Модуль 1. Знакомство с UNIX

История ОС

Multics (англ. Multiplexed Information and Computing Service) — одна из первых операционных систем с разделением времени исполнения программ. Изначально Multics была разработана для 36-битных мэйнфреймов GE-645.

Про ОС http://www.multicians.org/

Исходники: http://web.mit.edu/multics-history/

Литература:

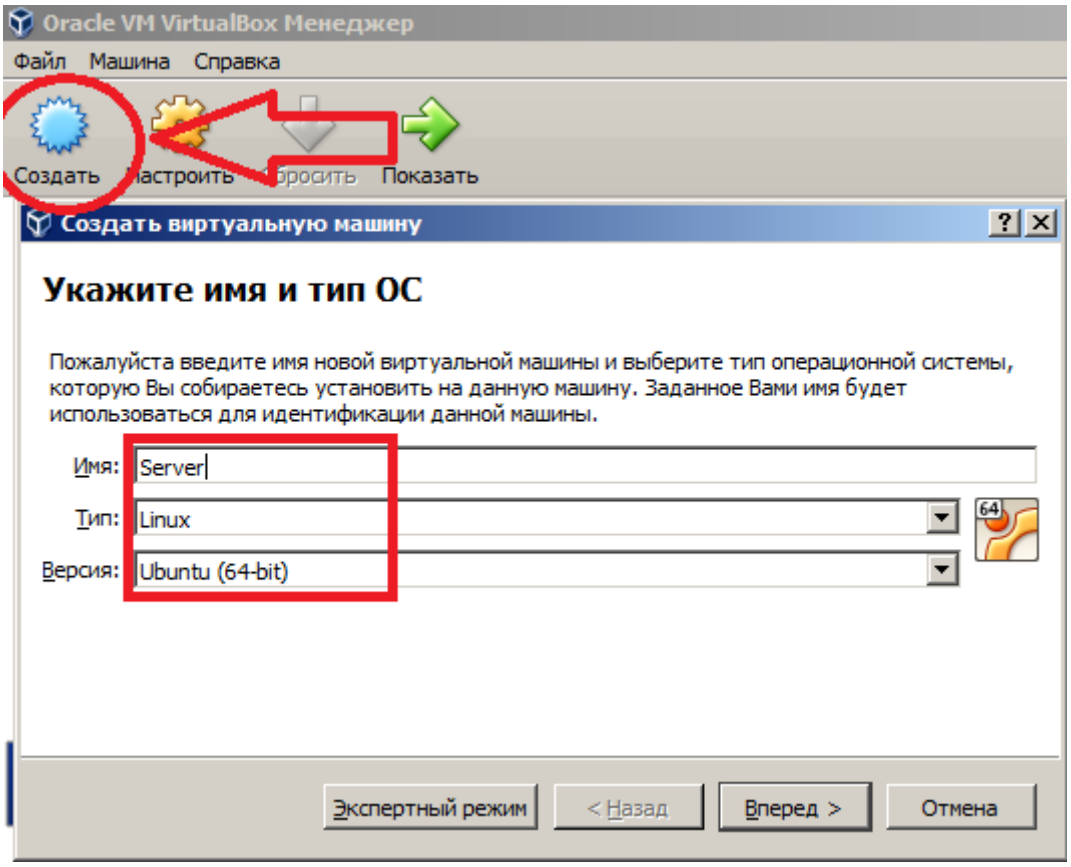
Скотт Граннеман «Linux - карманный справочник»

Андрей Робачевский. «Операционная система UNIX»

