


Beta [Try the new code view](#)

 master ▾

...

[Operating-Systems](#) / [Labs](#) / [lab1](#) / [README.md](#)



OmarMetmwah Update README.md ...

 History

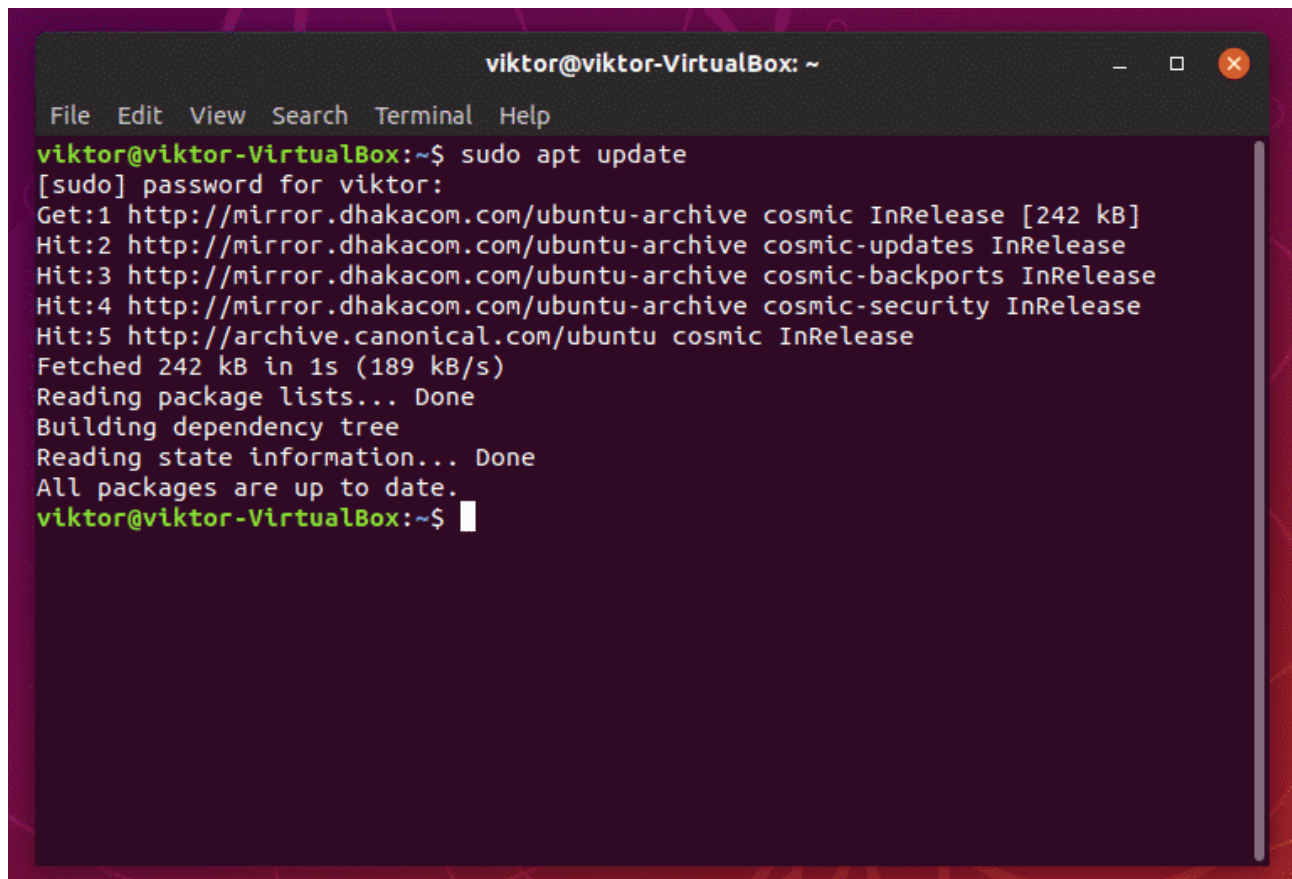
 3 contributors



215 lines (160 sloc) | 8.79 KB

...

