**Отчёт по лабораторной работе** № по курсу

студент группы М8О-107Б-18 Гамов Павел Антонович № по списку

Адрес (e-mail) pagamov@gmail.com

Работа выполнена: “ “ 2018г.

Преподаватель: асп. Каф. 806 Ридли А.Н.

Входной контроль знаний с оценкой

Отчёт сдан “ “ 2018 г итоговая оценка

Подпись преподавателя

1. Тема: встроенные утилиты os unix. Работа с потоками, файлами, форматами и данными.
2. Цель работы: изучить 39 приведенных ниже утилит и программ. Рассмотреть не менее 20 их них на конкретных примерах, расписать и узнать дополнительные функции, параметры и флаги каждой из утилит. Запротоколировать использование каждой на заранее подготовленных файлах.
3. **Оборудование** (лабораторное):

ЭВМ Pentium , процессор2.7 ГГц , имя узла сети с ОП 4096 МБ

НМД 2 ГБ. Терминал bash адрес . Принтер

Другие устройства

Оборудование ПЭВМ студента, если использовалось:

Процессор 2.7 GHz Intel Core i5, ОП 8 GB (1867 MHz LPDDR3), НМД 256 ГБ. Монитор встроенный

Другие устройства

1. Программное обеспечение (лабораторное):

Операционная система семейства Unix , наименование Ubuntu версия 18.04.1 LTS

Интерпретатор команд bash версия 4.3.48

Система программирования версия

Редактор текстов Emacs версия 7.4

Утилиты операционной системы

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

1. Программное обеспечение ЭВМ студента, если использовалось:

Операционная система семейства Unix , наименование Ubuntu версия 18.04.1 LTS

Интерпретатор команд bash версия 4.3.48

Система программирования версия

Редактор текстов GNU Emacs версия 25.2.2

Утилиты операционной системы

Прикладные системы и программы

Местонахождения и имена файлов программ и данных /home/pavel

1. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальное описание с пред- и постусловиями)

Рассмотреть функционал каждой функции/утилиты/программы на конкретных примерах.

1. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты, либо соображения по тестированию].

Ознакомиться с каждой из функций, утилит, программ используя встроенный man, используя справочные материалы в интернете и в научной литературе.

Подготовить необходимые для теста файлы.

Внести в протокол демонстрации основных возможностей каждой утилиты/программы/функции.

Внести пояснения/комментарии в процесс если необходимо.

Пункты 1-7 отчёта составляются **строго** **до** начала лабораторной работы.

Допущен к выполнению работы. Подпись преподавателя

1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем)
2. **Дневник отладки** должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

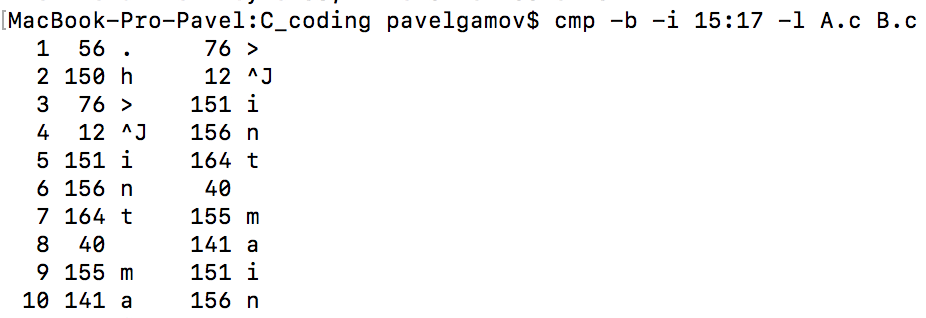
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб.  или  дом. | Дата | Время | Событие | Действие по исправлению | Примечание |
| 1 | дом | .2018 |  |  |  |  |

1. Замечание автора по существу работы:
2. Выводы: встроенные утилиты os unix позволяет пользователю очень быстро оперировать над множеством файлов и данных. Утилиты могут объединяться, использоваться вместе с другими посредством работы с потоками. Различные дополнительные опции и параметры такие как флаги позволяют расширить и без того не малый функционал, повышая потенциал каждой утилиты. Встроенные программы, такие как редакторы текста, позволяют облегчить пользовательское редактирование данных.
3. **Недочеты**, допущенные при выполнении задания, могут быть устранены следующим образом:

Подпись студента

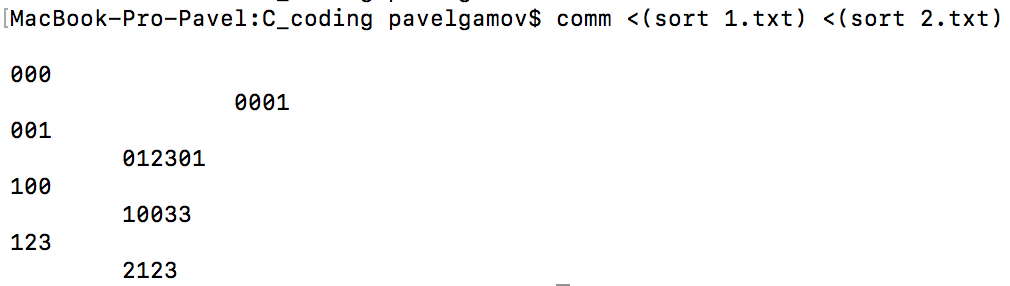
**сmp** – побитовое сравнение двух файлов.

-i – пропускает количество первых байт. -b – вывод значений отличающихся байтов. -l – вывод всех различных байтов.

