

COMP304 Project 1

Atakan Kara, Farrin Marouf Sofian

Part1:

In this section we used the `execv` command. This command takes two arguments, the first argument is the path to the command that we want to run and the second argument is the parameters that we want to pass to that function. Since the path to the executables begin with `/usr/bin`, we concatenated the command name (`command->name`) we get from the terminal with the parameters (`command->args`) and passed it to the `execv` function along with `NULL` as the last parameter.

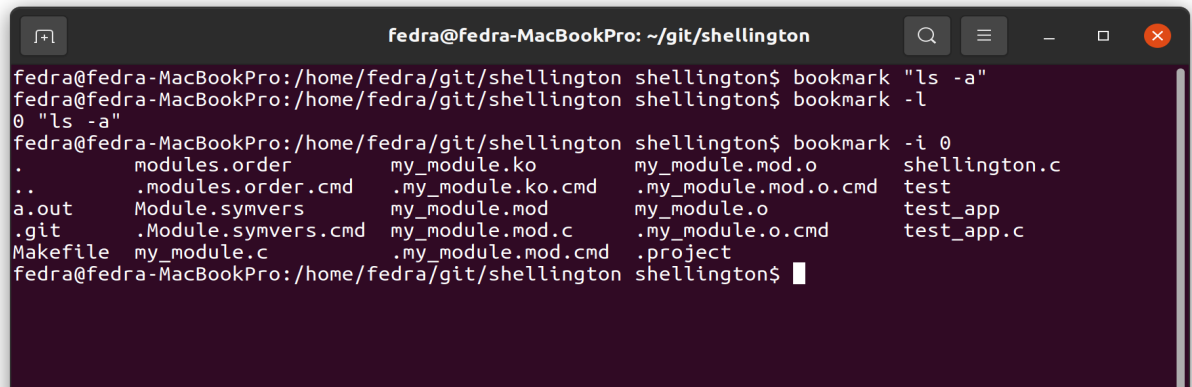
Part 2:

1) short

We used two files to keep track of both names and paths so that our command would work even the terminal session is terminated. When `set` executed, we add name to `names.txt` file and path to `paths.txt` file. When `jump` executed, we first search for name in `names.txt` file and use and run `cd` command on path at the very index.

2) Bookmark

First we check for the arguments written along with the bookmark command and divide the tasks into different parts depending on them. To store the bookmark command, we created a 2 dimensional array and stored the command using `command->args`. For deleting a bookmarked command, we deleted it with the given index and shifted the whole array. To list the bookmarks, we traversed the array and printed the strings. To run the bookmark command, we passed the command to `parse_command` and then `process_comand` functions.

A terminal window titled 'fedra@fedra-MacBookPro: ~/git/shellington'. The user enters 'bookmark "ls -a"', followed by 'bookmark -l', and then 'bookmark -i 0'. The output shows a directory listing with columns for file names and their corresponding command aliases. The files listed are: ., .modules.order, .modules.order.cmd, a.out, .git, Makefile, my_module.c, modules.order, my_module.ko, Module.symvers, my_module.mod, my_module.c, my_module.mod.cmd, my_module.ko, my_module.mod.o, my_module.o, my_module.o.cmd, shellington.c, test, test_app, and test_app.c.

```
fedra@fedra-MacBookPro: ~/git/shellington shellington$ bookmark "ls -a"
fedra@fedra-MacBookPro: ~/git/shellington shellington$ bookmark -l
0 "ls -a"
fedra@fedra-MacBookPro: ~/git/shellington shellington$ bookmark -i 0
..          modules.order      my_module.ko      my_module.mod.o    shellington.c
.           .modules.order.cmd .my_module.ko.cmd .my_module.mod.o.cmd test
a.out       Module.symvers     my_module.mod     my_module.o         test_app
.git        .Module.symvers.cmd  my_module.mod.c   .my_module.o.cmd    test_app.c
Makefile    my_module.c          .my_module.mod.cmd .project
fedra@fedra-MacBookPro: ~/git/shellington shellington$
```

3) remindme

This command takes in a time in format hour.minute and a string to be shown. We used `notify-send` to show the alert and `crontab -l` to schedule the command in the given time. We used `sprintf` to modify the text string and the following command in using `sh -c`. The following is the example code:

```
Crontab -l | { cat; echo * * * * * XDG_RUNTIME_DIR=/run/user/$(id -u)
/usr/bin/notify-send message'; } | crontab -
```

4) cwallpaper:

`cwallpaper` is one of our custom commands which will change desktop wallpaper. It uses the Unsplash random image generator and downloads the picture, stores it and later sets it as the desktop background using `wget` -) and `gsettings` commands along with parameters that make it possible to access and modify the wallpapers. They commands are first created in temporary arrays and then executed using `execv` command.

To use this functionality, use `cwallpaper`.

5) todo:

`Todo` command is our second custom command. This command takes in add parameter and a string and adds the task to a file which is later stored in a txt file. To mark a task as done, execute `todo done idx`, where `idx` is the index of the task that is finished. The `todo -l` will list all of the tasks in the list.

Possible commands:

`todo add sleep` //adds sleeping task

```
todo -l //lists all tasks  
todo done 5 //labels 5th task as done!  
todo clear //clears the list
```

Part 3:

Important note: You should run shellington.o as sudo user to use these functions.

- a) Pstraverse: We wrote a kernel module and implemented both DFS and BFS. We used ioctl to communicate between userspace and kernel space. Our module had the necessary functions so we directly called those functions and passed a PID as an argument. We iterated over child processes with respective order for -d and -b options. When exit command is given, my_module.ko is removed from the modules list. The command works as follows : pstraverse -d or -b <pid>

References:

1. <https://www.wikihow.com/Change-Your-Desktop-Wallpaper-on-Linux-Mint>
2. <http://derekmolloy.ie/writing-a-linux-kernel-module-part-2-a-character-device/>
3. <https://www.geeksforgeeks.org/crontab-in-linux-with-examples/>