2. ФОРТРАН И ТЕКУЩАЯ РЕАЛЬНОСТЬ

Есть всего два типа программ – те, на которые всё время ругаются и те, которые не используют.
Бьёрн Страуструп

Я должен изучать Фортран? Господи, помилуй…

А нужно ли изучать Фортран? Нет, не так, задам вопрос шире – а нужно ли вообще изучать новый язык программирования? Может быть из любопытства… А на сколько хватит любопытства? Вопрос довольно серьёзный. Тем более, если вы какой-то язык уже знаете, на нём работаете, к нему привыкли и, в принципе, он вас устраивает.

Надо сказать, что создавая новый язык программирования, разработчик не удовлетворён уже существующими языками и в своём он намерен сделать какую-то фишку, которая будет лучше всех. Например, язык Си разрабатывался, чтобы сильно не париться при написании операционной системы с Ассемблером. Язык Бэйсик разрабатывался как последователь Фортрана, но с более цивилизованной и понятной структурой описания программы. Ведь у Фортрана в то время синтаксис был довольно дикий. Язык Паскаль Вирт разработал исключительно потому, что переругался со всеми в комитете Алгола по поводу высокой сложности Алгол-68. А хотелось создать ясный в понимании и простой в реализации язык. Конечно же, есть своя фишка и у Фортрана. В чём она заключается? Давайте посмотрим.

Главное достоинство Фортрана – скорость вычислений. Имеется в виду, скорость при вычислениях с большими объёмами данных. С массивами 100 – 1 000 – 10 000 и даже может быть 100 000 ячеек, вполне удовлетворительно справится программа, написанная на любом языке. Многие трансляторы, кроме общематематической, имеют ещё и специализированные вычислительные библиотеки, которые одной функцией решают какой-либо сложный вопрос, например, вычисление определённого интеграла. Даже, к примеру, такой юный транслятор, как

FreeBasic, имеет свою специализированную математическую библиотеку, которая вам на раз вычислит всякие разные интегралы, дифференциалы и прочие романтические вещи, о которых юноши тихой лунной ночью беседуют с влюблёнными в них девушками. Даже интерпретатор Python имеет математическую библиотеку `numpy` с большим количеством математических функций. Но вот если речь пойдёт о совсем больших объёмах, то Фортран здесь даст фору подавляющему большинству языков. Итак, почему именно Фортран хорошо подходит для научных и инженерных расчётов:

• Скорость работы готовой программы. Если у вас нет возможности работать на суперкомпьютерах Cray или купить отличный интеловский компилятор Си, то Фортран по скорости работы будет вне конкуренции.

• Множество встроенных функций по работе с массивами. Больше ни у кого такого нет. Кроме того, Фортран может работать с частями массивов без явного применения циклов. Вроде мелочь, а приятная, т. к. код программы сокращается. К этому надо прибавить, что поэлементные арифметические операции тоже работают с массивами опять же без явного применения циклов. Два последних обстоятельства появились в стандарте Фортран-90.

• Скорость отладки программ. Отсутствие необходимости использования указателей резко сокращает время отладки по сравнению с Си.

• Специфические типы данных. Комплексный тип данных, как встроенный тип языка программирования, есть только в Фортране. Для всех остальных языков нужно подключать дополнительную библиотеку или модуль с поддержкой как простых вычислений с комплексными числами, так и различных математических функций для комплексных чисел. К примеру, электротехника без комплексных чисел никак обойтись не может.

• Скорость разработки. В отдельных случаях, когда например, программа небольшая и её несложно «охватить одним взглядом», можно даже не заботиться о предварительном объявлении переменных. И это большое достоинство. В новых языках, например в PHP, типов данных вообще не существует в классическом понимании этого термина. Правда, в некоторых

случаях, достоинство это сомнительное, а иногда даже вредное ☺.

• Большое количество элементарных математических функций. Это часто позволяет вообще не подключать какие-либо внешние библиотеки. Встроенные функции отлично работают со всеми без исключения встроенными типами данных, а также с массивами, что не наблюдается в других языках. И это также сильно способствует сокращению времени разработки.

Сразу хочу сказать, что не буду, себя не помня от пафоса, кричать, что «Никто, кроме Фортрана!» не имеет такой быстроты. Надо честно признаться, что в последнее время язык Си (не Си++ ☺) по скорости обработки практически сидит у Фортрана на пятках. Во многих случаях даже будет затруднительно определить, а кто же у нас отработал быстрее – Си или Фортран. Правда в этом случае стоит уточнить, что речь идёт об интеловском компиляторе Си ☺.

Последние полтора-два десятка лет разработчики Си трудились над этим языком просто не покладая рук. Так, что если вы уже работаете на Си, то можно с новым языком особенно и не торопиться. Ещё раз оговорюсь, что речь идёт об отлично вылизанном и хорошо оптимизированном для процессоров Intel компиляторе Intel Си, который входит в комплект Intel Parallel Studio XE. У других компиляторов дела не столь радостны. И стоит добавить, что процессоры Intel сегодня уже не в подавляющем большинстве на рынке вычислений. Процессоры ARM идут семимильными шагами к своей славе. Один из самых мощных вычислительных кластеров в мире – китайский «Sunway Taihulight» собран именно на ARM процессорах Sunway SW26010 260C 1.45 ГГц, так что использовать там интеловский компилятор никак не получится.

Далее, в практической части книги, приведён пример вычисления СЛАУ для четырёх компиляторов: GNU Fortran (gfortran), gcc, freepascal и freebasic. Про два последние говорить не будем ☺, но у двух первых результат очень интересен. Так что если у вас нет возможности платить большие деньги за отличный компилятор Intel Си, при условии, что вы работаете именно на процессорах от Intel, то GNU Фортран будет без преувели-

чения единственной возможностью получить быстродействующую программу, если фактор скорости выполнения является очень важным.

Отмечу, что в очень сладкой сишной бочке мёда есть ложка дёгтя и, может быть, даже не одна. Сначала расскажу про очень маленькую ложку, которая даже и не столовая ложка, а так – чайная. Программа на Си будет работать наравне с фортрановской только в том случае, если программу компилировать с максимально возможной оптимизацией. Например, если речь идёт о компиляторе gcc, то даже ключа `-O3` может быть недостаточно, нужно `-Ofast`, а возможно потребуется и `-ffast-math`. Правда в последнем случае нужно убедиться, что ваша программа вычисляет правильно, а то уже бывали прецеденты, когда приходилось выбирать – либо считать быстро, либо считать правильно… ☺ Фортран, в отличие от Си, даёт быструю программу даже и без всяких ключей вообще. Если ему добавить ещё и быстрые ключи, то скорость увеличивается, хотя и не всегда так уж намного, как в gcc, всё зависит от используемого алгоритма.

