

Скрэтчер#00



Online журнал для скрэтчеров — людей, чье хобби создавать вещи и технологии по следам уже существующих, в сотый раз изобретать велосипед, чтобы разобраться как оно работает, научиться делать самому, а возможно найти новый или забытый способ что-то сделать, и конечно получить удовольствие от процесса поиска.

Кустарь-одиночка с мотором



Редакция: <dponyatov@gmail.com>



Powered by L^AT_EX

Содержание

1	<i>Об этом журнале</i>	
	Сделай сам, расскажи другим	3
2	<i>КопиПаста</i>	
	Как сказать "Начать с нуля" ?	3
3	<i>КопиПаста</i>	
	Персональное производство еще один шаг к реконизму	4
4	<i>Новости технологий</i>	
	Злобный янки в 3D-танке	
	Полевые 3D-принтеры на службе американской армии	5
5	<i>Принципы скрэччера</i>	
	Чем оно отличается от прочего DIY	7
6	<i>Инструмент</i>	
	Кустарь-одиночка с мотором	9
6.1	Электроинструмент	10
6.1.1	Дрель	10
6.1.2	Лобзик	10
6.2	Паяльник	11
6.2.1	Паяльная станция	12
6.3	Жигитатель	12
6.3.1	Фрезерный шпиндель	14
6.4	Ручной инструмент	15
6.5	Pro'sKit	16
6.5.1	Инструмент до 1000 В	17
6.5.2	Хранение	17
6.5.3	Радиомонтаж	17
6.5.4	Наборы	19
6.6	Прочие	20
7	<i>Технологии</i>	
	Готовим травильный аквариум	21
7.1	Введение	21
7.2	Выбор концепции и комплектующих или начинаем готовку	21
8	<i>Хрюнекс</i>	
	Собираем Cross Linux	25
8.1	Linux для встраиваемых систем	26
8.2	Требования к системе сборки (BUILD)	26
8.3	Makefile и пакеты	27

8.4	Файловая структура проекта	27
8.4.1	mk.rc : скрипт генерации Makefile	27
8.4.2	cross/head.mk	28
8.4.3	dirs : создание рабочих каталогов	29
8.4.4	версии пакетов	30
8.4.5	пакеты	31
8.4.6	команды	31
8.4.7	*clean : очистка дерева проекта	32
8.4.8	debian : установка пакетов Debian Linux	32
8.4.9	набор макросов для configure	33
8.4.10	gz : зеркализование исходников	33
8.4.11	правила распаковки архивов исходников	34
8.5	Сборка BUILD -части	34
8.5.1	binutils : ассемблер и линкер	34
8.5.2	cclibs : библиотеки поддержки GCC	35
8.5.3	gcc : сборка компилятора GCC	36
8.6	Сборка базовой целевой системы	37
8.6.1	kernel : ядро Linux	37
8.6.2	libc : главная библиотека uClibc	38
8.6.3	bb : пакет UNIX-утилит BusyBox	38
8.7	root : Генерация файловой системы	39
8.7.1	boot : сборка загрузчика	39
8.7.2	user : сборка пользовательского ПО	40
8.8	qemu : Запуск готовой системы в эмуляторе QEMU	40
8.8.1	cross/qemu.mk	40
8.9	Конфигурирование ядра	40
8.10	Опции общие для всех вариантов сборки	41
8.10.1	rootfs в RAM	41
8.10.2	режим реального времени	41
8.10.3	часы и таймеры	41
8.10.4	вывод printk и отладка	42
8.10.5	форматы исполняемых файлов	42
8.10.6	носители данных	42
8.10.7	файловые системы	44
8.10.8	интернационализация	45
8.10.9	мультимедиа	45
8.11	Опции для архитектуры i386	45
8.11.1	опции процессора i486	48
8.11.2	опции для эмулятора qemu386	48

1 *Об этом журнале*

Сделай сам, расскажи другим

Наблюдая современные информационные тренды в Internete, можно заметить, что большое внимание уделяется различным самоделкам, DIY, 3D-принтерам, любительской электронике и концептам различных гаджетов.

Если попробовать взглянуть немного дальше, можно заметить все более и более заметное развитие такого явления как «Персональное производство» — большой интерес вызывает возможность создания и изготовления уникальных вещей, нужных только конкретному человеку.

С другой стороны, все усложняющиеся вещи и технологии вызывают у людей желание начать с нуля, создать что-то пользуясь старыми приемами. В клинических случаях попадаются особи, испытывающие дикий баттхерт от глобализации и массового производства, и бегущие подальше от цивилизации, прихватив с собой генератор и мобильник с Internetом Θ .

Вполне можно ожидать, что эти тенденции приведут к появлению и оформлению нового культурного течения, которое можно назвать скрэтчинг¹.

Этот журнал создан для скрэтчеворов — людей, чье хобби создавать вещи и технологии по следам уже существующих, в сотый раз изобретать велосипед, чтобы разобраться как оно работает, научиться делать самому, а возможно найти новый или забытый способ что-то сделать, и конечно получить удовольствие от процесса поиска.

2 *КопиПаста*

Как сказать "Начать с нуля" ?

- На английском — to start from scratch; to start over.
- На испанском — empezar de cero; empezar de nuevo.
- На итальянском — partire dal niente.

В общем-то во всех языках мы видим «кальку», выделяется только одно, содержащее слово «scratch» (царапина, черта).

С момента его возникновения, это выражение немного поменяло свое значение. Сейчас оно используется, когда мы хотим сказать «начать снова, начать с начала» в том смысле, что мы потерпели поражение при первой попытке.

Фраза родилась в конце 19-го века и тогда просто значила «начинать без преимуществ». Слово «scratch» использовалось с 18-го века как спортивный термин, обозначающий линию старта, прочерченную на земле. Впервые такая

¹ тут бы хорошо подошло слово «рукоблудие», но термин к сожалению уже занят, а другого русского аналога подобрать пока не удалось

линия упоминалась в описании игры в крикет — на ней стоял игрок, отбивающий мяч.

«Start from scratch» в качестве понятия «начинать с нуля» пришло к нам из бокса. Прочерченная линия определяла позиции боксеров, когда они стояли друг напротив друга в начале поединка. Отсюда также произошло выражение «up to scratch», (быть на должной высоте, в прекрасной форме), т.е. соответствовать стандартам, предъявляемым боксерам, делающим заявку на матч.

Позднее «scratch» стали называть любую стартовую точку в бегах. Термин стали использовать в «гандикап»-соревнованиях (handicap), в которых более слабый участник получает фору. Например, в велоспорте те, у кого нет преимуществ, стоят на линии, в то время как остальные стоят впереди. Другие виды спорта, особенно гольф, заимствовали переносное значение «scratch» как термин для обозначения «без преимуществ — начинать с нуля».

В The Fort Wayne Gazette (апрель 1887) содержится самое раннее упоминание «start from scratch» — в репортаже о «‘no-handicap» велосипедной гонке:

«It was no handicap. Every man was qualified to and did start from scratch.»

По моим наблюдениям, «start from scratch» употребляется чаще в письменной речи (например, уже несколько раз видела его в статьях в интернете), а «to start over» — в разговорной (слышала в американском сериале).

© Юлия Горбунова

3 *КопиПаста*

Персональное производство еще один шаг к реконизму

Один из важных моментов в построении реконицистической экономики — это трансформация традиционного, корпоративного производства, основанного на обязательной организации, как в смысле объединения людей, средств производства, финансовых и материальных ресурсов, так и в смысле появления так называемых юридических лиц как практически единственных субъектов производства. Такое производство в значительной части сфер деятельности будет вытесняться индивидуальным производством, когда любой желающий, используя так называемые микрофабрики — миниатюрный комплект универсального оборудования, сможет производить достаточно широкую линейку продукции, как для личного пользования, так и для продажи. Произойдет нечто вроде возврата к ремесленному производству средневековья и даже к натуральному хозяйству, но на неизмеримо более высоком технологическом уровне. Особую ценность в таких условиях обретет информация — продаваться будет не товар, а инструкция для микрофабрики, как данный товар изготовить. Конечно, такие инструкции будут не только продаваться, но и распространяться бесплатно, а также вороваться. Разумеется, это серьезно поменяет привычную нам социально-экономическую систему.

В последнем номере журнала «Наука и жизнь» (№8 за 2012 год), появилась

небольшая заметка, в которой рассказывается о разработке профессора Массачусетского технологического института Нила Гершенфельда, который предложил концепцию миниатюрной фабрики-лаборатории (Fab Lab). Фабрика-лаборатория представляет собой комплекс станков, совместно работающих под управлением персонального компьютера. Идея Гершенфельда получила широкое распространение и десятки университетов и исследовательских центров экспериментируют с такими мини-фабриками. В России первая такая фабрика создана в Московском институте стали сплавов, в ее составе фрезерный станок для обработки древесины, пластиков и мягких металлов, гравировальный прецизионный станок для производства печатных плат, установка лазерной резки, плоттер для раскюя гибких материалов и производства гибких микросхем, и 3D-принтер, предназначенный для изготовления любых изделий из ABS-пластика.

