Постнов Александр Вячеславович М8О-101Б-21

Вариант 17

**6. Команды ОС UNIX. Формат команд. Интерпретатор команд.**

Команды операционной системы Unix имеют следующий формат:

**command [-f] [-a flag\_parameter] [parameter]**

Каждая команда состоит из одного или нескольких слов и начинается с названия – (**command**).

За названием могут следовать флаги, параметры флагов и параметры, которые называются аргументами.

Аргумент, состоящий из знака минус '**-**', за которым следует одна буква, называется **флагом**. Флаги обычно задают режим работы команды.

За некоторыми флагами могут следовать аргументы, относящиеся только к этому флагу (flag\_parameter). Такие аргументы называются **параметрами флагов**.

Аргументы, задаваемые после последнего флага и параметры флага, называются **параметрами.** Обычно они задают объекты для обработки.

Все команды ОС Unix подчиняются следующим правилам (им следуют разработчики новых команд):

1) длина имени команды должна составлять от двух до девяти символов;

2) имя команды записывается только строчными (малыми) буквами и цифрами;

3) флаги должны состоять из одного символа;

4) флаги разделяются символом '-';

5) флаги без параметров флагов могут группироваться за одним разделителем '-';

6) все флаги должны быть перечислены до параметров в командной строке;

7) порядок флагов не является существенным.

Условно все команды можно разбить на группы: **информационные, служебные и работы с файловой системой и файлами:**

· Информационные команды позволяют получать различного рода справки о состояние системы и справочную информацию. **Date** - вывести и установить дату и время; **Ps**– получение информации о процессах;**Who**- получение информации о работающих в системе.

· Служебные команды дают возможность выполнять ряд специальных функций, установка среды пользователя, печать параметров, установка прав доступа и другие. В этой группе отметим команды:**Stty**– установить характеристики терминала;**Kill**– завершить процесс;**Chmod**– сменить права доступа.

· Самая многочисленная группа команд обеспечивает работу с файловой системой и отдельными файлами. Здесь следует отметить следующие команды:**Cd**– изменение текущей директории;**Mv**– пересылка и переименование файлов;**Cp**– копирование файла;**Rm**- уничтожение файлов;**Cat**- объединение файла.

Всего в Unix реализовано более 130 команд, оперируя которыми можно вести активный диалог в системе и выполнить представительный набор функций.

**Интерпретаторы команд ОС Unix**

В любой операционной системе существует командная оболочка (интерпретатор команд), которая анализирует введенные команды и их аргументы, проверяет синтаксис, допустимость тех или иных ключей и т.д. и запускает соответствующую программу, т.е. создает в Unix процесс и передает ему управление.

Помимо исполнения команд интерпретатор команд выполняет и другие важные работы: генерацию имён файлов; переназначение ввода и вывода; конвейеризацию; контроль среды окружения; интерпретацию и выполнение командных процедур.

**Основные реализации командных интерпретаторов**

Пользователь ОС UNIX общается с системой через командный интерпретатор(shell). Через него происходит доступ к командам, файловой системе и другим функциям ядра UNIX. Это обычная программа (т.е. не входит в ядро операционной системы UNIX). Ее можно заменить другой или использовать несколько разных версий. Наиболее известны следующие версии:

**sh –** классический интерпретатор версии UNIX V7, иначе называемый по фамилии автора Bourne shell.

**ksh** – интерпретатор Korn shell, дополняющий классический shell возможностями работы с заданиями пользователя, историей работы и позволяющий редактировать командную строку при помощи команд, аналогичных vi. Является фактически стандартом для POSIX-совместимых систем, в частности, UNIX System V.

**csh** – стандартный интерпретатор BSD UNIX и производных от него систем. Отличается улучшенными диалоговыми возможностями, способом присваивания и экспортирования переменных в среду, управляющими конструкциями и рядом других моментов; тоже поддерживает историю и редактирование командной строки. Главное и, по моему мнению, отрицательное его отличие от других интерпретаторов,- это свои управляющие конструкции, не совпадающие сsh.

