

Лабораторная работа №12

Дисциплина: Операционные системы

Гибшер Кирилл Владимирович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	8
4	Выполнение лабораторной работы	10
5	Вывод	14

Список иллюстраций

4.1	Скрипт первого задания	10
4.2	Результат	10
4.3	скрипт 2 задания	11
4.4	Положительный результат работы	11
4.5	Отрицательный результат работы командного файла	12
4.6	Скрипт третьего задания	12
4.7	Запуск и проверка работоспособности кода	13

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 алфа-

вита. Учтите, что \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767.

3 Теоретическое введение

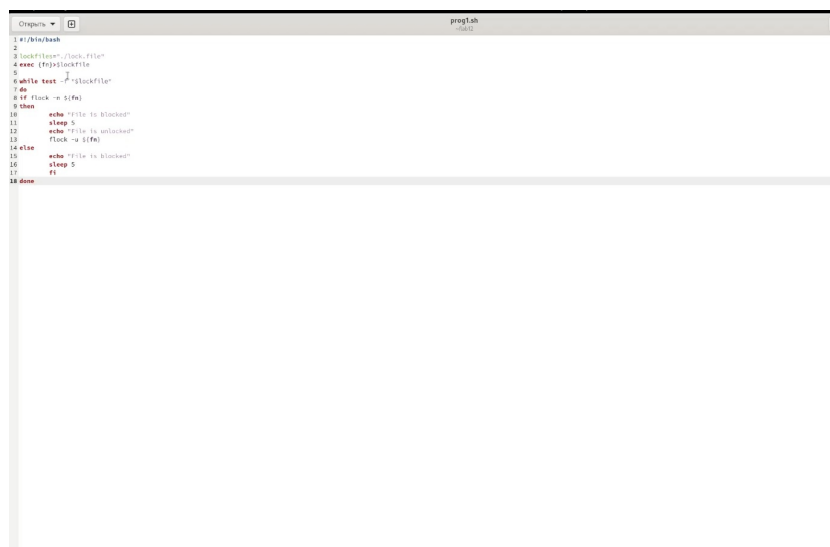
Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: - оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;

- C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).
- POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим

основные элементы программирования в оболочке `bash`. В других оболочках большинство команд будет совпадать с описанными ниже.

4 Выполнение лабораторной работы

1. Напишем скрипт для первого задания. (рис. [4.1])



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

Рис. 4.1: Скрипт первого задания

2. Запустим командный файл и проверим его работоспособность, прописав соответствующие опции в команде. (рис. [4.2])



```
File is blocked
File is unlocked
File is blocked
^C
```

Рис. 4.2: Результат

3. Напишем скрипт для второго задания. (рис. [4.3])

```
Outputs  prog2.sh
1 #!/bin/bash
2
3
4 if test -f /usr/share/man/man1/ls.1.gz
5 then less /usr/share/man/man1/ls.1.gz
6 else
7 echo "there is no such command"
8 fi
```

Рис. 4.3: скрипт 2 задания

4. Результат работы командного файла при использовании существующей команды , в данном случае ls (рис. [4.4])

```
kvgibsher@kvgibsher:~/lab12 — bash prog2.sh ls
ls - list directory contents
ESC[1mSYNOPSISESC[0m
ESC[1mls ESC[22mESC[4mOPTIONESC[24m]... ESC[4mFILEESC[24m]...
ESC[1mDESCRIPTIONESC[0m
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of ESC[1m-cftuvSUX ESC[22mnor ESC[1m-
-sort ESC[22mis speci-
fied.
Mandatory arguments to long options are mandatory for short options
too.
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
```

Рис. 4.4: Положительный результат работы

5. Результат работы командного файла при использовании несуществующей

команды. (рис. [4.5])

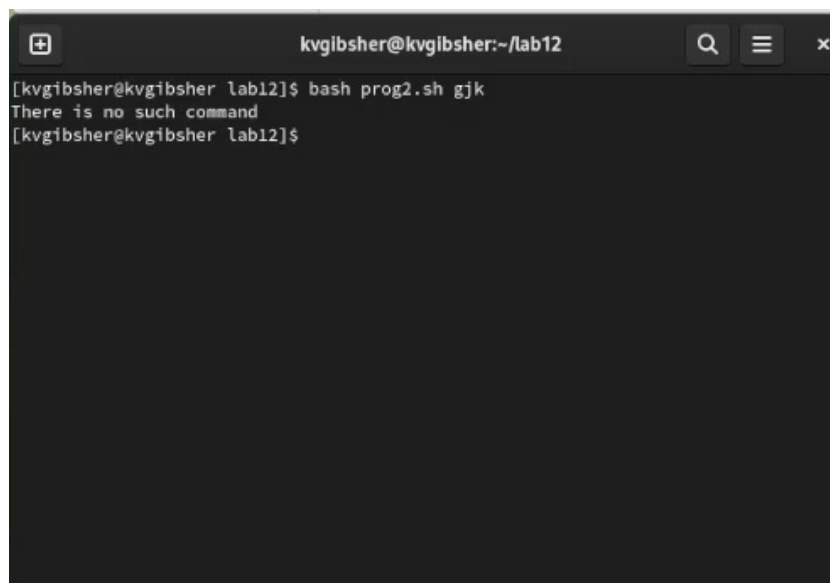


Рис. 4.5: Отрицательный результат работы командного файла

6. Напишем скрипт для третьего задания (рис. [4.6])

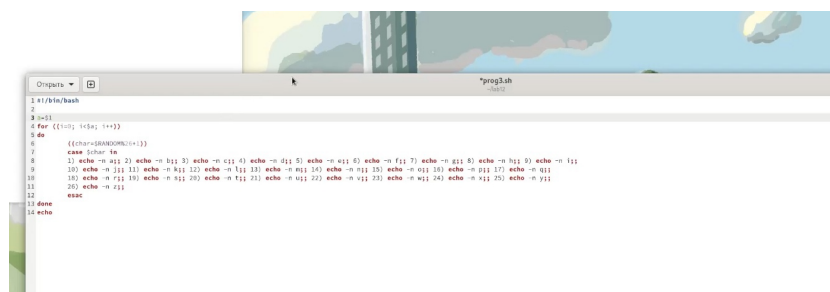


Рис. 4.6: Скрипт третьего задания

7. Запустим командный файл и увидим , что терминал в действительности выдает нам случайный набор из латинских букв в заданной пользователем размерности. (рис. [4.7])

```
kvgibsher@kvgibsher lab12]$ gedit prog3.sh
kvgibsher@kvgibsher lab12]$ bash prog3.sh 16
mastrdoqgmfdxp
kvgibsher@kvgibsher lab12]$ bash prog3.sh 10
nyobneccs
kvgibsher@kvgibsher lab12]$
```

Рис. 4.7: Запуск и проверка работоспособности кода

5 Вывод

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