Exp No:2 Title of the Exercise: Process Creation using fork()

and Usage of getpid(), getppid(), wait() functions

Date:30/08/2022

Aim: --

Process Creation using fork() and Usage of getpid(), getppid(), wait() functions.

Procedure:--

1)Fork():-

fork() system call is used to create child processes in a C program. fork() is used where parallel processing is required in your application. The fork() system function is defined in the headers sys/types.h and unistd.h

2)getpid():-

It returns the process ID (PID) of the calling process.

3)getppid (): -

It returns the process ID of the parent of the calling process.

4)Wait (): -

The wait () system call suspends execution of the calling thread until one of its children terminates.

Program: -

#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

#include <stdlib.h>

int main(int argc, char \*argv[])

{

printf("I am: %d\n", (int) getpid());

pid\_t pid = fork();

printf("fork returned: %d\n", (int) pid);

if (pid < 0) {

perror("Fork failed");

}

if (pid == 0) {

printf("I am the child with pid %d\n", (int) getpid());

printf("Child process is exiting\n");

exit(0);

}

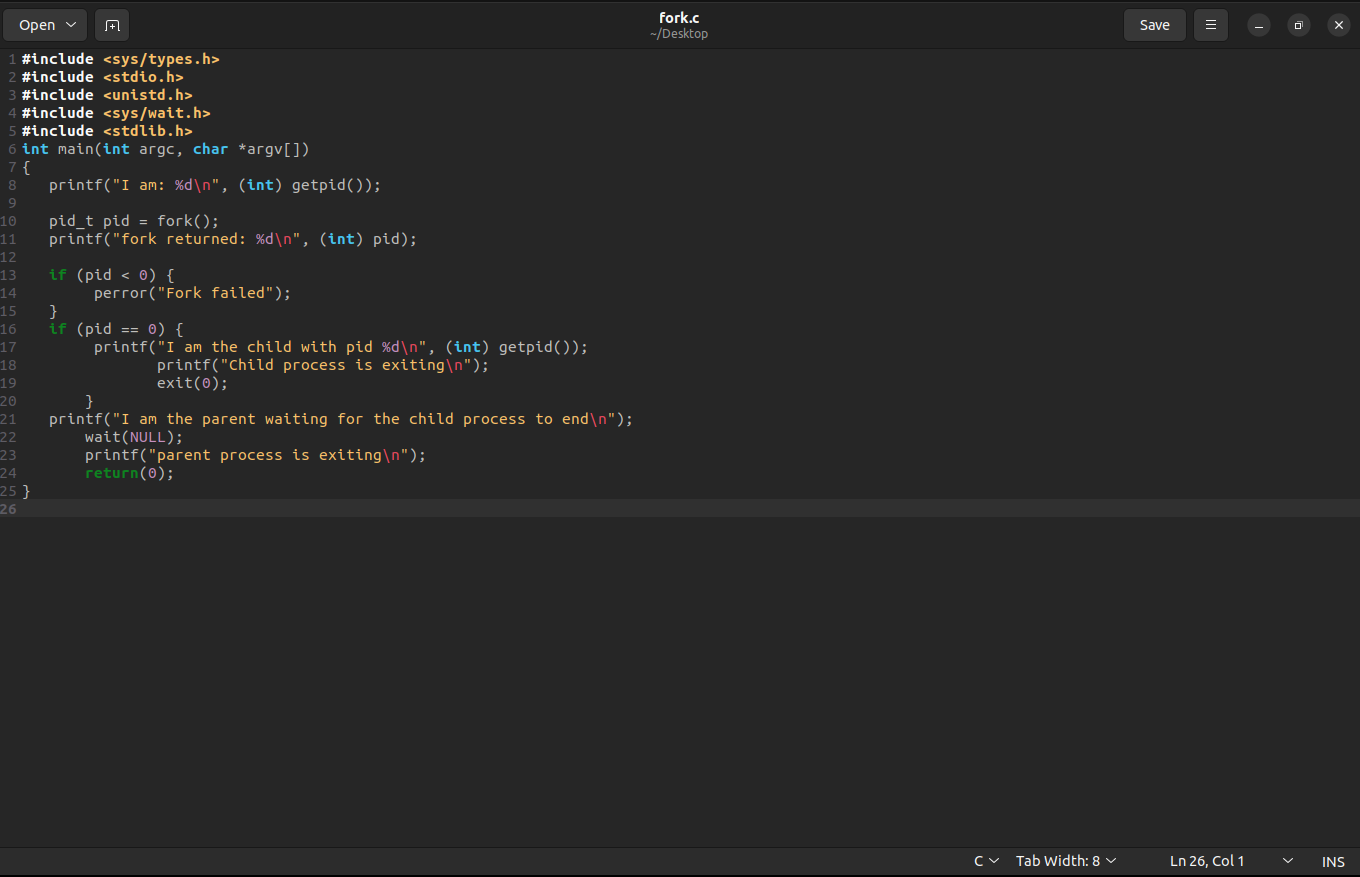
printf("I am the parent waiting for the child process to end\n");

wait(NULL);

