**tnylpo**(1) **tnylpo**(1)

**ИМЯ**

**tnylpo** - утилита для запуска программ, написанных для CP/M-80

**СИНТАКСИС**

**tnylpo [-abnrsw] [-c (<n> | @)] [-d <drive>] [-f <config-file>] [-l**

**(<n> | @)] [-t (<n> | @)] [-v <level>] <command> [<arg> ...]**

**tnylpo -h**

**ОПИСАНИЕ**

Утилита **tnylpo** позволяет выполнять программы, написанные для операционной системы CP/M-80 в среде Unix. Чтобы выполнить эту задачу, **tnylpo** запускает эмуляцию восьмибитового процессора Zilog Z80, которая на программном уровне переводит вызовы интерфейса прикладного программирования CP/M (вызовы BDOS и BIOS версии 2.2 CP/M-80) в библиотечные и системные вызовы Unix.

**tnylpo** даже не пытается воссоздать внешний вид операционной системы CP/M-80, эмулируя ее консольный командный процессор (CCP) или предоставляя реализацию служебных программ, таких как PIP или STAT. На самом деле, ее цель состоит в максимальной интеграции приложений и компиляторов CP/M-80 в интерфейс командной строки Unix, чтобы позволить их использование в сочетании с гораздо более мощными утилитами и редакторами операционной системы хоста. Пользователи **tnylpo** должны использовать cp(1) и dd(1) вместо PIP, chmod(1), df(1) и ls(1) вместо STAT, а также сценарии оболочки и перенаправления вместо SUBMIT и XS.

Файлы в среде CP/M напрямую отображаются в файлы Unix. Хотя это достигается ценой некоторых незначительных несовместимостей с CP/M-80 (см. ниже. ***Примечания***), это значительно упрощает взаимодействие между программами CP/M и Unix по сравнению с альтернативным подходом использования образов дисков.

Благодаря своему уровню трансляции системных вызовов, утилита **tnylpo** не содержит никакого кода из операционной системы CP/M и не нуждается в каком-либо дополнительном программном обеспечении, кроме реальных приложений, для выполнения которых она предназначена.

Поскольку программы CP/M-80 могут использовать в лучшем случае восьмибитные символы (многие из них даже разделяют твердую веру CP/M в семибитный мир ASCII), другой уровень трансляции является посредником между внутренним однобайтовым представлением текстового символа и почти универсальным многобайтовым символом (чаще всего на основе UTF-8) современных систем Unix. Этот слой трансляции влияет только на символьно-ориентированные каналы ввода/вывода CP/M и может быть отключен для принтеров, перфораторов и устройств чтения.

Все экранно-ориентированные программы CP/M-80 не способны справиться с возможностью графических пользовательских интерфейсов изменять размеры текстовых окон (фактически даже способность DEC VT100 переключаться между режимами столбцов 80 и 132 перегружает их). Чтобы решить эту проблему (и обеспечить возможность запуска одного и того же приложения CP/M в другом терминале эмуляции или на последовательных терминалах без перенастройки двоичных файлов CP/M), **tnylpo** включает в себя основанную на curses эмуляцию терминала DEC VT52 (с некоторыми расширениями функциональности) для использования в качестве консольного устройства CP/M. Утилита **tnylpo** предоставляет пользователю выбор между этой эмуляцией терминала и строчно-ориентированным консольным устройством, более подходящим для таких программ, как ассемблеры и компиляторы.

Верная своей цели обеспечения интеграции программ CP/M в среду Unix, **tnylpo** благополучно избавляется от некоторых несущественных, редко используемых или крайне не свойственных для Unix функций CP/M-80 (некоторые из которых, откровенно говоря, необоснованно сложно реализовать или интегрировать в Unix), среди них в том числе области пользователя, перенаправление на символьные устройства, реализуемое байтом ввода-вывода, атрибуты файла и эхо на принтер. Несмотря на то, что некоторые функции были упущены, предприняты меры, чтобы не обидеть программы CP/M (например, программы могут установить область пользователя, и по запросу **tnylpo** сообщит им текущее значение, но проигнорирует его в противном случае), Как правило, **tnylpo** или Unix предлагают лучшие решения в качестве замены нереализованных функций CP/M, например, различные конфигурации **tnylpo** (и диски/каталоги) для разных приложений или пользователей (области пользователя), перенаправление вывода оболочки (байт ввода/вывода), chmod(1) (атрибуты файла) или script(1) (протокол консоли).

**Эмуляция процессора**

Утилита **tnylpo** стремится обеспечить полную реализацию как документированных, так и недокументированных возможностей процессора Z80. Недокументированные инструкции и побочные эффекты поддерживаются в меру знаний автора.

Инструкции ввода/вывода доступны, но являются пустышками. Запись в порты ввода/вывода не оказывает прямого эффекта, а чтение из них всегда возвращает нулевой байт. То же самое верно для всех команд, связанных с прерываниями (в то время как регистр I может быть установлен и прочитан, а инструкции EI и DI имеют ожидаемые побочные эффекты, прерывание никогда не произойдет). Инструкция HALT не останавливает процессор (поскольку нет прерываний, это остановит эмуляцию), но действует так, как если бы NMI был получен сразу после входа в состояние остановки. (Приложения CP/M с хорошим поведением в любом случае не будут использовать ни одну из этих инструкций.)

Регистр **R** будет увеличен, как и ожидалось. Поскольку прикладные программы могут использовать текущее значение регистра **R** для генерации случайных чисел (процедура Randomize Turbo Pascal, похоже, делает это), регистр **R** инициализируется случайным значением (в противном случае повторные запуски одной и той же программы всегда будут генерировать одну и ту же последовательность случайных чисел).

Вся неиспользуемая память и остальные регистры в начале эмуляции устанавливаются равными нулю, а PC, SP и остальная часть основной памяти (кроме областей, затронутых аргументами командной строки, параметрами конфигурации и загрузкой исполняемого файла) всегда инициализируются одинаково.

