

# Simple Shell

Name : Hamza Hassan Mohammed

Id : 26

# Index

- 1- Description
- 2- Basic Idea
- 3- Main Functions
- 4- Simple Runs

# 1- Description

- The Shell provides an interface to take the input and execute the programs based on that input by either providing an internal implementation to the commands or calling executables existing in the bin folder.
- The shell is implemented in C and making system to calls using "execvp" command.

## 2- Basic idea

- The shell simply consists of an infinite loop which could be broken by the user entering 'exit' command.
- Each iteration of the loop consists mainly of 5 steps
  - Taking input from the user
  - Splitting the input as needed
  - Forking the parent process
  - Calling "execvp" in the child process to execute the entered command.
  - Wait for the child process to terminate (unless it is supposed to run in the background) .

# 3- Main Functions

- Handling the parent process properly

```
if (!background) {
    fprintf(debug, "Waiting for the child process to end \n");
    int status;
    fprintf(debug, "\n**We are in parent now with pid = %d and It is going to wait for the child **\n", getpid());
    while (wait(&status) != child_pid) {
    }
} else {
    fprintf(debug, "\n** We are in the parent now with pid = %d but we aren't going to wait for background child \n",
            getpid());
    fprintf(debug, "Parent will not wait deliberately for the child \n");
}
```

When we fork the parent process, the child process is created and executed parallelly with the parent process.

Waiting for the child process to end (using “waitpid” function) is necessary so that when the child process terminates, a signal is sent to the parent to clear his child’s memory and contents.

An exception here is when the child process is supposed to be a background process so in this case, we have to make the parent continue its execution and not to wait until the child end.

- Handler

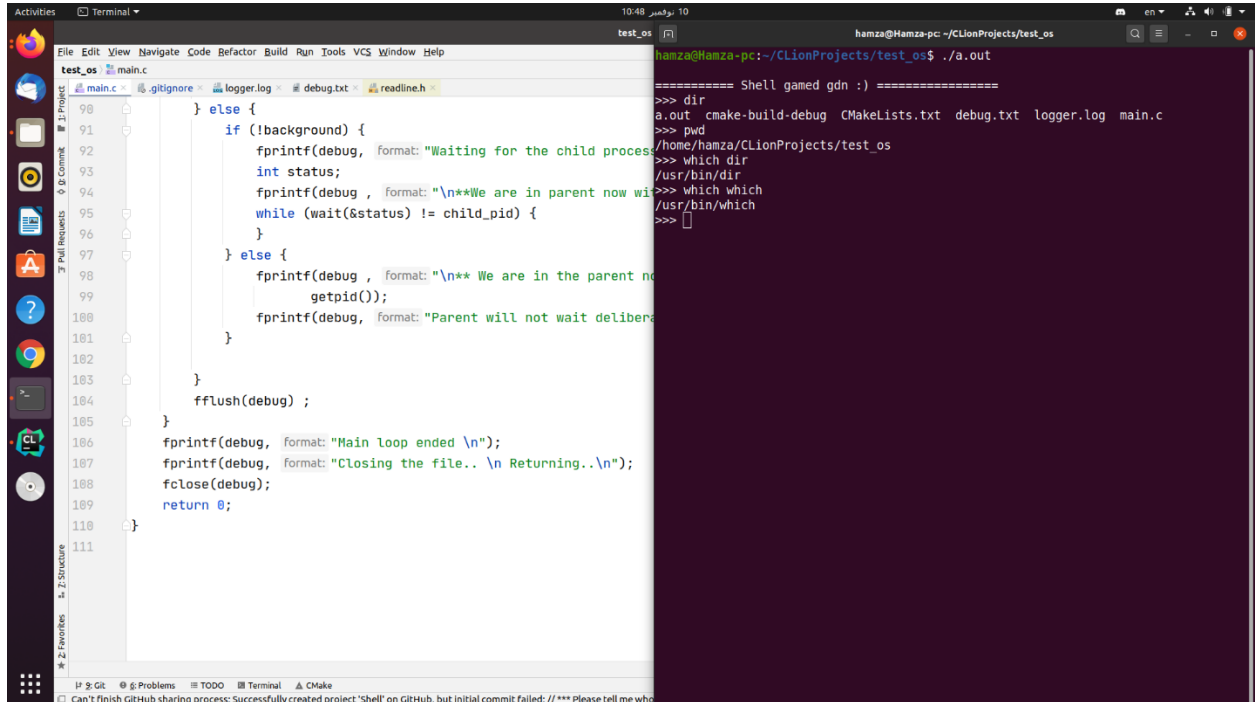
```
void handler() {  
    int status;  
    wait(&status); // clean the child :)  
    fprintf(logger, "Child process in terminated with status = %d \n ", status);  
    fflush(logger) ;  
    //    printf("Child process in terminated with status = %d \n ", status);  
    //signal(SIGINT , handler) ;  
}
```

