# Лекция 3

Unix не Windows, он не похож на него и приемы работы в unix отличаются от приемов работы в Windows. Unix – это не только операционная система, это еще и идеология работы с компьютером. Те правила, о которых мы с вами будем говорить сейчас лежат в основе изучения Linux, да и Unix вообще. Общий термин для них – Unix Way:

- *одна задача – одна программа.* В Unix не принято делать комбайны для выполнения «сразу всего». Программа делается таким образом, чтобы она могла выполнять одно простое действие, но выполняла его хорошо.
- *есть множество путей решения.* Для решений той или иной комплексной задачи каждый может выбирать свой набор простых компонент для ее решения.
- *все есть файл*. Самая замечательная концепция в unix. Действительно, в Unix все представлено в виде файлов – программы, настройки, системные данные и даже устройства. И с устройствами можно работать как с простыми файлами.
- Для обмена информацией (и для конфигурации) все программы используют формат, понятный человеку: *текстовый файл*
-

Будьте готовы к изучению не просто новых программ, а новых методов работы на компьютере.

На сегодняшний день Linux является полнофункциональным, открытым и бесплатным аналогом Unix. Но этого бы не произошло, не будь программного обеспечения в рамках проекта GNU (GNU's not Unix, GNU – это не Unix). Linux содержит много утилит GNU, включая трансляторы многих языков программирования (C, C++, Fortran, Pascal, LISP, Ada, BASIC, SmallTalk, Perl, PHP, Tcl/Tk и др.), отладчики, текстовые редакторы, утилиты печати и многое другое. Проект GNU развивается под эгидой фонда свободно распространяемого программного обеспечения – Free Software Foundation (FSF).

Давайте рассмотрим операционную систему как единый комплекс. Ниже приведен список того, что мы получим, установив ее:

**Ядро Linux:**

Ядро - это основная часть операционной системы. Оно отвечает за распределение памяти, управление процессами и периферийными устройствами. Для поддержки большего объема оперативной памяти по сравнению с физически установленной на компьютере, ядро позволяет использовать область подкачки, размещая страницы оперативной памяти на жестком диске.

Собственные файловые системы Linux (ext2fs...ext4fs) разработаны для оптимального использования дискового пространства и надежности.

**Утилиты GNU:**

Linux содержит множество утилит GNU, без которых была бы невозможна работа с операционной системой.

**X Window:**

Графический интерфейс пользователя представлен в Linux средой X Window. Различные оконные менеджеры (IceWM, WindowMaker, Fluxbox и прочие) и графические среды такие как KDE и GNOME, обеспечивают удобный интерфейс и работу со средствами мультимедиа.

**Интерфейсы DOS и Windows:**

Поскольку Linux была создана для компьютеров класса ПК, разработчики посчитали необходимым обеспечить совместимость с программами MS-DOS. В Linux предлагается эмулятор DOS как часть дистрибутива. Он позволяет исполнять DOS-

приложения непосредственно из-под Linux. Для запуска программ Microsoft Windows было разработано несколько средств. Наиболее известное из них – WINE – свободная реализация Windows API. Wine также входит в большинство дистрибутивов Linux.

Linux позволяет без проблем переносить файлы между файловыми системами DOS и Windows, напрямую обращаясь к соответствующим разделам на диске, хотя это и требует некоторой настройки.

**Сетевая поддержка:**

TCP/IP – основная сетевая система используемая Unix и Linux. TCP/IP – это целый набор протоколов, разработанных для Internet. Однако для объединения в локальные сети машин Unix тоже используется TCP/IP. Также Linux поддерживает другие протоколы, такие как IPX/SFX, AppleTalk и т.д.

**5.** как же узнать тип операционной системы, установленной у вас на компьютере. Для получения такой информации существует утилита uname (Unix NAME).

uname, запущенная без параметров, покажет базовое имя системы:

```
gserg@ADM:~$ uname
Linux
```

Также она может принимать следующие параметры:

-s – показывает название ядра системы

-r – имя релиза ядра системы

-v – имя версии, а также дату компиляции ядра

-o – операционную систему

-p – тип процессора

-m – тип оборудования (i386, i686, Alpha)

-a – всю информацию сразу

Это не все параметры uname. О справке Linux мы поговорим с вами на 5-м занятии.

Команда free показывает объем памяти и объем ее использования, а также использование swap:

```
gserg@ADM:~$ free
             total       used       free     shared    buffers     cached
Mem:        498916     483332      15584          0       4392     112924
-/+ buffers/cache:     366016     132900
Swap:      1453840     412532     1041308
```

Обратите внимание, что практически вся свободная память резервируется системой под дисковые буферы и дисковый кэш, что позволяет Linux более эффективно работать с дисками.