Simple Shell (Multi-Processing)



```
viktor@viktor-VirtualBox: ~  
File Edit View Search Terminal Help  
viktor@viktor-VirtualBox:~$ sudo apt update  
[sudo] password for viktor:  
Get:1 http://mirror.dhakacom.com/ubuntu-archive cosmic InRelease [242 kB]  
Hit:2 http://mirror.dhakacom.com/ubuntu-archive cosmic-updates InRelease  
Hit:3 http://mirror.dhakacom.com/ubuntu-archive cosmic-backports InRelease  
Hit:4 http://mirror.dhakacom.com/ubuntu-archive cosmic-security InRelease  
Hit:5 http://archive.canonical.com/ubuntu cosmic InRelease  
Fetched 242 kB in 1s (189 kB/s)  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
All packages are up to date.  
viktor@viktor-VirtualBox:~$
```

1. Objectives

1. Familiarity with system calls in Unix environment.
2. Introduction to processes and multi-processing.
3. Introduction to signal handling in Unix environment.

2. Problem Statement

It is required to implement a Unix shell program. A shell is simply a program that conveniently allows you to run other programs. Read up on your favorite shell to see what it does.

Your shell must support the following commands:

1. The internal shell command "exit" which terminates the shell
 - **Concepts:** shell commands, exiting the shell.
 - **System calls:** exit()
2. A command with no arguments
 - **Example:** ls, cp, rm ...etc

- **Details:** Your shell must block until the command completes and, if the return code is abnormal, print out a message to that effect.
- **Concepts:** Forking a child process, waiting for it to complete and synchronous execution.
- **System calls:** fork(), execvp(), exit(), waitpid()

3. A command with arguments

- **Example:** ls -l
- **Details:** Argument 0 is the name of the command.
- **Concepts:** Command-line parameters.

4. A command, with or without arguments, executed in the background using &.

- **Example:** firefox &
- **Details:** In this case, your shell must execute the command and return immediately, not blocking until the command finishes.
- **Concepts:** Background execution, signals, signal handlers, processes and asynchronous execution.
- **Requirements:** You have to show that the opened process will be nested as a child process to the shell program via opening the task manager found in the operating system like the one shown in figure 1. Additionally you have to write in a log file (basic text file) when a child process is terminated (main application will be interrupted by a SIGCHLD signal). So you have to implement an interrupt handler to handle this interrupt and do the corresponding action to it.

