James Peacemaker

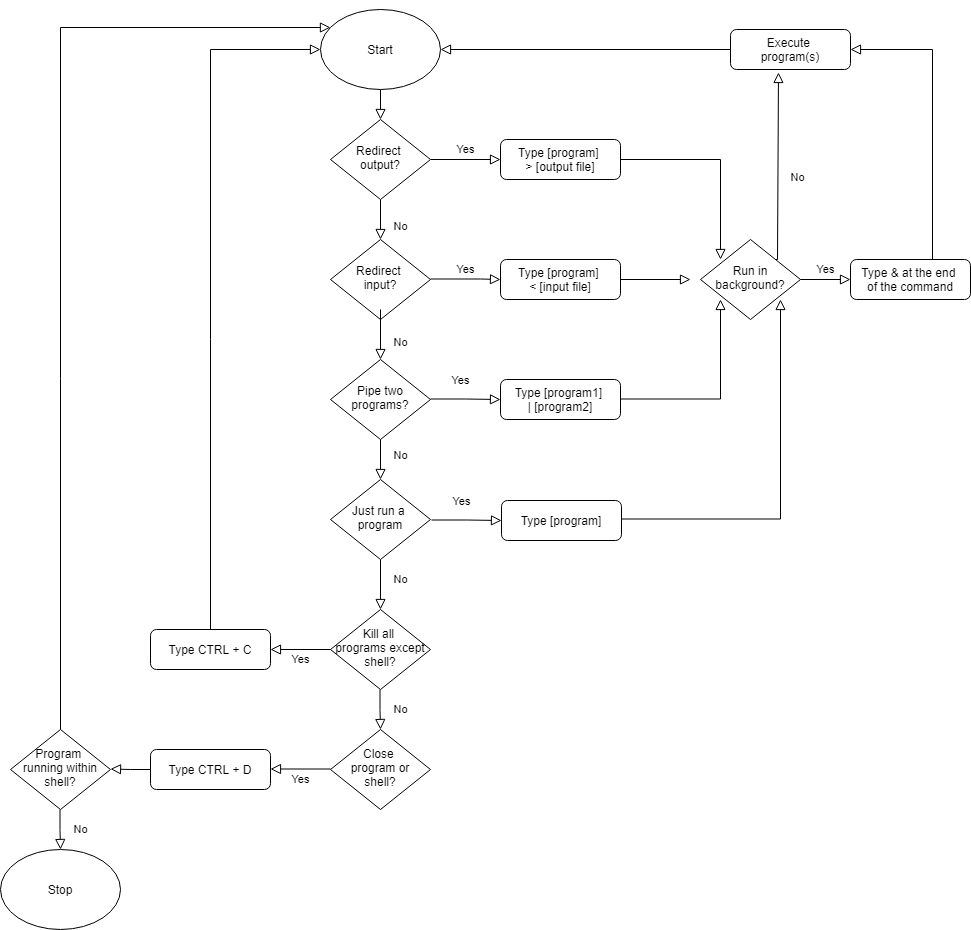
March 31, 2021

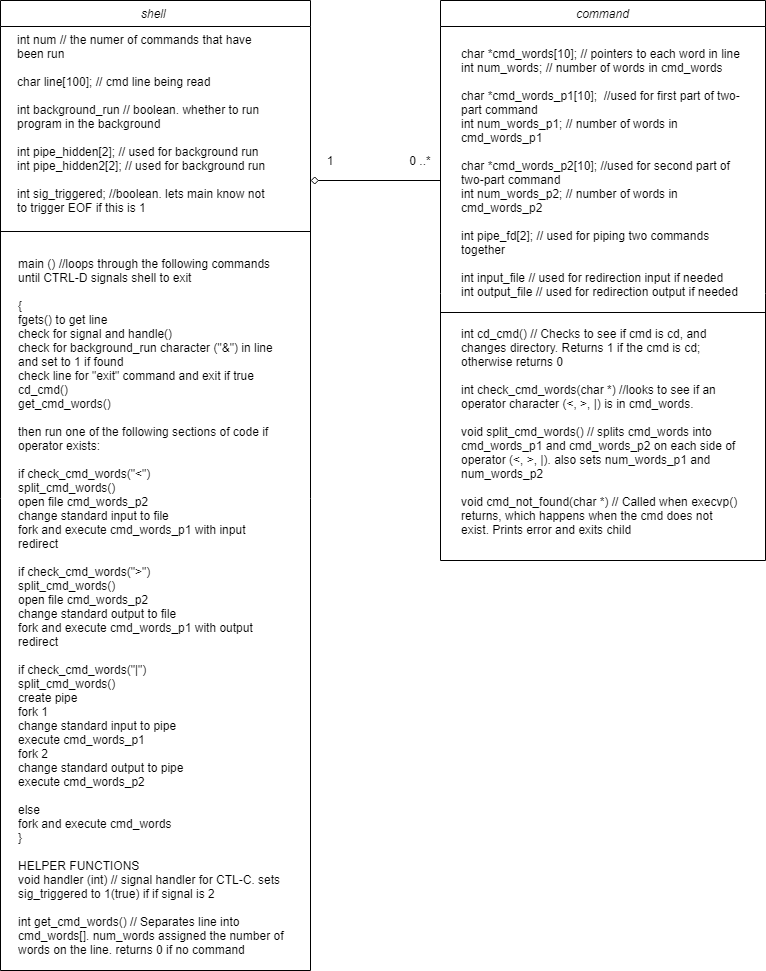
CPSC 405

Shell Project – Answer Document

Design

* Place the design of your implementation in this item. You must create a design artifact that shows the design of your shell. This should be two levels. A higher-level design shows the overall design of your shell - something that people can understand your design without getting into the details. The high level shows how users interact with your shell. A lower level design shows algorithmic flow and the underlying data structures. You can Search the Internet for design artifacts. In CPSC 240, you used UML to create items such as class diagrams and structure charts.



