

# Лабораторная работа №8

Поиск файлов. Перенаправление ввода-вывода. Просмотр запущенных процессов.

---

Буллер Т. А.

30 марта 2024

Российский университет дружбы народов, Москва, Россия

# Информация

---

- Буллер Татьяна Александровна
- студент группы НБИбд-01-23
- Российский университет дружбы народов

# Вводная часть

---

- инструменты поиска файлов
- инструменты фильтрации текстовых данных.
- команды по управлению процессами (и заданиями)
- команды по проверке использования диска и обслуживанию файловых систем

- Ознакомление с инструментами поиска файлов и фильтрации текстовых данных. Приобретение практических навыков: по управлению процессами (и заданиями), по проверке использования диска и обслуживанию файловых систем.

- Виртуальная машина Kali Linux
- Терминал bash
- Процессор pandoc для входного формата Markdown
- Результирующие форматы
  - pdf
  - html
- Автоматизация процесса создания: Makefile

# Выполнение лабораторной работы

---



## Перенаправление ввода-вывода.

Необходимо в файл `file.txt` названия файлов, содержащихся в каталоге `/etc`, и дописать в этот же файл названия файлов, содержащихся в домашнем каталоге. Для этого используем команду `ls` (просмотр файлов), но перенаправим ее вывод в нужный файл (символом `>` для первого перенаправления и `>>` для дописывания в существующий файл).

## Перенаправление ввода-вывода.

```
(tabuller@jordi)-[~]  
$ ls -a /etc > file.txt
```

```
(tabuller@jordi)-[~]  
$ ls -a >> file.txt
```

```
(tabuller@jordi)-[~]  
$ cat file.txt
```

```
.  
..  
adduser.conf  
adduser.conf.dpkg-save  
adduser.conf.update-old  
adjtime  
alsa  
alternatives  
apache2  
apparmor  
apparmor.d  
appstream.conf  
apt  
arp-scan  
avahi  
bash.bashrc  
bash_completion  
bash_completion.d  
bindresvport.blacklist  
binfmt.d  
bluetooth
```

Далее переходим в каталог `tmp`. Для этого используем команду `cd`, указав имя каталога, куда необходимо перейти. После того, как переход выполнен, путь до каталога указывается в командной строке.

## Фильтрация вывода.

Следующим шагом выведем имена всех файлов из `file.txt`, имеющих расширение `.conf`, после чего запишем их в новый текстовый файл `conf.txt`. Для фильтрации вывода используем команду `grep` с регулярным выражением: для того, чтобы точка отображалась именно как символ точки, а не один любой символ, дважды экранируем ее обратным слэшем (`\`), а чтобы после расширения файла не стояло ни единого символа (например, чтобы вместо `.conf` не выводилось `.config`), укажем символ `$`.

## Фильтрация вывода.

```
(tabuller@jordi)-[~]  
$ cat file.txt | grep \\.conf$ >> conf.txt  
|
```

Рис. 2: Фильтрация и перенаправление вывода

Проверим вывод файла, используя то же регулярное выражение для `grep`: это поможет подсветить нужные элементы в выводе. Как видим, запись прошла без ошибок.

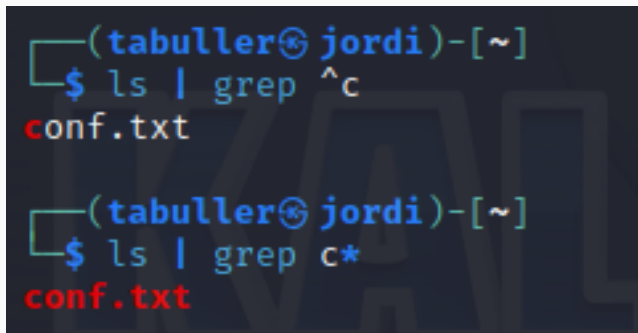
## Фильтрация вывода.

```
(tabuller@jordi)-[~]  
$ cat conf.txt | grep \\.conf  
adduser.conf  
appstream.conf  
ca-certificates.conf  
debconf.conf  
deluser.conf  
dns2tcpd.conf  
e2scrub.conf  
fuse.conf  
gai.conf  
hdparm.conf  
host.conf  
idmapd.conf  
ipsec.conf  
kernel-img.conf  
ld.so.conf  
libao.conf  
libaudit.conf  
locale.conf  
logrotate.conf  
miredo.conf  
mke2fs.conf  
nfs.conf  
nftables.conf  
nikto.conf  
nsswitch.conf  
pam.conf
```

Рис. 3: Проверка записи с фильтром

Далее по заданию нужно предложить несколько вариантов, как определить, какие файлы в домашнем каталоге имеют имена, начинавшиеся с символа с. Для этого используем тот же grep, на этот раз с други выражение. Самые простые варианты - указать только на начало строки (^ - начало, ^с - указание на то, что в начале обязательно стоит символ с) либо на то, что после первого символа с стоит любой другой (с\*).



