**Bahria University, Lahore Campus**

Department of Computer Sciences

Lab Journal 05

**(Spring 2024)**

|  |  |  |
| --- | --- | --- |
| Course: | **Operating System Lab** | Date: 21-3-2024 |
| Course Code: | CSL-320 | Max Marks: 20 |
| Faculty’s Name: | Abdullah |  |

Name: Muhammad Hammad Enroll No: 03-134221-024

**Objective(s):**

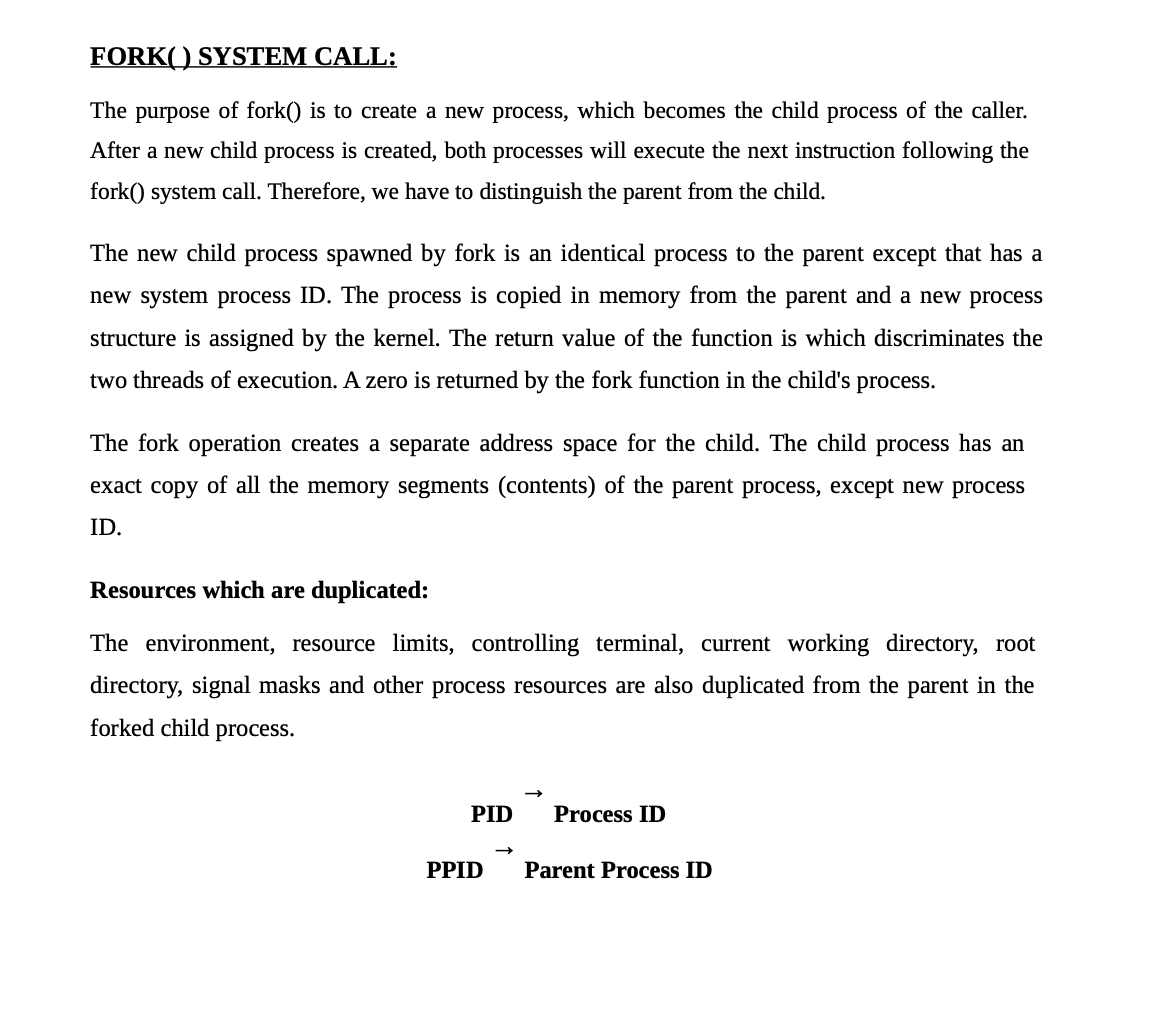
* To write a program to create a process in LINUX.
* To understand exec process.
* To create child with sleep and wait command.
* To understand getpid( ) and getppid( ).

