**Union Assignment**

1. Refer the code below and comment on size of the given structure considering
   1. Structure as union
   2. Structure as struct
   3. arr
   4. uarr

\_\_\_ Job

{

char name[32];

unsigned short ucount;

float salary;

int workerNo;

char \*orgname;

};

\_\_\_ Job myvar; //could of union or of struct

Struct Job arr[10];

Union Job uarr[10];

A:a:

**a**: Size of union Job: The size will be 32 bytes (the largest of the members, which is char name[32]).

For the array of union (uarr[10]), the total size will be:

Size of uarr[10]: 10 \* 32 bytes = 320 bytes

b. The total size of struct Job on a 64-bit system would be:

Size of struct Job: 32 + 2 + 2 + 4 + 4 + 8 = 52 bytes.

For the array of struct (arr[10]), the total size will be:

Size of arr[10]:

On a 32-bit system: 10 \* 48 bytes = 480 bytes.

On a 64-bit system: 10 \* 52 bytes = 520 bytes.

C: Size of arr[10]:

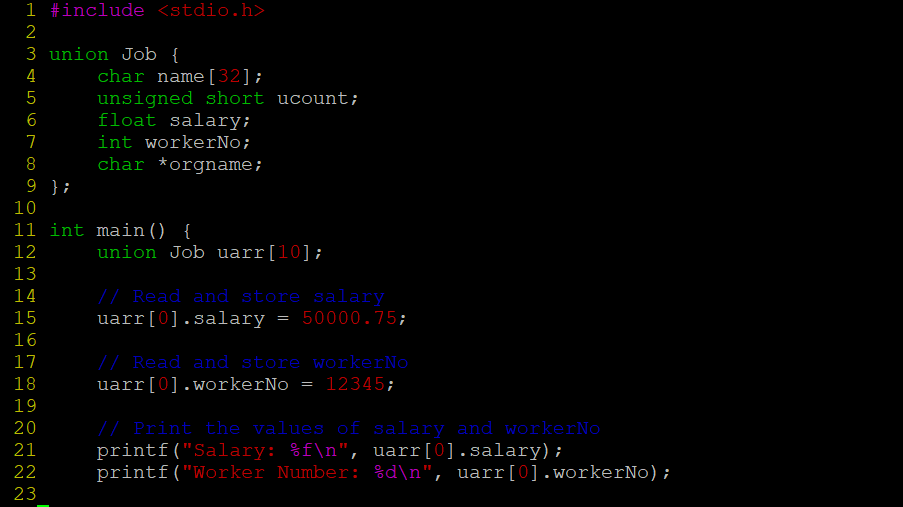
On a 32-bit system: 10 \* 48 bytes = 480 bytes.

On a 64-bit system: 10 \* 52 bytes = 520 bytes.

d: Size of uarr[10] (array of union): 320 bytes.

1. Refer Job datastructure in Q#1 above. Using uarr, perform below operations.
   1. Read and store salary
   2. Read and store workerNo

Comment on values of output if salary and workerNo are printed in order. Justify your statement.



OUTPUT:

A black background with numbers and symbols

Description automatically generated

1. Refer Job datastructure in Q#1 above. Assume that myvar is a structure variable. If I need to place 2 bytes (i.e 0x0102) as ucount using a char \*ptr then list all possible statements that can be used in \_\_\_\_\_.

[Let solutions include cases such as

* + 1. using base address of ucount
    2. using relative address of ucount w.r.t to base address of myvar]

int main()

{

char \*ptr = &myvar;

\_\_\_\_\_\_\_\_\_\_\_ = 0x01;

\_\_\_\_\_\_\_\_\_\_\_ = 0x02

}

A: i: Using the base address of ucount:

ptr[32] = 0x01;

ptr[33] = 0x02;

ii: ptr += 32; // Move ptr to ucount's address

ptr[0] = 0x01;

ptr[1] = 0x02;