**Руководство пользователя Z-System**

Ричард Якобсон {Richard Jacobson}

Брюс Морган {Bruce Morgen}

1. Введение 3

2. Концепции Z-System 4

2.1. Операционная система 4

Что такое операционная система? 4

Чем Z-System отличается от CP/M? 5

2.2. Основные понятия ZCPR3 6

Несколько команд в строке 6

Каталоги и области пользователя 6

Приглашение ZCPR3 7

Обработка команд 7

2.3. Основные понятия ZRDOS 8

3. Автозагрузка и системные сегменты 9

3.1. Введение в автозагрузку 9

3.2. Псевдонимы автозагрузки 9

3.3. Системные сегменты 11

3.4. Дескриптор среды 12

3.5. Пакет команд управления выполнением 12

3.6. Пакет резидентных команд 15

3.7. Именнованные каталоги 19

4. Другие команды Z-system 23

4.1. Встроенные команды 23

4.2. Редактирование строки и управление выводом 26

4.3. Составная командная строка ZCPR3 26

5. Псевдонимы 27

5.1. Что такое псевдоним? 27

5.2. Утилита ALIAS 28

5.3. Переменные ALIAS 28

5.4. Примеры псевдонимов 29

Пример 1: ASSEMBLE 29

Пример 2: ARCHIVE 29

Пример 3: I.COM 29

Пример 4: WSTAR 30

Пример 5: DIRB 30

Пример 6: NEWDISK 30

Пример 7: COMPILE 31

Пример 8: M8 31

Пример 9: WSTR 31

5.5. Исследования 32

6. ZEX, MENU и VFILER 32

6.1. Введение в расширенную обработку команд 32

6.2. ZEX 33

Пример 1: Интерактивный режим ZEX 34

Пример 2: Использование управления выполнением 35

Пример 3: ZASLINK 37

Пример 4: Выбор ассемблера в сочетании с проверкой ошибок 38

6.3. Подсистема меню 40

MENU 40

VMENU 42

6.4. VFILER 45

7. Взаимодействие с терминалами и принтерами 48

7.1. Введение 48

7.2. The TCAP 49

Получение информации TCAP в дескрипторе среды 49

7.3. Определение периферийного буфера 52

Выбор терминалов и принтеров 53

Примеры утилит, которые читают буфер определения периферии 54

7.4. Дальнейшие исследования 55

8. Инструменты Z-System (Z утилиты) 55

8.1. Утилиты ZCPR3 55

8.2. Утилиты ZRDOS 60

# Введение

Z-System (ZCPR3, ZRDOS и связанные с ними утилиты) является революционной операционной системой для микрокомпьютеров. Начиная с выпуска ZCPR3 в начале 1984 было установлено более 60,000 систем. Многие из этих установок теперь включают ZRDOS. (ZRDOS является одной из последних разработок. Много пользователей реализовало ZCPR3, прежде чем был выпущен ZRDOS в начале 1985.) Популярность Z-System была не менее блистательной.

Несмотря на ее широкое признание современной операционной системой для 8-разрядных компьютеров некоторые пользователи выразили беспокойство, что они будут иметь затруднения при изучении Z-System.

Руководство пользователя Z-System устраняет эту озабоченность, вызванную вашими проблемами с обучением и позволяет немедленно воспользоваться преимуществами Z-System.

Z-System предоставляет вам рабочую среду и способ управления этой средой, непревзойденный любой другой операционной системой для микрокомпьютеров. По сравнению с Z-System, CP/M-80 сырая, и в MS-DOS многое отсутствует.

Конечно, в той степени, в которой Z-System предоставляет вам непревзойденный уровень управления, она также ожидает от вас его использования. Когда вы сможете без проблем освоиться в Z-System, как в «обычной» CP/M-80, мы считаем, что скромные требования, выдвигаемые Z-System для получения полной справки обеспечивает ее самые главные преимущества - расширение знаний и повышение производительности.

Руководство пользователя Z-System может использоваться самостоятельно, хотя оно предполагает существование Руководства ZCPR3 Richard Conn. Его объем является исчерпывающим техническим справочником. Оно решает проблемы, уникальные для начинающего пользователя Z-System. Множество вариантов, представленных Z-System, сделало проблему выбора трудной и важной и, если вы считаете, что некоторые части системы несправедливо ущемлены или проигнорированы, можно сказать, что рамки наших возможностей, с учетом короткого промежутка человеческой жизни, были значительно сокращены.

За разного рода помощь и вдохновение мы в долгу перед Дэйвом Маккордом {Dave McCord}, Джей Сейджем {Jay Sage}, Полом Померло {Paul Pomerleau}, Стивом Коэном {Steve Cohen}, Джеффом Моро {Jeff Moro} и системными операторами Z-Node Ричардом Конном {Richard Conn} и Фрэнком Гауде {Frank Gaude}, которых можно назвать крестными отцами данного руководства, хотя их щедрые благодеяния для 8-битного сообщества далеко превысили эту степень ответственности.

Ричард Якобсон {Richard Jacobson}

Брюс Морган {Bruce Morgen}

Чикаго, Иллинойс

Уорминстер, Пенсильвания

Апрель 1986

# Концепции Z-System

## Операционная система

Z-System является революционной операционной системой, описывая ее возникают два вопроса: (1) Что такое операционная система? (2) Чем Z-System отличается от CP/M?

### Что такое операционная система?

Операционная система это программа, которая управляет основными служебными функциями вашего компьютера. Операционная система заставляет символы отображаться на экране, включает и выключает лампочки у дисковода, говорит дисководам читать дискеты и интерпретирует команды, которые вы вводите на клавиатуре. Операционная система ответственна за функции, которые поддерживают работу вашего компьютера и повинуется вашим командам. Она обеспечивает связь между компонентами оборудования вашего компьютера от дисководов до видео терминала, от клавиатуры до принтера и модема.

Во многих компьютерах часть операционной системы расположена в микросхемах ПЗУ (память только для чтения). Эти ПЗУ или ППЗУ (перепрограммируемая память только для чтения) являются полупроводниковыми устройствами, расположенными на системной плате вашего компьютера, и могут быть запрограммированы, для выполнения многих служебных задач, о которых мы говорили.

Другая часть операционной системы расположена на самых дальних от центра дорожках ваших "системных" дискет. Системная дискета является дискетой из которой ПЗУ компьютера "загружает" операционную систему, или загружает ее в память. Все компьютеры поставляются с программой для записи образа операционной системы на дискеты или на жесткий диск.

Когда вы включаете свой компьютер, машинные команды в ПЗУ говорят вашему компьютеру считать информацию, расположенную на системных дорожках в память. Если это является единственной функцией, предоставляемой ПЗУ, его называют "ПЗУ начальной загрузки". В некоторых компьютерах ПЗУ - больше, чем "ПЗУ начальной загрузки". Если это так, то после того, как изображение операционной системы на диске считано в память, резидентная система обращается к ПЗУ с просьбой выполнять необходимые функции. Несмотря на важное значение ПЗУ и его функции, когда мы говорим о Z-System или любой другой операционной системе, мы имеем в виду полнофункциональную служебную программу, расположенную на системных дорожках.

Операционная система, будь то Z-System или СР/М, имеет три основные составляющие: базовую систему ввода-вывода (BIOS), базовую дисковую операционную систему (BDOS) и процессор консольных команд (CCP).

BIOS представляет собой интерфейс между программным обеспечением и оборудованием. Он содержит подпрограммы, которые взаимодействуют с дисками, экранами, принтерами, модемами и т.п. Так как BIOS является лишь частью операционной системы, которая взаимодействует непосредственно с аппаратными устройствами (и, в некоторых компьютерах, таких как Kaypro, ПЗУ), она является единственной частью операционной системы, которая уникальна для компьютера.

BDOS расположена в памяти чуть ниже BIOS. Она действует как полнофункциональная сервисная служба, для программ и остальной части операционной системы, выполняя такие функции как чтение и запись на диски, поддержку структуры каталогов с файлами на диске, обработку информации, которая поступает в BIOS для окончательной обработки и выделения дискового пространства.

CCP, который находится в памяти непосредственно под BDOS, является программой, работающей в отсутствие любой другой программы. Он является интерфейсом между программами и пользователем. CCP обрабатывает команды, которые вы вводите в консольной клавиатуре. Когда вы видите знакомое приглашение A> или A0>, это CCP помещает его туда. Когда появляется это приглашение, CCP сообщает, что он ждет ваших команд.

Некоторые команды, такие как DIR, REN, ERA, TYPE и SAVE в системе CP/M обрабатываются непосредственно CCP. Эти команды вызывают "*резидентными командами*", потому что они находятся в памяти во время вашего сеанса с компьютером.

Если команда не является одной из этих резидентных команд, CCP CP/M обращается к текущему диску (диску B, если выведено приглашение B> или диску A, если выведено приглашение A>), и текущей области пользователя (об этом позже) для поиска файла COM, с тем же именем, что у введенной команды. Такие COM файлы называют "*транзитными*", потому что их доступность зависит от присутствия на диске.

После окончания выполнения некоторых COM файлов на вашем экране может появиться сообщение "Warm Boot" (теплая загрузка). Сообщение "Warm Boot" информирует вас о том, что CCP считывается с системных дорожек вашего дисковода и данные загружаются в соответствующие ячейки памяти. Почему CCP должен быть перезагружен в память? Многие COM файлы принимают на себя функции CCP во время работы. WordStar служит хорошим примером такой программы. Когда WordStar работает, он не обращает внимания на CCP и использует место CCP в памяти для своих собственных целей. Когда вы заканчиваете использовать WordStar, выполняется "Warm Boot" или повторное чтение CCP в память, потому что WordStar перезаписал области памяти, в которых расположен CCP. Если бы не было "Warm Boot" после окончания работы WordStar, то операционная система больше не была бы в состоянии обрабатывать команды, введенные с клавиатуры. В отличие от CCP и BIOS (в большинстве компьютеров) BDOS остается постоянно в памяти в течение всего времени включения вашего компьютера.

### Чем Z-System отличается от CP/M?

Z-System имеет ту же базовую структуру, что и СР/М. BIOS в Z-System подобен BIOS в CP/M, но с определенными дополнительными функциями во время "Холодной" загрузки или начального запуска компьютера. BDOS выполняет те же функции в Z-System, что и в CP/M, снова, с определенными значительными дополнениями и улучшениями. Однако, она имеет другое имя - ZRDOS. ZRDOS обозначает "Z80 замена дисковой операционной системы". CCP в Z-System называется ZCPR3 и является предметом гордости Z-System, значительно увеличивая функциональность и производительность по сравнению с CCP CP/M.

Несмотря на то, что Z-System полностью совместима практически со всем программным обеспечением, написанным для CP/M, и практически прозрачна для пользователей компьютеров управляемых CP/M, на этом сходство заканчивается.

Прежде чем мы продолжим исследовать некоторые основные понятия, давайте подчеркнем, что мы не обсуждаем, как установить Z-System на компьютер под управлением CP/M. Есть много способов выполнить это, начиная от ручной установки с нуля, используя информацию, предоставленную в ZCPR3: Руководство, добавленное в Z-News Эшелоном {Echelon}, файлы группы NAOG/ZSIG One-Eighty и в сети Z-Node Remote Access Systems (RAS), до автоматической начальной загрузки с помощью Vanguard Echelon или пакетов Z-Com. Сделайте ваш выбор, в зависимости от уровня вашего опыта общения с микрокомпьютером, но с этого момента, мы предполагаем, что у вас есть доступ к рабочей среде Z-System.

## Основные понятия ZCPR3

Командный процессор ZCPR3 Z-System, обеспечивает более функциональный, универсальный и осмысленный пользовательский интерфейс, чем CCP CP/M. Как процессор команд пользователя, ZCPR3 является беспрецедентным и непревзойденным по производительности и простоте использования.

### Несколько команд в строке

CP/M имеет ограничение "одна команда за один раз". Если вы хотите посмотреть каталог диска, стереть файл MYFILE.TXT и потом посмотреть каталог диска, чтобы убедиться в удалении, в CP/M вам нужно ввести три отдельные команды, каждая сопровождается возвратом каретки (<cr>). С ZCPR3 это не так. В ZCPR3 вы можете ввести одну командную строку, состоящую из группы команд, разделенных точкой с запятой, сопровождаемую <cr>. В ZCPR3 командная строка была бы "DIR; ERA MYFILE.TXT; DIR <cr>".

### Каталоги и области пользователя

Каталог в Z-System это логическое понятие. Физически диск содержит только одну область для своего каталога. Каталог в CP/M или Z-System является определенным логически (не физически) местом, где файлы группируются вместе. Область пользователя - дополнительный способ CP/M, для определения некоторой логической области, в которой файлы могут быть сгруппированы. В системах с двумя гибкими дисками есть два каталога CP/M - A и B и 16 доступных областей пользователя от 0 до 15 на каждом диске. В некоторых системах с жестким диском, таких как Kaypro 10, у вас есть два физических диска, жесткий диск и дискета, но три логических каталога - A, B и C. Снова, у каждого логического каталога есть 16 областей пользователя.

ZCPR3 использует термин каталог немного по-другому. В ZCPR3 "каталог" это логическая область на диске, будь то жесткий диск или дискета. Каталог ZCPR3 определяется буквой его диска и номером его пользователя. Таким образом A15 и B4 - оба "каталоги" в стиле ZCPR3. "A15" означает диска А, пользователь 15. "B4" означает диска B, пользователь 4. Этот способ обращения к каталогам ZCPR3 называют - формой "DU" (диск/пользователь). Кроме того, каталогу может быть присвоено мнемоническое имя. Имя "ROOT" может быть присвоено A15 (который мы создали на ваших дистрибутивных дисках), и имя "WORK" - B0. Этот способ обращения к каталогам ZCPR3 называют формой "DIR" (именованный каталог).

У формы "DIR" есть много преимуществ перед формой "DU", включая простое и очевидное - возможность присвоить некоторое содержательное имя диску/пользователю, чтобы помочь вспомнить типы хранящихся там файлов. Проще помнить "DBASE", чем запомнить, что вы сохранили ваши командные файлы DBASE в B7. Таким образом, вы просто присваиваете имя "DBASE" диску/пользователю B7 и когда хотите зарегистрироваться в эту область, вводите "DBASE: <cr>". Вы будете там в мгновение ока, в то время как ваши коллеги использующие CP/M все еще будут озадачены.

### Приглашение ZCPR3

Приглашение CP/M не предоставляет вам много информации. Если вы зарегистрировались на диске A вы получаете приглашение A> и если вы зарегистрировались на диске B вы получаете приглашение B>. Если вы изменяете область пользователя, вводя, например, "user 15" (вам не придется делать это в ZCPR3), вы по-прежнему будете видеть тоже же самое старое приглашение A>. Некоторые компьютеры показывают вам, в какую область пользователя вы зарегистрированы, таким образом, вы получаете приглашение вида A15>. Это - улучшение по сравнению со стандартным приглашением CP/M, но не столь же полезное как приглашение ZCPR3, доступное в Z-System.

В ZCPR3 пользователь-установщик может настроить строку приглашения. Самая популярная настройка, все равно, отображает и формы "DU" и "DIR". Приглашение, A15:ROOT>, показывает логический диск, область пользователя и имя каталога.

### Обработка команд

Обработка команды это то, что делает CCP после того, как вы (или программа суррогат SUBMIT в CP/M) задаете команду. Например, когда вы вводите "WS<cr>" процессор консольных команд (CCP) загружает и выполняет WordStar.

В CP/M обработка команд проста, почти примитивна. Система получает команду от пользователя или от файла SUBMIT. Затем она определяет, является ли команда резидентной командой CCP, такой как TYPE, REN, ERA, DIR, или SAVE и, если команда является резидентной, выполняет ее. Если команда не является резидентной, CCP определяет содержит ли текущий диск и область пользователя файл COM с тем же именем, что и команда. Если содержит, он загружает и выполняет файл COM. В противном случае CCP выдает сообщение об ошибке в виде нераспознанной команды сопровождаемой неинформативным "?".

В ZCPR3 обработка команды сложнее, функциональней и гибче. Также командный процессор ZCPR3 сообщает гораздо больше о том, что происходит с вашими командами, чем CP/M. После разбора (чтения) командной строки от пользователя или из пакетного файла, командный процессор ZCPR3 выполняет следующие операции:

1. Он проверяет, является ли команда командой потока, которая определяет дальнейшие действия системы, если такие команды имеются. Если состояние потока является истиной, то система будет продолжать обрабатывать команды. В противном случае он будет сбрасывать команду и затем продолжаться. (Помните, мы имеем дело с множественными командами.)
2. Он проверяет, является ли команда резидентным объектом ZCPR3 (резидентные объекты ZCPR3 расположены отдельно в памяти от самого ZCPR3 и более многочисленные и более функциональные, чем резидентные команды CP/M). Если команда является резидентным объектом, то командный процессор выполняет ее.
3. Если команда не является резидентным объектом, то командный процессор проверит, является, ли команда "частью" ZCPR3, т.е. встроенной командой ZCPR3. Если ZCPR3 определит, что команда является встроенной, то он выполняет ее.
4. Если команда не встроенная, то командный процессор ищет вдоль "пути поиска команды" файл COM, имеющий то же имя, что и команда. "Путь поиска" представляет собой последовательность дисков/пользователей, в которые вы можете направить систему, осуществлять поиск файлов COM. Если желаемый файл COM найден где-нибудь по пути поиска, он загружается и выполняется. Возможность управления поиском с помощью утилиты PATH, является одним из самых значительных достоинств Z-System.
5. Если требуемый файл COM не найден по пути поиска, выполняется "расширенный процессор команд", при наличии. Этот расширенный процессор команд может быть расположенным в памяти пакетным средством обработки файлов или некоторой другой подобной программой, которая затем пытается обработать команду.
6. Наконец, если все остальное действия неудачны, то система вызывает обработчик ошибок ZCPR3, если он был установлен. Обработчик ошибок является программой, которая корректно нейтрализует ошибки ввода данных пользователей и опционально допускает исправление командной строки. Если обработчик ошибок не был установлен, ZCPR3 распечатывает сообщение об ошибке, неверную команду и загадочный "?".

Существует очень много других различий между CCP CP/M и ZCPR3 Z-System. Мы еще исследуем их позже в интерактивном режиме.

## Основные понятия ZRDOS

ZRDOS, расшифровывается как Z80 замена дисковой операционной системы, и как подразумевает ее название, является заменой базовой дисковой операционной системы (BDOS), поставляемой Digital Research. ZRDOS полностью совместима с CP/M, но предлагает существенные улучшения.

В ZRDOS у вас больше нет проблемы смены дисков посреди сеанса и получения страшного сообщения "BDOS error", которое часто означает, "Скажите до свидания работе, которую вы делали в течение последнего часа". ZRDOS автоматически регистрирует в системе смененные диски. Это означает, что вы можете сосредоточиться на своей работе а не на вводе Ctrl-C каждый раз, когда вы меняете дискету.

Также ZRDOS позволяет защитить от записи некоторые файлы от непривилегированных пользователей. Вы можете установить пароль доступа. Если у другого пользователя нет пароля доступа, и вы защитили определенные файлы от записи, эти файлы не могут быть изменены или стерты никем, кроме вас. Кроме того, ZRDOS позволяет вам помечать записи файлов в каталоге, чтобы указать, что они были заархивированы. С помощью утилиты ZRDOS под названием AC.COM вы можете создавать резервную копию только тех файлов, которые еще не были "заархивированы". Наконец, если вы установили дисковод в состояние только для чтения, это состояние сохраняется после "Теплой" загрузки.

