

# Лабораторная работа №14

Операционные системы

---

Панявкина И.В.

16 мая 2025

Российский университет дружбы народов, Москва, Россия

Цель данной лабораторной работы - изучить основы программирования в оболочке ОС UNIX, научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

1. Написать командный файл, реализующий упрощённый механизм семафоров.

Командный файл должен в течение некоторого времени  $t_1$  дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени  $t_2 < t_1$ , также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (`> /dev/tty#`, где `#` — номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не фоновом, а в привилегированном режиме. Доработать программу так, чтобы имелась возможность взаимодействия трёх и более процессов.

2. Реализовать команду `man` с помощью командного файла. Изучите содержимое каталога `/usr/share/man/man1`. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой `less` сразу же просмотрев содержимое справки. Командный файл должен получать в виде аргумента командной строки название команды и в виде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге `man1`.

- Используя встроенную переменную `$RANDOM`, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтите, что `$RANDOM` выдаёт псевдослучайные числа в диапазоне от 0 до 32767.

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций; C-оболочка (или csh) — надстройка на оболочке Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке `bash`. В других оболочках большинство команд будет совпадать с описанными ниже.



Создаю командный файл для первой программы, пишу ее, проверяю ее работу (рис. 1).

```
[irinapanyavkina@irinapanyavkina ~]$ touch 121.sh
[irinapanyavkina@irinapanyavkina ~]$ chmod +x 121.sh
[irinapanyavkina@irinapanyavkina ~]$ bash 121.sh
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
```

Рис. 1: Создание и исполнение файла

Командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени  $t_1$  дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени  $t_2 < t_1$ , также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (`> /dev/tty#`, где `#` — номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не фоновом, а в привилегированном режиме. Доработать программу так, чтобы имелась возможность взаимодействия трёх и более процессов (рис. 2).

```
#!/bin/bash

lockfile="./lock.file"
exec {fn}>$lockfile

while test -f "$lockfile"
do
    if flock -n ${fn}
    then
        echo "File is blocked"
        sleep 5
        echo "File is unlocked"
        flock -u ${fn}
    else
        echo "File is blocked"
        sleep 5
    fi
done
```

Чтобы реализовать команду `man` с помощью командного файла, изучаю содержимое каталога `/usr/share/man/man1`. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой `less` сразу же просмотрев содержимое справки (рис. 3).

## Выполнение лабораторной работы

```
exempi.1.gz      gnuplot.1.gz      mtz-intel.1.gz
exit.1.gz         gold-tool.1.gz     mtz-metapo
exiv2.1.gz        gp-archive.1.gz    mtz-module
exo-open.1.gz     gpasswd.1.gz       mtz-packag
expand.1.gz       gp-collect-app.1.gz mtz-patter
export.1.gz       gp-display-html.1.gz mtz-pdf.1.
expr.1.gz         gp-display-src.1.gz mtz-plain.
extconv.1.gz      gp-display-text.1.gz mtz-profil
extractbb.1.gz    gpg.1.gz           mtz-rsync.
extractres.1.gz   gpg2.1.gz          mtxrunc.1.g
factor.1.gz       gpg-agent.1.gz     mtz-scite.
fadvice.1.gz      gpg-card.1.gz      mtz-server
fallocate.1.gz    gpg-check-pattern.1.gz mtz-spell.
false.1.gz        gpgconf.1.gz       mtz-textwo
fc.1.gz           gpg-connect-agent.1.gz mtz-timing
fc-cache.1.gz     gpgparsemail.1.gz  mtz-tools.
fc-cache-64.1.gz  gpg-preset-passphrase.1.gz mtz-unicod
fc-cat.1.gz       gpgsm.1.gz         mtz-unzip.
```

[irinapanyavkina@irinapanyavkina ~]\$ ls /usr/share/man/man1

Рис. 3: Изучение содержимого папки

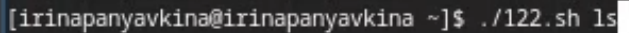
Командный файл должен получать в виде аргумента командной строки название команды и в виде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге `man1` (рис. 4).

```
#!/bin/bash

a=$1
if test -f "/usr/share/man/man1/$a.1.gz"
then less /usr/share/man/man1/$a.1.gz
else
echo "There is no such command"
fi
|
```

Рис. 4: Код программы

Проверяю работу командного файла (рис. 5).

A terminal window with a dark background. The prompt is [irinapanyavkina@irinapanyavkina ~]\$. The command being entered is ./122.sh ls, followed by a white cursor block.

```
[irinapanyavkina@irinapanyavkina ~]$ ./122.sh ls
```

Рис. 5: Исполнение программы



Командный файл работает так же, как и команда `man`, открывает справку по указанной утилите (рис. 6).

# Выполнение лабораторной работы

```
ESC[4mLESC[24m(1)
  ESC[4mLESC[24m(1)

ESC[1mNAMEESC[0m
  ls - list directory contents

ESC[1mSYNOPSISESC[0m
  ESC[1mls ESC[22m[ESC[4mOPTIONESC[24m]... [ESC[4mFILEESC[24m]...

ESC[1mDESCRIPTIONESC[0m
  List information about the FILES (the current directory by default). Sort entries alphabetically if none of ESC[

  Mandatory arguments to long options are mandatory for short options too.

  ESC[1m-aESC[22m, ESC[1m--allESC[0m
    do not ignore entries starting with .

  ESC[1m-AESC[22m, ESC[1m--almost-allESC[0m
    do not list implied . and ..

  ESC[1m--authorESC[0m
    with ESC[1m-lESC[22m, print the author of each file
```

Рис. 6: Результат работы программы

Создаю файл для кода третьей программы, пишу программу и проверяю ее работу (рис. 7).

```
[irinapanyavkina@irinapanyavkina ~]$ touch 123.sh
[irinapanyavkina@irinapanyavkina ~]$ chmod +x 123.sh
[irinapanyavkina@irinapanyavkina ~]$ bash 123.sh 20
lnvtvcjxvztmgdomabrf
[irinapanyavkina@irinapanyavkina ~]$
```

Рис. 7: Создание и исполнение файла

Используя встроенную переменную \$RANDOM, пишу командный файл, генерирующий случайную последовательность букв латинского алфавита. Т.к. \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767, ввожу ограничения так, чтобы была генерация чисел от 1 до 26 (рис. 8).

```
#!/bin/bash

a=$1

for ((i=0; i<$a; i++))
do
    ((char=$RANDOM%26+1))
    case $char in
        1) echo -n a;; 2) echo -n b;; 3) echo -n c;; 4) echo -n d;; 5) echo -n e;; 6) echo -n f;;
        7) echo -n g;; 8) echo -n h;; 9) echo -n i;; 10) echo -n j;; 11) echo -n k;; 12) echo -n l;;
        13) echo -n m;; 14) echo -n n;; 15) echo -n o;; 16) echo -n p;; 17) echo -n r;; 18) echo -n s;;
        19) echo -n t;; 20) echo -n q;; 21) echo -n u;; 22) echo -n v;;
        23) echo -n w;; 24) echo -n x;; 25) echo -n y;; 26) echo -n z;;
    esac
done
echo
```

Рис. 8: Код программы

## Выводы

---

При выполнении данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX, научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## Список литературы

---



1. Лабораторная работа №14 [Электронный ресурс] URL:  
<https://esystem.rudn.ru/mod/resource/view.php?id=1224395>