
Front matter

title: "Отчёт по лабораторной работе №12" subtitle: " Программирование в командном процессоре ОС UNIX. Расширенное программирование " author: "Федорина Эрнест Васильевич НКНбд-01-21"

Generic otions

lang: ru-RU toc-title: "Содержание"

Bibliography

bibliography: bib/cite.bib csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

Pdf output format

toc: true # Table of contents toc-depth: 2 lof: true # List of figures lot: true # List of tables
fontsize: 12pt linestretch: 1.5 papersize: a4 documentclass: scrreprt ## I18n polyglossia
polyglossia-lang: name: russian options: - spelling=modern - babelshorthands=true
polyglossia-otherlangs: name: english ## I18n babel babel-lang: russian babel-otherlangs:
english ## Fonts mainfont: PT Serif romanfont: PT Serif sansfont: PT Sans monofont: PT
Mono mainfontoptions: Ligatures=TeX romanfontoptions: Ligatures=TeX sansfontoptions:
Ligatures=TeX,Scale=MatchLowercase monofontoptions: Scale=MatchLowercase,Scale=0.9
Biblatex biblatex: true biblio-style: "gost-numeric" biblatexoptions: - parenttracker=true
- backend=biber - hyperref=auto - language=auto - autolang=other* - citestyle=gost-
numeric ## Pandoc-crossref LaTeX customization figureTitle: "Рис." tableTitle: "Таблица"
listingTitle: "Листинг" lofTitle: "Список иллюстраций" lotTitle: "Список таблиц" lolTitle:
"Листинги" ## Misc options indent: true header-includes: -

keep figures where there are in the text

– # keep figures where there are in the text

Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

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.

Теоретическое введение

При интерактивной работе с операционной системой пользователь постоянно сталкивается с необходимостью отдавать ей команды. В UNIX эту работу выполняет программа, которая называется командным процессором (shell). Иногда командный процессор называют шеллом или оболочкой, или интерпретатором команд (последнее неточно, потому что круг задач командного процессора шире, чем интерпретация команд).

Shell действует как посредник между вами и ядром операционной системы. Ядро – это часть операционной системы, которая всегда находится в памяти компьютера, это программа. Командный процессор преобразует ваши команды в инструкции для операционной системы, а операционная система превращает их в инструкции для

аппаратных средств компьютера. По сути, именно оболочка придает определенную “персонализацию” системам UNIX.

Командный процессор выполняет в системе следующие задачи:

- интерпретация команд пользователя, в том числе разбор командной строки;
- запуск программ;
- организация перенаправлений потоков между процессами;
- интерпретация языка скриптов и их выполнение;
- управление заданиями;
- интерпретация шаблонов имен файлов;
- подстановка имен файлов в командную строку.

Кроме того, shell является мощным языком программирования.

Любая команда, являющаяся отдельной программой, т.е. не встроенной в интерпретатор, будет выполняться одинаково, независимо от shell'a. Например, если вы хотите что-то напечатать, команда печати всегда работает одинаково.

Некоторые команды встроены в shell, т.е. они являются частью программы оболочки и, как следствие, могут выполняться по-разному в зависимости от оболочки, в которой они запускаются. Есть три вида команд, встроенных в shell:

- общие команды запускаются несколько быстрее, так как они являются частью оболочки;
- команды адаптации позволяют адаптировать оболочку;
- команды программирования образуют язык программирования оболочки.

При смене shell'a вы не заметите никакой разницы между общими командами, которые встроены просто для повышения быстродействия. Однако команды адаптации и программирования изменяются.

Примером общих команд, встроенных в оболочку, служат команды `cd`, `echo`, `pwd`, `login`, `umask`.

Адаптация включает в себя создание новых команд или новых имен для старых команд и привлечение новых возможностей. Примером общепринятой адаптации служит изменение стимула (или приглашения системы).

Можно персонализировать свою работу, привлекая удобные возможности, встроенные в различные оболочки. Почти все такие возможности связаны с этапом преобразования команды. Большинство различий между оболочками кроется в том, как они адаптированы.

