UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

CSCI 3020 / SOFE 3950: Operating Systems

Laboratory 2: Shell Project

October 30th, 2015

Lab group 11
Mohannad Abdo (100523158)
Luisa Rojas-Garcia (100518772)
Mirna Zohiry (100535658)

INTRODUCTION

The Shell or Command Line Interpreter a type of human-computer interaction that is text input and output based, and the fundamental user interface to an operating system. Through this program, the user is able to execute different tasks as well as different programs and since the shell is only one layer above the OS, the shell allows for operations that are not always valid or possible using the GUI (graphical user interphace).

OBJECTIVE

The objective of this project is to develop a command line interpreter – myshell- that runs the commands below, which are entered by the user through the terminal application. Note this program is to be designed for UNIX shell environment.

COMMAND	DESCRIPTION
`cd <directory>`	Change current directory to `<directory>`. If the argument is not present, report the current directory.
`clr`	Clear the screen.
`dir <directory>`	List the contents of directory .
`environ`	List all the environment strings.
`echo <comment>`	Display `<comment>` on the display followed by a new line (multiple spaces/tabs may be reduced to a single space).
`help`	Display the user manual.
`pause`	Pause operation of the shell until 'ENTER' is pressed.
`quit`	Quit the shell.
`myshell <batchprogram>`	Processes the commands read from the batch file provided.
`<program> &`	Launches `<program>` in the background.

`io < input > output`	Processes the commands read from the batch file, displays its output to the terminal and writes (or over-writes, if necessary) them to the output file.
`io < input >> output`	Processes the commands read from the batch file, displays its output to the terminal and appends them to the output file.

PROGRAM FLOW

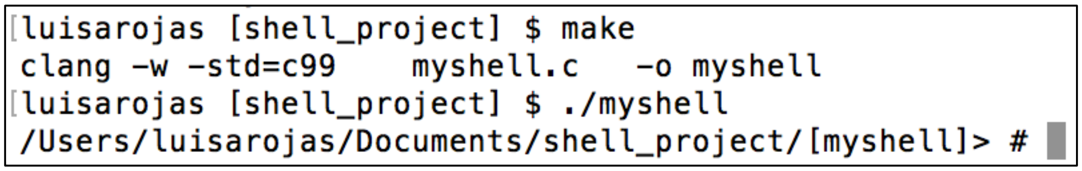
Once the shell starts, it will wait for the user to enter an input command. After entering the command, the user will have to press ‘Enter’ in order for the command to be executed. The shell will then check whether the command exists or not, and execute accordingly. It will wait for the termination of the executed process and will ask the user to enter a command again. Unless the user give a quit command, the shell will then terminate.

COMPILING THE PROGRAM

The program could have been executed using the following command:

```
clang –Wall –Wextra –std=c99 myshell.c –o myshell
```

However, for this project, a makefile was applied instead:



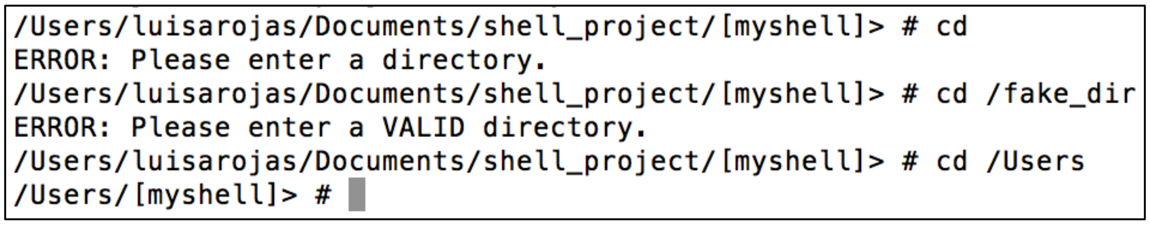
```
luisarojas [shell_project] $ make
clang -w -std=c99    myshell.c   -o myshell
luisarojas [shell_project] $ ./myshell
/Users/luisarojas/Documents/shell_project/[myshell]> # 
```

Figure 1. Makefile execution.

As shown in Figure 1, the myshell prompt is shown afterwards and the program is ready for user input.

COMMANDS SUPPORTED

`cd <directory>`: This command is used to change the directory the user is in to the one provided in the input. If the directory does not exist an appropriate error should be reported, as shown in Figure 2.

