Random Access in Files

Files are just streams of bytes. Whenever you read something from a file, it is done sequentially by default. That is, firstly first byte would be read, then second byte and then third byte and so on.

File Pointers

Every file maintains a file pointer which tells the current position in the file where writing or reading will take place.

Whenever reading or writing done onto a file then these two things happens
i) this operation takes place at the position of file pointer
ii) file-pointer advances by the specified number of bytes

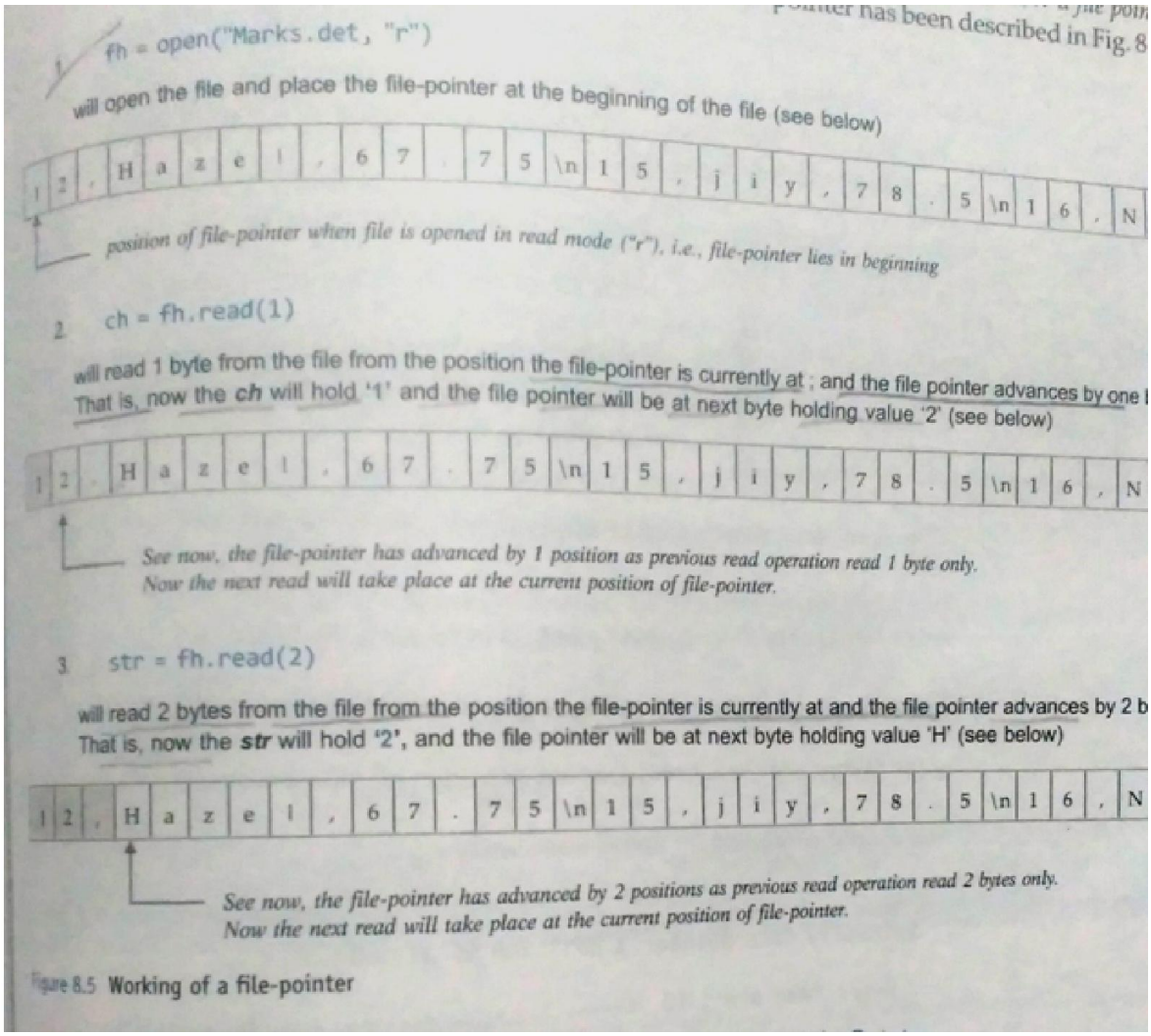


Figure 8.5 Working of a file-pointer

Random Access Functions

When you read or write in a file, then by default the access in sequential. Every read and write operation takes place at the current position of the file pointer. Python provides two functions that help you manipulate the position of file-pointer and thus you can read and write from desired position in the file.
The two random access functions of Python are : tell() and seek()

The tell() function

tell function return the current position of file pointer in the file.

```
<file-object>.tell()
```

The seek() function

seek() function changes position of file pointer by replacing, the file pointer at the specified position.

```
<file-object>.seek(offset, [mode])
```

