

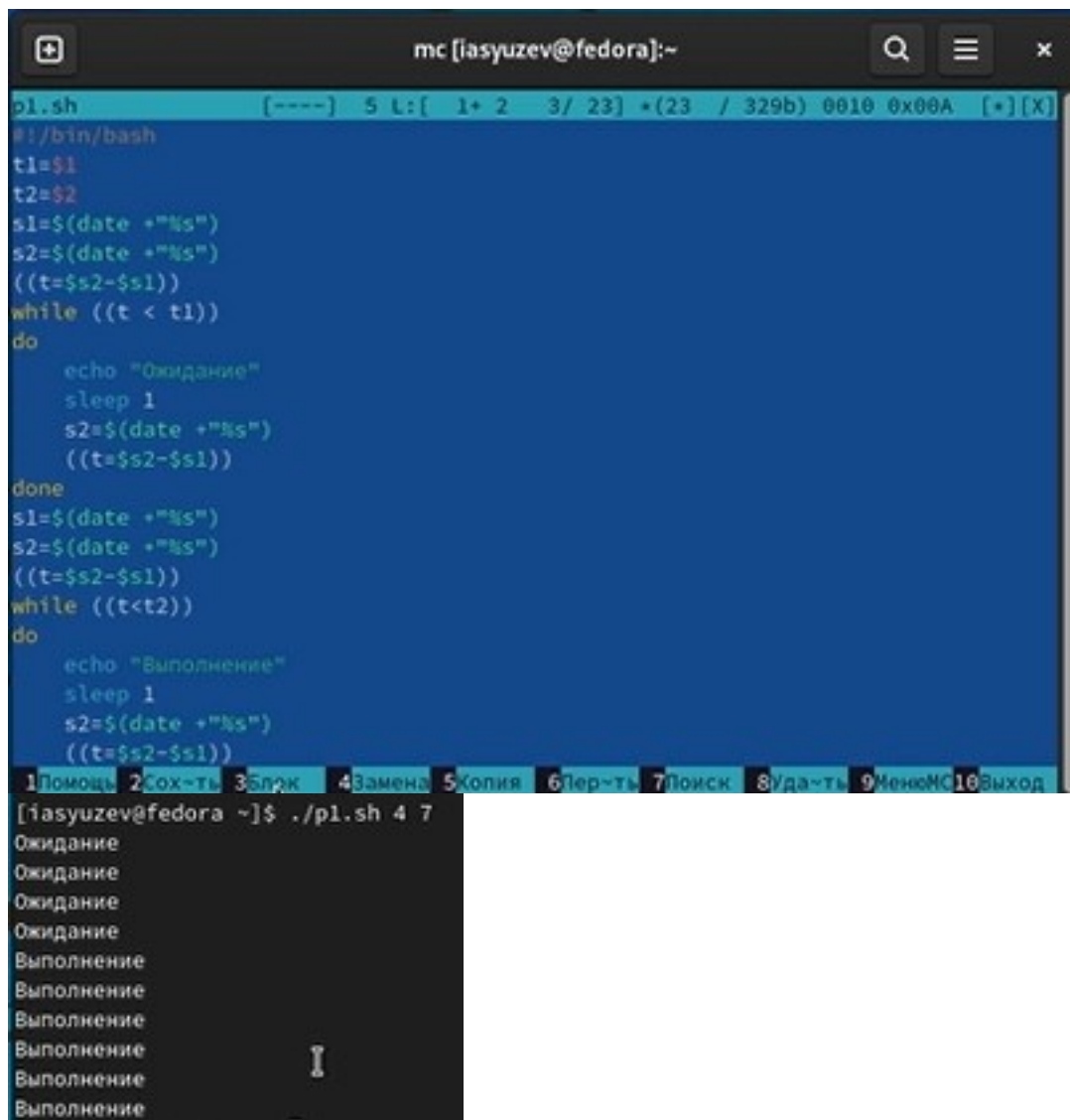
Отчет по лабораторной работе №12

Цель работы

Изучить основы программирования в оболочке ОС UNIX, научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов

Ход выполнения работы

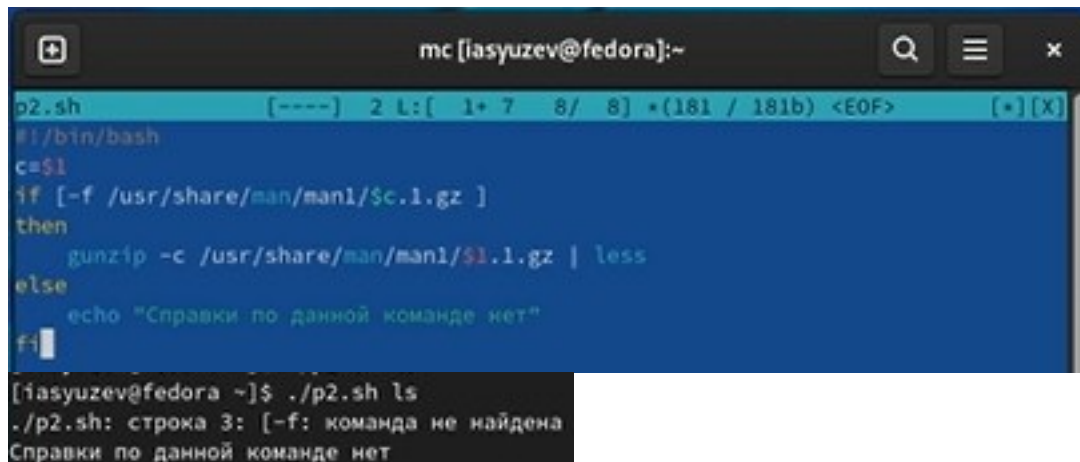
1. Написали командный файл, реализующий упрощенный механизм семафоров, и протестировали его



```
mc [iasyuzev@fedora]:~
p1.sh [-----] 5 L: [ 1+ 2 3/ 23] *(23 / 329b) 0010 0x00A [*][X]
#!/bin/bash
t1=$1
t2=$2
s1=$(date +%s)
s2=$(date +%s)
((t=$s2-$s1))
while ((t < t1))
do
    echo "Ожидание"
    sleep 1
    s2=$(date +%s)
    ((t=$s2-$s1))
done
s1=$(date +%s)
s2=$(date +%s)
((t=$s2-$s1))
while ((t < t2))
do
    echo "Выполнение"
    sleep 1
    s2=$(date +%s)
    ((t=$s2-$s1))
done

1Помощь 2Сох-ть 3Блэк 4Замена 5Копия 6Пер-ть 7Поиск 8/да-ть 9МенюMC10Выход
[iasyuzev@fedora ~]$ ./p1.sh 4 7
Ожидание
Ожидание
Ожидание
Ожидание
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
```

2. Реализовали команду `tap` с помощью командного файла



```
mc [iasyuzev@fedora]:~
p2.sh [----] 2 L: [ 1+ 7 8/ 8] *(181 / 181b) <EOF> [*][X]
#!/bin/bash
c=$1
if [-f /usr/share/man/man1/$c.1.gz ]
then
    gunzip -c /usr/share/man/man1/$1.1.gz | less
else
    echo "Справки по данной команде нет"
fi
[iasyuzev@fedora ~]$ ./p2.sh ls
./p2.sh: строка 3: [-f: команда не найдена
Справки по данной команде нет
```

3. Используя встроенную переменную `RANDOM` написали командный файл, генерирующий случайную последовательность букв латинского алфавита



```
mc [iasyuzev@fedora]:~
p3.sh [----] 27 L: [ 1+ 4 5/ 10] *(71 / 516b) 0010 0x00A [*][X]
#!/bin/bash
k=$1
for (( i=0; i<$k; i++))
do
    (( char=$RANDOM%26+1 ))
    case $char in
    <----->1) echo -n a;; 2) echo -n b;; 3) echo -n c;; 4) echo -n d;; 5) echo -n e
    esac
done
echo
[iasyuzev@fedora ~]$ ./p3.sh 10
fsixootcyh
```

Выводы

В ходе выполнения лабораторной работы я изучил основы программирования в оболочке ОС UNIX и научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов

Ответы на контрольные вопросы

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.

Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами:

Первый: VAR1=“Hello,

“VAR2=” World”

VAR3=“V A R 1VAR2”

echo “\$VAR3”

Результат: Hello, World

Второй: VAR1=“Hello,”

VAR1+=” World”

echo “\$VAR1”

Результат: Hello, World

3.

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.

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

5.

Отличия командной оболочки `zsh` от `bash`:

- В `zsh` более быстрое автодополнение для `cd` 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.

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

7.

Преимущества 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.