**Отчёт**

по лабораторной работе № 9

Подготовил: Зеневич Александр Олегович Т-091

Преподаватель: Сватко Игорь Борисович

Командная строка представляет собой программное средство ввода команд пользователем и получения результатов их выполнения на экране. В современных операционных системах семейства Windows, командную строку обеспечивает стандартное приложение **cmd.exe**, так же называемое командным процессором, интерпретатором команд и консолью. Приложение ”Командная строка” предоставляет пользователю текстовый интерфейс для ввода команд и получения результатов их выполнения. Фактически, командная строка является программным эмулятором классической консоли первых компьютерных систем, представляющей собой терминал с клавиатурой, используемый оператором в качестве средства общения с компьютером. Как и во времена первых компьютеров, командная строка поддерживает стандартное устройство ввода – клавиатуру, и стандартное устройство вывода – дисплей. Пользователь вводит команды с клавиатуры и получает результаты их выполнения на экране дисплея.

Запуск командной строки.

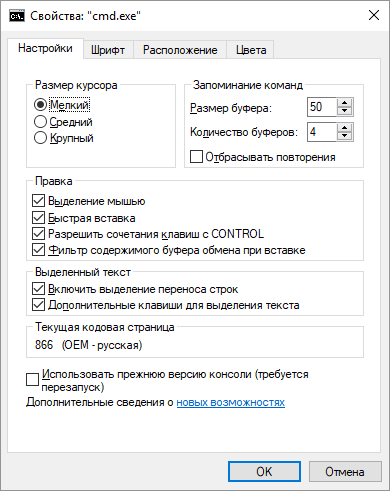
Для запуска командной строки можно воспользоваться одним из следующих способов:  
  
***Пуск - Выполнить*** (или клавиши Win+R) введите **cmd** и нажмите клавишу Enter;  
  
***Пуск - Все программы - Стандартные - Командная строка»;***  
  
***Пуск - Поиск - Командная строка***. Также, для запуска командной строки можно использовать заранее подготовленный ярлык, ссылающийся на исполняемый файл **%SystemRoot%\system32\cmd.exe** ( обычно – C:\Windows\system32\cmd.exe). Выполнение некоторых команд требует повышенных привилегий, поэтому, приложение командной строки должно быть запущено с использованием пункта контекстного меню ”Запуск от имени администратора”, вызываемого правой кнопкой мышки.

Настройка командной строки.

Стандартное окно командной строки – это окно с символами белого цвета на черном фоне. Параметры шрифтов, цвет и фон символов, использование буфера обмена и прочие свойства командной строки можно изменить в соответствии со своими предпочтениями. Настройки можно выполнить изменяя свойства ярлыка, с помощью которого выполняется запуск приложения командной строки, либо правкой параметров реестра, относящихся к командному процессору.  
  
[Страница, посвященная настройкам командного процессора CMD](https://ab57.ru/cmdtweak.html)  
  
Работа в командной строке предполагает ввод данных с клавиатуры, при чем, иногда довольно большого объема. Этот объем можно значительно сократить, используя некоторые приемы:

Использование буфера обмена.

Текст, выделенный в окне приложения, можно скопировать в буфер обмена, а также вставить его из буфера обмена в поле ввода. При стандартных настройках командной строки для выделения текста используется пункт ”Пометить” контекстного меню, вызываемого правой кнопкой мышки. При желании, в свойствах ярлыка, с помощью которого выполняется запуск можно включить режим выделения мышью:



В режиме выделения мышью, контекстное меню не используется, а текст выделяется с использованием левой кнопки мышки. Копирование и вставка выполняется при нажатии правой кнопки мышки. При включенном режиме ”Разрешить сочетания клавиш с CONTROL”, можно использовать стандартные сочетания клавиш:  
  
- CTRL+C (CTRL+Insert) – скопировать выделенный текст.  
  
- CTRL+V (Shift+Insert) – вставить выделенный текст.

Использование истории команд и горячих клавиш.

Для вызова ранее введенных команд используются клавиши со стрелками Стрелка Вверх - на одну команду назад и Стрелка Вниз - на одну команду вперед. Кроме того, можно использовать функциональные клавиши:  
  
**F1** - посимвольный вызов последней введенной команды. Каждое нажатие F1 приводит к последовательной подстановке в поле ввода одного символа из предыдущей команды.  
  
