Задачи Шаблоны

Как создать задачу?

Поиск по названию, содержанию и разделу

Грейд

По дате добавлен...

Раздел

- Flutter
- Тимлид
- Python
- 1C
- Аналитика данных
- C++
- Продуктовый дизайн
- Algorithms
- GO
  - Структуры данных
  - Скрининг
  - Go (теория)
  - Go (практика)
  - Базы данных
  - Архитектура
  - ОС, сети и эксплуатация
- Architecture (System Design)
- QA
- .NET
- Android
- iOS
- SQL
- BI
- Java
- Frontend
- Product
- DS
- Мотивация
- Базы данных(не для Data Engineer)
- Аналитик (бизнес)
- Scala
- Информационная безопасность
- Эксплуатация/DevOps
- Техническая документация
- Проектное управление
- Системная аналитика

Найдено задач: 63

Что такое страница памяти?

GO › ОС, сети и эксплуатация

19 - 20 грейд · 5 мин · 1 раз

Свернуть

Что такое страница памяти?

Ответ

19 грейд

Это единица структурирования виртуальной памяти, позволяет мапить виртуальную память на физическую либо на файл. Так же страничная организация позволяет защищать некоторые куски памяти от записи.

20 грейд

Дополнительные вопросы:

1. Если страницы мапятся на физическую память, то каждое обращение к памяти это поиск в некоторой мапе. Как это может работать быстро?
2. Что такое hugepages?

Ответы:

1. У процессора есть специальный кеш для этого мапинга, поэтому если программа работает с близкими адресами памяти, то это работает очень быстро
2. Это механизм через который можно менять размер страницы со стандартных 4Кб до бОльших значений, например 256Мб. Это нужно для высокопроизодительных программ типа баз данных как раз для того, чтобы уменьшить кол-во обращений к мапингу.

Что такое сисколл?

GO › ОС, сети и эксплуатация

18 - 19 грейд · 5 мин · 1 раз

Свернуть

Что такое сисколл?

Ответ

18 грейд

Вызов какой-то функции в операционной системе, например чтение файла.

19 грейд

Дополнительные вопросы:

1. Что происходит во время вызова сискола?
2. Что делает сисколл fork?
3. Что делает сисколл epoll?

Ответы:

1. Происходит переключение из User Space в Kernel Space, после чего система выполняет необходимые действия
2. Запускает копию процесса
3. Позволяет асинхоронно получать ивенты о том, что в из дескриптора можно вычитать кусок информации

Примитивы синхронизации

GO › ОС, сети и эксплуатация

L2 Application Support

18 - 20 грейд 5 мин 6 раз Свернуть

Какие знаешь примитивы синхронизации в ОС?

Ответ

18 грейд
Мьютексы

19 грейд
Мьютексы, семафоры, файлы

20 грейд
Может рассказать за spinlock, всякие гибридные схемы типа futex

User Space vs Kernel Space?

GO > ОС, сети и эксплуатация

18 - 19 грейд 5 мин 1 раз Свернуть

Что такое user space и kernel space? Зачем нужно такое разделение?

Ответ

18 грейд
Прилкадные программы работают в User Space, а система работает в Kernel Space

19 грейд
User space — окружение и режим процессора в котором ему доступна только виртуальная память текущего процесса, отсутствует часть низкоуровненых комманд процессора

Kernel space — соответственно окружение в котором работает система и ей доступен весь арсенал процессора в том числе доступ к устройствам, менеджмент памяти и тд.

Разделение нужно для безопасности так чтобы никакое прикладное ПО не могло сломать систему. Переключение между этими режимами происходит при вызове приложением сисколов (можно считать API операционной системы).

Что такое файловый дескриптор?

GO > ОС, сети и эксплуатация

18 - 20 грейд 5 мин 1 раз Свернуть

Что такое файловый дескриптор?

Ответ

18 грейд
При открытии файла тебе выдается некий идентификатор, который можно использовать для того чтобы совершать чтение/запись файла и другие операции

19 грейд
Это универсальный идентификатор, который выдает операционная система для работы со множеством сущностей: реальные файлы диске, виртуальные файлы в том числе файлы, замапленные на память, различные устройства ввода/вывода, пайпы, сокеты и тд.

Дополнительные вопросы:
1. Что такое stdin/stdout/stderr, какие дескрипторы назначены для этих сущностей?

Ответы:
1. Это стандартные интерфейсы для приложения для ввода информации, вывода и вывода ошибок. Обычно при запуске приложений из консоли stdout/stderr перенаправляется в консоль. Имеют идентификаторы 0, 1, 2

20 грейд
Дополнительные вопросы:

1. du/dh покызвает, что место в файловой системе еще есть, а записать ничего не возможно, т.к. “нет места”. Почему и как исправить?

Ответ:
1. При удалении файла реальное освобождение пространства происходит только при закрытии дескрипторов всеми приложениями. Поэтому нужно найти программу которая держит дескрипторы и заставить ее отпустить эти дескрипторы.

Утечка памяти

GO › ОС, сети и эксплуатация

18 - 20 грейд 5 мин 5 раз Свернуть

На голом сервере запущен какой-то сервис и в нем есть утечка памяти. Что произойдет, когда память закончится? Речь не про кубер и ограничения контейнеров, а про голый сервер.

Ответ

18 грейд
Система убьет процесс

19 грейд
Кандидат должен рассказать, что есть два варианта: включен своп или нет.

Если своп включен, то система может жить в таком состоянии довольно долго тк для памяти будет использоваться диск.

Если своп отключен, то придет OOM Killer и начнет убивать процессы. Не факт, что умрет именно наш сервис, но рано или поздно его тоже убьет.

20 грейд
Дополнительный вопрос: "На какой тип памяти система будет опираться при решении запустить OOM Killer?"

Важно, что система оценивает не вирутальную память, а только реально используемую память, которая мапится в физическую.

Дополнительный вопрос: "Зачем нужна виртуальная память?"

Виртуальная память это концепт современных операционных систем позволяющая через механизм страниц транслировать части виртуальной памяти в другие типы памяти: диск, физическая память, память устройств и тд. Так же виртуальная память обеспечивает разделение памяти между процессами и возможность защищать куски памяти от операций записи или от выполнения кода.

Процесс vs системный поток

GO › ОС, сети и эксплуатация

18 - 20 грейд 5 мин 10 раз Свернуть

Чем отличается процесс от системного потока?

Ответ

18 грейд
Системный поток запускается внутри процесса

19 грейд
Процессы не шарят память между собой, а потоки внутри процесса — шарят.

Дополнительный вопрос: "А как процессы могут обмениваться информацией?"
Ответ: сокеты, пайпы, сигналы

20 грейд
И то и другое это Task в терминах ядра Linux. Отличие в том, что процессы по-умолча не шарят память, а потоки в рамках процесса — шарят. При этом процессы тоже могу шарить, делается это через специальные механизмы типа shm_open или mmap.

Что такое GOMAXPROCS?

GO › Go (теория)

18 - 19 грейд 5 мин 11 раз Свернуть

Что такое GOMAXPROCS? Зачем это нужно?

Ответ

18 грейд

Ограничивает кол-во потоков, которые использует Go для запуска кода

19 грейд

Дополнительный вопрос: "Может ли приложение с GOMAXPROCS=4 потреблять больше 4ех ядер CPU"

Да может. Это происходит из-за того, что Go Runtime запускает дополнительные потоки на блокирующие сисколы. В эти микро-моменты кол-во используемых ядер может быть больше GOMAXPROCS

Ревью кода кеша

GO › Go (теория)

17 - 20 грейд 5 мин 18 раз Свернуть

Разработчик дал на ревью код своего нового кэша, нам необходимо провести код-ревью

Кэш будет использоваться под высокой нагрузкой в проде
Частота записи/чтения 20%/80% соответственно

Код:

```
package main

import (
    "fmt"
    "sync"
)

func main() {
    fmt.Println(GetOrCreate("hello", "world"))
    fmt.Println(Get("hello"))
}


var cache = make(map[string]string)

// GetOrCreate проверяет существование ключа key
// Если такого нет, то создает новое значение
func GetOrCreate(key, value string) string {
    var m sync.Mutex

    m.Lock()
    value = cache[key]
    m.Unlock()

    if value != "" {
        return value
    }

    m.Lock()
    cache[key] = value
    m.Unlock()
    return value
}

func Get(key string) string {
    var m sync.Mutex
```

```
    m.Lock()
    v := cache[key]
    m.Unlock()
    return v
}
```

Ответ

17 грейд

Заметил что мьютекс — локальная переменная
Дал рекомендацию оформить это в структуру

18 грейд

Посоветовал использовать defer для мьютексов
Предложил переписать GetOrCreate чтобы лочить мьютекс только один раз

19 грейд

Предложил использовать RWMutex
Заметил, что два отдельных лока в GetOrCreate могут привести к конкурентной записи

20 грейд

Предложил использовать sync.Map, рассказал чем подход с RWMutex (а за одно и посомневался почему RWMutex может оказаться медленнее)
Предложил сделать разбить кеш на несколько шардов с отдельным мьютексом

Как устроена хеш-таблица

GO › Структуры данных

17 - 20 грейд · 5 мин · 22 раза

Свернуть

Как устроена структура данных "хеш-таблица"? Интересует не детали реализации в каком-либо языке, а общая идея.

Допустим есть операция set. На вход подается ключ — строка, а значение — число. Что происходит внутри функции?

Ответ

17 грейд

Кандидат рассказал только про свойства хеш-таблиц и что операции обычно выполняются за сложность O(1)

18 грейд

Хеш-таблицы имеют под капотом массив определенной длины.

От ключа считается некий хеш с помощью хеш-функции, которая обеспечивает равномерное распределение. Хеш это какое-то большое число, которое потом нужно привести к длине массива. Приведение осуществляется остатком от деления.

Полученное число используется как индекс в массиве куда мы хотим положить пару "ключ-значение"

При таком подходе будут случаться коллизии, кандидат должен рассказать как они решаются: либо хранением списка внутри массива, либо поиском ближайшей свободной ячейки в массиве.

