**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: \_AFFAN AHMAD\_\_\_\_\_\_ Enroll No: \_03-134221-003\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Objective(s):**

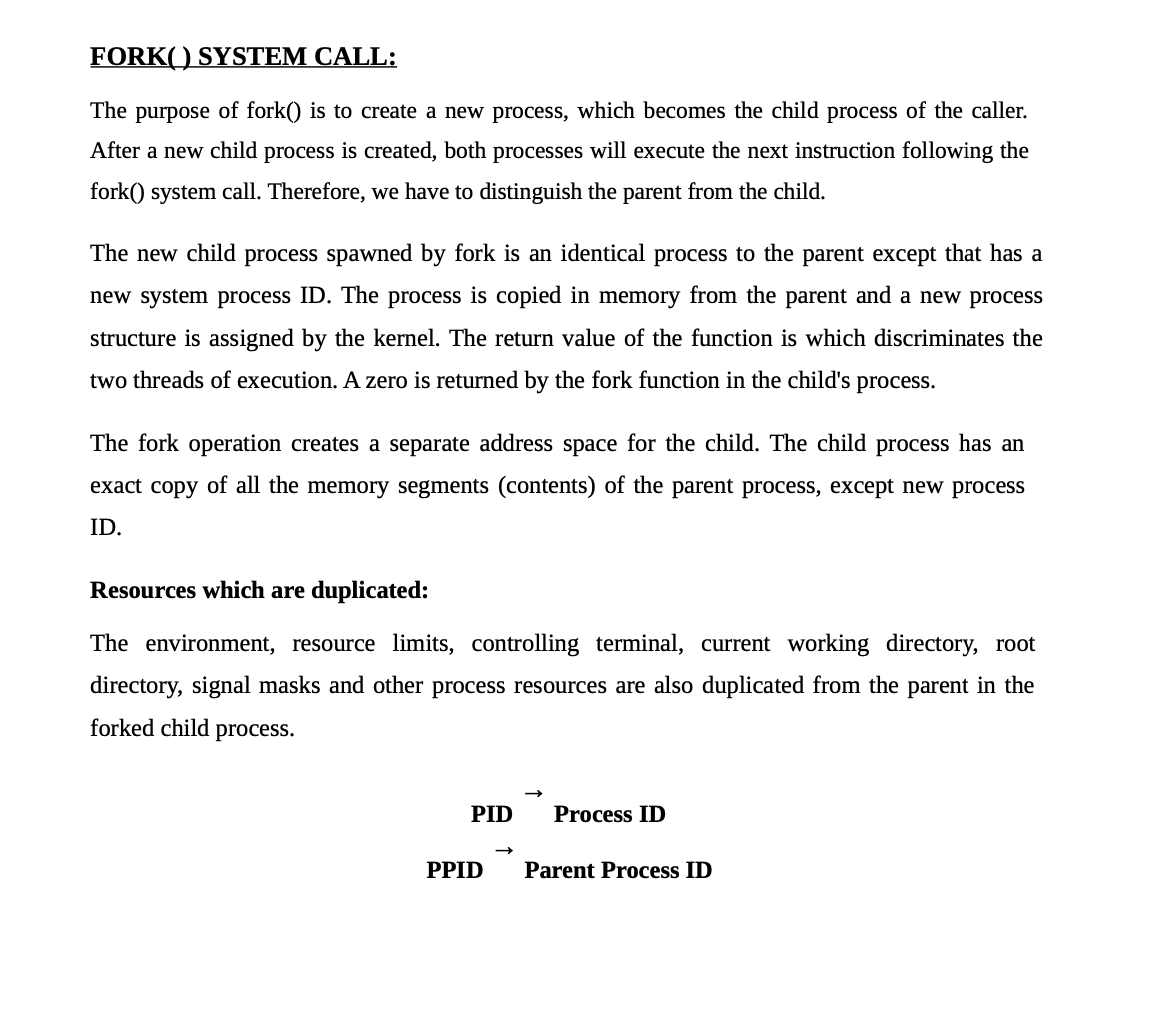
* To write a program to create a process in LINUX.
* To understand exec process.
* To create child with sleep and wait command.
* To understand getpid( ) and getppid( ).

