Unions in C

By: Yehia M. Abu Eita

Outlines

- Introduction
- Creating new union type
- Declaring a union variable
- Union variables in memory
- Accessing a union variable members

Introduction

- Union is a **non-primitive** data type that stores **different types of data**.
- Unions are used to **group different types of data together with a momentarily time access / shared memory**.
- You have to **create** the **new union** type **first**, and **then** you can **declare** union variables.
- Union **size differs** according to the **largest member size**.
- Using unions will **reduce memory** usage.

Creating new union type

- A creation of the new union data type must occur first, **written first in the .c** file, before any declaration occurs.
- **Creation** of a data type **doesn't mean** that we **allocate** anything from the **memory**.
- **Creation is just informing the compiler**.

```
union student
{
    unsigned char name[16];
    unsigned int id;
};
```

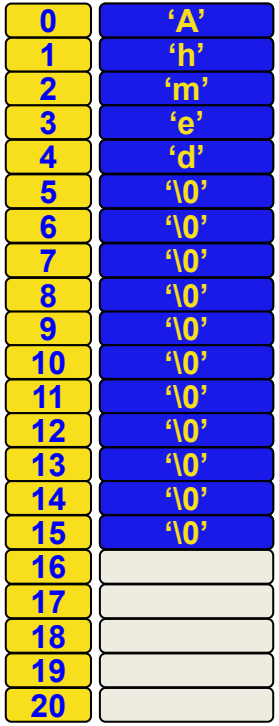
Declaring a union variable

- After creating a new union data type, you can declare a variable.
- Declaration Example:
  - `union student x;`
- Also you can set **only one initial value** for **either union** variable **members**.
- Definition Example:
  - `union student x = {"Ahmed"};`

Union variables in memory

- Defining a union variable:
  - `union student x = {"Ahmed"};`
- The size of this variable into the memory is **16 bytes**.
  - **16 bytes for the array of characters "name" because it is the largest member.**

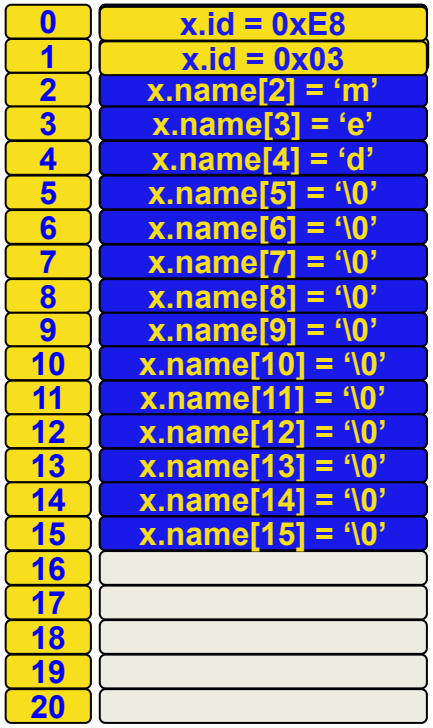
```
union student
{
      unsigned char name[16];
      unsigned int id;
};
```



Accessing a union variable members

- Use the **‘ . ’ operator** to access the union members.
- Access union members is for reading and writing.
- Examples:
  - `union student x = {“Ahmed”};`
  - `x.name[1] = ‘H’;`
  - `x.id = 1000;  // 0x03E8`

```
union student
{
      unsigned char name[16];
      unsigned int id;
};
```



Summary

- Now you are familiar with unions in C.
- You can create, declare and manipulate unions.
- You have learned that union size depends the largest member size
- Accessing union using the ‘ . ’ operator.
- Remember that only one initial value is given to a union member.
- You must control reading and writing to the union variables in order not to lose data.