- 进程
- 线程
- 协程

1、什么是进程

- 进程是计算机中的程序关于某数据集合的一次运行活动，是系统进行资源分配和调度的基本单位，是操作系统结构的基础。
- 在早期面向进程设计的计算机结构中，进程是程序的基本执行实体；
- 在当代面向进程设计的计算机结构中，进程是线程的容器。
- 程序是指令、数据及其组织形式的描述，进程是程序的实体。
- 狭义定义：进程是正在运行的程序的实例。
- 广义定义：进程是一个具有一定独立功能的程序关于某个数据集合的一次运行活动。它是操作系统动态执行的基本单元，在传统的操作系统中，进程既是基本的分配单元，也是基本的执行单元。
- 注意：同一个程序执行两次，就会在操作系统中出现两个进程，所以我们可以同时运行一个软件，分别做不同的事情也不会混乱。

2、进程调度

- 要想多个进程交替运行，操作系统必须对这些进程进行调度，这个调度也不是随机进行的，而是需要遵循一定的法则，由此就有了进程的调度算法。
  - 先来先服务调度算法
  - 短作业优先调度算法
  - 时间片轮转法
  - 多级反馈队列

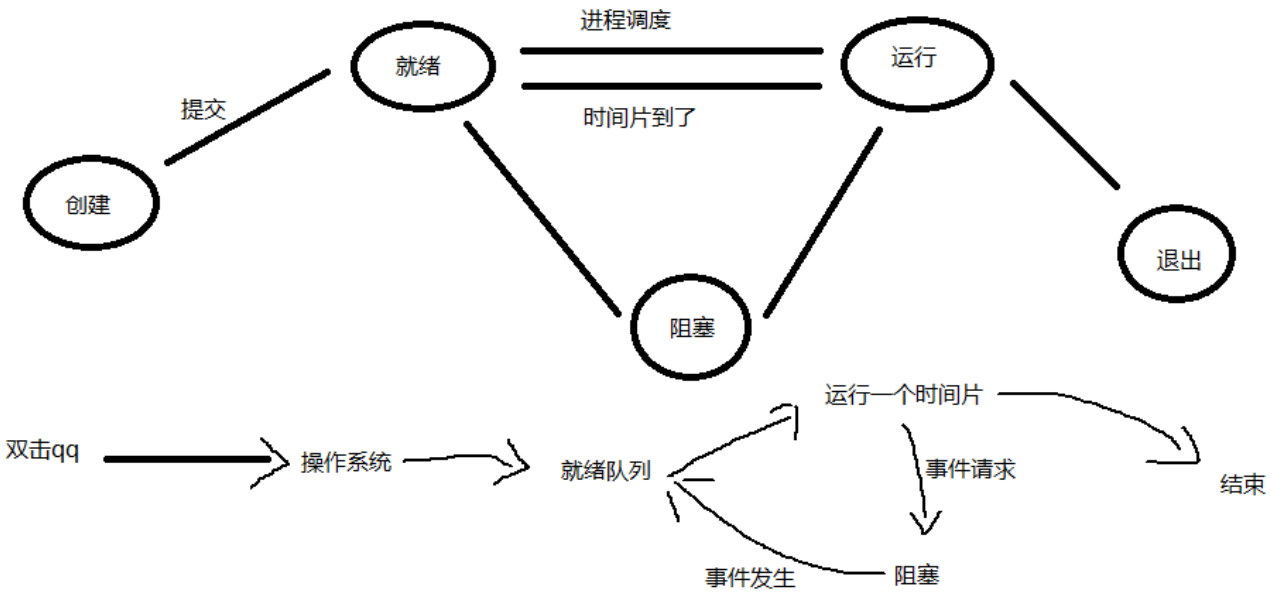
3、进程的并行和并发

a. 并行：并行是指两者同时执行，比如赛跑，两个人都在不停的往前跑；
   i. （资源够用，比如三个进程，四核的cpu）
b. 并发：并发是指资源有限的情况下，两者交替轮流使用资源，比如一座桥（单核cpu）同时只能过一个人，A走一段后，让给B，B用完继续给A，交替使用，目的是提高效率。
c. 区别：
   i. 并行是从微观上，也就是在一个精确的时间片刻，有不同的程序在执行，这就要求必须有多个处理器。

ii. 并发是从宏观上，在一个时间段上可以看出是同时执行的，比如一个服务器同时处理多个请求。

4、进程的状态

- 就绪
- 运行
- 阻塞



- 就绪状态：
  - 当进程已分配到除CPU以外的所有必要的资源，只要获得处理机可立即执行，这时的进程状态称为就绪状态。
- 执行、运行
  - 当程序已获得处理机，其程序正在处理机上执行，此时的进程状态称为执行状态
- 阻塞
  - 由于等待某个事件发生而无法执行时，便放弃处理机而处于阻塞状态。引进进程阻塞的事件可有多种，例如，等待I/O完成、申请缓冲区不能满足、等待信号等

