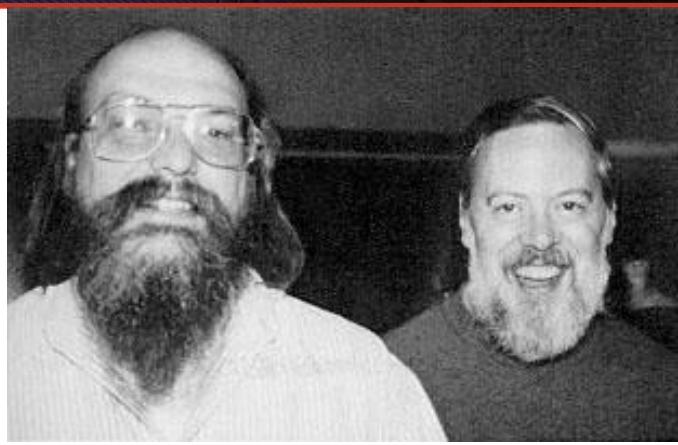
Linux Introduction

Aleksandr Mogylchenko (alatar.m)
Andrey Epifanov

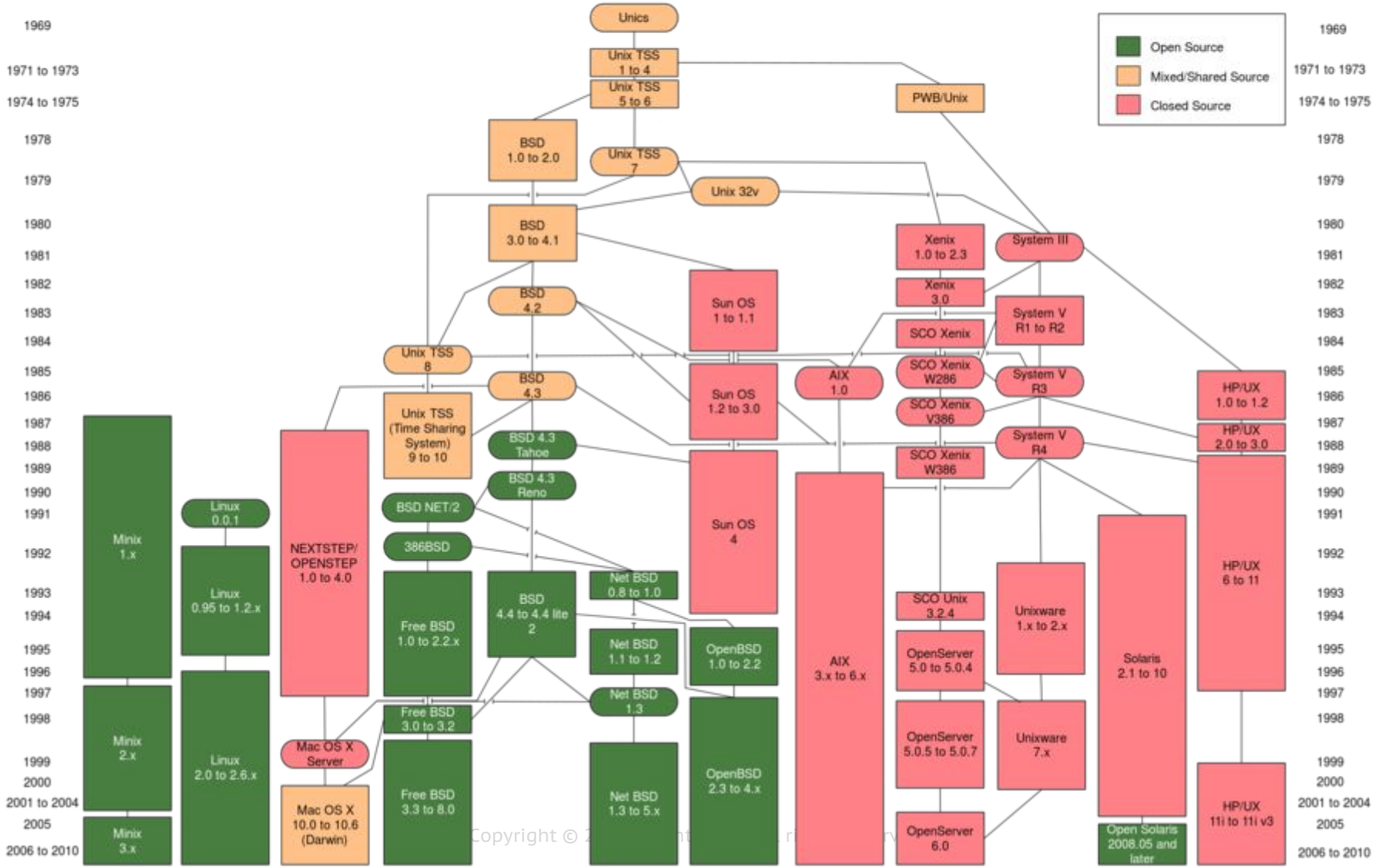
История



- **1969 - Кен Томпсон** и **Денис Ритчи**, работники корпорации **AT&T** создали ОС Unix для компьютера PDP-7.
- **1984 -** первые попытки финансирования GNU
- **1991 -** студент Хельсинского университета **Линус Торвальдс** выпустил первую версию Linux (на основе **Minix**).
- **март 1992 -** выпуск первого "почти безошибочного" релиза.
- …
- PROFIT :)

