

Отчет по лабораторной работе №14

**Программирование в командном процессоре ОС UNIX. Расширенное
программирование**

Разанацуа Сара Естэлл

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Выводы	13

Список иллюстраций

3.1	Создание и исполнение файла	8
3.2	Код программы	9
3.3	Изучение содержимого папки	10
3.4	Код программы	10
3.5	Результат работы программы	11
3.6	Создание и исполнение файла	11
3.7	Код программы	12

Список таблиц

1 Цель работы

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

2 Задание

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`.
3. 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.

/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 имела возможность взаимодействия трёх и более процессов (рис. fig. 3.2).



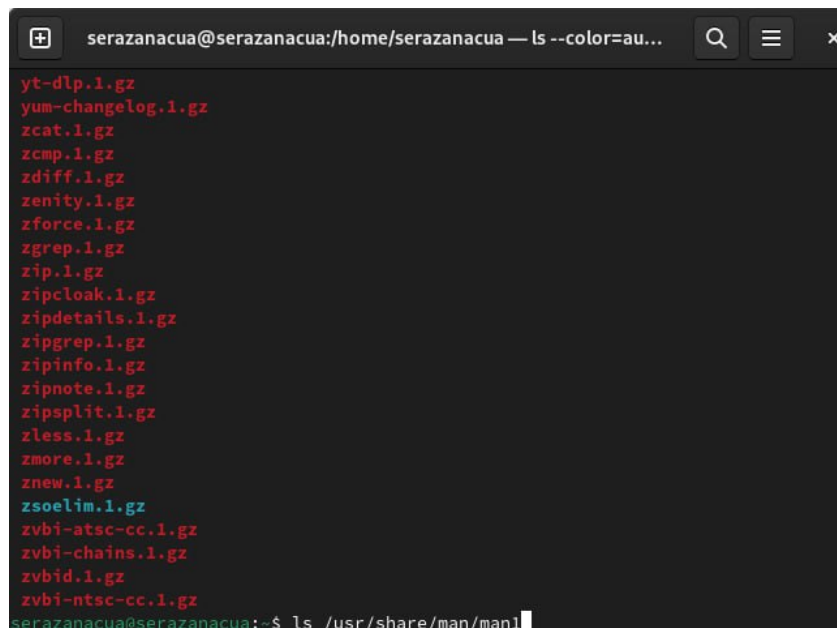
```
Открыть ▾ + • 121.sh ~/
#!/bin/bash

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

while test -f "$lockfile"
do
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
```

Рис. 3.2: Код программы

- 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. fig. 3.3).



```
serazanacua@serazanacua:/home/serazanacua — ls --color=au...
yt-dlp.1.gz
yum-changelog.1.gz
zcat.1.gz
zcmp.1.gz
zdiff.1.gz
zenity.1.gz
zforce.1.gz
zgrep.1.gz
zip.1.gz
zipcloak.1.gz
zipdetails.1.gz
zipgrep.1.gz
zipinfo.1.gz
zipnote.1.gz
zipsplit.1.gz
zless.1.gz
zmore.1.gz
znew.1.gz
zsoelim.1.gz
zvbi-atsc-cc.1.gz
zvbi-chains.1.gz
zvbid.1.gz
zvbi-ntsc-cc.1.gz
serazanacua@serazanacua:~$ ls /usr/share/man/man1
```

Рис. 3.3: 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. fig. 3.4).



```
122.sh
#!/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. 3.4: Код программы

- Командный файл работает так же, как и команда man, открывает справку по указанной утилите (рис. fig. 3.5).

```
serazanacua@serazanacua:/home/serazanacua — /bin/bash ./1...
ESC[4mLSESC[24m(1) User Commands
ESC[4mLSESC[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[1m-cftuvSUX ESC[22mnor ESC[1m-
-sort ESC[22mis speci-
fied.

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 ..

.
```

Рис. 3.5: Результат работы программы

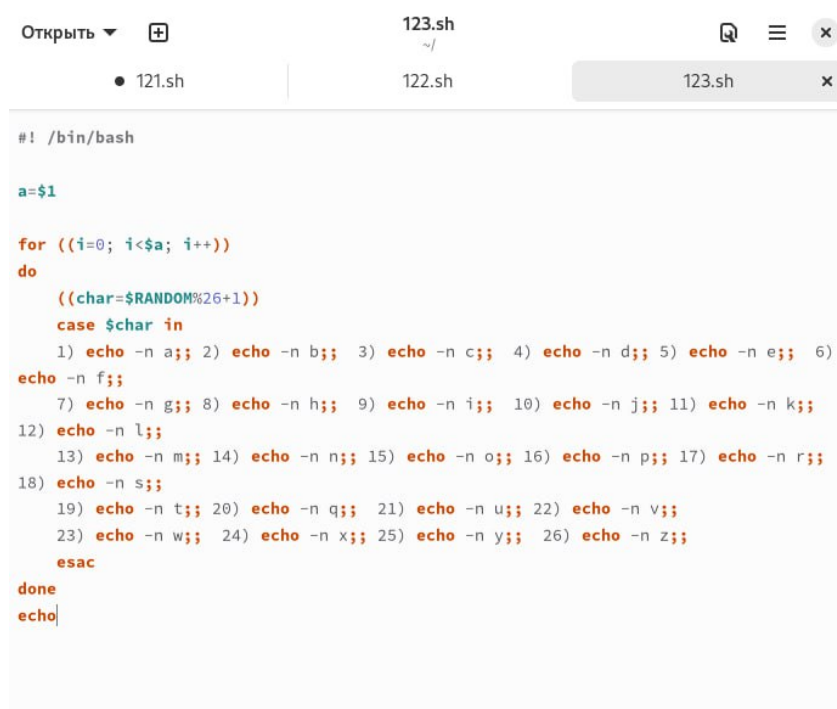
- Создаю файл для кода третьей программы, пишу программу и проверяю ее работу (рис. fig. 3.6).

```
serazanacua@serazanacua:/home/serazanacua — /bin/bash ./1...
zgrep.1.gz
zip.1.gz
zipcloak.1.gz
zipdetails.1.gz
zipgrep.1.gz
zipinfo.1.gz
zipnote.1.gz
zipsplit.1.gz
zless.1.gz
zmore.1.gz
znew.1.gz
zsoelim.1.gz
zvbi-atsc-cc.1.gz
zvbi-chains.1.gz
zvid.1.gz
zvbi-ntsc-cc.1.gz
serazanacua@serazanacua:~$ ./122.sh ls
serazanacua@serazanacua:~$ touch 123.sh
serazanacua@serazanacua:~$ chmod +x 123.sh
serazanacua@serazanacua:~$ bash 123.sh
```

Рис. 3.6: 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. fig. 3.7).



```
Открыть ▾ + 123.sh ~/ 121.sh 122.sh 123.sh x
#!/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. 3.7: Код программы

4 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.