

Отчёт по лабораторной работе

Лабораторная работа № 13.

Lukashov Nikita

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	13

List of Tables

List of Figures

1 Цель работы

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

2 Выполнение лабораторной работы

1. 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.

```
nalukashov@dk5n55 ~ $ touch lab10.sh
nalukashov@dk5n55 ~ $ emacs
bash: emacs: команда не найдена
nalukashov@dk5n55 ~ $ emacs lab10.sh
nalukashov@dk5n55 ~ $ rm -r lab10.sh
nalukashov@dk5n55 ~ $ ls
abc1      GNUstep      lab2.asm      play          qwert.map     tmp           лаба10П      Шаблоны
asdfg     lab03-1.asm  laboratory    public        README.md     work          лаба20П
asdfg.asm lab07        makefile      public_html   reports       Видео        Музыка
australia lab07.asm    may           qwert         script.sh     Документы    Общедоступные
c++       lab07.sh     monthly       qwert.asm     ski.plases    Загрузки     отчет_лаб_шаблон.odt
feathers   lab07.sh~    my_os         qwert.lst     '#sript.sh#'  Изображения  'Рабочий стол'

nalukashov@dk5n55 ~ $ touch lab10.sh
nalukashov@dk5n55 ~ $ emacs
bash: emacs: команда не найдена
nalukashov@dk5n55 ~ $ emacs lab10.sh
nalukashov@dk5n55 ~ $ lab10.sh
bash: lab10.sh: команда не найдена
nalukashov@dk5n55 ~ $ chmod +x lab10.sh
nalukashov@dk5n55 ~ $ lab10.sh
bash: lab10.sh: команда не найдена
nalukashov@dk5n55 ~ $ ./
asdfg      .gnupg/      lab10.sh      .pki/         .sage/         .vscode/      лаба10П/
australia/ GNUstep/     laboratory/   play/         script.sh      work/         лаба20П/
c++/       .gphoto/     .local/       public/        ski.plases/    Видео/        Музыка/
.config/   .ipython/    monthly/      public_html/   .ssh/          Документы/    Общедоступные/
.emacs.d/  .kde4/       .mozilla/     qwert          .texlive2020/ Загрузки/     Рабочий стол/
.git/      lab07.asm    my_os         reports/       tmp/           Изображения/  Шаблоны/

nalukashov@dk5n55 ~ $ ./lab10.sh
nalukashov@dk5n55 ~ $ bash lab10.sh
nalukashov@dk5n55 ~ $ emacs lab10.sh
nalukashov@dk5n55 ~ $ ./lab10.sh
lock
work
work
work
work
work
work
```

