Introduction to Programming

Week – *9*, Lecture – *3*

File Handling in C – Part 2

SAURABH SRIVASTAVA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

IIT KANPUR

Random Access within a File in C

What we discussed in the previous part of the lecture is essentially “sequential” access of a file…

- … i.e. we saw how we can write data in a sequence, and then, read it back in the same fashion

Random Access within a File in C

What we discussed in the previous part of the lecture is essentially “sequential” access of a file…

- … i.e. we saw how we can write data in a sequence, and then, read it back in the same fashion

Sometimes, you would like to access a part of the file directly, skipping through the content before it

- For instance, what if you just wanted to know the middle element of the list that we wrote?

Random Access within a File in C

What we discussed in the previous part of the lecture is essentially “sequential” access of a file…

- … i.e. we saw how we can write data in a sequence, and then, read it back in the same fashion

Sometimes, you would like to access a part of the file directly, skipping through the content before it

- For instance, what if you just wanted to know the middle element of the list that we wrote?

There are two functions which can help you in this regard – `ftell()` and `fseek()`

Random Access within a File in C

What we discussed in the previous part of the lecture is essentially “sequential” access of a file…

- … i.e. we saw how we can write data in a sequence, and then, read it back in the same fashion

Sometimes, you would like to access a part of the file directly, skipping through the content before it

- For instance, what if you just wanted to know the middle element of the list that we wrote?

There are two functions which can help you in this regard – `ftell()` and `fseek()`

The `ftell()` function can tell you the byte offset within the file…

- … where the next read or write will be done

Random Access within a File in C

What we discussed in the previous part of the lecture is essentially “sequential” access of a file…

- … i.e. we saw how we can write data in a sequence, and then, read it back in the same fashion

Sometimes, you would like to access a part of the file directly, skipping through the content before it

- For instance, what if you just wanted to know the middle element of the list that we wrote?

There are two functions which can help you in this regard – `ftell()` and `fseek()`

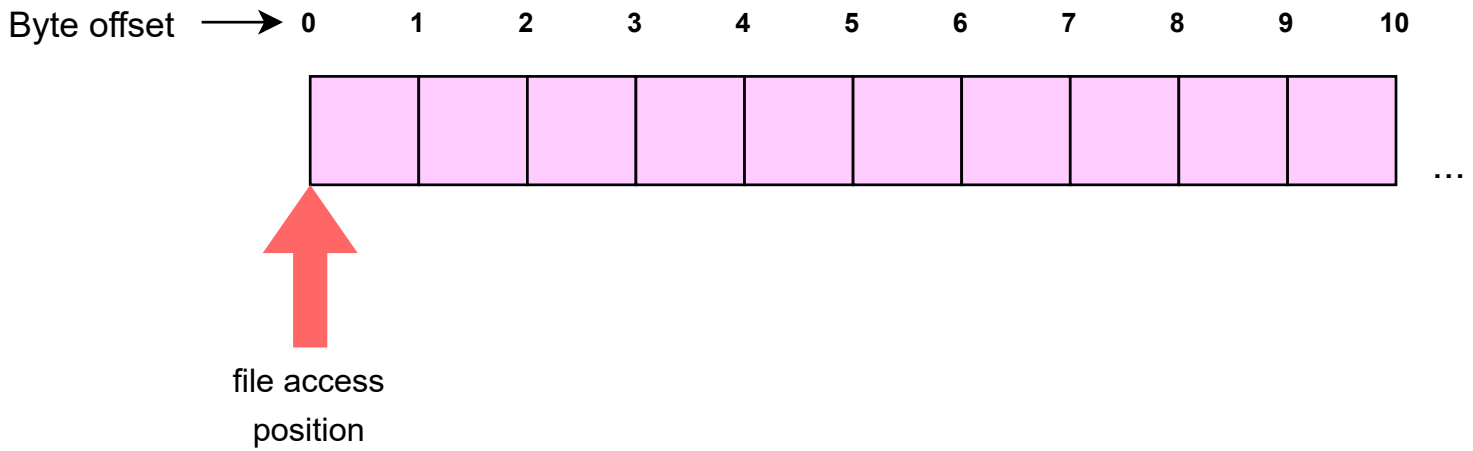
The `ftell()` function can tell you the byte offset within the file…

