1.程序执⾏到os.fork()时，操作系统会创建⼀个新的进程（⼦进程），然后

python基础语⾔

11进 程的创建-fork

复制⽗进程的所有信息到⼦进程中 然后⽗进程和⼦进程都会从fork()函数中得到⼀个返回值，在⼦进程中这 个值⼀定是0，⽽⽗进程中是⼦进程的 id号

在Unix/Linux操作系统中，提供了⼀个fork()系统函数，它⾮常特殊。

普通的函数调⽤，调⽤⼀次，返回⼀次，但是fork()调⽤⼀次，返回两次，因 为操作系统⾃动把当前进程（称为⽗进程）复制了⼀份（称为⼦进程），然 后，分别在⽗进程和⼦进程内返回。

⼦进程永远返回0，⽽⽗进程返回⼦进程的ID。

这样做的理由是，⼀个⽗进程可以fork出很多⼦进程，所以，⽗进程要记下 每个⼦进程的ID，⽽⼦进程只需要调⽤getppid()就可以拿到⽗进程的ID。

2.多进程中，每个进程中所有数据（包括全局变量）都各有拥有⼀份，互 不影响

3.multiprocessing

a.创建⼦进程时，只需要传⼊⼀个执⾏函数和函数的参数，创建⼀个 Process实例，⽤start()⽅法启动，这样创建进程⽐fork()还要简单。

b. join()⽅法可以等待⼦进程结束后再继续往下运⾏，通常⽤于进程间的同 步

4.ProcessProcess语法结构如下：

Process([group [, target [, name [, args [, kwargs]]]]])

target：表示这个进程实例所调⽤对象；

args：表示调⽤对象的位置参数元组；

kwargs：表示调⽤对象的关键字参数字典；

name：为当前进程实例的别名；

group：⼤多数情况下⽤不到；

Process类常⽤⽅法：

is\_alive()：判断进程实例是否还在执⾏；

join([timeout])：是否等待进程实例执⾏结束，或等待多少秒；

start()：启动进程实例（创建⼦进程）；

run()：如果没有给定target参数，对这个对象调⽤start()⽅法时，就将执 ⾏对象中的run()⽅法；

terminate()：不管任务是否完成，⽴即终⽌；

Process类常⽤属性：

name：当前进程实例别名，默认为Process-N，N为从1开始递增的整 数；

pid：当前进程实例的PID值；

5.multiprocessing.Pool常⽤函数解析：

apply\_async(func[, args[, kwds]]) ：使⽤⾮阻塞⽅式调⽤func（并⾏执 ⾏，堵塞⽅式必须等待上⼀个进程退出才能执⾏下⼀个进程），args为 传递给func的参数列表，kwds为传递给func的关键字参数列表；

apply(func[, args[, kwds]])：使⽤阻塞⽅式调⽤func

close()：关闭Pool，使其不再接受新的任务；

terminate()：不管任务是否完成，⽴即终⽌；

join()：主进程阻塞，等待⼦进程的退出，必须在close或terminate之后 使⽤

6.Queue

初始化Queue()对象时（例如：q=Queue()），若括号中没有指定最⼤可接收 的消息数量，或数量为负值，那么就代表可接受的消息数量没有上限（直到 内存的尽头）；

Queue.qsize()：返回当前队列包含的消息数量；

Queue.empty()：如果队列为空，返回True，反之False ；

Queue.full()：如果队列满了，返回True,反之False；

Queue.get([block[, timeout]])：获取队列中的⼀条消息，然后将其从列队 中移除，block默认值为True；

1）如果block使⽤默认值，且没有设置timeout（单位秒），消息列队如果为 空，此时程序将被阻塞（停在读取状态），直到从消息列队读到消息为⽌， 如果设置了timeout，则会等待timeout秒，若还没读取到任何消息，则抛 出"Queue.Empty"异常；

python基础语⾔

29进 程间通信-Queue

2）如果block值为False，消息列队如果为空，则会⽴刻抛 出"Queue.Empty"异常；

Queue.get\_nowait()：相当Queue.get(False)；

Queue.put(item,[block[, timeout]])：将item消息写⼊队列，block默认值 为True；

1）如果block使⽤默认值，且没有设置timeout（单位秒），消息列队如果已 经没有空间可写⼊，此时程序将被阻塞（停在写⼊状态），直到从消息列队 腾出空间为⽌，如果设置了timeout，则会等待timeout秒，若还没空间，则抛 出"Queue.Full"异常；

2）如果block值为False，消息列队如果没有空间可写⼊，则会⽴刻抛 出"Queue.Full"异常；

Queue.put\_nowait(item)：相当Queue.put(item, False)；