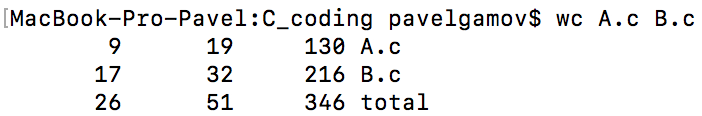


**comm** – принимает 2 файла предварительно отсортированные лексически.

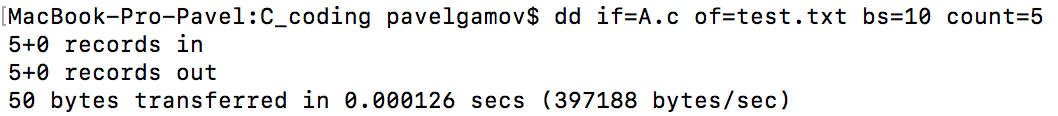
1 колонка выводит строчки только в 1 файле, 2 только во втором, 3 колонка в обоих. Атрибуты -1 -2 -3 подавляют вывод соответствующих колонок. -i убирает требование к регистру.



**wc** – выводит число переводов строк, слов и байт для каждого указанного файла



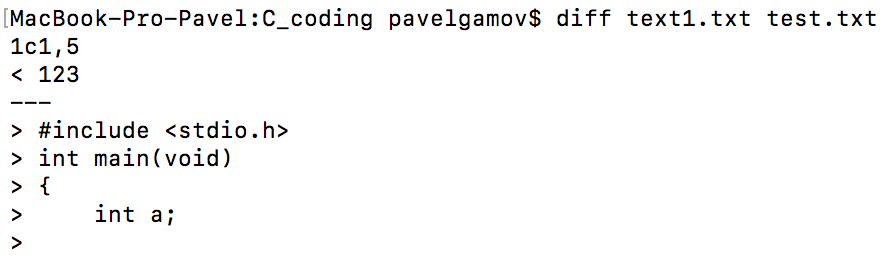
**dd** – копирование данных из а в б побайтово. С параметрами if – откуда, of – куда, bs – кол во данных, count – сколько порций.



**diff** – сравнение двух файлов или каталогов

-r – рекурсивный обход по папкам

Допустим diff -r -X musor dirr1 dirr2 > d.diff делает сравнение двух каталогов в которых может быть много мусора который выбрасывается при совпадении с musor. Результат записывается в файл \*.diff который видят почти все редакторы linux

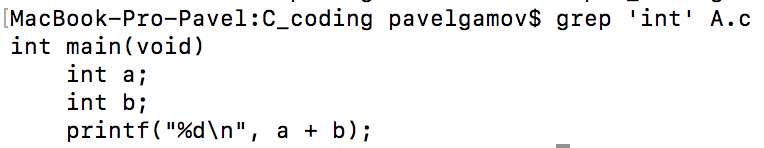


Строчки перед данными показывают какие строчки из каждых файлов надо добавить в другой для того, чтобы они стали идентичными.

Ключ -u показывает информацию об обрабатываемых файлах.

**grep** – global regular expression printer

В чистом виде grep ‘lololo’ /home/pavel/backup/log.txt получает строки которые содержат данный текст



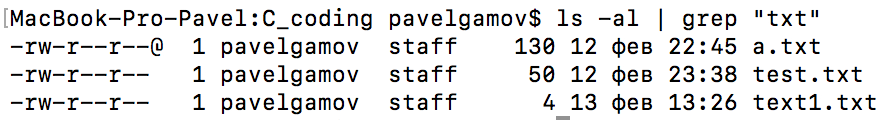
-v те строки в которых наоборот нет этой строки

Или использование grep вместе с потоками ls -l | grep ‘mp3’ вытаскивает из потока все mp3 файлы

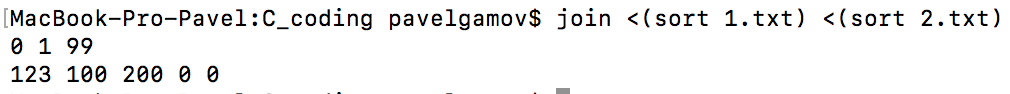
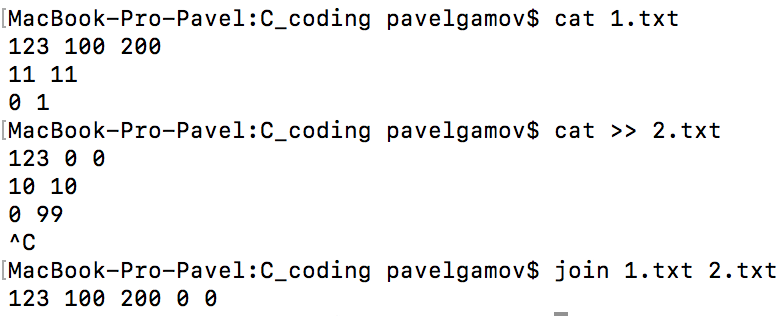
-I – убирает требование к регистру

-w – ищет только целые слова. grep ‘wood’ не найдет ‘Hollywood’.

-c – пишет кол во найденных слов.



**join** – ищет в файле строчки с одинаковыми первыми элементами и компонует их

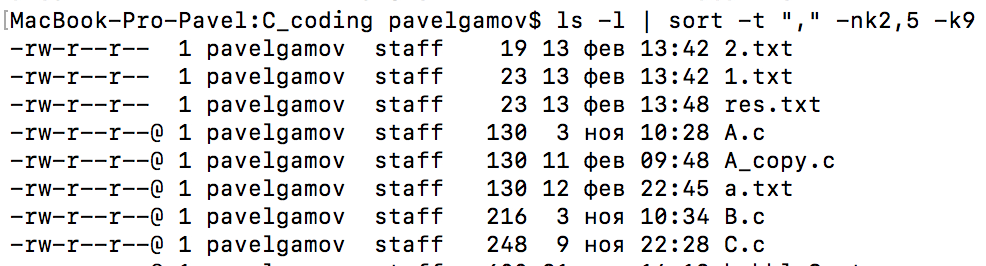


**sort** – сортирует файл по лексике.

-b - не учитывать пробелы. -d - использовать для сортировки только буквы и цифры.

