

Отчёта по лабораторной работе №11:

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

Кононов Алексей Сергеевич

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	7
4	Контрольные вопросы	11
5	Выводы	14
	Список литературы	15

Список иллюстраций

3.1	Задание 1	7
3.2	Выполнение и результат 1	8
3.3	man1	8
3.4	Задание 2	9
3.5	Выполнение и результат 2	9
3.6	Справка ls	9
3.7	Задание 3	10
3.8	Выполнение и результат 3	10

1 Цель работы

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

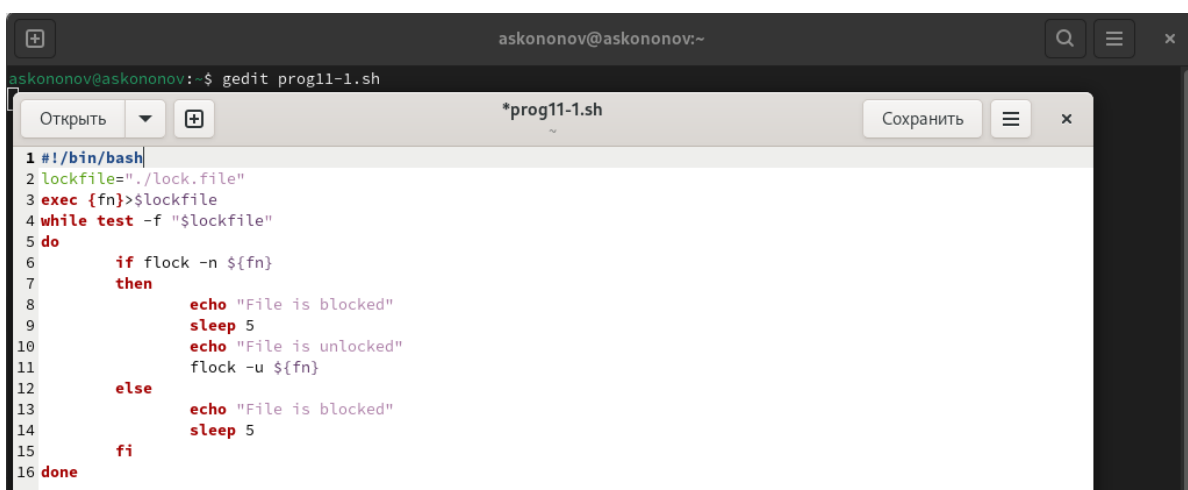
2 Задание

1. Написать командный файл, реализующий упрощённый механизм semaфоров. 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 имела 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

1. Напишем 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 имела возможность взаимодействия трёх и более процессов. (рис. 3.1):



```
askononov@askononov:~$ gedit prog11-1.sh
*prog11-1.sh
Сохранить

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

Рис. 3.1: Задание 1

Делаем файлы исполняемыми и выводим результат (рис. 3.2).

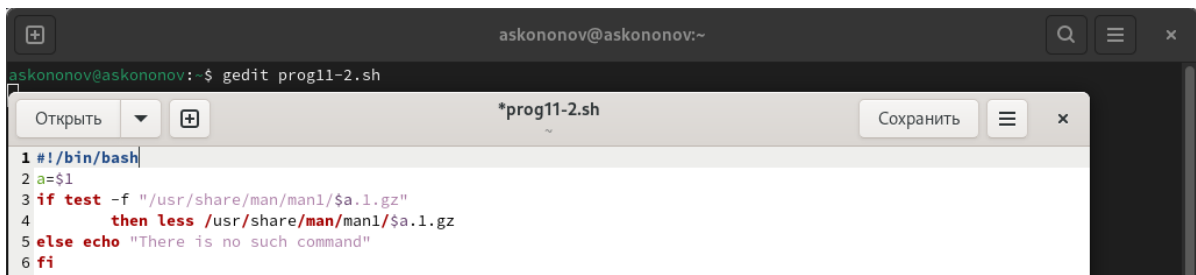
```
asknononov@asknononov:~$ gedit progll-1.sh
asknononov@asknononov:~$ chmod +x progll-1.sh
asknononov@asknononov:~$ bash progll-1.sh
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is unlocked
File is blocked
^C
asknononov@asknononov:~$
```

Рис. 3.2: Выполнение и результат 1

2. Реализуем 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`. (рис. 3.4):

