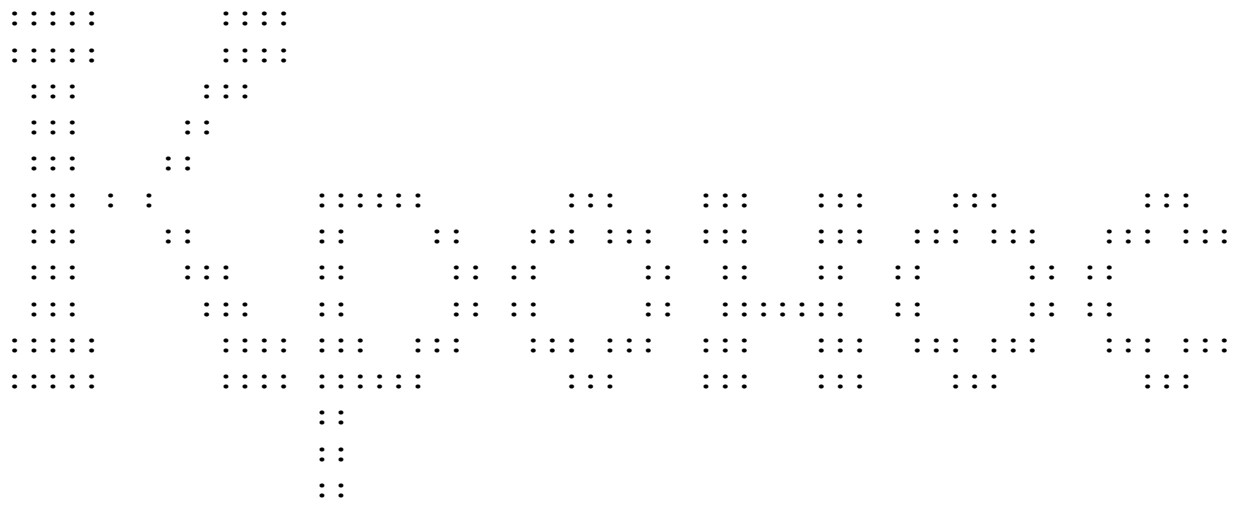
- Сибирское отделение АН СССР -

- Институт систем информатики -



УТИЛИТЫ ОС EXCELSIOR

Новосибирск - 92

Мы надеемся, что информация, изложенная в этом томе, является верной. Будем признательны, если читатель критически подойдет к ней и все возникшие вопросы, замечания, исправления и пожелания доведет до нашего сведения. Выражаем благодарность всем, кто прямо или косвенно принимал участие в подготовке и издании тома.

Наш адрес:

630090, Новосибирск-90, проспект ак. Лаврентьева, 6, ВЦ СОАН СССР, к.503, тел. 35-50-67.

Для сотрудников ВЦ СОАН тел. 8-97.

Последнее изменение 13.02.92

СОДЕРЖАНИЕ

ПРЕДВАРИТЕЛЬНЫЕ ЗАМЕЧАНИЯ

Пользовательский интерфейс в системе осуществляют системные утилиты.

Утилиты системы представляют собой несколько наборов программных модулей. Каждый из наборов предназначается для выполнения определенных функций.

Некоторые утилиты являются специфическими для конкретного оборудования и не описываются в данном томе.

Данный том является справочным пособием по утилитам ОС Excelsior. Основная часть тома посвящается подробному описанию утилит. В этой части изложение ведется в следующей форме:

- название утилиты;
- этимология названия;
- назначение;
- способы запуска;
- возможные ключи;
- примеры использования.

В описании некоторых утилит возможны и другие разделы.

Прежде чем перейти к изложению материала справочного характера, сделаем несколько общих замечаний, касающихся всех утилит и связанных с ними соглашений, принятых в системе.

1. Функции утилит

Утилиты ОС Excelsior можно условно разбить на разделы в соответствии с областью их применения. Заметим, что некоторые утилиты (утилиты администратора) не являются общепользовательскими, поскольку требуют специальной подготовки и налагают большую ответственность на использующее их лицо.

Приведем списки разделов утилит с указанием их назначения.

1.1. Утилиты работы с файлами:

1) cat - конкатенация файлов;
2) chmode - обслуживание защиты файлов;
3) cp - копирование файлов;
4) diff - посимвольное сравнение двух файлов;
5) du - информация об использовании диска;
6) find - поиск и обработка файлов;
7) grep - поиск образца в файлах;
8) ln - дополнительное именование файлов;
9) ls - информация о именах и атрибутах файлов;
10) mkdir - создание новой директории;
11) mkfs - создание файловой структуры;

12) mou - монтирование диска в файловую систему;
13) mv - переименование файлов;
14) rm - удаление указанных файлов.

1.2. Утилиты системы программирования Модула-2:

1) ex - универсальный редактор текстов;
2) hi - причины аварийного завершения задачи;
3) mx - Модула-компилятор;
4) pm - менеджер программных проектов;
5) vx - информация о кодофайле программы.

1.3. Утилиты сервиса:

1) ascii - информация о ASCII-кодировках;
2) dt - работа со временем;
3) msflx - работа с файловой системой MS/DOS;
4) pkr - упаковщик кодофайлов;
5) ptp - вывод на печатающее устройство;
6) tmspo - хранитель времени.

1.4. Утилиты администратора:

1) boo - перезагрузка ОС или других автономных задач;
2) config - подготовка конфигурации ОС;
3) dsu - управление устройствами;
4) fschk - проверка файловой структуры;
5) fsdb - редактирование файловой структуры;
6) human - обслуживание системы паролей;
7) login - вход в ОС;
8) mknode - создание специальхых файлов (узлов);
9) mshell - интерпретатор командных файлов;
10) shell - пользовательская оболочка.

2. Порядок запуска

Запуск утилиты (как и любой другой задачи) в системе обеспечивает пользовательская оболочка системы shell. Описание ее функций и команд можно найти в книге "ОС Excelsior для всех".

Все утилиты ОС Excelsior имеют стандартный интерфейс. Вот как выглядит запуск утилиты:

```
запуск утилиты::
     имя_утилиты { параметр | равенство | ключи }

параметр::
     строка

равенство::
     имя=строка
```

ключи::

```
-{ символ } | +{символ}
```

Примеры:

```
ls *.doc -cw
```

Имя утилиты 'ls', параметр '*.doc', ключи 'c' и 'w'.

Порядок ключей и равенств в командной строке не имеет значения. Так, все эти команды одинаковы:

```
ls *.doc -cw
ls -c *.doc -w
ls -cw *.doc
```

Ключи, которые утилита не использует, игнорируются. Внутри утилиты работа с параметрами ведется с помощью библиотеки tskArgs.

Ключ 'h' является стандартным и означает выдачу краткой подсказки о порядке запуска утилиты на стандартный вывод. Кроме этого, некоторые утилиты выдают подсказку в случае вызова их без параметров или с некорректным набором параметров.

3. Запрос на подтверждение

Практически все утилиты в процессе работы запрашивают подтверждение на выполнение каких-либо действий. Ответом на такой запрос может быть какой-либо символ. Семантика основных ответов следующая:

y (yes) - выполнить действие;
n (no) - не выполнять действие;
q (quit) - не выполнять действие и прекратить действия над остальными объектами;
a (all) - выполнить действия над всеми остальными объектами без запросов на подтверждение.

4. Параметры файловых утилит

Большая группа утилит предназначается для работы с файловой системой. Остановимся подробнее на особенностях этих утилит.

4.1. Образец группы файлов

Все файловые утилиты используют понятие образца группы файлов.

В этом разделе мы дадим лишь краткое и, по возможности, конкретное описание образца, пригодное для немедленного применения. Более точное и строгое описание дается в документации по ОС Excelsior в разделе, посвященном регулярным

выражениям.

Образец сравнивается с именами файлов; с одними он сопоставляется, с другими - нет. Таким образом, образец определяет некоторую группу имен файлов, с которыми он сопоставился. Необходимая для определения не одного, а множества имен файлов степень свободы достигается использованием в образце метасимволов.

4.1.1. Допустимые метасимволы

В образце могут встретиться следующие метасимволы:

? - обозначает произвольный символ.

```
aaa.?     =>  aaa.d  aaa.m aaa.@
ar?.doc   =>  ar1.doc ar2.doc ar3.doc
```

* - обозначает последовательность произвольных символов любой длины, возможно пустую;

```
*.*       => text.m   .profile   sys.doc
```

[символы] - обозначает один из перечисленных в скобках символов; символы могут быть заданы перечислением или диапазоном. Пример: [a,b,0-5].

```
ar[123].doc   => ar1.doc ar2.doc ar3.doc
StdIO.[dm]    => StdIO.d  StdIO.m
```

{символы} - обозначает последовательность из перечисленных в скобках символов любой длины, возможно пустую;

```
{abc}.t   => aaa.t  aba   .t   ca.t
```

| - бинарная операция ИЛИ. Указывает, что в этом месте может находиться последовательность символов, определяемая левым или правым операндом.

```
*.d|*.m  => xRef.d  xRef.m
```

& - бинарная операция И. Указывает, что в этом месте может находиться последовательность символов, которая должна сопоставляться как с левым, так и с правым операндом операции '&'.

```
*.d & a*   эквивалентно a*.d.
```

^ - унарная операция НЕ. Указывает, что в этом месте может находиться последовательность символов,

которая НЕ сопоставляется с операндом.

^*.? означает "все файлы, кроме *.?".

// - обозначает последовательность директорий произвольной глубины, возможно пустую. Может быть использован в образце только один раз.

/usr//txt => /usr/leo/wrk/txt /usr/txt

Неправильно: /usr//mydir/wrk//*.m

4.1.2. Использование скобок

Кроме перечисленного выше, в образце могут встретиться скобки, определяющие порядок выполнения операций. Например, запись (*.d|*.m)&a* определяет файлы с именами вида:
a*.d a*.m

4.1.3. Примеры

*.? - определяет группу файлов на текущей директории вида:

abc.m, ...

chapter{0-9}.doc

- определяет группу файлов на текущей директории вида:

chapter0.doc
chapter1.doc
chapter123.doc

/abc//*.cod - определяет группу файлов с расширением ".cod" на всех поддиректориях директории abc.

4.1.4. Порождение имени файла-получателя

Для таких утилит, как утилиты копирования или переименования, образец группы файлов может содержать, кроме этих, метасимволы, которые используются для порождения имени файла-получателя.

а) $цифра

Метасимвол $цифра может быть выставлен в образце вслед за метасимволами '*', '?', '[]', '{}', а также после

закрывающей круглой скобки. После этого в имени файла-получателя символ $цифра обозначает строку символов, которая сопоставляется с метасимволом, предшествовавшем $цифра в имени файла-источника.

Пример:

```
cp *$1  $1.bak

    aaaa        =>  aaaa.bak
    t.m         =>  t.m.bak

cp (abc*|def*)$1.*$2 $1_$2

    abc.cod     =>  abc_cod
    defMain.m   =>  defMain_m
```

б) $.

В имени файла-получателя можно использовать метасимвол '$.' для обозначения собственно имени файла (то есть его имени на директории, на которой он расположен). Например:

```
cp *.m  $.#

        a.m     => a.m#
        abcd.m  => abcd.m#
```

в) $@

Метасимвол '$@' обозначает последовательность имен директорий, соответствующих метасимволу '//' в имени файла-источника. Так, запись

```
cp /abc//*   /def/$@/$.
```

означает копирование файлового поддерева директории abc в директорию def, с сохранением имен поддиректорий и файлов, например:

```
/abc/usr/ned/txt.m => /def/usr/ned/txt.m.bac
```

ЗАМЕЧАНИЕ. Запись aaa/*|bbb/* означает не файлы вида aaa/* и bbb/*, а файлы вида aaa/*/* и aaa/bbb/* . А запись (aaa/*|bbb/*) неправильна, надо так: (aaa|bbb)/* (см. в документации по системе).

4.2. Параметры времени

В командной строке запуска файловой утилиты могут быть равенства со следующими именами: after, before, cafter,

cbefore. При этом в группы выбранных файлов попадают файлы, которые модифицированы до указанного времени (after), после указанного времени (before) или, соответсвенно, созданы до или после указанного времени (cafter, cbefore).

