Computer Systems Organization
CS-UH 2010

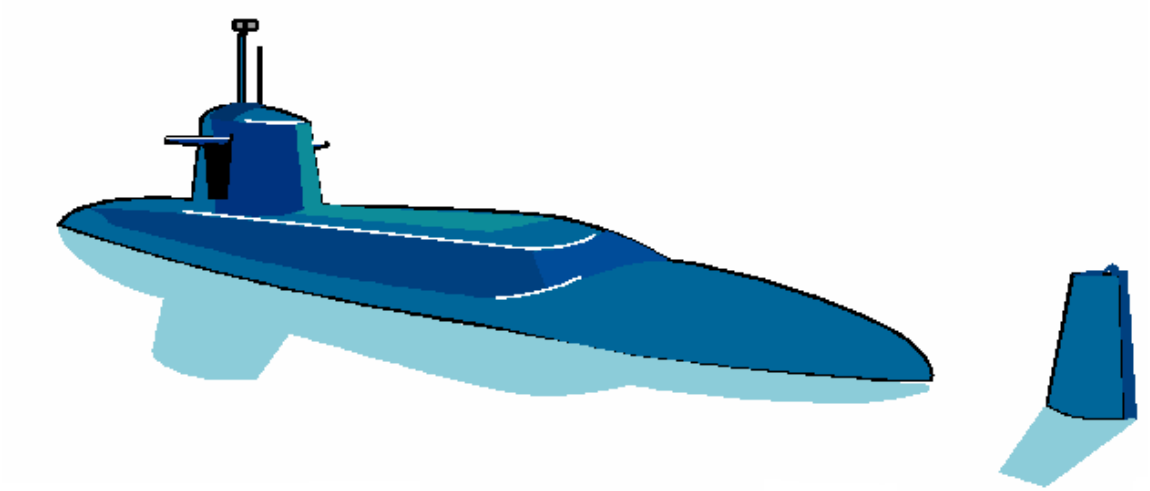
Recitation 2
Introduction to C

Khalid Mengal

Agenda

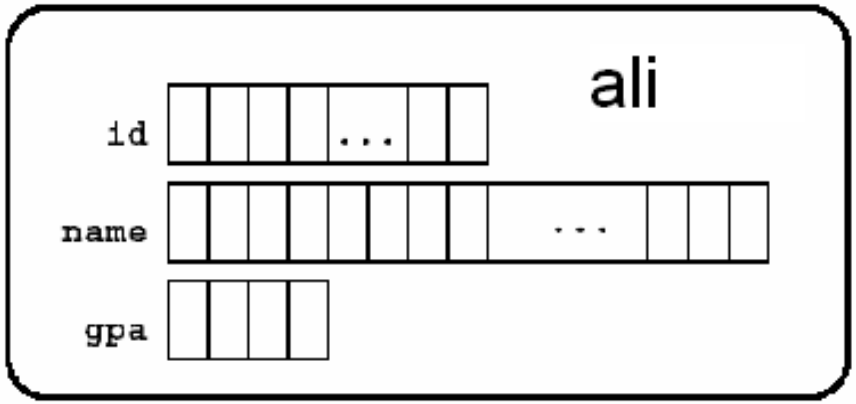
- Structures
- File Input/Output in C

Structures in C



Structures

- Colleciton of related data elemented
  - Possibily of different types
  - Share a single name



Struct type

- Structure type can be defined usign keyword struct

```
struct student
{
        char id[9];
        char name[26];
        float gpa;
};
```

NO variable has yet been defined

Declaring Structure variable

```
struct
{
        char id[9];
        char name[26];
        float gpa;
} ali;
```

- This defines a structure variable (ali)

Initializing Structures

```
struct student
{
                    char id[9];
                    char name[26];
                    float gpa;
};

struct student st1 = {"ak100","Ali Khan",3.5};
```

- Initial values should be constant values or constant expressions

Accessing members of structure

```
#include <stdio.h>
struct date
{
        int day;
        int month;
        int year;
};
int main(void)
{
    struct date today;
    today.day = 1;
    today.month = 1;
    today.year = 2003;
    if (today.day==1 && today.month==1)
        printf(“Happy New year");
    return 0;
}
```

Members of struct can be accessed using (.) operator

Type-Defined Structure

```
typedef struct
{
  char id[9];
  char name[26];
  float gpa;
} STUDENT;
```

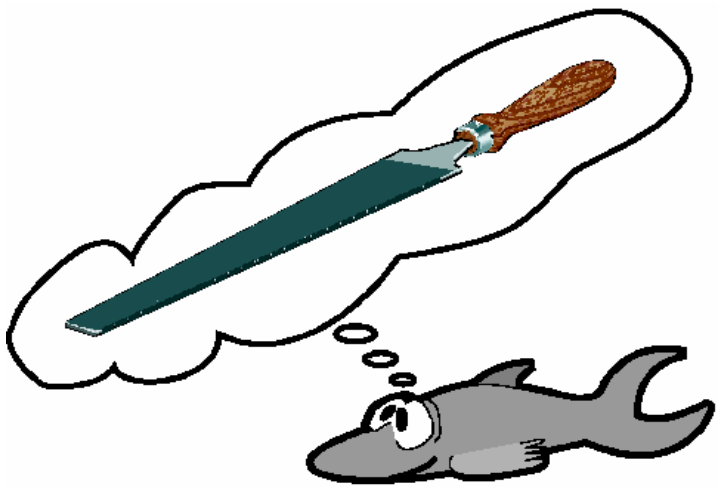
- This defines the data type STUDENT, which is a struct.
- Now we can write