**F2** - копирование до символа. После нажатия на F2 на экране отображается запрос символа, до которого будет скопирована предыдущая команда. Например, если предыдущая команда была **ping yandex.ru**, и в качестве символа задана точка, то в строку ввода команды будет занесено **ping yandex**, если задан пробел, то - **ping**.  
  
**F3** - вызов предыдущей команды.  
  
**F4** - удалить до символа. Выполняется удаление текста от текущей позиции курсора до заданного символа.  
  
**F5** и **F8** - вызов буфера ранее введенных команд. Вывод по нажатию клавиши F5 прекращается когда будет отображена первая введенная команда текущей сессии.  
  
**F7** - отображение ранее введенных команд в отдельном окне в виде списка. Для выполнения нужной команды нужно выбрать ее, используя клавиши со стрелками, и нажать **ENTER**  
  
**F9** - выполнить команду, номер которой запрашивается. Номер команды в списке истории можно получить при использовании **F7**.  
  
В Windows 10 / Windows Server 2016 появились возможности, отсутствующие в предыдущих реализациях командной строки:  
  
- Изменение прозрачности окна консоли при нажатии комбинаций **CTRL**+**Shift**+**-** или **CTRL**+**Shift**+**+**.  
  
- Включение / выключение полноэкранного режима при нажатии **CTRL+Enter**.  
  
- Расширение возможностей выделения текста и редактирования:  
  
**Shift+Home** – выделить текст от текущего положения курсора до начала строки.  
  
**Shift+End** – выделить текст от текущего положения курсора до конца строки.  
  
- Быстрое перемещение по буферу экрана и поиск по контексту:  
  
**CTRL+Home** – перейти в начало буфера экрана  
  
**CTRL+End** – перейти в конец буфера экрана.  
  
**CTRL+Стрелка Вверх** - перемещение на 1 строку вверх.  
  
**CTRL+Стрелка Вниз** - перемещение на 1 строку вниз.  
  
**CTRL+F** - открыть диалог поиска текста в буфере экрана.  
  
Окно командной строки в Windows 10 можно закрыть с помощию стандартной комбинации клавиш - **ALT+F4**.  
  
Для отключения новых возможностей CMD, необходимо в свойствах приложения установить галочку ”Использовать прежнюю версию консоли (требуется перезапуск)”

Автодополнение путей файлов

В командной строке Windows можно выполнять подстановку имен файлов и каталогов при нажатии клавиши **Tab**. Например, для перехода, в каталог с длинным именем **Program Files**наберите начальную часть имени каталога  
  
**CD Pro** и нажмите **Tab**. В строке ввода должна появиться команда :  
  
**CD "Program Files"**  
  
Как видим, выполняется подстановка недостающей части имени каталога, а в тех случаях, когда в нем имеется символ пробела, добавляются еще и двойные кавычки.  
  
Если имеется несколько совпадающих начальных частей имен файлов или каталогов, при каждом нажатии клавиши **Tab** будет подставляться следующее по алфавитному порядку имя. Для возврата на предыдущее - используется комбинация **Shift+Tab**

Перенаправление данных стандартного ввода /вывода консоли.

Как уже упоминалось, в качестве стандартного устройства ввода командной строки используется клавиатура, а в качестве устройства вывода – дисплей. Однако, существует возможность **перенаправления** ввода-вывода на другие устройства c использованием операторов перенаправления:

 **>**- перенаправление вывода. Выходные данные записываются в файл или передаются на указанное устройство.  
  
**ping –n 5 localhost > nul** - выполнить пинг петлевого интерфейса 5 раз с перенаправлением вывода в фиктивное устройство **nul**. Вывод результатов выполнения команды подавляется. Подобный прием используется для организации задержки в командных файлах, поскольку пинг петлевого интерфейса выполняется почти мгновенно, а интервал между пингами равен одной секунде, время выполнения данной команды определяется значением параметра **-n**  
  
**ping –n 100 yandex.ru > C:\ping-ya.txt** - выполнить 100 раз пинг узла yandex.ru c записью результатов выполнения команды в файл C:\ping-ya.txt. Если файл не существует, он будет создан, а если существует, его содержимое будет затерто.