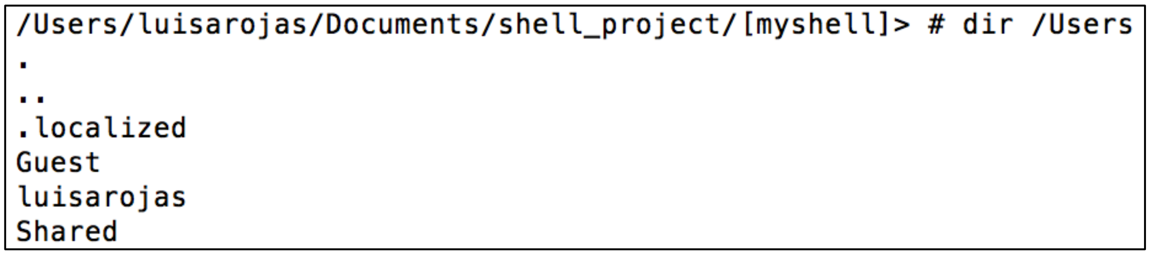


```
/Users/luisarojas/Documents/shell_project/[myshell]> # cd
ERROR: Please enter a directory.
/Users/luisarojas/Documents/shell_project/[myshell]> # cd /fake_dir
ERROR: Please enter a VALID directory.
/Users/luisarojas/Documents/shell_project/[myshell]> # cd /Users
/Users/[myshell]> #
```

Figure 2. cd <directory> command execution

`clr`: We can enter clr to clear the terminal screen.

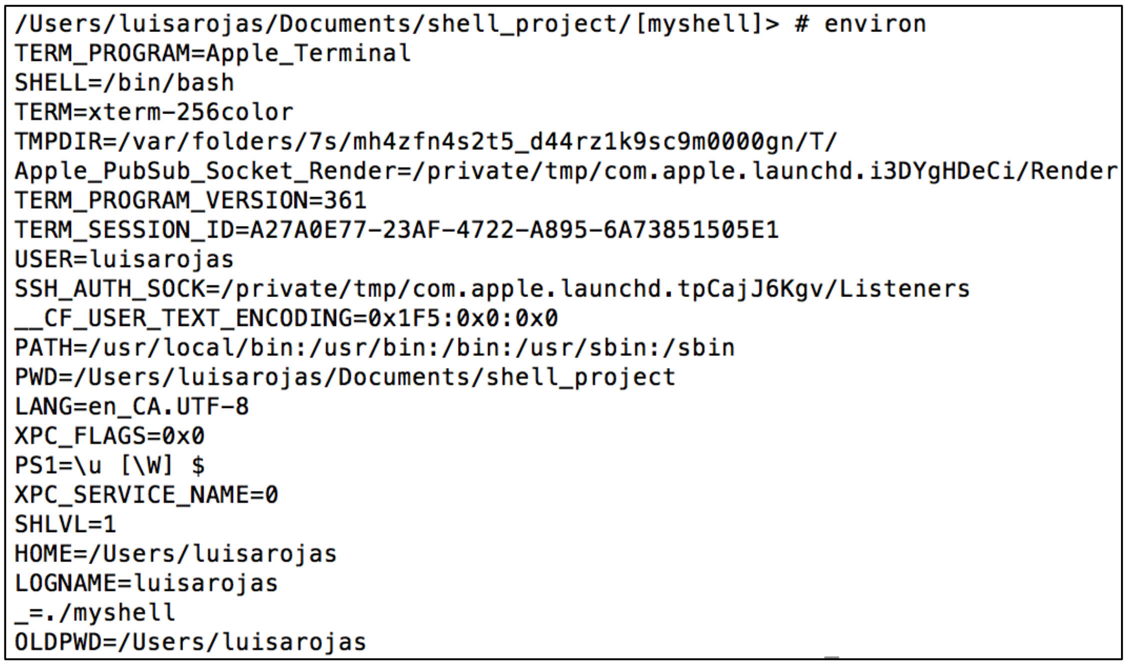
`dir <directory>`: Shows a list of all the contents of the directory the user inputs as parameter. If no directory is given, then an error is shown (see Figure 1).



```
/Users/luisarojas/Documents/shell_project/[myshell]> # dir /Users
.
..
.localized
Guest
luisarojas
Shared
```