- … where the next read or write will be done

The `fseek()` function can set this offset to any random position in the file…

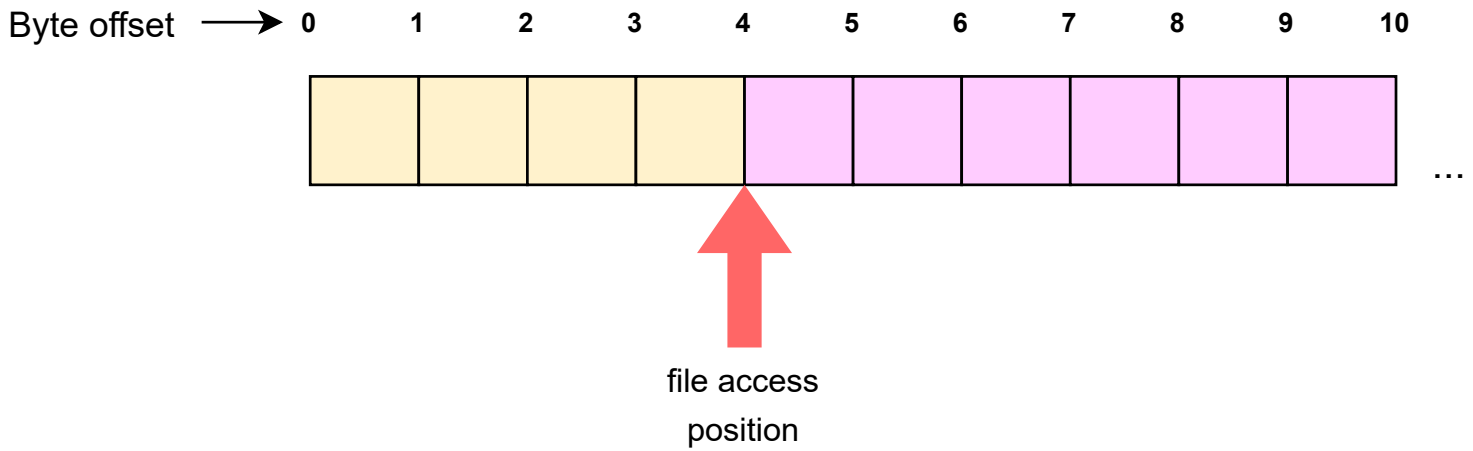
- … so the next read or write occurs from that position

Understanding read/write offset in a file



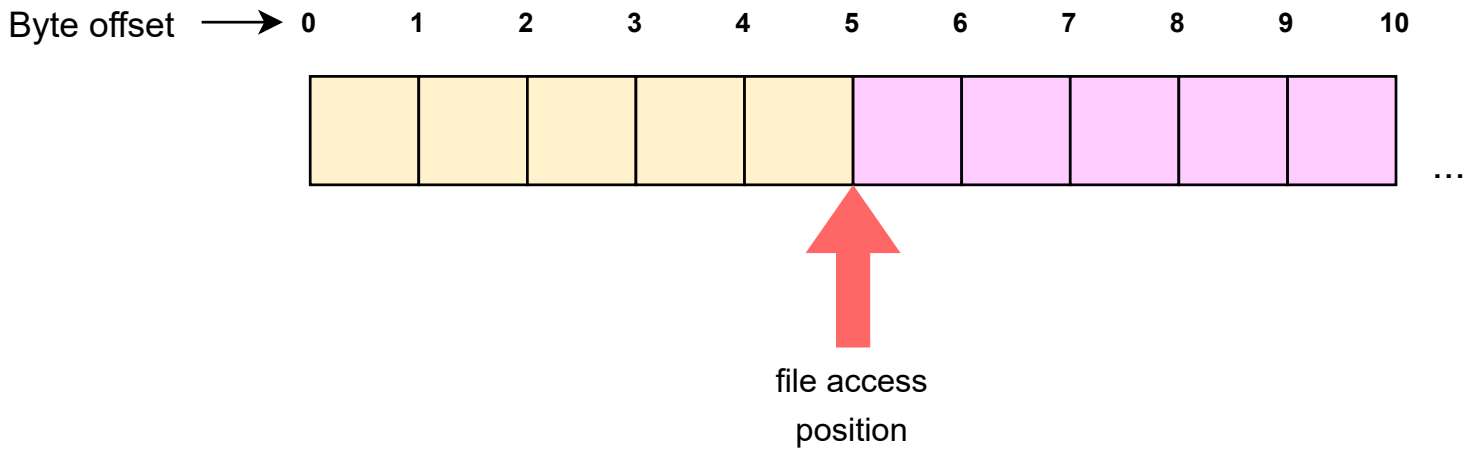
When the file is opened, the access position is at the offset 0

Understanding read/write offset in a file



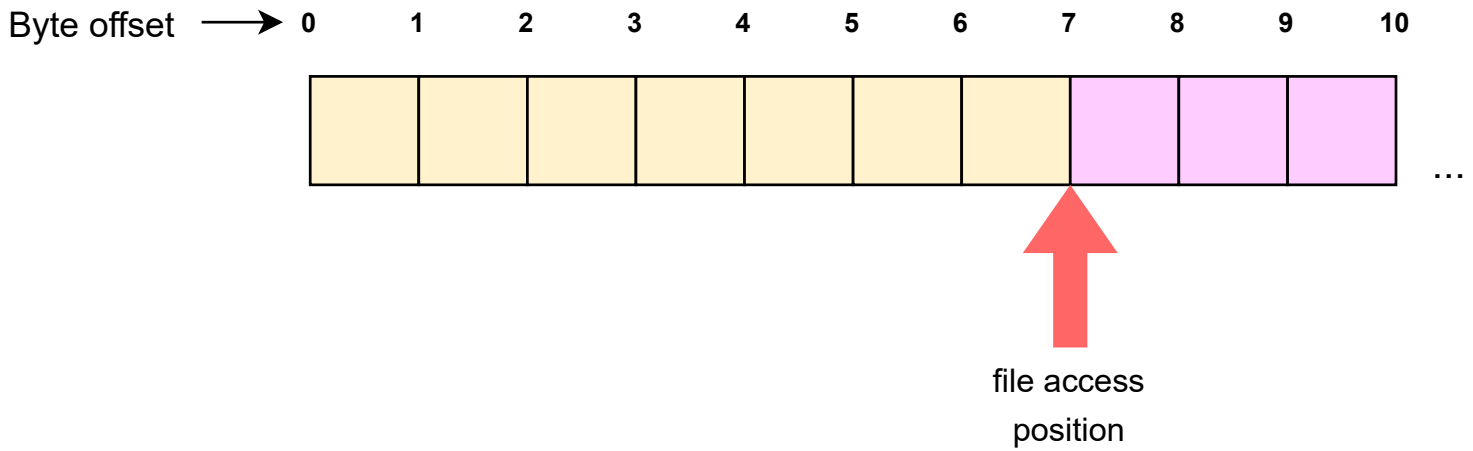
After reading or writing an `int`, the position changes to offset `4`

