

Лабораторная работа № 1

**Знакомство
с операционными системами
семейства *nix
на примере ОС ALT Linux Server**

**Командный интерпретатор и
основы программирования на shell**

Основы регулярных выражений

Рассматриваемые темы

- История развития ОС семейства Unix
- Организация файловой системы *nix
- Стандартная структура файловой системы
- Пользователи и процессы в системе
- Права доступа к файлам
- Командный интерпретатор и его роль в системе

- Основные команды системы
- Порядок выполнения команд
- Перенаправление ввода-вывода
- Основы регулярных выражений
- Создание скриптов
- Программирование на shell

- Работа с программными компонентами
- Менеджеры пакетов
- Управление работой демонов

История развития ОС семейства Unix

- 1964 – AT&T, GE и MТИ начали разработку MULTICS
- 1969 – AT&T (Bell Labs) выходит из проекта MULTICS
- 1969 – Ken Thompson, Dennis Ritchie, Douglas McIlroy, первая версия UNIX для PDP-7
- 1971, ноябрь – версия для PDP-11 (Edition 1)
- 1969-1973 – создание C
- 1973 – Edition 4, с ядром на C
- 1975 – Edition 5, полностью на C
- 1974 – распространение по университетам
- 1978 – BSD UNIX
- 1980 – начало коммерциализации UNIX, появление многочисленных ветвей системы
- 1988 – стандартизация систем, POSIX

Свободное программное обеспечение

1983 – Richard Stallman, манифест проекта GNU (GNU's Not Unix)

Свободы пользователей программ:

0. Свобода запускать программу в любых целях
1. Свобода изучения работы программы и адаптации её
2. Свобода распространять копии
3. Свобода улучшать программу и публиковать улучшения

Свободные лицензии:

- GNU General Public License, v. 3 (GNU GPL v.3)
- GNU General Public License, v. 2 (GNU GPL v.2)
- GNU Free Documentation License
- BSD
- MIT
- Artistic (Perl license)
- MPL (Mozilla Public License)
- ...