Figure 3. dir <directory> command execution

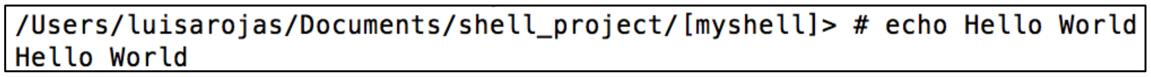
`environ`: Simply lists all the environment strings when used.



```
/Users/luisarojas/Documents/shell_project/[myshell]> # environ
TERM_PROGRAM=Apple_Terminal
SHELL=/bin/bash
TERM=xterm-256color
TMPDIR=/var/folders/7s/mh4zfn4s2t5_d44rz1k9sc9m0000gn/T/
Apple_PubSub_Socket_Render=/private/tmp/com.apple.launchd.i3DYgHDeCi/Render
TERM_PROGRAM_VERSION=361
TERM_SESSION_ID=A27A0E77-23AF-4722-A895-6A73851505E1
USER=luisarojas
SSH_AUTH_SOCK=/private/tmp/com.apple.launchd.tpCajJ6Kgv/Listeners
__CF_USER_TEXT_ENCODING=0x1F5:0x0:0x0
PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
PWD=/Users/luisarojas/Documents/shell_project
LANG=en_CA.UTF-8
XPC_FLAGS=0x0
PS1=\u [\W] $
XPC_SERVICE_NAME=0
SHLVL=1
HOME=/Users/luisarojas
LOGNAME=luisarojas
_=./myshell
OLDPWD=/Users/luisarojas
```

Figure 4. environ command execution

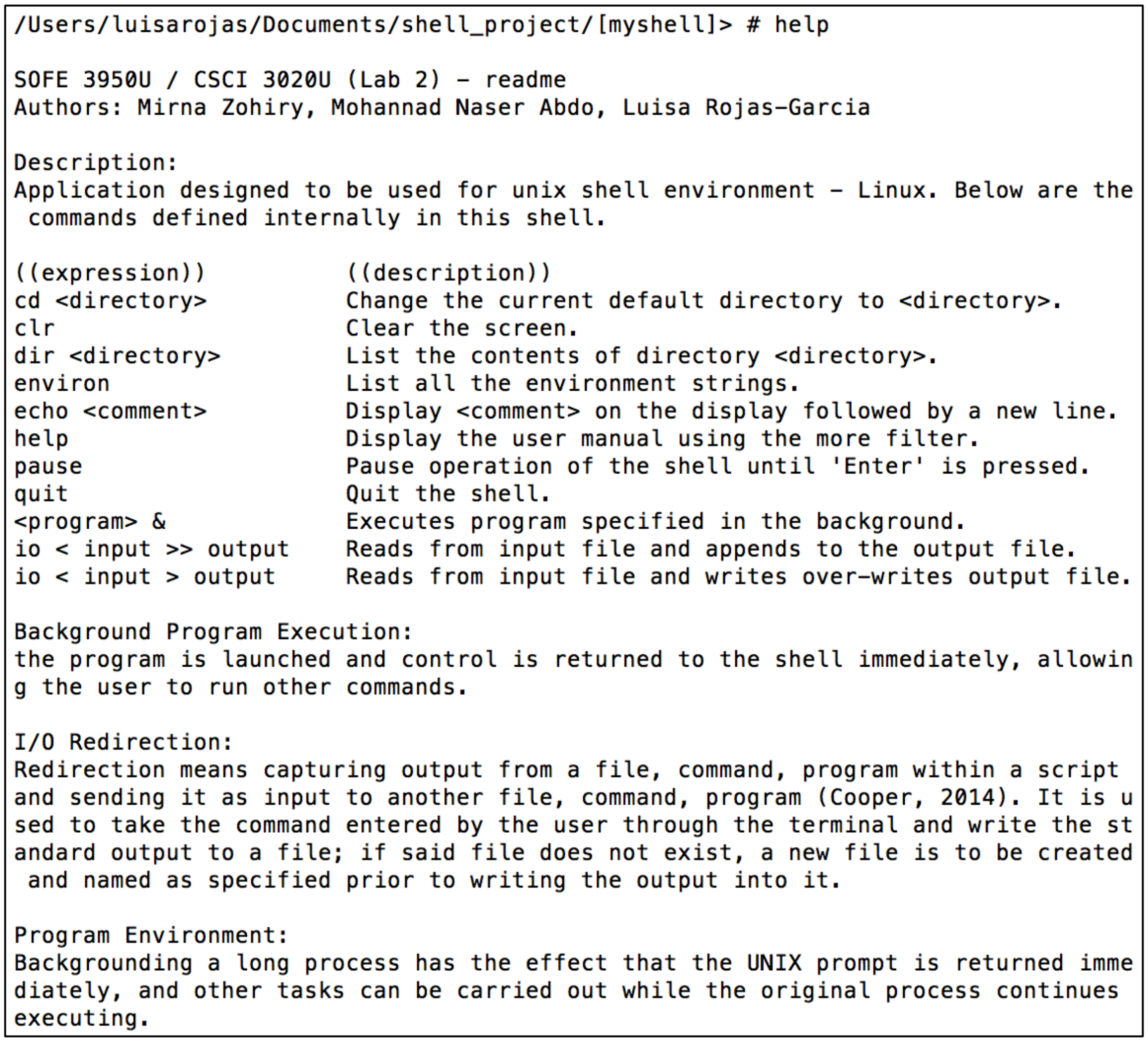
`echo <comment>:` The strings following the `echo` command will be displayed on the terminal.



```
/Users/luisarojas/Documents/shell_project/[myshell]> # echo Hello World
Hello World
```

