How i find the exact Address of Variable Buf

As reference, I'm using the following code:

```
#include <stdio.h>
#include <string.h>

int main (void) {
    char buf[100]; // ------> How do I find the address in gdb?

    printf ("Buffer is at memory location: %08x\n", &buf);
    strcpy (buf, "some random text");
    printf ("Text is [%s]\n", buf);

    return 0;
}
```

How can I get `gdb` to show me the address of the `buf` variable?

gdb buffer stack-overflow

edited Nov 1 '12 at 8:52
paxdiablo
401k 91 770 1224

asked Dec 16 '10 at 16:22
Neefra
37 1 1 3

2 Answers

`(gdb) p &a` if you need the address of variable `a`. A variable might be cached in a register though, in which case GDB would tell you `address requested for identifier "a" which is in register $xxx`.

Sidenote: do not use `gets`, see here.

edited Dec 16 '10 at 16:44

answered Dec 16 '10 at 16:29
Nikolai N Fetissov
55.5k 5 47 103

2 I don't thing `buf` will be cached in a register. Well, not unless it's a bloody *big* register :-) – paxdiablo Dec 16 '10 at 16:32

2 You mean your processor does not have 100-byte registers? Dude, what hardware are you running on? – Nikolai N Fetissov Dec 16 '10 at 16:36

Free BSD Inter Pentium 4...... – Neefra Dec 18 '10 at 13:24

If you enter the following into gdb, you'll get the address:

```
start
p &buf
```

as in the following transcript:

```
pax$ gdb ./qq.exe
GNU gdb 6.8.0.20080328-cvs (cygwin-special)
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-pc-cygwin"...
(gdb) start
Breakpoint 1 at 0x401144: file qq.c, line 2.
Starting program: /home/pax/qq.exe
[New thread 2912.0xf9c]
[New thread 2912.0x518]
main () at qq.c:2
```

```
2       int main (int argc, char **argv) {
(gdb) p &buf
$1 = (char (*)[100]) 0x22ccd0
(gdb)
```

answered Dec 16 '10 at 16:29

paxdiablo
401k 91 770 1224

Why two threads? – Nikolai N Fetissov Dec 16 '10 at 16:42

1 It's CygWin. Only $DEITY knows what's going on under the covers to emulate UNIX :-) – paxdiablo Dec 16 '10 at 16:54