```
askononov@askononov:~$ ls /usr/share/man/man1
:1.gz          nbdkit-service.1.gz
'[:1.gz'       nbdkit-sparse-random-plugin.1.gz
a2ping.1.gz    nbdkit-split-plugin.1.gz
ab.1.gz        nbdkit-ssh-plugin.1.gz
abrt.1.gz      nbdkit-swab-filter.1.gz
abrt-action-analyze-backtrace.1.gz nbdkit-tls.1.gz
abrt-action-analyze-c.1.gz          nbdkit-tls-fallback-filter.1.gz
abrt-action-analyze-cpp-locals.1.gz nbdkit-truncate-filter.1.gz
abrt-action-analyze-java.1.gz       nbdkit-zero-plugin.1.gz
abrt-action-analyze-oops.1.gz       nc.1.gz
abrt-action-analyze-python.1.gz     ncat.1.gz
abrt-action-analyze-vmcore.1.gz     ndctl.1.gz
abrt-action-analyze-vulnerability.1.gz ndctl-activate-firmware.1.gz
```

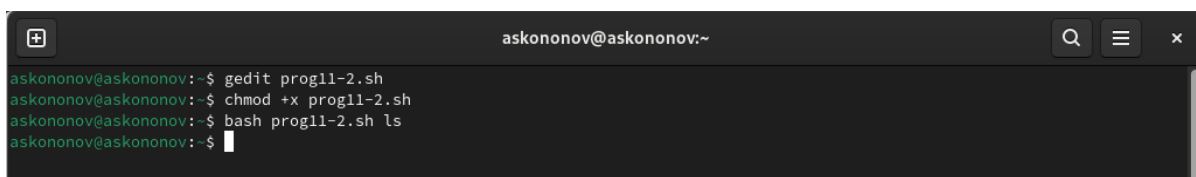
Рис. 3.3: man1



```
askononov@askononov:~$ gedit prog11-2.sh
*prog11-2.sh
1 #!/bin/bash
2 a=$1
3 if test -f "/usr/share/man/man1/$a.1.gz"
4     then less /usr/share/man/man1/$a.1.gz
5 else echo "There is no such command"
6 fi
```

Рис. 3.4: Задание 2

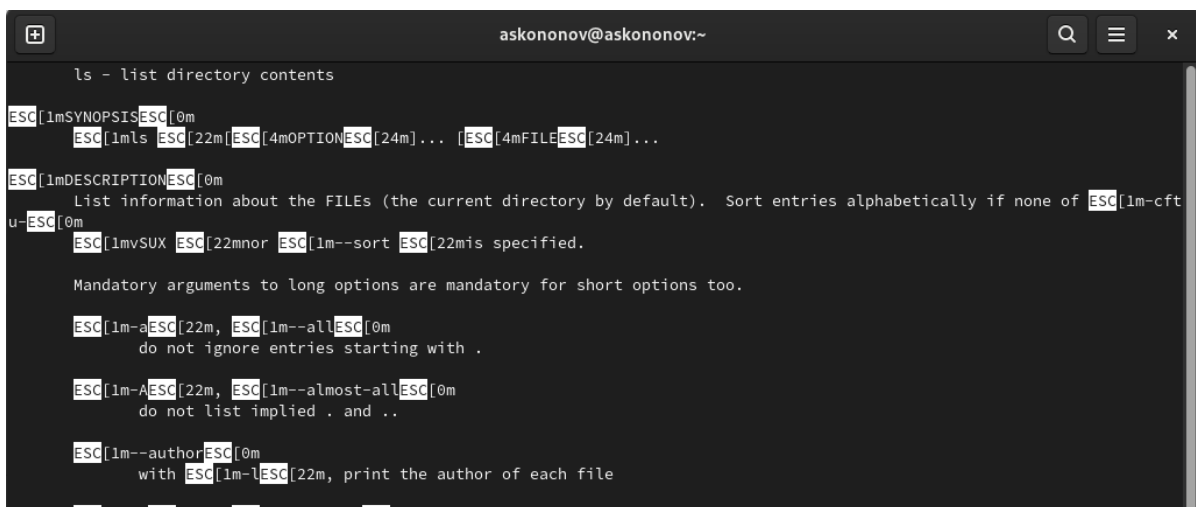
Делаем файлы исполняемыми и проверяем работу программы, запросив справку о команде `ls` (рис. 3.5)