UNIX HISTORY
1969 – 2010

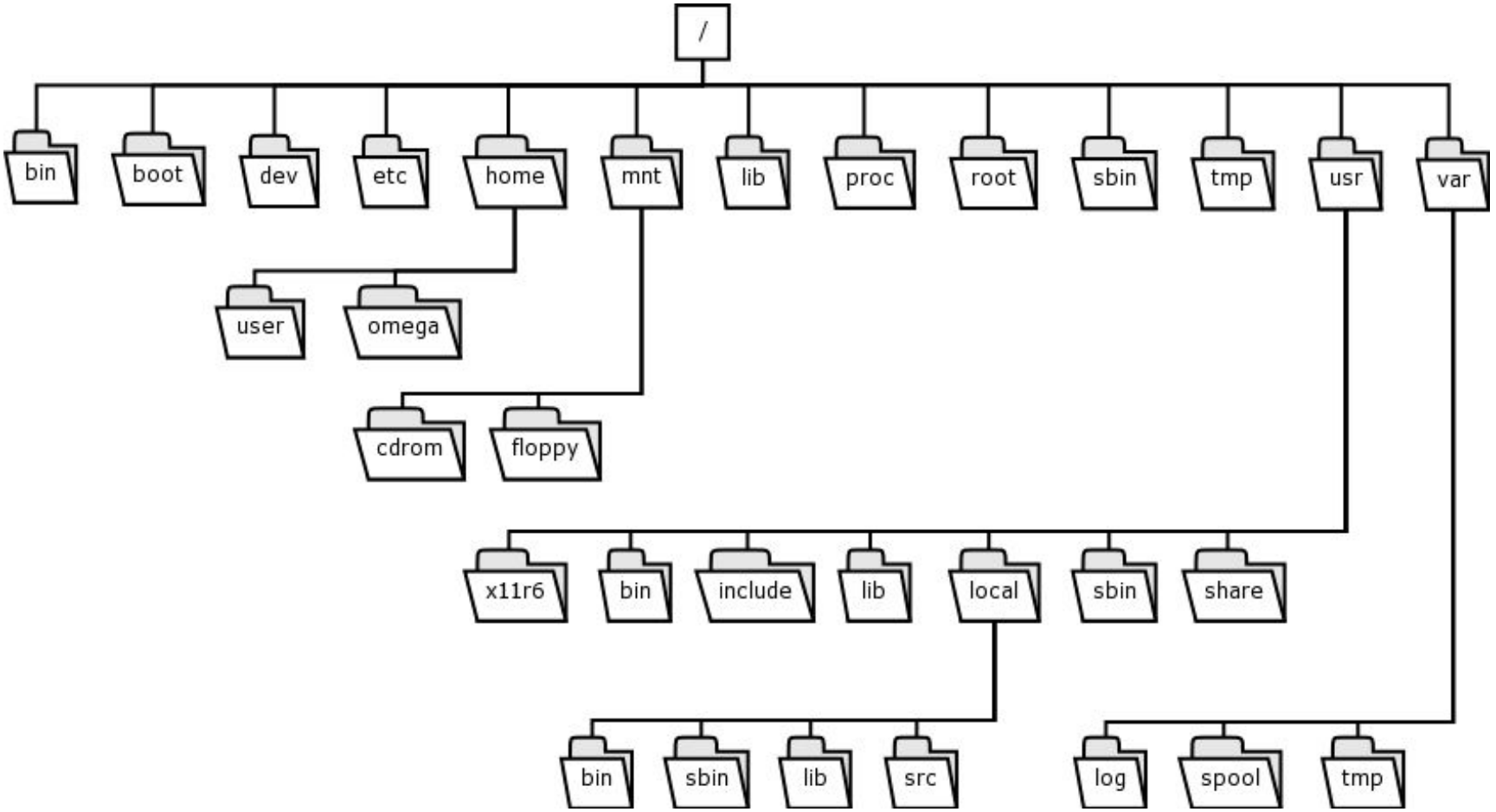
MIRANTIS
Pure Play OpenStack



Основные дистрибутивы Linux

- **Slackware Linux -** Старейший из существующих дистрибутивов. Создан **Патриком Фолкердингом** в **1992.** В настоящее время в основном выступает в качестве базовой системы для многочисленных кастомных решений.
- **SUSE Linux -** Впервые анонсирован в **1992** году как ответвление проекта **Slackware**. Свободно распространяемая ветка проекта - **openSUSE**.
- **Red Hat/CentOS/Fedora Linux -** Ведущий дистрибутив, создан в **1994** году. Развивается компанией **Red Hat**.
  Mageia/Mandriva/Mandrake Linux - впервые вышел в 1998 году как ветка дистрибутива Red Hat. В дальнейшем дистрибутив был коммерциализирован как Mandriva, но в 2010 году компания обанкротилась.
- **Debian/Ubuntu/Mint Linux -** Впервые выпущен в **1993** году. Разрабатывается как совместный проект большого сообщества (свыше **1000** разработчиков-добровольцев). В **2004** году выпущен дистрибутив **Ubuntu**, ориентированный преимущественно на использование в качестве десктопа. В **2006** году выпущен альтернативный, основанный на Ubuntu дистрибутив **Linux Mint**.

Файловая система



https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

Типы файлов

- файлы физических устройств
- именованные каналы (named pipes)
- сокеты (sockets)
- символические ссылки (symlinks)

Определить тип файла можно по первому символу вывода команды **ls -l**:

```
[student@ns lesson_2]$ ls -l
total 40
-rwxr-xr-x  1 root      root        2872 Aug 27  2001 arch
-rw-rw-rw-  1 root      root         612 Jun 25  2001 chain.b
brw-rw----  1 root      disk       3,  1 Feb  3 15:38 hda1
drwxrwxrwx  2 root      root       32768 Feb  3 15:38 ida
```

Интерпретация типов файлов:

- **-** обычный файл, двоичная запись на дисковом носителе; лист в дереве файловой структуры
- **d** каталог (директория); узел в дереве файловой структуры
- **l** ссылка на файл или каталог
- **b** файл - блочное устройство
- **c** файл - символьное устройство
- **s** файл-сокет

Файлы устройств

Имя файла	Описание
/dev/console	Системная консоль, т.е. монитор и клавиатура, физически подключенные к компьютеру
/dev/hd, /dev/sd	Жесткие диски с различными интерфейсами (IDE, SATA, SAS)
/dev/tty	Файлы поддержки пользовательских консолей. В Linux эти файлы обеспечивают работу виртуальных консолей
/dev/pty	Файлы поддержки псевдо-терминалов. Используются для обеспечения работы удаленных подключений (telnet, ssh)
