

# 04-1

栈与队列

队列应用

墙上一溜挂着五个烟斗。张大哥不等旧的已经不能再用了才买新的，而是使到半路就买个新的来；新旧替换着用，能多用些日子。

邓俊辉

deng@tsinghua.edu.cn

# 资源循环分配

❖ 一组客户 ( client ) 共享同一资源时 , 如何兼顾**公平**与**效率** ?

比如 , 多个应用程序共享CPU , 实验室成员共享打印机 , ...

❖ RoundRobin { //循环分配器

Queue Q( clients ); //共享资源的所有客户组成队列

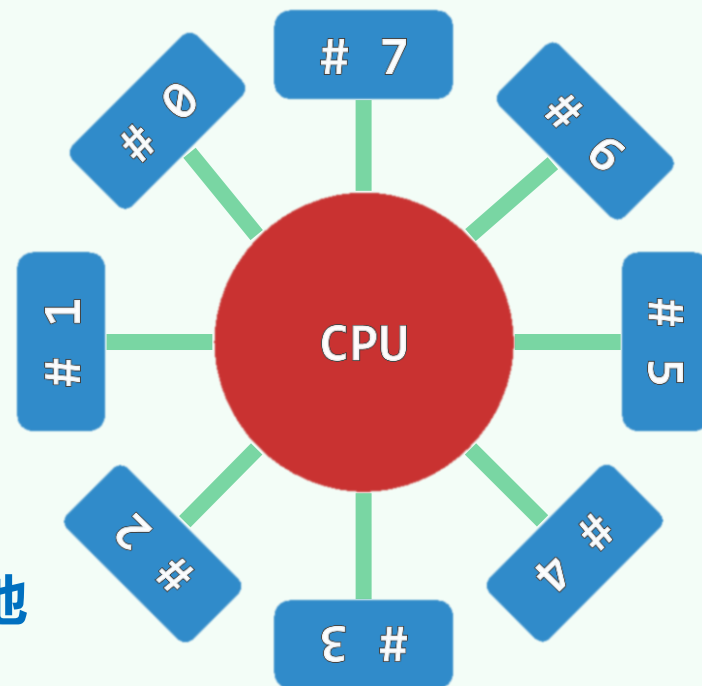
while ( ! ServiceClosed() ) { //在服务关闭之前 , 反复地

    e = Q.dequeue(); //令队首的客户出队 , 并

    serve( e ); Q.enqueue( e ); //接受服务 , 然后重新入队

}

}



# 银行服务模拟：模型

## ❖ 提供n个服务窗口

- 任一时刻，每个窗口至多接待一位顾客  
其他顾客排队等候
- 顾客到达后，自动地  
选择和加入最短队列（的末尾）

## ❖ 参数： nWin //窗口（队列）数目

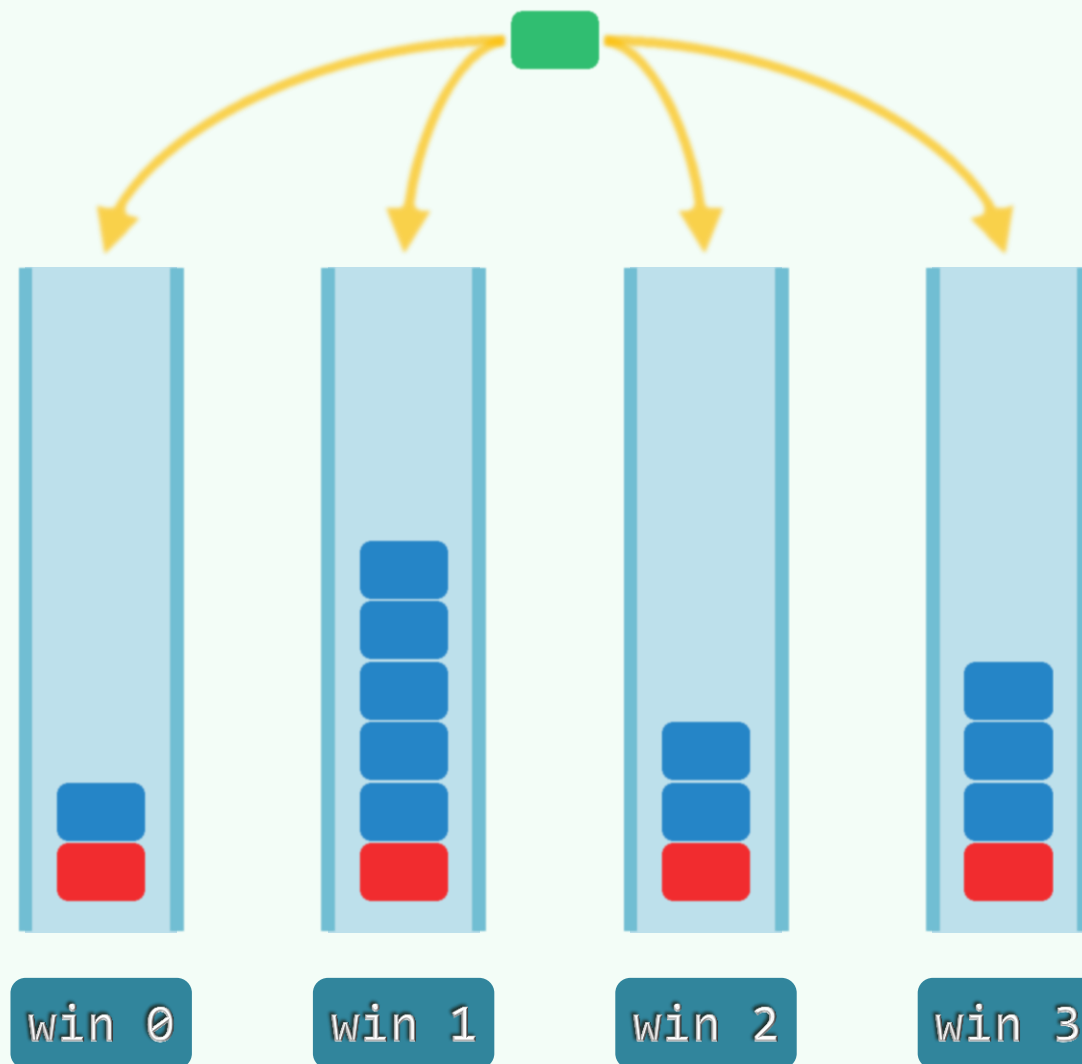
servTime //营业时长

## ❖ struct Customer { //顾客类

int window; //所属窗口（队列）

unsigned int time; //服务时长

};



## 银行服务模拟：实现

```
❖ void simulate( int nWin, int servTime ) {  
    Queue<Customer> * windows = new Queue<Customer>[ nWin ];  
    for ( int now = 0; now < servTime; now++ ) { //在下班之前，每隔单位时间  
        Customer c ; c.time = 1 + rand() % 50; //一位新顾客到达，其服务时长随机指定  
        c.window = bestWindow( windows, nWin ); //找出最佳（最短）服务窗口  
        windows[ c.window ].enqueue( c ); //新顾客加入对应的队列  
        for ( int i = 0; i < nWin; i++ ) //分别检查  
            if ( ! windows[ i ].empty() ) //各非空队列  
                if ( -- windows[ i ].front().time <= 0 ) //队首顾客接受服务  
                    windows[ i ].dequeue(); //服务完毕则出列，由后继顾客接替  
    } //for  
    delete [] windows; //释放所有队列  
}
```