Report for OS Assignment 1:

Question 2

The primary shell process operates by displaying the current working directory. Internal commands "word" and "exit" have been incorporated. External commands are supported if located in "/usr/bin" or "/usr/local/bin".

External commands "date" and "dir" have been implemented. This was achieved without utilizing "execvp()," which would necessitate moving executables to "/usr/local/bin" and demanding administrative privileges. Instead, a separate "bin" folder was created, containing compiled "date" and "dir" C files. The shell process employs `execl()` to run the executables directly from the "bin" directory, sidestepping the need for elevated permissions. Both "dir" and "date" programs run as child processes forked from the parent process.

Error handling is effective, employing `perror()` for function-related errors and `fputs()` to relay other errors to `stderr`.

"Word":
The "word" file resides in "src," while the header file is in the "include" folder. The core function, "wCountFunc()," primarily tallies words in a file. It identifies an alphanumeric character as the start and a non-alphanumeric character as the end, classifying this as one word in our approach. The "-n" option ignores newline characters and continues the loop. The "-d" option employs "wCountFunc" on both files and displays differences.

"Dir":
The "dir" program simply creates a directory via "mkdir()" and indicates an error if it already exists. With the "-r" option, the program monitors the exit status of "mkdir()." If it's -1, the directory is removed using "rmdir()" and then recreated.

The "-v" option combines default functionality with step-by-step execution through a single function, thanks to a boolean variable.

The parent process uses `waitpid()` to await the child process and store its exit status. If normal, the parent process changes its directory according to the input command-line argument's directory using chdir().

"Date":
This was more complex, involving the use of structs from `time.h`. The `status` function returns a timestamp. The `gmtime()` and `localtime()` functions convert the long integer to struct fields, returning a pointer.

The RFC 5322 string format is achieved through the `strftime()` function, returning the struct according to a custom format provided by "%a, %d %b %Y %H:%M:%S +0000\n"

Around 10 commands were implemented, including "today," "yesterday," "tomorrow," "in_a_minute," "in_an_hour," "in_a_day," "in_a_month," "in_a_year," "a_minute_ago," "an_hour_ago," "a_day_ago," "a_month_ago," and "a_year_ago." These were accomplished by manipulating the time struct. The default time format is "Day, DD MM YYYY HH:MM:SS IST."