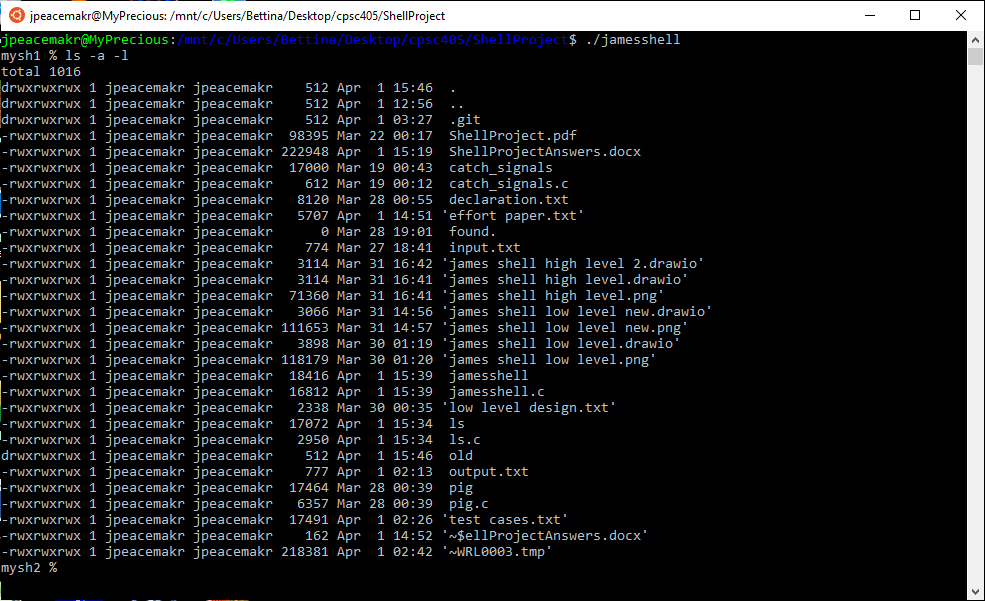


Features Completed

For each of the following features, indicate which you have designed and implemented in your shell. The gray boxes can be X-ed in by double clicking on it. Alternatively, just write yes or no. If you indicate yes for completing a feature, place a screen shot after the bullet showing your tests that demonstrate the feature.

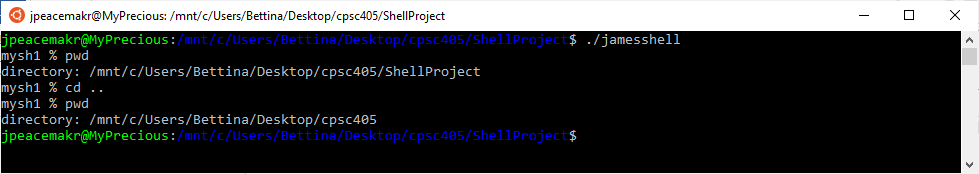
* Basic reading of a command line and breaking into its parts.

The starter shell gets a line of input and breaks it up into command words that can be used to fork and execute. It retains the options flags when it sends the command words to exec().