Самая большая и самая невкусная ложка в сишной программе – указатели. Если вы в своей программе на Си не используете указатели, считайте, что в тёплый и весёлый испанский город Малага вы едете через Улан-Удэ с пересадкой на Сахалине. В Фортране такой проблемы никогда не стояло, потому что параметры в функцию там всегда передавались по ссылке, что сильно способствовало скорости. Т. е. тратить время на создание копии данных для использования их в функции не нужно. А кроме того, до определённого момента, там не было и самих указателей, что сильно способствовало оптимизации в плане скорости работы ☺. Но если вы подумали, что теперь, узнав где у Си скорость, попадёте прямо в нирвану, то сильно ошиблись – вместе с указателями вы попадёте в сишное чистилище, т. к. по статистике самое большое количество ошибок программисты совершают именно, забравшись в болото указателей.

А сейчас можете подумать, что вам больше нравится – получать от скорости удовольствие, ощущение азарта и радости жизни или бессонные ночи с тяжёлыми и мучительными раздумьями о бренности всего сущего ☺.

Хочу обратить внимание на тот факт, что люди, программирующие на Си, любят упрекать все без исключения другие языки программирования всякими многочисленными и, в высшей степени, справедливыми упрёками. Правда при ближайшем рассмотрении их упрёки оказываются, все до одного взяты из 60...70-ых годов прошлого века. Конечно же, не обошли они своими упрёками и Фортран. Давайте разберём список традиционных фортраньих упрёков:

«Слишком много `GOTO`». Этот упрёк из 60-ых годов, когда структура программы копировалась с Ассемблера. Правда и Си тогда ещё не было, но это же неважно. Важно то, что Си в 60-ых совершенно не за что было упрекнуть ☺. После введения стандарта Фортран-77 количества `GOTO` заметно поубавилось. В Фортране-90 `GOTO` можно бы и вообще не употреблять. Но остались некоторые специфичные случаи, вроде «много вложенных циклов» из которых не-GOTO'шным методом быстро не выбраться. А ещё есть интересный момент: к примеру, реализации MPI пишут одновременно и на Си, и на Фортране. Так вот, самые последние версии MPI в сишном коде прямо так и пестрят `GOTO`. Ну, просто классика жанра – соринка и бревно ☺...

«Запутанные, непонятные тексты программ». Если говорить о Фортране, то сия претензия тесно связана с предыдущим пунктом о `GOTO`. Действительно, если тебя каждые две-три строчки куда-то посылают, разобраться будет сложно. Но можно ведь и не писать программы в стиле легендарных 60-ых, не так ли? Если используем формат записи, который ввели в стандарте Фортран-90, собственно непонятки и не появятся. А вот про Си этого не скажешь. Например, при задании циклов с заранее известным количеством повторений: что будем использовать «`for(i=0;i<MyMAX;i++)`» или «`for(i=0;i<MyMAX;++i)`», а, ребятки? ☺ В отличие от организации цикла в Фортране – это уже не вопрос пары минут. Я ведь даже не беру набившие оскомину сишные составные конструкции с плюсами-минусами-присваиваниями, в которых с первого взгляда и не распознать, что они там делают.

«Программист в Фортране может не объявлять тип переменных». Полная жесть, как же мы теперь без типов то проживём? ☺ Однако, если внимательно присмотреться – эта претензия вовсе не к языку, а к тому, кто на нём программирует. Фортран – язык типизированный, а вот то, что в программе можно не объявлять типы, когда-то было не недостатком, а вовсе даже и достоинством. Например, если надо по-быстрому составить программку для какого-то расчёта, можно много времени сэкономить на объявлении типов, положась на благоразумие Фортрана. Правда сиё благоразумие строилось на таких принципах: если название переменной начинается на одну из букв «**I**», «**J**», «**K**», «**L**», «**M**» или «**N**» – это будет целочисленная переменная, а если начинается с любой другой буквы – вещественная. Но это только для случая, если совсем не объявлять переменные заранее. Если же вы их объявляете, они будут именно того типа, которым его и назначили, вне зависимости от названия переменной. В Фортране-90 ввели явный контроль типов со стороны компилятора – команда `IMPLICIT NONE` строго следит, чтобы в программе не было необъявленных заранее переменных.

Есть и более «сегодняшние» претензии. Сишников мы больше слушать не будем. По-ихнему, если на каком-то языке нельзя написать драйвер, то это и не язык вовсе. А вот из серьёзных – отсутствие наличия (или наличие отсутствия ☺) встроенных средств работы с графикой и всевозможными пользовательскими графическими штучками-дрючками типа «кнопка» или «окошко». Тут надо сказать, что уже не первое десятилетие идёт совмещение фортрановских программ с сишными библиотеками. Или, если быть ближе к реальности – возможности безболезненного вызова сишных библиотечных функций из фортрановских программ. Таким образом, стало возможно писать модули-интерфейсы к различным компонентам типа «кнопка», «окошко» и т. п. Для GTK (сокращ. от GIMP ToolKit – библиотека графических элементов для редактора GIMP, которую позже стали использовать для графической оболочки в ОС Linux) есть готовые модули, которые можно взять на сайте github.com. А для взаимодействия Фортран и WinAPI даже целая книжка написана по этому поводу [27]. Если нужна безумно красивая графика – есть модуль взаимодействия с OpenGL. Так что для

любителей «окошек» – вперёд и с песней… Точно таким же образом мы можем взаимодействовать из своей программы с различными базами данных.

Ещё один весьма интересный вопрос – а где мы можем повстречаться с Фортраном в реальной жизни? Возможны весьма специфические случаи, когда студент, поступивший в высшее учебное заведение, может столкнуться с преподавателем программирования, старым и свирепым зубром, который ещё помнит фон Неймана, Тьюринга и Мао Цзэдуна. И который, исключительно веселья ради, решит обучать студентов программированию с помощью Фортрана, несмотря на стоны и вопли о пощаде этих несчастных замученных чипсами с кока-колой кроликов. С Фортраном вы обязательно встретитесь при работе на вычислительном кластере. Фактически на кластере у вас будет довольно богатый выбор из двух языков – Си или Фортран. И совсем не факт, что существующие расчётные программы для решения той или иной проблемы будут именно на Си.

