



---

## **Lab Assignment 1: Shell and System Calls**

---



Name : Remon Hanna Wadie.

ID : 21.

Subject : Operating System (Lab1).

## 1) Organization of the code and the main functions:

the whole program contents from eight methods:

a) The main method which runs the Shell

```
int main(int argc, char* argv[])
{
    signal (SIGCHLD, addLog);
    createLogFile();
    numberOfCommands=0;
    getEnvPaths();
    if(argc>1)
    {
        readFromFile(argv[1]);
    }
    else
    {
        runInteractiveMode();
    }
    return 0;
}
```

if the user send the path of the batch file to the program the Shell will execute the file line by line

else the Shell will run in the interactive mode.

---

## 2) getEnvPaths() method:

```
void getEnvPaths()
{
    char *path;
    path = getenv("PATH");
    char * splitter;
    splitter = strtok (path, ":");
    numberOfPaths=0;
    while (splitter != NULL)
    {
        if(splitter=='\n') continue;
        paths[numberOfPaths++]=splitter;
        splitter = strtok (NULL, ":");
    }
}
```

```
}  
}
```

this method return all environmental paths of the Operating System and save them in an arrays to search in them for the programs that we want to run.

---

3) runInteractiveMode() method:

```
void runInteractiveMode()  
{  
    while(1)  
    {  
        printf("Shell>");  
        background=0;  
        char str[600];  
        fgets(str,600,stdin);  
        int result=executeCommand(&str);  
        if(result==2)  
        {  
            continue;  
        }  
        else if(result==0)  
        {  
            return;  
        }  
    }  
}
```

this method runs the interactive mode of the Shell to allow the user to enter the commands and execute it immediately.

---

4) readFromFile() method:

```
void readFromFile(char*pathOfFile)
{
    batchFile=fopen(pathOfFile,"r");
    if (batchFile == NULL)
    {
        printf("Can not open this file!\n");
        return -1;
    }

    char str [600]; /* or other suitable maximum line
size */
    while ( fgets ( str, sizeof str, batchFile ) != NULL &&
!feof(batchFile)) /* read a line */
    {
        printf("%s",str);
        int result=executeCommand(&str);
        if(result==2)
        {
            continue;
        }
        else if(result==0)
        {
            return;
        }
    }
}
```

this method read the batch file line by line and execute it until reach to exit command or reach to the end of the file.

---

5) split() method:

```
void split(char *str)
{
    numOfArgs=0;
    char * splitter;
```

```

char* commandSplit[512];
spliter = strtok (str, " ");
int i=0;
while (spliter != NULL)
{
    if(spliter=='\n') continue;
    commandSplit[i++]=spliter;
    spliter = strtok (NULL, " \n");
    numOfArgs++;
}
command=commandSplit;
}

```

this method take the command line and make string splitting on it and put splitted strings in an arrays first element is the command and the next elements is the arguments of this command.

---

6) createLogFile() method:

```

void createLogFile()
{
    logFile = fopen("log.txt", "w");
    fclose(logFile);
}

```

this method creates log.txt file in the same directory of the project.

---

7) addLog() method:

```

void addLog(int signal)
{
    logFile = fopen("log.txt", "a");
    fprintf(logFile,"A child process %d is
finished.\n",numberOfChildFinished);
    numberOfChildFinished++;
    fclose(logFile);
}

```

this method is used for add new log in the file if a child process is terminated.

---

8) executeCommand() method:

```
int executeCommand(char*str)
{
    int command_Found=0;
    strcpy(history[numberOfCommands],str);
    numberOfCommands++;
    int i=0;
    background=0;
    command_Found=0;
    for(i=0; i<strlen(str); i++)
    {
        if(str[i]=='&')
        {
            background=1;
            str[i]=' ';
        }

        //Backspace \a, Form feed \f, Carriage return \r, Horizontal
        tab \t, Vertical tab \v

        if(str[i]=='\t' || str[i]=='\r' || str[i]=='\a' || str[i]=='\b' || str[i]=='\f' || str[i]=='\v')
        {
            str[i]=' ';
        }
    }

    if(strlen(str)>=512)
    {
        printf("Invalid too large command\n");
        return 2; //continue
    }
    if(strcmp(str,"\n")==0)return 2; //continue
    int foundEqual=0;
    for(i=0; i<strlen(str); i++)
    {
        if(str[i]=='=')
```

```

        {
            foundEqual=1;
            break;
        }
    }

    char * spliter;
    char* expression[512];
    if(foundEqual)
    {
        spliter = strtok (str," =");
        int i=0;
        while (spliter != NULL)
        {
            if(spliter=='\n') continue;
            expression[i++]=spliter;
            spliter = strtok (NULL, " \n=");
        }
        setenv(expression[0],expression[1],1);
        return 2; //continue
    }

    numOfArgs=1;
    int i2=0;
    int length=strlen(str);
    while(str[i2]==' ')i2++;
    while(i2<length)
    {
        if(str[i2]==' ' && str[i2+1]!=' ' && str[i2]!='\0' && i2+1<length
        && str[i2+1]!='\n')
        {
            numOfArgs++;
        }
        i2++;
    }

    split(str);
    if(command[0][0]=='#') return 2; //continue