Before even forking a process we define a signal as shown below.

`signal(SIGCHLD, handler);`

this signal is sent to the parent and calls the handler function whenever a process is terminated so that we avoid the Zombie processes to accumulating in the process table.

## 4- Sample runs and shots



The screenshot shows the CLion IDE interface. The main editor displays a C program named `main.c` with the following code:

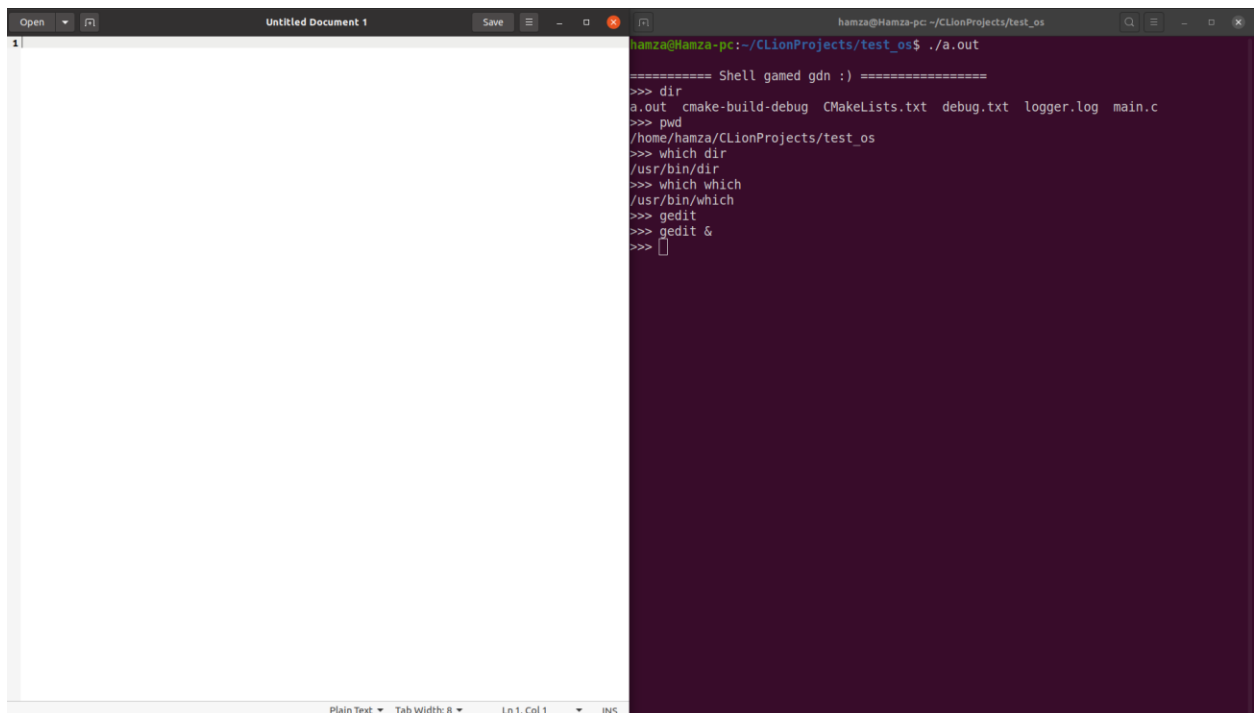
```
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112
```

```
if (!background) {  
    fprintf(debug, format: "Waiting for the child process\n");  
    int status;  
    fprintf(debug, format: "\n**We are in parent now waiting for child\n");  
    while (wait(&status) != child_pid) {  
    }  
} else {  
    fprintf(debug, format: "\n** We are in the parent now\n");  
    getpid();  
    fprintf(debug, format: "Parent will not wait deliberately\n");  
}  
fflush(debug);  
}  
fprintf(debug, format: "Main loop ended \n");  
fprintf(debug, format: "Closing the file.. \n Returning..\n");  
fclose(debug);  
return 0;  
}
```

