Dextutor

Program for wait() system call

Leave a Comment / Programs / By Baljit Singh Saini

When a process creates a child process, sometimes it becomes necessary that the parent process should execute only after the child has finished. wait() system call does exactly this. It makes the parent process wait for child process to finish and then the parent continues its working from the statement after the wait().
To be exact *wait* makes the parent wait for the child to change state. The state change can be : the child terminated; the child was stopped by a signal; or the child was resumed by a signal.

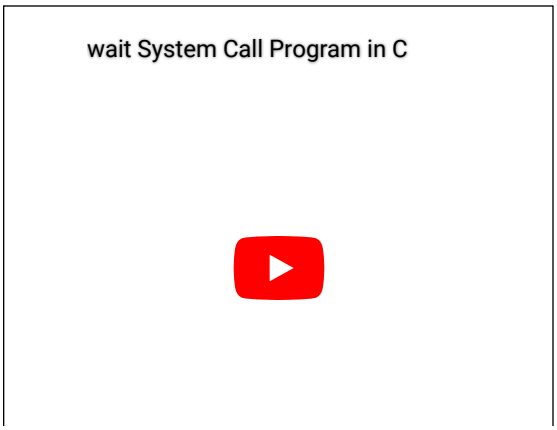
Synopsis

```
#include <sys/types.h>
#include <sys/wait.h>

pid_t wait(int *wstatus);
```

wait() system call takes only one parameter which stores the status information of the process. Pass NULL as the value if you do not want to know the exit status of the child process and are simply concerned with making the parent wait for the child. On success, wait returns the PID of the terminated child process while on failure it returns -1.

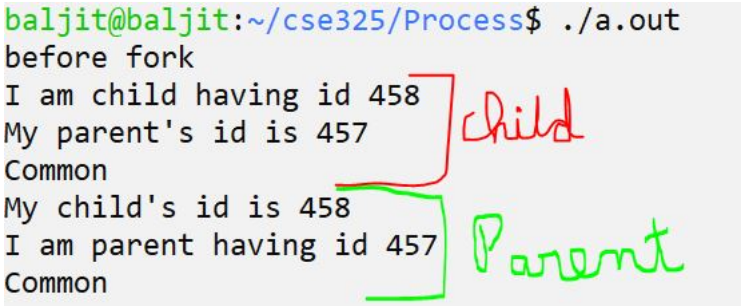
Important: wait() can be used to make the parent wait for the child to terminate(finish) but not the other way around.



Program for wait() system call which makes the parent process wait for the child to finish.

```
#include<sys/wait.h>
int main()
{
pid_t p;
printf("before fork\n");
p=fork();
if(p==0)//child
{
printf("I am child having id %d\n",getpid());
printf("My parent's id is %d\n",getppid());
}
else//parent
{
wait(NULL);
printf("My child's id is %d\n",p);
printf("I am parent having id %d\n",getpid());
}
printf("Common\n");
}
```

Output



How it works?

The execution begins by printing “before fork”. Then fork() system call creates a child process.
wait() system call is added to the parent section of the code. Hence, the moment processor starts processing the parent, the parent process is suspended because the very first statement is wait(NULL). Thus, first, the child process runs, and the output lines are all corresponding to the child process. Once the child process finishes, parent resumes and prints all its printf() statements. The NULL inside the wait() means that we are not interested to know the status of change of state of child process.

Viva questions on wait() system call

Q1. Can we use wait() to make the child process wait for the parent process to finish?
Q2. What does the wait() system call return on success?

Practice Program for wait() system call

Q1. Create a parent-child relationship between two processes. The parent should print two statements:
A) Parent (P) is having ID <PID>
B) ID of P’s Child is <PID_of_Child>
The child should print two statements:
C) Child is having ID <PID>
D) My Parent ID is <PID_of_Parent>

B

You are free to use any other relevant statement/printf as you desire and their order of execution does not matter.

Q2. Create a parent-child relationship between two processes such that the Child process creates a file named **Relation.txt** and the Parent process write some content into it by taking the input from the user

Q3. Write a program to create two child process. The parent process should wait for both the child to finish.

Relevant Programs

- Program for fork() system call
- How to create a Zombie process?
- Program for creating threads
- Program for execlp() function

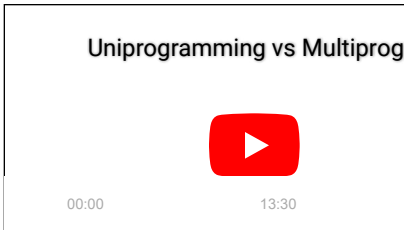
← Previous Post

Next Post →

Leave a Comment

You must be logged in to post a comment.

OS Theory



Categories

Database

- Entity-Relationship(ER)
- Introduction to Database

Linux

- commands