**Преобразование имени файла**

При переводе дисковых операций CP/M в дисковые операции Unix утилита **tnylpo** должна преобразовать имена файлов CP/M в имена файлов Unix и наоборот. Поскольку CP/M де‐факто в своих именах файлов допускает почти произвольные семибитные символы (включая нулевые байты и символы косой черты), а имена файлов Unix не привязаны к ограничению символов CP/M 8+3, **tnylpo** применяет общее подмножество: в именах файлов разрешены только десятичные цифры, 26 букв алфавита, знаки минус (‐), at (@), доллара ($) и решётки (#), которые ограничены длиной от одного до восьми символов. Имя может иметь расширение до трех символов из этого набора. Точка между именем и расширением может присутствовать только в том случае, если имеется хотя бы один символ расширения. На стороне Unix все буквы отображаются в нижнем регистре, в то время как **tnylpo** принимает (и возвращает) в CP/M FCB только буквы верхнего регистра. Файлы Unix с именами, содержащими прописные буквы или иным образом не соответствующие ограничениям **tnylpo**, недоступны программам CP/M. (Знаки доллара в именах файлов Unix являются помехой в сценариях оболочки, но поскольку они обычно используются в именах временных файлов CP/M, **tnylpo** не может запретить их. К счастью, программы CP/M, создающие файлы с именами файлов, содержащие знаки доллара, обычно удаляют их перед выходом.)

**Опции**

Многие опции командной строки **tnylpo** имеют соответствующие записи в файле конфигурации, поэтому они будут обсуждаться в этом контексте ниже (см. ***Параметры конфигурации***). Три опции не имеют аналогов:

**-a** Выбирает альтернативный набор символов из файла конфигурации перед началом выполнения (см. ***Наборы символов***)

**‐f** <файл-конфигурации>

предписывает программе **tnylpo** прочитать свою конфигурацию из указанного файла

**‐h** просит **tnylpo** показать краткий обзор командной строки и выйти (‐h не может использоваться в сочетании с любой другой опцией командной строки)

**Аргументы командной строки**

Утилита **tnylpo** интерпретирует свой первый аргумент командной строки как имя программы CP/M (файл .com), которую она должна выполнить. Если этот аргумент содержит символ косой черты (/), он интерпретируется как путь Unix, в противном случае он обрабатывается как имя файла CP/M, т. е. это может быть префикс спецификации диска и интерпретируется как относительный путь, соответствующий этому диску или диску по умолчанию согласно конфигурации **tnylpo** (см. ниже). В обоих случаях добавляется суффикс .com, если имя файла еще не заканчивается .com.

Все аргументы расположенные далее передаются в программу CP/M как аргументы командной строки (т.е. они преобразуются в верхний регистр и копируются в область прямого доступа по умолчанию в память по адресу 0x0080. Кроме того, FCB по умолчанию, расположенный по адресу 0x005c, инициализируется согласно второму и третьему аргументам).

**Конфигурационные файлы**

Хотя некоторыми функциями утилиты **tnylpo** можно управлять с помощью параметров командной строки, наиболее удобный способ настроить ее - использовать файл конфигурации. Если файл конфигурации явно указан в командной строке (опция **‐f**), он будет использоваться, в противном случае **tnylpo** ищет файл с именем **.tnylpo.conf** в текущем рабочем каталоге. Если ничего не найдено, **tnylpo** ищет файл **.tnylpo.conf** в домашнем каталоге пользователя. В качестве последнего средства, **tnylpo** использует свои встроенные настройки по умолчанию. Если в файле конфигурации и в командной строке указаны конфликтующие параметры, командная строка имеет приоритет.

Файл конфигурации **tnylpo** - это обычный текстовый файл. Пустые строки и строки, начинающиеся со знака решётки (#) или точка с запятой (;) игнорируются. Все остальные строки имеют вид:

**<ключевое слово> [<token> ...] = <token> <token> ...]**

**<token>** - это либо ключевое слово (последовательность буквенно-цифровых символов начиная с буквы), число (шестнадцатеричное, восьмеричное или десятичное с использованием обычного соглашения Unix с префиксом 0x, 0, соответственно несколько последующих цифр), строка в двойных кавычках или запятая.

**Опции конфигурации**

**drive <буква диска> = [readonly ,] <путь>**

При повторном использовании этой опции конфигурации можно определить до 16 накопителей. **<буква диска>** - это одна строчная буква в диапазоне a – p, а **<путь>** - строка, содержащая имя каталога на хосте компьютерной системы. Программы CP/M, пытающиеся создать или получить доступ к файлу на соответствующем диске, создадут или получат доступ к файлу в этом каталоге. Только обычные файлы размером до 8 МБ с именами, соответствующими правилу **tnylpo** о хороших именах файлов, подходящих как для CP/M, так и для Unix (см. выше), видны программам CP/M. Если полю **<путь>** предшествует необязательное ключевое слово **readonly**, программы, работающие в **tnylpo**, не смогут создавать новые файлы на этом диске или переименовывать, удалять или изменять существующие файлы (любая попытка изменить диск только для чтения завершит программу CP/M).

Соответствующий параметр командной строки отсутствует. Если в файле конфигурации **drive** (накопитель) не определен (или отсутствует файл конфигурации), **tnylpo** по умолчанию будет использовать

**drive a = "."**

т.е. текущий рабочий каталог будет доступен CP/M как диск A.

**default drive = <буква диска>**

опция командной строки ‐d <буква диска>

определяет диск, обозначенный в поле <буква диска> (одна строчная буква в диапазоне a – p) в качестве диска по умолчанию, т. е. все спецификации файлов, не включающие явное имя диска ссылаются на него. Этот диск должен быть назначен каталогу хост-системы неявно или с помощью параметра конфигурации **drive**. Если диск по умолчанию не указан, **tnylpo** использует диск A в качестве диска по умолчанию.

**close files = (true | false)**

опция командной строки **‐n**

Если **close** **files** имеет значение false (или если присутствует опция командной строки **‐n**), файлы, закрытые программой CP/M, остаются открытыми **tnylpo**, т. е. соответствующие FCB не являются недействительными. Это требуется некоторым программам CP/M (см. ***Закрытие файлов***), но по возможности этого следует избегать, поскольку в противном случае **tnylpo** может исчерпать файловые дескрипторы. По умолчанию **tnylpo** фактически закрывает файлы, закрытые программой CP/M.

