The main function of the shell is an infinite loop that grabs a command from the user using GNU readline and then passing that command to the processPipes function which handles all the execution from that point on. Once processPipes returns the memory allocated by readline is freed and the loop runs again. If a command is passed when mysh is invoked (e.g. `./mysh ls | cat`) the command passed as an argument is processed and then the loop terminates allowing the shell to exit.

The processPipes function splits the command on the ‘|’ char creating an array of commands. This array of sub commands is then lopped over to get the final output. If at any time the current command is `exit` then the shell immediately terminates. Processing of each sub command entails resolving subshells to get the actual command to be executed using the resolveSubshells function and then executing and capturing the output using the exec function. For each sub command after the first the output from the previous command is passed to exec to be used as the input for the next, allowing the chaining of pipes. If exec indicates an error then proper output is generated and the loop breaks so that no further commands in the chain of pipes are executed. Once the last command in the chain is executed the final output is printed to the terminal unless there was a redirection of output on that item.

The resolveSubshells function searches for (‘$(’, ‘)’) pairs and extracts the enclosed command. The preceding substring and remainder substring are saved for later use. The extracted command is execed and the output is captured. The preceding substring, the output from the subshell, and the remainder are glued back together to get the new string. This process is looped until no more subshells are detected in the command. This allows for resolution of multiple subshells but will not work for nested subshells. The resulting command after the loop has exited is returned as the new command.

The exec function first parses any redirections out of the command using the getRedirections function. Once that is done the command is broken on spaces so that we have an array of cstrings. The function is forked.

* In the child:

If previous input was provided or there is an input redirect then a pipe is created. First the previous input (if any) is written to the write end of the pipe then if there is an input redirect the input file is opened and data is read from it and written to the pipe and the read end is duped to stdin. If there is only an input redirect then the file is opened and the fd for that file is duped over stdin. If there is any error redirect then that file is opened and duped over stderr. A second pipe (created before the fork) has its write end duped over stdout so that the parent can see the results. The command is then execed using execvp.

* In the parent:

The parent waits for the child to complete and checks the return status. If it is non-zero then the proper errors are produced. The output from the child is read from the pipe into a buffer. If there was an output redirection then the file is opened and the output is written to the file. The result is still made available to the caller so that the output can be reused if there was a pipe as well as the redirection.

The resolveSubshells function loops while searching for any redirection delimiters. Each loop only the first delimiter is processed and the loop exits when no more redirection delimiters exist in the string. The name of the file redirected to is saved and the delimiter and the file name are removed from the command, overwritten by the remainder that came after it. If any redirection is specified multiple times it will be overwritten and only the last one matters. Once the loop exits the 3 redirections (NULL is used if none was specified) are made available to the caller for use.

**`myls –l`:**

/home/Students/mw1053/Documents/linux/proj2$ myls -l

total 304

-rw------- 1 mw1053 student 336 Apr 21 22:53 Makefile

-rwx------ 1 mw1053 student 12323 Apr 23 14:05 mycat

drwxr-xr-x 2 mw1053 student 4096 Apr 23 14:05 mycat\_src

-rwx------ 1 mw1053 student 18231 Apr 23 14:05 mycp

drwxr-xr-x 2 mw1053 student 4096 Apr 23 14:05 mycp\_src

-rwx------ 1 mw1053 student 14167 Apr 23 14:05 myls

drwxr-xr-x 2 mw1053 student 4096 Apr 23 14:05 myls\_src

-rwx------ 1 mw1053 student 25954 Apr 23 14:29 mysh

drwxr-xr-x 2 mw1053 student 4096 Apr 23 14:40 mysh\_src

-rw------- 1 mw1053 student 169 Apr 10 13:03 README.md

drwx------ 2 mw1053 student 4096 Apr 23 14:18 test

/home/Students/mw1053/Documents/linux/proj2$

**`ls –l`:**

/home/Students/mw1053/Documents/linux/proj2$ ls -l

total 152

-rw------- 1 mw1053 student 336 Apr 21 22:53 Makefile

-rwx------ 1 mw1053 student 12323 Apr 23 14:05 mycat

drwxr-xr-x 2 mw1053 student 4096 Apr 23 14:05 mycat\_src

-rwx------ 1 mw1053 student 18231 Apr 23 14:05 mycp

drwxr-xr-x 2 mw1053 student 4096 Apr 23 14:05 mycp\_src

-rwx------ 1 mw1053 student 14167 Apr 23 14:05 myls

drwxr-xr-x 2 mw1053 student 4096 Apr 23 14:05 myls\_src

-rwx------ 1 mw1053 student 25954 Apr 23 14:29 mysh

drwxr-xr-x 2 mw1053 student 4096 Apr 23 14:40 mysh\_src

-rw------- 1 mw1053 student 169 Apr 10 13:03 README.md

drwx------ 2 mw1053 student 4096 Apr 23 14:18 test

/home/Students/mw1053/Documents/linux/proj2$

**`mycat <Makefile >out`:**

/home/Students/mw1053/Documents/linux/proj2$ mycat <Makefile >out

/home/Students/mw1053/Documents/linux/proj2$ mycat out

SHELL\_FOLD = mysh\_src

CAT\_FOLD = mycat\_src

CP\_FOLD = mycp\_src

LS\_FOLD = myls\_src

.PHONY: clean run mysh mycat mycp myls

all: mysh myls mycat mycd

mysh:

$(MAKE) -C $(SHELL\_FOLD)

mycat:

$(MAKE) -C $(CAT\_FOLD)

mycd:

$(MAKE) -C $(CP\_FOLD)

myls:

$(MAKE) -C $(LS\_FOLD)

clean:

rm -f mysh mycat mycp myls

run:

@./mysh

/home/Students/mw1053/Documents/linux/proj2$

**`myls | mycat`:**

/home/Students/mw1053/Documents/linux/proj2$ myls | mycat

Makefile mycat\_src mycp\_src myls\_src mysh\_src README.md

mycat mycp myls mysh out test

/home/Students/mw1053/Documents/linux/proj2$