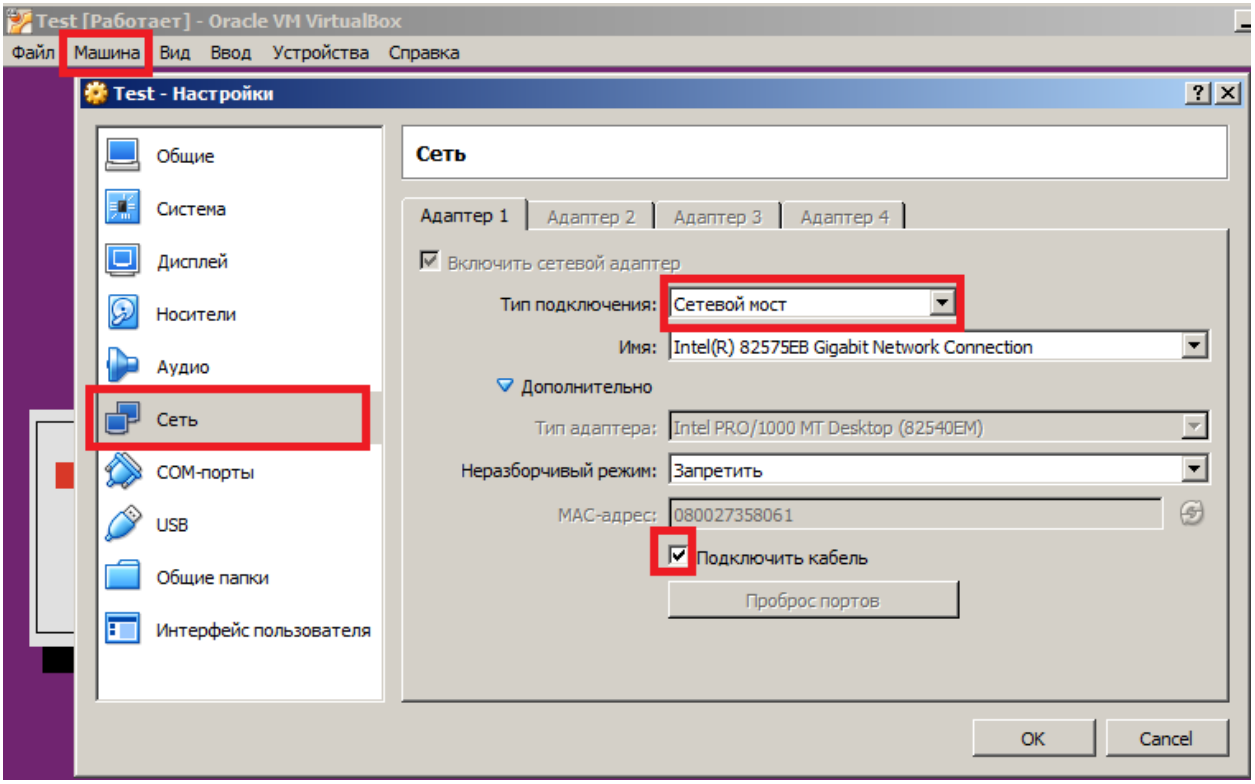
Э. Таненбаум. «Современные операционные системы»

Установка ОС на виртуальную машину.

1. **Создайте виртуальную машину в VirtualBox:**

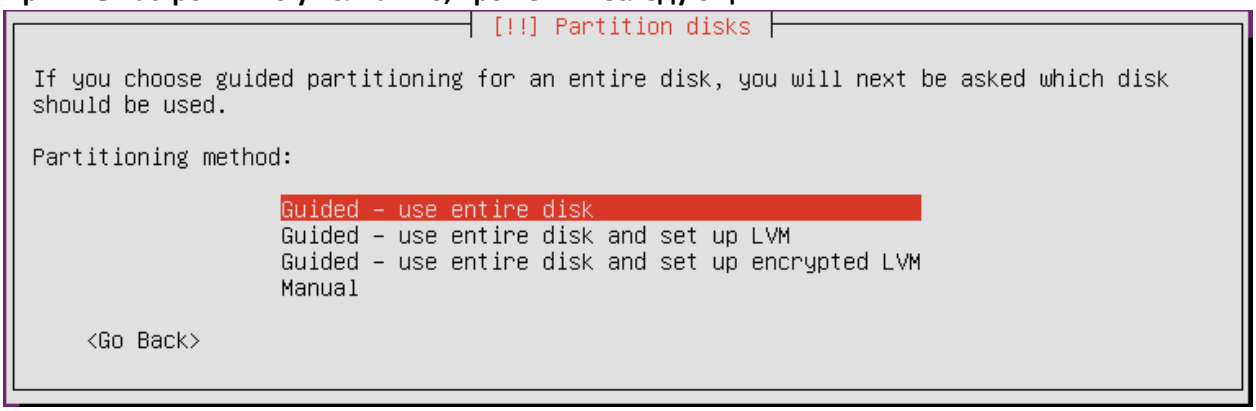


2. **Задайте объем памяти 1024 (2048 если позволяют ресурсы хоста).**
3. **Создать новый диск – тип VDI, формат: Динамический. Размер 127 Гб**
4. **Настроить сетевой адаптер:**

