11.3 Files and Streams

- Read/Write functions in standard library
  - `fgetc`
    - Reads one character from a file
    - Takes a `FILE` pointer as an argument
    - `fgetc( stdin )` equivalent to `getchar()`
  - `fputc`
    - Writes one character to a file
    - Takes a `FILE` pointer and a character to write as an argument
    - `fputc( 'a', stdout )` equivalent to `putchar( 'a' )`
  - `fgets`
    - Reads a line from a file
  - `fputs`
    - Writes a line to a file
  - `fscanf` / `fprintf`
    - File processing equivalents of `scanf` and `printf`

11.4 Creating a Sequential-Access File

- C imposes no file structure
  - No notion of records in a file
  - Programmer must provide file structure
- Creating a File
  - `FILE *cfPtr;`
  - Or `FILE* cfPtr;`
    - Creates a `FILE` pointer called `cfPtr`
  - `cfPtr = fopen("clients.dat", "w");`
    - Function `fopen` returns a `FILE` pointer to file specified
    - Takes two arguments – file to open and file open mode
    - If open fails, `NULL` returned

Personally, I like cfPtrW. W is a reminder for "w".

11.4 Creating a Sequential-Access File

- `fprintf`
  - Used to print to a file
  - Like `printf`, except first argument is a `FILE` pointer (pointer to the file you want to print in)
- `feof(` *`FILE pointer`* `)`
  - Returns true if end-of-file indicator (no more data to process) is set for the specified file
- `fclose(` *`FILE pointer`* `)`
  - Closes specified file
  - Performed automatically when program ends
  - Good practice to close files explicitly

- *Details*
  - Programs may process no files, one file, or many files
  - Each file must have a unique name and should have its own pointer

```c
/* Fig. 11.3: fig11_03.c
   Create a sequential file */
#include <stdio.h>

int main( void )
{
   int account;     /* account number */
   char name[ 30 ]; /* account name */
   double balance;  /* account balance */

   FILE *cfPtr;     /* cfPtr = clients.dat file pointer */

   /* fopen opens file. Exit program if unable to create file  */
   if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
      printf( "File could not be opened\n" );
   } /* end if */
   else {
      printf( "Enter the account, name, and balance.\n" );
      printf( "Enter EOF to end input.\n" );
      printf( "? " );
      scanf( "%d%s%lf", &account, name, &balance );
```

FILE pointer definition creates new file pointer

fopen function opens a file; w argument means the file is opened for writing

11.7 Creating a Random-Access File

- Data in random access files
  - Unformatted (stored as "raw bytes")
    - All data of the same type (`int`s, for example) uses the same amount of memory
    - All records of the same type have a fixed length
    - Data not human readable.
- What is human unreadable?
  - Use nodepad to open a pdf file, you will know.

11.7 Creating a Random-Access File

- Unformatted I/O functions
  - `fwrite`
    - Transfer bytes from a location in memory to a file
  - `fread`
    - Transfer bytes from a file to a location in memory
  - Example:

    `fwrite( &number, sizeof( int ), 1, myPtr );`
    - `&number` – Location to transfer bytes from
    - `sizeof( int )` – Number of bytes to transfer
    - `1` – For arrays, number of elements to transfer

      In this case, "one element" of an array is being transferred
    - `myPtr` – File to transfer to or from

11.7 Creating a Random-Access File

- Writing `struct`s

  `fwrite( &myObject, sizeof (struct myStruct), 1, myPtr );`
  - `sizeof` – returns size in bytes of object in parentheses
- To write several array elements
  - Pointer to array as first argument
  - Number of elements to write as third argument

```c
/* Fig. 11.11: fig11_11.c
   Creating a random-access file sequentially */
#include <stdio.h>

/* clientData structure definition */
struct clientData {
   int acctNum;           /* account number */
   char lastName[ 15 ];   /* account last name */
   char firstName[ 10 ];  /* account first name */
   double balance;        /* account balance */
}; /* end structure clientData */

int main( void )
{
   int i; /* counter used to count from 1-100 */

   /* create clientData with default information */
   struct clientData blankClient = { 0, "", "", 0.0 };
```