Время и дата могут быть представлены в двух форматах:

```
[ dd/mm/yy ][,][hh:mm[.ss]]
[ yy#dd#mm ][,][hh:mm[.ss]]
```

где dd - число, mm - месяц, yy - год, hh - час, mm - минуты, ss - секунды. Если определение даты отсутствует, то подразумевается текущий день. Если отсутствует спецификация времени, то в случая равенств after, cafter время равно 00:00.00, в случае равенств before, cbefore время равно 23:59.59.

Пример:

```
after=10/02/90   before=22/02/90,23:30
```

Фарфуркис прочитал вслух:

- "Вопрос: что у нея... гм... у нея внутри за лпч..." Лэпэче... Кэпэдэ, наверное? Что это за лэпэче?
- Лампочка, значит, - сказал старичок, хихикая и потирая руки. - Кодируем помаленьку.

А. и Б. Стругацкие. Сказка о Тройке

ascii

aMERICAN sTANDARD cODE FOR iNFORMATION iNTERCHANGE

Визуализирует принятую в системе кодировку.

ascii [-r]

Выдает по нажатии клавиши ее восьмеричную кодировку, в квадратных скобках - шестнадцатеричную и десятичную. Утилита прекращает работу после нажатия ^C. Ключ -r (raw - сырой) выключает все драйверные перекодировки при чтении с клавиатуры.

ascii -t

Выдает таблицу (table) соответствия КОДИРОВКА -> СИМВОЛ.

КЛЮЧИ:

- h - подсказка;
- r - без драйверной перекодировки;
- t - таблица соответствия кодов и символов.

```
boo
booT STRAP
```

Загружает в память автономную (stand-alone) задачу. Применяется для загрузки системы.

```
boo <имя_файла>
```

Если указанный файл является специальным дисковым файлом, то система загружается из файла с номером 1 на указанном диске.
Если же это обычный файл, то система загружается из него.

ПРИМЕРЫ:

```
boo /dev/wd0
```
пытается загрузить систему с диска wd0.

```
boo system.boot
```
загружает образ системы из файла system.boot.

КЛЮЧИ:

- h - подсказка.

cat

CONcatTENATE files

Слияние перечисленных файлов в указанном порядке. Результат по умолчанию выводится на терминал, но может быть стандартным образом направлен в файл.

cat {имя_файла}

КЛЮЧИ:

-h - подсказка.

ПРИМЕРЫ:

1)
cat a b c >d

Запись файлов a,b,c в файл d;

2)
cat a b

Объединение файлов a и b выводится на терминал;

3)
cat a

Вывод файла a на терминал.

4)
cat a >/dev/tty2

Вывод файла a на терминал номер 2. Если при этом вместо терминала номер 2 подключено последовательное печатающее устройство и на нем установлен терминальный драйвер, то файл напечатается на бумаге. Этот способ может оказаться полезным, если у вас нет соответствующего драйвера на печатающее устройство.

5)
cat /dev/dk1 >/dev/dk0

Запись содержимого диска dk1 на диск dk0. Происходит копирование носителей "один в один". При этом не требуется, чтобы носитель на dk0 был размечен (см. mkfs), поскольку после операции он будет представлять точную копию (или часть точной копии) носителя-источника, сохранив и его разметку. Естественно, при разных размерах и типах носителей правильность файловой структуры на носителе-приемнике не гарантируется.

chmode

chANGE FILE PROTECTION mode

Меняет защиту файла.

Маска защиты - битсет длиной 11. Ознакомиться с текущим состоянием маски можно при помощи утилиты ls с ключом -u.

Первые два бита имеют значение только для файлов, содержащих исполняемый код задачи: в первом выставляется режим привилегии для задачи, во втором делается пометка, будет ли владельцем задачи владелец кодофайла или пользователь, запустивший задачу.

Девять следующих битов разбиты на три группы. Первая группа определяет защиту по отношению к владельцу, вторая - по отношению к дружественным пользователям, третья - ко всем остальным. В каждой группе имеется бит, определяющий защиту файла от чтения, бит, определяющий защиту файла от записи, и бит, определяющий защиту файла от запуска кода, если таковой содержится в файле.

Подробности о защите файлов читайте в описании файловой подсистемы в книге "ОС Excelsior для всех".

chmode суперобразец ["pro="pro0] [-dq]

Задает маску защиты в явном виде следующим образом:

```
pro0 = PURWXRWXRWX
P    = ("P" | "p" | "-")
U    = ("U" | "u" | "-")
R    = ("R" | "r" | "-")
W    = ("W" | "w" | "-")
X    = ("X" | "x" | "-")
```

Здесь везде прочерк означает отсутствие разрешения, а значения символов следующие:

p - привилегированность запущенной задачи;
u - владельцем задачи является владелец файла, содержащего код;
r - разрешено чтение;
w - разрешена запись;
x - разрешено исполнение.

chmode суперобразец ["pro="{pro1}] [-dq] ["owner="user]

```
pro1 = ("o"|"a"|"g") {("+"|"-") ("r"|"w"|"x"|"p"|"u")}
```

Меняет значение указанных позиций в маске защиты. Здесь

o - владелец,
g - дружественная группа пользователей,
a - все остальные.

+ - разрешение действия, - отмена.

Значение символов, определяющих действие над файлов, описано выше.

chmode суперобразец ["owner="user]

Смена владельца файла. Может производиться одновременно с изменением маски защиты.

КЛЮЧИ:

q - не запрашивать подтверждение на изменение;
h - подсказка;
v - не выдавать никаких сообщений на экран (для открепленного запуска);
T - для изменения маски файлового дерева;
d - изменение маски для директорий.

Запрос утилиты

Если ключ -q не указан, утилита выдает запрос на подтверждение действия. Возможные ответы:

y - (yes) выполнить действие над предложенным объектом;
n - (no) не выполнять действие над предложенным объектом;
q - (quit) прекратить исполнение утилиты;
a,A - (all) все остальные объекты без вопросов;
i - (skip) пропустить поддерево;
CTRL_X - открепить утилиту.

config
system configuration utility program

Утилита реконфигурации системы.

config -m имя_головного_модуля [stack=N]

Сборка автономной (stand-alone) задачи. Результат - образ задачи - помещается в файл с именем головного модуля с расширителем .boot. Размер процедурного стека указывается в словах и по умолчанию равен 512 (2Кб).

config -c имя_системы {дополнительные_задачи}

Конфигурация системы.
В образе работоспособной системы содержатся:
- собственно система;
- список дополнительных кодофайлов;
- список задач, которые необходимо запустить при инициализации системы;
- список конфигурационных строк.

После инициализации система запускает задачи, указанные в списке. Как правило, этот список содержит необходимый комплект драйверов и shell.
Список конфигурационных строк определяет некоторые параметры системы, например, размер и местоположение дискового кэша.

Все эти дополнительные вещи берутся из файла имя_системы.cnf. В этом файле кофигурационные строки помечены в начале символом '$'. Остальные строки содержат имена задач. Если в конце строки стоит символ "&", то задача, указанная в строке, должна быть запущена после инициализации системы. Если "&" не стоит в конце строки, то коды указанной задачи только добавляются в образ системы. Размер стека в килобайтах может быть указан явно в начале строки в квадратных скобках. По умолчанию он равен 1Кб.

Дополнительные задачи могут быть указаны в строке аргументов при запуске утилиты config.
Ключи c и m можно указывать одновременно. При этом последовательно выполняется создание образа и конфигурация.

config -r имя_системы

Удаляет конфигурацию, оставляя собственно образ системы.

```
config -l  имя_системы
```

Выдает информацию о том, что есть в образе.

```
config -b имя_корнев_директории имя_системы
```

Система из указанного образа (из файла имя_системы.boot) копируется в файл с номером 1 на диск, имя корневой директории которого указано.

```
config -B имя_диска  имя_холодного_загрузчика
```

Записывает файл имя_холодного_загрузчика.boot в нулевой блок указанного диска.

```
config -I   имя_кодофайла_загрузчика
config -I   имя_устройства имя_кодофайла
```

Изготавливает холодный маленький (не более 4Кб) холодный загрузчик и записывает на устройство (например, ПЗУ), если оно указано. Ключ используется только для внутренних нужд.

-v - выключает выдачу информации при действиях, определяемых ключами -m, -c, -r.

-0 - не использовать буферов чтения/записи при создании файла системы; по умолчанию используется 4 буфера.

-h - подсказка.

ПРИМЕРЫ:

Приведем текст файла конфигурации:

```
[1] DKwsWD&
[1] DKwsFD&
[1] TTwsBM&
[5] shell $home -root :wd0&
$   CASH  MEM     128K
```

Файл содержит запуск трех драйверов и shell, а также указание, что дисковый кэш размером 128Кб необходимо разместить в основной памяти.

Приблизительный перечень действий для создания системного носителя:

```
config -m booter              сборка загрузчика
config -B /dev/wd0 booter     запись загрузчика в нулевой блок
config -mc system ex          сборка системы с редактором
config -b / system            копирование системы на место
```

- Я себе одну такую заведу, на пробу, - задумчиво сказал Рыцарь. - Или даже парочку. Одним словом, штук пятнадцать.

Л.Кэрролл. Алиса в Зазеркалье

cp
cOpY FILE

Физическое копирование файлов.

```
owner:
    [owner=user_name]
times:
    [after=time] [before=time] [cafter=time] [cbefore=time]
    time = [dd/mm/yy,][hh:mm[.s]]  (Europian style)
         | [yy#dd#mm,][hh:mm[.s]]  (American style)
query:
       (y|n|q|a|A|i|^X) = (Yes|No|Quit|all|All|skIp|IPR)
```

Leopold, Mar 1 90

cp {исходное_дерево} [дерево-получатель]
[-ключи] [время] [владелец]

Копирует файлы, заданные множеством исходных деревьев. Об образцах и получателях читайте в разделе "Параметры файловых утилит".

Если дерево-получатель не указано, копирование осуществляется на текущую директорию.

Если файл с именем, указанным в "получателе", уже существует, при копировании его старое содержимое будет удалено.

Замечание. Избыточное на первый взгляд многообразие ключей предоставляет широкие возможности для копирования сложных структур с помощью командных файлов, что мы и рекомендуем.

КЛЮЧИ:

h - подсказка;

q - не запрашивать подтверждение на копирование;

c - попытаться скопировать в уже существующий файл, если такой имеется (без создания нового файла); заметим, что если во время копирования возникнут ошибки (на устройстве), то возможна ситуация, когда часть уже существующего файла будет новой, а часть - старой (кто знает?);

r - удалить исходные файлы после удачного копирования; заметим, что если утилита запущена с ключом i, то любое копирование считается удачным;

m - эквивалентно -cr;

v - сравнить с исходным после копирования;

i - игнорировать ошибки чтения и записи;

d - создавать недостающие директории;

o - копировать только файлы, уже существующие на директории-получателе; заметим, что этот ключ несовместим с ключом x;

x - копировать только файлы, не существующие на директории-получателе;

b - копировать только файлы, отличающиеся по времени модификации или не существующие на получателе;

A - копировать только файлы, которых нет на получателе, или которые были записаны позднее (время записи больше), чем существующие на получателе;

B - копировать только файлы, которых нет на получателе, или которые были записаны раньше (время записи меньше), чем существующие на получателе;

C - записать текущее время в качестве времени создания файла (без этого ключа время создания файла-получателя равно времени создания файла-источника);

W - записать текущее время в качестве времени модификации файла (без этого ключа время модификации файла-получателя равно времени модификации файла-источника).

+H - записать копируемый файл как скрытый (-H - как видимый);

z - для копирования файлов-устройств (узлов); заметим, что при этом автоматически устанавливается ключ c;

y - для копирования файлов, помеченных как системные;

l - не выдавать никакой информации о копировании;

R - копирование на устройства (с устройств) подряд; позволяет большой файл скопировать на несколько

дискет подряд и обратно;

U - при копировании файлы-получатели наследуют владельца и маску защиты; без этого ключа владелец и маска берутся из окружения Shell.

