Работа с файлами

Скубачевский Антон

22 сентября 2022 г.

На семинаре было рассмотрено, как работать с файлами: процессы открытия, закрытия, чтения и записи.

FILE* fopen (const char* path, const char*mode): открыть файл для дальнейшей работы, и присвоить специальной переменной типа FILE* указатель на поток (да-да, stream) входных и выходных данный из файла. Path - имя файла, строка в кавычках. mode: права доступа, тоже строка, в кавычках. Типы прав: "r" - только чтение, "w" - только запись, причем если пишем в файл, то предыдущее его содержимое затираем (по-английски truncate), "a+" - чтение и запись, запись в конец файла после хранящихся там ранее данных, ничего не стирается. Все варианты прав - см man fopen.

пример работы:

FILE* f1 = fopen("1.txt" , "a+");

Функция закрытия: int fclose(FILE* f). Возвращает 0 в случае успешного закрытия.

Пример работы:

fclose(f1);

Функция чтения:

size_t fread(void* buff, size_t size, size_t nmemb, FILE* f) - прочитать из файла f nmemb кусков данных размером size каждый, и все прочитанное положить в массив (аля буффер) buff. size = 1(байт) для символов(char). Возвращает функция fread число считанных символов. Если nmemb>числа символов в файле, ничего страшного, функция просто прочитает весь файл. Поэтому если файлы небольшие, можно ставить например всегда nmemb = 1000000. Если мы хотим прочитать 10 символов из файла f1 и записать в массив buff, делаем так:

#include <stdio.h>

```
#include <stdlib.h>
int main() {
FILE* f1 = fopen("1.txt" , "r" );
void* buff = malloc(10000);
long int l = fread(buff, 1, 10, f1);
printf("%s " , (char*)buff);
printf("%ld " , l);
fclose(f);
return 0;
}
```

Прога выведет нам на экран то, что мы считали из файла (строку из 10 символов, ну или меньше, если длина файла меньше), а потом то, сколько символов мы считали.

Второй аргумент fread универсальнее записать так: sizeof(buff[0]), и не надо тогда будет вспоминать, сколько же в типе char байт.

Если мы 2 раза вызовем fread, то второй fread продолжит читать с того места файла, где закончил первый. Если же первый считал весь файл, и мы вызовем второй раз fread, то программа будет плеваться. Объяснить это просто: ведь FILE* - поток данных, который через fread "выливается"к нам (выливается значит выливается копия данных в файле. Данные при "выливании потока данных"из файла, разумеется, никуда не деваются.), и если он весь вылился, то нам читать больше нечего. Удостоверьтесь в этом с помощью следующей программы:

```
int main() {
FILE* f = fopen("2.txt" , "r" );
void* buff = malloc(5);
while (fread(buff, 1,1000, f)!=0) {
printf("%s&" , (char*)buff);
}
printf("%d " ,fread(buff, 1,1000, f));
fclose(f);
return 0;
}
```

Но мы можем создать второй поток данных из файла: FILE* f2, и его читать, если мы не начитались:

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main() { FILE* f = fopen("2.txt" , "r" );
FILE* f1 = fopen("2.txt" , "r" );
void* buff = malloc(5000);
void* buff1 = malloc(5000);
fread(buff, 1,1000, f);
printf("pervii fread schital: %s" , (char*)buff);
fread(buff1, 1,1000, f1);
printf("vtoroi fread schital: %s" ,(char*)buff1);
fclose(f);
fclose(f1);
return 0;
}
```

Обратите внимание на тип void. Если мы создаем массив данных этого типа, то это массив из кусков, каждый размером 1байт(как и размер символа, кстати), данные в котором могут оказаться в итоге любого типа. Элемент типа void может оказаться как числом (int, например), так и символом. Чтобы дать программе понять, что же все-таки в этом массиве, мы в функции printf используем явное приведение типов: (char*)buff.

Программу можно писать и без воида, возможно вам так будет проще разобраться:

```
int main() {
FILE* f1 = fopen("1.txt" , "r" );
char* buff = (char*)malloc(10000*sizeof(char));
long int l = fread(buff, 1, 10, f1);
printf("%s " , buff);
printf("%ld " , l);
fclose(f);
return 0;
}
```

То что первый аргумент fread имеет по ману тип void* ничего страшного: ведь void - значит может быть любым типом.

Функция записи:

size_t fwrite(void* buff, size_t size, size_t nmemb, FILE* f): все те же аргументы и возвращаемое значение, что и у чтения. Записать из buff в f nmemb кусков размером size. Вернет то, сколько символов записали. Вместо 3го аргумента рекомендую писать то значение, которое вернул fread, если мы считали им из одного файла, а потом с помощью fwrite пишем всю считанную инфу в другой файл.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
FILE* f1 = fopen("3.txt" , "r" );
FILE* f2 = fopen("4.txt" , "a+" );
//void* buff = malloc(100000);
char* buff = (char*)malloc(100000*sizeof(char));
long int l = fread(buff, sizeof(buff[0]), 100000, f1);
printf("size of element = %ld " , sizeof(buff[0]));
//printf("content of the first file: %s" , (char*)buff);
printf("content of the first file: %s" , buff);
printf("symbols read = %ld " , l);
fwrite(buff, sizeof(buff[0]), l, f2);
fclose(f1);
fclose(f2);
return 0;
}
```

Эта программа читает из файла 3.txt и пишем в файл 4.txt. Считаем, что в файле 3.txt не более 100000 символов.

Также есть функции fgets и fputs.

Отличие fgets от fread в том, что если fgets дошла до конца строки в файле, то больше она читать не будет, а также в том, что она будет читать не nmemb символов, а на 1 меньше. Ну и аргумента только 3. И еще возвращаемое значение - считанная строка, а не размер считанной строки.

char* fgets(void* buff, size_t nmemb, FILE* f)

Пример:

```
#include <stdio.h>
#include <stdlib.h>
int main()
FILE* f = fopen("2.txt" , "r" );
void* buff = malloc(5);
while (fgets(buff, 5, f)!=NULL) {//пока не дойдем до конца файла
printf("%s&" , (char*)buff);
}
fclose(f);
return 0;
}
```

В файле 2.txt напишите, например:
Hello world
asdasf
и посмотрите на работу программы.