

Оболочка Linux

2022

Пользователи

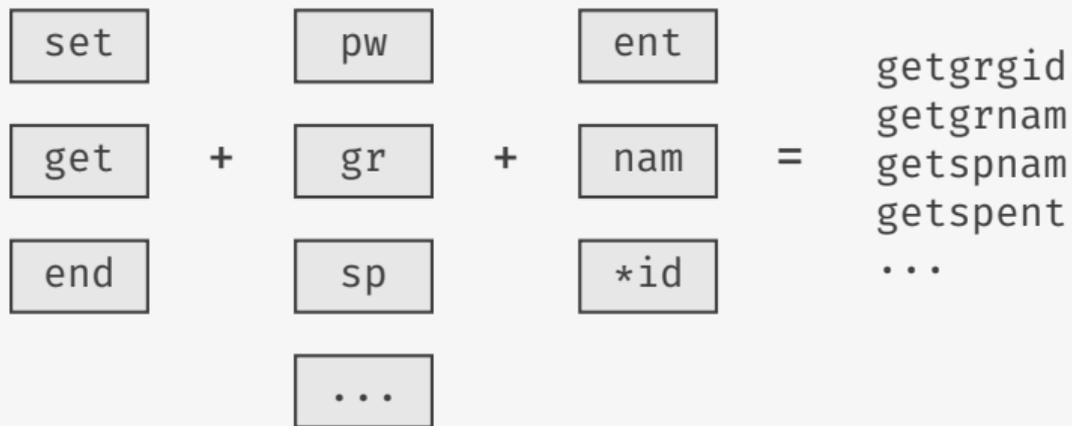
Вывести номера и имена всех пользователей:

```
setpwent(); // инициализировать базу данных пользователей
while (passwd* pw = getpwent()) { // получить все поля из БД
    std::cout << pw.pw_uid << ' ' << pw.pw_name << '\n';
}
endpwent(); // освободить ресурсы
```

Получить пользователя по номеру/имени:

```
passwd* pw = getpwnam("myuser"); // по имени
pw = getpwuid(1234); // по номеру
```

Базы данных



Пароли

```
$ cat /etc/passwd
myuser:x:1234:1234:My user:/home/myuser:/bin/bash
$ cat /etc/group
myuser:x:1234:
$ cat /etc/shadow
myuser:$6$YLLurBZUcMMbZxog$/19sjyJE1MWrNzNRTgAJ2...::0:99999:7:::
```

ID

соль

пароль (86 символов)

Права доступа

```
$ ls -l /etc/passwd /etc/group /etc/shadow
-rw-r--r-- 1 root root ... /etc/passwd
-rw-r--r-- 1 root root ... /etc/group
-- -- -- 1 root root ... /etc/shadow
```

права
пользователя

права
группы

права
остальных

владелец
(группа)

владелец
(пользователь)

Права доступа

```
struct stat st;  
stat("/etc/passwd", &st);  
std::cout << std::oct << st.st_mode << '\n';           // 100644  
std::cout << std::oct << (st.st_mode & 0777) << '\n'; // 644
```

Оболочка

Команда входа в систему:

```
$ login
mycomputer login: ...
...
[myuser@mycomputer ~]$
```

Выбор оболочки:

```
$ cat /etc/passwd
myuser:x:1234:1234:My user:/home/myuser:/bin/bash
$ cat /etc/shells
/bin/sh          # оболочка POSIX
/bin/bash        # POSIX-совместимая оболочка
/sbin/nologin    # вход запрещен
...
```

Базовые команды

```
$ cd /tmp           # перейти в директорию
$ pwd              # текущая директория
$ ls               # список файлов
$ cat FILE         # содержимое файла
$ echo Hello       # сообщение
$ test "123" = "456" # проверка равенства строк
$ echo $?         # результат проверки
```

Все команды:

```
$ ls /usr/bin /bin
...
[
cd
pwd
ls
...
```

Внешние команды

Встроенные команды:

```
x=10
y=20
if test "$x" = "$y"
then
    echo "Equal"
else
    echo Not equal
fi
```

