Лекция по курсу СПО

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 MTI 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
- Artisitic (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, CentOS, Mandriva, Suse, Slackware, ALT Linux, ASP Linux, ASTRA Linux, ...

Операционные системы

ЭВМ:

- Процессор
  - Арифметически-логическое устройство
  - Регистры
- ОЗУ
- Периферийные устройства

Операционные системы

- Однозадачные
- Многозадачные
  - Кооперативная многозадачность
  - Вытесняющая многозадачность

Операционные системы

Операционные системы

- Однопользовательские
- Многопользовательские
  - 1 пользовательский сеанс + фоновые задачи
  - N пользовательских сеансов + фоновые задачи

Unix – многозадачная многопользовательская ОС

Планировщик процессов:

- Текущее состояние выполнения программы
- Пользователь
- Группа пользователей
- ….

Архитектура Unix-систем

GUI *KDE Gnome ...* *X.org*	Пользовательские программы *vim, mc, ...*	Демоны *Apache, Postfix* *PostgreSQL, MySQL ...*
Компилятор C	Базовые утилиты	Оболочки
Стандартная библиотека C		Прочие библиотеки

Файлы устройств	**Ядро**	Сокеты
Процессы	Файловые системы	Протоколы
Управление памятью		Драйверы устройств

Оборудование

Дисковые устройства

Накопители данных:

- Блочные устройства
- HDD, SSD, оптические диски, сетевые диски, виртуальные диски, дисковые массивы, …

Таблицы разделов: MBR, GPT.

Дисковые массивы: RAID-0, RAID-1, RAID-5, RAID-6, др.

Менеджеры дисковых томов: LVM

Типы файловых систем

Общего назначения:

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

Псевдо-файловые системы

- procfs
- sysfs
- udevfs

Сетевые файловые системы

- NFS
- CIFS

Специализированные файловые системы

- tmpfs
- jffs2
- squashfs

Устройство файловых систем

FAT:
- таблица размещения файлов
- корневой каталог
- подкаталоги

EXT2:
- inode:
  - список блоков с данными файла
  - метаинформация: владелец, группа владельца, права доступа, даты создания, изменения и последнего доступа, количество ссылок на файл, расширенные аттрибуты
- каталоги:
  - имя файла и его тип
  - номер inode для файлов и каталогов

