

Сообщество EngEd

Программа инженерного образования Секции (EngEd) способствует созданию сообщества студентов университетов в областях, связанных с информатикой, для исследования и обмена темами, которые имеют отношение к инженерам в современном технологическом ландшафте. Более подробную информацию и рекомендации по программе вы можете найти в репозитории GitHub. Если вы в настоящее время обучаетесь в области, связанной с информатикой, и заинтересованы в участии в программе, пожалуйста, заполните эту форму.

Fork() на языке программирования Си

11 февраля 2021 года

Возможно, вы видели много процессов в диспетчере задач, если используете Windows. Или в вашем мониторе ресурсов, если вы используете Linux. Вы когда-нибудь задумывались о том, как они создаются?

В этом уроке мы поговорим о `fork()` функции, а затем реализуем некоторые примеры на языке программирования Си.

Мы используем файлы cookie для улучшения пользовательского опыта и анализа трафика веб-сайта. Нажав "Принять", вы соглашаетесь с использованием файлов cookie нашего веб-сайта, как описано в нашей [Политике использования файлов cookie](#). Вы можете изменить настройки файлов cookie в любое время, нажав кнопку "[Настройки](#)".

Предпочтения

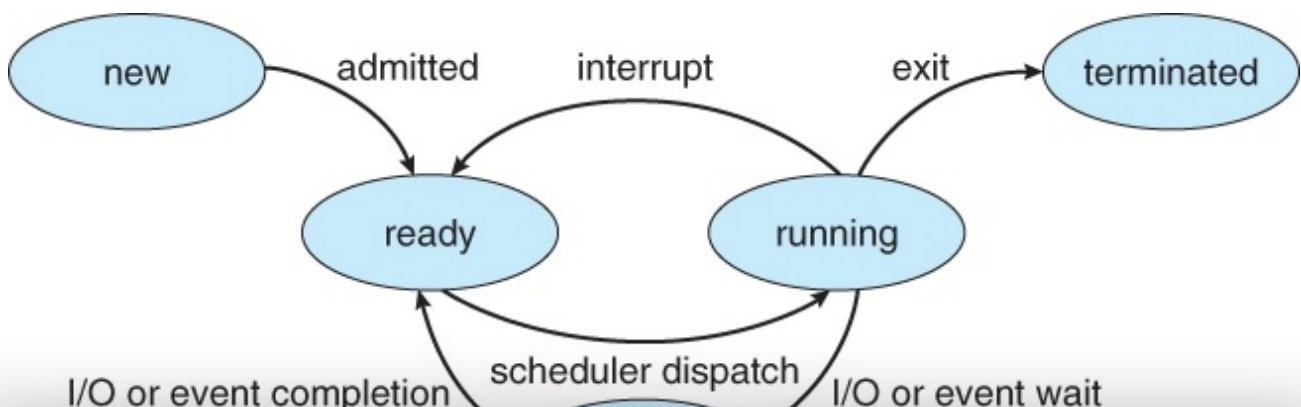
Принять

Согласно [Википедии](#), процесс – это экземпляр компьютерной программы, выполняемой одним или несколькими потоками. Содержит программный код и его действие. В зависимости от операционной системы (ОС) процесс может состоять из нескольких потоков выполнения, которые выполняют инструкции одновременно.

Что такое Fork()?

В вычислительной области `fork()` это основной метод создания процессов в Unix-подобных операционных системах. Эта функция создает новую копию, называемую *дочерней*, из исходного процесса, который называется *родительским*. Когда родительский процесс закрывается или по какой-либо причине выходит из строя, он также убивает дочерний процесс.

Давайте начнем с жизненного цикла процесса:



Мы используем файлы cookie для улучшения пользовательского опыта и анализа трафика веб-сайта. Нажав "Принять", вы соглашаетесь с использованием файлов cookie нашего веб-сайта, как описано в нашей [Политике использования файлов cookie](#). Вы можете изменить настройки файлов cookie в любое время, нажав кнопку "[Настройки](#)".

Предпочтения

Принять

принимает никаких параметров и возвращает значение в следующем образом:

Ноль: если это дочерний процесс (созданный процесс).