Если вы используете программы с оверлейными файлами (Wordstar является одним из популярных примеров) вы полюбите функцию общедоступных каталогов в ZRDOS. Общедоступными являются файлы, которые операционная система рассматривает, как будто они одновременно присутствуют во всех областях пользователя данного диска. Те из вас, кто владеет Kaypro 10, знают об абсурдной трате места на диске в CP/M из-за необходимости иметь копию двух больших оверлейных файлов Wordstar в каждой области пользователя, в которой вы работаете с Wordstar. В противном случае, на экране будет много странных сообщений об ошибках. Это не так в ZRDOS, которая предоставляет возможность настроить области пользователя с 1 по 8 в качестве общедоступных. Когда вы помещаете Wordstar и все его оверлейные программы в общедоступную область пользователя, вы обнаружите, что не только Wordstar работает в любой пользовательской области пользователя, но и функция "R" (запуска не Wordstar файла) наконец становится полезной.

Другие возможности ZRDOS полезны и будут упомянуты вкратце. Функция чтения буфера консоли (10) обрабатывает rubout (DEL), так же как backspace. Одноуровневая повторная входимость допускает компактные пакеты ввода-вывода. Специальные только для ZRDOS утилиты включают программы установки (отображения) атрибутов файла, дампа, сравнения, просмотра и Vtype. Существует также расширенная обработка ошибок с конкретными номерами ошибок , которые полностью объясняется в программе DOSERR.

Помимо возможности общедоступных файлов остальные функции ZRDOS могут быть для вас не столь очевидными. Но попробуйте этот эксперимент. После использования Z-System в течении нескольких недель, снова загрузите ваш старый диск CP/M. Дальнейшие вопросы по этой теме отпадут сами собой.

# Автозагрузка и системные сегменты

Автозагрузка (STARTUP) является концепцией Z-System, которую необходимо изучить для того, чтобы наиболее эффективно использовать Z-System. Как только вы освоите понятие автозагрузки, вы сможете использовать ее для создания пользовательских конфигураций Z-System для любого применения.

## Введение в автозагрузку

Интересующий нас файл, обычно называется STARTUP.COM, несмотря на то, что некоторые системы, как Micromint SB 180 используют аналогичные имена типа START.COM и другие, как Ampro Bookshelf, позволяют вам изменять имя файла автозапуска с помощью утилиты настройки BIOS. Теперь введите STARTUP<cr> или эквивалент в вашей системе. Это почти походит на нажатие кнопки сброса. Рассмотрим более внимательно STARTUP.COM и увидим, что она делает.

Введите "ALIAS STARTUP<cr>". Вы должны увидеть что-то вроде следующего, хотя то, что вы видите будет зависеть от используемого компьютера и конкретного содержания процедуры автозапуска:

ALIAS, Version 1.1

Alias Name: STARTUP

Old Alias Command Line:

1 --> 15:;

2 --> LDR SYS.ENV,SYS.FCP,SYS.RCP,SYS.NDR;

3 --> ERROR23;

4 --> VID HELLO;

5 --> TIME CD;

6 --> 0:;

7 --> CLS

Нажмите <cr>, чтобы прервать выполнение утилиты ALIAS. Мы теперь готовы проанализировать STARTUP.COM.

## Псевдонимы автозагрузки

STARTUP.COM представляет собой обычный файл COM. Давайте избавимся от той точки в начале. Да, он имеет некоторые особенности. Во-первых, STARTUP это ALIAS (псевдоним), что означает, что, хотя это обычный COM файл он имеет только одну функцию - передать группу команд в буфер составных команд. Буфер это зарезервированная область памяти для определенного использования. В Z-System буфер составных команд это область памяти (размером приблизительно 200 байт в большинстве реализаций Z-System), которая была зарезервирована для размещения группы команд. STARTUP передает группу команд в этот буфер, и Z-System затем выполняет их.

С другой стороны, STARTUP это больше, чем обычный файл COM. Она является ALIAS (псевдонимом) и, самое важное, утилитой Z-System, которую вы можете создать на лету. Утилита Z-System знает о ключевых адресах Z-System в силу своей способности найти и прочитать сегмент Z-System, перечисляющий эти адреса, расположенные в верхней памяти. (*Верхняя память* это область памяти выше области транзитных программ или "TPA", в которой обычно выполняются программы и сама операционная система.) Этот сегмент называют *дескриптором среды*. Утилиты Z-System знают, где в памяти расположен буфер составных команд , потому что они извлекают эту информацию из данных, содержащихся в дескрипторе среды.

Z-System пытается выполнить STARTUP при "Холодной" загрузке. Если Z-System находит STARTUP, она выполняет его. Директива системе искать STARTUP при "Холодной" загрузке "зашита" в BIOS. (Термин "зашита" означает, что определенный код, заставляющий систему искать STARTUP при "Холодной" загрузке закодирован в BIOS). Если файл, STARTUP.COM не существует в A0, когда системна выполняет "Холодную" загрузку, вы будете видеть "?" на вашем экране CRT. Это сообщение, как вы знаете из CP/M, означает, что система не может выполнить введенную команду. Если STARTUP будет найден в A0, то он будет выполнятся.

Теперь давайте посмотрим за передачей командной строки STARTUP в командный процессор.

Первая команда "15": область пользователя на диске A, которой соответствует каталог названный "ROOT". Она регистрирует вас в A15, где, по соглашению, обычно располагаются утилиты Z-System. Все остальные вызываемые STARTUP файлы COM, находятся в A15 или ROOT. Обратите внимание, что корректным способом переместиться из A0 в A15 является ввод 15:<cr>. Ввод A15:<cr> также работает. Не используйте команду CP/M "USER 15". Z-System не распознает ее, и мы больше не будем ее упоминать.

Следующая команда "LDR SYS.ENV,SYS.FCP,SYS.RCP,SYS.NDR". Это вторая из семи команд, которые STARTUP передает в систему и безусловно самая важная. Если вы выполните LDR.COM с параметром справки двойной наклонной черты, LDR //, то увидите следующий экран:

A0:COMMAND>**LDR //**

ZCPR3 LDR, Version 1.3

LDR Syntax:

LDR <list of packages/data files>

where entries in the list may be any of these types:

FCP - Flow Cmnd Package ENV - Z3 Environ

IOP - Input/Output Package NDR - Z3 Named Dir

RCP - Resident Cmnd Package Z3T - Z3TCAP Entry

The ENV file must be first if LDR is not installed.

A0:COMMAND>

## Системные сегменты

Пакеты, которыми мы интересуемся имеют расширения ENV, FCP, RCP и NDR. Их часто называют SYS.ENV, SYS.FCP, SYS.RCP и SYS.NDR. LDR загружает каждый из пакетов или системных сегментов в отведенное им место в памяти. Пакет может быть загружен динамично в любое время, выполнив утилиту LDR. У каждого сегмента есть характерное расширение файла. LDR распознает эти расширения файлов и загружает каждый системный сегмент по-разному. Z-System настроена для загрузки системного сегмента каждого типа, начиная с его собственного фиксированного адреса памяти. Чтобы взглянуть на эти адреса, запустите утилиту Z3LOC. Команда - Z3LOC Z<cr>.

Вы должны увидеть следующий вывод (конкретные адреса будут различаться в зависимости от того, в какой версия Z-System вы работаете):

A0:COMMAND>**Z3LOC Z**

Z3LOC Version 1.1

ZCPR3 Element Base Address

------------- ------------

CCP CC00 H

BDOS D406 H

BIOS E200 H

Env Descriptor F800 H

Pack: FCP F600 H

IOP 0000 H

RCP EE00 H

Buf: Cmd Line FF10 H

Ext FCB F9D0 H

Ext Path 0040 H

Ext Stk EDD0 H

Messages F980 H

Named Dir FA00 H

Shell Stk F900 H

Wheel Byte 003E H

Дескриптор среды (Env Descriptor) в этом примере загружен в памяти, начинающейся в F800H, пакет управления потоком (FCP) в F600H, пакет резидентных команд (RCP) в EE00H и буфер именованных каталогов (Named Dir) в FA00H. Эти адреса показывают начальную точку областей в памяти Z-System зарезервированные для конкретных пакетов и буферов.

Конкретные адреса справедливы только для определенной реализации Z-System. Есть много различных способов реализации Z-System, служащие различным целям и предлагающие различные возможности и разновидности Z. Некоторые Z-System реализованы с помощью пакета ввода-вывода или IOP. Вы можете сказать, глядя на адрес IOP равный 0000H в экране Z3LOC выше, что эта система не содержит сегмент IOP.

Системные сегменты являются главной особенностью Z-System. Каждый системный сегмент остается в памяти до тех пор, пока новый системный сегмент не загружен поверх него. Например, вы можете иметь несколько файлов на диске с расширением RCP, таких как WORDPRO.RCP для обработки текста, ASM.RCP для работы с ассемблером, CALC.RCP для электронных таблиц и т.д. Каждый файл будет содержать разный набор резидентных команд, используемых для различных целей. Если вы введете команду, "LDR WORDPRO.RCP", то утилита LDR загрузит резидентные команды WORDPRO.RCP в область памяти, отведенную для пакетов резидентных команд, и перезапишет любой другой набор ранее загруженных резидентных команд. Вот почему мы говорим, что пакеты в системе Z-System динамически изменяемые.

Теперь мы переходим к каждому из пакетов и сегментов, загруженных вашим файлом STARTUP.COM.

## Дескриптор среды

Дескриптор среды - это файл данных. Он загружается в зарезервированное для него место в памяти и предоставляет информацию всем утилитам Z-System о конфигурации Z-System, в которой работает эта утилита. Эта информация включает адреса самих системных сегментов, расположения буфера сообщения (в котором утилиты Z-System оставляют сообщения друг другу и операционной системе о различных системных вопросах), число подключенных дисков, максимальную область пользователя, различные флаги, показывающие, должны ли утилиты распечатать определенную информацию на CRT, или работать "тихо", и может ли утилита распознать DU и обозначения DIR или только обозначения DIR. Кроме того, дескриптор среды может включать ZCPR3 TCAP или запись возможностей терминала, который описывает атрибуты вашего терминала или консоли CRT, например коды, необходимые для очистки экрана, установки позиции курсора, активации повышенной яркости изображения и т.п. Наконец, дескриптор среды содержит определение двух логических терминалов и четырех логических принтеров. Эти терминалы и принтеры могут быть выбраны утилитой CPSEL.COM. Короче говоря, почти каждая значительная характеристика Z-System определена в дескрипторе среды.

## Пакет команд управления выполнением

Пакет команд управления выполнением содержит ZCPR3 команды управления выполнением. Команды управления выполнением - резидентные команды, которые дают вам многие возможности которые, на больших и малых ЭВМ называют языком управления заданиями. На простом уровне команды управления выполнением позволяют вам (даже если вы - абсолютно не программист) программировать ряд команд и устанавливать условия для выполнения одной или более команд. На системном уровне команды управления устанавливают одно из двух состояний управления - истина или ложь. Если состояние управления - истина (система автоматически проверяет это), могут быть обработаны все команды. Если состояние управления ложно, только команды управления выполнением могут быть обработаны. Основными командами управления выполнением являются IF, ELSE, FI и XIF. FI ("IF в обратном порядке") является эквивалентом ENDIF для тех из вас, кто выполнял некоторое программирование. XIF выход из всех незаконченных IF. Давайте попробуем некоторые команды управления выполнением.

Создайте файл под названием HELLO в вашем текстовом редакторе или текстовом процессоре и поместите некоторый текст в него. Введите DIR<cr>. Вы увидите файл HELLO в перечислении каталога. Введите следующую командную строку:

IF EXIST HELLO;TYPE HELLO;ELSE;ECHO HELLO DOES NOT EXIST;FI<cr>

Вы должны видеть текст, который вы ввели в файл HELLO. Теперь попробуйте это:

IF EXIST HELLOO;TYPE HELLOO;ELSE;ECHO HELLOO DOES NOT EXIST<cr>

Система вежливо скажет вам, что файл HELLOO не существует. (Теперь не говорите нам, что у вас был файл под названием HELLOO, находящийся без дела на вашем диске все это время. Мы вам не верим.)

Обычно доступны следующие команды управления выполнением:

|  |  |  |  |
| --- | --- | --- | --- |
| **Команда** | **Краткая форма** | **Команда** | **Краткая форма** |
| IF | отсутствует | ERROR | ER |
| ELSE | отсутствует | EXIST | EX |
| FI | отсутствует | INPUT | IN |
| XIF | отсутствует | NULL | NU |
| EMPTY | EM |  |  |

Синтаксис этих команд управления выполнением прост.

IF проверяет, является ли определенное условие True (истина). Если условие True, то состояние управления выполнением устанавливается в True. Это значит, что разрешено выполнение любой следующей команды. Если условие False (ложно), то состояние потока имеет значение False, и разрешается выполнить только команды управления выполнением.

Время для демонстрации. Введите, "IF IN" и ответьте на вопрос N. Теперь попробуйте некоторые нормальные команды (любые кроме команд управления выполнением), например DIR. Нет, ваш компьютер не поврежден. Что произошло? "IF IN" приглашает вас к вводу с клавиатуры. Если вы отвечаете T, Y, <cr> или пробел (<sp>), состояние управления выполнением устанавливается True, и вы можете продолжить выполнять другие команды. Если вы вводите что-либо другое, состояние управления выполнением устанавливается в False, и вы можете выполнить только команды управления выполнением. На самом деле, вы лучше выполните команду управления выполнением прямо сейчас, чтобы вернуть компьютер в более полезное состояние. Введите FI. Теперь попробуйте некоторые свои обычные команды, такие как DIR. Все "вернулось в норму".

FI завершает текущий уровень IF и возвращается к следующему, "активному состояние IF". Допустим, вы используете команду управления выполнением, которая имеет эффект установки состояния управления выполнения False. Тогда FI возвращает к предыдущему состоянию управления выполнением, которое, если оно True, теперь устанавливает в системе состояние управления выполнением в True. Проще говоря, FI имеет то же смысл что и ENDIF в стандартной терминологии программирования. В Z-System существуют девять состояний управления выполнением, или уровней, пустое или состояние "нет активных IF" (не следует путать с командой управления выполнения "IF EMPTY"), которая всегда является истиной и уровни 1 - 8, которые могут быть установлены пользователем.

Очень важно помнить, что командный процессор ZCPR3 всегда в курсе текущего состояния управления выполнением Z-System. Если он находится в состоянии True, командный процессор разрешает выполнение любой команды, если доступна команда пользователя. Если состояние управления выполнением - False, то могут быть выполнены только команды управления выполнением. Команда IF повышает в системе состояние управления выполнением до следующего уровня и устанавливает состояние управления выполнением в True или False в зависимости от результата тестирования некоторого условия. Когда вы используете команду IF IN, вы просите систему протестировать, данные введенные пользователем. Нажатие клавиш T, Y, <cr>, или <sp> устанавливает условное тестирование в состояние True, нажатие любой другой клавиши устанавливает его в состояние False. Так как вы нажали "любую другую клавишу", вы повысили состояние управления выполнением на один уровень и установили его в состояние False. Именно поэтому система больше не повиновалась ни одной из ваших команд, кроме команды управления выполнением. Когда вы ввели команду FI, вы опустили состояние управления выполнением на один уровень вниз. Так как состояние управления выполнением находилось в состоянии True, вы также установили состояние управления выполнением в True. Если бы предыдущее состояние управления выполнением было False (в результате предыдущего условного теста, который привел к состоянию управления выполнением False), то вам был бы нужен еще один FI, чтобы заставить систему повиноваться вашим командам, не являющимся командами управления.

Команда управления выполнением ELSE переключает состояние управления выполнением. Если текущее состояние управления выполнением True, ELSE переключает его в False. Если текущее состояние управления выполнением False и предыдущее состояние управления выполнением True, ELSE переключает состояние управления выполнением в True. Если предыдущее состояние управления выполнением - False, ELSE ничего не делает.

Команда управления выполнением XIF выходит из всех незавершенных **IF**, если состояние управления выполнением - True. Т.е. она уменьшает состояние потока до 0 или пустого состояния, которое всегда является True. Если состояние управления выполнением - False, XIF ничего не делает.

Очевидно, что команды управления выполнением наиболее полезны при пакетной обработке, когда вы хотите, чтобы компьютер выполнил некоторое действие при одном условии или иное действие при другом условии.

Прежде чем мы начнем некоторые практические упражнения, давайте ознакомимся до конца с остальной частью команд управления выполнением.

|  |  |  |
| --- | --- | --- |
| **Команда** | **Краткая форма** | **Значение** |
| EMPTY afn | EM | Если обозначенный файл пуст (размер - 0K), состояние управления выполнением установлено в True |
| ERROR | ER | Если флаг ошибки в программе ZCPR3 установлен, состояние управления выполнением установлено в True |
| EXIST afn | EX | Если обозначенный файл существует, состояние управления выполнением установлено в True |
| INPUT | IN | Пользователь получает запрос на ввод, и если ответ T, Y, <cr> или <sp>, состояние управления выполнением установлено в True - любой другой ответ устанавливает состояние управления выполнением в False |
| NULL afn | NU | Если 2-я запись (afn) в командной строке оставлена незаполненной, то состояние управления выполнением устанавливается в True |

Параметр "afn" в таблице команд управления выполнением означает "неоднозначное имя файла" или, по сути, это означает файл который вы выбираете, включая обозначение групповых файлов. Вспомните карточные игры! Подстановочные знаки могут иметь любое значение, которое вы присваиваете им. Наконец, ведущая тильда (~) перед условием инвертирует действие состояния. Если условие имеет значение False, состояние управления выполнением имеет значение True, и если условие имеет значение True, состояние управления выполнением имеет значение False. Таким образом "IF ~EXIST MYFILE.TXT" устанавливает состояние управления выполнением значение True, если файл MYFILE.TXT, не существует.

Не волнуйтесь, если это кажется немного сложным, в действительности это не так, и в любом случае, мы собираемся сделать некоторые практических упражнения вместе, чтобы вы практически могли изучить команды управления выполнением, даже если остальные из вас чувствует себя довольно уверенно в этом отношении.

Предполагая, что вы храните вашу ZCPR3 утилиту SHOW.COM в именованном каталоге с названием ROOT, войдите в ваш каталог ROOT, введя ROOT:<cr>. Теперь получите список файлов в каталоге с помощью команды DIR<cr>. Вы увидите файл SHOW.COM, перечисленный среди файлов в каталоге, таким образом, вы убедитесь, что он существует на вашем диске. Попробуйте выполнить следующую команду управления выполнением: "IF EX SHOW.COM;DIR;FI". Вы должны увидеть список файлов каталоге на вашем экране CRT. Что происходит? Вы говорите системе, если файл SHOW.COM существует поднять состояние управления выполнением на один уровень и установить его в True. Так как ясно, что SHOW.COM, существует, состояние управления выполнением устанавливается в True. Следовательно, программе DIR.COM разрешено выполниться. Завершающий FI опускает управления выполнением вниз на один уровень к состоянию 0, которое всегда является True. (Безусловно, имя каталога ROOT и файл SHOW.COM выбраны произвольно. Вы можете выполнить этот эксперимент в любом каталоге с любым файлом.)

Что происходит, если вы выбираете файл, который не существует? Давайте попробуем.

IF EX NOFILE.TXT;ECHO IT EXISTS;ELSE;ECHO IT DOES NOT EXIST;FI

Пожалуйста! Резидентная команда ECHO просто сказала, то что вы уже знали - файл, NOFILE.TXT не существует. Давайте попробуем другой вариант:

IF ~EXIST SHOW.COM;ECHO THE FILE SHOW.COM DOES NOT EXIST;ELSE;ECHO IT DOES;FI

Он действительно существует. Первое условие установило состояние управления выполнением в False, потому что файл, SHOW.COM действительно существует, и следующая команда, не являясь командой управления выполнением отключена. Вспомните, когда состояние управления выполнением False, могут быть выполнены только команды управления выполнением. Следующая команда управления выполнением ELSE. Если текущее состояние управления выполнением - False, которая является 0, и предыдущее состояние управления выполнением - True (это правда), то ELSE переключает состояние потока к True. Таким образом следующая команда может выполниться. Эта команда, ECHO IT DOES, отображает "IT DOES" на ваш экран.

