**17Python\_10多进程编程+多线程编程+urllib模块+paramiko模块**

**一 多进程编程-forking工作原理**

**1.1 什么是forking**

• fork(分岔)在Linux系统中使用非常广泛

• 当某一命令执行时,父进程(当前进程)fork出一个子进程

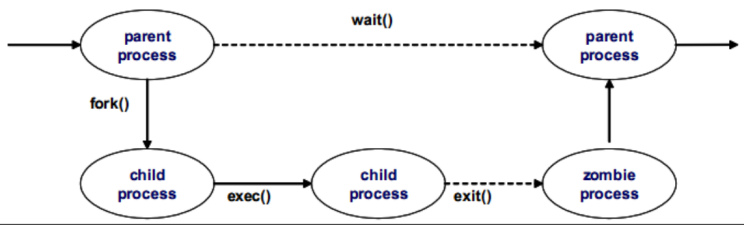
• 父进程将自身资源拷贝一份,命令在子进程中运行时,就具有和父进程完全一样的运行环境

**1.2 进程的生命周期**

• 父进程fork出子进程并挂起

• 子进程运行完毕后,释放大部分资源并通知父进程,这个时候,子进程被称作僵尸进程

• 父进程获知子进程结束,子进程所有资源释放



**1.3 僵尸进程**

• 僵尸进程没有任何可执行代码,也不能被调度

• 如果系统中存在过多的僵尸进程,将因为没有可用的进程号而导致系统不能产生新的进程

• 对于系统管理员来说,可以试图杀死其父进程或重启系统来消除僵尸进程

**二 多进程编程-forking编程**

**• Windows不支持多进程**

**• os.fork返回值在父子进程中是不一样的**

**父进程中,os.fork返回值不是0,而是子进程的PID**

**子进程中,os.fork返回值是0**

**• 多进程编程的思路**

**父进程只负责生成子进程**

**子进程做具体的工作**

**子进程的工作结束后,应该exit退出**

**2.1 forking编程基本思路**

• 需要使用os模块

• os.fork()函数实现forking功能

• python中,绝大多数的函数只返回一次,os.fork将返回两次

• 对fork()的调用,针对父进程返回子进程的PID;对于子进程,返回PID0

**案例:扫描存活主机**

1. 通过ping测试主机是否可达

2. 如果ping不通,不管什么原因都认为主机不可用

3. 通过fork方式实现并发扫描

**ping.py**

import subprocess

import time

import os

def ping(host):

result = subprocess.run(

'ping -c2 %s &> /dev/null' % host, shell=True

)

if result.returncode == 0:

print('%s:\tup' % host)

else:

print('%s:\tdown' % host)

if \_\_name\_\_ == '\_\_main\_\_':

ips = ('176.204.23.%s' % i for i in range(1, 255))

print(time.ctime())

for ip in ips:

retval=os.fork()

if not retval:

ping(ip)

exit()

print(time.ctime())

**案例: 模拟僵尸进程的产生**

**zb.py**

import os

import time

retval=os.fork()

if retval:

print('parent')

time.sleep(45)

print('parent done')

else:

print('child')

time.sleep(15)

print('child done')

# watch -n1 ps a #新开终端执行命令,原终端执行以上代码,查看进程的生成与消失

**2.2 解决zombie问题**

解决zombie问题

• 父进程通过os.wait()来得到子进程是否终止的信息

• 在子进程终止和父进程调用wait()之间的这段时间,子进程被称为zombie(僵尸)进程

• 如果子进程还没有终止,父进程先退出了,那么子进程会持续工作。系统自动将子进程的父进程设置为init进程,init将来负责清理僵尸进程

• python可以使用waitpid()来处理子进程

• waitpid()接受两个参数,第一个参数设置为-1,表示与wait()函数相同;第二参数如果设置为0表示挂起父进程,直到子程序退出,设置为1表示不挂起父进程

• waitpid()的返回值:如果子进程尚未结束则返回0,否则返回子进程的PID

**zb2.py**

import os

import time

retval = os.fork()

if retval:

print('parent')

#挂起父进程,处理完变成僵尸进程的子进程后才继续

result=os.waitpid(-1,0)

print(result) #result是(子进程PID,0)

time.sleep(10)

print('parent done')

else:

print('child')

time.sleep(15)

print('child done')

# watch -n1 ps a

**zb3.py**

import os

import time

retval = os.fork()

if retval:

print('parent')

# 不挂起父进程，子进程是僵尸进程就处理掉，不是则跳过

result = os.waitpid(-1, 1)

print(result) # result是(0, 0)

time.sleep(20)

print('parent done')

else:

print('child')

time.sleep(10)

print('child done')