Вы можете сказать – да ну, разве мы попадём ещё при этой жизни на работу со столь могучим инструментом как вычислительный кластер? Когда-то и я так думал. Но вот как-то раз я решил осчастливить российскую науку идеей о весьма дешёвом, но очень прыгучем, как блоха на барбоске, кластере. Естественно стал вентилировать вопрос у доступной мне учёной братии, а как там у нас обстоят дела с большими и серьёзными кластерами. Дела оказались печальны. И вовсе не потому, что кластеров нет или на них очередь желающих серьёзно вычислять до нового 2125 года, а потому что они работой не сильно то и загружены. Так что если у вас есть что посчитать, то кластер вполне будет доступен.

2.1 Чем можно пользоваться для программирования на Фортране

Дайте женщине пару хороших
туфель и она покорит мир!
Мэрилин Монро

Несмотря на давнишность Фортрана, до сих пор существует довольно много трансляторов для этого языка. Их список можно узнать из книжки [17]. Сразу предупреждаю – разделу «Books» не сильно верьте, какая-то там информация больно кусая, а вот всё остальное смотреть можно. Раздел «Compilers» актуален.

На сегодняшний день самые популярные трансляторы – фирмы Intel (проприетарный) и GNU Compiler Collection (сокращ. gcc, свободный). Первый – платный, наиболее оптимизированный, простой в использовании, поставляется в составе мощной интегрированной оболочки для разработки программ «Intel Parallel Studio XE» [18] и вместе с многочисленными подсобными инструментами. И, что самое интересное, с отличной математической библиотекой, которая раньше называлась IMSL, а теперь IMKL (Intel Math Kernel Library) [19]. Её, по уверениям Intel, можно скачивать и пользоваться совершенно бесплатно, только потребуется регистрация. Parallel Studio могут бесплатно пользоваться научные работники, преподаватели, студенты и разработчики программ с открытым исходным кодом. Естественно, прежде чем получить бесплатную лицензию, вы должны доказать, что ими и являетесь. Для трёх первых категорий, начальная стадия – это иметь почтовый ящик на почтовом сервере, который принадлежит научной организации или университету. Что там у них дальше с проверкой, я не выяснял. Для разработчика OpenSource необходимо иметь какой-то проект на сайте типа github.com или sourceforge.org, и они могут получить среду разработки только для операционной системы Linux.

У Parallel Studio есть один недостаток, который может полностью перечеркнуть все неоспоримые достоинства – эта штука работает исключительно на процессорах Intel. Собственно, другого от этой фирмы ожидать и не в праве. У меня, к примеру, рабочая станция на процессоре ARM, так что мой удел – ходить мимо с печальным видов и издавать громкие унылые вздохи, утирая со щеки скупую мужскую слезу…

Второй популярный транслятор – GNU Fortran или попросту gfortran [20] – вещь совершенно бесплатная и доступная каждому, причём для множества процессоров. Правда о совсем уж хорошей оптимизации по скорости тут говорить не приходится – слишком уж много точек приложения сил в отличие от Intel. Но, тем не менее, работает он очень хорошо и программы, сделанные с его помощью, тоже работают хорошо. У него очень много ключей компиляции, в которых разберёшься не сразу. В принципе, он делает быстрые программы и вообще без всяких ключей, но с ключами будет ещё быстрее.

Основные достоинства gfortran'а – конечно же, бесплатность, множество поддерживаемых процессоров и операционных систем. Однако надо сказать, что системы типа BSD (в частности FreeBSD, OpenBSD и NetBSD) от компиляторов GNU почему-то отказались в пользу LLVM с его Си-компилятором clang. LLVM, к сожалению, официального транслятора для Фортрана пока не сделал. Правда на github'е лежит проект фортрановского фронт-энда к LLVM, который называется flang. Работает этот компилятор только под UNIX. В самых новейших версиях дистрибутивов Linux он уже есть в готовом виде, так что можно им пользоваться. Официальным же фронт-эндом для языка Фортран в LLVM должен стать F18. Предварительно его включение в состав дистрибутива уже одобрено. Но к сожалению в готовом виде в дистрибутиве LLVM его пока нет. Желающие могут собрать F18 самостоятельно из исходников.

Небольшое лирическое отступление про LLVM

Несмотря на то, что язык Си мне не нравится, регулярно приходилось сталкиваться с ситуациями, когда кроме как Си использовать ничего было нельзя. И кроме чисто эстетических

причин, в языке Си мне не нравились его сообщения об ошибках. Порою приходилось тратить в 10 раз больше времени на то, чтобы понять, о чем говорит сообщение, чем на устранение ошибки.

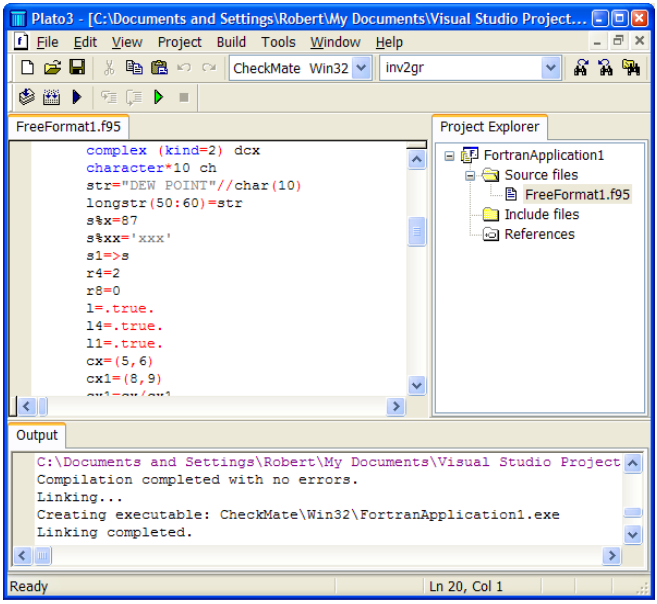
Конечно же, я начал искать, чем можно заменить GNU Си. Нашёл. Это оказался компилятор clang, который делался на основе LLVM – Low Level Virtual Machine или по-русски – низкоуровневая виртуальная машина. Изначально этот проект планировался как виртуальная RISC-машина, т. е. эмулятор RISC-процессоров. Но позже это вылилось в систему, переводящую некие условные обозначения в байт-код, как Java. Только этот байт-код эмулировал простые RISC-команды. и на этой основе создающий исполняемый код для определённого процессора.

В clang мне очень понравилась система сообщений об ошибках, которая была очень подробна и, что самое главное, понятна с первого взгляда. Кроме того, она давала конкретное место ошибки, а не просто номер строки, как в старых версиях GNU Си. Однако недолго длилась эта монополия на понятливость. Начиная с версии 5, GNU Си позаимствовал в LLVM систему сообщений, так что с пятой версии GNU Си сообщения тоже стало легко понимать ☺.

