# 状态表系统设计

1. 流程：

客户端操作->客户端状态判断->客户端条件判断->客户端操作发送到子服务器->子服务器状态判断->子服务器条件判断->子服务器状态切换(客户端状态切换)->子服务器执行操作。

1. 使用：

客户端、服务器都使用CheckOperInState(操作枚举)来判断是否可以执行这个操作。

服务器状态切换调用SetRealStateInGame(操作枚举),此函数自动会同步状态到客户端。

\*也可以通过IsInGameState(状态枚举)查询玩家是否拥有某种状态。

1. 详细说明：

1）添加PlayerState\_In.csv表，表中1表示当前状态可执行当前操作，0表示不可。



2）添加PlayerState\_Set.csv表，表中0表示当前操作对当前状态没有影响，1表示当前操作会添加当前状态，-1表示当前操作会取消当前状态。



3）将PlayerState\_In.csv表中的第2行和第2列使用脚本自动生成状态枚举和操作枚举，保存为PlayerState.h文件。

4）实现CheckOperInState、SetRealStateInGame、IsInGameState、GetState、GetOperate函数。

1. 操作前判断CheckOperInState(操作枚举)然后使用SetRealStateInGame(操作枚举)设置状态，停止移动或者寻路完成调用SetRealStateInGame(操作枚举)，瞬移等也是一样的。
2. 在SetRealStateInGame函数内多加了一次CheckOperInState用于检查之前的CheckOperInState是否调用，如未调用将会写入日志。