Устройство файловых систем

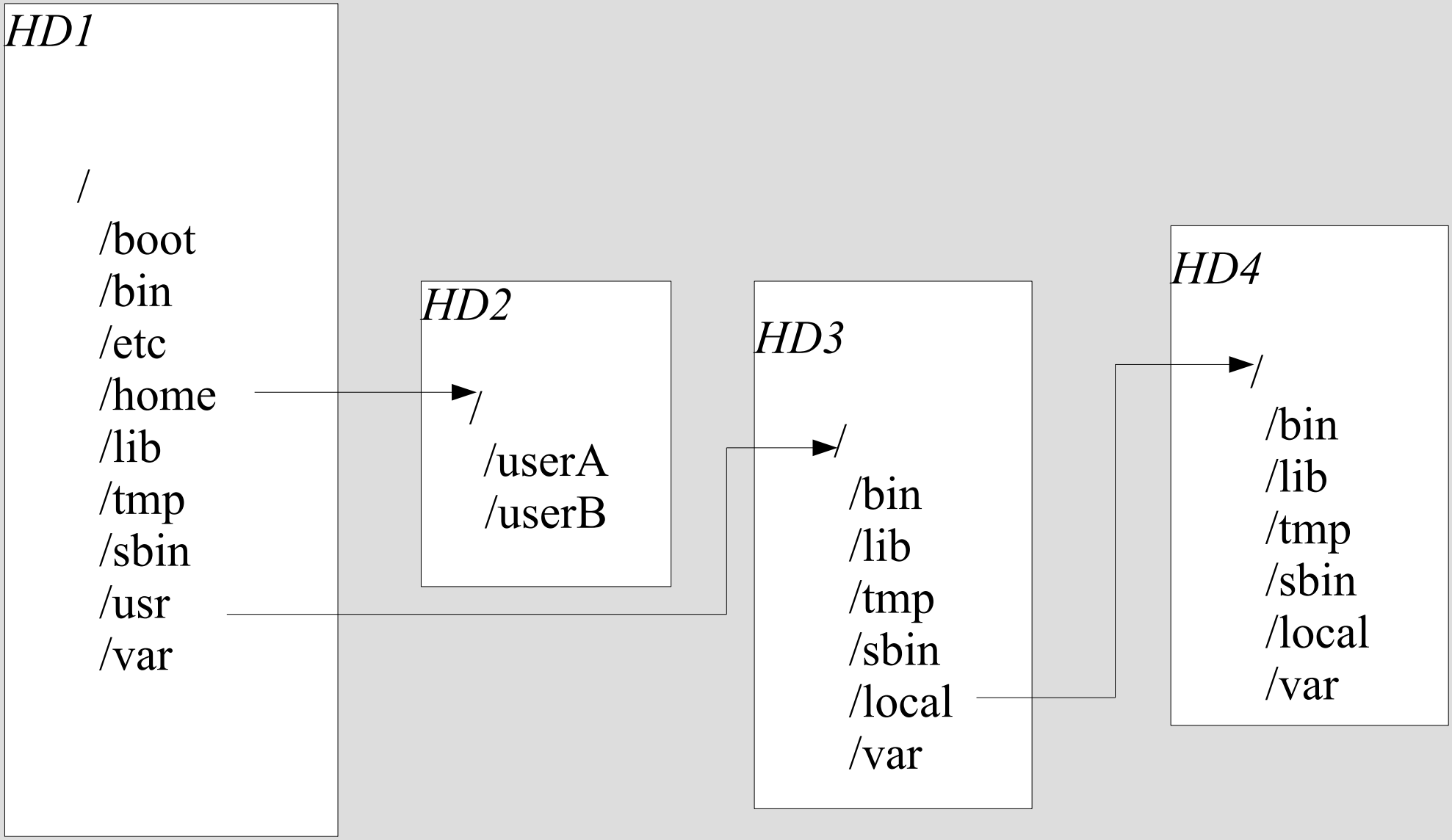
Типы файлов:
- обычный файл (file)
- каталог (directory)
- символьная ссылка (soft link)
- устройство (device): символьное, блочное (char, block)
- именованные каналы ввода-вывода (named pipe)
- сокеты (socket)
- двери (door)

Жёсткие ссылки (hard link) – записи в каталогах для inode

Отказоустойчивость файловых систем:
- ➔ Журналируемые файловые системы: ext3, ext4, XFS, …
- ➔ Файловые системы CoW: BtrFS, XFS+расширение reflink

Фрагментация файловых систем
Снимки состояния файловых систем

Стандартная структура файловой системы



Стандартная структура файловой системы

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

```
$ ls -1 /
bin
boot
dev
etc
home
lib
lib64
lost+found
media
mnt
opt
...
....
proc
root
sbin
srv
sys
tmp
usr
var
```

```
$ ls -1 /usr/
bin
etc
games
include
lib
lib64
libexec
local
sbin
share
X11R6
```

```
$ ls -1 /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
```

Права доступа к файлам

Биты доступа:

- SUID (Set User-ID)
- SGID (Set Group-ID)
- Sticky bit

```
$ ls -l /bin/su
-rws--x--- 1 root wheel 23712 Окт 18  2006 /bin/su

$ ls -l /usr/bin/crontab
-rwx--s--x 1 root crontab 39424 июн 30  2022 /usr/bin/crontab

$ ls -ld /var/spool/cron/
drwx-ws--T 2 root crontab 4096 фев 14 12:42 /var/spool/cron/
# ls -l /var/spool/cron/
total 4
-rw------- 1 webadmin crontab 1428 Feb 25 20:58 webadmin

$ ls -ld /tmp
drwxrwxrwt 21 root root 420 мар 16 01:11 /tmp
```

Общий вид POSIX-программы

C:

```
int main(int argc, char* argv[], char* env[]) {
...
get();
put();

return 0;
}
```

Pascal:

```
program main;
begin
    Writeln(...);
    Readln(...);
end.
```

Общий вид POSIX-программы

При запуске программе предоставляется:
- массив аргументов командной строки,
- массив переменных окружения,
- открытые файловые дескрипторы STDIN, STDOUT, STDERR.

