

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 13

дисциплина: операционные системы

Студент: Пиняева Анна Андреевна

Группа: НФИбд-02-20

МОСКВА

2021

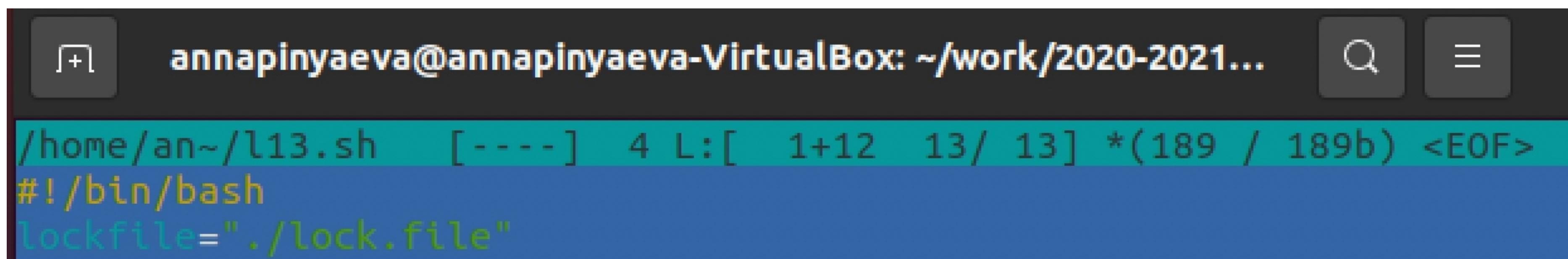
Цель работы

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

Ход работы

- Написать командный файл, реализующий упрощённый механизм семафоров. 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). 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.

Рисунок 1:



```
аннапиняева@аннапиняева-VirtualBox: ~/work/2020-2021...
/home/an-/l13.sh  [ ---- ]  4 L:[  1+12  13/ 13] *(189 / 189b) <EOF>
#!/bin/bash
lockfile="./lock.file"
```

```
exec {fn}>$lockfile
echo "lock"
until flock -n ${fn}
do<---->echo "not lock"
<----->sleep 1
<----->flock -n ${fn}
done
for ((i=0;i<=6;i++))
do echo "work"
    sleep 1
done
```

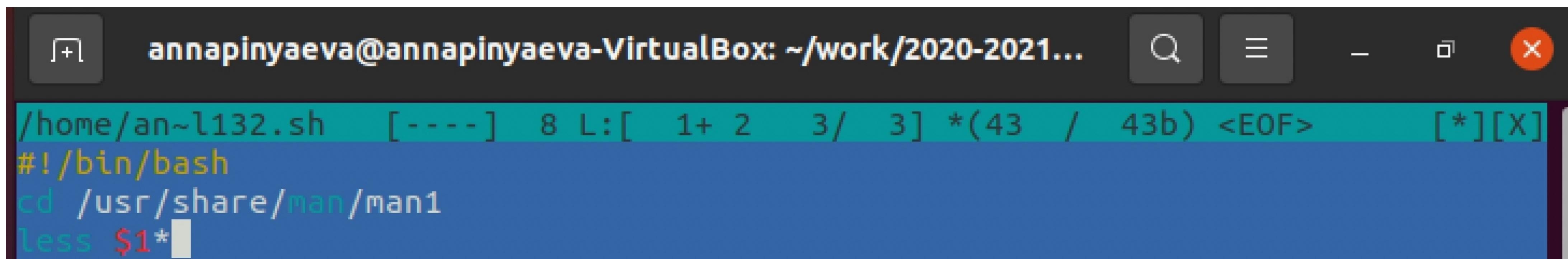
The screenshot shows a terminal window with the following details:

- Title Bar:** The title bar displays the session information: `annapinyaeva@annapinyaeva-VirtualBox: ~/work/2020-2021...`. It also includes standard window control buttons (minimize, maximize, close) and a search icon.
- Terminal Content:** The terminal window contains the following text:

```
annapinyaeva@annapinyaeva-VirtualBox:~/work/2020-2021/os/lab13$ touch l13.sh
annapinyaeva@annapinyaeva-VirtualBox:~/work/2020-2021/os/lab13$ mcedit l13.sh
annapinyaeva@annapinyaeva-VirtualBox:~/work/2020-2021/os/lab13$ chmod +x l13.sh
annapinyaeva@annapinyaeva-VirtualBox:~/work/2020-2021/os/lab13$ ./l13.sh
lock
work
work
work
work
work
work
work
annapinyaeva@annapinyaeva-VirtualBox:~/work/2020-2021/os/lab13$
```

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.

Рисунок 3:



The screenshot shows a terminal window with a dark theme. The title bar displays the user's name and session information: annapinyaeva@annapinyaeva-VirtualBox: ~/work/2020-2021... The window contains a command-line interface with the following text:

```
/home/an~l132.sh  [---]  8 L:[ 1+ 2  3/ 3] *(43  / 43b) <EOF> [*][X]
#!/bin/bash
cd /usr/share/man/man1
less $1*
```



Рисунок 4:

The screenshot shows a terminal window with a dark theme. The title bar displays the user's name and session information: `annapinyaeva@annapinyaeva-VirtualBox: ~/work/2020-2021...`. The window contains the following text, which is a man page for the 'less' command:

```
.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 [\-\-[+]aABcCdeEfFgGiIJKLMNOPQRSTUVWXYZ]"
.br
.B "      [\-\-b \fIspace\/\fP] [\-\-h \fIlines\/\fP] [\-\-j \fIline\/\fP] [\-\-k \fIkey
file\/\fP]"
.br
.B "      [\-\{o0\} \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]..."
```

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.

```
/home/an~l133.sh [----] 4 L:[ 1+ 9 10/ 11] *(179 / 180b) 0010 0x00A [*][X]
#!/bin/bash
M=10
c=1
d=1
while (( $c!=($M+1)))..
do echo $(for((i=1;i<=10;i++));do printf '%s' "${RANDOM;0;1}";done)|tr '[0-9]' 
echo $d
((c+=1))
((d+=1))
done
```

Рисунок 6:

```
annapinyaeva@annapinyaeva-VirtualBox:~/work/2020-2021/os/lab13$ ncedit l133.sh  
annapinyaeva@annapinyaeva-VirtualBox:~/work/2020-2021/os/lab13$ ./l133.sh  
bbfgbdidbg  
1  
cjdccbbibc  
2  
bgjcbjccdc  
3  
dfbbcigccc  
4  
fgbcciehcb  
5  
cbcdbbcfdc  
6
```

```
cbcbddjcjc  
7  
ccbccbcdbbb  
8  
bbhbddfbbj  
9  
cbbccbcccc  
10
```

записи оценки виртуальной машины - VirtualBox - /work/2020-2021/oc/Lab13S

Вывод: я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляемых конструкций и циклов.

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

Использование опции –rcfile 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 java-машины с байт-кодом часто превосходит скорость аппаратуры с кодами, получаемыми трансляторами с языков высокого уровня. Java-машина уступает по скорости только ассемблеру и лучшим оптимизирующими 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 фирмой 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, java 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)