[**bash**](http://ln.com.ua/~openxs/projects/man/solaris8/bash.html) – свободно распространяемый в виде исходных текстов интерпретатор, называемый "Bourne another shell", объединяющий все лучшее из остальных интерпретаторов с удобными возможностями редактирования командной строки и работы с историей команд. В настоящее время- фактический стандарт.

**10. Поиск и замена в emacs**

# **Поиск и замена**

Как и в других редакторах, в Emacs есть команды для поиска случаев появления какой-нибудь строки. Основная команда поиска необычна тем, что она является наращиваемой; она начинает поиск до того, как вы закончили набор строки поиска. Существуют также команды и для ненаращиваемого поиска, более похожие на аналогичные команды в других редакторах.

Кроме обычной команды replace-string, которая находит все случаи появления одной строки и заменяет их другой, Emacs имеет более сложную команду замены, названную query-replace, которая запрашивает в интерактивном режиме, в каких случаях надо произвести замену.

## Наращиваемый поиск

Наращиваемый поиск начинается, как только вы набрали первый знак строки поиска. По мере того, как вы набираете строку поиска, Emacs показывает вам, где эта строка (в том виде, в каком вы ее уже набрали) может быть найдена. Когда вы набрали достаточно знаков, чтобы определить желаемое место, вы можете остановиться. В зависимости от того, что вы собираетесь делать потом, вам может понадобиться, а может и не понадобиться прекратить поиск явно с помощью RET.

C-s

Наращиваемый поиск вперед (isearch-forward).

C-r

Наращиваемый поиск в обратном направлении (isearch-backward).

C-s начинает наращиваемый поиск. C-s считывает знаки с клавиатуры и располагает курсор в первом месте появления знаков, которые вы набрали. Если вы наберете C-s и затем F, то курсор встанет справа после первой найденной `F'. Наберите О, и увидите, что курсор встал за первой найденной `FO'. После еще одной О курсор встанет за первой `FOO', находящейся за местом, с которого вы начали поиск. На каждом шаге текст буфера, совпадающий со строкой поиска, подсвечивается, если терминал может это сделать; текущая строка поиска обновляется на каждом шаге в эхо-области.

Если вы сделали ошибку в наборе строки поиска, то вы можете сбросить знаки с помощью DEL. Каждый DEL отменяет последний знак строки поиска. Этого не происходит до тех пор, пока Emacs не будет готов считать следующий вводимый знак; сначала знак, который вы хотите сбросить, должен быть либо найден, либо нет. Если же вы не хотите ждать, пока это произойдет, используйте C-g так, как описано ниже.

Когда вы будете удовлетворены достигнутым местом, вы можете набрать RET, что остановит поиск, оставляя курсор там, куда его поместила команда поиска. Любая команда, не имеющая специального значения при поиске, также останавливает поиск и затем выполняется сама. Таким образом, набор C-a привел бы к выходу из поиска и затем передвинул бы курсор в начало строки. RET необходим только в том случае, если следующая команда, которую вы хотите набрать, является печатным знаком, DEL, RET или другим управляющим знаком, имеющим особое значение во время работы поиска (C-q, C-w, C-r, C-s, C-y, M-y, M-r или M-s).

Иногда вы ищете слово `FOO' и находите его, но это не то, что вам нужно. Было второе `FOO', о котором вы забыли, находящееся перед тем, которое вы ищете. В этом случае наберите C-s еще раз, чтобы продвинуться к следующему появлению строки поиска. Это можно проделывать неограниченное число раз. Если вы проскочили, то можете отменить некоторые число знаков C-s с помощью DEL.

После выхода из поиска вы можете снова искать ту же самую строку, просто набрав C-s C-s: первый C-s -- это ключ, который запускает наращиваемый поиск, а второй C-s означает "повтор поиска".

Чтобы вы могли снова использовать более ранние строки поиска, существует список поиска. Команды M-p и M-n передвигают по списку, чтобы вы могли подобрать нужную строку для повторного поиска. Эти команды оставляют выбранную строку поиска в минибуфере, где вы можете ее отредактировать. Для завершения редактирования и начала поиска наберите C-s или C-r.