 **>>**- то же, что и в предыдущем случае, но данные записываются в конец файла.  
  
**ping –n 100 yandex.ru >> C:\ping-ya.txt** - то же, что и в предыдущем примере, но, если файл не существует, то он будет создан, а если существует, то запись результатов будет выполняться в конец файла.

 **<**- перенаправление ввода. Данные считываются не с клавиатуры, а из файла или другого устройства.  
  
**cmd < 1.txt** - запустить командный процессор CMD и выполнить ввод данных из файла 1.txt. Если в файле поместить строку **ping –n 100 yandex.ru**, то выполнится команда, рассмотренная выше.

 **|**- вывод первой команды перенаправить на вход следующей за ней.  
  
Нередко, вывод одной команды нужно передать в качестве вводимых данных для другой, т.е. объединить команды в последовательную цепочку:  
  
**ping -n 100 microsoft.com | find "Превышен интервал"** - результат выполнения команды **ping -n 100 microsoft.com** передается в виде входных данных для команды поиска строк (**find**), содержащих текст "Превышен интервал".  
  
**ping -n 100 microsoft.com | find "Превышен интервал" > C:\ping-ya.txt** - то же, что и в предыдущем примере, но с перенаправлением выводимых результатов выполнения команды в текстовый файл.

 Использование дескрипторов ввода-вывода консоли.  
  
Каждому открытому файлу или устройству соответствует свой **дескриптор** ( **handle**) который представляет собой неотрицательное число, значение которого используется породившим поток ввода-вывода процессом. По умолчанию, для всех процессов, в том числе и для командного интерпретатора **cmd.exe** :  
  
**0** ( STDIN ) – дескриптор стандартного ввода (ввод с клавиатуры).  
  
**1** (STDOUT) – дескриптор стандартного вывода (вывод на экран).  
  
**2** (STDERR) – дескриптор вывода диагностических сообщений (сообщений об ошибках на экран).  
  
Дескрипторы можно использовать в тех случаях, когда требуется перенаправить (изменить) источники и приемники данных в стандартных потоках ввода-вывода. Например:  
  
**ping.exe –n 100 yandex.ru 2> C:\pinglog.txt** - стандартный поток сообщений программы **ping.exe** будет выводиться на экран, а ошибки ( стандартный вывод с дескриптором = 2 ) будут записаны в файл **C:\pinglog.txt** . В реальном случае для программы **ping.exe** приведенная конструкция значения не имеет, поскольку она выводит и диагностику, и результаты на экран.  
  
Для задания перенаправления в существующие дескрипторы используется амперсанд (&), затем номер требуемого дескриптора (например, &1):  
  
**ping –n 100 yandex.ru >log.txt 2>&1** - стандартный поток сообщений об ошибках (дескриптор=2) перенаправляется в стандартный поток вывода (дескриптор = 1) и все это перенаправляется в файл **log.txt** текущего каталога.  
  
**ping –n 100 yandex.ru >log.txt 1>&2** - стандартный вывод (дескриптор = 1) перенаправляется в вывод сообщений об ошибках (дескриптор=2) и все это записывается в текстовый файл.  
  
Если дескриптор не определен, то по умолчанию оператором перенаправления ввода **<**будет ноль (0), а оператором перенаправления вывода **>**будет единица.

Объединение нескольких команд в цепочку

В командной строке Windows существует возможность последовательного выполнения нескольких команд в зависимости от результатов их выполнения. Для чего используются символы объединения команд - **&**(амперсанд) и **|**(вертикальная черта)  
  
**&** - одиночный амперсанд используется для разделения нескольких команд в одной командной строке. Например:  
  
**echo ping ya.ru** - вывод на экран строки **ping ya.ru**  
  
**echo &ping ya.ru** - последовательное выполнение команды **echo** без параметров и команды **ping ya.ru**  
  
**&&** - условное выполнение второй команды. Она будет выполнена, если код завершения (значение которого передается в стандартную переменную ERRORLEVEL) первой команды равен нулю, т.е. команда выполнена успешно.  
  
