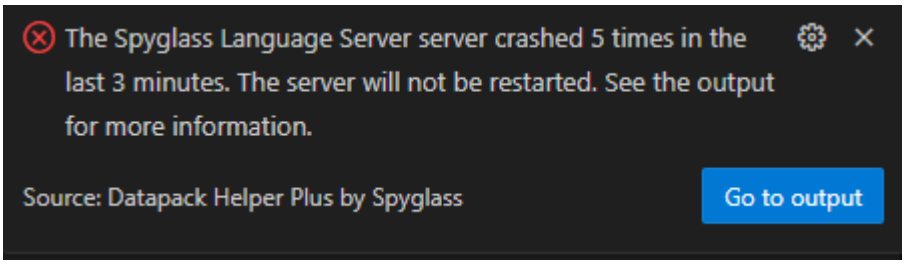
FAQ 常见问题集

目录

Spyglass（或 Datapack Helper Plus）4.0 以上版本无法运行

看到右下角出现这样的弹窗：



大概率为你的网络问题。这里提供三种解决方案，可以依次尝试：

方案（1）人为放置所需数据（目前适用 Windows）：

1. 下载 mc-je.zip
2. Windows 电脑上按 Win+R，在弹出的窗口输入 %localappdata%/spyglassmc-nodejs/
3. 回车。这时应该会弹出一个文件夹。（mac/linux 请自行寻找 spyglassmc-nodejs 文件夹的位置）
4. 打开文件夹里面的 Cache 文件夹（如果没有自行创建）
5. 打开 Cache 文件夹里的 downloader 文件夹（如果没有自行创建）
6. 如果有一个叫 mc-je 的文件夹就把它删除，然后将下载的 mc-je.zip 解压在此处。
7. 确认：downloader 文件夹下面有 mc-je 文件夹，打开 mc-je 以后可以看到各个游戏版本号。（或者再次按 Win+R，输入%localappdata%/spyglassmc-nodejs/Cache/downloader/mc-je/1.21　然后回车，如果正确打开就可以了）
8. 重启 vscode。你的 Spyglass 应该可以正常运行了。如果没有正常运行，请关闭所有其他插件，断网，并重复上述步骤。

方案（2）更改 DNS：

参考本视频

方案（3）尝试换源：

在数据包根目录下创建一个文件，名为 spyglass.json 。这个文件应该和 pack.mcmeta 与 data 文件夹在一个层级。文件包含以下内容：

```
{
  "env": {
    "env": "jsDelivr"
  }
}
```

重启或刷新 Vscode

如何编辑.nbt .dat .mca 文件

NBT 格式是一种二进制文件，游戏使用它来存储绝大部分信息，包括方块，实体，记分板等。
注意命令中使用的 NBT 全称为 SNBT，是 NBT 的字符串形式，与本问题关系不大。

想要在游戏外更改或查看 NBT 文件，需要使用特殊软件，任选其一：

1. Vscode 插件搜索 NBT Viewer
2. 安装 NBT Explorer。你可以在群文件里找到安装包。只支持 Windows

如何入手正版 Minecraft？地图制作需要正版么？

入手正版参见籽岷的视频：https://www.bilibili.com/video/BV1ta4y1R7Hy

地图制作过程中并不需要正版账号。其他事物不在本常见问题集的考虑或担保范围内，自行承担。

如何生成随机数？

不同的版本应当使用不同的方法。
1.20.2 或更高：

使用/random 命令配合/execute store 命令。前者生成随机数，后者把结果存储在记分板内。
假设已有记分板 random：

```
execute store result score test random run random value 1..100
```

本命令将 test 在 random 记分板上的数值设为 1 到 100 之间的随机数字。

详见：
- Wiki：命令/random
- Wiki：命令/execute#store

1.20.1 或更低：

此处推荐的是适用性最广的 UUID 随机法。原理是，当召唤了新实体时，游戏会为其分配一个 UUID。这是一个随机的数字，我们只需要读取这个数字即可。
假设已有记分板 random：

```
scoreboard players set %max random 100
summon marker ~ ~ ~ {Tags:["random"]}
execute store result score $result random run data get entity @e[type=marker,limit=1,tag=random,distance=..5] UUID[0]
kill @e[type=marker,tag=random]
scoreboard players operation $result random %= %max random
```

此时，$result 会被设置为 0 到 99 之间的随机数值。
$max 用来设置此次随机的范围。上文中设置为了 100，范围就是 0~99。设置为 1000 范围就会为 0~999。
在 1.16.5 或更低的版本中，marker 实体尚未存在。只需要把所有 marker 替换成盔甲架(armor_stand)即可。

