软件开发过程中平台兼容性分析

Windows 和 linux 之间的兼容性开发

叶茂青

June 19, 2020

总览

什么是兼容

- 二进制兼容：对于一个可执行文件，不需要做任何修改就可以直接运行
- 源码兼容：在新的系统上，不需要更改源码，只需要重新编译或利用解释器即可运行
- 向前兼容：新程序能接受老程序的输入
- 向后兼容：老程序能接受新程序的输入

1 什么是兼容

2 常见的 Windows 和 Linux 之间的行为差异

文件分隔符

```
path = "."

filename = "ppt.tex"

file = path + "\\" + filename

with open(file) as f:
    print(f.readline())
```

- 由于 DOS 系统中斜杆被表示为命令行参数
- Windows 使用\作为文件的分隔符
- 而 Linux 使用/作为文件分隔符

文件分隔符

```
import os

path = "."

filename = "ppt.tex"

file = os.path.join(path, filename)

with open(file) as f:
    print(f.readline())
```

解决方案：

- 统一用/
- 交给 os 库处理

换行符

```
with open("1.txt") as f1, open("2.txt") as f2:
    for _ in range(3):
        print(f1.readline() == f2.readline())
```

换行符

```
with open("1.txt") as f1, open("2.txt") as f2:
    for _ in range(3):
        print(f1.readline() == f2.readline())
```

code > 1.txt

```
1  line1
2  line2
3  line3
```

code > 2.txt

```
1  line1
2  line2
3  line3
```

```
False
False
True
```

多进程

```
from multiprocessing import Process

class T():
    def __init__(self, val):
        self.val = val

def f():
    print(t.val)

if __name__ == '__main__':
    t = T(42)

    f()

    p = Process(target=f, args=())
    p.start()
```

Linux 中由于创建新进程会使用 fork()，子进程可以直接获取父进程中的变量

多进程

```
from multiprocessing import Process

class T():
    def __init__(self, val):
        self.val = val

def f(t):
    print(t.val)

if __name__ == '__main__':
    t = T(42)

    f(t)

    p = Process(target=f, args=(t,))
    p.start()
```

在 Windows 中应该显式的进行传递或使用共享内存

Thank You!