

Лабораторная работа 13

Дисциплина: Операционные системы

Куликов Максим Игоревич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	12
5	Контрольные вопросы	13

Список таблиц

Список иллюстраций

3.1	Первый скрипт	8
3.2	Тест №1	9
3.3	Второй скрипт	9
3.4	Запуск скрипта	10
3.5	Тест №2	10
3.6	Содержание 3 скрипта	11
3.7	Тест №3	11

1 Цель работы

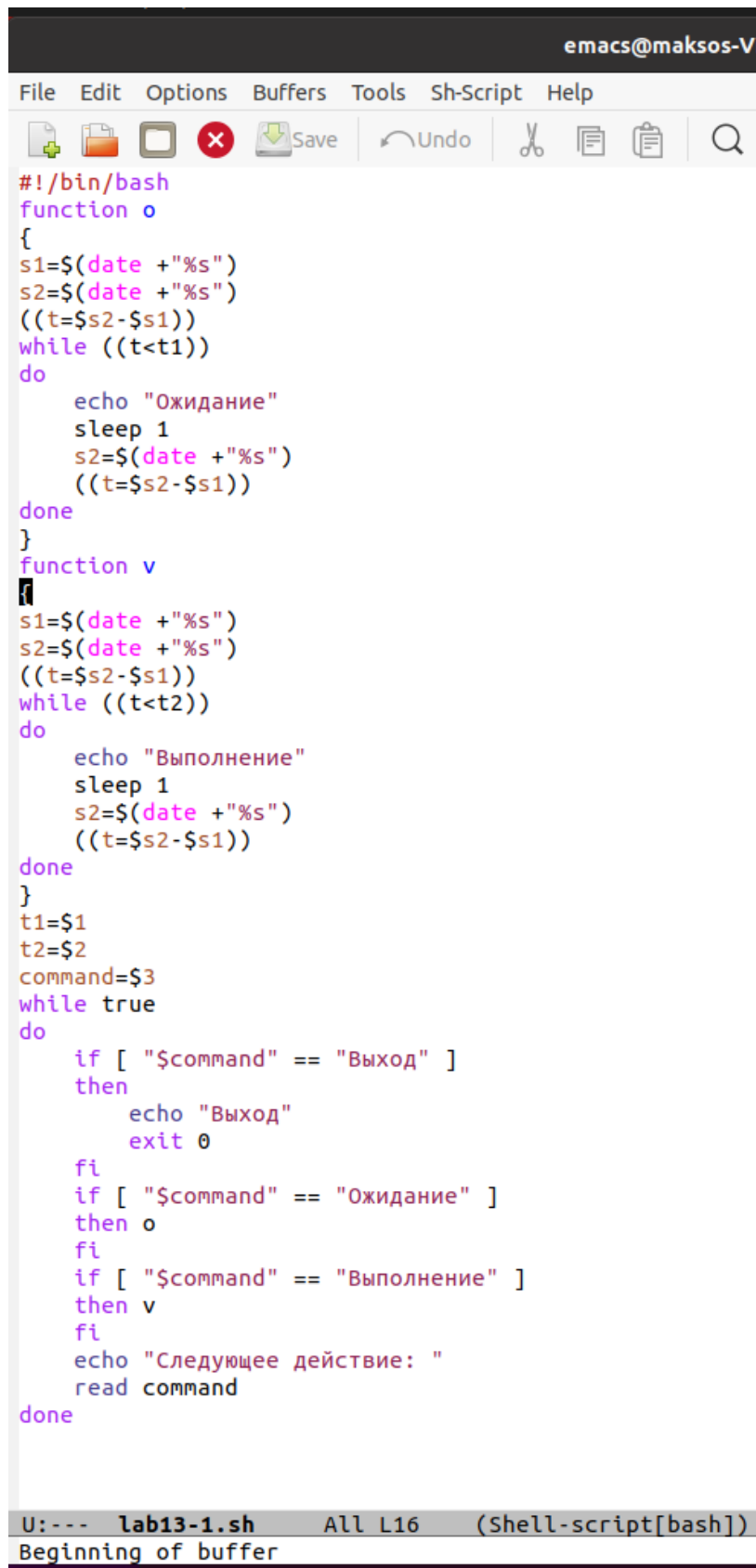
Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

2 Задание

1. Ознакомиться с теоретическим материалом.
2. Выполнить работу.