  `STUDENT ali;`

  rather than

  `struct STUDENT ali;`

Filing in C



Opening a File

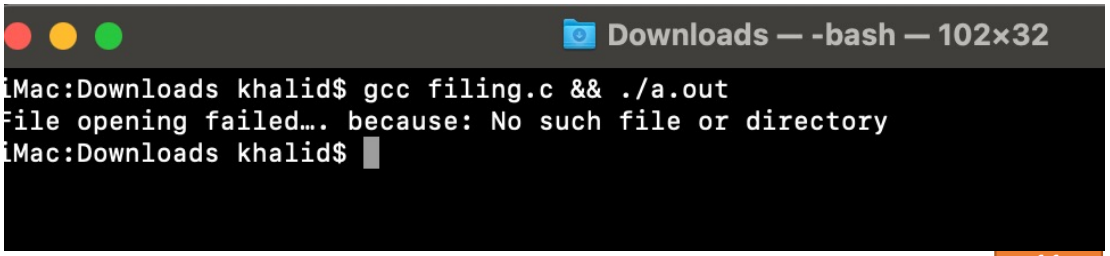
```
#include <stdio.h>
int main(void)
{
        FILE* fptr;
        fptr = fopen(”myfile.txt", "r");
```

Structure defined in Standard library (stido.h>

FILE* fopen(const char* name, const char* mode);

fopen() eturns NULL if it fails

```
FILE *fptr;
if((fptr = fopen("myfile.txt", "r")) == NULL)
{       perror(” File opening failed…. because");
        return 1;
}
```



File opening Modes

Modes for opening files	
"r"	open text file for reading, File must already exist
"w"	Open for writing/creation, if file already exist it data will be overwrite.
"a"	Open for append, data will be added in already exist file.
"rb"	open binary fie for reading
"wb"	open binary file for writing
"ab"	open binary file for appending

r, w and a can be appending with + character which means that the file is opened for both reading and writing.

Types of File I/O

- Character I/O
- String I/O
- Formatted I/O
- Record I/O

Character I/O

- Read/write a single character at a time
- fgetc: **Function to read a single character from Stream**

  ch = fgetc(in_stream)

- fputc: **Funciton to write a single character to the the stream**
  - fputc(ch,out_stream)

String I/O

- fgets: Funciton to read a string from stream
  - char* fgets(char* buffer, int size, FILE* stream);

- fputs: Fucntion to write a string to the stream
  - int fputs(const char* buffer, FILE* stream);

Formated I/O

- fscanf: funciton to read formated data from stream
  - int fscanf(FILE* stream, const char* format, ...);
  - fscanf(in_stream, "%I %f %39s", &j, &flt, word);

- fprintf: funciton to write formated data to stream
  - int fprintf(FILE* stream, const char* format, ...);
  - fprintf(out_stream, "%i %.1f %-39s", j, flt, word);

Record I/O, Binary I/O

- Read/Write a block of data at once, e.g. write a structure, array or string etc.

```
struct Student
   {
         char name[20];
         int  netid;
         char grade;
   } ;
struct Student st1={“ali”, 102, A}
fwrite(&st1, sizeof(st1), 1, ptr);   //1 = number of records
```

Record I/O (cont..)

- fread: a fucntion which reads from input stream and stores data in varaible/structures etc.

```
while( fread(&student,sizeof(student),1,ptr) == 1)
  {
   printf("Name        : %s\n",student.name);
   printf("Net I.d     : %d\n",student.netid);
   printf("Grade       : %c\n"student.grade);
   printf("----------------\n");
  }

fclose(ptr);
```

fseek(), ftell()

- **fseek():** Sets the position indicator associated with the *stream* to a new position.
  - int fseek (FILE * stream, long int offset, int origin);
  - stream: Pointer to a FILE object that identifies the stream.
  - offset:
    - Binary files: Number of bytes to offset from *origin*.
    - Text files: Either zero, or a value returned by ftell.
  - origin: Position used as reference for the *offset*
    - SEEK_SET Beginning of file
    - SEEK_CUR Current position of the file pointer
    - SEEK_END End of file *

- **ftell(): Returns the current value of the file position indicator for the file stream stream.**
    - Binary files: Returns the number of bytes from the beginning of the file.
    - Test files: unspecified, only meaningful as the input to std::fseek.

Example:

```
#include<stdio.h>
int main ()
{
   FILE *fp;
   int size;

   fp = fopen("myfile.txt", "r");
   if( fp != NULL )
   {
      fseek(fp, 0, SEEK_END);
      size = ftell(fp);
      fclose(fp);
      printf("Size of myfile.txt = %u bytes \n", size);
      fclose(fp);
   }
   return(0);
}
```