19 грейд

Все тоже самое что 18 грейд, но кандидат понимает как делать рехешинг, может примерно рассказать как написать хеш-функцию

Может привести примеры хеш-функций и рассказать какие из них хорошие и плохие (crc32, md5, aes, sha1 и тд)

20 грейд

Все тоже самое что 19 грейд, но кандидат может рассказать несколько способов решения коллизий, может порассуждать в какой момент лучше делать рехешинг, как его организовать рехешинг плавно.

Может дать детали реализации хеш-таблицы в конкретном языке программирования.

Может рассказать как написать хорошую хеш-функцию.

Самая сложная инженерная задача, которую приходилось решать

GO > Скрининг

Скрининг 5 мин 32 раза Свернуть

Расскажи самую сложную или интересную инженерную задачу, которую приходилось решать?

Дополнительные вопросы:
1. В чем была сложность?
2. Как удалось справиться с задачей?
3. Как бы ты решил задачу сейчас?

Ответ

Четкого ответа нет

select по каналам

GO

17 - 18 грейд 2 мин 35 раз Свернуть

Что выведет этот код

```
package main

func main(){
    ch := make(chan int, 1)

    for i := 0; i < 10; i++ {
        select {
            case x := <-ch:
                print(x)
            case ch <- i:
        }
    }
}
```

Ответ

17 грейд

0 2 4 6 8

18 грейд

Расскажет про панику из-за дедлока если сделать канал не буферизированным

LRU cache (Только теория)

GO > Go (теория)

17 - 19 грейд 10 мин 160 раз Свернуть

Рассказать, как устроен LRU cache

```
Кеш имеет фиксированный максимальный размер.
Если после вставки нового элемента превышается максимально допустимый
размер кеша, из него нужно удалить самый старый элемент.

Пример:
cache = new LRU(capacity=2)

cache.put(A, 1); // cache is {A=1}
```

```
cache.put(B, 2); // cache is {A=1, B=2}
cache.get(A);    // return 1
cache.put(C, 3); // {A=1, C=3}, element B=2 was evicted
cache.get(B);    // returns -1 (not found)
cache.put(D, 4); // {D=4, C=3}
cache.get(A);    // return -1 (not found)
cache.get(C);    // return 3
cache.get(D);    // return 4
```

Ответ

17 грейд

Если кандидат затрудняется с ответом, то спросить а какие вообще структуры данных он знает и сложность операций в них. Как минимум (массив, словарь/мапа, деревья)

18 грейд

Используем хештаблицу как основную структуру для кеша. Ключи в ней будут теми же, что и в LRU кеше, но со значениями сложнее - их мы оборачиваем в структуру Node - узел двусвязного списка. Node будет хранить ключ, значение, и ссылки на prev/next узлы.

Нам нужно реализовать работу с последовательностью операций, чтобы определять самый старый элемент. Здесь и нужен двусвязный список - элементы в нём располагаются в хронологическом порядке операций. Самый старый элемент будет в голове списка, новые и обновленные элементы будут перемещаться в хвост списка.

Если мы читаем по ключу не в первый раз, то нам может понадобиться переместить его из середины списка в хвост. Находим за O(1) нужный узел по ключу, за O(1) удаляем его из середины списка (обновляем ссылки соседних узлов) и перемещаем в конец.

19 грейд

-

Задача на РСУБД

GO › Базы данных

17 - 19 грейд · 20 мин · + 235 раз

Свернуть

Есть 3 сущности - пользователь, чат, сообщение

- У пользователя есть имя и дата регистрации
- У чата есть название и дата создания
- У сообщения есть текст, автор и дата создания
- Пользователь может состоять в нескольких чатах одновременно
- Сообщение обязательно принадлежит чату, сообщение не может принадлежать более чем 1 чату одновременно
- Нужно описать предметную область в виде таблиц

Ответ

17 грейд

```
create table users (
  id int primary key,
  name text not null,
  created_at timestamp not null
);

create table chats (
  id int primary key,
  name text not null,
  created_at timestamp not null
);

create table messages (
  id int primary key,
  content text not null,
  author_id int not null, -- ссылка на автора, связь N-1
  chat_id int not null, -- принадлежность к чату, связь N-1
  created_at timestamp not null
);
```

```
create table user_chats ( -- таблица для связи N-M
  user_id int,
  chat_id int,
  primary key (user_id, chat_id)
  -- primary key (chat_id, user_id) Тоже допускается
);
```

Дополнительные вопросы

Выбрать все чаты пользователя Вася в формате (chat_id,chat_name)

SQL

```
select uc.chat_id, c.name chat_name
from users u
join user_chats uc on uc.user_id = u.id
join chats c on c.id = uc.chat_id
where u.name = 'Вася'
```

- Какие еще виды соединений кроме join знаешь?
  Достаточно ответить про left join, right join
- Что будет, если вместо join применить left join в этом запросе?
  Ничего не изменится

18 грейд

Включает ответы на 17 грейд, задаем больше дополнительных вопросы

Дополнительные вопросы

- Что такое индекс? Сколько и какие индексы будут созданы по этой схеме?
  Индекс - это вспомогательная структура данных для ускорения доступа к таблице, неявно управляется ядром БД
  4 - для каждой таблицы для полей с указанием primary key будет создан уникальный B-TREE индекс
- По мере роста размеров таблиц этот запрос начинает все медленнее работать, можешь понять почему и исправить?
  Добавить индекс на user(name), в более сложных случаях профилировать запрос с помощью EXPLAIN [ANALYZE]
- **[Не спрашивать, если кандидат не знает про explain]** Что показывает эта команда?
  План запроса с оценкой сложности по узлам
- **[Не спрашивать, если кандидат не знает про explain]** Какой узел плана однозначно показывает, что не хватает индекса?
  SeqScan
- Как заполнять поле *id* при вставках в таблицы users, chats, messages
  Перейти на автоинкрементное поле

```
create table users (
    id serial primary key,
);
```

Или перейти на генерацию uuid на стороне приложения

19 грейд

Включает ответы на 18 грейд, задаем больше дополнительных вопросы

Дополнительные вопросы

- Чаты становятся популярными, число пользователей растет и нам скоро не хватит свободного места на диске для хранения всей переписки. Что нам делать?
  Нужно шардировать. Шардируем наиболее сильно растущие данные - в данном примере это таблица *messages*, потому что хранит генерируемый пользователем текст
- Какой первичный ключ лучше - числовой или uuid?

Задача на простой select с группировкой

GO › Скрининг

Скрининг · 5 мин · 328 раз

Свернуть

```
Дана таблица "orders":

+-----+----------+---------------------+-------------+
```

```
|  id |  user_id  |             created_at |  price_total |
+-----+-----------+------------------------+--------------+
|   1 |       111 |    2024-01-01 10:00:00 |        2 000 |
|   2 |       222 |    2024-01-02 10:00:00 |          100 |
|   3 |       111 |    2024-04-01 10:00:00 |       20 000 |
|   4 |       222 |    2024-05-01 10:00:00 |        5 000 |
|   5 |       333 |    2024-05-02 10:00:00 |       10 000 |
+-----+-----------+------------------------+--------------+
```

Напишите SQL-запрос, который вернёт количество заказов по каждому пользователю с price_total больше или равным 1000 в таком виде отсортированному по количеству заказов в обратном порядке:

```
+----------+-------------+
|  user_id | order_count |
+----------+-------------+
|      111 |           2 |
|      222 |           1 |
|      333 |           1 |
+----------+-------------+
```

Ответ

SELECT user_id, COUNT(*) AS order_count FROM orders WHERE price_total >= 1000 GROUP BY user_id ORDER BY order_count DESC

Если с группировкой не может, то предложить просто выбрать заказы с price_total >= 1000 (чтобы убедиться что вообще понимает что такое sql)

Работа с слайсами

GO › Скрининг

17 - 18 грейд · 5 мин · 697 раз

Свернуть

Что выведет данная программа. Почему так ?

```go
func main() {
    nums := []int{1, 2, 3}

    addNum(nums[0:2])
    fmt.Println(nums) // ?

    addNums(nums[0:2])
    fmt.Println(nums) // ?
}

func addNum(nums []int) {
    nums = append(nums, 4)
}

func addNums(nums []int) {
    nums = append(nums, 5, 6)
}
```

Ответ

17 грейд

1) 1 2 4
2) 1 2 4

Расскажет про правила работы append

18 грейд

Расскажет про синтаксис [0:2:2]

Modifying slice during iteration

GO › Скрининг

18 грейд 5 мин 264 раза Свернуть

Что выведет этот код, и почему?

```
package main

import "fmt"

func main() {
    lst := []string{"a", "b", "c", "d"}

    for k, v := range lst {
        if k == 0 {
            lst = []string{"aa", "bb", "cc", "dd"}
        }
        fmt.Println(v)
    }
}
```

Ответ

Это задача даже не на специфику работы оператора range, а просто на знание одного из основных принципов языка — *everything in Go is passed by value*. Массив является по сути *struct*'ом из трех полей — `ссылка на array`, `len` и `cap`, при вызове range происходит ровно то же самое, что и при вызове функции — эта структура копируется (а array на который ссылаемся — нет). Присваивая переменной новое значение мы никак не меняем исходный array, поэтому и выведется

```
a
b
c
d
```

А вот если менять значения в нижележащем array'е, то они будут меняться, и range их увидит

```
func main() {
    lst := []string{"a", "b", "c", "d"}

    for k, v := range lst {
        if k == 0 {
            lst[3] = "z"
        }
        fmt.Println(v)
    }
}
```

выведет

```
a
b
c
z
```

HTTP Protocol

GO › ОС, сети и эксплуатация

17 - 19 грейд 5 мин 2558 раз Свернуть

Что из себя представляет протокол HTTP? Из каких основных частей состоит HTTP запрос и HTTP ответ?