A terminal window with a dark background. The prompt is (tabuller@jordi)-[~]. The first command is \$ ls | grep ^c, which results in the output conf.txt. The second command is \$ ls | grep c\*, which also results in the output conf.txt.

```
(tabuller@jordi)-[~]  
$ ls | grep ^c  
conf.txt  
  
(tabuller@jordi)-[~]  
$ ls | grep c*  
conf.txt
```

Рис. 4: Поиск файлов по началу названия

То же самое сделаем для всех файлов, начинающихся с `h` в каталоге `/etc` и проверим вывод постранично, используя команду `more` (файлов и так немного, поэтому постраничный вывод от обычного не отличается):

```
(tabuller@jordi)-[~]  
$ ls /etc | grep ^h>>file1  
  
(tabuller@jordi)-[~]  
$ more file1  
hdparm.conf  
host.conf  
hostname  
hosts  
hosts.allow  
hosts.deny
```

Рис. 5: Поиск файлов по началу названия и постраничный вывод

Задание: запустить в фоновом режиме процесс, который будет записывать в файл ~/logfile файлы, имена которых начинаются с log. Для этого используем команду `find`. В параметре `-name` укажем “log\*” и попросим распечатать в logfile. Для того, чтобы перевести процесс в фоновый режим, укажем символ амперсанта - как только процесс выполнится, терминал об этом сообщит, а до этого можно выполнять другие задачи.

## Перевод процессов в фон и терминирование задач

```
(tabuller@jordi)-[~]  
$ find -name "log*" -print > logfile &  
[1] 30562  
  
(tabuller@jordi)-[~]  
$  
[1] + done find -name "log*" -print > logfile y
```

Рис. 6: Перевод процесса в фон

Проверим, что нашла команда, используя `grep` для подсветки начала названия (`log`). Можно видеть, что все файлы, начинающиеся с сочетания букв `log`, были записаны в `logfile`.

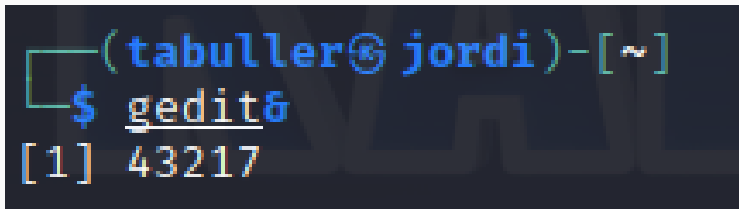
# Перевод процессов в фон и терминирование задач

```
taballier@jordi: ~$  
$ cat logfile | grep log  
./work/study/2023-2024/Операционные системы/os-intro/.git/modules/template/report/LOG  
./work/study/2023-2024/Операционные системы/os-intro/.git/modules/template/presentation  
/LOG  
./work/study/2023-2024/Операционные системы/os-intro/.git/LOG  
./local/share/pnpm/global/5/.ppm/ora5.4.1/node_modules/log-symbols  
./local/share/pnpm/global/5/.ppm/log-symbols@1.0  
./local/share/pnpm/global/5/.ppm/log-symbols@1.0/node_modules/log-symbols  
./local/share/pnpm/global/5/.ppm/handlebars@4.7.8/node_modules/handlebars/dist/cjs/handlebars/compilers.js  
./local/share/pnpm/global/5/.ppm/handlebars@4.7.8/node_modules/handlebars/dist/cjs/handlebars/helpers/log.js  
./local/share/pnpm/global/5/.ppm/handlebars@4.7.8/node_modules/handlebars/dist/amd/handlebars/compilers.js  
./local/share/pnpm/global/5/.ppm/handlebars@4.7.8/node_modules/handlebars/dist/amd/handlebars/helpers/log.js  
./local/share/pnpm/global/5/.ppm/handlebars@4.7.8/node_modules/handlebars/lib/handlebars/compilers.js  
./local/share/pnpm/global/5/.ppm/handlebars@4.7.8/node_modules/handlebars/lib/handlebars/helpers/log.js  
./local/share/pnpm/global/5/.ppm/neo-async@2.6.2/node_modules/neo-async/log.js
```

Рис. 7: Проверка результата записи

Далее переведем в фон процесс `gedit`. Используем символ амперсанта и видим, как процесс (в квадратных скобках указан номер задачи, за ним - `pid` процесса) запущен в фоновом режиме.





```
(tabuller@jordi)-[~]  
$ gedit&  
[1] 43217
```

Рис. 8: Запуск процесса в фоне

Определим идентификатор процесса `gedit`, используя команду `ps`, конвейер и фильтр `grep`. Получаем две строки: сам `gedit` (который на самом деле не установлен, поэтому в выводе видим процесс сообщения об отсутствующей команде) и `grep`, который ищет этот `gedit`. Нужный нам процесс - первый.

