

Отчёт по лабораторной работе №14

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

Сергей Витальевич Павлюченков

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Контрольные вопросы	14

Список иллюстраций

3.1	Код программы семафора	8
3.2	Код командного файла	9
3.3	Запуск программы	9
3.4	Структура папки man1	10
3.5	Код для 2-ой задачи	11
3.6	Запуск программы	11
3.7	Код для 3-й задачи	12

Список таблиц

1 Цель 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

Написал командный файл, реализующий упрощённый механизм семафоров. Командный файл в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использует его в течение некоторого времени $t_2 < t_1$, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом)

```
1 mc [svpavliuchnikov@fedora]:~/work/os/1
mc [svpavliuchnikov@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 "Process is locked"
    sleep $t1
done

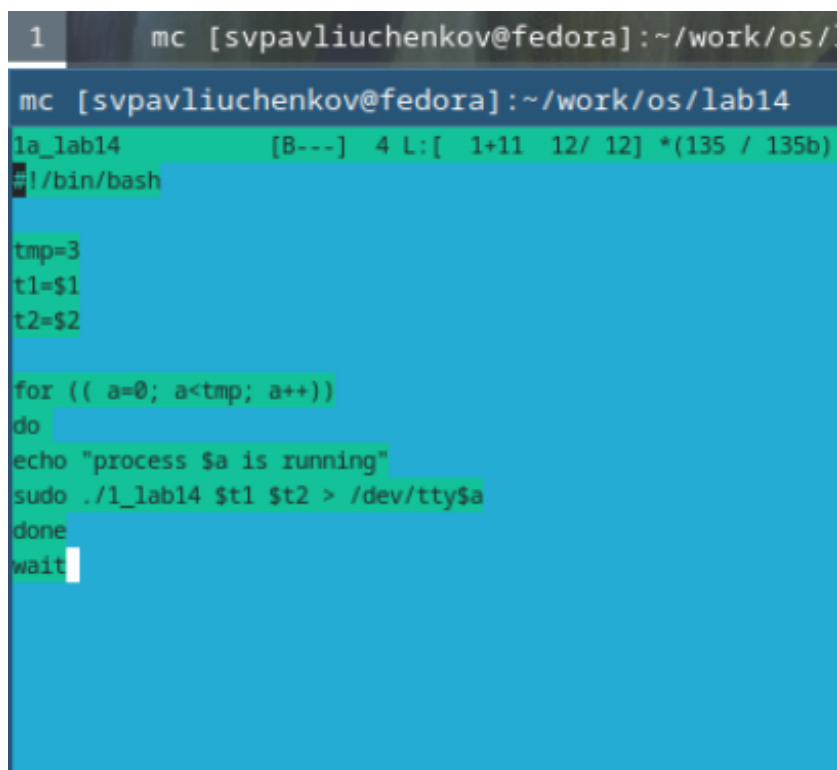
echo "Process $$ is running"
touch $myfile
sleep $t2

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

Рис. 3.1: Код программы семафора

Написал командный файл, который будет запускать код в одном виртуальном

терминале в фоновом режиме и перенаправит 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.



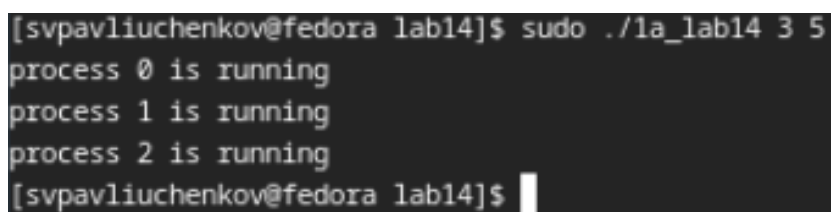
```
1 mc [svpavliuchnikov@fedora]:~/work/os/1
mc [svpavliuchnikov@fedora]:~/work/os/lab14
1a_lab14 [B---] 4 L:[ 1+11 12/ 12] *(135 / 135b)
#!/bin/bash

