

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №11

Дисциплина: Операционные системы

Студент: Акопян Изабелла Арменовна

Группа: НБИбд-01-2020

МОСКВА

2021 г.

Цель:

Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

Ход работы:

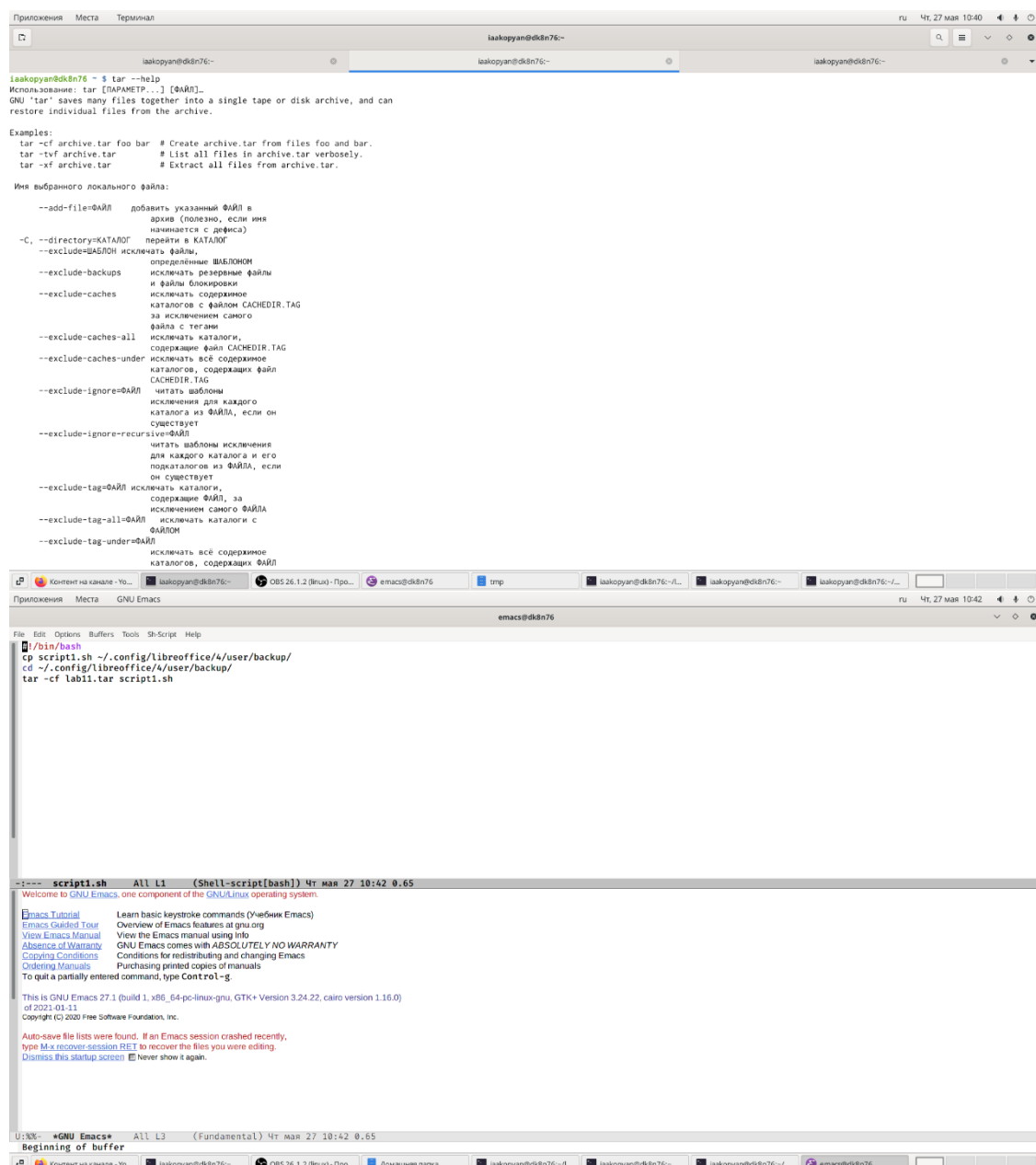
Работаю по материалам лабораторной работы №11:

[Ссылка1](#)

Далее мне пришлось сначала ознакомиться с дополнительными материалами про скрипты на Bash:

[Ссылка2](#)