ПРИМЕРЫ:

```
cp first second
```

копирование файла first в файл second.

```
cp *.m *.d имя_дир
```

копирование файлов с расширителями .m и .d на директорию -имя_дир- .

```
cp ./*  ../second/$.#
```

скопировать все файлы текущей директории, добавив к имени каждого '#', на директорию -second-, лежащую на директории, содержащей текущую.

```
cp /abc//* /def/$@/$. -d
```

скопировать все файлы поддерева abc с сохранением файловой структуры и имен исходных файлов, создавая при этом недостающие директории (-d).

```
cp abc/ ttt/  -b after=19/02/90
```

скопировать все файлы с директории abc, модифицированные после 19 февраля 1990 года, кроме тех файлов, которые уже были скопированы (-b), на директорию ttt с сохранением имен исходных файлов.

diff

diffERENCE

Сравнение файлов и выдача шестнадцатеричного дампа.

> Прицеливаясь взглядом то в одного барсука, то в другого, Похититель искал различия.
>
> "Ага, - думал он, - у этого барсука две ноги, а у этого три. Вот вам и различие".
>
> Ю.Коваль. Пять похищенных монахов

diff <имя_файла_1> <имя_файла_2> [-hlw]

Утилита осуществляет побайтовое сравнение двух файлов. Выдается длина обоих файлов в байтах.

Если найдены различия, выдаются шестнадцатеричный дамп и соответствующие символы ASCII для обоих файлов. На отличающиеся байты указывает символ '.'. Если различий нет, дамп выдается только по ключу.

diff <имя1> [-hlw] [ofs=<номер блока>]

Выдает дамп (шестнадцатеричное представление) указанного файла.

Замечание. Использование метасимволов в именах файлов не допускается.

КЛЮЧИ:

h - подсказка;
w - дамп выдается порциями с задержкой; для выдачи следующей порции нажимается любая клавиша;
l - выдача дампа в случае совпадения содержимого сравниваемых файлов;
ofs - номер байта, с которого начинать выдачу дампа.

ПРИМЕРЫ:

diff new.txt old.txt ofs=5000

сравнение двух файлов new.txt и old.txt, начиная с байта 5000;

diff txt -w
выдача дампа порциями;

```
diff /dev/fd0
```

выдача дампа диска на устройстве fd0. Обратите внимание - диск не обязательно должен быть монтирован в файловую систему.

dsu

dEVICE sETuP

Устанавливает параметры дисковых устройств, узлы которых расположены на текущей директории либо на директории /dev.

Перед использованием утилиты убедитесь, что никакая другая задача не работает в данный момент с устройством.

Могут изменяться следующие параметры:
s= число секторов на дорожке;
t= число дорожек на диске;
h= число головок (или сторон);
ss= размер сектора в байтах;
fs= номер первого сектора на дорожке;
ls= номер последнего сектора на дорожке;
v= число секторов на диске;
rs= число резервных секторов;
rate= размер шага головки (не меняетя);
prec= номер дорожки, с которой начинать прекомпенсацию (не меняется).

dsu -l
Выдает список устройств с указанием типа и состояния.

dsu [имя_устройства]
Выдает текущее состояние параметров диска. При этом в скобках выдается тип диска (гибкий, винчестерсткий) и тип формата (потрековый, посекторный).

dsu [имя_устройства] {<изменяемые_значения_параметров>}
Выдает текущее состояние параметров диска и устанавливает новые значения указанных параметров.

dsu [имя_устройства] -v
Проверяет диск.

dsu [имя_устройства] -m

Применяется для дисков, сформатированных в старых версиях системы. Производит запись параметров в нулевой сектор диска.

dsu [имя_устройства] -p
Отключает диск.

dsu [имя_устройства] -F {<изменяемые_параметры>}
Форматирует диск на указанном устройстве с указанными параметрами. Если вам подходят текущие значения параметров, их указывать не нужно.

dsu [имя_устройства] -f {<изменяемые_параметры>}
Форматирует диск на указанном устройстве с указанными

параметрами. При этом проверяется маркировка диска, после форматирования автоматически осуществляется верификация и разметка диска.

dsu [имя_устройства] -sf

"Мягкое" форматирование - с сохранением информации на форматируемых треках. Применяется при большом количестве поврежденных секторов и невозможности спасти информацию в другое место. Ясно, что информация с поврежденных секторов будет утрачена.

dsu -t

Выдает статистику файловой системы следующего характера: размер дискового кэша, сколько секторов прочитано с диска, сколько - из кэша с момента начала работы системы.

ПРИМЕРЫ:

dsu /dev/fd1 t=40 h=2 s=5 ss=1024 fs=1 ls=5 v=400 rs=0

Устанавливает на устройстве fd1 указанный формат. Заметим, что это формат, принятый для рабочей станции.

"What a funny watch!" she remarked. "It tells the day of the month, and doesn't tell what o'clock it is!"
"Why should it," muttered the Hatter. "Does your watch tell you what year it is?"

Lewis Carroll
Alice's adventures in Wonderland

dt

dATE & tIME SERVICE

Показывает и редактирует время в системе.

dt
Показывает текущее время.

dt -c
Работает как секундомер. Прекращается нажатием любой клавиши.

dt -e
Редактирование системного времени.
Время можно редактировать при помощи стрелок Up и Dw (увеличение и уменьшение, соответственно, дня, месяца, часа и т.д.). Для выхода из режима редактирования с установкой исправленного времени нажмите CR. Выход без записи - Esc.

dt <время>
Устанавливает указанное время. Строка <время> должна быть в одном из форматов:

```
<время>::=   [DD/MN/[19]YY,][HH:MM[.SS]]
           | [[19]YY#DD#MM,][HH:MM[.SS]]
```

, где DD обозначает день, MN - месяц, YY - две последние цифры года, HH - час, MM - минуты, SS - секунды.
Если дата не указана, по умолчанию устанавливается текущий день. Если не указаны часы, минуты и секунды, они по умолчанию полагаются равными нулю: 00:00.00.

КЛЮЧИ:

e - редактировать время;
c - секундомер;
h - подсказка;
- версия.

ПРИМЕРЫ:

```
dt -e
```

На экран выдается время, дата и день недели:

```
Fri Jul 14 15:31.55  1989
```

Все параметры теперь можно отредактировать.

- Мест нет! Мест нет! - дружно закричали Заяц и Шляпа, как только заметили Алису.
- Места сколько хочешь! - возмутилась Алиса.

du
dISK uSAGE

Информирует о занятости дискового пространства.

du <полное_имя_директории>

Выдает информацию о размерах (в килобайтах) свободного и занятого простанства диска, смонтированного на указанной директории. Если на директории диск не смонтирован, выдает информацию о диске, на котором расположена указанная директория.

du

Выдает информацию о диске, на котором расположена текущая директория.

КЛЮЧИ:

-h - подсказка.

ПРИМЕРЫ:

du

На экран выдается информация вида:

at "." total: 19,552K free: 588K (3%) busy: 18,964K (97%)

Это значит, что размер диска 19552 килобайта, что занято 18964 килобайта, а свободно 588 килобайт.

du /usr/flm/mnt

В случае, если на директории mnt смонтирован диск, выдается информация об этом диске:

at "/usr/flm/mnt" total: 790K free: 12K(1%) busy: 778K(99%)

Если же диск не смонтирован, выдается информация о диске, на котором расположена директория mnt:

at "." total: 19,552K free: 588K (3%) busy: 18,964K (97%)

ex
eDIT TExT

Экранный редактор текстов.

ex <имя_файла> [-w]

Позволяет редактировать файлы шириной до 256 символов. Максимальный размер файла зависит от доступной оперативной памяти. Редактор работает на любых терминалах, обслуживаемых ОС Excelsior. Для выхода из редактора нажмите клавиши CTRL E - выход с записью всех внесенных исправлений; CTRL C - выход без записи.

Есть возможности:
- изменение параметров редактирования;
- выход в shell редактора;
- позиционирование в файле;
- поиск в файле;
- контекстная замена;
- работа с файлами (переименование и пр.);
- работа с областью текста;
- форматирование абзаца текста;
- установка макросов;
- установка фильтров.

КЛЮЧИ:

-w (NOT wRITE) - разрешить только чтение файла.

РАЗДЕЛИТЕЛИ:

Редактор работает с текстом в кодировке КОИ-8. Строки могут быть разделены при чтении (см. библиотеку ASCII):

```
ASCII.HT  (11c)      - заменяется  при чтении на пробел;
ASCII.NL  (36c)      - формат  ОС Excelsior;
ASCII.NULL  (00c)    - формат ОС Excelsior;
ASCII.LF   (12c)     - Unix-формат;
ASCII.CR&LF (12c,15c) - RT-11-формат.
```

Файл пишется с разделителем ASCII.NL. Других контрольных символов в файле НЕТ. Последняя строка завершена NL.

КЛАВИАТУРА:

Редактор использует три управляющих клавиши F1, F0 и F2, а также набор традиционных управляющих символов - стрелки, дублирование и т.д..

1. Перемещения по тексту

переход вверх на строку	UP
переход вниз на строку	DOWN
переход на символ влево	LEFT
переход на символ вправо	RIGHT
табуляция вправо	TAB
табуляция влево	BACK TAB
переход на начало следующей строки	ENTER
переход на 16 строк вверх	PageUp
переход на 16 строк вниз	PageDw
переход на начало текущей строки	CR
переход на начало следующей строки со стиранием хвоста текущей строки	LF

2. Вставка и удаление

вставка символа с подвижкой хвоста строки вправо	InsCh
удаление символа с подвижкой хвоста строки влево	DelCh
замена символа слева на пробел с переходом на него курсора	BACKSPACE
удаление хвоста строки	EraseEOL
удаление хвоста строки с приклеиванием хвоста следующей	DelLine
вставка строки с отламыванием хвоста на следующую строку	InsLine
отмена последнего DelLn	F1 F2

3. Операции над строками

дублирование хвоста строки	DupLine
обмен хвостов вверх от курсора	SwapUp
обмен хвостов вниз от курсора	SwapDw
склеивание строк	F1 DelLn
разрыв строки	F1 InsLn
дублирование хвоста поверх следующей строки	F1 DupLn

4. Операции над единицами текста

Будем понимать под словом то, что понимается в русском (и

английском) языке, а СЛОВОМ назовем набор символов, не содержащий пробелов, ограниченный пробелами.

переход к началу слова	F1 UP
переход к концу слова	F1 DOWN
удаление слова справа от курсора	F1 DelCh
удаление слова слева от курсора	F1 BACKSPACE
переход к началу СЛОВА	F2 LEFT
переход к концу СЛОВА	F2 RIGHT
удаление СЛОВА справа от курсора	F2 DelCh
удаление СЛОВА слева от курсора	F2 BACKSPACE
переход к началу строки	F1 LEFT
переход за конец строки	F1 RIGHT
переход на конец текста	F1 PageDw
переход к началу текста	F1 PageUp
форматирование текущего абзаца	F0 1
центрирование заголовка	F0 2
возврат к первоначальному содержимому строки	F2 F1

5. Выход в другие режимы редактора

командный режим	F1 F1
установка параметров (SETUP-монитор)	F1 F0
вход в shell редактора (выход - ESC)	F0 PgDn

КОМАНДНЫЙ РЕЖИМ:

Переход в командный режим: F1 F1. На экране появится приглашение: >>> . Для возврата нажать CR.

Командная строка ::= {ПРЕФИКС} КОМАНДА [АРГУМЕНТЫ]

ПРЕФИКСЫ::= {! | $ | # | =}

"!" - начать поиск с начала файла;

"$" - искать в диапазоне строк, помеченном маркерами;
"#" - рассматривать диапазон как прямоугольник;
"=" - вставить прямоугольник с вертикальной раздвижкой.

Команды, допустимые в командном режиме, описаны далее.

ВЫХОД В SHELL:

Для выхода в shell редактора нажмите F0 PgDn. Shell редактора полностью аналогичен пользовательской оболочке системы. Таким образом, из shell редактора можно запускать любые задачи, в том числе все утилиты, компилятор, редактор и т.д.. Например, допустимо запускать один из другого несколько редакторов - только не потеряйтесь в уровнях.

Чтобы вернуться к режиму редактирования, нажмите Esc.