Состояние системы в данный момент, степень ее загруженности и время без перезагрузок показывает команда uptime:

```
gserg@ADM:~$ uptime
 14:24:08 up 1 day,  6:01,  2 users,  load average: 0.08, 0.19, 0.16
```

Первым идет текущее время, потом, после слова up – время, прошедшее с момента включения компьютера, потом показано сколько пользователей зарегистрировано сейчас в системе (это может быть и несколько регистраций одного и того же пользователя) и загрузка системы. Загрузка системы показывается в количестве процессов, одновременно работающих в системе, среднее значение за 1-ну, 5 и 15 минут. Система считается нагруженной, если это значение превышает 1 в расчете на 1 процессор.

Другим средством мониторинга производительности является команда vmstat:

```
[gserg@admin ~]$ vmstat
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0 268928 776168  15072 203316    1    2    10    14  207  225 13  3 84  0  0
```

Эта команда выдает за раз достаточно большой объем информации.

Раздел procs:

***r*** — количество ожидающих процессов

***b*** — количество спящих процессов

Раздел memory:

***swpd*** — объем используемой виртуальной памяти
***free*** — объем свободной виртуальной памяти
***buff*** — объем памяти, занятой под дисковые буферы
***cache*** - объем памяти, занятой под дисковый кэш
Раздел swap:
si — объем памяти, подкачанной с диска
so — объем памяти, выгруженной на диск
Раздел io:
bi — количество блоков, отправленных на блочное устройство
bo — количество блоков, прочитанных с блочного устройства
Раздел system:
in — количество прерываний в секунду
cs — количество переключений контекста в секунду
Раздел cpu:
us — время выполнения кода уровня пользователя (в процентах от общего времени)
sy — время выполнения кода уровня системы (в процентах от общего времени)
id — время простоя процессора (в процентах от общего времени)
wa — время ожидания ввода/вывода
st — время работы виртуальной машины уровня ядра

vmstat показывает при простом запуске усредненные показатели за все время с момента запуска системы. Но можно попросить vmstat вывести показатели за заданное количество времени:

```
[gserg@admin ~]$ vmstat 1 5
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0 268844 742148  16620 212452    1    2    10    14  216  230 13  3 84  0  0
 0  0 268844 742140  16628 212436    0    0     0    48 1097  414  6  2 92  0  0
 0  0 268844 742140  16628 212436    0    0     0     0 1105  392  5  1 94  0  0
 0  0 268844 742172  16628 212436    0    0     0     0 1090  345  4  1 95  0  0
 0  0 268844 742172  16628 212436    0    0     0     0 1107  403  6  1 93  0  0
```

В примере выведена информация за каждую секунду на протяжении 10 секунд. Если второй параметр (5) не указывать, то vmstat будет выводить информацию каждую секунду до нажатия Ctrl+C:

```
[gserg@admin ~]$ vmstat 1
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0 268844 740824  16824 212720    1    2    10    14  217  230 13  3 84  0  0
 0  0 268844 740856  16824 212720    0    0     0     0 1088  488  8  2 90  0  0
 0  0 268844 740856  16824 212720    0    0     0     0 1392  873 14  4 82  0  0
^C
```

Для просмотра размеров файловых систем используется команда df:

```
[gserg@admin ~]$ df
Файловая система     1K-блоков      Исп  Доступно  Исп% смонтирована на
/dev/hdb2             36733400  10074596  24762736  29% /
/dev/hdb1               101086     16228     79639  17% /boot
tmpfs                   647688         0    647688   0% /dev/shm
```

Без параметров команда выводит данные в виде количества блоков по 1 килобайту. Для человека это не очень удобная подача информации. У df существует ключ -h (или — human), позволяющий увидеть объемы в привычных нам единицах измерения:

```
[gserg@admin ~]$ df --human
Файловая система      Разм  Исп  Дост  Исп% смонтирована на
/dev/hdb2              36G  9,7G   24G  29% /
/dev/hdb1              99M   16M   78M  17% /boot
tmpfs                 633M     0  633M   0% /dev/shm
```

**Файловая система**

**1.** Файловая система Linux, в отличие от операционных систем семейства Windows не разделена по томам (дискам, устройствам), а имеет единую древовидную структуру, в основе которой лежит ***корневой каталог***. Корневой каталог - это уровень файловой системы, выше которого по дереву каталогов подняться невозможно. В Linux корневой каталог обозначается как / (именно / - слэш, а не \ - обратный слэш). Система позволяет устанавливать много корневых каталогов. Так например для некоторого пользователя ftp /home будет корневым каталогом и при обращении к клиенту ftp на смену каталога на корневой пользователь будет попадать в /home.