Написала скрипт, который при запуске делает резервную копию самого себя в директорию backup в моем домашнем каталоге. Файл архивируется tar. Изучила справку по tar.



```
laakopyan@dkn76:~$ tar --help
Использование: tar [ПАРАМЕТР...] [ФАЙЛ]...
GNU 'tar' saves many files together into a single tape or disk archive, and can
restore individual files from the archive.

Examples:
tar -cf archive.tar foo bar    # Create archive.tar from files foo and bar.
tar -tvf archive.tar          # List all files in archive.tar verbosely.
tar -xvf archive.tar           # Extract all files from archive.tar.

Имя выбранного локального файла:

--add-file=ФАЙЛ    добавить указанный ФАЙЛ в
                   архив (полезно, если имя
                   начинается с дефиса)
-C, --directory=КАТАЛОГ    перейти в КАТАЛОГ
--exclude=ШАБЛОН    исключать файлы,
                   определенные ШАБЛОНОМ
--exclude-backups    исключать резервные файлы
                   и файлы блокировки
--exclude-caches     исключать содержимое
                   каталогов с файлом SAVEDIR.TAG
                   за исключением самого
                   файла с тегами
--exclude-caches-all  исключать каталоги,
                   содержащие файл SAVEDIR.TAG
--exclude-caches-under  исключать всё содержимое
                   каталогов, содержащих файл
                   SAVEDIR.TAG
--exclude-ignore=ФАЙЛ    читать шаблоны
                   исключения для каждого
                   каталога из ФАЙЛА, если он
                   существует
--exclude-ignore-recursive=ФАЙЛ    читать шаблоны исключения
                   для каждого каталога и его
                   подкаталогов из ФАЙЛА, если
                   он существует
--exclude-tag=ФАЙЛ    исключать каталоги,
                   содержащие ФАЙЛ, за
                   исключением самого ФАЙЛА
--exclude-tag-all=ФАЙЛ    исключать каталоги с
                   ФАЙЛОМ
--exclude-tag-under=ФАЙЛ    исключать всё содержимое
                   каталогов, содержащих ФАЙЛ
```

```
U:~%~ - *GNU Emacs* - All L3 (Fundamental) Чт мая 27 10:42 0.65
Beginning of buffer

Welcome to GNU Emacs, one component of the GNU/Linux operating system.

[Emacs Tutorial] Learn basic keystroke commands (Учебник Emacs)
[Emacs Guided Tour] Overview of Emacs features at gnu.org
[View Emacs Manual] View the Emacs manual using Info
[Absence of Warranty] GNU Emacs comes with ABSOLUTELY NO WARRANTY
[Copying Conditions] Conditions for redistributing and changing Emacs
[Ordering Manuals] Purchasing printed copies of manuals
To quit a partially entered command, type Control-g.

This is GNU Emacs 27.1 (build 1, x86_64-pc-linux-gnu, GTK+ Version 3.24.22, cairo version 1.16.0)
of 2021-01-11
Copyright (C) 2020 Free Software Foundation, Inc.

Auto-save file lists were found. If an Emacs session crashed recently,
type M-x recover-session RET to recover the files you were editing.
Dismiss this startup screen [N] Never show it again.
```

Написала пример командного файла, обрабатывающего любое произвольное число аргументов командной строки.



Написала командный файл — аналог команды ls (без использования самой этой команды и команды dir). Он выдает информацию о каталогах и выводит информацию о возможностях доступа к файлам этого каталога.

```
File Edit Options Buffers Tools Sh-Script Help
emacs@dkn76

for A in *
do if test -d $A
then echo $A: is a directory
else echo -n $A: is a file and
if test -w $A
then echo writeable
elif test -r $A
then echo readable
else echo neither readable nor writeable
fi
done
```

U:~% - *GNU Emacs* All L3 (Fundamental) Чт мар 27 10:44 0.56

Написала командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и который показывает список этих файлов. Путь к директории передала в виде аргумента командной строки.

```
File Edit Options Buffers Tools Sh-Script Help
emacs@dkn76

/bin/bash
format=""
direct=""
echo "Введите формат"
read format
echo "Введите директорию"
read direct
cd "$direct"
ls | grep ".$format"
```

U:~% - *GNU Emacs* All L3 (Fundamental) Чт мар 27 10:44 0.28

Прочее

Создание и открытие для редактирования файлов:

```

iaakopyan@dk8n76 ~ $ touch script1.sh
iaakopyan@dk8n76 ~ $ emacs script1.sh
iaakopyan@dk8n76 ~ $ touch script2.sh
iaakopyan@dk8n76 ~ $ emacs script2.sh
iaakopyan@dk8n76 ~ $ touch script3.sh
iaakopyan@dk8n76 ~ $ emacs script3.sh
iaakopyan@dk8n76 ~ $ touch script4.sh
iaakopyan@dk8n76 ~ $ emacs script4.sh

