抽烟者问题。假设一个系统中有三个抽烟者进程，每个抽烟者不断地卷烟并抽烟。抽烟者卷起并抽掉一颗烟需要有三种材料：烟草、纸和胶水。一个抽烟者有烟草，一个有纸，另一个有胶水。系统中还有两个供应者进程，它们无限地供应所有三种材料，但每次仅轮流提供三种材料中的两种。得到缺失的两种材料的抽烟者在卷起并抽掉一颗烟后会发信号通知供应者，让它继续提供另外的两种材料。这一过程重复进行。  请用信号量机制编程，实现该问题要求的功能。

**项目4实验报告**

1. **实验内容简介**

一个系统有三个抽烟者进程和两个供应者进程。每个抽烟者需要有三种材料：烟草、纸和胶水，才可吸烟。三个抽烟者分别拥有三种材料中的一种。 供应者每次随机将两种材料放到桌子上，拥有剩下那种材料的抽烟者卷一根烟并抽掉它。材料被使用后，供应者按上述原则继续提供材料。

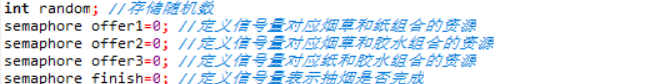
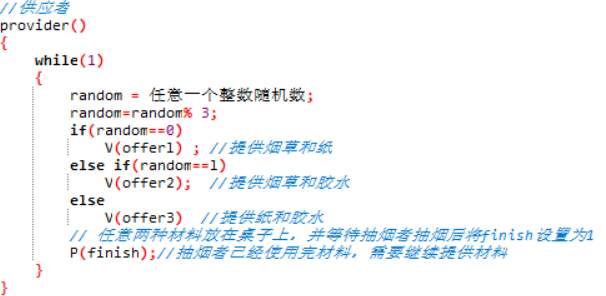
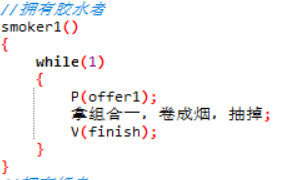
1. **问题分析**

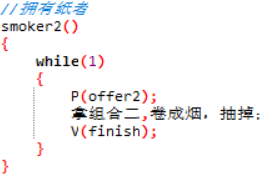
该问题本质是“可生产多种产品的多生产者-多消费者问题”。三个抽烟者相当于三个不同的消费者，他们每次只会有一个抽烟者可以抽到烟，其余两个则需要等待。而在这里默认空间大小只有一个单元，即供应者每次放下两种材料后都会停下来等待直到有消费者使用了这两种材料，他才会继续放另两种材料。相当于说缓冲区的大小为1。

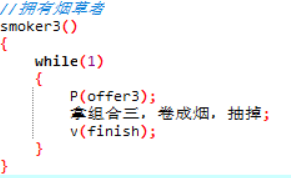
1. 关系分析：供应者与三个抽烟者分别是同步关系。由于供应者无法同时满足两个或 以上的抽烟者，三个抽烟者对抽烟这个动作互斥。
2. 进程分析：这里有五个进程。两个供应者作为生产者向三个抽烟者提供材料。
3. **信号量设置**
4. 信号量offer1、offer2、offer3分别表示烟草和纸组合的资源、烟草和胶水组合的资源、纸和胶水组合的资源。因为开始时没有资源，所以将唤醒消费者的三个信号的信号量初始值都设为0。
5. 随机数设置：生产者要能提供三种组合的原料：烟草，纸，烟草，胶水，或是纸和胶水，这就需要一个随机数，来决定提供哪种组合的原料。这里用random存储随机数。
6. 信号量finish用于互斥进行抽烟动作。finish初始值为0，表示无人抽烟。
7. **操作流程**

缓存区

1. **系统实现**





1. **吸烟者模拟程序测试结果**



**附件：源代码**

#include<iostream>

#include<string>

#include<algorithm>

#include<Windows.h>

#include<process.h>

using namespace std;

int random=0;

HANDLE offer1,offer2,offer3;

HANDLE finish;

unsigned \_\_stdcall threadProvider(void \*)

{

    while(1)

    {

        random=rand()%3;

        if(random == 0) ReleaseSemaphore(offer1,1,NULL);

        else if(random == 1) ReleaseSemaphore(offer2,1,NULL);

        else ReleaseSemaphore(offer3,1,NULL);

        WaitForSingleObject(finish,INFINITE);

    }

    return 1;

}

unsigned \_\_stdcall threadSmoker1(void \*)

{

    while(1)

    {

        WaitForSingleObject(offer1,INFINITE);

        cout<<"Smoker 1 is smoking now."<<endl;

        Sleep(1000);

        ReleaseSemaphore(finish,1,NULL);

    }

    return 2;

}

unsigned \_\_stdcall threadSmoker2(void \*)

{

    while(1)

    {

        WaitForSingleObject(offer2,INFINITE);

        cout<<"Smoker 2 is smoking now."<<endl;

        Sleep(1000);

        ReleaseSemaphore(finish,1,NULL);

    }

    return 3;

}

unsigned \_\_stdcall threadSmoker3(void \*)

{

    while(1)

    {

        WaitForSingleObject(offer3,INFINITE);

        cout<<"Smoker 3 is smoking now."<<endl;

        Sleep(1000);

        ReleaseSemaphore(finish,1,NULL);

    }

    return 4;

}

void process\_Smoker()

{

    offer1=CreateSemaphore(NULL,0,1,NULL);

    offer2=CreateSemaphore(NULL,0,1,NULL);

    offer3=CreateSemaphore(NULL,0,1,NULL);

    finish=CreateSemaphore(NULL,0,1,NULL);

    HANDLE hth1,hth2,hth3,hth4;

    hth1=(HANDLE)\_beginthreadex(NULL,0,threadProvider,NULL,0,NULL);

    hth2=(HANDLE)\_beginthreadex(NULL,0,threadSmoker1,NULL,0,NULL);

    hth3=(HANDLE)\_beginthreadex(NULL,0,threadSmoker2,NULL,0,NULL);

    hth4=(HANDLE)\_beginthreadex(NULL,0,threadSmoker3,NULL,0,NULL);

    WaitForSingleObject(hth1,INFINITE);

    WaitForSingleObject(hth2,INFINITE);

    WaitForSingleObject(hth3,INFINITE);

    WaitForSingleObject(hth4,INFINITE);

    CloseHandle(hth1);

    CloseHandle(hth2);

    CloseHandle(hth3);

    CloseHandle(hth4);

    CloseHandle(offer1);

    CloseHandle(offer2);

    CloseHandle(offer3);

    CloseHandle(finish);

}

int main()

{

    process\_Smoker();

    return 0;

}