

Отчёт по лабораторной работе №12

Дисциплина: Операционные системы

Подъярова Ксения Витальевна

Содержание

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

Список иллюстраций

2.1	Создание файла	6
2.2	Скрипт	7
2.3	Проверка работы	8
2.4	Скрипт	8
2.5	Скрипт	9
2.6	Проверка работы	9
2.7	Содержимое каталога /usr/share/man/man1	9
2.8	Создание файла	10
2.9	Скрипт	10
2.10	Проверка работы	10
2.11	Создание файла	11
2.12	Скрипт	11
2.13	Проверка работы	11

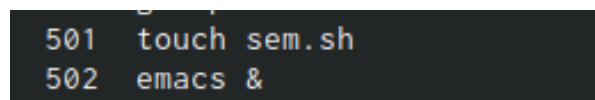
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

- 1) Написала командный файл, реализующий упрощённый механизм semaфоров. 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). Для данной задачи я создала файл `sem.sh` (рис. 2.1) и написала соответствующий скрипт (рис. 2.2).



```
501 touch sem.sh
502 emacs &
```

Рис. 2.1: Создание файла

```
#!/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
```

Рис. 2.2: Скрипт

- 2) Далее я проверила работу написанного скрипта (./sem.sh 4 7), предварительно предоставив файлу право на исполнение (chmod +x sem.sh). (рис. 2.3). Скрипт работает корректно

```

aorepina@dk3n52 ~ $ chmod +x sem.sh
aorepina@dk3n52 ~ $ ./sem.sh 3 5
Ожидание
Ожидание
Ожидание
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
aorepina@dk3n52 ~ $

```

Рис. 2.3: Проверка работы

- 3) После этого я изменила скрипт так, чтобы его можно было выполнять в нескольких терминалах и прверила его работу (например команда `./sem.sh 2 3 Ожидание > /dev/pts/1 &`) (рис. 2.4) (рис. 2.5). После проверила работу скрипта и увидела, что мне было отказано в доступе (рис. 2.6)

```

#!/bin/bash
function ozhidanie
{
    s1=$(date +%s")
    s2=$(date +%s")
    ((t=s2-s1))
    while ((t<t1))
    do
        echo "Ожидание"
        sleep 1
        s2=$(date +%s")
        ((t=s2-s1))
    done
}
function vipolnenie
{
    s1=$(date +%s")
    s2=$(date +%s")
    ((t=s2-s1))
    while ((t<t2))
    do
        echo "Выполнение"
        sleep 1
        s2=$(date +%s")
        ((t=s2-s1))
    done
}

```

Рис. 2.4: Скрипт

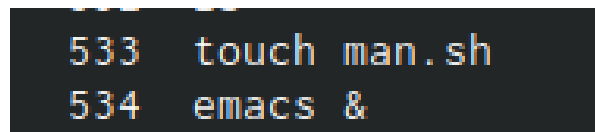
Рис. 2.5: Скрипт

Рис. 2.6: Проверка работы

[illegible]

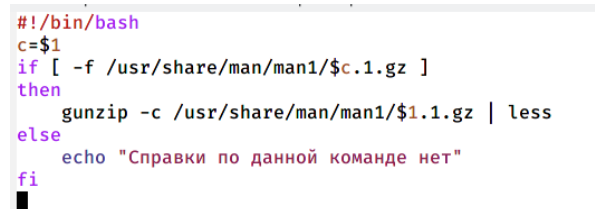
9

- 2) Для данной задачи я создала файл `man.sh` (рис. 2.8) и написала соответствующий скрипт (рис. 2.9)



```
533 touch man.sh
534 emacs &
```

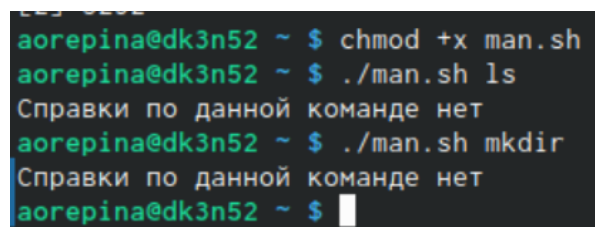
Рис. 2.8: Создание файла



```
#!/bin/bash
c=$1
if [ -f /usr/share/man/man1/${c}.1.gz ]
then
gunzip -c /usr/share/man/man1/${c}.1.gz | less
else
echo "Справки по данной команде нет"
fi
```

Рис. 2.9: Скрипт

- 3) Далее я проверила работу написанного скрипта (`./man.sh ls` и `./man.sh mkdir`), предварительно добавив право на исполнение файла (`chmod +x man.sh`) (рис. 2.10). Скрипт работает корректно.



```
aorepina@dk3n52 ~ $ chmod +x man.sh
aorepina@dk3n52 ~ $ ./man.sh ls
Справки по данной команде нет
aorepina@dk3n52 ~ $ ./man.sh mkdir
Справки по данной команде нет
aorepina@dk3n52 ~ $
```

Рис. 2.10: Проверка работы

3. 1) Используя встроенную переменную `$RANDOM`, написала командный файл, генерирующий случайную последовательность букв латинского алфавита. Учла, что `$RANDOM` выдаёт псевдослучайные числа в диапазоне от 0 до 32767. Для данной задачи я создала файл `mmm.sh` (рис. 2.11) и написала соответствующий скрипт (рис. 2.12)

3 Выводы

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

4 Ответы на контрольные вопросы

1. 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="VAR2" 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 LASTменьше, 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.