Так что, возможно, что лет через десять, для того чтобы поменять надевший мобильный телефон, мы будем заходить на сайт какой-нибудь Нокии, скачивать файл с данными, запускать его в программе на домашнем компьютере, а стоящий на тумбочке агрегат, очертаниями смахивающий на современное МФУ, погудев пару минут, выбросит в приемный лоток еще горячую, пахнущую свежим пластиком мобилку... ☺

© AG

4 *Новости технологий*

Злобный янки в 3D-танке

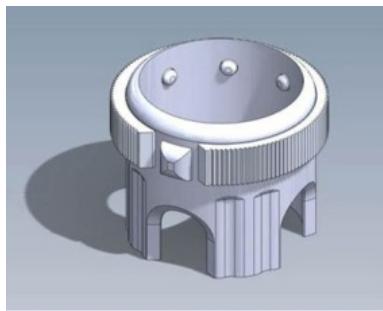
Полевые 3D-принтеры на службе американской армии

Пока специалисты в области 3D-печати рассуждают о перспективах применения технологии, а энтузиасты осторожно говорят о потенциальной возможности печати необходимого скарба сразу на лунной базе (чтобы не тащить лишнее с Земли), американская армия без всяких промедлений нашла применение 3D-печати уже сейчас. Военные США стали использовать мобильные лаборатории Expeditionary Lab Mobile с 3D-принтерами в комплекте.

Основными задачами лабораторий Expeditionary Lab Mobile (сокращённо — ELM) будет изготовление одноразовых инструментов для нужд армии, а также внесение корректирующих дополнений в уже существующее оборудование — «полевое» использование часто требует определённой доводки. В качестве примера приводится случай, когда войска получают партию карманных фонарей с дефектом — быстро выходящим из строя предохранителем выключателя. Находясь в кармане у военного, такой фонарь может самопроизвольно включаться и либо выдать местонахождение бойца, либо впустую разрядить батарейки. Однако, имея под рукой ELM, можно быстро допечатать предохранители, без необходимости отсылки всей партии обратно в США для замены.



Чебураторы на тропе войны



Ещё одним примером можно назвать реальный случай недоработки в конструкции миноискателя, приведший к тому, что время работы прибора из-за

иракской жары сократилось с восьми часов до 45 минут. В результате во время многодневных миссий солдаты были вынуждены носить большое количество дополнительных батарей. Использование ELM позволило сконструировать адаптер для использования батарей другого типа и увеличить время работы миноискателя до девяти часов.

Expeditionary Lab Mobile представляет собой стандартный грузовой контейнер ($6,1 \times 2,4$ м), внутри которого находятся 3D-принтер, специальные станки с ЧПУ (для изготовления более сложных деталей из стали и алюминия) и набор традиционных инструментов: резак, сварочный аппарат, циркулярная пила, маршрутизатор, лобзик и сабельная пила. Кроме того, в комплекте ELM имеется спутниковое оборудование связи для проведения телеконференций с чиновниками и инженерами в США – для оперативных корректировок работы. При каждой лаборатории будут находиться два инженера. Все лаборатории будут связаны между собой единой компьютерной сетью.

Стоит отметить, что подобный способ изготовления износившихся или недостающих деталей довольно дорог: стоимость каждой лаборатории составляет около 2,8 миллиона долларов. Планируется, что первые ELM будут испытаны в Афганистане. Кроме того, можно надеяться, что успешное применение новых технологий на «поле боя» будет способствовать их внедрению для мирных операций. Например, во время стихийных бедствий.

© Компьютерра, Николай Маслухин

5 *Принципы скрэтчера*

Чем оно отличается от прочего DIY

- Из говна и палок

Чем больше г и кривее палки, тем круче скрэтчер

- Сделай сам, расскажи другим

Необходим активный обмен информацией для минимизации и так больших расходов на изобретение колес

- Минимум покупных изделий

В идеале изготовление всего из чисто природных материалов и без старто-вого инструмента

- Все покупные ништяки должны быть всегда доступны в любом ближайшем магазине

Чтобы каждый мог легко и быстро повторить понравившийся хак.

Следует обратить внимание, что этому принципу противоречит использование техно-мусора, различных деталей от старой техники и т.п. — вот сколько сейчас у вас например сломанных стиралок, или дохлых телевизоров в доме ?

Еще одно противоречие — покупка комплектухи по почте в Китае, и заказ редких компонентов в магазинах

- Покупаться должны **самые дешевые** и самые кривые комплектующие Но при этом не нужно скатываться на использование раритета — см. доступность.
- Приоритетно использование более ранней ступени *технологического передела*

Например вместо использование готового заводского сверла взять хвостовик от сломанного, и выпилить сверло самому. Правильнее было бы взять твердосплавную заготовку, но это противоречит принципу доступности, т.к. их нет в доступных магазинах. Вариант использование куска проката из инструментального сплава лучше, потому что можно еще повыделываться с термичкой ☺

- Должно использоваться *открытое программное обеспечение*

Причем написанное целиком самостоятельно на ассемблере, ну или хотя бы собрать Cross Linux From Scratch для DIY компьютера, спаянного из отдельных деталей с помощью самодельного паяльника.

В процессе неплохо попутно изобрести пару уникальных языков программирования, написать на них операционную систему и комплект программного обеспечения.

- Желательно использовать нетиповые приемы работы и технологии
- При разработке конструкций нужно стремиться использовать малоизвестные и уникальные конструктивные решения
- Максимум самодельного инструмента
- Идеал скрэтчера — пройти всю технологическую цепочку от каменного рубила до обрабатывающего центра с ЧПУ

И с разгона заскочить еще дальше, обогнав текущие лабораторные разработки по 3D-печати, зональной плавке и прочим свежакам технологии

- Минимум повторов готовых изделий и унификации

Каждая поделка должна быть прекрасна в своей уникальности, и ее область применения должна быть максимально узкозаточенной под ваши задачи. Применение унификации, общезвестных конструктивных решений и принципов работы неприемлемо, т.к. какой смысл повторять уже готовое изделие, которое можно купить ?!

- Больше науки

Копайте книги по математике, физике и химии, больше статей и техрасчетов. Чем больше матана и самопала, тем выше левел. Не забывайте

про пропагандизм достижений на форумах (особенно нетематических) и в оффлайне.

- Больше синей изоленты
- Обязательно используйте ардуину

Даже если устройство вообще не предполагает использование электричества — прикрутите микроконтроллер изолентой, и подключите к нему компьютер.

- Для успеха проекта обязательно нужен ковер
- На демонстрационном видео должно что-нибудь отвалиться или чихнуть волшебным синим дымом

Набор принципов скрэтчера выглядит похоже на инструкцию «Как просрать полимеры», поэтому как и в любом другом деле, не нужно доводить их исполнение до фанатизма. *Новизна и уникальность* на первом месте, и не надо забывать что это все же хобби, а не жизненная миссия. Нужно всего лишь следить за соблюдением баланса между потраченными средствами, временем, и полученным от процесса удовольствием.

Главное достоинство отработанных вещей и технологий — на их доводку и проверку уже было потрачено гигантское количество ресурсов. Самодельные аналоги в любом случае будут хуже и на порядок дороже, чем серийное изделие, за редким исключением узконишевого использования, для которого готовое решение почему-то не подходит.

Из положительных эффектов скрэтчера можно отметить хорошие обще-технические знания, и умение при необходимости быстро слепить «костыль» (временное решение проблемы) из подручных ресурсов.

6 *Инструмент* Кустарь-одиночка с мотором

Отличная серия видео по изготовлению токарника

Возникает вопрос, где взять самый дешевый, легко доставаемый и универсальный электропривод, и нужно ли вообще пользоваться электричеством.

Вопрос по использованию электричества оставляем самым упоротым скрэтчерам 80-го уровня. Будем исходить из того, что хоть какое-то (под нагрузку мощностью от $100 \div 200$ Вт) сетевое электричество доступно сейчас всем, кроме туристов, огородников и прочих полевиков, не укомплектованных бензогенератором.

Соответственно в комплекте базового инструмента предполагаем наличие минимум электродрели и паяльника.

6.1 Электроинструмент

6.1.1 Дрель



Дрель ударная сетевая Praktyl-R PID13D01 400 Вт (!)395 р.



Дрель безударная сетевая Интерскол Д-11/530ЭР (с БЗП) 1120 р.

Дрель — одноразовая китайчатаина от 400 р. Цена крайне низкая, поэтому в целях тестирования взял один экземпляр на натурные испытания, результаты по живучести будут в следующих номерах. Подаются уже брендированные на Леруа Мерлен, наклейка «PID13D01 Ударная дрель 400 Вт, 13 мм». Скорость регулируется глубиной нажатия курка, крутилка на курке ограничивает глубину механически, фиксатор держит скорость близко к минимальной, запаха горелой пластмассы через несколько минут работы на холостом ходу нет.

По надежности рекомендуется Интерскол 1100+ р. Надежность Интерскола — не «китай», классика ДУ-580ЭР работает в хвост и гриву в университете ежедневно с ~2005 г., используется криворукими студентами, лежит в подвале в пыли от точила, и никаких вопросов даже со щетками.

Если не планируете много сверлить бетон, **берите дрель без ударного механизма**: отсутствуют лишние продольные перемещения, что может быть важно при использовании в качестве шпинделя сверлильного станка, и механизации других технологических поделок.

