Shell implementation

Implement a basic shell. Your shell implementation should not execute the existing shell program. You have to build these functionalities by yourself (using fork, exec, wait, etc.). By default, the shell program waits for user input from the stdin. After the user enters some command, the shell program parses the input to interpret I/O redirection, pipe, etc. After, interpreting the command as described below the shell program again waits for user input until the user enters the exit command. Below is the list of features you need to implement.

Syntax	Meaning
command	execute the command and wait for the command to finish, print error message if the command is invalid
command > filename	redirect stdout to file “filename”. If the file does not exist create one, otherwise, overwrite the existing file
command >> filename	If the filename already exists append the stdout output, otherwise, create a new file
1>filename	redirect stdout to filename
2>filename	redirect stderr to filename
2>&1	redirect stderr to stdout
command < filename	use file descriptor 0 (stdin) for filename. If command tries to read from stdin, effectively it will read from filename.
|	pipe command (as discussed in class)
exit	exit from the shell program

On ctrl+c (SIGINT), your program should kill the currently executing process (if any), and start waiting for an input command.

Your program should be able to handle the nested commands, e.g.,

“cat /bin/ls | sort | uniq | wc -l 2>&1 1>output.txt”

Implement everything in a single “.c” file, and submit it on the backpack. Follow the naming convention as “group_id.c”.