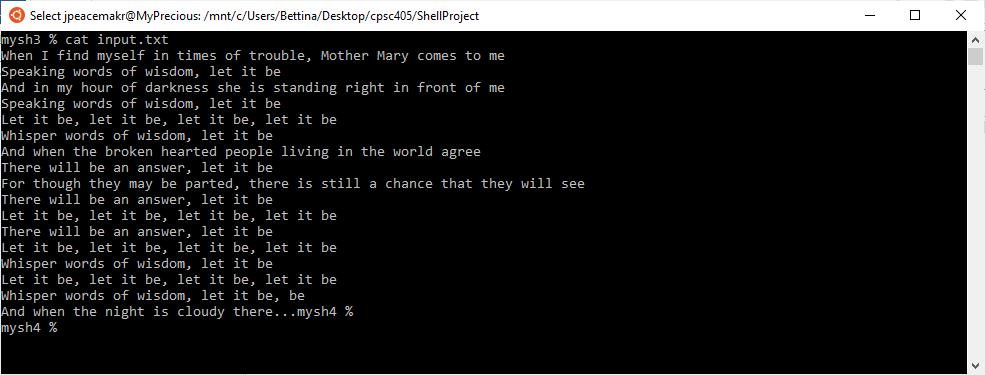


* cd and pwd commands

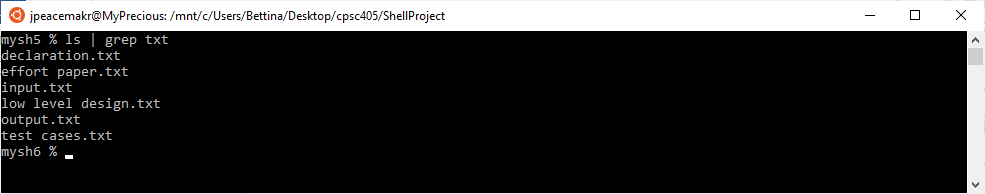
Cd reads the first three characters of line to see if they match “cd ”. It then runs chdir() on the remaining characters in line. If it can’t change to that directory, it gives an error message.



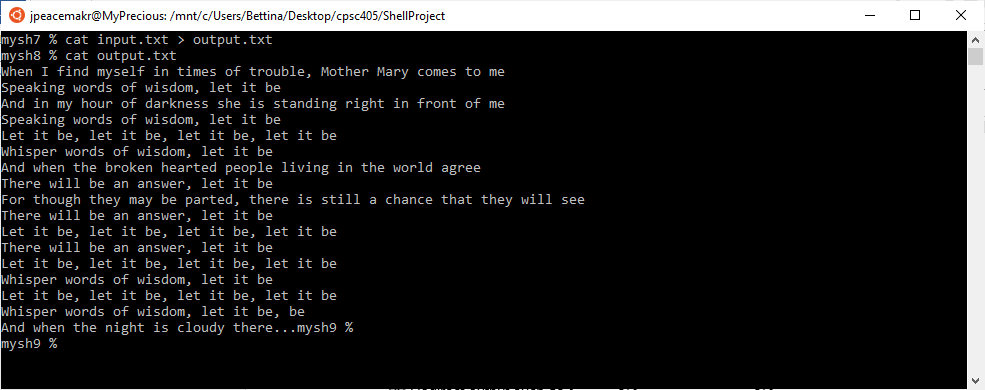
* fork() and exec() of command that are programs such as % cat file.txt



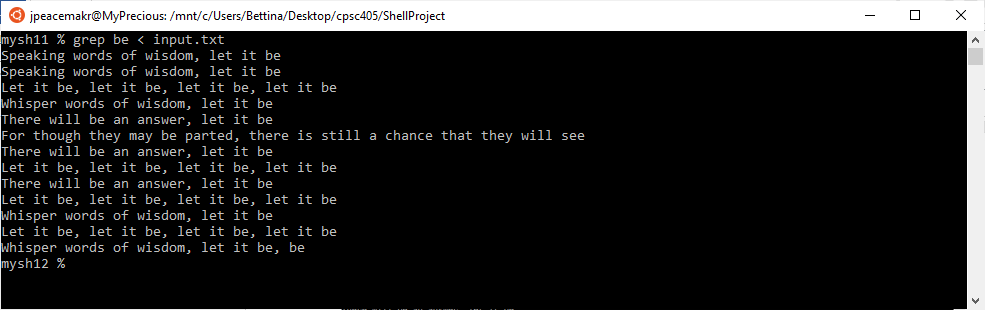
* Connect two commands with a pipe such as % cat file.txt | grep string



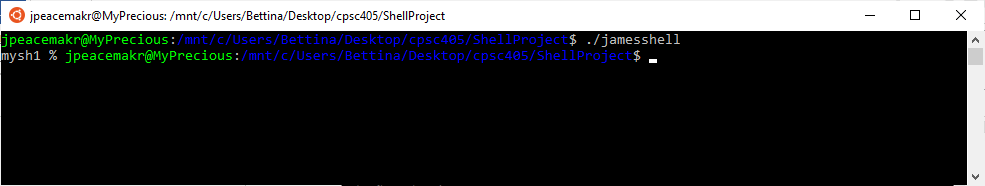
* Redirect output such as % cat file.txt > outputfile.txt



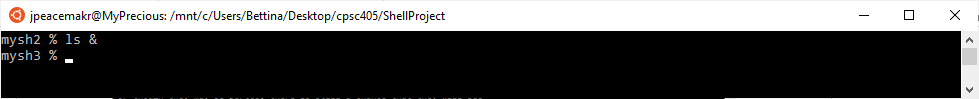
* Redirect input such as % grep string < inputfile.txt



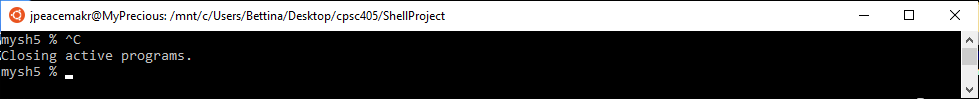
* Processes Control-D to exit the shell.



