SDCC-NOINIT

Алексей, aka SfS

Автоматизированная система сборки проектов для SPECCY

Как это началось?

Итак, я «изобрел велосипед». Очередной «велосипед», коих миллионы изобретены разными разработчиками. Зачем я это сделал? Да затем же, зачем и остальные — для упрощения жизни себя любимого.

Попробую рассказать, зачем я это сделал и как все случилось.

Каждый разработчик проходит несколько стадий, которые так или иначе отражаются на его развитии. И я — не исключение. Разумеется, эти стадии могут отличаться по внешнему облику, но суть у них примерно одна.

Стадия первая - «ура, я это могу!». Горящие глаза и первая программа уровня «Привет, Мир!». Обычно проходится в детстве или юности. На ZX эта стадия проходится обычно на великом Бейсике.

Затем следуют потуги в написании программ посложнее, освоение INKEY$, IN, OUT и так далее.

На данной стадии все кажется таким простым легким и прекрасным, что написание комментарием, структура программы и прочие подобные вещи кажутся ненужной глупостью, которую придумали злые люди, чтобы мешать полету мысли юного дарования.

Стадия вторая — «хочу написать игру!». После того, как освоены некоторые возможности ПК и «юное дарование» перестаёт путаться в том, чем отличается строка от числа и порт от ячейки памяти — приходит желание написать игру! Неважно, что это за игра — шахматы, стратегия или язык программирования. Все равно — это игра! То есть поделие для души, а не с целью делового использования.

И тут приходят первые прозрения и разочарования. Как правило, объём такой программы, даже если это SpaceIntruders — достаточно велик, чтобы было сложно запомнить все переменные, особенности алгоритма и всякие сложные программные ветви. Становится, наконец, понятным, что комментарии — это не выдумка бездушных формалистов, а очень полезная вещь; что процедурное программирование — это очень удобно по сравнению с огромными простынями программного кода, которые так любят писать начинающие. В общем - «юное дарование» немного взрослеет.

На дальнейших стадиях, которые могут выглядеть очень по-разному внешне, спектрумист, обычно, осваивает ассемблер пробует Мега-Лазер-Бета-бейсики, Паскаль, Форт или ещё что-нибудь. Но, в итоге — редко находит то, что ему подходит идеально.

В современном веке, как бы это ни было ужасно для ретроградов, практически все пишут на кросс-средствах.

Как бы не двигался путь развития разработчика программ на спектруме — в итоге он приходит к тому, что ему нужна универсальная и гибкая автоматизированная система сборки программ.

Я пришел к этому давно, но к сожалению, как и многие, давно уже не школьники — не мог взяться за свою систему сборки толком до этого года. Но, если чего-то очень хочется — то оно непременно получается. Рано или поздно. Так или иначе.

Что же такое SDCC-NOINIT?

Итак, что такое «автоматизированная среда сборки программ»? Попробую пояснить.

Когда вы пишите ПО, то волей или не волей у вас накапливается набор процедур, подпрограмм, скриптов и всякого инструментария и полуфабрикатов, которые вы используете для работы.

Если все это хранится, как часто бывает, бессистемной грудой файлов, распиханной по разным каталогам — то вы рискуете делать примерно следующие вещи:

- с криком хватаясь за волосы на голове (лысые — за волосы на заду) вы ищите какую-нибудь нужную утилиту или процедуру, которая «Ну вот же! была год назад! Точно помню! Блин, да где ж она! АААААА!»;
- ругаете по папе, маме и бабушке с дедушкой себя, кота, жену, любовницу и прочих, кого вспомните;
- лезете в интернет искать нужную вам вещь, если она там есть;
- пишите заново процедуру, если она уникально ваша, но найти её вы не могли.

Понятно, что такой подход непродуктивен. Да и валерьянка нонче не дешева, а в мире кризис.

