


Эксперименты в области сборочного программирования

Алексей Недоря, октябрь 2019

Личная история

- 1983-1984: Эдисон компилятор для МВК Эльбрус
- 1984: Эмулятор Кроноса на Burroughs 6700
- 1985-1989: Кронос: Модула-2 компиляторы (m0, m2, mx), ядро ОС Excelsior
- 1990-1991: M2C – Модула-2 => C транслятор
- 1991: Стажировка в ETHZ (Оберон/Portable Oberon compiler)
- 1992: X2C – Модула-2/Оберон-2 => C транслятор
- 1993-1995: XDS M2/O2, Oberon System Mithril, к.ф.-м.н.
- 1995-1999: XDS для заказчиков, XDS M2 для НПО ПМ
- 2002-2006: Эксперименты с компонентами (Delphi)
- 2007-2013: Среда сборочного программирования Вир
- 2016-2018: Продумывание Вир-2 и компонентного ассемблера
- 2019-?: Язык и экосистема в Huawei RRI

Статьи

- Всеплатформенная разработка или Если б я был султан, 06.2019
- Об изготовлении программ и ежиках в тумане, 05.2019
- Компонентный ассемблер. Часть 2. Дух языка, 01.2019
- Компонентный ассемблер для цифрового пространства, 12.2018
- Технология разработки мультиплатформенных программ на основе явных схем программ, 05.2018
- Интернет – в поиске чистого воздуха, 01.2018
- Забытое 40 лет назад новое и как оно может изменить нашу жизнь, 10.2017
- Вир. Явная схема программы, 11.2017
- Вир. Как устроена исполняемая программа, 11.2016
- Вир. Рабочие столы, 11.2016
- Вир. Столы, инструменты и узлы, 11.2016
- Среда разработки «Вир». Предисловие, 10.2016

Как мы делаем программы в 21 веке?

- **Бессистемное многообразие и аморфность**:
  - много языков, много сред, много парадигм,
  - много изолированных и слабо совместимых экосистем,
  - отсутствие стандартов взаимодействия,
  - отсутствие классификаций (множество классификаций),
  - недостаток (отсутствие) объективных критериев и измерений
- Highly likely подход и PR: Язык X + Framework Y (наверное) (чем-то) лучше для Z: пример: Dart/Flutter

Если мы хотим к звездам:

Систематизация

Структуризация

Стандартизация

Основа систематизации и стандартизации

В качестве альтернативы единому языку программирования мы выдвигаем понятие о языковой среде, которую мы называем лексиконом программирования.

… три формы программирования:

- синтезирующее;
- сборочное;
- конкретизирующее.

А. П. Ершов, Предварительные соображения о лексиконе программирования, 1983

Почему сборочное программирование?

- Размер «Hello» на Go:

  Hello.exe – 2M

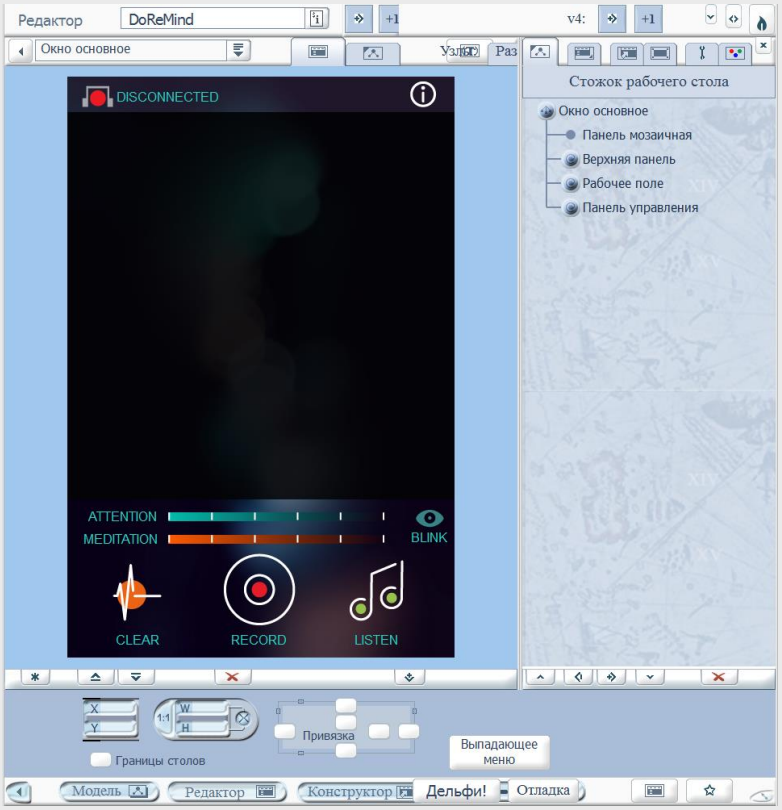
```
package main
import "fmt"
func main() {
    fmt.Println("Hello, world")
}
```

- Инструменты сборки: CMake, Gradle, Cargo (Rust), ….
- Неявная сборка используется ВСЕГДА
- А где явная сборка? Scratch? HTML + CSS + JS?
- Почему Component Pascal – это не сборка и не лексикон?
- Современные фреймворки – это больше конкретизация, чем сборка.
- Сборка позволяет сделать программу из компонент, которые могут быть синтезированы, конкретизированы или собраны из компонент меньшего уровня.