```

Вызов командных файлов, скрипта:

```

Приложения Места Терминал
iaakopyan@dk8n76:~
iaakopyan@dk8n76:~
iaakopyan@dk8n76:~
iaakopyan@dk8n76 ~ $ chmod +x script1.sh
iaakopyan@dk8n76 ~ $ ./script1.sh
iaakopyan@dk8n76 ~ $ cd ~/config/libreoffice/4/user/backup
iaakopyan@dk8n76 ~/config/libreoffice/4/user/backup $ ls
lab01.tar script1.sh
iaakopyan@dk8n76 ~/config/libreoffice/4/user/backup $ chmod +x script2.sh
chmod: невозможно получить доступ к 'script2.sh': Нет такого файла или каталога
iaakopyan@dk8n76 ~/config/libreoffice/4/user/backup $ cd
iaakopyan@dk8n76 ~ $ chmod +x script2.sh
iaakopyan@dk8n76 ~ $ ./script2.sh
Выведенное число:
iaakopyan@dk8n76 ~ $ ./script2.sh 1 2 3 4 5 6 7 8 9 10 11
Выведенное число: 1 2 3 4 5 6 7 8 9 10 11
iaakopyan@dk8n76 ~ $ ./script2.sh
Введите число:
12 3 5 4 3 4 5 6 7 8 9 1 12
12 3 5 4 3 4 5 6 7 8 9 1 12
iaakopyan@dk8n76 ~ $ chmod +x script3.sh
iaakopyan@dk8n76 ~ $ ./script3.sh
0: is a file andwriteable
0.txt: is a file andwriteable
1: is a file andwriteable
321: is a file andwriteable
abc1: is a file andwriteable
abcv: is a directory
absd: is a file andwriteable
absd.asm: is a file andwriteable
absd.lst: is a file andwriteable
absd.map: is a file andwriteable
addition.txt: is a directory
asd: is a file andwriteable
asd-: is a file andwriteable
asdfg: is a file andwriteable
asdfg.asm: is a file andwriteable
asdfg.o: is a file andwriteable
conf.txt: is a file andwriteable
file.txt: is a file andwriteable
GNUstep: is a directory
#konfig28: is a file andwriteable
konfig28-: is a file andwriteable
lab01-1.asm: is a file andwriteable
lab03-1.asm: is a file andwriteable
lab5.asm: is a file andwriteable
#lab-7.sh#: is a file andwriteable
lab-7.sh: is a file andwriteable
laboratory: is a directory
iaakopyan@dk8n76 ~ $ chmod +x script4.sh
iaakopyan@dk8n76 ~ $ ./script4.sh
Введите формат
pdf
Введите директорию
/afs/.dk.sci.pfu.edu.ru/home/i/a/iaakopyan/laboratory/lab03/шаблон/report
report.pdf
iaakopyan@dk8n76 ~ $ ./script4.sh
Введите формат
sh
Введите директорию
/afs/.dk.sci.pfu.edu.ru/home/i/a/iaakopyan
#lab-7.sh#
lab-7.sh
script1.sh
script1.sh~
script2.sh
script2.sh~
#script3.sh#
script3.sh
script3.sh~
script4.sh
script4.sh~
script.sh~
iaakopyan@dk8n76 ~ $ ./script4.sh
Введите формат
jpg
Введите директорию
/afs/.dk.sci.pfu.edu.ru/home/i/a/iaakopyan/work/os/lab06/lab/image
1.jpg
3.jpg
iaakopyan@dk8n76 ~ $

```

Вывод:

Я успешно изучила основы программирования в оболочке ОС UNIX/Linux. Научилась писать небольшие командные файлы.

Контрольные вопросы:

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются?

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

- оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
- C-оболочка (или csh) — надстройка на оболочкой Борна, использующая Сподобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).

2. Что такое POSIX?

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.

3. Как определяются переменные и массивы в языке программирования bash?

4. Каково назначение операторов let и read?

Команда let является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению.

Команда read позволяет читать значения переменных со стандартного ввода:

```
echo "Please enter Month and Day of Birth ?"
```

```
read mon day trash
```

5. Какие арифметические операции можно применять в языке программирования bash?

Для большинства команд используются следующие основания систем исчисления: 2 (двоичная), 8 (восьмеричная) и 16 (шестнадцатеричная). Простейшими математическими выражениями являются сложение (+), вычитание (-), умножение (*), целочисленное деление (/) и целочисленный остаток от деления (%).