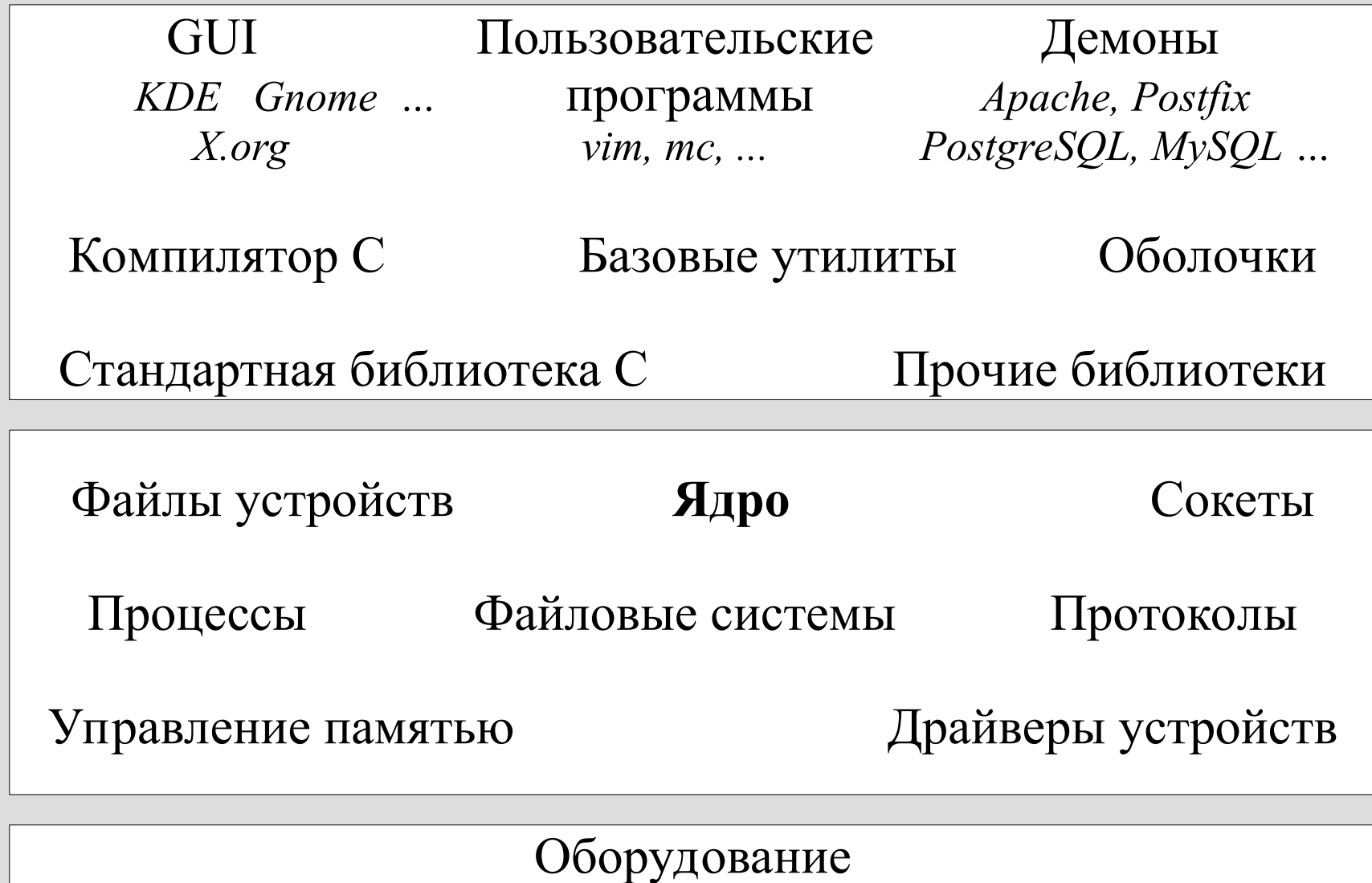
Linux-системы

- 1987 – MINIX
- 1991, сентябрь – анонс разработки Linux
- 1991, 5 октября – 0.02
- 1994, 6 марта – 1.0

- Состав Linux-системы:
ядро + утилиты GNU + сторонние приложения

- Дистрибутивы:
Debian, Ubuntu, RedHat, Fedora, Mandriva, Suse,
Slackware, ALT Linux, ASP Linux

Архитектура Unix-систем



Типы файловых систем

Общего назначения:

- Ext2
- Ext3
- Ext4
- XFS
- JFS
- ReiserFS
- Btrfs
- ISOFS (iso9660)
- UDF
- VFAT
- NTFS

