

Лабораторная работа №10

Операционные системы

Сабралиева Марворид Нуралиевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	11
4	Контрольные вопросы	12

Список иллюстраций

2.1	man tar	6
2.2	Создаем скрипт	7
2.3	скрипт	7
2.4	заполняем скрипт	7
2.5	скрипт	8
2.6	создаем и заполняем скрипт	8
2.7	скрипт	9
2.8	задание 3	9
2.9	задание 4	10

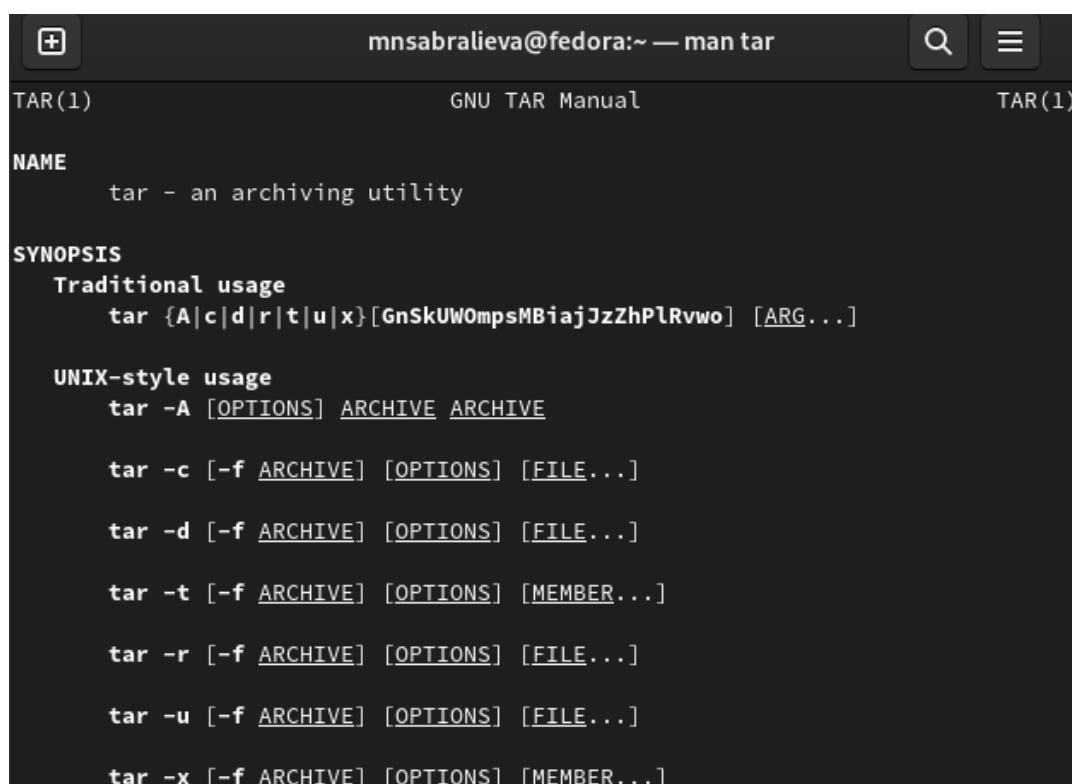
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