Внешние команды:

```
x=10
y=20
if /bin/test "$x" = "$y"
then
    /bin/echo "Equal"
else
    /bin/echo Not equal
fi
```

Директории для поиска команд:

```
$ echo $PATH
/usr/local/bin:/usr/bin
```

Циклы

Список всех файлов SVG:

```
ls *.svg
```

Преобразования все файлов SVG в EPS:

```
for i in *.svg
do
    inkscape -z -E $i.eps $i
done
```

Пользовательский ввод:

```
answer=
while test "$answer" != "yes" && test "$answer" != "no"
do
    echo "Delete? [yes/no]"
    read answer
done
```

Аргументы скрипта

Скрипт:

```
#!/bin/sh
while test -n "$1"
do
    case "$1" in
        -n|--nodes)
            shift
            num_nodes=$1
            shift
            ;;
        *)
            echo "Bad argument"
            exit 1
            ;;
    esac
done
```

Запуск скрипта:

```
$ chmod +x script.sh
$ ./script --nodes 10
$ echo $num_nodes

$ sh ./script --nodes 10
$ echo $num_nodes

$ . ./script --nodes 10
$ echo $num_nodes
10
```

Функции

Новая команда:

```
query() {  
    sqlite3 mydatabase.db "$1"  
}  
query "SELECT * FROM mytable"  
query "DELETE FROM mytable"
```

Сохранить результат в переменную:

```
all_names=$(query "SELECT name FROM table")  
echo "$all_names"      # вывести результат в неизменном виде
```

Перенаправление ввода/вывода

Перенаправление стандартных потоков:

```
$ ls 1>/tmp/stdout 2>/tmp/stderr 0</dev/null  
$ 1>/tmp/stdout 2>/tmp/stderr 0</dev/null ls
```

Строка как поток (шаблоны файлов):

```
name="myproject"  
cat > myfile << EOF  
project('$name', 'cpp', version: '0.1')  
EOF
```

Каналы

Количество файлов в директории:

```
$ ls | wc -l  
46
```

Контрольная сумма всех файлов во всех поддиректориях:

```
$ find . -type f | while read line; do sha1sum "$line"; done  
...  
b762797d42b044b22ffe0cbd025068ee97639650  build/spc-14.pdf  
...
```

Пример: частота повторения слов

Команды:

```
find . -type f |  
while read line; do cat "$line"; done |  
tr '[:upper:]' '[:lower:]' | # нижний регистр  
tr -d '[:punct:]' | # удаление пунктуации  
tr -s ' ' '\n' | # замена пробелов на '\n'  
sort | # сортировка по возрастанию  
uniq -c | # количество повторений  
sort -nrk1 | # сортировка первой колонки по убыванию  
head # первые 10 строк
```

Вывод:

```
3485 if  
2620 int  
2507 the  
2401 const  
2269 to  
2234 0  
1609 float  
1572 void  
1541 1  
1436 return
```

Еще примеры

Рекурсивный поиск в директориях:

```
$ grep -r 'include.*"' src
src/main.cc:#include "myguard.h"
src/main.cc:#include "myarray.h"
```

Объектно-ориентированные интерпретаторы:

```
$ sacct --helpformat
Account          AdminComment      AllocCPUS         AllocGRES
AllocNodes       AllocTRES         AssocID          AveCPU
...
$ sacct --helpformat | python3 -c "
import sys
print(','.join(sys.stdin.read().split()))
"
Account,AdminComment,AllocCPUS,AllocGRES,...
```

Пример: перемещение файлов

До 2006:

```
$ mv main2.cc main.cc
```

После 2006:

```
$ vdir
```

```
< отображается в редакторе >
```

```
42 ./Makefile
43 ./README
45 ./common.h
47 ./echoc.c
48 ./echod.c
50 ./server.c
51 ./server3.cpp
```

- ▶ Все есть строка.
- ▶ Команды — это исполняемые файлы.
- ▶ Не подходит для низкоуровневого программирования.
- ▶ Совмещает в себе язык шаблонов и анализ данных.

