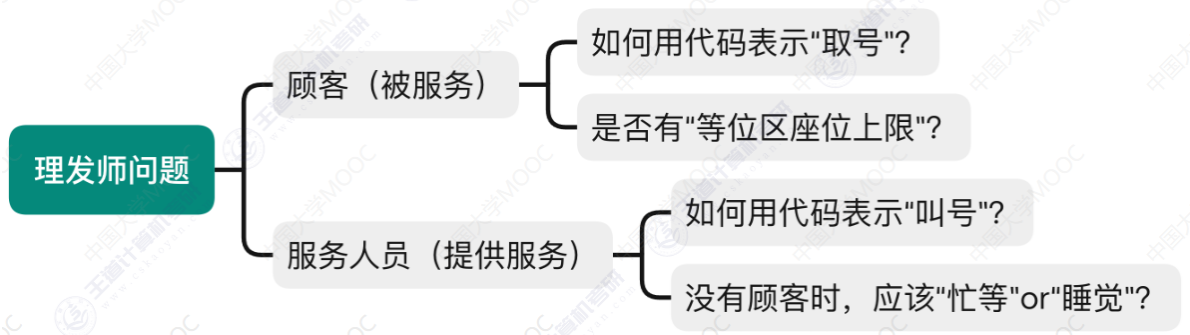


```
int waiting = 0; //正在等待的顾客数量
const int MAX = 10; //“等位区”座位的上限

semaphore customer = 0; //顾客数量（用来无顾客时阻塞自身，即“睡觉”）
semaphore service = 0; //叫号，被服务
semaphore mutex = 1; //互斥访问waiting变量

Customer_i(){ //一个顾客对应一个进程
    P(mutex); //防止多个顾客同时访问时，均检测到有空余名额，导致等位区人数超过MAX
    if (waiting == MAX) { //“等位区”已满，顾客离开
        V(mutex);
        return ; //离开
    }

    取号......
    waiting ++;
    V(mutex);
    V(customer);

    等待被叫号......

    P(service);
    被服务......
}

Server_j(){ //一个服务人员对应一个进程
    while(1){
        P(mutex); //防止多个服务人员同时检测到有顾客，导致一个顾客占用多份资源
        if (waiting > 0) { //当前有顾客
            叫号......
            waiting --;
            V(mutex);

            V(service); //即叫号，可进行服务
            提供服务......
        }
        else{
            V(mutex);
        }
        P(customer); //若无顾客，阻塞（则服务人员“睡觉”），有顾客，解除阻塞
    }
}
```