Чтобы облегчить жизнь себе и тем, кто вдруг захочет разобраться и использовать ваши «творения», необходимо, чтобы было понятно — какие и где утилиты лежат; рассортировать исходные тексты библиотек и программ, чтобы легко было найти нужную библиотеку или программу; и, наконец, разработать единый алгоритм для добавления и добавления программ и библиотек, понятный всем. Кроме того, все исходники необходимо хранить в системе контроля версий — в моём случае это GIT.

Тогда сразу, чудесным образом, исходники программ, библиотек и утилиты перестанут теряться, а волосы на теле останутся в целости.

Подходов тут много, можно спорить о том, что лучше и что хуже до старости. Но что сделано — то сделано. Чтобы экономить время, деньги на валерьянку и не костерить ни в чем ниповинных котов и жен, я поступил следующим образом.

Во-первых, все необходимые утилиты я объединил в один пакет. Благо эти утилиты доступны в виде исходных текстов. Таким образом, я не ищу утилиты, а просто беру этот пакет исходников, компилирую его и получаю набор конвертеров, компиляторов, утилит работы с образами и картинками. Назвал я эту чушь speccy-toolchain (https://github.com/salextpuru/speccy-toochain). Если мне нужна новая утилита — то я просто добавляю её в набор, но никогда ничего не удаляю. Что делать, во мне живёт внутренний Плюшкин:) Зато ничего не теряется.

Во-воторых, я положил в основу своего программизма на спектруме нелюбимый многими и обожаемый некоторыми кросс-компилятор языка С SDCC. Взять его можно на официальном сайте http://sdcc.sourceforge.net/snap.php#Source.

И в-третьих — я широко использую стандартные UNIX-Linux утилиты, такие как make, bash и прочие. Те, кто живёт под Windows (хотя что это за жизнь?) - могут насладиться использованием всех прелестей UNIX-Linux утилит, установив себе Cygwin (https://cygwin.com/install.html).

Что я получил в итоге? В итоге, я получил так называемую «сборочную среду». То есть набор программ, которые я могу использовать для сборки своих исходных текстов, подготовки графики и так далее. Этот набор я могу относительно быстро и просто установить на почти любой компьютер и работать на нём. Причем, ничего не потеряется и умрет — все доступно в инете.

Все что я описал — это только средства. А где же тут мои программы? И при чем тут автоматизация? Ну вот теперь и об этом.

Все исходные тексты и скрипты bash я упорядочил в системе каталогов, написал сборочные скрипты для make и назвал все это SDCC-NOINIT (https://github.com/salextpuru/sdcc-noinit).

Чем удобна для меня моя система SDCC-NOINIT ?

Попробую перечислить:

- жесткая упорядоченность по каталогам: не надо гадать где искать библиотеки, скрипты, программы или документацию - все очевидно из названий каталогов :);
- можно легко и просто добавить или выключить из сборки любую программу или библиотеку;
- можно использовать разные версии одной и той же библиотеки, переключая их конфигурацией;
- можно собирать сразу множество программ для ZX с разными конфигурациями памяти (расположены по разным адресам, с разным размером стека и так далее).

Кому нужна SDCC-NOINIT?

Как минимум — она нужна мне. Или тому, кому неинтересно писать в 100500й раз с нуля вывод символа на экран, простого спрайта, музыкальный плеер pt3 и stc или ещё какой-то велосипед. Многое в **SDCC-NOINIT** уже есть. Пусть не идеальное, не самое быстрое — но, есть! Поэтому тот, кто хочет проверить свою идею — может взять **SDCC-NOINIT**, быстренько накидать свою программу (примеры там есть) и заниматься проверкой своей идеи, а не добыванием руды для изготовления первого топора. И, если его идея, например, новой игры, ему понравится — то тогда он может потихоньку оптимизировать свои процедуры, переписывая их на ассемблере, если это надо или разработав новую библиотеку. Если кто-то будет добавлять свои библиотеки или программы — я буду только рад. Так что берите и смотрите.

Как установить SDCC-NOINIT?

Современный мир дает много возможностей. Лет 30 назад никто и не представить не мог, что можно будет находясь в лесу, купить билет на самолёт, вылетающий через полгода. А теперь этим никого не удивить.