ПОИСК В ФАЙЛЕ И ЗАМЕНА ОБРАЗЦА:

Поиск производится в командном режиме.

[!$#]"f"<образец> - поиск образца;
"n" - найти образец еще раз.

В этом случае вместо последовательности F1 F1 n CR можно набирать только F0 RIGHT.

[!]"p"<имя_процедуры> - найти процедуру;
[!]"e"<имя_процедуры> - найти конец процедуры.

Смысл префиксов:

"!" - начать поиск с начала файла;
"$" - искать в диапазоне строк, помеченном маркерами;
"#" - рассматривать диапазон как прямоугольник.

Например:
>>>fCASE
Редактор будет искать образец -CASE-, начиная с текущей строки.

Замена образца производится в командном режиме.

[!$#]"r"/"<образец>"/"<замена> - заменить образец.

Вместо символа "/" можно употребить любой разделяющий символ.
Например:
>>> #r$first$second
В прямоугольнике, помеченном маркерами, 'first' заменится на 'second'.

>>> r/first/second
Везде, начиная с текущей строки, при получении подтверждения, 'first' заменится на 'second'.

ПОЗИЦИОНИРОВАНИЕ:

Позиционирование в файле (N,M - числа) производится в командном режиме.

N - перейти на строку с номером N;
+N - перейти вперед на N строк;
-M - перейти назад на N строк;
N.M - перейти на строку с номером N, колонку с номером M.

РАБОТА С ФАЙЛАМИ:

Производится в командном режиме.

"w" (wRITE)

- записать отредактированный файл, не покидая редактора. То же действие можно произвести, нажав последовательно F0 и w.

"s"<имя_файла> (sET NAME)
- сменить имя файла на указанное, например:
>>>sStdIO.m
файлу присваивается имя StdIO.m.

"<"имя_донора
Считывается файл -имя_донора-, например:
>>> <../etc/myfile

РАБОТА С ОБЛАСТЬЮ ТЕКСТА:

ОБЛАСТЬ всегда маркируется как прямоугольная, но используется, по умолчанию, как диапазон строк. Префиксы #,= заставляют редактор произвести операцию с прямоугольной областью.

Область помечается маркерами:

F0 [- установка маркера на начало области;
F0] - установка маркера на конец области;
F0 . - вместо установки двух маркеров, если в качестве области взята одна строка.

После того, как маркеры установлены, можно выполнить действия (в скобках указаны допустимые префиксы):

F0 UP - спозиционироваться на начало области;
F0 DOWN - спозиционироваться на конец области.

Следующие действия производятся в командном режиме (F1 F1 для входа и CR для выхода):

/ - сбросить установленные маркеры;
d - delete удалить область (#);
o - over наложить область (#);
i - insert вставить область (#=);
m - move переместить область;
c - clear очистить область (#);
>имя_файла - записать область в файл -имя_файла-;
>>имя_файла - дописать область в конец файла -имя_файла-.

ФОРМАТИРОВАНИЕ:

Можно форматировать текущий абзац.

Параметры форматирования задаются двумя способами - устанавливаются в SETUP-мониторе (см. ex.setup) или явным способом на экране.

В SETUP-мониторе выставляются левый и правый края отформатированного текста и отступ красной строки. Эти параметры SETUP можно менять, не входя в монитор, прямо с экрана:

F0 | - устанавливает отступ красной строки;
F0 (- левый край отформатированного текста;
F0) - правый край отформатированного текста.

Все форматирование далее будет производиться с этими параметрами до их изменения. Для форматирования нажмите F0 и 1, находясь в любой позиции абзаца.

Редактор дает возможность разместить заголовок

симметрично относительно центра текста. Для этого наберите F0 и 2, находясь в любой позиции центрируемой строки.

МАКРОСЫ:

Для многократного выполнения произвольной последовательности действий можно использовать макросы.

Установка F2-макроса:

F2"{"<набор_действий>F2"}"

На экране появляется запрос:
<<< Кнопка для макроса:

После этого выбранная вами кнопка с литерой послужит для вызова макроса.

Вызов F2-макроса:
F2 <кнопка>

Установка и вызов F1-макроса производятся аналогично. Некоторые F1-макросы являются стандартными для редактора и устанавливаются при вызове редактора автоматически.

Сохранение макроса:

Чтобы установленные макросы сохранились после выхода из редактора, войдите в монитор установки параметров (SETUP-монитор) и воспользуйтесь командой SAVE. Макросы, установленные в редакторе, в том числе стандартные, запишутся в файл EX.SETUP. При дальнейших вызовах редактора на текущей директории макросы будут считаны из этого файла, поскольку редактор ищет файл макросов по установленному пути.

SETUP-МОНИТОР:

Установка параметров редактирования (SETUP-монитор).

F1 F0 - вход в SETUP-монитор.

В SETUP-мониторе:

Для передвижения по меню используйте стрелки. После изменения числовых параметров нажмите CR. Переключение on/off, выбор альтернативы осуществляются с помощью клавиши CR.
Для выхода из меню нажмите Esc.

LINES	- установить число строк на экране.
COLUMNS	- число колонок в тексте на экране. Если указанное количество символов не влезет на физический экран, продолжение строки помещается в следующую строку.
UP AREA	- с какой строчки начинать сдвиг страницы вниз.
DW AREA	- с какой строчки начинать сдвиг страницы вверх.
PAGE JUMP	- на сколько строк вниз/вверх переводит Page Dw/Page Up.
INS	- включить/отключить режим вставки.
BELL	- включить/отключить звуковой сигнал.
INFOLINE	- включить или отключить показ информационной строки.
PARAGRAPH	- отступ красной строки от левой границы текста (отрицательное число, если отступ влево).
LEFT MRG	- левая граница отформатированного текста.
RIGHT MRG	- правая граница отформатированного текста.
SAVE	- сохранить установленные параметры при выходе из редактора в файле .ex.$$$. При запуске редактора на текущей директории параметры будут взяты из этого файла.
RECALL	- установить первоначальные параметры (из файла .ex.$$$).
ORIGINAL	- установить параметры из файла bin/.ex.$$$.
INSTALL	- установка фильтра.

УСТАНОВКА ФИЛЬТРА:

Фильтр - программа, работающая с редактируемым текстом с помощью переменных процедурного типа из библиотеки myEditor. С помощью фильтров можно выполнять различные операции над редактируемым текстом.

Фильтр может быть запущен в shell'е редактора, как любая другая задача, а может быть установлен с помощью команды INSTALL в SetUp-мониторе следующим образом: после выбора альтернативы INSTALL в меню укажите произвольную удобную для вас литерную клавишу. После появления приглашения введите имя кодофайла, содержащего головной модуль устанавливаемого фильтра, например:

{+myEditor} abc

- устанавливает фильтр-словарь в режиме перевода;

{+myEditor} abc -f

- устанавливает фильтр-словарь в режиме пополнения словаря.

Фильтр установлен. В этом можно убедиться, запустив его: F0 <кнопка>.

find
FILE find & PROCESSING

Находит файлы по указанному образцу и применяет к ним указанное действие.

find {суперобразец} [cmd=действие] [-hqav] [время]

Строка "действие" должна включать символы %s. После того, как утилита найдет файл, сопоставившийся с образцом, символы %s заменяются на имя файла и осуществляется попытка исполнить получившуюся строку средствами библиотеки Shell.

Заметим, что если строка "действие" содержит пробелы, ее нужно заключить в кавычки.

Если встретился файл, над которым почему-либо не удалось произвести указанное действие, утилита завершает работу, игнорируя остальные файлы.

КЛЮЧИ:

h - подсказка;
q - не запрашивать подтверждение на исполнение командной строки;
f - не искать обычные файлы;
d - не искать файлы-директории;
y - не искать системные файлы;
z - не искать файлы-устройства (узлы);
v - не сообщать имена обрабатываемых файлов.

after=время
before=время
искать файлы, модифицированные после (или до) указанного времени (см. раздел 4.2 о параметрах времени).

ПРИМЕРЫ:

find /usr//*.ref cmd="rm %s -q" -q

Удалить все реффайлы на всех поддиректориях директории /usr, не запрашивая подтверждения ни на исполнение строки, ни на удаление.

fschk
fILE sYSTEM chECK

Проверка файловой системы на указанном диске.
Для осмысленного использования утилиты рекомендуется предварительно ознакомиться с разделом "Диск в файловой подсистеме" в Руководстве по ОС Excelsior.

fschk имя_файла_устройства [файл_сообщений]

Проверяет файловую систему на указанном диске.
Если указан файл сообщений, полученная информация об ошибках не только выдается на экран, но и заносится в этот файл. На экран выдаются сообщения о произведенных исправлениях, но исправления заносятся на диск только с ключом +W:

fschk имя_файла_устройства [файл_сообщений] +W

Не только проверяет, но и корректирует файловую систему на указанном диске. Если указан файл сообщений, полученная информация об ошибках заносится в него.

fschk -h

Выдает короткую подсказку.

ПРИМЕРЫ:

fschk /dev/wd0
Проверяет файловую систему диске wd0.

fschk /dev/dk0 +W
Проверяет и исправляет файловую систему диске wd0.

fschk /dev/dk0 /lost/errors"
Проверяет файловую систему диске wd0, занося обнаруженные ошибки в файл /lost/errors.

fsdb

fILE sYSTEM dEbUGGER

Утилита исправления файловой системы, проверки и исправления диска. Не рекомендуется пользоваться утилитой, не ознакомившись предварительно с разделом "Диск в файловой подсистеме" в Руководстве по ОС Excelsior.

fsdb имя_файла_устройства
Проверяет файловую систему на указанном диске.

fsdb имя_файла_устройства +W
Проверяет файловую систему на указанном диске и исправляет обнаруженные ошибки.

fsdb имя_файла_устройства -v
Проверяет диск на наличие испорченных блоков.

fsdb имя_файла_устройства -v +W
Проверяет диск на наличие испорченных блоков с закрытием таковых, если будут найдены. Закрытие означает включение плохого блока в файл номер 2 (при инициализации диска этот файл создается под именем "/BAD.BLOCKS"). Если плохих блоков на диске становится слишком много, то диск рекомендуется "мягко" отформатировать (см. "dsu").

fsdb [имя_файла_устройства] -m

Интерактивная отладка файловой системы на диске. Поскольку такая отладка содержит средства, позволяющие вскрыть защиту файлов на диске, утилита позволяет отлаживать диск только таким пользователям, которым разрешена запись в указанный файл-устройство (узел).
Если устройство указано при запуске, работа ведется с ним. В процессе работы можно указать или сменить обрабатываемое устройство.
Каждый этап работы в мониторе fsdb снабжен подсказками, поэтому дальнейшее описание утилиты опускается.

grep
gLOBAL repLACE PATTERN UTILITY

Поиск и замена образца в указанных файлах.

Поиск:

grep {дерево} p=образец [-qdoxmbap] [times] [sep=number]

Ищет вхождения образца в указанных файлах. Перед началом поиска в каждом файле выдает запрос на подтверждение поиска. Если образец найден в каком-либо файле, на экран выдается имя файла, номера строк и приводятся сами строки, в которых найден образец.

Замена:

grep {дерево} {дерево-получатель} образцы [-qdoxmbaprc]
[times] [sep=number]

образцы := p=что_менять r=на_что_менять

Находит в предложенных файлах образец и заменяет на указанный. Полученный измененный файл записывается в получатель. Исходный файл и получатель могут совпадать.

Строка times обозначает набор числовых ключей after=, before=, cafter=, cbefore=, имеющих стандартный смысл, описанный в разделе 4.2 о параметрах времени.

КЛЮЧИ:

-h - подсказка;
-q - не запрашивать подтверждение на обработку файла;
-d - с созданием недостающих директорий для получателя;
-o - обрабатывает только те файлы, которые уже есть на директории-получателе;
-x - обрабатывает только файлы, отсутствующие на директории-получателе;
-m - обрабатывать только исходные файлы, время последней модификации которых больше, чем у файлов-получателей;
-b - рассматривать файл как нетекстовый;
-a - найти все вхождения образца (только в режиме поиска);
-p - подсчитать количество вхождений образца;
-r - не спрашивать подтверждения на замену образца;
-c - подсчитать количество замен образца;

sep=<число> - задает код символа, считающегося разделителем строк, по умолчанию - ASCII.NL (036c) (для правильного подсчета строк в нестандартных файлах).
after=, before=, cafter=, cbefore= - ключи времени,

имеющие стандартный смысл (см. 4.2).