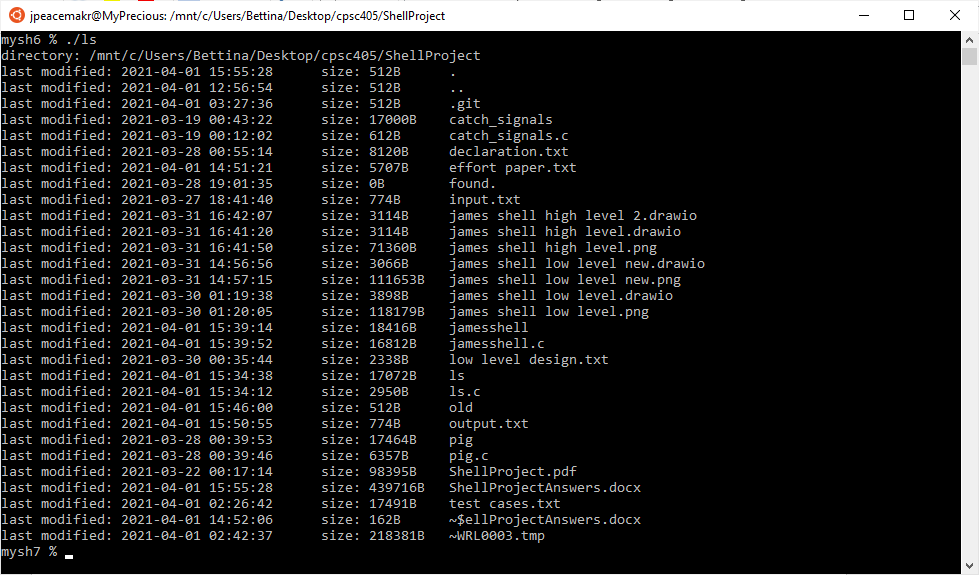
* Ability to run a program in the background such as % ls &



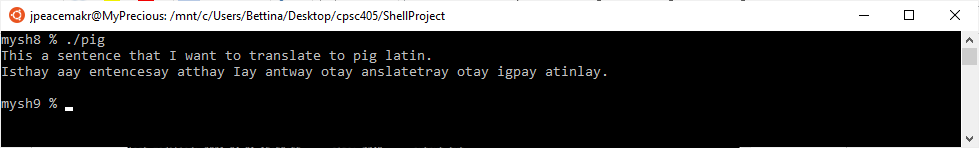
* Processes Control-C such that the shell does not terminate.



* Implemented your own version of the ls program.



* Implemented a second Linux program such as cat.



* Gusty, the code for my shell is so cool that I want you to read it.

Test Cases

* Place your test cases here. Your collection of test cases should demonstrate you have tested the features you completed. Your test cases should be a copy/paste of you applying your test cases to your program. If the test case is not obvious, you should annotate the copy/paste with descriptions.

I included a test cases.txt file in the folder too that has this same information that might be easier to read than in Word due to the narrow formatting.

Basic commands

This tests basic commands cd, pwd, ls.

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$ ./jamesshell

mysh1 % pwd

directory: /mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject

mysh2 % ls

ShellProject.pdf 'james shell low level new.drawio' lsbackgroundtest.txt pigold.c

ShellProjectAnswers.docx 'james shell low level new.png' lsoutput.txt pigold2.c

basichidden.txt 'james shell low level.drawio' output.txt pipe\_and\_exec.c

declaration.txt 'james shell low level.png' output2.txt redirect\_stdout.c

'effort paper.txt' jamesshell outputnew.txt sh.c

found. jamesshell.c outputnew2.txt simple\_pipe.c

input.txt jamesshellv1.c parent\_child\_pipe.c skipwhitespace.c

'james shell high level 2.drawio' 'low level design.txt' pig starter\_sh.c

'james shell high level.drawio' ls pig.c 'test cases.txt'

'james shell high level.png' ls.c pigbad.c

mysh3 % cd ..

mysh3 % ls

'C notes.txt' 'ProcessAPILab to turn in' doit lecture3-Scheduling.pdf

CPSC405SyllabusSpring21.pdf ScheduleProject doit.c lecture4-VirtualMem.pdf

'FirstLab to submit' 'ScheduleProject to turn in' doitwithdebug.c lecture5-Paging.pdf

'FirstLab to submit.zip' ShellProject 'exec types in c.txt' lecture6-TLB.pdf

FirstLab.pdf SignalLab gerbil lectureLocks.pptx

'FirstLabAnswers template.docx' 'Zoom test prep.docx' gerbil.c lectureThreads.pptx

Labs\_export ZoomPoints.xlsx gerbil2 linkedlist.c

Labs\_export.zip a.out gerbil2.c 'notes concurrency.txt'

Lecture1-Intro.pdf 'assembly programming notes.txt' gerbil3 pipeexample.c

Lecture2-VirtualCPU.pdf 'class info.txt' gerbil3.c program\_sections.c

MainArgsLab class1 gerbilelipsis q0\_demo.c

'MainArgsLab to turn in' class1.zip gerbilelipsis.c variable\_attributes.c

'MainArgsLab to turn in.zip' 'compiling with gcc.txt' hello 'zoom point questions.txt'

