Оглавление

Об этом сборнике

© Dmitry Ponyatov <dponyatov@gmail.com>

В этот сборник (блогбук) я пишу отдельные статьи и переводы, сортированные только по общей тематике, и добавляю их, когда у меняв очередной раз зачешется LaTeX.

Это сборник черновых материалов, которые мне лень компоновать в отдельные книги, и которые пишутся просто по желанию “чтобы было”. Заказчиков на подготовку учебных материалов подобного типа нет, бо̀льшая часть только на этапе освоения мной самим, просто хочется иметь некое слабоупорядоченное хранилище наработок, на которое можно дать кому-то ссылку.

Часть I

ML & функциональное программирование

Глава 1

Основы Standard ML © Michael P. Fourman

http://homepages.inf.ed.ac.uk/mfourman/teaching/mlCourse/notes/L01.pdf

1.1 Введение

ML обозначает “MetaLanguage”: МетаЯзык. У Robin Milner была идея создания языка программирования, специально адаптированного для написания приложений для обработки логических формул и доказательств. Этот язык должен быть **метаязыком** для манипуляции объектами, представляющими формулы на логическом **объектном языке**.

Первый *ML* был метаязыком вспомогательного пакета автоматических доказательств Edinburgh LCF. Оказалось что метаязык Милнера, с некоторыми дополнениями и уточнениями, стал инновационным и универсальным языком программирования общего назначения. Standard ML (SML) является наиболее близким потомком оригинала, другой — CAML, Haskell является более дальним родственником. В этой статье мы представляем язык SML, и рассмотрим, как он может быть использован для вычисления некоторых интересных результатов с очень небольшим усилием по программированию.

Для начала, вы считаете, что программа представляет собой последовательность команд, которые будут выполняться компьютером. Это неверно! Предоставление последовательности инструкций является лишь одним из способов программирования компьютера. Точнее сказать, что **программа — это текст *спецификации вычисления***. Степень, в которой этот текст можно рассматривать как последовательность инструкций, изменяется в разных языках программирования. В этих заметках мы будем писать программы на языке ML, который не является столь явно императивным, как такие языки, как Си и Паскаль, в описании мер, необходимых для выполнения требуемого вычисления. Во многих отношениях ML **проще** чем Паскаль и Си. Тем не менее, вам может потребоваться некоторое время, чтобы оценить это.

ML в первую очередь функциональный язык: большинство программ на ML лучше всего рассматривать как спецификацию **значения**, которое мы хотим вычислить, без явного описания примитивных шагов, необходимых для достижения этой цели. В частности, мы не будем описывать, и вообще беспокоиться о способе, каким значения, хранимые где-то в памяти, изменяются по мере выполнения программы. Это позволит нам сосредоточиться на **организации** данных и вычислений, не втягиваясь в детали внутренней работы самого вычислителя.

В этом программирование на ML коренным образом отличается от тех приемов, которыми вы привыкли пользоваться в привычном императивном языке. **Попытки транслировать ваши программисткие привычки на ML неплодотворны — сопротивляйтесь этому искушению!**

Мы начнем этот раздел с краткого введения в небольшой фрагмент на ML. Затем мы используем этот фрагмент, чтобы исследовать некоторые функции, которые будут полезны в дальнейшем. Наконец, мы сделаем обзор некоторых важных аспектов ML.

Крайне важно пробовать эти примеры на компьютере, когда вы читаете этот текст.[1]

---

[1] Пользовательский ввод завершается точкой с запятой ";". В большинстве систем, ";" должна завершаться нажатием [Enter]/[Return], чтобы сообщить системе, что надо послать строку в ML. Эти примеры тестировались на системе Abstract Hardware Limited's Poly/ML. В **Poly/ML** запрос ввода символ $>$ или, если ввод неполон — #.

Примечение переводчика Для целей обучения очень удобно использовать онлайн среды, не требующие установки программ, и доступные в большинстве браузеров на любых мобильных устройствах. В качестве рекомендуемых online реализаций Standrard ML можно привести следующие:

CloudML https://cloudml.blechschmidt.saarland/
описан в блогпосте B. Blechschmidt как онлайн-интерпретатор диалекта Moscow ML

TutorialsPoint SML/NJ http://www.tutorialspoint.com/execute_smlnj_online.php

Moscow ML (**offline**) http://mosml.org/ реализация Standart ML

- Сергей Романенко, Келдышевский институт прикладной математики, РАН, Москва
- Claudio Russo, Niels Kokholm, Ken Friis Larsen, Peter Sestoft
- используется движок и некоторые идеи из Caml Light © Xavier Leroy, Damien Doligez.
- порт на MacOS © Doug Currie.

