golang的默认编码——utf-8

1. 0???????
2. 110????? 10??????
3. 1110???? 10?????? 10??????

两阶段提交/什么时候使用redo log什么时候使用undo log

- 假设有一个事务正在执行, 执行过程中已经写入了 redo log, 而提交完后 binlog写入时发生异常, 那么在 binlog 中可能就没有对应的更新记录, 之后从库使用 binlog 恢复时, 导致少一次更新操作. 而主库用 redo log 进行恢复, 操作则正常. 最终导致这两个库的数据不一致.
- 于是 InnoDB存储引擎 使用两阶段提交方案 : 将 redo log 的写入拆成了两个步骤 prepare 和 commit

1. 执行事务时写入redo log (这时处于prepare)
2. 提交事务之前, 先写入 binlog
3. 最后提交事务, 并将 redo log 进行 commit
4. 若使用 redo log 恢复数据时, 发现处于 prepare 阶段, 且没有 binlog, 则会回滚该事务. 若 redo log commit 时异常, 但是存在对应 binlog, MySQL还是认为这一组操作是有效的, 并不会进行回滚.

IP协议的路由和寻址分别有作用

寻址：给设备分配唯一的 IP 地址
路由：决定数据从源IP到目标IP的最优路径

golang中Mutex自旋锁？可重入？

1. 部分自旋，抢不到就挂起
2. 不支持可重入，重复加锁会死锁
   可重入的含义就是 `mu.Lock() mu.Lock()` 这样没有问题

golang中引用数据类型

只有map、channel、slice、函数、interface、指针，其他的类型都是基本数据类型

new和&区别

new(T) == 创建一个零值对象并返回指针；
&T{} == 创建一个初始化对象并返回指针；

实际开发中，大多数情况下推荐使用 &T{}，因为它更灵活，语义更清晰。

nil的slice也能用append