MemoryLab 'crib notes.txt' hello.c

ProcessAPILab 'dictionary example' lecture2.c

mysh4 % cd ShellProject

mysh4 % ls

ShellProject.pdf 'james shell low level new.drawio' lsbackgroundtest.txt pigold.c

ShellProjectAnswers.docx 'james shell low level new.png' lsoutput.txt pigold2.c

basichidden.txt 'james shell low level.drawio' output.txt pipe\_and\_exec.c

declaration.txt 'james shell low level.png' output2.txt redirect\_stdout.c

'effort paper.txt' jamesshell outputnew.txt sh.c

found. jamesshell.c outputnew2.txt simple\_pipe.c

input.txt jamesshellv1.c parent\_child\_pipe.c skipwhitespace.c

'james shell high level 2.drawio' 'low level design.txt' pig starter\_sh.c

'james shell high level.drawio' ls pig.c 'test cases.txt'

'james shell high level.png' ls.c pigbad.c

mysh5 %

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$

Test the options flags on a program:

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$ ./jamesshell

mysh1 % ls -l

total 820

-rwxrwxrwx 1 jpeacemakr jpeacemakr 98395 Mar 22 00:17 ShellProject.pdf

-rwxrwxrwx 1 jpeacemakr jpeacemakr 204466 Mar 31 16:43 ShellProjectAnswers.docx

-rwxrwxrwx 1 jpeacemakr jpeacemakr 1112 Mar 28 19:55 basichidden.txt

-rwxrwxrwx 1 jpeacemakr jpeacemakr 8120 Mar 28 00:55 declaration.txt

-rwxrwxrwx 1 jpeacemakr jpeacemakr 1874 Mar 31 16:16 'effort paper.txt'

-rwxrwxrwx 1 jpeacemakr jpeacemakr 0 Mar 28 19:01 found.

-rwxrwxrwx 1 jpeacemakr jpeacemakr 774 Mar 27 18:41 input.txt

-rwxrwxrwx 1 jpeacemakr jpeacemakr 3114 Mar 31 16:42 'james shell high level 2.drawio'

-rwxrwxrwx 1 jpeacemakr jpeacemakr 3114 Mar 31 16:41 'james shell high level.drawio'

-rwxrwxrwx 1 jpeacemakr jpeacemakr 71360 Mar 31 16:41 'james shell high level.png'

-rwxrwxrwx 1 jpeacemakr jpeacemakr 3066 Mar 31 14:56 'james shell low level new.drawio'

-rwxrwxrwx 1 jpeacemakr jpeacemakr 111653 Mar 31 14:57 'james shell low level new.png'

-rwxrwxrwx 1 jpeacemakr jpeacemakr 3898 Mar 30 01:19 'james shell low level.drawio'

-rwxrwxrwx 1 jpeacemakr jpeacemakr 118179 Mar 30 01:20 'james shell low level.png'

-rwxrwxrwx 1 jpeacemakr jpeacemakr 18352 Mar 31 20:00 jamesshell

-rwxrwxrwx 1 jpeacemakr jpeacemakr 15404 Mar 31 20:01 jamesshell.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 10922 Mar 24 03:51 jamesshellv1.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 2338 Mar 30 00:35 'low level design.txt'

-rwxrwxrwx 1 jpeacemakr jpeacemakr 17072 Mar 26 15:45 ls

-rwxrwxrwx 1 jpeacemakr jpeacemakr 2950 Mar 26 15:47 ls.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 346 Mar 28 19:01 lsbackgroundtest.txt

-rwxrwxrwx 1 jpeacemakr jpeacemakr 382 Mar 28 20:12 lsoutput.txt

-rwxrwxrwx 1 jpeacemakr jpeacemakr 1112 Mar 28 00:40 output.txt

-rwxrwxrwx 1 jpeacemakr jpeacemakr 10780 Mar 28 00:55 output2.txt

-rwxrwxrwx 1 jpeacemakr jpeacemakr 1112 Mar 28 18:44 outputnew.txt

-rwxrwxrwx 1 jpeacemakr jpeacemakr 1112 Mar 28 18:45 outputnew2.txt

-rwxrwxrwx 1 jpeacemakr jpeacemakr 2075 Mar 22 00:17 parent\_child\_pipe.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 17464 Mar 28 00:39 pig

-rwxrwxrwx 1 jpeacemakr jpeacemakr 6357 Mar 28 00:39 pig.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 6493 Mar 27 22:07 pigbad.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 3518 Mar 27 15:26 pigold.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 6475 Mar 27 21:32 pigold2.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 1550 Mar 23 22:50 pipe\_and\_exec.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 965 Mar 22 00:17 redirect\_stdout.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 9526 Mar 22 00:17 sh.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 566 Mar 22 00:17 simple\_pipe.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 837 Mar 22 00:17 skipwhitespace.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 3415 Mar 22 00:51 starter\_sh.c

-rwxrwxrwx 1 jpeacemakr jpeacemakr 313 Mar 31 16:29 'test cases.txt'

mysh2 %

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$

Test my own list (./ls) program. It shows the current directory and lists the time and state modified, file size and name.