Глава 2

Programming in Standard ML'97

http://homepages.inf.ed.ac.uk/stg/NOTES/

© Stephen Gilmore
Laboratory for Foundations of Computer Science
The University of Edinburgh

Глава 3

Программирование в свободном синтаксисе: FSP

3.1 Типичная структура проекта FSP: *lexical skeleton*

Скелет файловой структуры FSP-проекта = lexical skeleton = skelex

Создаем проект **prog** из командной строки (*Windows*):

```
1 mkdir prog
2 cd prog
3 touch src.src log.log ypp.ypp lpp.lpp hpp.hpp cpp.cpp Makefile bat.bat .gitignore
4 echo gvim -p src.src log.log ... Makefile bat.bat .gitignore >> bat.bat
5 bat
```

Создали каталог проекта, сгенерили набор пустых файлов (см. далее), и запуститили батник-hepler который запустит **(g)Vim**.

Для пользователей GitHub `mkdir` надо заменить на

```
git clone -o gh git@github.com:yourname/prog.git
cd prog
git gui &
...
```

src.src		исходный текст программы на вашем скриптовом языке
log.log		лог работы ядра *bI*
ypp.ypp	**flex**	парсер **??**
lpp.lpp	**bison**	лексер **??**
hpp.hpp	C_+^+	заголовочные файлы **??**
cpp.cpp	C_+^+	код ядра **??**
Makefile	**make**	зависимости между файлами и команды сборки (для утилиты **make**)
bat.bat	*Windows*	запускалка **(g)Vim** **??**
.gitignore	**git**	список масок временных и производных файлов **??**

3.1.1 Настройки (g)Vim

При использовании редактора/IDE **(g)Vim** удобно настроить сочетания клавиш и подсветку **синтаксиса вашего скриптовго языка** так, как вам удобно. Для этого нужно создать несколько файлов конфигурации .vim: по 2 файла[1] для каждого диалекта скрипт-языка[2], и привязать их к расширениям через

[1] (1) привязка расширения файла и (2) подсветка синтаксиса

[2] если вы пользуетесь сильно отличающимся синтаксисом, но скорее всего через какое-то время практики FSP у вас выработается один диалект для всех программ, соответсвующий именно вашим вкусам в синтаксисе, и в этом случае его нужно будет описать только в файлах /.vim/(ftdetect|syntax).vim

dot-файлы **(g)Vim** в вашем домашнем каталоге. Подробно конфигурирование **(g)Vim** см. 7.

filetype.vim	**(g)Vim**	привязка расширений файлов (.src .log) к настройкам **(g)Vim**
syntax.vim	**(g)Vim**	синтаксическая подсветка для скриптов
/.vimrc	*Linux*	настройки для пользователя
/vimrc	*Windows*	
/.vim/ftdetect/src.vim	*Linux*	привязка команд к расширению .src
/vimfiles/ftdetect/src.vim	*Windows*	
/.vim/syntax/src.vim	*Linux*	синтаксис к расширению .src
/vimfiles/syntax/src.vim	*Windows*	

3.1.2 Дополнительные файлы

README.md	github	описание проекта для репоитория github
logo.png	github	логотип
logo.ico	*Windows*	
rc.rc	*Windows*	описание ресурсов: логотип, иконки приложения, меню,..