ЗАМЕЧАНИЯ

При перенаправлении стандартного вывода в файл утилита не выдает запросов, как при запуске с ключами -qr.

При работе в открепленном режиме утилита не выдает запросов, как при запуске с ключами -qr.

Если при запуске указан один образец для выбора файлов, то образец для получателя может быть опущен; в этом случае получателем выбирается текущая директория.

ОТВЕТЫ НА ЗАПРОСЫ:

Запросы выдаются в двух случаях: при выборе файла и для подтверждения замены.

y - (Yes) да;
n - (No) нет;
q - (Quit) прекращение работы;
^X - эквивалентно "y", но утилита переходит в откреплeнный режим.

При выборе файла для сканирования:

a - (all) все файлы на текущей директории;
A - (All) все оставшиеся файлы;
i - (skIp) пропустить оставшиеся файлы на текущей директории.

Подтверждение замены:

a - (all) все образцы в текущем файле;
A - (All) все образцы в оставшихся файлах;
i - (skIp) пропустить все образцы в текущем файле.

ПРИМЕРЫ:

```
grep //ex* p=pull -q
```

Выдается информация о вхождении образца pull в файлы вида ex* на всех поддиректриях файлового дерева:

```
ex.m
line   12:   lpull   = CHAR( ORD('W') MOD 32);
exScreen.m
line 1125: E pull;
exMain.m
line   16:   , pullleft, pullright, maxcol, size,
exTex.m
line   27: PROCEDURE pull_right(c1: INTEGER);
```

```
grep //ex*$1 my_ex/ex$1 p=pull r=out -q
```

Во всех файлах вида ex* образец pull заменяется на out и получившиеся измененные файлы записываются на директорию my_ex под исходным именем.

```
grep p=PROCEDURE *.[md] -aq
```

Найти и показать все вхождения образца "PROCEDURE" в файлах *.[md] на текущей директории.

```
grep p=PROC r=proc /usr/mydir//*  tmp/$@/ -r
```

Найти во всех файлах поддерева "/usr/mydir//*" образец "PROC" и заменить его, не спрашивая подтверждения на замену, на образец "proc".

```
hi
TASK hiSTORY
```

Выдает историю задачи.

```
hi [<номер_задачи>]
```

Если номер задачи не указан, то выдает историю последней аварийно завершившейся задачи. Номер интересующей вас задачи можно узнать с помощью команды shell "ps".

Утилита сообщает номер процесса, причину прекращения, последовательность вызова процедур с указанием имен модулей, которым принадлежат процедуры, номеров строк и позиций в строке (нумерация с нуля), из которых был произведен вызов, повлекший неудачу.

ПРИМЕРЫ:

```
hi
Выдается на экран:

#процесс: 1A383h  причина: процесс прерван (^C) [0050h]

cmd.            | MakeNapkin:54.6 <- one:67.4 <-
Files.          | LsOne:202.12 <-
FsDnode.        | Ls0:75.6 <-
Files.          | LsDir:207.8 <-
cmd.            | BEGIN:82.11 <-
Tasks.          | TaskStarter:318.4 <-
```

В этом примере показана история процесса, прерванного CTRL_C, посланным с терминала. Процесс прервался в момент выполнения оператора в 54-й строке процедуры -MakeNapkin- из модуля -cmd-, которая, в свою очередь, была вызвана в 67 строке процедуры -one-.

В случаях, когда утилите неизвестно имя процедуры, указывается ее номер и предваряется знаком '?'. Если утилите неизвестен номер строки, выдается ее позиция в коде относительно начала процедуры.

human

uSER manAGER

Утилита обслуживает систему пользователей. Утилита адресуется администратору системы.

human -e

Редактирует файл паролей.

Пользователь может изменить имя, под которым он входит в систему, свой пароль и свою рабочую директорию. Для этого необходимо войти в утилиту под своим именем и своим паролем. Для выхода из утилиты без записи нажмите Esc.

Изменить соответствующие параметры всех пользователей может только привилегированный пользователь (администратор системы). Для этого необходимо войти в утилиту под именем "su" и правильно указать пароль. Администратор может удалить или ввести в систему новых пользователей, изменить их привилегии или принадлежность группам, а также указать, какую задачу запустить пользователю в качестве shell и какая у него рабочая директория (директория, на которой лежит его командный файл profile.@, см. в Руководстве по ОС Excelsior раздел shell). Каким образом произвести все эти действия, ясно из системы меню, предлагаемых утилитой.

Вся перечисленная информация хранится в обусловленном файле PASS.WD, который создается копированием и последующим редактированием при установке системы на носителе. Он должен быть расположен на директории /usr/etc.

human -i

Загружает файл паролей (для работы с паролем после правки или создания без перезагрузки).

КЛЮЧИ:

-h - подсказка;
-e - редактирование файла паролей;
-i - загрузка файла паролей.

ln

LiNk files

Привязывание файла под новым именем.

ln {дерево} дерево-получатель [-qoxy] [+H|-H] [times]

Присваивает файлам на указанном дереве еще одно имя, определяемое деревом-получателем. При этом физического копирования файла не происходит, лишь число связей файла увеличивается на одну, отсюда термин "привязать файл". Все привязанные к файлу имена совершенно равноправны.

ln дерево [-qoxy] [+H|-H] [times]

Если дерево-получатель не указано, файлы привязываются на текущую директорию.

Строка times обозначает набор числовых ключей after=, before=, cafter=, cbefore=, имеющих стандартный смысл, описанный в разделе 4.2 о параметрах времени.

ЗАМЕЧАНИЕ:

Нельзя привязать к директории файл, физически размещенный на ином, чем директория, диске. В этом случае возбуждается ошибка "ссылка между устройствами".

КЛЮЧИ:

-h - подсказка;
-q - без запроса на подтверждение;
-x - привязать все файлы из исходного дерева, кроме уже существующих на получателе;
-o - привязать только те файлы из исходного дерева, которые уже существуют на получателе;
-y - для системных файлов;
+H - пометить файлы как скрытые;
-H - пометить файлы как видимые.

ПРИМЕРЫ:

```
ln /mnt/*.cod /mnt/bin/
ln /mnt/*.sym /mnt/sym/
```

Привязывает кодофайлы и симфайлы, расположенные на корневой директории гибкого диска, смонтированного на директории /mnt, на директории bin и sym соответственно.

ls
lIsT FILE NAMES AND ATTRIBUTES

Информация о файлах и их атрибутах.

ls {образец} [-ключи]

Выдает на экран имена файлов, удачно сопоставившихся с образцом. Информация о скрытых (hidden) файлах показывается только при запуске с ключом -a. Запуск ls без аргументов выводит на экран имена файлов текущей директории.

КЛЮЧИ:

Определение групп файлов:

-a показ всех файлов (включая скрытые);
-v показ только скрытых файлов;
-F показ только обычных файлов (не директорий);
-D показ только директорий.

Выдача атрибутов:

-l развернутая информация о файле, включающая размер файла, биты защиты файла и время модификации.
-i системные номера файлов;
-u владельцы файлов;
-y вывод времени в полном формате, содержащем год и секунды.
-c использование время создания вместо времени модификации.

Управление порядком имен:

-s в порядке хранения файлов в директории (без сортировки);
-t сортировка по времени модификации;
-r в обратном порядке;

По умолчанию ls выдает имена файлов в алфавитном порядке.

Управление представлением:

-1 выдача имен в одну колонку;
-f выдача полных имен в одну колонку;
-n выдача имен скрытых файлов без выделения; по умолчанию имена скрытых файлов выделяются цветом.

ПРИМЕРЫ:

ls
выдается список имен всех файлов на текущей директории.

ls -l

Имена файлов на текущей директории выдаются в "длинном" формате: указывается маска защиты, длина файла, дата и время последней модификации и, наконец, имя файла:

```
--rwxr-x---  1        968  Jan 18  13:08   a.m
--rwxr-x---  1      4,145  Dec 26   1989   gloss
--rwxr-x---  1        725  Jul 10   1989   his.t
--rwxr-x---  1    128,626  Jan 18  14:59   iV.doc
--rwxr-x---  1        000  Jan 18  14:57   t
--rwxr-x---  1      8,757  Jan 10  18:06   tt
--rwxr-x---  1     65,602  Oct  7   1989   usr1.doc
--rwxr-x---  1     63,067  Oct 10   1989   usr2.doc
428,451 byte in 10 files on ./  (2 invisible files)
```

ls -u

Выдает ту же информацию, но с указанием статуса и имени владельца.

```
--rwxr-x---  1 host flm         968  Jan 18  13:08   a.m
--rwxr-x---  1 host flm       4,145  Dec 26   1989   gloss
--rwxr-x---  1 host flm         725  Jul 10   1989   his.t
--rwxr-x---  1 host flm     128,626  Jan 18  14:59   iV.doc
--rwxr-x---  1 root sys         558  Jan 18  14:59   t
--rwxr-x---  1 host flm       8,757  Jan 10  18:06   tt
--rwxr-x---  1 host flm      65,602  Oct  7   1989   usr1.doc
--rwxr-x---  1 host flm      63,067  Oct 10   1989   usr2.doc
429,009 bytes in 10 files on ./  (2 invisible files)
```

ls -li

Добавились имена файлов в таблице инодов:

```
--rwxr-x---  1        968  Jan 18  13:08  i1318    a.m
--rwxr-x---  1      4,145  Dec 26   1989  i2808    gloss
--rwxr-x---  1        725  Jul 10   1989  i2291    his.t
--rwxr-x---  1    128,626  Jan 18  14:59  i1344    iV.doc
--rwxr-x---  1      1,217  Jan 18  15:00  i1316    t
--rwxr-x---  1      8,757  Jan 10  18:06  i3131    tt
--rwxr-x---  1     65,602  Oct  7   1989  i2456    usr1.doc
--rwxr-x---  1     63,067  Oct 10   1989  i2928    usr2.doc
429,668 bytes in 10 files on ./  (2 invisible files)
```

```
ls -liy
```

Теперь время выдается в полном формате:

```
--rwxr-x---  1        968  Jan 18 90  13:08.42 i1318  a.m
--rwxr-x---  1      4,145  Dec 26 89  12:39.09 i2808  gloss
--rwxr-x---  1        725  Jul 10 89  20:11.59 i2291  his.t
--rwxr-x---  1    128,626  Jan 18 90  14:59.10 i1344  iV.doc
--rwxr-x---  1      1,856  Jan 18 90  15:00.19 i1316  t
--rwxr-x---  1      8,757  Jan 10 90  18:06.22 i3131  tt
--rwxr-x---  1     65,602  Oct  7 89  23:09.10 i2456  usr1.doc
--rwxr-x---  1     63,067  Oct 10 89  13:55.34 i2928  usr2.doc
430,307 bytes in 10 files on ./  (2 invisible files)
```

```
ls -la
```

Добавилась информация о "невидимых" файлах, например, о кодофайлах:

```
--rwxr-x---  1        896  Jan 16  20:15   ../
--rwxr-x---  1        544  Jan 18  13:08   a.cod
--rwxr-x---  1        968  Jan 18  13:08   a.m
--rwxr-x---  1        350  Jan 18  13:08   a.ref
--rwxr-x---  1      4,145  Dec 26   1989   gloss
--rwxr-x---  1        725  Jul 10   1989   his.t
--rwxr-x---  1    128,626  Jan 18  14:59   iV.doc
--rwxr-x---  1      3,335  Jan 18  15:01   t
--rwxr-x---  1      8,757  Jan 10  18:06   tt
--rwxr-x---  1     65,602  Oct  7   1989   usr1.doc
--rwxr-x---  1     63,067  Oct 10   1989   usr2.doc
432,680 bytes in 12 files on ./
```

Можно использовать утилиту ls и для поиска файлов, например:

```
ls //my_file
```

Утилита будет разыскивать файл на всех директориях файлового дерева. Если файл найдется, на экран будет выдано его полное имя (с указанием маршрута до этого файла от корня):

```
/usr/my_dir/wrk/cmd/my_file.
```

mkdir
mAkE dirECTORY

Создает новую директорию.

mkdir {имя_директории}

ПРИМЕРЫ:

mkdir t1 t2 t3

создать на текущей директории поддиректории t1, t2, t3.

mkfs
mAkE fILE sYSTEM

Разметка диска в соответствии со стандартом ОС Excelsior (см. в Руководстве раздел о диске в файловой подсистеме). Производится на отформатированном носителе перед монтированием и записью информации. Если на диске уже содержалась информация, она будет утрачена.

mkfs {имя_блочного_устройства} [метка]

Метка - это символическое имя носителя, которое нигде не используется и может быть опущено. Некоторым пользователям нравится помечать носители разными именами, чтобы не путать.

