UNIX::Advanced

- Символ &;

UNIX::Advanced

- Символ &;
- переводит процесс в фоновый режим;
- посмотреть все фоновые процессы: команда *jobs*, переключать задачи в фон или выводить из него: команды *bg* и *fg*;

UNIX::Advanced

- Символ &;
- переводит процесс в фоновый режим;
- посмотреть все фоновые процессы: команда *jobs*, переключать задачи в фон или выводить из него: команды *bg* и *fg*;
- Разделитель команд в одной строке: ; echo 1; echo 2; echo 3;

UNIX::Advanced

- Символ &;
- переводит процесс в фоновый режим;
- посмотреть все фоновые процессы: команда *jobs*, переключать задачи в фон или выводить из него: команды *bg* и *fg*;
- Разделитель команд в одной строке: ; echo 1; echo 2; echo 3;
- Операторы && и || # аналогично Windows

UNIX::Advanced

- Символ &;
- переводит процесс в фоновый режим;
- посмотреть все фоновые процессы: команда *jobs*, переключать задачи в фон или выводить из него: команды *bg* и *fg*;
- Разделитель команд в одной строке: ; echo 1; echo 2; echo 3;
- Операторы && и || # аналогично Windows
- Оператор | # поток вывода команды слева становится потоком ввода команды справа;
  например: help | more или help | less;

UNIX::grep,cut,tr

- Задание: с помощью *grep* найти все строки в help, содержащие строку **abc**

UNIX::grep,cut,tr

- Задание: с помощью *grep* найти все строки в help, содержащие строку **abc**
- help | grep "abc"

UNIX::grep,cut,tr

- Задание: с помощью *grep* найти все строки в help, содержащие строку **abc**
- help | grep "abc"
- Задание: узнать все свои ip-адреса;

UNIX::grep,cut,tr

- Задание: с помощью *grep* найти все строки в help, содержащие строку **abc**
- help | grep "abc"
- Задание: узнать все свои ip-адреса;
- ip addr | grep "inet"

UNIX::grep,cut,tr

- Задание: с помощью *grep* найти все строки в help, содержащие строку **abc**
- help | grep "abc"
- Задание: узнать все свои ip-адреса;
- ip addr | grep "inet"
- ip addr show | grep "inet" | tr -s " " | cut -d " " -f 3

UNIX::grep,cut,tr

- Задание: с помощью *grep* найти все строки в help, содержащие строку **abc**
- help | grep "abc"
- Задание: узнать все свои ip-адреса;
- ip addr | grep "inet"
- ip addr show | grep "inet" | tr -s " " | cut -d " " -f 3
- ip addr show | grep "inet" | tr -s " " | cut -d " " -f 3 | cut -d "/" -f 1

UNIX::grep,cut,tr

- Создадим при помощи *nano* файл, в котором будут перечислены разные слова, и выведем его содержимое: cat words.txt

UNIX::grep,cut,tr

- Создадим при помощи *nano* файл, в котором будут перечислены разные слова, и выведем его содержимое: cat words.txt
- Отсортировать слова в файле: cat words.txt | sort

UNIX::grep,cut,tr

- Создадим при помощи *nano* файл, в котором будут перечислены разные слова, и выведем его содержимое: cat words.txt
- Отсортировать слова в файле: cat words.txt | sort
- В отсортированном файле вывести только первые 5 строк: cat words.txt | sort | head -n 5

UNIX::grep,cut,tr

- Создадим при помощи *nano* файл, в котором будут перечислены разные слова, и выведем его содержимое: cat words.txt
- Отсортировать слова в файле: cat words.txt | sort
- В отсортированном файле вывести только первые 5 строк: cat words.txt | sort | head -n 5
- Вывести последние 5 строк: cat words.txt | sort | tail -n 5

UNIX::grep,cut,tr

- Создадим при помощи *nano* файл, в котором будут перечислены разные слова, и выведем его содержимое: cat words.txt
- Отсортировать слова в файле: cat words.txt | sort
- В отсортированном файле вывести только первые 5 строк: cat words.txt | sort | head -n 5
- Вывести последние 5 строк: cat words.txt | sort | tail -n 5
- Как вывести строки с 3 по 6?

UNIX::grep,cut,tr

- Создадим при помощи *nano* файл, в котором будут перечислены разные слова, и выведем его содержимое: cat words.txt
- Отсортировать слова в файле: cat words.txt | sort
- В отсортированном файле вывести только первые 5 строк: cat words.txt | sort | head -n 5
- Вывести последние 5 строк: cat words.txt | sort | tail -n 5
- Как вывести строки с 3 по 6?
- cat words.txt | sort | head -n 6 | tail -n 3

UNIX::Advanced

- Существует необходимость вывод одних команд использовать в других командах

UNIX::Advanced

- Существует необходимость вывод одних команд использовать в других командах
- вывести строку, содержащую какой-то текст и путь до which:

  echo "which is located in ‘which which‘"
  echo "which is located in $(which which)"

UNIX::Advanced

- Существует необходимость вывод одних команд использовать в других командах
- вывести строку, содержащую какой-то текст и путь до which:

  echo "which is located in ‘which which‘"
  echo "which is located in $(which which)"
- Как вывести код программы *which* на экран?