Ответ

17 грейд

HTTP - текстовый протокол функционирующий по принципу запрос-ответ. HTTP запрос состоит из стартовой стороки (включающей метод, идентификатор ресурса и версию протокола), хедеров и тела сообщения. HTTP ответ похож по структуре на HTTP запрос, но стартовая строка вместо метода и URI содержит код состояния.

18 грейд

Кандидат может рассказать про cookie, привести примеры хедеров, основные методы, знает коды ответов.

19 грейд

Кандидат может самостоятельно написать http запрос текстом.
Кандидат может ответить, почему статику рекомендуется размещать на отдельном домене?

Дополнительные вопросы

Q: *Какие есть основные методы HTTP?*
A: GET, HEAD, POST, PUT, DELETE

Q: *Примеры HTTP хедеров?*
A: content-length, content-type, cookie, authorization, user-agent

Q: *Назовите классы кодов состояний HTTP*
A: 1xx (informational), 2xx (success), 3xx (redirect), 4xx (client error), 5xx (server error)

Мониторинг

GO > ОС, сети и эксплуатация

17 - 19 грейд · 10 мин · 1792 раза · Свернуть

Какие существуют инструменты для мониторинга и отладки микросервисов?

Ответ

17 грейд

Назвает логи, графики, говорит, что можно посмотреть информацию о загрузке CPU память на сервере, но без подробностей, назвает инфраструктурые средства мониторинга munin / zabbix

18 грейд

Рассказывает про мониторинг, метрики, называет prometheus, grafana, может описать систему алертинга.

19 грейд

Знает подробности трейсинга и профилирования, может описать свой опыт решения практическиз хадач.

Дополнительные вопросы

Можно изучить логи микросервиса для поиска сообщений об ошибках. При помощи метрик можно получить более детальную информацию о состоянии сервиса, например нагрузке или времени ответа. При помощи профайлера можно исследовать используемые ресурсы, например cpu или память. При помощи трассировки можно изучать взаимосвязи в больших микросервисных системах и находить узкие места.

Q: *Пример инструментов для сбора и отображения метрик?*
A: prometheus, grafana

Q: *Какая тулза используется для профилирования в go?*
A: pprof

Q: *Зачем микросервису на go нужен хендлер /debug/vars?*
A: Для выдачи отладочной информации (например об использовании памяти или gc)

Q: *Как посмотреть логи микросервиса живущего в kubernetes?*
A: С помощью команды kubectl logs

CPU 146%

GO › ОС, сети и эксплуатация

18 - 20 грейд · 5 мин · 2027 раз · Свернуть

Команда `top` показывает, что какой-то процесс потребляет 146% CPU. Это реальная ситуация? Если да, то нужно ли мне что-то с этим делать?

Ответ

18 грейд

Кандидат предположил, что это может быть из-за того, что используется несколько ядер. Может не знать нужно с этим что-то делать или нет.

19 грейд

Да, абсолютно реальная. Это означает, что процесс потребляет почти 1.5 ядра. При таких вводных нельзя сказать нужно ли что-то с этим делать или нет. В общем случае — это абсолютно нормально.

20 грейд

Кандидат знает про CFS, базово может рассказать про то как система распределяет процессорное время между тасками

Дополнительные вопросы

Красный флаг — если кандидат говорит "ну это наверное какая-то перегрузка системы"

Как убить процесс linux'e?

GO › ОС, сети и эксплуатация

17 - 18 грейд · 5 мин · 2019 раз · Свернуть

Как убить процесс linux'e?

Ответ

17 грейд

Дан по крайней мере один из вариантов

```
kill <pid> / kill -9 <pid>
killall <proccess_name>
top # найти процесс, нажать k
htop # найти процесс, нажать k
```

18 грейд

Указаны способы нахождения pid процесса.

```
ps aux и аналоги + grep
top / htop
```

Кандидат рассказал о сигналах и чем плох -9

Дополнительные вопросы

Дополнительные вопросы:
Q: *Что делать, если я вызываю kill и процесс продолжает работает?*
A: Отправить SIGKILL / -9.
Q: *Что такое `-9` ?*
A: Сигнал, отправляемый процессу

TCP / UDP

GO › ОС, сети и эксплуатация

17 - 19 грейд · 5 мин · 3575 раз · Свернуть

Чем различаются протоколы TCP и UDP?

Ответ

17 грейд

TCP требует установки соединения а UDP нет, TCP надежный а UDP датаграммы могут теряться/дублироваться/перепутываться, TCP медленнее UDP, UDP поддерживает broadcast датаграммы.

18 грейд

Q: Зачем нужен UDP, если он такой ненадежный?
A: Зато он быстрее, за счет отсутствия подтверждений

Q: Зачем нужен TCP, если он такой медленный?
A: Зато он надежнее, за счет механизма подтверждений

Рассказал анекдот

> - Знаю анекдот про UDP, но не факт, что он до вас дойдет...
> - Знаю анекдот про TCP. Если он до вас не дойдет, я повторю его снова.

19 грейд

Q: *Опишите высокоуровневую схему взаимодействия между клиентом и сервером по протоколу TCP*
A: Сервер создает сокет и начинает слушать входящие соединения на определенном порте (bind, listen, accept), клиент создает сокет и подключается к серверу при помощи connect, далее происходит обмен сообщениями (send/recv), в конце сокеты закрываются.

Дополнительные вопросы

Q: *Где применяется udp??*
A: Передача потокового видео/аудио, торренты, онлайн игры, DNS

Вопрос о медленном сервисе

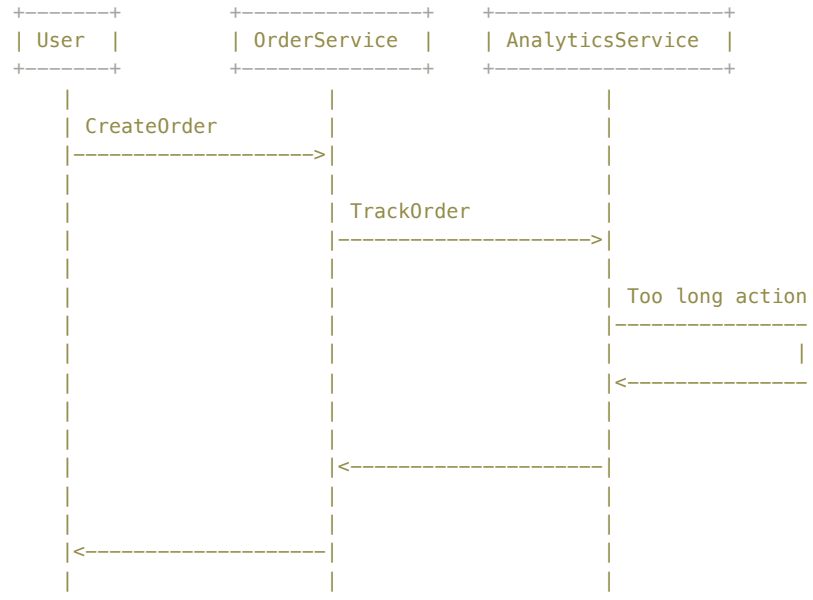
GO > Архитектура

18 грейд · 15 мин · 2378 раз

Свернуть

На примере создания заказа. Есть запрос на сервис и есть ответ, между этими двумя действиями мы сладываем в аналитику товары которые заказали (например для подсчета популярности товаров). Сервис аналитики переодически работает медленно или вовсе таймаутит, и мы не успеваем ответить, теряем заказы.

Что делать, что бы перестать терять заказы, и деньги соответсвенно?



Ответ

Придумать как распаралелить заказ и аналитику.

Дополнительные вопросы

// Если решение через БД Q: Как организовать воркеры, так, что бы они не брали в работу одни и те же данные? A: Придумать как не заселектить в несколько воркеров одинаковые данные для обработки, например мастер процесс который будет раздавать задачи.

Q: *Сервис аналитики не будет читать данные из нас или нашей очереди, что будем делать?* A: Придумать способ как самим пушить данные в аналитику из своей очереди

Q: *Отдел аналитики жалуется на высокий рпс от нас когда они под нагрузкой мы их сервис кладем. Как решить проблему?* A: Просто уменьшить рпс в конфигах плохая идея тогда будет копиться очередь. Нужно придумать экспоненциальное повышение или понижение нагрузки в зависимости от состояния аналитики

Q: *Проблема с аналитикой решена. Теперь проблема с местом занимаемым нашей очередью, инфра ограничила ресурсы(расширять дальше некуда), что делать? (На аналитику повлиять мы не можем, у них в планах переписать сервис и тд нужно свое решение.)* A: Надо придумать как мы будем просеивать данные в зависимости от нашей квоты на ресурсы и загруженности очередей. Например вытеснением или прореживанием данных для аналатики на входе нашего сервиса.

Rate limit

GO › Архитектура

18 - 19 грейд 5 мин 1213 раз Свернуть

Мой сервис используют внешний API для построения маршрутов, в рамках тарифа у нас есть ограничение по RPS. Нужно спроектировать ratelimiter.

Ответ

18 грейд

Есть два основных варианта решения этой задачи:

- С использованием внешнего хранилища. Например, мы можем использовать redis для этой задачи, в котором храним в счетчике кол-во запросов за нужное окно.
- Без использования внешнего хранилища. Каждый инстанс имеет ограничение x / n , где x - общее количество запросов, а n - количество инстансов.

Q: *Какие есть плюсы и минусы у одной/обоих схем?*
A: **внешнее хранилище**.
плюсы:

- простая реализация и конфигурация
- не имеет значения насколько сбалансированы запросы по инстансам сервиса минусы:
- увеличение времени запроса
- единая точка отказа

без внешнего хранилища.
плюсы:

- нет единой точки отказа
- нету дополнительных задержек

минусы:

- сложно конфигурировать
- балансировка запросов имеет большое значение

19 грейд

Q: *Как можно оптимизировать работу с redis'ом?*
A: Использовать lua и сделать процедуру, которая будет возвращать true/false.