Команды управления выполнением используются прежде всего для добавления интеллектуального автоматизированное принятие решений в пакетной обработке Z-System, а не в интерактивных практических упражнениях которые мы выполняли. Но мы будем подойдем к пакетной обработке - использовании псевдонимов и мощной замены Z-System SUBMIT/XSUB - ZEX немного позже.

В настоящее время, вы освоили азы управления выполнением и мы можем перейти к другим вопросам. Вспомните, мы недавно загрузили пакет управления потоком с помощью STARTUP.COM. Перейдем к следующему пакету, другому системному сегменту ZCPR3.

## Пакет резидентных команд

Резидентные команды постоянно (пока компьютер включен или до загрузки другого файла RCP) находятся в памяти, в отличие от нерезидентных команд, которые загружаются и выполняются из файлов COM. Резидентные команды ZCPR3 загружаются в память из файла на диске с помощью LDR.COM. Пакет резидентных команд обычно называется SYS.RCP. Если у вас есть подходящий макроассемблер, как ZAS Эшелона {Echelon} или аналогичный ему, вы можете ассемблировать комплект различных резидентных объектов и загружать различные пакеты резидентных команды на лету.

Чтобы просмотреть доступные резидентные команды, введите H<cr>. Экран должен выглядеть следующим образом:

B0:WORK>**H**

SYS 1.1A

CLS CP ECHO ERA

LIST P POKE R

REN TYPE

B0:WORK>

Эти десять команд постоянно находятся в памяти после загрузки некоторой конкретной SYS.RCP в составной командной строке, переданной системе программой STARTUP.COM. Из-за предпочтений пользователей и аппаратных особенностей, многие реализации Z-System будут иметь частично другие RCP.

|  |  |
| --- | --- |
| **Команда** | **Назначение** |
| **CLS** | Очищает экран. |
| **CP** | Команда копирования одного файла. |
| **ECHO** | Отправляет вывод на консоль, включая Escape последовательности. |
| **ERA** | Стирает файлы. |
| **LIST** | Распечатывает файл или группу файлов на принтере. |
| **P** | PEEK. Используется для изучения области памяти. |
| **POKE** | Изменение содержимого памяти. |
| **R** | Сброс дисковой системы (регистрирует новый диск). |
| **REN** | Изменяет имя файла. |
| **TYPE** | Отображает содержание файл или группу файлов ASCII на экране CRT. |

Многие резидентные команды очевидны, но некоторые заслуживают некоторого краткого обсуждения.

**CP**

CP копирует один файл из одного DU (диск/пользователь) к другой DU или из одного DU в тоже DU с другим именем файла. Команда CP очень быстрая, но не проверяет ошибки. Если вам нужно или вы хотите иметь проверку ошибок, используйте транзитную утилиту MCOPY.

Примеры:

CP MYFILE.TXT=OLDFILE.TXT

Копирует OLDFILE.TXT в новый файл в тоже DU с именем MYFILE.TXT.

CP WORK:=ROOT:MYFILE.TXT

Копирует файл MYFILE.TXT из каталога ROOT в каталог WORK.

CP WORK:=A15:MYFILE.TXT

Копирует файл MYFILE.TXT из DU A15 в каталог WORK.

CP B0:=MYFILE.TXT

Копирует файл MYFILE.TXT из текущего DU в DU B0.

CP B0:=A15:MYFILE.TXT

Копирует файл MYFILE.TXT из DU A15 в DU B0.

**ECHO**

Эхо «повторяет» сообщения на экране CRT. Она также может отправить Escape последовательности на экран CRT. Эти функции можно использовать в псевдонимах и других командных файлах для некоторых интересных визуальных эффектов.

Попробуйте это:

ECHO ^Z I HAVE JUST CLEARED THE SCREEN

или:

ECHO ^Z

ECHO ^Z является командой очистки экрана для наших терминалов. Вашему может потребоваться другая строка. В этом случае, ECHO [оказывается независимо от вашей строки очистки экрана] будет работать. Попытайтесь отправить некоторые последовательности управления в свой терминал с помощью команды ECHO. Получаете ли вы несколько идей больших видео эффектов?

Я наслаждаюсь ECHO в моей Z-System.

В самом деле, вы узнали о том, что ECHO преобразует символы нижнего регистра в верхний регистр.

**ERA**

ERA стирает файлы. Синтаксис - ERA dir:afn (где afn означает неоднозначное имя файла). Вы можете также ввести ERA dir:afn I, где I или параметр проверки предлагает приглашение Yes/No для каждого файла в списке.

Примеры:

ERA ROOT:\*.\* I

Удаление всех файлов в каталоге ROOT с запросом подтверждения для каждого файла перед стиранием.

ERA B0:\*.bak

Удаление всех файлов в B0 с расширением BAK без подтверждения.

ERA MYFILE.TXT

Удалить файл MYFILE.TXT без подтверждения перед стиранием.

**LIST**

LIST распечатывает один или группу файлов ASCII на устройство LST: (обычно принтер).

Примеры:

LIST B7:\*.ASM

Печать всех файлов в B7 с расширением ASM.

LIST ROOT:MYFILE.TXT

Печать MYFILE.TXT расположенного в ROOT.

**P**

Команда PEEK показывает байты в памяти. Ввод "P", без определения адреса памяти выводит на экран следующие 256 байтов памяти. Команда "P EA00" выводит на экран 256 байтов памяти, начиная с адреса EA00. Следующий "P" выводит на экран следующие 256 байтов. "P EA00 EAFF" выводит на экран память с адреса EA00 до адреса EAFF.

**POKE**

Команда POKE изменяет байты в памяти.

POKE помещает конкретные фрагменты данных в определенные места в памяти. Будьте очень осторожны с командой POKE, потому что она может поместить любое значение в любое место, включая области памяти, содержащие операционную систему.

Формат команды POKE следующий

POKE ADDRESS VALUE

где "ADDRESS" - адрес в памяти, и "VALUE" - значение, которое будет помещено в тот адрес. Команда

POKE 800 0

помещает шестнадцатеричное значение 0 в шестнадцатеричный адрес 800H.

POKE 800, "THIS IS A TEST"

(завершающая кавычка не обязательна) помещает значения ASCII символов "THIS IS A TEST" в ячейке памяти, начиная с 800H.

POKE 800 1 2 3 "THIS IS A TEST"

поместит значения 1, 2 и 3 в память, начиная с 800H, с последующими значениями ASCII символов "THIS IS A TEST" сразу после 3. Обратите внимание, что как только вы начинаете помещать "текст" в память (использующий кавычку, чтобы отметить начало текста), все, что следует далее интерпретируется как текстовый символ. Не существует возврата к вводу значений шестнадцатеричных чисел.

Давайте убедимся, что команда POKE делает то, что утверждает.

POKE 800 "Z IS A POWERFUL SYSTEM<cr>

теперь для подтверждения:

P 800<cr>

вы должны увидеть значения ASCII недавно введенные с помощью команды PEEK, проверяя правую сторону экрана, похожую на вывод старого доброго DDT. Конечно, команда POKE работает, но иногда неплохо убедиться в этом самостоятельно.

Команда POKE имеет огромное значение для пользователя Z-System, поскольку она допускает модификацию программы в памяти. Например, если вы выполняете WordStar, вы знаете, что можете использовать программу установки, чтобы создать много различных версий со многими различными значениями по умолчанию. Если вы знаете, где в WordStar находятся байты, которые управляют этими значениями по умолчанию, вы можете загрузить WS.COM в память с помощью команды "GET 100 WS.COM" и затем поместить в эти байты необходимые значения. Если у вас есть способ помещения много байтов с помощью одной команды, вы сможете использовать только одну копию WordStar и помещать необходимые значения в WS.COM во время выполнения. Мы покажем как это сделать позже. В данный момент просто запомните, что метод, который мы обсуждали, называют методом "Поместить и перейти". Это одна из многих вещей, которые сделали творческие пользователи Z-System для экономии времени, сил, места на диске и обеспечения более высокой производительности их аппаратного и программного обеспечения.

**R**

R сбрасывает дисковую систему. Вы когда-нибудь испытывали страх получить ошибку BDOS, сообщающую вам, диск только для чтения? Действительно, вы не должны получать такую ошибку в ZRDOS, но вы не можете не беспокоиться по этому поводу. По крайней мере, некоторые пользователи компьютера испытывают его. R позволяет удостоверится, что недавно вставленный диск зарегистрирован правильно. Также полезно убедиться, что информация о месте на диске, предоставленная программами каталога, верна для вновь вставленной дискеты.

**REN**

REN переименовывает файлы.

REN MYFILE.TXT=MY.DOC

переименовывает файл MY.DOC в MYFILE.TXT.

Вы не обязаны быть в том же DU где файл, чтобы переименовать его.

REN ROOT:MYFILE.TXT=MYFILE

переименовывает MYFILE в ROOT в MYFILE.TXT.

При попытке переименовать файл к имени файла, который уже существует, вас запросят, должен ли существующий файл быть удален. Вы можете переименовать файлы со статусом системный, но не со статусом только для чтения.

**TYPE**

Резидентная команда TYPE используется для отображения файлов на экране CRT или консоли.

TYPE MYNOVEL.TXT

отображает MYNOVEL.TXT на консоли с паузами после заполнения экрана, ожидая любого ввода с клавиатуры для продолжения отображения до следующего заполнения экрана. Это называют разбивкой на страницы.

TYPE MYNOVEL.TXT P

выводит на экран файл без разбивки на страницы, точно так же, как резидентная команда TYPE в CP/M.

TYPE допускает неоднозначное имя файла, поэтому вы можете последовательно отобразить несколько файлов с помощью одной команды, таких как TYPE \*.DOC, которая последовательно отображает все файлы с расширением DOC, независимо от их имени.

## Именнованные каталоги

Программа STARTUP загружает, его как последний пакет установки именованных каталогов.

Именованные каталоги присваивают имена каталогам Z-System. Каталоги Z-System, как мы уже обсуждали, являются логическими каталогами, известными по букве диска и номеру области пользователя. A0, B3, A15 и B0 все являются каталогами Z-System. Используя именованные каталоги ZCPR3, вы можете обратиться к каждому каталогу, например A0, по выбранному вами имени. Пакет именованных каталогов SYS.NDR на вашем дистрибутивном диске присваивает имена двум каталогам. A0 называется COMMAND и A15 называется ROOT. Мы выбрали эти имена. Вы можете предпочесть другие.

С именованными каталогами работа на компьютере более удобна и производительна. При выборе имен, вы можете исходить из своего комфорта. Многим людям нравится называть B0 WORK в системах использующих гибкие диски. В системе с жестким диском вы можете выделять рабочие области в зависимости от задачи выполняемой в этой области. B3 можно назвать ASM, потому что в ней будете ассемблировать программы. Вы можете выбрать свое имя в качестве имени каталога, если используете один компьютер совместно с другими пользователями. В системах с гибкими дисками вы можете сопоставить имена своих каталогов к виду задачи, для которой используете диск в дисководе A. Так, например, свою дискету используемую для обработки текста, можете назвать B0 - WORDPRO.

Утилита, которая позволяет вам изменить имена каталогов называется MKDIR. MKDIR создает файлы именованных каталогов для последующей загрузки утилитой LDR. Как любые утилиты Z-System, ее использование объясняется в файле встроенной справки, вызвать который можно, введя MKDIR //.

Так выглядит MKDIR когда вы вызываете его:

B0:WORK>**mkdir**

MKDIR, Version 3.2

MKDIR Command (? for Help)? **?**

MKDIR Commands are --

C -- Change Directory (Add/Rename/Delete Entries)

I -- Initialize Directory

P -- Print Directory

R -- Read Directory File

S -- Status of MKDIR Environment

W -- Write Directory File

X -- Exit Program

MKDIR Command (? for Help)? **R**

Name of File (<RETURN> = B 0: NAMES .NDR)? A15:SYS.NDR

MKDIR Command (? for Help)? **P**

DU : DIR Name - Password DU : DIR Name - Password

---- -------- -------- ---- -------- --------

A 0: COMMAND - A15: ROOT -

B 0: WORK -

MKDIR Command (? for Help)? **S**

\*\* MKDIR Status \*\*

3 Entries in Directory

Working File Name: SYS .NDR

No Changes made to Directory since Startup

File has been loaded

MKDIR Command (? for Help)? **X**

B0:WORK>

Этот рабочий сеанс показывает, как команда MKDIR "R" читает выбранный вами файл .NDR. Мы выбрали SYS.NDR, расположенный в A15: или в именованном каталоге с именем ROOT. Команда "P" распечатывает именованные каталоги на ваш экран CRT. Если вы хотите изменить какой-либо из именованных каталогов или добавить новые, вы используете команду "C". Если вы хотите A0 присвоить имя BASE вместо COMMAND, то вы должны ввести A0:BASE (в режиме "изменения", вызванном командой MKDIR "C" ), и имя будет изменено. Чтобы сохранить изменения и дополнения к файлу, вы должны выти из режима "изменения" с помощью команды "X", сопровождаемой возвратом каретки, и затем использовать команду записи "W". Так как вы уже сказали MKDIR, что имеете дело с SYS.NDR, расположенным в A15, вы просто нажимаете возврат каретки после команды "W" для сохранения новой версии SYS.NDR на ваш диск в дисководе A. Для загрузки новых именованных каталогов в память, вы вводите "LDR SYS.NDR". И у вас есть свой, недавно определенный набор именованных каталогов.

Прежде чем мы закончим, наше обсуждение MKDIR давайте посмотрим на сеанс, в котором каталог был изменен:

A0:REMOTE>**mkdir remote.ndr**

MKDIR, Version 3.2

MKDIR Command (? for Help)? **P**

DU : DIR Name - Password DU : DIR Name - Password

---- -------- -------- ---- -------- --------

A 0: SYSTEM - A 1: SB180 -

A 2: TEXTPROC - A 3: ASSEMBLE -

A 4: ZRDOS - A 5: DRICPM -

A 7: SYSOP - HORRORS! A 8: PUBLICS - WONTWORK

A 15: ROOT - NOTHERE!

B 0: UPLOADS - B 1: ZCPR3 -

B 2: UTILITY - B 3: COMPILE -

B 4: DOCUMENT -

MKDIR Command (? for Help)? **C**

\*\* MKDIR Change Mode \*\*

Directory Entry (?<RETURN> for Help)? **a1:naogzsig**

Renaming SB180

Directory Entry (?<RETURN> for Help)? **x**

MKDIR Command (? for Help)? **X**

Directory has changed since last Write

Do you want to write Directory to Disk (Y/N)? **Y**

Name of File (<RETURN> = A 0: REMOTE .NDR)? **test.ndr**

Writing Directory to Disk ... Done

Обратите внимание, MKDIR спрашивает вас об имени выходного файла перед созданием нового файла .NDR. Если пользователь ответит на запрос "Name Of File" (Имя файла) возвратом каретки, исходный REMOTE.NDR в A0 будет перезаписан. Создавая сначала новый именованный файл каталога в виде TEST.NDR, пользователь мудро оставляет исходный файл в покое, пока он не попробует измененную версию, после чего он может использовать команду REN или CP (или утилита) для изменения к первоначальному имени.

Утилита PWD позволяет увидеть, какие именованные каталоги вы определили и загрузили в память. Так выглядит PWD, когда вы вызываете его.

B0:WORK>**pwd**

PWD, Version 1.0

DU : DIR Name DU : DIR Name DU : DIR Name DU : DIR Name

---- -------- ---- -------- ---- -------- ---- --------

A 0: COMMAND A15: ROOT

B 0: WORK

B0:WORK>

Мы видим, что существуют три рабочих каталога, загруженные из нашего диска: COMMAND, ROOT и WORK. Каждый именованный каталог связан с диском и областью пользователя. Эти диски и области пользователя перечисляются перед именем под PWD. По существу PWD сочетает в себе функции команд MKDIR "R" и "P" и позволяет быстро посмотреть активные в данный момент каталоги, если вы забыли.

Команда смены каталога CD обеспечивает вам еще один способ перехода из одного именованного каталога в другой именованный каталог. Вы видели, что мы можем перейти из B0 в A15, введя A15:<cr>. Мы можем сделать то же самое, введя "ROOT:<cr>". CD позволяет нам вводить "CD A15:<cr>" или "CD ROOT:<cr>", чтобы сделать то же самое. Вы можете задаться вопросом, зачем надо вводить CD в дополнение к другой информации, если можно просто ввести ROOT:<cr>. Не вдаваясь в подробности, одно из преимуществ использования команды CD для перехода в новый именованный каталог состоит в том, что, когда CD переместит вас в новый каталог, первое что она делает - ищет файл с названием ST.COM. Если она находит ST.COM, то выполняет его. ST.COM может быть любой программой, которую вы можете переименовать в ST.COM. Она также может быть (и обычно является) псевдонимом. Псевдоним представляет собой файл COM, наподобие STARTUP (имя ST происходит от "STartup"), который просто передает ряд команд, встроенных в него в систему для выполнения. Таким образом, при регистрации в новый именованный каталог с помощью команды CD, может быть автоматически выполнен ряд команд. Более подробно о псевдонимах поговорим позже.

Именованные каталоги позволяют вычислять так, как вы думаете. Вы можете счесть трудным помнить, что выполняете ассемблирование в B3, но у вас будет меньше трудности с запоминанием именованного каталога под названием ASM. Допустим, вы хотите скопировать определенные файлы из вашего именованного каталога ASM в ваш каталог BACKUP. Легко запомнить, что ваш диск и область пользователя для резервных копий называется BACKUP и что ваш диск и область пользователя для работы с ассемблером называют ASM.

Чтобы скопировать файл под названием BIGPROG.ASM из каталога ASM в каталог BACKUP, вам просто нудно ввести:

CP BACKUP:=ASM:BIGPROG.ASM

Резидентная команда CP не осуществляет проверки ошибок. Современное оборудование довольно надежное, и многие пользователи обменивают проверку ошибок на скорость CP. Если вы хотите автоматически выполнять проверку ошибок в процессе чтения и записи, можете использовать программу MCOPY и ввести:

MCOPY BACKUP:=ASM:BIGPROG.ASM

Утилиты каталогов Z-System также знают об именованных каталогах. Чтобы посмотреть, какие файлы находятся в вашем каталоге ROOT, вы можете ввести "XDIR ROOT:", "XD ROOT:" или просто "DIR ROOT:", в зависимости от используемой утилиты каталога Z-System. Другие утилиты Z-System также знают об именованных каталогах. Именованные каталоги предоставляют вам мощный инструмент, позволяющий свести вычисления в область нормального повседневного английского языка. Это просто еще один пример того, как Z-System обеспечивает наиболее практичный и естественный пользовательский интерфейс, доступный сегодня на микрокомпьютерах.

Хорошо....что мы уже здесь В конце STARTUP. И мы только начали наше путешествие в Z. В этот момент вы должны чувствовать себя комфортно с понятиями управления выполнением, резидентных команд и именованных каталогов. Мы перейдем теперь к другим важным вопросам, включая резидентные команды, составную командную строку и псевдонимы.

# Другие команды Z-system

## Встроенные команды

ZCPR3 содержит некоторые "встроенные" команды, совершенно не связанные с резидентными командами и командами управления выполнением. Они существуют вне зависимости от факта загрузки пакетов команд управления выполнением и резидентных команд. Мы называем их "внутренние" команды.

Ваша система содержит следующие встроенные команды:

|  |  |  |
| --- | --- | --- |
| **GET** | - | Загрузка файла в память. |
| **GO** | - | Повторное выполнение последней транзитной команды. |
| **JUMP** | - | Ответвление к адресу в памяти. |
| **NOTE** | - | Создание комментария (отключает последующую интерпретацию команды). |
| **SAVE** | - | Запись образа памяти на диск. |

Эти команды являются краеугольными камнями некоторых из самых мощных приложений Z-System.

