

```
/*getpid()*/
```

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

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

```
int main(){
```

```
    printf("process identifiere(pid) of parent process:%d\n",(int)getpid());
```

```
    int f_val=fork();
```

```
    if(f_val==0){
```

```
        printf("after fork() system call the pid of child:%d\n",(int)getpid());
```

```
    }
```

```
    else{
```

```
        printf("after fork() system call the pid of parent:%d, fork return  
value:%d\n",(int)getpid(),f_val);
```

```
    }
```

```
    return 0;
```

```
}
```

```
// /*getpid()*/
```

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

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

```
// int main(){
```

```
//     printf("process identifiere(pid) of parent process:%d\n",getppid());
```

```
//     int f_val=fork();
```

```
//     if(f_val==0){
```

```
//         printf("after fork() system call the pid of child:%d\n",getpid());
```

```
//     }
```

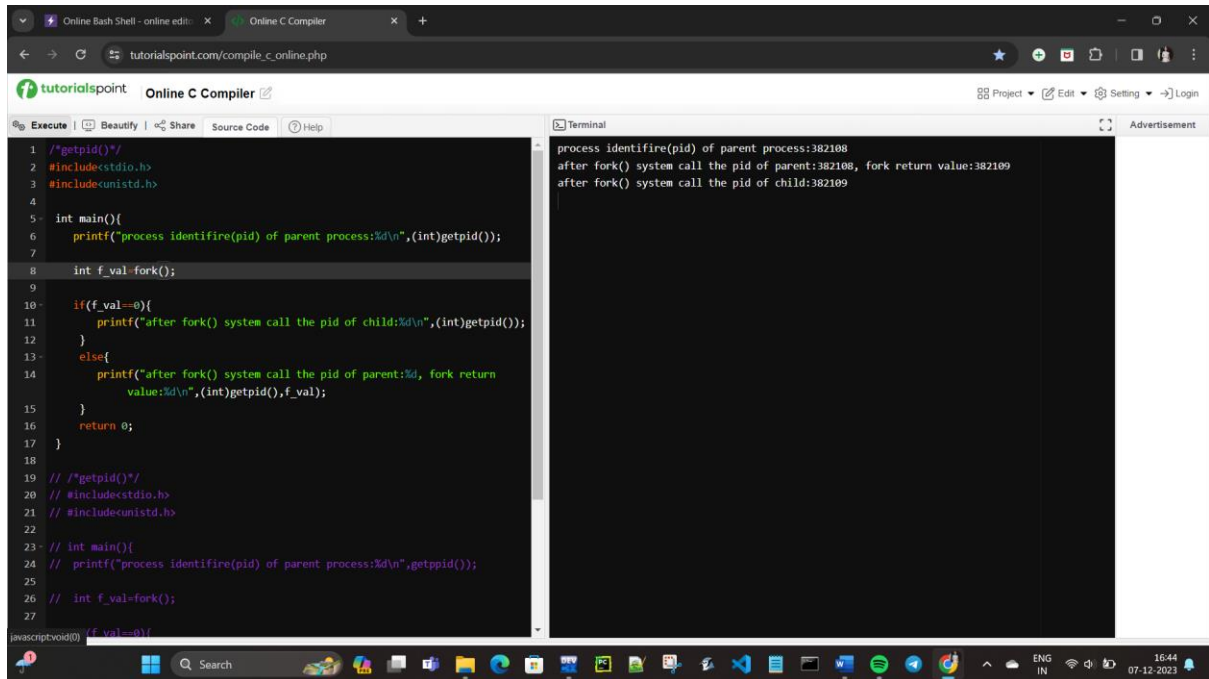
```
//      else{

//          printf("after fork() system call the pid of parent:%d, fork return
value:%d\n",getppid(),f_val);

//      }

//      return 0;

//}
```



The screenshot shows a web browser window with the URL `tutorialspoint.com/compile_c_online.php`. The page title is "Online C Compiler". The interface includes a code editor on the left and a terminal on the right. The code in the editor is a C program that demonstrates the `fork()` system call and `getpid()` and `getppid()` functions. The terminal output shows the execution results, including the parent and child process IDs.

```
1  /*getpid()*/
2  #include<stdio.h>
3  #include<unistd.h>
4
5  int main(){
6      printf("process identifiere(pid) of parent process:%d\n",(int)getpid());
7
8      int f_val=fork();
9
10     if(f_val==0){
11         printf("after fork() system call the pid of child:%d\n",(int)getpid());
12     }
13     else{
14         printf("after fork() system call the pid of parent:%d, fork return
value:%d\n",(int)getpid(),f_val);
15     }
16     return 0;
17 }
18
19 // /*getpid()*/
20 // #include<stdio.h>
21 // #include<unistd.h>
22
23 // int main(){
24 //     printf("process identifiere(pid) of parent process:%d\n",getppid());
25 //
26 //     int f_val=fork();
27
28 //     if(f_val==0){
```

```
process identifiere(pid) of parent process:382108
after fork() system call the pid of parent:382108, fork return value:382109
after fork() system call the pid of child:382109
```