详见：

- Wiki：命令/execute#store

如何找到某记分板上分数最高的玩家？

假设已有记分板 data：

```
scoreboard players operation $res data = @a[limit=1] data
#首先将$res 设置为任意玩家的分数
execute as @a run scoreboard players operation $res data > @s data
#对于每个玩家，如果自己的分数大于$res，则将$res 设置为自己的分数
execute as @a if score @s data = $res data run say 1
#这时分数等于$res 的玩家就是目标。这里让他 say  1

```

详见：

- Wiki：命令/execute#as
- Wiki：命令/scoreboard
- 教程：/scoreboard

什么是缓存 NBT？为什么要这么做？

使用命令访问玩家，非玩家实体，或方块实体的 NBT 是非常耗费性能的行为，尤其是非玩家实体和方块实体的 NBT，大量的频繁访问可能会出现一定的性能负担。

为了减轻性能负担，在需要大量运算，更改，或判断一串 NBT 时，应当先将其复制到一个 NBT 存储内再进行判定，更改。

例：
若你想检测玩家手中的物品做出对应的行为：

若每次都检测 SelectedItem：

```
## 函数

#如果手拿苹果就说 1
execute if data entity @s SelectedItem{id:"minecraft:apple"} run say 1
#如果手拿石头就说 2
execute if data entity @s SelectedItem{id:"minecraft:stone"} run say 2
#如果手拿泥土就说 3
execute if data entity @s SelectedItem{id:"minecraft:dirt"} run say 3
```

这会访问玩家的 NBT 三次。这时可以使用 NBT 缓存来降低一部分性能负担：

```
## 函数

#首先将玩家手持物品的信息复制到存储 foo:bar 的 temp 中，再直接检测 temp
data modify storage foo:bar temp set from entity @s SelectedItem
execute if data storage foo:bar temp{id:"minecraft:apple"} run say 1
execute if data storage foo:bar temp{id:"minecraft:stone"} run say 2
execute if data storage foo:bar temp{id:"minecraft:dirt"} run say 3
```

看似多使用了一条命令，然而运行速度会快上不少。穷举项越多越明显。

什么是射线追踪？

射线追踪是一种在空间中画直线的手段。基本原理是利用局部坐标不断向前递归一个侦测或功能函数。
想要使用射线追踪，你必须使用数据包/函数，否则将极为麻烦。

例 1 - 侦测玩家是否在看着石头：

```
execute as @a at @s anchored eyes run function raycast
#让所有玩家执行 raycast 函数，即射线检测。注意加入了 anchored eyes，以调整高度为玩家的眼睛高度。
##function raycast (以下为 raycast 函数内容):
```

```
execute if block ~ ~ ~ stone run say 在看石头
#如果本格是石头，说明玩家在看石头
execute unless block ~ ~ ~ stone unless entity @s[distance=..8] run say 没有在看石头，或者距离过远
#如果本格不是石头，且玩家距离当前位置超过 8 格，距离过远结束判定（自行修改最大距离）。
execute unless block ~ ~ ~ stone if entity @s[distance=..8] positioned ^ ^ ^0.1 run function raycast
#如果本格不是石头，玩家在 8 格以内，向前方步进 0.1 格后再次运行本函数（自行修改步进距离）。
```

如果版本至少为 1.20.4，可以使用/return run 命令简化 raycast 函数：

```
execute if block ~ ~ ~ stone run return run say 在看石头
execute unless entity @s[distance=..8] run return run say 没有在看石头，或者距离过远
execute positioned ^ ^ ^0.1 run function raycast
```

例 2 - 发射了一条直线粒子（类似子弹）：

类似地，替换 raycast 函数：

```
particle flame ~ ~ ~ 0 0 0 0 0 force @a

execute if entity @s[distance=..10] positioned ^ ^ ^0.2 run function raycast
```

你应该根据需求修改步进距离。更小的步进距离会增加判定次数，增加精度，但耗费更多性能。

详见：

- Wiki：命令/execute#positioned
- Wiki：命令/execute#anchored
- 教程：命令/tp，相对与局部坐标

如何强制同步实体 Motion

当你更改实体的 Motion 时，可能会观察到实体在运动改变之前先卡了一下。这是因为更改的 Motion 有可能并不会立刻同步到客户端。解决办法十分简单，只需要更改特定的 NBT 就可以了。其中一个是 Air：

```
data modify entity @s Motion set value [0.0d,2.0d,0.5d]

#假设我们更改了当前实体的 Motion。

execute store result entity @s Air short 1 run time query gametime

#立刻更新实体的 Air 来强制同步。这里选择把 gametime 的值填充进去只是为了保证不重复，如果你有内置的时钟之类（比如每秒一循环）会更好。
```