Глава 4

Язык *bI*

AST: Абстрактный Символьный Тип /Sym/

Использование класса **Sym** и виртуально наследованных от него классов, позволяет реализовать на C^{+}_{+} хранение и обработку **любых** данных в виде деревьев[1]. Прежде всего этот ***символьный тип*** применяется при разборе текстовых форматов данных, и текстов программ. **Язык *bI* построен как интерпретатор AST, примерно так же как язык *Lisp* использует списки**.

```
1 // ═══════════════════════════════════════════ ABSTRACT SYMBOLIC TYPE (AST)
2 struct Sym {
```

тип (класс) и значение элемента данных

```
1 // ──────────────────────────────────────────────────────────────────────────
```

[1] в этом АСТ близок к традиционной аббревиатуре AST: Abstract Syntax Tree

```
2     string tag;                          // data type / class
3     string val;                          // symbol value
```

конструкторы (токен используется в лексере)

```
1 // ------------------------------------------------------------ constructors
2     Sym(string,string);                  // <T:V>
3     Sym(string);                         // token
```

Хранение вложенных элементов реализовано через указатели на базовый тип **Sym**. Благодаря виртуальному наследованию и использованию RTTI, этими указателями можно пользоваться для работы с любыми другими наследованными типами данных[2]

AST может хранить (и обрабатывать) вложенные элементы

```
1 // ------------------------------------------------------------ nest[]ed elements
2     vector<Sym*> nest;
3     void push(Sym*);
4     void pop();
```

параметры (и поля класса)

```
1 // ------------------------------------------------------------ par{}ameters
2     map<string,Sym*> pars;
3     void par(Sym*);                      // add parameter
```

вывод дампа объекта в текстовом формате

```
1 // ------------------------------------------------------------ dumping
```

[2] числа, списки, высокоуровневые и скомпилированные функции, элементы GUI,..

```
2     virtual string dump(int depth=0);    // dump symbol object as text
3     virtual string tagval();             // <T:V> header string
4     string tagstr();                     // <T:'V'> Str-like header string
5     string pad(int);                     // padding with tree decorators
```

Операции над *символами* выполняются через использование набора *операторов*:

вычисление объекта

```
1 // ------------------------------------------------------ compute (evaluate)
2     virtual Sym* eval();
```

операторы

```
1  // ------------------------------------------------------------ operators
2      virtual Sym* str();                 // str(A)    string representation
3      virtual Sym* eq(Sym*);              // A = B     assignment
4      virtual Sym* inher(Sym*);           // A : B     inheritance
5      virtual Sym* member(Sym*);          // A % B,C   named member (class slot)
6      virtual Sym* at(Sym*);              // A @ B     apply
7      virtual Sym* add(Sym*);             // A + B     add
8      virtual Sym* div(Sym*);             // A / B     div
9      virtual Sym* ins(Sym*);             // A += B    insert
10 };
```

Часть II

em*Linux* для встраиваемых систем

Структура встраиваемого микро*Linux*а

syslinux Загрузчик

em*Linux* поставляется в виде двух файлов:

1. ядро `$(HW)$(APP).kernel`
2. сжатый образ корневой файловой системы `$(HW)$(APP).rootfs`

Загрузчик считывает их с одного из носителей данных, который поддерживается загрузчиком[3], распаковывает в память, включив защищенный режим процессора, и передает управление ядру *Linux*.

Для работы em*Linux* не требуются какие-либо носители данных: вся (виртуальная) файловая система располагается в ОЗУ. При необходимости к любому из каталогов корневой ФС может быть *подмонтирована* любая существующая дисковая или сетевая файловая система (FAT,NTFS,Samba,NFS,..), причем можно явно запретить возможность записи на нее, защитив данные от разрушения.

Использование rootfs в ОЗУ позволяет гарантировать защиту базовой ОС и пользовательских исполняемых файлов от внезапных выключений питания и ошибочной записи на диск. Еще большую защиту даст отключение драйверов загрузочного носителя в ядре. Если отключить драйвера SATA/IDE и грузиться с USB флешки, практически невозможно испортить основную установку ОС и пользовательские файлы на чужом компьютере.

kernel Ядро *Linux* 3.13.xx

[3] IDE/SATA HDD, CDROM, USB флешка, сетевая загрузка с BOOTP-сервера по Ethernet

ulibc Базовая библиотека языка Си

busybox Ядро командной среды UNIX, базовые сетевые сервера

дополнительные библиотеки

zlib сжатие/распаковка gzip
??? библиотека помехозащищенного кодирования
png библиотека чтения/записи графического формата .png
freetype рендер векторных шрифтов (TTF)
SDL полноэкранная (игровая) графика, аудио, джойстик

кодеки аудио/видео форматов: ogg vorbis, mp3, mpeg, ffmpeg/gsm

К базовой системе добавляются пользовательские программы ***/usr/bin***
и динамические библиотеки ***/usr/lib***.

