Lecture 12

CIS 208

Friday, March 3rd , 2005

File I/O review

- dbl_space.c

fread & fwrite

- reads and write data types larger than 1 byte (character)
- can read/write entire arrays

```
int fread(void *buffer, int numbytes, int count, FILE *fp)
int fwrite(void *buffer, int numbytes, int count, FILE *fp)
```

fread

int fread(void *buffer, int numbytes, int count, FILE *fp)

- *buffer: memory that will receive data
- numbytes: size of each item (use sizeof)
- count: number of items
- *fp: source file
- returns number of items read

fwrite

int fwrite(void *buffer, int numbytes, int count, FILE *fp)

- *buffer: source memory
- numbytes: size of each item
- count: number of items
- *fp : destination file

File Pointer placement

- Can be moved around
- But not with regular pointer arithmetic.
- Must use C functions for random access

rewind

void rewind(FILE *fp)

- moves the file pointer back to the beginning.

Random Access

- must tell pointer where to go.
- Every time a char is read, the file position indicator increments by 1

File position indicator

- a long int
- represents the number of bytes from beginning to current location
- starts with zero

ftell

long int ftell(FILE *_fp_)

- returns current value of file position indicator.
- returns –1 if unsuccessful.

fseek

int fseek(FILE **fp*, long int *numbytes,* int *origin*)

- moves file position indicator to origin + numbytes.

- if numbytes is positive, moves forward in file
- else, moves backwards in file.

fseek

- numbytes: number of bytes to move
  - 1 char = 1 byte.
  - doesn't really work this way, but makes it easier to think about
- can be the number returned by ftell

origin

- accepts 3 values.

SEEK_SET == 0 //beginning of file
SEEK_CUR == 1 //current location
SEEK_END == 2 // end of file.

fseek

- returns 0 if successful, nonzero else.
- Note: only guaranteed to work on binary files.
  - in Unix, binary file and text files are the same. So open in binary mode
  - not so in MS-DOS.

fseek

fseek(in, 0, SEEK_END)

fseek(in, 1, 1)

fseek(in, -1, SEEK_SET)

reading backwards?

- Reading a file backwards and printing to console.
- how would we do it?
- Any ideas?

ferror

int ferror(FILE *fp)

- determines if error has occurred with file pointer fp.
- each file operation sets an error flag.
- ferror checks that flag.
- returns 0 if no error
- non-zero if there was an error

ferror

- should be used immediately after file operation

```
ch = getc(stdin);
if (ferror(stdin)) printf("error\n");
```

Erasing Files

- don’t need file pointer. But file can’t be open.

int remove(const char *filename);

returns 0 if sucessful, else non-zero

Temporary Files

- Files opened by C for transfer purposes

```
FILE *tmpfile(void);
```

opens a FILE pointer to a new file.

File has unique name.

Don’t actually care what name is.

tmpfile()

- file is a binary-read-write file. ‘wb+’
- Automatically opens for you.
- Returns null pointer if it fails.

- You may use this file for anything.

Closure

- File is automatically deleted when:
  - FILE stream is closed
  - program terminates.

```
FILE *temp;
if ((temp = tmpfile())==NULL{
    printf("cannot Open Temporary work File.\n");
    exit(1);
}
```