ПРИМЕРЫ:

mkfs /dev/fd0 ws1

Разметка флоппи-диска с меткой ws1.

mknode
mAkE DEVICE node

Создает специальный файл - узел устройства.

mknode <маршрут> <имя_устройства>
[<защита>|"pro="<битсет защиты>]

защита =:: { ("o"|"g"|"a") ("+"|"-") ("r"|"w"|"x")}

Создает на указанной директории специальный файл, устанавливающий связь с указанным устройством. После этого к устройству можно обращаться как к файлу по полному имени этого файла. Специальный файл создается с указанной степенью защиты (см., например, утилиту cmode).

Обратите внимание: после создания специального файла изменить степень его защиты с помощью утилиты chmode нельзя.

КЛЮЧИ:

h - подсказка.

ПРИМЕРЫ:

Приведем полный перечень действий для подключения какого-нибудь устройства, например - принтера. Предположим, что принтер подключен к последовательному выводу с номером 3, а коды драйверов последовательного вывода и принтера расположены на директории /ipr.

1) Устанавливается драйвер последовательного вывода на порт 3. Отныне последовательный порт является устройством sr:

/ipr/SC5IO sr 3

2) На директории /dev созается специальный файл sr3, связанный с устройством sr:

mknode /dev/sr3 sr pro=--++-++-++-

3) Устанавливается драйвер принтера. Драйверу указывается устройство (отныне lp0 - печатающее устройство) и файл, в который будет осуществляться вывод:

/ipr/LPcm6329 lp0 /dev/sr3

4) На директории /dev созается специальный файл lp0, связанный с устройством lp0:

mknode /dev/lp0 lp0 pro=--++-++-++-

5) В окружение заносится параметр, определяющий принтер:

```
LP=/dev/lp0
```

Теперь можно печатать (см. утилиту ptp).

mou

mouNT DEVICE

Монтирование структурированных устройств (дисков) в файловую систему.

mou имя_директории имя_файла_устройства [-w12]

Монтирует устройство на указанную директорию. После окончания работы с диском он должен быть демонтирован, чтобы другие пользователи могли получить доступ к устройству.

mou имя_директории -r [-12]

Демонтирует устройство с указанной директории. Если демонтирование не удалось, используйте ключи -1 и -2, о смысле которых можно узнать в описании процедуры unmount из раздела о файловой подсистеме в книге Руководство по ОС Excelsior.

КЛЮЧИ:

-h - подсказка;
-w - монтирование без права записи на устройство.

ПРИМЕРЫ:

mou /mnt /dev/fd0
Монтирует флоппи-диск на директорию mnt.

mou /mnt -r
Демонтирует флоппи-диск с директории mnt.

msex

ms/DOS - exCELSIOR FILE EXCHANGE

Записывает файлы из ОС Excelsior в стандарте MS/DOS и наоборот.

msex

Вход в монитор утилиты. Работа в мониторе определяется альтернативами предлагаемых меню, имитирующих Norton-comander.

Обратите внимание на то, что при записи файлов в MS/DOS на устройство /dev/fd0 на нем должен быть установлен соответствующий формат (а диск сформатирован в этом формате). Используйте для этого утилиту dsu.

mshell
mODULA shell-INTERPRETER

Специальный интерпретатор командных файлов.

mshell <имя_файла> {<аргументы>}

Утилита выбирает из файла "file_name" строку за строкой и интерпретирует их в соответствии с описанным ниже синтаксисом.

Другой способ запуска - указать имя mshell в качестве параметра окружения SHELL (см. в Руководстве по ОС Excelsior раздел о пользовательской оболочке).

Разделители строк

Командным файлом утилита полагает текстовый файл, строки которого отделяются одна от другой либо символом ASCII.NL (код 036с - стандартный разделитель строк в ОС Excelsior), либо символом ASCII.LF (код 012с - дополнительный).

Текст, интерпретируемый mshell, представляет собой Последовательность Операторов языка, разделенных либо признаком конца строки, либо символом ";", либо, в некоторых случаях, символом ")". При этом комбинация "\;" разделителем не является и обозначает символ ";". В тех случаях, когда символ ")" является разделителем, комбинация "\)" разделителем не является и обозначает ")".

Слова, составляющие операторы, кроме вышеописанных разделителей, отделяются еще пробелами. Количество таких пробелов, если оно больше единицы, роли не играет и может быть использовано для более ясного структурирования текста.

Комментарии

Если в строке первым отличным от пробела символом является "%", то эта строка считается комментарием и не исполняется, например:

```
cp $1 $2 $3  my_dir/
% cp $1 $2 $3  dir/
  %    Конец копирования
```

Первая из приведенных строчка исполнится, а две следующие являются комментарием.

Результат оператора

Исполнение каждого оператора характеризуется его результатом - некоторым целым числом. В настоящей реализации ОС принято, что нулевой результат обозначает нормальное

исполнение оператора, все остальные значения результата свидетельствуют о какой-либо ошибке, произошедшей в ходе исполнения оператора.

Последовательность операторов

```
Посл_Операторов ::= { Оператор }
```

Исполнение последовательности операторов заключается в последовательном исполнении одного оператора за другим сверху вниз и слева направо по тексту.

Результатом последовательности операторов является результат последнего исполненного оператора. Если последовательность операторов пуста, то ее результатом будет 0 (ноль).

Операторы командного языка

```
Оператор ::= Команда_shell
           | Условный_Оператор
           | Цикл_До
           | Цикл_Пока
           | Бесконечный_Цикл
           | Выход_Из_Бесконечного_Цикла
           | Останов
           | "(" Посл_Операторов ")"
```

Команда shell

Синтаксис команд shell описан ранее. Отличие состоит в том, что перед исполнением строка, описывающая команду shell, предварительно обрабатывается следующим образом:

- "\\" заменяется на "\";
- "\$" заменяется на "$" с отключением специального смысла "$";
- "$1","$2".."$9" заменяются на параметры командного файла с номерами 1,2..9 или пустую строку;
- "$?" заменяется строкой, запрошенной с клавиатуры.

Далее обработанная строка передается для исполнения процедуре "Shell.system" (см. библиотеку).

Условный оператор

```
Условный_Оператор ::= "IF" УСЛОВИЕ "THEN"
                                Посл_Операторов
                      [ "ELSE" Посл_Операторов ]
                        "END" .
```

Если "ELSE" опущен, то считается, что соответствующая

последовательность операторов пуста.

Исполнение условного оператора заключается в вычислении условия и выполнении последовательности операторов либо THEN-части (если УСЛОВИЕ=TRUE), либо ELSE-части (если УСЛОВИЕ=FALSE).

Результатом условного оператора является результат исполненной последовательности операторов.

Цикл До

Цикл_До ::= "REPEAT" Посл_Операторов "UNTIL" УСЛОВИЕ .

Посл_Операторов внутри оператора называется телом цикла.

Исполнение этого оператора заключается в следующем:

- исполнение тела цикла;
- вычисление условия;
- если УСЛОВИЕ=TRUE, то переход к шагу 1, иначе - переход к исполнению первого оператора после УСЛОВИЯ.

Результатом является результат последнего исполненного тела цикла.

Цикл Пока

Цикл_Пока ::= "WHILE" УСЛОВИЕ "DO" Посл_Операторов "END" .

Посл_Операторов внутри оператора называется телом цикла.

Исполнение этого оператора заключается в следующем:

- вычисление условия;

- если УСЛОВИЕ=TRUE, то выполнение последовательности операторов и переход к шагу 1, иначе - переход к исполнению первого оператора после "END".

Результатом является результат последней исполненной последовательности операторов, либо 0, если тело цикла не было исполнено ни разу.

Бесконечный Цикл

Бесконечный_Цикл ::= "LOOP" Посл_Операторов "END" .

Исполнение этого оператора заключается в бесконечном повторении последовательности операторов. Прерывает бесконечное повторение специальный оператор выхода из цикла (см.ниже), он располагается непосредственно в тексте и

определяет результат оператора "loop".

Выход Из Бесконечного Цикла

```
Выход_Из_Бесконечного_Цикла ::= "EXIT"
```

Этот оператор прерывает исполнение последовательности операторов, вложенных в самый внутренний из Бесконечных Циклов и переходит к исполнению оператора, следующего за "END" этого цикла. Расположение оператора "EXIT" определяет результат прерванного им цикла "LOOP". Результатом цикла будет результат последовательности операторов, в которую входит "EXIT", но находящихся выше него по тексту; возможно, такая последовательность будет пустой, тогда результат - 0.

Оператор останова

```
Оператор_Останова ::= "HALT" [ "(" number ")" ]
```

Этот оператор прерывает интерпретацию командного файла. Параметр "number" будет проинтерпретирован как результат исполнения командного файла. При его отсутствии он считается равным "0".

Замечание. Символ "(" не является разделителем, поэтому (в отличие, например, от Modula-2) запись "HALT(10)" некорректна, а HALT (10) корректна.

Условие

Условия в условном операторе, Цикле Пока, Цикле До, не являются операторами, и их результат не воздействует на результат командного файла.

Синтаксис:

```
УСЛОВИЕ ::=  "$*"
           | Строка Операция Строка
           | Число  Операция "(" Посл_Операторов ")"
           | "(" Посл_Операторов ")" Операция Число
Операция ::= "=" | "#" | ">" | ">=" | "<=" | "<" .
```

Символ "$*" обозначает ввод ответа с клавиатуры. На экран выдается приглашение "[Y/N]?", а затем считываются символы с клавиатуры до тех пор, пока не встретится символ из набора ("y","Y","n","N"). Первые два при этом обозначают "TRUE", остальные - "FALSE".

В первом случае сравниваются строки символов, два следующих случая - проверка результата последовательности

операторов.

Строки

```
Строка ::=  "'" { Символ } "'"
          |'"' { Символ } '"'
          |"$1","$2".."$9"
          |"$?"
          |"$name" .

Символ  ::=  любой символ ASCII8, отличный от разделителя
             строк и открывающей кавычки.
```

Символы "$1".."$9" обозначают параметры командного файла с номерами соответственно 1..9, если их нет - то пустые строки.

Символ "$?" обозначает ввод строки с клавиатуры. Перед вводом на экран выдается приглашение "?".

Символ "$name" (где "name" - непустая последовательность символов, отличных от пробела и разделителей строк) обозначает параметр окружения с именем "name" (если параметр с таким именем в окружении отсутствует, то пустую строку).

Числа

```
Число ::= целое беззнаковое число в стандарте ОС | "$#" .
```

Символ "$#" обозначает ввод числа с клавиатуры. Для ввода числа на экране выдается приглашение "#", после чего можно вводить строки до тех пор, пока введенная строка не сопоставится с числом. На всякую строку, не сопоставившуюся с числом, выдается сообщение "illegal number".

Эхо

Эхо считается включенным, если в окружении "mshell" есть параметр с именем "ECHO" и его значение не равно "0". В противном случае эхо считается выключенным.

Если эхо включено, то все комментарии и каждая команда перед исполнением печатаются на экране.

Способ проверки корректности синтаксиса

Стоит заметить, что всякий командный файл просматривается и анализируется на корректность синтаксиса либо до конца, либо до первой синтаксической ошибки. Это свойство можно использовать при отладке командных файлов: достаточно в начале текста поставить команду "HALT", и командный файл не будет исполняться, а синтаксис его будет проверен. Если в нем присутствует ошибка, то в сообщении о ней указывается номер

строки и позиция в ней (считается, что строки нумеруются с 1).

