

# Выполнение 14 лабораторной работы

Программирование в командном процессоре ОС UNIX. Расширенное программирование

---

Павлюченков С.В.

07 сентября 2024

Российский университет дружбы народов, Москва, Россия

- Павлюченков Сергей Витальевич
- Студент ФФМиЕН
- Российский университет дружбы народов
- 1132237372@pfur.ru
- <https://serapshi.github.io/svpavliuchenkov.github.io/>



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`.
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.

## Выполнение лабораторной работы

---

## Код программы семафора

```
1 mc [svpavliuchenkov@fedora]:~/work/os/1
mc [svpavliuchenkov@fedora]:~/work/os/lab14
1_lab14 [B---] 0 L:[ 1+19 20/ 20] *(215 / 215b)
#!/bin/bash
t1=$1
t2=$2
myfile='lockedfile'

while test -f $myfile
do
    echo "Proccess is locked"
    sleep $t1
done

echo "Process $$ is running"
touch $myfile
sleep $t2

echo "Proccess $$ is done"
rm $myfile
```

```
1 mc [svpavliuchenkov@fedora]:~/work/os/]  
mc [svpavliuchenkov@fedora]:~/work/os/lab14  
1a_lab14 [B---] 4 L:[ 1+11 12/ 12] *(135 / 135b)  
#!/bin/bash  
  
tmp=3  
t1=$1  
t2=$2  
  
for (( a=0; a<tmp; a++))  
do  
echo "process $a is running"  
sudo ./1_lab14 $t1 $t2 > /dev/tty$a  
done  
wait
```



```
[svpavliuchenkov@fedora lab14]$ sudo ./1a_lab14 3 5  
process 0 is running  
process 1 is running  
process 2 is running  
[svpavliuchenkov@fedora lab14]$
```

Рис. 3: Запуск программы

## Структура папки man1.

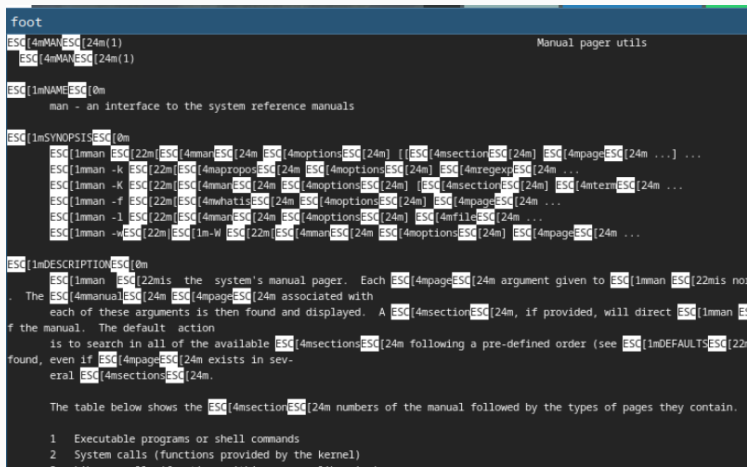
```
[svpavliuchenkov@fedora lab14]$ ls /usr/share/man/man1  
.:1.gz  
[.1.gz'  
a2ping.1.gz  
abrt.1.gz  
abrt-action-analyze-backtrace.1.gz  
abrt-action-analyze-c.1.gz  
abrt-action-analyze-ccpp-local.1.gz  
abrt-action-analyze-oops.1.gz  
abrt-action-analyze-python.1.gz  
abrt-action-analyze-vmcore.1.gz  
abrt-action-analyze-vulnerability.1.gz  
abrt-action-analyze-xorg.1.gz  
abrt-action-check-oops-for-hw-error.1.gz  
abrt-action-find-bodhi-update.1.gz  
abrt-action-generate-backtrace.1.gz  
abrt-action-generate-core-backtrace.1.gz  
abrt-action-list-dsos.1.gz  
abrt-action-notify.1.gz  
abrt-action-save-package-data.1.gz  
abrt-action-trim-files.1.gz  
abrt-applet.1.gz  
abrt-auto-reporting.1.gz  
abrt-bodhi.1.gz  
abrt-cli.1.gz  
abrt-dump-journal-core.1.gz  
abrt-dump-journal-oops.1.gz  
abrt-dump-journal-xorg.1.gz  
abrt-dump-oops.1.gz  
msgmerge.1.gz  
msgunfmt.1.gz  
msguniq.1.gz  
mshortname.1.gz  
mshowfat.1.gz  
msxlint.1.gz  
mtools.1.gz  
mtoolstest.1.gz  
mtrace.1.gz  
mtx-babel.1.gz  
mtx-base.1.gz  
mtx-bibtex.1.gz  
mtx-cache.1.gz  
mtx-chars.1.gz  
mtx-check.1.gz  
mtx-colors.1.gz  
mtx-context.1.gz  
mtx-dvi.1.gz  
mtx-epub.1.gz  
mtx-evohome.1.gz  
mtx-fcd.1.gz  
mtx-flac.1.gz  
mtx-fonts.1.gz  
mtx-grep.1.gz  
mtx-interface.1.gz  
mtx-metapost.1.gz  
mtx-modules.1.gz  
mtx-package.1.gz
```

```
[svpavliuchenkov@fedora lab14]$ ./2_lab14 man
/usr/share/man/man1man
[svpavliuchenkov@fedora lab14]$ cat 2_lab14
#!/bin/bash

file=$1
mydir="/usr/share/man/man1"
echo "$mydir$file"
if test -f "$mydir/$file.1.gz"
then
less "$mydir/$file.1.gz"
else
echo "no such file"
fi[svpavliuchenkov@fedora lab14]$
```

