

Отчет по лабораторной работе №14

Дисциплина: Операционные системы

Иванов Сергей Владимирович

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	7
4	Вывод	13

Список иллюстраций

3.1	Создание и исполнение файла	7
3.2	Код программы	8
3.3	Изучение содержимого папки	9
3.4	Код программы	9
3.5	Исполнение программы	10
3.6	Результат работы программы	10
3.7	Создание и исполнение файла	11
3.8	Код программы	11

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.

3 Выполнение лабораторной работы

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

```
[svivanov1@svivanov1 ~]$ touch 121.sh
[svivanov1@svivanov1 ~]$ chmod +x 121.sh
[svivanov1@svivanov1 ~]$ bash 121.sh
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
```

Рис. 3.1: 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).

```
1 #!/bin/bash
2
3 lockfile="./lock.file"
4 exec {fn}>$lockfile
5
6 while test -f "$lockfile"
7 do
8   if flock -n ${fn}
9   then
10     echo "File is blocked"
11     sleep 5
12     echo "File is unlocked"
13     flock -u ${fn}
14   else
15     echo "File is blocked"
16     sleep 5
17   fi
18 done
```

Рис. 3.2: Код программы

```
#!/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).

```
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
[svivanov1@svivanov1 ~]$ 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. 4).

```
1 #!/bin/bash
2
3 a=$1
4 if test -f "/usr/share/man/man1/$a.1.gz"
5 then less /usr/share/man/man1/$a.1.gz
6 else
7 echo "There is no such command"
8 fi
```

Рис. 3.4: Код программы

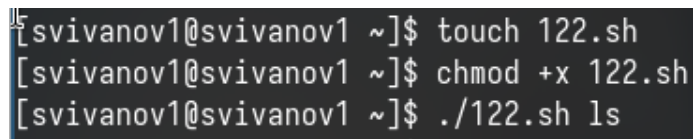
```
#!/bin/bash
```

```

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

```

Проверяю работу командного файла (рис. 5).



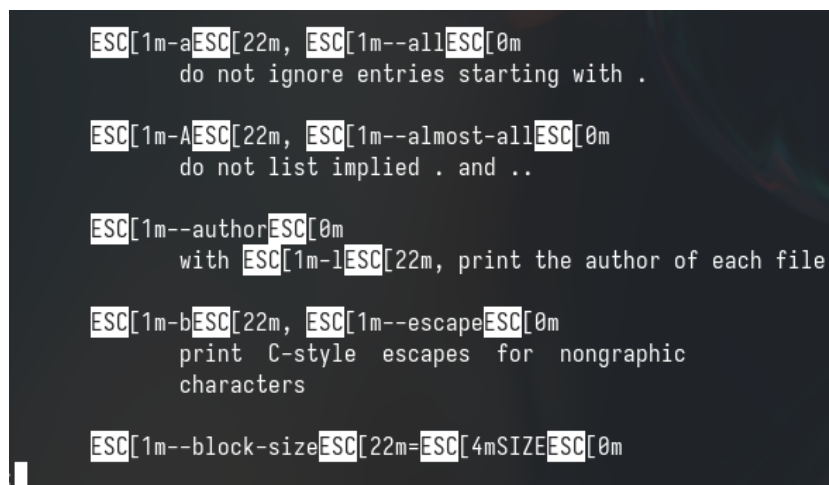
```

[svivanov1@svivanov1 ~]$ touch 122.sh
[svivanov1@svivanov1 ~]$ chmod +x 122.sh
[svivanov1@svivanov1 ~]$ ./122.sh ls

```

Рис. 3.5: Исполнение программы

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



```

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

ESC[1m-bESC[22m, ESC[1m--escapeESC[0m
print C-style escapes for nongraphic
characters

ESC[1m--block-sizeESC[22m=ESC[4mSIZEESC[0m

```

Рис. 3.6: Результат работы программы

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

```
[svivanov1@svivanov1 ~]$ touch 123.sh
[svivanov1@svivanov1 ~]$ chmod +x 123.sh
[svivanov1@svivanov1 ~]$ bash 123.sh 20
xlpsduafkftfpqxlcczdo
[svivanov1@svivanov1 ~]$
```

Рис. 3.7: 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).

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

Рис. 3.8: Код программы

```
#!/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;;
```

```
        7) echo -n g;; 8) echo -n h;; 9) echo -n i;; 10) echo -n j;; 11) echo -n k;
```

```
        13) echo -n m;; 14) echo -n n;; 15) echo -n o;; 16) echo -n p;; 17) echo -n r
```

```
        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
```

4 Вывод

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