Описание синтаксиса командного файла

```
Командный_Файл ::= Посл_Операторов .
Посл_Операторов ::= { Оператор } .
Оператор ::= Команда_shell'а
           | Условный_Оператор
           | Цикл_До
           | Цикл_Пока
           | Бесконечный_Цикл
           | Выход_Из_Бесконечного_Цикла
           | Останов
           | "(" Посл_Операторов ")"
Условный_Оператор ::= "IF" УСЛОВИЕ "THEN"
                            Посл_Операторов
                    [ "ELSE" Посл_Операторов ]
                      "END" .
Цикл_До ::= "REPEAT" Посл_Операторов "UNTIL" УСЛОВИЕ .
Цикл_Пока ::= "WHILE" УСЛОВИЕ "DO" Посл_Операторов "END" .
Бесконечный_Цикл ::= "LOOP" Посл_Операторов "END" .
Выход_Из_Бесконечного_Цикла ::= "EXIT"
Оператор_Останова ::= "HALT" [ "(" number ")" ]
УСЛОВИЕ ::=  "$*"
           | Строка Операция Строка
           | Число  Операция "(" Посл_Операторов ")"
           | "(" Посл_Операторов ")" Операция Число
Операция ::= "=" | "#" | ">" | ">=" | "<=" | "<" .
Строка ::=  "'" { Символ } "'"
           |'"' { Символ } '"'
           |"$1","$2".."$9"
           |"$?"
           |"$name" .
Символ  ::=  любой символ ASCII8, отличный от разделителя
             строк и открывающей кавычки .
Число ::= Десятичная_Цифра { Шестнадцатеричная_Цифра } [ Формат ]
.
Десятичная_Цифра ::= "0" | "1" | "2" | "3" | "4"
                   | "5" | "6" | "7" | "8" | "9" .
Шестнадцатеричная_Цифра ::= Десятичная_Цифра
                          | "A" | "B" | "C" | "D" | "E" | "F" .
Формат ::= "b" | "h" .
```

mv

FILES mOvE

Привязывает файлы в новое место файлового дерева и, возможно, под новыми именами. При этом файлы никуда не перемещаются физически.

mv [-qoxdy] [+H|-H] { дерево-источник }
дерево-получатель [время]

Отвязывает файлы с дерева-источника и привязывает к дереву-получателю.

mv [-qoxdy] [+H|-H] { дерево-источник } [время]

В качестве дерева-получателя используется текущая директория.

Строка times обозначает набор числовых ключей after=, before=, cafter=, cbefore=, имеющих стандартный смысл, описанный в разделе 4.2 о параметрах времени.

Ответы на запросы утилиты стандартные и описаны в п.3.

ЗАМЕЧАНИЕ:

Нельзя привязать к директории файл, физически размещенный на ином, чем директория, диске. В этом случае возбуждается ошибка "ссылка между устройствами".

КЛЮЧИ:

-h - подсказка;
-q - без запроса на подтверждение;
-x - привязать все файлы из исходного дерева, кроме уже существующих на получателе;
-o - привязать только те файлы из исходного дерева, которые уже существуют на получателе;
-y - для системных файлов;
-d - для директорий;
+H - пометить файлы как скрытые;
-H - пометить файлы как видимые.

ПРИМЕРЫ:

mv файл1 файл2
переименовать файл1 в файл2;

mv {<имя_файла>} другая/
перемещение файла 'имя_файла' с текущей директории в директорию 'другая'.

```
mx {<имя_файла>} [-ключи]
mODULA-ExTENDED COMPILER
```

Утилита запускает Модула-компилятор.

В качестве параметра утилите передаются имена файлов, содержащих определяющие, реализующие или программные модули задач. Имя файла должно иметь расширитель соответственно .d или .m. Если имя файла указано без расширителя, по умолчанию добавляется расширитель .m.

В результате компиляции определяющего модуля получается симфайл, реализующего модуля - реффайл и кодофайл одновременно, программного - ref-файл и кодофайл одновременно.

Имена симфайлов, реффайлов и кодофайлов получаются из имени модуля (не из имени файла) добавлением расширителей '.sym', '.ref' и '.cod' соответственно.

Подробное описание Модула-компилятора приведено в книге "ОС Excelsior для всех".

КЛЮЧИ:

Ключи компилятора могут включать и выключать перечисленные ниже действия. Для включения действия используется прописная буква, для выключения - строчная. В скобках указано текущее состояние компилятора по умолчанию.

-a (off) - печать всех ошибок в строке (по умолчанию показывается только первая ошибка в строке);
-f (off) - аварийное прекращение по первой ошибке;
-i (on) - инициализация указателей NIL'ом;
-n (off) - проверка на NIL для указателей и DYNARR'ов;
-r (on) - проверка границ отрезков;
-t (on) - проверка границ массивов;
e= прекращение компиляции после указанного числа ошибок (по умолчанию 8).

РЕЖИМЫ КОМПИЛЯЦИИ

Текст Модула-программы может содержать управляющие последовательности, которые мы будем называть прагматами. Прагматы служат для изменения режима компиляции внутри текста программы.

```
прагмат   = "(*$" директива "*)".
директива = { опция "+" | "-"  | "!" | "<" | ">" }
опция     = 'A'|'F'|'I'|'N'|'R'|'T'
```

Каждая опция означает действие, которое может быть включено (отключено) управляющей последовательностью. Смысл опций и их начальное состояние приводится в таблице.

N	Опция	Значение	По умолчанию
1	A	показ всех ошибок	off
2	F	аварийное окончание компиляции при первой встреченной ошибке	off
3	I	инициализация переменных типа указатель значением NIL; типа динмассив - парой (NIL,-1)	on
4	N	проверка на NIL для указателей и динмассивов. При включении автоматически включает опцию I	off
5	R	контроль выхода за границы отрезка	on
6	T	контроль индексов массивов	on

Остальные директивы определяют операции над стеком режимов трансляции.

1) < - запомнить текущее состояние опций в стеке;
2) > - восстановить последнее состояние опций из стека;
3) ! - счеркнуть весь стек режимов и перейти в исходное состояние.

ПРИМЕРЫ:

```
mx exf.m
```

Если в процессе компиляции не обнаружено ошибок, на экран выдается сообщение:

```
Modula X v0.3 /28-Sep-89/  "t.m"
lines: 1 time: 01cp+00io "t.cod" 1 words
```

В результате компиляции на текущей директории появятся файлы t.ref и t.cod.

Если в тексте модуля были обнаружены ошибки, о них выдается сообщение:

```
Ожидался символ ';'
 104:   PROCEDURE Next(i,j: INTEGER$,A: ARRAY OF CHAR);
Число ошибок: 1
```

Выдается сообщение о характере ошибки, приводится номер строки и сама строка, в которой допущена ошибка, а также помечается предположительно место в строке, где надо искать ошибку (символом '$'). Обратите внимание: символ "$" выставляется ПОСЛЕ лексемы, в которой обнаружена ошибка.

pkr
pACkING & ArCHIVING utility

Архивирует файлы в упакованном виде. Файл архива всегда записывается на текущую директорию с добавлением расширения 'pkr' к указанному имени файла архива. Если на текущей директории уже имеется файл с указанным именем и расширением 'pkr', то происходит его распаковка.

pkr имя_файла_архива [-ключи]{[@]файловое_дерево} [время]

@имя_файла означает, что описание файлового дерева должно быть взято из указанного текстового файла. В этом файле можно перечислить необходимые ветки файлового дерева, по одной на каждой строке, например:

```
/sys/util/*
/lib/def/*
....
```

Если имя вашего файла начинается с символа '@', предварите его тем же символом: '@@'.

В качестве файла-архива может быть указан файл-устройство (узел), например:

pkr /dev/fd0 /sys/util/major/*

КЛЮЧИ:

h - подсказка;
l - выдает список существующих архивов;
m - архивирование при ограничениях оперативной памяти;
T - упакованное дерево файлов будет восстановлено с сохранением структуры дерева;
r - базировать дерево относительно текущей (а не корневой) директории;
q - не запрашивать подтверждения на упаковку/распаковку.

О параметре 'время' читайте в разделе о параметрах файловых утилит.

pm
pROJECT mANAGER

Утилита предназначена для поддержки разработки программ, состоящих из многих модулей.

pm {дерево}

Показывает имена файлов, сопоставившихся с образцом. Среди этих файлов могут быть как программные (с расширителями .d и .m), так и обычные файлы. Совокупность программных модулей назовем проектом. Над проектом определены две основные операции: генерация проекта и перекомпиляция. Эти операции выполняются над выделенным пользователем множеством модулей.

При генерации будут скомпилированы выделенные модули и все модули проекта, используемые данными, в том числе косвенно.

При перекомпиляции будут скомпилированы выделенные модули модули проекта, которые от них зависят, в том числе косвенно.

pm

Если образец не указан, в качестве образца используется *.? .

pm {дерево} +E

Проект составляется в соответствии с созданным ранее и записанным в файл графом зависимости.

КОМАНДЫ:

В мониторе определены следующие команды:

```
F1      - подсказка;
e       - редактировать файл;
r       - заново составить список файлов;
q       - выход из утилиты;
R,SPACE - выделить/убрать выделение файла;
SC      - убрать все выделения;
F2 r    - прочитать файл с графом зависимости;
F2 w    - записать файл с графом зависимости;
F2 g    - создать граф зависимости;
F0 r    - перекомпиляция выделенных задач;
F0 g    - генерация выделенных задач;
F0 s    - сортировка в соответствии с графом зависимости;
F0 o    - установить опции компилятора;
F0 PgDw - выход в shell (ESC - возврат в монитор pm);
F5 r    - записать командный файл перекомпиляции;
F5 g    - записать командный файл генерации.
```

ptp

pRODUCER OF tOILET-pAPER

Предназначается для вывода на печать.

ptp {<образец>} [ключи]

Выводит на принтер файлы, сопоставившиеся с образцом. Утилита работает с любым принтером, обслуживаемым ОС Excelsior. Это достигается с помощью библиотеки Printer, у которой разным типам принтеров соответствуют разные реализующие модули. Во избежание путаницы реализующие модули для каждого принтера лежат в файлах с разными названиями.

Утилита позволяет использовать малофункциональные принтеры, не ограничивая в то же время набор возможностей, предоставляемых высококачественной печатающей техникой.

КЛЮЧИ:

-q - не запрашивать подтверждения на печать;
-h - подсказака;
+z - позволяет напечатать указанные несколько файлов, как один, то есть со сквозной нумерацией страниц и без инициализации параметров печати перед напечатанием каждого следующего файла.
+o - печать с одной стороны листа;
+r - (roll) печать на рулонной бумаге;
-H(T) - не печатать первую (последнюю) служебные строки;
+t - вывод на терминал;
+i - рассматривать @-последовательности в тексте как команды;

Для числовых ключей в квадратных скобках указаны их значения по умолчанию.

LP=n - печатать на устройстве с номером n, если в системе несколько устройств;
lines=n [59] - размер страницы в строках;
mo=n [5] - левая граница для нечетных страниц;
ma=n [8] - левая граница для всех страниц;
page=n [1] - начать нумерацию страниц с номера n;
from=n [1] - начать печать со страницы номер n;
font=n [0] - установить шрифт с номером n;
width=n [64] - установить ширину листа в n символов;
int=n [2] - установить интервал между строками в n полустрок;
gap=n [1] - (с ключом +r) промужуток в строках между страницами при печати на рулонной бумаге;
del=n - (с ключом -q) устанавливает время задержки (в секундах) перед печатью очередного листа);
upma=n [0] - размер верхнего поля (в строках).

При запросе подтверждения на печать страницы возможны ответы:

y - печатать страницу;
n - не печатать страницу;
q - прекратить печать;
a - напечатать предыдущую страницу еще раз.

Если нажать при запросе клавишу 'CR', произойдет захват вставленного диста бумаги; клавишу 'Up' (стрелка вверх) - бумага подастся на 1 строку вперед; при нажатии клавиши 'Dw' (стрелка вниз) - назад.