/dev/ttyS	Файлы, обеспечивающие работу с последовательными портами
/dev/random	Файл генератора случайных чисел
/dev/null	Устройство-черная дыра. Все, что записывается в него, теряется навсегда. Обычно используется для перенаправления вывода нежелательных сообщений

Типы файлов

Именованные каналы ***(man mkfifo)***

Файлы этого типа служат для организации обмена данными между разными приложениями. Каналы представляют собой буферы типа **FIFO** (first in - first out). Все, что помещается одним процессом в канал, может быть прочитано другим процессом.

Сокеты

Представляют собой некий **псевдо-**файл, в который можеи читать и писать. Являются неким аналогом каналов. После того, как связь установлена, данные передаются непосредственно между процессами с помощью ядра ОС и функций АПИ, например ***read()*** и ***write()***

AF_UNIX, **AF_LOCAL**	Локальная связь	unix(7)
AF_INET	IPv4	ip(7)
AF_INET6	IPv6	ipv6(7)
AF_NETLINK	Связь между ядром и юзерспейсом	netlink(7)
AF_APPLETALK	AppleTalk	ddp(7)
AF_PACKET	Низкоуровневый интерфейс работы с пакетами	packet(7)

Ссылки

Жесткие ссылки

Любой файл в Линукс может иметь более одного имени. В этом случае все имена файла называются "**жесткими ссылками" (hard link)**. Файл физически остается на диске до тех пор, пока у него есть хотя бы одно имя (жесткая ссылка). Другая особенность жестких ссылок - это то, что невозможно различить исходный файл и жесткие ссылки на него, созданные позднее.

Символические ссылки

Другой тип ссылок - **символические ссылки (symbolic links)**. В отличие от жестких ссылок, символические могут указывать на файлы, расположенные на другой файловой системе. Кроме того, символические ссылки не препятствуют удалению исходного файла. В этом случае символическая ссылка становится "висячей", то есть просто указывает в никуда.

Для создания ссылок используется команда **ln**. Символьную ссылку позволяет создать ключ -s:

```
$ ln /path/to/file /path/to/link

$ ln -s /home/ogelbukh/file1 ~/link1
$ ls -la ~/link1
lrwxrwxrwx 1 root root 31 Dec 13 21:13 links1 -> /home/root/file1
```

Навигация по ФС

MIRANTIS
Pure Play OpenStack

- **ls** **отображает содержимое текущего каталога**. Позволяет использовать различные ключи, наиболее важные: -l - расширенный формат вывода, содержит различные атрибуты файлов;-a - выводит данные о текущем каталоге (‘.’), родительском каталоге (‘..’) и скрытых элементах (начинающихся с ‘.’)