Understanding read/write offset in a file



Then, reading or writing a `char` will change the position to offset `5`

Understanding read/write offset in a file



Following it by reading or writing a `short` will further change the position to offset `7`, and so on…

Example – Modifying the list of integers

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ gcc -o ListSaver ListSaver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListSaver
Enter the number of integers in your list: 5
Enter the elements of the list:
7
11
3
29
23
Enter the name of the file to store the list (max 20 chars): list.bin
Saved your list in list.bin. Use the ListReader program to read it.
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$
```

Example – Modifying the list of integers

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ gcc -o ListSaver ListSaver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListSaver
Enter the number of integers in your list: 5
Enter the elements of the list:
7
11
3
29
23
Enter the name of the file to store the list (max 20 chars): list.bin
Saved your list in list.bin. Use the ListReader program to read it.
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$
```

Remember, we created this list with the `ListSaver` program; let us try to modify some element of the list

Example – Modifying the list of integers

```
fptr = fopen(file_name, "rb+"); //"r" - read, "b" - binary, "+" - allow write

// Read the number of elements in the list
fread(&number_of_elements, sizeof(int), 1, fptr);

if(number_of_elements > 0)
{
        printf("Number of elements: %d\n", number_of_elements);
        printf("The current position in the file is byte#%ld\n", ftell(fptr));
        printf("Which element should I fetch (starting from 1)?\n");
        scanf("%d", &index);
        if(index > 0 && index < number_of_elements)
        {
                fseek(fptr, (index-1)*sizeof(int), SEEK_CUR);
                fread(&temp, sizeof(int), 1, fptr);
                printf("The element at that position is %d\n", temp);
                printf("Do you want to change it? Y/N: ");
                clean_stdin();
                choice = getchar();
                if(choice == 'y' || choice == 'Y')
                {
                        printf("Enter the new integer: ");
                        scanf("%d", &temp);
                        fseek(fptr, -sizeof(int), SEEK_CUR);
                        fwrite(&temp, sizeof(int), 1, fptr);
                        printf("Done !!\n");
                }
        }
        else
                printf("Invalid index\n");
}
```

Example – Modifying the list of integers

```
fptr = fopen(file_name, "rb+"); //"r" - read, "b" - binary, "+" - allow write

// Read the number of elements in the list
fread(&number_of_elements, sizeof(int), 1, fptr);

if(number_of_elements > 0)
{
        printf("Number of elements: %d\n", number_of_elements);
        printf("The current position in the file is byte#%ld\n", ftell(fptr));
        printf("Which element should I fetch (starting from 1)?\n");
        scanf("%d", &index);
        if(index > 0 && index < number_of_elements)
        {
                fseek(fptr, (index-1)*sizeof(int), SEEK_CUR);
                fread(&temp, sizeof(int), 1, fptr);
                printf("The element at that position is %d\n", temp);
                printf("Do you want to change it? Y/N: ");
                clean_stdin();
                choice = getchar();
                if(choice == 'y' || choice == 'Y')
                {
                        printf("Enter the new integer: ");
                        scanf("%d", &temp);
                        fseek(fptr, -sizeof(int), SEEK_CUR);
                        fwrite(&temp, sizeof(int), 1, fptr);
                        printf("Done !!\n");
                }
        }
        else
                printf("Invalid index\n");
}
```

We start by reading the number of elements in the list

Example – Modifying the list of integers

```
fptr = fopen(file_name, "rb+"); //"r" - read, "b" - binary, "+" - allow write

// Read the number of elements in the list
fread(&number_of_elements, sizeof(int), 1, fptr);

if(number_of_elements > 0)
{
        printf("Number of elements: %d\n", number_of_elements);
        printf("The current position in the file is byte#%ld\n", ftell(fptr));
        printf("Which element should I fetch (starting from 1)?\n");
        scanf("%d", &index);
        if(index > 0 && index < number_of_elements)
        {
                fseek(fptr, (index-1)*sizeof(int), SEEK_CUR);
                fread(&temp, sizeof(int), 1, fptr);
                printf("The element at that position is %d\n", temp);
                printf("Do you want to change it? Y/N: ");
                clean_stdin();
                choice = getchar();
                if(choice == 'y' || choice == 'Y')
                {
                        printf("Enter the new integer: ");
                        scanf("%d", &temp);
                        fseek(fptr, -sizeof(int), SEEK_CUR);
                        fwrite(&temp, sizeof(int), 1, fptr);
                        printf("Done !!\n");
                }
        }
        else
                printf("Invalid index\n");
}
```