## Запуск программы

Командой `less` сразу же открывается просмотр содержимого справки о команде.



```
foot
ESC[4mMANESC[24m(1) Manual pager utils
ESC[4mMANESC[24m(1)
ESC[1mNAMEESC[0m
man - an interface to the system reference manuals
ESC[1mSYNOPSISESC[0m
ESC[1mman ESC[22mESC[4mmanESC[24m ESC[4moptionsESC[24m [[ESC[4msectionESC[24m] ESC[4mpageESC[24m ...] ...
ESC[1mman -k ESC[22mESC[4maproposESC[24m ESC[4moptionsESC[24m ESC[4mregexpESC[24m ...
ESC[1mman -K ESC[22mESC[4mmanESC[24m ESC[4moptionsESC[24m [[ESC[4msectionESC[24m] ESC[4mtermESC[24m ...
ESC[1mman -f ESC[22mESC[4mwhatISC[24m ESC[4moptionsESC[24m ESC[4mpageESC[24m ...
ESC[1mman -l ESC[22mESC[4mmanESC[24m ESC[4moptionsESC[24m ESC[4mfileESC[24m ...
ESC[1mman -wESC[22mESC[1m-W ESC[22mESC[4mmanESC[24m ESC[4moptionsESC[24m ESC[4mpageESC[24m ...
ESC[1mDESCRIPTIONESC[0m
ESC[1mman ESC[22mis the system's manual pager. Each ESC[4mpageESC[24m argument given to ESC[1mman ESC[22mis not
. The ESC[4manualESC[24m ESC[4mpageESC[24m associated with
each of these arguments is then found and displayed. A ESC[4msectionESC[24m, if provided, will direct ESC[1mman ESC
f the manual. The default action
is to search in all of the available ESC[4msectionsESC[24m following a pre-defined order (see ESC[1mDEFAULTSESC[22m
found, even if ESC[4mpageESC[24m exists in several
ESC[4msectionsESC[24m.

The table below shows the ESC[4msectionESC[24m numbers of the manual followed by the types of pages they contain.

1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within programs, their prototypes)
```

Рис. 6: Запуск программы

## Код с использованием встроенную переменную \$RANDOM

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

Рис. 7: Код для 3-й задачи

```
[svpavliuchenkov@fedora lab14]$ ./3_lab14  
aazgf  
[svpavliuchenkov@fedora lab14]$
```

Рис. 8: Запускаю программу

Я улучшил свои навыки работы с `bash`. Изучил команду `less`, использовал перенаправление в виртуальный терминал и научился пользоваться командой `exit()`.