Псевдо-файловые системы

- procfs
- sysfs
- udevfs

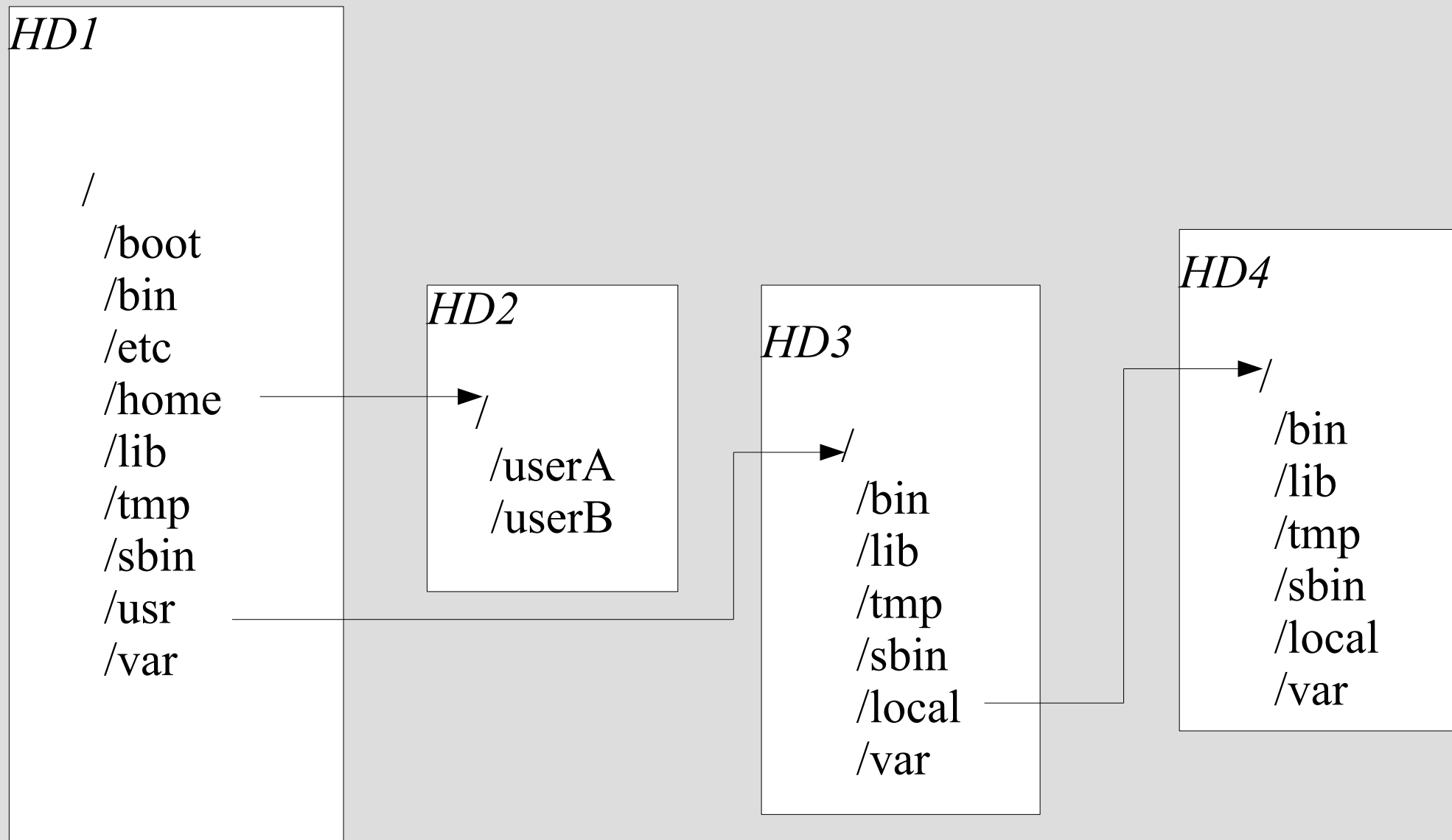
Сетевые файловые системы

- NFS
- CIFS

Специализированные файловые системы

- tmpfs
- jffs2
- squashfs

Стандартная структура файловой системы



Стандартная структура файловой системы

- структура файловой системы стандартизирована
- есть стандартные каталоги для размещения тех или иных файлов

```
$ ls -l /
```

```
bin
boot      ....
dev       proc
etc       root
home      sbin
lib       srv
lib64     sys
lost+found tmp
media     usr
mnt       var
opt
...
```

```
$ ls -l /usr/
```

```
bin
etc
games
include
lib
lib64
libexec
local
sbin
share
X11R6
```

```
$ ls -l /var
```

```
...
empty
lib
local
lock
Log
...
run
spool
tmp
...
```

Пользователи и процессы в системе

- Каждый процесс выполняется с правами определённого пользователя
- Система различает пользователей и группы пользователей
- UID, GID – числовые идентификаторы пользователя и группы
- /etc/passwd – список пользователей в системе
- /etc/group – список групп в системе

- Пользователи делятся на обычных и псевдо-пользователей
- Пользователь с UID=0 – администратор системы
- Традиционное имя для пользователя с UID=0 - root

- Первый процесс в системе – init. Запускается ядром с UID=0, GID=0
- Есть системные вызовы для смены UID и GID
- Изменить свой UID может только процесс с UID=0

Права доступа к файлам

Каждый файл имеет владельца, группу, и права доступа

Права доступа:

- Read – право на чтение из файла
- Write – право на запись в файл
- eXecute – право на выполнение файла

Каталоги также имеют владельца, группу, и права доступа

- Read – право на чтение списка файлов в каталоге
- Write – право на запись в каталог (создание/удаление файлов)
- eXecute – право на переход в каталог

Запись прав:

- rwx – право на чтение, запись и выполнение
- rw- – право на чтение и запись
- r-- – право на чтение и запись