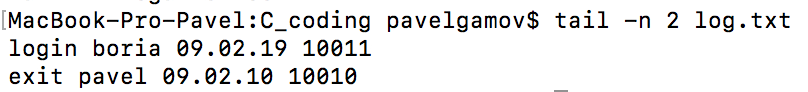
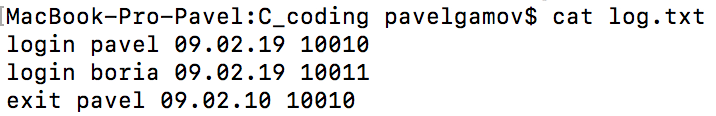
-i - сортировать только по ASCII символах. -r - сортировать в обратном порядке.

-u - удаляет дубликаты.



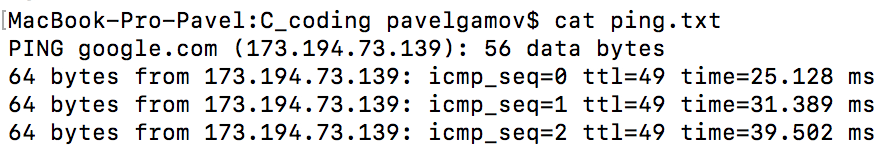
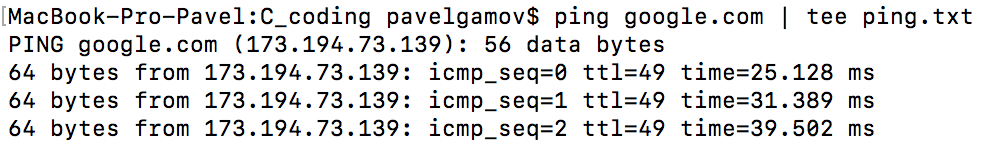
**tail** – выводит остаток файла по умолчанию 10, меняется ключом -n

ключ -f позволяет не заканчивать команду при окончании файла, таким образом можно выводить логи которые обновляются.

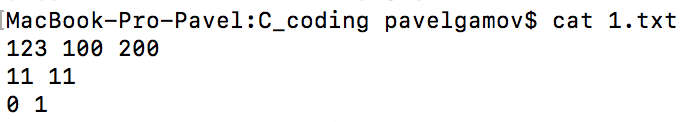


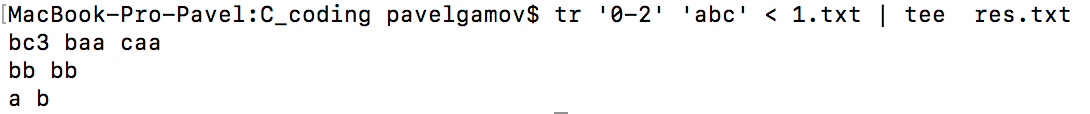
**tee** – позволяет записывать файл в консоль и файл одновременно

ping google.com | tee ping.txt



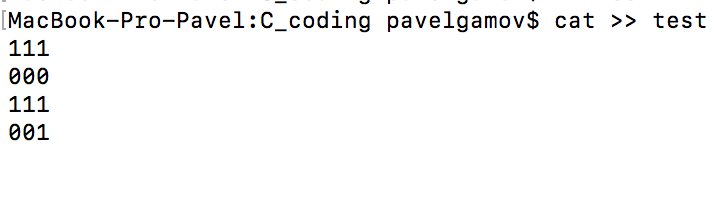
**tr** – заменяет некий символ на некий другой символ.

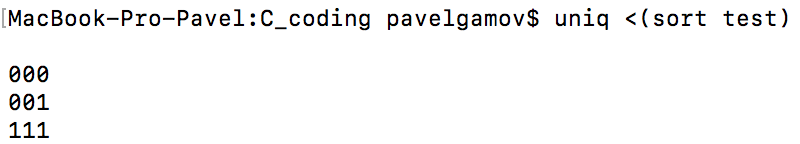




**uniq** - удаляет повторяющиеся строки в том числе разделители.

-f позволяет пропустить n количество первых строк.





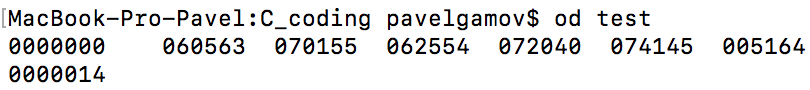
**od** – восьмеричный дамп файла. Печатает однозначное (по умолчанию байтовое восьмеричное) представление файла на стандартный вывод.

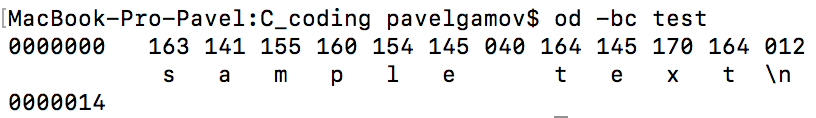
Дампом называется файл с полным или частичным содержимым памяти компьютера или базы данных в момент создания этого файла.

Чаще всего используется дамп памяти. Это такой процесс в результате которого появляется файл с памятью, где находилась та или иная программа, такой файл нужен для определеня ошибки при краше. Допустим синий экран, возникающий при ошибке системы, это и есть дамп памяти при ошибке.

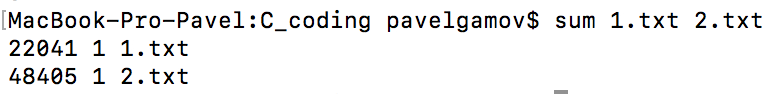
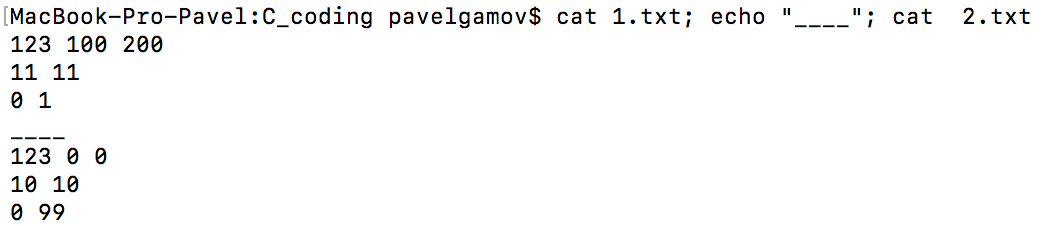
od [- опции] [-A в каком формате будет колонка позиций ] [-j начать с какого байта ] [-N длина ] [файл]