Шуруповерт — буржуйство, у него нет 43мм шейки для фиксации, поэтому как средство электропривода он практически бесполезен, и нужен собственно для заворачивания большого количества саморезов. Хотя наличие ограничителя крутящего момента и малые габариты удобны при сверлении и сборке поделок.

6.1.2 Лобзик



Лобзик Praktyl 350 Вт 356 р.



Лобзик Makite 4329 2260 р.

Лобзик опционален, и куда полезнее шуруповерта, китай-хлам 350+ р, чуть поприличнее 2000+ р. Не берите с маятником дешевле 5–7 тыс.р.

6.2 Паяльник



Паяльник ЭПСН-25/220



Паяльник 220В 25Вт, СВЕТОЗАР, SV-55310-25 230 р.

Паяльник — обязателен дешевый сетевой мощностью не менее 20 Вт, типа ЭПСН-25/220. Ограничитель мощности или регулятор температуры трубы скрэтчера должен собрать самостоятельно.

Для сборки электроники хорошо также иметь маленький монтажный 12 В 8 Вт от паяльной станции ZD-927 (~100 р), без самой станции.



Паяльник 220В 25Вт ZD-721N 175 р.



Паяльник для станции ZD-927
12 В 8 Вт 85 р.

6.2.1 Паяльная станция

Если не жалко 500 р., берите ZD-927 целиком, внутри простейший регулятор мощности, и вам не понадобится источник питания на 12 В, который вы еще не сделали. Но тру~~з~~ путь — конечно собрать свой паяльник целиком, из никрома, жала из толстой проволоки или медной шины, и самостоятельно выточенной ручки.



Паяльная станция ZD-927 520 р.

Паяльные станции типа Lukey 702/853D (3000+ р) естественно не рассматриваем ☺. Для работы или регулярного хобби паяльная станция с феном, а может даже и встроенным источником питания, вещь незаменимая, и не такая уж дорогая, но для скрэтчера слишком технологичная.

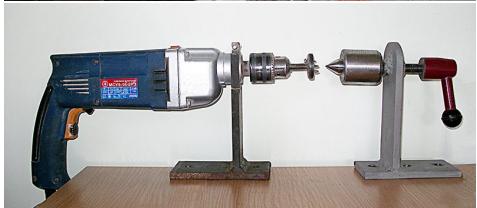


Паяльная станция LUKEY 702
3100 р.



Паяльная станция LUKEY 853D с
источником питания 5200 р.

6.3 Жигатель



Первый кандидат на место универсального электропривода достается той самой дрели, не забываем об обязательном наличии 43 мм монтажной шейки. Достоинство дрели как привода — прямое подключение к сети, встроенный редуктор, есть модели с простой регулировкой оборотов, резьба и отверстие под винт на валу, в комплекте есть патрон для зажима мелких деталей в точилке²

Ограниченно доставаемые двигатели от стиральных машин, отличаются мощностью и оборотистостью, особенно от старых моделей. Часто доступны сразу с готовым шкивом на валу, который иногда проще использовать, чем снять.

Автозапчасти: привод печки Камаза, двигатель постоянного тока 24 В 50 Вт

Ждвигатель Вятка-Автомат 19?? г.



² БЗП удобнее, патрон с ключем дает лучший зажим и возможно точнее



Двигатель печки Камаза



АИРЕ 56 В2, 0.2 КВт

Новые асинхронные двигатели АИРЕ 56 В2/В4 (3000/1500 об.) с заводским конденсатором, подключается к сети ~220 В, цена от 2500 р. С ростом размеров и мощности цена резко повышается. Следует обратить внимание на возможность монтажа на дополнительный фланцевый подшипниковый щит, (?) с моделями АИРЕ 80.

6.3.1 Фрезерный шпиндель

Съемные фрезерные шпинNELи, поставляются отдельно или в комплекте с насадкой ручного фрезера по дереву. Лучшие, со стальной шейкой — Kress, активно применяются хобби-ЧПУшниками.



Фрезерный двигатель KRESS
530/800/1050 FM(E) 5600+ р.



Шпиндель Интерскол ФМ-30/750
/снят с производства/

Попроще и сильно дешевле делал Интерскол, иногда попадается попадает попадает.

Недостаток как универсального привода — они высокоскоростные, возникают проблемы с понижающими передачами. Применение — приводной высокоскоростной инструмент: боры, фрезы по дереву, микроинструмент для граверов (микродиски, шарошки).



**Фрезер сетевой
Интерскол ФМ-55/1000 Э 5050 р.**

Китайские воздушные шпинделы постоянного тока с цанговыми патронами ER11: привод для сверлилки и микроинструмента. Требуют источник питания постоянного тока 9÷48 В.

6.4 Ручной инструмент

По мелкому ручному инструменту вопрос открыт.

Учитывая доступность и наличие дешевых вариантов, стоит ли использовать старый опыт мастеров-ремесленников, когда ученику давали только напильник, и он сам должен был изготовить себе весь инструмент ?

По крайней мере, вот этот вариант точно не подходит ☺:



РК-5308ВМ универсальный набор инструментов Pro'sKit

Но пара надфилей, заточной камень на дрель, комплект сверел и несколько листов наждачки вполне допускаются ☺.

Если хочется посложнее, можно ограничиться только парой электродвигателей:

1. относительно медленный высокомоментный АИРЕ 56 В2/4 на силовой привод и
2. высокоскоростной $10+$ тыс.об $^{-1}$ для сверления, шлифования насадками и т.п. операции допускающие работу с большими скоростями

6.5 Pro'sKit

Отдельного обзора заслуживает инструмент и наборы Pro'sKit / ги:

6.5.1 Инструмент до 1000 В

Для электромонтажных работ обязательно приобретите комплект высоковольтного инструмента до 1000 В:



PM-911 Пассатижи 1 кВ



PM-917 Кусачки (бокорезы) 1 кВ

6.5.2 Хранение



103-132D Кассетница для деталей и компонентов



SB-3428SB Портативная кассетница для саморезов и т.п.

6.5.3 Радиомонтаж



8PK-30D Кусачки миниатюрные



1PK-055S Длинногубцы изогну-
тые



1PK-709 Длинногубцы-кусачки



1PK-29 Круглогубцы



1PK-3001Е Клещи для зачистки проводов прецизионные (стриппер)



1PK-101Т Пинцет прямой



PD-374 Тиски на струбцине

6.5.4 Наборы



1PK-616В Набор инструментов для электроники профессиональный

1PK-813B Набор базовых инструментов для электроники



По личному опыту: в 1PK-813B не хватает мелкого мультиметра, стриппера 1PK-3001Е, микрокусачек типа 8PK-30D, канифоли, ножа, настроечную отвертку заменить индикаторной.

6.6 Прочие

Попалась интересная недорогая отвертка:

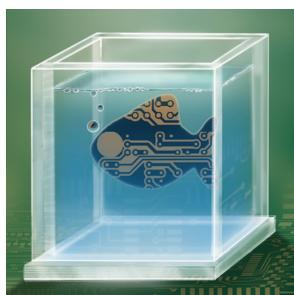


Фиксация четкая, исполнение очень неплохое, позволяет добраться до узких мест. Из минусов: ручка похоже не цельнометаллическая, при изломе есть риск распороть руку.

7 Технологии

Готовим травильный аквариум

7.1 Введение



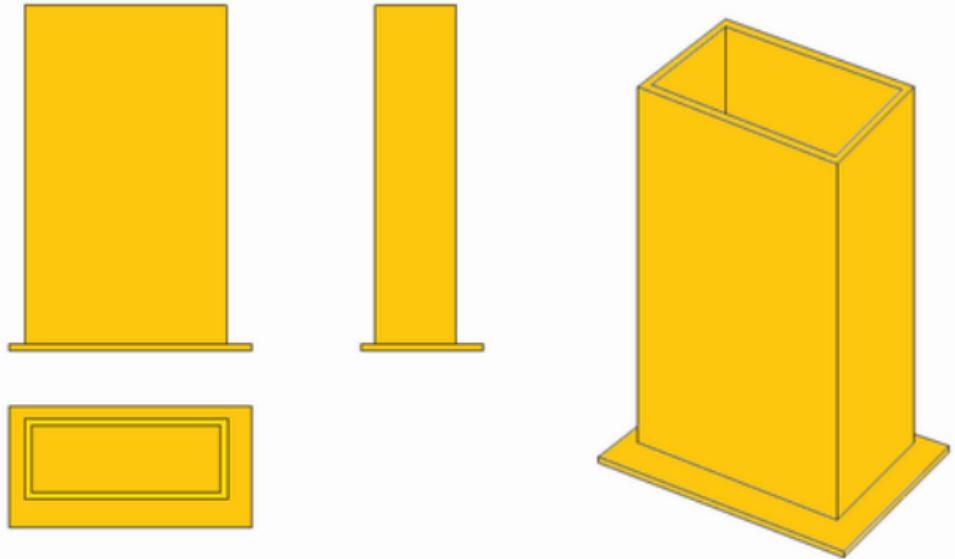
Данная разработка, как и любая другая у меня, делалась по принципу увидел клевую вещь — захотел такую же. Сей проект — это творчески доработанный аквариум для травления печатных плат в домашних условиях. Этих аквариумов уже гуляет солидное количество по сети.

