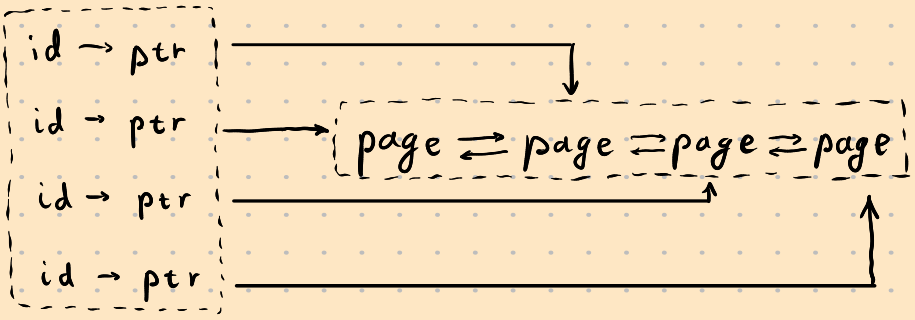
Константин Игоревич Владимиров, Syntacore

Лаборатория RISC-V со среды

От C к C++

LRU - cache - last recent used



Реализация на C

Тяжело: надо писать список + тесты, таблицу + тесты, драйвер + тесты

Дейкстра: „Скромный программист"

Стоит бить программу на модули: header + src file

! Нереально строить надёжные переиспользуемые абстракции (нет std списка например)

Реализация на C++

- Изначально компилится в C
- Одна из главных особенностей: объединение данных и методов их обработки
- Дурной тон: писать явный this (конец 2-го семестра)
- Обобщение данных и методов: template <typename T> (в C: макросы)

  Приходит ценой небольшой когнитивной сложности
- Бенчмарк: quick bench
- В cpp quicksort быстрее: компаратор инлайнится (т.к. qsort не предкомпилирован: он шаблонный).
- Такой подход к std позволяет иметь кучу обобщённых контейнеров (cpp reference)
- В LRU не надо писать ни hashtable, ни list
- Создавай любой объект максимально близко к т. использования
- Семантика значения: ведёт себя как int

А/З: сравнение ARC/2Q/LFU/LIRS с идеальным кэшем (знать будущее)

Стандарт C++ 20

Страуструп 4-th edition

Липман C++ primer

Лекция 3.

Объявления и определения

`int x` - определение (и объявление!)

`struct S;` - объявление

`extern void foo() {}` - определение

`extern S *ps;` - объявление

`int bar();` - объявление

`struct T {int x;};` - определение

`extern int y = 0;` - определение

Класс связывания

`static` - единица приватна для данной ед. трансляции

`extern` - публичный linkage specifier (доступен всем)

Три этапа:

`static` в ф-ии - глобальная пер-ая с обл-ю видимости ф-ии

`static` в `struct` - общая для всех объектов пер-ая

Чтобы тип был безопасен, он должен быть полностью одинаков во всех единицах трансляции, в к-рые попадает.

`inline` ф-ии - одна на все ед. трансляции - писать надо тем кто в header тело ф-ии

`static` ф-ии - по одной на каждую ед. трансляции - раздувает размер пр-ма

`static inline` ≡ `static` в этом смысле

метод внутри класса ≡ `inline` ф-ия (в header)

вне класса - ODR violation

module linkage - чтобы спрятать детали реализации в header-е - лучше чем include!

Модули пока плохо поддерживаются!

Область видимости

function scope, block scope

lifetime – все моменты времени, когда переменная валидна

Пример: int a = a; – scope начался, а lifetime – нет!

Проблемные ссылки и указатели – Beware!

Временный объект живёт до конца полного выражения.

Переопределение

C даёт гарантии по именам! Метка в asm = имя ф-ии (но почти зависит от Calling Conv)

C++ – нет, т.к. манглирование

Модули, скомпилённые разными компиляторами, могут не слинковаться!

Поэтому C будет жить вечно

C++ = C – гарантия имён