**Day-13(12-11-2024)**

* **Union:**
* Only one member of the union can only access at a time.
* It takes the size of the union of the large size of the data type
* Union item{

Int m;

Float x;

Char c;

};

* Here the size of the float(4 bytes) is more than the other members in the union so,the size of union is 4 bytes
* Program

// Online C compiler to run C program online

#include <stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct employee{

int id;

float sal;

char g;

char name[20];

} sEMP;

typedef union uemployee{

int id;

int sal;

char g;

char name[20];

} uEMP;

int main() {

printf("size of structure:%d\n",sizeof(sEMP));

printf("size of union:%d\n",sizeof(uEMP));

sEMP es;

uEMP eu;

uEMP \*eu1=NULL;

eu.id=101;

eu.sal=10000;

strcpy(eu.name,"ramyabalivada");

printf("%s\n",eu.name);

eu1=(uEMP\*)(malloc(sizeof(uEMP)));

eu1->id=103;

printf("%d\n",eu1->id);

return 0;

}

Output: size of structure:32

size of union:20

ramyabalivada

103

* **ENUM:**
* It a set of named integer constants specify all the legal values a variable of that type can have
* Syntax:

Enum enum-type-name{enumeration list} variable\_list;

* Enum always start with ‘0’

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Enomem is same as a enum
* **File handling:**
* A collection of logically related information.
* It is permanent
* There are start of the file and end of the file indicator
* Floppy disk(permanent storage)
* If you give the command cat it is in ram and it checks the files and takes the address using file pointer
* FILE is structure it contains file pointers
* Eg: an employee file with employee names,designation,salary
* Two types:

1. Sequential file :all files are in particular order
2. Random access file: all files are in random access
   * + - In c ,in file you can write anywhere
       - This structure to which the file pointer point to is of type FILE defined in the header file <stdio.h>
       - The only declaration needed for a file pointer is exemplified by:
3. FILE\*fp;
4. FILE \*fopen(char\*name,char\*mode);
5. fp=fopen(“file name”,”mode”);

* modes 🡪read only,writeonly,read write,append(add at the end of the file)
* The fopen() function opens the file whose name is the string pointed to by pathname and associates a stream with it.
* The argument mode points to a string beginning with one of the following sequences (possibly followed by additional characters, as described below):
* r 🡺 Open text file for reading. The stream is positioned at the beginning of the file.
* r+ 🡺 Open for reading and writing. The stream is positioned at the beginning of the file.
* w 🡺 Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.
* w+ 🡺 Open for reading and writing. The file is created if it doesnot exist, otherwise it is truncated. The stream is positioned at the beginning of the file.
* a 🡺 Open for appending (writing at end of file). The file is created if it does not exist. The stream is positioned at the end of the file.
* a+ 🡺 Open for reading and appending (writing at end of file). The file is created if it does not exist. The initial file position for reading is at the beginning of the file, but output is always appended to the end of the file.
* rb,wb🡺 binary read,binary write
  + - * once the function fopen() returns a FILE type pointer stored in a pointer of type FILE,this pointer becomes the medium through which all subsequent i/o can be performed
      * while trying to open the file if the file returns the address then it means file is present if it returns null it means file is not exist.
      * fseek is used to traverse the file
      * fclose 🡺 int fclose(FILE \*stream);
      * fprintf,fscanf🡺 formatted writing,reading respectively
      * fputs,fgets🡺 unformatted i/o operations
      * fwrite/fread🡺 reading or writing binary objects(whole structure or content)

A screen shot of a computer program

Description automatically generated

* + - * char \*fgets(char \*s, int size, FILE \*stream);
      * int fputc(int c, FILE \*stream);
      * int fputs(const char \*s, FILE \*stream);
      * fgetc() reads the next character from stream and returns it as an unsigned char cast to an int, or EOF on end of file or error.

int fgetc(FILE \*stream);

* + - * ctrl+c is used to stop the stream
      * for reading:

char ch;

while((ch)=fgetc(fd)!=EOF)

{

Fputc(ch,fd)

}

* + - * question: read the contents from the database the record of the employees

101|amit kumar| M| 8888|100001

101|amit kumar| M| 8888|100002

101|amit kumar| M| 8888|100001

Read the content from the file store in the structure and display the structure using fgets(it gives new line at the end so we use the null at the end)

And also give the value according to the number of records in memory allocation

**System programming**

* In system we cannot the execute certain linux commands
* Execl , excelp is used to execute a file
* Fseek-tells the position of the cursor in integer(or after which byte your pointer is positioned)
* The ftell() function in C is used to determine the current position of the file pointer within a file. It returns the position as a long integer, which represents the number of bytes from the beginning of the file.
* fseek () - It is used to moves the reading control to different positions using fseek function. ftell () - It **tells the byte location of current position in file pointer**. rewind () - It moves the control to beginning of a file.
* Fseek:

Int fseek(file \*stream,long offset,int whence);

SEEK\_SET,SEEK\_CUR,SEEK\_END in whence

* Here the offset become ‘0’
* Fseek(fp,0,SEEK\_END);🡺to know how many number of bytes in that particular file.
* Fseek(fp,-12,SEEK\_cur); 🡺 we can write the offset value negative and positive values
* Rewind:

(Void) fseek(stream,0L,SEEK\_SET);

* Errors is einval 🡺 EINVAL The whence argument to fseek() was not SEEK\_SET, SEEK\_END, or SEEK\_CUR. Or: the resulting file offset would be negative.
* Fread
  + - * reading or giving the contents to ptr
      * size\_t fread(void \*ptr, size\_t size, size\_t nmemb, FILE \*stream);
      * The function fread() reads nmemb items of data, each size bytes long, from the stream pointed to by stream, storing them at the location given by ptr.
      * Read the Data from file to pointer
* Fwrite:
* Write the contents from the ptr
* size\_t fwrite(const void \*ptr, size\_t size, size\_t nmemb, FILE \*stream);
* write the data from pointer to file
* In printf and scanf we write and read the text but in fread and fwrite it gives the binary values
* On success, fread() and fwrite() return the number of items read or

written. This number equals the number of bytes transferred only when size is 1. If an error occurs, or the end of the file is reached, the return value is a short item count (or zero).fread() does not distinguish between end-of-file and error, and callers must use feof(3) and ferror(3) to determine which occurred.