UNIX::Advanced

- Существует необходимость вывод одних команд использовать в других командах
- вывести строку, содержащую какой-то текст и путь до which:

  echo "which is located in ‘which which‘"
  echo "which is located in $(which which)"
- Как вывести код программы *which* на экран?
- cat $(which which)

UNIX::Advanced

- Обработать два идентичных файла или два схожих вывода команд одной командой:

  sort <(cat 1.txt) <(cat 2.txt)
  { cat 1.txt; cat 2.txt; } | sort
  sort <(cat 1.txt; cat 2.txt)

UNIX::Advanced

- Обработать два идентичных файла или два схожих вывода команд одной командой:

  sort <(cat 1.txt) <(cat 2.txt)
  { cat 1.txt; cat 2.txt; } | sort
  sort <(cat 1.txt; cat 2.txt)

- Обработка многострочного текста: << и <<<

UNIX::Advanced

- Обработать два идентичных файла или два схожих вывода команд одной командой:

  sort <(cat 1.txt) <(cat 2.txt)
  { cat 1.txt; cat 2.txt; } | sort
  sort <(cat 1.txt; cat 2.txt)

- Обработка многострочного текста: << и <<<

- sort<<END_OF_TEXT
  1
  3
  5
  7
  9
  $PWD
  11
  22
  END_OF_TEXT #$PWD будет раскрыта

UNIX::Advanced

- Обработать два идентичных файла или два схожих вывода команд одной командой:
  sort <(cat 1.txt) <(cat 2.txt)
  { cat 1.txt; cat 2.txt; } | sort
  sort <(cat 1.txt; cat 2.txt)
- Обработка многострочного текста: << и <<<
- sort<<END_OF_TEXT
  1
  3
  5
  7
  9
  $PWD
  11
  22
  END_OF_TEXT #$PWD будет раскрыта
- sort<<"EOF"
  1
  3
  2
  $PWD
  EOF #$PWD не будет раскрыта

UNIX::Advanced

- Обработать два идентичных файла или два схожих вывода команд одной командой:

  sort <(cat 1.txt) <(cat 2.txt)
  { cat 1.txt; cat 2.txt; } | sort
  sort <(cat 1.txt; cat 2.txt)

- sort<<<"1
  3
  2
  $PWD"# ввод без опознавательного знака EOF, $PWD раскрывается

UNIX::Advanced

- Обработать два идентичных файла или два схожих вывода команд одной командой:

  sort <(cat 1.txt) <(cat 2.txt)
  { cat 1.txt; cat 2.txt; } | sort
  sort <(cat 1.txt; cat 2.txt)

- sort<<<"1
  3
  2
  $PWD"# ввод без опознавательного знака EOF, $PWD раскрывается

- sort<<<'1
  3
  2
  $PWD' # раскрытия $PWD не будет

UNIX::Advanced

- Обработать два идентичных файла или два схожих вывода команд одной командой:

  sort <(cat 1.txt) <(cat 2.txt)
  { cat 1.txt; cat 2.txt; } | sort
  sort <(cat 1.txt; cat 2.txt)

- *xargs*: утилита для формирования списка аргументов и выполнения команды в UNIX-подобных операционных системах. Команда xargs объединяет зафиксированный набор заданных в командной строке начальных аргументов с аргументами, прочитанными со стандартного ввода, и выполняет указанную команду один или несколько раз

UNIX::Advanced

- Обработать два идентичных файла или два схожих вывода команд одной командой:

  sort <(cat 1.txt) <(cat 2.txt)
  { cat 1.txt; cat 2.txt; } | sort
  sort <(cat 1.txt; cat 2.txt)

- *xargs*: утилита для формирования списка аргументов и выполнения команды в UNIX-подобных операционных системах. Команда xargs объединяет зафиксированный набор заданных в командной строке начальных аргументов с аргументами, прочитанными со стандартного ввода, и выполняет указанную команду один или несколько раз

- echo A B C | xargs echo
  echo A B C | xargs -n 1 echo

UNIX::Advanced

- Обработать два идентичных файла или два схожих вывода команд одной командой:

  sort <(cat 1.txt) <(cat 2.txt)
  { cat 1.txt; cat 2.txt; } | sort
  sort <(cat 1.txt; cat 2.txt)

- *xargs*: утилита для формирования списка аргументов и выполнения команды в UNIX-подобных операционных системах. Команда xargs объединяет зафиксированный набор заданных в командной строке начальных аргументов с аргументами, прочитанными со стандартного ввода, и выполняет указанную команду один или несколько раз
- Регулярные выражения: сайт regexone.com

UNIX::Advanced

- Обработать два идентичных файла или два схожих вывода команд одной командой:

  sort <(cat 1.txt) <(cat 2.txt)
  { cat 1.txt; cat 2.txt; } | sort
  sort <(cat 1.txt; cat 2.txt)

- *xargs*: утилита для формирования списка аргументов и выполнения команды в UNIX-подобных операционных системах. Команда xargs объединяет зафиксированный набор заданных в командной строке начальных аргументов с аргументами, прочитанными со стандартного ввода, и выполняет указанную команду один или несколько раз
- Регулярные выражения: сайт regexone.com
- Использование регулярных выражений в grep:

  cat 1.txt | grep -P "\ d" # Perl-style

UNIX::Homework

- в файле /etc/group содержатся строки вида ГРУППА:x:n:USER1,USER2. Цель: написать программу, которая выводит все группы, в которых состоит пользователь (что есть группы пользователей будет позже), имя которого передали в параметрах;
- regexone.com - пройти уроки.