**GET**

Команда GET используется для загрузки файла в любое место памяти. На практике, так как файлы COM всегда загружаются по адресу 100H, вы будете использовать команду GET для загрузки файлов COM по их адресу выполнения. Таким образом, команда GET 100 WS.COM загружает WordStar по его рабочему адресу выполнения 100H. В сочетании с резидентной командой POKE и встроенной командой GO, GET обеспечивает возможность изменения программы на лету, непосредственно перед запуском. (Мы скоро будем развивать приложение WordStar более подробно.)

**GO**

Команда GO используется, чтобы повторно выполнить последнюю программу, загруженную в память в 100H, без ее повторной загрузка. Если вы вводите "DIR", и после просмотра списка каталогов, введете "GO", вы повторно выполните DIR.COM. "Что тут такого?" говорите вы. Действительно большое дело! В дополнение к предоставлению быстрого и простого способа повторного выполнения программы, команда GO является еще одним краеугольным камнем метода положить и перейти.

Метод положить и перейти позволяет вам иметь одну копию WordStar на вашем диске обработки текста и, все же, быть в состоянии загружать много различных версий WordStar с множеством различных значений по умолчанию. Предполагая, что WS.COM и его оверлейные программы находится на диске A, можно ввести следующую последовательность команд:

A0:COMMAND>**GET 100 WS.COM**

A0:COMMAND>**POKE 36B 4D**

A0:COMMAND>**POKE 36E 0**

A0:COMMAND>**GO**

Эта последовательность команд загружает WordStar по адресу его выполнения 100H, изменяет байт по адресу 36BH на 4DH, изменяет байт по адресу 36EH на 0 и выполняется (выполняет) WordStar измененный в памяти. Если вы знаете адреса точек исправления своего WordStar (и вы можете найти их, вызвав ваш местный Z-Node для получения файла, перечисляющего точки исправлений вашей версии WordStar), вы можете использовать, этот метод положить и перейти для исправления WordStar во время его загрузки в память, непосредственно перед выполнением. До свидания WINSTALL.COM! Конечно, есть более эффективные способы обработки ввода методом положить и перейти, чем вручную в командной строке, как мы увидим позже.

**JUMP**

Команда JUMP используется для ответвления или "перехода" к любому адресу в памяти. Вот интересная иллюстрация: введите "JUMP 0" и нажмите возврат каретки. Вы только что использовали JUMP для "Теплой" загрузки вашей системы! Дело в том, что команда JUMP может использоваться для перехода к любому адресу в памяти и, если она находит код для выполнения, начнет его выполнение. Еще более простой способ вызвать "Теплую" загрузку с помощью JUMP состоит в том, чтобы ввести только "JUMP" без операнда (хвоста команды) вообще, "0" подразумевается. Код, найденный в расположении 0, является вектором (командой перехода 8080/Z80) к подпрограмме BIOS, выполняющей "Теплую" загрузку (перезагрузку) ZCPR3.

**NOTE**

Команда NOTE позволяет Вам вводить комментарии, не получая сообщения об ошибках от системы. Попробуйте это:

THIS IS A NOTE<cr>

Что происходит? Вы получаете сообщение об ошибке. Теперь попробуйте это:

NOTE THIS IS A NOTE

Сообщение об ошибке нет! Почему? Команда NOTE, в действительности, отключила интерпретацию команды. Система больше не думает, что Вы пытаетесь выполнить команду под названием "THIS". Команда NOTE позволяет комментировать командные файлы, не вызывая сообщения об ошибках. Хотя точка с запятой в начале командной строки также отключает интерпретацию команды, как в CP/M, команда NOTE более полезна, потому что она может быть включена в составную командную строку.

**SAVE**

Команда SAVE используется для сохранения содержимого области транзитных программы (TPA) в файл на диске. Прежде всего мы используем ее для модификации программ с помощью отладчиков, таких как ZDM и DSD Эшелона {Echelon} или DDT Digital Research. Эта команда прежде всего для продвинутых пользователей, которые изменяют свою систему или "исправляют" файлы COM.

SAVE допускает два параметра (иначе говоря, две части информации, которые можно ввести после слова "SAVE", чтобы сказать ей что делать): число страниц для сохранения и имя файла, в который вы хотите сохранить страницы памяти. Если имя файла сопровождается "S", тогда Z-System предполагает, что вы хотите сохранить определенное число 128-байтовых секторов. Если "S" опущен, число относится к числу 256-байтовых страниц, которые вы хотите сохранить.

Давайте посмотрим конкретный терминальный сеанс, использующий команду SAVE для создании файла при генерации новой системы. Предположим, что существующая операционная система находится в файле с именем Z3TB58-3.SYS. (В случае, если вы любопытны, авторы устанавливали Advent/Plu\*Perfect TurboROM в свой Kaypro, поэтому, часть имени файла Z3TB относится к системе Z3 для TurboROM, и 58 напоминает нам, что это система 58K.) После накладывания измененной версии ZCPR3 мы создадим файл с именем Z3TB58-4.SYS, который содержит нашу измененную операционную систему.

Вот он:

B0:WORK>**DDT** **Z3TB58-3.SYS**

DDT VERS 2.2

NEXT PC

2500 0100

-**IZCPR3.HEX**

-**R3F00**

NEXT PC

2500 0000

-**G0**

"Теплая" загрузка

B0:WORK>**SAVE 36 Z3TB58-4.SYS**

Давайте внимательно посмотрим на то, что мы сделали.

Во-первых, вызван Dynamic Debugging Tool (DDT), который поставляется с большинством компьютеров с CP/M. Мы просим DDT загрузить файл Z3TB58-3.SYS.

Во-вторых, мы используем команду "I", чтобы установить блок управления файлом для файла ZCPR3.HEX. На некомпьютерном языке мы сказали системе, что собираемся прочитать файл ZCPR3.HEX.

В-третьих, мы используем команду "R", чтобы считать файл ZCPR3.HEX с соответствующим смещением для нашей системы. В этой конкретной системе для Kaypro с Advent TurboROM, мы используем смещение 3F00. Соответствующее смещение зависит от генерируемой вами системы.

В-четвертых, мы используем команду "G0" для "Теплой" загрузки системы. В этот момент на файл Z3TB58-3.SYS, находящийся в памяти, накладывается файл ZCPR3.HEX с заданным смещением.

Наконец, мы используем команду SAVE, для сохранения 36 256-байтовых страниц в файл с именем Z3TB58-4.SYS.

Как нам удалось подсчитать, сколько страниц сохранять? Ответ прост. Смотрите на шестнадцатеричное число под NEXT после того, как DDT загрузит Z3TB58-3.SYS. Это 2500. Для расчета числа сохраняемых страниц, возьмите два крайних левых числа и преобразуйте их в десятичное число. Шестнадцатеричное 25 или 25H равняются 2 х 16 плюс 5 х 1. Ответ 32 плюс 5 или 37. Если число под NEXT заканчивается двумя нулями, мы вычитаем 1, таким образом, число сохраняемых страниц равняется 36. Если последние два числа под NEXT не являются двумя нулями, вычитать 1 не требуется. Этот расчет необходим для каких-либо существенных работ по изменению операционной системы. Однако...

Z-System даже упрощает эту задачу - вычисление не требуется, если добавить "h" к шестнадцатеричному счетчику базовой страницы.

SAVE 25h Z3TB58-4.SYS<cr>

решает проблему!

## Редактирование строки и управление выводом

ZRDOS предоставляет функции редактирования строки Z-System. Функциями редактирования строки являются команды, которые вы можете использовать для редактирования командной строки, когда находитесь в системном приглашении, таком как A0:COMMAND>. Функции редактирования строки следующие:

|  |  |
| --- | --- |
| DEL | Удаление последнего символа, введенного в консоли. Это совпадает с Backspace. |
| Ctrl-H | Удаление последнего символа набранного в консоли и возврат на один символ. |
| Ctrl-U | Удаление всей командной строки. |
| Ctrl-X | Удаление всей командной строки и возврата в начало текущей строки. |
| Ctrl-E | Отмечает физический конец строки. Курсор возвращается, но командная строка не передается до нажатия клавиши RETURN. |
| Ctrl-M | Прерывает ввод с символом возврата каретки. |
| Ctrl-J | Прерывает ввод с символом перевода строки. |
| Ctrl-C | Перезагрузка Z-System ("Теплая" загрузка) |

Ниже приведены выходные функции:

|  |  |
| --- | --- |
| Ctrl-P | Копирует весь последующий консольный вывод на устройство LST:. Вывод идет к LST: устройство (принтер, обычно) и консоли до следующего нажатия Ctrl-P.  (Стоит отметить, что многие инструменты ZCPR3 и ZRDOS выводят символы через BIOS, а не ZRDOS.  Пакет IOREC Echelon (если ваша Z-System поддерживает сегменты IOP) или утилиты общественного достояния как SWAP или REDIR рекомендуются для захвата вывода консоли BIOS на устройство LST: или в файл на диске.) |
| Ctrl-S | Остановка консольного вывода до следующего ввода символа. Ввод Ctrl-C возвращает управление Z-System. |

Ваши входные строки могут быть довольно длинными. В дистрибутивной системе входная строка может быть в ширину приблизительно два с половиной экрана. Вы можете иметь такую длинную входную строку, потому что Z-System позволяет помещать несколько команд во входную строку, разделенных точкой с запятой (";").

## Составная командная строка ZCPR3

Составная командная строка, со многими методами доведения последовательности команд до командного процессора, является еще одним стержнем Z-System.

Что представляет из себя составная командная строка? ZCPR3 позволяет пользователю помещать несколько команд в одной командной строке, разделенные точкой с запятой. Например:

A0:COMMAND>WORK:;DIR \*.BAK;ERA \*.BAK;DIR;COMMAND:

допустимая командная строка. ZCPR3 интерпретирует эту командную строку в качестве запроса пользователя для входа в именованный каталог с названием WORK, вывода списка файлов в каталоге с расширением .BAK, удаления всех этих файлов и снова вывода списка файлов каталога, чтобы убедиться, что все эти файлы были фактически стерты и наконец возврата в именованный каталог COMMAND первоначальной регистрации.

Составная командная строка дает много преимуществ.

Во-первых, Вы можете ввести ряд сложных команд, нажать клавишу Return и оставить систему чтобы сделать что-то другое, пока не будут выполнены все команды.

Во-вторых, вы можете поместить любую составную командную строку (до заполнения предоставленного места в буфере, которое составляет приблизительно две с половиной ширины экрана) в упрощенный файл пакетной обработки, и уменьшите длинную, сложную команду до нажатия одной клавиши. Файлы псевдонима помещают составную командную строку в фактическую выполняемую программу. Процессор пакетной обработки команд ZEX, также работающий с составной командной строкой, и меню Z-System получают большую часть своей функциональности от возможности уменьшить составную командную строку до нажатия пользователем одной клавиши.

Таким образом составная командная строка, в сочетании с сценариями различных форм предоставляет вам возможности, немыслимые в CP/M. Сценарий представляет собой сохраненную на диске последовательность команд, которая может быть выполнена, как будто это всего одна команда. Таким образом, последовательность команд:

A0:COMMAND>WORK:;DIR \*.BAK;ERA \*.BAK;AC BACKUP:=\*.TXT;ERA \*.TXT;COMMAND:

может быть уменьшен до единственного командного сценария CLEANUP. Мы будем видеть, как создать его с помощью файлов ALIAS в следующей главе.

# Псевдонимы

## Что такое псевдоним?

Псевдоним (ALIAS) является для кого-то или чего-то альтернативным именем. Мы говорим "Билл Джонс" также известный как "Человек в маске". "Человек в маске" является псевдонимом Билла Джонса или другим именем.

Аналогично, Z-System ALIAS - также другое имя для чего-то. "Что-то" является рядом команд, в конечном счете отправленных в буфер командной строки для обработки ZCPR3.

Строго говоря псевдонимом является файл COM, созданный утилитой ALIAS, содержащий одну или несколько команд, разделенных точкой с запятой, которые передаются в буфер командной строки.

Утилита ALIAS помещает фактически любую последовательность команд в файл COM, ограниченную только размером буфера командной строки. Каждый раз, когда вы выполняете файл COM, созданный утилитой ALIAS, эта последовательность команд обрабатывается процессором команд ZCPR3. Эта последовательность команд имеет псевдоним (другое имя), и это другое имя (псевдоним), является именем файла COM, созданного вами. То что когда-то имело много имен, теперь сводится к одному имени. Что раньше требовало многих нажатий клавиш, теперь требует только сумму нажатий, необходимую для ввода имени псевдонима файла COM и его хвоста команды, если он имеется.

Ряд команд, сохраненных в файле COM, называют *сценариями* и COM файл, в котором хранится сценарий, помещает сценарий в буфер командной строки, когда имя псевдонима используется в качестве команды Z-System. В файл COM могут также передаться параметры из командной строки таким же способом, как в CP/M SUBMIT, с расширениями, и полностью расширенный (фактические параметры командной строки заменяют формальные символы), сценарий будет выполняться. Это станет понятней, когда мы будем подробно рассматривать реализацию псевдонимов.

## Утилита ALIAS

Утилита ALIAS создает файл COM, который передает команду или ряд команд в операционную систему. Утилита ALIAS вызывается следующим образом:

|  |  |  |
| --- | --- | --- |
| ALIAS | - | определить псевдоним файла COM |
| ALIAS filename | - | отобразить команды, хранящиеся в "filename.COM" и при необходимости редактировать эти команды. |

Ниже приводится подлинная терминальная сессия, демонстрирующая использование утилиты ALIAS:

B0:WORK>**ALIAS**

ALIAS, Version 1.1

Input Alias (RETURN to Abort)

--> **WORK:;DIR \*.BAK;DIR \*.REL;ERA \*.BAK;ERA \*.REL;DIR**

Name of ALIAS Command (RETURN to Abort)? **CLEANUP**

Alias Created

B0:WORK>

Мы уже создали псевдоним CLEANUP, состоящий из ряда команд, разделенных точкой с запятой. Теперь мы можем использовать утилиту ALIAS, для изучения команд, хранящихся в CLEANUP:

B0:WORK>**ALIAS CLEANUP**

ALIAS, Version 1.1

Alias Name: CLEANUP

Old Alias Command Line:

1 --> **WORK:;**

2 --> **DIR \*.BAK;**

3 --> **DIR \*.REL;**

4 --> **ERA \*.BAK;**

5 --> **ERA \*.REL;**

6 --> **DIR**

Input Alias (RETURN to Abort)

-->

B0:WORK>

Утилита ALIAS показывает каждую команду на отдельной строке для удобства пользователя.

## Переменные ALIAS

Ниже приводится краткое описание переменных (символов формальных параметров), которые вы можете использовать в теле псевдонима:

|  |  |
| --- | --- |
| $0 | Имя псевдонима |
| $n | Параметр командной строки (где 'n' - целое число от 1 до 9) |
| $\* | Хвост командной строки (все, что следует за командой) |
| $D | Домашний диск (диск, с которого вы начали) |
| $U | Домашний пользователь (область пользователя из которой вы начали) |
| $Fn | FILENAME.TYP системного файла 'n' (где 'n' - целое число от 1 до 4) |
| $Nn | FILENAME системного файла 'n' |
| $$ | Символ $ |

Параметр - то, что вы вводите после имени команды. Имя команды определяет команду. Когда вы вводите WS, чтобы вызвать WordStar, WS это имя команды. Когда вы вводите "WS MY.DOC", MY.DOC это параметр. Параметр следует за именем команды и говорит WordStar, какой файл вы хотите редактировать. Переменные $0 это имя самого псевдонима. Переменная $\* является хвостом командной строки - всей после имени команды, независимо от числа параметров.

Системные файлы, упомянутые выше, являются именами файлов, которые хранятся в памяти. Есть четыре таких имени, разрешенные ZCPR3. Утилита SETFILE определяет эти имена.

## Примеры псевдонимов

Теперь, давайте посмотрим на группу псевдонимов от ряда пользователей на различных уровнях знаний и опыта Z-System. Некоторые будут представлены как фактические экраны ALIAS.COM, другие будут простыми списками команд, которые вы можете превратить в псевдонимы с помощью ALIAS.COM, используя стандартный разделитель команд точку с запятой.

### Пример 1: ASSEMBLE

Вы устаете от ввода одних и тех же команд много раз, чтобы ассемблировать и загрузить файл:

ASM myfile.bbz

LOAD myfile

Решение Z-System состоит в том, чтобы создать псевдоним:

ASM $1.BBZ;LOAD $1

Давайте назовем псевдоним ASSEMBLE.COM. Теперь, когда вы ассемблируете файл с названием TEST.ASM, вы вводите, "ASSEMBLE TEST". Это - эквивалент ввода "ASM TEST.BBZ;LOAD TEST", но вы уменьшили число нажатий на клавиши.

### Пример 2: ARCHIVE

У вас есть система с жестким диском, и Вы хотите иметь возможность резервного копирования только тех файлы, которые изменились в определенном именованном каталоге. Попробуйте этот псевдоним:

AC BACKUP:=$1:\*.\* /A

Вы можете вызвать этот псевдоним ARCHIVE.COM. Если вы введете "ARCHIVE TEXT" архивная утилита резервного копирования AC скопирует файлы из именованного каталога TEXT, архивный бит которых не был установлен (т.е. новые файлы в этом каталоге) в именованный каталог BACKUP.

### Пример 3: I.COM

Каждый раз, когда вы устанавливаете утилиту, вы должны ввести "Z3INS ROOT:SYS filename.COM", где "filename" это имя новой утилиты. Как насчет псевдонима, чтобы сократить нажатия клавиш?

Z3INS ROOT:SYS $1.COM

Давайте вызовем псевдоним I.COM. Теперь, чтобы установить утилиту, filename.COM, введите "I filename", и все готово. С Z-System немного подумав сейчас можно избежать многих проблем позже.

### Пример 4: WSTAR

Вы выполняете WordStar и используете XtraKey, для загрузки измененной конфигурации клавиатуры, вам нравятся в качестве значений по умолчанию 78 столбцов, перенос слов выключен, и дефис помощи отключен. Посмотрите это:

ROOT:;

XK WS.XPD;

$D$U:;

GET 100 COMMAND:WS.COM;

POKE 36B 4D;

POKE 36E 00;

POKE 34F 00;

GO;

XK $$X

Этот псевдоним, который мы называем WSTAR.COM, входит в каталог ROOT, вызывает XtraKey, загружает файл XtraKey с определённой нами конфигурацией клавиатуры, возвращается к DU, в котором мы были зарегистрированы когда вводили WSTAR, загружаем WS.COM в память по его адресу выполнения 100H, помещаем необходимые значения в WordStar, чтобы изменить значения по умолчанию в соответствии с нашими пожеланиями и запускаем выполнение WordStar. После того, как мы заканчиваем с WordStar, XtraKey удаляет себя из памяти и псевдоним завершается.

### Пример 5: DIRB

Вы хотите иметь простой способ проверки содержимого дискеты в диске B, независимо от того, двухсторонняя она или односторонняя, проверяя все области пользователей. Помните, что внутренняя команда ZCPR3 JUMP вызывает "теплую" загрузку, если вызывается без операнда, см. 5.1.

JUMP;XDIR B:\*.\* U

Этот псевдоним вызывается введя DIRB.COM

### Пример 6: NEWDISK

Это - отредактированная версия составленного пользователем псевдонима с названием NEWDISK.COM. При создании нового системного диска, вы непременно захотите иметь на нем некоторые файлы, такие как ваш STARTUP.COM, ваши пакеты, LDR.COM и возможно другие файлы. Зачем вводить с клавиатуры имена всех этих файлов?

B0:WORK>ALIAS NEWDISK

ALIAS, Version 1.1

Alias Name: NEWDISK

Old Alias Command Line:

1 --> CP B:=A:SYS.ENV;

2 --> CP B:=A:LDR.COM;

3 --> CP B:=A:STARTUP.COM;

4 --> CP B:=A:SYS.FCP;

5 --> CP B:=A:SYS.RCP;

6 --> CP B:=A:SYS.NDR;

7 --> CP B:=A:PWD.COM;

Псевдоним использует резидентную RCP команду CP для копирования всех файлов, которые этот пользователь считает необходимыми на системных дисках.