В любом дистрибутиве Linux (кроме, пожалуй, Calculate Linux) есть и gfortran, который устанавливается без малейших проблем. Для Windows есть проекты Cygwin [21] и Mingw [22], в которые входит весь набор GNU-компиляторов, в том числе и gfortran.

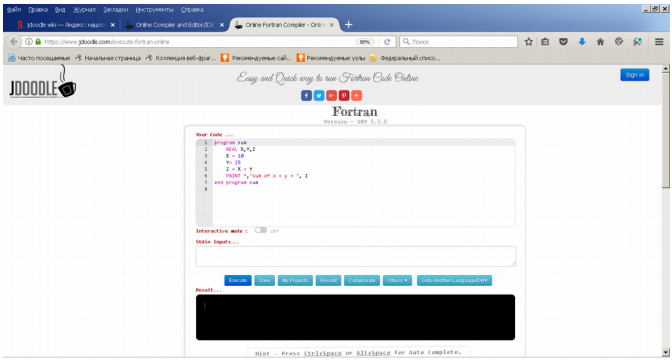
Gfortran ориентирован исключительно на командную строку. Если вам непременно нужна какая-нибудь IDE (сокращ. Integrated Development Environment – Интегрированная Среда Разработки), то это будет уже чья-то сторонняя разработка, в которой использование gfortran'а нужно специально указывать. Например, это может быть программистский редактор GEANY или что-то более тяжеловесное, типа `CODE::BLOCKS`.

Из бесплатных Фортранов есть ещё парочка интересных проектов. Например, FTN95 [23]. Работает только под Windows, версия Personal Edition бесплатная. Бонусом к нему прилагается IDE под названием Plato. Этот Фортран поддерживает стандарт Фортран-95 и для тех, кто любит работать исключительно в IDE'шках под Windows, самая идеальная штука, настроек никаких делать не надо. Конечно, там нет современных прибамбасов, типа `COARRAY` или ООП. Но, согласитесь, это не такая уж и необходимая штука в научных вычислениях, особенно если вы работаете только для себя любимого.



IDE для Фортрана FTN95

Ещё один проект несколько необычен тем, что вам можно не иметь у себя на компьютере вообще никакого транслятора и ваши файлы-программы будут доступны в любой точке земного шара, если там, конечно, есть интернет. Fortran-онлайн [24] –



Фортран-онлайн

это часть очень большого проекта JDoodle [25] по всевозможным языкам программирования, который последнее время всё чаще используется для различных курсов по программирова-

нию. Если вы просто из любопытства хотите посмотреть, что такое Фортран, то можно не заморачиваться с установками компиляторов, а просто поработать на этом сайте и посмотреть – если понравиться и захотите двигаться дальше, то можно тогда и озаботиться собственным компилятором.

Отдельно хотелось бы отметить ещё один компилятор Фортрана – g95 [47]. Правда на сайте разработчика уже давно царит затишье, тем не менее, это очень интересный продукт. Для программ, откомпилированных с его помощью и использующих `COARRAY` для распараллеливания вычислений, не обязательны какие-либо сторонние библиотеки (типа `MPI`). `COARRAY` реализован прямо в компиляторе. Для компиляции такой программы не нужны какие-либо дополнительные ключи. Для запуска в работу так же не требуется использование каких-либо менеджеров, типа `mpirun`. Запуск программы будет выглядеть так:

```
./ВашаПрограмма --g95 images=8
```

т. е. запустится 8 процессов одной и той же программы «`ВашаПрограмма`».

Такая возможность без библиотечного использования `COARRAY` поддерживается только в самой последней версии 0.94, которая вышла в январе 2013 года.

Небольшое лирическое отступление – сенсационное разоблачение gfortran ☺

G95 не канул в Лету по причине, которая случается не так уж и редко в современном мире – проект GNU переманил к себе разработчиков g95. Дело в том, что GNU уже довольно давно спонсируется софтверными гигантами, типа Intel, Oracle или Microsoft, поэтому разработчики там живут довольно шикарно, хотя и не жируют. Тем не менее, деньги у них водятся очень немаленькие. Так что ничего удивительного в том нет, когда разработчики g95 (правда за исключением своего предводителя – Энди Вота, который остался заниматься своим компилятором) переходят с чисто «для своего удовольствия» программирования на работу хоть и дополнительную, но оплачиваемую. Тем более что Windows там писать с них не требуют. Собственно, ничего кардинально, по большому счёту, не меняется, однако в кармане уже не так пусто и тоскливо, как было раньше ☺*…*

Так вот, если вы думаете, что GNU Фортран до сих пор тот же самый, что был с самого начала, то вы сильно ошибаетесь. В конце 90-ых годов попытка разработчиков GNU спроектировать и создать

современный Фортран (как минимум стандарта Фортран-90), похоже, не увенчалась успехом. До версии 4.0 у проекта GNU был очень неплохой Фортран-77, но вот с новым Фортраном что-то не сложилось.

Сильно не переживая по этому поводу примерно в 2003 году решили взять кодовую базу g95, благо что она была полностью открыта, и сделать то, что мы теперь знаем под именем gfortran. Затея удалась, но у g95 появился конкурент. И как только у GNU замаячили большие бабки, команда разработчиков g95 немедленно дунула в GNU. Вот такие вот пироги ☺…

2.2 Чем пользуется автор (GNU Fortran)

Великие дела нужно совершать,
а не обдумывать их бесконечно.
Гай Юлий Цезарь

Возможно вы и сами уже догадались, я пользуюсь тем, что легче всего достать – компилятором gfortran. Дело в том, что я использую две операционные системы – на работе стоит Windows (на большинстве компьютеров), а дома – Linux. На работе, на серверах, тоже стоит Linux, и я коварно пользуюсь (изредка ☺) этими серверами для распределённых вычислений. А кроме этих больших и серьёзных задач, у меня есть ещё совсем малюсенький проектик для процессора ARM. Поэтому в качестве рабочей среды программирования желательно видеть одно и то же везде (естественно, не платя денег ☺). Этому желанию удовлетворяет один только gfortran.

Небольшое лирическое отступление о том, как я познакомился с Фортраном

Надо сказать, что моё знакомство с Фортраном началось вовсе не с самого Фортрана. Когда я учился в магистратуре, у нас был предмет под названием «Вычислительные системы». И речь в нём шла не о железяках или системах программирования, как можно было бы подумать, а о вполне конкретной и довольно узкой теме – параллельных вычислениях.

