100 个 go 语言错误及避坑指南

代码和项目组织

#1 变量隐藏	意想不到的变量被隐藏
#2 不必要的嵌套代码	简单可读，尽早返回
#3 滥用 init 函数	谨慎使用
#4 过度使用 getter 和 setter	如果可能，直接访问字段
#5 接口污染	接口越大，抽象越弱
#6 生产者的接口	一般放在消费端维护
#7 返回接口	一般返回具体的实现
#8 any 意味着 nothing	避免过度泛化
#9 乱用泛型	通用数据结构和算法
#10 乱用嵌入类型	方法提升是一个双刃剑
#11 不使用函数选项模式	仁者见仁智者见智
#12 项目无组织	golang-standards 并非官方标准
#13 创建通用名的包	utils、commons 等不推荐
#14 变量和包名冲突	尽量避免
#15 缺少代码文档	每个导出对象都应该有文档
#16 不使用 linter	好工具为你减少错误

数据类型

#17 使用 8 进制字面量	很少有人直接使用
#18 忽视整数溢出	Go 对于溢出是静默的，要重视
#19 不了解浮点数	$1.0001 * 1.0001 \neq 1.00020001$
#20 不了解切片长度和容量	基本类型的实现原理要了解
#21 低效的切片初始化	初始化预期容量
#22 对 nil 和空切片迷惑	空是空，色是色
#23 不正确检查切片为空	$len(s) != 0$
#24 不正确复制切片	$copy(dst, src)$ 顺序、长度要记牢
#25 忽视 append 的副作用	理解切片数据的共享和扩容
#26 切片内存泄露	尤其是取子串、删除元素时
#27 低效的初始化 map	尤其是取子串、删除元素时
#28 map 内存泄露	删除不 shrink,NaN,clear 函数
#29 比较值时犯错	$reflect.DeepEqual$ 或第三方库

控制结构

#30 忽视遍历元素被复制	$for, a := range accounts$
#31 忽视遍历时参数咋求值	$for, v := rangeexp$, exp 求值
#32 忽视遍历时指针元素的影响	太普遍了，Go 1.21 尝试改变
#33 遍历 map 时想当然	顺序不保证，插入不可预知
#34 忽视 break 的工作机制	break 最里面的 for、switch、select
#35 在循环中使用 defer	别这样，一堆 defer 等待执行
#36 不理解 rune	rune 是 unicode 代码位

字符串

#37 不正确的字符串遍历	遍历的元素是 rune
#38 乱用 trim	你了解 TrimLeft 和 TrimRight 吗
#39 非优化的字符串连接	字符串连接和 []byte 互换等
#40 无用的字符串转换	尽量避免不必要的转换
#41 字符串内存泄露	子串占用母串

函数与方法

#42 不知道选择接收器类型	值接收器和指针接收器的区别
#43 从不使用命名的返回值	有时可以提高便利性
#44 使用命名返回值的副作用	因为初始化了，斟酌使用吧
#45 返回一个 nil 接收器	空不是空
#46 使用文件名作为函数输入	增加了实现和代码测试复杂性
#47 忽视 defer 和接收器的评估	参数和接收器的 evaluation

错误管理

#48 panic	谨慎使用 panic, 适时使用 recover
#49 搞不清何时使用 wrap	使用%w
#50 不准确的错误类型检查	使用 $errors.As$ 和 $errors.Is$
#51 错误的检查错误值	理解返回的错误类型
#52 两次处理同一个错误	每个错误只应该被处理一次
#53 忽略错误	大部分情况下都不应该被忽略
#54 忽略 defer 语句返回的错误	有可能导致资源未被释放

并发：基础

#55 混淆并发和并行	并发关于结构，并行关于执行
#56 认为并发总是更快	未必啊，还要看 overhead
#57 不知道选择并发锁还是通道	互补手段
#58 不理解竞争问题	数据竞争和竞争条件
#59 不知道工作负载类型	CPU 密集型和 I/O 密集型
#60 误解 Go context	context 不止上下文

并发：实践

#61 传播不恰当的上下文	谨慎传播
#62 不知道咋停止 goroutine	设计时要考虑，需要时要会停止
#63 粗心设计 goroutine 和循环变量	常见问题，Go 1.21 有改变
#64 使用 select/channel 保证确定性	select 是伪随机的
#65 没有使用通知类型 channel	常用 $chanstruct$
#66 没有使用 nil channel	更多的情况下我们是避免 nil channel
#67 对 channel 缓存区大小迷惑	理解缓存区有无、多大就好
#68 忘记字符串格式化的副作用	$String()$ 可能不是并发安全的，甚至死锁
#69 使用 append 导致数据竞争	切片 append 不是并发安全的
#70 错误使用互斥锁	小心锁的边界
#71 错误使用 WaitGroup	容易导致 panic
#72 忘记还有 sync.Cond	其实忘记了也好。通知和广播
#73 没有使用 errgroup	使用扩展包事半功倍
#74 复制同步原语	go vet 工具能够检查出来

标准库

#75 提供错误的持续时间	使用 $time.Second$ 而不是 1e9
#76 time.After 内存泄露	使用 $time.After$ 要谨慎
#77 常见的 JSON 处理错误	一些常见 JSOn 解析错误
#78 常见的 SQL 错误	一些常见的数据库访问错误
#79 没有关闭临时资源	比如 $resp.Body$
#80 处理 http 请求时没有返回	$http.Error$ 之后立即返回
#81 使用默认的 http 客户端和服务端	容易被意外更改

测试

#82 未区分测试种类	适应 build tag、环境变量、短模式区分
#83 未打开 -race 开关	强烈建议开启
#84 未使用测试执行模式	$-parallel\ -shuffle$
#85 未使用表格驱动型测试	推荐使用
#86 在单元测试中休眠	默认超时 30 秒
#87 没有有效使用 time	mock 或者重构时间相关代码
#88 未使用测试工具包	$httptest\ iotest$ 等
#89 实现不准确的基准测试	一些基础测试的误区
#90 未探索所有的 Go 测试特性	代码覆盖率等等

优化

#91 不了解 CPU 缓存	写 CPU 缓存友好的代码
#92 编写导致伪共享的并发代码	通过填充或通信来防止伪共享
#93 不考虑指令级并行性	充分利用指令流水线并行
#94 不了解数据对齐	对齐保证更好的空间局部性
#95 不了解栈与堆	尽量避免在堆上分配内存
#96 不了解如何减少分配	初始容量和池化技术
#97 没有依赖内联	内联是提升性能的有效手段
#98 没有使用 Go 诊断工具	$profile\ trace$
#99 不了解 GC 的工作原理	了解一些 GC 细节
#100 不了解容器对 Go 程序的影响	$automaxprocs$

2023 整理 by 晁岳攀 (@ 鸟窝) https://colobu.com。新书推荐: ↓