# 操作系统作业2 学号2022132006 姓名杨佳伲

**2.3.2 进程的创建**

查下java,c等高级语言创建一个进程的语句,有哪些参数,对应什么资源

**Java**：通过ProcessBuilder或Runtime.getRuntime().exec()来创建进程，参数有command: 要执行的命令或可执行文件的路径arg1, arg2, ...: 命令的参数列表。对应资源有环境变量和工作目录，并获取进程的标准输入、输出和错误流。

**C**：通过fork和exec系列函数来创建进程，需要手动处理子进程的创建和等待，可以通过execle等函数显式设置环境变量，并可以通过文件描述符标志来控制文件描述符的继承。

1.写出一种高级语言（C,C++,java，python均可）的进程通信方式及其对应程序语句；

2.写出一种高级语言（C,C++,java，python均可）的线程创建语句,中止语句，编程实现一个线程的创建和推出，该线程的功能是在节目打印出“我是一个线程，线程id为\*\*\*\*，我创建了”，中间打印几行自己的学号，然后显示“线程id，线程终止退出”

答;

1. **选择语言：python**
2. 、管道（pipe）通信

代码示例：

from multiprocessing import Process, Pipe

import time

def consumer(p, name):

left, right = p

left.close()

while True:

try:

baozi = right.recv()

print('%s 收到包子:%s' % (name, baozi))

except EOFError:

right.close()

break

def producer(seq, p):

left, right = p

right.close()

for i in seq:

left.send(i)

time.sleep(1)

else:

left.close()

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

left, right = Pipe()

c1 = Process(target=consumer, args=((left, right), 'c1'))

c1.start()

seq = (i for i in range(10))

producer(seq, (left, right))

right.close()

left.close()

c1.join()

print('进程间通信-管道-主进程')

部分运行代码截图：



1. 、队列

from multiprocessing import Process, Queue

def worker(q):

q.put("Hello from worker")

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

q = Queue()

p = Process(target=worker, args=(q,))

p.start()

p.join()

while not q.empty():

print(q.get())

1. 、共享内存

from multiprocessing import Process, Value, Array

def worker(a, arr):

a.value = 3.1415927

for i in range(len(arr)):

arr[i] = -arr[i]

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

num = Value('d', 0.0) # 创建一个共享的双精度浮点数

arr = Array('i', range(10)) # 创建一个共享的整数数组

p = Process(target=worker, args=(num, arr))

p.start()

p.join()

print(f"num: {num.value}")

print(f"arr: {list(arr)}")

1. 、事件处理
2. 、信号量机制

from multiprocessing import Process, Semaphore

import time

def worker(sem, index):

with sem:

print(f"Worker {index} is in the critical section")

time.sleep(1)

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

sem = Semaphore(2) # 允许两个进程同时进入临界区

processes = []

for i in range(5):

p = Process(target=worker, args=(sem, i))

processes.append(p)

p.start()

for p in processes:

p.join()

6）、套接字（socket）

import socket

def server():

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.bind(('localhost', 9999))

s.listen(1)

conn, addr = s.accept()

print(f"Connected by {addr}")

data = conn.recv(1024)

print(f"Received: {data.decode('utf-8')}")

conn.close()

def client():

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.connect(('localhost', 9999))

s.sendall(b"Hello from client")

s.close()

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

import threading

server\_thread = threading.Thread(target=server)

server\_thread.start()

time.sleep(1) # 等待服务器启动

client()

server\_thread.join()

1. **线程实现（使用高级语言：python）**

**代码示例：**

import threading  
def thread\_function():  
 thread\_id = threading.current\_thread().ident  
 print(f"我是一个线程，线程id为{thread\_id}，我创建了")  
 for \_ in range(3):  
 print('学号：2022132006\n')  
 print(f'线程id{thread\_id}，线程终止退出')  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 thread\_1 = threading.Thread(target=thread\_function)  
 thread\_1.start()  
 thread\_1.join()

部分运行截图：



# 操作系统作业3 学号2022132006 姓名杨佳伲

**2.4 进程同步**

**编程实现生产者-消费者问题，编程语言不限制。要求：**

**打印出每轮操作的以下信息：**

**自己的学号**

**生产者生产的第n个商品被放入缓冲区**

**缓冲区目前空余k位**

**消费者从缓冲区拿走第m个商品**

**作业贴出完整代码,并附上执行截图**

提交作业时，保留以上文字，在每个作业的下面填写自己的答案（代码截图等），作业2-3，同一个文档提交

示例代码：

import threading  
import time  
import queue  
  
BUFFER\_SIZE = 6  
buffer = queue.Queue(BUFFER\_SIZE)  
  
id="2022132006"  
def producer():  
 for n in range(1,11):  
 buffer.put(n) *#put item* k=BUFFER\_SIZE-buffer.qsize() *#residual* print(f"学号：{id}\n生产者生产的第{n}个商品被放入缓冲区\n缓冲区目前空余{k}位") *#print* time.sleep(3)  
  
def consumer():  
 for n in range(1,11):  
 item = buffer.get()  
 print(f"消费者从缓冲区拿走第{item}个商品\n\n") *# print* buffer.task\_done()  
 time.sleep(5)  
  
producer\_thread = threading.Thread(target=producer)  
consumer\_thread = threading.Thread(target=consumer)  
  
producer\_thread.start()  
consumer\_thread.start()  
  
producer\_thread.join()  
consumer\_thread.join()

部分过程运行截图：