Scaling

GO › Архитектура

18 грейд 10 мин 1440 раз Свернуть

Есть два сервиса A и B, сервис A отправляет запросы в сервис B по HTTP. Какие существуют варианты для масштабирования сервиса B? Можно использовать любые технологии.

Ответ

Здесь нету одного однозначно правильного ответа, т.к. все очень сильно зависит от конкретной ситуации, но основные направления такие:

- serverside balancing. Добавляем промежуточный балансировщик (nginx / haproxy).
- clientside balancing. Используем DNS / consul / etcd из которого A узнает где запущен B.

Дополнительные вопросы

Q: *Какие есть плюсы и минусы у одной/обоих схем?*

Q: *Сколько балансировщиков будет в первой схеме?*
A: Как минимум два

Q: *Если их больше, чем один, то как приложение узнает где они запущены? (возвращаемся к оригинальной задаче)*
A: Балансеры можно находить через DNS.

Q: *Как именно будет выглядеть discovery через DNS?*
A: Ожидается ответ про A-записи, SRV-записи

REST / RPC

GO › Архитектура

18 грейд · 5 мин · 1333 раза · Свернуть

В чем разница между REST и RPC подходами?

Ответ

REST строится вокруг ресурсов и операций (GET, PUT, DELETE и т.д.)
RPC - вызов методов сервиса

Дополнительные вопросы

Q: *Обязательно ли делать REST поверх HTTP?*
A: да

Q: *Обязательно ли делать RPC поверх HTTP?*
A: нет, можно делать поверх чего угодно.

Q: *Расскажи как RPC запрос можно переложить на HTTP-запрос?*
A: Передавать имя сервиса/метода в url'е/заголовке и т.д.

Q: *А как это сделать используя только TCP?*

Proxy / reverse proxy

GO › Архитектура

18 грейд · 5 мин · 372 раза · Свернуть

В чем разница между proxy / reverse-proxy?

Ответ

В случае proxy клиент знает, что запрос/соединение пойдет через proxy в точку назначения. Если используется reverse proxy, то клиент ничего не знает о его существовании и не знает кем конкретно будет обработан его запрос.

Задачка про датафикс

GO › Базы данных

18 - 19 грейд · 5 мин · 2154 раза · Свернуть

Есть таблица на 1 миллиард записей, которая активно используется в продашкне. Нуж сделать датафикс, который модифицирует 10 миллионов строк. Как бы ты подошел к решению задачи?

Ответ

18 грейд

Делал бы update батчами, с фильтром по индексируемым полям.

19 грейд

Из жизни — чаще всего надо поправить только ряды за сегодня, а на остальное забить, это сразу переводит задачу в класс более простых.
То есть есть какой-то соседний индекс коррелирующий с появлением испорченных записей, и по этому индексу можно радикально сократить кол-во перебираемых рядов.

Если же реально данные равномерно размазаны по всей таблице:

1) Если индекс, позволяющий найти нужны ряды есть — все проще, пишем код, который отдельно набирает ID рядов, где условие выполняется, потом накатываем изменения пачками с `where ID in(...)`.

2) Если индекса по нужному нам условию нет, и надо перебирать все поля

- выбираем ID начиная от которого баг исправлен
- далее пишем скрипт который читает кусками данные из **слейва** и сохраняет в файл ID рядов где надо менять данные
- накатываем пачками на мастер по 1000-10 000 рядов, беря ID из файла

Типы блокировок в PostgreSQL

GO › Базы данных

18 грейд · 10 мин · 972 раза · Свернуть

Какие типы локов (блокировок) есть в PostgreSQL? Кто берет эти локи?

Ответ

Лок на таблицу, лок на запись, пользовательские (рекомендательные) локи.
Далее важно, чтобы человек порассуждал на тему что это за локи, как они конфликтуют между собой.
Локи на таблицу могут брать стандартные команды SQL (Update, Delete, Select, Alter table...) Например:SELECT берет лок на таблицу типа ACCESS SHARE, который конфликтует, например, с самым строгим ACCESS EXCLUSIVE.
Локи на строчки в таблице берут стандартные команды SQL, которые меняют данные у каких-либо записей. Например, есть тип блокировки FOR UPDATE, который конфликтует сам с собой. Эти же типы локов можно указывать у оператора SELECT. Пример: Select * from t1 for update nowait (skip locked).
Пользовательские локи можно брать, например, когда нужен общий ресурс для нескольких приложений и нет возможности шарить его в другой инфре.

Аномалии при параллельном исполнении транзакций

GO › Базы данных

18 - 19 грейд · 5 мин · 869 раз · Свернуть

Какие аномалии могут быть при исполнении несколько транзакций параллельно? Расскажи кратко о них.

Ответ

18 грейд

Перечислил аномалии, без подробностей

19 грейд

Перечислил аномалии с краткими подробностями

Дополнительные вопросы

грязное чтение, неповторяемое чтение, фантомное чтение, аномалия сериализации.

Транзакции

GO › Базы данных

17 - 18 грейд · 5 мин · 3272 раза · Свернуть

Что такое транзакция? Какие уровни изоляции транзакций существуют в РСУБД?

Ответ

17 грейд

Транзакция - это группа выполняющихся команд, которые можно подтвердить/отменить целиком или с возвратом SavePoint.
Кандидат может назвать пример использования. Обычно это банковский перевод с одного счета на другой.

18 грейд

Кандидат может назвать уровни изоляции.
Уровни изоляции - Read uncommited (можно спросить есть ли он в PostgreSQL), Read committed, Repeatable read, Serializable.

Индексы

GO › Базы данных

Грейд неизвестен · 2193 раза · Свернуть

Построить оптимальный индекс для

```
SELECT * FROM employee WHERE sex = 'm' AND salary > 300000 AND age = 20
ORDER BY created_at
```

Ответ

TODO: @mkabischev

Constraint

GO › Базы данных

18 грейд · 5 мин · 1022 раза · Свернуть

Какие существуют общие типы ограничений (constraint) в реляционных базах данных? Расскажите кратко о назначении каждого ограничения.

Ответ

Для реляционных баз данных существуют следующие общие constraint-ы:

- Ограничения-проверки (CHECK): наиболее общий тип ограничений, предназначен для контроля задаваемого полю таблицы значения заданному в выражении условию;
- NOT NULL: указывает, что столбцу нельзя присваивать значение NULL;
- Ограничения уникальности (UNIQUE): гарантируют, что данные в определённом столбце или группе столбцов уникальны среди всех строк таблицы;
- Первичные ключи (PRIMARY KEY): означает, что образующий ограничение столбец (или группа столбцов) может быть уникальным идентификатором строк в таблице;
- Внешние ключи (FOREIGN KEY): указывает, что значения столбца (или группы столбцов) должны соответствовать значениям в некоторой строке другой таблицы
- *Ограничения-исключения (Postgres) (EXCLUDE): гарантирует, что при сравнении любых двух строк по указанным столбцам или выражениям с помощью заданных операторов, минимум одно из этих сравнений возвратит false или NULL

Дополнительные вопросы

Q: Что произойдет если нарушить CHECK?
A: Ошибка модификации данных на уровне самой базы данных и rollback транзакции

Q: Можно ли как-то визуально улучшить сообщения об ошибках нарушения ограничения CHECK средствами БД?
A: Можно задать ограничению отдельное имя - в этом случае при ошибке будет возвращено имя нарушенного ограничения.

Q: Может ли ограничение CHECK или UNIQUE работать на всю таблицу целиком? Если да, приведите пример.
A: CHECK (так же как и UNIQUE) может ссылаться на несколько столбцов, в этом случае ограничение проверяет данные всех включенных столбцов при модификации:

```
CREATE TABLE TEST_TABLE (
    A bigint,
    B bigint,
    C bigint,
    CHECK(A - B >= C),
    UNIQUE (b, c)
                    );
```

Q: Что такое значение NULL?
A: Это указатель на отсутствие реального значения поля таблицы.

Q: Допустим мы создаем таблицу:

```
CREATE TABLE TEST_TABLE (
    A serial,
    B bigint,
    C bigint);
```

и выполняем команду на вставку строки:

```
insert into test_table(b) values (5);
```

Что вернет следующий запрос?

```
select b - c as delta from test_table;
```

A: Запрос вернет NULL: недопустимо осуществлять операции сравнения или арифметические с столбцами, где разрешено хранить NULL без обертки такого столбца в COALESCE или аналогичную по назначению функцию. Автоматического преобразования к значению по умолчанию базового типа данных столбца не происходит.

Q: (На внимательность) Допустим мы создаем таблицу:

```
CREATE TABLE TEST_TABLE (
    A serial,
    B bigint,
    C bigint,
    PRIMARY KEY (a, c));
```

и выполняем команду на вставку строки:

```
insert into test_table(b) values (5);
```

что вернет следующий запрос?

```
select b - COALESCE(c, 0) as delta from test_table;
```

A: Ничего не вернет, так как команда insert не отработает - будет ошибка нарушения первичного ключа: для данных в ключе недопустимы NULL-ы

Q: Что происходит с таблицей при добавлении первичного ключа?
A: Автоматически создаётся уникальный индекс (B-дерево) для столбца (или группы столбцов), перечисленных в первичном ключе, и данные столбцы помечаются как NOT NULL

Q: Для чего нужны FOREIGN KEY?
A: Для поддержания ссылочной целостности данных средствами самой базы данных.

Q: Допустим мы имеем базу данных из следующих таблиц

```
CREATE TABLE products (
    product_no integer PRIMARY KEY,
```

```
    name text,
    price numeric
);

CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    shipping_address text
);

CREATE TABLE order_items (
    product_no integer REFERENCES products,
    order_id integer REFERENCES orders,
    quantity integer,
    PRIMARY KEY (product_no, order_id)
);
```

Как сделать так, чтобы

- при удалении записи из таблицы products гарантированно не удалялись все записи из таблицы order_items с тем же продуктом?
- при удалении из таблицы заказов автоматически удалялись все записи этого заказа из order_items?