Возникает вопрос, а как тогда разные физические устройства участвуют в формировании единой файловой системы? Сделаем небольшой экскурс в историю. В то время, когда создавалась ОС Юникс устройства – накопители информации представляли собой ящик размером с письменный стол и назывались магнитными барабанами. В то время не было необходимости подключать и отключать его по нескольку раз в час. Поэтому не был выработан и механизм быстрой смены. Для подключения любого устройства к файловой системе используется так называемая ***точка монтирования*** – каталог, все вложенные уровни которого являются файловой системой на устройстве-носителе. Например, при ***монтировании*** дискеты обычно используется каталог /media/floppy. То есть, все каталоги и файлы, находящиеся внутри /media/floppy на самом деле содержаться на дискете, вставленной в дисковод компьютера. Для подключения, или монтирования, устройств используется специальная команда, которую мы изучим на следующих занятиях. Таким образом подключаются и сетевые файловые системы, то есть такие системы, которые реально находятся где-то на сервере сети, однако различий в работе с ними пользователь не ощущает и видит сетевые файлы и каталоги, как если бы они были расположены на локальном компьютере.

Есть у файловой системы Linux и еще одна особенность. Каждому пользователю в ней выделяется ***домашний каталог*** – специальный каталог, необходимый для хранения пользователем своих личных данных. При входе пользователя в систему, он сразу оказывается в своем домашнем каталоге. Обычно права доступа к домашнему каталогу пользователя выставлены таким образом, что доступ к каталогу запрещен всем кроме владельца и администратора.

**2.** В файловой системе Линукс различают несколько типов файлов. Понятие «файл» включает в себя также и интерфейсы работы с периферийными устройствами, и каналы, позволяющие разным процессам в системе обмениваться данными.

```
[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
```

3. Навигация по файловой системе является одним из самых важных навыков при работе с операционной системой Linux. Основными командами, используемыми при навигации по файловой системе, являются:

**pwd** – показывает полное имя каталога, в котором находится пользователь.

```
[student@ns student]$ pwd
/home/student
[student@ns student]$_
```

**cd** – изменяет текущий каталог на указанный. cd без параметров или с параметром ~ изменяет текущий каталог на домашний. cd с параметром .. изменяет каталог на тот, который находится на один уровень выше по дереву каталогов.

```
[student@ns student]$ pwd
/home/student
[student@ns student]$ cd primer
```

```
[student@ns primer]$ pwd
/home/student/primer
[student@ns primer]$ cd ..
[student@ns student]$ pwd
/home/student
[student@ns student]$ cd /home/student/primer
[student@ns primer]$ pwd
/home/student/primer
[student@ns primer]$ cd
[student@ns student]$ pwd
/home/student
[student@ns student]$ cd /bin
[student@ns bin]$ pwd
/bin
[student@ns bin]$ cd ~
[student@ns student]$ pwd
/home/student
[student@ns student]$_
```

**pushd, popd** – эти команды работают в связке. Команда pushd изменяет каталог на указанный. pushd с параметром .. изменяет каталог на тот, который находится на один уровень выше по дереву каталогов. Основное отличие этой команды от cd в том, что вся история смены каталогов запоминается в стек и потом может быть использована для быстрой обратной навигации с помощью команды popd.

```
[student@ns student]$ pushd /var
/var ~
[student@ns var]$ pushd log
/var/log /var ~
[student@ns log]$ popd
/var ~
[student@ns var]$ popd
~
[student@ns student]$
```

4. Команда **cp** используется для копирования файлов. Её синтаксис таков:
*cp [параметры] <имя файла источника> <имя каталога приемника>*

Наиболее часто используемым параметром является параметр -R, позволяющий рекурсивно копировать каталоги, т.е со всем их содержимым.

```
[student@ns primer_3]$ cd ../primer_1/in_primer_1
[student@ns in_primer_1]$ ls
[student@ns primer_3]$ cd ../primer_3
[student@ns primer_3]$ cp in_primer_3 ../primer_1/in_primer_1/
[student@ns primer_3]$ cd ../primer_1/in_primer_1
[student@ns in_primer_1]$ ls
in_primer_3
[student@ns primer_2]$ cd ../primer_2
[student@ns primer_2]$ ls
in_primer_2  in_primer_2_2
[student@ns primer_3]$ cp -R * ../primer_2
[student@ns primer_3]$ cd ../primer_2
[student@ns primer_2]$ ls
in_primer_2  in_primer_2_2  in_primer_3
[student@ns primer_2]$
```

