# **лабораторная работа №5**

Лабораторная работа состоит из двух частей^

* Лб. 5.1 посвящена изучению основных команд и операторов управления интерпретатора BASH и созданию на их основе скрипт-файлов.
* Лб. 5.2 посвящена изучению основные системные вызовы и функции в ОС UNIX/Linux для работы с файлами и каталогами.

**Лабораторная работа 5.1**

**5.1.1 Команды системы**

Операционная система ОС ***Linux*** создана на основе ОС ***UNIX*** и во многом имеет схожую структуру и систему команд. Пользователь может работать в текстовом режиме с помощью командной строки, или с использованием графического интерфейса ***X Window*** и одного из менеджеров рабочего стола (например, ***KDE*** или ***GNOME***). Причем, одновременно в системе могут работать 7 пользователей (6- в текстовом режиме консоли и 1 – в графическом режиме), переключение между пользователями осуществляется по нажатию клавиш:

.

***Ctrl***

***Alt***

***F7***

***Ctrl***

***Alt***

***F1***

В табл. 1 приведены основные команды системы

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| Команда | Аргументы/ключи | Пример | Описание |
| ***dir*** | каталог | ***dir***  ***dir /home*** | Выводит на консоль содержимое каталога |
| ***ls*** | ***-all***  и другие (см. man) | ***ls -all*** | Выводит на консоль содержимое каталога |
| ***ps*** | ***-a***  ***-x*** и другие (см. man) | ***ps -a*** | Выводит на консоль список процессов |
| ***mkdir*** | имя каталога | ***mkdir stud11*** | Создает каталог |
| ***rmdir*** | имя каталога | ***rmdir stud11*** | Удаляет каталог |
| ***rm*** | файл | ***rm myfile1*** | Удаляет файл |
| ***touch*** | имя файла | ***touch 1.txt*** | Создает файл |
| ***mv*** | файл новое имя | ***mv myfile1 myf1*** | Переименование файла |
| ***cat*** | файл | ***cat 1.txt*** | Вывод файла на консоль |
| ***cd*** | имя каталога | ***cd home*** | Переход по каталогам |
| ***grep*** | (см. man) | ***grep "^a" "words.txt"*** | Поиск строки в файле |
| ***kill*** | ***pid*** процесса | ***kill 12045*** | Уничтожает процесс |
| ***top*** |  |  | Выводит на консоль список процессов |
| ***htop*** |  |  | Выводит на консоль полный список запущенных процессов |
| ***su*** |  |  | Переход в режим root |
| ***chmod*** | права доступа файл | ***chmod 777 1.txt*** | Изменение прав доступа к файлам |
| ***mount*** | устройство каталог | ***mount /dev/cdrom /MyCD*** | Монтирование устройств |
| ***dd*** | ***if=***файл ***of=***файл ***bs=n count=n*** | ***dd if=/dev/hda1***  ***of=/F.bin bs=512 count=1*** | Копирование побайтное |
| ***ln*** | файл1 файл2  ***-l*** | ***ln файл1 файл2***  ***ln –l файл1 файл2*** | Создать жёсткую или символическую ссылку на файл |
| ***uname*** | ***-a*** | ***uname –a*** | Информация о системе |
| ***find*** | dir файл | ***find /home –name a1.txt*** | Поиск файлов |
| ***man*** |  | ***man fgetc*** | Справка по системе |
| ***info*** |  | ***info fgetc*** | Справка по системе |

Linux и Windows используют различные файловые системы для хранения и организации доступа к информации на дисках. В Linux используются файловые системы- ***Ext2/Ext3***, ***RaiserFS, FFS*** и другие. Все файловые системы имеют поддержку ***журналирования***. ***Журналируемая*** файловая система сначала записывает изменения, которые она будет проводить в отдельную часть файловой системы (***журнал***) и только потом вносит необходимые изменения в остальную часть файловой системы. После удачного выполнения всех транзакций, записи удаляются из ***журнала***. Это обеспечивает лучшее сохранение целостности системы и уменьшает вероятность потери данных. Следует отметить, что ***Linux*** поддерживает доступ к ***Windows***-разделам.