- **cd** позволяет **изменить текущий каталог** на указанный параметром команды. При запуске без параметров позволяет перейти в домашний каталог текущего пользователя.

- **pwd** **показывает полное имя текущего каталога.**

- **pushd** изменяет текущий каталог на указанный параметром, **аналогично команде cd**. Кроме того, команда **позволяет сохранять историю** посещенных каталогов в виде стека.

- **popd** изменяет текущий каталог, **выбирая каталог назначения из стека истории посещенных каталогов**, сформированного командой pushd.

Создание, удаление, копирование файлов

- **cp** - Копирование существующих файлов. Эта команда принимает два параметра: имя файла источника и имя файла приемника. Если в качестве имени файла источника указывается директория, производится копирование файла в эту директорию без изменения имени. Часто используются следующие ключи:
  - -p сохранить атрибуты файлов при копировании (режим, владелец, время изменения)
  - -R, -r рекурсивное копирование директорий
  - -s создать символическую ссылку вместо копирования файла
  - -l создать жесткую ссылку вместо копирования файла
- **touch** - Для создания файлов может использоваться эта команда. Кроме того, создать файл позволяют некоторые программы, например, текстовые редакторы.
- Наконец, можно использовать для создания файлов оператор перенаправления вывода ('**>**').
- **rm** - Удаление файлов. Наиболее часто используются ключи:
  - -f игнорирование несуществующих файлов и аргументов
  - -r **рекурсивное удаление директории и файлов, которые в ней содержатся**
- **mv** - Перемещение файлов, то есть копирование файла с последующим удалением источника.

Вывод текстовой информации

Для вывода текстовой информации наиболее часто используются команды **cat, more, less, tail и head.**

Команда **cat** выводит на экран содержимое текстового файла.

```
[user@localhost ~]$ # cat /etc/timezone
Europe/Moscow
[user@localhost ~]$
```

more - эта команда позволяет просматривать файл по частям, перемещаясь к следующей строке по нажатию клавиши *Enter* и к следующему экрану по *Space*.

Команды **head** и **tail** позволяют просмотреть часть файла, не открывая его полностью. По умолчанию они выводят по 10 строк, соответственно, первых и последних в файле. Флаг **-n N** позволяет указать другое число выводимых строк (**N**). Флаг **-c N** заставляет команду вывести **N** байт.

less - эта команда позволяет прокручивать текст в любом направлении, а также искать вхождения групп символов. *Users should realize that less(1) provides more(1) emulation plus extensive enhancements.*

Создание, удаление директорий

Основные операции с каталогами в целом производятся теми же командами, что и операции с файлами.

- **mkdir** Создать каталог. Ключи:
  - -p позволяет создать также все промежуточные каталоги, вместо того, чтобы выдать ошибку, если какой-то из них не существует.
- **rmdir** Эта команда позволяет удалить только пустой каталог
- **mv** Универсальный инструмент перемещения файлов и/или директорий

man mv

(опционально) ***man rsync***

Учетные записи, пароли

Linux- многопользовательская операционная система. Это означает, что для **каждого файла** в системе заданы следующие группы атрибутов: **имя пользователя-владельца, имя группы владельцев, а также права на действия**, которые имеют право произвести с файлом владелец, пользователи-члены группы владельцев и прочие пользователи: **чтение, запись и исполнение.**

Учетная запись пользователя представляет собой пару "**имя-пароль**", а также ряд связанных с именем пользователя свойств, в том числе:

- имя пользователя
- цифровой идентификатор пользователя (UID)
- идентификатор группы (GID)
- полное имя пользователя
- путь к домашнему каталогу
- командная оболочка

Данные об учетной записи пользователя хранятся в нескольких файлах:

- **/etc/passwd** - параметры учетной записи, помимо пароля. Пример записи в файле:

```
root:x:0:0:root:/root:/bin/bash
```

Учетные записи, пароли ч.2

/etc/shadow - пароль пользователя в зашифрованном виде

Пример записи в файле:

```
root:$6$1sLl.
zsTdfHXKgtl$wdBf0PWOWXMh9HNSMVeY58dzqwRp6Xe3PtAq2qBbJx02NoPfy86yjoZUp.
v5DnKbr0ZgoGMwAnIBsIGJMIM1P1:15324:0:99999:7:::
```

Значения полей:

имя пользователя:шифрованный пароль с солью:число дней с последнего изменения пароля: число дней перед изменением пароля:число дней, после которого пароль должен быть изменен:число дней, за которое пользователь начнет получать предупреждение о том, что пароль устаревает:число дней между устареванием пароля и блокировкой учетной записи: число дней до блокировки учетной записи:зарезервированное поле

/etc/group - идентификаторы групп и пользователи, входящие в группу

man crypt
man shadow
man group

Создание, удаление пользователей

Учетная запись суперпользователя, а в некоторых дистрибутивах - также запись обычного пользователя, создается в процессе инсталляции системы.