Права доступа к файлам

```
$ ls -l ~
итого 8
drwx----- 2 student student 4096 Фев 19 17:30 Documents
-rw-r--r-- 1 student student 0 Фев 20 08:03 file.txt
drwx----- 2 student student 4096 Фев 19 15:59 tmp
$ ls -l /var
итого 72
drwxr-xr-x 2 root root 4096 Апр 19 2007 adm
drwxr-xr-x 4 root root 4096 Фев 15 08:32 cache
drwxr-xr-x 2 root root 4096 Апр 19 2007 db
dr-xr-xr-x 2 root root 4096 Апр 19 2007 empty
drwxr-xr-x 11 root root 4096 Фев 9 15:29 lib
drwxr-xr-x 14 root root 4096 Фев 20 07:32 log
lrwxrwxrwx 1 root root 10 Фев 5 13:22 mail -> spool/mail
drwxr-x--- 2 root nobody 4096 Апр 19 2007 nobody
drwxrwxrwt 2 root root 4096 Апр 19 2007 tmp
drwxr-xr-x 3 root root 4096 Фев 15 09:24 www
drwx----- 2 root root 4096 Апр 19 2007 yp
$ ls -l /bin/su
-rws--x--- 1 root wheel 23712 Окт 18 2006 /bin/su
```

Командный интерпретатор и его роль в системе

- Обеспечивает пользовательский интерфейс командной строки
- Позволяет пользователю запускать программы
- Предоставляет возможность создания и исполнения файлов с последовательностями команд – скриптов
- Имеет набор встроенных команд
- Часть системных утилит, в т.ч. управления процессами запуска / остановки системы, написаны на языке командного интерпретатора

Существует ряд командных интерпретаторов:
sh, csh, tsh, bash, zsh ...

Общий формат вызова команды выглядит следующим образом:

```
$ command -f --flag --key=parameter argument1 argument2 ...
```

Основные команды системы

Смена каталога:	cd
Просмотр каталога:	ls
Создание каталога:	mkdir
Удаление каталога:	rmdir
Удаление файла:	rm
Изменение даты/создание пустого файла:	touch
Просмотр текстового файла:	cat
Поэкранный просмотр файла:	less
Просмотр начала файла:	head
Просмотр конца файла:	tail
Изменение прав доступа к файлу:	chmod
Изменение владельца файла:	chown
Редактор:	vi / vim

Выполнение программ

- Получение кода возврата:

```
$ echo 'Hello, world!'  
$ echo $?
```

- Последовательное выполнение программ:

```
$ cd; ls
```

- Логическое “И”

```
$ cd /tmp/0 && ls
```

- Логическое “ИЛИ”

```
$ cd /tmp/0 || mkdir /tmp/0
```

Выполнение программ

Фоновый режим выполнения команд:

- Перевести команду из переднего плана в фон:

```
Ctrl+Z; bg
```

- Запустить команду в фоне:

```
$ command &
```

- Получить список команд в фоновом режиме:

```
$ jobs
```

- Вывести команду из фона на передний план:

```
$ fg
```

- Прервать выполнение команды переднего плана:

```
Ctrl+C
```


Выполнение программ

Получить список процессов:

```
$ ps; ps aux
```

Послать сигнал процессу:

```
$ kill -<signal> <pid>
```

Остановить процесс:

```
$ kill -SIGSTOP <pid>
```

Продолжить выполнение:

```
$ kill -SIGCONT <pid>
```

Список сигналов:

```
$ kill -l
```

SIGKILL – уничтожить процесс

SIGTERM – завершить процесс

SIGQUIT - завершить процесс

SIGSTOP – остановить процесс

SIGCONT - продолжить процесс

ПОТОКИ ВВОДА-ВЫВОДА

Стандартные потоки ввода-вывода:

- `STDIN` – стандартный поток ввода
- `STDOUT` – стандартный поток вывода
- `STDERR` - стандартный поток ошибок