A: Модифицировать таблицу order_items следующим образом:

```
CREATE TABLE order_items (
    product_no integer REFERENCES products ON DELETE RESTRICT,
    order_id integer REFERENCES orders ON DELETE CASCADE,
    quantity integer,
PRIMARY KEY (product_no, order_id)
);
```

Q: Приведите плюсы и минусы практики применения FOREIGN KEY
A:
Плюсы: гарантированный контроль целостности данных на уровне базы данных без лишних усилий.
Минусы:

- возможно поймать неожиданное поведение RESTRICT / CASCADE и потерять данные либо привести к ситуации когда при кажущейся целостности базы будут сломаны контроли данных на уровне приложения вплоть до полной неработоспособности
- сложности в модификации схем данных при работе с шардами баз
- (самое важное что хотелось бы услышать) Для нагруженных приложений будут вызываться каскадные блокировки страниц данных всех связанных ключами таблиц, что скажется на производительности базы данных и времени отклика модифицирующих запросов. Так же, это приведет к сложностям при массовых манипуляциях с данными вплоть до полной потери производительноти до тех пор, пока не будет завершена большая модифицирующая транзакция. Большие риски поймать deadlock-и

Библиотека

GO › Базы данных

17 - 18 грейд · 15 мин · 3043 раза · Свернуть

Нужно описать модель библиотеки. Есть 3 сущности: "Автор", "Книга", "Читатель"
Физически книга только одна и может быть только у одного читателя. Нужно составить таблицы для библиотеки так что бы это учесть.

Ответ

17 грейд

Таблицы для библиотеки созданы в псевдокоде.
Задачи решаются с подзапросами без использования join.

18 грейд

Таблицы для библиотеки созданы в одном из диалектов SQL
Созданы требуемые индексы
Учтено, что у книги может быть несколько авторов
Решены все три задачи преимущественно без подзапросов.

Дополнительные вопросы

Q: Написать запрос - выбрать названия всех книг которые на руках
A: (как примерно должно выглядеть) Inner join по кникам на руках и таблице "Книга"

Q: Написать запрос - выбрать названия всех книг в библиотеке у которых больше 3 авторов
A: (как примерно должно выглядеть) Left join по кникам на руках, таблице "Книга" и "Атовр" с условием что книга не на руках и группировкой по автору. Условие что count авторов > 3

Q: Написать запрос - выбрать имена топ 3 читаемых авторов на данный момент
A: (как примерно должно выглядеть) Inner join по кникам на руках, таблице "Книга" и "Атовр" и группировкой по автору. Сортировка по count limit 3

Интерактивная задача на кодинг, приближенный к условиям работы

GO › Go (практика)

17 - 19 грейд 15 мин 2095 раз Свернуть

Есть набор урлов.

```
package main

func main() {
  var urls = []string{
    "http://ozon.ru",
    "https://ozon.ru",
    "http://google.com",
    "http://somesite.com",
    "http://non-existent.domain.tld",
    "https://ya.ru",
    "http://ya.ru",
    "http://ёёёё",
  }
}
```

Напишите программу, которая:

1. Поочередно выполнит http запросы по предложенному списку ссылок
   - в случае получения http-кода ответа на запрос "200 OK" печатаем на экране "адрес url - ok"
   - в случае получения http-кода ответа на запрос отличного от "200 OK" либо в случае ошибки печатаем на экране "адрес url - not ok"

2. Модифицируйте программу таким образом, чтобы использовались каналы для коммуникации основного потока с горутинами. Пример:
   - Запросы по списку выполняются в горутинах.
   - Печать результатов на экран происходит в основном потоке

3. Модифицируйте программу таким образом, чтобы нигде не использовалась длина слайса урлов. Считайте, что урлы приходят из внешнего источника. Сколько их будет заранее - неизвестно. Предложите идиоматичный вариант, как ваша программа будет узнавать об окончании списка и передавать сигнал об окончании действий далее.
4. (необязательно, можно обсудить устно, чтобы убедиться, что кандидат понимает идею контекста, либо предложить как домашнее задание) Модифицируйте программу таким образом, что бы при получении 2 первых ответов с "200 OK" остальные запросы штатно прерывались.
При этом необходимо напечатать на экране сообщение о завершении запроса.
5. (необязательно, можно обсудить устно) Предложите отрефакторить код. Какие тесты кандидат написал бы к этому коду?
Предложите написать код теста и интерфейсы, для которых будут генериться моки. (Как показывает практика это самая сложная часть задачи)

Ответ

17 грейд

Написал синхронное решение, которое идет последовательно и выполняет http-запро

18 грейд

Написал решение с использованием канала (куда отправляются ссылки для скачивания), горутин (воркеров), которые выполняют запросы и sync.WaitGroup для ожидания завершения.
Может реализовать остановку выполнения воркеров через context или управляющий канал.

19 грейд

Может написать решение, чтобы ограничивать количество исходящих запросов.
Может самостоятельно написать моки.

Дополнительные вопросы

Первый пункт очень прост и хорош для разминки кандидата, ознакомления с сервисов, чтобы кандидат запустил код, посмотрел как работает. По каждому пункту можно дополнительно обсудить использованные им примитивы, почему выбрано то или иное решение.
Например:

- какие вообще есть примитивы синхронизации в go, как их можно использовать
- как используются закрытые каналы
- что будет если писать или читать в закрытый канал
- как сделать счетчик, агрегирующий информацию из разных горутин (либо atomic либо отдельная горутина)
- как использовать контекст

Перебор паролей

GO › Go (практика)

18 грейд · 15 мин · 526 раз

Свернуть

У нас есть база данных с паролями пользователей, пароли захешированы (функция hashPassword), а так же известен набор символов которые могут быть использованы в паролях (переменная alphabet).
Наша задача реализовать функцию RecoverPassword так, чтобы она восстанавливала пароль по известному хэшу и TestRecoverPassword завершился успешно

Базовые требования:
Решить как угодно

```
package main

import (
    "crypto/md5"
    "testing"
)

var alphabet = []rune{'a', 'b', 'c', 'd', '1', '2', '3'}

func RecoverPassword(h []byte) string {
    // TODO: implement me
    return ""
}

func TestRecoverPassword(t *testing.T) {
    for _, exp := range []string{
        "a",
        "12",
        "abc333d",
    } {
        t.Run(exp, func(t *testing.T) {
```

```
            act := RecoverPassword(hashPassword(exp))
            if act != exp {
                t.Error("recovered:", act, "expected:", exp)
            }
        })
    }
}

func hashPassword(in string) []byte {
    h := md5.Sum([]byte(in))
    return h[:]
}
```

Ответ

Это вопрос на написание кода, который потом будет приложен к протоколу собеседования. Простой пример по перебору (максимальное число перебираемых комбинаций - math.MaxInt64)

```
func RecoverPassword(h []byte) string {
    var step int

    for ; ; step++ {
        guess := genPassword(step)
        if bytes.Equal(hashPassword(guess), h) {
            return guess
        }
    }
}

func genPassword(step int) (res string) {
    for {
        res = string(alphabet[step%len(alphabet)]) + res
        step = step/len(alphabet) - 1
        if step < 0 {
            break
        }
    }

    return
}
```

Дополнительные вопросы

Q: Как сделать подбор константным по сложности, если мы можем ограничить длину пароля?
A: Использовать rainbow table.

Q: Вычислительная сложность подбора пароля?
A: O(a^n), а для n-битовой хеш-функции сложность нахождения первого прообраза составляет O(2^n)

Q: Как атакующий может скомпрометировать криптосистему?
A: Через timing-attack. Защитой будет сравнение за константное время, кол-во попыток, ограничение по времени в случае одноразовых паролей из SMS.

Q: Как разработчики сервиса могли бы усложнить подбор паролей?
A: крипто стойкое хеширование, соль, сравнение за константное время

Ожидание завершения горутин

GO › Go (практика)

17 - 18 грейд 5 мин 1892 раза Свернуть

Что выведет на экран эта программа?

```
func main() {
    for i := 0; i < 5; i++ {
```

```
        go func() {
            fmt.Println(i)
        }()
    }
}
```

Ответ

17 грейд

Поведение не определено - скорее всего ничего не будет выведено на экран поскольку программа завершится не дождавшись горутин. Также в программе есть вторая проблема - даже если дождаться всех горутин на экран скорее всего будет выведено одно и то же число поскольку все горутины используют одну и ту же переменную i (исправлено в go 1.22 и выше)

18 грейд

Знает, что с go 1.22 проблема переменной уже не актуальна.
Решил через sync.WaitGroup
Предпочтительны решения с использованием defer

Дополнительные вопросы

Q: *Как можно было бы изменить программу чтобы дожидаться всех созданных горутин?*
A: Например можно использовать sync.WaitGroup:

```
func main() {
    wg := sync.WaitGroup{}
    for i := 0; i < 5; i++ {
        wg.Add(1)
        go func() {
            defer wg.Done()
            fmt.Println(i)
        }()
    }
    wg.Wait()
}
```

Q: *Как можно было бы изменить программу чтобы каждая горутина правильно выводила свой индекс?*
A: Требуется использовать из горутины локальную копию i или явно передавать i в горутину как аргумент:

```
func main() {
    var wg sync.WaitGroup

    for i := 0; i < 5; i++ {
        wg.Add(1)
        go func(k int) {
            defer wg.Done()
            fmt.Println(k)
        }(i)
    }

    wg.Wait()
}
```

In-memory cache

GO › Go (практика)

18 - 20 грейд 15 мин 2339 раз Свернуть

Нужно написать простую библиотеку in-memory cache.
Для простоты считаем, что у нас бесконечная память и нам не нужно задумываться о
удалении ключей из него.
Реализация должна удовлетворять интерфейсу:

```
type Cache interface {
    Set(k, v string)
    Get(k string) (v string, ok bool)
}
```

Ответ

18 грейд

Решил в лоб с sync.(RW)Mutex'ом, без применения шардирования, правильно инициализировал map в конструкторе.