1. Написать скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации необходимо узнать, изучив справку.



```
mnsabralieva@fedora:~ — man tar
TAR(1)                                GNU TAR Manual                                TAR(1)

NAME
    tar - an archiving utility

SYNOPSIS
    Traditional usage
        tar {A|c|d|r|t|u|x}[GnSkUWompsMBiajJzZhPlRvwo] [ARG...]

    UNIX-style usage
        tar -A [OPTIONS] ARCHIVE ARCHIVE

        tar -c [-f ARCHIVE] [OPTIONS] [FILE...]

        tar -d [-f ARCHIVE] [OPTIONS] [FILE...]

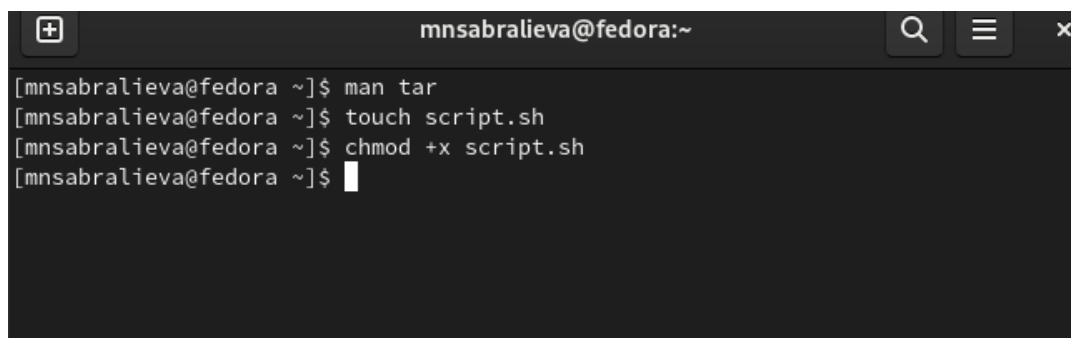
        tar -t [-f ARCHIVE] [OPTIONS] [MEMBER...]

        tar -r [-f ARCHIVE] [OPTIONS] [FILE...]

        tar -u [-f ARCHIVE] [OPTIONS] [FILE...]

        tar -x [-f ARCHIVE] [OPTIONS] [MEMBER...]
```

Рис. 2.1: man tar



```
mnsabralieva@fedora:~  
[mnsabralieva@fedora ~]$ man tar  
[mnsabralieva@fedora ~]$ touch script.sh  
[mnsabralieva@fedora ~]$ chmod +x script.sh  
[mnsabralieva@fedora ~]$
```

Рис. 2.2: Создаем скрипт



Рис. 2.3: скрипт



Рис. 2.4: заполняем скрипт

```
[mnsabralieva@fedora ~]$ ./script.sh
[mnsabralieva@fedora ~]$ ls ~/backup/
backup.sh.gz
[mnsabralieva@fedora ~]$
```

Рис. 2.5: скрипт

2. Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов. `for i` - для всех переданных аргументов `do echo $1` - выводим первый аргумент `shift`- удаляем первый аргумент, смещаем все аргументы `done`- конец цикла



```
[mnsabralieva@fedora ~]$ touch script2.sh
[mnsabralieva@fedora ~]$ chmod +x script2.sh
[mnsabralieva@fedora ~]$
```

Открыть ▾ + script2.sh ~/

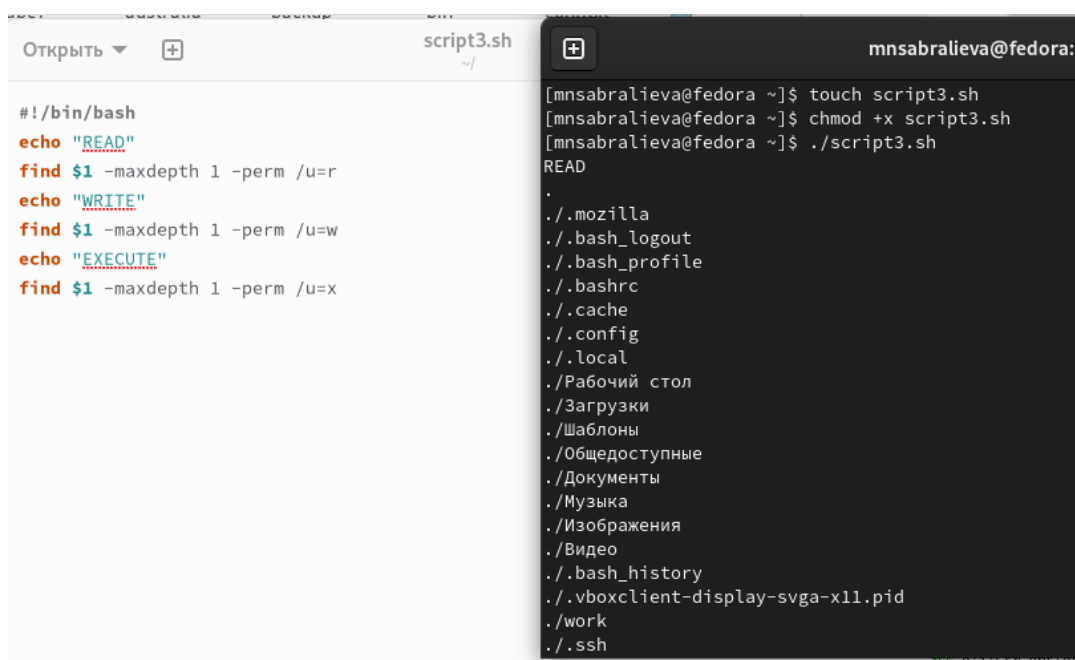
```
#!/bin/bash
for i
do echo $1
shift
done
```