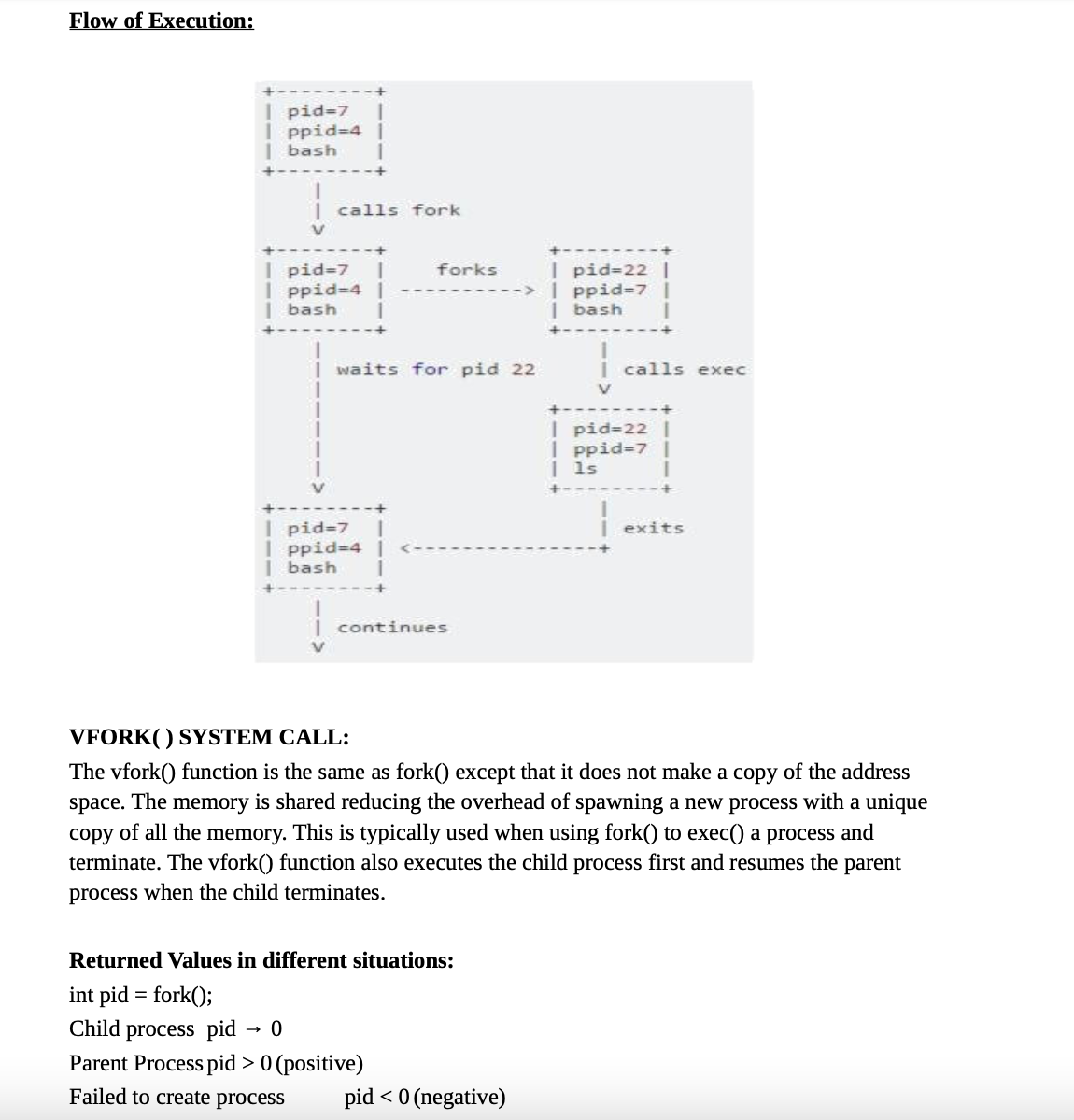
**Tool(s) used:**

Ubuntu, VIM Editor

**Lab Grading Sheet :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Max Marks** | **Obtained Marks** | **Comments(*if any*)** |
| 1. | 05 |  |  |
| 2. | 05 |  |  |
| 3. | 05 |  |  |
| 4. | 05 |  |  |
| **Total** | **20** |  | **Signature** |

****

****

**Task 1:** Write the program for process creation using fork command.

**STEP 1:** Start the program.

**STEP 2:** Declare pid as integer.

**STEP 3:** Create the process using Fork command.

**STEP 4:** Check pid is less than 0 then print error else if pid is equal to 0 then execute command else parent process wait for child process.

