

Отчёт по лабораторной работе №14

Операционные системы

Бекауов Артур Тимурович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Выводы	12
5	Ответы на онтрольные вопросы	13

Список иллюстраций

3.1	Программа №1 - Создание файла	8
3.2	Программа №1 - Выполнение	9
3.3	Программа №2 - создание файлов	9
3.4	Программа №2 - выполнение 1	10
3.5	Программа №2 - выполнение 2	10
3.6	Программа №3 - создание файла	10
3.7	Программа №3 - выполнение	11

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.

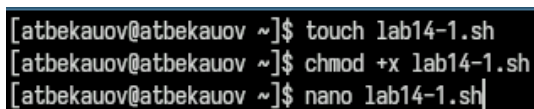
2 Задание

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

Создаю файл lab14-1.sh для новой программы меняю права доступа, разрешая его выполнение, таким образом файл становится исполняемым. (рис. 3.1).



```
[atbekauov@atbekauov ~]$ touch lab14-1.sh
[atbekauov@atbekauov ~]$ chmod +x lab14-1.sh
[atbekauov@atbekauov ~]$ nano lab14-1.sh
```

Рис. 3.1: Программа №1 - Создание файла

Открываю файл в редакторе nano и записываю следующий код программы:

```
#!/bin/bash

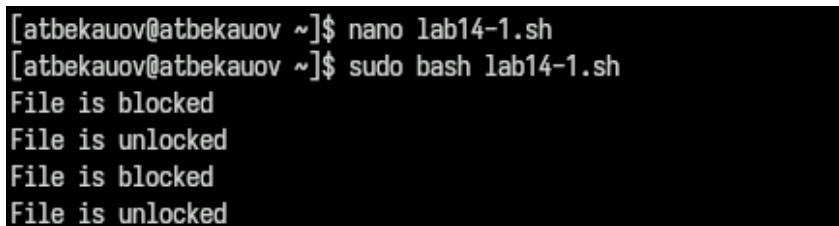
lockfile="/.lock.file"
exec {fn}>$lockfile

while test -f "$lockfile"
do
if flock -n ${fn}
then
    echo "File is blocked"
    sleep 5
    echo "File is unlocked"
    flock -u ${fn}
else
```



```
    echo "file is blocked"
    sleep 5
fi
done
```

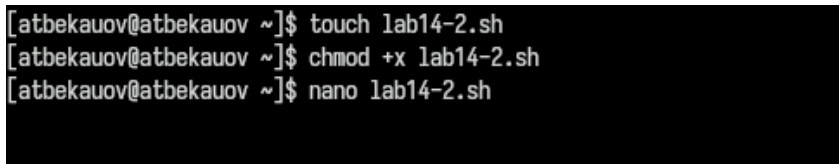
Сохраняю файл и закрываю редактор nano, далее запускаю исполняемый файл с помощью команды bash. Затем проверяю, что выполняет поставленную задачу(рис. 3.2).



```
[atbekauov@atbekauov ~]$ nano lab14-1.sh
[atbekauov@atbekauov ~]$ sudo bash lab14-1.sh
File is blocked
File is unlocked
File is blocked
File is unlocked
```

Рис. 3.2: Программа №1 - Выполнение

Создаю файл lab14-2.sh, меняю права доступа, разрешая его выполнение. (рис. 3.3).



```
[atbekauov@atbekauov ~]$ touch lab14-2.sh
[atbekauov@atbekauov ~]$ chmod +x lab14-2.sh
[atbekauov@atbekauov ~]$ nano lab14-2.sh
```

Рис. 3.3: Программа №2 - создание файлов

Затем открываю в nano файл lab14-2.sh и ввожу следующую программу:

```
#!/bin/bahs

a=$1
if test -f "/usr/share/man/man1/$a.1.gz"
then less /usr/share/man/man1/$a.1.gz
else
```

```
echo "There is no man for $a"  
fi
```

Сохраняю файл, выхожу из nano и запускаю файл через bash. Ввожу в качестве аргумента ls.(рис. 3.4).

```
[atbekauov@atbekauov ~]$ bash lab14-2.sh ls  
[atbekauov@atbekauov ~]$ |
```

Рис. 3.4: Программа №2 - выполнение 1

Вижу, что командный файл открыл мне справку по команде ls.(рис. 3.5).

```
bash lab14-2.sh ls  
ESC[4mL$ESC[24m(1) User Commands  
ESC[4mL$ESC[24m(1)  
ESC[1mNAME$ESC[0m  
ls - list directory contents  
ESC[1mSYNOPSIS$ESC[0m  
ESC[1mls ESC[22m[ESC[4mOPTIONESC[24m]... [ESC[4mFILEESC[24m]...  
ESC[1mDESCRIPTION$ESC[0m  
List information about the FILES (the current directory by default). Sort entries alphabetically if  
ESC[1m--sort ESC[22mis  
specified.
```

Рис. 3.5: Программа №2 - выполнение 2

Создаю файл lab14-3.sh, меняю права доступа, разрешая его выполнение. Открываю файл в nano (рис. 3.6).

```
[atbekauov@atbekauov ~]$ touch lab14-3.sh  
[atbekauov@atbekauov ~]$ chmod +x lab14-3.sh  
[atbekauov@atbekauov ~]$ nano lab14-3.sh |
```