ПОТОКИ ВВОДА-ВЫВОДА

Перенаправление потоков ввода-вывода

STDOUT

Вывод в файл:

```
$ cat > file
```

Запись в конец файла:

```
$ cat >> file
```

STDIN

Ввод из файла:

```
$ cat < file
```

Ввод до разделителя:

```
$ cat <<END
```

```
Hello, world!
```

```
END
```

Ввод из файла и вывод в файл:

```
$ cat <file >file1
```

ПОТОКИ ВВОДА-ВЫВОДА

Конвейеры:

```
$ ls | sort  
$ cat file | head -n 10 | tail -n 5  
$ cat file | grep 'http://'
```

Устройства для перенаправления потоков ввода-вывода:

`/dev/null` - “пустое” устройство, в которое можно
записывать

`/dev/zero` - “нулевое” устройство, из которого можно
прочитать нули

Регулярные выражения

- Язык описания шаблонов текста
- Позволяют:
 - проверить наличие заданного шаблона в тексте
 - выделить в соответствии с шаблоном одну или несколько подстрок из текста

Простейшие шаблоны:

шаблон	-	совпадение с подстрокой 'шаблон'
(шаблон)	-	совпадение и выделение совпавшего текста

Регулярные выражения

<code>one two</code>	- 'one' или 'two'
<code>(one) {1, 2}</code>	- одна или две подстроки 'one'
<code>(one) {1, }</code>	- одна или больше подстроки 'one'
<code>(one) {, 2}</code>	- от нуля до двух подстрок 'one'
<code>(one) +</code>	- одна или больше подстроки 'one'
<code>(one) ?</code>	- ноль или одна подстрока 'one'
<code>(one) *</code>	- сколько угодно подстрок 'one'
<code>к (и о) т</code>	- 'КИТ' или 'КОТ'
<code>ки+т</code>	- 'КИТ', 'КИИИТ', ...
<code>ки?о?т</code>	- 'КТ', 'КИТ', 'КОТ', 'КИОТ'
<code>кии*т</code>	- 'КИТ', 'КИИИТ', ...

Регулярные выражения

- - любой символ
- [abc] - перечисление символов
- [^abc] - символы, кроме перечисленных
- [a-d] - диапазоны символов
- ^ - начало строки
- \$ - конец строки

- ^[A-Z][a-z]+ - слово с заглавной буквы, в начале строки.
- \.\$ - строки, кончающиеся на точку.

Регулярные выражения:

- жадные - $a \cdot^* a$ → лабораторная
- ленивые - $a [^a]^+$ → лабораторная

Утилита `grep`

`grep` - фильтр текста.

- \$ `grep` шаблон [файл] - ПОИСК И ВЫВОД СОВПАДАЮЩИХ СТРОК
- \$ `grep -v` шаблон [файл] - ПОИСК И ВЫВОД НЕ СОВПАДАЮЩИХ СТРОК

Примеры использования:

- \$ `ls /bin | grep '^ [a-c] . * a '`
- \$ `ls /bin | grep '^ [a-b] . * [n-z] $ '`
- \$ `grep -v '^ * \ (# \ | $ \) '`
- \$ `grep -E -v '^ * (# | $) '`
- \$ `egrep -v '^ * (# | $) '`

Утилита sed

sed – строковый редактор

Поиск и замена текста с sed:

```
$ sed 's/шаблон/замена/[ig]'
```

Примеры:

```
$ date
```

```
Пнд Окт 13 09:55:26 MSK 2014
```

```
$ date | sed 's/Окт/Янв/'
```

```
Пнд Янв 13 09:55:56 MSK 2014
```

```
$ date | sed 's/Окт/Янв/' | sed 's/^[^ ]\+ \+//'
```

```
Янв 13 09:56:41 MSK 2014
```

```
$
```

Утилита awk

awk – скриптовый язык обработки текстовой информации

Общий вид программы awk:

```
$ awk '/шаблон/ {действие;} /шаблон/ {действие;} ...'
```

Примеры:

```
$ ls -l /bin | head -4
```

```
total 5596
```

```
lrwxrwxrwx 1 root root          4 Feb 25 05:30 awk -> gawk
-rwxr-xr-x 1 root root    19064 Apr 20  2008 basename
-rwxr-xr-x 1 root root  549368 Mar 27  2008 bash
```

```
$ ls -l /bin | awk '/^-/ {print $9"\t->\t"$3":"$4"\t"$1;}' \
| head -5
```

```
basename      ->          root:root          -rwxr-xr-x
bash          ->          root:root          -rwxr-xr-x
bzip2         ->          root:root          -rwxr-xr-x
bzip2recover  ->          root:root          -rwxr-xr-x
cat           ->          root:root          -rwxr-x
```