Вир: Экспериментальная среда разработки

Основная идея: программа собирается из бинарных компонентов различных уровней из общего репозитория.

- 2006-2007: Прототип
- 2008-2013: Активное развитие и использование:
  - Использована для разработки нескольких коммерческих продуктов, таких как Sitecraft (www.sitecraft.ru)
  - И самого Вира
- 2014-2016: использовалась для прототипирования приложений (DoReMind, Анализатор Текстов)



Основные идеи

- Программа строится по **явной схеме**, которая определяет, какие компоненты используются и как они взаимодействуют.
- Есть **несколько видов (уровней) компонент** - от низкоуровневых до высокоуровневых. Каждый уровень имеет свои определенные правила взаимодействия и т. д. Конфигурация компонент встроена в схему.
- Каждая компонента работает в **явно заданном контексте** в любой точке исполнения (локальный контекст компоненты + общий контекст: управление памятью, доступ к API и др.).
- Независимость от базовой ОС.

Явная схема программы

- Схема - это дерево деревьев (верстаков). Каждый верстак – дерево компонентов
- При запуске программы загрузчик создает экземпляр верхнего узла: экземпляры всех компонентов на этом узле. Это может быть окно GUI или нет.
- Все остальные верстаки создаются явными операциями
- Двоичный образ программы состоит из:
  - Схемы (текстовый или двоичный формат)
  - Файла кода содержит код всех используемых компонент
  - Файл ресурсов - все ресурсы (в основном картинки), используемые в программе

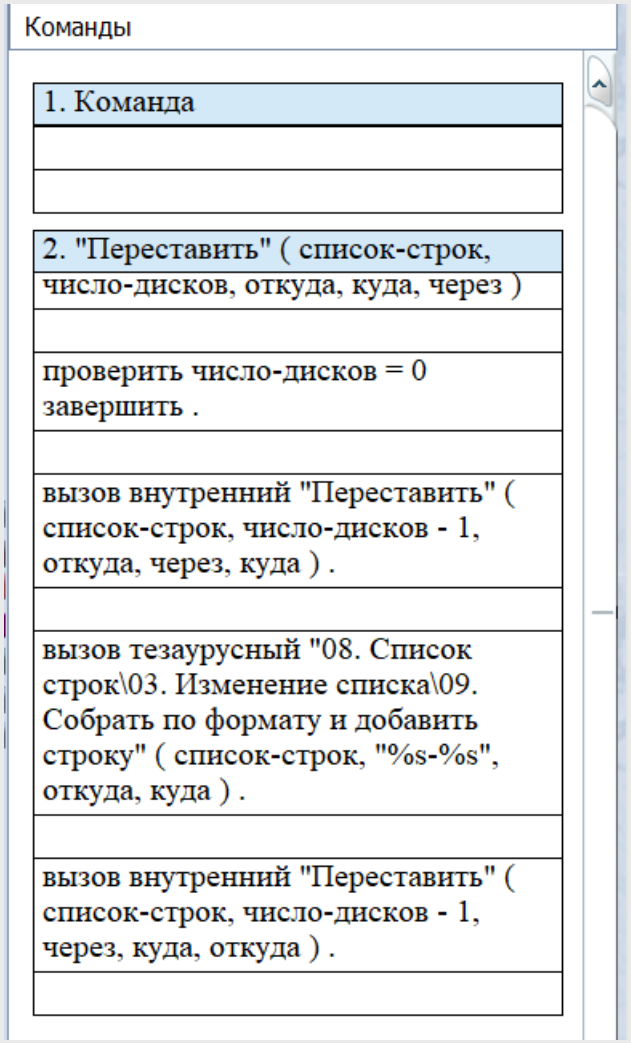
Иерархия компонент

От нижнего уровня до верхнего:

- Статья (тезауруса)
  - Единица компиляции «экспортирующая» функцию
  - Используется для Инструмента, имеет доступ к контексту
- Инструмент
  - Лист в схеме приложения или компоненты
  - «Экспортирует» набор функций
  - Определяет локальный контекст
- Узел
  - Дерево компонент, которые могут быть использованы как единное целое (пример: текстовый редактор)
- Верстак
  - Компонента верхнего уровня

Язык программирования Вир/А0

- Язык программирования Вир.А0 - это **вспомогательный инструмент**, используемый для построения компонент. Если все необходимые компоненты доступны, то нет необходимости писать код (нового компонента).
- Очень простой язык используется с вполне удовлетворительными результатами.
- Для более широкого использования следует использовать более "обычный" язык



Вир: Не очевидные следствия

- Общий репозиторий для всех программ
  - Каждая новая программа использует предыдущие компоненты и добавляет новые компоненты
  - Каждая следующая программа требует меньше усилий
  - DoReMind: только 5% новых компонентов
  - Анализ текста: только 2% новых компонентов
- Унифицированные (программно-независимые) сервисы
  - Журналы, профилирование, обновление, горячая перезагрузка, защита
- Явная архитектура, встроенная в программу. Легко понять, не нужно запоминать, не оторвана от кода.
- Легкий рефакторинг и расширение функциональности.

Продолжение работ

Huawei Research:

- разработка языка и экосистемы для создания приложений
  - Делать унифицированным образом приложения для разных устройств и платформ
  - Увеличить производительность разработчика и Joy of Programming

Вопросы