**STEP 5:** Exit the process.

**Program**#include<iostream>

#include <unistd.h>

using namespace std;

int main()

{

int pid = fork();

if(pid<0)

{

cout<<"Error" <<endl; //will print error

}

else if(pid==0)

{

cout<<"Child procces";

//will execute child procces

}

else

{

cout<<"Parent procces";

//will execute parent procces

}

return 0;

}

**Program Execution   
Task 1:** Write the program for process creation using fork command.

**STEP 1:** Start the program.

**STEP 2:** Declare pid as integer.

**STEP 3:** Create the process using Fork command.

**STEP 4:** Check pid is less than 0 then print error else if pid is equal to 0 then execute command else parent process wait for child process.

**STEP 5:** Exit the process.

**OUTPUT**

A screenshot of a computer

Description automatically generated

**Task 1.2:** Write a program illustrating the sleep command during process creation.

**STEP 1:** Start the program.

**STEP 2:** Create process using fork and assign into a variable.

**STEP 3:** If the value of variable is < zero print not create and > 0 process create and else print child create.

**STEP 4:** Create child with sleep of 20.

**STEP 5:** Stop the program.

**Program**

#include<iostream>

#include <unistd.h>

using namespace std;

int main()

{

int pid = fork();

if(pid<0)

{

cout<<"Error" <<endl; //will print error

}

else if(pid>0)

{

cout<<"Parent procces";

//will execute parent procces

}

else

{

sleep(20);

cout<<"Child procces : ";

cout<<"Child is asleep";

//will execute child procces

}

return 0;

} **Program Execution**

**STEP 1:** Start the program.

**STEP 2:** Create process using fork and assign into a variable.

**STEP 3:** If the value of variable is < zero print not create and > 0 process create and else print child create.

**STEP 4:** Create child with sleep of 20.

**STEP 5:** Stop the program.

**OUTPUT**

A black background with white text

Description automatically generated

**Task 1.3:** Write a program illustrating the wait command during process creation.

**STEP 1**: Start the execution  
**STEP 2**: Create process using fork and assign it to a variable  
**STEP 3**: Check for the condition pid is equal to 0  
**STEP 4**: If it is true print the value of i and terminate the child process   
**STEP 5**: If it is not a parent process has to wait until the child terminate   
**STEP 6**: Stop the execution

**Program**

#include<iostream>

#include <unistd.h>

#include <sys/wait.h>

using namespace std;

int main()

{

int pid = fork();

int i = 5;

if(pid<0)

{

cout<<"Error" <<endl; //will print error

}

else if(pid==0)

{

cout<<"Child procces " <<endl;

//will execute child procces

cout<<"Value of I is : " << i <<endl ;

}

else

{

cout<<"Parent procces";

//will execute parent procces

wait(NULL);

}

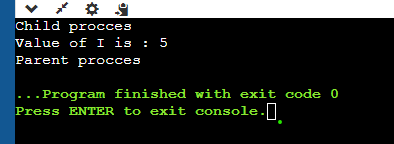
return 0;

}

**Program Execution**

**STEP 1**: Start the execution  
**STEP 2**: Create process using fork and assign it to a variable  
**STEP 3**: Check for the condition pid is equal to 0  
**STEP 4**: If it is true print the value of i and terminate the child process   
**STEP 5**: If it is not a parent process has to wait until the child terminate   
**STEP 6**: Stop the execution

**OUTPUT**



**Task 1.4** Write the output of a program illustrating the wait command during process creation. **Algorithm**

**STEP 1**: Start the execution  
**STEP 2**: Create process using fork and assign it to a  
variable  
**STEP 3**: Check for the condition pid is equal to 0  
**STEP 4**: If it is true print the value of i and terminate the child process  
**STEP 5**: If it is not a parent process has to wait until the child terminate  
**STEP 6**: Stop the execution

**Program**

#include<iostream>

#include <unistd.h>

#include <sys/wait.h>

using namespace std;

int main()

{

int pid = fork();

int i = 5;

if(pid<0)

{

cout<<"Error" <<endl; //will print error

}

else if(pid==0)

{

cout<<"Child procces " <<endl;

//will execute child procces

cout<<"Value of I is : " << i <<endl ;

}

else

{

cout<<"Parent procces";

//will execute parent procces

wait(NULL);

}

return 0;

}

**Program Execution**

**STEP 1**: Start the execution  
**STEP 2**: Create process using fork and assign it to a variable  
**STEP 3**: Check for the condition pid is equal to 0  
**STEP 4**: If it is true print the value of i and terminate the child process   
**STEP 5**: If it is not a parent process has to wait until the child terminate   
**STEP 6**: Stop the execution

**OUTPUT**

**A screenshot of a computer

Description automatically generated**

**Exec System Call:**

Exec system call replaces the contents and data segments of the currently running process with the information from the program file whose name is passed as a parameter to exec( ).

