Operating System Practice

Dr. Sanjaya Kumar Panda
Asst. Professor, Dept. of CSE
NIT Warangal
sanjayauce@gmail.com
+91-9861126947

What is a System Call?

- open(), read(), write(), fork()
- System calls are how a program enters the kernel to perform some task.
  - Creating processes
  - Doing network and file IO
- As an application developer, you don't typically need to think about how exactly a system call is made. You simply include the appropriate header file and make the call as if it were a normal function.

fork()

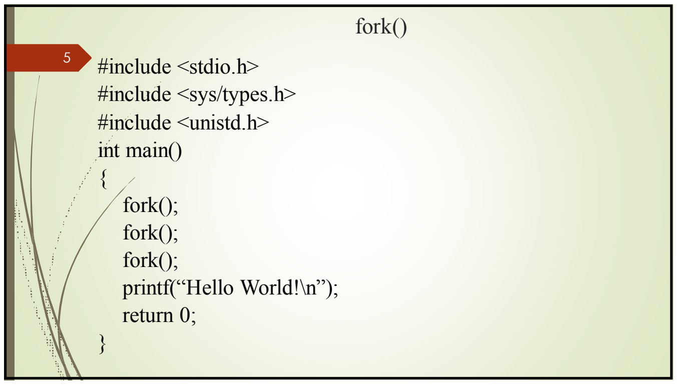
- fork() system call is used for creating a new process, which is called child process.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
   fork();
   printf("Hello World!\n");
   return 0;
}
```

fork()

- Output

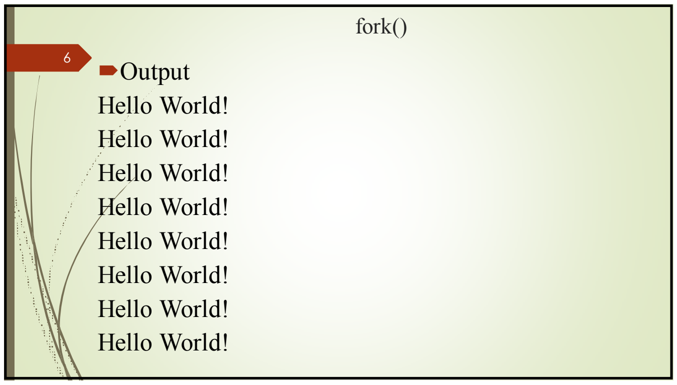
```
Hello World!
Hello World!
```



fork()

5

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    fork();
    fork();
    fork();
    printf(“Hello World!\n”);
    return 0;
}
```

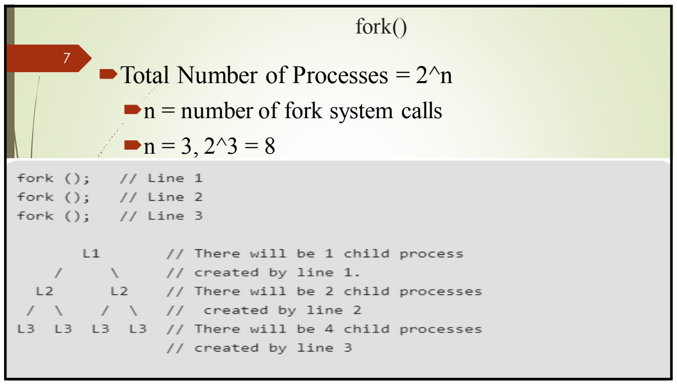


fork()

6

- Output

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

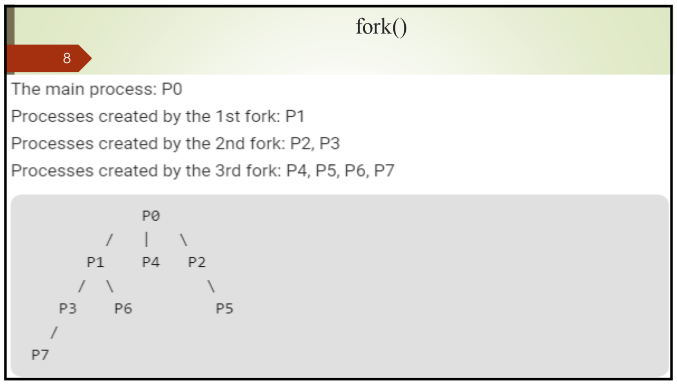


fork()