# Перевод процессов в фон и терминирование задач

```
(tabuller@jordi)-[~]  
$ ps aux | grep gedit  
tabuller  43219  0.0  0.4 32204 19840 pts/0    TN   03:31   0:00 /usr/bin/python3 /usr/  
lib/command-not-found -- gedit  
tabuller  43509  0.0  0.0  6344  2176 pts/0    S+   03:31   0:00 grep --color=auto ge  
dit
```

Рис. 9: Определение идентификатора процесса

Теперь необходимо от этого процесса избавиться. Сделать это можно двумя способами: по номеру задачи либо по идентификатору процесса. Для того, чтобы убить процесс по идентификатору, используем опцию -9 с самим идентификатором процесса. Для убийства по задаче используем символ % с номером задачи.

# Перевод процессов в фон и терминирование задач

```
(tabuller@jordi)-[~]
$ gedit
[1] 43862

(tabuller@jordi)-[~]
$ Command 'gedit' not found, but can be installed with:
sudo apt install gedit
Do you want to install it? (N/y)
[1] + suspended (tty input) gedit
(tabuller@jordi)-[~]
$ kill -9 43862

[1] + killed gedit
(tabuller@jordi)-[~]
$ gedit
[1] 44070

(tabuller@jordi)-[~]
$ Command 'gedit' not found, but can be installed with:
sudo apt install gedit
Do you want to install it? (N/y)
[1] + suspended (tty input) gedit , the more you at
(tabuller@jordi)-[~]
$ kill %1

[1] + terminated gedit
```

Рис. 10: Завершение процесса двумя способами

Для работы с диском рассмотрим две команды `du` и `df`. Первая говорит нам о том, что и как диск занимает непосредственно. Вторая - о том, насколько диск заполнен.

```
DF(1) fileutils utilities \(df\) User Commands DF\(1\)  
  
NAME  
    df - report file system space usage  
  
SYNOPSIS  
    df \[OPTION\] ... \[FILE\] ...  
  
DESCRIPTION  
    This manual page documents the GNU version of df. df displays the amount of  
    space available on the file system containing each file name argument. If no
```

Рис. 11: Руководство df

# Проверка использования диска

```
du(1) [FILE:000 width=700] User Commands du(1)
NAME
  du - estimate file space usage
SYNOPSIS
  du [OPTION]... [FILE]...
  du [OPTION]... --files0-from=F
DESCRIPTION
  Summarize device usage of the set of FILES, recursively for directories.
  Mandatory arguments to long options are mandatory for short options too.
```

Рис. 12: Руководство du



Попробуем исполнить эти команды и видим, соответственно, сколько места использовано и осталось на дисках (df) и то, сколько какой файл занимает (du).

# Проверка использования диска

```
(tabuller@jordi)-[~]  
$ df  
Filesystem      1K-blocks    Used Available Use% Mounted on  
udev            1945568         0   1945568    0% /dev  
tmpfs           397188      1364   395824    1% /run  
/dev/sda1       45142924 36614720  6202600   86% /  
tmpfs           1985924         0   1985924    0% /dev/shm  
tmpfs            5120         0      5120    0% /run/lock  
tmpfs           397184      120   397064    1% /run/user/1002
```

Рис. 13: Вывод df

```
1988  ./mozilla/firefox/qsre64kg.default
4     ./mozilla/firefox/qsre64kg.default
8     ./mozilla/firefox/qsre64kg.default
2936  ./mozilla/firefox/qsre64kg.default
4     ./mozilla/firefox/qsre64kg.default
12    ./mozilla/firefox/qsre64kg.default
408   ./mozilla/firefox/qsre64kg.default
31040 ./mozilla/firefox/qsre64kg.default
31076 ./mozilla/firefox
31084 ./mozilla
36    ./bashrc.d
8     ./gnupg/openpgp-revocs.d
12    ./gnupg/private-keys-v1.d
40    ./gnupg
4     ./Templates
4     ./gvfs
8     ./java/.userPrefs/burpome, the
12    ./java/.userPrefs
16    ./java
1139724 .
```

Рис. 14: Вывод du

Последним шагом с помощью команды `find` выведем имена всех файлов, имеющих в домашнем каталоге. Для этого укажем несколько параметров: во-первых то, что ищем мы в домашнем каталоге и только в нем (глубина поиска 0), во-вторых - то, что ищем мы директории.

```
(tabuller@jordi)-[~]  
$ find ~/* -maxdepth 0 -type d  
/home/buller/Desktop  
/home/buller/Documents  
/home/buller/Downloads  
/home/buller/Music  
/home/buller/Pictures  
/home/buller/Public  
/home/buller/Templates  
/home/buller/Videos  
/home/buller/work
```

Рис. 15: Просмотр директорий в домашнем каталоге

## Выводы

---

Приобретены практические навыки по работе с инструментами поиска файлов и фильтрации текстовых данных, по управлению процессами (и заданиями), по проверке использования диска и обслуживанию файловых систем.