Processes

Lab 03

اللهم علمنا ما ينفعنا،،، وانفعنا بما علمتنا،،، وزدنا علماً

Lab Objective

- To practice creating child process using `fork()`.

The `fork` Function

- In computing, when a process forks, it creates a copy of itself, which is called a "***child process***." The original process is then called the "***parent process***".
- The `fork()` function is used from a "*parent*" process to create a duplicate process, the "*child*".

```
1 #include <iostream>
2 #include <unistd.h>      //fork()
3 #include <stdlib.h>
4 #include <sys/types.h>    //pid_t
5
6 using namespace std;
7
8 int main()
9 {
10
11
12     fork();
13
14     cout<<"Hello CS323 students \n PID= "<< getpid()<<"\n";
15
16      return(0);
17   }//end main
18
19
```

The `fork` Function

- In computing, when a process forks, it creates a copy of itself, which is called a "***child process***." The original process is then called the "***parent process***".
- The `fork()` function is used from a "*parent*" process to create a duplicate process, the "*child*".
- The parent and the child processes can tell each other apart by examining the return value of the `fork()` system call

The `fork` Function

- The parent and the child processes can tell each other apart by examining the return value of the `fork()` system call

```
pid_t = fork(void);
```

- If successful, the `fork` function returns twice:

Parent

returns PID of the newly-created child process

Child

returns 0

- On failure, the `fork` function returns once:

Parent

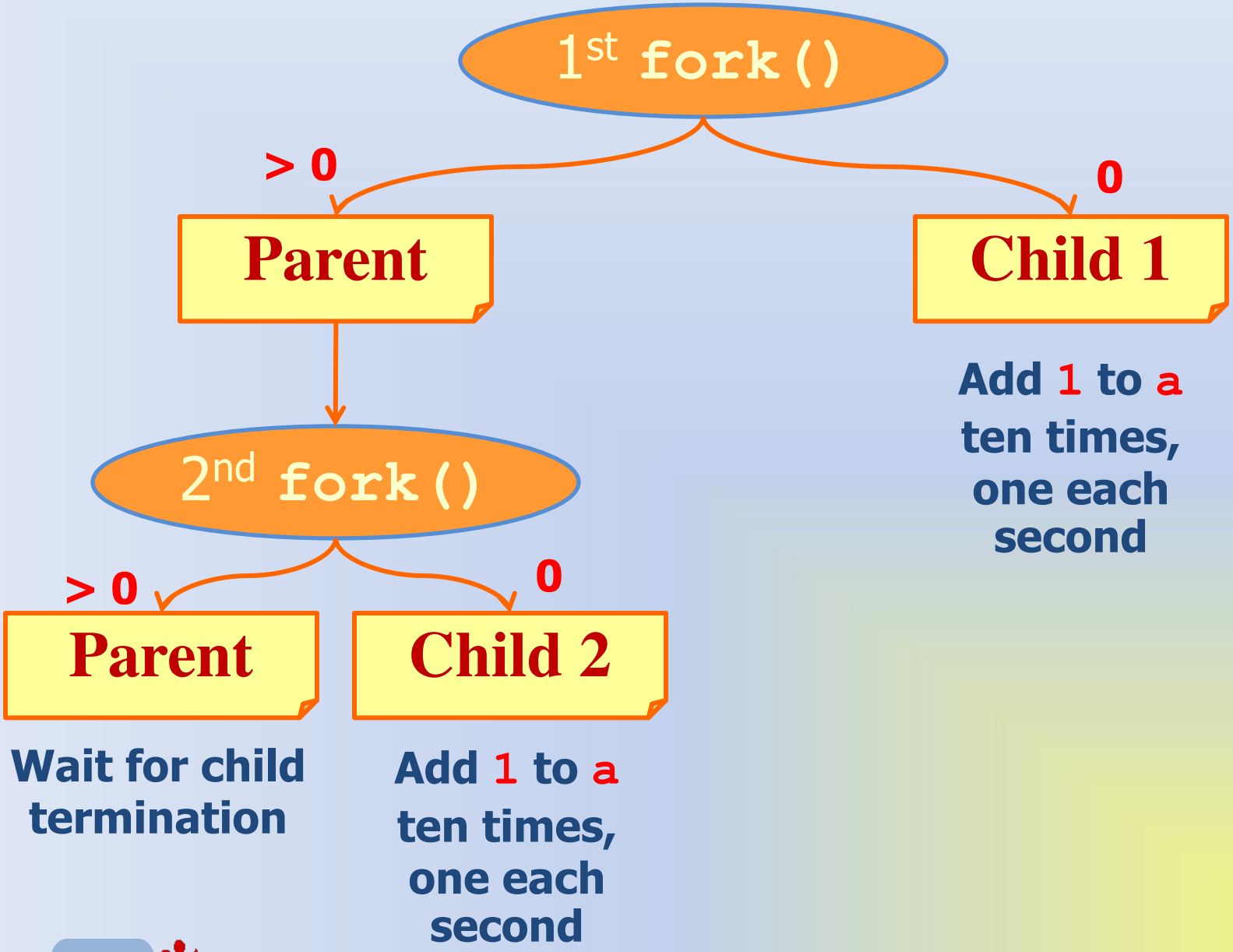
returns -1

Parent and Child

- A child inherits its parent's permissions, working-directory, root-directory, open files, etc.
- All descriptors that were open in the parent before the call to `fork` are shared with the child after the `fork` returns.

Practice

- In the following C++ program, the main process forks two children.
- Every child repeats adding the value 1 to the variable “a” ten times.
- Write, compile and run the program in Linux.


1st fork()
> 0
0
Parent
Child 1
Add 1 to a
ten times,
one each
second
2nd fork()
> 0
0
Parent
Child 2
Wait for child
termination
Add 1 to a
ten times,
one each
second

```
#include <iostream>
#include <stdlib.h>      /* exit() */
#include <unistd.h>      /* fork() */
#include <sys/types.h>   /* pid_t */
#include <sys/wait.h>    /* wait() */
using namespace std; //to replace std::cout with cout

int main()
{
        pid_t pid1, pid2, cpid;
        int i, j, a, status;
        a = 0;
        cout<<"\n Parent process pid="<<getpid()<<"\n";
        pid1 = fork();   //fork child 1 process
        if (pid1 < 0)    //error occurred
        {
                cout<<"First Fork Failed\n";
                exit(-1);
        }/end if
        else if (pid1 == 0)      //child 1 process
        {
         cout<<"\n Child1 process pid="<<getpid()<<"\n";
                for (i=0; i<10; i++)
                {
                        a++;
                        cout<<"Child1: a = "<<a<<"\n";
                        sleep(1);
                }//end for
        }//end else if
```

The main process forks child 1

Error

When `fork()` returns a negative number, an error happened

Child 1

When `fork()` returns 0, we are in the child 1 process

Here child 1 Add 1 to 'a' ten times

a = 10

```
        else    //parent process
        {
                pid2 = fork();  //fork child 2 process
                if (pid2 < 0)   //error occurred
                {
                cout<<"Second Fork Failed\n";
                exit(-1);
                }//end if
                else if (pid2 == 0) //child 2 process
                {
      cout<<"\n Child2 process pid="<<getpid()<<"\n";
                        for (j=0; j<10; j++)
                        {
                                a++;
                                cout<<"Child2: a = "
                                    <<a<<"\n";
                                sleep(1);
                        }//end for
                }//end else if
                else    //parent process
                {
                        cpid = wait(&status);
                        cout<<"\n*****Parent is
Closing*****\n";
                        exit(0);
                }
        }//end else
}//end main
```

