

Отчет по лабораторной работе №12

Дисциплина: операционные системы

Лобанова Полина Иннокентьевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Контрольные вопросы	10
4	Вывод	12

Список иллюстраций

2.1	Создание файлов.	6
2.2	Первый скрипт.	7
2.3	Результат первого скрипта.	7
2.4	Второй скрипт.	8
2.5	Запуск второго скрипта.	8
2.6	Результат второго скрипта.	8
2.7	Третий скрипт.	9
2.8	Результат третьего скрипта.	9

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.

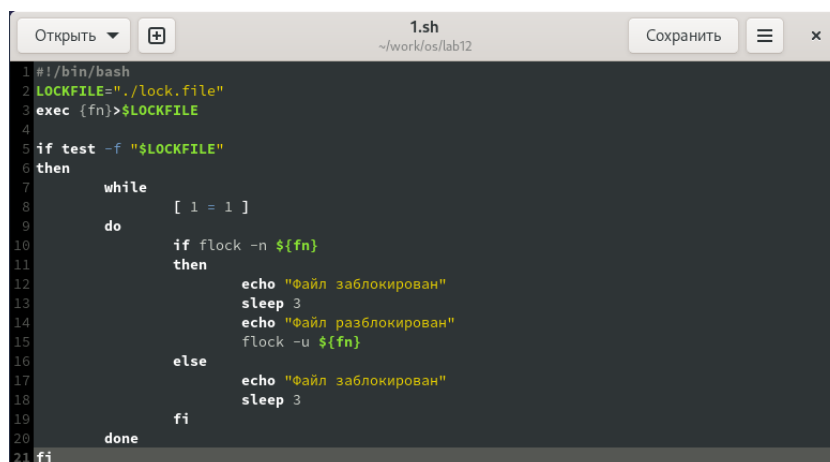
2 Выполнение лабораторной работы

1. Создадим все необходимые для дальнейшей работы файлы.

```
[pilobanova@fedora ~]$ cd ~/work/os/  
[pilobanova@fedora os]$ mkdir lab12  
[pilobanova@fedora os]$ cd lab12  
[pilobanova@fedora lab12]$ touch 1.sh  
[pilobanova@fedora lab12]$ touch 2.sh  
[pilobanova@fedora lab12]$ touch 3.sh
```

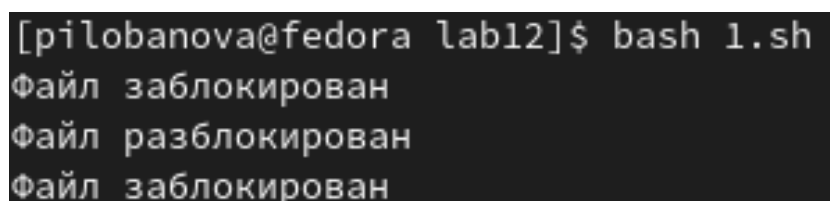
Рис. 2.1: Создание файлов.

2. 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). Запустим 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. Доработаем программу так, чтобы имелась возможность взаимодействия трёх и более процессов.



```
1 #!/bin/bash
2 LOCKFILE="./lock.file"
3 exec {fn}>$LOCKFILE
4
5 if test -f "$LOCKFILE"
6 then
7     while
8         [ 1 = 1 ]
9     do
10         if flock -n ${fn}
11         then
12             echo "Файл заблокирован"
13             sleep 3
14             echo "Файл разблокирован"
15             flock -u ${fn}
16         else
17             echo "Файл заблокирован"
18             sleep 3
19         fi
20     done
21 fi
```

Рис. 2.2: Первый скрипт.



```
[pilobanova@fedora lab12]$ bash 1.sh
Файл заблокирован
Файл разблокирован
Файл заблокирован
```

Рис. 2.3: Результат первого скрипта.