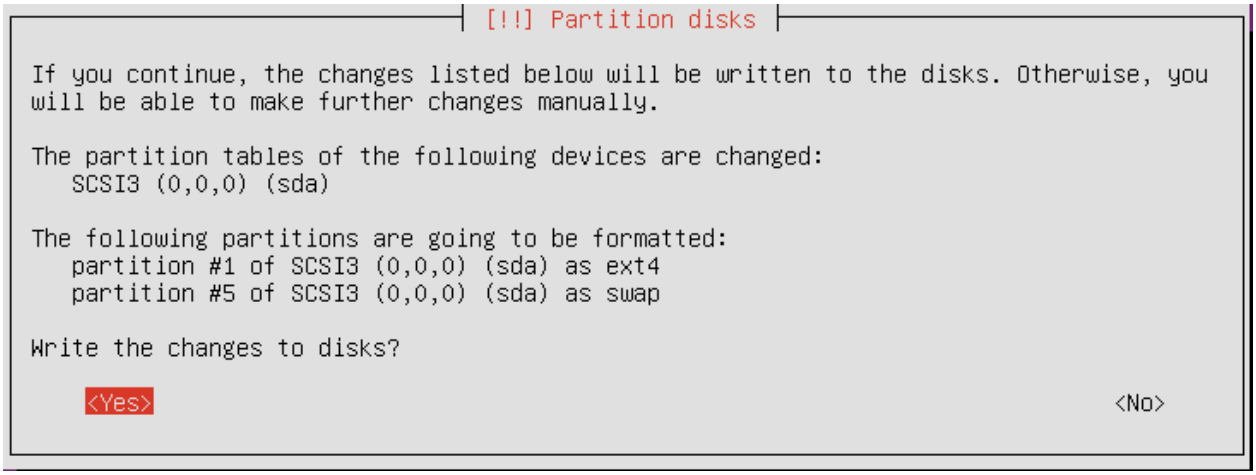


5. **Запустить виртуальную машину и указать путь к файлу ubuntu-16.04.2-server-amd64.iso:**

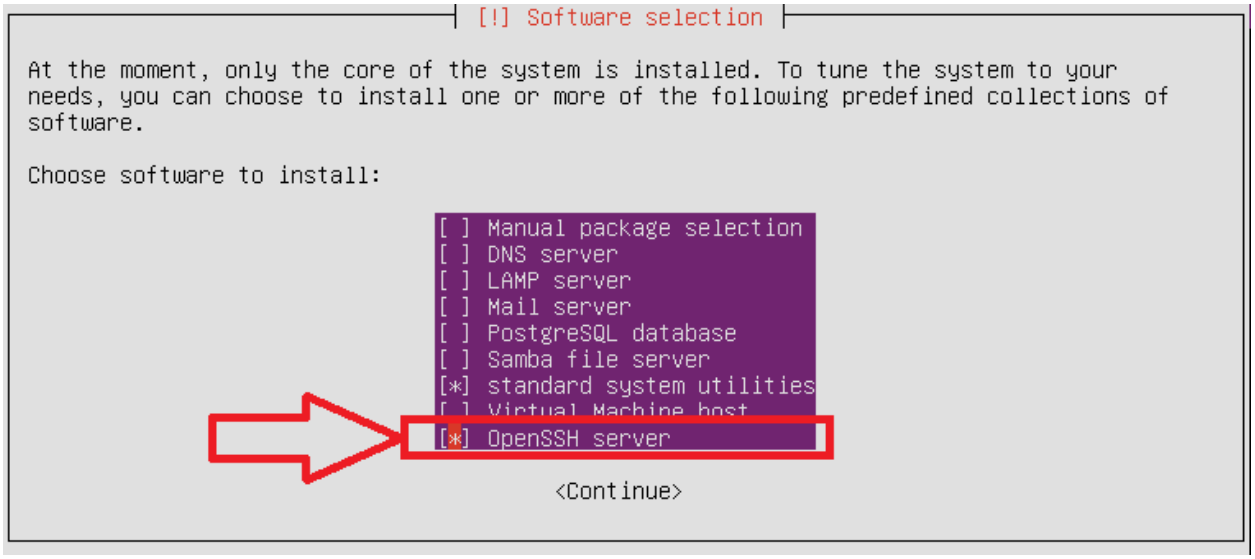
6. **Принять настройки по умолчанию, кроме нижеследующих:**



7.



