**Лабораторная работа №5. Выполнение базовых команд**

**Шпаргалка наиболее частых команд**[**​**](https://iu5edu.ru/wiki/devops/docs/labs/linux/commands/#%D1%88%D0%BF%D0%B0%D1%80%D0%B3%D0%B0%D0%BB%D0%BA%D0%B0-%D0%BD%D0%B0%D0%B8%D0%B1%D0%BE%D0%BB%D0%B5%D0%B5-%D1%87%D0%B0%D1%81%D1%82%D1%8B%D1%85-%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4)

* cd (change directory) - сменить рабочий каталог
* mv (move) - переместить объект файловой системы
* cp (copy) - скопировать объект файловой системы
* mkdir (make directory) - создать каталог
* echo - вывод аргументов на экран
* cat - вывод объекта (например, файла) на экран
* less - утилита для удобного просмотра больших текстовых файлов
* sort - утилита для сортировки
* grep - фильтрация строк по подстрокам

к сведению

Разберем несколько:

* ls — покажет содержимое директории, в которой вы находитесь. Если после команды ввести адрес конкретной папки, то она покажет то, что хранится в ней. Типы файлов. Система прав. Полезные аргументы:
  + -R в выводе команды появятся файлы из поддиректорий
  + -l отобразить списком с метаданными (права доступа и прочая информация из inode)
  + -a отобразить скрытые (начинающиеся с точки) объекты
* cd — change directory, сменить директорию. Из названия понятно, что с помощью этой команды можно перемещаться между каталогами. Вписываем её, а потом путь, например: cd /path/to/dir .

Есть сокращенные варианты:

* + cd - для перемещения в предыдущую директорию
  + cd (без аргументов) для прыжка в домашнюю папку
* cp - copy - скопировать объект. Из полезных опций -r позволяет делать рекурсивно (для всех подкаталогов) и -p позволяет сохранить все права на файл как у исходного.
* pwd — укажет полный путь до директории, в которой вы находитесь. Если вы перепрыгнули на рабочий стол с помощью cd ~/Desktop/, то pwd покажет что-то вроде /home/mak/Desktop.
* rm — команда для удаления файлов, каталогов и их содержимого. Удаляет жесткие ссылки,фактически данные остаются на диске, просто мы теряем на них указатели и эти блоки данных теперь считаются пустыми. На форумах часто ходит злая шутка для новичков, которым советуют починить что-нибудь в их систему с помощью rm -rf /, что потенциально стирает все данные.

Полезные аргументы:

* + -r удалить каталог и все вложенные объекты
  + -f удалить без подтверждения
* grep — это полнотекстовый поиск в файле. Очень полезная команда, когда вам нужно что-то найти. Например, grep значение /путь/до/файла покажет строки в которых есть "значение". А команда grep -ril значение /путь/до/каталога/\* покажет все файлы, в которых есть искомое значение.

Полезные аргументы:

* + -r рекурсивный поиск
  + -i игнорировать регистр (case **i**nsensetive)
  + -l вывести результат списком
  + -v инвертировать работу - пропускать только строки, в которых **НЕТ** значения
  + -E использовать регулярные выражения в "значении"
* df — disk free - показывает данные по ФС, в т.ч. сколько еще свободно места на каких ФС.
  + -h human-friendly, переводит байты в читаемое
  + -i inode, показывает информацию по айнодам на ФС
* du — disk usage - показывает данные по всем неудаленным файлам на ФС.
  + -h human-friendly, переводит байты в читаемое
* chown пользователь:группа путь — команда позволяет изменить владельца файла или каталога. -R для рекурсивного.
* chmod права путь — изменяет права на файл или каталог. Подробнее в разделе "Система прав в GNU/Linux".
* find — с помощью этой утилиты можно искать (и даже удалять) файлы по определённым параметрам. Например: find /mnt/data/backup/ -mtime +3 -type f -iname "\*.sql.gz" - выведет все файлы (-type f) в директории /mnt/data/backup 3-х дневной давности (дата изменения modify time +3 - -mtime +3) с именем попадающим в маску \*sql.gz. При добавлении опции -delete удаляет найденные файлы. Часто используется в таком виде в скриптах для очистки старых логов или резервных копий.
* awk — си-подобный сценарный язык построчного разбора и обработки входного потока по заданным шаблонам. Ключей много. Примеры и статья на хабре:echo word1 word2 word3 word4 | awk '{ print $2,$4 }' выведет word2 word4 для каждой строки (тут всего одна). echo word1:word2:word3:word4 | awk -F: '{ print $1,$3 }' выведет word1 word3.
* tail — позволяет вывести только конец файла/потока.
  + +2 - выкинуть первую строку
  + -n X - оставить только Х последних строк
* head — аналогично tail, только для начала файла/потока.
* ps — посмотреть список процессов, с аргументом aux показывает все процессы в системе.

**Pipeline**

Для выполнения работы вам непременно потребуется знание о том, что такое конвейер (pipeline). Небольшое напоминание о работе конвейера:

cat file.txt | grep bmstu | sort -u

* cat file.txt - вывести содержимое файла, передаем это утилите grep
* grep bmstu - оставить только строки, содержащие bmstu, передать вывод утилите sort
* sort -u - отсортировать в алфавитном порядке и оставить только уникальные значения

**Shell/Bash**

Пример скрипта, который создает файл с содержимым первого и второго аргументов:

#!/bin/sh  
  
echo "$1" > test.txt  
echo "$2" >> test.txt

Пример того, как сделать файл "исполняемым":

chmod +x file.sh

Пример того, как запустить "исполняемый" файл c аргументами из текущей директории:

./file.sh 10 15

к сведению

А почему исполняемый файл нельзя запустить просто написав file.sh как в windows?

Все дело в безопасности. Это очень плохая идея позволять подменять стандартные утилиты файлами из текущей директории, есть риск исполнить не то, что подразумевал пользователь, а то, что подложил ему в каталог злоумышленник. Поэтому текущая директория **не добавляется** к переменной PATH, списку директорий в которых производится поиск исполняемых файлов.