The terminal window on the right shows the execution of the program:

```
hamza@Hamza-pc:~/CLionProjects/test_os$ ./a.out  
===== Shell gamed gdn :) =====  
>>> dir  
a.out  cmake-build-debug  CMakeLists.txt  debug.txt  logger.log  main.c  
>>> pwd  
/home/hamza/CLionProjects/test_os  
>>> which dir  
/usr/bin/dir  
>>> which which  
/usr/bin/which  
>>>   

```



The screenshot shows the CLion IDE interface with a terminal window. The terminal window displays the execution of the program:

```
hamza@Hamza-pc:~/CLionProjects/test_os$ ./a.out  
===== Shell gamed gdn :) =====  
>>> dir  
a.out  cmake-build-debug  CMakeLists.txt  debug.txt  logger.log  main.c  
>>> pwd  
/home/hamza/CLionProjects/test_os  
>>> which dir  
/usr/bin/dir  
>>> which which  
/usr/bin/which  
>>> gedit  
>>> gedit &  
>>>   

```

Activities
System Monitor
Processes
Resources
File Systems
10:51
بوفمبر 10

Process Name	User	Stat	Virtual Mem %	CPU	Started	ID	Memory	Session	Disk read tot	Disk write
cron	root	Sleep	9.3 MIB	0	Today 7:1	635	172.0 KIB	N/A		
acpid	root	Sleep	2.5 MIB	0	Today 7:1	631	72.0 KIB	N/A		
accounts-daemon	root	Sleep	236.3 MIB	0	Today 7:1	630	100.0 KIB	N/A		
systemd-udev	root	Sleep	23.0 MIB	0	Today 7:1	288	404.0 KIB	N/A		
systemd-journald	root	Sleep	120.2 MIB	0	Today 7:1	258	1.4 MIB	N/A		
dbus-daemon	messagebus	Sleep	8.8 MIB	0	Today 7:1	637	1.7 MIB	N/A		
kerneloops	kernoops	Sleep	11.0 MIB	0	Today 7:1	788	84.0 KIB	N/A		
kerneloops	kernoops	Sleep	11.0 MIB	0	Today 7:1	786	96.0 KIB	N/A		
systemd	hamza	Sleep	20.6 MIB	0	Today 7:1	1044	1.9 MIB	1.4 GIB	672.0 KIB	
gnome-terminal-server	hamza	Sleep	939.6 MIB	0	Today 10:	10798	12.5 MIB	25.3 MIB		
bash	hamza	Sleep	10.6 MIB	0	Today 10:	10809	1.5 MIB	512.2 MIB	28.0 KIB	
a.out	hamza	Sleep	2.4 MIB	0	Today 10:	14906	104.0 KIB	13.2 MIB		
gedit	hamza	Sleep	801.4 MIB	0	Today 10:	15019	18.8 MIB	N/A		
firefox	hamza	Runn	3.0 GIB	0	Today 10:	15285	140.6 MIB	104.0 MIB		
Web Content	hamza	Runn	2.5 GIB	0	Today 10:	15363	135.1 MIB	6.1 MIB		
WebExtens	hamza	Runn	2.3 GIB	0	Today 10:	15412	25.2 MIB	644.0 KIB		
Web Content	hamza	Runn	2.3 GIB	0	Today 10:	15467	34.3 MIB	3.7 MIB		
MainThread	hamza	Runn	2.3 GIB	0	Today 10:	15483	15.9 MIB	N/A		
gnome-shell	hamza	Runn	4.7 GIB	20	Today 7:1	1459	288.6 MIB	243.8 MIB		
nautilus	hamza	Sleep	1.0 GIB	0	Today 10:	14674	31.8 MIB	27.8 MIB		
