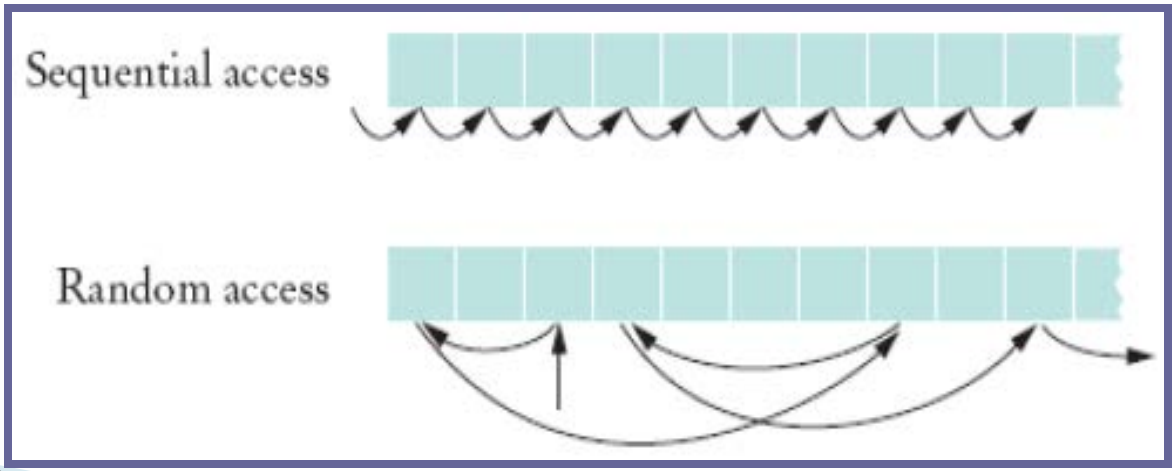
Lecture 7

Random access to files

Random Access Files

- Random access files are files in which records can be accessed in any order
  - Also called direct access files
  - More efficient than sequential access files

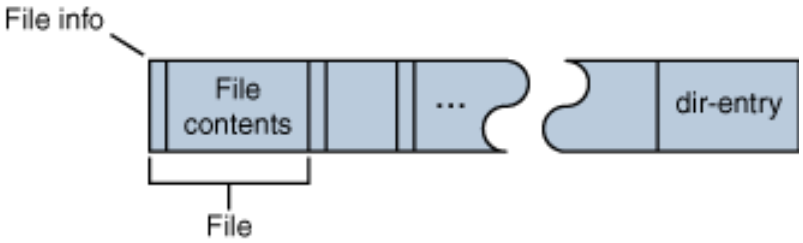


Need for Random Access Files

- Real-time applications require immediate response
  - Example: respond to customer query about a bill
  - Sequencing through records for account is time-intensive
- Random (immediate) access meets real-time need
  - Directly read from or write to desired record

Example

- Consider the zip format. A ZIP archive contains files and is typically compressed to save space. It also contain a directory entry at the end that indicates where the various files contained within the ZIP archive begin



Accessing a specific file using sequential access

- Open the ZIP archive.
- Search through the ZIP archive until you locate the file you want to extract.
- Extract the file.
- Close the ZIP archive.

On an average, we have to read half of the zip archive to find the required file

Accessing a specific file using random access

- Open the ZIP archive.
- Seek to the directory entry and locate the entry for the file you want to extract from the ZIP archive.
- Seek (backward) within the ZIP archive to the position of the file to extract.
- Extract the file.
- Close the ZIP archive.

  This is more efficient as you read only the directory entry and file that you want to extract.

RandomAccessFiles class

- The RandomAccessFile class contains the same read(), write() and close() methods as Input and OutputStream
- Also contains seek() that lets you select a beginning position within the file before reading or writing data
- Includes capabilities for reading and writing primitive-type values, byte arrays and strings

- **`fseek`**
  - Sets file position pointer to a specific position
  - **`fseek(`** *pointer, offset, symbolic_constant* **`);`**
    - *pointer* – pointer to file
    - *offset* – file position pointer (0 is first location)
    - *symbolic_constant* – specifies where in file we are reading from
    - **`SEEK_SET`** – seek starts at beginning of file
    - **`SEEK_CUR`** – seek starts at current location in file
    - **`SEEK_END`** – seek starts at end of file

ftell

The C library function **long int ftell(FILE *stream)** returns the current file position of the given stream.

```
#include <stdio.h>
 int main ()
{ FILE *fp;
int len;
 fp = fopen("file.txt", "r");
if( fp == NULL )
 {
 perror ("Error opening file");
 return(-1);
}
fseek(fp, 0, SEEK_END);
len = ftell(fp);
 fclose(fp);
 printf("Total size of file.txt = %d bytes\n", len);
return(0); }
```