Команда **touch** позволяет создавать файлы. Её применение наиболее просто: touch <имя файла>. Если файл с заданным именем существует в текущей директории, команда touch обновит его время создания на текущее.

```
[student@ns lesson_3]$ ls
primer_1  primer_2  primer_3
[student@ns lesson_3]$ touch file
[student@ns lesson_3]$ ls
```

```
file  primer_1  primer_2  primer_3
[student@ns lesson_3]$_
```

Команда **rm** используется для удаления файлов. Основные параметры, используемые с командой rm это -i (удаление с подтверждением удаления), -r (рекурсивное удаление) и -f (удаление всех файлов без подтверждения), -v (подробное описание производимых действий). Параметры -r и -f используются для удаления большого количества файлов. Но при их использовании необходимо быть предельно осторожным, т.к. с помощью этих параметров можно уничтожить систему.

```
[student@ns lesson_3]$ rm -iv ./file
rm: remove `./file'? y
removing `./file'
[student@ns lesson_3]$_
```

**5**. Операции с каталогами также важны для пользователя Linux, как и основные операции с файлами. Основные команды, используемые при работе с каталогами это – rmdir и mkdir.

Команда mkdir позволяет создать каталог:

```
[student@ns student]$ ls
file  primer_1  primer_2  primer_3
[student@ns student]$ mkdir catalog
[student@ns student]$ ls
catalog file  primer_1  primer_2  primer_3
[student@ns student]$_
```

rmdir, наоборот, позволяет удалить каталог:

```
[student@ns student]$ ls
catalog file  primer_1  primer_2  primer_3
[student@ns student]$ rmdir catalog
[student@ns student]$ ls
file  primer_1  primer_2  primer_3
[student@ns student]$_
```

Обращаю ваше внимание на то, что команда rmdir, без использования дополнительных параметров, может удалять ТОЛЬКО ПУСТЫЕ КАТОЛОГИ.

**6**. Файловая система Linux, как и любой другой unix-подобной операционной системы, имеет строгую структуру каталогов. Каждый дистрибутив Linux может несколько изменять структуру в зависимости от предпочтений разработчиков. Мы рассмотрим те каталоги, которые используются в каждом дистрибутиве:

| Имя каталога | Описание |
|---|---|
| /bin | в этом каталоге находятся основные исполняемые файлы, жизненно необходимые для функционирования системы |
| /boot | содержит ядро операционной системы и карты загрузки, а также конфигурационные файлы загрузчиков (lilo, grub) |
| /dev | содержит файлы, которые являются интерфейсом с периферийными устройствами |
| /etc | содержит основные файлы настроек приложений Linux |
| /home | содержит домашние папки пользователей |
| /lib | содержит основные библиотеки, необходимые для нормальной работы системы |
| /lost+found | информация, восстановленная при проверке файловой системы на наличие ошибок |
| /media | точки монтирования отключаемых устройств (usb-диски, CD, floppy) |
| /mnt | точки монтирования ISO-образов, сетевых файловых систем, других |

| | |
|---|---|
| | постоянных файловых систем |
| /opt | альтернатива usr, для коммерческого ПО или ПО, не входящего в основной дистрибутив |
| /proc | внутри этого каталога находится виртуальная файловая система *proc*, создаваемая ядром Linux “на лету”. Содержит общую информацию о системе и подробную о процессах. |
| /root | домашний каталог пользователя root |
| /sbin | утилиты суперпользователя |
| /srv | файлы, выкладываемые для доступа всевозможных внешних служб (например, tftp) |
| /sys | внутри этого каталога также находится виртуальная файловая система, только она содержит подробную информацию о процессах |
| /tmp | в этом каталоге находятся временные файлы, используемые запущенными в данный момент процессами |
| /usr | программы, библиотеки и другие данные пользовательских приложений |
| /var | "Переменная часть" |
| /var/log | содержит файлы журналов |

# Команды mc

Tab - сменить панель
F3 - просмотреть содержимое
F4 - редактировать
F5 - копировать
Shift + F4 - создать файл
ESC (дважды) - выйти
F10 - выйти из mc

# Командная строка

- **history** – Отображает всю историю введенных команд
  - N – N-строк истории коштшманд
  - -d*N* — удалить N-ю строку в истории команд (например введенный пароль)
- **!!** — последняя введенная команда
- **!N** — N-я команда в истории
- **!-N** — команда, введенная N-шагов назад
- **!string —** последняя команда, начинающаяся со **string**
- **!$ —** последнее слово из предыдущего события
- **!?string?** -последняя команда содержащая **string**
- **!!:s/новый/старый —** замена в последней введенной команде, фразы *старый* на *новый*