Скрипты

- Текстовые файлы с последовательностями команд:
 - необходимо указать, что это программа:
 - право выполнения.
- Могут быть на разных программных языках:
 - необходимо указать shell как интерпретатор:
 - специальный формат первой строки файла

```
$ cat >hello.sh <<END
#!/bin/sh
echo 'Hello, world!'
END
$ chmod a+x hello.sh
$ ./hello.sh
Hello, world!
$
```

Переменные shell

Переменные:

- окружения
- пользователя

Список переменных:

```
$ set
```

Присваивание значений:

```
$ A=10  
$ A=A  
$ A='Текст с пробелами'  
$ A="Текст с переменными"
```

Использование значений:

```
$ B=$A  
$ C="B=$B"  
$ echo $C, "C=$C"
```

Запись вывода команды в переменную:

```
$ DATE=`date`; echo $DATE  
$ A=`ls /bin | grep '^bash' | head -n 1`; echo $A
```

Управляющие структуры shell

Условное выполнение:

```
if ... ; then ....; else ...; fi
```

```
$ if /bin/true; then echo 'True'; else echo 'False'; fi  
True
```

```
$ if /bin/false; then echo 'True'; else echo 'False'; fi  
False
```

```
$ if ls /bin/ | grep -q 'true'; then  
    echo 'True'  
else  
    echo 'False'  
fi
```

```
$ /bin/true && echo 'True'  
$ /bin/false || echo 'False'
```

Управляющие структуры shell

Проверка условий: `/usr/bin/test ; /usr/bin/[`

- Для файлов:

- | | | |
|---------------------------|---|---------------------------------|
| <code>-f /bin/bash</code> | - | файл существует |
| <code>-x /bin/bash</code> | - | файл есть и может быть выполнен |
| <code>-r /bin/bash</code> | - | файл есть и может быть прочитан |
| <code>-w ~/.bashrc</code> | - | файл есть и может быть записан |
| <code>-d /bin</code> | - | каталог существует |

Пример:

```
$ [ -w ~/.bashrc ] && echo "yes"
```

```
$ if [ -w ~/.bashrc ]; then echo "yes"; fi
```

Управляющие структуры shell

Проверка условий:

- Для чисел:

'10' -eq '10'	-	равно
'10' -ne '5'	-	не равно
'10' -gt '5'	-	больше
'10' -lt '20'	-	меньше

- Для строк:

-n 'строка'	-	строка не пустая
-z ''	-	строка пустая
'строка' == 'строка'	-	строки совпадают
'Строка' != 'строка'	-	строки не совпадают

Управляющие структуры shell

Циклы:

```
while команда; do список команд; done  
until команда; do список команд; done  
for переменная in список значений; do команды; done
```

Примеры:

```
$ while /bin/true; do echo "Y"; sleep 1s; done  
$ until /bin/false; do echo "N"; sleep 1s; done  
  
$ for i in `ls /bin`; do echo $i; done  
$ for i in `seq 1 10`; do echo $i; done
```


Работа с программными компонентами

Основная часть программного обеспечения доступна в виде исходных кодов.