8. **Пробелом поставить * в позиции установки OpenSSH Server:**



9. **После перезагрузки войти в систему и произвести настройку сети:**
   Просмотреть доступные интерфейсы:
   Команда ip a

Настройка адреса через конфиг:

vi /etc/network/interfaces

добавить в конец файла:

auto enp0s3

iface enp0s3 inet static

address 172.16.1.X

netmask 255.255.255.0

gateway 172.16.1.254

dns-nameservers 172.16.1.254

dns-search corpX.un

Применить без перезагрузки:

ifdown enp0s3 && ifup enp0s3

Структура команд

 # структура команд и формат ключей.

Команда –ключ (или опция) аргумент (над чем производится действие)

Напр #find --help

Регистрозависимая ОС

who

who | wc –l

Документация

Для чтения на русском добавить переменную в ~/.bashrc

export MANOPT="-L ru"

или

man -L ru man

если есть соотв мануал на русском (см. в /usr/share/man/ru/man1)

чтобы определить, какая файловаяw система на разделе /dev/sda1, наберите команду file с ключом -s:

sudo file -s /dev/sda1

 # df -T - показывает тип файловой системы.

df -h *отображает информацию о смонтированных разделах с отображением общего, доступного и используемого пространства*

/dev/sda1

sd [abcd] [12345] – диск – экземпляр – раздел

Дисковое пространство принимается за 110%

du /var *- подсчитывает и выводит размер, занимаемый директорией*

du -h -s /var 2>/dev/null

du --max-depth=1 /usr/share/

 # ls

отобразить содержимое указанной директории

pwd

.имяфайла - скрытые файлы

Какими командами вывести сведения, когда создавался, менялся, был прочитан файл:

ls -l --time=ctime testfile - когда создавался, менялся

ls -l --time=atime testfile - когда был прочитан

- регулярные файлы

d – директории

Каталоги

pwd - показать текущую директорию

drwxr-xr-x . –текущий каталог

drwxr-xr-x . . – родительский каталог

ls -l ..

Переходы между директориями:

cd .. перейти в директорию уровнем выше

cd ../.. перейти в директорию двумя уровнями выше

cd перейти в домашнюю директорию

cd ~user перейти в домашнюю директорию пользователя user

cd - перейти в директорию, в которой находились до перехода в текущую директорию

cd /user/user1

cd /../../bin/bash

ls -a ~ (~ - домашний каталог)

id

whoami

mkdir

ls -l

rm – файла

rm -r

rmdir – пустого каталога

cp что куда

cp /etc/passwd .

ls

mv что куда

Переименование

Перемещение

перемещение под другим именем.

Создание ссылок в Linux

 # ln файл ссылка

 # ln file1 file2

Символьную ссылку можно создать при помощи команды **ln** с ключом **-s** (от "symbolic"). Например:

ln -s /root/lnfile1 /var/softlfile1

Поиск всех жестких ссылок файла (по inode)

touch file1

ln file1 /var/file2

ls -li

5644346 *-rw-r--r--* ***2*** *root root 0 Sep 27 21:39* ***file1***

find / -inum 5644346 2>/dev/null

Программы поиска файлов:

which – показывает путь к каталогу указанного файла

which vi

locate - *позв. найти файл, указанный по имени*

locate hello

cat *(concatenation, firstly, In pair with split command)*

сегодня используется для вывода на экран содержимого файла

tac *file1 -вывести содержимое файла file1 на стандартное устройство вывода в обратном порядке (последняя строка становиться первой и т.д.)*

Пример создания и заполнения файла командой cat:

~# cat > file1

12345

^C

~# cat >> file1

23456

^C

~# cat file1

~# tac file1

split *разбивала файл на дискеты по 1440, cat потом собирала файл:*

Пример:

 # dd if=/dev/zero of=bigfile1 bs=1024 count=20000

$ split -a 1 -d -b 2M bigfile1 bigfile1.part

 # rm bigfile1

$ cat bigfile1.part > bigfile1.iso*

*# rm bigfile1.part**

more - скроллинг только вниз

more /etc/passwd

less - скроллинг вверх и вниз, PgUp / PgDwn работает

Например, man форматирует для вывода на устройство, a less выводит

Выход из команды: q

Для просмотра логов - tail. По умолчанию, выводит 10 последних строк. Можно переопределить параметром –n.

tail -n 20 /var/log/auth.log

tail -f -n 0 /var/log/auth.log

выводим на экран 0 старых записей (-n 0) и ждем появления для отображения новых (-f).