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
```

Рис. 3.2: Код командного файла

Запускаю программу, как видно вывод программы семафора переенаправляется в другой терминал.



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

Рис. 3.3: Запуск программы

Приступаю к выполнению 2-го задания. Изучаю структуру 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

```

Рис. 3.4: Структура папки man1

Реализовал команду man с помощью командного файла. Командный файл получает 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.

```
[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/$file.1.gz"
else
echo "no such file"
fi[svpavliuchenkov@fedora lab14]$
```

Рис. 3.5: Код для 2-ой задачи

Открывается командой less сразу же просмотр содержимое справки о коман-
де.

```
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[1man ESC[22mESC[4mmanESC[24m ESC[4moptionsESC[24m [ESC[4msectionESC[24m] ESC[4mpageESC[24m ...] ...
ESC[1man -k ESC[22mESC[4maproposESC[24m ESC[4moptionsESC[24m] ESC[4mregexESC[24m ...
ESC[1man -K ESC[22mESC[4mmanESC[24m ESC[4moptionsESC[24m] ESC[4msectionESC[24m] ESC[4mtermESC[24m ...
ESC[1man -f ESC[22mESC[4mwhatISC[24m ESC[4moptionsESC[24m] ESC[4mpageESC[24m ...
ESC[1man -l ESC[22mESC[4mmanESC[24m ESC[4moptionsESC[24m] ESC[4mfileESC[24m ...
ESC[1man -wESC[22mESC[1m-W ESC[22mESC[4mmanESC[24m ESC[4moptionsESC[24m] ESC[4mpageESC[24m ...
ESC[1mDESCRIPTIONESC[0m
ESC[1man ESC[22mis the system's manual pager. Each ESC[4mpageESC[24m argument given to ESC[1man 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[1man ESC[22m
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 sev-
eral 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 C library routines (functions provided by the kernel)
4 Shared library routines (functions provided by the kernel)
5 Macro definitions
6 File formats and conventions
7 Miscellaneous
8 System administration
9 Reference
10 Miscellaneous
```

Рис. 3.6: Запуск программы

Приступаю к 3 задаче. Используя встроенную переменную \$RANDOM, напи-
сал командный файл, генерирующий случайную последовательность букв ла-
тинского алфавита.

```
1 mc [svpavliuchenkov@fedora]:  
mc [svpavliuchenkov@fedora]:~/work/  
3_lab14 [B---] 8 L:[ 1+35 36/ 36]  
#!/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. 3.7: Код для 3-й задачи

Как видно, код генерирует случайную строку из латинских букв.

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

Выводы

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

4 Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке: `while [$1 != "exit"]` в том, что данная конструкция скорее всего никогда не закончится, так как `$1` не меняется. Еще не хватает пробелов после `[` и перед `]`.
2. Как объединить (конкатенация) несколько строк в одну? `a=${A+$B}`
3. Найдите информацию об утилите `seq`. Какими иными способами можно реализовать её функционал при программировании на `bash`? например `for ((a=1; a<80; a++))` а будет иметь те же значения, что и `seq`.
4. Какой результат даст вычисление выражения `$((10/3))`? Выдаст 3
5. Укажите кратко основные отличия командной оболочки `zsh` от `bash`.

ZSH является расширенным аналогом BASH и имеет с ним обратную совместимость, добавляя ему большое количество улучшений. Ключевые особенности ZSH: Встроенное автозаполнение с расширенным функционалом.

6. Проверьте, верен ли синтаксис данной конструкции `1 for ((a=1; a <= LIMIT; a++))`

Не хватает пробелов после `((` и перед `)`

7. Сравните язык `bash` с какими-либо языками программирования. Какие преимущества у `bash` по сравнению с ними? Какие недостатки? `Bash` позволяет напрямую взаимодействовать с консолью, что не позволяет `python`. Из недостатков