Next, we ask the user for the exact element that must be read from the file

Example – Modifying the list of integers

```
fptr = fopen(file_name, "rb+"); //"r" - read, "b" - binary, "+" - allow write

// Read the number of elements in the list
fread(&number_of_elements, sizeof(int), 1, fptr);

if(number_of_elements > 0)
{
        printf("Number of elements: %d\n", number_of_elements);
        printf("The current position in the file is byte#%ld\n", ftell(fptr));
        printf("Which element should I fetch (starting from 1)?\n");
        scanf("%d", &index);
        if(index > 0 && index < number_of_elements)
        {
                fseek(fptr, (index-1)*sizeof(int), SEEK_CUR);
                fread(&temp, sizeof(int), 1, fptr);
                printf("The element at that position is %d\n", temp);
                printf("Do you want to change it? Y/N: ");
                clean_stdin();
                choice = getchar();
                if(choice == 'y' || choice == 'Y')
                {
                        printf("Enter the new integer: ");
                        scanf("%d", &temp);
                        fseek(fptr, -sizeof(int), SEEK_CUR);
                        fwrite(&temp, sizeof(int), 1, fptr);
                        printf("Done !!\n");
                }
        }
        else
                printf("Invalid index\n");
}
```

Next, we ask the user for the exact element that must be read from the file

Note that we have not made any attempt to read the whole list

Example – Modifying the list of integers

```
fptr = fopen(file_name, "rb+"); //"r" - read, "b" - binary, "+" - allow write

// Read the number of elements in the list
fread(&number_of_elements, sizeof(int), 1, fptr);

if(number_of_elements > 0)
{
        printf("Number of elements: %d\n", number_of_elements);
        printf("The current position in the file is byte#%ld\n", ftell(fptr));
        printf("Which element should I fetch (starting from 1)?\n");
        scanf("%d", &index);
        if(index > 0 && index < number_of_elements)
        {
                fseek(fptr, (index-1)*sizeof(int), SEEK_CUR);
                fread(&temp, sizeof(int), 1, fptr);
                printf("The element at that position is %d\n", temp);
                printf("Do you want to change it? Y/N: ");
                clean_stdin();
                choice = getchar();
                if(choice == 'y' || choice == 'Y')
                {
                        printf("Enter the new integer: ");
                        scanf("%d", &temp);
                        fseek(fptr, -sizeof(int), SEEK_CUR);
                        fwrite(&temp, sizeof(int), 1, fptr);
                        printf("Done !!\n");
                }
        }
        else
                printf("Invalid index\n");
}
```

The `fseek()` function can take us to a particular byte offset in the file

Example – Modifying the list of integers

```
fptr = fopen(file_name, "rb+"); //"r" - read, "b" - binary, "+" - allow write

// Read the number of elements in the list
fread(&number_of_elements, sizeof(int), 1, fptr);

if(number_of_elements > 0)
{
        printf("Number of elements: %d\n", number_of_elements);
        printf("The current position in the file is byte#%ld\n", ftell(fptr));
        printf("Which element should I fetch (starting from 1)?\n");
        scanf("%d", &index);
        if(index > 0 && index < number_of_elements)
        {
                fseek(fptr, (index-1)*sizeof(int), SEEK_CUR);
                fread(&temp, sizeof(int), 1, fptr);
                printf("The element at that position is %d\n", temp);
                printf("Do you want to change it? Y/N: ");
                clean_stdin();
                choice = getchar();
                if(choice == 'y' || choice == 'Y')
                {
                        printf("Enter the new integer: ");
                        scanf("%d", &temp);
                        fseek(fptr, -sizeof(int), SEEK_CUR);
                        fwrite(&temp, sizeof(int), 1, fptr);
                        printf("Done !!\n");
                }
        }
        else
                printf("Invalid index\n");
}
```

The `fseek()` function can take us to a particular byte offset in the file

To go to the beginning of the i^{th} element in the list (i >= 1), we will need to skip (i-1) integers

Example – Modifying the list of integers

```
fptr = fopen(file_name, "rb+"); //"r" - read, "b" - binary, "+" - allow write

// Read the number of elements in the list
fread(&number_of_elements, sizeof(int), 1, fptr);

if(number_of_elements > 0)
{
        printf("Number of elements: %d\n", number_of_elements);
        printf("The current position in the file is byte#%ld\n", ftell(fptr));
        printf("Which element should I fetch (starting from 1)?\n");
        scanf("%d", &index);
        if(index > 0 && index < number_of_elements)
        {
                fseek(fptr, (index-1)*sizeof(int), SEEK_CUR);
                fread(&temp, sizeof(int), 1, fptr);
                printf("The element at that position is %d\n", temp);
                printf("Do you want to change it? Y/N: ");
                clean_stdin();
                choice = getchar();
                if(choice == 'y' || choice == 'Y')
                {
                        printf("Enter the new integer: ");
                        scanf("%d", &temp);
                        fseek(fptr, -sizeof(int), SEEK_CUR);
                        fwrite(&temp, sizeof(int), 1, fptr);
                        printf("Done !!\n");
                }
        }
        else
                printf("Invalid index\n");
}
```

Then we read i^{th} element and show it (just that element, none other)