Положительное значение: если это родительский процесс.

Отрицательное значение: если произошла ошибка.

Примечание: Следующий код выполняется только в операционных системах на базе Linux и UNIX. Если вы используете Windows, то я рекомендую вам использовать [Cygwin](#).

Давайте перейдем к практическому разделу, где мы создадим примеры от простого уровня до продвинутого.

Привет, мир!

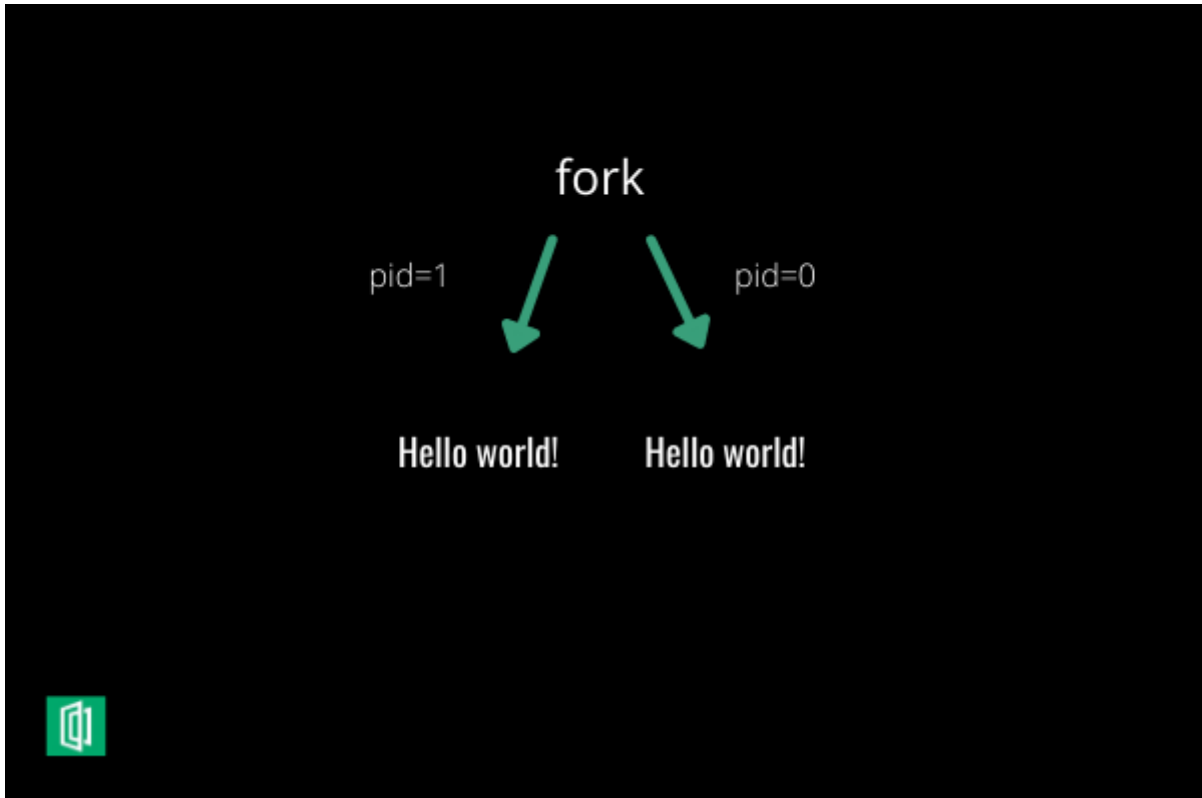
```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    /* fork a process */
    fork();
    /* the child and parent will execute every line of code after the fork (ea
```

Мы используем файлы cookie для улучшения пользовательского опыта и анализа трафика веб-сайта. Нажав "Принять", вы соглашаетесь с использованием файлов cookie нашего веб-сайта, как описано в нашей [Политике использования файлов cookie](#). Вы можете изменить настройки файлов cookie в любое время, нажав кнопку "[Настройки](#)".

Предпочтения

Принять

Where one of the output came from the parent process and the other one from the child process.