Файловая система ***Linux*** имеет лишь один корневой каталог, который обозначается косой чертой ( ***/*** ). В файловой структуре ***Linux*** нет дисков ***A, B, C, D*** …., а есть только каталоги. В ***Linux*** различаются прописные и строчные буквы в командах, именах файлов и каталогов. В ***Windows*** у каждого файла существует лишь одно имя, в ***Linux*** их может быть много. Это – «***жесткие***» ссылки, которые указывают непосредственно на индексный дескриптор файла. Жесткая ссылка – это один из принципов организации файловой системы ***Linux***.

Структура каталогов ОС ***Linux*** представлена в табл. 1. Есть также несколько полезных сокращений для имен каталогов:

* Одиночная точка (**.**) обозначает текущий рабочий каталог.
* Две точки (**..**) обозначают родительский каталог текущего рабочего.
* Тильда (**~**) обозначает домашний каталог пользователя (обычно это каталог, который является текущим рабочим при запуске Bash).

Таблица 1

|  |  |
| --- | --- |
| ***/*** | Корневой каталог |
| ***/bin*** | Содержит исполняемые файлы самых необходимых для работы системы программ. Каталог ***/bin*** не содержит подкаталогов. |
| ***/boot*** | Здесь находятся само ядро системы (файл ***vmlinuz***-...) и файлы, необходимые для его загрузки. |
| ***/dev*** | Каталог ***/dev*** содержит файлы устройств (драйверы). |
| ***/etc*** | Это каталог конфигурационных файлов, т. е. файлов, содержащих информацию о настройках системы (например, настройки программ). |
| ***/home*** | Содержит домашние каталоги пользователей системы. |
| ***/lib*** | Здесь находятся библиотеки (функции, необходимые многим программам). |
| ***/media*** | Содержит подкаталоги, которые используются как точки монтирования для сменных устройств (CD-ROM'ов, floppy-дисков и др.) |
| ***/mnt*** | Данный каталог (или его подкаталоги) может служить точкой монтирования для временно подключаемых файловых систем. |
| ***/proc*** | Содержит файлы с информацией о выполняющихся в системе процессах. |
| ***/root*** | Это домашний каталог администратора системы. |
| ***/sbin*** | Содержит исполняемые программы, как и каталог ***/bin***. Однако использовать программы, находящиеся в этом каталоге может только администратор системы (***root***). |
| ***/tmp*** | Каталог для временных файлов, хранящих промежуточные данные, необходимых для работы тех или иных программ, и удаляющиеся после завершения работы программ. |
| ***/usr*** | Каталог для большинства программ, которые не имеют значения для загрузки системы. Структура этого каталога фактически дублирует структуру корневого каталога. |
| ***/var*** | Содержит данные, которые были получены в процессе работы одних программ и должны быть переданы другим, и файлы журналов со сведениями о работе системы. |

**5.1.2 Интерпретатор Bash**

***Bash*** - это ***shell***-совместимый интерпретатор командного языка, выполняющий команды, прочитанные со стандартного входного потока или из файла. ***Скрипт-файл*** – это обычный текстовый файл, содержащий последовательность команд ***bash***, для которого установлены права на выполнение. Пример скрипта, выводящего содержимое текущего каталога на консоль (dir) и в файл (dir > 1.txt). Файл 1.txt должен быть предварительно создан, например, с помощью команды ***touch 1.txt*** ::

***#!/bin/bash***

***echo “My script”***

***dir***

***dir > 1.txt***

Первая строка — решётка, за которой следует восклицательный знак (эту последовательность называют [шебанг](https://ru.wikipedia.org/wiki/%D0%A8%D0%B5%D0%B1%D0%B0%D0%BD%D0%B3_(Unix))) и путь к bash, указывают системе на то, что сценарий создан именно для интерпретатора bash.

Команда ***echo “My script”*** выводит на экран текст *My script*

Для создания скрипта надо открыть редактор теста, например gedit набрать текст и сохранить, например, под именем *myscript.*

Сделаем файл myscript исполняемым.