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$ ./jamesshell

mysh1 % ./ls

directory: /mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject

last modified: 2021-03-31 20:00:23 size: 512B .

last modified: 2021-03-31 16:21:40 size: 512B ..

last modified: 2021-03-31 16:46:54 size: 512B .git

last modified: 2021-03-28 19:55:42 size: 1112B basichidden.txt

last modified: 2021-03-28 00:55:14 size: 8120B declaration.txt

last modified: 2021-03-31 16:16:12 size: 1874B effort paper.txt

last modified: 2021-03-28 19:01:35 size: 0B found.

last modified: 2021-03-27 18:41:40 size: 774B input.txt

last modified: 2021-03-31 16:42:07 size: 3114B james shell high level 2.drawio

last modified: 2021-03-31 16:41:20 size: 3114B james shell high level.drawio

last modified: 2021-03-31 16:41:50 size: 71360B james shell high level.png

last modified: 2021-03-31 14:56:56 size: 3066B james shell low level new.drawio

last modified: 2021-03-31 14:57:15 size: 111653B james shell low level new.png

last modified: 2021-03-30 01:19:38 size: 3898B james shell low level.drawio

last modified: 2021-03-30 01:20:05 size: 118179B james shell low level.png

last modified: 2021-03-31 20:00:23 size: 18352B jamesshell

last modified: 2021-03-31 20:01:26 size: 15404B jamesshell.c

last modified: 2021-03-24 03:51:39 size: 10922B jamesshellv1.c

last modified: 2021-03-30 00:35:44 size: 2338B low level design.txt

last modified: 2021-03-26 15:45:14 size: 17072B ls

last modified: 2021-03-26 15:47:26 size: 2950B ls.c

last modified: 2021-03-28 19:01:35 size: 346B lsbackgroundtest.txt

last modified: 2021-03-28 20:12:15 size: 382B lsoutput.txt

last modified: 2021-03-28 00:40:46 size: 1112B output.txt

last modified: 2021-03-28 00:55:40 size: 10780B output2.txt

last modified: 2021-03-28 18:44:57 size: 1112B outputnew.txt

last modified: 2021-03-28 18:45:40 size: 1112B outputnew2.txt

last modified: 2021-03-22 00:17:13 size: 2075B parent\_child\_pipe.c

last modified: 2021-03-28 00:39:53 size: 17464B pig

last modified: 2021-03-28 00:39:46 size: 6357B pig.c

last modified: 2021-03-27 22:07:21 size: 6493B pigbad.c

last modified: 2021-03-27 15:26:49 size: 3518B pigold.c

last modified: 2021-03-27 21:32:30 size: 6475B pigold2.c

last modified: 2021-03-23 22:50:52 size: 1550B pipe\_and\_exec.c

last modified: 2021-03-22 00:17:14 size: 965B redirect\_stdout.c

last modified: 2021-03-22 00:17:14 size: 9526B sh.c

last modified: 2021-03-22 00:17:14 size: 98395B ShellProject.pdf

last modified: 2021-03-31 16:43:35 size: 204466B ShellProjectAnswers.docx

last modified: 2021-03-22 00:17:14 size: 566B simple\_pipe.c

last modified: 2021-03-22 00:17:14 size: 837B skipwhitespace.c

last modified: 2021-03-22 00:51:09 size: 3415B starter\_sh.c

last modified: 2021-03-31 20:07:11 size: 7498B test cases.txt

mysh2 %

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$

Testing my pig (./pig) program. Rather than cat or dog, I made pig, which translates text into pig latin. It is shown here translating two sentences from standard input then I use input redirection to translate a file containing lyrics from one of my favorite songs.

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$ ./jamesshell

mysh1 % ./pig

This is a sentence that I want to translate.

Isthay isay aay entencesay atthay Iay antway otay anslatetray.

It also handles "punctuation" and (Capitalization).

Itay alsoay andleshay "unctuationpay" anday (Apitalizationcay).

^C

Closing active programs.

mysh2 % ./pig < input.txt

Enwhay Iay indfay elfmysay inay imestay ofay oubletray, Othermay Arymay omescay otay emay

Eakingspay ordsway ofay isdomway, etlay itay ebay

Anday inay ymay ourhay ofay arknessday eshay isay andingstay ightray inay ontfray ofay emay

Eakingspay ordsway ofay isdomway, etlay itay ebay

Etlay itay ebay, etlay itay ebay, etlay itay ebay, etlay itay ebay

Isperwhay ordsway ofay isdomway, etlay itay ebay

Anday enwhay ethay okenbray eartedhay eoplepay ivinglay inay ethay orldway agreeay

Erethay illway ebay anay answeray, etlay itay ebay

Orfay oughthay eythay aymay ebay artedpay, erethay isay illstay aay ancechay atthay eythay illway eesay

Erethay illway ebay anay answeray, etlay itay ebay

Etlay itay ebay, etlay itay ebay, etlay itay ebay, etlay itay ebay

Erethay illway ebay anay answeray, etlay itay ebay

Etlay itay ebay, etlay itay ebay, etlay itay ebay, etlay itay ebay

Isperwhay ordsway ofay isdomway, etlay itay ebay

Etlay itay ebay, etlay itay ebay, etlay itay ebay, etlay itay ebay

Isperwhay ordsway ofay isdomway, etlay itay ebay, ebay

Anday enwhay ethay ightnay isay oudyclay erethay...

mysh3 %

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$

Testing the redirect output operator >.