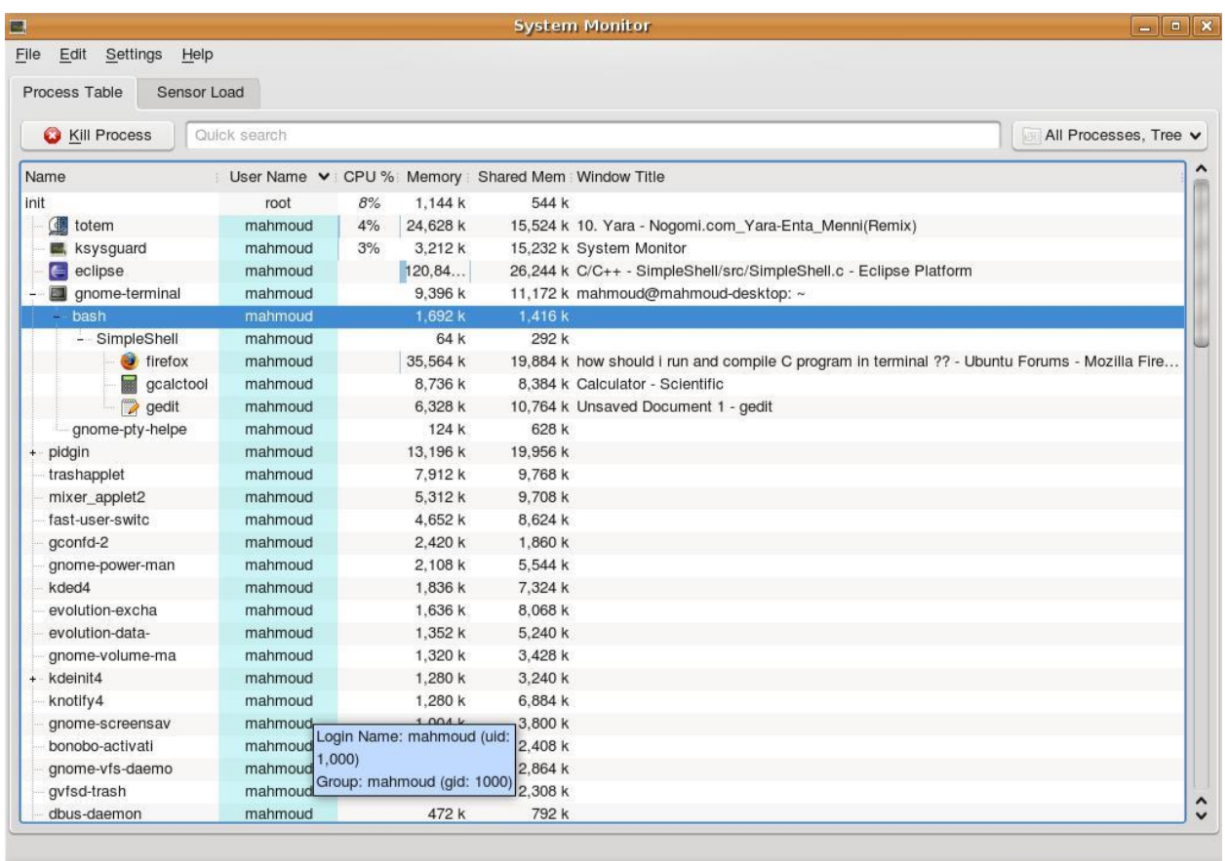
5. Shell builtin commands

- **Commands:** cd & echo
- **Details:** for the case of:
 - **cd:** Cover all the following cases (assume no spaces in path):
 - cd
 - cd ~
 - cd ..
 - cd absolute_path
 - cd relative_path_to_current_working_directory
 - **echo:** Prints the input after evaluating all expressions (assume input to echo must be within double quotations).
 - echo "wow" => wow
 - export x=5
 - echo "Hello \$x" => Hello 5

6. Expression evaluation

- **Commands:** export

- **Details:** Set values to variables and print variables values. No mathematical operations is needed.
- **Export Details:** Accept input of two forms, either a string without spaces, or a full string inside double quotations.
- **Example:**
 - `export x=-l`
 - `ls $x =>` Will perform `ls -l`
 - `export y="Hello world"`
 - `echo "$y" =>` Hello world



Name	User Name	CPU %	Memory	Shared Mem	Window Title
init	root	8%	1,144 k	544 k	
totem	mahmoud	4%	24,628 k	15,524 k	10. Yara - Nogomi.com_Yara-Enta_Menni(Remix)
ksysguard	mahmoud	3%	3,212 k	15,232 k	System Monitor
eclipse	mahmoud		120,84...	26,244 k	C/C++ - SimpleShell/src/SimpleShell.c - Eclipse Platform
gnome-terminal	mahmoud		9,396 k	11,172 k	mahmoud@mahmoud-desktop: ~
bash	mahmoud		1,692 k	1,416 k	
SimpleShell	mahmoud		64 k	292 k	
firefox	mahmoud		35,564 k	19,884 k	how should i run and compile C program in terminal ?? - Ubuntu Forums - Mozilla Fire...
gcalctool	mahmoud		8,736 k	8,384 k	Calculator - Scientific
gedit	mahmoud		6,328 k	10,764 k	Unsaved Document 1 - gedit
gnome-pty-helpe	mahmoud		124 k	628 k	
pidgin	mahmoud		13,196 k	19,956 k	
trashapplet	mahmoud		7,912 k	9,768 k	
mixer_applet2	mahmoud		5,312 k	9,708 k	
fast-user-switc	mahmoud		4,652 k	8,624 k	
gconfd-2	mahmoud		2,420 k	1,860 k	
gnome-power-man	mahmoud		2,108 k	5,544 k	
kded4	mahmoud		1,836 k	7,324 k	
evolution-excha	mahmoud		1,636 k	8,068 k	
evolution-data-	mahmoud		1,352 k	5,240 k	
gnome-volume-ma	mahmoud		1,320 k	3,428 k	
kdeinit4	mahmoud		1,280 k	3,240 k	
knotify4	mahmoud		1,280 k	6,884 k	
gnome-screensav	mahmoud		1,004 k	3,800 k	
bonobo-activati	mahmoud		1,000 k	2,408 k	
gnome-vfs-daemo	mahmoud		1,000 k	2,864 k	
gvfsd-trash	mahmoud		1,000 k	2,308 k	
dbus-daemon	mahmoud		472 k	792 k	

Figure 1 Firefox, Calculator and Gedit are child processes to the SimpleShell process*

3. Problem Description

1. Your command shell should take the user command and its parameter(s), i.e., "ls" and "-l" in this example, and convert them into C strings. (Recall that a C string terminates with a null string, i.e., \0.)
2. The command shell should create a child process via `fork()`.
3. The child process passes the C strings—the command and parameter(s)—to `execvp()`.
4. The child exits if `execvp()` returns error.

