3）向场景服务发送 enter 消息（见 3.13.3 节），请求进入场景。如果成功进入场景，会给 s.snode 和 s.sname 赋值。

代码 3-51 service/agent/scene.lua

```
local skynet = require "skynet"
local s = require "service"
local runconfig = require "runconfig"
local mynode = skynet.getenv("node")

s.snode = nil --scene_node
s.sname = nil --scene_id

s.client.enter = function(msg)
    if s.sname then
        return {"enter",1," 已在场景 "}
    end
    local snode, sid = random_scene()
    local sname = "scene"..sid
    local isok = s.call(snode, sname, "enter", s.id, mynode, skynet.self())
    if not isok then
        return {"enter",1," 进入失败 "}
    end
    s.snode = snode
    s.sname = sname
    return nil
end
```

随机选择场景的 random_scene 方法，如代码 3-52 所示。按照 3.2.4 节的分析，agent 应尽可能地进入个同节点的 scene。为模拟合适的匹配机制，random_scene 返回同节点场景服务的概率是其他节点的数倍。

具体做法是，先把所有配置了场景服务的节点都放在表 nodes 中，同一节点（mynode）会插入多次，使它能有更高被选中的概率。插入完成后在 nodes 表随机选择一个节点（scenenode）。再在选出的节点中随机选出一个场景（sceneid）。

代码 3-52 service/agent/scene.lua 中新增的内容

```
local function random_scene()
    -- 选择 node
    local nodes = {}
    for i, v in pairs(runconfig.scene) do
        table.insert(nodes, i)
        if runconfig.scene[mynode] then
            table.insert(nodes, mynode)
        end
    end
    local idx = math.random( 1, #nodes)
    local scenenode = nodes[idx]
    -- 具体场景
    local scenelist = runconfig.scene[scenenode]
    local idx = math.random( 1, #scenelist)
```