Отчёт по лабораторной работе №12

Дисциплина: Операционные системы

Горпинич Елена Михайловна

Содержание

Цель работы

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

1. Ознакомиться с теоретическим материалом.
2. Сделать отчёт по лабораторной работе №12 в формате Markdown.
3. Изучить основы программирования в оболочке ОС UNIX/Linux.
4. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

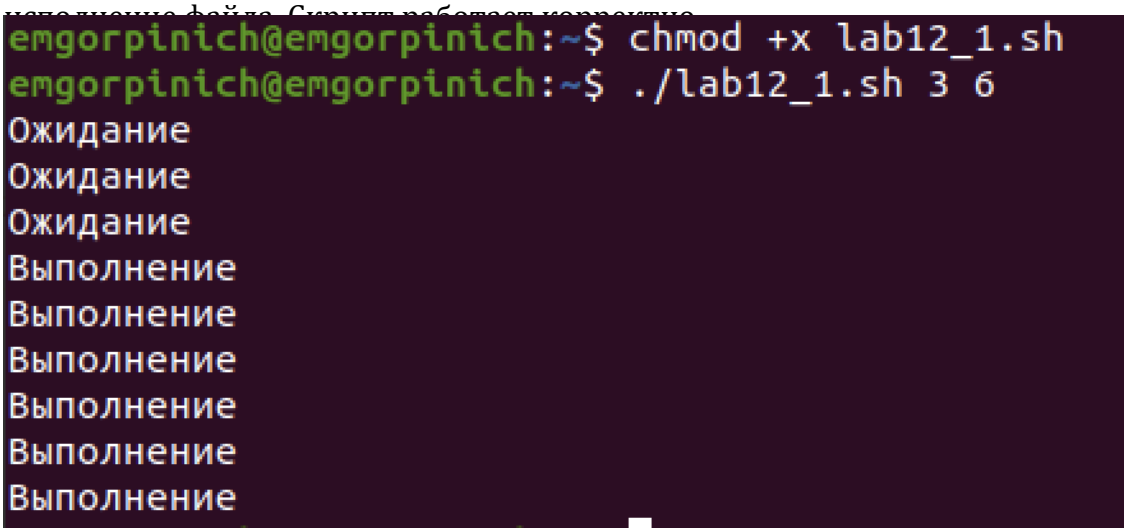
Выполнение лабораторной работы

1) Написала 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). Для данной задачи я создала файл и написала соответствующий скрипт.

```
emgorpinich@emgorpinich:~$ touch lab12_1.sh
emgorpinich@emgorpinich:~$ emacs &
[1] 2749
```

```
#!/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
```

Далее я проверила работу написанного скрипта, предварительно добавив право на исполнение файла. Скрипт работает корректно.



После этого я изменила скрипт так, чтобы его можно было выполнять в нескольких терминалах и проверила его работу

```
#!/bin/bash
function ogidania
{
    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
}
```

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

```
emgorpinich@emgorpinich:~$ ./l2.sh 2 3 Ожидание > /dev/pts/1 &
[6] 4374
emgorpinich@emgorpinich:~$ Ожидание
Ожидание
Следуещее действие:
./l2.sh 2 3 Выполнение > /dev/pts/1 &
[7] 4381

[6]+  Stopped                 ./l2.sh 2 3 Ожидание > /dev/pts/1
emgorpinich@emgorpinich:~$ Выполнение
Выполнение
Выполнение
Следуещее действие:


[7]+  Stopped                 ./l2.sh 2 3 Выполнение > /dev/pts/1
```

