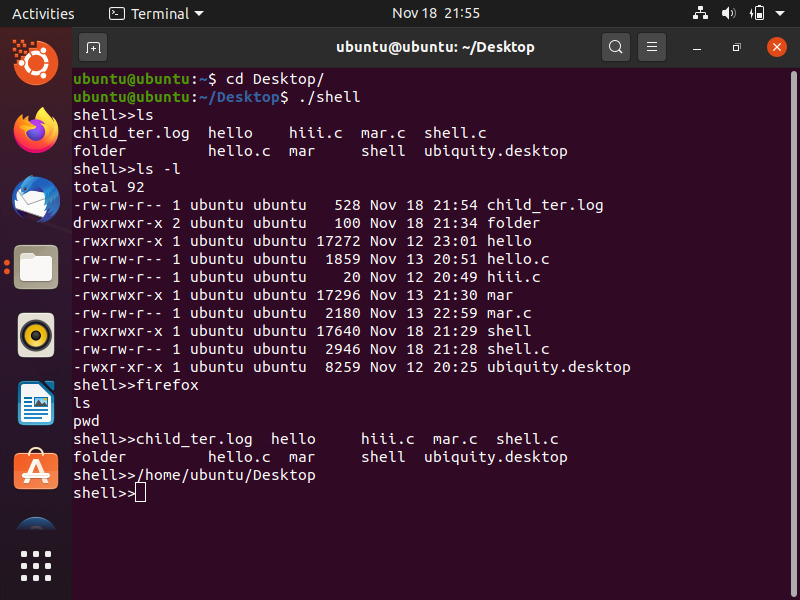
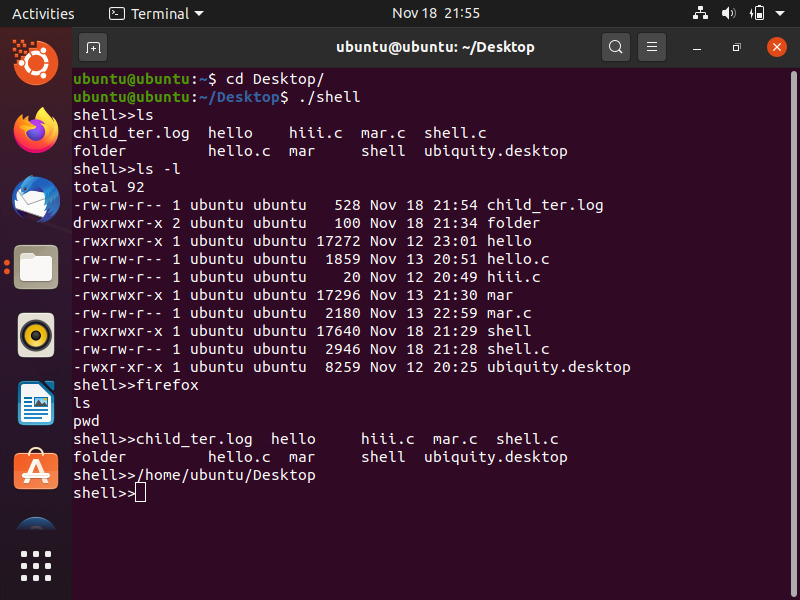
**Lab 1**

