

Язык *bI*
чтобы никто не догадался

© Dmitry Ponyatov <dponyatov@gmail.com>

25 августа 2015 г.

Оглавление

Язык <i>bI</i>	2
Литература	3
Файлы проекта	4
I Синтаксис	5
Комментарии	6
Точечные команды	7
II Система типов	8
1 Нативные	9
1.1 c: одиночный символ <i>/[c]har/</i>	10
1.2 s: строка <i>/[s]tring/</i>	10
1.3 i: целое <i>/[i]nt/</i>	10
1.4 f: число с плавающей точкой <i>/[f]loat/</i>	10

III	Расширение ☹eclipse	11
	Манифест	12
	Перспектива <i>bI</i>	13

Язык *bI*

- DSL-ориентированный
- динамический
- объектный
- параллельный

язык программирования и проектирования программно-аппаратных систем.

Язык *bI* был создан в экспериментах с синтаксическим анализом, символьными вычислениями, и чтении книг типа [1, 2, 3]. Движущей силой было желание создать [инструмент для метапрограммирования: описания прикладных программ и программно-аппаратных систем в свободном синтаксисе](#), с применением смешанных методов программирования, и без необходимости полностью описывать все объекты программы во всех тонкостях и с соблюдением всех заморочек языка реализации, библиотек, платформ и ОС.

Идея о свободном синтаксисе предполагала, что программист имеет полный контроль над языком, так как сам написал его реализацию, и по мере необходимости меняет работу существующего функционала языковой системы, и дописывает дополнительные фичи. В процессе работы над различными проектами в итоге формируется некий клон базового языка, заточенный под решаемые задачи, под используемые платформы, и с определенным набором средств автоматизации рутинных вопросов, типа генерации кода на Си, автоматического отслеживания зависимостей между файлами и т.п.

Литература

[1] SICP:

Харольд Абельсон, Джеральд Джей Сассман, Джули Сассман

Структура и интерпретация компьютерных программ

[2] Книга Дракона

А.Ахо, Р.Сети, Д.Ульман

Компиляторы: принципы, технологии, инструменты

[3] *Ю.А.Кирютенко, В.А.Савельев*

Объектно-ориентированное программирование Язык SmallTalk

Файлы проекта

Makefile	файл зависимостей
lexer.lpp	лексический анализатор на языке flex
parser.ypp	синтаксический парсер на языке bison
core.cpp	runtime-ядро на C_+^+
bI.hpp	определения встроенных типов
bI.bI	описание модели языка <i>bI</i> следующей версии
doc/	документация в формате \LaTeX


Часть I

Синтаксис

bI построен на bypass-принципе: все, что языковая система получила на вход, и не распознала как элемент языка, отправляется на выход без изменений.

Также используется литературное программирование¹: решение задачи описывается на удобном языке разметки (L^AT_EX, DocBook,...), а *bI* **вычленяет и выполняет блоки кода**. Поэтому комментарии в программах на *bI* **не используются**.

Для упрощения отладки и разработки "ядерная" часть языка *bI*₀ выдает на stdout html-заголовок, позволяет использовать в исходном тексте html-тэги, и при вычислении выражений результат также обрамляется в стиливые тэги в соответствии с типом полученного значения.

Использование на начальном этапе разработки языка такой "обрезанной" html-разметки вместо полноценного документирования в формате L^AT_EX позволяет исключить использование относительно медленного T_EX-процессора для просмотра результата: используется быстрый минибраузер, встроенный в IDE  ECLIPSE. Кроме того, этот подход оказывается удобным и для встраиваемых систем с веб-интерфейсом.

Комментарии

Тем не менее из условий запуска скриптов в UNIX требуется, чтобы первая строка имела формат `#!/bin/bI`, поэтому в язык все же вводятся лексемы комментариев:

`#!/bin/bI`

`#` строчный комментарий

`#|` блочный

комментарий `|#`

¹ [literate programming](http://en.cppreference.com/w/cpp/string/basic/basic_literate_programming)

Точечные команды

Точечные dot-команды используются для документирования и управления форматами вывода bI -системы².

.html	режим html-форматирования, для bI_0 единственный поддерживаемый режим
.tex	\LaTeX -разметка
.sec	секция документации
.sec- .sec+	изменение текущего уровня секционирования (глава, раздел,...)

