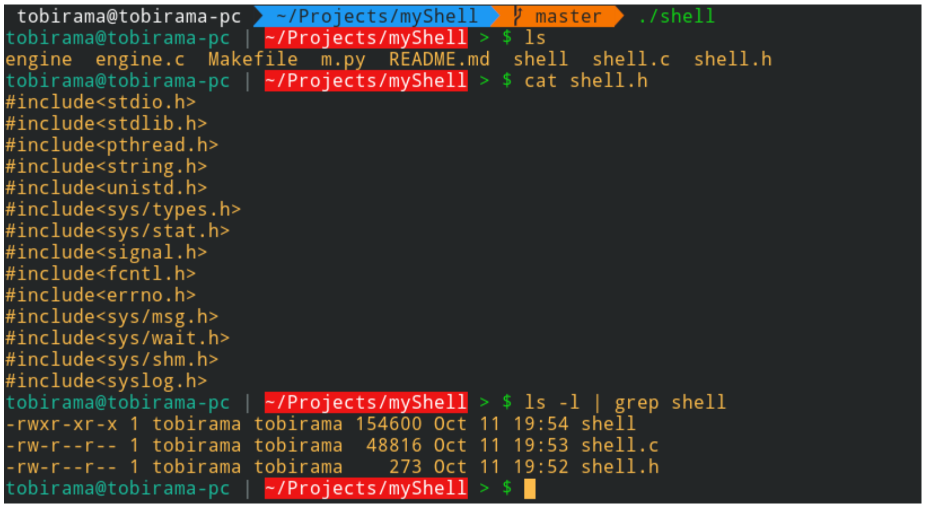
MyShell

This is a simple UNIX Shell made from scratch as part of a Systems programming exercise, implemented in C. This supports basic UNIX shell features and a few more custom commands.

Screenshot



```
 tobirama@tobirama-pc  ~/Projects/myShell  master  ./shell
tobirama@tobirama-pc | ~/Projects/myShell > $ ls
engine  engine.c  Makefile  m.py  README.md  shell  shell.c  shell.h
tobirama@tobirama-pc | ~/Projects/myShell > $ cat shell.h
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<string.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<signal.h>
#include<fcntl.h>
#include<errno.h>
#include<sys/msg.h>
#include<sys/wait.h>
#include<sys/shm.h>
#include<syslog.h>
tobirama@tobirama-pc | ~/Projects/myShell > $ ls -l | grep shell
-rwxr-xr-x 1 tobirama tobirama 154600 Oct 11 19:54 shell
-rw-r--r-- 1 tobirama tobirama  48816 Oct 11 19:53 shell.c
-rw-r--r-- 1 tobirama tobirama    273 Oct 11 19:52 shell.h
tobirama@tobirama-pc | ~/Projects/myShell > $
```

Compile and Run (For POSIX based Systems)

- Compile using : `make`
- Run the program with : `./shell`

Features

- Basic Unix Shell Operations like Job Control, Process Groups, execution of `UNIX` commands.
- Can change the current working directory of the Shell using `cd` command. Home directory can also be interchanged with `~` when changing directories.
- Allows the user to add any directory to the `.myshellrc` configuration file to add to the local `PATH` variable for the Shell. The Shell looks into all these directories one by one, and tries executing each line until there is a success. Otherwise, a suitable error message is printed.
- Can execute `filename` having suitable execute permissions in the current working directory using `./filename`. (Dot Slash commands)
- Foreground and Background Job Control using `fg` and `bg` commands.

Usage (Both fg and bg):

```
fg 1000
```

```
fg %python
```

- Pipe `|` Operator for redirecting commands to another command.
- Redirection `>`, `>>` and `<` Operators for input / output redirection from / to a file respectively.

Usage :

```
ls | grep out > output.txt
```

- Support for System V Message Queues using `##` operator. This passes the output of an input command via Message Queues, which then pipes it to other commands. (Note : Unfortunately enough, this is the comment syntax for bash)
- Support for Shared Memory Redirection using `SS` operator. This is similar to the `##` operator, but uses the system's Shared Memory for faster execution and does not have the limitation of a fixed buffer size like Message Queues.

Usage for `##` and `SS` :

```
ls ## wc , sort ## Uses System V Message Queues
```

```
ls SS wc , sort ## Uses Shared Memory
```

- Can start daemon processes with the `daemonize` command.

Usage (Executes `bash script.sh` as a daemon process) :

```
daemonize bash script.sh
```

TODO

- Add some more builtin commands.
- Have implemented a basic AutoComplete Engine using Tries. Incorporate it into the Shell, possibly with the help of the `ncurses` library.

Bugs

- Suspend signal does not work as intended, if the background job waits for `tty` as input.
- `&` does not allow support with pipes and redirection at present.
- `##` Operation for piping via System V Message Queues does not presently support chaining with pipes and redirection operators.
- `SS` Operation for piping via Shared Memory does not presently support chaining with pipes and redirection operators.

- As of now, any command executes as expected only if there is atleast one space between any two keywords. Even if you want to get the output to multiple commands using `,`, you still need to insert a space before and after the comma.