***chmod +x ./myscript***

Запускаем скрипт на выполнение

***./myscript***

Командному файлу можно передать аргументы, которые записываются после имени командного файла при его запуске. Следующие переменные используются командным интерпретатором.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***$0,$1,$2,$3…*** | | | Значения аргументов командной строки при запуске скрипта. Где ***$0***-имя самого файла скрипта, ***$1***- первый аргумент, ***$2***- второй аргумент, и т.д. | |
| ***$@*** | | | Все аргументы командной строки, каждый в кавычках | |
| ***$?*** | | Код возврата последней команды | | | |
|  |  | | |

Пример простого скрипта, выводящего на консоль и в файл содержимое каталога ***home/stud*** , где имя каталога передаётся скрипту в качестве аргументов при запуске:

Запуск скрипта: ***>./mydir*** ***/home/stud***

Скрипт:

***#!/bin/bash***

***dir $1***

***dir $1 > 1.txt***

Можно создать собственную переменную и присвоить ей значение:

***A=121***

***A=”121”***

***let A=121***

***let “A=А+1”***

Вывод значения на консоль: ***echo $A***

Проверка условия: ***test[expr]***

где ***expr***: а) для строк: ***S1 = S2*** ***S1***содержит ***S2***

***S1 != S2******S1***не содержит ***S2***

***-n S1*** если длина ***S1 >0***

***-z S1*** если длина ***S1 =0***

б) целые ***i1***и ***i2***

***i1 – ge i2***

***i1 – gt i2***

***i1 – ie i2***

***i1 – et i2***

***i1 – nt i2***

в) файлы

***-d name\_file*** является ли файл каталогом

***-f name\_file*** является ли файл обычным файлом

***-r name\_file*** доступен ли файл для чтения

***-s name\_file*** имеет ли файл ненулевую длину

***-w name\_file*** доступен ли файл для записи

***-x name\_file*** является ли файл исполняемым

г) логически операции

***!exp*** логическое отрицание (не)

***exp1 –a exp2*** умножение условий (и)

***exp1 –o exp2*** сложение условий (или)

Проверка условия: ***if [expr ]***

***then com 1*** Если условие ***expr=true*** то команда

***… com 1… com n***

***com n***

***(elif expr2***

***com1***

***…***

***com n***

***)***

***else***

***com 1***

***…***

***com n***

***fi***

Проверка нескольких условий: ***case string1 in***

***str 1)***

***com 1***

***…***

***com n***

***; ;***

***str 2)***

***com 1***

***…***

***com n***

***; ;***

***str 3)***

***com 1***

***…***

***com n***

***; ;***

***\*)***  // default

***com 1***

***…***

***com n***

***; ;***

***esac***

Функция пользователя: ***fname2 (arg1,arg2...argN)***

***{  
commands  
}***

Организация циклов:

1. ***for var1 in list***

***do***

***com1***

***…***

***com n***

***done***

2. ***while exp***

***com1***

***…***

***com n***

***end***

3. ***until exp // аналог do-while***

***do***

***com1***

***…***

***com n***

***done***

Полезные ссылки

<https://tldp.org/LDP/abs/html/index.html>

<https://habr.com/ru/company/ruvds/blog/325522/>

<https://selectel.ru/blog/tutorials/linux-bash-scripting-guide/>

**5.2 Работа с файлами и каталогами в ОС Linux**

Функция ***int main( int argc , char \*argv[ ] [, char \*envp[ ] ] );***

Данное объявление позволяет удобно передавать аргументы командной строки и переменные окружения. Определение аргументов:

***argc*** - количество аргументов, которые содержатся в ***argv[]*** (всегда больше либо равен 1);

***argv*** - в массиве строки представляют собой параметры из командной строки, введенные пользователем программы. По соглашению, **argv [0] –** это команда, которой была запущена программа, **argv[1]** – первый параметр из командной строки и так далее до argv[argc] – элемент, всегда равный NULL;

***envp*** - это массив строк, которые представляют собой переменные окружения. Массив заканчивается значением ***NULL***.