Примером команд адаптации могут служить следующие команды:

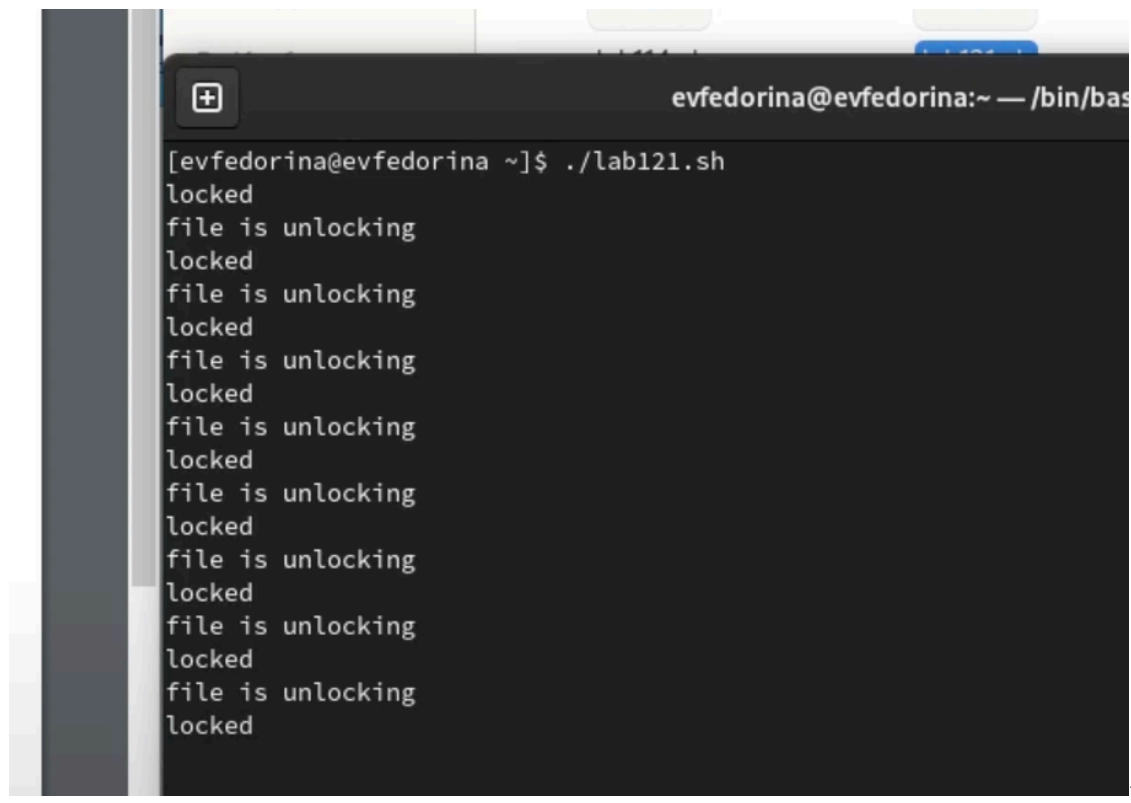
- alias – создает новое имя для команды;
- export – создает переменную среды;
- set – включает и выключает некоторые опции shell'a;
- unalias – удаляет псевдоним команды.

Выполнение лабораторной работы

Написал 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) (рис.1,2)

```
1 #!/bin/bash
2 lockfile="./locking.file"
3 exec fn >${lockfile}
4 if test -f "${lockfile}"
5 then while [ 1!=0 ]
6 do
7 if flock -n ${fn}
8 then
9 echo "locked"
10 sleep 3
11 echo "file is unlocking"
12 flock -u ${fn}
13 else
14 echo "is locked"
15 sleep 3
16 fi
17 done
18 fi
```