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$ ./jamesshell

mysh1 % ls > output.txt

mysh2 % cat output.txt

ShellProject.pdf

ShellProjectAnswers.docx

basichidden.txt

declaration.txt

effort paper.txt

found.

input.txt

james shell high level 2.drawio

james shell high level.drawio

james shell high level.png

james shell low level new.drawio

james shell low level new.png

james shell low level.drawio

james shell low level.png

jamesshell

jamesshell.c

jamesshellv1.c

low level design.txt

ls

ls.c

lsbackgroundtest.txt

lsoutput.txt

output.txt

output2.txt

outputnew.txt

outputnew2.txt

parent\_child\_pipe.c

pig

pig.c

pigbad.c

pigold.c

pigold2.c

pipe\_and\_exec.c

redirect\_stdout.c

sh.c

simple\_pipe.c

skipwhitespace.c

starter\_sh.c

test cases.txt

mysh3 %

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$

Testing the redirect input operator <. I use an output.txt file from ls > output.txt.

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$ ./jamesshell

mysh1 % grep txt < output.txt

basichidden.txt

declaration.txt

effort paper.txt

input.txt

killout.txt

low level design.txt

lsbackgroundtest.txt

lsoutput.txt

output.txt

output2.txt

outputnew.txt

outputnew2.txt

signalout.txt

siout.txt

test cases.txt

mysh2 %

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$

Testing the ability to pipe two programs together. The output of ls is used as the input for grep txt:

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$ ./jamesshell

mysh1 % ls | grep txt

basichidden.txt

declaration.txt

effort paper.txt

input.txt

low level design.txt

lsbackgroundtest.txt

lsoutput.txt

output.txt

output2.txt

outputnew.txt

outputnew2.txt

test cases.txt

mysh2 %

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$

Testing the ability to run a program in the background. Here I test catch\_signals. It takes 4 CTRL C signals to kill it but does not show output. The ps program is used to show process that are running.

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$ ./jamesshell

mysh1 % ./catch\_signals &

mysh2 % ps

PID TTY TIME CMD

7 tty1 00:00:01 bash

680 tty1 00:00:00 jamesshell

681 tty1 00:00:01 catch\_signals

682 tty1 00:00:00 ps

mysh3 % ^C

Closing active programs.

mysh3 % ^C

Closing active programs.

mysh3 % ps

PID TTY TIME CMD

7 tty1 00:00:01 bash

680 tty1 00:00:00 jamesshell

681 tty1 00:00:07 catch\_signals

683 tty1 00:00:00 ps

mysh4 % ^C

Closing active programs.

mysh4 % ^C

Closing active programs.

mysh4 % ps

PID TTY TIME CMD

7 tty1 00:00:01 bash

680 tty1 00:00:00 jamesshell

681 tty1 00:00:11 catch\_signals <defunct>

684 tty1 00:00:00 ps

mysh5 % jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$

Running in the background with redirected output:

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$ ./jamesshell

mysh1 % ls > output.txt &

mysh2 % cat output.txt

FPE\_signal

FPE\_signal.c

ShellProject.pdf

ShellProjectAnswers.docx

basichidden.txt

catch\_signals

catch\_signals.c

declaration.txt

effort paper.txt

found.

input.txt

james shell high level 2.drawio

james shell high level.drawio

james shell high level.png

james shell low level new.drawio

james shell low level new.png

james shell low level.drawio

james shell low level.png

jamesshell

jamesshell.c

jamesshellv1.c

killme\_with\_a\_signal

killme\_with\_a\_signal.c

killout.txt

low level design.txt

ls

ls.c

lsbackgroundtest.txt

lsoutput.txt

my\_kill

my\_kill.c

output.txt

output2.txt

outputnew.txt

outputnew2.txt

parent\_child\_pipe.c

pig

pig.c

pigbad.c

pigold.c

pigold2.c

pipe\_and\_exec.c

redirect\_stdout.c

sh.c

signalout.txt

simple\_pipe.c

siout.txt

skipwhitespace.c

starter\_sh.c

test cases.txt

mysh3 % jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$

Testing CTRL-C and CTRL-D. CTRL C sends signal for program to end. It is caught by the program ./catch\_signals until the fifth time. CTRL-D exits out of the shell.

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$ ./jamesshell

mysh1 % ./catch\_signals

Ok, let's go, kill me (371) if you can!

^CSignal 2 ouch, that hurt!

Closing active programs.

^CSignal 2 ouch, that hurt!

Closing active programs.

mysh2 % ^CSignal 2 ouch, that hurt!