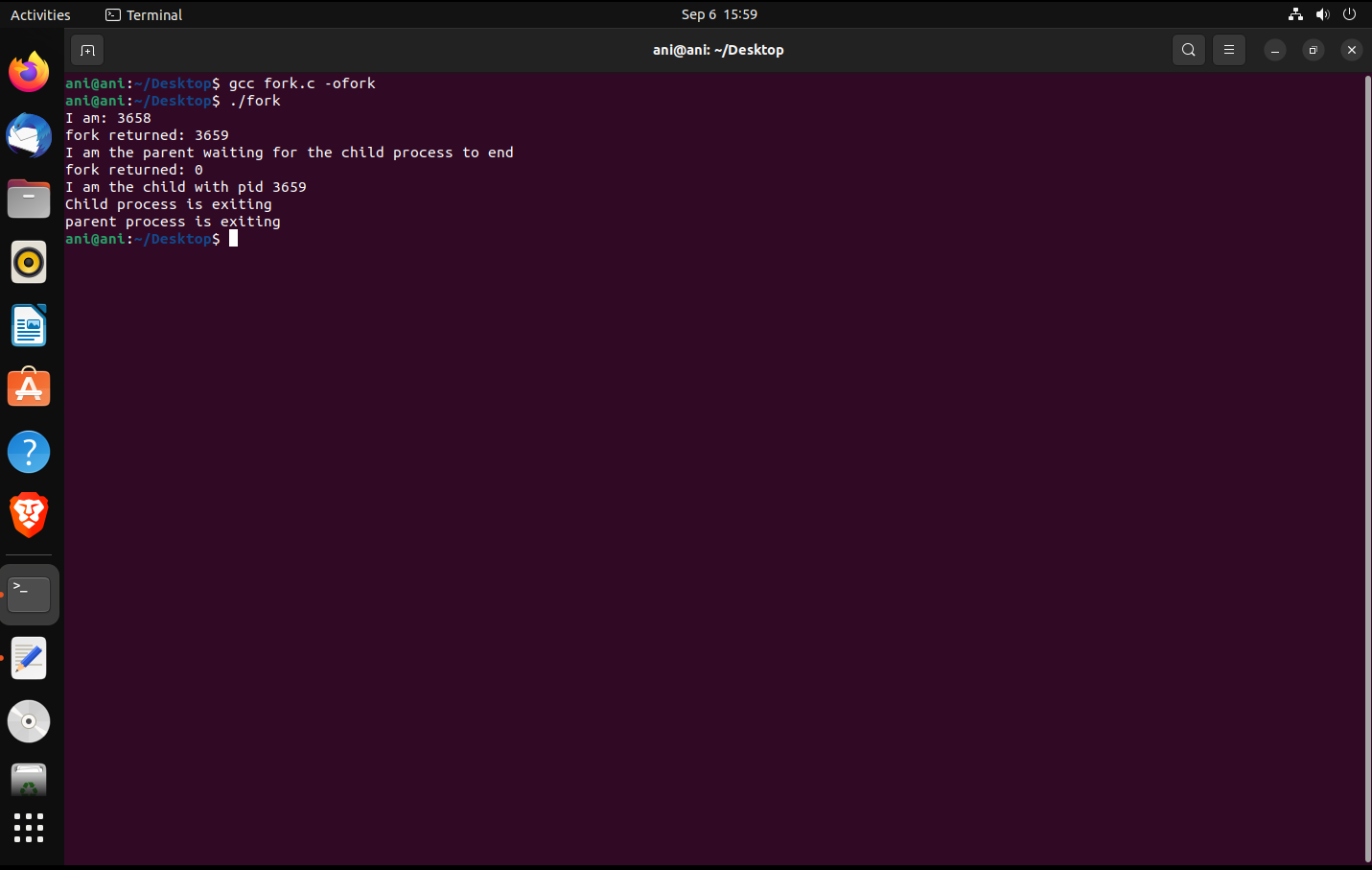
printf("parent process is exiting\n");

return(0);