7

- Total Number of Processes = 2^n
  - n = number of fork system calls
  - n = 3, 2^3 = 8

```
fork ();   // Line 1
fork ();   // Line 2
fork ();   // Line 3

       L1       // There will be 1 child process
    /     \     // created by line 1.
  L2       L2   // There will be 2 child processes
 /  \    /  \   //  created by line 2
L3  L3  L3  L3  // There will be 4 child processes
                // created by line 3
```



fork()

8

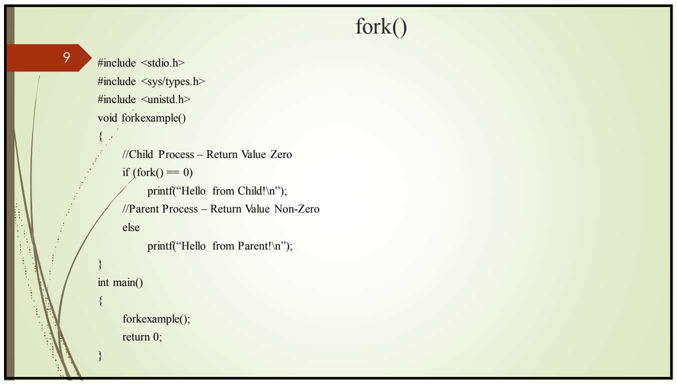
The main process: P0

Processes created by the 1st fork: P1

Processes created by the 2nd fork: P2, P3

Processes created by the 3rd fork: P4, P5, P6, P7

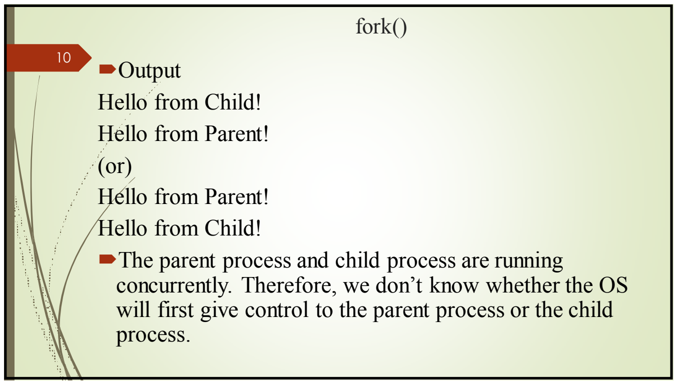
```
              P0
          /   |   \
        P1    P4    P2
       /  \            \
     P3    P6           P5
    /
  P7
```



fork()

9

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void forkexample()
{
    //Child Process – Return Value Zero
    if (fork() == 0)
        printf(“Hello  from Child!\n”);
    //Parent Process – Return Value Non-Zero
    else
        printf(“Hello  from Parent!\n”);
}
int main()
{
    forkexample();
    return 0;
}
```



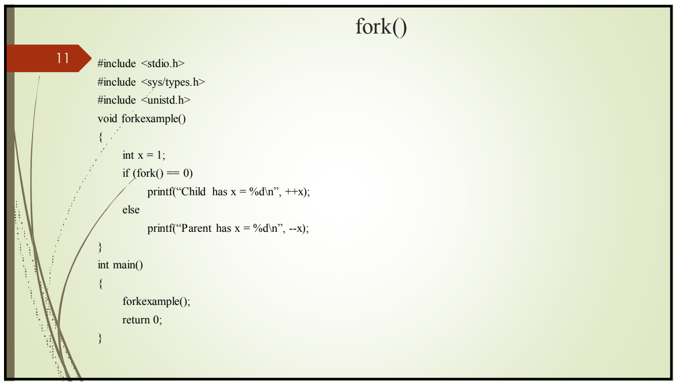
fork()

10

- Output