### Пример 7: COMPILE

Еще один от того же пользователя. Он ненавидит вводить с клавиатуры ряд команд для выполнения компиляции с помощью своего компилятора C (да, некоторые люди используют высокоуровневые языки хотя все из нас, кто стремится писать эффективные CP/M-совместимые программы, должны изучить современные ассемблерные методы с SYSLIB Ричарда Конна {Richard Conn}, это не так сложно). Это его псевдоним:

B0:WORK>ALIAS COMPILE

ALIAS, Version 1.1

Alias Name: COMPILE

Old Alias Command Line:

1 --> CC $1 $2 $3 $4;

2 --> A:M80 =$1;

3 --> L80 $1,A:CRUNLIB/S,$1/N/E;

4 --> ERA $1.MAC;

5 --> ERA $1.REL

Предполагая, что этот пользователь выполняет свою работу на языке C будучи зарегистрированным на диске B, вы можете обнаружить ненужную команду здесь? Какой значительный элемент Z-System не задействует этот пользователь? Средство поиска PATH, конечно! В строке 2 ему не нужен "A:M80". Его путь поиска - A0 A15. Даже при том, что он зарегистрирован на диске B, командный процессор будет искать файл M80.COM на диске A в области пользователя 0. Еще один совет новым пользователям Z-System. Забудьте свои старые привычки. Расширьте свое мышление для , использования новых методов экономящих время.

### Пример 8: M8

Этот от одного из опытных пользователей. На самом деле он является одним из основных авторов Z-System. Для ассемблирования утилит ZCPR3 он рекомендует следующее:

M80 =$1;L80 $1/P:100,VLIB/S,Z3LIB/S,SYSLIB/S,$1/N/E

Давайте посмотрим содержимое его ALIAS, названного M8.COM.

Вы запускаете M8, вводя "M8 NEWUTIL", где NEWUTIL - файл с расширением MAC. Файл MAC является исходным файлом на ассемблере, подготовленным для компиляции ассемблером M80 Microsoft. (Многие пользователи Z-System заменили M80 прекрасным ассемблером ZAS, поставляемым Эшелоном {Echelon} и ZAS, конечно, является хорошим выбором, если у вас имеется только древний ASM.COM.)

Что же делает М8 и как он это делает? Первая команда, переданная ALIAS, является "M80 =$1". Если вы введете "M8 NEWUTIL" псевдоним будет знать, что первый параметр - NEWUTIL и развернет команду до "M80 =NEWUTIL". Эта команда ассемблирует файл NEWUTIL.MAC, создавая файл NEWUTIL.REL. Вызывается редактор связей L80 и NEWUTIL собирается (ALIAS подставляет NEWUTIL вместо $1 во время его расширения), с начальным адресом 100H, извлекая необходимые подпрограммы из трех стандартных библиотек Z-System: VLIB, Z3LIB и SYSLIB.

### Пример 9: WSTR

У вас есть два принтера - параллельный матричный Okidata и последовательный принтер. Вы используете версию Z-System с перенаправляемым вводом-выводом, и у вас есть две версии WordStar - одна для параллельного принтера Okidata и другая настроенная для последовательного принтера. Смотрите это (команды слева):

**Сценарий Значение**

IF OKI=$2 Проверить, является ли 2-й параметр OKI

DEV L LPT Если да, назначить устройство LST к LPT

WSO $1 и выполнить WS в версии OKI

ELSE Если нет ...

DEV L TTY назначить LST к TTY

WST $1 и выполнить WS в версии TTY

FI и установить состояние управлением команд в True

Этот ALIAS вызывается вводом WSTR.COM. Если вы вводите "WSTR NEW.DOC", без 2-го параметра, то OKI=$2 - False и псевдоним расширится до "DEV L TTY;WST NEW.DOC". Если вы вводите "WSTR NEW.DOC OKI", OKI=$2 (где $2 расширяется до OKI) - True, и псевдоним расширится до "DEV L LPT;WSO". Будет выполняться версия WordStar для параллельного принтера Okidata. Разумеется, вам необходимо реализовать пакет IO, чтобы выполнить этот ALIAS, а также соответствующие модификации BIOS, но если у вас есть система с IOP, вы готовы к работе.

## Исследования

Упомянутые выше сценарии псевдонимов являются всего несколькими из бесчисленно возможных. Каждый раз, когда вы используете повторные нажатия клавиш для любого процесса, или каждый раз, когда вы находите, что забыли некоторую сложную команду и должны искать ее, вспоминайте об ALIAS - начинайте смотреть на программы как на инструменты, соединенные командными сценариями для выполнения задач. Псевдонимы являются одной из самых мощных возможностей Z-System, которая поможет вам достичь максимальной эффективности в работе с компьютером.

# ZEX, MENU и VFILER

## Введение в расширенную обработку команд

В этой главе мы рассмотрим четыре самые мощные в Z-System (некоторые могли бы сказать необычные) программы: ZEX, размещенный в памяти процессор командных файлов, MENU и VMENU, самые популярные "пользовательские интерфейсы" для Z-System и VFILER, который сочетает экранно-ориентированное управление файлами с программируемыми пользователями параметрами в стиле меню. Мы не пытаемся задокументировать каждую функцию этих инструментов. Это всесторонне сделано в ZCPR3: Руководстве и системе файловой справки ZCPR3. Наша единственная цель состоит в том, чтобы убедить вас пробовать силы в освоении их. Они были специально разработаны, чтобы предоставить пользователю Z-System средство настройки компьютеризированного рабочего места, адаптации его в соответствии с индивидуальными требованиями. Они обеспечивают возможность программирования, без программирования в общепринятом смысле этого слова. Если подходить с осторожностью и пытливым умом, они позволяют вам настроить свою вычислительную среду до такой степени, которая недоступна ни одной другой операционной системе для микрокомпьютеров.

## ZEX

ZEX - размещенный в памяти процессор командных файлов, сочетающий функций CP/M SUBMIT и XSUB, но добавляющий скорость и возможности далеко за пределами того, что эти программы предлагают в CP/М. По ряду причин ZEX часто предпочтительнее псевдонимов в качестве инструмента выполнения командного сценария:

1. Длина командных сценариев в файлах ZEX не ограничена размером буфера составной командной строки ZCPR3. Расширение параметров командной строки в псевдониме позволяет легко столкнуться с этим ограничением.
2. ZEX имеет гораздо больше функций и возможностей, чем предлагает ALIAS (или, если на то пошло, SUB). Отображения на экране и пользовательские подсказки настраиваемы в очень элегантный пользовательский интерфейс.
3. ZEX может посылать заданные сценарием команды в интерактивную программу так же, как командная строка ZCPR3. ZEX может обеспечить ввод для текстового редактора, текстового процессора или отладчика. Например, файл ZEX может вызвать WordStar, послать WordStar внутренние команды для выполнения поиска и замены, выйти в систему, вызвать MCOPY или CP для помещения копии измененного файла во многие назначенные каталоги, загрузить образ системы с помощью DDT или ZDM Эшелона {Echelon}, выполнить операцию внесения исправлений внутри отладчика, выйти и выполнить операцию генерации системы, все без присмотра.
4. ZEX имеет более мощные возможности условного исполнения чем псевдоним или SUB, который позволяет истинное однонаправленное условное ветвление путем использования GOTO.COM, являясь специальным инструментом ZCPR3. Эта функция добавляет еще один аспект использования управления выполнением в командных скриптах.
5. ZEX позволяет параметры по умолчанию которые будут созданы, используя их в расширении скрипта в случае, если соответствующие параметры в исходной командной строке отсутствуют.
6. Как в SUB, но в отличие от псевдонима, исполнение ZEX может быть легко остановлено с клавиатуры с помощью Ctrl-C.

ZEX имеет два режима: интерактивный режим и режим командного файла.

Чтобы войти в интерактивный режим, введите ZEX<cr>. В интерактивном режиме пользователь вводит командные строки до тех пор, пока больше не останется команд для ввода. В этот момент, введя пустую командную строку (нажатие RETURN в пустой строке) заставит ZEX начать выполнение команд в порядке, в котором они были введены. Вы не можете передавать параметры из исходной командной строке в интерактивном режиме, но вы можете установить их в середине сессии. Вот одна из демонстрации ZEX в интерактивном режиме автора ZCPR3 Ричарда Конна {Richard Conn}. Вы можете "ощутить" то, что Ричард делал на своей собственной клавиатуре просто введя в ZEX.

### Пример 1: Интерактивный режим ZEX

Этот пример иллюстрирует ZEX в интерактивном режиме.

B0:WORK>**zex**

ZEX, Version 3.1

1: **^$ this is fun** <-- Определение 3 параметров

2: **echo $1 $2 $3**

3: **^$ hello from happy acres** <-- Определение 4 параметров

4: **echo $1 $2 $3 $4**

5:

(ZEX активен) <-- ZEX сейчас работает

B0:WORK>echo this is fun

THIS IS FUN

B0:WORK>echo hello from happy acres

HELLO FROM HAPPY ACRES

B0:WORK>

(ZEX завершен)

Вашей командой >

Давайте исследуем, что делается в примере сеанса работы с терминалом. Во-первых, ZEX вызван без хвоста команды. Директива ZEX ^$ (два символа, а не Ctrl-$) определяет последующие записи, разделенные пробелами в одной строке, как параметры ZEX. Затем команда RCP ECHO говорят вывести на экран эти вновь созданные параметры. Затем мы снова используем ^$, чтобы назвать четыре новых параметра и выполнить ECHO для вывода их на экран. Ввод RETURN (возврат каретки) в пустой строке говорит ZEX начинать обрабатывать команды, которые он делает, выполняя как команду ECHO, так и "скрывая" директивы ZEX на экране.

Чтобы войти в командный файл, или режим пакетной обработки, вводят "ZEX filename parameters", где "filename" - любой пакетный файл ZEX с расширением SUB или ZEX, и "parameters" - предоставленные параметры любого надлежащего пользователя. В этом режиме ZEX будет искать вдоль пути filename.ZEX или filename.ZEX и, если он находит такие файлы, начнет обрабатывать их. Если каталог по пути пользователя содержит файлы и с расширением ZEX и SUB с одинаковыми именами, то ZEX обработает файл ZEX.

В любой момент ZEX может быть прерван вводом Ctrl-C в консоли. ZEX не поддерживает вложения, поэтому если процессор ZEX встретится с другим вызовом ZEX в пакетном файле, он прервется. Предусмотрены много мощных, встроенных, расширенных команд управления, делая ZEX значительно более эффективным, чем XSUB CP/M при действии в качестве замены "консоли робота" для ввода данных пользователя в прикладные программы.

Следующий командный файл ZEX использует управление выполнением команд ZCPR3 для выбора между двумя ассемблерами, стандартным ассемблером CP/M ASM.COM и ZAS Эшелона {Echelon}. Кстати мощный, современный ассемблер аналогичный ZAS является обязательным, если вы хотите попытаться поправить "внутренности" Z-System и ее утилит. Программа MLOAD NightOwl Software является значительно улучшенной заменой команды CP/M LOAD и доступна в свободном доступе по модему с Z-Node или в группах пользователей, включая NAOG/ZSIG.

### Пример 2: Использование управления выполнением

Этот файл использует расширения файла \*.ASM и \*.Z80 для идентификации двух различных диалектов ассемблеров и выбирает необходимый ассемблер. Имя выходного файла, отличное от источника, может быть указанно в исходной командной строке.

;

; Ассемблирование с Echelon ZAS 2.2a или DRI ASM 2.0

; с загрузкой с помощью MLOAD 2.4 от NightOwl

;

; ^& Suppress FALSE IF Printout

;

IF NUL $1

ECHO \*\* Для ассемблирования требуется имя файла \*\*

ELSE

IF EX $1.Z80 ; NOTE Если Мнемоника Zilog

ZAS $1 H ; NOTE Выполнить ассемблировани с ZAS

ELSE

ASM $1.BBZ ; NOTE нет, использовать ASM (надеюсь нет макросов)

FI

IF INPUT Abort if Errors Exist (^C), <cr> to Load

ERA $1.BAK ; NOTE Удалить файл BAK

IF ~NUL $2 ; NOTE Если существует файл с выходным именем

ERA $2 ; NOTE удалить его текущего владельца и

MLOAD $2=$1 ; NOTE загрузить HEX файл с этим именем

ELSE

ERA $1.COM ; NOTE Нет, поэтому просто удалить старый COM-файла

MLOAD $1 ; NOTE и обычным образом загрузить Hex файл

FI

FI

ERA $1.HEX ; NOTE Удалить HEX файл в любом случае

FI

;

; Операция завершена

;

Давайте разберем файл, который называется ASMBL.ZEX. Все строки, начинающиеся с точек с запятой и все команды NOTE, будут выведены на экран, но не вызовут любых действий ZCPR3. Директиве ^& повинуется ZEX даже при том, что ей предшествует точка с запятой. Это говорит ZEX не выводить любые строки на экран, когда, IF имеет состояние - FALSE. Первая строка проверяет присутствие операнда командной строки и распечатывает сообщение об ошибке, если они отсутствуют. Обратите внимание, что состояние IF затем переключается в FALSE с командой ELSE, поэтому если операнды отсутствуют, выполняются только команды управления выполнением. Если операнд существует, он используется в качестве имени файла следующих команд IF EX. При выполнении файла командой "ZEX ASMBL FILE" эта строка расширится до "IF EX FILE.Z80". Расширения имен \*.Z80 используются для исходных файлов с мнемониками Zilog, которые обрабатываются ZAS. Если Файл Z80 не существует, то следующая команда ELSE считает (с некоторым риском), что существует FILE.ASM, использующий мнемоники Intel, который будет ассемблирован с помощью ASM.

Затем следует команда FI (ENDIF), которая выходит из уровня IF, основанного на расширении файлов. Любой ассемблер производит выходной файл \*.HEX, так как, в данный момент мы должны иметь файл FILE.HEX и некоторое сообщение на экране, указывающее об успешности или неуспешности процесса ассемблирования. Это где, команда "IF INPUT", с ее консольным сообщением опроса, просит разрешение продолжить (<cr>, возврат каретки) или сигнал аварийного прекращения работы (^C, Ctrl-C). Если разрешение получено, следующая команда стирает файл резервной копии (\*.BAK) файл, оставленный редактором, который произвел исходный файл. Проверка "IF ~NUL $2", подразумевает действует, "если существует второй параметр командной строки", и, раз так MLOAD будет проинструктирован, использовать его в качестве выходного имени файла. Она позволит вам ассемблировать FILE.Z80 или FILE.ASM в MYFILE.COM или что-то подобное. Без второго параметра имя исходного файла будет использоваться для выходного файла \*.COM. Одно упражнение, оставленное вам, проследить выполнение команд сценария, если будет отсутствовать существенный первый параметр в предположении, что вы просто вводите "ZEX ASMBL", что произошло бы и почему?

Важно понимать, что число символов, задействованных в ASMBL.ZEX, исключает реализацию его в виде ALIAS, таким образом, теперь становится очевидным основное преимущество ZEX. ZEX может обработать длинные сценарии с многословными консольными сообщениями, ALIAS не может.

ASMBL.ZEX очень мало использует специальные внутренние директивы ZEX, в основном он осуществляет управление выполнением команд, которое можно реализовать с помощью SUB со значительным снижением скорости. Следующий пример, слегка отредактированный из одного из написанных автором ZRDOS Деннисом Райтом {Dennis Wright}, показывает, как красиво ZEX может управлять отображением консоли и иллюстрирует силу внутренних директив ZEX. Он является сложным примером использования ZEX, созданным вокруг возможности ассемблера ZAS "сообщить" об ошибках ассемблирования в ZCPR3 через флаг ошибки программы, зарезервированный байт в буферной области памяти сообщений ZCPR3. Он также иллюстрирует способ, который используется для создания большинства утилит ZCPR3 и ZRDOS - главный исходный файл на языке ассемблера компилируется в перемещаемый объектный модуль, который соединяется с подпрограммами, извлеченными связывающим загрузчиком ZLINK из библиотек подпрограмм. Публикация Эшелона {Echelon}, ZCPR3: Библиотеки, исследует этот способ более подробно.

### Пример 3: ZASLINK

Это - сценарий условного ассемблирования, использующий управление выполнением команд с Z-System ассемблером ZAS Эшелона {Echelon} и связывающим загрузчиком ZLINK. "Структурированный" стиль этого файла упрощает отслеживание различных уровней активности присутствующих IF. Чем дальше вправо начинается команда, тем выше уровень активного IF.

^.

^#

^<^|

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;^|

; ;^|

; ZASLINK.ZEX -- Z-System ZAS Macro Assembler and ZLINK Linker ;^|

; with ZAS error checking. ;^|

; ;^|

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;^|

^>

^&

if ~exist $1.Z80 ;note if file does not exist^|

ECHO ^G \*\*\* $1.Z80 NOT FOUND \*\*\*^|

else^|

ZAS $1^|

if ~er ;note if no assembly errors^|

ERA $1.COM^|

ZLINK $1,A:PDMLIB/, A:VLIB/,A:Z3LIB/,A:SYSLIB/ $$C100^|

ERA $1.REL^|

ERA $1.BAK^|

Z3INS A15:SYS $1.COM^|

ECHO -- ZASLINK COMPLETE^|

else^|

ECHO ^G \*\*\* FATAL ERROR IN ASSEMBLY, ZASLINK ABORTED \*\*\*^|

fi ;note if ~er^|

fi ;note if ~exist^|

^#

Вместо того, чтобы пройти через ZASLINK.ZEX шаг за шагом вместе, вероятно, более полезно на данном этапе исследовать его самостоятельно, обращаясь к следующему списку расширенных команд управления, доступных в ZEX. Управление выполнением команд прямое, и более автоматизировано, чем в ASMBL.ZEX, потому что он использует возможность ZAS оповестить ZCPR3 о наличии ошибок ассемблирования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Команда** | **Значение** |  | **Команда** | **Значение** |
| | | Вставить <CR> |  | ^| | Вставить <CR><LF> |
| ^: | Повторно выполните командный файл |  | ^. | Подавить печать символов |
| ^# | Переключить сообщения ZEX |  | ^$ | Определить параметры по умолчанию |
| ^? | Ожидать от пользователя <CR> |  | ^/ | Выдать звуковой сигнал и ожидать <CR> |
| ^\* | Выдать звуковой сигнал |  | ^" | Разрешить ввод данных пользователем |
| ^< | Отображать только символы |  | ^> | Остановить отображение |
| $& | Подавить отображение когда, IF имеет состояние - FALSE | | | |
| ;; | Комментарий ZEX |  | $n | 1<=n<=9 для параметров |
| $$ =$ |  |  | $^ =^ |  |
| $| =| |  |  | ^c | Вставить символ Ctrl-C |

Эти команды могут быть встроены в текст командного файла или предоставлены пользователем в интерактивном режиме, и они будут выполняться после того как начинается обработка ZEX.

Следующие команды ZEX являются командами вставки символов. Т.е. они используются для вставки символов в поток команд ZEX:

|  |  |  |  |
| --- | --- | --- | --- |
| | | Вставить <CR> | ^| | Вставить пару <CR><LF> |
| $$ | Вставить один символ $ | $^ | Вставить один символ ^ |
| $| | Вставить один символ | | ^c | Вставить управляющий символ |

^\* Заставляет ZEX выдать звуковой сигнал на консоли

;; Комментарий ZEX

^< и ^> Символы между этими "скобками" отображаются на экране пользователя во время выполнения, но не обрабатываются как часть потока команд.

^# Переключает подавление информативных сообщений ZEX

^. Заставляет прекратить вывод на консоль до появления следующиего "^."