Процедура сборки

Глава 5

clock: коридорные электронные часы = контроллер умного дурдома

Глава 6

gambox: игровая приставка

Часть III

Микроконтроллеры Cortex-Mx

Часть IV

Технологии

Часть V

Сетевое обучения

Часть VI

Прочее

Глава 7

Настройка редактора/IDE (g)Vim

При использовании редактора/IDE **(g)Vim** удобно настроить сочетания клавиш и подсветку синтаксиса языков, которые вы используете так, как вам удобно.

7.1 для вашего собственного скриптового языка

Через какое-то время практики FSP у вас выработается один диалект скриптов для всех программ, соответсвующий именно вашим вкусам в синтаксисе, и в этом случае его нужно будет описать только в файлах /.vim/(ftdetect|syntax).vim, и привязать их к расширениям через dot-файлы **(g)Vim** в вашем домашнем каталоге:

filetype.vim	**(g)Vim**	привязка расширений файлов (.src .log) к настройкам **(g)Vim**
syntax.vim	**(g)Vim**	синтаксическая подсветка для скриптов
/.vimrc	*Linux*	настройки для пользователя
/vimrc	*Windows*	
/.vim/ftdetect/src.vim	*Linux*	привязка команд к расширению .src
/vimfiles/ftdetect/src.vim	*Windows*	
/.vim/syntax/src.vim	*Linux*	синтаксис к расширению .src
/vimfiles/syntax/src.vim	*Windows*	

Книги must have любому техническому специалисту

Математика, физика, химия

- Бермант **Математический анализ** [19]
- Савельев **Курс общей физики** [20]

Программирование

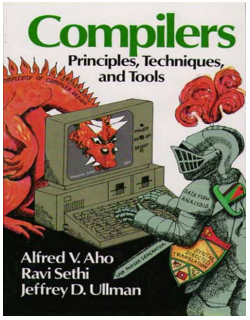
- **Система контроля версий Git и git-хостинга GitHub**
  хранение наработок с полной историей редактирования, правок, релизов для разных заказчиков или вариантов использования
- **Язык Python** [17]
  написание простых скриптов обработки данных, автоматизации, графических оболочек и т.п. утилит
- **Язык C_+^+, утилиты GNU toolchain** (gcc/g++, make, ld)
  базовый Си, ООП очень кратко, без излишеств профессионального программирования, чисто вспомогательная роль для написания вычислительных блоков и критичных к скорости/памяти секций,

использовать в связке с Python. Знание базового Си **критично при использовании микроконтроллеров**

- Использование утилит **flex/bison**
  обработка текстовых форматов данных, часто необходимая вещь

Литература

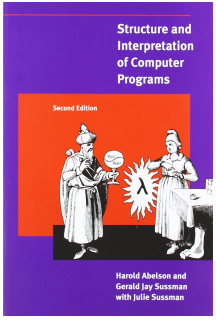
Разработка языков программирования

[1] 

Dragon Book
Компиляторы. Принципы, технологии, инструменты.
Альфред Ахо, Рави Сети, Джеффри Ульман.
Издательство Вильямс, 2003.
ISBN 5-8459-0189-8

[2] **Compilers: Principles, Techniques, and Tools**
Aho, Sethi, Ullman

Addison-Wesley, 1986.
ISBN 0-201-10088-6



[3] **SICP**
Структура и интерпретация компьютерных программ
Харольд Абельсон, Джеральд Сассман
ISBN 5-98227-191-8
EN: web.mit.edu/alexmv/6.037/sicp.pdf

[4] **Функциональное программирование**
Филд А., Харрисон П.

