



Rabia Batool

2022-BSE-064

Group#B

Operating system

LAB#06

Submitted to Sir shehzad

Viva questions:

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?

The wait() function in the context of process management is used by a parent process to wait for its child processes to complete. It is not typically used to make a child process wait for the parent process to finish. The wait() function is used to collect the exit status of a child process and perform cleanup, such as releasing resources associated with the child process.

(2)

The wait() system call in Unix-like operating systems returns the process ID of the terminated child process on success. It allows the parent process to obtain information about the child process that has terminated, including its exit status.

Task#1

- Q1. Write a program to create two child process. The parent process should wait for both the child to finish.

```
#include<unistd.h>

#include<sys/types.h>

#include<stdio.h>

#include<sys/wait.h>

int main()
```

```
{  
pid_t p,q;  
printf("before fork\n");  
  
p=fork();  
q=fork();  
if(p==0)//child  
{  
printf("I am child having id %d\n",getpid());  
printf("My parent's id is %d\n",getppid());  
}  
if(q==0)//child  
{  
printf("I am child q 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());  
}  
}
```

```
GNU nano 6.2 rabiaa.c
#include<unistd.h>
#include<sys/types.h>
#include<stdio.h>
#include<sys/wait.h>
int main()
{
    pid_t p,q;
    printf("before fork\n");

    p=fork();
    q=fork();
    if(p==0)//child
    {
        printf("I am child having id %d\n",getpid());
        printf("My parent's id is %d\n",getppid());
    }
    if(q==0)//child
    {
        printf("I am child q 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());
    }
}
```

Help Exit Write Out Read File Where Is Replace Cut Paste Execute Justify Location Go To Line Undo Redo Set Mark Copy To Bracket Where Was

Output:

```
~$ touch rabiaa.c
~$ nano rabiaa.c
~$ gcc rabiaa.c
~$ ./a.out
before fork
I am child q having id 1097
My parent's id is 1095
I am child having id 1096
My parent's id is 1095
My child's id is 1096
I am parent having id 1095
I am child having id 1098
My parent's id is 1096
I am child q having id 1098
My parent's id is 1096
~$ My child's id is 0
I am parent having id 1096
```

Task#2

Q2. Create a parent-child relationship between two process. The parent should print two statements:

A) its own PID

B) PID of its child

The child should print two statements:

C) its own PID

D) PID of its parent

Make use of wait() in such a manner that the order of the four statements A, B, C and D is:

A

C

D

B

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

```
#include<unistd.h>
```

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

```
#include<stdio.h>
```

```
#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);

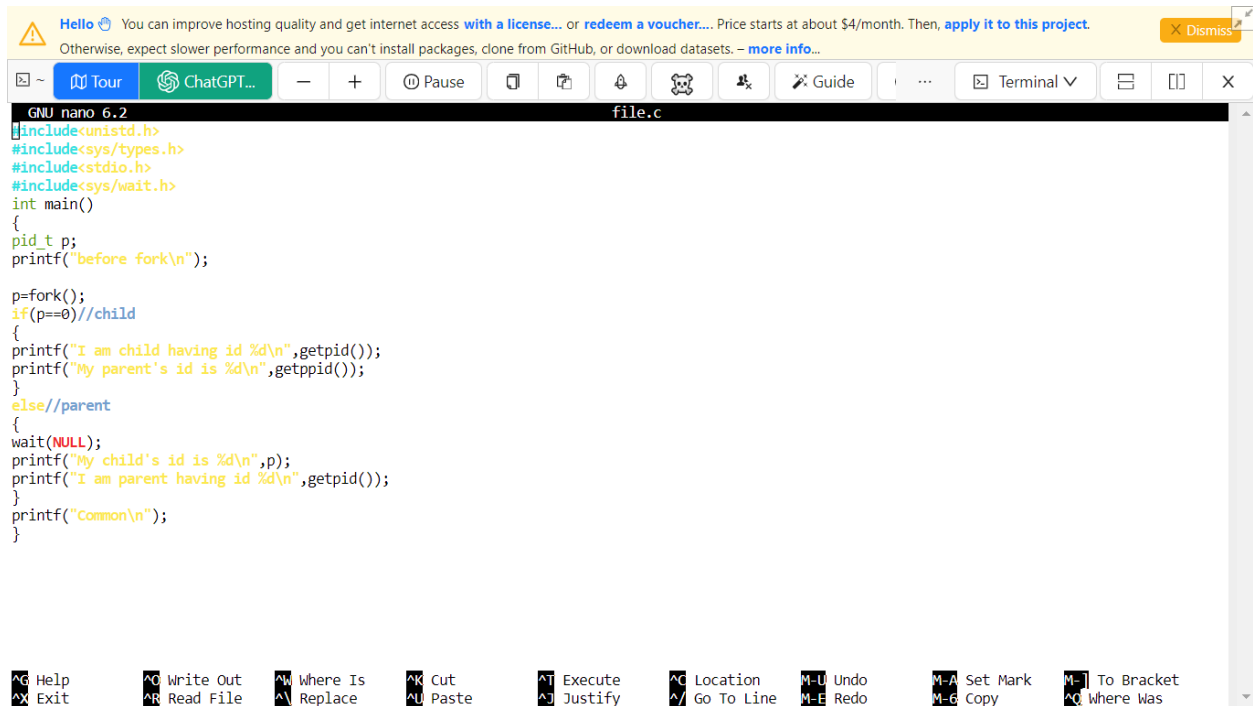
printf("I am parent having id %d\n",getpid());

}

printf("Common\n");

}

```



GNU nano 6.2 file.c

```

#include<unistd.h>
#include<sys/types.h>
#include<stdio.h>
#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");
}

```

Help Exit Write Out Read File Where Is Replace Cut Paste Execute Justify Location Go To Line Undo Redo M-A Set Mark M-G Copy M-I To Bracket M-Q Where Was

Output:

```

~$ touch file.c
~$ nano file.c
~$ gcc file.c
~$ ./a.out
before fork
I am child having id 1940
My parent's id is 1939
Common
My child's id is 1940
I am parent having id 1939
Common
~$ █

```