РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Фундаментальная информатика и информационные технологии

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 13

дисциплина: Операционные системы

Студент: Кузнецов Юрий

Группа: НФИбд-01-20

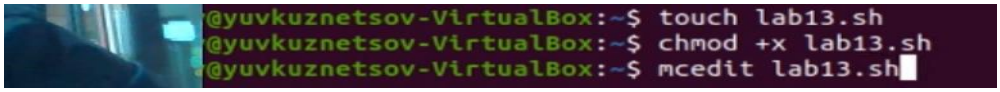
МОСКВА

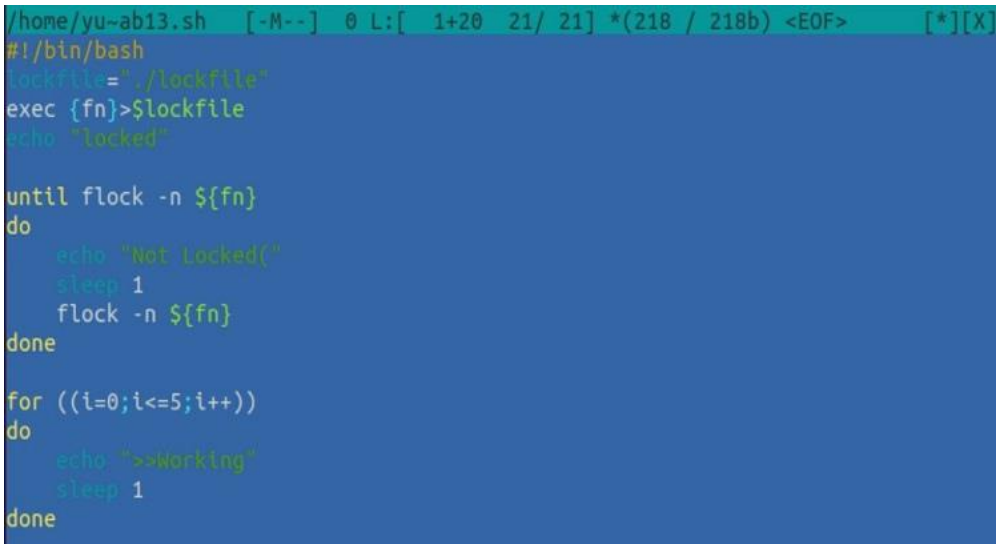
2021 г.

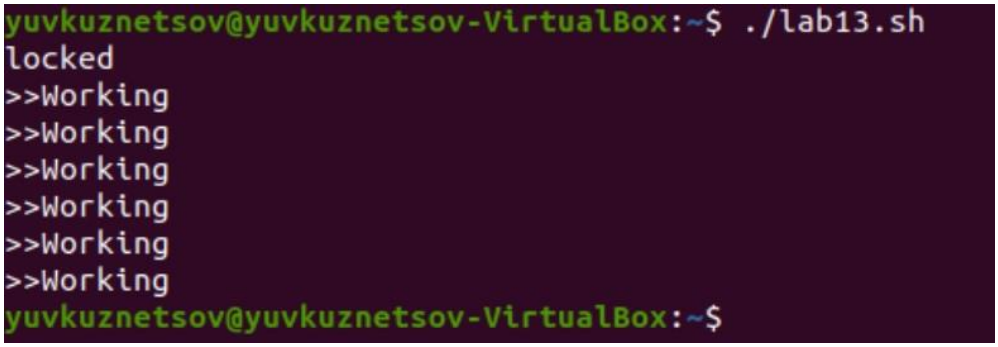
Цель работы: изучить основы программирования в оболочке **ОС UNIX**, научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Ход работы:

1.Написал 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). Запустил 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. Реализовал команду **man** с помощью командного файла. Изучил 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.

```
yuvkuznetsov@yuvkuznetsov-VirtualBox:~$ touch lab13_1.sh
yuvkuznetsov@yuvkuznetsov-VirtualBox:~$ chmod +x lab13.sh
yuvkuznetsov@yuvkuznetsov-VirtualBox:~$
```

```
/home/yu~13_1.sh   [----]  8 L:[  1+ 2   3/  3] *(43  /  43b) <EOF>       [*][X]
#!/bin/bash
cd /usr/share/man/man1
less $1*
```

```
yuvkuznetsov@yuvkuznetsov-VirtualBox:~$ ./lab13_1.sh less
```

```
.TH LESS 1 "Version 551: 11 Jun 2019"
.SH NAME
less \- opposite of more
.SH SYNOPSIS
.B "less \-?"
.br
.B "less \-\-help"
.br
.B "less \-V"
.br
.B "less \-\-version"
.br
.B "less [\-[+]aABcCdeEfFgGiIJKLmMnNqQrRsSuUVwWX~]"
.br
.B "     [\-b \fIspace\/\fP] [\-h \fIlines\/\fP] [\-j \fIline\/\fP] [\-k \fIkey
file\/\fP]"
.br
.B "     [\-{oO} \fIlogfile\/\fP] [\-p \fIpattern\/\fP] [\-P \fIprompt\/\fP] [\
-t \fItag\/\fP]"
.br
.B "     [\-T \fItagsfile\/\fP] [\-x \fItab\/\fP,...] [\-y \fIlines\/\fP] [\-[z
] \fIlines\/\fP]"
.br
.B "     [\-# \fIshift\/\fP] [+[+]\fIcmd\/\fP] [\-\-] [\fIfilename\/\fP]..."
.br
(See the OPTIONS section for alternate option syntax with long option names.)

.SH DESCRIPTION
less.1.gz (file 1 of 5)
```

```
LESS(1)                   General Commands Manual                   LESS(1)

NAME
       less - opposite of more

SYNOPSIS
       less -?
       less --help
       less -V
       less --version
       less [-[+]aABcCdeEfFgGiIJKLmMnNqQrRsSuUVwWX~]
            [-b space] [-h lines] [-j line] [-k keyfile]
            [-{oO} logfile] [-p pattern] [-P prompt] [-t tag]
            [-T tagsfile] [-x tab,...] [-y lines] [-[z] lines]
            [-# shift] [+[+]cmd] [--] [filename]...
       (See  the OPTIONS section for alternate option syntax with long option
       names.)

DESCRIPTION
       Less is a program similar to more (1), but it has many more  features.
       Less  does  not have to read the entire input file before starting, so
       with large input files it starts up faster than text editors  like  vi
       (1).    Less  uses termcap (or terminfo on some systems), so it can run
       on a variety of terminals.  There is even limited support for hardcopy
       terminals.   (On a hardcopy terminal, lines which should be printed at
       the top of the screen are prefixed with a caret.)

       Commands are based on both more and vi.  Commands may be preceded by a
 Manual page less(1) line 1 (press h for help or q to quit)
```

3.Используя встроенную переменную **$RANDOM**, написал командный файл, генерирующий случайную последовательность букв латинского алфавита. Учёл, что **$RANDOM** выдаёт псевдослучайные числа в диапазоне от **0** до **32767**.

```
yuvkuznetsov@yuvkuznetsov-VirtualBox:~$ touch lab13_2.sh
yuvkuznetsov@yuvkuznetsov-VirtualBox:~$ chmod +x lab13_2.sh
yuvkuznetsov@yuvkuznetsov-VirtualBox:~$ mcedit lab13_2.sh
```

```
/home/yu~13_2.sh   [----] 15 L:[  1+ 1   2/  2] *(27  /  48b) 0116 0x074 [*][X]
#1/bin/bash
echo $RANDOM | tr '[0-9]' '[a-zA-Z]'
```

```
yuvkuznetsov@yuvkuznetsov-VirtualBox:~$ ./lab13_2.sh
cjdhg
yuvkuznetsov@yuvkuznetsov-VirtualBox:~$ ./lab13_2.sh
bhdj
yuvkuznetsov@yuvkuznetsov-VirtualBox:~$ ./lab13_2.sh
ijaj
yuvkuznetsov@yuvkuznetsov-VirtualBox:~$ ./lab13_2.sh
cica
yuvkuznetsov@yuvkuznetsov-VirtualBox:~$
```

Вывод: изучил основы программирования в оболочке ОС UNIX, научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Ответы на контрольные вопросы:

1.В строке while [$1 != "exit"] квадратные скобки надо заменить на круглые.