**logfile = <путь>**

**<путь>** - строка в кавычках, содержащая путь к файлу, к которому **tnylpo** добавляет сообщения об ошибках и другую информацию о регистрации (фатальные сообщения об ошибках также пишутся в stderr). Если опция **logfile** (файл журнала) не используется, информация о регистрации отсутствует и не будет записана. Соответствующая опция командной строки отсутствует.

**loglevel = <уровень>**

Объем данных, записанных в файл журнала, контролируется параметром конфигурации журнала **loglevel**. Его эквивалент командной строки **-v**. Оба принимают числовой аргумент. Чем больше число, тем больше информации записывается (что приводит к постепенному замедлению эмуляции). Допустимые уровни журнала:

0 записываются только сообщения об ошибках.

1 дополнительно подсчитает машинные инструкции, выполненные эмулятором. При завершении программы, **tnylpo** выведет таблицы, показывающие, какие инструкции и как часто были выполнены.

2 дополнительно, позволяет проследить функции FDOS (т.е. функции BDOS, связанные с файловым вводом/выводом).

3 дополнительно отобразит дамп FCB для функций FDOS, использующих FCB.

4 дополнительно, создаст записи файла дампа, прочитанные и записанные функцией FDOS.

5 дополнительно проследит все остальные системные вызовы (BDOS и BIOS функции). Так как отслеживаются все символьные функции ввода/вывода, это даст много выхода.

Средство ведения журнала является пережитком после разработки и отладки самого **tnylpo**. Поскольку оно может дать важные подсказки, когда приложения не работают должным образом, оно было сохранено.

**console = (full | line)**

параметры командной строки **‐s** или **‐b**

предписывает **tnylpo** использовать полноэкранную эмуляцию VT52 (**full**, **‐s**) или строчно-ориентированный (**line**, **‐b**) интерфейс консоли. Несколько других опций конфигурации, такие как, **lines**, **columns**, **application cursor** и **screen delay** действуют только в режиме **full**. По умолчанию **tnylpo** использует строчно-ориентированный интерфейс консоли.

**screen delay = (<число> | клавиша)**

опция командной строки **‐t** (<число> | **@**)

определяет количество секунд, которые **tnylpo** должен ждать между завершением программы и сбросом дисплея. Если указана клавиша (соответственно **@**), **tnylpo** ожидает нажатия клавиши, прежде чем выйти из эмуляции VT52. Эта опция позволяет пользователю увидеть окончательный экран приложения CP/M, даже если сброс настроек дисплея восстанавливает исходное содержимое экрана или очищает экран.

**lines = (<число> | current)**

**columns = (<число> | current)**

Параметры команды -l (<число> | @) и ‐c (<число> | @)

определяют размер экрана, используемого эмуляцией терминала. Количество строк должно быть от 5 до 95, количество столбцов-от 20 до 95. Использование ключевого слова **current** (соответственно @ в случае параметров командной строки) указывает **tnylpo** использовать текущий размер устройства отображения. Если размер файла не указан в файле конфигурации или в командной строке, по умолчанию **tnylpo** использует 24 строки с 80 столбцами.

**application cursor = (true | false)**

опция командной строки **‐w**

Если для **application cursor** установлено значение **true** (истина) (или указана опция командной строки **‐w**), нажатие клавиш курсора вверх, влево, вправо или вниз отправит работающей программе CP/M управляющие символы ^E (0x05), ^S (0x13), ^D (0x04) или ^X (0x18) (т.е. соответствующие команды перемещения курсора для таких программ, как WordStar или Turbo Pascal). В противном случае клавиши курсора будут генерировать escape-последовательности VT52 по умолчанию, <esc>A, <esc>D, <esc>C и <esc>B. Этот параметр действует только в полноэкранном режиме консоли.

**exchange delete = (true | false)**

опция командной строки **‐r**

Если для **exchange delete** задано значение **true** (истина) (или в командной строке указана опция **‐r**), в полноэкранном режиме клавиша возврата (^H, 0x08) и клавиша удаления (0x7f) меняются местами.

**[alt] char <number> = <string>**

**[alt] charset = (ascii | vt52 | latin1 | tnylpo)**

**unprintable = <string>**

служат для определения основных и альтернативных наборов символов, используемых **tnylpo**. Они не имеют соответствующих параметров командной строки и описаны ниже (см. ***Наборы символов***).

**(printer | punch | reader) file = <path>**

**(printer | punch | reader) mode = (text | raw)**

определяют путь и формат файлов данных, представляющих символьные устройства ввода-вывода CP/M LST:, PUN: и RDR:. Соответствующие параметры командной строки отсутствуют. Подробности описаны ниже (см. ***Символьные устройства***).

**Эмуляция терминала**

**tnylpo** предоставляет основанную на curses эмуляцию терминала DEC VT52, которую можно использовать вместо стандартной линейно-ориентированной консоли для размещения полноэкранных приложений. Эта эмуляция терминала выбирается параметром командной строки **-s** соответствующей записи: console = full в файле конфигурации.

Эта эмуляция терминала довольно точно имитирует VT52, но предлагает ряд расширений, среди которых возможность поддерживать размеры экрана до 95 на 95 символов (это ограничение связано с ограничениями команды прямого позиционирования курсора VT52), восьми разрядные операции, динамически переключаемый набор альтернативных символов, команды вставки и удаления строки, а также выделение жирным, подчеркнутым, инвертированным и мигающим (т. е. всеми любимые) отображения символов. Чтобы защитить приложение CP/M (или его пользователя) от эффектов изменения размера экрана, эмуляция терминала обеспечивает область экрана фиксированного размера (обычно 80 столбцов на 24 строки, но она может быть изменена опциями командной строки или конфигурационного файла) внутри фактического дисплея (эмулятор терминала, такой как xterm (1) или экран фактического последовательного терминала). Если устройство отображения / окно больше этой области, справа и под областью отображения VT52 будут пустые поля. Если он меньше, часть вывода из эмулятора будет невидима для пользователя, но появятся (повторно), как только окно будет увеличено.