Simply, we can tell that the result is 2 power of n, where n is the number of fork() system calls.

Мы используем файлы cookie для улучшения пользовательского опыта и анализа трафика веб-сайта. Нажав "Принять", вы соглашаетесь с использованием файлов cookie нашего веб-сайта, как описано в нашей [Политике использования файлов cookie](#). Вы можете изменить настройки файлов cookie в любое время, нажав кнопку "[Настройки](#)".

Предпочтения

Принять

```
fork();
fork();
printf("Hello world!\n");
return 0;
}
```

The result is:

```
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
```

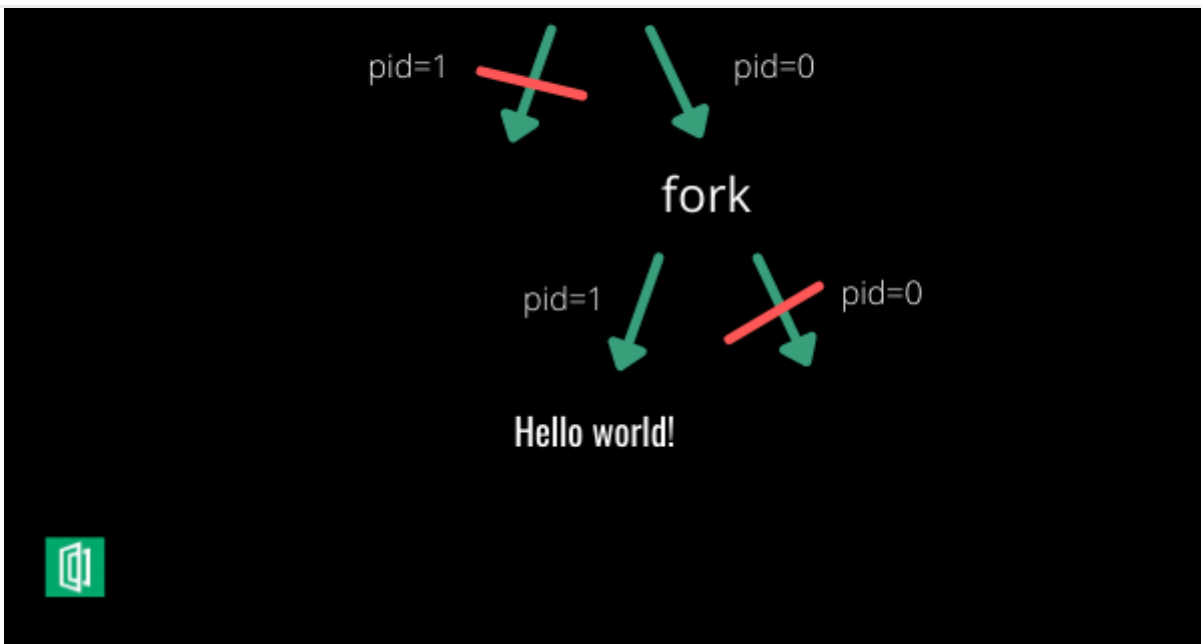
Another example is:

```
int main() {
    if(fork() == 0)
        if(fork())
            printf("Hello world!!\n");
}
```

Мы используем файлы cookie для улучшения пользовательского опыта и анализа трафика веб-сайта. Нажав "Принять", вы соглашаетесь с использованием файлов cookie нашего веб-сайта, как описано в нашей [Политике использования файлов cookie](#). Вы можете изменить настройки файлов cookie в любое время, нажав кнопку "[Настройки](#)".

Предпочтения

Принять



Inside the first `if` condition a fork has occurred and it is checking if it is the child process, it then continues to execute its code. Otherwise (if it's the parent process) it will not go through that `if`. Then, in the second `if`, it will only accept the parent process which holds the positive id.

As a result, it will print only one "Hello world!".