5. The parent process, i.e., the command shell, should wait, via `waitpid(pid_t pid, int *statusPtr, int options)`, for the child process to finish.
6. The command shell gets the next command and repeats the above steps. The command shell terminates itself when the user types `exit`.
7. No zombie process should ever exist, you can read more about zombie processes and how to handle them at the [Reading & Resources section](#).

In case a user wants to execute the command in background (i.e. as a background process), he/she writes `&` at the end of the command. For example, a user command can be:

```
firefox &
```

In this case, your command shell should not wait for the child by skipping the Step 5.

You should keep a log file (basic text file) for your shell program such that whenever a child process terminates, the shell program appends the line "Child process was terminated" to the log file. To do this, you have to write a signal handler that appends the line to the log file when the SIGCHLD signal is received.

Pseudocode

The shell program should be written as following pseudocode:

```
function parent_main()
    register_child_signal(on_child_exit())
    setup_environment()
    shell()
```

```
function on_child_exit()
    reap_child_zombie()
    write_to_log_file("Child terminated")
```

```
function setup_environment()
    cd(Current_Working_Directory)
```

```
function shell()
    do
        parse_input(read_input())
        evaluate_expression():
```

```

        switch(input_type):
            case shell_builtin:
                execute_shell_bultin();
            case executable_or_error:
                execute_command();

while command_is_not_exit

function execute_shell_bultin()
    swirch(command_type):
        case cd:
        case echo:
        case export:

function execute_command()
    child_id = fork()
    if child:
        execvp(command parsed)
        print("Error)
        exit()
    else if parent and foreground:
        waitpid(child)

```

Notes

- You should register the SIGCHLD signal at the beginning of your main as shown in [this example](#), so when a child dies, the parent process receives SIGCHLD (or SIGCLD) signal.
- To see the set of all signals supported on your system, type, kill -l.
- Use a process monitor package to monitor your processes. Provide a screenshot for your shell parent process and some child processes spawned as background processes. Suggested packages: KSysguard or Gnome-System-Monitor.
- Reading [this article about waitpid\(pid_t pid, int *statusPtr, int options\)](#) is a must.

4. Deliverables

- Complete C source code, commented thoroughly and clearly.
- A report that includes:
 - A description of the overall organization of your code and the major functions.

- Sample runs.
- screenshots for the processes hierarchy in KSysguard (or any similar package) during the execution of your shell program.
- A 3-min (tolerance of 30 seconds only) video that shows the output of the next test case.

Test Case

Open your shell.

Execute the following commands.

```
ls
mkdir test
ls
ls -a -l -h
export x="-a -l -h"
ls $x
```

Execute the following commands in the same session, but **show** that shell is stuck and cannot execute other commands while firefox is open.

Note: In order for this test to work correctly, firefox has to be closed before executing this command. Also, if firefox is not available, you may use 'gedit' instead.

```
firefox
```

Close the firefox.

Open the log file, show us its content, then close it.

Execute the following commands in the same session, but **show** that shell is not stuck and can execute other commands while firefox is open.

```
firefox &
```

Open the log file, show us its content, then close it.

Open the system monitor and expand all running processes under your shell process.

Open the system monitor and **Search** for all firefox processes.

Execute the following command in the same session, it should show us an error message.

```
hey
```

Execute the following command in the same session, it should close the shell from the first time you execute it.

```
exit
```

That's it!

Lab requirements

- Students will be working individually.
- You can only use C programming language.
- Use Ubuntu operating system for development.
- Submit the deliverables using the submission form provided in your class.
- If the submitted code cannot replicate the same video scenario, then this will be considered cheating and the "cheating and plagiarism" policy will be applied.

Readings & Resources

- [What is a zombie process in Linux?](#)
- [Process states & dealing with zombie processes](#)
- [Linux Signals](#)
- [Important notes on wait\(\) and reaping zombies](#)
- [sigaction\(2\) — Linux manual page](#)
- [signal\(2\) — Linux manual page](#)

[Give feedback](#)