```

```
char type[50];
strcpy(type,command[0]);
int j=0;
while(j<50)
{
    if(type[j]=='\n')
    {
        type[j]='\0';
        break;
    }
    j++;
}

command[numOfArgs]=NULL;
pid_t pid;
if(strcmp("exit",type)==0 || strcmp("Ctrl-D",type)==0)
{
    printf("Goodbye!\n");
    return 0;//terminate
}
else if(strcmp(type,"history")==0)
{
    i=0;
    for(; i<numberOfCommands; i++)
    {
        printf("%s",history[i]);
    }
    return 2; //continue
}
char *program=command[0];
j=0;
while(j<50)
{
    if(program[j]=='\n')
    {
        program[j]='\0';
        break;
    }
}
```



```

        j++;
    }

    if(strcmp(program,"cd")!=0)
    {
        for(i=0; i<numberOfPaths; i++)
        {
            strcpy(type,paths[i]);
            strcat(type,"/");
            strcat(type,command[0]);
            int j=0;
            while(j<50)
            {
                if(type[j]=='\n')
                {
                    type[j]='\0';
                    break;
                }
                j++;
            }
            int result=access(type,X_OK);
            if(result==0)
            {
                pid=fork();
                command_Found=1;
                break;
            }
        }
        if(command[0][0]=='/')
        {
            strcpy(type,command[0]);
            command_Found=1;
            pid=fork();
        }
    }
    else if(strcmp(program,"cd")==0)
    {
        chdir(command[1]);
    }
}

```

```

    return 22; //continue
}
if(strcmp(command[0],"echo")==0)
{
    command_Found=1;
    int i=0;
    int j=0;
    while(i<strlen(command[1]))
    {

        char c=command[1][i];
        if(c!='"' && c!='\\')
            newStr1[j++]=c;
        i++;
    }
    newStr1[i++]='\0';
    command[1]=newStr1;

    i=0;
    j=0;
    while(i<strlen(command[numOfArgs-1]))
    {
        char c=command[numOfArgs-1][i];
        if(c!='"' && c!='\\')
            newStr2[j++]=c;
        i++;
    }
    newStr2[i++]='\0';
    command[numOfArgs-1]=newStr2;

    i=1;
    for(; i<numOfArgs; i++)
    {
        if(command[i][0]=='$')
        {
            char *env;

            int n=0;

```

```

        char * splitter;
        char* splited[50];
        splitter = strtok (command[i], " $\n");
        int i=0;
        while (splitter != NULL)
        {
            if(splitter=="\n") continue;
            splited[i++]=splitter;
            splitter = strtok (NULL, " $\n");
            n++;
        }
        env=splited[0];
        command[i]=getenv(env);
    }
}

int status;
if(command_Found)
{
    if(pid!=0)
    {
        if(!background)
        {
            waitpid(pid, &status, 0);
        }
    }
    else
    {
        if(type[strlen(type)-1]=='/'||type[strlen(type)-1]=='\\')return
0;
        execv(type, command);
        perror("execv");
        return 2;
    }
    command_Found=0;
}
else
{

```

```
        fprintf(stderr, "Command not found!\n");  
    }  
}
```

this method is used to execute a line that sent to it.

2) Compiling and running the code:

to compile the source code Shell.c :

1)create a make file:

gcc Shell.c -o Shell

2)In the terminal run the make file by these commands:

chmod +x make

./make

3) run the Shell program:

./Shell


### 3)Sample runs:

```
remon@remon: ~/Desktop
remon@remon:~/Desktop$ ./Shell
Shell>ls
10615322_706382549453070_2929406405072117231_n.jpg  make
aa.mm~                                              make~
adt-bundle-linux-x86_64-20140702                  MP
course-materials                                  password.txt~
file.sh                                             play.txt~
file.sh~                                           shell
log.txt                                             shell.c
Shell>ps
  PID TTY          TIME CMD
 22208 pts/13    00:00:00 bash
 22655 pts/13    00:00:00 Shell
 22660 pts/13    00:00:00 ps
Shell>cal
      November 2014
Su Mo Tu We Th Fr Sa
                1
 2   3   4   5   6   7   8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
Shell>date
Sun Nov  2 02:28:12 EET 2014
Shell>echo hello
hello
Shell>echo "hello world!"
hello world!
Shell>echo 'hello form Shell'
hello form Shell!
Shell>
```

```

Shell>date
Sun Nov  2 02:28:12 EET 2014
Shell>echo hello
hello
Shell>echo "hello world!"
hello world!
Shell>echo 'hello form Shell'
hello form Shell!
Shell>remon = 50118833
Shell>echo remon
remon
Shell>echo $remon
50118833
Shell>ps
  PID TTY          TIME CMD
 22208 pts/13        00:00:00 bash
 22655 pts/13        00:00:00 Shell
 22768 pts/13        00:00:00 ps
Shell>

