Потоки данных (1/2)

Поток - это абстракция, подразумевающая некую последовательность данных.

Можно представить поток как последовательность элементов, которые двигаются по конвейерной ленте и обрабатываются по одному.

Различают **текстовые** и **бинарные** потоки.

Бинарный поток - упорядоченная последовательность байтов. Можно записать в такой поток данные из произвольного объекта, а затем считать их в том же виде, без изменений.

Потоки данных (2/2)

Текстовый поток - последовательность символов, разбитая на строки, конец каждой из которых обозначен специальным символом завершения строки (каким именно - зависит от реализации).

Текстовые потоки гарантируют нормальную работу только с:

- печатными символами;
- символами переноса строки и табуляции.

Символ перевода не должен следовать за пробелами.

Буферизация потоков

Потоки могут буферизоваться. При работе с потоком создаётся буфер, в который записываются данные, и только после заполнения буфера они будут извлечены из потока.

Это связано с тем, что передача данных из потока, как правило, требует обращения к ОС (выполнения системных вызовов), а это - дорогостоящая операция.

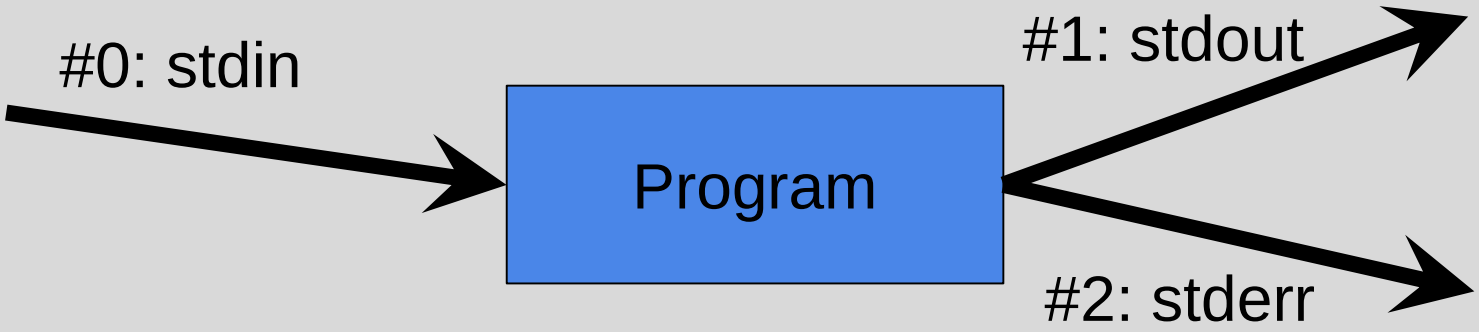
Буферизация позволяет сократить количество таких операций.

Стандартные потоки ввода-вывода

В начале работы программы доступны три потока: **stdin**, **stdout** и **stderr** (ввода, вывода и диагностики).

Поток **stderr** может быть небуферизован.

У каждого потока есть номер (дескриптор), по которому к нему можно обратиться из командной оболочки (shell-а).



Работа с потоками ввода-вывода

Для выполнения операций ввода-вывода в заголовочном файле **stdio.h** объявлены функции, константы (в виде макросов) и типы.

Для работы со стандартными потоками ввода-вывода существуют функции:

Чтение из stdin	Запись в stdout	Запись в stderr
scanf, vscanf	printf, vprintf	puterror
gets	puts	
getchar, fgetchar	putchar, fputchar	

С документацией к этим функциям следует ознакомиться самостоятельно.

Работа с потоками ввода-вывода - пример (1/2)

```
#define MY_STR_LENGTH 13

int main() {
        char nextStr[MY_STR_LENGTH];
        memset(nextStr, 0, sizeof(char) * MY_STR_LENGTH);
        printf("Enter someting: ");
        scanf("%s", nextStr);
        printf("%s", nextStr);
}
```

```
>Enter something: Hello, world!
>Hello,world!
```

Работа с потоками ввода-вывода - пример (2/2)

```
#define MY_STR_LENGTH 14

int main() {
        char str[MY_STR_LENGTH];
        memset(str, 0, sizeof(char) * MY_STR_LENGTH);
        printf("Enter someting: ");
        gets(str);
        printf("%s\n", str);
        scanf("%c", &str[4]);
        printf("%s", str);
}
```

Файловая система, файлы

Файловая система - это набор правил организации и алгоритмов доступа к именованным объектам, инкапсулирующим данные или функциональность.

Файл - это именованная область данных, объект в составе файловой системы, доступ к которому ей же и управляется.

Для файла могут быть определены операции:

создания/удаления;

открытия/закрытия;

чтения/записи;

Потоки данных и работа с файлами

При открытии файла с ним ассоциируется поток, при закрытии выполняется освобождение ресурсов потока.

Информацию для работы с открытым файлом содержит объект типа **FILE**.

Существует константа **EOF**, соответствующая концу файла, а также **BUFSIZ**, в которой хранится размер буфера записи, которая на данной системе даёт лучшие результаты с точки зрения производительности.

Какие ещё константы определены в заголовочном файле stdio.h? Зачем они нужны?

Работа с файлами в C - stdio.h

Операции по открытию и закрытию файла доступны в виде функций **fopen**, **fclose**.

Функции посимвольного ввода-вывода: **fgetc**, **fputc**, **getc**, **putc**, **ungetc**.

Функции для чтения и записи строк: **fgets**, **fputs**.

Функции для чтения и записи наборов байт: **fread**, **fwrite**.

Функции для позиционирования потока в файле: **fgetpos**, **fseek**, **fsetpos**, **ftell**.

Работа с файлами - пример (1/2)

```
#define ARR_SIZE 6

#define ARR_FILENAME "array.bin"

int main() {
        int i = 0;
        int inArr[ARR_SIZE] = { 1, 2, 3, 4, 5, 6 };
        int *outArr = (int*)calloc(ARR_SIZE, sizeof(int));
        FILE *arrPtr = fopen(ARR_FILENAME, "wb");
        fwrite(inArr, sizeof(int), ARR_SIZE, arrPtr);
        fclose(arrPtr);
        ...
```

Работа с файлами - пример (2/2)

```
    ...
    arrPtr = fopen(ARR_FILENAME, "rb");
        fseek(arrPtr, sizeof(int), SEEK_SET);


        fread(outArr, sizeof(int), ARR_SIZE - 1, arrPtr);
        fclose(arrPtr);


        for (i = 0; i < ARR_SIZE; i++)
                printf("%i ", outArr[i]);
}
```

Пример - перенаправление потоков

```
#include <stdio.h>


int main()
{
        freopen("stdout.txt", "w", stdout);


        printf("Hello, world!");
}
```

Будет создан файл stdout.txt, и строчка “Hello, world!” записана в него.