Я хоть травлю платы не слишком часто, но иногда случается и так получается, что эти самые платы у меня плохо получаются. ☺ И вот однажды в поисках секрета получения хороших плат я наткнулся на вот эту штуку.

Мне она понравилась и я захотел такую же. Но как и любая идея которая у меня зреет, эта ждала своего часа довольно долго. Пока в один прекрасный день я не уволился с работы, а до устройства на следующую у меня оставались три недели лишнего отпуска. Их-то я и решил потратить с пользой. Что из этого получилось я и хочу вам показать.

7.2 Выбор концепции и комплектующих или начинаем готовку

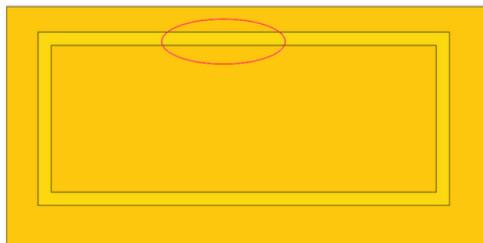
Итак задача ясна и я начал думать как её осуществить. Тупо повторить разработку **JeckDigger** (ссылка выше) мне было неинтересно, ведь впереди целых три недели! И я начал думать, что же такого интересного можно сделать. С самим аквариумом из оргстекла все ясно, тут что-то новое придумать сложно. Единственное, если у JeckDigger аквариум изготавливается из подручных средств, то у меня был доступ к фрезерному станку и я решил им воспользоваться по полной. Быстро наваял чертежи аквариума и мне их по дружбе сформировали. В качестве материала я использовал оргстекло толщиной 5 мм. Получилось как-то так :



Чертеж аквариума

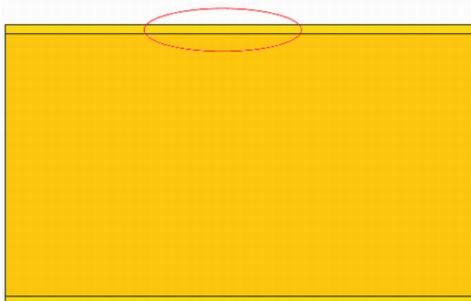
Файлы чертежей в формате .dwg

А вот и детализировка:



Основание

Красным выделено углубление 2 мм. Которое играет роль своеобразных направляющих. В них устанавливаются стенки аквариума. Сделал я их для лучшего позиционирования при слеивании аквариума ибо придавливать вертикальную стенку аквариума смазанную kleem к гладкой плоскости основание то еще удовольствие, добиться параллельности всех стенок очень сложно. А в таком варианте все удобно становится, да и клей в углубление хорошо заливается и не размазывается по всей поверхности.



Фронтальная стенка

Красным выделено углубление 2 мм в виде ступеньки. Сделано оно опять же для удобства склеивания и для того, чтобы на выходе получить аккуратный параллелепипед аквариума. ☺



Боковая стенка

Здесь никаких извращений. Обычный прямоугольник. ☺

Итак с самим аквариумом разобрались. Но этого конечно же мало для полного счастья. Как и у JeckDigger я решил добавить туда стандартный аквариумный распылитель.



Аквариумный распылитель

Распылитель служит своеобразной «палкой-мешалкой» травящего раствора. Ходят слухи, что это помогает ускорить процесс травления. Скажу честно, я без понятия правда это или нет, но идея мне понравилась.

Для распылителя был необходим компрессор. Я выбрал самый миниатюрный какой только нашел — AC-500.



Аквариумный компрессор

Объем нашего аквариума небольшой поэтому такого компрессора вполне хватит.

Но если вы думаете, что это все, то вы плохо меня знаете. ☺ Разойдясь я уже не мог остановиться. Я решил прикрутить к всему этому добру нагреватель с температурной регулировкой. Опять же ходит слух, что в теплой воде процесс травления идет быстрее³. Сказано — сделано. С качестве нагревателя был использован обычный аквариумный нагреватель мощностью 75 Вт.



AquaPili

Аквариумный нагреватель

Примечание 2 — Не обращайте внимания, что на картинке указана мощность 50 Вт. Просто достойной фотографии нагревателя, который приобрел я сделать не удалось, поэтому в срочном порядке был подключен Google ☺.

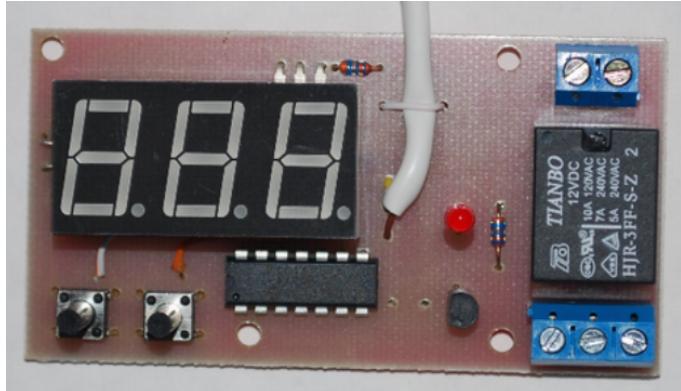
Правда нагреватель пришлось доработать, но об этом речь пойдет дальше.

Итак, нагреватель есть, осталось прикрутить к нему терморегулятор. И тут появилась дилема: либо собирать схему самому либо купить уже готовый. После непродолжительных дебатов с самим собой решение было принято в пользу

³ а если объединить компрессор и нагреватель, то плата должна травиться вообще в две секунды ☺

покупного. Ибо, делать плату самому было лень. Да и как её сделаешь, если аквариум для травления еще не готов? ☺ Налицо парадокс. ☺

Свободного времени было хоть отбавляй, поэтому не откладывая дело в долгий ящик я пошел на радио рынок. После непродолжительных поисков я нашел, что искал:



Плата регулятора тем-

пературы

По параметром он конечно немного превосходил мои скромные запросы, но я был не в обиде.

Итак, вроде с большего было понятно что делать, оставалось обдумать детали.

Например, я захотел не только регулировать температуру кнопками, но и включать/выключать распылитель при помощи кнопки. Так же появилась небольшая проблема связанная с питанием. Заключалась она в том, что нагреватель и компрессор работают от напряжения ~220 В, плюс для питания остальной электроники планировалось использовать зарядное устройство для сотового телефона, а оно так же подключалось в сети ~220 В. А это означало, что для работы аквариума необходимо было занимать аж три розетки, что меня нисколько не устраивало. Поэтому я решил сделать, так чтобы для питания прибора можно было обойтись одним сетевым кабелем. Так же я решил всю электронику, включая компрессор запихнуть в деревянное основание, на которое уже устанавливать сам аквариум из оргстекла. Что же, концепция, с большего была ясна — осталось дело за малым. ☺

Александр Ковалчук

8 *Хрюникс*

Собираем Cross Linux

В качестве примера применения возьмем относительно простое приложение: многофункциональные настенные часы, с синхронизацией времени через Internet, с будильником, медиапроигрывателем, блэкджеком и плюшками.

8.1 Linux для встраиваемых систем

Linux для встраиваемых систем⁴ — популярный метод быстрого создания комплекса ПО для больших сложных приложений, работающих на достаточно мощном железе, особенно предполагающих интенсивное использование сетевых технологий.

За счет использования уже существующей и очень большой базы исходных текстов ядра, библиотек и программ для Linux, бесплатно доступных в т.ч. и для коммерческих приложений, можно на порядки сократить стоимость разработки собственных программных компонентов, и при этом получить очень мощную команду бесплатных стронних разработчиков, уже знакомых с созданием ПО для Linux.

Из недостатков можно отметить:

- Отсутствие полноценной поддержки режима жесткого реального времени;
- Тяжелое ядро;
 - Поддерживаются только мощные семейства процессоров;
 - Значительные требования по объему RAM и общей производительности;
- Дремучесть техспециалистов, контуженных ТурбоПаскалем и Windowsом;

Для сборки emLinux-системы используется метод кросс-компиляции, когда используется *кросс-тулчейн*, компилирующий весь комплект ПО для компьютера с другой архитектурой⁵.

emLinux очень широко применяется на рынке мобильных устройств⁶, и устройств интенсивно использующих сетевые протоколы (роутеры, медиацентры).

8.2 Требования к системе сборки (BUILD)

Требования жесткие — 4x-ядерный процессор, 4+ Гб RAM, 64x-битный дистрибутив Linux (рекомедую Debian), и никаких виртуалок.

Возможна установка системы на флешку, в этом случае требования к RAM еще более ужесточаются — потребуется каталоги с временными файлами смонтировать как `tmpfs`.

Сборка под MinGW/Cygwin совершенно неживая. Если совсем никак без винды — используйте виртуалки, и будьте готовы ждать часами.

Можно попытаться сделать **билд-сервер** и на худшем железе, но будьте готовы к тормозам или внезапному окончанию памяти — ресурсоемка сборка тяжелых библиотек типа `libQt` или крупных пакетов типа `gcc`.

⁴ будем называть его emLinux

⁵ типичный пример — сборка ПО на ПК с процессором Intel i7 для Raspberry Pi или планшета на процессоре AllWinner/Tegra/...

⁶ в т.ч. является основой Android

В этом номере Скрэтчера описана сборка только базовой системы. Вы можете попробовать поставить Linux на виртуалку, на флешку, и на жесткий диск (если найдете место) и оценить возможности этих вариантов на сборке пакета `gcc`.