Для выполнения операций записи и чтения данных в существующем файле его следует открыть при помощи вызова ***open ()****.*

***int open (const char \*pathname, int flags, [mode\_t mode]);***

***int fopen (const char \*pathname, int flags, [mode\_t mode]);***

Второй аргумент системного вызова ***open*** *-* ***flag****s -* имеет целочисленный тип и определяет метод доступа. Параметр ***flags***принимает одно из значений, заданных постоянными в заголовочном файле ***fcnt1.h***. В файле определены три постоянных:

***O\_RDONLY*** – открыть файл только для чтения,

***O\_WRONLY*** – открыть файл только для записи,

***O\_RDWR*** – открыть файл для чтения и записи,

или ***“r”***, ***“w”***, ***“rw”*** для ***fopen().***

Третий параметр ***mode*** устанавливает права доступа к файлуи является необязательным, он используется только вместе с флагом ***O\_CREAT***. Пример создания нового файла:

***# include <sys / types.h>***

***# include <sys / stat.h>***

***# include <fcnt1.h>***

***int Fd1;***

***FILE \*F1;***

***F1=fopen (“Myfile2.txt”, “w”, 644);***

***Fd1=open (“Myfile1.txt”, O\_CREAT, 644);***

Системные вызовы ***sta****t* и ***fstat*** позволяют процессу определить значения свойств в существующем файле.

***#include <sys/types.h>***

***#include <sys/stat.h>***

***int stat (const char \*pathname, struct stat \*buf);***

***int fstat (int filedes, struct stat \*buf);***

Пример: ***stat(“1.exe”, &st1);***

Где ***pathname*** – полное имя файла, ***buf*** –структура типа ***stat***. Эта структура после успешного вызова будет содержать связанную с файлом информацию.

Поля структуры stat включает следующие элементы:

***struct stat {***

***dev\_t st\_dev; /\**** логическое устройство, где находится файл ***\*/***

***ino\_t st\_ino; /\**** номер индексного дескриптора ***\*/***

***mode\_t st\_mode; /\**** права доступа к файлу ***\*/***

***nlink\_t st\_nlink; /\**** количество жестких ссылок на файл ***\*/***

***uid\_t st\_uid; /\**** ID пользователя-владельца ***\*/***

***gid\_t st\_gid; /\**** ID группы-владельца ***\*/***

***dev\_t st\_rdev; /\**** тип устройства ***\*/***

***off\_t st\_size; /\**** общий размер в байтах ***\*/***

***unsigned long st\_blksize; /\**** размер блока ввода-вывода ***\*/***

***unsigned long st\_blocks; /\**** число блоков, занимаемых файлом ***\*/***

***time\_t st\_atime; /\**** время последнего доступа ***\*/***

***time\_t st\_mtime; /\**** время последней модификации ***\*/***

***time\_t st\_ctime; /\**** время последнего изменения ***\*/***

***};***

Права доступа в ***Linux***. Права доступа к файлам представлены в виде последовательности бит, где каждый бит означает разрешение на запись (***w***), чтение (***r***) или выполнение (***x***). Права доступа записываются для владельца-создателя файла (***owner***); группы, к которой принадлежит владелец–создатель файла (***group***); и всех остальных (***other***). Например, при выводе команды ***dir*** запись типа:

-***rwx r-x r-w***  ***1.exe***

означает, что владелец файла ***1.exe*** имеет права на чтение, запись и выполнение, группа имеет права только на чтение и выполнение, все остальные имеют права только на чтение. В восьмеричном виде получится значение ***0754***. В действительности манипулирует файлами не сам пользователь, а запущенный им процесс. Для просмотра прав доступа можно использовать функцию ***stat***.

Для записи прав доступа служит функция ***chmod***:

***#include <sys/types.h>***

***#include <sys/stat.h>***

***int chmod(const char \*pathname, mode\_t mode);***

Пример: ***chmod(“1.exe”, 0777);***

Каталоги в ОС ***Linux*** –это особые файлы. Для открытия или закрытия каталогов существуют вызовы:

***#include <dirent.h>***

***DIR \*opendir (const char \*dirname);***

***int closedir( DIR \*dirptr);***

Для работы с каталогами существуют системные вызовы:

***int mkdir (const char \*pathname, mode\_t mode)*** – создание нового каталога,

***int rmdir(const char \*pathname)*** – удаление каталога. Первый параметр – имя создаваемого каталога, второй – права доступа:

***retval=mkdir(“/home/s1/t12/alex”,0777);***

***retval=rmdir(“/home/s1/t12/alex”);***

Заметим, что вызов ***rmdir(“/home/s1/t12/alex”)*** будет успешен, только если удаляемый каталог пуст, т.е. содержит записи “точка” (.) и “двойная точка” (..). Для чтения записей каталога существует вызов:

***struct dirent \*readdir(DIR \*dirptr);***

Структура dirent такова: ***struct dirent {***

***long d\_ino;***

***off\_t d\_off;***

***unsigned short d\_reclen;***

***char d\_name [1];***

***};***

Поле ***d\_ino*** - это число, которое уникально для каждого файла в файловой системе. Значением поля ***d\_off*** служит смещение данного элемента в реальном каталоге. Поле ***d\_name*** есть начало массива символов, задающего имя элемента каталога. Данное имя ограничено нулевым байтом и может содержать не более ***MAXNAMLEN*** символов. Тем самым описываемая структура имеет переменную длину, хранящуюся в поле ***d\_reclen***.

Пример вызова:

***DIR \*dp;***

***struct dirent \*d;***

***d=readdir(dp);***

При первом вызове функция ***readdir*** в структуру ***dirent*** будет считана первая запись каталога. После прочтения всего каталога в результате последующих вызовов ***readdir*** будет возвращено значение ***NULL***. Для возврата указателя в начало каталога на первую запись существует вызов:

***void rewindir(DIR \*dirptr);***

Чтобы получить имя текущего рабочего каталога существует функция:

***char \*getcwd(char \*name, size\_t size);***

Время в ***Linux*** отсчитывается в секундах, прошедшее с начала этой эпохи (***00:00:00 UTC, 1 Января 1970 года***). Для получения системного времени можно использовать следующие функции:

***#include <sys/time.h>***

***time\_t time (time\_t \*tt);***

***int gettimeofday(struct timeval \*tv, struct timezone \*tz);***

***struct timeval {***

***long tv\_sec; /\* секунды \*/***

***long tv\_usec; /\* микросекунды \*/***

***};***

**ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ**

# Варианты индивидуальных заданий к лабораторной 5.1

1. Написать скрипт для поиска файлов заданного размера в заданном каталоге (имя каталога задаётся пользователем в качестве третьего аргумента командной строки). Диапазон (мин.- мах.) размеров файлов задаётся пользователем в качестве первого и второго аргумента командной строки. Вывести на консоль первые найденные 20 файлов в виде: полный путь, имя файла, его размер.
2. Написать скрипт с использованием цикла ***for***, выводящий на консоль размеры и права доступа для всех файлов в заданном каталоге и всех его подкаталогах (имя каталога задается пользователем в качестве первого аргумента командной строки). На консоль выводится общее число просмотренных файлов.
3. Написать скрипт для поиска заданной пользователем строки во всех файлах заданного каталога и всех его подкаталогов (строка и имя каталога задаются пользователем в качестве первого и второго аргумента командной строки). На консоль выводятся полный путь и имена файлов, в содержимом которых присутствует заданная строка, и их размер. Если к какому-либо каталогу нет доступа, необходимо вывести соответствующее сообщение и продолжить выполнение.
4. Написать скрипт поиска одинаковых по содержимому файлов в двух каталогов, например, ***Dir1*** и ***Dir2***. Пользователь задаёт имена ***Dir1*** и ***Dir2*** в качестве первого и второго аргумента командной строки. В результате работы программы файлы, имеющиеся в ***Dir1***, сравниваются с файлами в ***Dir2*** по их содержимому. На экран выводятся число просмотренных файлов и результаты сравнения.
5. Написать скрипт, находящий в заданном каталоге и всех его подкаталогах все файлы, владельцем которых является заданный пользователь. Имя владельца и каталог задаются пользователем в качестве первого и второго аргумента командной строки. Скрипт выводит результаты в файл (третий аргумент командной строки) в виде полный путь, имя файла, его размер. На консоль выводится общее число просмотренных файлов.
6. Написать скрипт, находящий в заданном каталоге и всех его подкаталогах все файлы заданного размера. Диапазон (мин.- мах.) размеров файлов задаётся пользователем в качестве первого и второго аргумента командной строки. Скрипт выводит результаты поиска в файл (третий аргумент командной строки) в виде: полный путь, имя файла, его размер. На консоль выводится общее число просмотренных файлов.
7. Написать скрипт подсчитывающий суммарный размер файлов в заданном каталоге и всех его подкаталогах (имя каталога задаётся пользователем в качестве аргумента командной строки). Скрипт выводит результаты подсчёта в файл (второй аргумент командной строки) в виде каталог (полный путь), суммарный размер файлов число просмотренных файлов.
8. Написать скрипт, находящий в заданном каталоге и всех его подкаталогах все файлы заданного расширения и создающий для каждого найденного файла жесткую ссылку в заданном каталоге. Расширение файла и каталог для жестких ссылок задаются в качестве первого и второго аргумента командной строки.
9. Написать скрипт, находящий все каталоги и подкаталоги начиная с заданного каталога и ниже на заданной глубине вложенности (аргументы 1 и 2 командной строки). Скрипт выводит результаты в файл (третий аргумент командной строки) в виде полный путь, количество файлов в каталоге. На консоль выводится общее число просмотренных каталогов.