На первой же лабораторной работе, я полез в интернет, чтобы узнать, что думает по этому вопросу мировая научная общественность. И первое, на что я наткнулся – код программы именно на Фортране. Правда преподаватель от нас хотел получить текст на Си, так что я заодно нашёл и пару программ на Си. Сравнивая оба варианта,

оказалось, что на Фортране программы как-то понятнее. Решившись, я понёс показывать своё произведение искусства именно в фортрановском варианте.

Рассматривая мой код, преподаватель как-то странно на меня поглядывал. Потом спросил с неким сомнением в голосе – а, в каком году, сей не в меру оригинальный магистрант, заканчивал институт. Я, не моргнув глазом, браво подкрутив воображаемый ус, с гусарской лихостью ответил – в прошлом, Ваша светлость. У меня создалось впечатление, что он мне ни капли не поверил и даже сказал – ну-ну… Только позже я понял, что он, мягко говоря, не совсем понял мой код на Фортране. По всей видимости, учась в институте, он как раз Фортран изучал, но это был старый Фортран, а мой код был уже на новом Фортране. В общем, как бы там ни было, а свою пятёрку я получил и от этого сразу полюбил Фортран всей душой ☺.

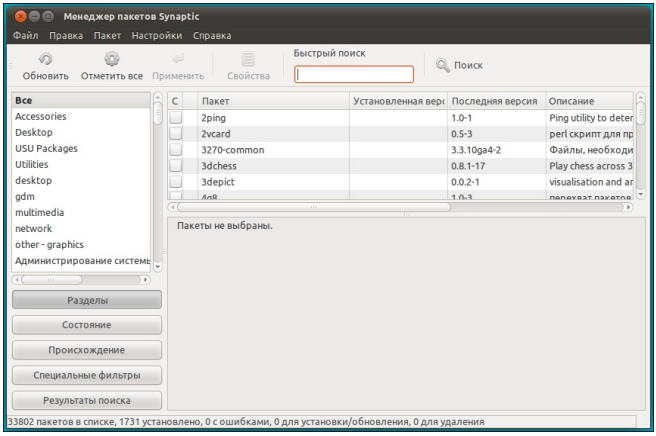
2.3 Установка GNU Fortran (gfortran) в дистрибутивах Linux

Верховным судьей всякой физической теории
является опыт. Без экспериментаторов
теоретики скисают.
Лев Ландау

В UNIX-системах установка gfortran самая простая – с помощью репозитория той ОС, которой вы пользуетесь. Обычно установка программ производится путём вызова графической оболочки для установщика и поиска в нём необходимой программы. Правда в последних версиях Ubuntu в качестве графического установщика используется нечто невразумительное, которое для обычного пользователя вполне годится, но вот мне отчего-то не нравится и главное – я там почему-то вообще не могу найти какой-нибудь компилятор ☺. Видимо современные разработчики Ubuntu решили, что все программы, которые нужны современному пользователю компьютера, уже написаны. И ничем, кроме браузера, офиса и соцсетей пользоваться уже не нужно… В общем – шуты гороховые ☺.

В Ubuntu я предпочитаю дополнительно установить себе программу – менеджер пакетов Synaptic, которая на более ран-

них Ubunt'ах стояла, и которая обычно есть в других операционках. Там искать программы удобнее.



Графическая оболочка для установки программ в Ubuntu и AltLinux

Так же вполне удобно пользоваться консольными установщиками, типа apt, yum, urpmi, где кроме самого установщика надо только прописать слово `install` и название пакета `gfortran`. Впрочем, я уверен, что вы и без меня прекрасно знаете, как ими пользоваться. В любом случае, указав только имя gfortran, установщик подтянет все необходимые для него зависимости.

Итак, если вы работаете в Ubuntu, то можно установить gfortran в консоли, введя команду:

```
sudo apt install gfortran
```

Sudo спросит ваш собственный пароль (ведь вы наверняка при установке указали администратором себя любимого ☺), после введения которого, установщик apt потратит некоторое время на поиск Фортрана в репозитории, найдёт его и спросит, то ли он нашёл. Если найденное, то что нужно, введёте букву «**Y**» и установится Фортран и необходимые ему зависимости.

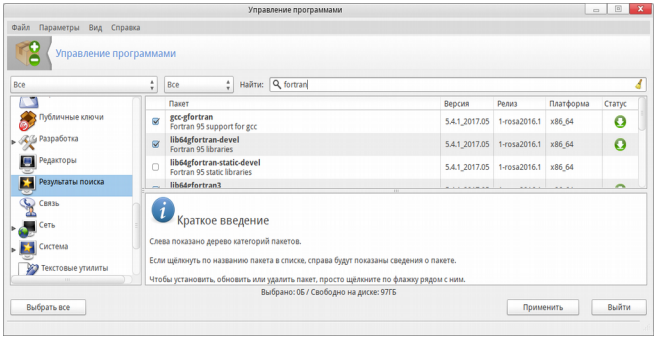
Если вы непременно хотите воспользоваться графической оболочкой, то нужно сначала установить ту оболочку, которая найдёт вам Фортран, под названием Synaptic. В разных ОС внешний вид может незначительно отличаться, например, вме-

сто поля «Быстрый поиск» присутствовать только кнопка «Поиск», нажатие на которую открывает окошко для поиска.

В поле поиска нужно ввести «fortran» и нажать кнопку «Поиск», после чего в окошке ниже высветятся найденные по этому слову файлы пакетов. Обычно, пакет с Фортраном, оказывается не в первых рядах найденного и вообще, список может быть излишне велик, поэтому придётся поработать линейкой прокрутки.

Пометив галочкой нужный пакет, нажимаете на кнопку «Применить» и некоторое время наблюдаете за процессом установки.

У дистрибутивов типа Mandriva вид графической оболочки немного другой, но принцип работы точно такой же.



Графическая оболочка для установки программ в ОС Mandriva или Mageia

Мне кажется, что для такой простой вещи, как установка Фортрана проще всего воспользоваться консольным установщиком. Как это сделать в Ubuntu написано выше. Для ОС типа RedHat (CentOS и подобные) нужно написать:

```
sudo yum install gcc-gfortran
```

Именно так – `gcc-gfortran`, но установится только gfortran, просто пакет так интересно назвали.

В ОС Fedora пока ещё работает установщик yum, но в новых ОС лучше воспользоваться новым установщиком:

```
sudo dnf install gcc-gfortran
```

Если вы захотите использовать компилятор g95, то его можно просто скачать с сайта разработчика [47], распаковать и сразу же начать использовать. В установке он не нуждается.

