

Лабораторная работа №12

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

Панченко Денис Дмитриевич

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Вывод	10
4	Контрольные вопросы	11

Список иллюстраций

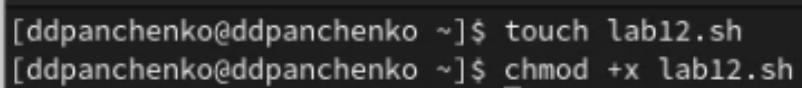
2.1	Создание файла	5
2.2	Программа	6
2.3	Запуск и вывод	7
2.4	Создание файла	7
2.5	Программа	7
2.6	Запуск	7
2.7	Вывод	8
2.8	Создание файла	8
2.9	Программа	8
2.10	Запуск и вывод	9

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

Напишем командный файл, реализующий упрощённый механизм семафоров (рис. 2.1 - 2.3).

A terminal window with a dark background and light-colored text. It shows two commands being executed in sequence. The first command is 'touch lab12.sh' and the second is 'chmod +x lab12.sh'. Both commands are preceded by the prompt '[ddpanchenko@ddpanchenko ~]\$'.

```
[ddpanchenko@ddpanchenko ~]$ touch lab12.sh  
[ddpanchenko@ddpanchenko ~]$ chmod +x lab12.sh
```

Рис. 2.1: Создание файла

```
#!/bin/bash
lockfile="./lockfile"
exec {fn}>$lockfile
echo "lock"
until flock -n ${fn}
do
    echo "not lock"
    sleep 1
    flock -n ${fn}
done
for ((i=0;i<=5;i++))
do
    echo "work"
    sleep 1
done
```

Рис. 2.2: Программа

```
[ddpanchenko@ddpanchenko ~]$ ./lab12.sh
lock
work
work
work
work
work
work
work
[ddpanchenko@ddpanchenko ~]$
```

Рис. 2.3: Запуск и вывод

Реализуем команду `man` с помощью командного файла (рис. 2.4 - 2.7).

```
[ddpanchenko@ddpanchenko ~]$ touch lab12-2.sh
[ddpanchenko@ddpanchenko ~]$ chmod +x lab12-2.sh
```

Рис. 2.4: Создание файла

```
#!/bin/bash
cd /usr/share/man/man1
less $1*
```

Рис. 2.5: Программа

```
[ddpanchenko@ddpanchenko ~]$ ./lab12-2 less
```

Рис. 2.6: Запуск

```
LESS(1)                                General Commands Manual                                LESS(1)

ESC[1mNAMEESC[0m
less - opposite of more

ESC[1mSYNOPSISESC[0m
ESC[1mless -?ESC[0m
ESC[1mless --helpESC[0m
ESC[1mless -VESC[0m
ESC[1mless --versionESC[0m
ESC[1mless [-[+]aAbcCdEfFgGiIjKlMnNqQrRsSuUVvWx~]ESC[0m
ESC[1m[-b ESC[4mESC[22mspaceESC[24mESC[1m] [-h ESC[4mESC[22mlinesESC[24mESC[1m] [-j ESC[4mESC[22mlineESC[24mESC[1m] [-k ESC[4mESC[22mkeyfileESC[24mESC[1m]ESC[0m
ESC[1m[-{oO} ESC[4mESC[22mlogfileESC[24mESC[1m] [-p ESC[4mESC[22mpatESC[24mESC[1m] [-P ESC[4mESC[22mpromptESC[24mESC[1m] [-t ESC[4mESC[22mtagESC[24mESC[1m]ESC[0m
ESC[1m[-T ESC[4mESC[22mtagsfileESC[24mESC[1m] [-x ESC[4mESC[22mtabESC[24mESC[1m,...] [-y ESC[4mESC[22mlinesESC[24mESC[1m] [-z ESC[4mESC[22mlinesESC[24mESC[1m]ESC[0m
ESC[1m[-# ESC[4mESC[22mshiftESC[24mESC[1m] [+][+]ESC[4mESC[22mcmdESC[24mESC[1m] [--] [ESC[4mESC[22mfilenameESC[24mESC[1m]...ESC[0m
(See the OPTIONS section for alternate option syntax with long option
```

Рис. 2.7: Вывод

Используя встроенную переменную \$RANDOM, напомним командный файл, генерирующий случайную последовательность букв латинского алфавита (рис. 2.8 - 2.10).

```
[ddpanchenko@ddpanchenko ~]$ touch lab12-3.sh
[ddpanchenko@ddpanchenko ~]$ chmod +x lab12-3.sh
```

Рис. 2.8: Создание файла

```
#!/bin/bash
M=10
c=1
d=1
echo
echo "10 random words:"
while (($c!=$((M+1))))
do
    echo $(for((i=1;i<=10;i++)); do printf '%s' "${RANDOM:0:1}"; done) | tr '0-9' '[a-z]'
    echo $d
    ((c+=1))
    ((d+=1))
done
```

Рис. 2.9: Программа


```
[ddpanchenko@ddpanchenko ~]$ ./lab12-3.sh  
  
10 random words:  
ccdhdhbcbb  
1  
cbeccbcccc  
2  
dccgjbdcbb  
3  
ccccjccbbc  
4  
bdecccbcc  
5  
bchbbbcbb  
6  
dcbccbccbi  
7  
cccbgdbccb  
8  
cdcbibcbbf  
9  
cebcjfbccb  
10
```

Рис. 2.10: Запуск и вывод

3 Вывод

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

4 Контрольные вопросы

1. Синтаксическая ошибка в данной строке заключается в том, что в `bash` используется двойное квадратное скобочное выражение для условных операторов, а не круглые скобки. Правильно будет написать так: `while [$1 != "exit"]`.
2. Несколько строк можно объединить с помощью оператора конкатенации строк `-`, например: `str1="Hello"; str2="world"; result=$str1$str2; echo $result`. В данном примере результатом будет строка `"Hello-world"`.
3. Утилита `seq` используется для генерации последовательностей чисел. Пример использования: `seq 1 10`. Эта команда выведет последовательность чисел от 1 до 10. Альтернативными способами генерации последовательностей являются использование циклов `for` или `while` с использованием оператора счетчика, например: `for i in {1..10}; do echo $i; done`.
4. Выражение `$((10/3))` даст результат 3, так как в `bash` при делении целых чисел используется целочисленное деление без округления.
5. Основные отличия командной оболочки `zsh` от `bash`:
 - `zsh` имеет более продвинутый и удобный интерфейс командной строки;
 - `zsh` имеет множество дополнительных функций и возможностей, таких как автодополнение путей и команд, подсветка синтаксиса и др.;
 - `zsh` имеет более продвинутую систему настройки и использования алиасов и функций.

6. Данный синтаксис верен. Он используется для цикла `for` с заданным начальным значением, конечным значением и шагом счетчика.
7. Bash - это язык скриптовой обработки командной строки, который удобен для быстрого и простого выполнения повседневных задач в системе Unix/Linux. Он имеет простой синтаксис и понятную логику, что делает его легко доступным для новичков. Однако, в отличие от других языков программирования, bash не имеет широких возможностей для создания сложных алгоритмов и программ. Также, bash может быть медленнее в выполнении сложных задач по сравнению с более производительными языками программирования, такими как Python или Java.