Если ваша строка вообще не найдена, то эхо-область говорит `Failing I-Search'. Курсор располагается после того места, где Emacs нашел из вашей строки всё, что смог. Таким образом, если вы ищете `FOOT', а такой строки нет, вы можете увидеть курсор после `FOO' в слове `FOOL'. С этого места вы можете сделать несколько вещей. Если ваша строка неправильно набрана, вы можете что-то стереть из нее и исправить. Если вы довольны найденным местом, вы можете набрать RET или любую другую команду Emacs, чтобы "принять то, что предложил этот поиск", или вы можете набрать C-g, что уничтожит из строки поиска знаки, которые не были найдены (`Т' в `FOOT'), оставляя те, что нашлись (`FOO' в `FOOT'). Второй C-g в этом месте отменяет поиск полностью, возвращая точку туда, где она была, когда поиск начался.

Если строка поиска содержит заглавную букву, то поиск производится с учетом регистра. Если вы удалите заглавные буквы из строки поиска, эта особенность исчезает.

Если поиск был неудачным и вы просите повторить его, набирая C-s еще раз, то он начинается снова с начала буфера. Повторение неудачного поиска в обратном направлении при помощи команды C-r начинает новый поиск с конца. Такой поиск называется круговым. Как только это произошло, в подсказке поиска появляется слово `Wrapped'. Если вы пройдете через точку, где начался поиск, это слово заменяется на `Overwrapped', что означает, что вы снова проходите через уже виденные вами совпадения.

Знак "выхода" C-g поступает во время поиска особым образом. Что именно он делает, зависит от статуса поиска. Если поиск нашел то, что вы хотели, и ожидает ввода, то C-g полностью отменяет поиск. Курсор возвращается туда, откуда вы начали поиск. Если C-g набирается, когда в строке поиска есть ненайденные знаки -- Emacs все еще ищет их, или он не смог их найти -- тогда эти ненайденные знаки сбрасываются из строки поиска. Сброс этих знаков делает поиск успешным, и он ждет дальнейшего ввода, таким образом, второй C-g отменит поиск полностью.

Чтобы найти символ перевода строки, введите C-j. Для поиска другого управляющего знака, такого как control-S или возврат каретки, вы должны отменить их специальное значение, набирая перед ними C-q. Эта функция C-q аналогична ее назначению как команды для вставки она заставляет трактовать следующий знак так, как в этом контексте трактовался бы любой "обычный" знак. Вы также можете задать знак по его восьмиричному коду: введите C-q и затем последовательность восьмиричных цифр.

Вы можете изменить направление поиска на обратное при помощи C-r. Вам следует поступить так, если поиск оказался неудачным, потому что место, с которого вы его начали, находилось слишком близко к концу файла. Повторение C-r продолжает поиск следующих случаев появления в обратном порядке, а C-s начинает поиск опять вперед. C-r в поиске может быть отменена при помощи DEL.

Если вы заранее знаете, что вам нужно вести поиск в обратном порядке, то чтобы начать поиск, вы можете использовать C-r вместо C-s, так как C-r также является ключом, запускающим команду (isearch-backward) для поиска в обратном порядке. Обратный поиск находит совпадения, которые расположены перед начальной точкой, так же как поиск вперед находит совпадения, начинающиеся после точки, где поиск начался.

Знаки C-y и C-w могут использоваться в наращиваемом поиске для захвата текста из буфера в строку поиска. Это делает удобным поиск другого случая появления того текста, который находится в точке. C-w копирует слово после точки в строку поиска, продвигая точку вперед через это слово. Следующая команда C-s для повторения поиска будет затем искать строку, включающую это слово. C-y подобна C-w, только копирует в строку поиска весь остаток текущей строки. И C-y, и C-w преобразуют копируемый текст к нижнему регистру, если поиск сейчас ведется без учета регистра; таким образом поиск остается регистронезависимым.

Команда M-y копирует в строку поиска текст из списка уничтожений. Она использует тот же текст, который был бы восстановлен командой C-y.

Когда вы выходите из наращиваемого поиска, метка устанавливается в то место, где точка была до начала поиска. Это удобно для возврата к этому месту. В режиме Transient Mark наращиваемый поиск устанавливает метку, не активизируя ее, если только метка уже не активна.

Чтобы настроить специальные знаки, которые понимает наращиваемый поиск, измените их привязки в таблице ключей isearch-mode-map. Для получения перечня привязок посмотрите документацию на isearch-mode с помощью C-h f isearch-mode RET.