Example – Modifying the list of integers

```
fptr = fopen(file_name, "rb+"); //"r" - read, "b" - binary, "+" - allow write

// Read the number of elements in the list
fread(&number_of_elements, sizeof(int), 1, fptr);

if(number_of_elements > 0)
{
        printf("Number of elements: %d\n", number_of_elements);
        printf("The current position in the file is byte#%ld\n", ftell(fptr));
        printf("Which element should I fetch (starting from 1)?\n");
        scanf("%d", &index);
        if(index > 0 && index < number_of_elements)
        {
                fseek(fptr, (index-1)*sizeof(int), SEEK_CUR);
                fread(&temp, sizeof(int), 1, fptr);
                printf("The element at that position is %d\n", temp);
                printf("Do you want to change it? Y/N: ");
                clean_stdin();
                choice = getchar();
                if(choice == 'y' || choice == 'Y')
                {
                        printf("Enter the new integer: ");
                        scanf("%d", &temp);
                        fseek(fptr, -sizeof(int), SEEK_CUR);
                        fwrite(&temp, sizeof(int), 1, fptr);
                        printf("Done !!\n");
                }
        }
        else
                printf("Invalid index\n");
}
```

Then we read i^{th} element and show it (just that element, none other)

We ask if the user wishes to update that element to something else…

Example – Modifying the list of integers

```
fptr = fopen(file_name, "rb+"); //"r" - read, "b" - binary, "+" - allow write

// Read the number of elements in the list
fread(&number_of_elements, sizeof(int), 1, fptr);

if(number_of_elements > 0)
{
        printf("Number of elements: %d\n", number_of_elements);
        printf("The current position in the file is byte#%ld\n", ftell(fptr));
        printf("Which element should I fetch (starting from 1)?\n");
        scanf("%d", &index);
        if(index > 0 && index < number_of_elements)
        {
                fseek(fptr, (index-1)*sizeof(int), SEEK_CUR);
                fread(&temp, sizeof(int), 1, fptr);
                printf("The element at that position is %d\n", temp);
                printf("Do you want to change it? Y/N: ");
                clean_stdin();
                choice = getchar();
                if(choice == 'y' || choice == 'Y')
                {
                        printf("Enter the new integer: ");
                        scanf("%d", &temp);
                        fseek(fptr, -sizeof(int), SEEK_CUR);
                        fwrite(&temp, sizeof(int), 1, fptr);
                        printf("Done !!\n");
                }
        }
        else
                printf("Invalid index\n");
}
```

… if yes, then we must “rewind” back by one integer (i.e., usually 4 bytes) from the current position…

Example – Modifying the list of integers

```
fptr = fopen(file_name, "rb+"); //"r" - read, "b" - binary, "+" - allow write

// Read the number of elements in the list
fread(&number_of_elements, sizeof(int), 1, fptr);

if(number_of_elements > 0)
{
        printf("Number of elements: %d\n", number_of_elements);
        printf("The current position in the file is byte#%ld\n", ftell(fptr));
        printf("Which element should I fetch (starting from 1)?\n");
        scanf("%d", &index);
        if(index > 0 && index < number_of_elements)
        {
                fseek(fptr, (index-1)*sizeof(int), SEEK_CUR);
                fread(&temp, sizeof(int), 1, fptr);
                printf("The element at that position is %d\n", temp);
                printf("Do you want to change it? Y/N: ");
                clean_stdin();
                choice = getchar();
                if(choice == 'y' || choice == 'Y')
                {
                        printf("Enter the new integer: ");
                        scanf("%d", &temp);
                        fseek(fptr, -sizeof(int), SEEK_CUR);
                        fwrite(&temp, sizeof(int), 1, fptr);
                        printf("Done !!\n");
                }
        }
        else
                printf("Invalid index\n");
}
```

… if yes, then we must “rewind” back by one integer (i.e., usually 4 bytes) from the current position…

… and then write the new integer at that location

Example – Modifying the list of integers

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListReader
Enter the name of the file containing the list (max 20 chars): list.bin
This is what I read:
7
11
3
29
23
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ gcc -o ListElementModifier ListElementModifier.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListElementModifier
Enter the name of the file containing the list (max 20 chars): list.bin
Number of elements: 5
The current position in the file is byte#4
Which element should I fetch (starting from 1)?
4
The element at that position is 29
Do you want to change it? Y/N: Y
Enter the new integer: 41
Done !!
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListReader
Enter the name of the file containing the list (max 20 chars): list.bin
This is what I read:
7
11
3
41
23
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$
```

Example – Modifying the list of integers

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListReader
Enter the name of the file containing the list (max 20 chars): list.bin
This is what I read:
7
11
3
29
23
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ gcc -o ListElementModifier ListElementModifier.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListElementModifier
Enter the name of the file containing the list (max 20 chars): list.bin
Number of elements: 5
The current position in the file is byte#4
Which element should I fetch (starting from 1)?
4
The element at that position is 29
Do you want to change it? Y/N: Y
Enter the new integer: 41
Done !!
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListReader
Enter the name of the file containing the list (max 20 chars): list.bin
This is what I read:
7
11
3
41
23
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$
```