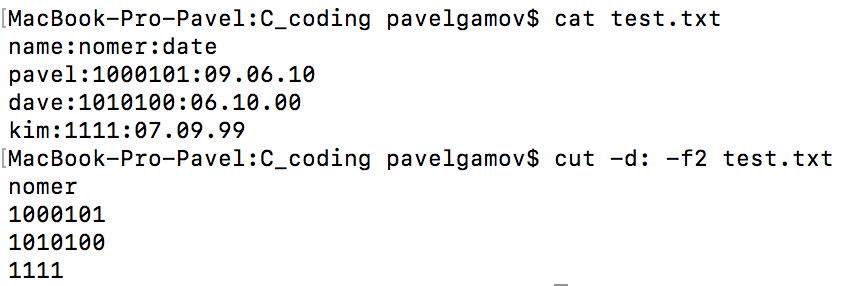


**sum** - подсчет контрольной суммы и количества блоков в файле



**cut** – вырезка нужных элементов и символов в строке

-d# определяет новый разделитель(и), по умолчанию tab. -f# выделяет нужный столбец(ы).

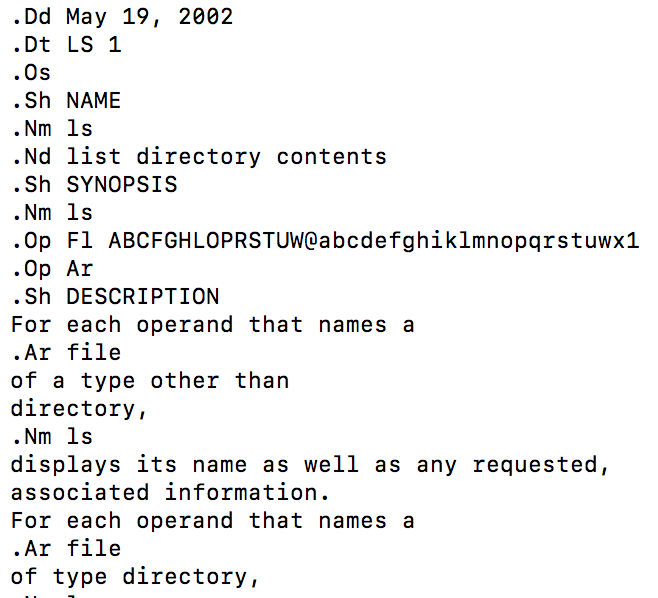


2-5 - будут выведены все байты, символы и поля со второго по пятый.  
-3 - будут выведены все байты, символы и поля до четвертого.  
1,3,6 - будут выведены только первый, третий и шестой байты, символы или поля.

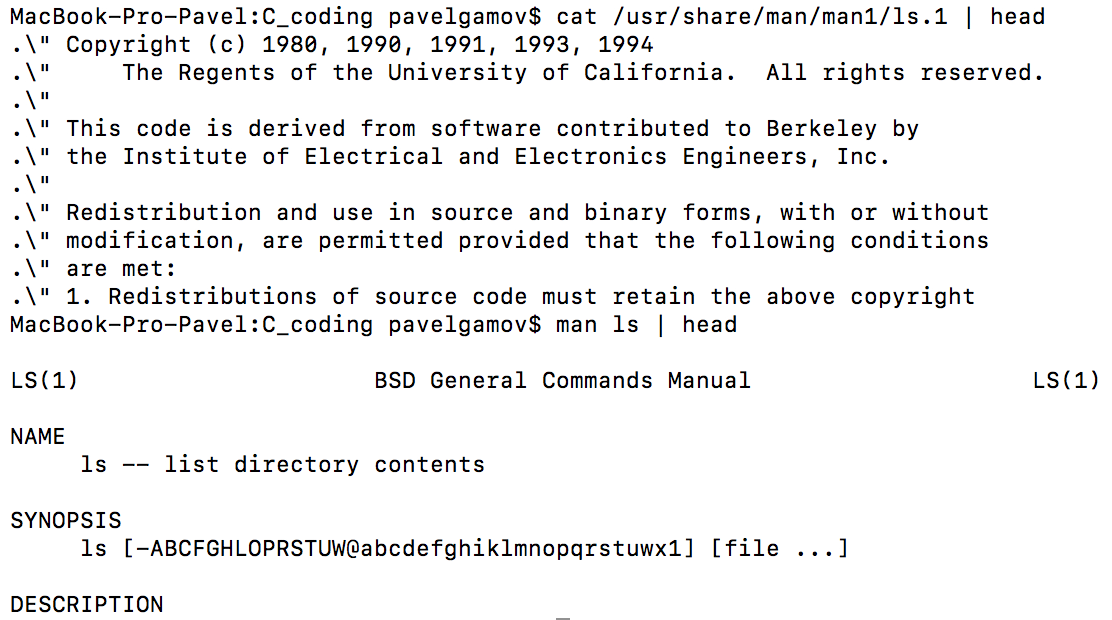
-с определяет срез в байтах от начала строки.

**nroff (groff)** – утилита которая работает автоматом при вызове man. Она преобразует файлы содержащие информацию о функциях и прочую документацию, форматируя ее и выдавая нам удобном виде.





Найдя тот самый файл, который содержит в себе информацию о ls мы видим несколько сотен строк какой-то мешанины.