Figure 5. echo command execution

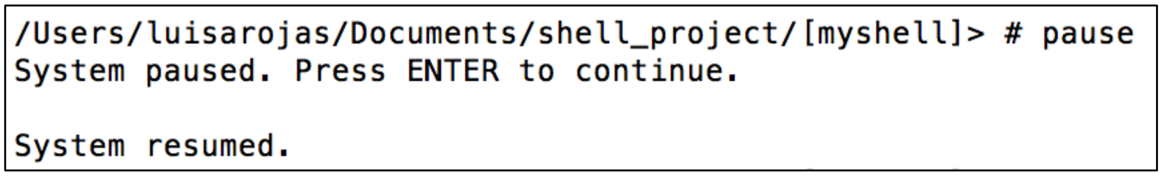
`help:` Displays the readme file (manual) to the user, which contains all the commands supported what the shell, what their use is, the syntax, and other useful information.



```
/Users/luisarojas/Documents/shell_project/[myshell]> # help

SOFE 3950U / CSCI 3020U (Lab 2) - readme
Authors: Mirna Zohiry, Mohannad Naser Abdo, Luisa Rojas-Garcia

Description:
Application designed to be used for unix shell environment - Linux. Below are the
 commands defined internally in this shell.

((expression))          ((description))
cd <directory>          Change the current default directory to <directory>.
clr                     Clear the screen.
dir <directory>         List the contents of directory <directory>.
environ                 List all the environment strings.
echo <comment>          Display <comment> on the display followed by a new line.
help                    Display the user manual using the more filter.
pause                   Pause operation of the shell until 'Enter' is pressed.
quit                    Quit the shell.
<program> &             Executes program specified in the background.
io < input >> output    Reads from input file and appends to the output file.
io < input > output     Reads from input file and writes over-writes output file.

Background Program Execution:
the program is launched and control is returned to the shell immediately, allowin
g the user to run other commands.

I/O Redirection:
Redirection means capturing output from a file, command, program within a script
and sending it as input to another file, command, program (Cooper, 2014). It is u
sed to take the command entered by the user through the terminal and write the st
andard output to a file; if said file does not exist, a new file is to be created
 and named as specified prior to writing the output into it.

Program Environment:
Backgrounding a long process has the effect that the UNIX prompt is returned imme
diately, and other tasks can be carried out while the original process continues
executing.
```

Figure 6. help command execution

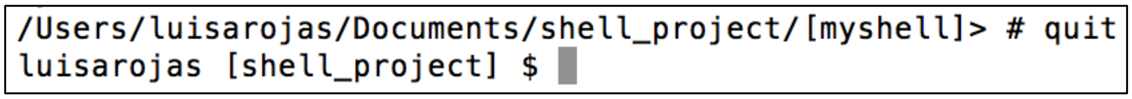
pause: Does a pause operation of the shell until the 'ENTER' key is pressed.



```
/Users/luisarojas/Documents/shell_project/[myshell]> # pause
System paused. Press ENTER to continue.

System resumed.
```