This is what the list looked initially

Example – Modifying the list of integers

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListReader
Enter the name of the file containing the list (max 20 chars): list.bin
This is what I read:
7
11
3
29
23
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ gcc -o ListElementModifier ListElementModifier.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListElementModifier
Enter the name of the file containing the list (max 20 chars): list.bin
Number of elements: 5
The current position in the file is byte#4
Which element should I fetch (starting from 1)?
4
The element at that position is 29
Do you want to change it? Y/N: Y
Enter the new integer: 41
Done !!
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListReader
Enter the name of the file containing the list (max 20 chars): list.bin
This is what I read:
7
11
3
41
23
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$
```

We then change the fourth element in the list

Example – Modifying the list of integers

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListReader
Enter the name of the file containing the list (max 20 chars): list.bin
This is what I read:
7
11
3
29
23
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ gcc -o ListElementModifier ListElementModifier.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListElementModifier
Enter the name of the file containing the list (max 20 chars): list.bin
Number of elements: 5
The current position in the file is byte#4
Which element should I fetch (starting from 1)?
4
The element at that position is 29
Do you want to change it? Y/N: Y
Enter the new integer: 41
Done !!
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./ListReader
Enter the name of the file containing the list (max 20 chars): list.bin
This is what I read:
7
11
3
41
23
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$
```

… and the changes persist in the file

Writing Structures to Files

We cover one more common task that you'll use quite often – writing structures to files

Writing Structures to Files

We cover one more common task that you'll use quite often – writing structures to files

Remember how we talked about some sort of format, when we deal with binary data?

- One simple way to organise different type of data, while writing to a file, is to enclose it in a structure

Writing Structures to Files

We cover one more common task that you'll use quite often – writing structures to files

Remember how we talked about some sort of format, when we deal with binary data?

- One simple way to organise different type of data, while writing to a file, is to enclose it in a structure

The common practice is to write structure variables in a file, one followed by another

Writing Structures to Files

We cover one more common task that you'll use quite often – writing structures to files

Remember how we talked about some sort of format, when we deal with binary data?

- One simple way to organise different type of data, while writing to a file, is to enclose it in a structure

The common practice is to write structure variables in a file, one followed by another

- In this case, you do not need to read or write individual member variables of the structure…
- … instead, you can write and read them in one go

Reading/Writing structure variables

```
struct sample
{
    int data1;
    char data2;
    short data3;
};
```

Reading/Writing structure variables

```
struct sample
{
    int data1;
    char data2;
    short data3;
};
```

Assume that we have a structure with three member variables

Reading/Writing structure variables

```
struct sample
{
    int data1;
    char data2;
    short data3;
};
```

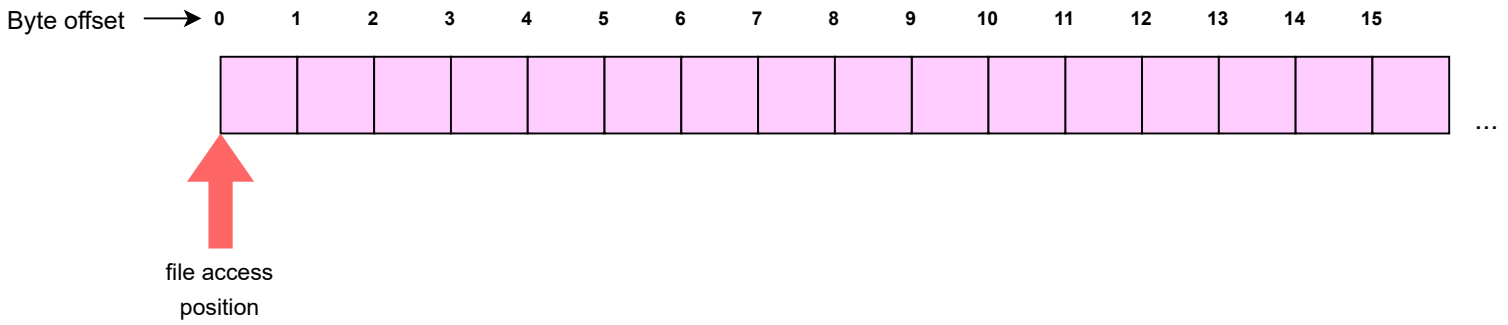
Assume that we have a structure with three member variables

The space that a structure variable of type sample needs (without any padding) is 7 bytes

Reading/Writing structure variables

```
struct sample
{
    int data1;
    char data2;
    short data3;
};
```

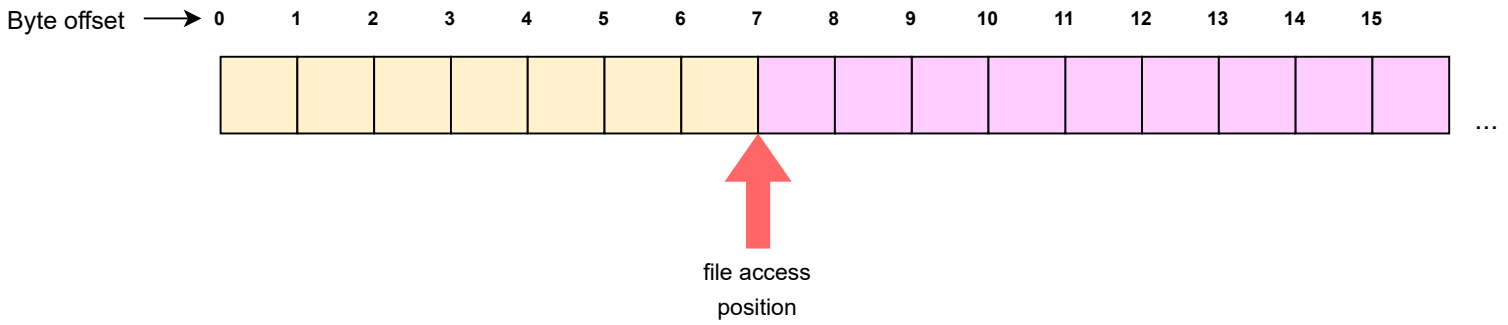
Let us say we wish to write them in a file



