

Лабораторная работа №13. Программирование в командном процессоре ОС UNIX. Расширенное программирование.

Гекишева Анастасия Дмитриевна, НБИ-01-20,
30 апреля, 2021

¹RUDN University, Moscow, Russian Federation

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.
2. Реализовать команду `man` с помощью командного файла.
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`.
3. Используя встроенную переменную `$RANDOM`, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита.

Выполнение 1-го пункта задания

Командный файл, реализующий упрощённый механизм семафоров

```
1 #! /bin/bash
2
3 lockfile="lockfile"
4
5 exec {fn}>$lockfile
6
7 until flock -n ${fn}
8 do
9     echo "не удалось заблокировать"
10    sleep 1
11 done
12
13 for (( i=0;i<=10;i++ ))
14 do
15     echo "файл используется"
16     sleep 1
17 done
18
19 flock -u ${fn}
```

Рис. 1: Командный файл, реализующий упрощённый механизм семафоров

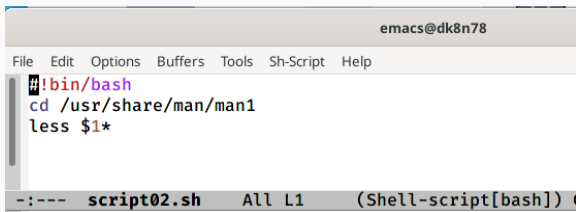
Результат выполнения командного файла

```
для более детальной информации смотрите /usr/bin/lsusb(1).
adkeki sheva@dk8n78 ~ $ ./script01.sh
файл используется
файл используется
файл используется
файл используется
файл используется
файл используется
файл используется
файл используется
файл используется
файл используется
adkeki sheva@dk8n78 ~ $ ./script01.sh
не удалось заблокировать
не удалось заблокировать
не удалось заблокировать
не удалось заблокировать
не удалось заблокировать
не удалось заблокировать
не удалось заблокировать
не удалось заблокировать
не удалось заблокировать
не удалось заблокировать
^C
adkeki sheva@dk8n78 ~ $
```

Рис. 2: Результат выполнения командного файла

Выполнение 2-го пункта задания

Командный файл реализующий команду man



The image shows a screenshot of an Emacs editor window. The title bar at the top reads "emacs@dk8n78". Below the title bar is a menu bar with the following items: "File", "Edit", "Options", "Buffers", "Tools", "Sh-Script", and "Help". The main editing area contains a shell script with the following lines:

```
#!/bin/bash
cd /usr/share/man/man1
less $1*
```

At the bottom of the window, there is a status bar that displays: "-:--- script02.sh All L1 (Shell-script[bash])".

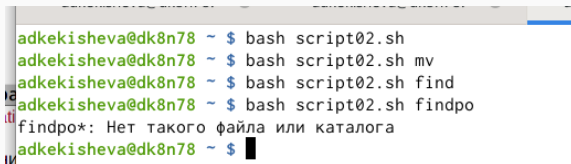
Рис. 3: Командный файл реализующий команду man

Содержимое каталога /usr/share/man/man1

```
adkekisheva@dk8n78 /usr $ cd share/man/man1
adkekisheva@dk8n78 /usr/share/man/man1 $ ls
411toppm.1.bz2
7z.1.bz2
7za.1.bz2
7zr.1.bz2
a2ps.1.bz2
a2x.1.bz2
a52dec.1.bz2
aacplusenc.1.bz2
ab.1.bz2
aclocal-1.11.1.bz2
aclocal-1.12.1.bz2
aclocal-1.13.1.bz2
aclocal-1.14.1.bz2
aclocal-1.15.1.bz2
aclocal-1.16.1.bz2
aconnect.1.bz2
acyclic.1.bz2
adddebug.1.bz2
addedge.1.bz2
addftinfo.1.bz2
addrinfo.1.bz2
advdef.1.bz2
advnmng.1.bz2
nvme-lnvm-diag-bbtbl.1.bz2
nvme-lnvm-diag-set-bbtbl.1.bz2
nvme-lnvm-factory.1.bz2
nvme-lnvm-id-ns.1.bz2
nvme-lnvm-info.1.bz2
nvme-lnvm-init.1.bz2
nvme-lnvm-list.1.bz2
nvme-lnvm-remove.1.bz2
nvme-netapp-ontapdevices.1.bz2
nvme-netapp-smdevices.1.bz2
nvme-ns-descs.1.bz2
nvme-ns-rescan.1.bz2
nvme-read.1.bz2
nvme-reset.1.bz2
nvme-resv-acquire.1.bz2
nvme-resv-register.1.bz2
nvme-resv-release.1.bz2
nvme-resv-report.1.bz2
nvme-sanitize.1.bz2
nvme-sanitize-log.1.bz2
nvme-security-recv.1.bz2
nvme-security-send.1.bz2
nvme-self-test-log.1.bz2
```