3. Реализуем команду `man` с помощью командного файла. Изучим 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`.

```

1 #!/bin/bash
2 command=""
3
4 while getopts :n: opt
5 do
6     case $opt in
7         n) command="$OPTARG";;
8         esac
9     done
10
11 if test -f "/usr/share/man/man1/$command.1.gz"
12 then less /usr/share/man/man1/$command.1.gz
13
14 else
15     echo "no such command"
16
17 fi

```

Рис. 2.4: Второй скрипт.

```

[pilobanova@fedora lab12]$ bash 2.sh -n kill

```

Рис. 2.5: Запуск второго скрипта.

```

KILL(1)                                User Commands                                KILL(1)

ESC[1mNAMEESC[0m
    kill - terminate a process

ESC[1mSYNOPSISESC[0m
    ESC[1mkill ESC[22mESC[1m-ESC[4mESC[22msignalESC[24mESC[1m-s ESC[4mESC[22msignalESC[24mESC[1m-pESC[22mESC[1m-q ESC[4mESC[22mvalueESC[24mESC[1m-aESC[22mESC[1m--timeout ESC[4mESC[22mmillisecondsESC[0m
    ESC[4msignalESC[24mESC[1m--ESC[22mESC[4mpidESC[24mESC[4mnameESC[24m
    ...

    ESC[1mkill -l ESC[22mESC[4mnumberESC[24m | ESC[1m-LESC[0m

ESC[1mDESCRIPTIONESC[0m
    The command ESC[1mkill ESC[22msends the specified ESC[4msignalESC[24m to the specified processes or process groups.

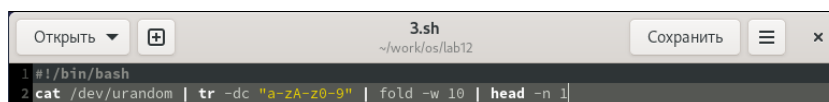
    If no signal is specified, the ESC[1mTERM ESC[22msignal is sent. The default action for this signal is to terminate the process. This signal should be used in preference to the ESC[1mKILL ESC[22msignal (number 9), since a process
/usr/share/man/man1/kill.1.gz

```

Рис. 2.6: Результат второго скрипта.

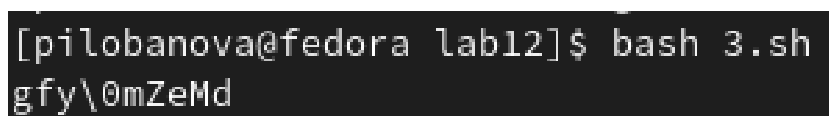
- Используя встроенную переменную \$RANDOM, напишем командный файл, генерирующий случайную последовательность букв латинского алфавита.

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



```
3.sh
~/work/os/lab12
1 #!/bin/bash
2 cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 10 | head -n 1
```

Рис. 2.7: Третий скрипт.



```
[pilobanova@fedora lab12]$ bash 3.sh
gfy\0mZeMd
```

Рис. 2.8: Результат третьего скрипта.

3 Контрольные вопросы

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

\$1. Так же между скобками должны быть пробелы. В противном случае скобки и рядом стоящие символы будут восприниматься как одно целое

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

```
[pilobanova@fedora ~]$ cat file.txt | xargs | sed -e 's/\./.\n/g'
```

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

`seq` - выдает последовательность чисел. Реализовать ее функционал можно командой `for n in {1..5} do done`

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

Ответ: 3

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

`Zsh` очень сильно упрощает работу. Но существуют различия. Например, в `zsh` после `for` обязательно вставлять пробел, нумерация массивов в `zsh` начинается с 1 (что не особо удобно на самом деле). Если вы собираетесь писать скрипт, который легко будет запускать множество разработчиков, то я рекомендую `Bash`. Если скрипты вам не нужны - `Zsh` (более простая работа с файлами, например)

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

Верен

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

`Bash` позволяет очень легко работать с файловой системой без лишних конструкций (в отличие от обычного языка программирования). Но относительно обычных языков программирования `bash` очень сжат. Тот же `C` имеет гораздо более широкие возможности для разработчика.

4 Вывод

Я научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.