То же и со средствами разработки. Обилие программ, утилит, конвертеров и прочих средств настолько велико, что трудно найти нужное.

Поскольку я не маньяк и писать все с нуля не хочу, не могу и не буду — то я по максимуму пользуюсь тем, что уже написано благородными людьми и отдано публике.

Поэтому я предваряю сей труд описанием того, какие именно программы и продукты нам понадобятся для работы с SDCC-NOINIT.

Итак, **прежде всего** — это утилиты такие как **bash** и **make**. Если вы счастливый обладатель Linux — то у вас проблем с этим нет. Но если вы несчастный и потерянный заложник Windows — то вам необходимо поставить CygWin, который позволит вам пользоваться юникс-утилитами в виндовс. Взять его можно на официальном сайте: https://cygwin.com/install.html.

Второе, что нам понадобиться — это компилятор GCC. И так же — для Linux он обычно есть в системе, а для Windows — в CygWin.

Третье — это набор утилит, который вы соберете с помощью компилятора GCC. Это специфичные спектрум-утилиты. Для удобства я собрал их в один «типа-пакет» **speccy-toochain**, который можно взять с GitHub и откомпилировать: https://github.com/salextpuru/speccy-toochain.

Четвертое, что нам необходимо — это компилятор SDCC. Тут все просто: идем на его сайт http://sdcc.sourceforge.net/.

Далее - или качаем бинарную сборку https://sourceforge.net/projects/sdcc/files/ или исходники http://sdcc.sourceforge.net/snap.php#Source и собираем сами.

Ну и, наконец, **пятое**, самое последнее — это собственно SDCC-NOINIT. Находится она все на том же GitHube, как раз рядышком с коллекцией исходников спектрум-утилит: https://github.com/salextpuru/sdcc-noinit.

Скачали? Поставили ? Зайдите в корень SDCC-NOINIT и наберите:

```
# make
```

Если у вас все верно (все утилиты есть, пути к ним прописаны и т. п.), то после сборки в каталоге **bin** появятся программки и плагины WC.

Можете их посмотреть, запуская на спектруме или (фи!) его эмуляторе.

Ну а если что-то не пошло — то думайте, чего вам не хватает и исправляйте. Какой интерес жить, не решая проблем ? :)

Пример простой программы

Как и всегда в подобных случаях, покажу, как написать свой «Hello, world!» с использованием **SDCC-NOINIT**.

Если вы успешно и без ошибок скомпилировали примеры, то всё у вас получится.

Итак, заходим в каталог **apps** и создаём каталог программы **hello**.

```
# mkdir hello
```

Далее, любым доступным способом копируем из каталога с примером example-0 во вновь созданный каталог hello три фала: **Makefile**, **build.mk** и **config.mk**.

Создаём в каталоге **hello** файл с программой **hello.c**, содержимое которого приведено ниже:

```
#include <conio.h>
void main(){
        ccls(6);
        printf("Hello world!");
        while(1){}
}
```

Теперь в каталоге **apps/hello** имеются файлы: **hello.c**, **Makefile**, **build.mk** и **config.mk**.

Открываем файл **build.mk** и добавляем в него имя программы и файлы, из которых она состоит. Файл **build.mk** выглядит примерно так:

```
# App name (Имя программы)
APP=hello

# Object files (Список объектных файлов. Заметьте, что система
#       сама разбирается из какого исходного файла создавать объектник)
OBJ=hello.rel
```