По окончанию выполнения программа выдаёт код возврата.

Программа может использовать внешние (динамические) библиотеки:
- стандартные библиотеки POSIX,
- дополнительные библиотеки

Общий вид запуска программ

```
int main(void)
{
 pid_t pid = fork();

 if (pid == -1){ perror("fork failed"); exit(1);}
 else
   if (pid == 0) {
      printf("Hello from the child process!\n");
      exit(0);
   } else {
      int status;
      (void)waitpid(pid, &status, 0);
   }
 return 0;
}
```

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 agrument2 ...
```

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

Списки открытых файловых дескрипторов процесса – в /proc/<pid>/fd/:

```
# cat /proc/33943/cmdline ;ls -l /proc/33943/fd/
/bin/bash
total 0
lrwx------ 1 root root 64 Mar 16 02:27 0 -> /dev/pts/0
lrwx------ 1 root root 64 Mar 16 02:27 1 -> /dev/pts/0
lrwx------ 1 root root 64 Mar 16 02:27 2 -> /dev/pts/0
lrwx------ 1 root root 64 Mar 16 02:27 255 -> /dev/pts/0

[root@lab-00 ~]# cat /proc/1905/cmdline; ls -l /proc/1905/fd/
/usr/sbin/lighttpd -D -f /etc/lighttpd/lighttpd.conf
total 0
lrwx------ 1 root root 64 Mar 16 02:27 0 -> /dev/null
lrwx------ 1 root root 64 Mar 16 02:27 1 -> /dev/null
lrwx------ 1 root root 64 Mar 16 02:27 2 -> 'socket:[550039]'
l-wx------ 1 root root 64 Mar 16 02:27 3 -> /run/lighttpd.pid
lrwx------ 1 root root 64 Mar 16 02:27 4 -> 'socket:[550043]'
l-wx------ 1 root root 64 Mar 16 02:27 5 -> /var/log/lighttpd/error.log
l-wx------ 1 root root 64 Mar 16 02:27 6 -> /var/log/lighttpd/access.log
lrwx------ 1 root root 64 Mar 16 02:27 7 -> 'anon_inode:[eventpoll]'
lr-x------ 1 root root 64 Mar 16 02:27 8 -> 'pipe:[550052]'
l-wx------ 1 root root 64 Mar 16 02:27 9 -> 'pipe:[550052]'
```

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

`one|two`	-	'one' или 'two'
`(one){1,2}`	-	одна или две подстроки 'one'
`(one){1,}`	-	одна или больше подстроки 'one'
`(one){,2}`	-	от нуля до двух подстрок 'one'
`(one)+`	-	одна или больше подстроки 'one'
`(one)?`	-	ноль или одна подстрока 'one'
`(one)*`	-	сколько угодно подстрок 'one'
`к(и|о)т`	-	'кит' или 'кот'
`ки+т`	-	'кит', 'киииит', ...
`ки?о?т`	-	'кт', 'кит', 'кот', 'киот'
`кии*т`	-	'кит', 'киииит', ...

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.*a` → л**абораторна**я