Для разделения результатов вывода м. исп. ========== “Enter”

Для выхода исп. Ctrl + C – сигнал «завершить работу»

Противоположно предыдущей команде, head выводит начало файла.

Работа с историей команд (history)

Ctrl + r – шаблон поиска – поиск по истории

для просмотра списка ранее введенных команд в bash — имеется команда history. По умолчанию все пишиться в файл ~/.bash_history, а его размер — **500** команд.

Если хотим хранить историю в другом файле, то нужно в .bashrc, задать команду
HISTFILE=~/.my_history.
HISTSIZE — определяет число строк, хранящихся в списке истории (в памяти интерпретатора).
HISTFILESIZE — максимальное количество команд хранящихся в файле

$ export HISTSIZE=1000

$ export HISTFILESIZE=1000

возможность указать
количество выводимых строк (команд):

$ history 20

все команды имеют номер, с помощью которого к ней можно обратится.
Если нам надо повторить 28 команду, то просто набираем в терминале:

$!28

Список наиболее распространенных команд:

!! — ссылается на предыдущую команду;

!n — ссылается на команду под номером n;

!-n — ссылается на команду по номером „текущая минус n“;

history -c — очистить историю команд, удалив все записи

history -d n — удалить из истории запись под номером n

history -a — дописать команды, введенные в текущей сессии bash, в конец файла $HISTFILE

Так же можно сохранить дату и время для каждой команды в истории, для этого в конец .bashrc дописываем:
$ vi .bashrc

```
export HISTTIMEFORMAT="%h/%d-%H:%M:%S"
```

Мод. 3. Регулярные выражения. Редакторы.

Регулярные выражения

 # man re_format

Общая схема регулярного выражения

регулярное выражение состоит из трёх основных частей:

1. Якорь – определяет позицию шаблона в строке текста:
    - ^ – якорь, определяющий начало строки;
    - $ – якорь, определяющий конец строки.
2. Шаблон - набор (последовательность) символов – для поиска соответствий в заданных позициях строки текста:
    - символ "точка" (.) соответствует любому произвольному символу;
    - алфавитно-цифровые символы и пробел представляют сами себя;
    - прочие символы – интерпретация зависит от диалекта.
3. Модификатор – задаёт количество повторов предыдущего символа или набора символов (в зависимости от диалекта):
    - * – любое количество повторов символа/набора, в том числе и нулевое;
    - ? – соответствует нулю или одному экземпляру символа/набора;

Символы базовых регулярных выражений:

^ - соответствует началу строки;

Пример: grep '^s' /etc/passwd

$ - соответствует концу строки;

Пример: grep 'sh$' /etc/passwd

[] - любой символ из числа заключенных в скобки (символы могут быть разделены запятыми, указание диапазона - [0-9]);

Пример: [012345789] – соответствует одному цифровому символу из заданного набора;

Предназначены для задания подмножества символов. Квадратные скобки, внутри регулярного выражения, считаются одним символом, который может принимать значения, перечисленные внутри этих скобок. Метасимвол ^ означает отрицание множества:

[^] - любой символ кроме тех что указаны в скобках;

Пример: grep '^[rs]' /etc/passwd

\ - Служит для экранирования специальных символов, т.е. отменяет спец. значение следующего за ним метасимвола;

Пример: grep 'bin\/sh' /etc/passwd

-- \<...\> -- Экранированные "угловые скобки" - задают границы слова (не работает в sed).

Пример: grep -R '\<sed\>' /usr/share

. - Означает не менее одного любого символа;

* - Означает любое количество символа в строке, предшествующего «звездочке», в том числе и нулевое число символов;

Например, имеется шаблон для поиска любого количества символов, заключённых в кавычки:

".*"

-- \(\) --Экранированные "круглые скобки"

Предназначены для выделения групп регулярных выражений. Они полезны при использовании с оператором «\|» и при извлечении подстроки.