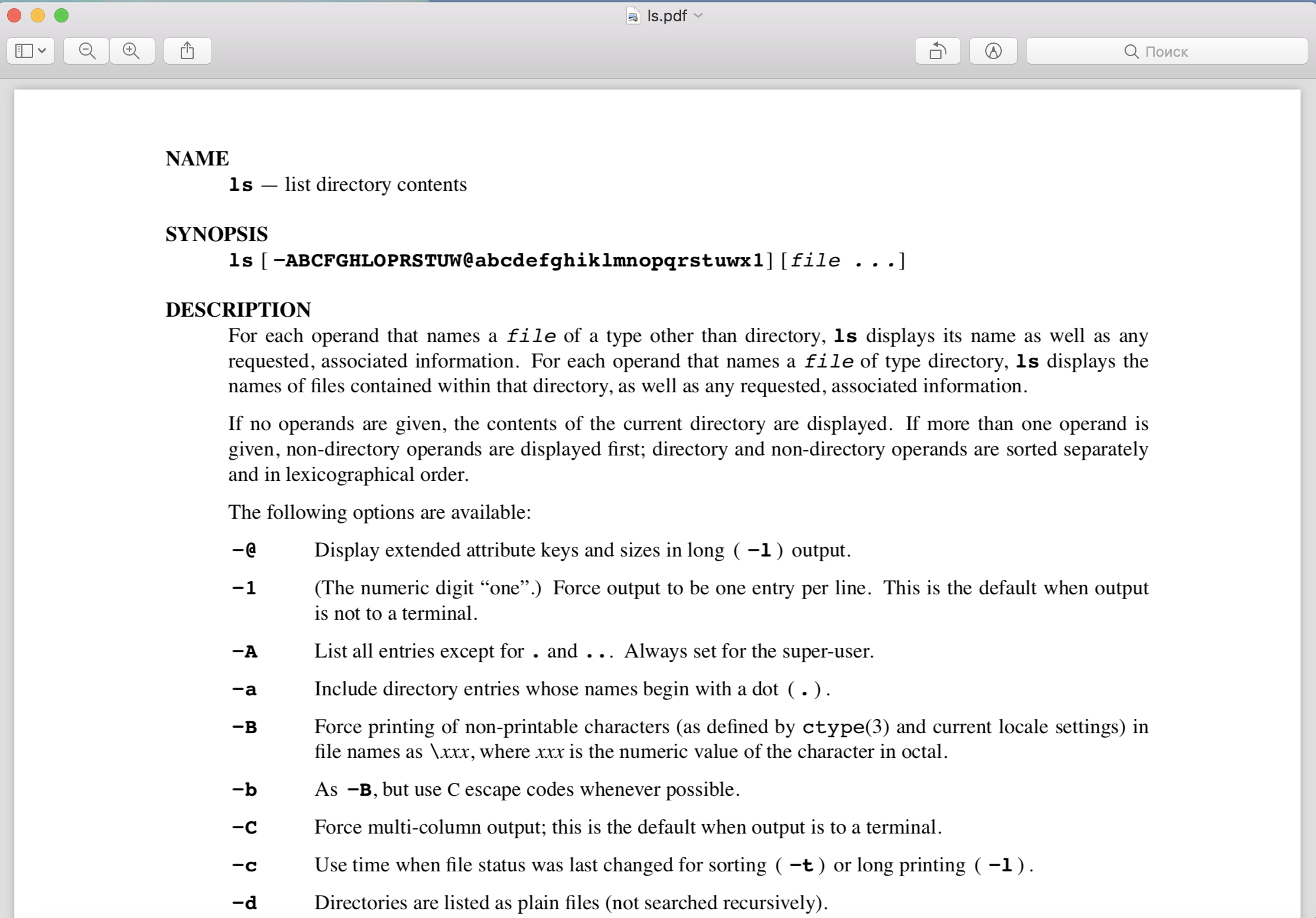


Вызов man дает нам красивую документацию.

cat /usr/share/man/man1/ls.1 | groff -mandoc ascii будет равен выводу обычного man.

Можно сделать очень красиво:

cat /usr/share/man/man1/ls.1 | groff -mandoc > ls.ps



Получили очень красивую книжку документации по ls.

Утилита ps2pdf позволит изменить форматы и подготовить файл к печати.

**Vi/vim** – текстовый редактор vi и его расширенная версия vim уже очень давно являются стандартом в мире Linux. С одной стороны, vim является очень мощным редактором, имеющим огромное количество возможностей, а с другой стороны, vi обычно присутствует в любой системе (даже минимальной, получаемой при загрузке в rescue-режиме с установочного компакт-диска дистрибутива), поэтому в некоторых случаях он может быть единственным имеющимся текстовым редактором.

vi(m) существует в 3 вариантах: vi - небольшой консольный редактор, имеющий немного возможностей; vim - консольный редактор с расширенными возможностями; gvim - графический вариант.

**mc** – один из файловых менеджеров с текстовым интерфейсом типа Norton Commander для UNIX-подобных операционных систем, а также Microsoft Windows и Windows NT, OS/2, Mac OS X (но не Mac OS 9).

Его возможности:

- Работа с различными архивами и образами файловых систем, как если бы они являлись обычными каталогами.

- Работа с сетевыми файловыми системами Samba, FTP, SFTP.

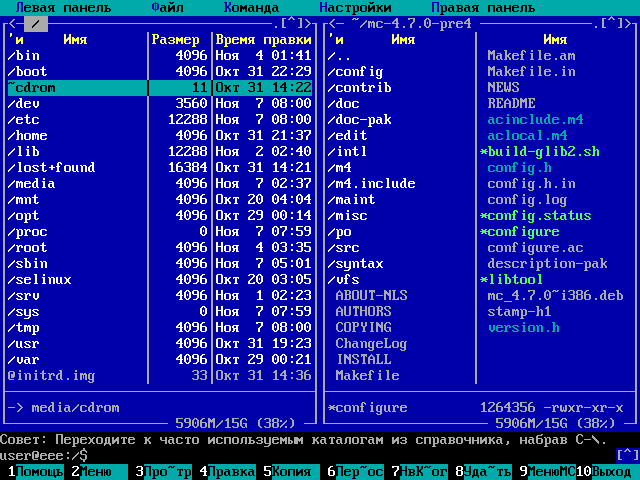
- Выполнение операций копирования и перемещения файлов в фоновом режиме.

- Встроенный текстовый редактор с подсветкой синтаксиса.

- Поддержка многоязыкового интерфейса.

