# 《现代交换原理》实验报告

实验名称　　摘挂机检测实验

班　　级　　2019211306

学　　号　　2019211397

姓　　名　　毛子恒

指导教师　　赵　学　达

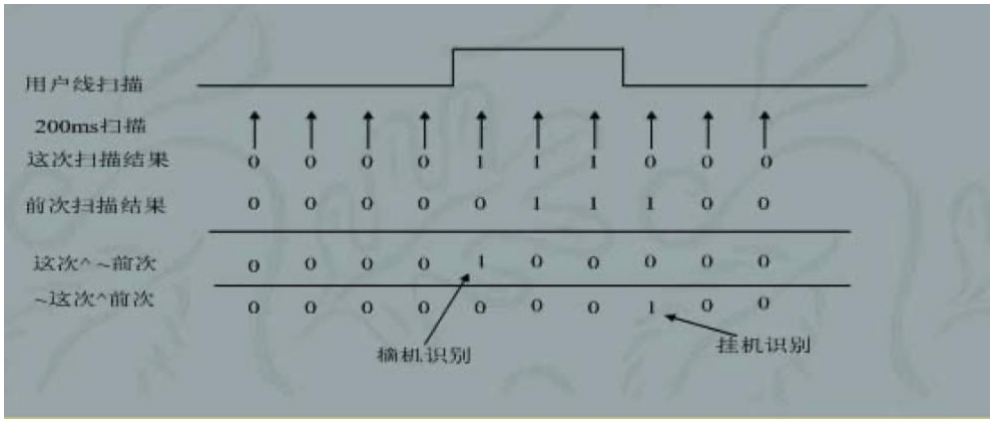
# 实验 2　摘挂机检测实验

## 一、实验目的

摘挂机检测实验用来考查学生对摘挂机检测原理的掌握情况。

## 二、实验内容和实验步骤

设用户在挂机状态时扫描输出为“0”，用户在摘机状态时扫描输出为“1”，摘挂机扫描程序的执行周期为 200ms，那么摘机识别，就是在 200ms 的周期性扫描中找到从“0”到“1”的变化点，挂机识别就是在 200ms 的周期性扫描中找到从“1”到“0”的变化点，该原理的示意图如下所示：




在我们的实验中，我们把前 200ms 的线路状态保存以备这次可以读取，同时读出这次的线路状态，把前 200ms 的线路状态取反与这次的线路状态相与，如果为 1，就说明检测到摘机消息了。同理，我们把这次的线路状态取反再与前 200ms 的线路状态相与，如果为 1 就说明检测到挂机消息了，然后把摘挂机信号作为事件放入摘挂机队列中。

实验主要数据结构：

函数功能为：检测到摘、挂机事件，并把该事件放入到摘挂机事件队列中。

函数原型：`void scanfor200(int linestate200[LINEMAX],int linestate[LINEMAX],UpOnnode * head1, UpOnnode* end1);`

其中 LINEMAX 为线路总数，是定义在 bconstant.h 中的一个宏，`linestate200[LINEMAX]`为已保存的 200ms 前线路状态，`linestate[LINEMAX]`为当前的线路状态，head1,end1 为摘挂机队列的首尾指针，该队列已经在主程序中进行了初始化。我们所要做的就是把检测到的摘挂机事件以摘挂机队列节点的形式插入到摘挂机事件队列中。