Parent

The parent forks another child

Error

When `fork()` returns a negative number, an error happened

Child 2

When `fork()` returns 0, we are in the child 2 process

Here child 2 Add 1 to 'a' ten times

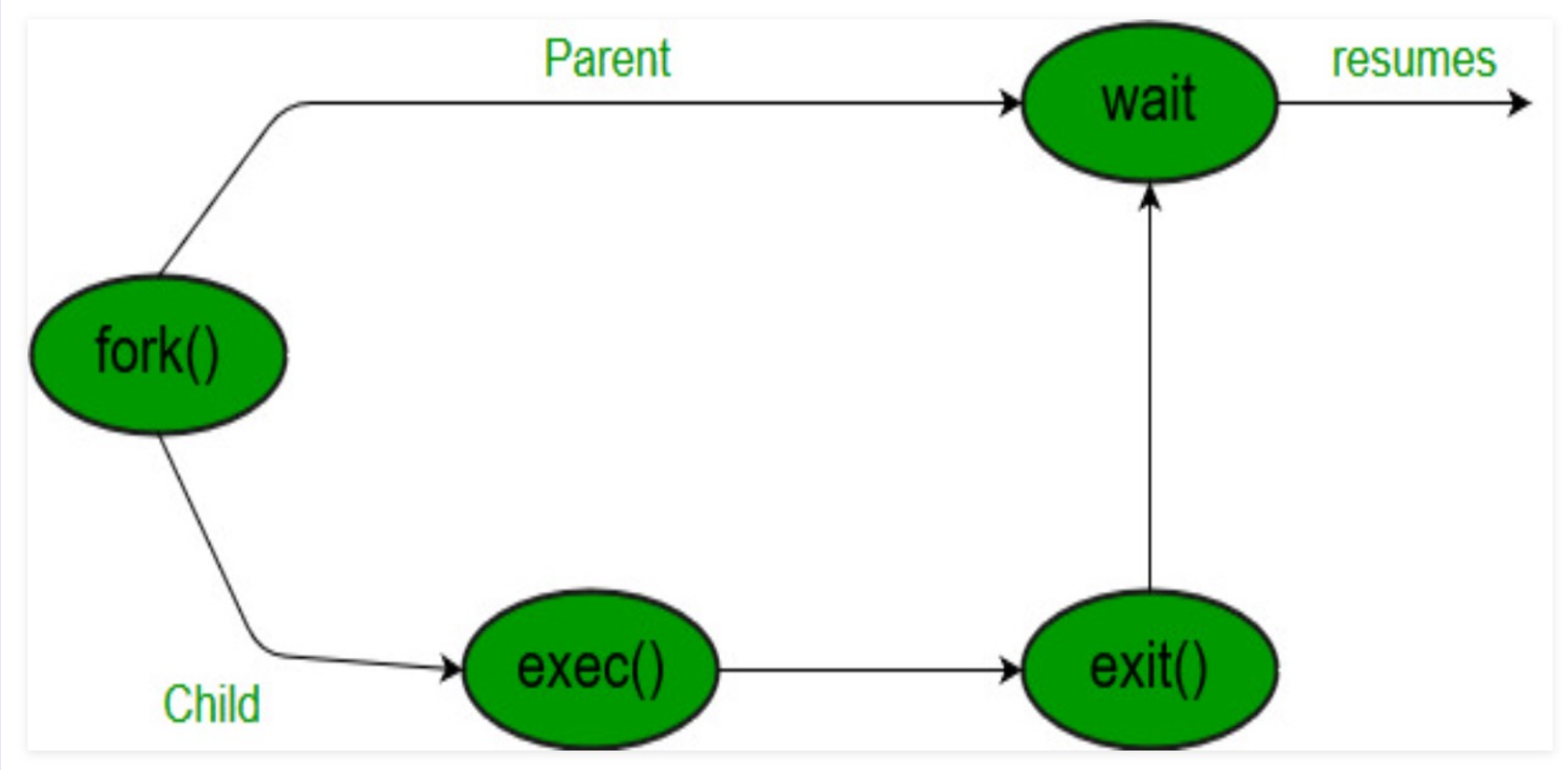
a = 10

Parent

When `fork()` returns a positive number, we are in the parent process

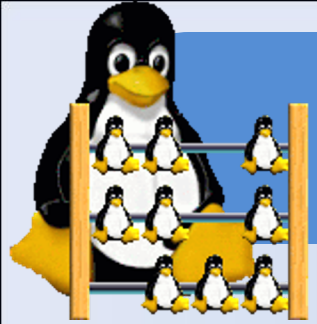
The parent waits for children termination

Process Termination



Process Termination

- Process executes last statement and asks the operating system to delete it (**exit**)
- Parent may terminate execution of children processes (**abort**)
  - Child has exceeded allocated resources
  - Task assigned to child is no longer required
  - If parent is exiting
    - Some operating systems do not allow child to continue if its parent terminates.



Check Off

1) Why the final value of `a` is 10 and not 20?
2) Use the command `ps -all` in a separate window while the above program is running. Write down the `PID` of the processes related to the program.
3) Kill child 1 and then child 2 while the program is running. Briefly explain what will happen.
4) Kill the main process while the program is running. Briefly explain what will happen.

??? ANY QUESTIONS ???

☺