5、同步异步

- 所谓同步就是一个任务的完成需要依赖另一个任务时，只有等待被依赖的任务完成后，依赖的任务才能完成，这是一种可靠的任务序列。要么成功都成功，失败都失败，两个任务的状态可以保持一致。
- 所谓异步是不需要等待被依赖的任务完成，只是通知被依赖的任务要完成什么工

作，依赖的任务也立即执行，只要自己完成了整个任务就算完成了，至于被依赖的任务最终是否完成，依赖它的任务无法确定，所以它是不可靠的任务序列。

```
# 同步    两件事      一件做完了去做另一件事
# 异步    两件事      同时做
```

6、进程的创建

- 但凡是硬件，都需要有操作系统去管理，只要有操作系统，就有进程的概念，就需要有创建进程的方式。
- 对于通用操作系统（跑很多应用程序），需要有系统运行过程中创建或撤销进程的能力，主要分成4种形式创建新的进程：
  - 系统初始化（运行在后台并且只在需要时才唤醒的进程）
  - 一个进程在运行过程中开启了子进程（如nginx开启多进程）
  - 用户的交互请求，而创建一个进程（如双击暴风影音）
  - 一个批处理作业的初始化（只在大型机的批处理系统中应用）
- 无论哪一种，新进程的创建都是由一个已近存在的进程执行了一个用于创建进程的系统调用而创建的。