Where offset - is a number specifying number of bytes
mode - is a number 0 or 1 or 2
0 for beginning of file(i.e. to move file pointer w.r.t. beginning of file) it is default position.
1 for current position of pointer (i.e. to file pointer w.r.t current position of it)
2 for end of file (i.e. to move file-pointer w.r.t end of file)

```
fh = open("Marks.dat", "r")
fh.seek (30)
# will  place the file pointer at 30th byte from
# the beginning of file(default)
fh.seek(30, 1)
# will place the file pointer at 30th byte ahead
#from the current position.
fh.seek(-30, 2)
# will place file pointer at 30 bytes behind
#(backward direction) from end of file.
```

```
# check the position of file pointer after read() function
fh = open(r"e:\Marks.dat", "r")
print(fh.read())
print("File - pointer is now at byte : ", fh.tell())
```

Output:
12,Ramesh,78.33
34,Yatin,78.5
56,Noor,68.9
59,Akshra,78.9
70,Jivin,89.5
File - pointer is now at byte : 75

read the last 15 bytes of the file Marks.dat

```
fh = open(r"e:\Marks.dat","rb")
'''n text files (those opened without a b in the mode string), only seeks relative to the beginning of
the file are allowed (the exception being seeking to the very file end with seek(0, 2)).
if we open the file in "r" mode it gives io.UnsupportedOperation: can't do nonzero end-relative seeks '''

fh.seek(-15, 2)
str1 = fh.read(15).decode('utf-8')
print("Last 15 bytes of file contain : ", str1)
```

Renaming and Deleting File in Python

Sometimes, you need to rename or delete/remove a file from within a program, For this Python provides os module. Python os module provides methods for file procession operations, like renaming and deleting files. In order to use the function defined in this module, you need to import it first.

The rename() method

This function is used to rename a file existing on the disk. It is used as per following syntax:

```
os.rename(<current-file-name>, <new-file-name>)
```

For example to rename an existing file "Myfile.txt" to "Yourfile.txt"

```
import os
os.remove("Myfile.txt", "Yourfile.txt")
```

The remove() method

This function is used to remove or delete a file existing on the disk.

```
os.remove(<file-name>)
```

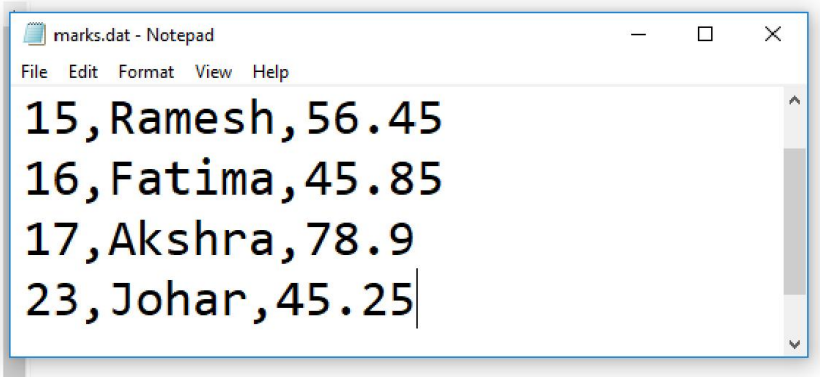
for e.g. to delete a file say "MYfile.txt" from the disk you may write:

```
import os
os.remove("Myfile.txt")
```

Usages of rename() and remove() Function

The functions rename() and remove() play an important role in file manipulation. Using these functions, you can insert a new record at desire position in a file or you can even delete and modify records.

For example if you want to insert into a sorted file without disturbing the order of records. Consider the marks.dat that contains student record in the following order:



Now, if you want to add a roll no, (say 18) that fall in between, we follow these steps:

1. Get the appropriate position. By applying some logics.
2. Copy the new records prior to determined position to a temporary file say temp.dat.
3. Append the new record in the temporary file temp.dat(Opened in append mode(a)).
4. Now append the rest of the record in the temporary file temp.dat
5. Delete file student.dat by issuing command.
   ```
   remove("student.dat")
   ```
6. Rename file temp.dat as student.dat as follows:
   ```
   rename("temp.dat", "student.dat")
   ```

```
import os
infile = open("marks.dat", "r")
outfile = open("temp.dat", "w")
nrollno = int(input("Enter rollno :"))
nname= input("Enter name :")
nmarks = float(input("Enter marks :"))
nrec = str(nrollno)+ "," + nname + "," + str(nmarks)+"\n"
count=0
rec=""
while rec:
    rec = infile.readline()
    count += 1
    if count == 5:
           outfile.write(nrec)
           outfile.write(rec)
    else:
            outfile.write(rec)
infile.close()
outfile.close()
os.remove("marks.dat")
os.rename("temp.dat", "marks.dat")
print("New record sucessfully added")
```