Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Новосибирский государственный технический университет»

Кафедра теоретической и прикладной информатики

Лабораторная работа № 3

Структура системы управления вводом-выводом в ОС UNIX

по дисциплине «Управление ресурсами в вычислительных системах»

Факультет:	ПМИ
Группа:	ПМ-72
Вариант:	8
Студенты:	Антонов С.С.
Преподаватели:	Стасышин В.М. Сивак М.А.

Новосибирск

2020

1. Цель работы

Ознакомиться с системой управления вводом-выводом в ОС UNIX и основными структурами данных, используемыми этой системой. Исследовать механизм работы системы управления вводом-выводом.

2. Задание

Процесс создал новый файл и переназначил на него стандартный ввод. Разработайте программу, демонстрирующую динамику создания таблиц, связанных с этим событием (таблица описателей файла, таблица файлов, таблица открытых файлов процесса). Например, сценарий программы может быть следующим:

- неявное открытие стандартного файла ввода;
- неявное открытие стандартного файла вывода;
- неявное открытие стандартного файла вывода ошибок;
- чтение из стандартного файла ввода 5 байт;
- открытие пользовательского файла;
- закрытие стандартного файла ввода (моделирование **`close(0)`**);
- получение копии дескриптора пользовательского файла (моделирование **`dup(fd)`**, где **`fd`** - дескриптор пользовательского файла);
- закрытие пользовательского файла (моделирование **`close(fd)`**, где **`fd`** - дескриптор пользовательского файла);
- чтение из "стандартного" файла ввода 10 байт.

После каждого из этапов печатаются таблица описателей файлов, таблица файлов, таблица открытых файлов процессов.**Описание метода решения задачи**

Компиляция:

Программа написана на языке С, может быть скомпилирована компилятором языка С, с помощью make-файлов, либо без них.

Запуск программы осуществляется командой:

./hello [каталог]

По умолчанию вывод происходит на экран, но может быть легко перенаправлен в файл стандартными средствами интерпретаторов.

```
[pmi-b7608@students laba3]$ make
g++ main.c -o hello
[pmi-b7608@students laba3]$ ./hello
```

3. Используемые функции

int fstat(char *filename, struct stat *statbuf) - Функция fstat() вносит в структуру, на которую указывает statbuf, информацию, содержащуюся в файле, связанном с указателем filename.

int dup(int handle) - Функция dup() возвращает новый дескриптор файла, который полностью описывает (т.е. дублирует) состояние файла, связанного с handle. В случае успеха возвращается неотрицательная величина, а в противном случае —1.

int close(int fd) - При вызове функции close() с действительным дескриптором файла она закрывает связанный с ним файл, осуществив предварительно очистку буфера записи, если это необходимо

4. **Формат вывода результата**

Таблица:
descr name perm inode nlink UID GID

5. **Алгоритм**

1) Неявно открываем стандартные файлы (ввода, вывода, вывода ошибок).
2) Печатаем необходимые таблицы.
3) Открываем пользовательские файлы "file.txt" и "file1.txt".
4) Печатаем необходимые таблицы.
5) Закрываем стандартный файл ввода.
6) Печатаем необходимые таблицы.
7) Получаем копию дескриптора пользовательского файла "file.txt".
8) Печатаем необходимые таблицы.
9) Закрываем пользовательский файл "file1.txt".
10) Печатаем необходимые таблицы.

6. **Текст программы:**

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#define n 6

struct table
{
    int descr;
    struct stat p;
};


int main()
{
    char *name[2];
    int i, index;
    struct table mas[n];


   //Вывод таблицы
printf("Implicit opening of standard files\n");
   printf("-----------------------------------------------------------------\n");
    printf("descr\tname\t\tperm\tinode\tnlink\tUID\tGID\n");
    printf("-----------------------------------------------------------------\n");
    for(i = 0; i < 3; i++)
    {
        mas[i].descr = i;
        fstat(i , &mas[i].p);
        printf("%d\t\t\t\t%d\t%d\n", mas[i].descr, mas[i].p.st_ino,
mas[i].p.st_nlink);

    }
```

```
    //Открытие файлов
     printf("\nOpen file\n");
     name[0] = (char*)"file.txt";
     name[1] = (char*)"file1.txt";

     errno = 0;

    mas[3].descr = creat(name[0], 0666);
    if (errno != 0){
     perror("Error ");
    return 0;
    }

     errno=0;
    mas[4].descr = creat(name[1], 0666);
    if (errno != 0){
     perror("Error ");
    return 0;
    }

    stat(name[0],&mas[3].p);
    stat(name[1],&mas[4].p);

