**Exercise 9 Understanding of wait (),sleep(),exit()**

**Date:**

**Question:** Write a Simple C program using wait(),sleep() and exit().

**Aim:**

**Prerequisite :**

**Wait(),sleep(),exit()**

Concept :

The Parent process creates a child process using fork() system call. Then, the parent process should not proceed till the child process completes the task. Hence, wait() is used to ensure that the child process is terminated.

**Procedure:**

Begin

Create a child process using fork();

Child process

{

// perform some computation

// exit()

}

// parent portion

// wait for the child process to complete – **wait()**

Perform some computations

end

**B.** Replace **wait()** with **sleep()** and note what’s happening?

**Questions:**

1. What is the function of exit() ?
2. Is the exit() returning any status to the wait()function? If so what is it and how it is interpreted by wait()?
3. What exactly happenes when sleep() is executed ?
4. Under which circumstances sleep() is used?

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No:** | **Assessment Process Description** | | **Mark(s)** |
| 1 | Documentation/Procedure(2) | |  |
| 2 | Program(5) | |  |
| 3 | Program Execution and viva(3) | |  |
|  | Total(10) | |  |
| Remarks |  | | |
| Date of Completion: | | Signature | |

**Result:**

**Exercise 10.a INTER PROCESS COMMUNICATION – Pipe()**

**Date:**

**Question:** Write a Program to create a child process and establish communication between parent and child using pipes.

**Aim:**

**Prerequisites : pipe(), read(),write(),fork()**

**Concept :**

From the previous exercises 4a,4b and 4c it is understood that the Parent and child processes remain in separate address spaces.

To have some sort of communication between them, IPC is used. Here, a kernel data structure named pipe() is used. Initially using a pipe() , a pipe is created. The pipe system call uses 2 int type file descriptors , for reading and writing messages from it.

A child process is then created using fork(). Since the child process is created after the pipe(), The file descriptors being a part of pipe , also duplicated in the child address space Hence, the child process and parent process can access the pipe using the descriptors.

**Procedure:**

**Begin**

**//** Create a pipe using pipe() with integer file descriptors fd[2].

// Check the return value of Pipe() for pipe error; if no errors proceed with the step 3.

// create a child process use fork()

Check for fork() error ; if no errors proceed otherwise exit;

If child region

{

**// use read() with appropriate file descriptor for reading from pipe**

Print the message read from the pipe.

}

// parent portion

**Use write() with appropriate file descriptors to write the message in the pipe.**

end

**Questions**:

1. What is the basic mode of operation in pipes?

2. Why the pipe is not a permanent fixture?

3. What are the characteristics of a pipe in Linux/Unix ?

4. What happens when a process tries to read an empty pipe() ? Will it wait or end ?

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No:** | **Assessment Process Description** | | **Mark(s)** |
| 1 | Documentation/Procedure(2) | |  |
| 2 | Program(5) | |  |
| 3 | Program Execution and viva (3) | |  |
|  | Total(10) | |  |
| Remarks |  | | |
| Date of Completion: | | Signature | |

**Result :**

**Exercise 10b INTERPROCESS COMMUNICATION : Two Pipes**

**Date:**

**Question:** Create 2 pipes for IPC between parent and child processes.

**Aim:**

**Prerequisites : pipe(), read(), write(),fork().**

**Concept :**

**Since, a pipe() behaves in a half duplex mode. To establish communication between parent and child and vice-versa , Two pipes(pipe1,pipe2) are used. Using a pipe1 a message is sent from parent to child , the child after receiving the message from pipe1 should respond in pipe2, which in turn read by the parent process.**

**Procedure:**

**{ parent should send a message “ Hello” to child using pipe1.**

**Child receiving should respond with “fine” using pipe2 }**

Create two pipes as pipe(fd1[2]) – pipe1, pipe(fd2[2]) – pipe2;

Create a child process : fork()

Child process

{

// read the message from pipe1 with read() system call

Print the message received.

// Reply the message in pipe2 using write() system call

}

// parent process

write the message in pipe1 using write() system call

read the reply message from pipe2 sent by the child process using read () system call.

End.

Questions :

1. What are the restrictions in using pipe system call.
2. What is a NAMED pipe? Why it is used in LINUX.
3. Which alternatives can be choosen for IPC (a) Pipe (b) NAMED Pipes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No:** | **Assessment Process Description** | | **Mark(s)** |
| 1 | Documentation/Procedure(2) | |  |
| 2 | Program(5) | |  |
| 3 | Program Execution and viva (3) | |  |
|  | Total(10) | |  |
| Remarks |  | | |
| Date of Completion: | | Signature | |

**Result :**