**команда1 && команда2** - выполняется **команда1**, а **команда2** выполняется, только если первая была выполнена успешно. Например:  
  
**ping ya.ru -err & ping –n 2 ya.ru** - при использовании единичного амперсанда, первая команда **ping ya.ru –err** выдаст сообщение о неверном параметре **-err**, а вторая, **ping –n 2 ya.ru** выполнит двукратный пинг узла **ya.ru**  
  
**ping ya.ru -err && ping –n 2 ya.ru**- при использовании двойного амперсанда, первая команда **ping ya.ru –err** выдаст сообщение о неверном параметре **-err** и соответственно, сформирует код завершения не равный нулю (ошибка), что приведет к тому, что вторая команда (**ping ya.ru -n 2**) выполнена не будет.  
  
Двойная вертикальная черта **||** - условное выполнение второй команды. Если первая команда завершилась с кодом возврата не равным нулю (неуспешно), то выполняется команда, следующая за двойной вертикальной чертой.  
  
**команда1 || команда2** - если **команда1** выполнена неуспешно, то запускается на выполнение **команда2**  
  
**ping –n 1 ya.ru && ping –n2 ya.ru || ping –n 3 ya.ru** - сначала выполнится 1-я команда **ping ya.ru –n 1** - однократный пинг узла **ya.ru**, затем выполнится команда **ping ya.ru -n 2** - двукратный пинг узла **ya.ru**. Третья команда **ping ya.ru -n 3**, трехкратный пинг узла **ya.ru** выполнена не будет.  
  
**ping –n 1 –err ya.ru && ping ya.ru -n 2 || ping ya.ru -n 3** - первая команда **ping ya.ru –n 1 –err**выполнится с ошибкой из-за неверного параметра **-err**, вторая команда **ping –n 2 ya.ru** - двукратный пинг узла **ya.ru** не будет выполнена из-за условия двойного амперсанда, и в результате выполнится третья команда **ping –n 3 ya.ru**, трехкратный пинг узла **ya.ru**.  
  
В некоторых случаях может возникнуть необходимость запрета обработки служебных символов и трактовки их в качестве простого текста. Например, если ввести в командной строке  
  
**ECHO ping -n 1 ya.ru & ping -n 2 ya.ru  
  
то вместо текста ping -n 1 ya.ru & ping -n 2 ya.ru будет выведена его часть до символа амперсанда ping -n 1 ya.ru и затем выполнена команда после амперсанда - ping -n 2 ya.ru . Проблему решает использование специального экранирующего символа ^ , который позволяет использовать служебные символы как текст:  
  
ECHO ping -n 1 ya.ru ^& ping -n 2 ya.ru - вывод на экран текста ping -n 1 ya.ru & ping -n 2 ya.ru  
  
Логика условной обработки команд, реализуемая с помощью конструкций && и || действует только на ближайшую команду, то есть, при вводе команды  
  
TYPE C:\plan.txt && DIR & COPY /?  
  
команда COPY /? запустится в любом случае, независимо от результата выполнения команды TYPE C:\plan.txt. Но, несколько команд можно сгруппировать с помощью скобок. Например, есть 2 командные строки:  
  
TYPE C:\plan.txt && DIR & COPY /?  
  
TYPE C:\plan.txt && (DIR & COPY /?)  
  
В первой из них символ условной обработки && действует только на команду DIR, во второй — одновременно на две команды: DIR и COPY. В качестве наглядного эксперимента, попробуйте выполнить вторую команду при условиях наличия и отсутствия файла C:\plan.txt. Для создания пустого файла можно воспользоваться копированием из фиктивного устройства nul:  
  
copy nul C:\plan.txt  
  
Для удаления файла используется команда erase c:\plan.txt или del C:\plan.txt**

Командные файлы

**Командные файлы (сценарии, скрипты) – это обычные текстовые файлы с заранее подготовленным набором команд для их выполнения командным процессором cmd.exe. Стандартно, такие файлы имеют расширение .bat или .cmd. Строки командных файлов могут содержать специфические команды самого процессора команд, например - FOR, ECHO, REM и т.п. или имена исполняемых модулей – reg.exe, sc.exe, auditpol.exe., которые можно использовать без расширения – reg, sc, auditpol. Пример простого командного файла:  
  
REM Создается текстовый файл со списком каталога Windows  
dir C:\Windows > %TEMP%\winlist.txt  
REM выполняется задержка на 5 секунд  
ping -n 5 localhost > nul  
REM Файл открывается в редакторе WordPad  
write %TEMP%\winlist.txt  
REM После завершения работы Wordpad, текстовый файл удаляется.  
erase C:\winlist.txt**