如何阻止 TNT 爆炸破坏方块

方案是把所有即将爆炸的 TNT 替换成即将爆炸的苦力怕，再更改游戏规则 mobGriefing 为 false 来阻止苦力怕破坏方块。

注意修改 mobGriefing 也会导致其他现象的发生，比如凋零等其他怪物均无法破坏方块，猪灵无法捡起地面上的金锭等。如果你不想要这些效果，那么就只能使用数据包完全模拟爆炸的伤害和冲击了，这里不做讨论。

```
#选中所有即将爆炸的 TNT，使其运行函数 fake_tnt：

execute as @e[type=tnt,nbt={fuse:1s}] at @s run function fake_tnt
```

```
##function fake_tnt:

summon creeper ~ ~ ~ {Fuse:1s,ignited:1b,CustomName:'"TNT"',ExplosionRadius:4}

#在自己的位置生成一个苦力怕，它的引信是 1 tick，已经被点燃，爆炸威力 4（同 TNT），且名字叫 TNT。

kill

#清除这个 TNT
```

详见：

- Wiki：TNT#实体数据
- Wiki：苦力怕#实体数据

- Wiki：游戏规则
- 教程：/gamerule

如何检测玩家使用右键

在最新版本中，侦测右键使用的方式已经有非常多了。此处介绍最经典也是最简单的钓竿方法。

胡萝卜钓竿和 1.16.5 新增的诡异菌钓竿，在右键时会触发记分板的 used 准测，即使玩家没有骑在猪或炽足兽上。因此通过创建记分板我们可以非常轻松地侦测玩家有没有手持这两个物品按下了右键。

通过在钓竿上植入自定义 NBT 或组件，可以实现在一种钓竿上制作出多个可右键的物品。

首先来看没有自定义 NBT 的版本：

首先创建记分板，准则为使用胡萝卜钓竿：

```
scoreboard objectives add carrot_stick used:carrot_on_a_stick
```

接下来就可以让所有该记分板至少有一分的玩家循环运行函数 right_click：

```
execute as @a[scores={carrot_stick=1..}] run function right_click
```

函数 right_click：

```
scoreboard players set @s carrot_stick 0
#首先重置玩家的分数

say 我刚刚拿着胡萝卜钓竿右键乐
#随后运行任何触发的内容
```

使用诡异菌钓竿同理，只需要替换记分板准则即可。

想要用胡萝卜钓竿制作多个右键触发，并区分彼此，需要在其内添加自定义 NBT。比如下面三条命令分别给了带着 3 个不同 NBT 的胡萝卜钓竿（1.20.4 之前）：

```
give @s carrot_on_a_stick{trigger:{id:"苹果"}}

give @s carrot_on_a_stick{trigger:{id:"香蕉"}}
```

```
give @s carrot_on_a_stick{trigger:{id:"葡萄"}}
```

若在 1.20.5+，需要使用物品组件：

```
give @s carrot_on_a_stick[custom_data={trigger:{id:"苹果"}}]
give @s carrot_on_a_stick[custom_data={trigger:{id:"香蕉"}}]
give @s carrot_on_a_stick[custom_data={trigger:{id:"葡萄"}}]
```

现在，替换 right_click 函数：

1.20.4 之前：

```
data modify storage trigger:right temp set value ""
#重置缓存用的存储
execute if data entity @s SelectedItem.tag{trigger:{}} run function right_click/mainhand
#侦测玩家主手是不是拿着触发物品，因为主手的优先级比副手高。如果是，执行函数：
```

```
    scoreboard players set @s carrot_stick 0
    #首先重置分数
    data modify storage trigger:right temp set from entity @s SelectedItem.tag.trigger.id
    #将其触发器 id 存储到缓存用的存储中
```

```
execute if score @s carrot_stick matches 1.. run data modify storage trigger:right temp set from entity @s Inventory[{Slot:-106b}].tag.trigger.id
#此时若分数仍为 1，则上面的函数没有触发，那么物品在玩家副手，将副手（-106 槽位）的触发器 ID 存进存储缓存

execute if data storage trigger:right {temp:"苹果"} run say 触发了苹果
execute if data storage trigger:right {temp:"香蕉"} run say 触发了香蕉
execute if data storage trigger:right {temp:"葡萄"} run say 触发了葡萄
#根据触发器 id 来判断触发了什么
```