**Operation Systems**

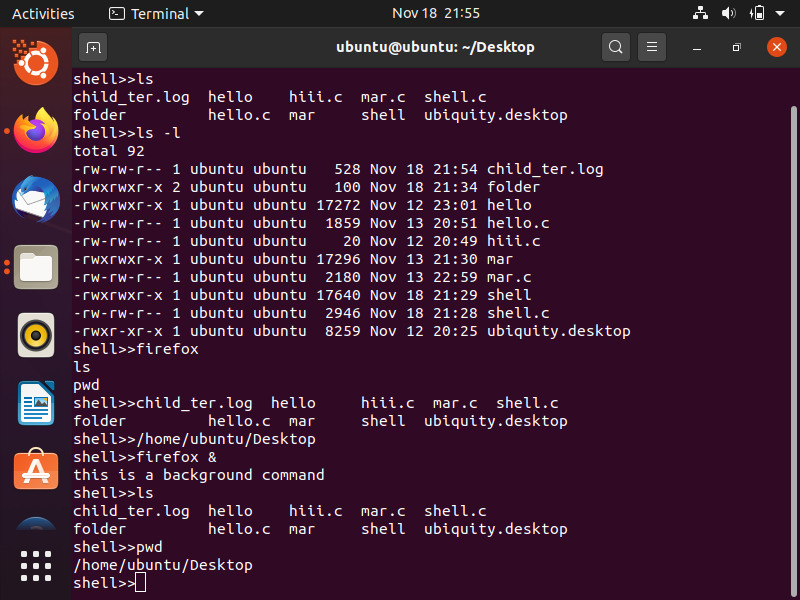
**Name: Toka Sherif Id:5492**



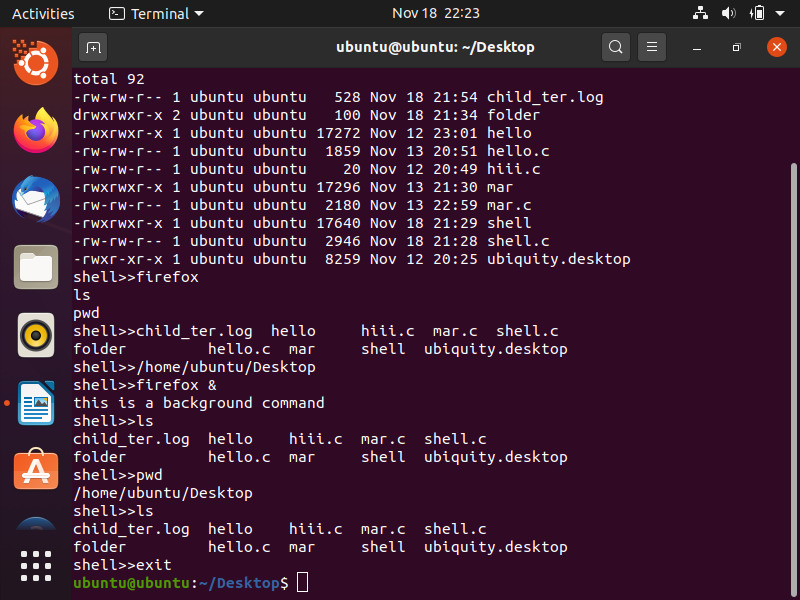
1. Performing command with no arguments



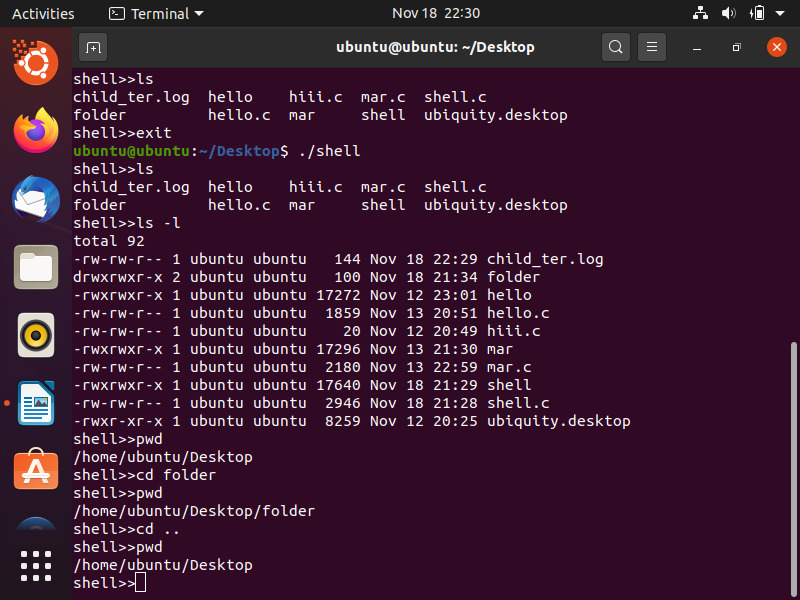
1. Performing command with arguments



1. Performing command in foreground
2. Performing command in background



5)performing exit command



6)performing cd command

**Code:**

#include <stdio.h>

#include <unistd.h>

#include <string.h>

#include<sys/wait.h>

#include<errno.h>

#include <stdlib.h>

#include <signal.h>

#include <sys/types.h>

#include <time.h>

int forking(int flag ,char\* argv[])

{

pid\_t pid;

pid = fork();

if(pid >= 0)

{

if(pid == 0)

{

if(execvp(argv[0], argv)<0)

{

perror("\nError");

exit(1);

//exit(-1);

}

}

else if(pid > 0)

{

if(flag==0)

{

waitpid(pid,NULL,0);

}

else

{

flag=0;

return flag ;

}

}

}

else

{

perror("\nError in fork");

exit(1);

}

}

void file\_handler(int sig)

{

/\*FILE \* f;

f = fopen ("data.txt","a");

fprintf (f, "The process terminated\n");

fclose(f);\*/

time\_t current\_time;

char\* c\_time\_string;

/\* Obtain current time. \*/

current\_time = time(NULL);

if (current\_time == ((time\_t)-1))

{

(void) fprintf(stderr, "Failure to obtain the current time.\n");

exit(EXIT\_FAILURE);

}

/\* Convert to local time format. \*/

c\_time\_string = ctime(&current\_time);

if (c\_time\_string == NULL)

{

(void) fprintf(stderr, "Failure to convert the current time.\n");

exit(EXIT\_FAILURE);

}

FILE \*fp ;

fp = fopen("child\_ter.log","a");

fprintf(fp,"%s My child terminated \n ", c\_time\_string);

fclose(fp);

}

int cd(char \*path) {

return chdir(path);

}

int main()

{

//reading line

char str[80];

char\* command\_argument[80] ;

//tokenization

const char s[2] =" ";

char \*token;

//termination

int state;

while(1)

{

int bg = 0 ;

int counter = 0 ;

printf("shell>>");

gets(str);

//("%[^\n]",str);

if(strcasecmp(str,"exit") == 0)

{

exit(0);

}

token = strtok(str, s);

while( token != NULL )

{

command\_argument[counter] = token ;

token = strtok(NULL, s);

counter ++ ;

}

//check in background

if(strcmp(command\_argument[counter-1],"&") == 0)

{

command\_argument[counter-1]=NULL;

printf("this is a background command\n");

bg = 1 ;

counter--;

}

//check cd command

else if(strcmp(command\_argument[0], "cd") == 0){

if (cd(command\_argument[1]) < 0) {

perror(command\_argument[1]);

}

continue;

}

else{

command\_argument[counter]=NULL;

}

signal(SIGCHLD, file\_handler);

bg = forking (bg,command\_argument);

}

}