By the format of the program execution syntax, three sections [OPTION], [BUFFER\_SIZE] and [INPUT\_FILE] are indicated. To properly start the program execution, these three sections must be validated first. [OPTION] section is optional. The first argument argv[1] is not necessary for [OPTION] input, but [OPTION] will always in argv[1] if specific. Since the syntax of [OPTION] starts with the character ‘-’, checking argv[1][0] can determine if any [OPTION] are specified. Loop over to check all characters within argv[1]. To get the correct size to loop over argv[1], a function lenOf() is created to get argv[1]’s length. An integer variable with default value 0 is created for each [OPTION]. If any of the OPTION ‘nwcI’ is found specified in the loop, cover their representation variable value to 1. In the case of no ‘nwcI’ are specificized, if all option variables’ value is 0 then change all to 1 as default use.

To ensure the position of BUFFER\_SIZE on the input command line, int variable ‘optc’ is used to tell if [OPTION] is specific or not. Argument argv[optc+1] get in two scenarios, 1)OPTION specific: BUFFER\_SIZE will be argv[2]. 2)No OPTION: will be argv[1].

[INPUT\_FILE] is also an optional argument. It optimal property combined with [OPTION] lead to 4 different scenarios. Variable ‘optc’ is used to help get the correct start position of [INPUT\_FILE]. ‘optc+2’ will indicate the first input file if it is on the command line. But using ‘optc+2’ as start point for the loop will never loop if argc=optc+2, which is the case that no [INPUT\_FILE] specifies. So, an end integer variable introduced as the end loop point. In the case argc=optc+2, end will set to argc+1 to ensure to loop at least once in the program. Since only .exe and [BUFFER\_SIZE] is required, the condition ‘argc-optc==2’ is always true for the no [INPUT\_FILE] scenarios, and “prj1inp.txt” will default as input file when this condition met.

Fork() is designed to invoke outside the main process loop. But, when the program processing POSIX message queue. Found out that the parent process needs to call the wait() function to wait child process done writing message to the queue. Since wait() is waiting for a child to return to resume, child must return after finishing writing. If there are multiple input files, then there will be no child process to process information for parents except the first read file. To solve this situation, simply change to invoked fork() once in every loop and return the child at end of writing, that keeping always one child exist at once time.