Эмуляция терминала (как и VT52) не выполняет автоматический перенос строки (т. е. курсор не перемещается в первый столбец следующей строки, если символ отображается в последнем столбце строки) и поддерживает (или, по крайней мере, допускает) все управляющие последовательности VT52:

**<bel>** (0x07)

дает акустический (или визуальный) сигнал

**<bs>** (0x08)

перемещает курсор влево, останавливается на столбце 1

**<tab>** (0x09)

если курсор находится непосредственно перед позицией табуляции (позиции табуляции находятся в столбцах 9, 17, 25, ...), перемещает его на следующую позицию табуляции, в противном случае перемещает курсор вправо на один столбец (если курсор находится в последнем столбце, ничего не происходит)

**<lf>** (0x0a)

перемещает курсор вниз на одну строку, прокручивает вверх в нижней строке

**<cr>** (0x0d)

перемещает курсор на первый столбец текущей строки

**<esc>** (0x1b)

отмечает начало escape-последовательности (см. ниже)

Все остальные символы в диапазоне управляющих символов ASCII (0x00 - 0x1f, 0x7f) игнорируются. Escape-последовательности VT52:

**<esc>** **)** (0x1b 0x29) и **<esc>** **=** (0x1b 0x3d)

переключают клавиатуру для использования, в обычном (числовом) режиме (без эффекта в **tnylpo**)

**<esc> A** (0x1b 0x41)

перемещает курсор вверх на одну строку, останавливается на верхней строке

**<esc> B** (0x1b 0x42)

перемещает курсор вниз на одну строку, останавливается в нижней строке

**<esc> C** (0x1b 0x43)

перемещает курсор вправо на один столбец, останавливается на последнем столбце

**<esc> D** (0x1b 0x44)

перемещает курсор влево на один столбец, останавливается на первом столбце

**<esc> F** (0x1b 0x46) и **<esc> G** (0x1b 0x47)

отображает коды символов 0x5e - 0x7e в графическом режиме и как символы ASCII соответственно (см. ниже)

**<esc> H** (0x1b 0x48)

перемещает курсор в левый верхний угол дисплея

**<esc> I** (0x1b 0x49)

перемещает курсор вверх на одну строку, прокрутка вниз до первой строки

**<esc> J** (0x1b 0x4a)

очищает отображение от текущей позиции курсора до конца экрана

**<esc> K** (0x1b 0x4b)

очищает отображение от текущей позиции курсора до конца строки

**<esc> Y <line> <column>** (0x1b 0x59 0xll 0xcc)

перемещает курсор в указанную позицию дисплея. Номера строк и столбцов даны в виде графических символов в диапазоне от 0x20 (позиция 1) до 0x7e (позиция 95). Если номер столбца больше ширины дисплея, горизонтальное положение не изменяется. Номер строки больше высоты дисплея перемещает курсор на последнюю строку.

**<esc> Z** (0x1b 0x5a)

запрашивает тип терминала. Эмулятор терминала отвечает отправкой последовательности **<esc>** **/K** (0x1b 0x2f 0x4b), т.е. е. он идентифицирует себя как VT52 без печатного устройства

**<esc> [** (0x1b 0x5b) и **<esc> <esc> \** (0x1b 0x5c)

осуществляют вход и соответственно выход из режима "hold screen" удержания экрана (см. ниже)

Кроме того, эмуляция терминала в **tnylpo** поддерживает следующие расширения для escape-последовательностей VT52:

**<esc> E (0x1b 0x45)**

Очищает дисплей, перемещает курсор в верхний левый угол дисплея

**<esc> L** (0x1b 0x4c)

вставляет пустую строку в положение курсора, перемещает строки расположенные ниже вниз на одну строку (последняя строка будет потеряна)

**<esc> M** (0x1b 0x4d)

удаляет строку в положении курсора, перемещает строки расположенные ниже на одну строку вверх (в нижней части дисплея появится пустая строка)

**<esc> N** (0x1b 0x4e)

вставляет пустой символ пробела в положение курсора, перемещает символы справа на одну позицию вправо (последний символ в строке будет потерян)

**<esc> O** (0x1b 0x4f)

удаляет символ в позиции курсора, перемещает символы справа на одну позицию влево (в последнем столбце строки появляется пустой символ)

**<esc> a** (0x1b 0x61) и **<esc> b** (0x1b 0x62)

делают курсор невидимым и соответственно видимым

**<esc> c** (0x1b 0x63) и **<esc> d** (0x1b 0x64)

переключают на альтернативный или соответственно основной набор символов (см. ниже)

**<esc> e** (0x1b 0x65) и **<esc> f** (0x1b 0x66)

включают и соответственно выключают утолщение символов

**<esc> g** (0x1b 0x67) и **<esc> h** (0x1b 0x68)

включают и соответственно выключают подчеркивание символов

**<esc> i** (0x1b 0x69) и **<esc> j** (0x1b 0x6a)

включают и соответственно выключают инверсное отображение символов

**<esc> k** (0x1b 0x6a) и **<esc> l** (0x1b 0x6c)

включают(ррр!) и соответственно выключают (фу!) мигающие символы

**<esc> m** (0x1b 0x6d)

выключает отображение утолщенных, подчеркнутых, мигающих и инверсных символов, а также режим выделения (см. ниже)

**<esc> n** (0x1b 0x6e) и **<esc> o** (0x1b 0x6f)

переключают клавиши курсора в режим приложения и соответственно возвращают в обычный режим (VT52): в обычном режиме клавиши курсора отправляют последовательности <esc> A (вверх), <esc> B (вниз), <esc> C (вправо) и <esc> D (влево). В режиме приложения они отправляют WordStar-совместимые команды, а именно ^E (0x05, вверх), ^X (0x18, вниз), ^D (0x04, вправо) и ^S (0x13, влево)

**<esc> p** (0x1b 0x70) и **<esc> q** (0x1b 0x71)

включают и соответственно выключают режим ожидания. Режим ожидания является наиболее заметным символьным атрибутом, предоставляемым curses (обычно это инвертированное видео, поэтому использование <esc> p и <esc> q обычно эквивалентно использованию <esc> i и <esc> j)

Режим "Удержание экрана" является функцией терминала VT52: он вводится и завершается либо компьютером, отправляющим соответствующую последовательность управления, либо пользователем, нажав клавишу "Удержание экрана" (в **tnylpo** это клавиша F5). В режиме "Удержания экрана" попытка прокрутить экран вверх (с помощью <lf> в нижней строке экрана) блокирует дальнейший вывод до выхода из режима "Удержания экрана" или до нажатия пользователем клавиши "Прокрутки страницы" (F6 в **tnylpo**) или клавиша "Прокрутка строки" (F7 в **tnylpo**), которые разрешают вывод еще одного экрана или соответственно еще одной строки.