seahorse	hamza	Sleep	457.5 MIB	0	Today 7:4	5808	17.0 MIB	4.0 MIB		
gnome-calendar	hamza	Sleep	836.9 MIB	0	Today 7:4	5803	14.8 MIB	8.2 MIB		
gvfs-metadata	hamza	Sleep	158.6 MIB	0	Today 7:1	2827	380.0 KIB	520.0 KIB		
gnome-snap-store	hamza	Sleep	1.2 GIB	0	Today 7:1	1749	14.8 MIB	37.3 MIB		
gsd-printer	hamza	Sleep	334.3 MIB	0	Today 7:1	1733	596.0 KIB	124.0 KIB		
gsd-sessions	hamza	Sleep	340.9 MIB	0	Today 7:1	1620	4.7 MIB	608.0 KIB		
gsd-wwan	hamza	Sleep	310.9 MIB	0	Today 7:1	1619	312.0 KIB	44.0 KIB		
gsd-wacom	hamza	Sleep	267.8 MIB	0	Today 7:1	1604	4.5 MIB	212.0 KIB		
gsd-usb-protection	hamza	Sleep	378.9 MIB	0	Today 7:1	1602	428.0 KIB	156.0 KIB		
gsd-sound	hamza	Sleep	314.2 MIB	0	Today 7:1	1598	296.0 KIB	32.0 KIB		
gsd-smartcard	hamza	Sleep	310.7 MIB	0	Today 7:1	1590	340.0 KIB	2.3 MIB		
gsd-sharing	hamza	Sleep	458.0 MIB	0	Today 7:1	1589	544.0 KIB	1.3 MIB		
gsd-screensaver-proxy	hamza	Sleep	230.1 MIB	0	Today 7:1	1588	276.0 KIB	168.0 KIB		
gsd-rfkill	hamza	Sleep	446.5 MIB	0	Today 7:1	1587	288.0 KIB	176.0 KIB		
gsd-print-notifications	hamza	Sleep	242.5 MIB	0	Today 7:1	1586	548.0 KIB	496.0 KIB		
gsd-power	hamza	Sleep	412.8 MIB	0	Today 7:1	1585	1.5 MIB	868.0 KIB		
gsd-media-keys	hamza	Sleep	1.1 GIB	0	Today 7:1	1584	6.2 MIB	3.0 MIB		

```

hamza@Hamza-pc:~/ClionProjects/test_os$ ./a.out
===== Shell gamed gdn :) =====
>>> dir
a.out cmake-build-debug CMakeLists.txt debug.txt logger.log main.c
>>> pwd
/home/hamza/ClionProjects/test_os
>>> which dir
/usr/bin/dir
>>> which which
/usr/bin/which
>>> gedit
>>> gedit &
>>> firefox &

```

End Process

The screenshot shows a code editor window titled "test\_os - logger.log". The editor displays a log file named "logger.log" with the following content:

```
1 Child process in terminated with status = 21909
2 Child process in terminated with status = 21909
3 Child process in terminated with status = 21909
4 Child process in terminated with status = 21909
5 Child process in terminated with status = 21909
6 Child process in terminated with status = 0
7
```

The editor interface includes a sidebar on the left with sections for "Project", "Commit", and "Pull Requests". The top bar shows the file name "test\_os - logger.log" and various menu options like "File", "Edit", "View", "Navigate", "Code", "Refactor", "Build", "Run", "Tools", "VCS", "Window", and "Help". The right sidebar shows a "Database" section.