emacs@dk5n55

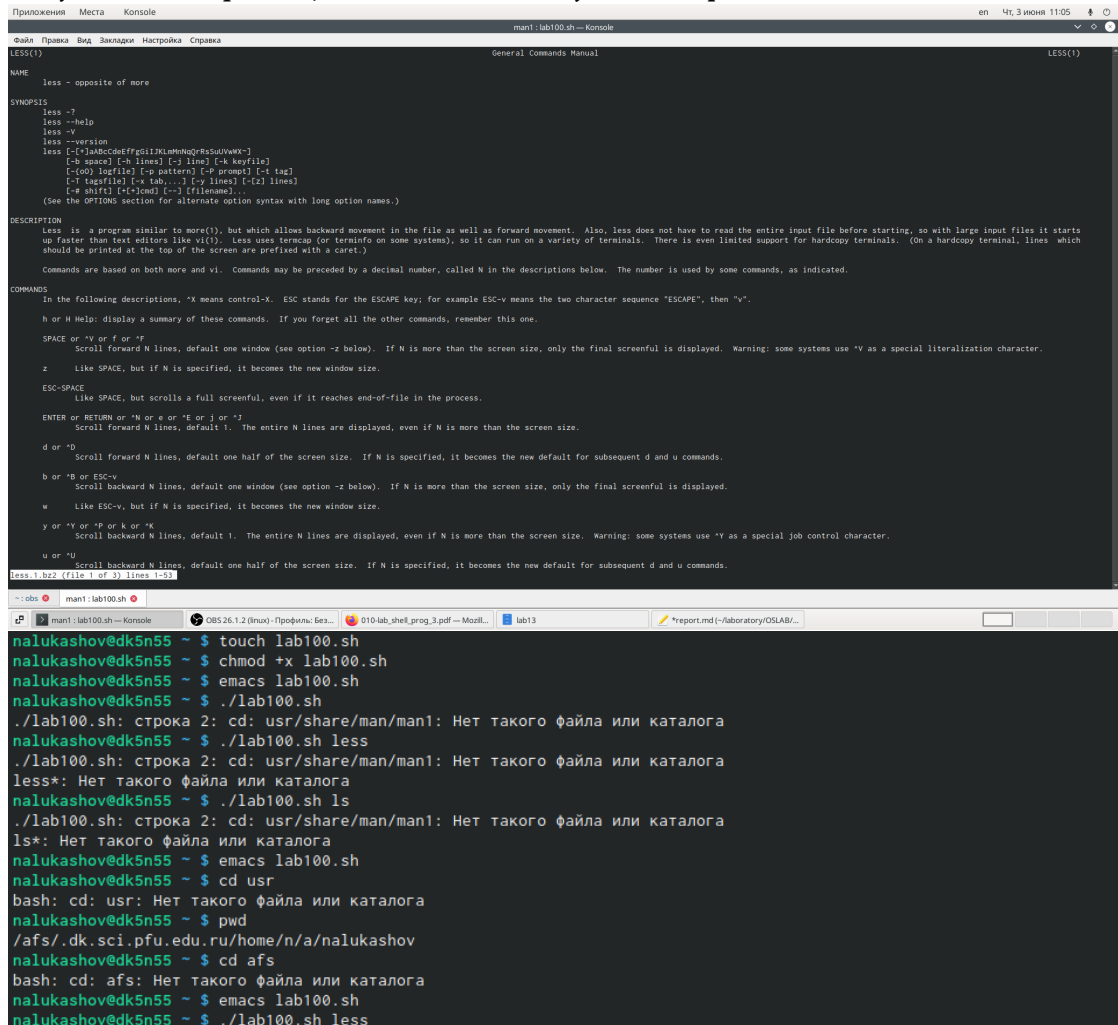
File Edit Options Buffers Tools Sh-Script Help

```
#!/bin/bash
lockfile="./lockfile"
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
```