ПРИМЕРЫ:

```
ptp filename page=12 from=14 lines=64 +t
```

Печатает файл filename с размером страницы 64 строки, первая страница текста будет иметь номер 12, а печать начнется со страницы номер 14 (третьей страницы файла). Вывод будет осуществляться на терминал (+t).

УПРАВЛЯЮЩИЕ СТРОКИ:

Входной текст может содержать управляющие строки, которые опознаются по первому слову в строке и в напечатанном тексте опускаются.

.PAGE
- начать новую страницу.

.HEAD n <формат>
- установить новый вид заголовка.

.TAIL n <формат>

- установить новую нижнюю строку. Здесь n принимает значения 0 и 1. n=1 означает, что указанный заголовок устанавливается для всех нечетных страниц, n=0 - для всех четных. Это используется в случае печати на обеих сторонах листа. При печати с одной стороны листа заголовки будут совпадать в части <формат>.

<формат> совпадает с форматом стандартного вывода, принятого в ОС Excelsior (см. раздел о стандартном выводе в Руководстве по ОС), за исключением следующего:

- все изображаемые в строке числа рассматриваются как номер страницы или дата, поэтому отсутствуют базы -f-, -e-, -g-, применяемые для изображения вещественных чисел;

- отсутствует модификатор "точность";

- базы d,h,i используется для обозначения номера страницы. Например, %6d в строке формата будет означать номер страницы, напечатанный справа в поле ширины 6;

- база x,X используется для обозначения пробелов. Например, %6x в строке формата будет означать вставку шести пробелов;

- база s используется для изображения имени печатаемого файла, например, строка

```
"                       %s"
```

отобразится при печати файла text.t как

```
"                       text.t".
```

%K - печать строковой константы "(c) KRONOS";

%D - печать текущей даты, например:

```
"%K                     %D" отобразится при печати как
"(c) KRONOS"            12-01-90".
```

%T - печать текущего времени в формате ЧЧ:ММ.СС.

УПРАВЛЯЮЩИЕ ПОСЛЕДОВАТЕЛЬНОСТИ:

Входной текст может содержать специальные управляющие символьные последовательности, которые переключают размер, тип и режим печати шрифта.

<управляющая_последовательность> ::= {"@@"[d]<char>}
d - необязательный числовой параметр в десятичном виде;
<char> - идентификатор команды.

Последовательности управляющих символов в печатаемом тексте опускаются. Для желающих все же напечатать символ '@@' необходимо его удвоить, или же переопределить его путем вставки управляющей последовательности @@=<char>, где C - новый управляющий символ, который может быть переопределен тем же способом (C=B - и уже B становится управляющим).
@@-последовательности рассматриваются как управляющие только при наличии ключа +i !!!

Допустимы следующие идентификаторы команд:

< - перевести каретку на начало текущей строки;
s - сохраняет номер фонта и все моды в стек;
e - восстанавливает номер фонта и все моды из стека,
E - восстанавливает номер фонта и все моды из стека, не счеркивая элемент стека (эквивалентно @e@s);
^ - устанавливает каретку на половину строки вверх;
_ - устанавливает каретку на половину строки вниз;
R(r) - включает (выключает) инверсию символов;

A(a) - включает (выключает) оттенение (ужирнение) символов;

U(u) - включает (выключает) подчеркивание символов;

H(h) - включает (выключает) вертикальную растяжку символов;

W(w) - включает (выключает) горизонтальную растяжку символов;

{d}g - устанавливает фон для текста;
{d}L - устанавливает параметр lines (см. числовые ключи);
{d}P - устанавливает номер текущей страницы;
{d}M - устанавливает параметр ma (см. числовые ключи);
{d}m - устанавливает параметр mo (см. числовые ключи);
{d}f - устанавливает параметр font (см. числовые ключи).
{d}. - сдвинуть каретку на указанное число точек вправо;
{d}l - установить интервал между строками в полустроках.
{d}# - изобразить один из примитивов псевдографики, изображенных на рисунке схематически:

```
0  __    1 _ _  2 __
  |         |        |

3 |-     4  +     5 -|

6 |_     7 _|_   8 _|
```

Номер примитива, указанный слева от него, подставляется вместо d.
{d}_ - изобразить псевдографический примитив "черта" указанное количество раз;
| - изобразить псевдографический примитив "вертикальная черта".

rm
rEmOVE FILES

Удаляет указанные файлы, отвязывая их от имени. Файл утрачен для пользователя, если он не привязан ни к одному имени (см. также утилиту ln).

rm {файловое_дерево} [-ключи] [время]

КЛЮЧИ:

q - не запрашивать подтверждение на удаление;
T - удаление дерева с поддиректориями;
d - для удаления директории; удаляется только пустая директория;
y - для удаления системных файлов;
z - для удаления файлов-устройств (узлов);
l - не сообщать имена удаляемых файлов.

О параметре 'время' читайте в разделе о параметрах файловых утилит.

ПРИМЕРЫ:

rm имя_файла
удаление файла 'имя_файла' с текущей директории;

rm * -q
удаление всех файлов с текущей директории;

rm *.cod
удаление всех кодовых файлов с текущей директории;

rm /dev/lp0 -z
удаление файла-устройства принтера.

shell
USER shell

Пользовательская оболочка системы. Утилита универсальна и имеет несколько функций, которые зависят от ключа, с которым запускается утилита. С подробным описанием можно ознакомиться в Руководстве по ОС Excelsior.

Раскрутка системы после загрузки

shell -root <имя_устройства>

используется в файле конфигурации системы при ее раскрутке.

При раскрутке системы выполняются следующие действия:

- устройство с указанными именем монтируется на корень файловой системы;
- текущая директория устанавливается на корень файловой системы;
- на текущей директории ищется командный файл с именем "profile.@" и исполняется;
- shell заканчивается.

Ведение диалога с пользователем

shell -home <имя_директории_пользователя>

Запуск осуществляется утилитой входа в систему login или администратором непосредственно. При этом выполняются следующие действия:

- текущая директория устанавливается на указанную директорию;
- исполняется командный файл с именем $TTY_up.@;
- исполняется командный файл "profile.@" с текущей директории;
- в окружение заносятся параметр HOME с именем указанной директории и параметр USER с именем пользователя;
- начинается диалог с пользователем (см. Главу 1).

Интерпретация командных файлов

shell "<"имя_командного_файла { аргументы }

Может быть запущена пользователем для исполнения командного файла.

По пути, указанному в ETC, ищется файл с указанным именем, интерпретируется как командный файл, после чего утилита завершается.

Запуск без аргументов

```
shell
```

Ведет диалог, который заканчивается либо по команде bye, либо по нажатии клавиши Esc.

tmspo
tImE spoOLER

tmspo <имя_файла>

Читает время из специального файла при запуске системы и записывает текущее время в этот файл каждые пять минут в течение работы системы.

vx
CODEFILE vISUALIZER

Визуализатор кода программы.

Утилита работает в интерактивном режиме.

vx <имя_модуля> [-r]

Вход в утилиту с одновременным выбором альтернативы PROC (см.), то есть с показом процедурной таблицы указанного модуля.

Утилита ищет кодофайл с названием '<имя_модуля>.cod' и реффайл с названием '<имя_модуля>.ref' на директориях, указанных в окружении задачи. Если найден хотя бы один из этих файлов, то утилита работает с ним.

vx [-r]

Вход в монитор с выбором альтернативы MODULEs (см.)

Подсказка

В любой момент работы утилиты для получения подсказки нажмите клавишу F1.

Завершение работы с утилитой

Чтобы закончить работу с утилитой, перейдите в дежурное меню и нажмите клавишу ESC.

Для всех меню

Все меню допускают:

- движение по меню с помощью стрелок и клавиш PgUp, PgDn;
- выбор альтернативы клавишей ENTER.
- переход между имеющимися на экране меню (клавиша TAB - в одну сторону и ALT TAB - в другую);
- удаление текущего меню (клавиша ESC);
- изменение положения окна на экране. Для этого, находясь в меню, нажать F2 и двигать окно стрелками, табулятором (TAB, ALT TAB) и клавишами PgUp, PgDn, Home, End.
- изменение размеров окна.

Для этого, находясь в меню, нажать F2.

Для увеличения вертикального размера на строку нажмите F3, для уменьшения - F4.

Для увеличения ширины колонки на символ нажмите Ins, для уменьшения - Del.

Для увеличения числа колонок нажмите '+', для уменьшения - '-'.

При всех манипуляциях окно изображается пустой рамкой.

Выбрав подходящие размер и место, нажмите ENTER. Окно заполнится содержимым.

Дежурное меню

Верхняя (нулевая) строка экрана выделена для меню (board), которое мы будем здесь называть дежурным, потому что его нельзя удалить (если даже ничего нет, уж оно-то останется). Выглядит дежурное меню следующим образом:

PROC EXT INFO STR CONST TYPEs GLOB MULTI MODULEs

Переход в дежурное меню из любой точки осуществляет клавиша F10.

Альтернативы дежурного меню

PROC - выдача процедурной таблицы.

Таблица выдается в виде окна, в котором размещено процедурное меню. Его альтернативами являются шестнадцатеричные номера процедур в модуле. Если найден реффайл, то выдаются и имена процедур.

В первой строке появляется информация о процедуре, на которую указывает курсор: смещение и, если найден реффайл, номер строки текста, в которой начинается процедура.

Процедурное меню

В процедурном меню имеется возможность спозиционироваться на процедуру:

- с заданным смещением (клавиша 'f');
- с указанным именем, если найден реффайл (клавиша 'n');

После выбора альтернативы создаются окно с кодом процедуры и (при наличии реффайла) окно с описанием процедуры (имя и тип процедуры, имена и типы параметров и локалов).

Код процедуры

Код процедуры выдается в следующем виде:

глобальный PC
локальный PC
мнемоника M-кода
непосредственные операнды (если есть);
номер строки и позиция соответствующей лексемы в тексте модуля
(если есть реффайл).

Имеются дополнительные возможности:

- позиционирование в конец процедуры (клавиша '_');
- прыжки по командам, осуществляющим условные и безусловные переходы (клавиша ENTER);
- пометить кусок кода (клавиши '[' и ']') и записать в файл с указанным именем (клавиша 'w');
- поиск в процедуре глобального PC (клавиша 'g*);
- поиск в процедуре локального PC (клавиша 'l').

EXT - список импортируемых модулей

Выдается меню, альтернативы которого - номер модуля и его имя. При выборе модуля попадаем в уже знакомое меню, приведенное в начале описания утилиты, но для выбранного модуля.

Для возвращения в предыдущий модуль выйдите в дежурное меню и нажмите ESC.

INFO - выдача общих сведений

Выдается следующая информация:
имя модуля,
время создания кодофайла,
имя и версия компилятора,
число процедур в модуле,
количество глобалов,
количество мультиглобалов,
количество внешних модулей,
размер строкового пула,
размер кода,
версия кодофайла,
минимальный необходимый размер процедурного стека,
прикидка компилятора о размере стека,
признак уникальности (если уникальный - 1),
размер кодофайла.

STR - выдача строкового пула

Слева выдается шестнадцатеричный дамп, справа - текстовая интерпретация.

CONST - выдача структурных констант, если есть реффайл. Выдается имя константы и ее значение в шестнадцатеричном виде.

TYPEs - выдача описанных и импортируемых типов,
если есть реффайл.

Появляется окно с именами типов. В первой (информационной) строке указанный курсором тип расшифровывается.

GLOB - выдача глобалов модуля, если есть реффайл.

Появляется окно с именами глобалов. В первой (информационной) строке выдается тип, с которым описан указанный курсором глобал.

MULTI - выдача мультиглобалов модуля.

Выдается окно с таким содержимым:
номер мультиглобала;
смещение;
размер.

MODULEs

При выборе этой альтернативы выдается окно с меню, альтернативами которого являются имена кодофайлов (реффайлов) на текущей директории, а также имена директорий, по которым можно гулять.

После выбора модуля работа с ним происходит, как и в случае выбора импортируемого модуля.

КЛЮЧИ:

-# - версия утилиты;
-h - подсказка;
-r - игнорировать реффайлы и содержащуюся в них информацию.