3 Выполнение лабораторной работы

1. Создаю файл с расширением “sh”, в котором пишу скрипт, который является упрощенным механизмом семафоров (рис. -fig. 3.1)

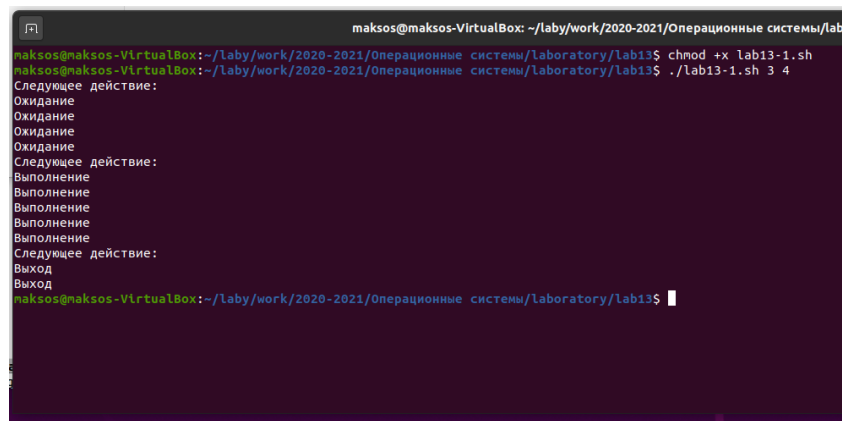


```
emacs@maksos-V
File Edit Options Buffers Tools Sh-Script Help
Save Undo
#!/bin/bash
function o
{
s1=$(date +%s")
s2=$(date +%s")
((t=$s2-$s1))
while ((t<t1))
do
echo "Ожидание"
sleep 1
s2=$(date +%s")
((t=$s2-$s1))
done
}
function v
{
s1=$(date +%s")
s2=$(date +%s")
((t=$s2-$s1))
while ((t<t2))
do
echo "Выполнение"
sleep 1
s2=$(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 [ "$command" == "Ожидание" ]
then o
fi
if [ "$command" == "Выполнение" ]
then v
fi
echo "Следующее действие: "
read command
done

U:--- lab13-1.sh All L16 (Shell-script[bash])
Beginning of buffer
```

Рис. 3.1: Первый скрипт

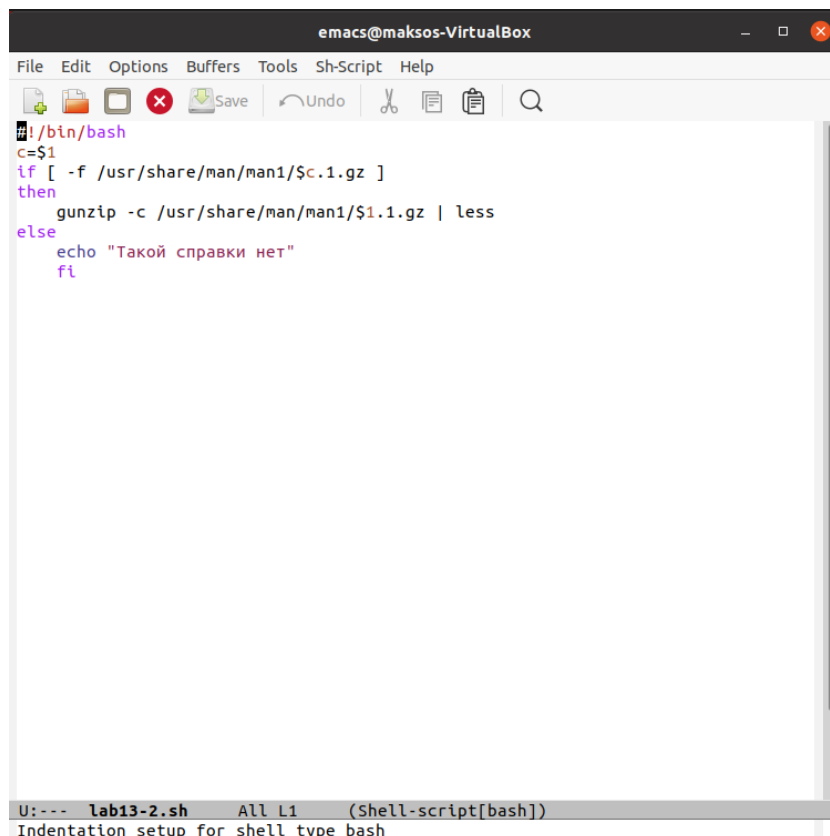
2. Тест скрипта. Работает исправно (рис. -fig. 3.2)



```
maksos@maksos-VirtualBox: ~/laby/work/2020-2021/Операционные системы/lab
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/Laboratory/lab13$ chmod +x lab13-1.sh
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/Laboratory/lab13$ ./lab13-1.sh 3 4
Следующее действие:
Ожидание
Ожидание
Ожидание
Ожидание
Следующее действие:
Выполнение
Выполнение
Выполнение
Выполнение
Следующее действие:
Выход
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/Laboratory/lab13$
```