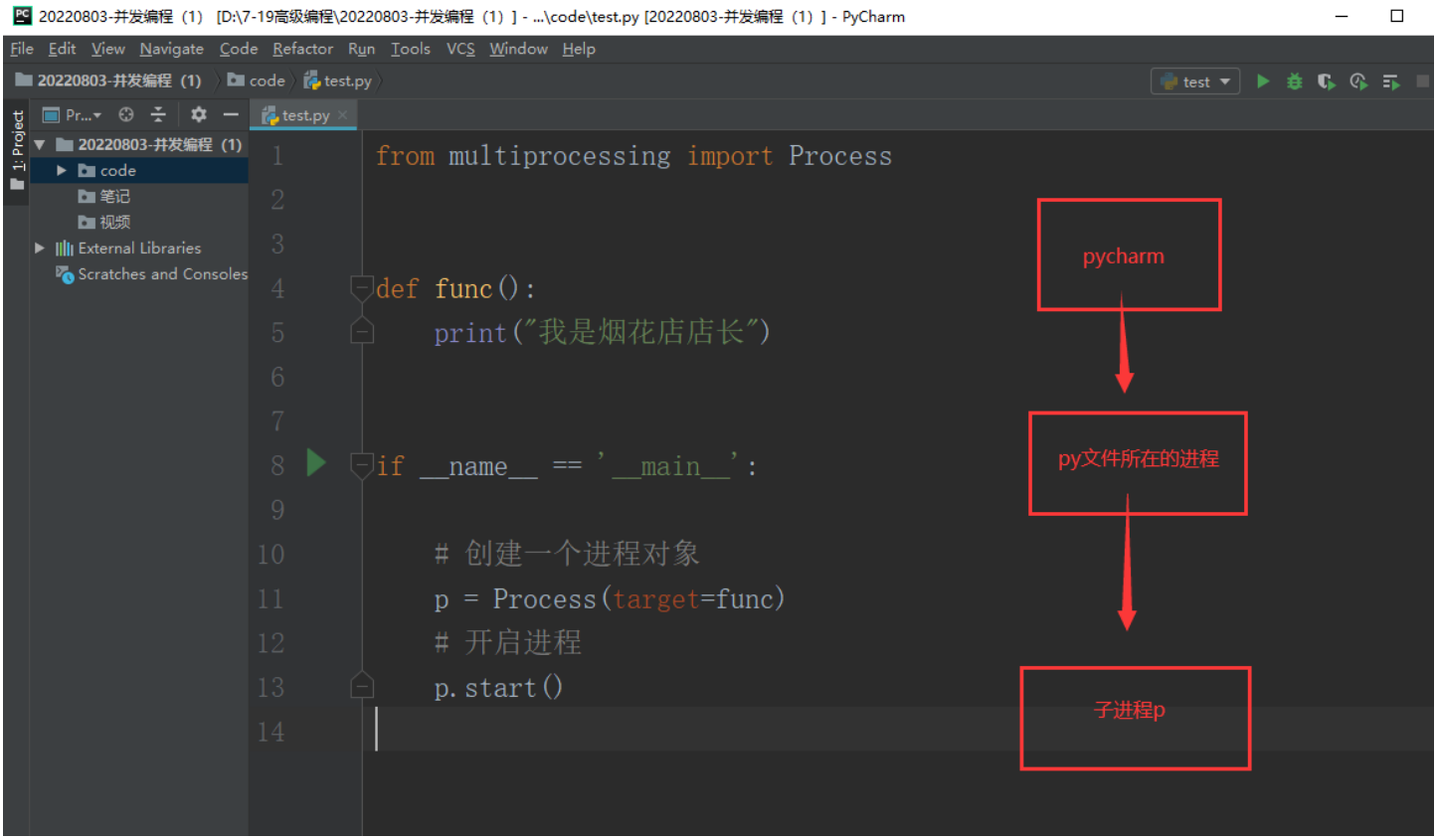
- 进程的结束
- 正常退出
- 出错退出
- 严重错误
- 被其他进程杀死

7、python程序中进程的操作

- 之前我们已经了解了很多进程相关的理论知识，了解进程是什么应该不再困难了，刚刚我们已经了解了，运行中的程序就是进程。所有的进程都是通过它的父进程来创建的。因此，运行起来的Python程序也是一个进程，那么我们也可以在程序中再创建进程。多个进程可以实现并发效果，也就是说，当我们的程序中存在多个进程的时候，在某些时候，就会让程序的执行速度变快。以我们之前所学的知识，并不能实现创建进程这个功能，所以我们就需要借助Python程序中强到大的模块。
- multiprocess包

- 仔细来说，multiprocess不是一个模块而是Python中一个操作、管理进程的包。之所以叫multi是取自multiple的多功能的意思，在这个包中几乎包含了和进程有关的所有子模块。由于提供的子模块非常多，为了方便大家归类记忆我们分成四个部分：创建进程部分、进程同步部分、进程池部分、进程之间数据共享。

- multiprocess.process模块
  - process模块是一个创建进程的模块，借助这个模块，就可以完成进程的创建。



- 这个三个进程是同步还是异步的呢?

```python
import os
import time
from multiprocessing import Process


def func():
    for i in range(10):
        time.sleep(0.5)
        print("子进程", os.getpid(), os.getppid())


if __name__ == '__main__':
    print("主进程", os.getpid(), os.getppid())
```

```
    # 创建一个进程对象
    p = Process(target=func)
    # 开启进程
    p.start()
    for i in range(10):
        time.sleep(0.3)
        print("*" * i)
```

- 从结果看是异步的。

- 为什么要写 if __name__ == '__main__':?
  - （只是在windows上必须写）（原因是不同操作系统之间创建子进程的方式不一样）
- 能不能给子进程传参数?

```
from multiprocessing import Process


def func(num1, num2):
    print(num1 + num2)


if __name__ == '__main__':
    # 创建一个进程对象
    p = Process(target=func, args=(100, 200))

    p.start()
```

- 进程之间的数据隔离问题（数据不共享）

```
from multiprocessing import Process
import time


count = 100

def func():
    global count
    count -= 1
    print("子进程", count)
```

```
11
12
13 if __name__ == '__main__':
14     p = Process(target=func)
15     p.start()
16     time.sleep(5)
17     print("主进程", count)
```

- 如何开启多进程

```
1  from multiprocessing import Process
2  import time
3
4
5  def func(name):
6      print("我是学员{}".format(name))
7
8
9  if __name__ == '__main__':
10
11     names = ["陈绪东", "饭团", "生鲜鱼白", "阿莫西林", "一帆"]
12     for i in range(5):
13         # p = Process(target=func, args=(names[i], ))
14         # p.start()
15         Process(target=func, args=(names[i],)).start()
```

- 子进程和父进程的关系

```
1  from multiprocessing import Process
2  import time
3  import os
4
5  # 子进程和父进程之间的关系
6  def func(arg):
7      print('子进程%s: ' % arg, os.getpid(), os.getppid())
8      time.sleep(5)
9      print('子进程end')
10
11
12 if __name__ == '__main__':
```

```python
    for i in range(10):
        Process(target=func, args=(i,)).start()
    print('父进程***********')


# 父进程和子进程的启动是异步的
# 父进程只负责通知操作系统启动子进程
# 接下来的工作由操作系统接手， 父进程继续执行
# 父进程的代码执行完毕之后并不会直接结束程序
# 而是会等待所有的子进程都执行完毕之后才结束
# 父进程要负责回收子进程的资源
```