Отображение графических символов вместо символов ASCII для диапазона байтов 0x5e - 0x7f является еще одной функцией терминала VT52, которая позволяет получить доступ к определенным дополнительным формам, таким как подстрочные цифры или знак +-. В **tnylpo** графический режим позволяет отображать формы, определенные для позиций символов 0x00 - 0x1f и 0x7f, которые не могут быть отображены напрямую.

**Наборы символов**

Переключение между основным и альтернативным набором символов является расширением, предоставляемым **tnylpo**: в файле конфигурации могут быть определены два полных набора символов по 256 форм (каждая из которых содержит отдельный набор графических символов в позициях 0x00 - 0x1f и 0x7f). Программы могут переключаться между этими двумя наборами, используя **<esc> c** **<esc>** **d** соответственно. Переключение наборов символов не меняет символы, уже записанные на дисплей. Эта функция позволяет программам использовать национальный вариант семибитного набора символов ISO 646 параллельно стандартным символам ASCII.

Определение набора символов выполняется в файле конфигурации с помощью параметра

**[alt] char <number> = <string>**

который определяет **<string>** (односимвольный строковый литерал в двойных кавычках) как представление кода **<number>** (число в диапазоне 0 - 255) в первичном (или альтернативном, если строка начинается с префикса alt) наборе символов. Символы, не определенные таким образом, берутся из набора символов по умолчанию, который может быть выбран опцией

**[alt] charset = (ascii | vt52 | latin1 | tnylpo)**

Возможные значения соответствуют четырем встроенным наборам символов ASCII, VT52 (ASCII плюс аппроксимация графических символов VT52), восьмиразрядному набору символов ISO 8859-1 (Latin 1), который дополняет код ASCII символами, используемыми в западноевропейских языках в позициях 0xa0 - 0xff, и, наконец, расширенный набор кодов Windows 1252 (и, следовательно, набор ISO 8859-1 и ASCII), который поддерживает блочную графику и символы рисования блоков VT100 в качестве графического набора символов.

Если набор символов не указан, **tnylpo** по умолчанию использует набор символов VT52.

Используя параметр командной строки **-a**, программа может быть запущена с выбранным альтернативным набором символов. Эффект почти идентичен программе, которая в качестве первого действия выдает <esc> c. (Существует разница в полноэкранном режиме консоли: во время инициализации экрана **tnylpo** передает символ пробела из выбранного набора символов в библиотеку curses для использования в качестве фонового символа.. Поскольку сама программа может выбрать альтернативный набор символов только после завершения этой инициализации, curses получит символ пробела из основного набора символов. С помощью параметра командной строки -a она получит символ пробела из альтернативного набора символов. Это различие в основном академическое, так как в любом случае не рекомендуется переопределять символ пробела, см. ниже.)

Выходные символы, которые не определены в текущем выбранном наборе символов, игнорируются эмуляцией терминала. Параметр конфигурации

**unprintable = <string>**

заменяет неопределенные символы на выходе значением **<string>** (которое должно содержать один символ).

**Функциональные клавиши**

Помимо F5, F6 и F7, которые используются для реализации функции "Удержание экрана" VT52 (см. выше), эмуляция терминала **tnylpo** определяет функциональные клавиши F1, F2 и F3 как эквиваленты трех немаркированных клавиш терминал VT52. При нажатии они возвращают последовательности <esc> P (0x1b 0x50), <esc> Q (0x1b 0x51) и <esc> R (0x1b 0x52). F4 заставляет эмуляцию терминала перерисовывать его отображение, что полезно, если какая-то другая программа или операционная система хоста испортила экран пользователя. F10 отправляет SIGINT на **tnylpo**, вызывая внезапную остановку эмуляции, но позволяя самому **tnylpo** корректно завершить работу (эта клавиша является последним средством остановки потерянной программы CP/M).

**Символьные устройства**

Помимо двунаправленного консольного устройства, CP/M поддерживает три однонаправленных символьных устройства, принтер только для вывода и устройства перфорации бумажной ленты LST: и PUN: и устройство считывания бумажной ленты только для ввода RDR:. Утилита **tnylpo** представляет эти устройства с помощью файлов хост-системы, в которые записываются данные, предоставляемые программами CP/M, и соответственно, из которых считываются данные, представленные в программы CP/M.

Эти файлы определяются параметрами конфигурации

**(printer | punch | reader) file = <путь>**

где **<путь>** - это строка в двойных кавычках, содержащая путь к файлу Unix. В случае устройств вывода (LST: и PUN:) данные, записанные программой CP/M, добавляются к содержимому именованных файлов RDR: Введите начало в начале файла считывателя. Если программа CP/M читает после конца файла считывателя, **tnylpo** продолжит возвращать ^Z (0x1a) байтов в качестве указания конца файла.

В зависимости от параметров конфигурации символьные устройства **tnylpo** могут работать в двух режимах,

**(printer | punch | reader) mode = (text | raw)**

В режиме **raw** (необработанном) байты записываются в файл в точности так, как они были сгенерированы программой CP/M, и соответственно передаются в программу CP/M точно так же, как они считываются из файла (байты ^Z по-прежнему возвращаются как указание EOF). Переключение набора символов консолью (с помощью escape-последовательностей <esc>c и <esc>d) повлияет на этот перевод, но переключение в графические символы (<esc>F и <esc>G) влиять не будет. Кроме того, в этом режиме **tnylpo** преобразует маркеры конца строки из CP/M (^M^J, 0x0d 0x0a) в Unix (^J, 0x0a) и наоборот.

Программы CP/M могут получать доступ к символьным устройствам (включая консольное устройство) не только посредством вызова функций BDOS (1 - 6 и 9 -11), но и более напрямую путем вызова соответствующих записей BIOS (CONST, CONIN, CONOUT, LIST, PUNCH, READER и LISTST), которые (в отличие от конкретных записей дискового ввода/вывода BIOS) в **tnylpo** полнофункциональны.