Рис. 3.2: Тест №1

3. Создаю файл с расширением “sh”, в котором пишу скрипт, который реализовывает команду man (рис. -fig. 3.3)

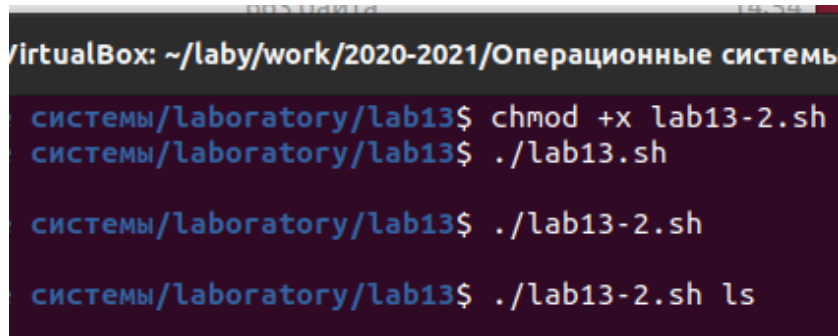


```
emacs@maksos-VirtualBox
File Edit Options Buffers Tools Sh-Script Help
Save Undo
#!/bin/bash
c=$1
if [ -f /usr/share/man/man1/$c.1.gz ]
then
  gunzip -c /usr/share/man/man1/$c.1.gz | less
else
  echo "Такой справки нет"
fi

U:--- lab13-2.sh All L1 (Shell-script[bash])
Indentation setup for shell type bash
```

Рис. 3.3: Второй скрипт

4. Запускаю скрипт. Он должен должен выдать справку об команде “ls” (рис. -fig. 3.4)



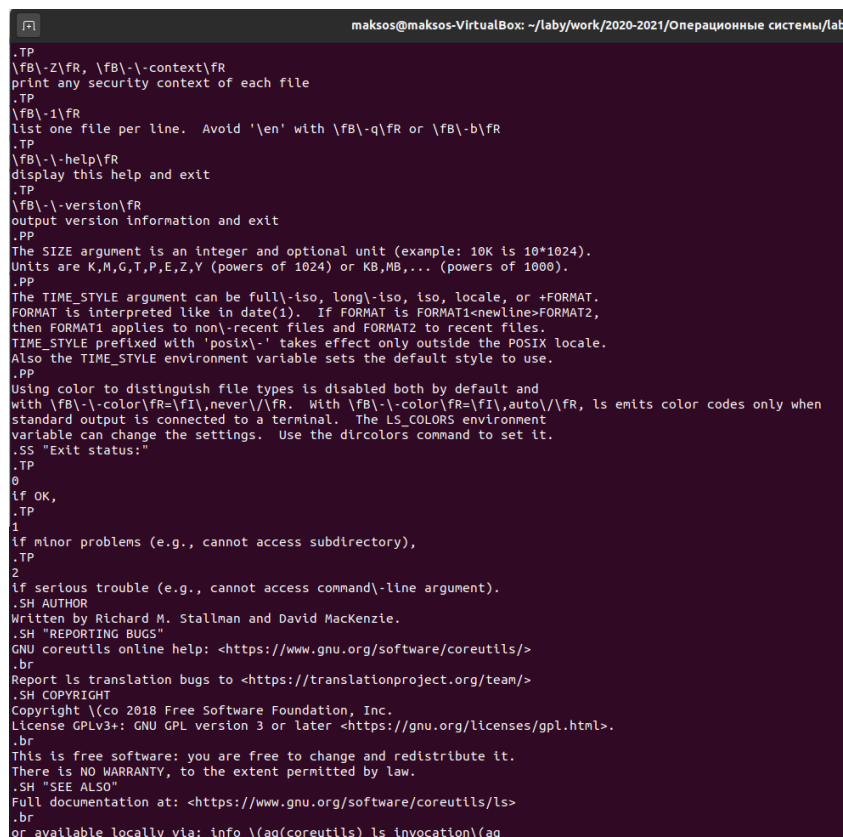
```
VirtualBox: ~/laby/work/2020-2021/Операционные системы/
системы/laboratory/lab13$ chmod +x lab13-2.sh
системы/laboratory/lab13$ ./lab13.sh

системы/laboratory/lab13$ ./lab13-2.sh

системы/laboratory/lab13$ ./lab13-2.sh ls
```