Reading/Writing structure variables

```
struct sample
{
    int data1;
    char data2;
    short data3;
};
```

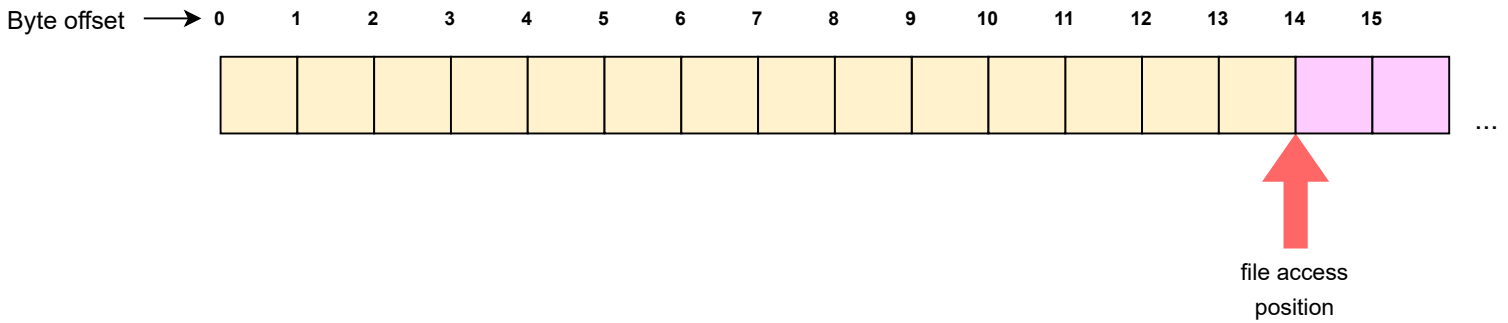
After writing one structure variable by calling `fwrite()` once, the byte offset changes to 7



Reading/Writing structure variables

```
struct sample
{
    int data1;
    char data2;
    short data3;
};
```

Another call to `fwrite()` with another structure variable will take the offset to `14`… and so on

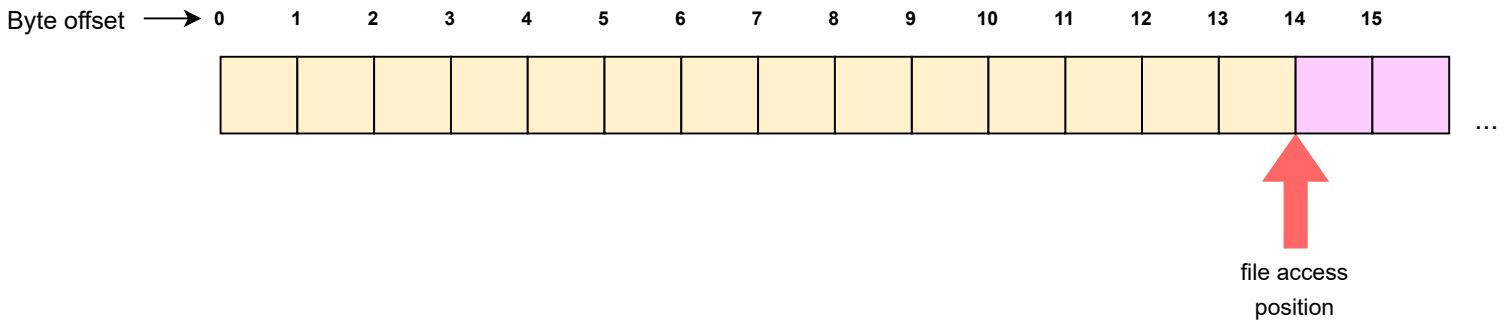


Reading/Writing structure variables

```
struct sample
{
    int data1;
    char data2;
    short data3;
};
```

Another call to `fwrite()` with another structure variable will take the offset to `14`… and so on

The same is true for calls to `fread()` for reading a structure variable



Example – Writing a Car Catalog to a file

```c
#include<stdio.h>
#include<string.h>
#include "Car.h"

int main()
{
        Car cars[3];
        int i;
        FILE *fptr = fopen("catalog.bin", "wb");

        strcpy(cars[0].name, "Toyota Innova Crysta");
        cars[0].capacity = 8;
        cars[0].price_in_lakhs = 16.27;

        strcpy(cars[1].name, "Hyundai Creta");
        cars[1].capacity = 5;
        cars[1].price_in_lakhs = 10;

        strcpy(cars[2].name, "Kia Seltos");
        cars[2].capacity = 5;
        cars[2].price_in_lakhs = 9.9;

        printf("Creating the Cars Catalog\n");
        fwrite(cars, sizeof(Car), 3, fptr);
        printf("Done...\n");
}
```