Рис. 2.6: создаем и заполняем скрипт


```
[mnsabralieva@fedora ~]$ ./script2.sh 1 2 3 4
1
2
3
4
[mnsabralieva@fedora ~]$
```

Рис. 2.7: скрипт

3. Написать командный файл — аналог команды `ls` (без использования самой этой ко- манды и команды `dir`). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.

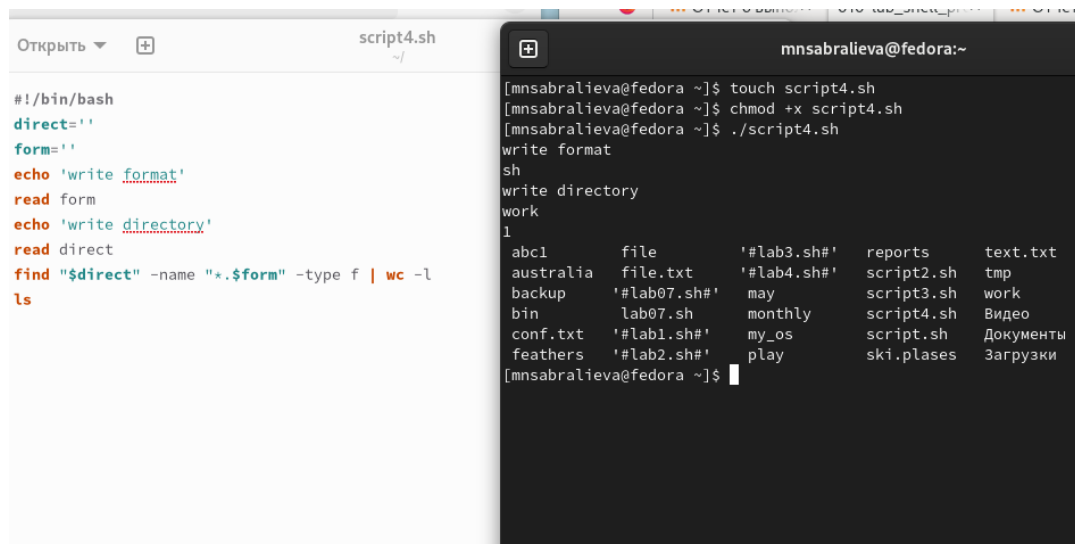


```
#!/bin/bash
echo "READ"
find $1 -maxdepth 1 -perm /u=r
echo "WRITE"
find $1 -maxdepth 1 -perm /u=w
echo "EXECUTE"
find $1 -maxdepth 1 -perm /u=x

[mnsabralieva@fedora ~]$ touch script3.sh
[mnsabralieva@fedora ~]$ chmod +x script3.sh
[mnsabralieva@fedora ~]$ ./script3.sh
READ
.
./.mozilla
./.bash_logout
./.bash_profile
./.bashrc
./.cache
./.config
./.local
./Рабочий стол
./Загрузки
./Шаблоны
./Общедоступные
./Документы
./Музыка
./Изображения
./Видео
./.bash_history
./.vboxclient-display-svga-x11.pid
./work
./.ssh
```

