

# **Отчёта по лабораторной работе №12**

**НКНбд-02-21**

Акондзо Жордани Лади Гаэл

# Содержание

1	Цель работы	4
2	Задание	5
3	Ход работы	7
4	Выводы	11
5	Контрольные вопросы	12

## Список иллюстраций

3.1	Написание программы 1 . . . . .	8
3.2	Вывод программы 1 на экран . . . . .	8
3.3	Написание программы 2 . . . . .	9
3.4	Вывод программы 2 на экран . . . . .	9
3.5	Написание программы 3 . . . . .	10
3.6	Вывод программы 3 на экран . . . . .	10

# 1 Цель работы

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

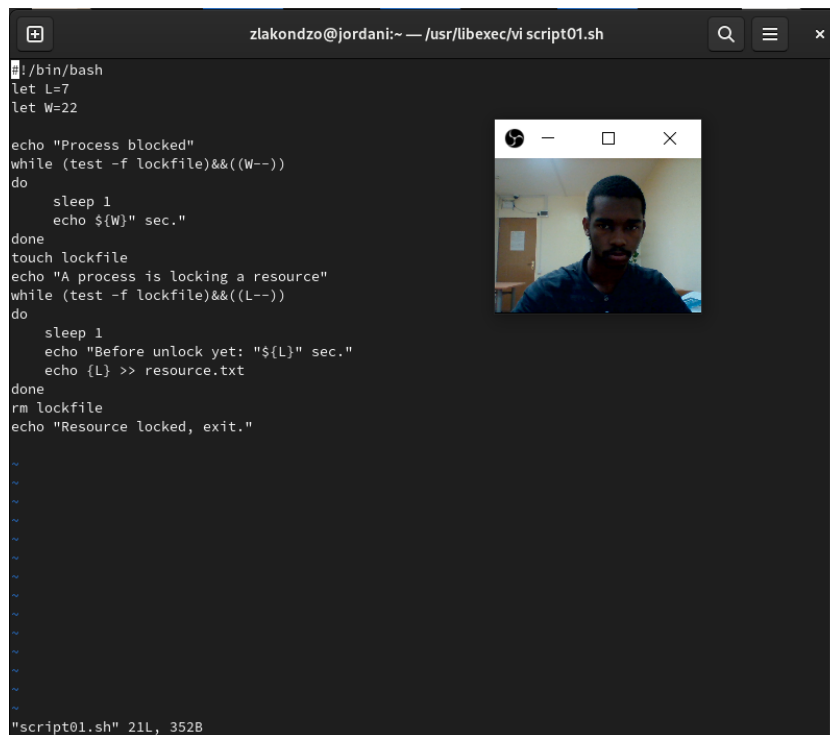
## 2 Задание

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.
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`.
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.

### 3 Ход работы

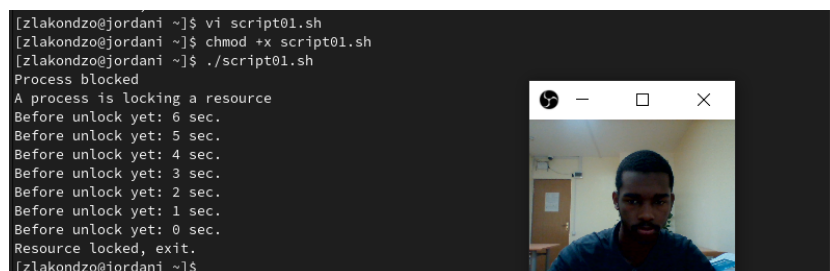
1. Написал командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени  $t_1$  дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени  $t_2 < t_1$ , также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). 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. (рис. 3.1, 3.2)



```
#!/bin/bash
let L=7
let W=22

echo "Process blocked"
while (test -f lockfile)&&((W--))
do
    sleep 1
    echo "${W}" sec."
done
touch lockfile
echo "A process is locking a resource"
while (test -f lockfile)&&((L--))
do
    sleep 1
    echo "Before unlock yet: "${L}" sec."
    echo {L} >> resource.txt
done
rm lockfile
echo "Resource locked, exit."
```

Рис. 3.1: Написание программы 1



```
[zlakeondzo@jordani ~]$ vi script01.sh
[zlakeondzo@jordani ~]$ chmod +x script01.sh
[zlakeondzo@jordani ~]$ ./script01.sh
Process blocked
A process is locking a resource
Before unlock yet: 6 sec.
Before unlock yet: 5 sec.
Before unlock yet: 4 sec.
Before unlock yet: 3 sec.
Before unlock yet: 2 sec.
Before unlock yet: 1 sec.
Before unlock yet: 0 sec.
Resource locked, exit.
[zlakeondzo@jordani ~]$
```

