

# 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: *Нгуен Хоанг Нам* Группа: *НФИбд-03-20*

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  $t_1$  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). 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.

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

```
#!/bin/bash
lockfile="./blockfile"
exec {fn}>$lockfile
echo "lock"
until flock -n ${fn}
do
    echo "not lock"
    sleep 1
    flock -n ${fn}
done
for ((i=0;i<=5;i++))
do
    echo "work"
    sleep 1
done
```

lab13.sh All L1 (Shell-script[bash])

```
namnguyen@namnguyen-VirtualBox:~$ ./lab13.sh
lock
work
work
work
work
work
work
work
```

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.

```
namnguyen@namnguyen-VirtualBox:~$ chmod +x lab13-1.sh
```

```
#!/bin/bash
cd /usr/share/man/man1
less $1*
```

-- lab13-1.sh All L4 (Shell-script[bash])

```

.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.3.
.TH TEST "1" "September 2019" "GNU coreutils 8.30" "User Commands"
.SH NAME
test \- check file types and compare values
.SH SYNOPSIS
.B test
.I EXPRESSION
.br
.B test
.br
.\" \& tells doclifter the brackets are literal (Bug#31803).
.B [\&
.I EXPRESSION
.B ]\&
.br
.B "[\& ]\&"
.br
.B [\&
.I OPTION
.SH DESCRIPTION
.\" Add any additional description here
.PP
Exit with the status determined by EXPRESSION.
.TP
\fb\-\-help\fr
display this help and exit
.TP
\fb\-\-version\fr
output version information and exit
.PP
An omitted EXPRESSION defaults to false. Otherwise,
EXPRESSION is true or false and sets exit status. It is one of:
.TP
( EXPRESSION )
EXPRESSION is true
.TP
! EXPRESSION
EXPRESSION is false
.TP
EXPRESSION1 \fb\-a\fr EXPRESSION2
both EXPRESSION1 and EXPRESSION2 are true
.TP
EXPRESSION1 \fb\-o\fr EXPRESSION2
either EXPRESSION1 or EXPRESSION2 is true
.TP
\fb\-n\fr STRING
the length of STRING is nonzero
.TP

```

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

NAME
    less - opposite of more

SYNOPSIS
    less -?
    less --help
    less -V
    less --version
    less [-[+]aABcCdeEffGgIiJkLmMnNqQrRsSuUVvWwX~-]
        [-b space] [-h lines] [-j line] [-k keyfile]
        [-{o0} logfile] [-p pattern] [-P prompt] [-t tag]
        [-T tagfile] [-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 termi-
    nals. (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 decimal number,
    called N in the descriptions below. The number is used by some commands, as indicated.

COMMANDS
    In the following descriptions, ^X means control-X. ESC stands for the ESCAPE key; for ex-
    ample ESC-v means the two character sequence "ESCAPE", then "v".

    h or H Help: display a summary of these commands. If you forget all the other commands,
    remember this one.

    SPACE or ^V or f or ^F
        Scroll forward N lines, default one window (see option -z below). If N is more than
        the screen size, only the final screenful is displayed. Warning: some systems use
        ^V as a special literalization character.

    z
        Like SPACE, but if N is specified, it becomes the new window size.

    ESC-SPACE
        Like SPACE, but scrolls a full screenful, even if it reaches end-of-file in the
        process.

    ENTER or RETURN or ^N or e or ^E or j or ^J
        Scroll forward N lines, default 1. The entire N lines are displayed, even if N is
        more than the screen size.

    d or ^D
        Scroll forward N lines, default one half of the screen size. If N is specified, it
        becomes the new default for subsequent d and u commands.

    b or ^B or ESC-v
        Scroll backward N lines, default one window (see option -z below). If N is more
        than the screen size, only the final screenful is displayed.

    w
        Like ESC-v, but if N is specified, it becomes the new window size.
```

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

```
namnguyen@namnguyen-VirtualBox:~$ chmod+x lab13-2
```

```
#!/bin/bash
m=10
c=1
d=1
echo
echo "random words"
while (($c!=($m+1)))
do
    echo $(for((i=1;i<=10;i++)); do printf '%s' "${RANDOM:0:1}"; done) | tr '0-9' '[a-z]'
    echo $d
    ((c+=1))
    ((d+=1))
done
```

lab13-2.sh All L1 (Shell-script[bash])

```
namnguyen@namnguyen-VirtualBox:~$ ./lab13-2.sh

random words
cibcbibccj
1
gbbcdbbbed
2
ccibgibbch
3
jbcbbccbbcc
4
deicbdebjc
5
ibidgccicb
6
bbcbcfcbdc
7
bchciccbcc
8
ccdghcdcdg
9
ccdbbccjdf
10
```

**Вывод:** изучил основы программирования в оболочке ОС UNIX, научил писать более сложные командные файлы с использованием логических управляющих 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 – `porcs`, 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 `exes`, 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`, `awk`) 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, iss, ...) 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 ask(5,2,3).