Figure 7. pause command execution

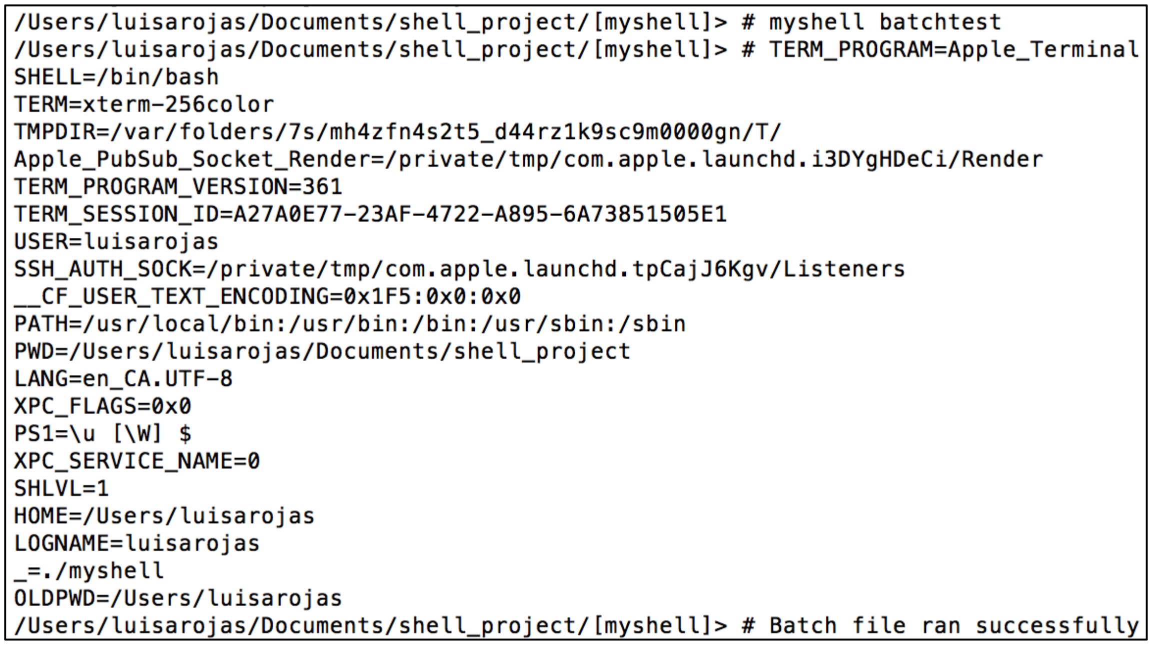
quit: Terminates the myshell program.



```
/Users/luisarojas/Documents/shell_project/[myshell]> # quit
luisarojas [shell_project] $
```

Figure 8. quit command execution

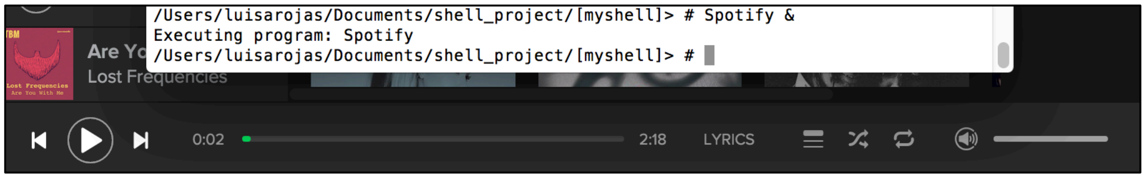
myshell <batchprogram>: The batchfile contains a set of command lines for the shell to process. The shell exits when the end of file is reached. If the shell is invoked without a command line argument, it solicits input from the user via a prompt on the display.



```
/Users/luisarojas/Documents/shell_project/[myshell]> # myshell batchtest
/Users/luisarojas/Documents/shell_project/[myshell]> # TERM_PROGRAM=Apple_Terminal
SHELL=/bin/bash
TERM=xterm-256color
TMPDIR=/var/folders/7s/mh4zfn4s2t5_d44rz1k9sc9m0000gn/T/
Apple_PubSub_Socket_Render=/private/tmp/com.apple.launchd.i3DYgHDeCi/Render
TERM_PROGRAM_VERSION=361
TERM_SESSION_ID=A27A0E77-23AF-4722-A895-6A73851505E1
USER=luisarojas
SSH_AUTH_SOCK=/private/tmp/com.apple.launchd.tpCajJ6Kgv/Listeners
__CF_USER_TEXT_ENCODING=0x1F5:0x0:0x0
PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
PWD=/Users/luisarojas/Documents/shell_project
LANG=en_CA.UTF-8
XPC_FLAGS=0x0
PS1=\u [\W] $
XPC_SERVICE_NAME=0
SHLVL=1
HOME=/Users/luisarojas
LOGNAME=luisarojas
_=./myshell
OLDPWD=/Users/luisarojas
/Users/luisarojas/Documents/shell_project/[myshell]> # Batch file ran successfully
```