Рис. 3.6: Программа №3 - создание файла

Затем ввожу в файл текст программы:

```
#!/bin/bash
```

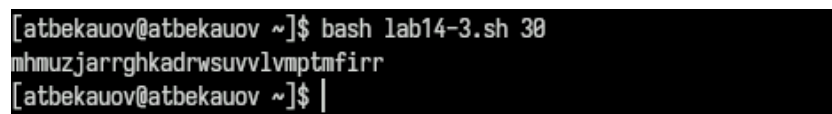
```
a=$1
```

```

for ((i=0; i<$a; i++))
do
    ((char=$RANDOM%26 +1))
    case $char in
        1) echo -n a;; 2) echo -n b;; 3) echo -n c;; 4) echo -n d;; 5) echo -n e;; 6)
        8) echo -n h;; 9) echo -n i;; 10) echo -n j;; 11) echo -n k;; 12) echo -n l;;
        15) echo -n o;; 16) echo -n p;; 17) echo -n q;; 18) echo -n r;; 19) echo -n s
        22) echo -n v;; 23) echo -n w;; 24) echo -n x;; 25) echo -n y;; 26) echo -n z
    esac
done
echo

```

Сохраняю файл, выхожу из nano и запускаю файл через bash с аргументом 30, программа выводит 30 случайных английских букв. (рис. 3.7).



```

[atbekauov@atbekauov ~]$ bash lab14-3.sh 30
mhmuzjarrghkadrwsuvvlvmptmfirr
[atbekauov@atbekauov ~]$ |

```

Рис. 3.7: Программа №3 - выполнение

4 Выводы

В ходе данной лабораторной работы я научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

5 Ответы на онтрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке: `1 while [$1 != "exit"]`

В данной строчке допущены следующие ошибки: не хватает пробелов после первой скобки `[` и перед второй скобкой `]` выражение `$1` необходимо взять в `"`, потому что эта переменная может содержать пробелы Таким образом, правильный вариант должен выглядеть так: `while ["$1" != "exit"]`

2. Как объединить (конкатенация) несколько строк в одну?

Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами: Первый: `VAR1="Hello," VAR2=" World" VAR3="$VAR1$VAR2" echo "$VAR3"` Результат: Hello, World Второй: `VAR1="Hello," VAR1+=" World" echo "$VAR1"` Результат: Hello, World

3. Найдите информацию об утилите `seq`. Какими иными способами можно реализовать её функционал при программировании на `bash`?

Команда `seq` в Linux используется для генерации чисел от ПЕРВОГО до ПОСЛЕДНЕГО шага INCREMENT. Параметры: `seq LAST`: если задан только один аргумент, он создает числа от 1 до LAST с шагом шага, равным 1. Если LAST меньше 1, значение `is` не выдает. `seq FIRST LAST`: когда заданы два аргумента, он генерирует числа от FIRST до LAST с шагом 1, равным 1. Если LAST меньше FIRST, он не выдает никаких выходных данных. `seq FIRST INCREMENT LAST`: когда заданы три аргумента, он генерирует числа от FIRST до LAST на шаге INCREMENT. Если LAST меньше, чем FIRST, он не производит вывод. `seq -f «FORMAT» FIRST`

INCREMENT LAST: эта команда используется для генерации последовательности в форматированном виде. FIRST и INCREMENT являются необязательными. seq -s «STRING» ПЕРВЫЙ ВКЛЮЧЕНО: Эта команда используется для STRING для разделения чисел. По умолчанию это значение равно /n. FIRST и INCREMENT являются необязательными. seq -w FIRST INCREMENT LAST: эта команда используется для выравнивания ширины путем заполнения начальными нулями. FIRST и INCREMENT являются необязательными.

4. Какой результат даст вычисление выражения $\$((10/3))$?

Результатом данного выражения $\$((10/3))$ будет 3, потому что это целочисленное деление без остатка.

5. Укажите кратко основные отличия командной оболочки zsh от bash.

Отличия командной оболочки zsh от bash: В zsh более быстрое автодополнение для cd с помощью Tab В zsh существует калькулятор zcalc, способный выполнять вычисления внутри терминала В zsh поддерживаются числа с плавающей запятой В zsh поддерживаются структуры данных «хэш» В zsh поддерживается раскрытие полного пути на основе неполных данных В zsh поддерживается замена части пути В zsh есть возможность отображать разделенный экран, такой же как разделенный экран vim

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

`for ((a=1; a <= LIMIT; a++))` синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать \$ перед переменными ().

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

Преимущества и недостатки скриптового языка bash:

- Один из самых распространенных и ставится по умолчанию в большинстве дистрибутивах Linux, MacOS
- Удобное перенаправление ввода/вывода
- Большое количество команд для работы с файловыми системами Linux
- Можно писать собственные скрипты, упрощающие работу в Linux Недостатки скриптового языка bash:
- Дополнительные библиотеки других языков позволяют выполнить больше действий
- Bash не является языком общего назначения
- Утилиты, при выполнении скрипта, запускают свои процессы, которые, в свою очередь, отражаются на скорости выполнения этого скрипта
- Скрипты, написанные на bash, нельзя запустить на других операционных системах без дополнительных действий