数据结构说明：

头文件：bconstant.h (以下的数据结构都已在该文件中定义）

LINEMAX：最大线路数；

`int linestate200[LINEMAX],linestate[LINEMAX]`：线路从 0 开始编号；状态：1：有电流，0 无电流；

`enum UporOn {ehandup,ehandon}`：为摘挂机区别符：ehandup 表示摘机，ehandon 表示挂机；

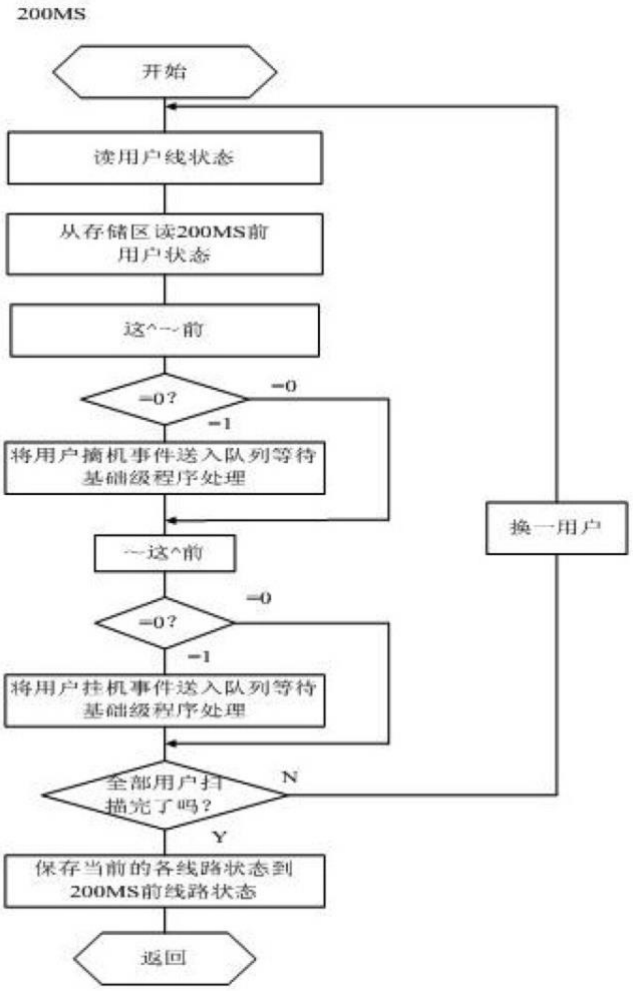
```
struct UpOnnode              //摘挂机队列节点结构
{
    UporOn phonestate;       //摘挂机区别符;
```

```
    int linenum;            //线路号（从 0 开始）；
    struct UpOnnode* next;  //指向下一节点的指针；
};
```

注意事项：

1. 我们编写的模块是基础实验部分预加载的本局交换系统的一个模块而已，在系统中 head1 头指针和 end1 尾指针已经完成初始化。为方便起见，我们的摘挂机事件队列是一个包含头节点的单向链表，并且头指针指向该头节点，尾指针在初始化时也指向了该节点。所以在我们的函数编写中应保证头指针始终指向该头节点上、尾指针指向摘挂机事件队列的最末一个节点。
2. 注意把这次扫描的线路状态值保存在前 200ms 扫描线路状态数组中,以便主程周期调用。

实验主体流程图：




## 三、源代码

```
#include "bconstant.h"

extern "C" _declspec(dllexport) void scanfor200(int
linestate200[LINEMAX], int linestate[LINEMAX], UpOnnode *head1,
UpOnnode *end1)
{
    int up, down;
    for (int i = 0; i < LINEMAX; i++)
    {
        struct UpOnnode *now = new struct UpOnnode;
        up = (!linestate200[i]) && linestate[i];    // !前 & 这 => 摘机
        down = (!linestate[i]) && linestate200[i];  // !这 & 前 => 挂
机
        if (up || down) // 检测到摘机或挂机
        {
            if (up)
                now->phonestate = ehandup;
            else
                now->phonestate = ehandon;
            now->linenum = i;   // 设置线路号
            now->next = 0;      // now 的下个节点为空
            end1->next = now;   // 原队尾的下个节点为 now
            end1 = now;         // now 成为新队尾
        }
    }
    return;
}

extern "C" _declspec(dllexport) void freenode(UpOnnode *node)
{
    delete node;
}
```

## 四、实验结果

能够正常进行通话并且检测摘挂机。

## 五、实验心得

本次实验的代码主要是实现一个简单的队列，很快就可以编写完成。通过本次实验，我对摘挂机检测的原理有了更深的理解。