Рис. 3.2: Вывод программы 1 на экран

2. Реализовал 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`. (рис. 3.3, 3.4)



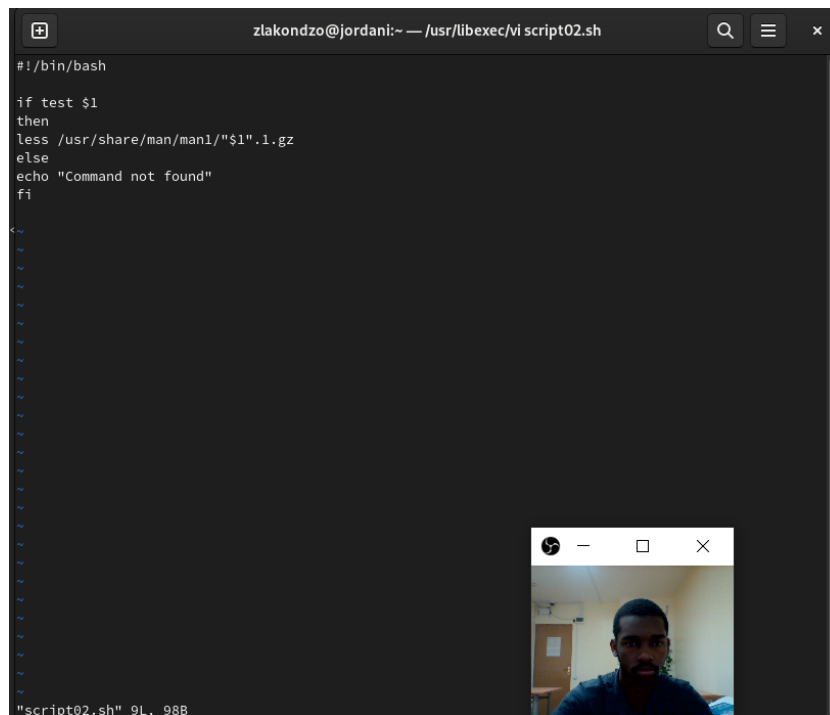


Рис. 3.3: Написание программы 2

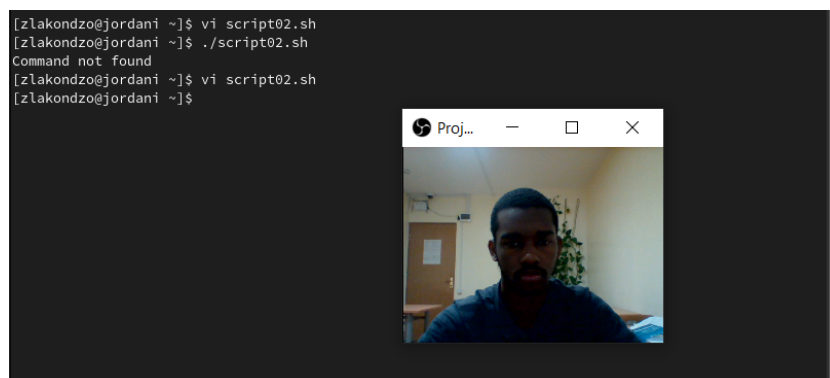


Рис. 3.4: Вывод программы 2 на экран

- 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. рис. 3.5, 3.6)