Может подсветить проблемы в использовании: переполнение памяти, дублирование памяти из-за нескольких процессов, нету механизма для инвалидации кеша (ключи будут жизни вечно),

19 грейд

Рассказал про различие sync.Mutex + sync.RWMutex. Решил с шардированием и рассказал про алгоритмы хеширования и смог либо написать свою реализацию, либо вызов из std lib. Предложил решение проблем, описанных в 18 грейде.
Может рассказать про плюсы/минусы такого решения в сравнении с, например, memcached.

20 грейд

Может порассуждать на темы:

- нужно ли кешировать ответы с ошибками или отсутствие данных
- что лучше: LRU или TTL?
- когда нужно использовать централизованное кеширование или распределенное
- есть ли смысл в кеше, который не помещается в память и частично сохраняется на диск

Дополнительные вопросы

Q: *Если кандидат написал map + sync.Mutex, то спросить можно ли найти решение лучше?*
Q: *Если для решения использовался sync.RWMutex, то спросить в каких случаях это будет работать медленно?*
A: При большом количестве запросов lock contention будет расти и любая запись будет тормозить чтение.

Q: *Как это можно решить?*
A: Шардировать данные, попросить написать решение с учетом шардинга

Q: *Как шардировать данные?*
A: Ожидается ответ про алгоритмы хеширования (md5, crc и т.д.)

Q: *Сколько шардов нужно делать?*
A: Не меньше, чем кол-во потоков, а лучше больше и кратно этому значению.

Q: *можно ли сделать еще более быстрый кеш, если у нас мало ключей и мы заранее знаем, что они идут от "1" до "10000" и других ключей не бывает?*
A: Можно использовать слайс, заранее выделенного размера.

Вопрос про "звездочку"

GO › Go (практика)

18 грейд · 5 мин · 501 раз

Свернуть

Это почти реальный пример из одного из наших сервисов. Этот код - обертка над кешем, который соотвественно пишет и читает данные из кэша. Не смотря на то, что внедрение данного кэша должно было облегчить основное хранилище, однако это не произошло. Почему?

```
type Storage struct {
    cache *lru.Cache
}

func (s *Storage) Set(wh *warehouse.Warehouse) {
    s.cache.Put(wh.Id, *wh)
}
```

```
func (s *Storage) Get(id types.WarehouseId) *warehouse.Warehouse {
    item, ok := s.cache.Get(id)

    if ok {
        if wh, ok := item.(*warehouse.Warehouse); ok {
            return wh
        }
    }

    return nil
}
```

Ответ

В кэш кладется `warehouse.Warehouse`, а assert делается с `*warehouse.Warehouse`.

Merge каналов

GO › Go (практика)

18 грейд · 10 мин · 1827 раз

Свернуть

Написать код функции, которая делает merge N каналов. Весь входной поток перенаправляется в один канал.

```
func merge(cs ...<-chan int) <-chan int {
    ...
}
```

Ответ

Это вопрос на написание кода, который потом будет приложен к протоколу собеседования. Пример реализации:

```
func merge(cs ...<-chan int) <-chan int {
    var wg sync.WaitGroup
    out := make(chan int)

    // Start an output goroutine for each input channel in cs.  output
    // copies values from c to out until c is closed, then calls wg.Done.
    output := func(c <-chan int) {
        for n := range c {
            out <- n
        }
        wg.Done()
    }
    wg.Add(len(cs))
    for _, c := range cs {
        go output(c)
    }

    // Start a goroutine to close out once all the output goroutines are
    // done.  This must start after the wg.Add call.
    go func() {
        wg.Wait()
        close(out)
    }()
    return out
}
```

Закрытие канала

GO › Go (практика)

18 грейд · 5 мин · 996 раз

Свернуть

Что выведет данная программа? Она отработает корректно?

```
func main() {

    ch := make(chan int)

    go func() {
        for i := 0; i < 5; i++ {
            ch <- i
        }
    }()

    for n := range ch {
        fmt.Println(n)
    }
}
```

Ответ

В программе содержится ошибка, она сначала выведет числа от 0 до 4, а потом упадет с сообщением "all goroutines are asleep - deadlock!". Чтобы починить программу надо в конце горутины закрывать канал:

```
func main() {

    ch := make(chan int)

    go func() {
        for i := 0; i < 5; i++ {
            ch <- i
        }
        close(ch)  // <<<
    }()

    for n := range ch {
        fmt.Println(n)
    }
```

Пользовательский тип ошибки

GO › Go (практика)

17 грейд · 3 мин · 1268 раз

Свернуть

Напишите функцию которая бы возвращала ошибку не импортируя для этого никаких пакетов:

```
func main() {
    println(handle())
}

func handle() error {
    //...
}
```

Ответ

Это вопрос на знание типа error и понимание основ работы с ошибками. error является интерфейсом, чтобы решить задачу требуется создать свою структуру реализующую этот интерфейс и вернуть из функции экземпляр структуры:

```
func main() {
    println(handle())
}

func handle() error {
    return &customError{}
```

```
}

type customError struct {}

func (e *customError) Error() string {
    return "Custom error!"
}
```

pprof

GO > Go (теория)

17 - 21 грейд 5 мин 839 раз Свернуть

pprof. Что это такое и зачем нужно?

Ответ

17 грейд

Слышал/где-то читал, что это нужно для того, чтобы снять профили.

18 грейд

Может назвать как позапускать, какие флаги есть у консольной команды. Есть реальный опыт работы, может привести пример из реальной жизни.

19 грейд

Может на практике рассказать, что он видит на подготовленных картинках.

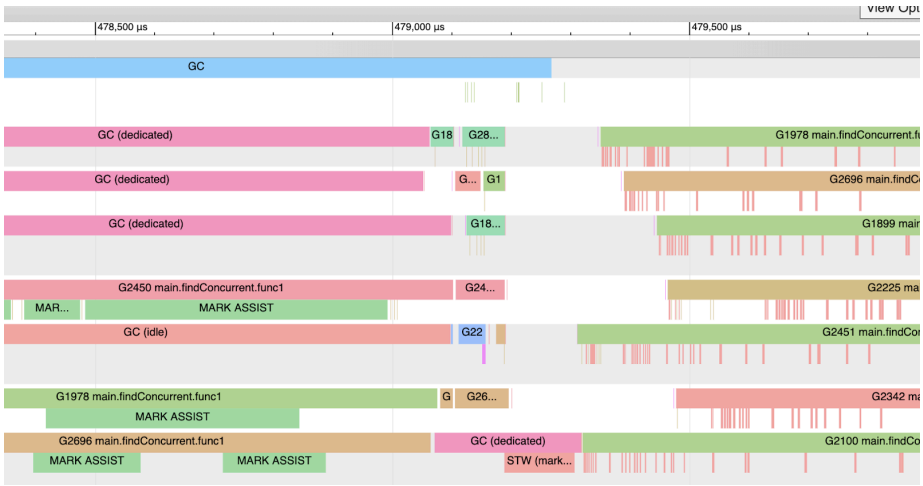
НУЖНО ДОБАВИТЬ ПРИМЕР ГРАФА и ФЛЕЙМГРАФА

20 грейд

Может рассказать как работает семплирующий профайлер: Профайлер сохраняет не каждое событие, а каждое N-ное. В случае CPU-профайлера это значит, что стектрейсы. Может рассказать почему при получении профиля по heap/alloc - это происходит мгновенно, а для CPU нужно подождать какое-то время.

21 грейд

Может рассказать что здесь происходит.
https://www.ardanlabs.com/blog/2018/12/garbage-collection-in-go-part1-semantics.html



Дополнительные вопросы

Q: *Как можно снять профили?"
A: Можно снимать профили, используя программный API, либо подключив пакет `net/http/pprof`.

Q: *Что такое семплирующий профайлер?*
A: Профайлер сохраняет не каждое событие, а каждое N-ное. В случае CPU-профайле это значит, что стектрейсы.

Q: *Есть ли оверхед от использования pprof'a?*
A: Для профилей heap, alloc, mutex, block - оверхед минимальный, по сути эти данные собираются всегда. CPU собирается только по требования и явно добавляет оверхед, конкретные значения нужно замерять для конкретного приложения.

Q: *Что такое trace и чем он отличается от других профилей?*
A: Trace предоставляет очень подробную информацию о выполнении приложений:

- создание / блокировка / разблокировка горутин
- syscalls
- работа GC
- размер heap'a, количество тредов и горутин

Т.к. trace очень подробная штука, то он добавляет больше оверхеда, чем остальные профили.

Q: Показать картинку с флеймграфом и попросить рассказать он там видит.

Maps

GO › Go (теория)

17 - 18 грейд · 5 мин · 1888 раз · Свернуть

Что выведет данный код?

```
m := map[string]int{"a":1,"b":2,"c":3}

for a, b := range m {
    fmt.Println(a, b)
}
```

Ответ

17 грейд
Выведется все элементы map'ы в виде [ключ, значение]

18 грейд
Элементы мапы будут выведены в случайном порядке. Это сделано специально, чтобы код не был завязан на конкретную реализацию мапы. Если важен порядок, то нужно использовать слайсы или массивы.

Incomparable types

GO › Go (теория)

17 - 18 грейд · 5 мин · 469 раз · Свернуть

```
a := map[B]int{}
a[d] = 0
e, ok := a[d]
```

В каких случаях данный код даст ошибку?

Ответ

17 грейд

1. B - incomparable type, т.е. типы, которые не поддерживается операторы сравнение `==` и `!=`
2. B - не тип
3. d - не является типом B
4. a или e уже определены выше, B/d не определены
5. race condition из-за конкурентной записи в map

Если называет ~2/3 из этого списка, то 17 грейд

18 грейд
Должен назвать все пункты из 17 грейда.