8.3 Makefile и пакеты

Сборка выполняется утилитой `make`, описание структуры проекта для которой присано в файлах **Makefile** и ***.mk**.

В обычных дистрибутивах пакетами называют архивы скомпилированных программ, устанавливаемых в Linux-систему. В нашем случае кросс-компиляции, будем называть *пакетом* архив исходных текстов определенной программы или компонента системы, вместе с секциями мейкфайла, выполняющего ее компиляцию.

Также пакет может быть не связан с компиляцией ПО, а выполнять какую-то вспомогательную работу.

Пакеты запускаются по своему имени вручную с помощью команды

```
1 user@builder: make [-f Makefile] <packname1> [<packname2>...]
```

8.4 Файловая структура проекта

Получите последнюю версию системы **cross** командой:

```
git clone  
1 user@builder: cd ~  
2 user@builder: git clone --depth=1 -o gh \  
3     https://github.com/ponyatov/cross.git cross
```

gz/	локальное зеркало архивов исходных текстов
src/	временный каталог разлаковки исходников
tmp/	каталог out-of-tree сборки пакетов
build/	каталог установки \$(BUILD)-части пакетов
\$(TARGET)/	rootfs целевой системы
user/	
cross/mk.rc	

8.4.1 mk.rc: скрипт генерации Makefile

Главный **Makefile** генерируется по частям скриптом:

mk.rc

```
1#!/bin/sh  
2 cat \  
3     head.mk dirs.mk \  
4
```

```

4 versions.mk packages.mk \
5 debian.mk \
6 commands.mk clean.mk srcrules.mk \
7 gz.mk \
8 cfg.mk \
9 binutils.mk cclibs.mk gcc.mk \
10 kernel.mk libc.mk busybox.mk \
11 user.mk boot.mk \
12 qemu.mk \
13> Makefile
14 cat \
15 config/ramdisk.kernel \
16 config/binformats.kernel \
17 config/realtime.kernel \
18 config/debug.kernel \
19 config/storage.kernel \
20 config/filesystems.kernel \
21 config/media.kernel \
22 config/clock.kernel \
23 config/i10n.kernel \
24> config/kernel.all

```

Это было сделано для удобства вставки частей (.mk файлы) в документацию, в т.ч. и в эту статью.

8.4.2 cross/head.mk

- **HW** железка, на которой будет запускаться
- **APP** приложение

Во вложенных файлах прописываются переменные:

- **ARCH** архитектура железки
- **CPU** процессор и
- **TARGET** триплет целевой платформы

head.mk

```

1
2 HW = qemu386
3 APP = clock
4
5 include config/hw/$(HW).mk
6 include config/arch/$(ARCH).mk

```

```
7 include config/cpu/$(CPU).mk  
8 include config/app/$(APP).mk
```

config/hw/qemu386.mk

```
1 # hw: qemu386 QEMU i386 emulated PC  
2 ARCH = i386  
3 CPU = i486
```

config/arch/i386.mk

```
1 # arch: i386  
2 TARGET = $(CPU)-linux-uclibc  
3 ARCH_KERNEL = x86
```

config/cpu/i486.mk

```
1 # cpu: i486
```

8.4.3 dirs: создание рабочих каталогов

GZ зеркало архивов исходных текстов

SRC сюда распаковываются исходные тексты

TMP каталог для out-of-tree сборки

BUILD кросс-компилятор и другие программы **BUILD**-системы

ROOT целевая файловая система **rootfs** (initrd)

BOOT файлы, относящиеся к процессу загрузки

Пакет **dirs** создает дерево каталогов.

dirs.mk

```
1  
2 GZ = $(PWD) / gz  
3 TMP = $(PWD) / tmp  
4 SRC = $(PWD) / src  
5 BUILD = $(PWD) / build  
6 ROOT = $(PWD) / $(TARGET)  
7  
8 BOOT = $(ROOT) / boot  
9 USR = $(ROOT) / usr  
10 BIN = $(USR) / bin  
11
```

```

12 DIRS = \
13   $(GZ) $(TMP) $(SRC) \
14   $(BUILD) $(ROOT) \
15   $(BOOT) $(USR) $(BIN)
16
17 .PHONY: dirs
18 dirs:
19   mkdir -p $(DIRS)

```

8.4.4 версии пакетов

В этой части прописаны версии пакетов.

versions.mk

```

1
2 # gcc libs
3 GMP_VER = 5.1.3
4 MPFR_VER = 3.1.2
5 MPC_VER = 1.0.2
6
7 # cross compiler
8 BINUTILS_VER = 2.24
9 GCC_VER = 4.9.1
10
11 # core
12 KERNEL26_VER = 2.6.39.4
13 KERNEL3_VER = 3.16.3
14 LIBC_VER = 0.9.33.2
15 BB_VER = 1.22.1
16
17 # libs
18 ZLIB_VER = 1.2.8
19 SDL_VER = 1.2.15
20 FT2_VER = 2.5.2
21 SDL_TTF_VER = 2.0.11
22
23 # arch:i386
24 SYSLINUX_VER = 6.02
25 GRUB_VER = 0.97
26 GRUB2_VER = 2.00
27 MEMTEST_VER = 5.01

```

8.4.5 пакеты

Здесь прописаны полные имена пакетов вместе с версиями.

packages.mk

```
1
2 # gcc libs
3 GMP = gmp-${GMP_VER}
4 MPFR = mpfr-${MPFR_VER}
5 MPC = mpc-${MPC_VER}
6
7 # cross compiler
8 BINUTILS = binutils-${BINUTILS_VER}
9 GCC = gcc-${GCC_VER}
10
11 # core
12 KERNEL26 = linux-${KERNEL26_VER}
13 KERNEL3 = linux-${KERNEL3_VER}
14 LIBC = uClibc-${LIBC_VER}
15 BB = busybox-${BB_VER}
16
17 # libs
18 ZLIB = zlib-${ZLIB_VER}
19 SDL = SDL-${SDL_VER}
20 FT2 = freetype-${FT2_VER}
21 SDL_TTF = SDL_ttf-${SDL_TTF_VER}
22
23 # arch: i386
24 SYSLINUX = syslinux-${SYSLINUX_VER}
25 GRUB = grub-${GRUB_VER}
26 GRUB2 = grub2-${GRUB2_VER}
27 MEMTEST = memtest86+-${MEMTEST_VER}
```

8.4.6 команды

Макросы команд, используются далее:

commands.mk

```
1
2 CPU_CORES ?= $(shell grep processor /proc/cpuinfo | wc -l)
3
4 MAKE = make -j $(CPU_CORES)
5 INSTALL = make install-strip
6 DISTCLEAN = make distclean
7
```

```

8 WGET = wget -N -P $(GZ)
9
10 CCACHE = ccache
11
12 BCC = $(CCACHE) gcc -pipe
13 BCXX = $(CCACHE) g++ -pipe
14
15 TCC = $(CCACHE) $(TARGET)-gcc
16 TCXX = $(CCACHE) $(TARGET)-g++

```

8.4.7 *clean: очистка дерева проекта

Для каталогов, прописанных в **/etc/fstab** как **tmpfs** (файловая система в RAM) для экономии ресурса флешки и максимального ускорения сборки:

/etc/fstab

```

1 tmpfs /home/user/cross/tmp tmpfs auto ,uid=user ,gid=user 0 0
2 tmpfs /home/user/cross/src tmpfs auto ,uid=user ,gid=user 0 0

```

существует пакет **ramclean**, выполняющий их очистку.

Если вы используете такие каталоги, добавляете его вызов в конце вторым + пакетом, чтобы освободить RAM для следующей сборки.

clean.mk

```

1
2 .PHONY: distclean
3 distclean:
4     make ramclean
5     rm -rf $(BUILD) $(TARGET)
6     make dirs
7
8 .PHONY: ramclean
9 ramclean:
10    rm -rf $(SRC)/* $(TMP)/*

```

8.4.8 debian: установка пакетов Debian Linux

Перед сборкой нужно поставить несколько пакетов разработчика:

debian.mk

```

1
2 .PHONY: debian
3 debian:
4     sudo aptitude install \

```

```
5      ccache gcc g++ \
6      flex bison m4 \
7      bc make git \
8      xz-utils bzip2 \
9      ncurses-dev \
10     qemu
```

8.4.9 набор макросов для configure

- **CFG** общая часть запуска **configure**
- **BCFG** для **BUILD**-пакетов
- **TCFG** для **TARGET**-пакетов
- **--disable-nls** диагностические сообщения и документация только на английском
- **--prefix** каталог установки скомпилированного пакета

cfg.mk

```
1 CFG = configure --disable-nls
2
3
4 BCFG = $(CFG) CC="$(BCC)" CXX="$(BCXX)" --prefix=$(BUILD)
5 TCFG = $(CFG) CC="$(TCC)" CXX="$(TCXX)" --prefix=$(TARGET)
```

8.4.10 gz: зеркалирование исходников

Пакет **gz** выполняет загрузку из Interneta архивов исходных текстов пакетов.