^: Заставляет ZEX перезапустить выполнение всего потока команд с самого начала

$n Заставляет параметр, указанный или по умолчанию, использовать значение из командной строки. где n - значение от 1 до 9

^$ Заставляет рассматривать остальную часть строки в качестве новый набора параметров, разделенных пробелами

^? ZEX прекращает обработку и ожидает <SP> или <CR>, чтобы продолжить обработку (Ctrl-C прерывает)

^/ Как команда ^?, но выдает звуковой сигнал для привлечения внимания пользователя

^" Заставляет ZEX прекратить принимать входные данные из потока команд и ждать ввода пользователя (который будет выключен символом ">").

Как вы видите только по числу функций, ZEX является командным процессором с большими возможностями, но также и довольно сложным. Мы предполагаем, что экспериментирование - лучший способ изучить его возможности.

В качестве последнего примера командных файлов ZEX, приведен сценарий, который сочетает в себе автоматизированный выбор ассемблирования ASMBL.ZEX с проверкой ошибок ZCPR3 в ZASLINK.ZEX. Также, эта сокращенная версия ASMREL.ZEX знакомит с утилитой GOTO, которая позволяет ZEX по условию пропускать любое количество командных строк, возобновляя выполнение в метке GOTO, которая принимает форму ";=LABEL". Это позволяет ASMREL.ZEX включать дополнительные библиотеки подпрограмм при компоновке в качестве параметра исходной командной строки. ASMREL.ZEX основывается на M80.ZEX составленного Ричардом Конном {Richard Conn}.

### Пример 4: Выбор ассемблера в сочетании с проверкой ошибок

Этот командный файл ZEX использует два ассемблера, один специально разработанный, для использования ZCPR3 флага ошибки программы и другой измененный для этого. В дополнение к двум библиотекам подпрограмм по умолчанию у пользователя есть параметр для включения в начальную командную строку до четырех библиотек специального назначения во время выполнения компоновки.

;

; Echelon ZAS 2.3 and DRI RMAC 1.1 Assemblers

; with NightOwl PROLINK 1.5 Linking Loader

;

; ^& Suppress FALSE IF Printout

;

IF NUL $1

ECHO \*\* File Name Required for Assembly \*\*

ELSE

IF EX $1.Z80 ;NOTE Zilog Mnemonics?

ZAS $1 ;NOTE Perform Assembly With ZAS

ELSE

RMAC $1 $$-S PZ;NOTE No, Use Modified RMAC

FI

IF ~ER

ERA $1.BAK ;NOTE Cleanup BAK File

ERA $1.COM ;NOTE Cleanup Old COM File

IF ~NUL $5 ;NOTE Link 4 Additional Libraries

PROLINK ORIGIN 100&LINK $1&SEARCH $2,$3,$4,$5,Z3LIB,SYSLIB&EXIT

GOTO DONE

FI

IF ~NUL $4 ;NOTE Link 3 Additional Libraries

PROLINK ORIGIN 100&LINK $1&SEARCH $2,$3,$4,Z3LIB,SYSLIB&EXIT

GOTO DONE

FI

IF ~NUL $3 ;NOTE Link 2 Additional Libraries

PROLINK ORIGIN 100&LINK $1&SEARCH $2,$3,Z3LIB,SYSLIB&EXIT

GOTO DONE

FI

IF ~NUL $2 ;NOTE Link 1 Additional Library

PROLINK ORIGIN 100&LINK $1&SEARCH $2,Z3LIB,SYSLIB&EXIT

GOTO DONE

ELSE ;NOTE Standard Link

PROLINK ORIGIN 100&LINK $1&SEARCH Z3LIB,SYSLIB&EXIT

;=DONE Done with Link

FI ;NOTE on IF ~NUL Tests

FI ;NOTE on IF ~ER

ERA $1.REL

FI ;NOTE on IF NUL

;

; Операция завершена

;

Эти примеры дают некоторое представление о возможностях и силе ZEX, но ваша задача найти дополнительные подходящие приложения. Достаточно сказать, что ZEX может быть бесценным инструментом в автоматизации вычислительных задач. Как и большая часть Z-System, он хорошо выполняет простые задачи и является достаточно мощным для роста возможностей с пользователем, по мере приобретения им знаний и уверенности. Просто начните и продолжите экспериментировать. ZEX (и Z-System) вознаградит ваши усилия.

## Подсистема меню

"Подсистема меню" ZCPR3 опирается на два ориентированных на меню командных процессора - MENU и VMENU. Обе эти утилиты считывают свои данные из файлов ASCII, которые могут быть созданы любым стандартным текстовым редактором, включая WordStar. MENU использует файлы c расширением MNU. VMENU использует файлы c расширением VMN. Формат этих файлов может быть проверен утилитами MENUCK и VMENUCK.

MENU представляет собой строчно-ориентированный процессор. VMENU экранно-ориентированный и полагается на присутствие TCAP ZCPR3. В остальном, эти две программы очень похожи.

### MENU

MENU - оболочка ZCPR3, которая читает текстовый файл с расширением \*.MNU, распечатывает экран согласно настроек дисплея(ев), установленных в том файле, и обрабатывает команды из сценариев, которые он содержит.

Синтаксис MENU:

**MENU** <-- выполнить MENU.MNU

или

**MENU ufn** <-- выполняет меню содержащееся в файле с однозначным именем файла

Когда программа MENU вызывается пользователем, она ищет файл MENU.MNU в текущем каталоге. Если она находит этот файл, то загружает его и начинает обрабатывать. Если она не находит файл, она ищет файл с именем и расширением .MNU. В случае неудачи она прерывается. Давайте исследуем MENU.MNU и объясним некоторые функции MENU в дальнейшем.

**Пример 1:**

Этот MENU.MNU организует группу программ в взаимосвязанную и простую в использовании систему каталогизации дисков. Пакет Эшелона {Echelon} DISCAT является гораздо более изысканной реализацией этой концепции.

#DPX

**Master Catalog System**

Defaults: Master Catalog on drive A:

Update disk on drive B:

**1 ) Create** a new Master Catalog file

**2 ) Label** a disk Volume

**3 ) Update** the Master Catalog

**4 ) Find** specific file(s) with a match key

**5 ) List** the entire Master Catalog

**6 ) View** the disk directory

**7 ) Purge** a disk Volume

**8 ) Examine** a disk Volume

**Enter your selection (1 - 8):**

#

1 era mast.cat;zex create <<

"List filenames.ext to be excluded, separated by commas: "

2 era b0:-\*.\* i;save 0 b0:-"Volume label <<

(VOLUME.NBR, 'VOLUME'=7 characters or less): "

3 mcat b:;echo CONTROL-C ABORTS MAST.LST UPDATE;xcat

4!find51 mast.lst "Enter matching key (string): "

5!type mast.lst

6!dir b:

7 zex purge ."Enter volume number to be purged (all 3 <<

digits): ";echo CONTROL-C ABORTS MAST.LST UPDATE;xcat

8!find51 mast.lst "Enter volume number to be examined (3 digits): "

##

Буквы после первого знака решетки (#) обозначают доступные параметры данного меню. Буква D означает, что параметры меню отображаются, P означает, что экран разбит на страницы, так, чтобы экран меньшего размера, чем полный экран увеличивается при нажатии возврата каретки и параметр X обеспечивает выход в меню ZCPR3. Единственным пунктом меню, который не используется в этом файле, является C, который выводит на экран командную строку, построенную MENU, которая будет выполнена, полезен в основном для отладки.

Все от строки ниже строки параметров до следующего знака решетки - фактический экран меню, отображаемым пользователю. Вот и все, это так просто. MENU использует командную строку, представленную внизу экрана, который для многоэкранного MENU.MNU выглядит следующим образом:

Command (<CR>=Menu,^C=ZCPR3,\*=1st Menu,<=Prev Menu,>=Last Menu) -

Раздел MENU.MNU с фактическими командами MENU отправляемыми в ZCPR3 для выполнения начинается ниже второго знака решетки и заканчивается строкой выше двойного знака решетки. Команды не выводятся на экран, и правила синтаксиса являются (неукоснительно) строгими. Первый символ каждой командной строки является ее "инициатором", единственное нажатие на клавишу, которого вызовет ее. Фактическая командная строка начинается сразу после этого символа, пробелы недопустимы. Сами командные строки очень похожи на те, что вводятся в командную строку ZCPR3 или ALIAS.COM с расширениями. Так как некоторые командные строки в файле слишком длинные для нормальных книжных полей, поэтому мы взяли на себя смелость разбить их двумя стрелками указывающими влево "<<", чтобы указать, где они объединяются. Просто запомните, фактическая командная строка MENU должна располагаться только на одной строке.

Первая из этих командных строк, вызываемая символом "1", инициализирует новый файл основного каталога диска MAST.CAT. Старый файл удаляется, и вызывается командный файл ZEX для выполнения фактической инициализации. Текст между кавычками не является частью командной строки. Кавычки - важная функция MENU, которая позволяет пользователю обеспечить ввод командной строки. В этом случае пользователь вводит список имен файлов, разделенных запятыми, которые будут исключены из файла MAST.CAT. MENU отправляет эту информацию в ZCPR3, который поочередно передает ее в ZEX, который в данном случае работает изнутри WordStar, чтобы выполнить фактическую работу.

Вторая командная строка, вызываемая "2", также использует заключенную в кавычки функцию ввода данных пользователем MENU, на сей раз чтобы обеспечить ввод для встроенной команды ZCPR3 SAVE. SAVE создает пустой файл с корректным именем, который служит меткой тома диска.

Следующая командная строка, вызываемая с помощью "3" фактически вызывает дисковую утилиту каталогизации MCAT, за которой находится команда ECHO, напоминающая пользователю, как прервать, заключительную команду в строке XCAT.

Нажатие "4" вызовет утилиту поиска строки, снова используя заключенную в кавычки функцию ввода данных пользователем для сканирования файла MAST.LST для любой вставленной строки ASCII.

Следующая команда просто выводит MAST.LST на экран с помощью резидентной команды TYPE. Пользователь ни чего не вводит.

Если пользователь нажмет "6", то результатом будет простой каталог диска B, который находится в этом дисководе и будет каталогизироваться (в предположении, что система с двумя дисководами).

Шестая командная строка, активизируемая по нажатию "7", вызывает другой файл ZEX, который запускает WordStar, на сей раз с целью найти-и-удалить для исключить все ссылки на номер тома диск, предоставленный пользователем с помощью заключенной в кавычки функции ввода. Затем вызывается XCAT, чтобы обновить MAST.LST после ECHO, напоминающего пользователю, как прервать этот процесс.

Последняя команда, к которой получают доступ, нажимая "9", использует утилиту поиска строки для вывода на экран содержания всего дискового тома.

Более полное обсуждение MENU и составления MENU.MNU находятся в ZCPR3: Руководство. Как и другие основные программы ZCPR3, эта вознаграждает тех, кто учится и экспериментирует.

Ниже представлен список команд MENU кроме запрограммированных в MENU.MNU:

|  |  |
| --- | --- |
| **Команда** | **Функция** |
| <CR> | Обновление изображения меню (клавиша RETURN) |
| ^C | Выход в ZCPR3 (Ctrl-C) |
| \* | Переход к первому меню |
| < или , | Переход к предыдущему меню |
| > или . | Переход к следующему меню |
| $ | Переход к системному меню (требуется пароль) |
| другая | Меню с параметрами или недопустимая команда. Буквы автоматически преобразуются в верхний регистр, таким образом, a=A |

### VMENU

VMENU работает во многом как MENU, за исключением того, что она рассчитана на работу с экраном, поэтому, требует TermCap (часть SYS.ENV в некоторых Z-System), и намного более мощная. VMENU читает файл \*.VMN и получает из него визуальное представление и команды от него.

При вызове VMENU загружает имена файлов в текущем каталоге, загружает \*.VMN файл и выводит на экран до шестнадцати файлов в текущем каталоге плюс первое меню из файла меню.

VMENU вызывается следующим образом:

**VMENU** <-- запустить MENU.VMN для всех файлов в директории

или

**VMENU afn** <-- запустить MENU.VMN для файлов выбранныз afn

или

**VMENU afn ufn** <-- запустить меню (ufn) для выбранных файлов

Файлы VMENU MENU.VMN очень напоминают файлы MENU MENU.MNU с небольшими изменениями. Следующий MENU.VMN является главным меню параметров для обработки текстов журнальных статей.

**Пример 1:**

Начальное меню для обработки нескольких секции статей журнала.

#X

^AWRITNG, EDITING AND TEXT TRANSMISSION SYSTEM MENU^B

^A-------------------------------------------------^B

F - GO TO FEATURES SECTION P - GO TO PRODUCTS SECTION

H - GO TO HIGHLIGHT SECTION E - GO TO LEADING EDGE SECTION

O - GO TO OUTLOOK SECTION M - GO TO MISCELLANY

W - WORKING DISK DIRECTORY D - SYSTEM DISK DIRECTORY

S - SEE LIST OF SECTIONS R - RUN THE POINTED `COM' FILE

Z - RUN ANY ZCPR3 COMMAND LINE

#

F SHCTRL P;B0:;VMENU

P SHCTRL P;B1:;VMENU

H SHCTRL P;B2:;VMENU

E SHCTRL P;B3:;VMENU

O SHCTRL P;B4:;VMENU

M SHCTRL P;B14:;VMENU

S!CLR;PWD

D!CLR;XDIR A:\*.\* U

W!CLR;XDIR B:\*.\* U

R$PN "ANY PARAMETERS? OTHERWISE, <cr> "

Z"YOUR COMMAND LINE PLEASE: "

##

Из нашего предыдущего изучения MENU.MNU следует, что этот файл предназначен для общего назначения и служебных целей. Он не требует много пояснений, но несколько моментов заслуживают отдельного упоминания. В файле в области отображения экрана ^A и ^B включают и выключают функцию выделения на экране, также используемую в MENU. Обратите внимание на использование в области командной строки восклицательного знака (!) в качестве второго символа в нескольких командных строках. Он заставляет VMENU (или MENU) делать паузу прежде, чем восстановить управление системой, удобную для просмотра каталогов, и т.д. Единственной функцией VMENU является параметр $PN, который является именем файла, указывающего только на файл в файле отображения экрана VMENU. Большинство команд здесь просто освобождает VMENU, "выталкивая" стек оболочки ZCPR3 с помощью утилиты SHCTRL и выполняет регистрацию в назначенный DU. Затем VMENU вызывается повторно, используя найденный в новой рабочей области файл MENU.VMN. Рабочая область MENU.VMN может выглядеть следующим образом:

#X

^AWRITING, EDITING AND TEXT TRANSMISSION MENU^B

D-SHOW FILES IN THIS SECTION |S-SET WORKING FILE TO POINTER

E-WRITE ON/EDIT WORKING FILE |T-TRANSMIT WORKING FILE

C-CHECK SPELLING OF WORKING FILE |M-EXAMINE/MAINTAIN FILES

?-TIME, DAY AND DATE DISPLAY |W-CONVERT WORKING FILE TO WS

A-CONVERT WORKING FILE TO ASCII |K-KILL WORKING FILE & BACKUP

R-REMOVE ALL BACKUP FILES |G-GET BACK ERASED WORKING FILE

N-NAME OF WORKING FILE |X-ERASE SPELLING CHECK WORKFILE

L-LISTING OF SECTION NAMES |Q-QUIT TO ANOTHER SECTION

B-BACK TO SYSTEM MENU

#

D!CLS;;DIR

S SETFILE 1 $PF

E WS $F1

T erab0:send.txt;cls;cpb0:send.txt=$F1;mex114ez read A0:xmit

C TW $F1

M VFILER

?!CLR;TIME

A CLS;RUN FILTX $F1

O CLS;RUN RESTORE

W ECHO ERASING OLD BACKUP FILE;ERA $N1.BAK;ECHO RENAMING WORKING <<

FILE TO N1.BAK;REN $N1.BAK=$F1;ECHO CREATING WORDSTAR FILE; <<

WSDOCON $N1.BAK $F1;ECHO CONVERSION COMPLETE!

R ERA \*.BAK

K ERA $N1.\* I

G UNERASE $F1

N!CLR;ECHO;ECHO;ECHO;ECHO;ECHO;ECHO;ECHO;ECHO;DIR $F1

L!CLR;PWD

X ERA ERRWORDS.TXT

B SHCTRL P;SYSTEM:;VMENU

Q"SECTION NAME PLEASE: ":

z!"OK - do it!> "

##

При рассмотрении этого MENU.VMN полезно отстраниться и понять сложность некоторых командных строк, которые VMENU вызывает для выполнения нажатием одной клавиши. Например, строка вызываемая нажатием "W", полностью изменяет способ работы программы, утилиты преобразования текста ASCII в WordStar, делающей обработку этого файла, отправляя полезные отчеты о выполнении на экран. Команда "T" связывает среду VMENU с коммуникационной программой управления файлами. Использование утилиты SETFILE в сочетании с возможностью указателя VMENU является еще одной интересной особенностью. Некоторые общие справочные сведения о VMENU приведены ниже.

Полная командная строка меню выглядит следующим образом:

Command (<CR>=Menu,^C=Z3,\*=1st Menu,<=Prev Menu,>=Last Menu) -

Команды VMENU:

|  |  |
| --- | --- |
| **Команда** | **Функция** |
| ^R | Обновление изображения меню (клавиша RETURN) |
| ^C | Выход в ZCPR3 (Ctrl-C) |
| \* | Переход к первому меню |
| < или , | Переход к предыдущему меню |
| > или . | Переход к следующему меню |
| другая | Меню с параметрами или недопустимая команда. Буквы автоматически преобразуются в верхний регистр, таким образом, a=A |

Внутренние команды файла \*.VMN:

|  |  |
| --- | --- |
| **Команда** | **Функция** |
| :nn | Перейти к меню nn, где первым меню является Меню 1 |
| ! | Подождать выполнения командной строки прежде, чем обработать меню |
| "Prompt" | Запросить у пользователя ввод данных и принять его |

Переменными VMENU являются:

|  |  |
| --- | --- |
| **Переменная** | **Заменяется на** |
| $D | Текущий диск |
| $U | Текущий пользователь |
| $Fn | FILENAME.TYP системного файла n |
| $Nn | Имя системного файла n |
| $Tn | Расширение системного файла n |
| $PF | FILENAME.TYP указанного файла |
| $PN | Имя указанного файла |
| $PT | Расширение указанного файла |
| $$ | $ |

*Примечание*: Системные файлы могут быть определены командой SETFILE.

Встроенные символы выделения:

|  |  |
| --- | --- |
| **^A** | Включить выделение |
| **^B** | Выключить выделение |

*Примечание*: Рекомендуется, выключение выделения располагать в той же строке.

Следующие символы ASCII не могут быть использованы в качестве команд, так как они используются в других местах:

<пробел> # % , . < > \* <DEL>

<Любой символ меньше прбела>

VMENU активно взаимодействует с системными файлами ZCPR3, которые определены как часть дескриптора среды ZCPR3. Существует четыре системных файла, и три из них используются VMENU для различных целей:

|  |  |
| --- | --- |
| **Файл** | **Назначение** |
| 2 | Имя текущего файла |
| 3 | Имя файла меню |
| 4 | Имя (содержащее подстановочные знаки), используемое для выбора файлов VMENU для экрана с файлами. |

## VFILER

VFILER, оболочка управления файлами ZCPR3, является современной, рассчитанной на работу с экраном утилитой манипулирования файлами, имеющей большое количество возможностей для программирования пользователем. Несмотря на то, что она не полностью настраиваемая как MENU или VMENU, обилие полезных встроенных функции и программируемость сделала ее одним из самых популярных инструментов Z-System. Если MENU и VMENU могут считаться "чистыми листами", на которых пользователь пишет для удовлетворения специальных действий, VFILER - полностью реализованная рабочая среда, в которую пользователь может добавить много своих функций.