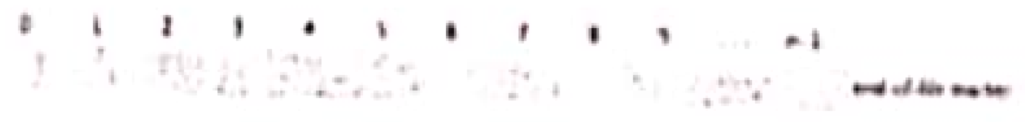
Outline

fig11_11.c

(1 of 2)

11.3 Files and Streams

- C views each file as a sequence of bytes
  - File ends with the *end-of-file marker*
    - Or, file ends at a specified byte



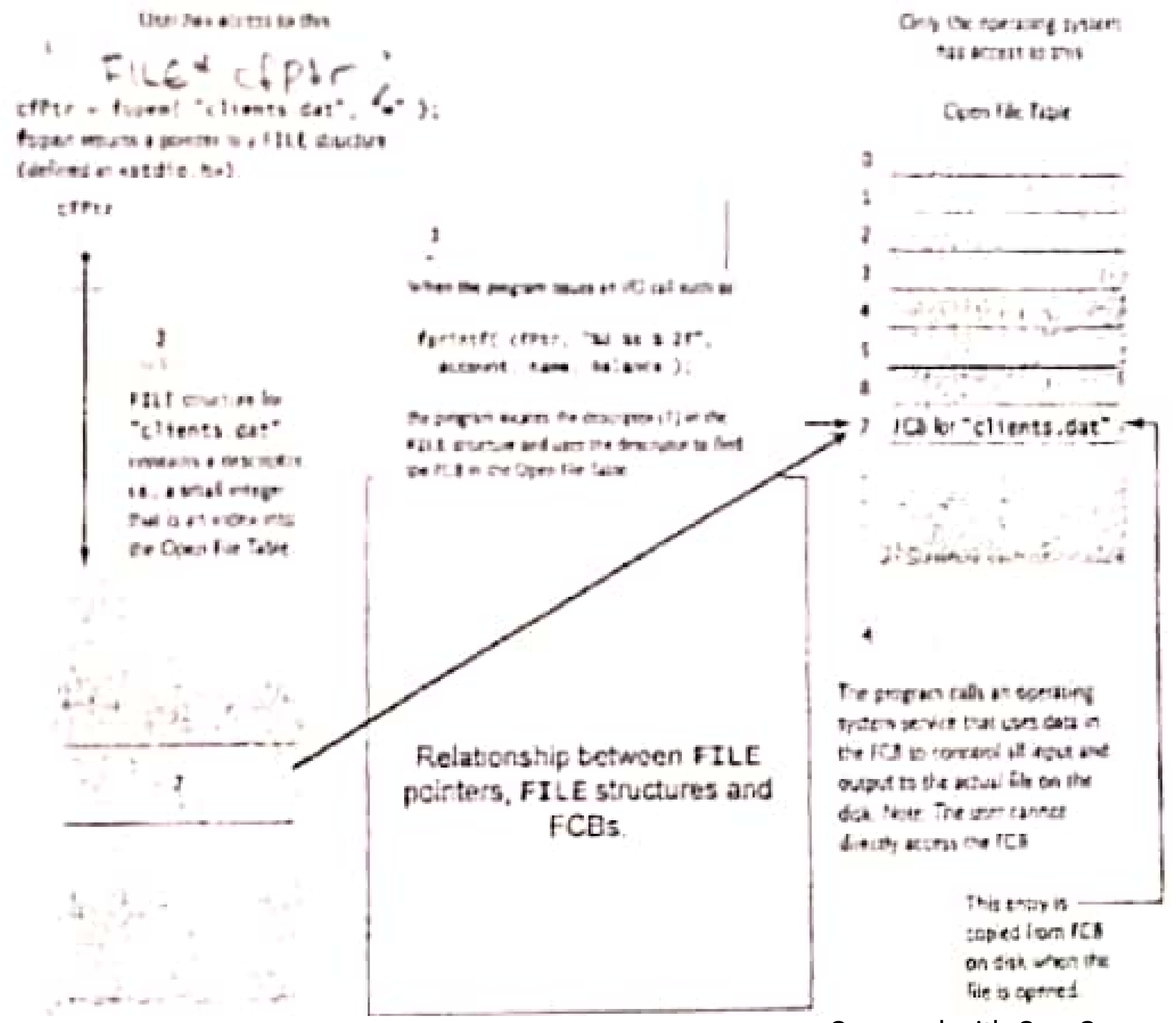
Operating system	Key combination
Linux/Mac OS X/UNIX	*<Ctrl> d*
Windows	*<Ctrl> z*