М.: Мир, 1993
ISBN 5-03-001870-0

[5] 

Функциональное программирование: применение и реализация
П.Хендерсон
М.: Мир, 1983

Lisp/Sheme

Haskell

ML

[6] http://homepages.inf.ed.ac.uk/mfourman/teaching/mlCourse/notes/L01.pdf
Basics of Standard ML

© Michael P. Fourman
перевод 1

[7] http://www.soc.napier.ac.uk/course-notes/sml/manual.html
A Gentle Introduction to ML
© Andrew Cumming, Computer Studies, Napier University, Edinburgh

[8] http://www.cs.cmu.edu/~rwh/smlbook/book.pdf
Programming in Standard ML
© Robert Harper, Carnegie Mellon University

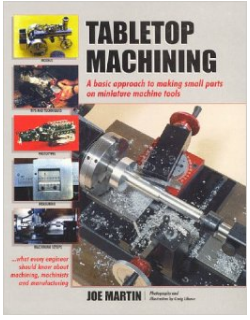
Электроника и цифровая техника

[9] **An Introduction to Practical Electronics, Microcontrollers and Software Design**
Bill Collis
2 edition, May 2014
http://www.techideas.co.nz/

Конструирование и технология

Приемы ручной обработки материалов

Механообработка



[10] **Tabletop Machining**
Martin, Joe and Libuse, Craig
Sherline Products, 2000

[11] **Home Machinists Handbook**
Briney, Doug, 2000

[12] **Маленькие станки**
Евгений Васильев
Псков, 2007
http://www.coilgun.ru/stanki/index.htm

Использование OpenSource программного обеспечения

LaTeX

[13] **Набор и вёрстка в системе LaTeX**
С.М. Львовский
3-е издание, исправленное и дополненное, 2003
http://www.mccme.ru/free-books/llang/newllang.pdf

[14] **e-Readers and LaTeX**
Alan Wetmore
https://www.tug.org/TUGboat/tb32-3/tb102wetmore.pdf

[15] **How to cite a standard (ISO, etc.) in BibLaTeX ?**
http://tex.stackexchange.com/questions/65637/

Математическое ПО: Maxima, Octave, GNUPLOT,..

[16] **Система аналитических вычислений Maxima для физиков-теоретиков**
В.А. Ильина, П.К.Силаев
http://tex.bog.msu.ru/numtask/max07.ps

САПР, электроника, проектирование печатных плат

Программирование

Python

[17] **Язык программирования Python**
Россум, Г., Дрейк, Ф.Л.Дж., Откидач, Д.С., Задка, М., Левис, М., Монтаро, С., Реймонд, Э.С., Кучлинг, А.М., Лемburг, М.-А., Йи, К.-П., Ксиллаг, Д., Петрилли, Х.Г., Варсав, Б.А., Ахлстром, Дж.К., Роскинд, Дж., Шеменор, Н., Мулендер, С.

http://rus-linux.net/MyLDP/BOOKS/python.pdf

Python является простым и, в то же время, мощным интерпретируемым объектно-ориентированным языком программирования. Он предоставляет структуры данных высокого уровня, имеет изящный синтаксис и использует динамический контроль типов, что делает его идеальным языком для быстрого написания различных приложений, работающих на большинстве распространенных платформ. Книга содержит вводное руководство, которое может служить учебником для начинающих, и справочный материал с подробным описанием грамматики языка, встроенных возможностей и возможностей, предоставляемых модулями стандартной библиотеки. Описание охватывает наиболее распространенные версии Python: от 1.5.2 до 2.0.

Разработка операционных систем и низкоуровневого ПО

[18] **OSDev Wiki**
http://wiki.osdev.org

Базовые науки

Математика

[19] 

Краткий курс математического анализа для ВТУЗов
Бермант А.Ф, Араманович И.Г.
М.: Наука, 1967
https://drive.google.com/file/d/0B0u4WeMjO894U1Y1dEJ6cncxU28/view?usp=sharing

Пятое издание известного учебника, охватывает большинство вопросов программы по высшей математике для инженерно-технических специальностей вузов, в том числе дифференциальное исчисление

функций одной переменной и его применение к исследованию функций; дифференциальное исчисление функций нескольких переменных; интегральное исчисление; двойные, тройные и криволинейные интегралы; теорию поля; дифференциальные уравнения; степенные ряды и ряды Фурье. Разобрано много примеров и задач из различных разделов механики и физики. **Отличается крайней доходчивостью и отсутвием филонианов и "легко догадаться".**

Физика

[20] Савельев И.В.

Химия

Стандарты и ГОСТы

[21] 2.701-2008 **Схемы. Виды и типы. Общие требования к выполнению**
http://rtu.samgtu.ru/sites/rtu.samgtu.ru/files/GOST_ESKD_2.701-2008.pdf