Figure 9. myshell <batchprogram> command execution

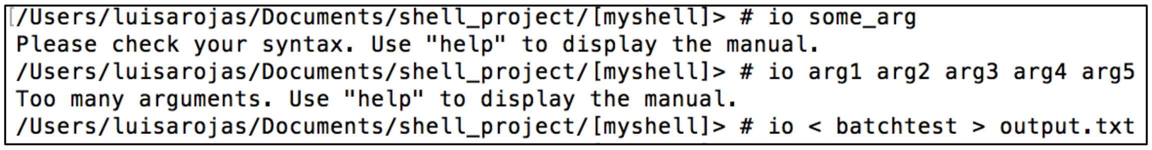
`<program> &:` Runs <program> in the background, returning to the command line prompt and making itself available for more input immediately after launching the program specified.



```
/Users/luisarojas/Documents/shell_project/[myshell]> # Spotify &
Executing program: Spotify
/Users/luisarojas/Documents/shell_project/[myshell]> #
```

Figure 10. <program> & command execution

`io < input >/>> output:` Reads the contents of the `input` file, executes them and then it either overwrites on an existing file (>), creates a new file and saves the output there (>), or appends the information to an existing file (>>).



```
/Users/luisarojas/Documents/shell_project/[myshell]> # io some_arg
Please check your syntax. Use "help" to display the manual.
/Users/luisarojas/Documents/shell_project/[myshell]> # io arg1 arg2 arg3 arg4 arg5
Too many arguments. Use "help" to display the manual.
/Users/luisarojas/Documents/shell_project/[myshell]> # io < batchtest > output.txt
```

Figure 11. io command execution

RELEVANT TERMS

I/O REDIRECTION

Redirection means capturing output from a file, command, program within a script and sending it as input to another file, command, program (Cooper, 2014). It is used to take the command entered by the user through the terminal and write the standard output to a file; if said file does not exist, a new file is to be created and named as specified prior to writing the output into it. The command line: programname arg1 arg2 < inputfile > outputfile, will execute the program <programname> with arguments arg1 and arg2, the stdin FILE stream replaced by inputfile and the stdout FILE stream replaced by outputfile. For output redirection; redirection character > will create an outputfile if it does not exist and truncates if it does. While redirection token >> will create an outputfile if it does not exist and appends to if it does (‘UNIX Introduction “, 2000).

PROGRAM ENVIRONMENT

Program or software environment is the term commonly used to refer to support an application. A program environment for a particular application could include the operating system, the database system, specific development tools or compiler (“Software Environment”, webopedia). In this lab we are using the C programming language on UNIX platform to implement myshell. UNIX is an operating system that is made up of three parts: the kernel, the shell, and the programs; we will be focusing on the shell, which acts as an interface between the user and the kernel. The shell is a command line interpreter in which it takes the user’s inputs and execute them depending on the commands prompted by the user. myshell is implemented in a way to support certain commands as specified in the previous sections (‘UNIX Introduction “, 2000).

BACKGROUND PROGRAM EXECUTION

UNIX is a multitasking operating system in which many processes can run at the same time. There are two ways to run a process on UNIX: The first is when the running process is in the foreground, in which the user launches a process and waits for it to be executed and returned back to the shell so that he/she would launch another process. The second, is when then running process is in the background. In this case, the program is launched and control is returned to the shell immediately, allowing the user to run other commands. In order to execute a process in the background, ampersand character ‘&’ should be added to the end of the command line (‘UNIX Introduction “, 2000).

CONCLUSION

In this project, we have learned how to implement a shell that runs given commands and display results through the terminal. We have also learned how to develop software concurrently in a team using Git, and important terminology like I/O redirection, program environment and background execution. By working on this assignment we also gained experience using makefiles and developing multisource file projects in the C programming language.

GITHUB REPOSITORY

https://github.com/MirnaZohiry/Lab-2-OS/tree/master

RESOURCES USED

UNIX Introduction. (2000, October 9). Retrieved October 30, 2015, from http://www.ee.surrey.ac.uk/Teaching/Unix/unixintro.html

Cooper, M. (2014, March 10). Chapter 20. I/O Redirection. Retrieved October 31, 2015, from http://www.tldp.org/LDP/abs/html/io-redirection.html

Software Environment. (n.d.). Retrieved October 30, 2015, from http://www.webopedia.com/TERM/S/software_environment.html