Example – Writing a Car Catalog to a file

```c
#include<stdio.h>
#include<string.h>
#include "Car.h"

int main()
{
        Car cars[3];
        int i;
        FILE *fptr = fopen("catalog.bin", "wb");

        strcpy(cars[0].name, "Toyota Innova Crysta");
        cars[0].capacity = 8;
        cars[0].price_in_lakhs = 16.27;

        strcpy(cars[1].name, "Hyundai Creta");
        cars[1].capacity = 5;
        cars[1].price_in_lakhs = 10;

        strcpy(cars[2].name, "Kia Seltos");
        cars[2].capacity = 5;
        cars[2].price_in_lakhs = 9.9;

        printf("Creating the Cars Catalog\n");
        fwrite(cars, sizeof(Car), 3, fptr);
        printf("Done...\n");
}
```

This writes the whole array of structure variables in the file

Example – Writing a Car Catalog to a file

```c
#include<stdio.h>
#include<string.h>
#include "Car.h"

int main()
{
        Car cars[3];
        int i;
        FILE *fptr = fopen("catalog.bin", "wb");

        strcpy(cars[0].name, "Toyota Innova Crysta");
        cars[0].capacity = 8;
        cars[0].price_in_lakhs = 16.27;

        strcpy(cars[1].name, "Hyundai Creta");
        cars[1].capacity = 5;
        cars[1].price_in_lakhs = 10;

        strcpy(cars[2].name, "Kia Seltos");
        cars[2].capacity = 5;
        cars[2].price_in_lakhs = 9.9;

        printf("Creating the Cars Catalog\n");
        fwrite(cars, sizeof(Car), 3, fptr);
        printf("Done...\n");
}
```

This writes the whole array of structure variables in the file

As long as you plan to use the binary data file on the same system, you don't need to worry about how individual members of the structure are stored in the file

Example – Reading the Catalog back

```c
#include<stdio.h>
#include "Car.h"

int main()
{
        Car car;
        int i;
        FILE *fptr = fopen("catalog.bin", "rb");

        printf("Reading the Cars Catalog\n");

        for(i = 0; i < 3; i++)
        {
                fread(&car, sizeof(Car), 1, fptr);
                printf("-------------------------\n");
                printf("%s\n", car.name);
                printf("Capacity: %d people\n", car.capacity);
                printf("Price: %05.2f lakhs\n", car.price_in_lakhs);
                printf("-------------------------\n");
        }
}
```

Example – Reading the Catalog back

```c
#include<stdio.h>
#include "Car.h"

int main()
{
        Car car;
        int i;
        FILE *fptr = fopen("catalog.bin", "rb");

        printf("Reading the Cars Catalog\n");

        for(i = 0; i < 3; i++)
        {
                fread(&car, sizeof(Car), 1, fptr);
                printf("-------------------------\n");
                printf("%s\n", car.name);
                printf("Capacity: %d people\n", car.capacity);
                printf("Price: %05.2f lakhs\n", car.price_in_lakhs);
                printf("-------------------------\n");
        }
}
```

… and that's how we read it back

Example – Reading the Catalog back

```c
#include<stdio.h>
#include "Car.h"

int main()
{
        Car car;
        int i;
        FILE *fptr = fopen("catalog.bin", "rb");

        printf("Reading the Cars Catalog\n");

        for(i = 0; i < 3; i++)
        {
                fread(&car, sizeof(Car), 1, fptr);
                printf("-------------------------\n");
                printf("%s\n", car.name);
                printf("Capacity: %d people\n", car.capacity);
                printf("Price: %05.2f lakhs\n", car.price_in_lakhs);
                printf("-------------------------\n");
        }
}
```

… and that's how we read it back

Again, we don't need to read individual member variables; reading it directly into a structure variable works !!

Example – Reading the Catalog back

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ gcc -o CarsCatalogCreator CarsCatalogCreator.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./CarsCatalogCreator
Creating the Cars Catalog
Done...
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ gcc -o CarsCatalogReader CarsCatalogReader.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./CarsCatalogReader
Reading the Cars Catalog
-------------------------
Toyota Innova Crysta
Capacity: 8 people
Price: 16.27 lakhs
-------------------------
-------------------------
Hyundai Creta
Capacity: 5 people
Price: 10.00 lakhs
-------------------------
-------------------------
Kia Seltos
Capacity: 5 people
Price: 09.90 lakhs
-------------------------
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$
```

Example – Reading the Catalog back

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ gcc -o CarsCatalogCreator CarsCatalogCreator.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./CarsCatalogCreator
Creating the Cars Catalog
Done...
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ gcc -o CarsCatalogReader CarsCatalogReader.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$ ./CarsCatalogReader
Reading the Cars Catalog
-------------------------
Toyota Innova Crysta
Capacity: 8 people
Price: 16.27 lakhs
-------------------------
-------------------------
Hyundai Creta
Capacity: 5 people
Price: 10.00 lakhs
-------------------------
-------------------------
Kia Seltos
Capacity: 5 people
Price: 09.90 lakhs
-------------------------
saurabh@saurabh-VirtualBox:~/C/examples/Week 9$
```

Something like this…

Homework !!

Write another program called `CarCatalogModifier.c`, which shows the user all the cars…

- … then allow her to change any one Car's details (e.g. change its name to something else)
- Use `fseek()` elegantly for this !!

Additional Reading

It is tricky to write structures with pointers because only the address gets written…

- … not the content it points to

In such case, you may have to write individual member variables of a structure separately

Read more about such cases

- This page has some good discussion on this: https://stackoverflow.com/questions/2763438/write-pointer-to-file-in-c