- ленивые – `a[^a]+` → л**абор**аторная

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:

`-f /bin/bash`	-	файл существует
`-x /bin/bash`	-	файл есть и может быть выполнен
`-r /bin/bash`	-	файл есть и может быть прочитан
`-w ~/.bashrc`	-	файл есть и может быть записан
`-d /bin`	-	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

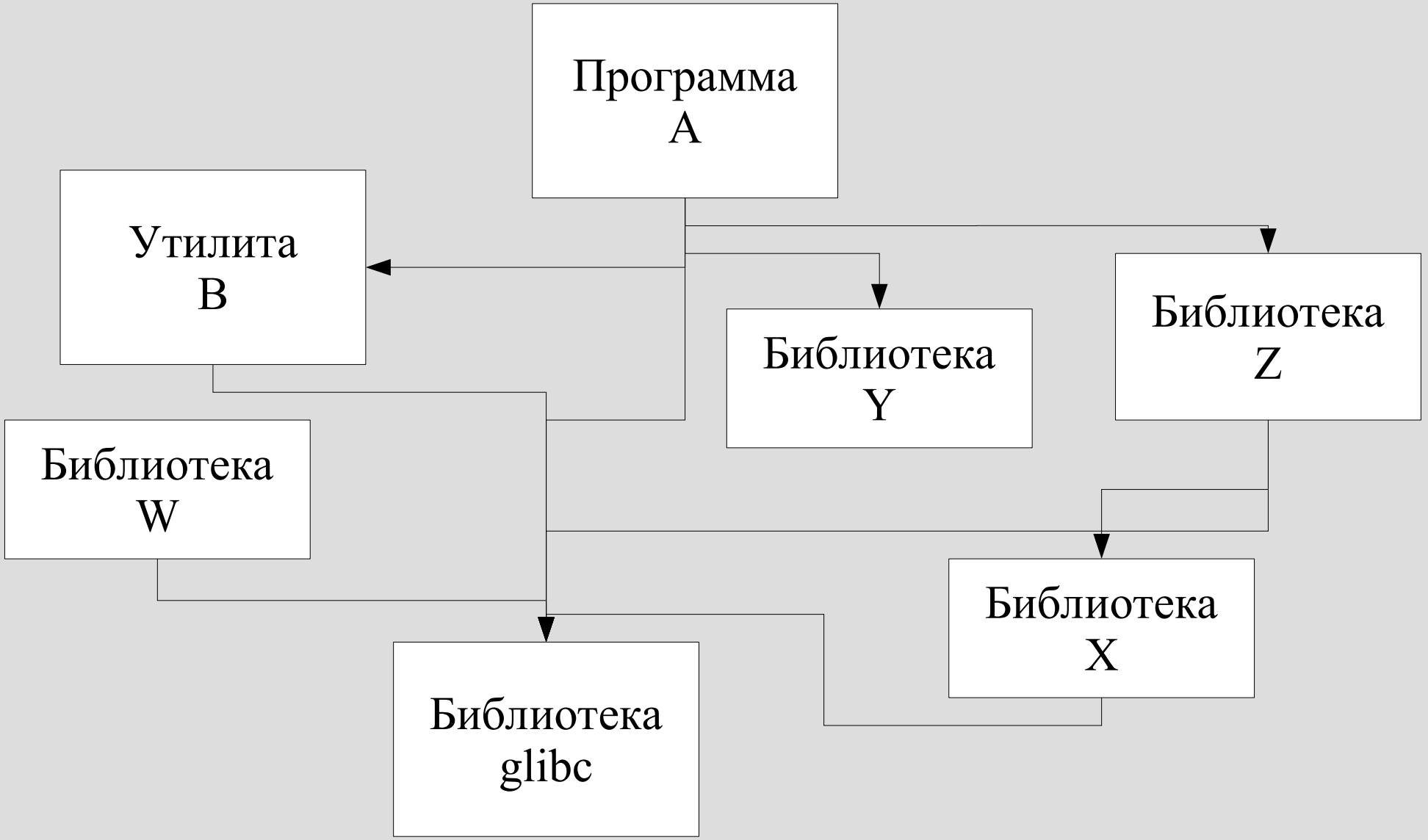
Основная часть программного обеспечения доступна в виде исходных кодов.

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

Существует большое число различных дистрибутивов:
- общего назначения: Debian, Ubuntu, RedHat, CentOS, Arch, Mandriva, ALT Linux, ASP Linux, ASTRA Linux.
- специального назначения: Mobian, Manjaro, 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 APT: /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 APT:

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

Порядок загрузки системы

- Подача питания на процессор
- Инициализация процессора
- Запуск на выполнение кода по фиксированному адресу из ПЗУ
- Определение, проверка и инициализация ОЗУ, основых устройств
- Опционально - инициализация адаптеров периферийных устройств
- Определение загрузочного накопителя

- Чтение и запуск первичного загрузчика
- Инициализация загрузочного накопителя
- Чтение и запуск вторичного загрузчика
- Выбор варианта и параметров загрузки операционной системы

- Загрузка и запуск ядра операционной системы
- Инициализация внутренних структур ядра
- Поиск и инциализация основных устройств системы
- Запуск первого процесса – init, PID=0

Системы инициализации: sysvinit

Управлением порядком загрузки занимаются системы инициализации: systemd, sysvinit, Upstart, Runit, Launcd, 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 командой init

Демон – традиционное название для неинтерактивных программ.
Запуск

Система инициализации sysvinit

sysvinit:
- действия при смене уровней загрузки выполняются скриптами командного интерпретатора
- размещение скиптов – в /etc/rc.d/
- запуск демонов – скриптами из /etc/rc.d/init.d/ (/etc/init.d),
- порядок запуска/остановки демонов для соответствующего уровня загрузки – символьные ссылки на скрипты демонов в /etc/rc.d/rc<N>.d/ ,
- запуск и остановка демонов выполняется последовательно.

Включение / выключение автоматической загрузки:

```
# chkconfig <daemon> on; chkconfig <daemon> off
```

Добавление / удаление ссылок в /etc/rc.d/rc<N>.d/:

```
# chkconfig <daemon> --add; chkconfig <daemon> --del
```

Запуск / остановка / состояние демона в ручном режиме:

```
# service <daemon> start; service <daemon> stop
# service <daemon> status
```

Система инициализации systemd

systemd:
- действия выполняются отдельным демоном systemd;
- описание действий – файлы конфигурации;
- порядок запуска/остановки указывается относительно других сервисов;
- запуск и остановка демонов выполняется по-возможности параллельно;
-

Включение / выключение автоматической загрузки:

```
# systemctl enable  [--now] <daemon>
# systemctl disable [--now] <daemon>
```

Запуск / остановка / состояние демона в ручном режиме:

```
# systemctl start  <daemon>; systemctl stop <daemon>
# systemctl status <daemon>
```

Помимо собственно системы инициализации в состав systemd входят: journald, udevd, logind, networkd, containerd, ...

Системные журналы

sysvinit: syslogd / syslogd-ng / rsyslogd
- отдельный независимый от sysvinit демон, получающий от программ и записывающий в файлы журналов записи;
- файлы журналов – текстовые файлы;
- просмотр журналов – стандартными утилитами;
- для ротации журналов требуется отдельный сервис logrotate, запускающийся по расписанию.

systemd: journald.
- входит в состав systemd;
- файлы журналов – бинарные, сжаты, с расширенной информацией о записях;
- просмотр журналов – отдельной утилитой journalctl;
- ротация журналов ведётся journald.

Демоны и приложения: могут использовать syslogd / journald, и/или вести журналы самостоятельно.

Выполнение команд в заданное время

- Запуск программ в нужное время обеспечивает демон crond.
- Получить / изменить настройки crond для пользователя:

```
$ crontab -l; crontab -e
```

- Формат файла crontab:

```
SHELL=/bin/sh
MAILTO=user@domain.tld
5   0   * * *    $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
15 14   1 * *    $HOME/bin/monthly
23 */2  * * *    echo "run at 00:23,02:23, ..."
5  4    * * sun  echo "run at 04:05 every sunday"
@reboot  $HOME/bin/init_after_reboot
```

- Файлы расписаний процессов пользователей: /var/spool/cron/
- Файлы расписаний для системных процессов: /etc/crontab, /etc/cron.d/, /etc/cron.daily/, /etc/cron.hourly/, /etc/cron.weekly/, /etc/cron.montly/ .