**СТАТУС ЗАВЕРШЕНИЯ ПРОГРАММЫ**

Утилита **tnylpo** завершает работу со статусом 0, если она не обнаружила фатальную ошибку, а статус 1 в противном случае. К фатальным ошибкам относятся ошибки командной строки и конфигурации, а также некоторые ошибки в программе CP/M, которые **tnylpo** считает неустранимыми, например, попытка записать файл только для чтения или файл на диск, настроенный как доступный только для чтения, недопустимые аргументы для системных вызовов или недопустимая последовательность операций (например, попытка чтения из файла на диске, который уже был закрыт).

Поскольку CP/M-80 (по крайней мере, в версии 2.2) не имеет понятия статуса завершения программы, сама программа CP/M не имеет простого способа сообщить о неудачном выполнении в среде Unix.

**ФАЙЛЫ**

**./.tnylpo.conf**

Файл **.tnylpo.conf** в текущем рабочем каталоге используется в качестве файла конфигурации, если он присутствует и в командной строке не указан файл конфигурации.

**~/.tnylpo.conf**

Файл **.tnylpo.conf** в домашнем каталоге пользователя используется в качестве файла конфигурации, если он присутствует, если в командной строке не был указан файл конфигурации, и если **.tnylpo.conf** отсутствует в текущем рабочем каталоге.

Использование информации о конфигурации из файла в текущем рабочем каталоге (**./.tnylpo.conf**) удобно во многих ситуациях, но представляет потенциальную угрозу безопасности. Главным образом по этой причине **tnylpo** отказывается работать, если эффективный идентификатор ее пользователя равен 0.

**ПРИМЕЧАНИЯ**

**Различия между tnylpo и СР/М-80**

По замыслу, существуют некоторые несовместимости между CP/M-80 и эмуляцией, предоставляемой **tnylpo**:

**Прямой доступ к областям BDOS и BIOS**

Поскольку она не содержит никакого фактического кода CP/M, Все программы, пытающиеся сопоставить BDOS или иным образом сделать предположения о макете операционной системы или ее внутренних структурах данных, потерпят неудачу при запуске под **tnylpo**.

Эмуляции функций BDOS и BIOS активируются имитируемым процессором, выполняющим инструкцию RET, извлеченную из одного из самых верхних 19 адресов пространства памяти CP/M (0xffed для записи BDOS, с 0xffee до 0xfffe для 17 записей BIOS CP/M-80 2.2 и 0xffff для одной подпрограммы задержки специфичной для **tnylpo**, скрываемой как 18-ая запись BIOS). Такое использование «магических адресов» может сбить с толку отладчиков, пытающихся отследить системные вызовы.

Область BIOS (начинающаяся за три байта до адреса, сохраненного в 0x0001) содержит только вектор перехода BIOS из 17(+1) элементов, фиктивные структуры диска (см. Ниже) и вышеупомянутые 19 инструкций RET. Область BDOS (начиная с адреса, хранящегося в 0x0006) еще короче, содержит только переход к 0xffed и таблицу целевых адресов для четырех состояний фатальной ошибки BDOS (несуществующий диск, поврежденный сектор, диск только для чтения и файл только для чтения. Эти адреса могут быть изменены прикладной программой, но они полностью игнорируются **tnylpo**). Любая программа, ожидающая, что области BDOS или BIOS будут иметь размеры и характеристики выравнивания реальной среды CP/M-80, будет разочарована.

Серийный номер ОС, хранящийся в шести байтах непосредственно перед областью BDOS, является (мы надеемся, безобидным) манекеном.

**Прямой доступ к дискам**

Все программы, пытающиеся получить прямой доступ к структуре диска CP/M (например, редакторы дисков), не будут работать, поскольку отсутствует базовая структура диска CP/M (**tnylpo** переводит вызовы FDOS в операции над файлами Unix). Все связанные с диском вызовы BIOS реализованы как фиктивные функции (те немногие, которые могут возвращать индикацию ошибки (SELDSK, READ и WRITE), будут делать это). Функции BDOS, возвращающие информацию, связанную со структурой диска (27 и 31), будут возвращать фиктивные структуры, содержащие бессмысленные (но непротиворечивые) данные. Например, все дисководы будут иметь размер блока 16 КБ, 2048 записей каталога и емкость 8 МБ, из которых 8128 КБ (8 МБ ниже четырех блоков каталога) являются свободными (для экономии места все диски используют один и тот же фиктивный вектор распределения, что, конечно, невозможно в реальных CP/M-80). Вся информация блока, возвращенная функциями 17 и 18 BDOS, не имеет смысла. Независимо от содержимого поля S2 (смещение FCB 14), все остальные совпадающие файлы возвращаются только один раз (в отличие от одного на физический экстент в CP/M-80). Аналогично, вся информация о блоках в FCB открытых файлов (смещения байтов FCB с 16 по 31) не имеет смысла (но не может быть нарушена, поскольку **tnylpo** хранит там некоторую информацию о состоянии, см. Ниже). Эмуляция достаточно хороша для программ, которые ищут шаблоны имен файлов или просто выводят каталог диска, но не сможет выполнить те, которые пытаются проанализировать блочную структуру эмулируемого диска на основе возвращенной информации.

**Процессор консольных команд**

Эмулятор **tnylpo** не эмулирует CCP, поэтому любая программа, использующая его функции (например, создавая файл $$$.SUB и ожидая, что CCP выполнит его содержимое), не будет работать правильно. Окончание программы простым возвращением в CCP поддерживается. Эмулятор **tnylpo** инициализирует 8-уровневый стек CCP в конце TPA и помещает адрес записи горячей загрузки в вектор BIOS, поэтому программа, пытающаяся вернуться к CCP, будет корректно завершена.

**Атрибуты файлов**

Атрибуты файла (только для чтения и системный атрибут или четыре атрибута, доступных для пользовательских программ) не поддерживаются **tnylpo**. Соответствующая функция 30 BDOS является фиктивной. Символы имен файлов всегда будут возвращаться с сбросом старших значащих битов.