² точка была взята из языка *Forth*, подразумевает конструкции языка, относящиеся к выводу

Часть II

Система типов

Глава 1

Нативные

Нативные типы реализуются на уровне bI_{VM}^1 , являются типами самого низкого уровня, и не покрываются в полной мере средствами отладки, трассировки, и контроля. При кросс-трансляции преобразуются в базовые типы целевого языка, и элементы данных с аппаратной поддержкой².

Особенно аккуратно их нужно использовать при трансляции для 8/16-битных платформ, так как низкая разрядность нативных типов и облегченный контроль со стороны bI -машины дает весь спектр побочных эффектов: переполнения, знаковые ошибки, потерю точности и т.п.

¹ виртуальная машина, рантайм-движок языковой системы bI

² машинное слово, блок памяти, элемент стека мат.сопроцессора и т.п.

1.1 **c**: одиночный символ /**[c]**har/

1.2 **s**: строка /**[s]**tring/

1.3 **i**: целое /**[i]**nt/

1.4 **f**: число с плавающей точкой /**[f]**loat/

Часть III

Расширение ☾ eclipse

plugin.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?eclipse version="3.4"?>
3 <plugin>
4 <extension point="org.eclipse.ui.perspectives">
5 <extension point="org.eclipse.ui.perspectiveExtensions">
6 <extension point="org.eclipse.ui.activities">
7 <extension point="org.eclipse.ui.newWizards">
8 </plugin>
```

Манифест

META-INF/MANIFEST.MF

```
1 Manifest-Version: 1.0
2 Bundle-ManifestVersion: 2
3 Bundle-Name: bl
4 Bundle-SymbolicName: org.eclipse.bl; singleton:=true
5 Bundle-Version: 0.0.1.alpha
6 Bundle-Activator: org.eclipse.bl.Activator
7 Bundle-Vendor: Dmitry Ponyatov <dponyatov@gmail.com>
8 Require-Bundle: org.eclipse.core.runtime, org.eclipse.core.resources,
9   org.eclipse.ui, org.eclipse.ui.console
10 Bundle-RequiredExecutionEnvironment: JavaSE-1.8
11 Bundle-ActivationPolicy: lazy
```

Перспектива *bl*

org.eclipse.ui.perspectives

```
1<extension point="org.eclipse.ui.perspectives">
2    <perspective
3        class="org.eclipse.bl.Perspective"
4        id="org.eclipse.bl.perspective"
5        icon="META-INF/icon.png"
6        name="bl">
7    </perspective>
8</extension>
```

org.eclipse.bl.Perspective

```
1package org.eclipse.bl;
2
3import org.eclipse.ui.IFolderLayout;
4import org.eclipse.ui.IPageLayout;
5import org.eclipse.ui.IPerspectiveFactory;
6import org.eclipse.ui.console.IConsoleConstants;
7
8public class Perspective implements IPerspectiveFactory {
9
10    public static final String ID = "org.eclipse.bl.perspective";
11
12    @Override
13    public void createInitialLayout(IPageLayout layout) {
14        defineActions(layout);
15    }
16}
```

```
15     defineLayout(layout);
16 }
17
18 private void defineActions(IPageLayout layout) {
19     // new menu items
20     layout.addNewWizardShortcut("org.eclipse.ui.wizards.new.file");
21     layout.addNewWizardShortcut("org.eclipse.ui.wizards.new.folder");
22     layout.addNewWizardShortcut("org.eclipse.bl.new");
23     // views
24     layout.addViewShortcut(IPageLayout.ID_PROJECT_EXPLORER);
25     layout.addViewShortcut(IConsoleConstants.ID_CONSOLE_VIEW);
26 }
27
28 private void defineLayout(IPageLayout layout) {
29     String editorArea = layout.getEditorArea();
30     IFolderLayout left = layout.createFolder("left", IPageLayout.LEFT, (float) 0.2,
31     IFolderLayout bottom = layout.createFolder("bottom", IPageLayout.BOTTOM, (float) 0.8);
32     left.addView(IPageLayout.ID_PROJECT_EXPLORER);
33     bottom.addView(IConsoleConstants.ID_CONSOLE_VIEW);
34 }
35
36 }
```