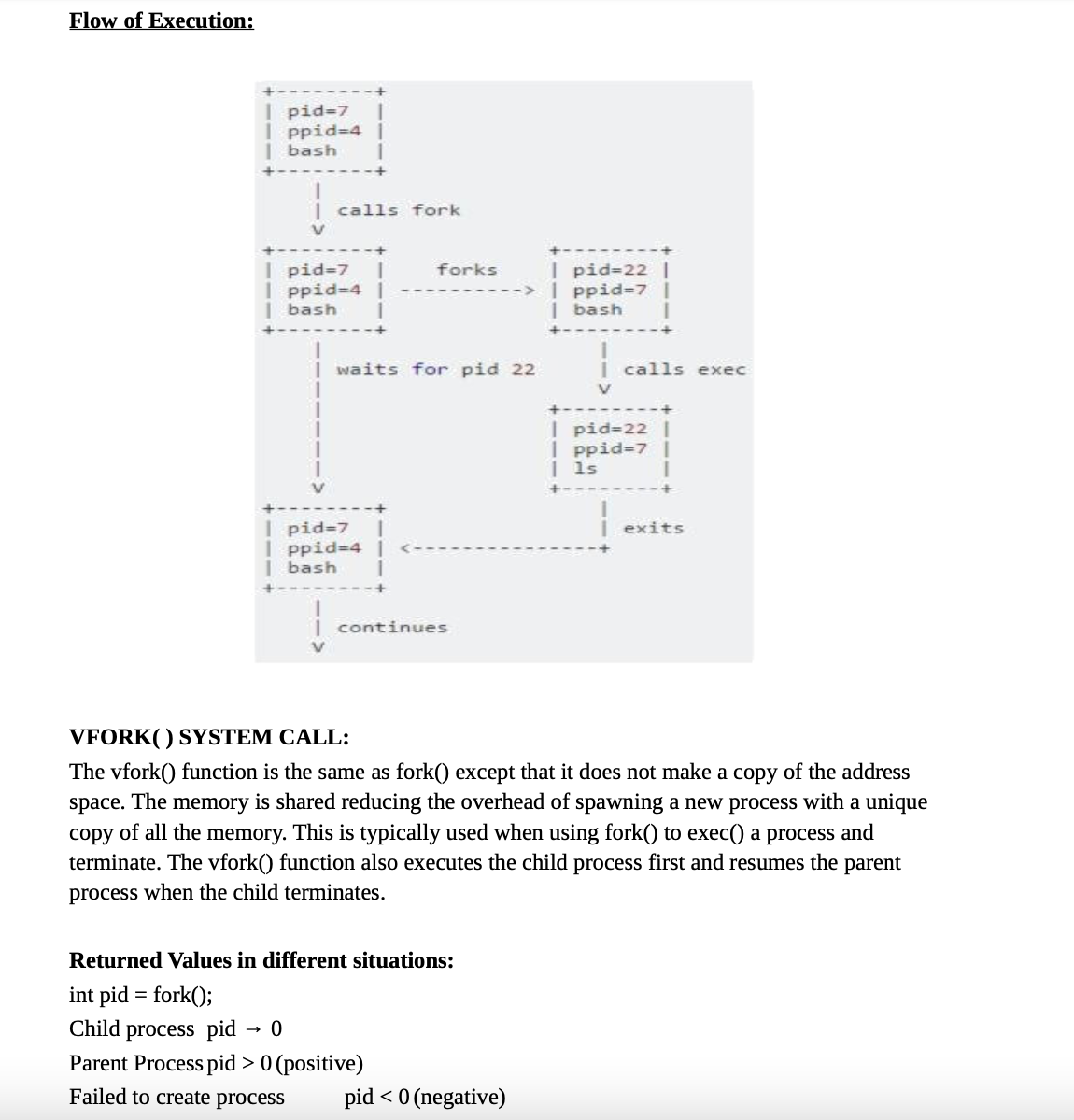
**Tool(s) used:**

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()**

**{**

**pid\_t fork(void);**

**pid\_t c\_pid = fork();**

**if (c\_pid == -1) {**

**perror("fork");**

**exit(EXIT\_FAILURE);**

**}**

**else if (c\_pid > 0) {**

**cout << "printed from parent process " << getpid() << endl;**

**}**

**else {**

**cout << "printed from child process " << getpid() << endl;**

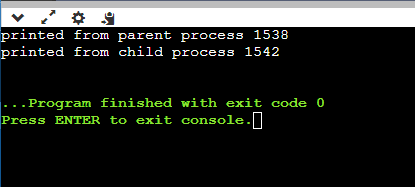
**}**

**return 0;**

**}**

**Program Execution**

**OUTPUT**



**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.

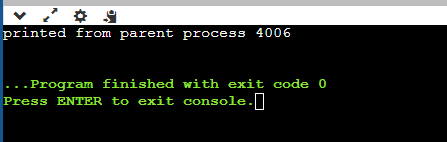
**STEP 4:** Create child with sleep of 20.

