Отчёт по лабораторной работе №12

Операционные системы

Сячинова Ксения Ивановна

Содержание

Список иллюстраций

1 Цель работы

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

2 Задание

1. Написать 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.

2. Реализовать команду man с помощью командного файла. Изучите 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.

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

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

3 Выполнение лабораторной работы

1. Создадим рабочее пространство для скриптов данной лабораторной работы. также создадим файл, в который запишем наш скрипт (рис. 3.1).

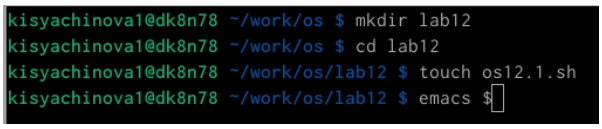

Рис. 3.1: Создание папки

Напишем скрипт для выполнения здадния.(рис. 3.2).

```
#!/bin/bash
t1=$1
t2=$2
s1=$(date +"%s")
s2=$(date +"%s")
((t=$s2-$s1))
while ((t < t1))
do
    echo "Ожидайте"
    sleep 1
    s2=$(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. 3.2: Скрипт

Даём право на выолнение и проверяем работу. Скрип работает корректно.(рис. 3.3).

Рис. 3.3: Проверка

Далее изменим скрипт там, чтобы можно было его выполнять в нескольких терминалах.При проверке мне было отказано в доступе(рис. 3.4).(рис. 3.5).

```
done
}
t1=$1
t2=$2
command=$3
while true
do
    if [ "$command" = "Выход" ]
    then
        ehco "Выход"
        exit 0
    fi
    if [ "$command" = "Ожидание" ]
    then ogidanie
    fi
    if [ "$command" = "Выполнение" ]
    then vipolnenie
    fi
    echo "Следующее действие"
    read command
done
}
```

Рис. 3.4: Скрипт

3.1

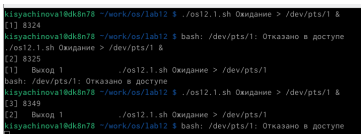

Рис. 3.5: Проверка

2. Выполним второе задание. Для начала изучим содержимое файла “/usr/share/man/man1”(рис. 3.6).

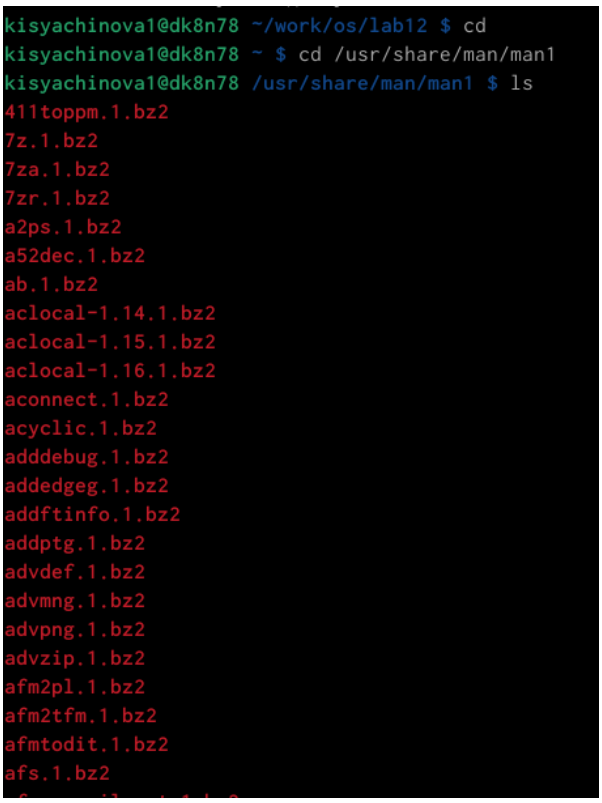

Рис. 3.6: Сожержимое файла

Создадим файла для скрипта и напишем его. Далее добавим право на выполнение и проверим его. (рис. 3.7).(рис. 3.8).

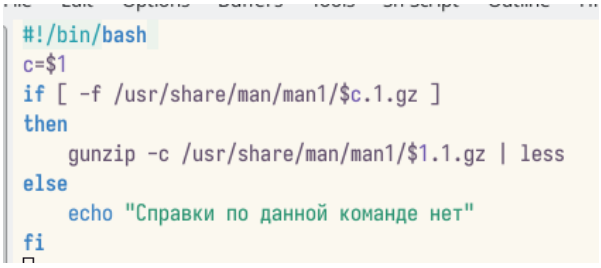

Рис. 3.7: Скрипт

```
kisyachinova1@dk8n78 ~/work/os/lab12 $ chmod +x os12.2.sh
kisyachinova1@dk8n78 ~/work/os/lab12 $ ./os12.2.sh ls
Справки по данной команде нет
kisyachinova1@dk8n78 ~/work/os/lab12 $ ./os12.2.sh mkdir
Справки по данной команде нет
```

Рис. 3.8: Проверка

3. Напишем скрипт для выполнение третьего задания. (рис. 3.9).(рис. 3.10).

```
#!/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. 3.9: Скрипт

```
kisyachinova1@dk8n78 ~/work/os/lab12 $ ./os12.3.sh 5
owqea
kisyachinova1@dk8n78 ~/work/os/lab12 $ ./os12.3.sh 10
afwazyktea
```

Рис. 3.10: Проверка

4 Выводы

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

5 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=“⊠ ⊠⊠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 zshot bash:

- В zsh более быстрое автодополнение для cdc 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 поддерживаетсязаменачастипути
- 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 умолчаниюв 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 языков общего назначения
- Утилиты, при выполнении скрипта, запускают свои процессы, которые, в свою очередь, отражаются на быстроте выполнения этого скрипта
- Скрипты, написанные на bash, нельзя запустить на других операционных системах без дополнительных действий