**Области пользователя**

Концепция пользовательских областей не реализована **tnylpo**. Хотя код пользователя может быть установлен функцией 32 BDOS (и будет возвращен при запросе), он не влияет на файловые операции.

**Байт ввода/вывода**

Аналогично, функциональность байта ввода/вывода не реализована. Хотя байт ввода/вывода может быть запрошен и установлен (функциями BDOS 7 и 8) и сохранен по адресу 0x0003, он не влияет на символьные устройства.

**Структуры FCB**

Вызовы FDOS переводятся в файловые операции Unix. Чтобы это работало, **tnylpo** поддерживает отдельный список соответствующих файлов Unix. Ссылки на записи этого списка хранятся в FCB открытых файлов CP/M (16-разрядный ссылочный номер хранится в смещениях 16 и 17 в FCB, и то же самое число XORed 0xafcb хранится на смещениях 18 и 19). Это позволяет программам копировать FCB открытых файлов и использовать копии для дальнейшего доступа к тем же файлам (например, программы, написанные на Turbo Pascal, делают такие вещи). На части диска и имени файла FCB ссылаются только функции BDOS 15, 17, 19, 22, 23, 30 и 35, все остальные функции (особенно функции чтения и записи) используют ссылочный номер файла для идентификации соответствующего Unix файла. Текущий номер записи в последовательных операциях ввода/вывода сохраняется в полях FCB EX, S2 и RC (смещения 12, 14 и 32).

**Разреженные файлы**

Поскольку файловые операции CP/M напрямую сопоставляются с файловыми операциями Unix, некоторые характеристики файлов Unix видны для программ CP/M, работающих на **tnylpo**: Попытка считать запись в незаписанном блоке в середине разреженного файла приведет к индикация ошибки (чтение незаписанных данных) в CP/M-80, но просто возвращает запись нулевых байтов в **tnylpo**, поскольку нераспределенные области в середине файла Unix читаются как ноль. Хотелось бы надеяться, немного программ будут обижаться на это различие.

**Закрытие файла**

Обычно закрытие файла в программе CP/M (функция 16 BDOS) заставит **tnylpo** закрыть соответствующий файл Unix и освободить его запись в списке файлов. Это вызывает проблемы с определенным программным обеспечением CP/M (например, dBase II), которое продолжает использовать FCB для файловых операций после вызова функции BDOS 16. Поскольку операция закрытия просто записывает обновленные данные FCB обратно в каталог диска (но не изменяется FCB), это возможно в CP/M-80 (хотя, на мой взгляд, немного грязно). Наоборот, **tnylpo** (в дополнение к закрытию соответствующего файла Unix) удаляет ссылку на свой файл из FCB, тем самым помечая его как недопустимый для дальнейшего ввода-вывода, что приводит к сбою последующих операций программой CP/M.

Чтобы приспособиться к таким программам, **tnylpo** предоставляет параметр командной строки **-n** (и соответствующий параметр конфигурации close files = false), который предотвращает закрытие файла Unix и удаление ссылки на файл из FCB (фактически делая функцию BDOS 16 фиктивной операцией). Если многие файлы открываются (последовательно или одновременно) программой CP/M, это может привести к исчерпанию файлов **tnylpo**, так как закрытие всех файлов Unix откладывается до завершения программы.

**Формат текстового файла**

Дисковый ввод/вывод файлов всегда выполняется непереведенным, то есть содержимое дисковых файлов, записываемых программами, выполняющимися в **tnylpo**, всегда находится в наборе символов, используемом внутренне для эмуляции CP/M. Аналогично, соглашение о конце строки в текстовых файлах соответствует условию CP/M (^M^J, 0x0d 0x0a), а текстовые файлы, которые не заканчиваются точно в конце записи файла CP/M длиной 128 байт, будут содержать байт ^Z (0x1a) как логический маркер EOF (в идеале они должны быть заполнены байтами ^Z до следующей границы записи, но в действительности они обычно заканчиваются одним ^Z, за которым следует произвольный мусор). Это необходимо учитывать при использовании утилит Unix для обработки файлов, сгенерированных **tnylpo**.

С другой стороны, текстовые файлы, созданные в Unix, должны быть преобразованы в соглашение о конце строки CP/M перед обработкой их с помощью программ, работающих под **tnylpo** (команда set ff = dos может пригодиться людям, использующим Vim!). Соглашение CP/M о маркировке логического конца текстовых файлов выполняется автоматически, поскольку эмуляция BDOS **tnylpo** дополняет файлы Unix до размера записи CP/M 128 байтов байтами ^Z.

Преобразование текстового файла между Unix и форматом CP/M может быть удобно осуществлено с помощью сопутствующей программы **tnylpo-convert** (1).

**Производительность**

Утилита **tnylpo** была оптимизирована для мобильности, а не для производительности. Некоторые части эмуляции процессора (например, операции сложения и вычитания) могут быть смехотворно неэффективными по сравнению с оптимизированной вручную версией ассемблера (или даже версией C, в которой умеренно используются вещи, специфичные для платформы, такие как порядок байтов или представление чисел). С другой стороны, **tnylpo** должна компилироваться и запускаться на любой платформе Unix, поддерживающей ISO C99 (для поддержки широких и многобайтовых символов), версии библиотеки ncurses с поддержкой широких символов и переменных int, содержащих более 16 бит. Тем не менее, я обнаружил, что **tnylpo** ослепительно быстра по сравнению с реальным, даже на устаревшем оборудовании, которое я использовал для его разработки.

**Процедура задержки**

Поскольку CP/M-80 версии 2.2 не предлагает никаких функций для хранения времени или задержек, программы вынуждены использовать время цикла определенных инструкций в сочетании с тактовой частотой процессора, если им нужно задержать выполнение программы. Этот подход обречен под эмулятором, особенно если он работает на многопрограммной системе.

Чтобы позволить программам CP/M задерживать выполнение на определенный период времени (wall clock), **tnylpo** предоставляет процедуру задержки, маскирующуюся под 18-ю запись BIOS: она ожидает один параметр, 16-разрядное число без знака в регистре BC и ожидает это число миллисекунд прежде, чем возвратиться к вызывающей стороне..