-- \{ \} -- Экранированные "фигурные скобки"

Задают число вхождений предыдущего выражения. Для уточнения количества повторений наборов символов применяется модификатор \{min,max\}.

Пример: grep '\(ro.*\)\{2\}' /etc/passwd

Пример: ip a | grep '[1-9][0-9]\{0,2\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}'

> Зная, что в каждой части IP-адреса может содержаться от одной до трёх цифр, запишем модификатор в виде \{1,3\}. Символы "обратный слэш" перед точками необходимы для того, чтобы отменить их специальный статус универсального метасимвола.

\{n\} - указывает на то что предыдущий шаблон встречается ровно n раз;

\{n,\} - указывает на то что предыдущий шаблон встречается не менее n раз;

\{,m\} - указывает на то что предыдущий шаблон встречается не более m раз;

\{n,m\} - указывает на то что предыдущий шаблон встречается не менее n раз и не более m раз;

n|m - выбор из двух шаблонов

\> - признак конца слова;

Классы символов:

[:upper:] - [A-Z];

[:lower:] - [a-z];

[:digit:] - [0-9];

[:alnum:] - [0-9a-zA-Z];

[:space:] - символ пробела;

[:alpha:] - [A-Za-z];

Пример: # grep [[:upper:]] /etc/passwd

*grep [параметры] регулярное_выражение [файл]

-c - задает отображение только числового значения, сколько строк соответствует шаблону;

-i - игнорировать регистр;

-h - подавляет вывод имен файлов, включающих найденные строки;

-l - только отображение имен файлов, содержащих найденные строки;

-n - нумерация выводимых строк;

-s - подавляет вывод сообщений о несуществующих или нетекстовых файлах и ошибках;

-v - отображение строк, не соответствующих шаблону;

-E - включение расширенных регулярных выражений;

POSIX делит регулярные выражения на две категории:

BRE (Basic Regular Expressions) и ERE (Extended Regular Expressions).

В обеих категориях поддерживаются метасимволы . и *, якоря ^ и $, группирование символов в скобках (для BRE скобки экранируются обратным слэшем), применение квантификаторов \{min,max\} к группам в скобках. Запоминание и повторное использование \1...\9 поддерживает только категория BRE, а квантификаторы + и ? и конструкцию выбора – только категория ERE.

Символы расширенных регулярных выражений

Многие символы, экранируемые в базовых выражениях – () {} | – но не – <> – используются без экранирования.

Знак вопроса -- ? --

Означает, что предыдущий символ или регулярное выражение встречается 0 или 1 раз.

$ grep -E '^r?o' /etc/passwd

Знак "плюс" -- + --

Указывает на то, что предыдущий символ или выражение встречается 1 или более раз

Как вывести из конфига только незакомментированные строки? Использовать **рег выражение**:

grep '^#' /etc/ssh/sshd_config (спец символы засовываем в ‘’)

grep '^#' /etc/ssh/sshd_config (^# начинаются с #)

grep -v '^#' /etc/ssh/sshd_config (-v начинаются не с #)

grep -v '^#\|^$' /etc/ssh/sshd_config (при отрицании -v «и» (оператор «**\|**»)меняется на «или» , т.е. не содержат пробелы ^$)

Вывести общее кол-во папок в указанной директории:

ls -l ~ | grep "^d" | wc -l

Потоковые редакторы

Потоковый редактор sed

Формат команды

sed команды_редактирования [имя_файла]

-i – редактировать файл

sed -i 's/string1/string2/' example.txt	в файле example.txt заменить "string1" на "string2", результат вывести на стандартное устройство вывода.
sed -i '/^$/d' example.txt	удалить пустые строки из файла example.txt
sed -i '/ *#/d; /^$/d' example.txt	удалить пустые строки и комментарии из файла example.txt
-e – добавить к команде скрипт	
sed -e '1d' example.txt	удалить первую строку из файла example.txt
sed -n '/string1/p' example.txt	отобразить только строки содержащие "string1"
sed -e 's/ *$//' example.txt	удалить пустые символы в конце каждой строки
sed -e 's/string1//g' example.txt	удалить строку "string1" из текста, не изменяя всего остального
sed -n '1,8p;5q' /etc/passwd	взять из файла с первой по восьмую строки и из них вывести первые пять
sed -n '5p;5q' /etc/passwd	вывести пятую строку
sed -e 's/0*/0/g' /etc/passwd	заменить последовательность из любого количества нулей одним нулём