## Ненаращиваемый поиск

В Emacs также есть удобные команды ненаращиваемого поиска, которые требуют от вас полностью набрать строку поиска до начала работы.

C-s RET строка RET

Поиск заданной строки.

C-r RET строка RET

Поиск строки в обратном направлении.

Чтобы начать ненаращиваемый поиск, наберите сначала C-s RET. Эта команда входит в минибуфер для считывания строки поиска; ограничьте эту строку с помощью RET, и поиск начнется. Если строка не будет найдена, команда поиска выдает ошибку.

Способ работы C-s RET заключается в следующем: C-s запускает наращиваемый поиск, который специально запрограммирован так, что запускает ненаращиваемый поиск, если заданный вами аргумент является пустым. (Такой пустой аргумент в других случаях был бы бесполезен). C-r RET работает аналогично.

Однако, запрошенный с помощью C-s RET ненаращиваемый поиск не запускает непосредственно search-forward. Первым делом проверяется, не будет ли следующим знаком C-w, что запустит поиск слов.

Прямой и обратный ненаращиваемый поиск осуществляются командами search-forward и search-backward. Эти команды могут быть привязаны к ключам обычным способом. Возможность их запуска через наращиваемый поиск имеет исторические причины и, помимо этого, существует для того, чтобы вам не нужно было находить для них подходящие последовательности ключей.

## Поиск слов

Поиск по словам применяется для отыскания последовательности слов независимо от того, как эти слова разделены. Более подробно, вы набираете строку из нескольких слов, используя для их разделения одиночные пробелы, и эта строка может быть найдена, даже если в оригинале слова разделены несколькими пробелами, переводами строки, либо любыми знаками препинания.

Поиск слов полезен при редактировании печатных документов, подготовленных в программах для форматирования текста. Если вы редактируете, просматривая уже напечатанную, отформатированную версию, то вы не можете сказать, где прерывается строка в исходом файле. При помощи же поиска слова вы можете искать, не имея этой информации.

C-s RET C-w слова RET

Ищет слова, игнорируя пунктуацию между ними.

C-r RET C-w слова RET

Ищет слова в обратном направлении, игнорируя пунктуацию между ними.

Поиск слов -- это специальный случай ненаращиваемого поиска, и он вызывается с помощью C-s RET C-w. За этим следует строка поиска, которая всегда должна быть ограничена RET. Будучи ненаращиваемым, поиск не начинается до тех пор, пока аргумент не завершен.

Для обратного поиска слов используйте C-r RET C-w.

Прямой и обратный поиск слов реализован в командах word-search-forward и word-search-backward. Эти команды могут быть привязаны к ключам обычным способом. Возможность их запуска через наращиваемый поиск существует по историческим причинам и для того, чтобы вам не нужно было находить для них подходящие последовательности ключей.

## Команды замены

Глобальные команды поиска и замены не нужны в Emacs так часто, как в других редакторах, но они доступны. Кроме простой команды M-x replace-string, которая аналогична такой же команде в большинстве редакторов, существует команда M-x query-replace, которая для каждого появления образца спрашивает вас, надо ли его заменять.

Команды замены обычно работают с текстом от точки до конца буфера; однако, в режиме Transient Mark они действуют на область, когда метка активна. Все команды замены заменяют одну строку (или регулярное выражение) одной строкой замены. Можно выполнить параллельно несколько замен, используя команду expand-region-abbrevs.

### Безусловная замена

M-x replace-string RET строка RET новая-строка RET

Заменяет каждое вхождение строки на новую-строку.

M-x replace-regexp RET regexp RET новая-строка RET

Заменяет каждое совпадение с regexp на новую-строку.

Чтобы заменить каждый случай вхождения `foo' после точки на `bar', используется команда M-x replace-string с двумя аргументами `foo' и `bar'. Замещение происходит только в тексте после точки, так, если вы хотите охватить весь буфер, вы должны сначала отправиться в его начало. Все экземпляры вплоть до конца буфера будут заменены; чтобы ограничиться заменой в части буфера, сузьте его до этой части перед выполнением замены. В режиме Transient Mark, если область активна, замена ограничена этой областью)

Когда вы выходите из replace-string, точка остается на месте последней замены. Значение точки в момент, когда была запущена команда replace-string, запоминается в списке пометок. C-u C-SPC перемещает вас обратно.