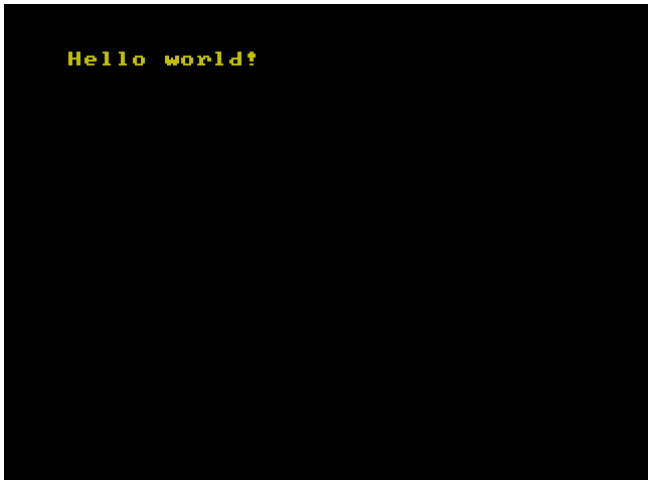
И, наконец, добавляем наш проект в систему сборки. Открываем файл **configs/apps.mk** и добавляем в его конец строчку:

```
# Hello
APPLICATIONS+=hello
```

Всё! Теперь осталось скомпилировать наш проект. Выходим в корневой каталог **SDCC-NOINIT** и наберите:

 # make

Если у вас все верно, то после сборки в каталоге **apps/hello** появятся файлы с нашей программой: **hello-basic.tap**, **hello.$C** и **hello-code.tap**. Это наша программа в разных форматах. При её запуске, мы увидим следующую красоту:

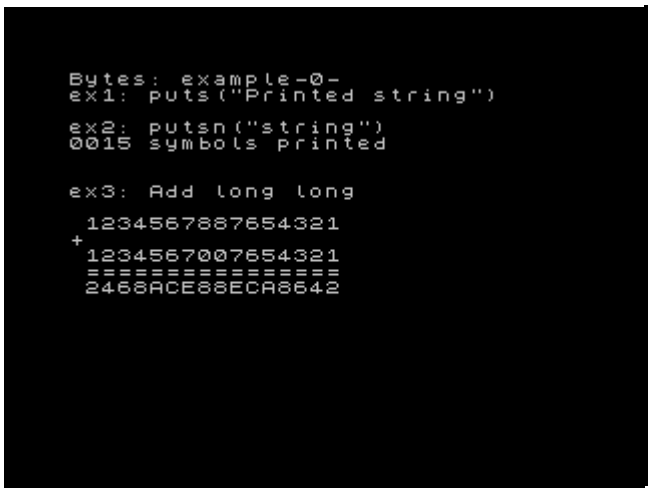


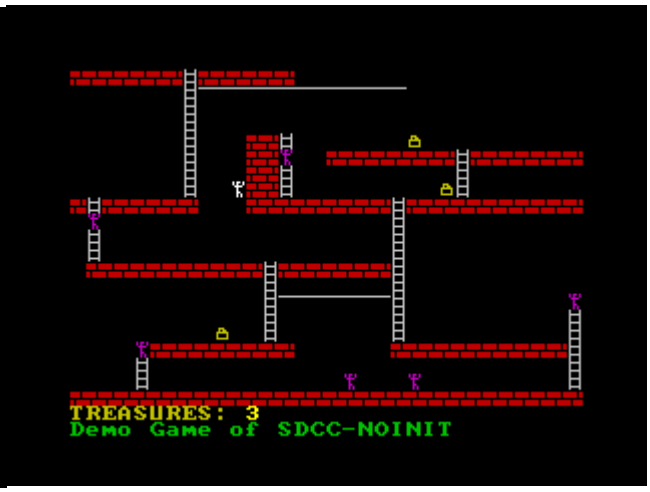
Поздравляю! Вы теперь дипломированный специалист по SDCC-NOINIT!