**STEP 5:** Stop the program.

**Program**

**Program Execution**

**OUTPUT**



**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>**

**using namespace std;**

**#include<stdlib.h>**

**#include<sys/wait.h>**

**#include<unistd.h>**

**int main()**

**{**

**pid\_t cpid;**

**if (fork()== 0)**

**exit(0);**

**else**

**cpid = wait(NULL);**

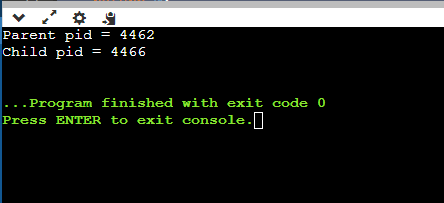
**cout <<"Parent pid = "<< getpid()<< endl;**

**cout << "Child pid = "<< cpid<< endl;**

**return 0;**

**}  
Program 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>**

**using namespace std;**

**#include<stdlib.h>**

**#include<sys/wait.h>**

**#include<unistd.h>**

**int main()**

**{**

**pid\_t cpid;**

**if (fork()== 0)**

**exit(0);**

**else**

**cpid = wait(NULL);**

**cout <<"Parent pid = "<< getpid()<< endl;**

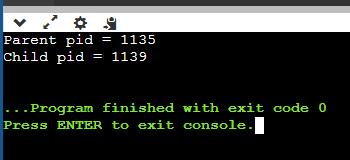
**cout << "Child pid = "<< cpid<< endl;**

**return 0;**

**}**

**Program Execution**

**OUTPUT**



**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.

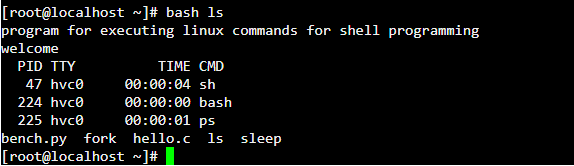
**STEP 3:** Stop the execution.

**Program**

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

page59image57780384page59image57783920

**OUTPUT**



**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>**

**using namespace std;**

**#include <unistd.h>**

**#include <sys/types.h>**

**int main() {**

**pid\_t child\_pid;**

**child\_pid = fork();**

**if (child\_pid < 0) {**

**cout << "Fork failed." << endl;**

**return 1;**

**} else if (child\_pid == 0) {**

**cout << "Child process: PID = " << getpid() << ", PPID = " << getppid() << endl;**

**sleep(5);**

**cout << "Child process after sleep: PID = " << getpid() << " PPID = " << getppid() << endl;**

**} else {**

**sleep(10);**

**cout << "Parent process after sleep: PID = " << getpid() << std::endl;**

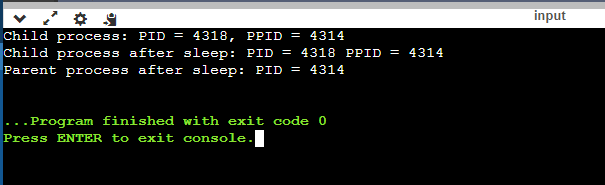
**}**

**return 0;**

**}**

**Program Execution**

**OUTPUT**



**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 <stdio.h>**

**#include <unistd.h>**

**#include <sys/types.h>**

**int main() {**

**pid\_t child\_pid;**

**child\_pid = fork();**

**if (child\_pid < 0) {**

**fprintf(stderr, "Fork failed.\n");**

**return 1;**

**} else if (child\_pid == 0) {**

**printf("Child process: PID = %d, PPID = %d\n", getpid(), getppid());**

**int n;**

**printf("Enter a number: ");**

**scanf("%d", &n);**

**printf("Square of %d is: %d\n", n, n \* n);**

**} else {**

**printf("Parent process: PID = %d\n", getpid());**

**int num;**

**printf("Enter a number: ");**

**scanf("%d", &num);**

**printf("Table of %d:\n", num);**

**for (int i = 1; i <= 10; ++i) {**

**printf("%d x %d = %d\n", num, i, num \* i);**

**}**

**}**

**return 0;**

**}**

**Program Execution**

**OUTPUT**