Длительная и потребляющая трафик операция, нужен онлайн. По-хорошему архив исходников тут было бы желательно загружать через пиринговые сети, а не нагружать зеркала.

gz.mk

```
1
2 .PHONY: gz
3 gz:
4     # gcc libs
5     $(WGET) ftp://ftp.gnu.org/pub/gmp/$(GMP).tar.bz2
6     $(WGET) http://www.mpfr.org/mpfr-current/$(MPFR).tar.bz2
7     $(WGET) http://www.multiprecision.org/mpc/download/$(MPC).tar.bz2
8     # cross compiler
9     $(WGET) http://ftp.gnu.org/gnu/binutils/$(BINUTILS).tar.bz2
```

```

10 $(WGET) http://gcc.skazkaforyou.com/releases/$(GCC)/$(GCC).tar.xz
11 # core
12 $(WGET) https://www.kernel.org/pub/linux/kernel/v2.6/$(KERNEL_V2)
13 $(WGET) https://www.kernel.org/pub/linux/kernel/v3.x/$(KERNEL_V3)
14 $(WGET) http://www.uclibc.org/downloads/$(LIBC).tar.xz
15 $(WGET) http://busybox.net/downloads/$(BB).tar.bz2
16 # libs
17 $(WGET) http://www.zlib.net/$(ZLIB).tar.xz
18 $(WGET) http://www.libsdl.org/release/$(SDL).tar.gz
19 $(WGET) http://download.savannah.gnu.org/releases/freetype/$(FREETYPE)
20 $(WGET) http://www.libsdl.org/projects/SDL_ttf/release/$(SDL_TTF)
21 # arch-specific
22 make gz_$(ARCH)
23
24 .PHONY: gz_i386
25 gz_i386:
26   $(WGET) https://www.kernel.org/pub/linux/utils/boot/syslinux/9.2.0/syslinux-9.2.0.tar.xz
27   $(WGET) ftp://ftp.gnu.org/gnu/grub/$(GRUB2).tar.xz
28   $(WGET) ftp://alpha.gnu.org/gnu/grub/$(GRUB).tar.gz
29   $(WGET) http://www.memtest.org/download/$(MEMTEST_VER)/$(MEMTEST)

```

8.4.11 правила распаковки архивов исходников

srcrules.mk

```

1 $(SRC)/%/README: $(GZ) %.tar.gz
2   cd $(SRC) && zcat $< | tar x && touch $@
3 $(SRC)/%/README: $(GZ) %.tar.bz2
4   cd $(SRC) && bzcat $< | tar x && touch $@
5 $(SRC)/%/README: $(GZ) %.tar.xz
6   cd $(SRC) && xzcat $< | tar x && touch $@

```

8.5 Сборка BUILD-части

8.5.1 binutils: ассемблер и линкер

Пакет **binutils** включает ассемблер, линкер и вспомогательные программы для работы с объектными файлами в формате *ELF*.

- **-target** триплет целевой системы
- **-with-sysroot** каталог с include/lib файлами целевой системы
- **-with-native-system-header-dir** относительно **SYSROOT**

binutils.mk

```
1 CFG_BINUTILS = \
2   --target=$(TARGET) \
3   --with-sysroot=$(ROOT) \
4   --with-native-system-header-dir=/include
5
6 .PHONY: binutils
7 binutils: $(SRC)/$(BINUTILS)/README
8   rm -rf $(TMP)/$(BINUTILS) && \
9     mkdir $(TMP)/$(BINUTILS) && \
10    cd $(TMP)/$(BINUTILS) && \
11    $(SRC)/$(BINUTILS)/$(BCFG) $(CFG_BINUTILS) && \
12    $(MAKE) && $(INSTALL)
13
```

8.5.2 cclibs: библиотеки поддержки GCC

cclibs.mk

```
1 .PHONY: cclibs gmp mpfr mpc
2 cclibs: gmp mpfr mpc
3
4 CFG_CCLIBS = \
5   --with-gmp=$(BUILD) \
6   --with-mpfr=$(BUILD) \
7   --with-mpc=$(BUILD) \
8   --disable-shared
9
10 CFG_GMP = $(CFG_CCLIBS)
11 CFG_MPFR = $(CFG_CCLIBS)
12 CFG_MPC = $(CFG_CCLIBS)
13
14 gmp: $(SRC)/$(GMP)/README
15   rm -rf $(TMP)/$(GMP) && \
16     mkdir $(TMP)/$(GMP) && \
17     cd $(TMP)/$(GMP) && \
18     $(SRC)/$(GMP)/$(BCFG) $(CFG_GMP) && \
19     $(MAKE) && $(INSTALL)
20
21 mpfr: $(SRC)/$(MPFR)/README
22   rm -rf $(TMP)/$(MPFR) && \
23     mkdir $(TMP)/$(MPFR) && \
24     cd $(TMP)/$(MPFR) && \
```

```

26 $(SRC) / $(MPFR) / $(BCFG) $(CFG_MPFR) && \
27 $(MAKE) && $(INSTALL)
28
29 mpc: $(SRC) / $(MPC)/README
30 rm -rf $(TMP) / $(MPC) && \
31 mkdir $(TMP) / $(MPC) && \
32 cd $(TMP) / $(MPC) && \
33 $(SRC) / $(MPC) / $(BCFG) $(CFG_MPC) && \
34 $(MAKE) && $(INSTALL)

```

8.5.3 gcc: сборка компилятора GCC

gcc.mk

```

1 CFG_GCC = $(CFG_BINUTILS) $(CFG_CCLIBS) \
2   --without-headers --with-newlib \
3   --disable-threads \
4   --enable-languages="c"
5
6
7 .PHONY: gcc
8 gcc: $(SRC) / $(GCC)/README
9 rm -rf $(TMP) / $(GCC) && \
10 mkdir $(TMP) / $(GCC) && \
11 cd $(TMP) / $(GCC) && \
12 PATH=$(BUILD)/bin : $(PATH) \
13 $(SRC) / $(GCC) / $(BCFG) $(CFG_GCC)
14
15 cd $(TMP) / $(GCC) && \
16 PATH=$(BUILD)/bin : $(PATH) \
17 $(MAKE) all-gcc
18
19 cd $(TMP) / $(GCC) && \
20 PATH=$(BUILD)/bin : $(PATH) \
21 $(MAKE) install-gcc
22
23 cd $(TMP) / $(GCC) && \
24 PATH=$(BUILD)/bin : $(PATH) \
25 $(MAKE) all-target-libgcc
26
27 cd $(TMP) / $(GCC) && \
28 PATH=$(BUILD)/bin : $(PATH) \
29 $(MAKE) install-target-libgcc
30
31 .PHONY: tc

```

```
32 tc : binutils cclibs gcc
```

8.6 Сборка базовой целевой системы

8.6.1 kernel: ядро Linux

kernel.mk

```
1
2 CFG_KERNEL = \
3     ARCH=$(ARCH_KERNEL) \
4     INSTALL_HDR_PATH=$(ROOT)
5
6 KERNEL = $(KERNEL3)
7
8 .PHONY: kernel
9 kernel: $(SRC)/$(KERNEL)/README
10    cd $(SRC)/$(KERNEL) && \
11        PATH=$(BUILD)/bin:$(PATH) \
12        $(DISTCLEAN)
13
14    cd $(SRC)/$(KERNEL) && \
15        PATH=$(BUILD)/bin:$(PATH) \
16        $(MAKE) $(CFG_KERNEL) allnoconfig
17
18    echo "CONFIG_CROSS_COMPILE=\"$(TARGET)-\" >> $(SRC)/$(KERNEL)/.config
19    echo "CONFIG_LOCALVERSION=\"$(HW) $(APP)\" >> $(SRC)/$(KERNEL)/.config
20    echo "CONFIG_DEFAULT_HOSTNAME=\"$(HW) $(APP)\" >> $(SRC)/$(KERNEL)/.config
21
22    cat config/kernel.all >> $(SRC)/$(KERNEL)/.config
23    cat config/arch/$(ARCH).kernel >> $(SRC)/$(KERNEL)/.config
24    cat config/cpu/$(CPU).kernel >> $(SRC)/$(KERNEL)/.config
25    cat config/hw/$(HW).kernel >> $(SRC)/$(KERNEL)/.config
26
27    cd $(SRC)/$(KERNEL) && \
28        PATH=$(BUILD)/bin:$(PATH) \
29        $(MAKE) $(CFG_KERNEL) menuconfig
30
31    cd $(SRC)/$(KERNEL) && \
32        PATH=$(BUILD)/bin:$(PATH) \
33        $(MAKE) $(CFG_KERNEL)
34
35    cp \
36        $(SRC)/$(KERNEL)/arch/$(ARCH)/boot/bzImage \
37        $(BOOT)/$(HW) $(APP).kernel
```

```

38
39 cd $(SRC) / $(KERNEL) && \
40 PATH=$(BUILD) / bin : $(PATH) \
41 $(MAKE) $(CFG_KERNEL) headers_install