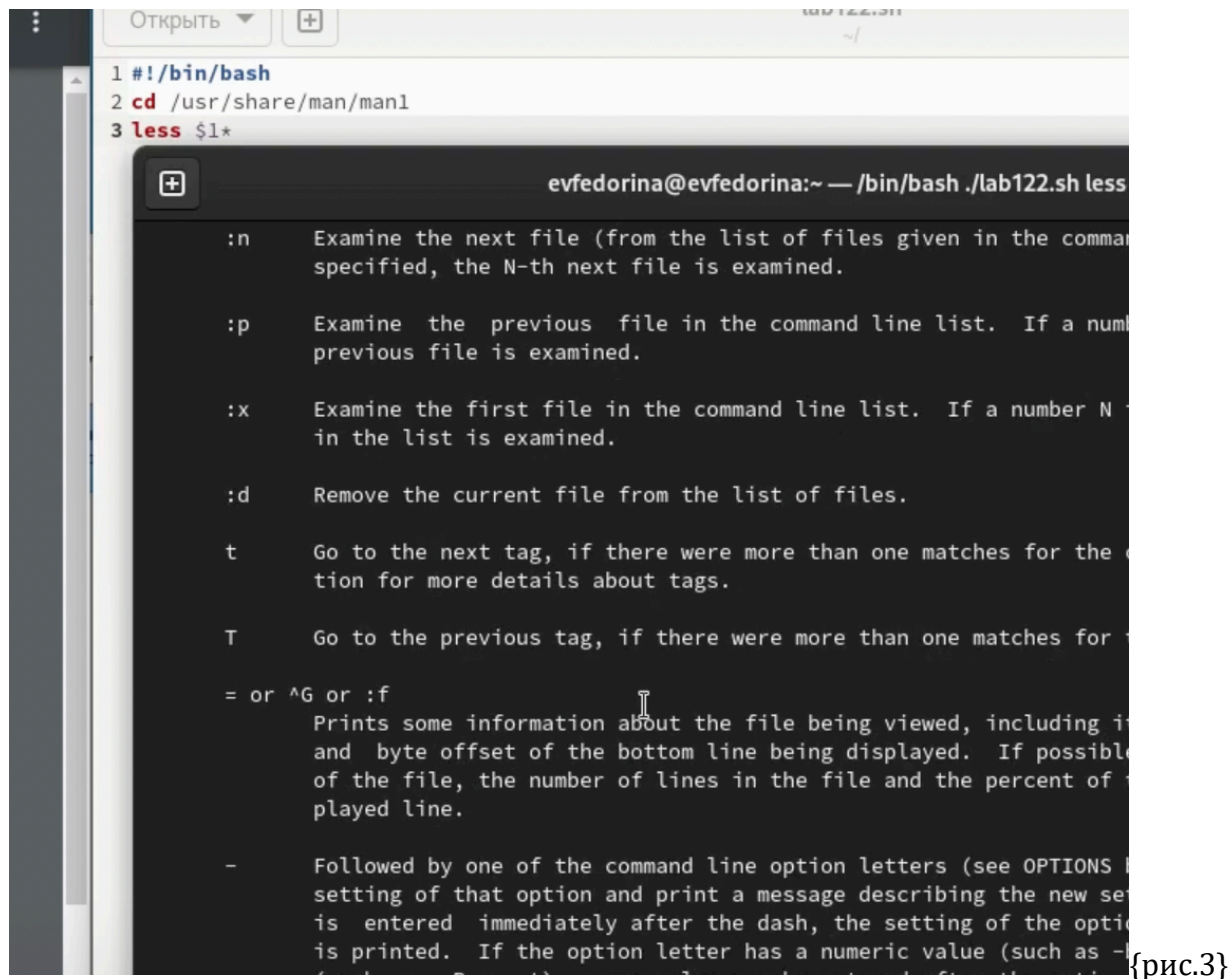
{pic.1}



```
evfedorina@evfedorina:~ — /bin/bas
[evfedorina@evfedorina ~]$ ./lab121.sh
locked
file is unlocking
locked
file is unlocking
locked
file is unlocking
locked
file is unlocking
locked
file is unlocking
locked
file is unlocking
locked
file is unlocking
locked
file is unlocking
locked
```

{рис.2}

Реализовать команду map с помощью командного файла.(рис.3)



The image shows a terminal window with a light gray title bar containing the text "Открыть" and a "+" icon. The terminal content is as follows:

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

Below this, a window titled "evfedorina@evfedorina:~ — /bin/bash ./lab122.sh less" displays the man page for 'less'. The man page lists various navigation and display options:

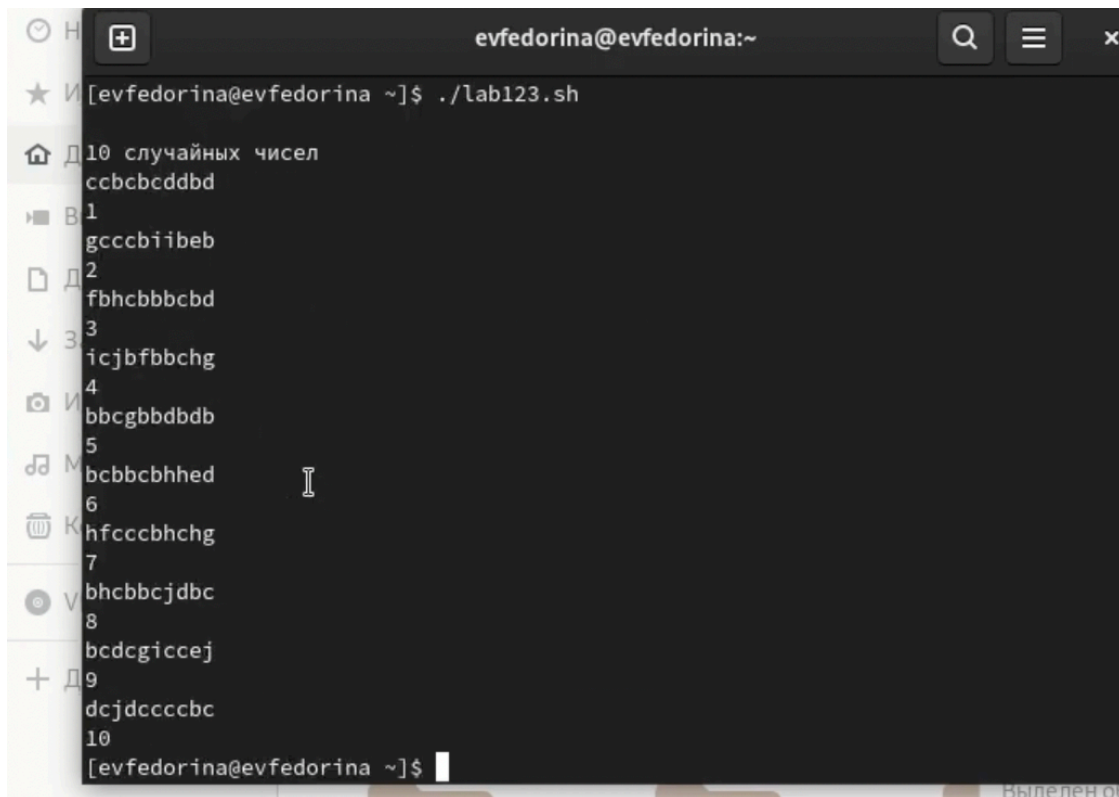
- `:n` Examine the next file (from the list of files given in the command line, the N-th next file is examined).
- `:p` Examine the previous file in the command line list. If a number N is given, the N-th previous file is examined.
- `:x` Examine the first file in the command line list. If a number N is given, the N-th file in the list is examined.
- `:d` Remove the current file from the list of files.
- `t` Go to the next tag, if there were more than one matches for the tag, go to the first one. For more details about tags.
- `T` Go to the previous tag, if there were more than one matches for the tag, go to the last one.
- `= or ^G or :f` Prints some information about the file being viewed, including its name, size, and byte offset of the bottom line being displayed. If possible, it also prints the number of lines in the file and the percent of the file that has been displayed.
- `-` Followed by one of the command line option letters (see OPTIONS), the setting of that option and print a message describing the new setting. If the option letter has a numeric value (such as `-n 10`), the setting is entered immediately after the dash, the setting of the option is printed. If the option letter has a numeric value (such as `-n 10`), the setting is entered immediately after the dash, the setting of the option is printed.

Используя встроенную переменную \$RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита.(рис.4,5)



```
1 #!/bin/bash
2 a=10
3 b=1
4 c=1
5 echo
6 echo "10 случайных чисел"
7 while ((b!=((a+1))))
8 do
9     echo $(for i=1;i<=10;i++; do printf '%s' "${RANDOM:0:1}"; done) | tr ' [0-9]' ' [a-z]'
10    echo $c
11    ((b+=1))
12    ((c+=1))
13 done
```

{рис.4}



```
evfedorina@evfedorina:~$ ./lab123.sh
10 случайных чисел
ccbcbbcd
1 gcccbiibeb
2 fbhcbbbc
3 icjbfbbchg
4 bcbgbbdb
5 bcbcbhhed
6 hfccbbchg
7 bhcbbcjdb
8 bcdcgiccej
9 dcjdccc
10
[evfedorina@evfedorina ~]$
```

Выделен об {рис.5}

Выводы

Изучил основы программирования в оболочке ОС UNIX. Научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.