Рис. 3.4: Запуск скрипта

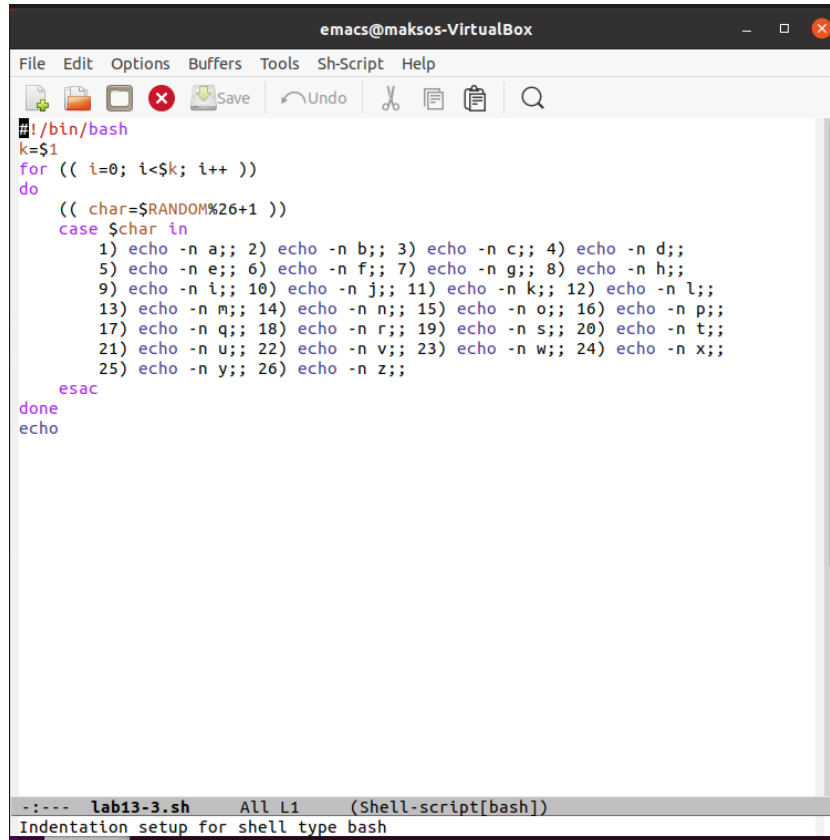
5. Скрипт работает исправно (рис. -fig. 3.5)



```
maksos@maksos-VirtualBox: ~/laby/work/2020-2021/Операционные системы/lab
.TP
\fb\-Z\fr, \fb\-.context\fr
print any security context of each file
.TP
\fb\-l\fr
list one file per line. Avoid '\en' with \fb\-q\fr or \fb\-b\fr
.TP
\fb\-.help\fr
display this help and exit
.TP
\fb\-.version\fr
output version information and exit
.PP
The SIZE argument is an integer and optional unit (example: 10K is 10*1024).
Units are K,M,G,T,P,E,Z,Y (powers of 1024) or KB,MB,... (powers of 1000).
.PP
The TIME_STYLE argument can be full\-iso, long\-iso, iso, locale, or +FORMAT.
FORMAT is interpreted like in date(1). If FORMAT is: FORMAT<newline>FORMAT2,
then FORMAT1 applies to non\-recent files and FORMAT2 to recent files.
TIME_STYLE prefixed with 'posix\-' takes effect only outside the POSIX locale.
Also the TIME_STYLE environment variable sets the default style to use.
.PP
Using color to distinguish file types is disabled both by default and
with \fb\-.color\fr=\fi\,never\|fr. With \fb\-.color\fr=\fi\,auto\|fr, ls emits color codes only when
standard output is connected to a terminal. The LS_COLORS environment
variable can change the settings. Use the dircolors command to set it.
.SS "Exit status:"
.TP
0
if OK,
.TP
1
if minor problems (e.g., cannot access subdirectory),
.TP
2
if serious trouble (e.g., cannot access command\-line argument).
.SH AUTHOR
Written by Richard M. Stallman and David MacKenzie.
.SH "REPORTING BUGS"
GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
.br
Report ls translation bugs to <https://translationproject.org/team/>
.SH COPYRIGHT
Copyright (c) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
.br
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
.SH "SEE ALSO"
Full documentation at: <https://www.gnu.org/software/coreutils/ls>
.br
or available locally via: info \ag\coreutils ls invocation\ag
```