```

CPU		remon	140 K	100 K
▼ sh		remon	104 K	552 K
▼ Shell		remon	344 K	472 K
 firefox		remon	108,420 K	46,092 K Mozilla Firefox
firefox		remon	zombie	
firefox		remon	zombie	

```

Shell>firefox &
Shell>
(process:30290): GLib-CRITICAL **: g_slice_set_config: assertion 'sys_page_size == 0' failed

(firefox:30290): GLib-GObject-WARNING **: Attempt to add property GnomeProgram::sm-connect after class was initialised

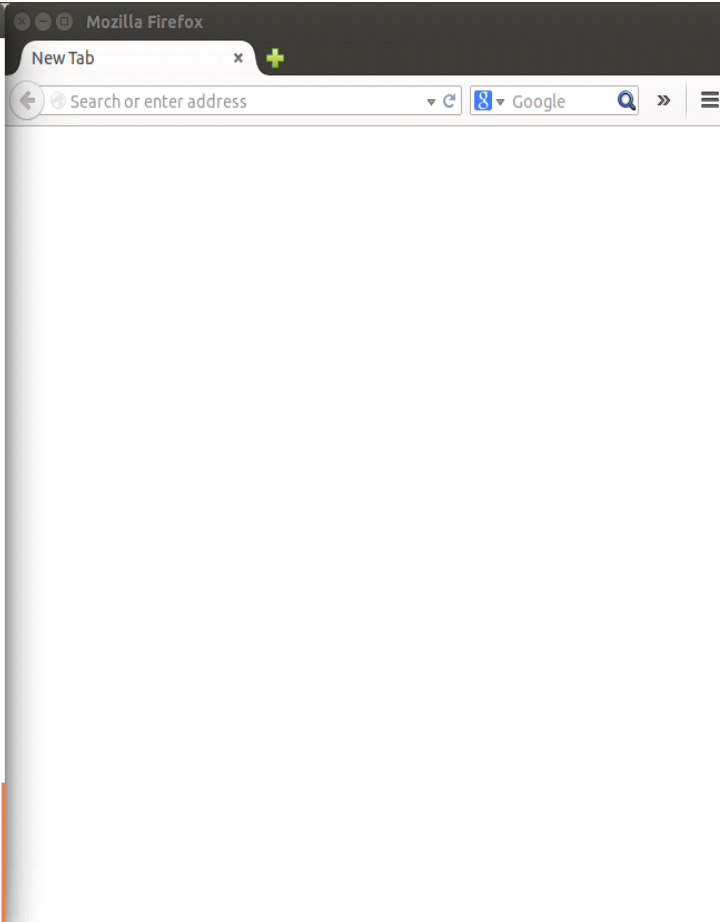
(firefox:30290): GLib-GObject-WARNING **: Attempt to add property GnomeProgram::show-crash-dialog after class was initialised

(firefox:30290): GLib-GObject-WARNING **: Attempt to add property GnomeProgram::display after class was initialised

(firefox:30290): GLib-GObject-WARNING **: Attempt to add property GnomeProgram::default-icon after class was initialised

Shell>
Shell>ps
  PID TTY          TIME CMD
 29825 pts/13    00:00:00 cb_console_runn
 29826 pts/13    00:00:00 sh
 29827 pts/13    00:00:00 Shell
 29941 pts/13    00:00:02 firefox <defunct>
 30216 pts/13    00:00:01 firefox <defunct>
 30290 pts/13    00:00:01 firefox
 30347 pts/13    00:00:00 ps
Shell>cd /home/remon/Desktop
Shell>pwd
/home/remon/Desktop
Shell>

```



▼ sh	remon	104 K	552 K
▼ Shell	remon	252 K	472 K
firefox	remon	108,220 K	47,784 K Mozilla Firefox
firefox	remon	zombie	

File Edit View Search Terminal Help

```
ihell>  
ihell>  
ihell>  
ihell>firefox
```

```
[process:30040]: GLib-CRITICAL **: g_slice_set_config: assertion 'sys_page_size == 0' failed
```

```
[firefox:30040]: GLib-GObject-WARNING **: Attempt to add property GnomeProgram::sm-connect after class was initialised
```

```
[firefox:30040]: GLib-GObject-WARNING **: Attempt to add property GnomeProgram::show-crash-dialog after class was initialised
```

```
[firefox:30040]: GLib-GObject-WARNING **: Attempt to add property GnomeProgram::display after class was initialised
```

```
[firefox:30040]: GLib-GObject-WARNING **: Attempt to add property GnomeProgram::default-icon after class was initialised
```

```
|
```

Mozilla Firefox

New Tab

Search or enter address

Google