}



OUTPUT: --



Result: -

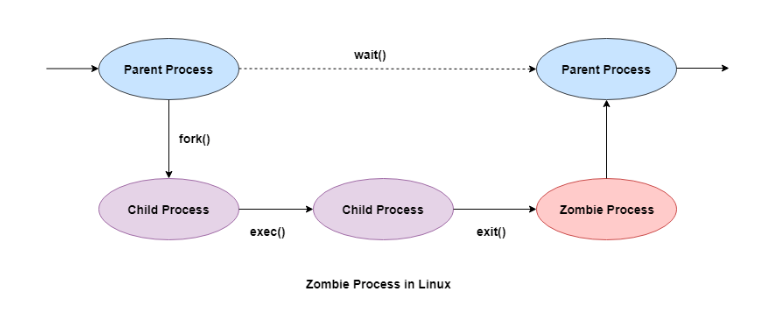
Thus we have successfully created process using fork() and Usage of getpid(),

getppid(), wait() functions.

Zombie Process

Procedure: --

A zombie process is a process whose execution is completed but it still has an entry in the process table. Zombie processes usually occur for child processes, as the parent process still needs to read its child’s exit status. Once this is done using the wait system call, the zombie process is eliminated from the process table. This is known as reaping the zombie process



Code: --

#include <stdio.h>

#include <stdlib.h>

#include<sys/wait.h>

#include<unistd.h>

int main() {

int pid = fork();

if (pid > 0) {

sleep(20);

printf("Parent process is exiting\n");

}

else {

printf("Child process is exiting\n");

printf("This is now considered as zombie process\n");

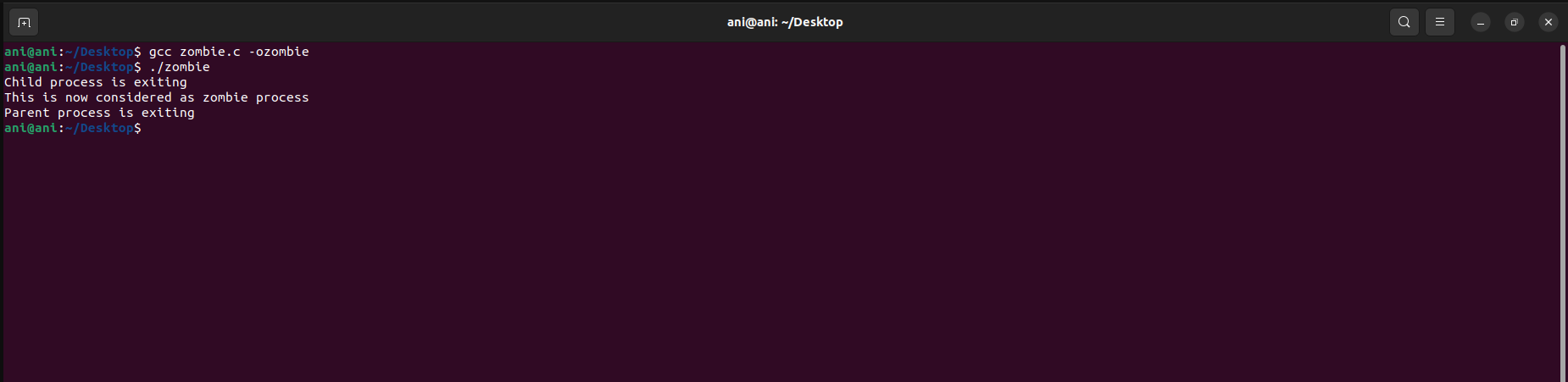
exit(0);

}

return 0;

}

Output: --



Orphan Process: --

A process whose parent process no more exists i.e., either finished or terminated without waiting for its child process to terminate is called an orphan process. Parent process finishes execution and exits while the child process is still executing and is called an orphan process .

Code: --

#include<stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main() {

int pid;

pid = fork();

if(pid == 0)

{

printf("I am the child, my process ID is %d\n",getpid());

printf("My parent's process ID is %d\n",getppid());

sleep(2);

printf("\nAfter sleep\nI am the child, my process ID is %d\n",getpid());

printf("My parent's process ID is %d\n",getppid());

exit(0);

}

else

{

sleep(3);

printf("I am the parent, my process ID is %d\n",getpid());

printf("Parent terminates\n");

}

return 0;

}

Output: --