6. Что означает операция (())?

Для облегчения программирования можно записывать условия оболочки bash в двойные скобки — (()).

7. Какие стандартные имена переменных Вам известны?

– HOME — имя домашнего каталога пользователя. Если команда `cd` вводится без аргументов, то происходит переход в каталог, указанный в этой переменной.

– IFS — последовательность символов, являющихся разделителями в командной строке, например, пробел, табуляция и перевод строки (new line).

– MAIL — командный процессор каждый раз перед выводом на экран промптера проверяет содержимое файла, имя которого указано в этой переменной, и если содержимое этого файла изменилось с момента последнего ввода из него, то перед тем как вывести на терминал промптер, командный процессор выводит на терминал сообщение You have mail (у Вас есть почта).

– TERM — тип используемого терминала.

– LOGNAME — содержит регистрационное имя пользователя, которое устанавливается автоматически при входе в систему

8. Что такое метасимволы?

Такие символы, как ' < > * ? | \ " &, являются метасимволами и имеют для командного процессора специальный смысл

9. Как экранировать метасимволы?

Снятие специального смысла с метасимвола называется экранированием метасимвола. Экранирование может быть осуществлено с помощью предшествующего метасимволу символа `\`, который, в свою очередь, является метасимволом.

10. Как создавать и запускать командные файлы?

`touch` файл

`bash` файл [аргументы]

`./файл`

11. Как определяются функции в языке программирования bash?

Группу команд можно объединить в функцию. Для этого существует ключевое слово `function`, после которого следует имя функции и список команд, заключённых в фигурные скобки. Удалить функцию можно с помощью команды `unset` с флагом `-f`.

12. Каким образом можно выяснить, является файл каталогом или обычным файлом?

`ls -lrt` Если есть `d`, то является файл каталогом

13. Каково назначение команд `set`, `typeset` и `unset`?

Оболочка `bash` позволяет работать с массивами. Для создания массива используется команда `set` с флагом `-A`. За флагом следует имя переменной, а затем список значений, разделённых пробелами.

Если использовать `typeset -i` для объявления и присвоения переменной, то при последующем её применении она станет целой. Также можно использовать ключевое слово `integer` (псевдоним для `typeset -i`) и объявлять таким образом переменные целыми. Выражения типа `x=y+z` будет восприниматься в это случае как арифметические.

Изъять переменную из программы можно с помощью команды `unset`

14. Как передаются параметры в командные файлы?

Символ `$` является метасимволом командного процессора. Он используется, в частности, для ссылки на параметры, точнее, для получения их значений в командном файле. В командный файл можно передать до девяти параметров. При использовании где-либо в командном файле комбинации символов `$i`, где $0 < i < 10$, вместо неё будет осуществлена подстановка значения параметра с порядковым номером i , т.е. аргумента командного файла с порядковым номером i . Использование комбинации символов `$0` приводит к подстановке вместо неё имени данного командного файла

15. Назовите специальные переменные языка `bash` и их назначение.

- `$*` — отображается вся командная строка или параметры оболочки;
- `$?` — код завершения последней выполненной команды;
- `$$` — уникальный идентификатор процесса, в рамках которого выполняется командный процессор;
- `$!` — номер процесса, в рамках которого выполняется последняя вызванная на выполнение в командном режиме команда;
- `$-` — значение флагов командного процессора;
- `${#*}` — возвращает целое число — количество слов, которые были результатом `$*`;
- `${#name}` — возвращает целое значение длины строки в переменной `name`;
- `${name[n]}` — обращение к n -му элементу массива;
- `${name[*]}` — перечисляет все элементы массива, разделённые пробелом;
- `${name[@]}` — то же самое, но позволяет учитывать символы пробелы в самих переменных;
- `${name:-value}` — если значение переменной `name` не определено, то оно будет заменено на указанное `value`;
- `${name:value}` — проверяется факт существования переменной;
- `${name=value}` — если `name` не определено, то ему присваивается значение `value`;
- `${name?value}` — останавливает выполнение, если имя переменной не определено, и выводит `value` как сообщение об ошибке;
- `${name+value}` — это выражение работает противоположно `${name-value}`. Если переменная определена, то подставляется `value`;
- `${name#pattern}` — представляет значение переменной `name` с удалённым самым коротким левым образцом (`pattern`);
- `${#name[*]}` и `${#name[@]}` — эти выражения возвращают количество элементов в массиве `name`.