Еще пример:

 # g > output.txt

 # sed -e 's/\(.*\)/\U\1/' sshd_config > output.txt

Утилита Tr

Команда tr служит для перевода выбранных символов в другие символы или удаления их. **tr не принимает имен файлов в качестве аргумента.**

echo 'test' | tr '[:lower:]' '[:upper:]' преобразовать символы из нижнего регистра в верхний

 # cat sshd_config | tr '[:lower:]' '[:upper:]'

Редактор Vi

Vi – есть везде.

vimtutor – встроенный учебник

Cp /etc/passwd ~

Cd

Vi passwd

J - вниз

K – вверх

L – вправо

G – в начало на первый символ

Редактирование текста

Режим вставки

i - ввод текста с текущей позиции

o - ввод текста с новой строки

Командный режим

J - склеить строки

x - удалить текст (DEL)

X - удалить текст (BACKSPACE)

yy - копировать строку в буфер

dd - вырезать строку в буфер

p - вставить строку из буфера под выбранной строкой

P - вставить строку из буфера над выбранной строкой

u - отменить последнее действие

Командный режим – во всем тексте замени значение X на 1

:%s/X/1/g

Редактирование конфигов

Перед редактированием конфига, его оригинал нужно сохранять

Кроме cp /etc/ssh/sshd_config /etc/ssh/sshd_configXXXXXX

Можно и лучше использовать систему контроля версий (только ее нужно сначала установить

apt-get install rcs

RCS - это Система Управления Исправлениями (revision control system)

RCS включает в себя следующие программы:

rcs, которая управляет атрибутами архивного файла RCS;

ci и **co**, проверяющие старые и измененные архивы RCS;

ident, которая производит поиск в архивах RCS по ключевому слову;

rcsclean, программа которая удаляет нерабочие или неизмененные файлы;

rcsdiff, которая запускает **diff** для сравнения версий;

rcsmerge, которая объединяет результаты работы двух пользователей над файлом в один работающий файл;

rlog, которая выводит сообщения из журнала RCS.

Создать каталог для репозитория

mkdir RCS

Затем импортировать файл:

ci testfile

Исходный файл _перемещается_ в репозиторий (если он там уже есть, то под новой версией).

[ci -l /etc/ssh/sshd_config – запомнить состояние файла.]

Предложит ввести комментарий, если идей нет просто ставим “. “

Извлечь файл из репозитория можно командой:

co testfile

(файл будет иметь права доступа 444)

Чтобы изменить файл, нужно установить его блокировку и установить права доступа, разрешающие запись

rcs -l testfile

chmod o+w testfile

cat >> testfile

23232323

Правим файл, смотрим рез:

rcsdiff testfile – смотрим, что (на что) поменялось.

Строка - измененная строка (1a2)

1a2 – первой добавилась вторая

> 23232323 – что добавилось (операция)

Чтобы записать изменения нужно снова выполнить

ci testfile

rlog ./testfile – сколько и какие версии и ревизии версий

что поменялось относительно указанной версии:

co testfile

rcsdiff -r1.1 testfile

что поменялось между указанными версиями:

rcsdiff -r1.1 -r1.2 testfile

Мод. 4. Процессы

Упр сервисами и процессами:

Ps

ps aux - отобразить все существующие процессы.

ps a -все интерактивные процессы.

ps ax -все процессы.

pstree – выводит дерево процессов

Каждый запущенный процесс в любой момент времени находится в одном из следующих состояний (статусов):

- **Активен (R=Running)** – процесс находится в очереди на выполнение, то есть либо выполняется в данный момент, либо ожидает выделения ему очередного кванта времени центрального процессора.
- **«Спит» (S=Sleeping)** – процесс находится в состоянии прерываемого ожидания, то есть ожидает какого-то события, сигнала или освобождения нужного ресурса.