В дальнейшем добавить пользователя в систему можно с помощью команды **adduser** (в некоторых системах также существует команда **useradd**, которая является более низкоуровневым интерфейсом). Полезные опции при использовании **useradd**:

- ***-m*** Создать домашнюю директорию
- ***-s*** Указать шелл пользователя
- ***-G*** Список групп куда пользователь будет входить
- ***-U*** Создать группу с таким же именем
- ***-p*** указать пароль (можно поменять пароль позднее с помощью **passwd**)

Так же может быть полезна команда ***useradd -D***

Для удаления учетной записи пользователя используется команда **userdel**.

Права доступа

Базовые механизмы разграничения доступа сохранились практически в неизменном виде с первых версий ОС Linux.

В индексном дескрипторе каждого файла записаны имя так называемого **владельца** файла и **группы**, которая имеет права на этот файл. Первоначально, при создании файла его **владельцем объявляется тот пользователь, который этот файл создал**. Точнее — тот пользователь, от чьего имени запущен процесс, создающий файл. **Группа тоже** назначается при создании файла — по идентификатору группы процесса, создающего файл.

Владельца и группу файла можно поменять командами **chown** и **chgrp** соответственно.

Также в дескрипторе файла определяются права доступа для **владельца**, **группы** и **остальных пользователей**. Каждой из этих трех категорий пользователей соответствуют **три бита** в формате дескриптора.

- Старший бит определяет право на чтение файла для категории пользователей,
- Средний бит - право на запись в файл,
- Младший бит - право на выполнение программы, записанной в файле.

```
[user]$ ls -l /bin/ls
-rwxr-xr-x 1 root root 49940 Sep 12 1999 /bin/ls
```

Управление правами

Для изменения прав доступа к файлу используется команда **chmod**. Установить права с ее помощью можно двумя способами:

- указать права в формате XOY или XOY, где
  - X определяет категорию пользователей (**u** - владелец, **g** - группа, **o** - прочие, **a** - все категории);
  - O определяет операцию (**+** - предоставить право, **-** - лишить права, **=** - установить указанные права вместо имеющихся);
  - Y определяет соответствующее право (r, w, x).

Например, команда, дающая всем право на выполнение файла file_name:

```
[user]$ chmod a+x file_name
```

Эта команда эквивалентна команде chmod +x file_name, поскольку по умолчанию права меняются для всех категорий пользователей.

Следующая команда удаляет право на чтение и запись для всех, кроме владельца файла:

```
[user]$ chmod go-rw file_name
```

Управление правами ч.2

- указать права в виде десятичных цифр, соответствующих двоичной записи прав в дескрипторе, например:

```
[user]$ chmod 760  file_name
```

Эта команда задает права на чтение, запись и выполнение для владельца (7 = 111), чтение и запись группе (6 = 110) и никаких прав для остальных пользователей (0 = 000).

Использовать команду **chown** может только владелец файла и суперпользователь. Использовать команду **chgrp** могут также члены группы владельцев файла.

Права на каталоги обозначаются так же, как права доступа к файлу, но **имеют немного отличное значение**.

- Флаг **r** дает **право на просмотр** содержимого каталога (например, командой ls).
- Флаг **w** позволяет **создавать и удалять** файлы в каталоге. При этом необязательно иметь право на изменение собственно файла.
- Флаг **x** дает право с**делать этот каталог текущим** (например, командой cd).

man chmod

THANK YOU

SUID, SGID, Sticky bit

man chmod

Sticky bit:
при установке бита на директории ограничивает разрешает удаление файлов только владельцу файла или **root.** Без него любой пользователь с **WX** правами на директории может удалять/переименовывать файлы в ней.

chmod +t /usr/local/tmp

SUID, SGID

- при установке флага на файле позволяет запустить файл с правами того пользователя, кому он принадлежит.
- при установке флага на директории все новые файлы в этой директории унаследуют **gid** папки, а не пользователя который ее создал.

chmod u+s /bin/ping

~$ stat -c '%A %a %n' /bin/ping

-rwsr-xr-x 4755 /bin/ping

Capabilities

man capabilities

Linux разделяет все процессы на две категории:

- привелигированные (euid=0), для которых ядро не выполняет никаких проверок прав доступа
- непривелигированные (euid!=0), для которых проводится полный набор проверок

Начиная с Linux 2.2 полномочия root начали выносить в отдельные **сущности.**

setcap cap_net_raw+p /bin/ping

getcap /bin/ping

~$ grep CAP_NET_RAW /usr/src/linux-headers-3.13.0-61/include/uapi/linux/capability.h