2) Реализовала команду man с помощью командного файла. Изучила 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

```
emgorpinich@emgorpinich:~$ cd /usr/share/man/man1
emgorpinich@emgorpinich:/usr/share/man/man1$ ls
'[.1.gz'
 7z.1.gz
 7za.1.gz
 7zr.1.gz
 aa-enabled.1.gz
 aa-exec.1.gz
 aarch64-linux-gnu-addr2line.1.gz
 aarch64-linux-gnu-ar.1.gz
 aarch64-linux-gnu-as.1.gz
 aarch64-linux-gnu-c++filt.1.gz
 aarch64-linux-gnu-cpp.1.gz
 aarch64-linux-gnu-cpp-9.1.gz
 aarch64-linux-gnu-dwp.1.gz
 aarch64-linux-gnu-elfedit.1.gz
 aarch64-linux-gnu-gcc.1.gz
 aarch64-linux-gnu-gcc-9.1.gz
 aarch64-linux-gnu-gcc-ar.1.gz
 aarch64-linux-gnu-gcc-ar-9.1.gz
 aarch64-linux-gnu-gcc-nm.1.gz
 aarch64-linux-gnu-gcc-nm-9.1.gz
```

Для данной задачи я создала файл и написала соответствующий скрипт

```
emgorpinich@emgorpinich:~$ touch man.sh
emgorpinich@emgorpinich:~$ emacs &
[1] 3056
```

```
#!/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
```

Далее я проверила работу написанного скрипта , предварительно добавив право на исполнение файла. Скрипт работает корректно.

```
emgorpinich@emgorpinich:~$ chmod +x man.sh
emgorpinich@emgorpinich:~$ ./man.sh ls
emgorpinich@emgorpinich:~$ ./man.sh zip
emgorpinich@emgorpinich:~$ ./man.sh cd
Справки по данной команде нет
```

```
.\" ============================================================================
.\" Copyright (c) 1990-2008 Info-ZIP.  All rights reserved.
.\"
.\" See the accompanying file LICENSE, version 2007-Mar-4 or later
.\" (the contents of which are also included in zip.h) for terms of use.
.\" If, for some reason, all these files are missing, the Info-ZIP license
.\" also may be found at:  ftp://ftp.info-zip.org/pub/infozip/license.html
.\" ============================================================================
.\"
.\" zip.1 by Mark Adler, Jean-loup Gailly and  R. P. C. Rodgers
.\" updated by E. Gordon for Zip 3.0 (8 May 2005, 24 December 2006,
.\" 4 February 2007, 27 May 2007, 4 June 2007 by EG; 12 June 2007 by CS;
.\" 30 August 2007, 27 April 2008, 25 May 2008, 27 May 2008 by EG,
.\" 7 June 2008 by SMS and EG; 12 June 2008 by EG)
.\"
.TH ZIP 1 "16 June 2008 (v3.0)" Info-ZIP
.SH NAME
zip \- package and compress (archive) files
.SH SYNOPSIS
.B zip
.RB [\- aABcdDeEfFghjklLmoqrRSTuvVwXyz!@$ ]
[\-\-longoption ...]
.RB [\- b " path]"
```

3) Используя встроенную переменную $RANDOM, написала командный файл, генерирующий случайную последовательность букв латинского алфавита. Для данной задачи я создала файл: random.sh и написала соответствующий скрипт

```
emgorpinich@emgorpinich:~$ touch random.sh
emgorpinich@emgorpinich:~$ emacs &
[2] 3115
```

```
#!/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) e
cho -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
```

Далее я проверила работу написанного скрипта, предварительно добавив право на исполнение файла. Скрипт работает корректно.

```
emgorpinich@emgorpinich:~$ chmod +x random.sh
emgorpinich@emgorpinich:~$ ./random.sh 4
jety
emgorpinich@emgorpinich:~$ ./random.sh 29
xeeuturunzmzkwtlwcouwrlmofvea
```

15

Вывод

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

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=“$VAR1$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 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 выравнива- ния 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 cdc помощью Tab • В zsh существует калькулятор zcalc,способныйвыполнятьвычислениявнут- ри терминала • В zsh поддерживаются числа с плавающей запятой • В zsh поддерживаются структуры данных «хэш» •В zsh поддерживается раскрытие полного пути на основе не полных данных • В zsh поддерживается замена части пути •В zsh есть возможность отображать разделенный экран,такой же какразделенный 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,привыполнениискрипта,запускаютсвоипроцессы,которые,в свою очередь, отражаются на быстроте выполнения этого скрипта • Скрипты,написанныенаbash,нельзязапуститьнадругихоперационных системах без дополнительных действий.