# **18. Обработка файлов и потоков. Файлы и потоки.** Файл - это упорядоченная и именованная последовательность байтов, которая имеет постоянное хранилище. Поток — это последовательность байтов, которую можно использовать для записи или чтения из устройств хранения информации (например, дисков или памяти). Для работы с файлами существуют файловые потоки. Есть несколько видов запоминающих устройств и существует несколько видов потоков, помимо файловых потоков, например сетевые потоки, потоки памяти и потоки каналов. Прежде чем использовать поток для доступа к файлу, его необходимо соединить с файлом. Эта информация хранится в структуре типа FILE, поэтому поток имеет вид FILE\* (их часто отождествляют). Когда поток соединяют с файлом, то говорят, что файл открывают; когда отсоединяют - говорят, что файл закрывают.

# Каждый поток может работать в 2 режимах:         текстовый         бинарный Режим работы потока задается при его соединении с файлом. В текстовом режиме поток записывает и читает текстовые строки, которые заканчиваются символом ‘\n’ и могут содержать символы ‘\t’. При записи и чтении данных текстовым потоком может происходить преобразование этих данных. В бинарном режиме поток записывает и читает данные один к одному.

# **18.2 Обработка текстовых файлов 18.2.1 Открытие файла.** FILE\* fopen (const char\* fname, const char\* mode); FILE \*f1; FILE \*f2,\*f3; f1 = fopen("input" , "r"); f2 = fopen("h:\ex.dat", "w+"); f3 = fopen("work", "a+");

# "w+" - новый *текстовый файл* открывается для записи и последующих многократных исправлений. Если *файл* уже существует, то предыдущее содержимое стирается. Последующие после *открытия файла* запись и чтение из него допустимы в любом месте *файла*, в том числе запись разрешена и в конце *файла*, т.е. *файл* может увеличиваться.

# "r+" - существующий *текстовый файл* открывается как для чтения, так и для записи в любом месте *файла*; однако в этом режиме невозможна запись в конец *файла*, то есть недопустимо увеличение размеров *файла*.

# "a+" - *текстовый файл* открывается или создается, если *файла* нет, и становится доступным для изменений, т.е. для записи и для чтения в любом месте; при этом в отличие от режима "w+"можно открыть существующий *файл* и не уничтожать его содержимое; в отличие от режима "r+" в режиме "a+" можно вести запись в конец *файла*, то есть увеличивать его размеры.

# **18.2.2 Перенаправление потока.** FILE\* freopen (const char\* filename, const char\* mode, FILE\* stream); Закрывает файл, соединенный с потоком stream, и соединяет этот поток с файлом filename в режиме mode. **FILE \*f2,\*f3; f3 = freopen("input1.dat", "r", f2 );**

# **18.2.3 Закрытие файла.** int fclose(FILE\* stream); Закрывает файл, при этом освобождая все буферы потока. В случае удачи возвращает 0, иначе – EOF. **fclose(f3);**

# **18.2.4 Проверка на конец файла.**   Каждый поток содержит индикатор конца файла, который хранится в структуре файла и устанавливается в ненулевое значение функцией чтения из файла при достижении конца файла. Состояние определяется функцией: **int feof(FILE\* file); F1=fopen(“inp.txt", "r"); while (!feof( F1)) { //….. обработка}**

# **18.2.5. Функция для записи строк: int fputs(const char\* str, FILE\* stream);** Пишет строку str в поток stream не включая завершающий нулевой байт. В случае удачи возвращает 0, иначе возвращает EOF. **fputs(str,out);**

# **18.2.5. Функция для чтения строк:** **char\* fgets(const char\* str, int n, FILE\* stream);** Читает строку из потока stream в строку str, остановится: ·     прочитан n+1 символ; ·     встретился символ новой строки; ·     достигнут конец файла. Символ копируется в строку, к концу строки добавляется нулевой байт. Возвращает при успехе указатель на str, иначе возвращает NULL. Строка str не изменяется, если ни один символ не прочитан и встретился конец файла.

# **Пример 18.1 Чтение из текстового файла void ReadStringFile( FILE \*in, char\* Fname )** **{ char \*str; str= new char [81];   if ((in = fopen(Fname, "r") ) != NULL) { if (feof(in)) return; fgets(str, 80, in); do**

# **{ puts(str); // При выводе распознает специальные символы \n fgets(str,80,in); } while ( !feof ( in )); fclose (in ); } else printf (" Файл не открывается\n"); }**

# **Пример 18.2. Создание текстового файла void CreateStringFile( FILE \*out, char\* Fname ) {int k; char str[80]; out = fopen(Fname, "w") ; gets(str); while(!feof(stdin)) { fputs(str,out); gets(str); // Не заносит в строку символ возврата каретки // Заносит \0 } fclose(out); }**

# **18.2.6 Символьный ввод-вывод.**   Используется с текстовыми потоками. **int fputc(int c, FILE\* stream);** Записывает символ **c** в поток stream и продвигает индикатор позиции на следую-щий символ. При успехе возвращает c, в случае неудачи EOF (-1) . **int fgetc(FILE\* stream);** Работает аналогично fputc, но читает символ. В случае достижения конца файла возвращает EOF и устанавливает индикатор конца файла.

# **void ReadChFile( FILE \*in, char\* Fname )** {………… do **{ ch = fgetc(in); putc(ch,stdout); } while (ch != EOF);**

# **18.2.7 Форматированный ввод-вывод. int fprintf(FILE\* stream, const char\* format, *объекты вывода);*** Форматированный вывод в файл stream. Работает аналогично sprintf, возвращает число выводимых символов, иначе -1. **int fscanf(FILE\* stream, const char\* format, *объекты вывода*);** Форматированный ввод из файла stream, в случае успеха возвращает количество единиц прочитанных данных, иначе - EOF.