Рис. 2.8: задание 3

4. Написать командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки.



```
#!/bin/bash
direct=''
form=''
echo 'write format'
read form
echo 'write directory'
read direct
find "$direct" -name "*.$form" -type f | wc -l
ls
```

```
[mnsabralieva@fedora ~]$ touch script4.sh
[mnsabralieva@fedora ~]$ chmod +x script4.sh
[mnsabralieva@fedora ~]$ ./script4.sh
write format
sh
write directory
work
1
abc1      file      '#lab3.sh#'  reports  text.txt
australia file.txt   '#lab4.sh#'  script2.sh tmp
backup    '#lab07.sh#' may        script3.sh work
bin        lab07.sh   monthly    script4.sh Видео
conf.txt   '#lab1.sh#' my_os      script.sh  Документы
feathers    '#lab2.sh#' play       ski.places Загрузки
[mnsabralieva@fedora ~]$
```

Рис. 2.9: задание 4

3 Выводы

Мы изучили основы программирования в оболочке ОС UNIX/Linux. Научились писать небольшие командные файлы.

4 Контрольные вопросы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются? Ответ:
 - a) sh — стандартная командная оболочка UNIX/Linux, содержащая базовый, полный набор функций
 - b) csh — использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд
 - c) ksh — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна
 - d) bash — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна
2. Что такое POSIX? Ответ: 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? Ответ: Переменные вызываются \$var, где var=чему-то, указанному пользователем, неважно что бы то не было, название файла, каталога или еще чего. Для массивов используется команда set -A
4. Каково назначение операторов let и read? Ответ: let — вычисляет далее заданное математическое значение read — позволяет читать значения переменных со стандартного ввода
5. Какие арифметические операции можно применять в языке программирования?

- вания `bash`? Ответ: Прибавление, умножение, вычисление, деление), сравнение значений, экспонирование и др.
6. Что означает операция `(())`? Ответ: Это обозначение используется для облегчения программирования для условий `bash`
 7. Какие стандартные имена переменных Вам известны? Ответ: Нам известны `HOME`, `PATH`, `BASH`, `ENV`, `PWD`, `UID`, `OLDPWD`, `PPID`, `GROUPS`, `OSTYPE`, `PS1` - `PS4`, `LANG`, `HOSTFILE`, `MAIL`, `TERM`, `LOGNAME`, `USERNAME`, `IFS` и др.
 8. Что такое метасимволы? Ответ: Метасимволы это специальные знаки, которые могут использоваться для сокращения пути, поиска объекта по расширению, перед переменными, например «\$» или «*» .
 9. Как экранировать метасимволы? Ответ: Добавить перед метасимволом метасимвол «\»
 10. Как создавать и запускать командные файлы? Ответ: При помощи команды `chmod`. Надо дать права на запуск `chmod +x название файла`, затем запустить `bash ./название файла` Например у нас файл `lab` Пишем: `chmod +x lab ./lab`
 11. Как определяются функции в языке программирования `bash`? Ответ: Объединяя несколько команд с помощью `function`
 12. Каким образом можно выяснить, является файл каталогом или обычным файлом? Ответ: Можно задать команду на проверку директория ли это `test -d директория`
 13. Каково назначение команд `set`, `typeset` и `unset`? Ответ: `Set` — используется для создания массивов `Unset` — используется для изъятия переменной `Typeset` — используется для присваивания каких-либо функций
 14. Как передаются параметры в командные файлы? Ответ: Добавлением аргументов после команды запуска `bash` скрипта
 15. Назовите специальные переменные языка `bash` и их назначение. Ответ: `-$*` — отображается вся командная строка или параметры оболочки; `-$?` — код завершения последней выполненной команды; `-$$_code> — уникальный идентификатор процесса, в рамках которого выполняется -$! — номер процесса, в рамках которого выполняется последняя вызванная -$- — значение флагов`

командного процессора; # Список литературы{unnumbered}