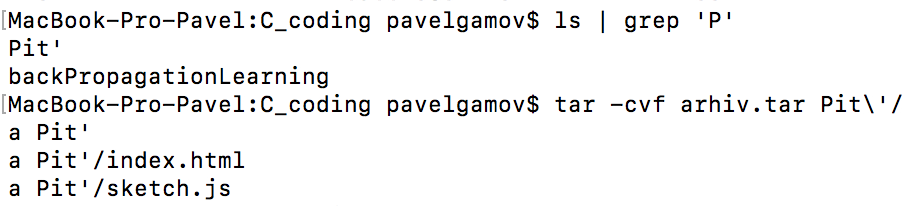
- Поддержка кодировки UTF-8.

- Настраиваемое пользовательское меню, вызывается по F2. Каждый пункт меню выполняет набор команд оболочки (shell).

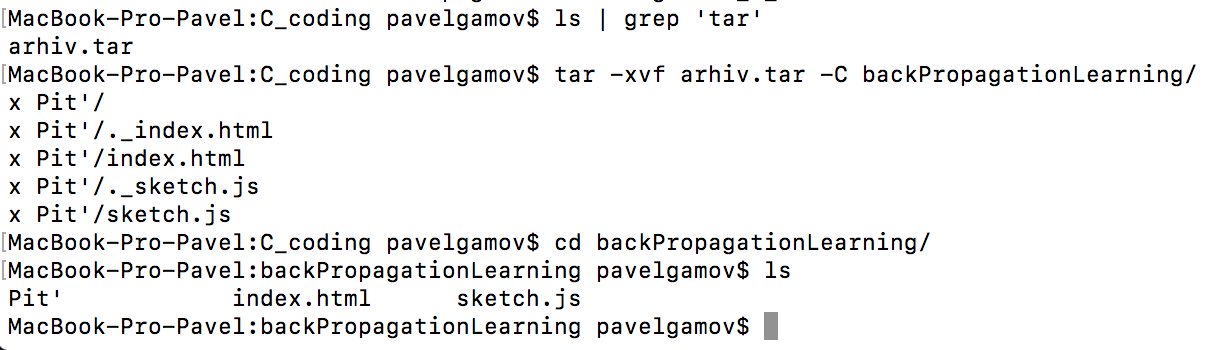


**tar** –создание не сжатого архив, который используется большим количеством системных администраторов Linux для работы с накопителями для резервного копирования.

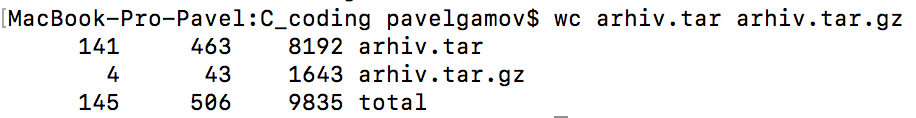
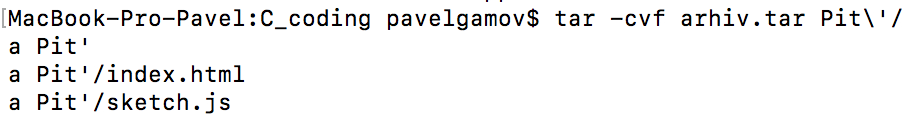
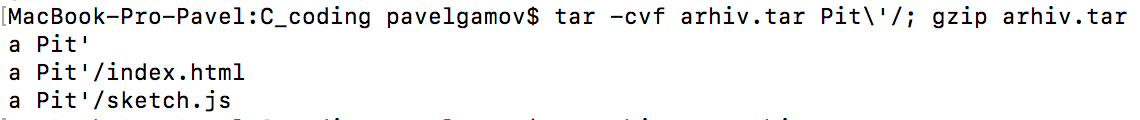
-с – создание нового архива, -v – показывается прогресс, -f – тип имени файла архива.



Распаковка в директорию с флагом -C и -х.



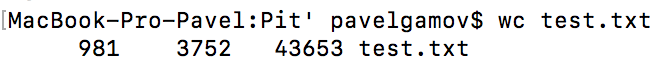
**gzip** – архиватор в привычном плане, не работает с папками, эффективен вместе с утилитой tar.



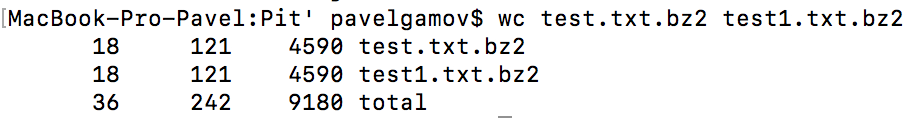
**bzip2** – модифицированная версия архиватора, еще сильнее позволяет сжать данные. Требует больше оперативной памяти в общем, а также больше времени при разархивировании.

Также работает с tar и имеет степени сжатия от -1 до -9 (минимум максимум соответственно).

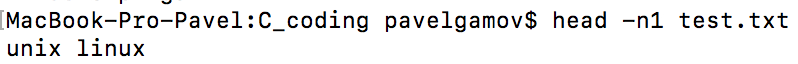
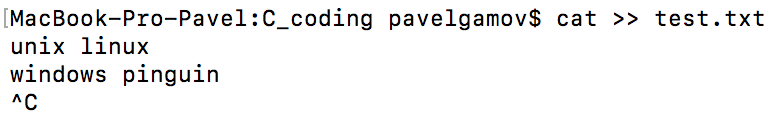
-d – разархивирование. -к – сохранение оригинала.