```
askononov@askononov:~$ gedit prog11-2.sh
askononov@askononov:~$ chmod +x prog11-2.sh
askononov@askononov:~$ bash prog11-2.sh ls
askononov@askononov:~$
```

Рис. 3.5: Выполнение и результат 2

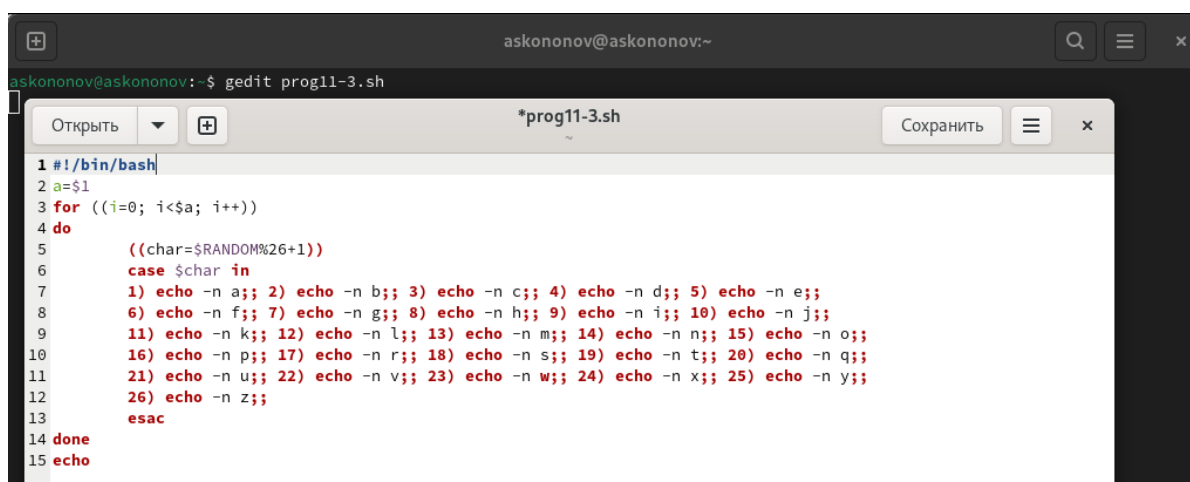
В итоге мы получаем справку команды `ls`, которую запрашивали (рис. 3.6).



```
ls - list directory contents
SYNOPSIS
ls [OPTION]... [FILE]...
DESCRIPTION
List information about the FILES (the current directory by default). Sort entries alphabetically if none of -cftu- is specified.
Mandatory arguments to long options are mandatory for short options too.
Options
-a, --all do not ignore entries starting with .
-A, --almost-all do not list implied . and ..
--author with -l, print the author of each file
```

Рис. 3.6: Справка ls

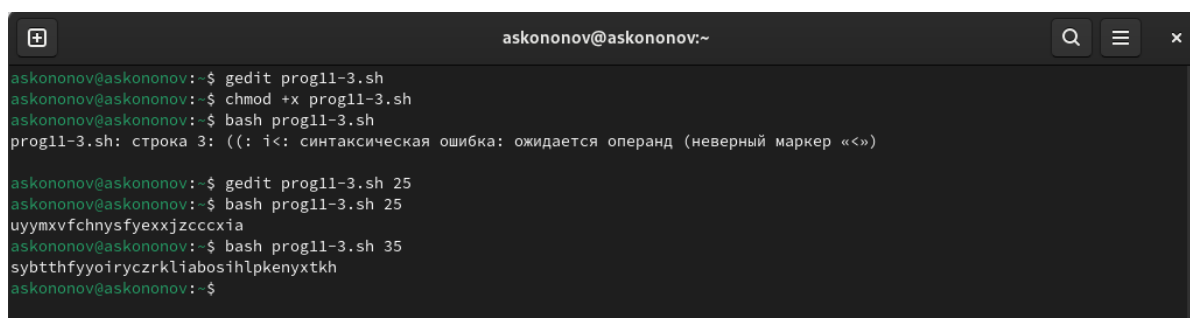
- Используя встроенную переменную `$RANDOM`, напомним командный файл, генерирующий случайную последовательность букв латинского алфавита. (рис. 3.7):



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

Рис. 3.7: Задание 3

Делаем файлы исполняемыми и выводим результат (рис. 3.8).