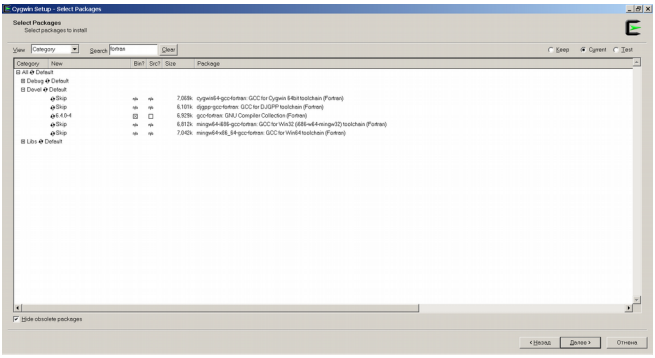
2.4 Установка GNU Fortran (gfortran) в Windows

Можно было бы сказать, что это самая любимая операционная система в мире, но повторять американские враки что-то не хочется ☺. Самая распространённая – так будет правильно. Фирма Microsoft, в отличие от Apple или IBM, довольно долго не обращала ни малейшего внимания на массовые нелицензионные установки своей ОС. И что мы теперь видим? И это отнюдь не равнодушие к своим потенциальным доходам, а грамотно рассчитанная умными маркетологами рекламная политика. Создать у людей зависимость, а потом спокойно на этой основе выгребать деньги из карманов тех наивных фраеров, которые соблазнились, как папуасы, на блестящие стекляшки ☺.

На ноябрь 2017 года общий объём всяких Windows составляет 38,11 %. Для сравнения: Android – 37,91 % ☺ [46]. Но давайте перейдём к более интересной теме – установке GNU Фортран.

Для Windows, если вы не купили Intel Фортран, ситуация с установкой маленько посложнее, чем в Linux. Осложнение вызывает более широкий выбор источников gfortran'а. Но мы не будем долго раздумывать над этим изощрённым и крайне философским вопросом, а возьмём дистрибутив CYGWIN, где GNU Фортран (gfortran) сравнительно свежий – версии 6.4. Правда и в MINGW версия не сильно старая – 6.3 ☺. CYGWIN я выбрал потому, что у него имеются намного больше сайтов-зеркал, с которых скачиваются необходимые пакеты и в случае, если какой-то сайт работает по каким-то причинам медленно, можно всегда выбрать другой. У MINGW источник всего один – сайт

sourceforge.net и выбрать более быстрое зеркало sourceforge в



Установка gfortran из дистрибутива CYGWIN

установщике нет возможности.

Для установки CYGWIN необходимо запустить его программу setup_x86.exe, если мы хотим установить 32-ух битный компилятор. Если же 64-ёх битный, то setup_x86_64.exe. Скачивается он с главной страницы сайта [21]. Установщик задаст несколько вопросов, на которые надо ответить согласием, проверит соединение с интернет (скачивание пакетов будет происходить с какого-либо сайта-зеркала CYGWIN), после чего выдаст окошко для выбора устанавливаемых программ. В этом окошке в поле «View» надо выбрать «Full», потому, что по умолчанию там стоит пункт обновления программ, но у нас-то пока ещё ничего не установлено. А в поле «Search» нужно написать «fortran», т. к. выбор устанавливаемых программ там богат, иначе в полном списке искать Фортран вы будете до утра следующего понедельника. После того, как в окошке ниже появится строка, которая относится к Фортрану, в ней, в колонке «New» (где по умолчанию стоит слово «Skip») нужно щёлкнуть мышкой и там появится текущая версия этой программы. Потом нажимаем на кнопку внизу «Далее», соглашаемся с предложенным списком устанавливаемых зависимостей и ждём, когда перестанет бегать прогресс-бар и в окошке появится сообщение, что программа установлена.

После окончания установки, желательно путь, куда вы поставили Фортран, внести в системную переменную «`PATH`» и

можно будет компилировать тексты своих программ без указания полного пути к компилятору.

2.5 Установка GNU Fortran (gfortran) в BSD-системах

Любой может спроектировать быстрый процессор.
Фокус в том, чтобы спроектировать быструю систему.
Сэймур Крэй

Тут могут появиться возмущённые реплики, типа: «Да разве ж можно программировать в ОС BSD на Фортране, ведь они все без исключения серверные?» Вопрос резонный, но нелогичный ☺. Ведь для BSD кто-то писал до этого момента программы и ничего, как-то же они там работают. И чаще всего работают хорошо. Если посмотреть с другой стороны, то все прекрасно знают, что BSD-системы очень трепетно относятся к ресурсам компьютера, поэтому для работы прикладных программ в них остаётся больше памяти, больше процессорного времени, чем в ОС типа Linux, не говоря уж о Windows.

В качестве дополнительной мотивации к использованию BSD стоит упомянуть, что количество программных пакетов в репозитории FreeBSD (у них это называется «Порты» – «Ports») сегодня составляет 31 000 штук. Правда не все они откомпилированы и доступны в списке бинарных пакетов. Но ведь никто не мешает откомпилировать исходники, тем более что в FreeBSD, в отличие от Linux, для этого вовсе не надо заботиться о поиске зависимых библиотек или программ. Всё будет найдено автоматически. Несколько большим количеством пакетов могут похвастаться только Ubuntu и Debian.

Небольшое лирическое отступление о том, как я познакомился с BSD

Впервые с этой ОС, не теоретически, а практически, я столкнулся, устраиваясь на свою нынешнюю работу. Интернет-шлюз работал на ОС FreeBSD 5.1. При приёме меня спросили, работал ли я когда-нибудь с Unix. Уверенный, что речь идёт о Linux'ах, с которыми я действительно работал, ответил «Да». Оказалось, что я ошибся. Впрочем, больших

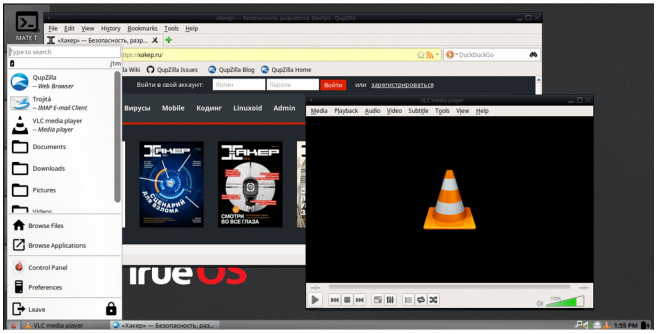
затруднений не возникло, т. к. практически все консольные программы были с такими же названиями, что и в Linux, правда, кроме файервола. Впрочем, небольшая практика помогла наверстать упущенное.