Файловая система

```
$ ls /  
bin      # исполняемые файлы  
boot     # ядро и загрузчик системы  
dev      # устройства  
etc      # файлы с настройками  
home     # домашние директории  
lib      # библиотеки  
proc     # процессы  
root     # домашняя директория суперпользователя  
sbin     # исполняемые файлы для суперпользователя  
sys      # взаимодействие с ядром  
tmp      # временные файлы  
usr      # неизменяемые системные файлы  
var      # изменяемые системные файлы
```

Работа с устройствами

Проигрывание видео-потока:

```
$ mplayer tv:// -tv device=/dev/video0
```

Захват видео-потока:

```
$ ffmpeg -s 640x480 -f video4linux2 -i /dev/video0 myvideo.mpeg
```

Специальные файлы для ввода/вывода в терминал:

```
$ tty  
/dev/pts/2 # название псевдо-терминала  
$ echo 'Hello world' >/dev/pts/2 # в другом терминале  
Hello world # в первом терминале
```

«Черная дыра»:

```
$ find / -type f 2>/dev/null # перенаправление потока ошибок
```

Работа с дисками

ЛР — логический раздел:

```
$ tree /dev/disk/by-id
/dev/disk/by-id
|- ata-Optiarc_DVD_RW_AD-5200A -> ../../sr0 # DVD
|- ata-ST3250318AS -> ../../sda      # физический диск
|- ata-ST3250318AS-part1 -> ../../sda1 # физический раздел 1
|- ata-ST3250318AS-part2 -> ../../sda2 # физический раздел 2
|- dm-name-vg0-home -> ../../dm-2 # ЛР с домашними директориями
|- dm-name-vg0-root -> ../../dm-0 # ЛР с корневой файловой системой
|- dm-name-vg0-swap -> ../../dm-1 # ЛР подкачки
`- usb-USB_Mass_Storage_Device_816820130806-0:0 -> ../../sdb
```

Заполнение файла нулями:

```
$ dd if=/dev/zero of=/dev/sda      # заполнить жесткий диск нулями
```

Работа с процессами

Загрузка системы:

```
$ cat /proc/loadavg
```

```
56.43 56.39 56.53 57/3855 45562
```

за 1 мин.

за 5 мин.

за 15 мин.

номер последнего
процесса

всего процессов

работающих процессов

Информация о процессоре

```
$ cat /proc/cpuinfo
```

```
...
```

```
$ lscpu
```

```
Architecture:          x86_64
```

```
CPU(s):                56
```

```
Thread(s) per core:   2
```

```
Core(s) per socket:   14
```

```
Socket(s):             2
```

```
NUMA node(s):         2
```

```
Model name:            Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz
```

```
CPU MHz:               3192.295
```

```
CPU max MHz:           3500.0000
```

```
CPU min MHz:           1200.0000
```

Монтирование файловых систем

```
$ mount
/dev/mapper/vg0-home on /home type xfs
/dev/mapper/vg0-root on / type xfs
/dev/sda1 on /boot type ext2
devtmpfs on /dev type devtmpfs
proc on /proc type proc
sysfs on /sys type sysfs
tmpfs on /dev/shm type tmpfs
tmpfs on /tmp type tmpfs
```

Файлы для компилятора

Заголовочные файлы C++:

```
$ find /usr/include/c++  
/usr/include/c++/8/bits/vector.tcc  
/usr/include/c++/8/bits/stl_vector.h  
/usr/include/c++/8/bits/stl_bvector.h  
/usr/include/c++/8/vector  
...
```

Библиотеки:

```
$ ls /usr/lib64/libstdc++.*      # библиотека C++  
/usr/lib64/libstdc++.so.6  
/usr/lib64/libstdc++.so.6.0.25  
$ ls /usr/lib64/libc.*         # библиотека C  
/usr/lib64/libc.so  
/usr/lib64/libc.so.6
```


Ссылки

- ▶ [Права доступа к файлу.](#)
- ▶ [Гранулированные права доступа.](#)
- ▶ [Права доступа процесса.](#)
- ▶ [Иерархия файловой системы.](#)
- ▶ [Файловая система процессов.](#)
- ▶ [Command line foo.](#)

© 2018–2022 Ivan Gankevich i.gankevich@spbu.ru

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License. The copy of the license is available at <https://creativecommons.org/licenses/by-sa/4.0/>.