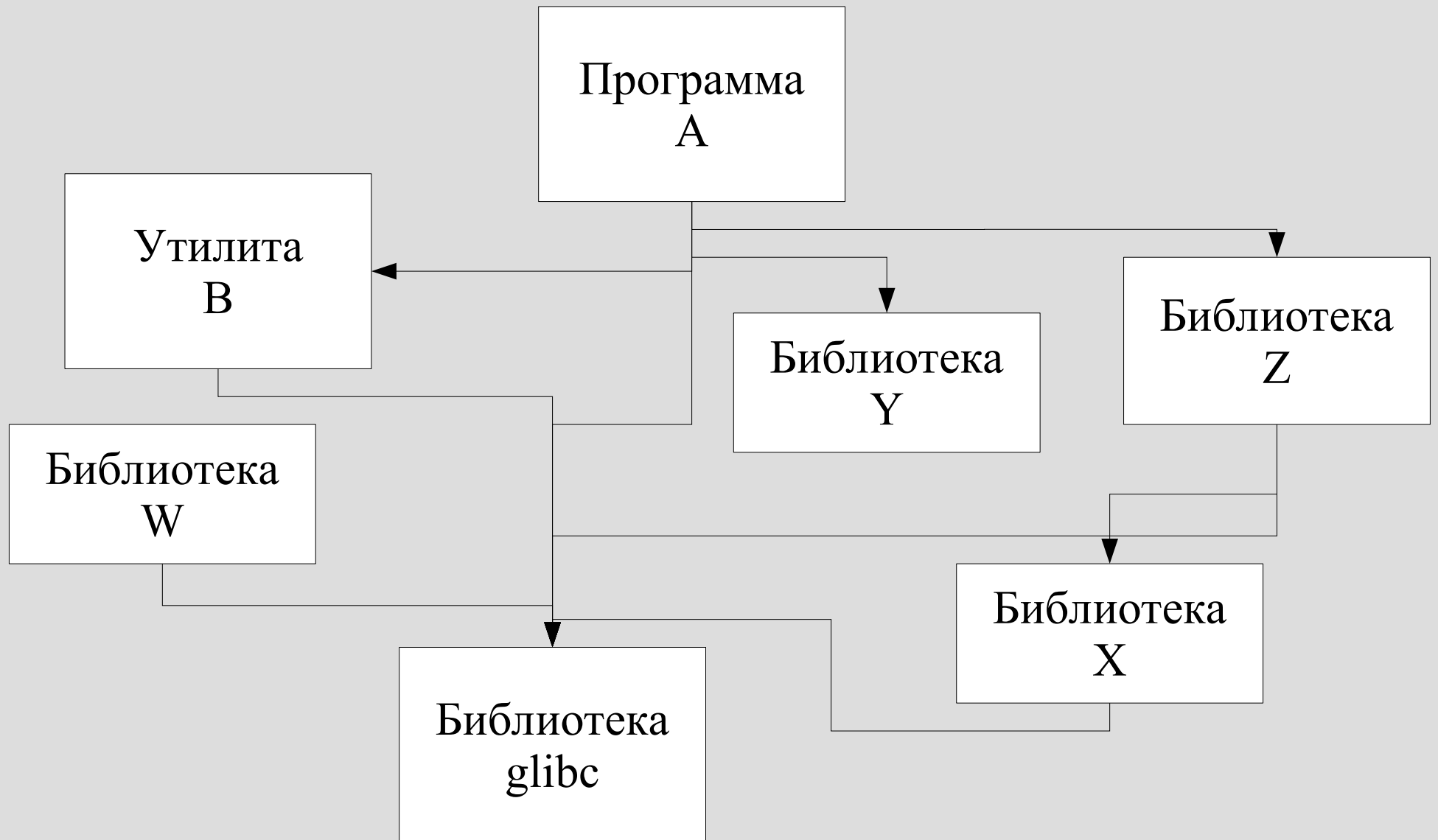
Дистрибутив – набор программных пакетов, настроенных для совместной работы в системе.

Существует большое число различных дистрибутивов:

- общего назначения: Debian, Ubuntu, RedHat, Fedora, Suse, Mandriva, ALT Linux, ASP Linux.
- специального назначения: OpenWRT, RIP, DBAN, ...

Как правило, программные пакеты используют другие программные пакеты – в виде разделяемых библиотек, внешних исполняемых файлов, и т.п.

Работа с программными компонентами



Менеджеры пакетов

Менеджеры пакетов:

- .deb – dpkg (Debian Package manager) - Debian, Ubuntu
- .rpm – RPM Package Manager – RedHat, Fedora, Suse, ALT Linux, ...

Основная утилита RPM – rpm

RPM отслеживает зависимости в отдельных пакетах .rpm.

Совокупность пакетов – репозиторий.

В репозитории (в идеале) зависимости между пакетами замкнуты.

