TAGS
hello world, linux 3.0, system call, Ubuntu

Adding hello world system call to Linux

October 5, 2012
I just finished doing this and so thought I'd write about it in my blog since it took me a while to get this done after many an experimentation I suppose . So here goes!

I did this on an Ubuntu 12.04.1 LTS OS running on an 64bit machine but this post should mostly work on any Linux OS(with a few differences here and there).

Step 1: Get the source

The first step is to download the source code of the Linux kernel. I used the one available from the repositories but feel free to get the sources from **kernel.org (http://kernel.org)**.

```
1 | apt-get source linux-image-$(uname -r)
```

This would download all the archives and unpack it into a directory linux-3.2.0.

Step 2: Add system call to system call table

Open the file arch/x86/kernel/syscall_table_32.S and add the following line.

```
1 | .long sys_hello
```

Step 3: Define macros associated with system call

Open the file arch/x86/include/asm/unistd_32.h. You will notice that a macro is defined for each system call. At the end of the huge macro definition, add a definition for our new system call. I added the following line:

```
1 | #define __NR_hello 349
```

and accordingly incremented the value of the macro NR_SYSCALLS:

```
1 | #define NR_syscalls 350
```

Also, add the macro definition to the file arch/x86/include/asm/unistd_64.h

```
1 | #define __NR_hello 312
2 |  __SYSCALL(__NR_hello, sys_hello)
```

Now to the file include/linux/syscalls.h, add the prototype of the system call.

```
1 | asmlinkage long sys_hello(void);
```

Now, in the root directory of the kernel sources, create a directory named hello and in it, a file hello.c with the following content:

```
#include <linux/kernel.h>

asmlinkage long sys_hello(void)
{
    printk("Hello world\n");
    return 0;
}
```

printk is similar to printf function of C but writes to the kernel log instead of the screen. asmlinkage is a key word used to indicate that all parameters of the function(here none of course) would be available on the stack.

After creating the function definition, create a file named Makefile within the hello directory and the following content to the file:

```
obj-y := hello.o
```

This is to ensure that our hello.c file is compiled and included in the kernel.

Now, to the Makefile in the root directory of the kernel sources, edit the following line:

```
core-y          += kernel/ mm/ fs/ ipc/ security/ crypto/ bloc
```

to:

```
y           += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ hello/
```

This is to tell the compiler that the source files of our new system call are in present in the hello directory.

That's it-you have added your own system call! Now all you need to compile the kernel. [1] is a good place to get information on how to compile the kernel. You needn't do all the steps since you have the sources-read it carefully :)!

After you compile and reboot into the kernel you just compiled, try running the following program.

```
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>

#define __NR_hello 312 //349 if you are running a 32bit kerne

long hello_syscall(void)
{
    return syscall(__NR_hello);
}

int main(int argc, char *argv[])
{
    long int a = hello_syscall();

```

```
17          printf("System call returned %ld\n", a);
18          return 0;
    }
```

The output of the program would be:

```
1  System call returned 0
```

The printk's output get written to the kernel log. To view it, run the command

```
1  dmesg
```

and sure enough you will see the Hello World on the very last line.

Congratulations you have just added a system call to Linux!

Helpful resources

[1] **http://www.howopensource.com/2011/08/how-to-compile-and-install-linux-kernel-3-0-in-ubuntu-11-04-10-10-and-10-04/ (http://www.howopensource.com/2011/08/how-to-compile-and-install-linux-kernel-3-0-in-ubuntu-11-04-10-10-and-10-04/)**

[2] **http://bluegrit.cs.umbc.edu/~lsebald1/cmsc421/new-syscall.php (http://bluegrit.cs.umbc.edu/~lsebald1/cmsc421/new-syscall.php)**

[3] **https://wiki.ubuntu.com/Kernel/BuildYourOwnKernel (https://wiki.ubuntu.com/Kernel/BuildYourOwnKernel)** (easier method)

[4] **https://help.ubuntu.com/community/Kernel/Compile (https://help.ubuntu.com/community/Kernel/Compile)**

From → Operating Systems, Ubuntu

22 Comments

1. **Rob permalink**
   Thanks you so much, this really helped me.
   I actually already had the system call correctly created and the kernel compiled, but I was having trouble figuring out what syntax to use when making the system call and what to #include.
   Thank you sir!