**head** – выводит первые n строк (-n #) файла, или первые n байт (-с #) файла.

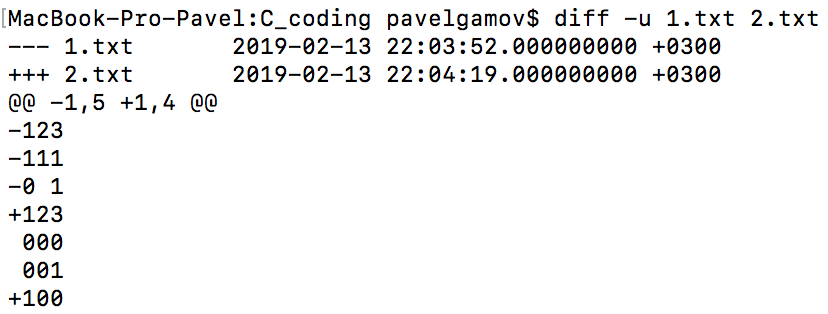


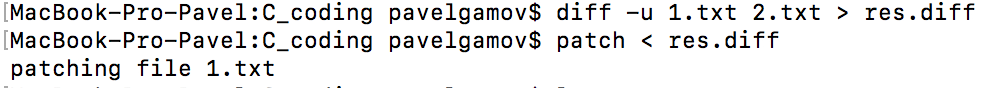
**iconv** – утилита перекодировки файла. --list позволяет узнать какие кодировки доступны.

Синтаксис iconv -f <from> -t <to> <filein> -0 <fileout>.

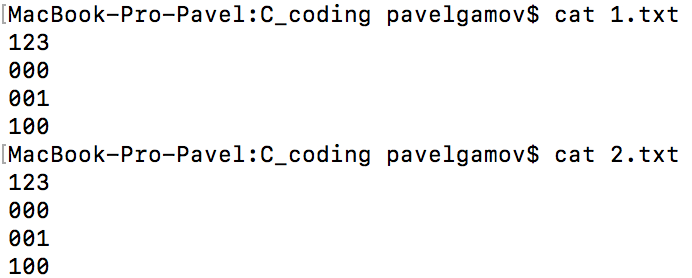
**patch** – в мире Linux широко используется пара программ diff и patch. diff создает файл, в который записываются различия между 2 файлами или 2 каталогами (т.н. патч), а patch позволяет "наложить" этот патч, т.е. имея патч и один из двух файлов (каталогов), получить другой.

Как пример ядро Linux, упакованное в .tar.bz2, весит более 30 Мб независимо от его версии. Но файл различий между версиями 2.6.6 и 2.6.7, упакованный в .bz2, весит всего 3 Мб, поэтому для получения исходников ядра 2.6.7 из исходников ядра 2.6.6 достаточно скачать лишь 3 Мб, не выкачивая все 32 Мб заново.



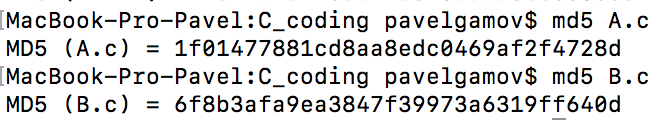


Создание патча.



Таким образом мы создали патч разницы файлов 1 и 2 и пропатчили первый, то есть изменили его.

**md5** – 128-битный алгоритм хеширования. Широко применялся для проверки целостности информации и хранения хешей паролей.



**du** – отображает размер дискового пространства, занятого файлами или каталогами.

–a выводит информацию о том сколько памяти занимает каталог и все ее внутренние файлы.

–b выводит информацию в байтах (а не в килобайтах).

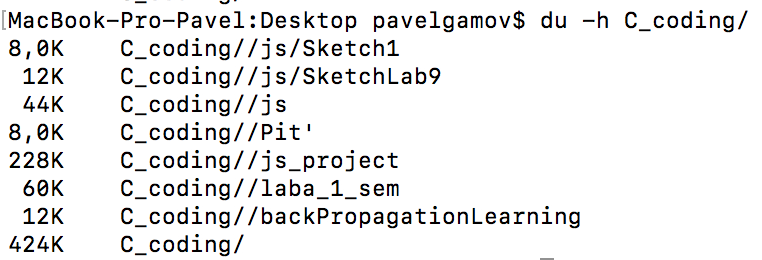
–c выводит итоговую информацию об использовании дисковой памяти.

–h вывести информацию в человекочитаемом виде.

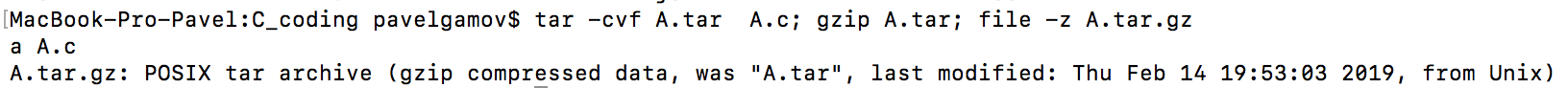
–m выводит общую занимаемую память.

–с выводит общий размер в отдельную строку.

--time скажет, когда был изменен каждый файл.



**file** – утилита, нужная для определения типа данных в файле, в том числе и в сжатых архивах



**touch** – утилита позволяющая изменять дату и время последнего изменения файла. Также создает пустой файл.

# touch -t 200701310846.26 index.html

# touch -d '2007-01-31 8:46:26' index.html

# touch -d 'Jan 31 2007 8:46:26' index.html

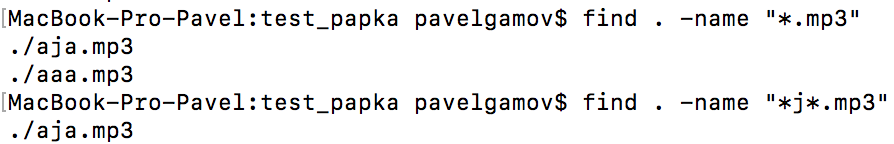
Различные форматы ввода даты и времени.