- Находится в состоянии **непрерываемого ожидания (D=Direct)** – процесс ожидает определенного («прямого») сигнала от аппаратной части и не реагирует на другие сигналы;
- **Приостановлен (T)** – процесс находится в режиме трассировки (обычно такое состояние возникает при отладке программ).
- **«Зомби» (Z=Zombie)** – это процесс, выполнение которого завершилось, но относящиеся к нему структуры ядра по каким-то причинам не освобождены.

Потоки данных

У каждой программы существует 3 системных потока: stdout, stderr, stdin

Linux предоставляет специальные команды для перенаправления каждого потока

Команды с одной угловой скобкой переписывают существующий контент целевого файла:

•> — стандартный вывод

•< — стандартный ввод

•2> — стандартная ошибка

Команды с двойными угловыми скобками не переписывают содержимое целевого файла:

•>> — стандартный вывод

•<< — стандартный ввод

•2>> — стандартная ошибка

Можно создать каналы (pipes). между двумя процессами, в который один процесс сможет писать поток байтов, а другой процесс сможет его читать.

При помощи каналов организуются конвейеры оболочки. Когда оболочка видит строку вроде

команда1 | команда2

Переменные окружения.

Пример:

date

unset LANG ##отменить унаследованную переменную

date

export LANG=ru_RU.UTF-8 ##вернуть значение взад. Экспортировать потомкам

date

СИГНАЛЫ.

сигналы позволяют управлять программным обеспечением

отличие сигналов от других средств взаимодействия между процессами заключается в том, что их обработка программой обычно происходит сразу же после поступления сигнала (или не происходит вообще), независимо от того, что программа делает в данный момент. Сигнал

прерывает нормальный порядок выполнения инструкций в программе и передает управление специальной функции – обработчику сигнала.

SIGHUP (номер 1) изначально был предназначен для того, чтобы информировать программу о потере связи с управляющим терминалом (терминалы часто подключались к системе с помощью модемов, так что название сигнала происходит от hung up – повесить трубку). В ответ на получение SIGHUP демон обычно перезапускается (или просто повторно читает файл конфигурации).

Например, поменяли порт в /etc/ssh/sshd_conf- (sed -i 's/22/222/' /etc/ssh/sshd_config) сделали

ps aux | grep user1

kill -s HUP <PID>

или

kill -HUP <PID>

оно же

kill -1 <PID>

и не надо перезагрузки сервиса.

SIGINT (номер 2) обычно посылается процессу, если пользователь терминала дал команду прервать процесс (обычно эта команда – сочетание клавиш Ctrl-C) .

SIGKILL (номер 9) завершает работу программы. Программа не может ни обработать, ни игнорировать этот сигнал.

SIGCONT (номер 18) возобновляет выполнение процесса, остановленного сигналом SIGSTOP

SIGSTOP (номер 19) приостанавливает выполнение процесса. Как и SIGKILL, этот сигнал не возможно перехватить или игнорировать.

Пример: работает юзер через терминал. Смотрим сеансы pts (ps ax | grep userX)

[netstat –apnt | grep ssh | grep ‘*’]

Kill –STOP <PID> ## остановка процесса

Kill –CONT <PID> ## продолжить выполнение остановленного процесса

Kill –KILL <PID>

Установка ОС на VirtualBox VM

Server.corpX.un

Ubuntu-64

2048 ОЗУ

125 GB hdd

Сеть: мост

1.Guided – use entire disk

После первого запуска настройка ip адреса.

Пример:

sudo ifconfig eth0 inet 172.16.1.X/24

sudo route add default gw 172.16.1.254

Или

sudo ifconfig enp0s3 inet 172.16.1.X/24

sudo route add default gw 172.16.1.254

Настройка адреса через конфиг:

vi /etc/network/interfaces

auto eth0

iface eth0 inet static

    address 172.16.1.X

    netmask 255.255.255.0

    gateway 172.16.1.254

    dns-nameservers 172.16.1.254

    dns-search corpX.un

Старый способ именования интерфейсов задается в /etc/default/grub

GRUB_CMDLINE_LINUX=""

to

GRUB_CMDLINE_LINUX="net.ifnames=0"

Then, type in:

sudo update-grub

and reboot your system

sudo reboot

Настройка файлов конфигурации.

sudo vi /etc/hostname и /etc/hosts

Указать имя компьютра, например для машины server:

server