**Task 2** Write the output of a program for execution of ls command using exec.

**Algorithm  
STEP 1:** Start the program.

**STEP 2:** Execute the command in the shell program using exec ls.

**STEP 3:** Stop the execution.

**Program**

echo “Program for executing LINUX command using Shell Programming”   
echo “Welcome”  
ps  
exec ls

page59image57780384page59image57783920

**OUTPUT**

A black screen with white text

Description automatically generated

**Task 3**

Create a program for getting the pid and ppid while and use the sleep command.

**Algorithm**

**STEP 1:** Start the execution and create a process using fork( ) command.  
**STEP 2:** Make the parent process to sleep for 10 seconds.   
**STEP 3:** In the child process print it pid and it corresponding pid.

**STEP 4:** Make the child process to sleep for 5 seconds.  
**STEP 5:** Again print it pid and it parent pid.  
**STEP 6:** After making the sleep for the parent process for 10 seconds print it pid.  
**STEP 7:** Stop the execution.

**Program**

#include<iostream>

#include <unistd.h>

#include <sys/wait.h>

using namespace std;

int main()

{

int pid = fork();

if(pid<0)

{

cout<<"Error" <<endl; //will print error

}

else if(pid==0)

{

cout<<"Child procces " <<endl;

cout<<"Child pid = " <<getpid() << " Corresponsdin pid = " << getppid()<<endl;

sleep(5);

cout<<"Child pid after child woke up = " <<getpid() <<endl;

cout<<"Parent pid = " <<getppid() <<endl;

}

else

{

cout<<"Parent procces"<<endl;

sleep(10);

cout<<"Parent pid after parent woke up = " <<getppid()<<endl;

}

return 0;

}

**Program Execution**

**STEP 1:** Start the execution and create a process using fork( ) command.  
**STEP 2:** Make the parent process to sleep for 10 seconds.   
**STEP 3:** In the child process print it pid and it corresponding pid.

**STEP 4:** Make the child process to sleep for 5 seconds.  
**STEP 5:** Again print it pid and it parent pid.  
**STEP 6:** After making the sleep for the parent process for 10 seconds print it pid.  
**STEP 7:** Stop the execution.

**OUTPUT**

**A screenshot of a computer screen

Description automatically generated**

**Task 4** Write a program in C to create two process Parent and child through fork.

* Differentiate between parent and child through conditional statements.
* Parent: should ask the user a number and then print its table 1st 10 times.
* Child: should take a number n from user and print its Square.
* Both processes should display their IDs and parent ID/Child ID once.

**Program**

#include<iostream>

#include <unistd.h>

#include <sys/wait.h>

using namespace std;

int main()

{

int pid = fork();

int num;

if(pid<0)

{

cout<<"Error" <<endl; //will print error

}

else if(pid==0)

{

cout<<"Child procces " <<endl;

cout<<"Child pid = " <<getpid() << " Parent pid = " << getppid() <<endl;

cout<<"Enter a number for Child procces : ";

cin>>num;

cout << "Square of " << num << " is " << num \* num << endl;

}

else

{

cout<<"Parent procces"<<endl;

cout<<"Parent pid = " <<getpid() << " Child pid = " <<getppid() <<endl;

cin.ignore();

cout<<"Enter a number for parent procces : ";

cin>>num;

for(int i =1; i<= num ;i++)

{

cout<< num << " x " << i << " = " << num\*i <<endl;

}

wait(NULL);

}

return 0;

}

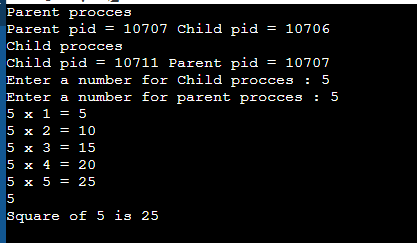
**Program Execution**

**STEP 1:** Start the execution and create a process using fork( ) command.  
**STEP 2:** Display Parent Id and Child Id in Parent Process   
**STEP 3:** Display Parent Id and Child Id in Child Process

**STEP 4:** Take Input For Parent Process and make its table upto 10 and display table.

**STEP 5:** Take Input For Childe Process and Square it and display   
**STEP 6:** Stop the execution.

**OUTPUT**

****