# **18.2.8 Стандартные потоки.** Каждой программе предоставляется три стандартных потока, которые по умолчанию соединены с консолью. Указатели на эти потоки возвращают макрокоманды: stdin, stdout, stderr. Для работы со стандартными потоками предназначены функции: **int putchar(int c); // stdout int getchar(void); // stdin int puts(const char\* str); // stdout char\* gets(char\* str); // stdin void perror(const char\* str); // stderr** Использование: выводим строку str в стандартный поток stderr, после этого выводим пробел и сообщаем об ошибке, связанной со значением переменной errno.

# **18.2.9 Дополнительные функции. int remove(const char\* filename);** Удаляет файл с именем filename. В случае удачи возвращает 0, иначе возвращает не 0. **int rename(const char\* old\_filename, const char\* new\_filename);** Переименовывает файл. В случае удачи возвращает 0, иначе возвращает не 0.

# **FILE\* tmpfile(void);** Создает временный файл в режиме w+b. В случае неудачи возвращает NULL. **char\* tmpname(char\* str);** Возвращает имя временного файла. Максимальное число имен равно (для Windows) 32767, максимальная длина имени – 255. В случае удачи возвращает str, иначе возвращает NULL.

# **Пример 18.3. int remFile(char \*filename) { int flag; flag=rename(filename, "C:\\emp.txt"); if (flag==1 ){ printf("There is no such a file.\n"); return 0; } printf("The file was renamed.\n"); return 1;**

# **flag** = **remove(filename); if (flag) { printf("There is no such a file.\n"); return 0; }   printf("The file was deleted.\n");   return 1; }**

# **18.3. Обработка бинарных файлов**  бинарный режим: **“rb”, “wb”, “ab”** – чтение, запись, присоединение;  бинарный режим: **“rb+”, ‘‘wb+”, “ab+”** или **“r+b”, “w+b”, “a+b”** – чтение и запись.

# **18.3.1 Открытие файла.** FILE\* fopen (const char\* fname, const char\* mode); FILE \*f1; FILE \*f2,\*f3; f1 = fopen("input" , "rb"); f2 = fopen("h:\ex.dat", “wb"); f3 = fopen("work", “ab");

# **18.3.2 Блочный ввод-вывод.**   Блоком называется область оперативной памяти, содержимое которой записывается в байтах. **Ввод-вывод блоками используется бинарными потоками**. Запись блоков в файл: **size\_t fwrite (const void\* ptr, size\_t size, size\_t nitems, FILE\* stream );**

# где **ptr** указывает на блок памяти, который записывается в файл ; длина блока указывается как **size\*nitems**. Функция возвращает число записанных единиц. В случае удачи это число должно быть равно nitems.

# Чтение: **size\_t fread(void\* ptr, size\_t size, size\_t nitems, FILE\* stream); Параметры имеют то же смысл, что и в fwrite.**

# **Пример 18.4 Создание бинарного файла void CreateBinFile (void) { struct emp { int code; char name[20]; } FILE\* out; struct emp s;   if(!(out = fopen("z.bin", "wb"))) { printf("Open file failed.\n"); return ; }**

# **printf("Input code, name.\n"); printf("Press Ctrl+z to exit.\n");   printf("1. "); int i=2; scanf("%d%s", &s.code, &s.name);**

# **while (!feof(stdin)) { fwrite(&s, sizeof(struct emp), 1, out);   printf("%d. ",i); i++; scanf("%d%s", &s.code, &s.name); } fclose(out); }**

# **Пример 18.5 Чтение бинарного файла void ReadBinFile(void)   struct emp { int code; char name[20]; };   FILE\* in; struct emp s; unsigned i;**

# **if(!(in = fopen("z.bin", "rb"))) { printf("Open file failed.\n"); return ; } fread(&s, sizeof(struct emp), 1, in); do { printf("\tcode = %d name = %s \n",s.code, s.name); fread(&s, sizeof(struct emp), 1, in); } while (!feof(in)); fclose(in);}**

# **18.3.3 Работа с индикатором позиции файла.**   Для каждого файла после его открытия определяется индикатор позиции, который указывает на смещение от начала файла в байтах. Для работы с индикатором позиции применяются следующие функции: **void rewind(FILE\* stream);** Устанавливает индикатор позиции на начало файла. Индикаторы ошибки и конца файла сбрасываются.

# **int fseek(FILE\* stream, long offset, int mode); Сдвигает индикатор позиции файла на offset байт. Параметр mode определяет режим сдвига и может принимать следующие значения:         seek\_set – от начала файла;         seek\_cur – от текущей позиции;         seek\_end – от конца файла**

# При работе с текстовым потоком должны использоваться только следующие комбинации значений параметра: **mode** **offset** seek\_set 0 или ftell seek\_cur 0 seek\_end 0

# **int fsetpos(FILE\* stream, const fops\_t\* pos);** Устанавливает индикатор позиции файла в позицию, на которую указывает pos. Индикатор конца файла сбрасывается. Код возврата: удача – 0, неудача – не 0. **long ftell(FILE\* stream);** В случае успеха возвращает текущую позицию файла, иначе возвращает -1. **int fgetpos(FILE\* stream, fops\_t\* pos);** Работает аналогично ftell, но возвращает текущую позицию в pos.

# **printf("Input an index: "); scanf("%u", &i); // устанавливает указатель на нужную запись fseek(in, i\*sizeof(struct emp), SEEK\_SET); if(!fread(&s, sizeof(struct emp), 1, in)) { printf("The wrong index.\n"); return; } printf("\tcode = %d name = %s \n", s.code, s.name);**