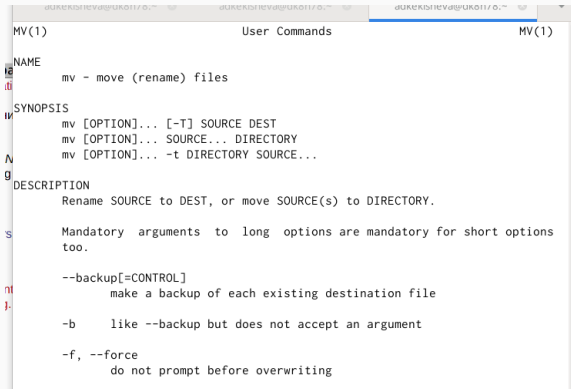
Рис. 4: Содержимое каталога /usr/share/man/man1

Вызов командного файла программы



```
adkekisheva@dk8n78 ~ $ bash script02.sh
adkekisheva@dk8n78 ~ $ bash script02.sh mv
adkekisheva@dk8n78 ~ $ bash script02.sh find
adkekisheva@dk8n78 ~ $ bash script02.sh findpo
findpo*: Нет такого файла или каталога
adkekisheva@dk8n78 ~ $
```

Рис. 5: Вызов командного файла программы



```
MV(1)                                User Commands                                MV(1)

NAME
    mv - move (rename) files

SYNOPSIS
    mv [OPTION]... [-T] SOURCE DEST
    mv [OPTION]... SOURCE... DIRECTORY
    mv [OPTION]... -t DIRECTORY SOURCE...

DESCRIPTION
    Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

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

    --backup[=CONTROL]
        make a backup of each existing destination file

    -b
        like --backup but does not accept an argument

    -f, --force
        do not prompt before overwriting
```

Рис. 6: Результат работы программы: справка программы mv

Выполнение 3-го пункта задания

Командный файл, генерирующий случайную последовательность букв латинского алфавита

```
1 #!/bin/bash
2 echo "Введите количество комбинаций: "
3 read number
4 M=number
5 echo "Введите количество букв для генерирования комбинаций"
6 read num
7 c=1
8 d=1
9 echo
10 echo "Рандомные комбинации букв:"
11 while (($c!=($M+1)))
12 do
13     echo $d
14     echo $(for((i=1;i<=$num;i++));
15 do printf '%s' "${RANDOM:0:1}"; done) | tr '[:0-9:]' '[:a-z:]'
16     ((c+=1))
17     ((d+=1))
18 done
19
```

Рис. 7: Командный файл, генерирующий случайную последовательность букв латинского алфавита

Результат работы программы

```
adkekisheva@dk8n78 ~ $ bash script03.sh
Введите количество комбинаций:
4
Введите количество букв для генерирования комбинаций
6

Рандомные комбинации букв:
1
bfbbbg
2
bchbid
3
cbgcid
4
bbbbdc
adkekisheva@dk8n78 ~ $ bash script03.sh
Введите количество комбинаций:
3
Введите количество букв для генерирования комбинаций
2

Рандомные комбинации букв:
1
fb
2
cb
3
cg
```

Рис. 8: Результат работы программы

Я продолжила изучение основ программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

1. Ссылка 1
2. Ссылка 2