```

8.6.2 libc: главная библиотека uClibc

libc.mk

```

1 CFG_LIBC = PREFIX=$(ROOT)
2
3 .PHONY: libc
4 libc: $(SRC) / $(LIBC) / README
5   cd $(SRC) / $(LIBC) && \
6     PATH=$(BUILD) / bin : $(PATH) \
7     $(MAKE) $(CFG_LIBC) distclean
8
9   cd $(SRC) / $(LIBC) && \
10    PATH=$(BUILD) / bin : $(PATH) \
11    $(MAKE) $(CFG_LIBC) allnoconfig
12
13 echo "KERNEL_HEADERS=\\"$(ROOT) / include\\\" >> $(SRC) / $(LIBC) / .config
14 echo "CROSS_COMPILER_PREFIX=\\"$(TARGET)-\\\" >> $(SRC) / $(LIBC) / .config
15
16 cat config/libc.all >> $(SRC) / $(LIBC) / .config
17 cat config/arch/$ (ARCH).libc >> $(SRC) / $(LIBC) / .config
18 cat config/cpu/$ (CPU).libc >> $(SRC) / $(LIBC) / .config
19 cat config/hw/$ (HW).libc >> $(SRC) / $(LIBC) / .config
20
21 cd $(SRC) / $(LIBC) && \
22 PATH=$(BUILD) / bin : $(PATH) \
23 $(MAKE) $(CFG_LIBC) menuconfig
24
25 cd $(SRC) / $(LIBC) && \
26 PATH=$(BUILD) / bin : $(PATH) \
27 $(MAKE) $(CFG_LIBC) install
28

```

8.6.3 bb: пакет UNIX-утилит BusyBox

busybox.mk

```

1
2 CFG_BB = \

```

```

3 CONFIG_PREFIX=$ (ROOT) \
4 CROSS_COMPILE=$ (TARGET)- \
5 SYSROOT=$ (ROOT)
6
7 .PHONY: bb
8 bb: $(SRC) / $(BB) / README
9   cd $(SRC) / $(BB) && \
10  PATH=$(BUILD) / bin : $(PATH) \
11  $(MAKE) $(CFG_BB) distclean
12
13 cd $(SRC) / $(BB) && \
14 PATH=$(BUILD) / bin : $(PATH) \
15 $(MAKE) $(CFG_BB) allnoconfig
16
17 cp config/app / $(APP).busybox $(SRC) / $(BB) /. config
18
19 cd $(SRC) / $(BB) && \
20 PATH=$(BUILD) / bin : $(PATH) \
21 $(MAKE) $(CFG_BB) menuconfig
22
23 cp $(SRC) / $(BB) /. config config/app / $(APP).busybox
24
25 cd $(SRC) / $(BB) && \
26 PATH=$(BUILD) / bin : $(PATH) \
27 $(MAKE) $(CFG_BB) install

```

8.7 root: Генерация файловой системы

8.7.1 boot: сборка загрузчика

boot.mk

```

1
2 ROOTREX = "./(boot | lib | include)"
3
4 .PHONY: root
5 root:
6   rm -rf $(ROOT)/etc ; cp -r config/etc $(ROOT)/etc
7   cp README $(ROOT)/etc/
8   chmod +x $(ROOT)/etc/init.d/*
9   cd $(ROOT) && find . | egrep -v $(ROOTREX) | cpio -o -H newc >
10  cat $(BOOT) / $(HW) $(APP).cpio | gzip -9 > $(BOOT) / $(HW) $(APP).rootfs
11
12 .PHONY: boot
13 boot: user root

```

8.7.2 user: сборка пользовательского ПО

user.mk

```
1 TDEBUG = -g0
2 TOPT = -Ofast
3 TPEDANTIC = -Wall -Wextra -Werror -ansi -pedantic-errors
4 TCFLAGS = $(TOPT) $(TPEDANTIC) $(TDEBUG)
5
6
7 .PHONY: user
8 user: $(BIN)/hello
9 #     make $(BIN)/hello
10
11 COMPILE_RULE = \
12     PATH=$(BUILD)/bin : $(PATH) \
13     $(TCC) $(TCFLAGS) -o $@ $< && \
14     PATH=$(BUILD)/bin : $(PATH) \
15     $(TARGET)-size $@ \
16 #     && \
17 #     PATH=$(BUILD)/bin : $(PATH) \
18 #     $(TARGET)-strip $@ \
19 $(ROOT)/%: user/%.c
20     $(COMPILE_RULE)
21 $(BIN)/%: user/%.c
22     $(COMPILE_RULE)
```

8.8 qemu: Запуск готовой системы в эмуляторе QEMU

8.8.1 cross/qemu.mk

На этом этапе мы имеем только базовую систему, которую можно запустить в эмуляторе, или на реальном x86-ом компьютере.

qemu.mk

```
1
2 .PHONY: qemu
3 qemu:
4     qemu -m 32M -kernel $(BOOT)/$(HW)$(APP).kernel -initrd $(BOOT)/
```

8.9 Конфигурирование ядра

Отдельно рассмотрим опции конфигурации ядра. Эта информация поможет вам адаптировать конфигурацию ядра для вашей конкретной железки, создав

собственные файлы конфигурации **config/hw/*.kernel**, **config/arch/*.kernel**, **config/cpu/*.kernel**.

Формирование результирующего конфигурационного файла см. **mk.rc** и секцию мейкфайла для пакета **kernel**.

8.10 Опции общие для всех вариантов сборки

8.10.1 rootfs в RAM

config/ramdisk.kernel

```
1 # boot with rootfs (initrd) in ram
2 CONFIG_BLK_DEV_INITRD=y
3 CONFIG_BLK_DEV_RAM=y
4 CONFIG_BLK_DEV_RAM_COUNT=1
5 CONFIG_DEVTMPFS=y
6 CONFIG_DEVTMPFS_MOUNT=y
7 CONFIG_PROC_FS=y
8 CONFIG_SYSFS=y
```

8.10.2 режим реального времени

config/realtim.e.kernel

```
1 # realtime options
2 CONFIG_PREEMPT=y
3 CONFIG_HZ_1000=y
4 CONFIG_SWAP=n
5 CONFIG_CC_OPTIMIZE_FOR_SIZE=n
```

8.10.3 часы и таймеры

config/clock.kernel

```
1 CONFIG_RTC_CLASS=y
2 CONFIG_RTC_HCTOSYS=y
3 CONFIG_RTC_SYSTOHC=y
4 # interfaces
5 #     interfaces
6 CONFIG_RTC_INTF_SYSFS=y
7 CONFIG_RTC_INTF_PROC=y
8 CONFIG_RTC_INTF_DEV=y
```

8.10.4 вывод printk и отладка

config/debug.kernel

```
1 # debug
2 CONFIG_PRINTK_TIME=y
3 CONFIG_MAGIC_SYSRQ=y
4 CONFIG_PRINTK=y
5 CONFIG_EARLY_PRINTK=y
6 CONFIG_SLUB_DEBUG=n
7 # sub
8 CONFIG_USB_ANNOUNCE_NEW_DEVICES=y
9 # panic
10 CONFIG_PANIC_TIMEOUT=0
11 # disable unneeded
12 CONFIG_SCHED_DEBUG=n
13 CONFIG_DEBUG_PREEMPT=n
```

8.10.5 форматы исполняемых файлов

config/binformats.kernel

```
1 # bin formats
2 CONFIG_BINFMT_ELF=y
3 CONFIG_CORE_DUMP_DEFAULT_ELF_HEADERS=y
4 CONFIG_BINFMT_SCRIPT=y
```

8.10.6 носители данных

config/storage.kernel

```
1 # disk drives and filesystems
2 CONFIG_BLOCK=y
3 CONFIG_LBDAF=n
4 CONFIG_BLK_DEV_BSG=y
5 CONFIG_PARTITION_ADVANCED=y
6 CONFIG_MSDOS_PARTITION=y
7 CONFIG_LDM_PARTITION=y
8 CONFIG_EFI_PARTITION=y
9 CONFIG_MAC_PARTITION=y
10 # block device drivers
11 CONFIG_BLK_DEV=y
12 CONFIG_BLK_DEV_LOOP=y
13 CONFIG_BLK_DEV_LOOP_MIN_COUNT=4
```

```
14 # USB mass storage
15 CONFIG_USB_STORAGE=y
16
17 ## Linux SoftRAID
18 #CONFIG_MD=y
19 #CONFIG_BLK_DEV_MD=y
20 #CONFIG_MD_AUTODETECT=y
21 #CONFIG_MD_RAID0=y
22 #CONFIG_MD_RAID1=y
23 #CONFIG_MD_RAID10=y
24 #CONFIG_MD_RAID456=y
25 #CONFIG_MD_FAULTY=y
26
27 # SCSI (emulation) layer
28 CONFIG_SCSI=y
29 CONFIG_SCSI_SCAN_ASYNC=y
30 # all Hard Disks & Flash drives
31 CONFIG_BLK_DEV_SD=y
32 # all CDROMs
33 CONFIG_BLK_DEV_SR=y
34
35 # IDE/SATA drives
36 CONFIG_ATA=y
37 CONFIG_ATA_SFF=y
38 CONFIG_ATA_BMDMA=y
39 # Legacy Generic PATA/SATA
40 CONFIG_ATA_GENERIC=y
41 CONFIG_PATA_LEGACY=y
42 # platform specific: copy it to $(HW) files
43 # Intel (S)ATA
44 #CONFIG_ATA_PIIX=y
45 # AMD/nVidia IDE on Athlon boards
46 #CONFIG_PATA_AMD=y
47 # JMicron IDE
48 #CONFIG_PATA_JMICRON=y
49 # Old Intel PIIX PATA IDE
50 #CONFIG_PATA_OLDPIIX=y
51 # PIO-only IDE on old ISA soundcards
52 #CONFIG_PATA_ISAPNP=y
53 # Intel MPIIX
54 #CONFIG_PATA_MPIIX=y
55 # embedded IDE systems
56 #CONFIG_PATA_PLATFORM=y
57
58 # Flash Cards MMC/SD cardreaders
```

```
59 CONFIG_MMCI=y
60 CONFIG_MMCI_BLOCK=y
61 # SDHC controllers wide used
62 CONFIG_MMCI_SDHCI=y
63 CONFIG_MMCI_SDHCI_PCI=y
```