#define CAP_NET_RAW 13

Процессы

MIRANTIS
Pure Play OpenStack

```
  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
    1             20   0 24680  2580  1352 S  0.0  0.0  0:00.89 /sbin/init
 4848 kai         20   0  159M 12328  4720 S  0.0  0.2  0:00.07 ├─ xterm
 4850 kai         20   0  109M  4956  1684 S  0.0  0.1  0:00.15 │  └─ bash
 4905 kai         20   0  111M  2420  1436 R  0.0  0.0  0:00.14 │     └─ htop
 4775 kai         20   0 15956   416   212 S  0.0  0.0  0:00.00 ├─ /usr/bin/xsel -s -i
 3809 kai         20   0  159M 12328  4720 S  0.0  0.2  0:02.33 ├─ xterm
 3811 kai         20   0  109M  4960  1684 S  0.0  0.1  0:00.15 │  └─ bash
 3896 kai         20   0  111M  2464  1436 S  1.0  0.0  0:39.11 │     └─ htop
 3328 kai         20   0 26932  2284  1276 S  0.0  0.0  0:02.55 ├─ tmux new-session -d -s kai-T420s
 4118 kai         20   0  112M  8600  1764 S  0.0  0.1  0:00.33 │  ├─ -bash
 4231 kai         20   0  271M 27796  8044 S  0.0  0.3  0:07.83 │  │  └─ emacs -nw --eval (menu-bar-mode 0) Makefile
 3361 kai         20   0  109M  4964  1684 S  0.0  0.1  0:00.20 │  ├─ bash
 3354 kai         20   0  109M  5152  1764 S  0.0  0.1  0:00.28 │  ├─ bash
 4088 kai         20   0  103M  1672  1052 S  0.0  0.0  0:00.01 │  │  └─ man -a getpid
 4101 kai         20   0   99M  1000   820 S  0.0  0.0  0:00.00 │  │     └─ pager -s
 3341 kai         20   0  109M  5020  1728 S  0.0  0.1  0:00.20 │  ├─ bash
 3335 kai         20   0  109M  5088  1768 S  0.0  0.1  0:00.32 │  ├─ bash
 4845 kai         20   0 14644   384   296 R 199.  0.0  3:11.70 │  │  └─ ./pthr_create
 4846 kai         20   0 14644   384   296 R 99.0  0.0  1:35.84 │  │     └─ ./pthr_create
 3329 kai         20   0  112M  8680  1760 S  0.0  0.1  0:00.43 │  └─ -bash
 4820 kai         20   0  103M  1664  1044 S  0.0  0.0  0:00.01 │     └─ man -a htop
 4831 kai         20   0   99M  1000   820 S  0.0  0.0  0:00.00 │        └─ pager -s
```

PID 1 - главный процесс, который запускается ядром и отвечает за запуск и остановку ОС. Все остальные процессы созданы этим процессом с помощью ***fork()*** или ***clone()***

Процессы (статусы)

Возможные статусы процессов:

- R - процесс выполняется или готов к выполнению (состояние готовности)
- D - процесс в "беспробудном сне" - ожидает дискового ввода/вывода
- T - процесс остановлен (stopped) или трассируется отладчиком
- S - процесс в состоянии ожидания (sleeping)
- Z - процесс-зобми

Уточняющие:

- < - процесс с отрицательным значением nice
- N - процесс с положительным значением nice
- "+" означает что процесс на “переднем плане” (foreground)
- маленькая "s" означает что процесс - лидер сессии

Процессы (идентификаторы)

Существует 4 пары идентификаторов процесса:

uid, gid

Идентификатор пользователя и группы от которой запущен процесс

effective uid and gid

Некоторые программы меняют **uid** и **gid** запущенного процесса на свои собственные. Они известны как ***setuid*** программы, и они полезны для ограничения доступа к сервисам (особенно работающим от какого-либо другого пользователя), например к сетевым. Таким образом, **effective uid and gid** принадлежат ***setuid*** программе, и хранятся отдельно. Ядро проверят их наравне с **uid, gid** при проверке прав доступа.

file system uid and gid

Обычно совпадают с **effective uid and gid,** и нужны для некоторых дополнительных проверок (например при работе с NFS)

saved uid and gid

Требуются POSIX и используются для сохранения **uid** и **gid** на время их изменения.

