Subject: unexpected output while reading the size of structure.
Posted by Shwetha on Sat, 10 Dec 2011 04:39:11 GMT
View Forum Message <> Reply to Message

Hii,

Please find the attached test case for the reference..in which I am trying to read the size of the following structure.

```
struct read_data
  {
   int i;
   char c;
   double d;

  };
 read_data r;
```

Output :

Size of int = 4
Size of char = 1
Size of double = 8

Size of r.i = 4
Size of r.c = 1
Size of r.d = 8

size of struct read_data = 16

Expected size of the struct read_data is 13 bytes.But giving the wrong output..Would you please let me know the issue behind this.

Thank You,

Shwetha S

File Attachments

1) SizeofDataTypes.zip, downloaded 406 times

Subject: Re: unexpected output while reading the size of structure.
Posted by Novo on Sat, 10 Dec 2011 04:58:45 GMT

Data structure alignment
Structure Alignment Examples

Subject: Re: unexpected output while reading the size of structure.
Posted by Shwetha on Sat, 10 Dec 2011 07:54:12 GMT

Hi,

Using struct PACKED we can read the size of the structure correctly.

```
#ifndef PACKED
#define PACKED  __attribute__((__packed__))
#endif

typedef struct  PACKED
  {
   int i;
   char c;
   double d;

  }read_data;

 read_data r;


};
```

Thanks & Regards,

Shwetha S

Subject: Re: unexpected output while reading the size of structure.
Posted by cbpporter on Mon, 12 Dec 2011 08:41:27 GMT

The size of 16 is correct. There are very good technical reasons for it to be so and Novo pointed out some links. Unless you have very good reasons to use an unaligned/packed struct, I would recommend against it.

One thing that you could try is reorder the elements. There is this old trick of ordering elements,

with the largest/more alignment constrained ones first. So put the double first, then the int, then the char.

If you are using binary storage an rely on fixed size (like 13) the solution is to read/write each field separately.