Числовой аргумент ограничивает замену совпадениями, которые окружены ограничителями слов. Значение аргумента роли не играет.

### Команды замены и регистр букв

Если первый аргумент в команде замены набран в нижнем регистре, во время поиска вхождений для замены регистр игнорируется -- при условии, что case-fold-search не равна nil. Если case-fold-search установлена в значение nil, регистр учитывается во всех типах поиска.

Кроме того, когда аргумент новая-строка весь или частично написан строчными буквами, команды замены пытаются сохранить образец использования регистра в каждом вхождении. Таким образом, команда

M-x replace-string RET foo RET bar RET

заменяет `foo' в нижнем регистре на `bar' в нижнем регистре, `FOO' в верхнем регистре на `BAR', а `Foo' с первой заглавной буквой на `Bar'. (Три эти альтернативы: все строчные буквы, все заглавные и первая заглавная -- единственные варианты, которые может распознать replace-string.)

Если в строке подстановки использованы буквы верхнего регистра, то они остаются такими при каждой вставке этого текста. Если буквы верхнего регистра используются в первом аргументе, то второй аргумент всегда вставляется в том виде, в котором он дан, без изменения регистра. Аналогично, если переменная case-replace или case-fold-search установлена равной nil, замещение происходит без изменения регистра.

### Замена с подтверждением

M-% строка RET новая-строка RET

M-x query-replace RET строка RET новая-строка RET

Заменяет некоторые вхождения строки на новую-строку.

C-M-% regexp RET новая-строка RET

M-x query-replace-regexp RET regexp RET новая-строка RET

Заменяет некоторые совпадения с regexp на новую-строку.

Когда вам показывают вхождение строки или совпадение с регулярным выражением regexp, вы можете набрать следующее:

SPC

чтобы заменить это вхождение на новую-строку.

DEL

чтобы перейти к следующему вхождению, не заменяя это.

, (Запятая)

чтобы заменить это вхождение и показать результат. Затем у вас запрашивают ввод еще одного знака, чтобы узнать, что делать дальше. Так как замена уже произведена, то DEL и SPC в этой ситуации эквивалентны; обе переходят к следующему вхождению. Вы можете набрать в этом месте C-r (смотрите ниже), чтобы изменить замененный текст. Вы можете также набрать C-x u, чтобы отменить сделанную замену; эта команда выходит из query-replace, так что если вы хотите делать дальнейшие замены, вы должны использовать C-x ESC ESC RET, чтобы запустить замену заново.

RET

чтобы выйти без осуществления дальнейших замен.

. (Точка)

чтобы заменить этот экземпляр и затем выйти без продолжения поиска следующих вхождений.

!

чтобы заменить все оставшиеся экземпляры без повторных запросов.

^

чтобы вернуться к положению предыдущего вхождения (или к тому, что им было), если вы изменили его по ошибке. Это делается при помощи выталкивания из списка пометок. Можно использовать только один `^' подряд, так как во время работы query-replace хранится только одна предыдущая позиция замены.

C-r

чтобы войти в новый уровень рекурсивного редактирования, в том случае, когда экземпляр нуждается скорее в редактировании, чем просто в замене его новой-строкой. Когда вы сделаете это, выйдите из этого уровня рекурсивного редактирования, набрав C-M-c, чтобы перейти к следующему вхождению.

C-w

чтобы удалить это вхождение и потом войти в новый уровень рекурсивного редактирования, как в C-r. Используйте рекурсивное редактирование для вставки текста и замены удаленного вхождения строки. Когда вы закончите, выйдите из этого уровня рекурсивного редактирования с помощью C-M-c, чтобы перейти к следующему вхождению.

C-l

чтобы восстановить изображение экрана. Потом вы должны набрать еще один знак, чтобы указать, что делать с этим вхождением.

C-h

чтобы просмотреть сообщение, резюмирующее эти варианты. Потом вы должны набрать еще один знак, чтобы указать, что делать с этим вхождением.