# Варианты индивидуальных заданий к дабораторной 5.2

1. Отсортировать в заданном каталоге (аргумент 1 командной строки) и во всех его подкаталогах файлы по следующим критериям (аргумент 2 командной строки, задаётся в виде целого числа):1 – по размеру файла, 2 – по имени файла. Записать без сохранения структуры каталогов отсортированные файлы общим списком, в новый каталог (аргумент 3 командной строки). В связи с индексированием файлов в каталогах для файловых систем ext 2,3,4 перед запуском программы необходимо временно отключить опцию индексирования файловой системы следующим образом:

***sudo*** ***tune2fs –O +dir\_index /dev/sdaXY***

Проверить результат, используя, ***ls -l –f.***

1. Найти в заданном каталоге (аргумент 1 командной строки) и всех его подкаталогах заданный файл (аргумент 2 командной строки). Вывести на консоль полный путь к файлу, размер, дату создания, права доступа, номер индексного дескриптора. Вывести также общее количество просмотренных каталогов и файлов.
2. Для заданного каталога (аргумент 1 командной строки) и всех его подкаталогов вывести в заданный файл (аргумент 2 командной строки) и на консоль полный путь, размер и дату создания, удовлетворяющих заданным условиям: 1 – размер файла находится в заданных пределах от ***N1*** до ***N2*** (***N1,N2*** задаются в аргументах командной строки), 2 – дата создания находится в заданных пределах от ***M1*** до ***M2*** (***M1***,***M2*** задаются в аргументах командной строки).
3. Найти совпадающие по содержимому файлы в двух заданных каталогах (аргументы 1 и 2 командной строки) и всех их подкаталогах. Вывести на консоль и в файл (аргумент 3 командной строки) полный путь, размер, дату создания, права доступа, номер индексного дескриптора.
4. Подсчитать суммарный размер файлов в заданном каталоге (аргумент 1 командной строки) и для каждого его подкаталога отдельно. Вывести на консоль и в файл (аргумент 2 командной строки) название подкаталога, количество файлов в нём, суммарный размер файлов, имя файла с наибольшим размером.
5. Написать программу, находящую в заданном каталоге и всех его подкаталогах все файлы, заданного размера. Имя каталога задаётся пользователем в качестве первого аргумента командной строки. Диапазон от ***N1*** до ***N2*** задается в аргументах командной строки. Программа выводит результаты поиска в файл (четвертый аргумент командной строки) в виде полный путь, имя файла, его размер. На консоль выводится общее число просмотренных файлов.
6. Найти все дубликаты (с одинаковым содержимым) файлов в заданном диапазоне размеров от ***N1*** до ***N2*** (***N1, N2*** задаются в аргументах командной строки), начиная с исходного каталога и ниже. Имя исходного каталога задаётся пользователем в качестве первого аргумента командной строки.
7. Подсчитать для заданного каталога (первый аргумент командной строки) и всех его подкаталогов суммарный размер занимаемого файлами на диске пространства в байтах и суммарный размер файлов. Вычислить коэффициент использования дискового пространства в %. Для получения размера занимаемого файлами на диске пространства использовать команду ***stat.***
8. Написать программу, находящую в заданном каталоге (первый аргумент командной строки) и всех его подкаталогах все файлы заданного расширения и создающий для каждого найденного файла жесткую ссылку в заданном каталоге. Расширение файла и каталог для жестких ссылок задаются в качестве второго и третьего аргументов командной строки.