VFILER содержит два стандартных встроенных экрана, один с полноэкранным отображением файлов в текущем каталоге с функцией указателя, подобной VMENU, другой полное меню стандартных функций VFILER, который имеет следующий вид:

-- Tagging Commands -- --------- File Operations -----------

T - Tag File C - Copy File D - Delete File

U - Untag File F - File Size R - Rename File

W - Wildcard T/U Q - File usQ

G - Group Copy/Delete/FSize/usQ/Tag/Untag

-- File Print & View -- --- User Functions ---

-- Cursor -- P - Print V - View 0-9 - Execute # - Help

^E

^ -- Movement Commands -- ---- Miscellaneous ----

^S <-+-> ^D <SP> - File Forward A - Toggle Alpha Sort

v <BS> - File Backward H - Help File

^X + - Screen Forward L,N - New DIR

- - Screen Backward S - Disk Status

-- Screen -- J - Jump to a File Z - ZCPR3 Command

^A Left E - Refresh Screen ^C,X - Exit

^F Right

Ввод вопросительного знака (?) или косой черты (/) используется для переключения между двумя экранами. Независимо от того, какой экран отображается, в настоящее время указанный файл отображается на экране в верхнем правом углу. Многие функции VFILER будут знакомы тем, кто использовал программы CP/M как SWEEP Роберта Фишера {Robert Fisher}, DISK7 Фрэнка Гауди {Frank Gaude} и NSWP Дэйва Рэнда {Dave Rand}. На самом деле DISK7 был основой, на которой Ричард Конн {Richard Conn} разработал VFILER, несмотря на то, что она сильно изменилась от своего оригинала в CP/M.

Использование встроенных команд VFILER не вызывает затруднений, как только вы поймете, что большинство его операций направлены на указанный в настоящее время файл или на группу "помеченных" файлов. Программирование 10 (пронумерованных от 0 до 9) функций пользователя является необязательным, но здесь Вы можете добавить некоторые свои собственные задачи и настройки к общей функциональности VFILER. Эти функции программируются, созданием текстового файла с названием VFILER.CMD, который похож по своей концепции на MENU.VMN и MENU.MNU, но несколько отличается в синтаксисе. Еще одним заметным отличием является то, что VFILER ищет VFILER.CMD по пути поиска команд, если он не будет найден в текущем DU.

**Пример 1:**

TITLE: General CMD Macro File

0 xdir %d%u:\*.\* ogoh'Options (p=to printer, d=disk, u=all users):'

1 get 100 ws.com;poke 392 ff;go %$

2 spell %$ $slic;review;markfix %$ $m#;if ex errwords.txt; <<

ws %$;era errwords.txt i;fi

3 vdo %d:%f

4 %d%u:;nulu -o %f -l

5 CAT:;catscan 'Master Catalog Search (e.g. \*.doc): ' \*.\*;%d%u:

6 tmaker get income10.ei e;ac BACKUP:=%d:income10.ei /a

7 mex

8 DBASE:;dbase mail;%d%u:

9 protect %$ 'Attribute (CR=r/w, r=r/o, s=system): '

#

>>> SIMPLE WRITING, DATABASE and COMMUNICATIONS Menu <<<

-----------------------------------------------------------------

0 - Directory of Current Disk (XDIR) - 0

1 - EDIT Current File (non-document mode) (WordStar) - 1

2 - Spelling Check and Correction (SPELL+) - 2

3 - Edit Current File using VDO (VDO25) - 3

4 - Enter Current Library (.LBR) (NULU151) - 4

5 - Scan Master CATALOG from Console (CATSCAN) - 5

6 - Financial Tally (T/Maker) - 6

7 - Telecommunicate via modem (MEX) - 7

8 - Mailing List Management System (dBASE II & MAIL-ACG) - 8

9 - Set Current File Attributes (R/O,R/W,System) (PROTECT) - 9

-----------------------------------------------------------------

Обратите внимание, что в файлах VFILER.CMD, командные строки на первом месте, в отличии от соглашения MENU.MNU и MENU.VMN, и продолжаются до первого и единственного знака решетки. Весь текст после знака решетки является экраном справки, показанным, когда пользователь VFILER вводит "#" в любом из двух стандартных экранов VFILER. Этот раздел не может быть больше, чем длина предусмотренная экраном. Вы должны были также заметить, что все командные строки начинаются со своей соответствующей вызову цифры. Распознаются только цифры от 0 до 9, и в отличие от MENU и VMENU, за ними должен следовать пробел, после которого начинается фактическая командная строка. Строки, которые (как TITLE в этом файле) начинаются с чего-либо кроме 10 одноразрядных цифр, рассматривают как то, что программисты называют "комментарии исходного кода". VFILER игнорирует их, но они могут быть удобными пометками для себя, если и когда VFILER.CMD повторно редактируется.

Первая командная строка, инициируемая с помощью "0", вызывает программу ZCPR3 XDIR с некоторым количеством параметров. Любой дополнительный параметр(ы) XDIR предоставляются пользователем через заключенную в кавычки функцию ввода. Обратите внимание, что VFILER требует апострофы (') вместо кавычек (").

Вторая функция пользователя, вызываемая "1", использует метод поместить и перейти, чтобы перевести WordStar (версия 3.3) в рабочее состояние в режиме недокумента для редактирования указанного в данный момент файла. Параметр %$ будет расширен VFILER до хвоста командной строки DU:FILENAME.TYP. WordStar (и большинство других не ZCPR программ) будет игнорировать номер пользователя.

Нажатие "2" переводит в рабочее состояние следующую функцию, которая использует программы из пакета WORD Plus Oasis Systems для проверки орфографию в указанном настоящий момент файле. Отметьте использование команды управления выполнением IF EX для проверки присутствия файла ошибок WORD Plus ERRWORDS.TXT, и вызвать WordStar, если возникли ошибки. В этом случае ERROWRDS.TXT удаляется с подтверждением пользователя при выходе из WordStar.

Четвертая командная строка использует альтернативную программу редактирования, которая быстрее, но менее универсальна, чем WordStar, программа общественного достояния VDO. Так как VDO испытывает затруднения при обработке формы DU:, указанный файл определяется с помощью комбинации параметров %d:%f, которую VFILER разворачивает до D:FILENAME.TYP.

Следующая команда инициируемая "4", вызывает утилиту управления файлами библиотек NULU Мартина Мюррея {Martin Murray}, которая пытается открыть указанный файл в формате Gary Novosielski \*.LBR и отобразить его содержание. Внутренняя проверка ошибок NULU зависится от обработки любых входных ошибок.

Нажатие "5" временно регистрирует в именованный каталог CAT: и вызывает программу Эшелона {Echelon} DISCAT CATSCAN с заключенным в кавычки вводом данных пользователя, определяющим поиск через неоднозначную спецификацию файла. Параметры VFILER %d и %u, сопровождаемые двоеточием, возвращают в каталог, из которого функция была запрошена.

Следующая строка, инициированная "6", переводит в рабочее состояние файл данных INCOME10.EI с помощью многофункциональной программы T/Maker и использует утилиту AC ZRDOS, чтобы поместить копию этого файла в именованный каталог BACKUP: если были внесены какие-либо изменения.

Следующая командная строка вызывает коммуникационную программу MEX. Вы можете, конечно, заменить на корректную командную строку вызова MDM740, COMM7, Crosstalk или TERM3 в зависимости от своих предпочтений.

Девятая функция пользователя инициированная "8", регистрирует в именованный каталог DBASE: и переводит в рабочее состояние командный файл dBase II MAIL.CMD, возвращаясь к исходному каталогу по окончании.

Заключительная командная строка, инициированная "9", расширяет возможности манипулирования файлами VFILER, вызывая утилиту установки атрибутов файла ZCPR3 PROTECT с заключенным в кавычки вводом данных пользователя.

# Взаимодействие с терминалами и принтерами

## Введение

Дескриптор среды ZCPR3 включает TCAP (Запись возможностей терминала) и буферы для определения периферийных устройств. TCAP, определения периферийных буферы и утилиты, которые читают данные из этих буферов, предлагают гибкую, мощную систему управления терминалам и принтерам.

Краеугольный камень возможностей Z System для взаимодействия с терминалами TCAP, является областью данных в верхней памяти, которая описывает атрибуты терминалов или консолей CRT (экраны). Под "атрибутами" мы имеем в виду управляющие коды или последовательности, необходимые для выполнения терминальных функций, таких как очистка экрана, перемещение курсора и стирание до конца строки. Ориентированные на экран инструменты для Z System знают о TCAP и выполняют различные экранно-ориентированные функции, используя свои знания записи TCAP для вашего терминала. SHOW и VFILER - две известных утилиты Z System, которые используют данные в TCAP для визуального выделения и детализации. Существует множество подобных инструментов для Z System и вы найдете их, по мере изучения системы.

В буфере определения периферии есть место для определения характеристик двух CRT и 4 принтеров. Некоторые утилиты Z System, такие как PRINT и PAGE, считывают информацию из этого буфера. PRINT считывает данные, описывающие один из четырех принтеров (вы выбираете принтер с помощью утилиты CPSEL) - информацию о возможностях принтера для прогона страницы, строк текста на одной странице и ширины принтера. Утилита PAGE, необычная версия резидентной команды TYPE, считывает данные, описывающие один из двух терминалов для ширины и высоты экрана. Обратите внимание, что, когда мы говорим о настройках принтера и терминала в буфере определения периферии, мы говорим о логических принтерах и терминалах. Слово "логический", при обращении к устройству, такому как принтер или терминал, не относится к фактическим физическим характеристикам устройства, а к его атрибутам как указано в буфере определения периферии. У вашего принтера может быть физическая ширина 132 столбца, используя шаг 10 точек, но вы можете определить его как имеющий логическую ширину 80. С точки зрения ZCPR3 и его утилит, таких как PRINT, ваш принтер будет теперь функционировать, как будто у него только 80 столбцов.

## TCAP Описания характеристик терминала

Ваша запись характеристик терминала TCAP, занимает 128 байт памяти и содержит всю необходимую информацию для видео-ориентированных инструментов Z System для корректной работы с терминалом.

### Получение информации TCAP в дескрипторе среды

Существует три способа получения информации о вашем терминале в 128-байтном буфере в верхней части дескриптора среды.

#### SYS.ENV

Во-первых, вы можете поместить определение своего терминала в SYSENV.LIB (конфигурационный файл для SYSENV.ASM) и создать новый SYS.ENV (SYSENV.COM переименовывается в SYS.ENV после того, как он ассемблирован и загружен). Как вы помните, файл .ENV загружается с помощью LDR.COM. Вот запись SYSENV.LIB для одного из наших компьютеров Kaypro 484:

B0:WORK>PAGE ASMFIL.DOC PL

PAGE, Version 2.0

PAGE File: B 0: ASMFIL .DOC

1: ;

2: ; Terminal Capabilities Data

3: ;

4: envorg2:

5: DB 'Kaypro 484 ' ;Name of Terminal

6: DB 'K'-'@' ;Cursor UP

7: DB 'J'-'@' ;Cursor DOWN

8: DB 'L'-'@' ;Cursor RIGHT

9: DB 'H'-'@' ;Cursor LEFT

10: DB 00 ;CL Delay

11: DB 00 ;CM Delay

12: DB 00 ;CE Delay

13: DB 'Z'-'@',0 ;CL String

14: DB 1bh,'=%+ %+ ',0 ;CM String

15: DB 18h,0 ;CE String

16: DB 1Bh,'B1',0 ;SO String

17: DB 1Bh,'C1',0 ;SE String

18: DB 0 ;TI String

19: DB 0 ;TE String

20:

21: ds 80H-($-envorg2) ; make exactly 80H bytes long

22:

23: ;

24: ; End of Environment Descriptor

25: ;

26: endm

Всего несколько пояснений. CL (строки 10 и 13) указывает последовательность очистки экрана, CM (строки 11 и 14) последовательность движения курсора, CE (строки 12 и 15) последовательность очистки до конца строки. SO (строка 16) означает начать выделение, SE (строка 17) конец выделения, TI (строка 18) инициализация терминала и TE (строка 19) восстановление терминал в нормальный режим. "'Z'-'@'" означает Ctrl Z. (Символ очистки экрана для Kaypro 484 - Ctrl Z). Символ процента (%) говорит интерпретатору движения курсора (подпрограмма, находящаяся в ориентированных на экран утилитах Z-System), что следующий символ является директивой интерпретатора. Последовательность "%+ " (строка 14), например, говорит интерпретатору последовательности движения курсора добавлять 32 или 20H к текущему значению строки или столбца и выводить это значение в двоичном виде. (32 десятичное - в ASCII пробел.) Формула для определения движения курсора на Kaypro 484 является "ESC = ROW + 32, COLUMN + 32". Формула выражена в терминах, которые интерпретатор последовательности движения курсора может понять на строке 14. Эти понятия могут казаться сложными, но сложность побеждена игрой. Вы можете играть немного на этом этапе с помощью этих методов для получения данных в вашем TCAP. Вот задание. Просмотрев вышеприведенный код, можете ли Вы выяснить, как изменить выдающийся режим от видео полуинтенсивности, чтобы инвертировать видео. Подсказку? Последовательностью для ввода обратного видео в Kaypro 484 является "ESC B 0", и последовательность для выхода из обратного видео является "ESC C 0". Вы можете сделать то же самое со своим собственным терминалом?

#### TCMAKE

Второй метод генерации терминальных данных и загрузки его в TCAP полагается на утилиту TCMAKE. TCMAKE позволяет вам определить свои терминальные атрибуты и сохранить определения в системном сегменте, который мы еще не обсудили - файле .Z3T. Если вы используете терминал Wyse, вы могли бы использовать TCMAKE для создания файла WYSE.Z3T, который может быть загружен в слот TCAP с помощью LDR.COM. Команда была бы "LDR WYSE.Z3T". Утилита TCMAKE – управляемая с помощью меню. Мы не будем вдаваться в подробности его использования. Эта информация доступна в ZCPR3: Руководство, стр. 309-324.

#### TCSELECT

Третий и заключительный метод, чтобы получить данные для вашего TCAP, используя утилиту TCSELECT, является безусловно самым простым и тем, который мы рекомендуем в начале. TCSELECT выбирает терминальное определение из стандартного файла базы данных терминалов (более чем 40) и или создает .Z3T файл для загрузки LDR.COM или непосредственно загружает данные в резидентный TCAP. Стандартный файл базы данных обычно называют Z3TCAP.TCP.

Утилита TCSELECT выбирает запись для терминала из файла Z3TCAP.TCP и либо загружает эту запись в TCAP или записывает ее в файл с расширением .Z3T для загрузки в память с помощью LDR.COM. Это - то, что показывает вам параметр справки "//" для TCSELECT (запомните, что почти все утилиты Z-System используют параметр справки "//"):

B0:WORK>**tcselect //**

TCSELECT, Version 1.1

TCSELECT - Select Entry from Z3TCAP.TCP

Syntax:<

TCSELECT outfile -or- TCSELECT outfile.typ

where "outfile" is the file to be generated by

the execution of TCSELECT. If no file type is

given, a file type of Z3T is the default.

Syntax:

TCSELECT

where this alternate form may be used to store

the Z3TCAP entry for the selected terminal directly

into the Z3 Environment Descriptor.

Теперь давайте выполним TCSELECT и выберем терминал, используя опцию записать терминал нашего выбора в файл .Z3T:

B0:WORK>**tcselect myterm.z3t**

TCSELECT, Version 1.1

\*\* Terminal Menu 1 for Z3TCAP Version 1.5 \*\*

A. AA Ambassador K. Apple //e PCPI

B. ADDS Consul 980 L. Apple ///

C. ADDS Regent 20 M. Bantam 550

D. ADDS Viewpoint N. CDC 456

E. ADM 2 O. Concept 100

F. ADM 31 P. Concept 108

G. ADM 3A Q. CT82

H. ADM 42 R. DEC VT52

I. Apple //e ALS S. DEC VT100

J. Apple //e MS T. Dialogue 80

Enter Selection, + for Next, or ^C to Exit - +

\*\* Terminal Menu 2 for Z3TCAP Version 1.5 \*\*

A. Direct 800/A K. HP 2621

B. Epson GENEVA L. IBM 3101

C. Epson QX-10 M. Kaypro II

D. General Trm 100A N. Kaypro 10

E. Hazeltine 1420 O. Micro Bee

F. Hazeltine 1500 P. Microterm ACT IV

G. Hazeltine 1510 Q. Microterm ACT V

H. Hazeltine 1520 R. NorthStar Advant

I. H19 (ANSI Mode) S. Osborne I

J. H19 (Heath Mode) T. P Elmer 1100

<

Enter Selection, - for Last, + for Next, or ^C to Exit - N

Selected Terminal is: KP-10 Kaypro -- Confirm (Y/N)? Y

File MYTERM .Z3T Created

Теперь, когда мы создали MYTERM.Z3T, давайте загрузим его в память, таким образом, мы можем протестировать его.

B0:WORK>**ldr myterm.z3t**

ZCPR3 LDR, Version 1.4

Loading MYTERM.Z3T

B0:WORK>;и мы сделали.

## Определение периферийного буфера

Буфер определений периферии, расположенный в дескрипторе среды, содержит информацию об атрибутах двух CRT и четырех принтерах. Для выбора одного из двух CRT и одного из этих четырех принтеров используется утилита CPSEL. Некоторые утилиты ZCPR3, такие как PAGE и PRINT читают определения выбранного терминала и принтера и регулируют их вывод исходя из этого.

Мы предоставили исходный код для этой части SYS.ENV, содержащий определения терминала и принтера, сохраненные в периферийном буфере определения периферии после того, как SYS.ENV (или любой другой .ENV файл) будет загружен в память LDR.COM.

db 0 ; CRT selection (0=CRT 0, 1=CRT 1)

db 1 ; Printer selection (n=Printer n)

db 80 ; width of CRT 0

db 24 ; number of lines on CRT 0

db 22 ; number of lines of text on CRT 0

db 132 ; width of CRT 1

db 24 ; number of lines on CRT 1

db 22 ; number of lines of text on CRT 1

db 80 ; width of Printer 0

db 66 ; number of lines on Printer 0

db 58 ; number of lines of text on Printer 0

db 1 ; form feed flag (0=can't formfeed, 1=can)

db 102 ; width of Printer 1

db 66 ; number of lines on Printer 1

db 58 ; number of lines of text on Printer 1

db 1 ; form feed flag (0=can't formfeed, 1=can)

db 80 ; width of Printer 2

db 66 ; number of lines on Printer 2

db 58 ; number of lines of text on Printer 2

db 0 ; form feed flag (0=can't formfeed, 1=can)

db 132 ; width of Printer 3

db 88 ; number of lines on Printer 3

db 77 ; number of lines of text on Printer 3

db 1 ; form feed flag (0=can't formfeed, 1=can)

Ассемблерный источник для буфера определения периферии использует директиву ассемблера "DB", чтобы определить байты. Первый байт буфера, который является 48-м байтом дескриптора среды, содержит выбор вашего CRT, и следующий байт содержит выбор вашего принтера. Остающиеся байты определяют характеристики двух возможных логических CRT и четыре возможных логических принтеров. Обратите внимание, что последняя запись определяет принтер 3 как имеющий 88 строк с 77 строками фактического текста. Один из нас использует принтер 3, когда он распечатывает в режиме сжатой печати 8 линий на дюйм. Определение принтера 3 как имеющего 88 строк за страницу просто говорит утилите PRINT не продвигать принтер к новой странице, пока не введены 88 строк текста. Еще раз, напоминаем об этом, чтобы подчеркнуть разницу между логическими и физическими устройствами.

### Выбор терминалов и принтеров

Существует несколько способов выбора логического терминала и логического принтера для предоставления необходимой информации соответствующим утилитам ZCPR3.

#### CPSEL

Утилита CPSEL выбирает одну из двух консолей CRT и один из четырех принтеров, в соответствии с определениями в буфере определения периферии в дескриптора среды. Она также предоставляет информацию о выбранной консоли CRT и выбранном принтере. Наконец, CPSEL показывает вам определение остальных, невыбранных CRТ и принтеров.