Чем мне больше всего понравилась эта система – временем работы без перезагрузки. Самый большой период продлился аж 8 месяцев. В течение этого времени надолго электричество не отключали. С серверами на основе Windows такой длительности работы достигнуть ни разу не удавалось ☺.

Начнём с самой распространённой системы – FreeBSD. Сюда же можно отнести и некоторые её клоны, например, DragonFly BSD или TrueOS. Последняя, кстати говоря, существует в двух вариантах – для сервера и для рабочей станции. Версия для рабочей станции снабжается по умолчанию графической оболочкой, так что на рабочую станцию она вполне похожа, хотя русификация там пока что неполная.

Чем привлекательна именно FreeBSD – у неё в пакетах есть самая последняя версия GNU Фортран. Мало того, даже можно установить и версию, которая находится в разработке. Впрочем, последнее – уже для профи в этом деле.

Однако при установке может возникнуть лёгкий казус. Де-



Рабочая станция на основе FreeBSD – TrueOS

ло в том, что отдельного пакета с Фортраном здесь нет. Попытка поиска закончится тем, что пакетный менеджер выдаст вам несколько библиотек, в имени которых встречается слово «fortran». Чтобы установить именно компилятор Фортран, необходимо поставить «малый джентльменский набор» GCC. Пакет называется «gccX», где X - это номер версии. Туда входят компиляторы Си, Си++ и Фортран.

```
pkg install gcc7
rehash
```

В данном случае мы устанавливаем GCC версии 7. Здесь вторая команда «rehash» нужна только в том случае, если вы захотите опробовать Фортран сразу же, без повторного входа в систему. Хотя в современных версиях FreeBSD (наверное начиная с 11 версии) «rehash» уже не сильно нужен, но на всякий случай...

Второй казус ожидает вас, как только вы захотите откомпилировать программу. Опять покажется, что «gfortran» начисто отсутствует в системе, несмотря на то, что вы буквально только что его установили ☺.Сильно не пугайтесь, просто здесь этот файл называется «gfortran7», если вы устанавливали именно седьмую версию. Видимо, это сделано специально на тот случай, если вы захотите одновременно иметь несколько версий «GCC». FreeBSD это благосклонно позволяет. Если вам вполне достаточно только одной версии, и вы категорически не желаете писать «gfortran7», то в каталоге «/usr/local/bin/» можно создать ссылку с именем «gfortran» на файл «gfortran7» и забыть об этой проблеме:

```
ln -s /usr/local/bin/gfortran7/usr/local/bin/gfortran
```

Правда, если пакет через некоторое время обновится, то ссылку придётся создавать заново ☺.

Компилятор g95 [47] здесь тоже можно использовать и тоже без установки.

Небольшое лирическое отступление о DragonFly BSD

Когда-то давно один из разработчиков FreeBSD не сойдясь во мнении со своими единомышленниками о том, как надо оптимизировать ядро ОС, отделился от них и создал свой собственный проект – DragonFly. От FreeBSD он отличается системой управления процессами и наличием кое-каких дополнительных драйверов оборудования, которых до сих пор нет во FreeBSD. Содержимое пакетной базы одинаковое.

По утверждению Мэттью Диллона, одного из разработчиков, ему удалось запустить на своей машине 900 000 тестовых процессов. Конечно же, машина была не простая –

128 ГБ ОЗУ и, по косвенным данным, 2 процессора Intel Xeon. Так же он утверждает, что количество запускаемых процессов теперь ограничивается только величиной максимально допустимого PID – до 6 цифр. Так что имейте в виду ☺.

Следующая по распространённости BSD-система – OpenBSD. Она аналогична по управлению пакетами NetBSD, но вот с пакетной базой у неё дела похуже и в сравнении с FreeBSD, и в сравнении с NetBSD. Здесь мы столкнёмся с той же ситуацией, что и во FreeBSD – будем искать в списках пакетов «fortran» по названию и не найдём ничего. Потому что здесь два отдельных и разных Фортрана:

• g77 – это Фортран-77. Поддерживает только старый синтаксис и подойдёт тем, кто непременно хочет испытать ностальгию по легендарным временам ☺.

• g95 – несмотря на название, это именно «gfortran» от GNU, только стандарта Фортран-95, т. е. в нём нет ни ООП, ни `COARRAY`. Но в остальном это нормальный современный Фортран и я рекомендую устанавливать именно его.

Перед установкой пакета, нужно чтобы в специальной системной переменной «`PKG_PATH`» был прописан правильный путь к ftp-каталогу пакетов для вашей версии OpenBSD. Команда установки будет такой:

```
pkg_add g95
```

После установки Фортран будет называться без всяких выкрутасов – «gfortran», поэтому компиляция программ проходит самым обычным образом:

```
gfortran ВашаПрограмма.f90
```

Самая малоизвестная BSD система – NetBSD. Однако если взглянуть в её репозиторий с Фортранами, то это настоящий праздник души и сердца ☺:

• gcc3-f77 – это, как и в OpenBSD, компилятор старого Фортран-77. Работает, судя по названию, совместно с библиотеками gcc3.

• g95 – это самый настоящий g95, вот только версии 0.93, поэтому с `COARRAY` работать не будет. В принципе с сайта [47] можно скачать исходники последнего компилятора и скомпилировать g95 самостоятельно. Судя по присутствию бинарного пакета, проблем с этим не должно быть.

• gcc7 – так же как и во FreeBSD это «малый джентльменский набор» компиляторов GNU – gcc, g++ и gfortran. Компиляторы версии 7.1. Правда на сайте описания пакетов почему-то написано, что они должны быть версии 7.2 и «большой джентльменский набор». Не исключено, что это описание для следующей версии NetBSD – 8.0, т. к. на ftp-сервере проекта выложена уже восьмая версия.

Перед установкой надо так же, как и в OpenBSD, прописать в системную переменную «`PKG_PATH`» путь к ftp-каталогу вашей версии ОС.

Установка проходит тем же способом, что и в OpenBSD:

```
pkg_add gcc7
```

2.6 Как компилировать программы с помощью gfortran

Кто-то утверждает, что каждое уравнение,
включённое в мою книгу, вдвое снижает её продажи.
Стивен Хокинг «Краткая история времени»

Перед тем, как перейти к теории с практикой, нужна практика по использованию инструмента, который мы будем использовать (Во, завернул! ☺). Обычно я компилирую gfortran'ом вообще без каких-нибудь дополнительных ключей, поскольку он и так даёт достаточно быстрые программы:

```
gfortran НазваниеПрограммы.f90
```

На выходе получается файл `a.out` в Linux или `a.exe` в Windows. Эта особенность Linux'овых компиляторов давать такое затейливое имя откомпилированной программе. Если

нужно получить программу с вменяемым именем, то приходится использовать специальный ключ компиляции:

```
gfortran -o НазваниеПрограммы НазваниеПрограммы.f90
```

тогда у откомпилированной программы будет такое имя, которое вы указали после ключа «-o».