11.3 Files and Streams

- Stream created when a file is opened
  - Provide communication channel between files and programs
  - Opening a file returns a pointer to a `FILE` structure
    - Example file pointers:
    - `stdin` - standard input (keyboard) This is a file
    - `stdout` - standard output (screen) This is a file
    - `stderr` - standard error (screen) This is a file

11.3 Files and Streams

- `FILE` structure
  - File descriptor
    - Index into operating system array called the open file table
- File Control Block (FCB)
  - Found in every array element, system uses it to administer the file


User has access to this
FILE* cfPtr ;
cfPtr = fopen("clients.dat", "w");
fopen returns a pointer to a FILE structure (defined in <stdio.h>).
cfPtr
2
FILE structure for "clients.dat" contains a descriptor, i.e., a small integer that is an index into the Open File Table.
3
When the program issues an I/O call such as
fprintf(cfPtr, "%d %s %.2f", account, name, balance);
the program locates the descriptor (7) in the FILE structure and uses the descriptor to find the FCB in the Open File Table.
Only the operating system has access to this
Open File Table
0
1
2
3
4
5
6
7 FCB for "clients.dat"
4
The program calls an operating system service that uses data in the FCB to control all input and output to the actual file on the disk. Note: The user cannot directly access the FCB.
This entry is copied from FCB on disk when the file is opened.

Relationship between `FILE` pointers, `FILE` structures and FCBs.

```
Enter request
 1 - List accounts with zero balances
 2 - List accounts with credit balances
 3 - List accounts with debit balances
 4 - End of run
? 1

Accounts with zero balances:
300         White              0.00

? 2

Accounts with credit balances:
400         Stone            -42.16

? 3

Accounts with debit balances:
100         Jones             24.98
200         Doe              345.67
500         Rich             224.62

? 4
End of run.
```

11.5 Reading Data from a Sequential-Access File

- **Sequential access file**
  - Cannot be modified without the risk of destroying other data
  - Fields can vary in size
    - Different representation in files and screen than internal representation
    - 1, 34, -890 are all ints, but have different sizes on disk
  - Note, int 1, char '1', and string "1" have no difference on disk.

Anlamadım

11.6 Random-Access Files

- Random access files
  - Access individual records without searching through other records
  - Instant access to records in a file
  - Data can be inserted without destroying other data
  - Data previously stored can be updated or deleted without overwriting
- *Implemented using fixed length records*
  - Sequential files do not have fixed length records

Anlamadım

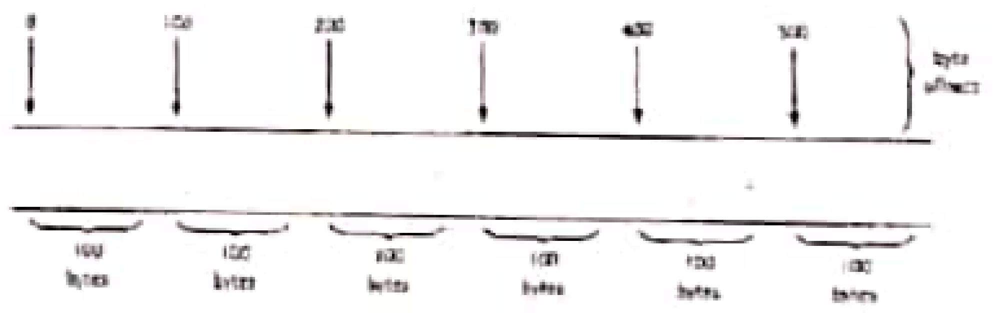


Fig. 11.10 | C's view of a random-access file