1.20.5+：

```
scoreboard players set @s carrot_stick 0
```

```
#首先重置分数
data modify storage trigger:right temp set value ""
#重置缓存用的存储
execute if data entity @s Inventory[{Slot:-106b}].components."minecraft:custom_data"{trigger:{}} run data modify storage trigger:right temp set from entity @s Inventory[{Slot:-106b}].components."minecraft:custom_data"{trigger:{}}.trigger.id
#如果玩家副手物品有自定义数据，就将其触发器 id 存储到缓存用的存储中
execute if data entity @s SelectedItem.components."minecraft:custom_data"{trigger:{}} run data modify storage trigger:right temp set from entity @s SelectedItem.components."minecraft:custom_data"{trigger:{}}.trigger.id
#如果玩家主手物品有自定义数据，就将其触发器 id 存储到缓存用的存储中
#这里的主手物品优先级高于副手物品
execute if data storage trigger:right {temp:"苹果"} run return run say 苹果
execute if data storage trigger:right {temp:"香蕉"} run return run say 香蕉
execute if data storage trigger:right {temp:"葡萄"} run return run say 葡萄
#根据触发器 id 来判断触发了什么
```

如何安装资源包？

在 Minecraft 中，装载资源包共有三种方式。
两种为手动装载，余下一种为地图内置资源包，自动加载。
注①: 你装载的资源包内必须存在 pack.mcmeta 文件，否则游戏将无法识别。（可以是文件夹/压缩包）
注②: 以压缩包为格式的资源包，后缀名称必须为.zip。
注③: 在游戏加载资源包时，会按照列表上的顺序依次加载它们的资源。上层资源包可以替代、合并或移除下层资源包的资源，因此资源包的位置顺序会影响游戏对某些资源的读取和使用。

手动装载 1 - 拖放法：

1. 在你已启动的游戏中，点击**选项...**
2. 点击**资源包...**

在打开**资源包...**选项卡后你将看到一个窗口。接下来你只需要将你要装载的**资源包拖放至此窗口**。
拖放完成后，Minecraft 将弹出一个提示框，提示**你是否要将这些包添加进 Minecraft 中？**选择 **是"**即可。
然后你就可以在**资源包窗口的可用栏**处看到你刚刚拖放进去的资源包。

最后你只需要将鼠标放置在你**刚刚托放进来的资源包**上，点击**资源包图标**上的一个**三角形**，它就会去到**已选栏**。点击**完成**，**等待加载完毕**即可。

手动装载 2- 目录安装法：

若你开启了**版本隔离：**

1. 在**启动器目录**找到**.minecraft** 文件夹。
2. 前往**.minecraft/version/VersionName/resourcepack。**
3. 将**资源包文件**放置在 **resourcepack 文件夹**中**。**
4. **打开游戏，装载资源包。**

注④: **VersionName** 为你的**游戏版本名称。**请确保启动版本与安装资源包的版本为同一个版本。

若你未开启**版本隔离：**

1. 在**启动器目录**找到**.minecraft** 文件夹。
2. 打开**.minecraft** 目录下的 **resourcepack** 文件夹。
3. 将**资源包文件**放置在 **resourcepack 文件夹**中**。**
4. **打开游戏，装载资源包。**

自动装载 1- 内置资源包（仅适用于地图作者）：

在 Minecraft 中，你可以将**资源包内置**在**地图**里。当玩家们打开这张地图时，就会自动加载你内置的资源包。但这个方法仅支持主机，也就是打开这张地图的玩家。不适用从局域网联机进来的其他玩家，也不适用于服务器（服务器需要在 **server.properties** 中规定资源包 ）。

若你开启了**版本隔离：**

1. 在**启动器目录**找到**.minecraft** 文件夹。
2. 往**.minecraft/version/VersionName/saves/MapName。**
3. 将**资源包重命名为 resource.zip**，将**资源包**放置在 **MapName** 的根目录中。

注⑤: **MapName** 为你的**地图文件夹。**

例：
我的地图叫做 **Datura。**
那么就在**.minecraft/version/VersionName/saves/Datura** 这个目录中，放入 **resource.zip**。

若你未开启**版本隔离：**

1. 在**启动器目录**找到**.minecraft** 文件夹。
2. 往**.minecraft/saves/MapName。**
3. 将**资源包重命名为 resource.zip**，将**资源包**放置在 **MapName** 的根目录中。

详见：

- Wiki：资源包
- Wiki：教程/加载资源包
- Wiki：教程/制作资源包