При написании параллельной программы на основе OpenMP нужно добавить такой ключ:

```
gfortran -fopenmp НазваниеПрограммы.f90
```

Если в фортрановской программе вы захотите использовать сишный препроцессор (мало ли для чего это будет нужно, раз уж есть такая возможность, отчего бы и не воспользоваться), то можно добавить такую опцию:

```
gfortran -cpp НазваниеПрограммы.f90
```

Если у вас получилась длинная программа и там очень много ошибок, то чтобы не сильно портить нервную систему, количество выводимых на экран ошибок за один раз можно ограничить:

```
gfortran -fmax-errors=N НазваниеПрограммы.f90
```

где N – это количество ошибок выводимых за один раз.

Если нужно получить программу совсем быструю, то можно добавить такие ключи:

```
gfortran -Ofast НазваниеПрограммы.f90
```

После компиляции программа может выйти более «толстая», зато более быстрая. Правда применение быстрого ключа для Фортрана не всегда даёт такой потрясающий прирост скорости, как для Си.

Если нужно чтобы линкер подключил какую-нибудь библиотеку, например, при работе с базой данных, то используется такая опция:

```
gfortran -lбиблиотека НазваниеПрограммы.f90
```

Компиляция модулей, т. е. функций, которые расположены в отдельных файлах и программой не являются, тоже требуют отдельного ключа. В отличие от предыдущих примеров, где создавалась полноценная самостоятельно запускаемая программа, здесь компиляция происходит только наполовину. Т. е. компиляция идёт как обычно, вот только после неё, программа-линковщик не приделывает в начало откомпилированного файла специальный модуль со служебной информацией, где определяется, к примеру, как размещается в памяти программа. На выходе мы тоже видим двоичный файл, но самостоятельного значения он не имеет, это что-то вроде библиотеки функций:

```
gfortran -c НазваниеМодуля.f90
```

При этом образуются два файла – `НазваниеМодуля.o` и `НазваниеМодуля.mod`. Первый – это объектный скомпилированный файл с двоичным кодом функций, но который не обработал линковщик и не присоединил туда атрибуты исполняемой программы. Второй – это файл с описанием интерфейсной (заголовки подпрограмм\функций, видимые типы, константы, переменные) части модуля. Файл с расширением «`.o`» нужно будет указывать при компиляции других программ, где необходимо использовать данный модуль:

```
gfortran НазваниеМодуля.o НазваниеПрограммы.f90 -o НазваниеПрограммы
```

При таком способе компиляции общее время компиляции основной программы сокращается, если вы не переделываете код модулей, поскольку модули не компилируются, а просто подставляются в нужное место.

Кроме объектных файлов при компиляции программ с использованием дополнительных модулей, в строке компиляции можно прописывать и названия файлов с исходными кодами модулей:

```
gfortran Модуль1.f90 Модуль2.f90 Программа.f90
```

Учтите, что обработка исходников проходит последовательно в том порядки, в каком они прописаны в строке компиляции. Поэтому исходник главной программы должен стоять на последнем месте, т. к. к моменту её компиляции модули уже должны существовать, иначе неоткуда будет вставлять двоичный код.

Опций компиляций довольно много, их можно посмотреть на страничке компилятора [26].

По поводу компиляции своих и чужих исходников, которых так много в интернете. По собственному опыту могу сказать, что без переделки кода можно компилировать фортрановские программы начиная с версии компилятора 4.8.2, именно такой стоит на моём нетбуке. В версиях до 5.1. нет полноценной поддержки `COARRAY`, но это отнюдь не самая необходимая штука в Фортране. А вот любой gfortran, начиная с версии 5.1, содержит все современные возможности стандартов Фортрана, по крайней мере Фортран-2008 точно.

Если вы берёте исходники из интернета, то там, в половине случаев, код программ от старого Фортрана. Отличить его от современного можно по внешнему виду – строки кода начинаются не с первой колонки, а с некоторым отступом. И там очень много `GOTO` ☺. В большинстве случаев старый код будет компилироваться без какой-либо переделки. Правда показывать его преподавателю будет слегка стыдновато ☺.

В конце следующего раздела, после кратких теоретических сведений о синтаксисе современного Фортрана, мы немного поговорим о том, как старому коду придать более благообразный «осовремененный» вид, какие устаревшие конструкции на какие новые можно и нужно заменить.

Все современные версии Linux содержат пакеты с современным Фортраном. В CYGWIN и MINGW для Windows Фортран тоже современный, там только различаются версии gfortran, например 6.4 и 6.3. Так что все без исключения возможности Фортрана можно использовать в любой современной ОС.

Небольшое лирическое отступление по поводу моего нетбука

Давным-давно, ещё до американских санкций, я соблазнился махоньким, лёгоньким и, что самое главное, сравнительно дешёвеньким нетбуком. В нём проживал процессор Intel Atom N2800, который имел 4 вычислительных ядра, что тогда мне показалось явным достоинством. Увы, через некоторое время выяснилось, что по поводу «достоинств» смотрел я совсем не туда.

Нет, сначала всё шло вполне благополучно – поставил ОС Ubuntu 14.04, которая вполне исправно заработала. Свою ошибку с выбором процессора я понял только, когда пришло время обновлять ОС до следующей версии. Встроенная в процессор видеокарта перестала работать. Вообще перестала, даже подсветка экрана отсутствовала. И вообще, любая одна другая Linux-ОС на этом процессоре с ядром новее, чем 3.13 не хотела работать. Мало того, CentOS, у которой ядро, как и у RedHat версии 3.10, тоже какое-либо видео показывать отказалась.

Проблема, конечно же, крылась в интеловских дровах, так как VESA-драйвер работал вполне благополучно. Проблема с VESA была в разрешении экрана – у моего нетбука экран 1024x600, а у VESA можно было поставить только либо 800x600, либо 1024x768. В первом случае можно было ностальгировать по началу 90-ых прошлого века, но никак не работать, во втором – не видеть полностью своего экрана.

Что интересно, эта проблема только у N2800, другие Atom'ы с интеловскими дровами работают вполне благополучно. Мало того, несмотря на то, что N2800 64-ёх битный процессор, на сайте Intel 64-ёх битного видеодрайвера (под Windows, естественно) долгое время не было. Вот вам, бабушка и Юрьев день. Не процессор, а выродок какой-то. Так что имейте в виду…