Про bash и пр.

25 сентября 2020

Имя файла

Полное имя файла

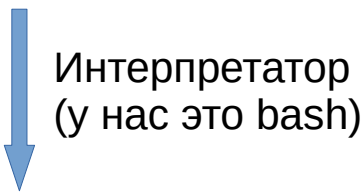
Путь к файлу (полное имя директории, в которой лежит файл)

```
/P/y20/term1/pr4/files/TySQkQ.tmp
```

[краткое] имя файла

Командная строка — последовательность байтов

```
kodomo:~$ abc def ghi klm
```

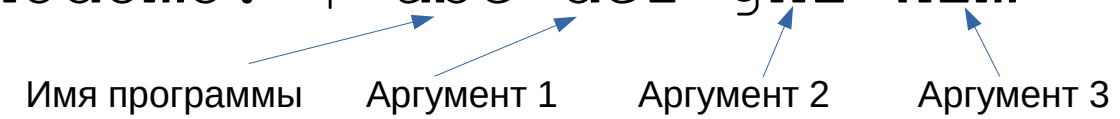


Интерпретатор командной строки
(у нас это bash)

ОС (в случае kodomo — Debian Linux)

Программа и её аргументы

```
kodomo:~$ abc def ghi klm
```



Пробел — тоже байт! Но bash интерпретирует его как разделитель аргументов

Спецсимволы bash

Ряд символов воспринимается интерпретатором командной строки bash не буквально, а как указания произвести некоторые операции. Это следующие символы:

- пробел (а также табулятор и перенос строки);
- кавычки (" и '), обратная косая черта (\ )
- `*  ?  [  ]  {  }`
- `>  <  |  ;  &`
- `$  !  (  )  ` ` ~  #  %  ^`

Спецсимволы bash

К следующему разу нужно:
(а) помнить весь список спецсимволов
(б) знать, зачем нужны те, что выделены красным

- **пробел** (а также табулятор и перенос строки);
- **кавычки (" и '), обратная косая черта (\ )**
- *** ? [ ] { }**
- **> < | ; &**
- $! ( ) ` ~ # % ^ /

Не используйте спецсимволы (в том числе пробел) в именах файлов и директорий!
(а ещё не используйте в этих именах русские буквы)

Спецсимволы bash: защита символов

Кавычки и обратная косая черта используются для передаче программам параметров, содержащих спецсимволы.

- В одинарных кавычках все символы воспринимаются буквально.
- В двойных кавычках символы $ и ` (обратные кавычки) сохраняют специальное значение (подстановка содержания переменной и вывода другой команды соответственно); кроме того, сочетание \$ превращается в (буквальный) символ $, \" — в символ ", а \` — в символ ` (обратная кавычка). Все остальные символы воспринимаются буквально.
- Любой символ после \ воспринимается буквально (в том числе сам \)

Спецсимволы bash: как проверить, что получается?

Команда echo просто выдаёт то, что она получила от bash, на экран

Поэтому с помощью echo легко проверить, во что bash превращает вашу строку

Спецсимволы bash: маски файлов

- Звёздочка * в именах файлов заменяет любое количество любых символов
- Вопросительный знак ? в именах файлов заменяет один любой символ
- Пара квадратных скобок заменяет любой символ из стоящих в этих скобках

  bash превращает выражение с этими символами в список аргументов изо всех имён файлов, подходящих под маску
  (если же таких файлов не нашлось, эти символы передаются программе в неизменном виде)

Стандартные потоки

- Каждое консольное приложение (то есть, попросту говоря, каждая программа, с которой можно работать из командной строки) имеет дело с тремя т.н. потоками: **stdin**, **stdout** и **stderr**.
- Поток stdin по-русски называется «стандартный поток ввода» и по умолчанию содержит то, что вы набираете на клавиатуре во время работы программы (например, в ответ на вопросы программы).
- Потоки stdout и stderr — это, соответственно, «стандартный поток вывода» и «стандартный поток ошибок», и по умолчанию их содержание отображается на экране. Хотя это два разных потока, для пользователя они обычно сливаются вместе. Но можно перенаправить stdout одной программы на stdin другой программы или в файл, при этом stderr будет продолжать выводиться на экран.

Перенаправление вывода

- Символ > используется в bash для перенаправления стандартного вывода (stdout) команды в файл.
- Например, команда

  ```
  ls > cur_dir.txt
  ```

  приведет к тому, что список файлов текущей директории окажется в файле cur_dir.txt
  При этом, если файл с таким именем уже существовал, то его старое содержимое исчезнет.

Перенаправление вывода

- Чтобы дописать stdout команды в конец уже существующего файла, используется сочетание символов >>. Например, команда

  ```
  cat file2.txt >> file1.txt
  ```

  приведет к тому, что в файле file1.txt после его старого содержимого окажется еще и содержимое файла file2.txt

  (из примера нетрудно догадаться, что делает программа cat).

Перенаправление вывода

- Для направления stdout одной команды на вход (stdin) другой команды используется "pipe" (конвейер), задаваемый знаком |
- Например:

  ```
  ls -l | less
  ```

  позволяет просмотреть программой less информацию о файлах в текущей директории (less в такой ситуации возьмёт текст со стандартного ввода)

Программа "grep"

- Программа grep предназначена для вытаскивания из текстового файла строк, содержащих некоторое слово. Например, если написать
  `grep abc some.txt`
  то на экране появятся строки файла some.txt, содержащие сочетание букв abc. Выдача идет в stdout, поэтому её легко перенаправить в файл или просмотреть программой less.
- Важная особенность программы grep: если файл не указан, то grep берет данные для поиска строчек из stdin.

Программа "grep"

- Опция -c позволяет посмотреть не сами строки, а их число:
  `grep -c abc some.txt`
  на экране появится число строки файла some.txt, содержащих сочетание букв abc.
- Программе можно указывать несколько файлов:
  `grep abc file1.txt file2.txt`
  или
  `grep abc *.txt`
  при этом будут выданы не только строки, но и имена файлов, в которых строки нашлись (обязательно посмотрите на примерах, как это выглядит!)
- Иногда удобно делать конвейер из grep'ов, например:
  `grep abc file.txt | grep -c def`

Программа "grep"

- В выражении для поиска можно использовать знаки ^ и $:
  - ^ означает искать только в начале строк:
    `grep '^abc' file.txt`
    выдаст все строки файла, начинающиеся с abc
  - $ означает искать только в концах строк:
    `grep 'abc$' file.txt`
    выдаст все строки файла, кончающиеся на abc
  - Кавычки в примерах защищают спецсимволы