```
askononov@askononov:~$ gedit prog11-3.sh
askononov@askononov:~$ chmod +x prog11-3.sh
askononov@askononov:~$ bash prog11-3.sh
prog11-3.sh: строка 3: ((: i<: синтаксическая ошибка: ожидается операнд (неверный маркер «<»)

askononov@askononov:~$ gedit prog11-3.sh 25
askononov@askononov:~$ bash prog11-3.sh 25
uuyymxvfchnysfyexxjzcccxia
askononov@askononov:~$ bash prog11-3.sh 35
sybtthfyoyioryczrkliabosihlpkenyxtkh
askononov@askononov:~$
```

Рис. 3.8: Выполнение и результат 3

4 Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке: `while [$1 != "exit"]` Ошибка в том что после `[` и до `]` должен идти пробел, еще имеет смысл обернуть `$1` в двойные кавычки, чтобы эта переменная корректно обрабатывалась даже если она содержит пробелы.
2. Как объединить (конкатенировать) несколько строк в одну? Есть множество способов конкатенации строк в `bash`, я разберу два основных. Предположим у нас есть переменные `a` и `b`, чтобы добавить к строке `a` строку `b` справа можно использовать следующий синтаксис: `a+=$b`. Записать результат конкатенации в новую переменную можно так: `c="ab"`.
3. Найдите информацию об утилите `seq`. Какими иными способами можно реализовать её функционал при программировании на `bash`? Утилита `seq` используется для генерации последовательностей, ее можно использовать в циклах. Например: `for i in $(seq 0 2 10)`, переменная `i` будет принимать значения от 0 до 10 включительно с шагом 2. Очевидно, что чтобы добиться похожего результата можно использовать Си-подобный цикл `for`: `for ((i=0;i<=10;i+=2))`
4. Какой результат даст вычисление выражения `$((10/3))`? Результат будет 3, так как будет выполнено целочисленное деление.
5. Укажите кратко основные отличия командной оболочки `zsh` от `bash`.
 - Интерактивность: `zsh` известен своими улучшенными интерактивными возможностями, такими как более продвинутый автодополнение команд и файлов, что делает его более удобным для интерактивного

использования.

- Темы и плагины: zsh поддерживает темы и плагины, что позволяет пользователям настраивать свой рабочий интерфейс и расширять функциональность оболочки.
- Синтаксис: zsh обладает более гибким синтаксисом, включая улучшенные возможности для работы с массивами и ассоциативными массивами.
- Совместимость: zsh во многом совместим с bash, но включает множество дополнительных возможностей, которые могут не работать в bash без изменений.
- Контекстное автодополнение: zsh предлагает более продвинутое контекстное автодополнение, которое может учитывать не только имена файлов и команд, но и их параметры.
- Модульность: zsh разработан с учётом модульности, что позволяет легко добавлять новые функции и интегрировать внешние скрипты.

6. Проверьте, верен ли синтаксис данной конструкции

`for ((a=1; a <= LIMIT; a++))` Да, синтаксис верен. В двойных круглых скобках можно использовать переменные без знака доллара.

7. Сравните язык bash с какими-либо языками программирования. Какие преимущества у bash по сравнению с ними? Какие недостатки?

- Преимущества bash:
 - Скорость разработки: bash-скрипты обычно проще и быстрее писать для простых задач, особенно для автоматизации командной строки и системных операций.
 - Встроенная поддержка UNIX-команд: bash нативно интегрируется с UNIX-командами и утилитами, что делает его мощным инструментом для системного администрирования.
 - Портативность: bash-скрипты легко переносить между различными UNIX-подобными системами без изменений.

- Недостатки bash:
 - Ограниченные возможности программирования: bash не имеет такого богатого набора функций программирования, как C++ или Python, например, объектно-ориентированное программирование или обширные стандартные библиотеки.
 - Медленная производительность: Для сложных задач или задач, требующих интенсивных вычислений, bash может работать медленнее, чем C++ или Python.
 - Сложность синтаксиса: Некоторые аспекты синтаксиса bash могут быть непривычными или запутанными для новичков, особенно при работе с текстовыми строками и файлами.

5 Выводы

В этой лабораторной работе мы закрепили свои знания об основах программирования на языке `bash`. Написав простейшие реализации семафора и команды `map`, используя переменную `$RANDOM` для генерации последовательности букв английского алфавита.

Список литературы