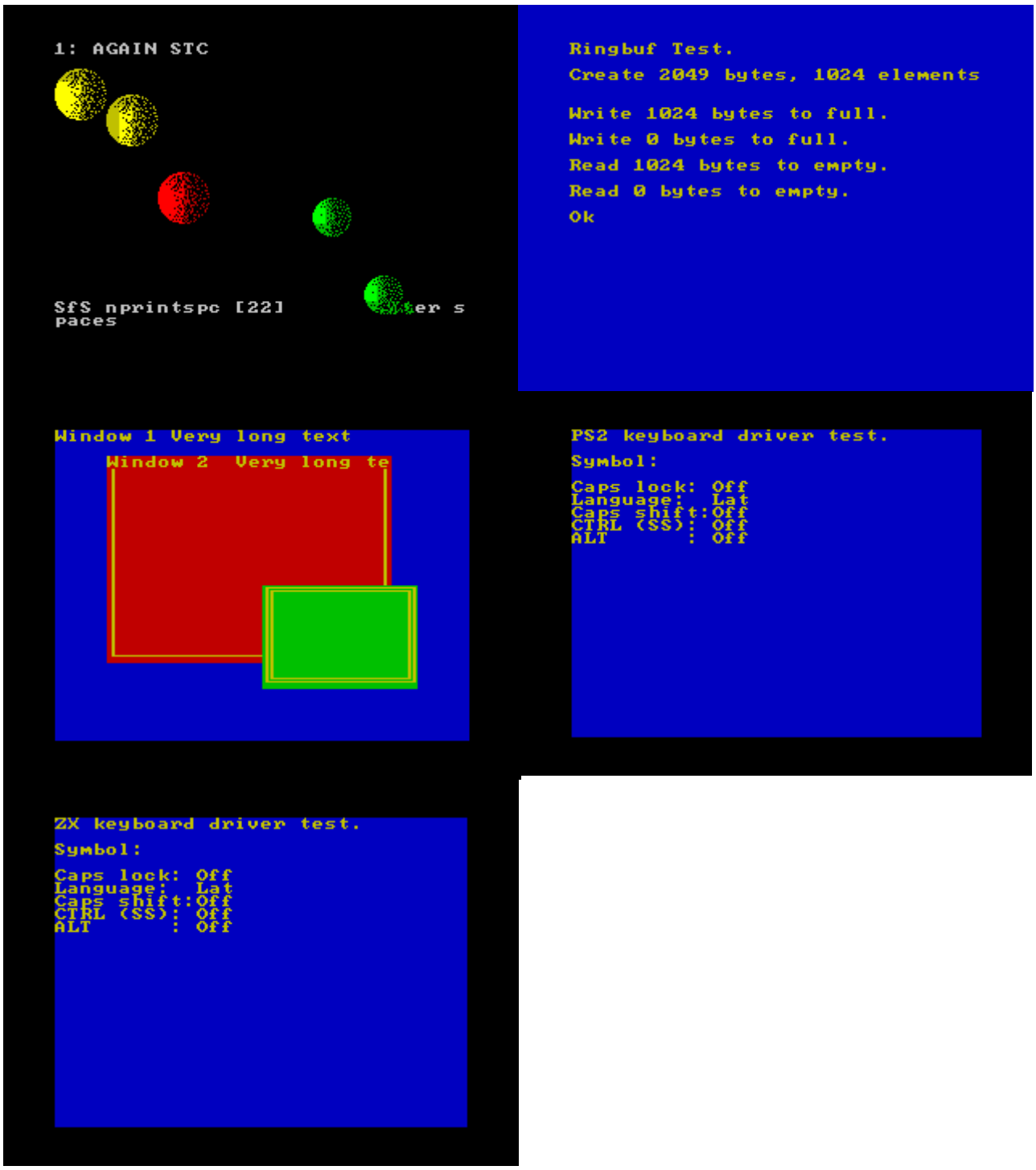
Заключение

Вроде бы написал немного, а уже вышло несколько страниц. Понятно, что подробно о конфигурации и прочем - писать в короткой статье не получится. Кому интересны подробности — загляните в каталог **sdcc-noinit/doc**. Там вы увидите довольно обширный, но всё ещё не оконченный документ sdcc-noinit.pdf, в котором есть много интересного не только о самой системе сборки, но и о компиляторе SDCC.

И, чтобы внести немного красок, несколько картинок с экранами примеров, которые есть в **SDCC-NOINIT**.







Всегда рад ответить на ваши вопросы на форуме **http://zx-pk.ru/forum.php**. Если вопрос простой и мелкий — можно в личку. Если глобальный и большой — то создавайте тему:)

Алексей, aka SfS