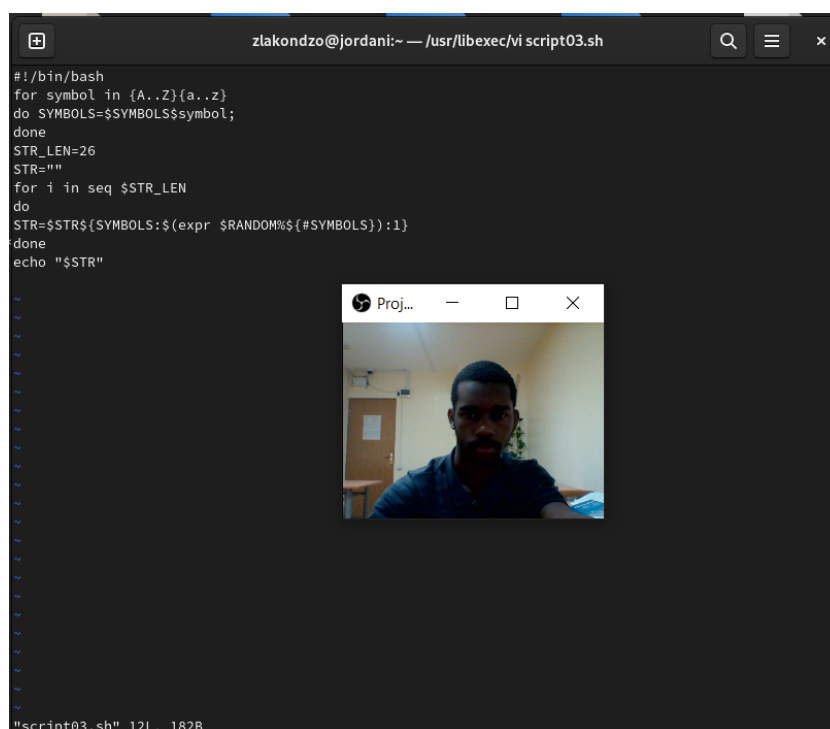


Рис. 3.5: Написание программы 3

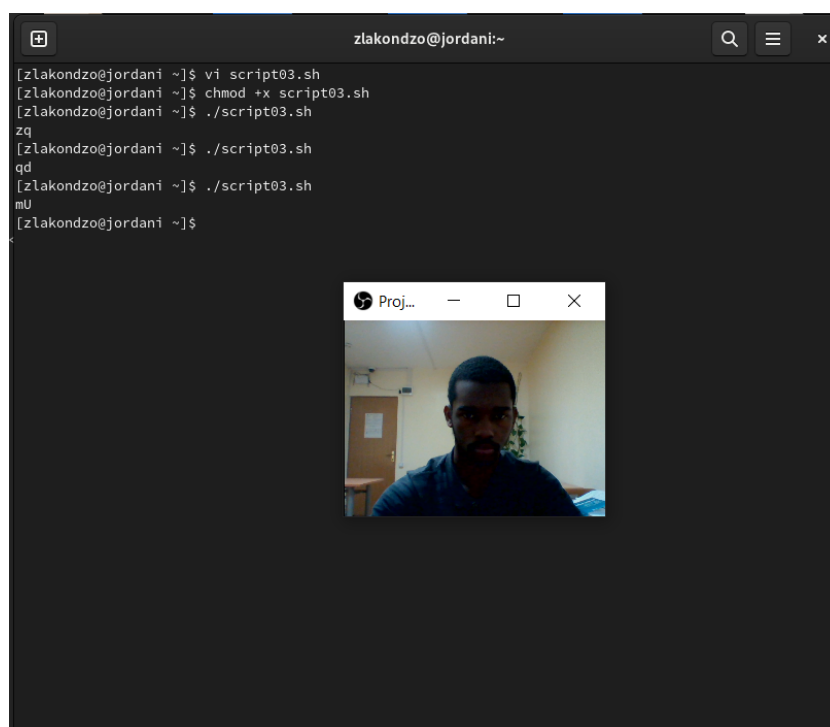


Рис. 3.6: Вывод программы 3 на экран

## 4 Выводы

Во время выполнения работы, мы изучили основы программирования в оболочке ОС UNIX. Научились 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 `>|` можно объединить несколько строк в одну.

3. Найдите информацию об утилите `seq`. Какими иными способами можно реализовать её 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))`?

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

**6. Проверьте, верен ли синтаксис данной конструкции for ((a=1; a <= LIMIT; a++))**

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

**7. Сравните язык bash с какими-либо языками программирования. Какие преимущества у bash по сравнению с ними? Какие недостатки?**

Язык bash и другие языки программирования:

- a. Скорость работы программ на ассемблере может быть более 50% медленнее, чем программ на си/си++, скомпилированных с максимальной оптимизацией;
- b. 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;
- c. 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;
- d. Скорость кодов, генерируемых компилятором языка си фирмы Intel, оказалась заметно меньшей, чем компилятора GNU и иногда LLVM;
- e. Скорость ассемблерных кодов x86-64 может меньше, чем аналогичных кодов x86, примерно на 10%;

- f. Оптимизация кодов лучше работает на процессоре Intel;
- g. 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;
- h. 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;
- i. В рассматриваемых версиях gawk, php, perl, bash реализован динамический стек, позволяющий использовать всю память компьютера. Но perl и, bash используют стек настолько экстенсивно, что 8-16 ГБ не хватает для расчета  $\text{ack}(5,2,3)$  особенно,