ЛАБ 19

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 <fnmatch.h>
#include <unistd.h>

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

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

        int flag = 0;
        while((dp = readdir(dirp)) != NULL) {
                if (fnmatch(pattern, dp->d_name, 0) == 0) {
                        printf("%s\n", dp->d_name);
                        flag = 1;
                }
        }

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

        closedir(dirp);
}


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

        char bckSlash = '/';
        char space = ' ';
        char* ch;
        if((ch = strchr(argv[1], bckSlash)) != NULL){
```

```
                perror("'/' cannot be used!\n");
                exit(EXIT_FAILURE);
        }

        if((ch = strchr(argv[1], space)) != NULL){
                perror("Wrong format!\n");
                exit(EXIT_FAILURE);
        }

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

▼ STRCHR()

Функция `strchr` выполняет поиск первого вхождения символа `symbol` в строку `string`. Возвращает указатель на первое вхождение символа в строке. Завершающий нулевой символ считается частью Си-строки. Таким образом, он также может быть найден для получения указателя на конец строки.

В Си эта функция определена так:

```
1  char * strchr( const char * string, int symbol);
```

Параметры:

- **string**
  Указатель на строку.
- **symbol**
  Искомый символ, передается в функцию как целое число. После того как соответствующее значение будет найдено, функция преобразует его в символ.

Указатель на вектор символов - argv.

argv - является указателем на массив указателей на строки

▼ OPENDIR

DIR *opendir(const char *dirname) - функция, открывает поток каталога и возвращает указатель на структуру типа DIR, которая содержит информацию о каталоге

Возврат:

- **Успех:** указатель на структуру DIR, которая используется в качестве параметра к readdir(3C), closedir(3C)
- **Неуспех**: null + уствновка errno

Тип ***DIR (***определенный в заголовке <dirent.h>) представляет поток каталога, который представляет собой упорядоченную последовательность всех записей каталога в конкретном каталоге. Записи каталога представляют файлы.

closedir(3C) - функция, закрывает дескриптор директории, заданный параметром dirp, и освобождает связанную с ним структуру

Закрывает поток каталога, на который указывает ptr.

▼ READDIR

struct dirent *readdir(DIR *dirp) - *read directory*

Функция ***readdir(3C)*** используется для чтения записей из директории.

Возврат:

- указатель на структуру dirent, которая содержит следующую непустую запись в директории, заданной с помощью dirp.

readdir(3C) возвращает указатель на внутренний буфер, который может быть переиспользован следующим вызовом readdir(3C).

- нулевой указатель при достижении конца потока каталога

```
struct dirent{
  ino_t d_ino; /* inode number */
  off_t d_off; /* offset to the next dirent */
  unsigned short d_reclen;/* length of this record */
  unsigned char d_type; /* type of file; not supported by all file system types */
  char d_name[256]; /* filename */
};
```

▼ **Example from MAN *readdir***

Example 1 Search the current directory for the entry name.

```
    The following sample program will search the current directory for each
of the arguments supplied on the command line:

  #include <sys/types.h>
  #include <dirent.h>
  #include <errno.h>
  #include <stdio.h>
  #include <strings.h>

  static void lookup(const char *arg)
  {
          DIR *dirp;
          struct dirent *dp;

          if ((dirp = opendir(".")) == NULL) {
                  perror("couldn't open '.'");
                  return;
          }

          do {
                  errno = 0;
                  if ((dp = readdir(dirp)) != NULL) {
                          if (strcmp(dp->d_name, arg) != 0)
                                  continue;

                          (void) printf("found %s\\n", arg);
                          (void) closedir(dirp);
                          return;
                  }
          } while (dp != NULL);

          if (errno != 0)
                  perror("error reading directory");
          else
                  (void) printf("failed to find %s\\n", arg);
          (void) closedir(dirp);
          return;
  }

  int main(int argc, char *argv[])
  {
          int i;
          for (i = 1; i < argc; i++)
                  lookup(argv[i]);
          return (0);
  }
```

▼ **FNMATCH**

int fnmatch(const char *pattern, const char *string, int flags) - сравнивает имя файла и путь

Функция **fnmatch()** проверяет, совпадает ли строка *string* с параметром *pattern*, который является шаблоном (набором символов для текущей оболочки).

Возврат:

- 0, если строка *string* совпадает с шаблоном *pattern*
- **FNM_NOMATCH,** если строка и шаблон не совпадают, или другое ненулевое значение, если есть какая-либо ошибка в шаблоне.

The name fnmatch() is intended to imply filename match, rather than pathname match.

https://digitology.tech/docs/python_3/library/fnmatch.html - fnmatch

https://docs.oracle.com/cd/E23823_01/html/816-5168/fnmatch-3c.html#scrolltoc - man fnmatch

> Для совпадения литерал заключите мета-символы в скобки. Например, `'[?]'` соответствует символ `'?'`.
>
> Обратите внимание, что сепаратор имени файла (`'/'` в Unix) является специальным предложением *не* к этому модулю. См. модуль `glob` для расширения имени пути (`glob` использует `filter()` для сопоставления сегментов имени пути). Аналогично, имена файлов, начинающиеся с периода, не являются специальными для этого модуля и соответствуют шаблонам `*` и `?`.