Процессы ч.2

Для наблюдения за процессами можно использовать команду ***top***

kos@linux.home: /home/kos – Консоль

Файл Сеансы Настройки Помощь

```
  8:42pm  up  1:37,  2 users,  load average: 1,26, 1,21, 1,07
61 processes: 59 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:  3,9% user, 96,0% system,  0,0% nice,  0,0% idle
Mem:   191296K av,  185164K used,    6132K free,    2944K shrd,   12532K buff
Swap:   64508K av,   39320K used,   25188K free                  130384K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
 1102 kos       17   0 83148   75M 69052 R     94,8 40,6  52:06 vmware
  937 root       9   0 21868   14M  8520 S      2,1  7,7   2:59 X
 1099 kos        9   0  3876  3340  3136 S      1,3  1,7   0:02 kdeinit
 1284 root      15   0  1164  1164   920 R      1,1  0,6   0:00 top
 1092 kos        9   0  4652  4176  3612 S      0,5  2,1   0:06 kdeinit
 1286 kos        9   0   720   720   608 S      0,3  0,3   0:00 xwd
    1 root       8   0   132    80    80 S      0,0  0,0   0:03 init
    2 root       9   0     0     0     0 SW     0,0  0,0   0:00 keventd
    3 root       9   0     0     0     0 SW     0,0  0,0   0:34 kapm-idled
    4 root       9   0     0     0     0 SW     0,0  0,0   0:00 kswapd
    5 root       9   0     0     0     0 SW     0,0  0,0   0:00 kreclaimd
    6 root       9   0     0     0     0 SW     0,0  0,0   0:00 bdflush
    7 root       9   0     0     0     0 SW     0,0  0,0   0:01 kupdated
  591 rpc        9   0   172   124   124 S      0,0  0,0   0:00 portmap
  623 root       9   0   288   244   244 S      0,0  0,1   0:00 syslogd
  631 klogd      9   0   804   168   168 S      0,0  0,0   0:00 klogd
```

Процессы (сбор информации)

ps [-опции]
ps [опции]
ps [-- длинное_имя_опции [-- длинное_имя_опции] ...]

Все процессы в системе
[user]$ ps –e

Фильтрация по конкретным полям:
[user]$ ps –eo pid,user,cmd
[user]$ ps -eo pid,gid,uid,ppid,euid,egid,cmd

Для того, чтобы увидеть все процессы в системе, с применением графического отображения отношения "предок-потомок":
[user]$ ps –ef

Для того, чтобы увидеть, сколько % ЦПУ и памяти занимают запущенные вами процессы:
[user]$ ps –u

Чтобы узнать приоритет процесса и значение nice, воспользуйтесь опцией -l:
[user]$ ps –l

Приоритеты процессов

от +20 (наименьший приоритет), до -20 (наивысший приоритет)

устанавливается для каждого процесса в момент порождения этого процесса

Для запуска процесса с нестадратным приоритетом:
nice [- adnice] command [args]
где adnice — значение (от –20 до +19), добавляемое к значению nice процесса-родителя.

Для изменения значения nice для уже выполняющихся процессов:
renice priority [[-p] PID] [[-g] grp] [[-u] user]

Например
[root]# renice -1 987 –u daemon –p 32
увеличивает на 1 приоритет процессов с PID 987 и 32, а также всех процессов пользователя daemon.

Сигналы

man 7 signal

1	HUP	Hangup. Отбой
2	INT	Interrupt.
3	QUIT	Как правило, сильнее сигнала Interrupt
9	KILL	Всегда прекращает выполнение процесса
11	SEGV	Segmentation Violation. Доступ к недозволенной области памяти
15	TERM	Требование закончить процесс (программное завершение)

Для посылки сигнала процессу (или группе процессов) существует команда ***kill***:

kill [options] <pid> [...]

Например

kill 123 543 2341 3453

Послать сигнал по умолчанию (SIGTERM) всем этим процессам

kill -l

Вывести доступные сигналы в виде таблицы

Linux Shells

Unix shell – CLI в Unix-like системах, может группировать команды, представляет собой интерпретатор языка программирования.

- **sh** – **Bourne shell** – первоначальный командный интерпретатор Unix, написанный **Стефаном Борном** (AT&T)
- **csh** – **C shell** – создан **Биллом Джоем** – выпускником университета Berkeley – **C-style shell**
- **tcsh** – **улучшенный csh:** встроенная история команд, автодополнение и т.д.
- **bash** – основная командная оболочка в большинстве дистрибутивов Linux – **Bourne-again shell** – написанный **Брайаном Фоксом** для GNU проекта как замена sh (**1989**) – включает в себя многие возможности из других оболочек.
- **zsh** – ориентированная на интерактивную работу пользователя оболочка

BASH (клавиатуные сокращения)

MIRANTIS
Pure Play OpenStack

Передвижение курсора:

Ctrl + a Перейти в начало строки (Home)
Ctrl + e Перейти в коней строки (End)
Ctrl + p Предыдущая команда (Up arrow)
Ctrl + n Следующая команда (Down arrow)
Alt + b Назад (влево) на одно слово
Alt + f Вперед (вправо) на одно слово
Ctrl + f Вперед на один символ
Ctrl + b Назад на один символ

История:

Ctrl + r Поиск по истории набранных команд по мере печати
Ctrl + p Предыдущая команда в истории
Ctrl + n Следующая команда
Ctrl + o Выполнить найденную через Ctrl+r команду
Ctrl + g Выйти из поиска
!! Повтор последней команды
!N Выполнить команду под номером N из истории
!$ Последний аргумент предыдущей команды
!* Все аргументы предыдущей команды
^*abc*-^-*def* Выполнить предыдущую команду, заменяя *abc* на *def*

Многозадачность. Управление заданиями

Основная характеристика системы Linux – многозадачность. Она поддерживается в том числе и в консоли. Во-первых, возможно запустить несколько командных оболочек в разных виртуальных консолях. Переключение между ними производится клавишами <***Alt+F[1-6]***>.