8.10.7 файловые системы

config/filesystems.kernel

```
1 # file systems
2
3 # Ext / Reiser
4 CONFIG_EXT2_FS=y
5 CONFIG_EXT2_FS_XATTR=n
6 CONFIG_EXT3_FS=y
7 CONFIG_EXT3_FS_XATTR=n
8 CONFIG_EXT4_FS=y
9 CONFIG_REISERFS_FS=y
10 CONFIG_REISERFS_FS_XATTR=n
11
12 # CDRs
13 CONFIG_ISO9660_FS=y
14 CONFIG_JOLIET=y
15 CONFIG_UDF_FS=y
16
17 # FAT
18 CONFIG_MSDOS_FS=y
19 CONFIG_VFAT_FS=y
20 # NTFS
21 CONFIG_NTFS_FS=y
22 # looped fs image r/w
23 CONFIG_NTFS_RW=y
24
25 # misc
26 CONFIG_MISC_FILESYSTEMS=y
27 # Apple
28 CONFIG_HFS_FS=y
29 CONFIG_HFSPLUS_FS=y
30 # packaged r/o filesystems
31 CONFIG_SQUASHFS=y
32 CONFIG_SQUASHFS_FILE_DIRECT=y
33 CONFIG_SQUASHFS_4K_DEVBLK_SIZE=y
34 CONFIG_ZISOFS=y
35 # QNX
```

```
36 CONFIG_QNX4FS_FS=y  
37 CONFIG_QNX6FS_FS=y
```

8.10.8 интернационализация

config/i10n.kernel

```
1 # internationalization  
2 # compiled-in NLS  
3 CONFIG_NLS=y  
4 CONFIG_NLS_DEFAULT="koi8-r"  
5 CONFIG_NLS_CODEPAGE_437=y  
6 CONFIG_NLS_CODEPAGE_866=y  
7 CONFIG_NLS_CODEPAGE_1251=y  
8 CONFIG_NLS_ISO8859_1=y  
9 CONFIG_NLS_ISO8859_5=y  
10 CONFIG_NLS_KOI8_R=y  
11 CONFIG_NLS_MAC_CYRILLIC=y  
12 CONFIG_NLS_UTF8=y  
13 # ru FAT options  
14 CONFIG_FAT_DEFAULT_CODEPAGE=866  
15 CONFIG_FAT_DEFAULT_IOCHARSET="koi8-r"
```

8.10.9 мультимедиа

config/media.kernel

```
1 # audio/video  
2 CONFIG_MEDIA_SUPPORT=y  
3 CONFIG_MEDIA_CAMERA_SUPPORT=y  
4 CONFIG_MEDIA_USB_SUPPORT=y  
5 CONFIG_USB_VIDEO_CLASS=y  
6 CONFIG_USB_VIDEO_CLASS_INPUT_EVDEV=y  
7 CONFIG_USB_GSPCA=n
```

8.11 Опции для архитектуры i386

config/arch/i386.kernel

```
1 # arch: i386  
2 CONFIG_X86_GENERIC=y  
3 CONFIG_NOHIGHMEM=y  
4 CONFIG_X86_MCE=y
```

```
5 # buses
6 CONFIG_PCI=y
7 CONFIG_ISA=y
8 CONFIG_PNP=y
9 CONFIG_ISAPNP=y
10 # ACPI
11 CONFIG_ACPI=n
12 CONFIG_ACPI_PROCESSOR=y
13 CONFIG_ACPI_THERMAL=y
14 CONFIG_ACPI_FAN=y
15 CONFIG_ACPI_AC=y
16 CONFIG_ACPI_BATTERY=y
17 CONFIG_ACPI_BUTTON=y
18 CONFIG_INTEL_IDLE=y
19 # input/output layer
20 CONFIG_INPUT=y
21 CONFIG_TTY=y
22 CONFIG_VT=y
23 CONFIG_VT_CONSOLE=y
24 CONFIG_UNIX98_PTYS=n
25 CONFIG_LEGACY_PTYS=n
26
27 # serial system console & main external IO interface
28 CONFIG_SERIAL_8250=y
29 CONFIG_SERIAL_8250_DEPRECATED_OPTIONS=n
30 CONFIG_SERIAL_8250_CONSOLE=y
31 CONFIG_SERIAL_8250_NR_UARTS=4
32 CONFIG_SERIAL_8250_RUNTIME_UARTS=4
33 CONFIG_SERIAL_8250_EXTENDED=n
34 # parallel port i/o
35 CONFIG_PARPORT=y
36 CONFIG_PARPORT_PC=y
37 CONFIG_PARPORT_1284=y
38
39 # VGA 80x25 console
40 CONFIG_VGA_CONSOLE=y
41 # at/ps2 kbd support (main input)
42 CONFIG_INPUT_KEYBOARD=y
43 CONFIG_KEYBOARD_ATKBD=y
44 # USB HID kbd/mouse
45 CONFIG_HID=y
46 CONFIG_HID_GENERIC=y
47
48 # USB
49 CONFIG_USB_SUPPORT=y
```

```
50 #      USB client disabled
51 CONFIG_USB_GADGET=n
52 #      USB HOSTS
53 CONFIG_USB=y
54 #          USB 1.1 OHCI
55 CONFIG_USB_OHCI_HCD=y
56 #          Intel/VIA USB
57 CONFIG_USB_UHCI_HCD=y
58 #          USB 2 EHCI
59 CONFIG_USB_EHCI_HCD=y
60 #          USB 3 XHCI
61 CONFIG_USB_XHCI_HCD=y
62 #      Device Classes
63 #          Wireless WDM/AT modems
64 CONFIG_USB_WDM=y
65 CONFIG_USB_SERIAL_OPTION=y
66 #CONFIG_USB_SERIAL_QUALCOMM=y
67 #      Mass Storage s
68 # see config/storage.kernel
69 #      Serial Converters
70 CONFIG_USB_SERIAL=y
71 CONFIG_USB_SERIAL_CONSOLE=y
72 CONFIG_USB_SERIAL_GENERIC=y
73 CONFIG_USB_SERIAL_CP210X=y
74 CONFIG_USB_SERIAL_FTDI_SIO=y
75 CONFIG_USB_SERIAL_PL2303=y
76 #CONFIG_USB_SERIAL_IR=y
77 #CONFIG_USB_CYPRESS_CY7C63=y
78 #CONFIG_USB_EZUSB_FX2=y
79
80 # CMOS/hwclock
81 CONFIG_NVRAM=y
82 CONFIG_RTC_DRV_CMOS=y
83
84 # mouse
85 CONFIG_INPUT_MOUSE=y
86 CONFIG_INPUT_MOUSEDEV=y
87 CONFIG_INPUT_MOUSEDEV_PSAUX=n
88 CONFIG_MOUSE_PS2=y
89 CONFIG_MOUSE_SERIAL=y
90
91 # video drivers
92 CONFIG_FB=y
93 CONFIG_FRAMEBUFFER_CONSOLE=y
94 #CONFIG_FIRMWARE_EDID=y
```

```
95 CONFIG_FB_VESA=y
96 CONFIG_LOGO=y
97 CONFIG_LOGO_LINUX_MONO=y
98 CONFIG_LOGO_LINUX_VGA16=n
99 CONFIG_LOGO_LINUX_CLUT224=n
100
101 # sound
102 CONFIG_SOUND=y
103 # ALSA
104 CONFIG_SND=y
105 CONFIG_SND_SUPPORT_OLD_API=n
106 CONFIG_SND_VERBOSE_PROCFS=n
107 # MIDI
108 CONFIG_SND_DRIVERS=n
109 #CONFIG_SND_SEQUENCER=y
110 #CONFIG_SND_VIRMIDI=y
111 #CONFIG_SND_SERIAL_U16550=y
112 # ISA cards
113 CONFIG_SND_ISA=n
114 # Adlib
115 #CONFIG_SND_ADLIB=y
116 # PCI cards
117 CONFIG_SND_PCI=n
118 # integrated AC97
119 #CONFIG_SND_INTEL8X0=y
```

8.11.1 опции процессора i486

config/cpu/i486.kernel

```
1 # cpu: i486sx
2 CONFIG_M486=y
3 CONFIG_MATH_EMULATION=y
```

8.11.2 опции для эмулятора qemu386

config/hw/qemu386.kernel

```
1 # hw: qemu386
2 # disable realtime
3 CONFIG_HZ_100=y
4 # disable FPU emulation
5 CONFIG_MATH_EMULATION=n
6 # disable unsupported generic i386 hw
```

```
7 CONFIG_PARPORT=n  
8 # enable Adlib sound  
9 CONFIG_SND_ISA=y  
10 CONFIG_SND_ADLIB=y
```