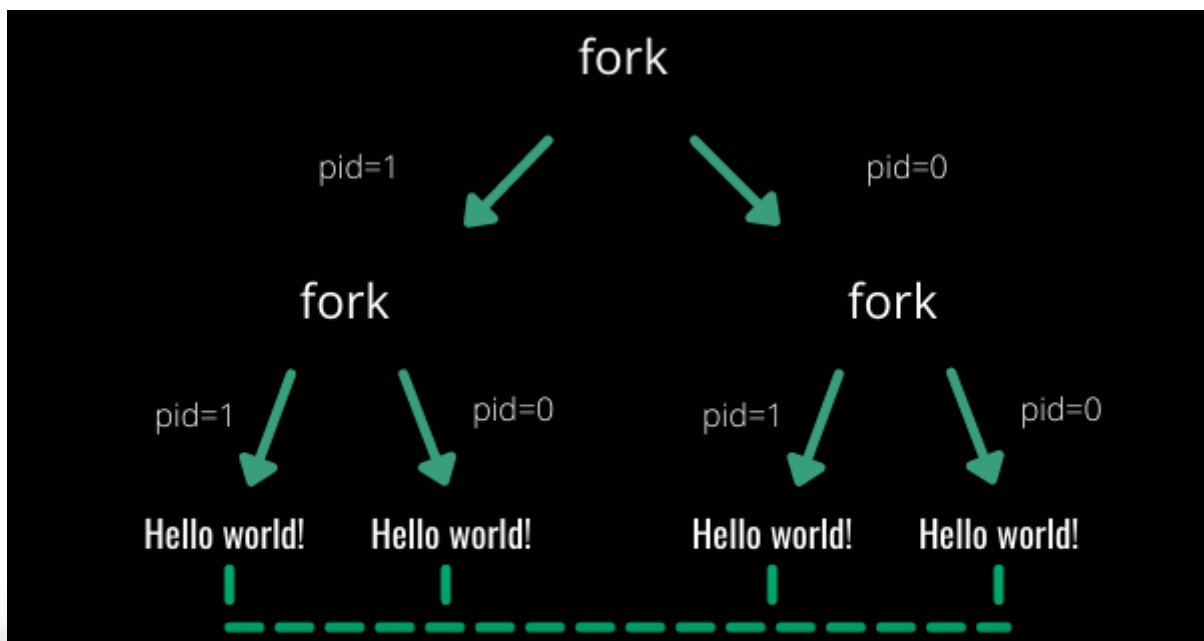
Now try to execute the following code and compare your result with ours:

Мы используем файлы cookie для улучшения пользовательского опыта и анализа трафика веб-сайта. Нажав "Принять", вы соглашаетесь с использованием файлов cookie нашего веб-сайта, как описано в нашей [Политике использования файлов cookie](#). Вы можете изменить настройки файлов cookie в любое время, нажав кнопку "[Настройки](#)".

[Предпочтения](#)[Принять](#)

The result will be:

```
Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!
```



Мы используем файлы cookie для улучшения пользовательского опыта и анализа трафика веб-сайта. Нажав "Принять", вы соглашаетесь с использованием файлов cookie нашего веб-сайта, как описано в нашей [Политике использования файлов cookie](#). Вы можете изменить настройки файлов cookie в любое время, нажав кнопку "[Настройки](#)".

Предпочтения

Принять

print that hello world! then exits.

Advanced example

When a process creates a new process, then there are two possibilities for the execution exit:

The parent continues to execute concurrently with its child.

The parent waits until some or all of its children have terminated.

```
#include <sys/types.h>
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main(int argc, char *argv[]) {

    /* fork a child process */
    pid_t pid = fork();

    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }

    else if (pid == 0) { /* child process */
```

Мы используем файлы cookie для улучшения пользовательского опыта и анализа трафика веб-сайта. Нажав "Принять", вы соглашаетесь с использованием файлов cookie нашего веб-сайта, как описано в нашей [Политике использования файлов cookie](#). Вы можете изменить настройки файлов cookie в любое время, нажав кнопку "[Настройки](#)".

Предпочтения

Принять

Система `wait call wait(NULL)` заставит родительский процесс ждать, пока дочерний процесс не выполнит все его команды.

