ZIP FOLDER CONTAINS:

1. shell.c
2. README_15CS01022.pdf
3. io.c - simple code to test execution in the terminal and for file I/O redirection.
4. bg.c - simple code to run the process in the background.
5. input.txt -file for file I/O redirection

COMPILATION: gcc shell.c -o shell -lreadline

EXECUTION: ./shell

LIST OF COMMANDS SUPPORTED:

message- message for you
help - Displays the commands supported by the shell.
jobs- Lists the jobs running in the background.
cd- Changes current directory
If only cd is given then it goes to home directory.
history- Displays the history of commands used.
clrhistory- Clears the history of commands used.
clear- Clears the terminal
exit- Exits from the shell
kill % jobid- Kills the job using its job id
ls or /bin/ls - Lists the files in the current working directory (Both relative
or absolute pathnames)"

INSTRUCTION:

Give proper syntax for the commands to execute.

cd:

• **Command: cd**
• Changes current directory
• If only cd is given then it goes to home directory.

exit:

• **Command: exit**
• Exits from the shell
• If there are jobs (background processes running) then you will not be allowed to exit unless you kill all the running jobs for which you can use the killAll command.

Clear:

• **Command: clear**

• Clears the terminal

Help:

• **Command: help**

• Displays the commands supported by the shell.

Run as background process using '&':

• Use bg.c only

• Use: bg.c code

• Compile: gcc bg.c -o bg

• **Syntax: Command &**

• example: ./bg & (./code space &)

• Placing & at the end of the command runs the corresponding process as background process.

• You can check if the process is running as background process or not by using jobs command.

• You can also use top command and check using the corresponding pid of the process.

Input Output file redirection:

• Use: io.c code

• Compile: gcc io.c -o io

• **Syntax: Command < infile > outfile**

• example: ./io < input.txt > output.txt

• (./io space < space input.txt space > space 2.txt)

• Here the input.txt is attached.The output.txt will be automatically created when the above command is executed.

• Please refresh or reopen the directory to check the created output.txt.

• The input to the code (io.c) is taken through the file input.txt and the all the output which it will print in the terminal is printed only in the output.txt but not in the terminal.

Jobs:

• **Command: jobs**

• First run some processes in the background by using the command ./code &.

• Then type command jobs

• You can see the job id, status (Running or Done) and pid.

Kill:

• **Command: kill % jobid (kill space % space jobid)**

• ****You can kill the jobs only by using job ids and not pids.**

• You can kill any running job using valid job id.
• After killing the job, you can check if the job is killed or not by using
the jobs command where you can see the job status i.e running or done.
• You can also check by using top command to check if the specific job is killed or not by using its corresponding pid.

KillAll:

• **Command: killAll**
• Kills all the running jobs at once
• If you are trying to exit from the shell but if the backgound processes
(jobs) are still running,then you will be notified that you can not exit with out killing all the running jobs.
• You can use the killAll command to kill all the running jobs at once or else you can kill each job by using kill % jobid command.

History:

• **Command: history**
• To display the history of commands used.
• **Command: ! Num (! space num)**
• Num is the serial number of the commands in the history.
• It displays the history command corresponding to the serial number.
• **Command: !- Num (!- space num)**
• Num is the serial number which displays the command which is recently used in the history.

Clear History:

• **Command: clrhistory**
• To clear the history of commands used.