# watch -n1 ps a

**三 多线程编程-多线程工作原理**

• **各类型操作系统均支持多线程**

• 程序,就是在磁盘上存储的可执行文件

• 当程序运行时,就会出现进程.可以说,进程就是加载内存中的一系列指令。每个进程都拥有自己独立的运行空间。

• 进程还可以拥有多个线程。同一进程内的所有线程,共享进程的运行空间。

• 多线程没有僵尸进程的问题

• 多线程的编程思路:

主线程(类似于父进程),负责生成工作线程

工作线程(类似于子进程),做具体的工作

**3.1 多线程的动机**

• 在多线程(MT)编程出现之前,电脑程序的运行由一个执行序列组成,执行序列按顺序在主机的中央处理器(CPU)中运行

• 无论是任务本身要求顺序执行还是整个程序是由多个子任务组成,程序都是按这种方式执行的

• 即使子任务相互独立,互相无关(即,一个子任务的结果不影响其它子任务的结果)

• 如果并行运行这些相互独立的子任务可以大幅度地提升整个任务的效率

**3.2 多线程任务的特点**

• 它们本质上就是异步的,需要有多个并发事务

• 各个事务的运行顺序可以是不确定的,随机的,不可预测的

• 这样的编程任务可以被分成多个执行流,每个流都有一个要完成的目标

• 根据应用的不同,这些子任务可能都要计算出一个中间结果,用于合并得到最后的结果

**3.3 什么是进程**

• 计算机程序是磁盘中可执行的、二进制(或其它类型)的数据文件

• 进程(有时被称为重量级进程)是程序的一次执行

• 每个进程都有自己的地址空间、内存以及其它记录其运行轨迹的辅助数据

• 操作系统管理在其上运行的所有进程,并为这些进程公平地分配时间

当程序运行时,就会出现进程.可以说,进程就是加载内存中的一系列指令.每个进程都有自己独立的运行空间.

**3.4 什么是线程**

• 线程(有时被称为轻量级进程)跟进程有些相似。不同的是,所有的线程运行在同一个进程中,共享相同的运行环境

• 一个进程中的各个线程之间共享同一片数据空间,所以线程之间可以比进程之间更方便地共享数据以及相互通讯

进程可以拥有多个线程,同一进程内的所有线程,共享进程的运行空间

多线程没有僵尸进程的问题

**四 多线程编程-多线程编程**

**4.1 多线程相关模块**

• thread和threading模块允许程序员创建和管理线程

• thread模块提供了基本的线程和锁的支持,而threading提供了更高级别、功能更强的线程管理功能

• 推荐使用更高级别的threading模块

**4.2 传递函数给 Thread类**

• 多线程编程有多种方法,传递函数给threading模块的Thread类是介绍的第一种方法

• Thread对象使用start()方法开始线程的执行,使用join()方法挂起程序,直到线程结束

**mtping.py(传递函数给Thread类)**

import subprocess

import threading

def ping(host):

result = subprocess.run(

'ping -c2 %s &> /dev/null' % host, shell=True

)

if result.returncode == 0:

print('%s:up' % host)

else:

print('%s:down' % host)

if \_\_name\_\_ == '\_\_main\_\_':

ips = ('176.204.23.%s' % i for i in range(1, 255))

for ip in ips:

# 创建工作线程

t = threading.Thread(target=ping, args=(ip,))

#启动工作线程,就是调用相应的函数,函数结束,工作线程也就结束了

t.start() #调用target(\*args)

(nsd1906) [root@room9pc01 day10]# python3 mtping.py | grep :up | wc -l

19

**4.3 传递可调用类给Thread类**

• 传递可调用类给Thread类是介绍的第二种方法

• 相对于一个或几个函数来说,由于类对象里可以使用类的强大的功能,可以保存更多的信息,这种方法更为灵活

**mtping2.py(传递可调用类给Thread类)**

import subprocess

import threading

class Ping:

def \_\_call\_\_(self,host):

result = subprocess.run(

'ping -c2 %s &> /dev/null' % host, shell=True

)

if result.returncode == 0:

print('%s:up' % host)

else:

print('%s:down' % host)

if \_\_name\_\_ == '\_\_main\_\_':

ips = ('176.204.23.%s' % i for i in range(1, 255))

for ip in ips:

# target是ping的实例

t = threading.Thread(target=Ping(), args=(ip,))

#启动工作线程,就是调用相应的函数,函数结束,工作线程也就结束了

t.start() #调用target(\*args)

**mtping3.py**

import subprocess

import threading

class Ping:

def \_\_init\_\_(self, host):

self.host = host

def \_\_call\_\_(self):

result = subprocess.run(

'ping -c2 %s &> /dev/null' % self.host, shell=True

)

if result.returncode == 0:

print('%s:up' % self.host)

else:

print('%s:down' % self.host)

if \_\_name\_\_ == '\_\_main\_\_':

ips = ('176.204.23.%s' % i for i in range(1, 255))

for ip in ips:

# target是Ping的实例

t = threading.Thread(target=Ping(ip))

t.start() # 调用target(\*args)

**五 urllib模块-urllib基础**

**5.1 urllib简介**

• 在Python2版本中,有urllib和urlib2两个库可以用来实现request的发送。而在Python3中,已经不存在urllib2这个库了,统一为urllib

实现http客户端功能

• urllib中包括了四个模块

– urllib.request可以用来发送request和获取request的结果

– urllib.error:定义错误,实现异常

– urllib.parse用来解析和处理URL

– urllib.robotparse用来解析页面的robots.txt文件

**5.2 爬取网页**

• 先需要导入用到的模块:urllib.request

• 在导入了模块之后,我们需要使用urllib.request.urlopen打开并爬取一个网页

• 读取内容常见的有3种方式:

– read()读取文件的全部内容,read()会把读取到的内容赋给一个字符串变量。

– readlines()读取文件的全部内容,readlines()会把读取到的内容赋值给一个列表变量。

– readline()读取文件的一行内容。

>>> from urllib import request

>>> url='http://www.163.com'

>>> html=request.urlopen(url)

>>> html.readline()

b' <!DOCTYPE HTML>\n'

>>> html.read(10)

b'<!--[if IE'

**5.3 下载网络资源**

• urllib不仅可以下载网页,其他网络资源均可下载

• 有些文件比较大,需要像读取文件一样,每次读取一部分数据

import urllib.request

html=urllib.request.urlopen('http://172.40.50.116/python.pdf')

fobj= open('/tmp/python.pdf','ab')

while True:

data=html.read(4096)

if not data:

break

fobj.write(data)

fobj.close()

**download.py**

from urllib import request

import sys

def download(url, fname):

html = request.urlopen(url)

with open(fname, 'wb') as fobj:

while 1:

data = html.read(4096)

if not data:

break

fobj.write(data)

if \_\_name\_\_ == '\_\_main\_\_':

url = sys.argv[1]

fname = sys.argv[2]

download(url, fname)

**5.4 模拟客户端**

• 有些网页为了防止别人恶意采集其信息所以进行了一些反爬虫的设置,而我们又想进行爬取

• 可以设置一些Headers信息(User-Agent),模拟成浏览器去访问这些网站

>>> url='http://www.jianshu.com'

>>> heads={'Mozilla/5.0 (X11; Linux x86\_64; rv:52.0) Gecko/20100101 Firefox/52.0'}

>>> r=request.Request(url,headers=heads)

>>> html=request.urlopen(r)

>>> html.read()

import urllib.request

url='http://www.tedu.cn'

hearder={

'User-Agent':'Mozilla/5.0 (X11; Fedora; Linux x86\_64)

AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110

Safari/537.36'

}

html=urllib.request.Request(url,headers=header)

data=urllib.request.urlopen(request).read()

**六 urllib模块-urllib进阶**

**6.1 数据编码**

• 一般来说,URL标准中只会允许一部分ASCII字符,比如数字、字母、部分符号等

• 而其他的一些字符,比如汉字等,是不符合URL标准的。此时,我们需要编码。

• 如果要进行编码,可以使用urllib.request.quote()进行

>>> **urllib.request.quote**('hello world!')

'hello%20world%21'

>>> **urllib.request.unquote**('hello%20world%21')

'hello world!'

>>> url='https://www.sogou.com/web?query=元旦'

>>> html=request.urlopen(url) #报错,因为汉字是不允许的

>>> url='https://www.sogou.com/web?query=' + **request.quote**('元旦放假')

>>> url

'https://www.sogou.com/web?query=%E5%85%83%E6%97%A6%E6%94%BE%E5%81%87'

#将url的值复制到浏览器地址栏,能正常访问

**6.2 HTTP异常处理**

• 如果访问的页面不存在或拒绝访问,程序将抛出异常

• 捕获异常需要导入urllib.error模块

>>> html = urllib.request.urlopen('http://172.40.50.116/a.html')

urllib.error.HTTPError: HTTP Error 404: Not Found

>>> html = urllib.request.urlopen('http://172.40.50.116/aaa')

urllib.error.HTTPError: HTTP Error 403: Forbidden

**使用wget模块**

(nsd1906) [root@room9pc01 day10]# pip install wget

>>> import wget

>>>url='https://img01.sogoucdn.com/app/a/07/80f2e20a98944a0fa662d3fc5f472e56'

>>> wget.download(url,'/tmp/sea.jpg')

'/tmp/sea.jpg'