Дополнительные вопросы

Q: *Какие типы в go являются incomparable?*
A: go не поддерживает сравнение для таких типов:

- slice
- функции
- структуры с incomparable полями
- array с incomparable типами

Интерфейсы

GO › Go (теория)

17 - 18 грейд 10 мин 1445 раз Свернуть

1. Что такое интерфейс в Go? Как он устроен?

```
type Foo struct{}

func (f *Foo) A() {}
func (f *Foo) B() {}
func (f *Foo) C() {}

type AB interface {
    A()
    B()
}

type BC interface {
    B()
    C()
}

func main() {
    var f AB = &Foo{}
    y := f.(BC) // сработает ли такой type-assertion?
    y.A() // а этот вызов?
    _ = y

}
```

Ответ

17 грейд

Интерфейс в Go — это абстракция поведения других типов: определение, описывающее конкретные методы, которые должны быть у какого-то другого типа. В этом случае можно утверждать, что "тип удовлетворяет интерфейсу".
Интерфейсы позволяют определить обобщенную реализацию без привязки к конкретному типу.
Когда в Go объявление (переменной, параметра функции или поля структуры) имеет интерфейсный тип, можно использовать объект любого типа, пока он удовлетворяет интерфейсу в объявлении.

Рассказал что такое duck-typing.

Может назвать несколько интерфейсов из stdlib: error, io.Writer, io.Reader, context.Context, stringer, etc

18 грейд

Ответил на вопросы из примера кода.

Дополнительные вопросы

Q: *Что такое "пустой интерфейс" (interface{})?*
A: Пустой интерфейсный тип не описывает методы. Любой объект удовлетворяет пустому интерфейсу.

Q: *Когда допустимо применять пустой интерфейс?*
A: В случае реализации кода, который должен уметь работать с любым типом данных
Например, функция fmt.Print стандартной библиотеки.

Q: *Каковы основные отличия применения интерфейса в Go от интерфейсов классического ООП-языка (Java, C#, etc)?*
A: Разница в том, что обычно в ООП-языках имплементация интерфейса указывается для класса явно, в Go имплементация неявная (duck typing).

Горутины и потоки

GO › Go (теория)

17 грейд · 5 мин · 3471 раз

Свернуть

Чем горутины отличаются от потоков?

Ответ

Горутины гораздо более легковесны и требуют меньше памяти, поэтому их можно создать больше. Горутины шедулятся рантаймом языка в юзерспейсе, в то время как потоки шедулятся операционной системой в кернелспейсе. Из этого следует также и меньшее количество переключений контекста в случае использования горутин.

Дополнительные вопросы

Q: *Что можно сказать про стек горутины и стек потока?*
A: Изначально у горутины небольшой стек который может расти до 1 гб на 64-битных ОС и до 256 мб на 32-битных системах. У потока же изначально большой стек который не может динамически увеличиваться.
https://github.com/golang/go/blob/go1.22.0/src/runtime/proc.go#L157

Q: *Сколько можно создать горутин?*
A: Количество горутин не ограничено, можно создавать столько на сколько хватит памяти

Q: *Что такое GOMAXPROCS?*
A: Это переменная ограничивающая количество потоков ОС используемых для исполнения горутин

Defer

GO › Go (теория)

17 - 19 грейд · 5 мин · 1618 раз

Свернуть

Что такое defer? Что выведет следующий код?

```
type X struct {
    V int
}

func (x X) S() {
    fmt.Println(x.V)
}

func main() {
    x := X{123}
    defer x.S()
    x.V = 456
}
```

После ответа - спросить, что изменится, если изменить код

```
...
func main() {
    x := X{123}
    defer func(){ x.S() }()
    x.V = 456
}
...
```

Ответ

17 грейд

Defer гарантирует отложенный вызов функциии по мере выполнения программы. Defer добавляет функцию в стек вызова приложения.
В первом случае ответ: `123`, во втором - `456` (передача по ссылке)

18 грейд

Q: *Что произойдет, если писать несколько defer подряд?*
A: Они вызовутся в обратном порядке, так как это стек вызова функций.

19 грейд

Q: В каких случаях defer вызван не будет?
A: os.Exit(), паника, log.Fatal (внутри вызывает os.Exit)

Итерационные переменные

GO › Go (теория)

17 - 18 грейд · 5 мин · 1217 раз

Свернуть

Что выведет данный код?

```
func main() {
    values := []int{1,2,3,4,5}
    for _, val := range values {
        go func() {
            fmt.Println(val)
        }()
    }
    time.Sleep(100 * time.Millisecond)
}
```

После ответа - как исправить?

Ответ

17 грейд

Одинаковые адреса и одинаковые значения

```
5
5
5
5
5
```

Исправить:

```
func main() {
    values := []int{1,2,3,4,5}
    for _, val := range values {
        val := val // Копируем переменную
        go func() {
            fmt.Println(val)
        }()
    }
    time.Sleep(100 * time.Millisecond)
}
```

или

```
func main() {
    values := []int{1,2,3,4,5}
    for _, val := range values {
        go func(val int) {
            fmt.Println(val)
        }(val)
    }
    time.Sleep(100 * time.Millisecond)
}
```

18 грейд

Смог решить без использования time.Sleep.

Знает, что начиная с go 1.22 были изменены правила определения области видимости для переменных в циклах.

Замыкания

GO › Go (теория)

18 грейд · 5 мин · 717 раз

Свернуть

Что такое замыкание? Приведите пример использования замыкания.

Ответ

Замыкание - это функция, которая ссылается к переменным вне ее тела. Функция имеет доступ к связанным переменным, а также может присваивать им значения
Пример:

```
func accum() func(int) int {
    sum := 0
    return func(x int) int {
        sum += x
        return sum
    }
}
```

Функция accum возвращает замыкание

Мьютексы

GO › Go (теория)

17 - 18 грейд · 5 мин · 2335 раз

Свернуть

Что такое mutex и зачем он нужен?

Ответ

17 грейд

Это примитив синхронизации, который нужен для эксклюзивного доступа к какому-то ресурсу. Если другому потоку нужен доступ к ресурсу, то он блокируется до тех пор, пока другой поток не освободит блокировку.

18 грейд

Кандидат может рассказать как устроен mutex "под капотом", знает про RWMutex и его разницу с Mutex.

Дополнительные вопросы

Q: Чем отличается RWMutex от Mutex? A: RWMutex нужен в том случае, если у нас есть потоки, которым нужен доступ на чтение ресурса, но не нужна запись. В этом случае читать может много потоков одновременно.

Каналы

GO › Go (теория)

18 - 19 грейд · 10 мин · 3138 раз

Свернуть

Что такое каналы? Зачем нужны?

Ответ

18 грейд

Каналы - механизм взаимодействия между горутинами. Потокобезопасны, то есть к одному каналу могут обращаться несколько горутин одновременно.

Q: *Какие они бывают?*

A: Буферизированные(асинхронные) и небуферизированные(синхронные)

Q: *К чему приведут операции чтения/записи над закрытым каналом?*
A: Чтение вернет нулевое значение, запись вызовет панику

Q: *К чему приведут операции чтения/записи над nil каналом*
A: Чтение из nil канала приведет к блокировке навсегда.
Запись в nil канал приведет к блокировке навсегда.
Закрытие такого канала приводит к panic.

Q: *Какой еще способ блокировки навсегда?*
A:

```
select {
}
```

(Если это не совсем очевидно, можно сначала спросить про select, а потом вернуться к этому вопросу)

Q: *Что такое select и как он работает*
A: Если вкратце: switch для каналов

Q: *Другие примитивы синхронизации*
A: Mutex, WaitGroup

19 грейд

Q: *Как устроен внутри?*
A: Структура с метаданными по типу "размер буффера", "закрыт/открыт", "связанные горутины" и тд. Также в этой структуре есть мьютекс для безопасного доступа к каналу

Q: *Паттерны конкурентного программирования, использующие каналы*
A: Генераторы, Fan-In, Fan-Out, Управляющий канал и тд

Race condition

GO › Go (теория)

18 - 19 грейд · 5 мин · 1488 раз · Свернуть

Можно ли передать переменную в несколько горутин? Пример (с ошибкой) - что будет?

```
x := make(map[int]int, 1)
go func() { x[1] = 2 }()
go func() { x[1] = 7 }()
go func() { x[1] = 10 }()
time.Sleep(100 * time.Millisecond)
fmt.Println("x[1] =", x[1])
```

Ответ

18 грейд

Можно, нужно учесть race condition (mutex) Что будет - будет паника `concurrent map writes` (при условии GOMAXPROCS != 1)
Решение:

```
x := make(map[int]int, 1)
lock := sync.RWMutex{}
go func() {
    lock.Lock()
    x[1] = 2
    lock.Unlock()
}()

go func() {
    lock.Lock()
    x[1] = 7
    lock.Unlock()
}()
```

```
go func() {
    lock.Lock()
    x[1] = 11
    lock.Unlock()
}()

time.Sleep(100 * time.Millisecond)
fmt.Println("x[1] =", x[1])
```

19 грейд

Может продемонстрировать решение без sync.Mutex/sync.RWMutex

Дополнительные вопросы

Q: *Какие знаешь средства для предотвращения RC*
A: Например mutex (может свой вариант сделать)

Q: *какие mutex бывают и чем отличаются для чего нужны?*
A: Бывают полностью блокирующие или блокировка только на запись (RWMutex)

Слайс

GO › Go (теория)

17 - 18 грейд 5 мин 2318 раз Свернуть

Что такое слайс? Как он устроен?

Ответ

17 грейд

Это абстракция, построенная на основе массива. Обобщенно, слайс это тип данных, построенный как обертка над массивом, представляющая собой часть массива от 0 до N элемента.
От массива слайс отличает следующее:

1. это не массив, но структура, содержащая ссылку на часть массива;
2. может менять свой размер и динамически выделять память;

Внутри слайс представляет собой структуру вида

```
type slice struct {
    array unsafe.Pointer
    len   intcap   int
}
```

Где:

- len (length, длина) — текущая длина слайса
- cap (capacity, вместимость) — длина внутреннего массива: размер массива под который выделена память
- array - собственно ссылка на массив с данными

18 грейд

Знает, особенности создания слайсы при использовании двух аргументов make.
Может рассказать о росте памяти при append.