Многозадачностью можно воспользоваться и в рамках одной консоли. Для поддержки многозадачности в bash можно использовать следующие средства:

- <***Ctrl+Z***> – комбинация клавиш, отправляющая текущему процессу немаскриуемый сигнал SIGSTOP. Выполнение процесса приостанавливается, управление передается родительскому процессу, в данном случае, командной оболочке.
- **command &** – символ **&** позволяет запустить ее в **фоновом режиме**.
- **jobs** – выводит список текущих заданий командного интерпретатора.
- **bg <#job>** – переводит задание **<#job>** в фоновый режим. При этом задание должно находиться в остановленном состоянии. Номер задания можно не указывать, если оно единственное.
- **fg <#job>** – передает управление консолью заданию номер **<#job>**. Задание должно находиться в остановленном состоянии или в фоновом режиме.

Фильтрация строковых данных GREP

Для того, чтобы найти в текстовых файлах вхождения известных подстрок или данные в известном формате, можно использовать фильтры и регулярные выражения.

Основным средством для фильтрации файлов в Linux является программа **grep**. В качестве аргументов для этой команды должны быть указаны регулярное выражение и имя файла или каталога: **grep regexp filename.**

Регулярными выраженями называются средства указания шаблона для поиска его в тексте. Это может быть просто слово или его часть, а может быть намного более сложная структура, позволяющая выбирать данные различного формата в рамках одного выражения.

Из наиболее частых опций используются следующие:

- **-f** - бытрый поиск, ищет только совпадения (без шаблонов). Для удобства есть отдельная команда **fgrep**
- **-r** - рекурсивный поиск и во вложенных папках.
- **-e** - расширенный вариант regexp. Есть также **egrep**

Если аргумент **filename** не указан, команда grep осуществляет поиск в тексте, поступающем на стандартный вход. Это можно использовать для поиска в выводах других команд, перенаправляя их вывод на вход grep через канал (pipe).

```
[user@localhost ~]$ cat /var/log/messages | grep ERROR
```

Фильтрация строковых данных GREP

Для фомирования шаблонов наиболее часто используются следующие специальные символы:

^ – начало строки;

$ – конец строки;

[] – любой символ из заключенных в скобки. Поддерживает диапазоны, например, [1-6] - любая цифра от 1 до 6, и т.п.;

[^] – любой символ, кроме указанных в скобках после ^;

\ }} – превращает управляющий символ в обычный, {{\$ означает символ '$', а не конец строки;

. – любой символ;

***** – предыдущий шаблон встречается 1 или более раз в тексте.

Фильтрация с GREP - примеры

Поиск строки, начинающейся с **root:**

cathy ~> **grep ^*root* /etc/passwd**
root:x:0:0:root:/root:/bin/bash

Поиск строки, заканчичающейся на `:`

cathy ~> **grep *:$* /etc/passwd**
news:x:9:13:news:/var/spool/news:

Поиск всех строк, имеющих **y** или **f**

cathy ~> **grep *[yf]* /etc/group**
sys:x:3:root,bin,adm
tty:x:5:
mail:x:12:mail,postfix
ftp:x:50:
nobody:x:99:

Поиск всех 5и символьных слов начинающихся на **c** и заканчивающихся на **h**
cathy ~> **grep '\<*c...h*\>' /usr/share/dict/words**
catch
clash
cloth
coach
…

Поиск всех слов, начинающихся на **c** и заканчивающихся на **h**
cathy ~> **grep '\<*c.*h*\>' /usr/share/dict/words**
caliph
cash
catch
cheesecloth
cheetah
--output omitted--

Поиск файлов и директорий

man find

Найти файл по имени:

 # find . -name tecmint.txt

Найти все директории по имени **Tecmint**

 # find / -type d -name Tecmint

Найти все PHP файлы:

 # find . -type f -name "*.php"

Найти все файлы с правами доступа 777:

 # find . -type f -perm 0777 -print

Найти все исполняемые файлы:

 # find / -perm /a=x

Найти все mp3 файлы и удалить их:

 # find . -type f -name "*.mp3" -exec rm -f {} \;

Найти пустые файлы:

 # find /tmp -type f -empty

Найти все файлы, которые менялись 50 дней назад:

 # find / -mtime 50

Найти все файлы, которые менялись как минимум два дня назад:

 # find / -mtime +1

Найти все файлы, которые модифицировались в последний час:

 # find / -cmin -60

Найти все файлы размером от 50Мб до 100Мб:

 # find / -size +50M -size -100M

Полезные ссылки

1. Сетевые Операционные Системы
2. Анатомия ядра Linux
3. Структура ОС Linux
4. Карта ядра Linux
5. ProcFS
6. Linux Documentation project
7. Linux Performance tools
8. LWN.net
9. ARCHLinux wiki

Thank you!