//Вывод таблицы
    printf("---------------------------------------------------------------------\n");
     printf("descr\tname\t\tperm\tinode\tnlink\tUID\tGID\n");
    printf("---------------------------------------------------------------------\n");
     for(i = 0; i < 3; i++)
         printf("%d\t\t\t\t%d\t%d\n", mas[i].descr, mas[i].p.st_ino,
mas[i].p.st_nlink);
     printf("%d\t%s\t%o\t%d\t%d\t%d\t%d\n", mas[3].descr, name[0], mas[3].p.st_mode,
mas[3].p.st_ino, mas[3].p.st_nlink, mas[3].p.st_uid, mas[3].p.st_gid);
     printf("\nClose standard input\n");

//Закрытие стандартного ввода
     close(0);
     fstat(0,&mas[0].p);

//Вывод таблицы
    printf("---------------------------------------------------------------------\n");
     printf("descr\tname\t\tperm\tinode\tnlink\tUID\tGID\n");
     printf("---------------------------------------------------------------------\n");
     for(i = 1; i < 3; i++)
         printf("%d\t\t\t\t%d\t%d\n", mas[i].descr, mas[i].p.st_ino,
mas[i].p.st_nlink);
     for(i = 3; i < 5; i++)
         printf("%d\t%s\t%o\t%d\t%d\t%d\t%d\n", mas[i].descr, name[i-3],
mas[i].p.st_mode,
                mas[i].p.st_ino, mas[i].p.st_nlink, mas[i].p.st_uid, mas[i].p.st_gid);

     printf("\nGetting a copy of the file descriptor \n");

//Получение копии дескриптора файла
     mas[5].descr=dup(mas[3].descr);//Копирование дескриптора
     fstat(mas[5].descr,&mas[5].p);

//Вывод таблицы
    printf("---------------------------------------------------------------------\n");
     printf("descr\tname\t\tperm\tinode\tnlink\tUID\tGID\n");
     printf("---------------------------------------------------------------------\n");
     for(i = 1; i < 3; i++)
         printf("%d\t\t\t\t%d\t%d\n", mas[i].descr, mas[i].p.st_ino,
mas[i].p.st_nlink);
     for(i = 3; i < 6; i++)
     {
```

```
        if(i == 4)
            index=1;
        else
            index=0;
        printf("%d\t%s\t%o\t%d\t%d\t%d\t%d\n", mas[i].descr, name[index],
mas[i].p.st_mode,
              mas[i].p.st_ino, mas[i].p.st_nlink, mas[i].p.st_uid,mas[i].p.st_gid);
    }

    printf("\n Closing file1.txt\n");
   printf("-----------------------------------------------------------------\n");
    printf("descr\tname\t\tperm\tinode\tnlink\tUID\tGID\n");
    printf("-----------------------------------------------------------------\n");
//Закрытие файла
    close(mas[4].descr);
//Вывод таблицы
    for(i = 1; i < 3; i++)
        printf("%d\t\t\t\t%d\t%d\n", mas[i].descr, mas[i].p.st_ino,
mas[i].p.st_nlink);
    for(i = 3; i < 6; i++)
    {
        if(i != 4)
printf("%d\t%s\t%o\t%d\t%d\t%d\t%d\n",mas[i].descr,name[0],mas[i].p.st_mode,
                  mas[i].p.st_ino, mas[i].p.st_nlink, mas[i].p.st_uid,
mas[i].p.st_gid);
    }
    return 0;
}
```

7. Тесты

Результат выполнения программы(без ошибок)

```
[pmi-b7608@students laba3]$ ./hello
Implicit opening of standard files
------------------------------------------------------------------
descr   name            perm    inode   nlink   UID     GID
------------------------------------------------------------------
0                               74      1
1                               74      1
2                               74      1

Open file
------------------------------------------------------------------
descr   name            perm    inode   nlink   UID     GID
------------------------------------------------------------------
0                               74      1
1                               74      1
2                               74      1
3       file.txt        100644  2235502 1       6124    3031

Close standard input
------------------------------------------------------------------
descr   name            perm    inode   nlink   UID     GID
------------------------------------------------------------------
1                               74      1
2                               74      1
3       file.txt        100644  2235502 1       6124    3031
4       file1.txt       100644  2235966 1       6124    3031

Getting a copy of the file descriptor
------------------------------------------------------------------
descr   name            perm    inode   nlink   UID     GID
------------------------------------------------------------------
1                               74      1
2                               74      1
3       file.txt        100644  2235502 1       6124    3031
4       file1.txt       100644  2235966 1       6124    3031
0       file.txt        100644  2235502 1       6124    3031

 Closing file1.txt
------------------------------------------------------------------
descr   name            perm    inode   nlink   UID     GID
------------------------------------------------------------------
1                               74      1
2                               74      1
3       file.txt        100644  2235502 1       6124    3031
0       file.txt        100644  2235502 1       6124    3031
```