(nsd1906) [root@room9pc01 day10]# eog /tmp/sea.jpg

**下载网易首页上的所有图片**

•找到所有图片的url; •下载

1. 下载网页首页,保存为一个文件
2. 从网易文件的每一行

**getweb.py**

import wget

import os

import re

def get\_url(fname,patt,encoding=None):

result=[]

cpatt=re.compile(patt)

with open(fname,encoding=encoding) as fobj:

for line in fobj:

m=cpatt.search(line)

if m:

result.append(m.group())

return result

if \_\_name\_\_ == '\_\_main\_\_':

img\_dir='/tmp/163'

fname163='/tmp/163.163.html'

url163='http://www.163.com'

#如果不存在保存图片的目录,则创建

if not os.path.exists(img\_dir):

os.mkdir(img\_dir)

#如果网页首页文件不存在,则下载

if not os.path.exists(fname163):

wget.download(url163,fname163)

#取出网易首页中所有的图片地址

img\_patt='(http|https)://[-\w/.]+\.(jpg|png|jpeg|gif)'

#网页首页使用的是gbk编码

img\_list=get\_url(fname163,img\_patt,'gbk')

print(img\_list)

#下载图片

for url in img\_list:

wget.download(url,img\_dir)

**七 paromiko模块**

**7.1 安装paramiko模块**

**• 本地安装**

# yum install -y gcc gcc-c++ python-devel

# tar xzf paramiko-1.15.4.tar.gz

# python setup.py install

**• 网络安装**

(nsd1906) [root@room9pc01 day10]# pip install paramiko

**7.2 基础使用介绍**

• SSHClient

– 创建用于连接ssh服务器的实例

>>> ssh = paramiko.SSHClient()

• paramiko.AutoAddPolicy

– 设置自动添加主机密钥

• ssh.connect

– 连接ssh服务器

• ssh.exec\_comand

– 在ssh服务器上执行指定命令

**7.3 paramiko实例**

• 编写用于实现ssh访问的脚本

– 创建SSHClient实例;– 设置添加主机密钥策略;– 连接ssh服务器

– 执行指定命令;– 在shell命令行中接受用于连接远程服务器的密码以及在远程主机上执行的命令

(nsd1906) [root@room8pc16 day02]# pip install paramiko\_pkgs/\*

>>> import paramiko

**创建SSHClient对象**

>>> ssh = **paramiko.SSHClient()**

自动接收服务器发来的密钥，相当于是自动回答yes

>>> ssh.set\_missing\_host\_key\_policy(**paramiko.AutoAddPolicy()**)

**#连接SSH服务器并登陆**

>>> **ssh.connect**('192.168.1.10',username='root', password='123456')

**执行命令**

>>> result = **ssh.exec\_command**('id root; id zhangsan')

**分析返回结果，result是一个长度为3的元组,保存在内存中**

>>> result

(<paramiko.ChannelFile from <paramiko.Channel 0 (closed) -> <paramiko.Transport **at 0x90f45fd0** (cipher aes128-ctr, 128 bits) (active; 0 open channel(s))>>>, <paramiko.ChannelFile from <paramiko.Channel 0 (closed) -> <paramiko.Transport **at 0x90f45fd0** (cipher aes128-ctr, 128 bits) (active; 0 open channel(s))>>>, <paramiko.ChannelFile from <paramiko.Channel 0 (closed) -> <paramiko.Transport **at 0x90f45fd0** (cipher aes128-ctr, 128 bits) (active; 0 open channel(s))>>>)

>>> len(result)

3

**result中的三项分别是输入、输出和错误的类文件对象。**

>>> stdin, stdout, stderr = result ??? stdin不能读出吗?

**读取输出信息**

>>> out = stdout.read()

**读取错误信息**

>>> err = stderr.read()

>>> out

b'uid=0(root) gid=0(root) \xe7\xbb\x84=0(root)\n'

>>> err

b'id: zhangsan: no such user\n'

**将bytes转为str**

>>> out.decode()

'uid=0(root) gid=0(root) 组=0(root)\n'

**关闭连接**

>>> ssh.close()

**rcmd.py**

**import** paramiko  
  
**def** rcmd(host, user=**'root'**, passwd=**None**, port=22, cmd=**None**):  
 ssh = paramiko.SSHClient()  
 ssh.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())  
 ssh.connect(host, username=user, password=passwd)  
 stdin, stdout, stderr = ssh.exec\_command(cmd)  
 out = stdout.read()  
 err = stderr.read()  
 **if** out: #如果有输出  
 print(**'[%s] OUT:\n%s'** % (host, out.decode()))  
 **if** err: #如果有输出  
 print(**'[%s] ERROR:\n%s'** % (host, err.decode()))  
 ssh.close()  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 rcmd(**'192.168.1.10'**, passwd=**'123456'**, cmd=**'id root;id zhangsan'**)

案例:多线程ssh并发访问

• 编写脚本程序

1. 在文件中取出所有远程主机IP地址

2. 在shell命令行中接受远程服务器IP地址文件、远程服

务器密码以及在远程主机上执行的命令

3. 通过多线程实现在所有的远程服务器上并发执行命令