```
Hello from Child!
Hello from Parent!
(or)
Hello from Parent!
Hello from Child!
```

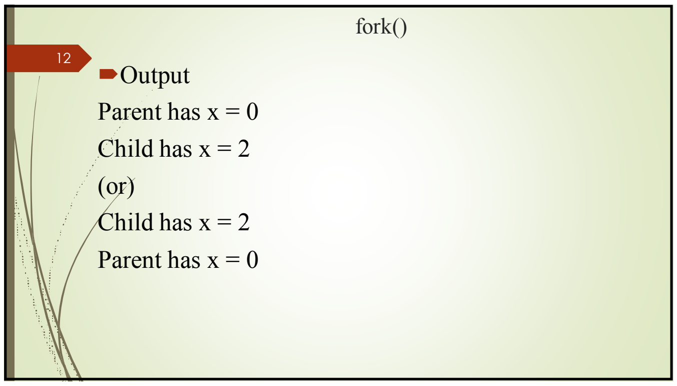
- The parent process and child process are running concurrently. Therefore, we don’t know whether the OS will first give control to the parent process or the child process.



fork()

11

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void forkexample()
{
    int x = 1;
    if (fork() == 0)
        printf(“Child  has x = %d\n”, ++x);
    else
        printf(“Parent has x = %d\n”, --x);
}
int main()
{
    forkexample();
    return 0;
}
```

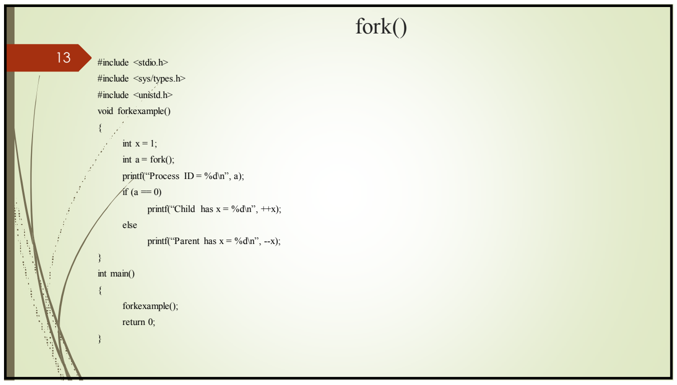


fork()

12

- Output

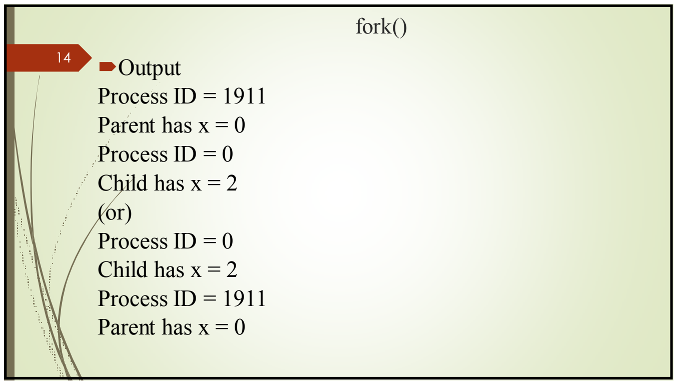
```
Parent has x = 0
Child has x = 2
(or)
Child has x = 2
Parent has x = 0
```



fork()

13

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void forkexample()
{
        int x = 1;
        int a = fork();
        printf("Process ID = %d\n", a);
        if (a == 0)
                printf("Child has x = %d\n", ++x);
        else
                printf("Parent has x = %d\n", --x);
}
int main()
{
        forkexample();
        return 0;
}
```

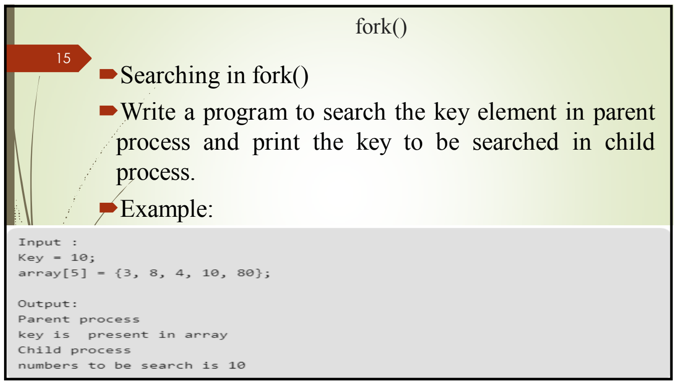


fork()

14

- Output