Преимущество этой подпрограммы ограничено отсутствием. Поскольку это расширение, ни одно из существующих приложений CP/M (для которых **tnylpo** был создан в первую очередь) не использует его, и даже в случае, если кто-то разработает новое программное обеспечение CP/M, использование функции задержки **tnylpo** нарушит совместимость со всеми остальными машинами или эмуляторами CP/M.

**Название программы**

"**tnylpo**" - это фантастическое слово. Это не аббревиатура и не имеет никакого смысла, о котором я знаю (это главная причина, по которой я его выбрал). "**tnylpo**" произносится как носитель немецкого языка (например, я) наивно произносил бы "**tnylpo**" (или как говорящий на финском языке произносил бы "**tnylpo**", при условии, что оно выживет в исходной группе согласных). В своем образовании "**tnylpo**" с его двумя слогами, заканчивающимися на "**po**", хорошо совпадает с другими бессмысленными словами, которые я придумал ранее (например, "**ilpo**" или "**sampo**"), - пока я не переехал в Финляндию и обнаружил, что большинство из них, предполагаемо глупых слов имеют здесь смысл ...

**ОШИБКИ**

Смотрите комментарии за ***Примечаниями***. Большинство из них могут быть интерпретированы как ошибки в реализации, включая имя программы.

**Переключение набора символов**

Программа, работающая в полноэкранном режиме (эмуляция VT52), может изменить свой набор символов отображения с основного набора символов на альтернативный набор и обратно путем выдачи соответствующей escape-последовательности. Это переключение также влияет на перевод набора символов других символьных устройств (LST:, PUN: и RDR:), если они не работают в необработанном режиме.

**Переопределение символов**

Не рекомендуется переопределять какие-либо символы, которые могут появляться в именах файлов CP/M, их эквивалентах Unix или в полях имени/расширения файла FCB, к ним относятся десятичные цифры (0x30-0x39), верхние - и символы нижнего регистра (0x41-0x5a и 0x61-0x7a), пробел (0x20), знак доллара (0x24), звездочка (0x2a), минус (0x2d), точка (0x2e), двоеточие (0x3a) и знак вопроса (0x3f).

**Языки хост-системы**

Программа **tnylpo** была реализована с намерением работать со всеми возможными региональные параметрами (проф. жарг. лока́ль от англ. locale) Unix CTYPE (другие аспекты текущей локали игнорируются). К сожалению, поддерживаются только системы и локали на основе ASCII, поскольку нет независимого способа представления для некоторых элементов управления CP/M (например, ^C и ^Z).

В настоящее время утилита **tnylpo** всегда устанавливает, что коды символов 0x00-0x1f и 0x7f набора одно или многобайтовых символов операционной системы хоста соответствуют управляющим символам ASCII. Честно говоря, утилита **tnylpo** была тщательно протестирована только для работы с локалями на основе UTF-8.

**Конфликты пространства имен файлов**

**tnylpo** пытается скрыть неподходящие файлы в каталогах хостов, используемых в качестве дисков CP/M от программ CP/M. При поиске файлов и запросах открытия будут игнорироваться все, кроме обычных файлов размером до 8 МБ с допустимыми именами. Однако эта попытка скрыть другие объекты далека от совершенства: программа CP/M, пытающаяся, например, создать файл, имя которого совпадает с именем (скрытого) каталога, таинственно потерпит неудачу (операция переименования может столкнуться с аналогичными проблемами). Поэтому рекомендуется зарезервировать каталоги, используемые в качестве дисков CP/M, для соответствующих файлов или дать всем другим объектам имена, не приемлемые в качестве имен файлов CP/M (ведущая точка может творить чудеса!).

**ПРИМЕР**

tnylpo -f ws.conf ws hugo.txt

будет использовать файл конфигурации ws.conf и запустит программу ws.com, передав ей строку hugo.txt в качестве параметра командной строки.

Если файл ws.conf содержит следующие строки

#

# Конфигурация WordStar

#

drive a = "."

printer file = "./printer.txt"

printer mode = text

logfile = "./.tnylpo.log"

loglevel = 0

#

# Использовать немецкий набор символов

#

charset = ascii

char 0x40 = "S"

char 0x5b = "A"

char 0x5c = "O"

char 0x5d = "U"

char 0x7b = "a"

char 0x7c = "o"

char 0x7d = "u"

char 0x7e = "B"

unprintable = "c"

console = full

lines = 24

columns = 80

application cursor = true

screen delay = 2

Эмулятор **tnylpo** будет искать ws.com (и другие файлы) в локальном каталоге, который будет использоваться в качестве диска A в эмуляции. Консольный ввод/вывод будет обрабатываться в полноэкранном режиме с помощью эмуляции VT52, которая использует экран из 24 строк по 80 столбцов. Клавиши курсора будут переведены на ^E (вверх), ^S (влево), ^D (вправо) и ^X (вниз), а набор символов будет соответствовать немецкому варианту ISO/IEC 646. (символы Umlaut вместо скобок и фигурных скобок ASCII). Символы в диапазоне от 0x80 до 0xff будут отображаться в виде перевернутых вопросительных знаков.

Вывод на принтер будет записан в printer.txt в текущем каталоге, будет в наборе символов Unix и будет следовать соглашениям о конце строки Unix. Журнал будет добавлен в файл .tnylpo.log в текущем каталоге и будет содержать только сообщения об ошибках. После завершения программы **tnylpo** будет ждать две секунды, прежде чем перезагрузить экран.

**АВТОР**

Программу

([tnylpo@gmx.at](mailto:tnylpo@gmx.at)), программист, администратор информационных систем и партизанский египтолог.

**СМОТРИТЕ ТАКЖЕ**

tnylpo-convert(1)

Реализация эмуляции процессора Z80, особенно функций, не охваченных официальной документацией (Zilog Inc., Руководство пользователя процессора Z80, редакция 11: август 2016 г. [UM008011-0816]), находится под сильным влиянием "The Undocumented Z80 Documented" Шона Янга {Sean Young}. (Версия 0.91, 18 сентября 2005 г.).

Эмуляция системного интерфейса CP/M-80 2.2 основана на руководстве по операционной системе CP/M Digital Research International (Третье издание: сентябрь 1983 года).

2019-06-13 [tnylpo(1)](manpage:tnylpo(1))