2.Есть несколько видов конкатенации строк. Например,

VAR1="Hello,"

VAR2=" World"

VAR3="$VAR1$VAR2"

echo "$VAR3"

3.Команда seq выводит последовательность целых или действительных чисел, подходящую для передачи в другие программы. В bash можно использовать seq с циклом for, используя подстановку команд. Например,

$ for i in $(seq 1 0.5 4)

do

echo "The number is $i"

done

4. Результатом вычисления выражения $((10/3)) будет число 3.

5. Список того, что можно получить, используя Z Shell вместо Bash:

Встроенная команда zmv поможет массово переименовать файлы/директории, например, чтобы добавить ‘.txt’ к имени каждого файла, запустите zmv –C '(*)(#q.)' '$1.txt'.

Утилита zcalc — это замечательный калькулятор командной строки, удобный способ считать быстро, не покидая терминал.

Команда zparseopts — это однострочник, который поможет разобрать сложные варианты, которые предоставляются скрипту.

Команда autopushd позволяет делать popd после того, как с помощью cd, чтобы вернуться в предыдущую директорию.

Поддержка чисел с плавающей точкой (коей Bash не содержит).

Поддержка для структур данных «хэш».

Есть также ряд особенностей, которые присутствуют только в Bash:

Опция командной строки –norc, которая позволяет пользователю иметь дело с инициализацией командной строки, не читая файл .bashrc

Использование опции –rcfile <filename> с bash позволяет исполнять команды из определённого файла.

Отличные возможности вызова (набор опций для командной строки)

Может быть вызвана командой sh

Bash можно запустить в определённом режиме POSIX. Примените set –o posix, чтобы включить режим, или —posix при запуске.

Можно управлять видом командной строки в Bash. Настройка переменной PROMPT_COMMAND с одним или более специальными символами настроит её за вас.

Bash также можно включить в режиме ограниченной оболочки (с rbash или --restricted), это означает, что некоторые команды/действия больше не будут доступны:

Настройка и удаление значений служебных переменных SHELL, PATH, ENV, BASH_ENV

Перенаправление вывода с использованием операторов ‘>’, ‘>|’, ‘<>’, ‘>&’, ‘&>’, ‘>>’

Разбор значений SHELLOPTS из окружения оболочки при запуске

Использование встроенного оператора exec, чтобы заменить оболочку другой командой

6. Синтаксис конструкции for ((a=1; a <= LIMIT; a++)) верен.

7. Язык bash и другие языки программирования:

-Скорость работы программ на ассемблере может быть более 50% медленнее, чем программ на си/си++, скомпилированных с максимальной оптимизацией;

-Скорость работы виртуальной ява-машины с байт-кодом часто превосходит скорость аппаратуры с кодами, получаемыми трансляторами с языков высокого уровня. Ява-машина уступает по скорости только ассемблеру и лучшим оптимизирующим трансляторам;

-Скорость компиляции и исполнения программ на яваскрипт в популярных браузерах лишь в 2-3 раза уступает лучшим трансляторам и превосходит даже некоторые качественные компиляторы, безусловно намного (более чем в 10 раз) обгоняя большинство трансляторов других языков сценариев и подобных им по скорости исполнения программ;

-Скорость кодов, генерируемых компилятором языка си фирмы Intel, оказалась заметно меньшей, чем компилятора GNU и иногда LLVM;

-Скорость ассемблерных кодов x86-64 может меньше, чем аналогичных кодов x86, примерно на 10%;

-Оптимизация кодов лучше работает на процессоре Intel;

-Скорость исполнения на процессоре Intel была почти всегда выше, за исключением языков лисп, эрланг, аук (gawk, mawk) и бэш. Разница в скорости по бэш скорее всего вызвана разными настройками окружения на тестируемых системах, а не собственно транслятором или железом. Преимущество Intel особенно заметно на 32-разрядных кодах;

-Стек большинства тестируемых языков, в частности, ява и яваскрипт, поддерживают только очень ограниченное число рекурсивных вызовов. Некоторые трансляторы (gcc, icc, ...) позволяют увеличить размер стека изменением переменных среды исполнения или параметром;

-В рассматриваемых версиях gawk, php, perl, bash реализован динамический стек, позволяющий использовать всю память компьютера. Но perl и, особенно, bash используют стек настолько экстенсивно, что 8-16 ГБ не хватает для расчета ack(5,2,3)