running. We can also reset the buffer when we find that the cmd is not 0 before we write anything into the buffer to avoid chaos.

matches(): This is a string compare method that is used throughout our code to see if two strings are the same, in our case, the user input and one of the commands.

main_loop_data(): This method reads the signals and sends them to the arduino. To do this we first close the correct ends of the pipes. Then, we need to validate the user's command. From here we analyze the user's input. If the user inputs "exit" then the method will return. If the user inputs "pause" then we mark the is_paused variable to true by assigning it a 1. Then we store the message in the to_send variable. We call the send function and then sleep. For resume we change the is_paused variable to 0 and record the message in the to_send variable. Then we send this to the arduino. The rest of the method builds logic for the pause. In this method we also read and accumulate a 16 byte reply. Based on this we update the record or the hist. If the client is request we write the reply to the data type.