

HDFS脚本语法修正



为什么mkdir创建时递归用的参数-p不是-r?



很好的问题！你注意到了`mkdir`和`rm`在递归操作时使用了不同的参数，这其实是因为它们的操作含义和目标是不同的。

`mkdir` 命令和 `-p` 参数：

- `mkdir` 用于创建目录。
- **-p 参数**：表示**递归创建目录**。当你用`-p`参数时，`mkdir`会创建指定的目录路径中所有不存在的父目录。如果某些父目录已经存在，`mkdir -p`不会报错，它会忽略这些已经存在的目录，只会创建那些不存在的目录。

举例：假设你想在`/home/user`下创建一个多层嵌套的目录`/home/user/projects/java/`，但只有`/home/user`目录存在，其他的`/projects`和`/java`目录都不存在。

- 如果你不使用`-p`参数，`mkdir`会报错，因为它尝试创建一个路径中的父目录，但是这些父目录尚不存在。

```
bash
```

```
mkdir /home/user/projects/java  
# 会报错: No such file or directory
```

- 如果你使用`-p`参数：

```
bash
```

```
mkdir -p /home/user/projects/java
```

它会递归地创建`projects`和`java`目录，前提是`/home/user`已经存在。执行这个命令后，所有的目录都会被创建，而不会报错。

`rm` 命令和 `-r` 参数：

- `rm` 用于删除文件或目录。
- **-r 参数**：表示**递归删除**。它的作用是删除一个目录及其下的所有文件和子目录。当你删除一个目录时，`rm -r`会递归地删除该目录中的所有内容，包括所有的文件和子目录。
 - 如果没有`-r`参数，`rm`只能删除文件，不能删除目录。
 - 使用`-r`时，`rm`会递归删除目录中的所有内容，并且也会删除目录本身。

举例： 假设你有如下目录结构：

```
arduino

/home/user
├── projects
│   └── java
│       └── file1.txt
```

如果你执行：

```
bash

rm -r /home/user/projects/java
```

它会删除 `/home/user/projects/java` 目录以及其中的 `file1.txt` 文件。

为什么 `mkdir` 使用 `-p` 而不是 `-r`？

- **递归创建目录：** `mkdir -p` 的目的是在你给定的路径中创建所有不存在的父目录。如果路径中某些父目录已存在，`mkdir -p` 不会报错，而是跳过它们。这是因为你可能想要递归地创建一条完整的目录路径。
 - 例如，你可能希望创建 `~/data/2024/December/`，但如果中间的某些目录已经存在，你并不希望因此而失败，`mkdir -p` 就能处理这个问题。
- **`-r` 和删除操作的关系：** `rm` 使用 `-r` 来递归地删除目录及其内容。这个操作和创建目录时的操作是相反的：你删除目录时需要递归删除其中的所有内容，而创建目录时，你不需要递归地删除内容，只需要递归地创建缺失的父目录。

主要区别：

- `mkdir -p`：创建目录及所有缺失的父目录。如果目录已经存在，不会报错。
- `rm -r`：删除目录及其所有内容（包括子目录和文件）。

总结：

- `mkdir -p` 的 `-p` 参数表示递归创建目录，在路径中创建不存在的父目录。
- `rm -r` 的 `-r` 参数表示递归删除目录及其内容。

这两者虽然都涉及“递归”操作，但它们的目的不同：`mkdir` 是创建目录，`rm` 是删除目录，因此它们使用不同的参数（`-p` 用于创建，`-r` 用于删除）。