

Re: How to read/write an 'address' from a memory location.....

## Re: How to read/write an 'address' from a memory location.....

---

*Source:* <http://coding.derkeiler.com/Archive/General/comp.arch.embedded/2007-10/msg00796.html>

---

- *From:* [aeroboro221@xxxxxxxxxxxxxxxx](mailto:aeroboro221@xxxxxxxxxxxxxxxx)
  - *Date:* Tue, 23 Oct 2007 05:09:04 -0700
- 

On Oct 23, 2:28 am, CBFalconer <cbfalco...@xxxxxxxx> wrote:

aeroboro...@xxxxxxxxxxxxxxxx wrote:

... snip ...

My problem is: I have two processors exchanging data over a shared section of memory. Now before processor 1 can write the data for processor 2 (in the shared memory), processor 1 needs to tell processor 2 where exactly it will be writing. That's where I was coming from – processor 1 basically needs to write an "address" into a predefined location for processor 2 and that "address" will hold the actual data for processor 2.

That is outside the abilities of the C language, which has no intrinsic knowledge of multiple processes and managing them. However, if your description is adequate, the following might work:

Global (common memory):  
int running = 1;  
Type \*ptr = NULL;

Sender code fragment:  
do {  
/\* prepare data \*/  
while (ptr && running) continue;  
if (running) {  
ptr = malloc(sizeof \*ptr);  
\*ptr = data;  
}  
} while (running)

Receiver code fragment:  
do {  
while (!ptr && running) continue;  
if (running) {

Re: How to read/write an 'address' from a memory location.....

```
rcvdata = *ptr;
free(*ptr);
ptr = NULL;
/* process rcvdata */
}
} while (running);
```

Note that either process can shut down the interaction by setting running to zero. Sender has a local variable named data, and receiver has one named rcvdata. The processes are in strict sync, i.e. one item of data causes one item to be transmitted, and nothing more is done until that item is received. Similarly the receiver can do nothing while waiting for new data to be prepared.

Note that malloc failing in sender is a fatal error. The initialization values ensure that the sender runs first. The definition of 'Type' controls the amount of data transferred. The 'continue' statements can be changed into function calls to do other things.

—

Chuck F (cbfalconer at mainline dot net)  
Available for consulting/temporary embedded and systems.  
<<http://cbfalconer.home.att.net>>

—

Posted via a free Usenet account from <http://www.teranews.com>

The algorithm of exchanging data once the addresses are known to both processors is pretty much done.

The way to communicate the address is where I was kind of stuck.  
I just tested one of the ways described here and it seems to work!

Thanks again!

Regards,  
Rintu

.