# ЛАБОРАТОРНАЯ РАБОТА №2 РАБОТА С ФАЙЛАМИ И КАТАЛОГАМИ ОС UNIX

Цель работы – изучить основные системные вызовы и функции в ОС UNIX для работы с файлами и каталогами.

**Теоретическая часть**

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 \*/***

***};***

**Порядок выполнения работы**

1. Изучить теоретическую часть лабораторной работы.
2. Написать программу вывода сообщения на экран.
3. Написать программу ввода символов с клавиатуры и записи их в файл (в качестве аргумента при запуске программы вводится имя файла). Для чтения или записи файла использовать ***только*** функции посимвольного ввода-вывода ***getc(),putc()***, ***fgetc()***,***fputc()***. Предусмотреть выход после ввода определённого символа (например: ***ctrl-F***). Предусмотреть контроль ошибок открытия/закрытия/чтения файла.
4. Написать программу вывода содержимого текстового файла на экран (в качестве аргумента при запуске программы передаётся имя файла, второй аргумент (***N***) устанавливает вывод по группам строк (по ***N*** –строк) или сплошным текстом (***N***=0)). Для вывода очередной группы строк необходимо ожидать нажатия пользователем любой клавиши. Для чтения или записи файла использовать ***только*** функции посимвольного ввода-вывода ***getc(),putc()***, ***fgetc()***,***fputc()***. Предусмотреть контроль ошибок открытия/закрытия/чтения/записи файла.
5. Написать программу копирования одного файла в другой. В качестве параметров при вызове программы передаются имена первого и второго файлов. Для чтения или записи файла использовать ***только*** функции посимвольного ввода-вывода ***getc(),putc()***, ***fgetc()***,***fputc()***. Предусмотреть копирование прав доступа к файлу и контроль ошибок открытия/закрытия/чтения/записи файла.
6. Написать программу вывода на экран содержимого текущего и корневого каталогов. Предусмотреть контроль ошибок открытия/закрытия/чтения каталога.

## Варианты индивидуальных заданий

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.