

Лабораторная работа №12. Программирование в командном процессоре ОС UNIX. Расширенное программирование

Подготовил:

Королев Адам Маратович

Группа: НПИбд-02-21

Студенческий билет: № 1032217060

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

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

- С-оболочка (или `csh`) - надстройка на оболочке Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд
- оболочка Корна (или `ksh`) - напоминает оболочку С, 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.

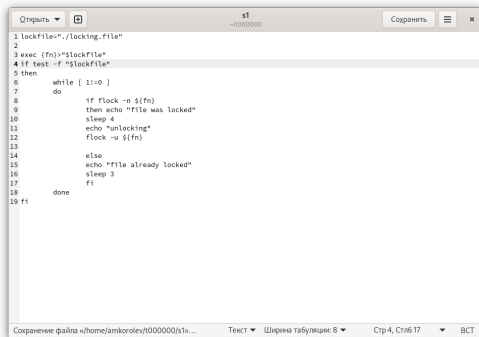
Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода.

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.

Выполнение лабораторной работы:

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

Пишем скрипт.

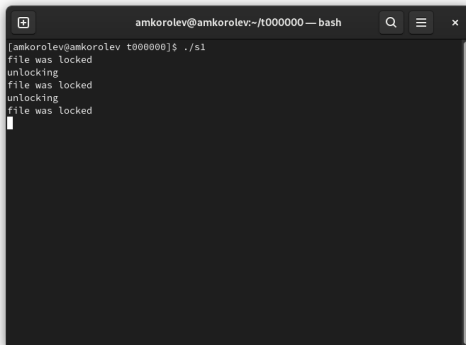


The image shows a terminal window titled 's1' with a path '~root00000'. The window contains a shell script with the following content:

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

The status bar at the bottom of the window displays: "Сохранение файла «/home/amikorelev/t000000/s1»...", "Текст", "Шрифт: табуляция: 8", "Стр 4, Стб 17", and "ВСТ".

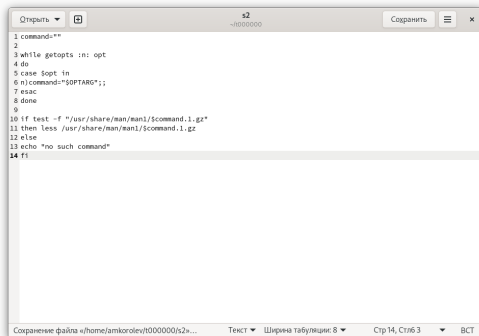
Figure 1: Пишем скрипт

A terminal window with a dark background and light text. The title bar at the top reads 'amkorolev@amkorolev:~/t000000 — bash'. The terminal content shows the execution of a script named 's1'. The output consists of five lines: 'file was locked', 'unlocking', 'file was locked', 'unlocking', and 'file was locked'. A cursor is visible on the line following the last output.

```
amkorolev@amkorolev:~/t000000 — bash
[amkorolev@amkorolev t000000]$ ./s1
file was locked
unlocking
file was locked
unlocking
file was locked
```

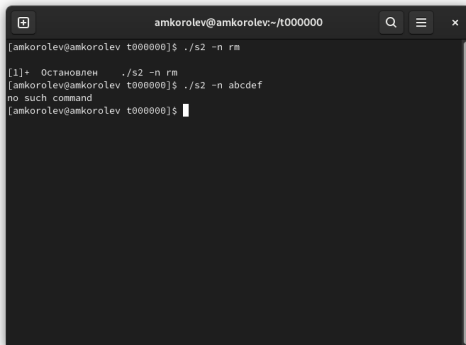
Figure 2: Выполняем скрипт

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



```
1 command=""
2
3 while getopts :n: opt
4 do
5 case $opt in
6 n) command="$OPTARG";;
7 esac
8 done
9
10 if test -f "/usr/share/man/man1/$command.1.gz"
11 then less /usr/share/man/man1/$command.1.gz
12 else
13 echo "no such command"
14 fi
```

Figure 3: Пишем скрипт

A terminal window with a dark background and light text. The title bar shows 'amkorolev@amkorolev:~/t000000'. The terminal content shows a sequence of commands and their outputs. The first command is './s2 -n rm', which results in a message '[1]+ Остановлен ./s2 -n rm'. The second command is './s2 -n abcdef', which results in the error 'no such command'. The prompt returns to the user.

```
amkorolev@amkorolev:~/t000000$ ./s2 -n rm
[1]+  Остановлен  ./s2 -n rm
amkorolev@amkorolev:~/t000000$ ./s2 -n abcdef
no such command
amkorolev@amkorolev:~/t000000$
```

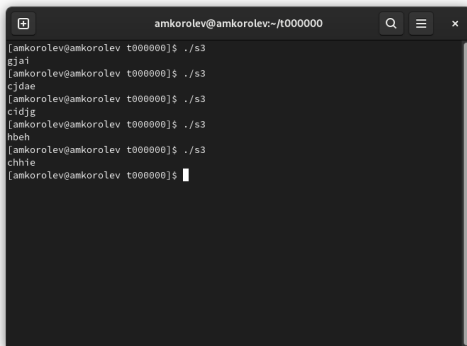
Figure 4: Выполняем скрипт

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.

Пишем скрипт.



Figure 5: Пишем скрипт

A terminal window with a dark background and light text. The title bar at the top reads 'amkorolev@amkorolev:~/t000000'. The terminal shows a series of commands and their outputs. The first command is './s3', which outputs 'gjai'. This is followed by another './s3' command outputting 'cjdae', then another './s3' outputting 'cidjg', then another './s3' outputting 'hbeh', and finally another './s3' outputting 'chhie'. The last line shows the prompt '[amkorolev@amkorolev t000000]\$' with a cursor, indicating the script execution is complete.

```
amkorolev@amkorolev:~/t000000
[amkorolev@amkorolev t000000]$ ./s3
gjai
[amkorolev@amkorolev t000000]$ ./s3
cjdae
[amkorolev@amkorolev t000000]$ ./s3
cidjg
[amkorolev@amkorolev t000000]$ ./s3
hbeh
[amkorolev@amkorolev t000000]$ ./s3
chhie
[amkorolev@amkorolev t000000]$
```

Figure 6: Выполняем скрипт

Выводы:

- В процессе выполнения работы научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.