```
Process ID = 1911
Parent has x = 0
Process ID = 0
Child has x = 2
(or)
Process ID = 0
Child has x = 2
Process ID = 1911
Parent has x = 0
```

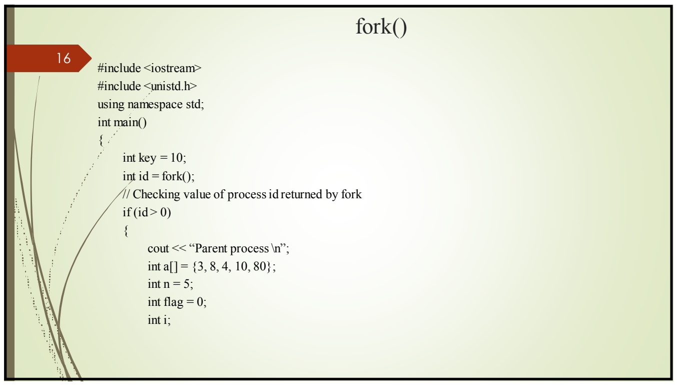


fork()

15

- Searching in fork()
- Write a program to search the key element in parent process and print the key to be searched in child process.
- Example:

```
Input :
Key = 10;
array[5] = {3, 8, 4, 10, 80};

Output:
Parent process
key is  present in array
Child process
numbers to be search is 10
```



fork()

16

```
#include <iostream>
#include <unistd.h>
using namespace std;
int main()
{
        int key = 10;
        int id = fork();
        // Checking value of process id returned by fork
        if (id > 0)
        {
                cout << "Parent process \n";
                int a[] = {3, 8, 4, 10, 80};
                int n = 5;
                int flag = 0;
                int i;
```

fork()

17

```
        for (i = 0; i < n; i++)
        {
            if (a[i] != key)
                flag = 0;
            else    { flag = 1; break;}
        }
        if (flag == 0)
            cout << "key is not present in array";
        else
        {
            cout << "key is present in array";
            cout << "\n";
        }
    }
    // If n is 0, i.e., we are in child process
    else
    {
        cout << "Child process \n";
        cout << "numbers to be search is ";
        cout << key;
    }
    return 0;
}
```

fork()

18

- Output

Parent process

key is present in array

Child process

numbers to be search is 10

- At some instance of time, it is not necessary that child process will execute first or parent process will be first allotted CPU, any process may get CPU assigned, at some quantum time. Moreover process id may differ during different executions.

fork()

19

- Sorting in fork()
- Write a program to sort the numbers in parent process and print the unsorted numbers in child process.
- Example:

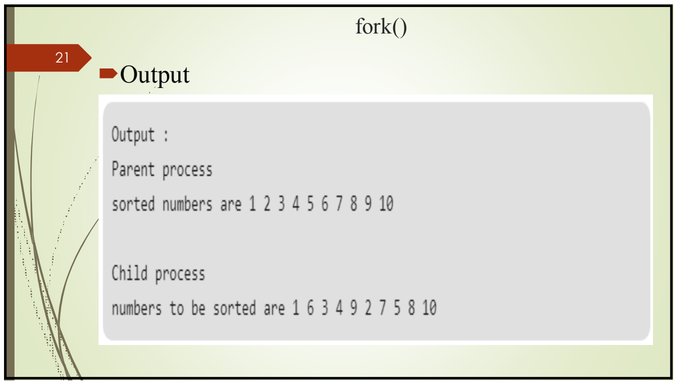
```
Input : 5, 2, 3, 1, 4

Output :
Parent process
sorted numbers are
1, 2, 3, 4, 5


Child process
numbers to sort are
 5, 2, 3, 1, 4
```

fork()

20

```
#include <iostream>
#include <unistd.h>
#include <algorithm>
using namespace std;
```

```
int main()
{
    int a[] = { 1, 6, 3, 4, 9, 2, 7, 5, 8, 10 };
    int n = sizeof(a)/sizeof(a[0]);
    int id = fork();

    // Checking value of process id returned by fork
    if (id > 0) {
        cout << "Parent process \n";

        sort(a, a+n);

        // Displaying Array
        cout << " sorted numbers are ";
        for (int i = 0; i < n; i++)
            cout << "\t" << a[i];

        cout << "\n";

    }

    // If n is 0 i.e. we are in child process
    else {
        cout << "Child process \n";
        cout << "\n  numbers to be sorted are ";
        for (int i = 0; i < n; i++)
            cout << "\t" << a[i];
    }

    return 0;
}
```



fork()

21

- Output

```
Output :
Parent process
sorted numbers are 1 2 3 4 5 6 7 8 9 10

Child process
numbers to be sorted are 1 6 3 4 9 2 7 5 8 10
```