Рис. 3.5: Тест №2

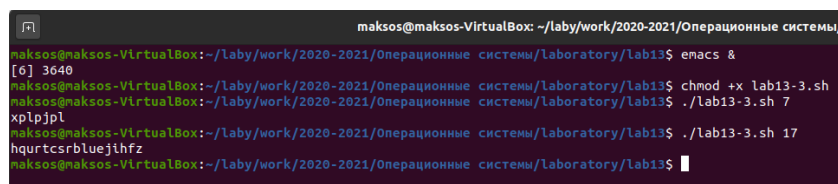
6. Создаю файл с расширением “sh”, в котором пишу скрипт, который принимает значение с клавиатуры и выводит на экран строку из случайных символов, длина которой равна числу, введённому с клавиатуры (рис. - fig. 3.6)



```
#!/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) echo -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
```

Рис. 3.6: Содержание 3 скрипта

7. Тестирую скрипт №3. Работает верно (рис. -fig. 3.7)



```
maksos@maksos-VirtualBox: ~/laby/work/2020-2021/Операционные системы/
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/laboratory/Lab13$ cat lab13-3.sh
[6] 3640
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/laboratory/Lab13$ chmod +x lab13-3.sh
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/laboratory/Lab13$ ./lab13-3.sh 7
xplpjpl
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/laboratory/Lab13$ ./lab13-3.sh 17
hqurctsrbluejihfz
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/laboratory/Lab13$
```

Рис. 3.7: Тест №3

4 Выводы

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

5 Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке `while [$1 != "exit"])$1` следует внести в кавычки («»)
2. Как объединить (конкатенация) несколько строк в одну? С помощью знака `>|`
3. Найдите информацию об утилите `seq`. Какими иными способами можно реализовать ее функционал при программировании на `bash`? Эта утилита выводит последовательность целых чисел с заданным шагом. Также можно реализовать с помощью утилиты `jot`.
4. Какой результат даст вычисление выражения `$((10/3))`? Результат: 3.
5. Укажите кратко основные отличия командной оболочки `zsh` от `bash`. В `zsh` можно настроить отдельные сочетания клавиш так, как вам нравится. Использование истории команд в `zsh` ничем особенным не отличается от `bash`. `Zsh` очень удобен для повседневной работы и делает добрую половину рутинных за вас. Но стоит обратить внимание на различия между этими двумя оболочками. Например, в `zsh` после `for` обязательно вставлять пробел, нумерация массивов в `zsh` начинается с 1, чего совершенно невозможно понять. Так, если вы используете `shell` для повседневной работы, исключающей написание скриптов, используйте `zsh`. Если вам часто приходится писать свои скрипты, только `bash`! Впрочем, можно комбинировать. Как установить `zsh` в качестве оболочки по-умолчанию для отдельного пользователя: `o`
6. Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))` Синтаксис верен.

7. Сравните язык `bash` с языками программирования, которые вы знаете. Какие преимущества у `bash` по сравнению с ними? Какие недостатки?

1. Скорость работы программ на ассемблере может быть более 50% медленнее, чем программ на `си/си++`, скомпилированных с максимальной оптимизацией;
2. 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;
3. 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;
4. Скорость кодов, генерируемых компилятором языка `си` фирмы Intel, оказалась заметно меньшей, чем компилятора GNU и иногда LLVM;
5. Скорость ассемблерных кодов `x86-64` может меньше, чем аналогичных кодов `x86`, примерно на 10%;
6. Оптимизация кодов лучше работает на процессоре Intel;
7. 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;
8. 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;

9. 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 $ask(5,2,3)$