tedu_devops_day01

多进程

linux采用fork方式执行程序。当运行程序（命令）时，系统将会把父进程的资源拷贝一份，生成子进程。程序、命令在子进程中运行。当命令执行完毕后，子进程销毁。windows系统不支持多进程。

执行shell的方式

- bash xxx.sh: 指定使用bash解释脚本，fork执行
- ./xxx.sh: 根据脚本第一行指定的解释器选择解释方法，fork执行
- source xxx.sh: 在当前进程中执行指令，不采用fork方式

os.fork()方法

os.fork()针对父子进程都有返回值。父进程的返回值是非0值（子进程的ID号）， 子进程返回0

os.fork()编程思路

- 父进程负责生成子进程
- 子进程负责做具体的工作
- 子进程工作结束后需要彻底结束

僵尸进程

当子进程已经结束，但是父进程还未结束，父进程又没有处理僵尸进程的代码，僵尸进程就会一直存在，直到父进程工作结束。如果父进程结束了，子进程会变成孤儿进程，它将会被 systemd接管。如果systemd发现子进程是僵尸进程，就会处理。

多线程

程序是计算机硬盘上存储的一些可执行文件，当程序运行后，就会加载到内存，产生进程。所以进程也可说是加载到内存中的一系列指令，一个进程可以包含很多线程。每个进程都有自己独立的运行环境，如内存等；但是同一进程内的多个线程，共享进程的运行环境。

多线程编程思路

- 主线程生成工作线程
- 工作线程做具体工作。当工作线程工作完成后，它就自动结束了，也不会产生僵尸进程。

urllib模块

```
>>> from urllib import request
>>> html = request.urlopen('https://upload-images.jianshu.io/upload_images/12347101-
bc5e84e92e23c692.jpg')
>>> data = html.read()
>>> with open('/tmp/fork.jpg', 'wb') as fobj:
...     fobj.write(data)
[root@room8pc16 day01]# eog /tmp/fork.jpg
```

模拟客户端访问

当客户机访问服务器时，客户机发送的请求头中使用User-Agent说明自己用的客户端是什么。服务器将在它的访问日志中记录。可以修改请求头，模拟正常的客户端访问，而不是通过python代码的访问。

```
[root@room8pc16 httpd]# tail -f /var/log/access_log
>>> from urllib import request
>>> headers = {'User-Agent': "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101
Firefox/52.0"}
>>> req = request.Request(url='http://127.0.0.1', headers=headers)
>>> html = request.urlopen(req)
```

编码

url只允许一部分字符，如果需要用到其他字符，需要对这些字符进行编码。 如果直接使用汉字，将会报错：

```
>>> html = request.urlopen('https://www.sogou.com/web?query=中国')
```

需要进行以下转换：

```
>>> url = 'https://www.sogou.com/web?query=' + request.quote('中国')
>>> url
'https://www.sogou.com/web?query=%E4%B8%AD%E5%9B%BD'
>>> html = request.urlopen(url)
```

异常处理

- http://127.0.0.1/abc 不存在
- http://127.0.0.1/ban 无权限

wget模块

```
[root@room8pc16 ~]# pip3 install wget
>>> import wget
>>> wget.download('https://upload-images.jianshu.io/upload_images/12347101-
bc5e84e92e23c692.jpg', '/tmp')
```