Это справка "двойной наклонной черты", предоставляемая CPSEL:

B0:WORK>**cpsel //**

CPSEL Version 1.0

CPSEL cmd1 cmd2 cmd3,...

Commands:

Cc, c=0 or 1 -- Select CRT 0 or 1

Pp, p=0,1,2,3 -- Select Printer 0, 1, 2, or 3

Dd, d=A (All), C (CRT), P (Printer)

-- Display Selection Values

И это дисплей, когда мы говорим CPSEL показать нам все выбранные значения:

B0:WORK>**cpsel da**

CPSEL Version 1.0

Current CRT Selection: 0

\* CRT 0: Width = 80 Actual/Text Lines = 24/ 22

CRT 1: Width = 132 Actual/Text Lines = 24/ 22

Current Printer Selection: 1

Prt 0: Width = 80 Actual/Text Lines = 66/ 58 Form Feed? Yes

\* Prt 1: Width = 102 Actual/Text Lines = 66/ 58 Form Feed? Yes

Prt 2: Width = 80 Actual/Text Lines = 66/ 58 Form Feed? No

Prt 3: Width = 132 Actual/Text Lines = 88/ 77 Form Feed? Yes

Примечание "\*" указывает выбранное логическое устройство.

Для тех, кто хочет пробовать свои силы в Z-System, как насчет того, чтобы использовать резидентную команду POKE? Если ваш дескриптор среды начинается в F300H, вы можете попробовать "POKE F32F 01" с целью изменения выбора вашего CRT с CRT 0 на CRT 1. Не торопитесь, использовать команду PEEK для изучения буфера определений периферии. Знание расположение адресов различных байтов определения позволит вам создать сценарии псевдонима, для ввода различных определений в буфер и выбора различных консолей CRT и принтеров, не используя CPSEL. Иногда длинный обходной путь оказывается кратчайшим путем.

### Примеры утилит, которые читают буфер определения периферии

PAGE и PRINT считывают данные, находящиеся в буфере определения периферии дескриптора среды. Эта способность обеспечивает мощность и гибкость при управлении терминалами и принтерами.

#### PAGE

Утилита PAGE печатает текстовые файлы на вашем экране. Она является полнофункциональной, изящная версией TYPE. В частности, с целью нашего исследования дескриптора среды, PAGE считывает информации о системной консоли CRT и выполняет перенос слов если длина строки превышает ширину экрана, определенной для выбранной консоли CRT в буфере определение периферии. Опять же, возможность "справки наклонной черты" предоставляет информацию о функциях PAGE.

B0:WORK>page //

PAGE, Version 2.0

Syntax:

PAGE file1,file2,...,filen o...

Options:

0-9 Select Delay Constant

I Inspect and Select Files First

L Toggle Numbering of Each Line

P Toggle Paging

Snnnn Skip to Specified Page before Printing

Examples:

PAGE MYFILE.TXT,\*.MAC LI

-- Number Lines, Inspect Files

PRINT MYFILE.\* S25

-- Skip to Page 25

Commands during printout:

^C - abort PAGE ^X - skip to next file

^S - suspend output P - toggle paging

0-9 - change speed

Обратите внимание, что PAGE разрешает вам просматривать файлы последовательно и пропускать любой файл в списке.

#### PRINT

Утилита PRINT является эквивалентом PAGE для принтера. Она также читает определения логических устройств в дескрипторе среды и предлагает полностраничную печать с выбором характеристик принтера. PRINT контролирует заголовки страниц, нумерацию страниц и последовательную печать файлов. Кроме того всегда полезная двойная наклонная черта предоставляет вам достаточную информацию для начала работы.

B0:WORK>print //

PRINT III, Version 2.1

PRINT file1,file2,...,filen o...

Options:

E Exact Print (expand tabs, form feed, no line

or page numbers, no heading) default no

F Toggle File Name Display default yes

H@head@ Specify Page Heading (@ is any printing char)

I Inspect and Select Files First default no

L Toggle Numbering of Each Line default no

M Toggle Multiple Runs (MR=No TOF Msg) default yes

N Toggle Numbering of Each Page default yes

Onn Offset Printout by nn Characters from Left default 0

Snnnn Skip to Specified Page before Printing default no

T Toggle Time Display in header default yes

Examples:

PRINT MYFILE.TXT,\*.MAC LH'SAMPLE'

-- Number Lines, SAMPLE is Heading

PRINT MYFILE.\* S25E

-- Skip to Page 25, Exact Print

At any time, ^C aborts PRINT and ^X skips to next file

Как только вы начнете работать с логическими определениями принтеров вы обнаружите, что утилита PRINT является одним из наиболее часто используемых вами инструментов ZCPR3.

## Дальнейшие исследования

После того, как вы освоили операции с дескриптором среды для настройки взаимодействия между компьютером и его терминалами и принтерами, вы можете изучить интерпретатор языка управления DPROG, используемый для программирования этих физических устройств. DPROG позволяет создавать программы или сценарии с понятными английскими словами для управления вашими периферийными устройствами. Много превосходных программ для DPROG доступны на Z-Node во всем мире. Эти программы были созданы для Epson, Okidata и других популярных принтеров, а также для многих широко используемых терминалов. Один из лучших способов изучить DPROG или программирование устройств, состоит в том, чтобы посмотреть на то, что сделали другие.

# Инструменты Z-System (Z утилиты)

В этом разделе кратко описываются основные утилиты ZCPR3. Для более подробных объяснений проверьте ZCPR3: Руководство или соответствующий файл справки ZCPR3. Большинство утилит ZCPR3 выводят на экран краткое сообщение справки с синтаксисом, если они вызваны с "//" в качестве хвоста команды.

## Утилиты ZCPR3

**ALIAS**

Утилита ALIAS создает файлы псевдонима и выводит на экран их содержание. Псевдоним - сценарий, который может быть передан как ряд команд, во время выполнения файла COM, содержащий сценарий.

**CD**

Утилита CD обозначает "смена каталога". Она позволяет перемещаться из одного каталога в другой. Каждый раз при регистрации в новом каталоге, она ищет файл COM с названием ST.COM. Если она находит ST.COM, то выполняет его. ST.COM обычно - ALIAS, который устанавливает требуемые условия в области каталога. Таким образом ST.COM может загрузить новые наборы RCP, FCP и NDR. При регистрации в каталоге ASM, вы можете изменить свой RCP, например, предназначенный для вашей работы с ассемблером. Взаимодействие между CD и ST - еще один пример направленности Z-System на точную настройку среды пользователя.

**CLEANDIR**

Эта утилита "чистит" ваш каталог физического диска. CLEANDIR загружает каталог диска, сортирует его и записывает его на диск, заполняя неиспользованные записи каталога E5 - ускоряя вывод списка файлов в каталоге и увеличивая вероятность успешного восстановления случайно стертых файлов.

**COMMENT**

Помещает компьютер с Z-System в режим, подобный началу всех строк команд с точкой с запятой: никакие команды не выполняются, видны на экране, а также могут быть отправлены в устройство LST: с помощью команды переключателя.

**CMD**

Инструмент специального назначения, разработанный, чтобы улучшить утилиту настройки оболочки SHSET. Предоставляет прямой доступ к командной строке ZCPR3, включая программу SHCTRL, которая может завершить оболочку.

**CMDRUN**

Простая демонстрация программа, предназначенная для демонстрации функции ZCPR3 CMDRUN. Обычно вместо функциональных расширенных команд процессора, таких, как ZEX.COM переименованных в CMDRUN.COM.

**CPSEL**

CPSEL позволяет вам выбирать одну из двух консолей и один из четырех принтеров, как определено в дескрипторе среды. Это полезно, когда Вы поручили своему принтеру, например, печатать 8 линий на дюйм и хотите, чтобы страница была заполнена. Таким образом вы будете использовать CPSEL для выбора логический принтер, настроенного как имеющего больше, чем обычное число строк за страницу. Имейте в виду, что действие CPSEL может быть обнаружено и использовано только утилитами ZCPR3.

CRC

Эта утилита - расширенная версия ZCPR3 программы CRCK Кейта Петерсена {Keith Petersen}. Она выполняет контроль с помощью циклического избыточного кода для указанных файлов и и при необходимости создает файл на диске с их значениями. Эта утилита используется, чтобы гарантировать, что два файла являются фактически идентичными.

**DEV**

DEV управляет перенаправляемыми драйверами устройств ввода-вывода ZCPR3. она используется, чтобы отобразить и выбрать выходные устройства в Z-System, реализованные с пакетами ввода-вывода. Она принимает инструкции в командной строке и таким образом более полезна, чем DEVICE (см. ниже), при вызове из командного скрипта.

**DEVICE**

DEVICE привязывает физические устройства к логическим, в интерактивном режиме отображает имена доступных физических драйверов устройств, и, в отличие от DEV, работает только в интерактивном режиме. DEVICE требует, чтобы необходимый пакет ввода-вывода был реализован и загружен в память.

**DIFF**

DIFF сравнивает два файла и сообщает вам, если и насколько они разные. Такое сравнение полезно, когда вы пытаетесь определить одинаковы ли два файла или в чем они различаются.

**DIR**

Эта утилита отображает текущие или запрашиваемые файлы каталога, показывает размеры файла, емкость диска, используемое и остающееся дисковое пространство. DIR является самой быстрой из утилит каталогов ZCPR3, но испытывает недостаток во многих расширенных функциях XD и XDIR.

**DPROG**

DPROG - интерпретатор, используемый для программирования физических устройств, таких как терминалы и принтеры. DPROG использует сравнительно простые английские команды управления физическими устройствами.

**DU3**

DU3 - версия ZCPR3 дисковой утилиты DU, которая позволяет выполнять операции с дисками на уровне байтов. DU3 является гораздо более мощной, чем предыдущие варианты DU.

**ERASE**

ERASE - транзитный дубликат ERA и намного более мощный, предлагающий пользователю полный контроль над всеми операциями удаления, со многими параметрами.

**ERROR1**, **ERROR2**, **ERROR3** and **ERROR4**

Эти утилиты являются обработчиками ошибок ZCPR3. Каждый обработчик ошибок может быть установлен или удален динамично (обработчик ошибок с удаляется с помощью ERRORX), и у каждого есть немного отличающаяся разновидность. ERROR2 - Экранно-ориентированный. Обработчики ошибок позволяют вам делать корректные восстановления после команд с одиночными или многократными ошибками.

**HELP**

Обеспечивает полную онлайн-документацию системы ZCPR3. ZCPR3: Руководство, доступное от Echelon, Inc., является Библией системы ZCPR3, и система HELP содержит основное подмножество информации из руководства для интерактивного использования в консоли. HELPCK и HELPPR используются для проверки синтаксиса файлов HLP при создании и распечатки бумажных копий файлов HLP.

**IF**

Несколько более мощная и гибкая альтернатива, в виде файла COM, резидентной FCP команде IF.

**IFSTAT**

Простой инструмент диагностики, полезный в написании управляемых потоком командных сценариев. Отображение текущего уровня IF. Состояние IF должно быть переключено к TRUE для работы IFSTAT.

**LDR**

Загрузчик системных сегментов ZCPR3. Корректно располагает в памяти ZCPR3 дескрипторы среды (\*.ENV), пакеты управления выполнением команд (\*.FCP), пакеты резидентных команд (\*.RCP), пакеты ввода-вывода (\*.IOP) и сегменты TCAP (\*.Z3T).

**MCOPY**

Эта утилита копирует файлы от одного каталога в другой. В отличие от резидентной команды CP, MCOPY допускает проверку CRC и имеет другие мощные опции. В отличие от CP, она не позволяет копировать файлы в тот же DU или DIR с измененным именем.

**MENU**

MENU выполняет подсистему меню ZCPR3. Меню MNU создаются в текстовом редакторе с использованием синтаксических правил подсистемы MENU, описанных в Руководстве и файле справки MENU. MENU является оболочкой ZCPR3, программой, которая может при желании заменить стандартное приглашение ZCPR3 в качестве интерфейса между пользователем и компьютером.

**MKDIR**

MKDIR создает именованные каталоги. У каждого каталога может быть имя длиной до восьми символов. Дополнительно каждому каталогу доступна защита паролем.

**MU**

ZCPR3 экранно-ориентированная утилита редактирования памяти. Доступная и как COM файл (MU3), и как RCP (DEBUGRCP). Более сложная реализация RCP команд P (чтение из ячейки оперативной памяти) и POKE, с дополнительными функциями.

**PAGE**

PAGE распечатывает текстовые файлы на вашем экране. Вы можете просмотреть файлы последовательно и пропустить любой файл в списке. PAGE выполнит экранный переход на новую строку, если длина строки превысит ширину "логического" экрана.

**PATH**

PATH отображает и динамично изменяет путь поиска команд.

**PRINT**

PRINT предлагает полностраничную печать текстовых файлов на принтере. CPSEL выбирает характеристики принтера. PRINT контролирует заголовки страниц, нумерацию страниц и последовательную печать файлов.

**PROTECT**

PROTECT устанавливает атрибуты файла - только для чтения, системный, архив или чтение и запись. Альтернатива ZRDOS утилите SFA.

**PWD**

PWD распечатывает на CRT имена доступных именованных каталогов.

**QUIET**

QUIET уменьшает сумму информации, выведенной на экран утилитами ZCPR3, запрограммированными реагировать на системный "тихий байт".

**RECORD**

Специальный инструмент для управления работой IOREC Эшелона {Echelon} (регистратор ввода-вывода) или любой правильно разработанный пакет ввода-вывода (\*.IOP), реализующий маршрутизацию вывода консоли в файл на диске или периферийное устройство.

**REG**

Утилита, довольно полезная в сложных управляемых потоком командных сценариях. Загружает, арифметически манипулирует и читает серии из 10 (0-9) зарезервированных однобайтовых буферов, называемых регистрами ZCPR3.

**RENAME**

Более функциональный файл COM, эквивалентный ZCPR3 резидентной команде RCP и встроенной команде REN. Допускает работу со списками файлов и поддерживает неоднозначные ссылки в имени файла.

**SAK**

SAK означает "нажать любую клавишу" и используется в командных строках для приостановки выполнения на заданное время или разрешения при необходимости отмены следующих операций. SAK может, при необходимости, предупредить пользователя, что требуется ввод сигналом на консоли.

**SETFILE**

Эта утилита используется для объявления до четырех имен системных файлов для использования с псевдонимами и с программными оболочками, такими как VMENU. Утилита SHOW выводит на экран активные системные файлы.

**SHOW**

SHOW отображает состояние среды ZCPR3 - адреса расположения буферов, длину командной строки, системную скорость процессора, максимально разрешенные номер диска и пользователя и многое другое на одиннадцати информационных экранах. Команда "SHOW E" инсталлирует SHOW в качестве обработчика ошибок.

**SHCTRL**

SHCTRL - утилита манипулирования оболочкой ZCPR3. Она позволяет пользователю просматривать стек оболочки ZCPR3, очищать все оболочки или выходить из активной в настоящее время оболочки.

**SH**, **SHDEFINE**, **SHFILE**, **SHVAR**

SH является ZCPR3 оболочкой именованных переменных, остальные программы являются утилитами, довольно полезными в сочетании с SH. Пользовательские приложения для этих программ довольно малочисленные и редкие, но они действительно обеспечивают, очень впечатляющую демонстрацию возможностей оболочки ZCPR3 - многофункциональность и универсальность.

**SHSET**

SHSET позволяет вызвать любую исполняемую программу в качестве оболочки ZCPR3. Текстовые процессоры, коммуникационные программы и многие другие программы интерактивного характера являются хорошими кандидатами оболочек.

**SUB**

Очень расширенная ZCPR3 версия утилиты SUBMIT CP/M. Командные сценарии находящиеся на диске, выполняются значительно медленнее, чем псевдонимы или основанная на памяти программа ZEX и была в основном вытеснена ими в пользовательском сообществе.

**UNERASE**

Эта утилита перечисляет и восстанавливает случайно стертые файлы.

**TCCHECK**, **TCMAKE**, **TCSELECT**

Эти утилиты поддерживают ZCPR3 TCAP (возможности терминалов) и малоинтересны для пользователей автономных систем как Kaypro. Они незаменимы, если вы эксплуатируете вашу систему с внешним терминалом.

**VFILER**

VFILER - ориентированная на экран утилита манипулирования файлом, подобная DISK7 или NSWP, но намного более мощная в среде ZCPR3. Она позволяет вам указывать с помощью стрелки файл в вашем каталоге и выполнять с ним много операций. VFILER является оболочкой ZCPR3.

**VMENU**

Выполняет подсистему VMENU (видео-ориентированное MENU) ZCPR3. Используя созданные пользователем командные файлы \*.VMN, VMENU предлагает полное управление меню, перечисляет части каталога наверху экрана и имеет возможность доступа к файлу. VMENU является оболочкой ZCPR3.

**WHEEL**

Разрешает изменение состояния системы от безопасного до небезопасного и обратно. Если система установлена в безопасное состояние, могут быть необходимы пароли для получения доступа к определенным каталогам и защищенным файлам. Некоторые утилиты, такие как MKDIR и ALIAS, не могут использоваться, когда wheel байт сброшен.

**XD**

Утилита каталога XD, немного функциональней по сравнению с DIR, но немного медленнее. Хотя она не столь многофункциональна как XDIR, зато быстрее в работе.

**XDIR**

XDIR является самой мощной из утилит каталога ZCPR3, а также самой медленной.

**Z3INS**

Утилита установки программ ZCPR3. Z3INS должна использоваться, для настройки программ ZCPR3, полученных от Эшелона {Echelon} или из Z-Node прежде, чем выполнять их в вашей системе. Внедряет адрес дескриптора среды в целевой файл COM.

**Z3LOC**

Отображает все главные адреса Z-System. Быстрее, чем SHOW, но предоставляет менее подробную информацию.

**ZEX**

ZEX - очень мощная, основанная на памяти ZCPR3 альтернатива CP/M программ SUBMIT и XSUB. Позволяет не только командные сценарии практически неограниченной длины, но и контролировать работу многих программ "изнутри" с вводом из скриптов. Очень быстрая и универсальная, она очень тесно взаимодействует с ZCPR3 через специальные байты сообщений.

## Утилиты ZRDOS

Эти утилиты работают только с BDOS заменой ZRDOS.

**AC**

Утилита архивного копирования AC разрешает вам автоматически выбирать для копирования те файлы, которые не были отмечены, как заархивированные. У нее есть много параметров, включая стирание исходного файла после завершения копирования, сброса дисковой системы перед сменой носителей и другие.

**COMP**

COMP - видео-ориентированная утилита сравнения файлов, более мощная, чем DIFF, которая графически отображает различия между двумя файлами. Она также управляет регистрами ZCPR3, полезная функция в условных командных сценариях на основе результатов сравнения файлов.

**DFA**

Отображает атрибуты файла, которые могут быть установлены утилитами ZRDOS SFA или ZCPR3 PROTECT.

**DUMP**

Выводит содержание дампа файла, в расширенном стиле DDT - шестнадцатеричном и ASCII на экран или принтер.

**PUBLIC**

Эта утилита устанавливает диски и пользовательские области как общедоступные или частные. Общественные области - великолепные места для размещения оверлейных программ для таких программ как WordStar, разрешая полный доступ к программе из любой пользовательской области.

**SFA**

Устанавливает атрибуты файла - защита колеса (wheel, вы не можете записать в файл с защитой колеса, если байт колеса сброшен), системный, только для чтения и архив.

**VIEW**

Сканер текстового файла. Позволяет переключатель отображение управляющих символов и разрешает перемещение вперед и назад в файле.

**VTYPE**

Утилита сканирования файла, разрешающая просмотр текстового файла вперед и назад с управляемым уровнем прокрутки, поиском строки, быстрая переходом, перемещением в конец и начало файла, и т.д. Полностью экранно-ориентированная со встроенной системой контекстной справки.