   Reply
   - **Arvind permalink**
     Hello Rob,
     Thanks for pointing out! I have updated the C program that calls the system call. Missed out a header.

     Reply
2. **aman gupta permalink**
   http://simplyeazy.com/how-to-add-a-system-call-to-linux-kernel/

   try this link its working perfectly for fedora…

Reply

3. **Jason permalink**
   I am doing a assignment similar to the instructions on the site. How did you compile the kernel? I am having problems with the compilation and then how did you run the last program where you check if the system call worked.

   Reply
   - **Arvind permalink**
     Resource [1] contains the instructions I followed to compile the Linux kernel. To test the system call, I compiled the last program using gcc and ran the executable created. Just be sure that you specify the correct system call number and it should work.

     Reply
4. **Jason permalink**
   I follow every single line and the resource for the compilation. I made a c program with the program you mention to see if the compilation worked and the output returned a -1, not a 0. I'm not sure what went wrong.

   Reply
   - **Arvind permalink**
     Did you specify the correct system call number for the kernel you compiled? I can't think of any other issues. Maybe you could also check if dmesg has any useful error messages.

     Reply
5. **jason permalink**
   I am currently using 12.04.1 (32-bit). So the system call number is 349. When I did enter the "dmesg" command, I got

   "50.977596 init: plymouth-stop prestart process (1123) terminiated with status 1
   137.669229 hrtimer: interrupt took 6521308 ns"

   Those are the last 2 lines I saw. Is this helpful? I appreciate your help. I am a novice at this type of work.

   Reply
6. **Arvind permalink**
   Hmmm I'm not sure if that's of any use. Maybe you should check out the other messages from dmesg. I'm afraid I do not have much experience in this so I can't really help out a real lot .

   Reply
7. **Aqib Rashid permalink**
   hello arvid bro, i follow all the steps for adding system call. now i want to compile the kernel but i didnt get the steps mentioned in link [1]. can u please help me in your way that what should i do now because they way you explained here is incredible and easy for beginers. thanks

   Reply
   - **Arvind permalink**
     Hey Aqib!
     Sorry for late response! I think [1] is a pretty good link. Just take your time and read it – it's pretty straightforward .

Reply

8. **mubashir permalink**
   i have recieved message system call return -1

   Reply
   - **Arvind permalink**
     Hey Mubashir!
     Sorry for late response! Could you check what error messages are displayed by dmesg?

     Reply
9. **marzi permalink**
   hi
   thanks for useful article
   i do as you say above , but when i try to compile my new kernel i face an fatal error
   like :
   hello\hello.c:1:25: fatal error: linux/kernel: no such file or directory compilation terminated.
   make[1]:***[hello/hello.o] Error 1
   make : *** [hello] Error 2

   Reply
   - **Arvind permalink**
     Hey Marzi!
     Sorry for the late response! I think you've included the wrong file – it's linux/kernel.h. Could you check that again? If that is the correct line, maybe you've not installed the packages required for building the kernel.

     Reply
10. **gsaro permalink**
    I followed instructions and after compiling, my system call is not recognised. error-undefined reference to mycall. when i checked my /include/i386-linux-gnu/unistd_32.h i found that 1

    #define __NR_mycall 349 is not defined in spite of adding it before compiling

    Reply
    - **Arvind permalink**
      Hey gsaro!
      I think you've not defined the function mycall properly. That macro adds the system call to the system call table but doesn't define the function that is executed when the system call is invoked.

      Reply
11. **Shabbir permalink**
    Hello ! i've checked everything thoroughly, but still i get "System call returned -1″ , what should i do now ?

    Reply
    - **Arvind permalink**
      Did you try and determine why the system call returned -1?

      Reply
      - **Shabbir permalink**

Well, i’ve resolved my issue, it was just a minor mistake, i was trying to test the system call without installing the new kernel. Sorry to bother you….

2. **chaudhary permalink**
   i have recieved message system call return -1
   quickly responsed please

   Reply
   - **Arvind permalink**
     Did you try and determine why the system call returned -1?

     Reply