**find** – команда позволяющая находить различные файлы по различным параметрам в папках и каталогах.

. - поиск в текущем каталоге; / - поиск от корневого каталога;

~ - поиск в домашнем каталоге. -name – поиск файлов по имени, используя приведенный шаблон;

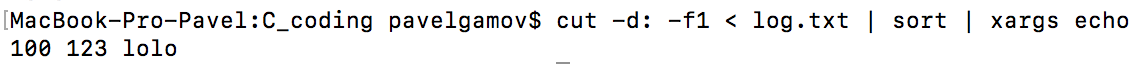
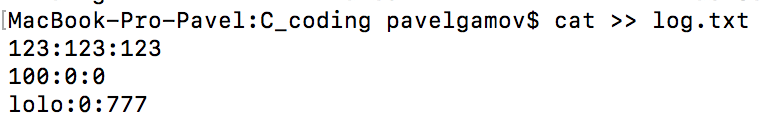




**xargs** – утилита имеет интересный принцип работы, не обрабатывает файлы, а перенаправляет их из входного потока и направляет в другие программы с нужными параметрами.

find . -name "\*.sh"| xargs rm – позволяет удалить все файлы в папке с расширением sh.

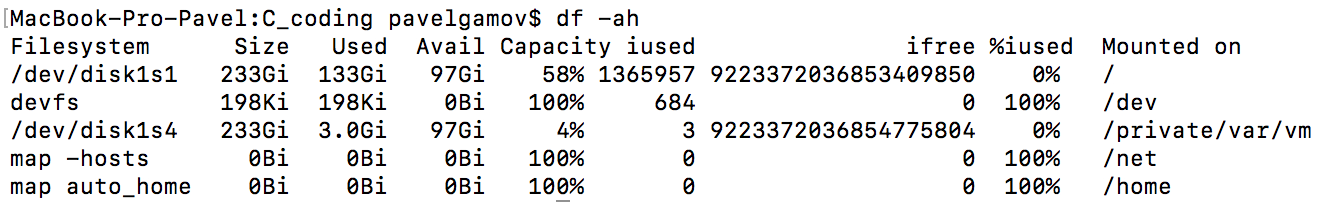
cut -d: -f1 < log.txt | sort | xargs echo – позволяет вывести из лога первый столбик, отсортировать и вывести в стандартный поток вывода.



echo dir1 dir2 dir3 | xargs -P 3 -I NAME tar czf NAME.tar.gz NAME

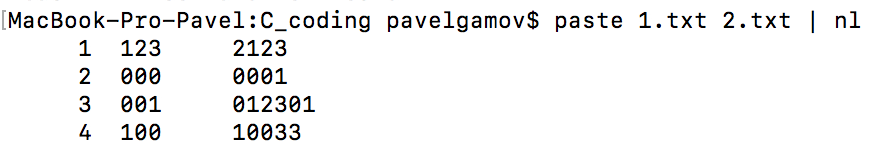
такая сложная команда позволяет параллельно запаковать 3 каталога под их именами. В данном случае ключ -P n показывает сколько у нас циклов. Иногда, как тут, утилиту xargs можно сравнить с обычным циклом for.

**df** – утилита позволяет узнать размер свободного пространства области памяти.



-a показывает все диски. -h в читабельном виде.

**paste** – позволяет вывести бок о бок два файла построчно с разделителем tab. Есть конечно команда diff но вот так.



Команда nl# нумерует строчки.

**сpp** – препроцессор для языков семейства Си. Препроцессор – программа, подготавливающая код программы к компиляции. Выполняет лексический анализ кода, выполнение подстановок и прочее. Генерирует ошибки и предупреждения.

Так как язык может обновляться, утилита может вести себя иногда неадекватно. Язык препроцессора не является полным по Тьюрингу хотя бы потому, что с помощью директив невозможно заставить препроцессор зависнуть.

**indent** – утилита позволяющая изменить файл содержащий код C/С++. В зависимости от параметров утилита добавляет пробелы/табы, удаляет лишние пустые строки. Обладая большим потенциалом, имеет как короткие флаги, так и их длинные синонимы. Список их очень велик. Поэтому лучше создать файл, в который можно записать все необходимые флаги и просто запускать их для нужного файла.

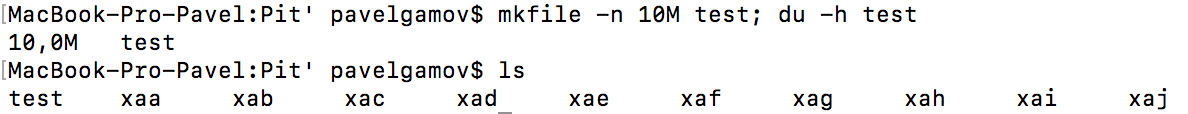
**split** – утилита разрезающая файлы на части. -b – эта опция задает размер выводимых файлов. -n – задает количество выводимых файлов.

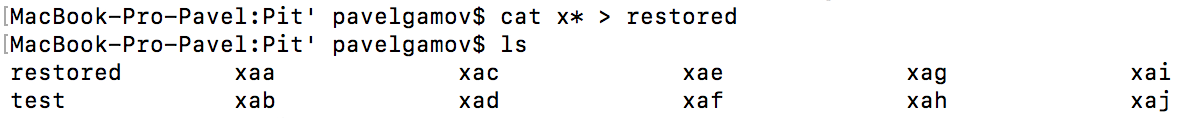


Создание пустого файла размером 10мб (буква М прописная).

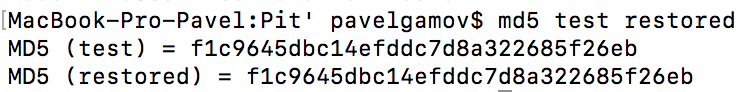


Разделение файла на меньшие, размером не больше 1мб (тут уже м строчная).





Восстановление файла через команду cat.



Сравнение контрольных сумм, файлы идентичны.

**mktemp** – создание временного файла с случайным именем. Нашел пример для этой команды.

Допустим вы программист и пишете код echo something > /tmp/temporary-file.

Кто-то подходит и пишет строчку привязки вашего файла к своему.

ln -s /etc/passwd /tmp/temporary-file. Все сливается к нему в файл.

Такая запись поможет избежать подобной проблемы давая файлу случайное имя.

TEMP=$(mktemp /tmp/temporary-file.XXXXXXXX)

echo something > ${TEMP}

В любом случае найденные ошибки говорят о том, что иногда функция не дает нужный результат, поэтому необходимо использовать mkstemp().