ЛАБ 20

Задание:

19. Шаблоны имен файлов

Напишите программу, которая приглашает пользователя ввести шаблон имени файла, аналогичный тому, который используется в shell. Синтаксис шаблона таков:

- * соответствует последовательности любых символов кроме /, имеющей любую длину; возможно - пустой последовательности.
- ? соответствует любому одному символу.
- / не может встречаться.
- любой другой символ соответствует самому себе.

Символы * и ? в шаблоне могут встречаться в любом количестве и в любом порядке. Затем программа должна найти и распечатать имена всех файлов в текущем каталоге, соответствующих шаблону. Если таких файлов нет, программа должна распечатать сам шаблон.

Совет: используйте **readdir**, чтобы считать все имена файлов в текущем каталоге, и выберите из них соответствующие шаблону.

20. Шаблоны имен файлов (2)

Измените предыдущую программу так, чтобы в шаблоне могли встречаться символы /. При этом программа должна распечатывать все файлы, путевые имена которых соответствуют шаблону. Так, шаблону */* соответствуют все файлы во всех подкаталогах текущего каталога.

```
#include <stdio.h>
#include <dirent.h>
#include <stdlib.h>
#include <string.h>
#include <glob.h>

void matchPattern(char *pattern) {
        DIR *dirp;
        struct dirent *dp;

        if((dirp = opendir(".")) == NULL) {
                perror("Couldn't open directory\n");
                exit(0);
        }

        int flag = 0, count = 0;
        glob_t pglob;

        if (glob(pattern, GLOB_MARK, NULL, &pglob) == 0) {
                for(int i = 0; i < pglob.gl_pathc; ++i) {
                        printf("%s\n", pglob.gl_pathv[i]);
                }
                flag = 1;
        }

        if (flag == 0)
                printf("%s\n", pattern);

        closedir(dirp);
        globfree(&pglob);
}



void main(int argc, char* argv[]){
        if (argc != 2) {
                perror("Wrong pattern");
                exit(EXIT_FAILURE);
        }

        char space = ' ';
        char* ch;
        if((ch = strchr(argv[1], space)) != NULL){
                perror("' ' cannot be used!\n");
                exit(EXIT_FAILURE);
```

```
        }

        matchPattern(argv[1]);
}
```

▼ **GLOB** - поиск имен файлов по заданному шаблону

https://docs.oracle.com/cd/E23823_01/html/816-5168/glob-3c.html

int glob (const char *restrict** pattern**, int** flags**, int(*errfunc)(**const char *epath, int eerrno**), glob_t *restrict** pglob**);

Ключевое слово `restrict` позволяет программисту сообщить компилятору, что объявляемый указатель адресует область памяти, на которую не ссылается никакой другой указатель. Гарантию того, что на участок памяти не будут ссылаться более одного указателя, даёт программист. При этом оптимизирующий компилятор может генерировать более эффективный код

- ***pattern*** - указатель на шаблон имени пути. Функция glob() сопоставляет все доступные имена путей с этим шаблоном и составляет список всех совпадающих имен путей. Чтобы иметь доступ к имени пути, glob() требует разрешения на поиск для каждого компонента пути, кроме последнего, и разрешения на чтение для каждого каталога любого компонента имени файла шаблона, который содержит любой из следующих специальных символов

- ***flags -*** *А*ргумент flags используется для управления поведением glob(). Значение флагов представляет собой побитовое ИЛИ с нулевым или более из следующих констант:
  - **GLOB_MARK -** К каждому пути, который является каталогом, соответствующим *шаблону* , добавляется косая черта.

- ***pglob -*** *структура типа glob_t:*

```
size_t   gl_pathc;     /* число совпавших путей by pattern */
char     **gl_pathv;   /* указатель на список совпавших path names */
size_t   gl_offs;      /* slots to reserve at beginning of gl_pathv */
```

  Функция ***glob()*** сохраняет количество совпадающих имен путей в ***pglob->gl_pathc*** и указатель на список указателей на имена путей в ***pglob->gl_pathv***.

  Указатель после последнего имени пути - NULL.

  Если шаблон не соответствует ни одному имени пути, возвращаемое количество совпадающих путей устанавливается равным 0, а содержимое pglob->gl_pathv зависит от реализации. Вызывающий объект обязан создать структуру, на которую указывает pglob.

  Функция glob() выделяет другое пространство по мере необходимости, включая память, на которую указывает gl_pathv. Функция globfree() освобождает все пространство, связанное с pglob из предыдущего вызова glob().

- ***errfunc и epath*** - Если во время поиска встречается каталог, который нельзя открыть или прочитать, а errfunc не является указателем NULL, функция glob() вызывает (*errfunc) с двумя аргументами:
  1. Аргумент ***epath*** — это указатель на путь, по которому произошел сбой.
  2. Аргумент ***eerrno*** представляет собой значение ***errno*** из-за сбоя, установленное функциями opendir (3C) или readdir (3C)

  Если (**errfunc*) вызывается и возвращает ненулевое значение, или если флаг GLOB_ERR установлен в *flags* , glob() останавливает сканирование и возвращает GLOB_ABORTED после установки *gl_pathc* и *gl_pathv* в *pglob* для отражения уже просканированных путей.

Если GLOB_ERR не установлен и либо *errfunc* является указателем NULL , либо (**errfunc*) возвращает 0, ошибка игнорируется.

Возврат:

Успех: 0.

Аргумент *pglob*-> gl_pathc возвращает количество совпадающих имен путей, а аргумент *pglob*-> gl_pathv содержит указатель на список совпадающих и отсортированных путей, заканчивающийся нулем.

Иначе: ≠ 0

Произошла ошибка. Аргументы *pglob*-> gl_pathc и *pglob*-> gl_pathv по-прежнему установлены, как определено выше.

globfree() - освобождает всю память, выделенную функцией glob() , связанной с *pglob* .