В ALT Linux для организации репозитория используется

APT (Advanced Packaging Tool)

Менеджеры пакетов

Настройки АРТ: `/etc/apt/sources.list`, `/etc/apt/sources.list.d/*`

Запись о репозитории:

```
#rpm [updates] ftp://ftp.altlinux.org/pub/distributions/ALTLinux/p9/branch x86_64 classic  
#rpm [updates] ftp://ftp.altlinux.org/pub/distributions/ALTLinux/p9/branch noarch classic
```

Работа с АРТ:

```
# apt-get update  
# apt-get dist-upgrade  
# apt-get install <package>  
# apt-get remove <package>  
# apt-cache search <название>
```

Управление работой ДЕМОНОВ

Демон – традиционное название для неинтерактивных программ.

Управлением порядком загрузки занимаются системы инициализации: `systemd`, `sysvinit`, `Upstart`, `Runit`, `Launchd`, `Initng`, ...

Уровни загрузки системы для `sysvinit`:

0 — уровень остановки системы

1 — однопользовательская система

2 — многопользовательская система без сетевой поддержки

3 — многопользовательская система

4 — предоставлено для конкретных систем

5 — многопользовательская система с поддержкой графики

6 — уровень перезагрузки системы

Переход между уровнями осуществляет `init`

Выполнение команд в заданное время

- Запуск программ в нужное время обеспечивает демон `crond`.
- Получить настройки `crond` для пользователя:
`$ crontab -l`
- Изменить настройки `crond` для пользователя:
`$ crontab -e`

Задание на лабораторную работу

- Выполнить удалённую регистрацию в системе.
- Провести ознакомление с операционной системой. Изучить структуру каталогов сервера, посмотреть доступные команды в системе, вызвать справочное руководство. Создать текстовый файл, используя редактор `vi`.
- Используя команду `su`, получить привилегии суперпользователя.
- Изменить пароли пользователя и суперпользователя системы.
- Создать новую учётную запись пользователя.
- Зарегистрироваться в системе под созданным пользователем, убедиться в возможности использования им команды `su`.
- Удалить учётную запись пользователя `student`.
- Изучить список пакетов, установленных в системе.
- Настроить список репозиториев пакетов для системы *APT*.
- Провести обновление системы до текущего состояния репозитория.
- Установить веб-сервер `lighttpd`, запустить сервер.
- Проверить работу веб-сервера. Настроить его автоматический запуск при загрузке системы.

Задание на лабораторную работу

- Перезагрузить систему. Убедиться, что веб-сервер `lighttpd` автоматически запустился после перезагрузки системы.
- Доставить в систему всё необходимое для работы скриптов сбора и отображения статистики программное обеспечение.
- Адаптировать приведённые в описании работы скрипты, получающие значения статистических параметров и записывающие их в журналы.
- Обеспечить периодическое регулярное выполнение скриптов.
- Адаптировать приведённые в описании работы скрипты для отображения записываемых данных из журналов, обеспечить их выполнение из командной строки.
- Настроить `lighttpd` для удалённого обращения из браузера к указанным скриптам и отображения собираемых данных в веб-браузере на удалённом рабочем месте.
- Обеспечить безопасное выполнение скриптов.

Формат данных к лабораторной работе

Виртуальный сервер: lab-99.edu.cbias.ru

VEID: 230199

Login: student
Password: password

Доступ к виртуальному серверу по SSH:

Имя сервера: ssh.edu.cbias.ru

Номер порта: 22199

Доступ к серверу по HTTP: <http://lab-99.edu.cbias.ru>

Установленная в виртуальном сервере система:

ALT Linux Server 9.1 x86_64 (branch p9)

Список репозиториев:

rpm [p9] <ftp://ftp-distr/ALTLinux/M90P/branch> x86_64 classic

rpm [p9] <ftp://ftp-distr/ALTLinux/M90P/branch> noarch classic

rpm <ftp://ftp.ossrg.ru/APT/APT> x86_64 ALT_M90P OSSG_M90P

rpm <ftp://ftp.ossrg.ru/APT/APT> noarch ALT_M90P OSSG_M90P

Описания лабораторных работ и прочие материалы:

<http://edu.cbias.ru>

Вопросы, комментарии, предложения: edu-2020@cbias.ru