Дополнительные вопросы

Q: На что влияет параметр вместимости слайса cap(acity)?
A: Cap — ключевой параметр для выделения памяти, влияет на производительность вставки в слайс: в случае, если вместимость на момент определения была задана недостаточной, вставка в слайс приведет к частому выделению памяти по мере добавления новых элементов. В идеале, capacity слайса в момент создания должна быть указана максимально близкой к плановому количеству элементов.

Q: Какие способы безопасного копирования слайсов вы знаете?

A: 1) Через функцию copy() 2) присвоить возвращаемое append() значение слайса новой переменной в случае добавления элемента при len == cap

Q: Как происходит выделение памяти при росте слайса через append? К каким негативным последствиям может неправильное понимание будущего размера слайса в момент его создания?
A: При размере слайса менее 1024 элементов размер памяти увеличивается вдвое. При размере среза > 1024 элементов, срез увеличивается на четверть текущего размера. Операция вставки в слайс имеет серьезные последствия для памяти - при увеличении capacity массив будет скопирован в новый и размер выделенной памяти будет расти по своей внутренней логике, лишь отчасти связанной с требуемой емкостью.

Q: Когда сборщик мусора удалит массив под слайсом?
A: Массив будет удален только после того, как не остается ни одного слайса, который ссылается на элементы этого массива.

Q: Какие особенности получения нового среза от ранее существующего слайса вы знаете?
A: Слайсниг не производит копирование данных слайса. Создаётся новое значение слайса, указывающее на исходный массив.

Q: Чем опасно создание через make слайса с явно указанными len & cap?
A: При создании слайса с указаной длиной и емкостью будет выделена память под емкость и при этом инициализированы значениями по умолчанию (для типа элементов слайса) len элементов массива. В дальнейшем они будут участвовать в итерировании и являться реальными значениями массива, которые могут внести искажение в данные.

Строка в Go

GO › Go (теория)

17 - 18 грейд · 5 мин · 1467 раз

Свернуть

Что из себя представляет строка в Go?

Ответ

17 грейд

В Go строкой является срез (slice) байтов, который доступен только для чтения.

18 грейд

Кандидат знает, как иттерироваться посимольно (рунами) по строке, знает как подсчитать количество рун в строке.

Дополнительные вопросы

Q: *Чем отличается байт от руны в Go?*
A: Руны это тип в GO, int32, предназначен для представления Unicode и символов для кодирования которых нужно 32 бита

SQL запрос на JOIN нескольких таблиц с условием

GO › Скрининг

Скрининг · 10 мин · 5328 раз

Свернуть

Есть база с такой схемой данных

```
// user
id | firstname | lastname | birth
1  | Ivan      | Petrov   | 1996-05-01
2  | Anna      | Petrova  | 1999-06-01
3  | Anna      | Petrova  | 1990-10-02

// purchase
sku| price | user_id | date
1  | 5500  | 1       | 2021-02-15
1  | 5700  | 1       | 2021-01-15
2  | 4000  | 1       | 2021-02-14
3  | 8000  | 2       | 2021-03-01
```

```
4  | 400   | 2       | 2021-03-02

// ban_list
user_id | date_from
1       | 2021-03-08
```

Нужно вывести:

1. Вывести уникальные комбинации пользователя и id товара для всех покупок, совершенных пользователями до того, как их забанили. Отсортировать сначала по имени пользователя, потом по SKU
2. Найти пользователей, которые совершили покупок на сумму больше 5000р. Вывести их имена в формате id пользователя | имя | фамилия | сумма покупок

Ответ

1) Здесь мы проверяем что человек умеет в джоины, distinct, where, order

```
SELECT distinct u.id, firstname, lastname, p.item_id
FROM users u
     join purchase p ON u.id = p.user_id
     left join ban_list bl ON u.id = bl.user_id -- не забыть left join, а
то в запрос попадут только покупки забаненных пользователей.
WHERE bl.user_id IS NULL                        -- пользак не забанен
   OR bl.date_from > p.date                     -- или забанен позже, чем
совершена покупка
ORDER BY lastname, firstname, u.id, p.item_id   -- лучше бы кандидат
догадался или спросил, что в сортировке по имени надо сначала ставить
фамилию, потом имя.
```

2) Здесь мы проверяем что человек умеет в HAVING, знает, чем having отличается от WHERE

```
SELECT u.id, u.firstname, u.lastname, SUM(p.price)
FROM users u join purchase p ON u.id = p.user_id
GROUP BY u.id, u.firstname, u.lastname -- и знает, что аггрегирующие
функции без group by не будут работать
HAVING SUM(p.price) > 5000


-- В принципе, вариант

SELECT * FROM (
    SELECT u.id, u.firstname, u.lastname, SUM(p.price) s
    FROM users u join purchase p ON u.id = p.user_id
    GROUP BY u.id, u.firstname, u.lastname
) WHERE s > 5000

-- тоже является корректным, но все равно надо спросить про HAVING
```

Определить является ли слайс монотонным

GO › Скрининг

Скрининг 10 мин 2172 раза Свернуть

Является ли слайс монотонным?
Монотонная функция - функция одной переменной, определённая на некотором подмножестве действительных чисел, которая либо везде (на области своего определения) не убывает, либо везде не возрастает.

```
{1,7} - true
{1,1} - true
{3,3,1} - true
{9,5,1} - true
{23,5,23} - false
```

Ответ

```
func isMonotonic(in []int) bool {
    isUp, isDown := true, true
    for i := 1; i < len(s); i++ {
        isDown = isDown && s[i - 1] >= s[i]
        isUp = isUp && s[i - 1] <= s[i]
    }

    return isUp || isDown
}
```

Фильтрация элементов слайса

GO › Скрининг

Скрининг 10 мин + 669 раз Свернуть

Дан слайс целых чисел. Напишите функцию remove, удаляющую все нули

```
Примеры:
    remove([]) -> []
    remove([0]) -> []
    remove([1,0,0,2]) -> [1,2]
```

Ответ

```
func remove(in []int) []int {
    i := 0
    j := 0

    for ; i < len(in); i++ {
        if in[i] != 0 {
            in[j] = in[i]
            j++
        }
    }

    return in[:j]
}
```

Дополнительные вопросы

Оценить сложность решения

Функция zip

GO › Скрининг

Грейд неизвестен + 1565 раз Свернуть

Требуется реализовать функцию zip, которая соединяет элементы двух слайсов в слайс пар

```
func main() {
    s1, s2 := []int{1, 2, 3}, []int{4, 5, 6, 7, 8}
    fmt.Println(zip(s1, s2)) // [[1 4] [2 5] [3 6]]
}

func zip(s1 []int, s2 []int) [][]int {
    //...
}
```

Ответ

```
func zip(s1 []int, s2 []int) [][]int {

    minLen := len(s1)
    if len(s2) < minLen {
        minLen = len(s2)
    }

    res := make([][]int, 0, minLen)
    for i := 0; i < minLen; i++ {
        res = append(res, []int{s1[i], s2[i]})
    }

    return res
}
```

Дополнительные вопросы

Q: Реализовать версию функции zip которая сможет соединять произвольное количество слайсов
A:

```
func zip(s ...[]int) [][]int {

    if len(s) == 0 {
        return [][]int{}
    }

    minLen := len(s[0])
    for i := 1; i < len(s); i++ {
        if len(s[i]) < minLen {
            minLen = len(s[i])
        }
    }

    res := make([][]int, 0, minLen)
    for i := 0; i < minLen; i++ {
        x := make([]int, 0, len(s))
        for k := 0; k < len(s); k++ {
            x = append(x, s[k][i])
        }
        res = append(res, x)
    }

    return res
}
```

Генерация N уникальных чисел

GO › Скрининг

Скрининг 5 мин + 4236 раз Свернуть

Требуется реализовать функцию uniqRandn, которая генерирует слайс длины n уникальных, рандомных чисел.

```
import (
    "fmt"
    "math/rand"
)

func main() {
    fmt.Println(uniqRandn(10))
}

func uniqRandn(n int) []int {
```

```
    //...
}
```

Ответ

```
func uniqRandn(n int) []int {
    res, resMap := make([]int, 0, n), make(map[int]struct{}, n)
    for len(res) < n {
        val := rand.Int()
        if _, ok := resMap[val]; ok {
            continue
        }
        res = append(res, val)
        resMap[val] = struct{}{}
    }
    return res
}
```

Задачка про слайсы 3

GO › Скрининг

Грейд неизвестен + 1859 раз

Свернуть

Что выведет программа? Почему?

```
func main() {
    timeStart := time.Now()
    _, _ = <-worker(), <-worker()
    println(int(time.Since(timeStart).Seconds()))
}

func worker() chan int {
    ch := make(chan int)
    go func() {
        time.Sleep(3 * time.Second)
        ch <- 1
    }()
    return ch
}
```

Ответ

```
6
```

Задачка про строки

GO › Скрининг

Скрининг 3 мин + 1783 раза

Свернуть

Что выведет программа?

```
s := "test"
println(s[0]) // ответ кандидата
s[0] = "R"
println(s) // ответ кандидата
```

Ответ

Слабый ответ: выведет 116 (номер байта) и не заменит R, будет test.
Правильный: этот код не скомпилится вовсе.
Продвинутый: поменять этот код так, чтобы он менял байты и все же работал.

Задачка про слайсы 1

GO › Скрининг

Грейд неизвестен + 2575 раз Свернуть

Что выведет программа?

```
func a() {
    x := []int{}
    x = append(x, 0)
    x = append(x, 1)
    x = append(x, 2)
    y := append(x, 3)
    z := append(x, 4)
    fmt.Println(y, z)
}

func main() {
    a()
}
```

Ответ

```
[0 1 2 4] [0 1 2 4]
```