~:--- lab10.sh All L1 (Shell-script[bash]) Чт июн 3 11:03 1.63

2. Реализовать команду manps 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.



```
man1: lab100.sh - Konsole
LESS(1)
NAME
less - opposite of more
SYNOPSIS
less -?
less --help
less -v
less --version
less [-*]aAbCdEffgIjKlMmNnQqRrSaSuVvwX-
[-b space] [-h lines] [-j line] [-k keyfile]
[-oO] [-of file] [-p pattern] [-P pattern] [-t tag]
[-T tagfile] [-x tab,...] [-y lines] [-z lines]
[-e shift] [-c load] [-s] [-r name]...
(See the OPTIONS section for alternate option syntax with long option names.)
DESCRIPTION
Less is a program similar to more(1), but which allows backward movement in the file as well as forward movement. Also, 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 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 example 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.
y or ^Y or ^P or k or ^K
Scroll backward N lines, default 1. The entire N lines are displayed, even if N is more than the screen size. Warning: some systems use ^Y as a special job control character.
u or ^U
Scroll backward N lines, default one half of the screen size. If N is specified, it becomes the new default for subsequent d and u commands.
less: lab100 (file 1 of 3) lines 1-53
man1: lab100.sh - Konsole
nalukashov@dk5n55 ~$ touch lab100.sh
nalukashov@dk5n55 ~$ chmod +x lab100.sh
nalukashov@dk5n55 ~$ emacs lab100.sh
nalukashov@dk5n55 ~$ ./lab100.sh
./lab100.sh: строка 2: cd: usr/share/man/man1: Нет такого файла или каталога
nalukashov@dk5n55 ~$ ./lab100.sh less
./lab100.sh: строка 2: cd: usr/share/man/man1: Нет такого файла или каталога
less*: Нет такого файла или каталога
nalukashov@dk5n55 ~$ ./lab100.sh ls
./lab100.sh: строка 2: cd: usr/share/man/man1: Нет такого файла или каталога
ls*: Нет такого файла или каталога
nalukashov@dk5n55 ~$ emacs lab100.sh
nalukashov@dk5n55 ~$ cd usr
bash: cd: usr: Нет такого файла или каталога
nalukashov@dk5n55 ~$ pwd
/afs/.dk.sci.pfu.edu.ru/home/n/a/nalukashov
nalukashov@dk5n55 ~$ cd afs
bash: cd: afs: Нет такого файла или каталога
nalukashov@dk5n55 ~$ emacs lab100.sh
nalukashov@dk5n55 ~$ ./lab100.sh less
```

- 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.


```
emacs@dk5n55
File Edit Options Buffers Tools Sh-Script Help

#!/bin/bash
M=10
c=1
d=1
echo
echo "10 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

--:--- lab1000.sh All L1 (Shell-script[bash]) Чт июн 3 11:07 0.89

./lab1000.sh: строка 9: синтаксическая ошибка рядом с неожиданным маркером «do»
./lab1000.sh: строка 9: `    echo $(for((i=1;i<=10;i++)); do printf '%s' "${RANDOM:0:1}"; done | tr '[:0-9]' '[:a-z]''
nalukashov@dk5n55 ~ $ emacs lab1000.sh
nalukashov@dk5n55 ~ $ ./lab1000.sh

10 random words:
./lab1000.sh: строка 9: неожиданный конец файла во время поиска «)»
./lab1000.sh: строка 15: синтаксическая ошибка: неожиданный конец файла
nalukashov@dk5n55 ~ $ emacs lab1000.sh
nalukashov@dk5n55 ~ $ ./lab1000.sh

10 random words:
./lab1000.sh: строка 9: синтаксическая ошибка рядом с неожиданным маркером «do»
./lab1000.sh: строка 9: `    echo $(for((i=1;i<=10;i++)); do printf '%s' "${RANDOM:0:1}"; done | tr '[:0-9]' '[:a-z]''
nalukashov@dk5n55 ~ $ emacs lab1000.sh
nalukashov@dk5n55 ~ $ ./lab1000.sh

10 random words:
./lab1000.sh: строка 9: неожиданный конец файла во время поиска «)»
./lab1000.sh: строка 15: синтаксическая ошибка: неожиданный конец файла
nalukashov@dk5n55 ~ $ emacs lab1000.sh
nalukashov@dk5n55 ~ $ ./lab1000.sh

10 random words:
./lab1000.sh: строка 9: синтаксическая ошибка рядом с неожиданным маркером «do»
./lab1000.sh: строка 9: `    echo $(for((i=1;i<=10;i++)); do printf '%s' "${RANDOM:0:1}"; done | tr '[:0-9]' '[:a-z]''
nalukashov@dk5n55 ~ $ emacs lab1000.sh
nalukashov@dk5n55 ~ $ ./lab1000.sh

10 random words:
bbbbggbccb
1
bbgcbbbcb
2
cbdfcbfccc
3
edccbhdccd
4
dbcbfdbbc
5
cibddccgbb
6
bbbcbcbbi
7
icbbdcbedc
8
beddbdbbcd
9
dcbcdcecfj
10
```

#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="VAR1VAR2"  
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` 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, 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 ask(5,2,3)

3 Выводы

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