Отчет по лабораторной работе №12

дисциплина: Операционные системы

Старков Никита Алексеевич

Содержание

Список иллюстраций

1 Цель работы

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)Напишем командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени t2<>t1, 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.

Создаем файл sem.sh и открываем emacs.

```
nastarkov@dk6n55 ~ $ touch sem.sh
nastarkov@dk6n55 ~ $ emacs &
```

Рис. 2.1: Создание файла

Пишем скрипт, удовлетворяющий условиям задачи

```
#!/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 №1

Проверяем работу написанного скрипта, предварительно открыв доступ на исполнение файла

```
nastarkov@dk6n55 ~ $ ./sem1.sh 4 7
Ожидание
Ожидание
Ожидание
Ожидание
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
```

Рис. 2.3: Проверка работы скрипта

Дорабатываем программу в соответствии с условием задачи

```
#!/bin/bash
function ogidania
{
    s1=$(date +"%s")
    s2=$(date +"%s")
    ((t=$s2-$s1))
    while ((t < t2))
    do
        echo "Выполнение"
        sleep 1
        s2=$(date +"%s")
        ((t=$s2-$s1))
    done
}
t1=$1
t2=$2
command=$3
while true
do
    if [ "$command" == "Выход" ]
    then
        echo "Выход"
        exit 0
    fi
    if [ "$command" == "Выполнение" ]
    then ogidanie
    fi
    if [ "$command" == "Выполнение" ]
    then vipolnenie
    fi
    echo "Следующее действие: "
    read command
done
```

Рис. 2.4: Доработанный скрипт №1

Проверяем работу

```
nastarkov@dk6n55 ~ $ chmod +x sem.sh
nastarkov@dk6n55 ~ $ ./sem.sh 2 3 Ожидание > /dev/pts/1 &
[2] 18934
nastarkov@dk6n55 ~ $ bash: /dev/pts/1: Отказано в доступе

[2]+  Выход 1             ./sem.sh 2 3 Ожидание > /dev/pts/1
nastarkov@dk6n55 ~ $ ./sem.sh 2 3 Ожидание > /dev/pts/2 &
[2] 19315
nastarkov@dk6n55 ~ $ bash: /dev/pts/2: Отказано в доступе

[2]+  Выход 1             ./sem.sh 2 3 Ожидание > /dev/pts/2
nastarkov@dk6n55 ~ $ ./sem.sh 2 5 Ожидание > /dev/pts/2 &
[2] 19344
nastarkov@dk6n55 ~ $ bash: /dev/pts/2: Отказано в доступе

[2]+  Выход 1             ./sem.sh 2 5 Ожидание > /dev/pts/2
```

Рис. 2.5: Проверка работы скрипта

2)Реализуем команду man с помощью командного файла. Изучаем 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.

Заходим в каталог /usr/share/man/man1 и просматриваем содержимое

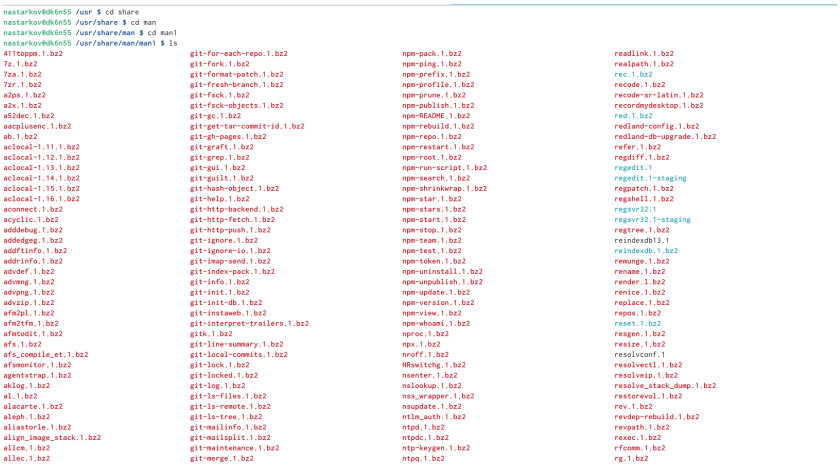


Рис. 2.6: Просмот содержимого каталога /usr/share/man/man1

Создаем файл man.sh и пишем скрипт, удовлетворяющий условиям задачи

```
#!/bin/bash
c=$1
if [ -f /usr/share/man/man1/$c1.1.gz ]
then
    gunzip -c /urs/share/man/man1/$1.1.gz | less
else
    echo "Справки по данной команде нет"
fi
```

Рис. 2.7: Скрипт №2

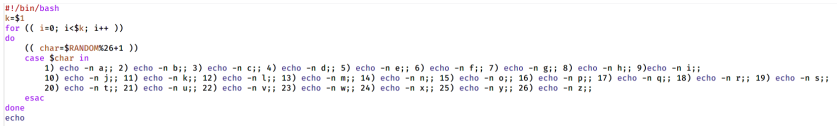
Проверяем работу скрипта, предварительно открыв доступ на исполнение файла

```
nastarkov@dk6n55 ~ $ chmod +x man.sh
nastarkov@dk6n55 ~ $ ./man.sh ls
Справки по данной команде нет
nastarkov@dk6n55 ~ $ ./man.sh mkdir
Справки по данной команде нет
```

Рис. 2.8: Проверка работы скрипта

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

Создаем файл random.sh и пишем скрипт, удовлетворяющий условиям задачи



```
#!/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;; 6) echo -n f;; 7) echo -n g;; 8) echo -n h;; 9)echo -n i;;
        10) echo -n j;; 11) echo -n k;; 12) echo -n l;; 13) echo -n m;; 14) echo -n n;; 15) echo -n o;; 16) echo -n p;; 17) echo -n q;; 18) echo -n r;; 19) echo -n s;;
        20) echo -n t;; 21) echo -n u;; 22) echo -n v;; 23) echo -n w;; 24) echo -n x;; 25) echo -n y;; 26) echo -n z;;
    esac
done
echo
```

Рис. 2.9: Скрипт №3

Далее проверяем работу написанного скрипта, предварительно открыв право на исполнение.

```
nastarkov@dk6n55 ~ $ ./random.sh 10
wwakqcofic
nastarkov@dk6n55 ~ $ ./random.sh 5
wyylm
```

Рис. 2.10: Проверка работы скрипта

3 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 про

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=“$VAR1VAR2”
echo “$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

seq FIRST LAST: когда заданы два аргумента, он генерирует числа от FIRST до LAST

seq FIRST INCREMENT LAST: когда заданы три аргумента, он генерирует числа от FIRS

seq -f «FORMAT» FIRST INCREMENT LAST: эта команда используется для генерации посл

seq -s «STRING» ПЕРВЫЙ ВКЛЮЧЕНО: Эта команда используется для STRING для разделен

seq -w FIRST INCREMENT LAST:эта команда используется для выравнивания ширины путе
```

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

5). Отличия командной оболочки zshот bash:

```
В zsh более быстрое автодополнение для cdc помощью Tab

В zsh существует калькулятор zcalc, способный выполнять вычисления внутри термина

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 поддерживаетсязаменачастипути
```

В zsh есть возможность отображать разделенный экран, такой же как разделенный экр

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 умолчаниюв большинстве дистрибутивах

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 языков общего назначения

Утилиты, при выполнении скрипта, запускают свои процессы, которые, в свою очередь

Скрипты, написанные на bash, нельзя запустить на других операционных системах без

4 Вывод

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