Результатом будет:

```
I'm the child
Child Complete
```

Еще один пример:

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
    printf("I am: %d\n", (int) getpid());
```

Мы используем файлы cookie для улучшения пользовательского опыта и анализа трафика веб-сайта. Нажав "Принять", вы соглашаетесь с использованием файлов cookie нашего веб-сайта, как описано в нашей [Политике использования файлов cookie](#). Вы можете изменить настройки файлов cookie в любое время, нажав кнопку "[Настройки](#)".

[Предпочтения](#)[Принять](#)

```
/* parent process */  
printf("I am the parent waiting for the child process to end\n");  
wait(NULL);  
printf("parent process is exiting\n");  
return(0);  
}
```

Результатом будет что-то вроде:

```
I am: 2337  
fork returned: 2338  
I am the parent waiting for the child process to end  
fork returned: 0  
I am the child with pid 2338  
Child process is exiting  
parent process is exiting
```

На сегодня все! 🎉

Заключение

На уникальных примерах мы узнали, что может сделать `fork()`, и как это реализовать на языке программирования Си. Если вас больше интересуют

Мы используем файлы cookie для улучшения пользовательского опыта и анализа трафика веб-сайта. Нажав "Принять", вы соглашаетесь с использованием файлов cookie нашего веб-сайта, как описано в нашей [Политике использования файлов cookie](#). Вы можете изменить настройки файлов cookie в любое время, нажав кнопку "[Настройки](#)".

Предпочтения

Принять

1. [Подробнее о fork](#)
2. [Труба с вилкой](#)
3. [Что такое семафор](#)
4. [Введение в семафор](#)

Материалы экспертной оценки: [Saiharsha Balasubramaniam](#)

Did you find this article helpful?



14

0 Comments

Sort By Best ▼

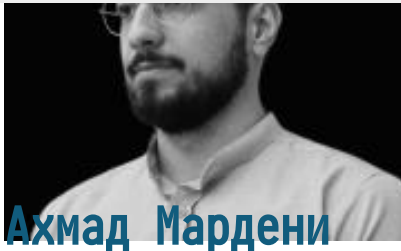
The EngEd community is subject to Section's [moderation policy](#).

Be the first to comment...

[LOGIN](#) [SIGNUP](#)

Мы используем файлы cookie для улучшения пользовательского опыта и анализа трафика веб-сайта. Нажав "Принять", вы соглашаетесь с использованием файлов cookie нашего веб-сайта, как описано в нашей [Политике использования файлов cookie](#). Вы можете изменить настройки файлов cookie в любое время, нажав кнопку "[Настройки](#)".

[Предпочтения](#)[Принять](#)

**Ахмад Мардени**

Ахмад - страстный инженер-программист. Имеет опыт работы в области машинного обучения и Data Science. Он выиграл несколько хакатонов и конкурсов по программированию. Он считает, что Web3 лежит в основе интернета ценностей и возвращает пользователю контроль.

[Посмотреть полный профиль автора](#) →

Join our Slack community

Add to Slack

Company

About

Careers

Legals

Resources

Мы используем файлы cookie для улучшения пользовательского опыта и анализа трафика веб-сайта. Нажав "Принять", вы соглашаетесь с использованием файлов cookie нашего веб-сайта, как описано в нашей [Политике использования файлов cookie](#). Вы можете изменить настройки файлов cookie в любое время, нажав кнопку "[Настройки](#)".

Предпочтения

Принять

Partners



Support


[Docs](#)


[Community Slack](#)


[Help & Support](#)


[Platform Status](#)

Section supports many open source projects including:

 [varnish cache logo](#)

 [cloud native computing foundation logo](#)

 [the linux foundation logo](#)

 [lf edge logo](#)



© 2021 Section

[Privacy Policy](#) [Terms of Service](#)

Мы используем файлы cookie для улучшения пользовательского опыта и анализа трафика веб-сайта. Нажав "Принять", вы соглашаетесь с использованием файлов cookie нашего веб-сайта, как описано в нашей [Политике использования файлов cookie](#). Вы можете изменить настройки файлов cookie в любое время, нажав кнопку "[Настройки](#)".

Предпочтения

Принять