race condition

概述

系统运行的结果有时会受到事件发生先后顺序的影响。当这些事件如果不受控制，没有按照开发者想要的顺序执行就可能会出现bug，这种情况就是条件竞争。在计算机中较为容易出现条件竞争的程序是多线程程序和分布式程序。

条件

- 并发，至少存在两个并发的执行流，包括线程，进程，任务等。
- 共享资源，多个并发流回访问同一个对象。常见的共享对象有共享内存，文件系统，信号。一般来说，这些共享对象是用来使得多个程序执行流相互交流。此外，我们称访问共享对象的代码为临界区。
- 改变对象，至少有一个控制流回改变竞争对象的状态。如果只有读操作，那么不同顺序执行的结果是一样的，就不会产生条件竞争。

影响

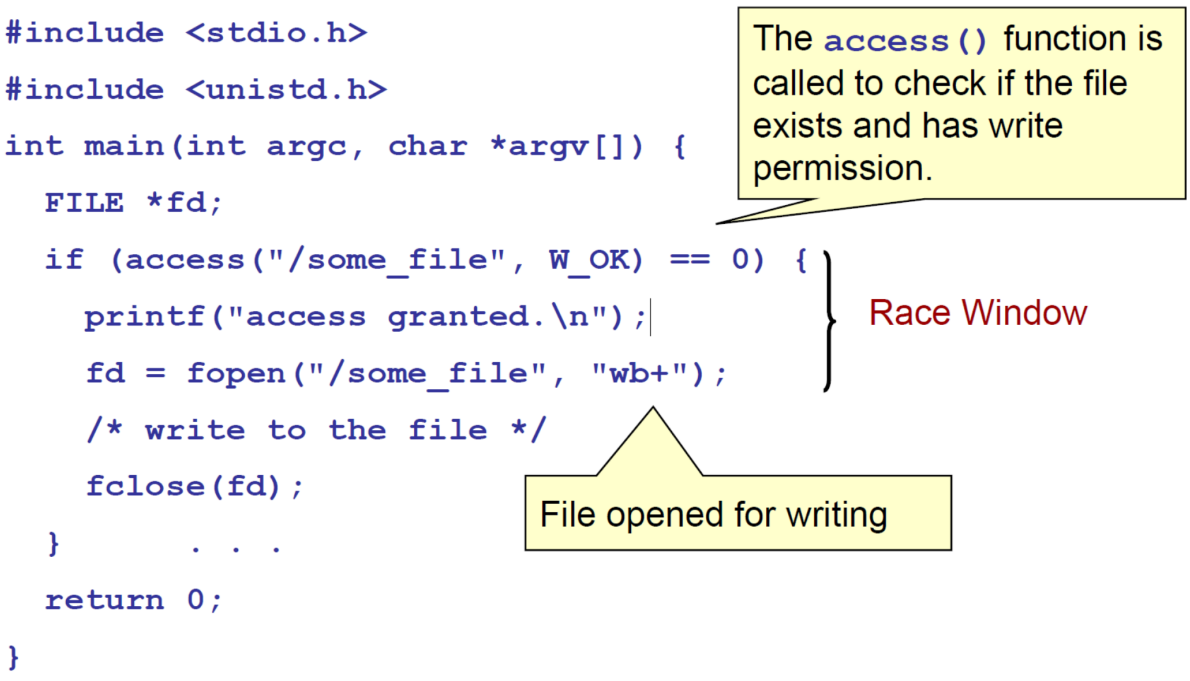
由于在并发时，执行流的不确定性很大，条件竞争相对难察觉，并且在复现和调试方面会比较困难。这给修复条件竞争也带来了不小的困难。条件竞争造成的影响也是多样的，轻则程序异常执行，重则程序崩溃。如果条件竞争漏洞被攻击者利用的话，很有可能会使得攻击者获得相应系统的特权。

形式

TOCTOU

TOCTOU (Time-of-check Time-of-use) 指程序在使用资源前会做检查，但是当检查过了之后再使用该资源时资源被修改了。

例如，程序在访问某个文件之前，会检查是否存在，之后会打开文件然后执行操作。但是如果在检查之后，真正使用文件之前，攻击者将文件修改为某个符号链接，那么程序将访问错误的文件。



例子

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
int index =0;
char content[60];
void *t2(void *arg){
   puts("break in!");
   for(;;){
       {
          if (index == 30)
              index = 0;
          usleep(rand()%2);
          printf("%d%c",*(int*)arg,content[index++]);
          usleep(rand()%2);
       }
       if(index > 60)
       {
           puts("get it");
           exit(0);
       }
   }
   puts("t2 exit");
}
```

```
int main(){
   setvbuf(stdout, 0, 2, 0);
   FILE *f1 = fopen("secret.txt","r");
   FILE *f2 = fopen("/tmp/common.txt","r");
   fread(content,30,1,f2);
   fread(content+30,30,1,f1);
   fclose(f1);
   fclose(f2);
   pthread_t th1;
   pthread_t th2;
   int pstr1 = 1;
   int pstr2 = 2;
   int * th_ret = NULL;
   int ret1;
   int ret2;
   ret1 = pthread_create(&th1,NULL,t2,&pstr1);
   ret2 = pthread_create(&th2,NULL,t2,&pstr2);
   pthread_join(th1,&th_ret);
   pthread_join(th2,&th_ret);
   return 1;
}
```

防范

同步原语

- 锁变量

  确保同一时间只有一个线程或进程可以访问该资源。访问共享资源之前获取锁，访问完成后释放。

- 信号量

  可以设置允许访问资源的线程数量，当资源被占用时信号量减少，资源释放时信号量增多。

- 原子操作

  使用原子操作来确保对共享资源的操作时不可分割的，可以保证在多线程环境中对共享资源的操作时原子性的，不会被其他线程中断。

- 避免共享数据

  尽可能避免多个线程或进程共享同一块数据。如果数据不需要共享，尽量使用局部变量来避免。

- 同步机制和顺序化访问

通过使用同步机制和严格的访问顺序，对共享资源进行有序的访问。确保每个线程按照特定的顺序访问共享资源，避免并发执行导致的问题。