Closing active programs.

mysh2 % ^CSignal 2 ouch, that hurt!

I've had enough!

Closing active programs.

mysh2 % ps

PID TTY TIME CMD

7 tty1 00:00:01 bash

370 tty1 00:00:00 jamesshell

372 tty1 00:00:00 ps

mysh3 %

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/ShellProject$

Effort Paper

* Place your paper here. Write a paper that (1) describes your effort on the project (include a log of days/hours), (2) describes your difficulties on the project, and (3) describes the features completed in your shell.

For me, I think this is one of those projects where 5 percent of the problem takes 50 percent of the effort.

I used the starter shell, which was a big help. The starter shell gets a line of input and breaks it up into command words that can be used to fork and execute. It also sets up basic commands like cd, pwd and exit. It also handles it if no command or a bad command is typed.

The first step for me was to be able to break up the command words into two parts, one part for each side of an operator (<,>,|).

Then I could handle each command differently depending on which operator was in the middle.

For the input operator (<), I changed standard input to a file. The command words on the left side of the operator became the command and the right side became the file name.

For the output operator (>), I changed standard output to a file. The command words on the left side of the operator became the command and the right side became the file name.

For the pipe operator (|), I created a pipe to pass data from one program to another. The command words on the left of the operator became the program that passed output into the pipe, while the command words on the right became the program that pulled input out of the pipe.

The final symbol I had to handle was the &, which made the process run in the background. For this task, I switched over standard input and/or output over to a pair of pipes that were just never visible. I first tried a single pipe but then quickly realized the program would just loop, outputting to one side and then using that as input. I also made it so that when a command runs in the background, the parent (main) will not wait for it to finish.

For the two program, I did a simple version of ls and a program called pig. You said we could do the cat program or come up with our own like “dog.” I thought a program called pig that translated test into pig latin.

The ls program seemed simple enough, but I had to look up the structure that went with the file information.

The pig program turned out to be harder than I expected because I wanted it to be able to handle punctuation, capitalization and numbers. I also had to come up with how to handle words that started with multiple consonants, started with vowels or words that didn’t have a vowel at all, like “why.”

I spent about 40 hours working on this project overall.

3/21: 2 hours. Set up git. Read project description and code. Started pseudocode to do redirection and pipes.

3/22: 3 hours. Got it to split up command words into two sets of command words if it contained a <, > or |. First half of command words executes now. It gives feedback on correct usage if words are missing before or after the <, > or |.

3/23: 5 hours. I got the pipe command | done. I spent several hours trying to figure out why the grep command would not close. The shell was just stuck waiting. I had to make sure both ends of the pipes were closed in each fork and it finally worked.

3/25: 3 hours. I got a basic ls program running with name, file size and last modified time. It took some time to figure out the structures that stored information about each file.

3/26: 4 hours. I got the majority of my pig program done.

3/27: 8 hours. I had to change the pig program from scanf to fget to get it to handle the new lines right and for piping. Fixed a bug where it would loop when no vowels were in a word. Added handling uppercase letters. Spent a lot of time trying to figure out why it would print fine on standard input / output but had problems when using redirection to and from files through linux.

3/28: 5 hours. Spent a lot of time trying to get the shell to not trigger an EOF when doing CTRL-C. Basically CTRL-C was doing what CTRL-D does in addition to the signal. Worked on getting program to run in the background. I think I got it working but need to test more.

3/29: 3 hours on diagrams. Struggled trying to a diagram that I liked. I didn't use objected oriented design on the program and there were not a lot of parts, just a lot of conditions and actions.

3/31: 6 hours. Testing and tweaking. For some reason, the basic wait(NULL) was not working as expected. It took two of them to get the parent to wait. I switched them to waitpid() and they worked fine. I spent most of the time trying to figure out how to have the parent process wait for a child that has been sent a CTRL C signal. Wrote most of the effort paper.

4/1: 2 hours. Finished up the effort paper and added a few things to the answer document. Created a makefile.

I don't think the concepts are that hard, but there are a lot of them intertwined and pitfalls trying to do relatively minor things.

The biggest problems I ran into were the idiosyncrasies of the C language. This was likely due to my lack of formal training with C and there being gaps in my knowledge. A lot of the functions and structures don't seem intuitive to me.

A few of the obstacles that took the most time to figure out:

- I was running into a problem where the program I was trying to execute would keep running. This was because I did not understand the importance of closing both ends of the pipes -- and closing them in the parent and every child that uses them.

- I spent a lot of time trying to get the shell to not trigger an End Of File when doing CTRL-C. Basically CTRL-C was doing what CTRL-D does in addition to the signal.

- I spent a lot of time trying to figure out how to get a parent child to wait for a child to fully exit that had been sent a CTRL-C signal. I still couldn't figure it out. Once the signal was sent and handled by the child, the parent didn't want to wait and they just output on top of each other.

If I had more time I would probably add the ability to do multiple pipes and combine them with input and output redirection.

I would also create a /bin folder to store programs like ls so you wouldn't need the ./ in front of it to run.

Code

* Submit your code and a makefile.