Adding a System Call to UML

Alexandros Karypidis

This brief guide attempts to provide an introduction to creating new system calls for UML. Please send bug reports to karypid@inf.uth.gr.

1 Define the entry point

The first thing you need to do is modify the file `unistd.h` in `include/asm/arch`. In this file, you need to add a line providing an id for your system call. Locate the bunch of lines of the form:

```
#define __NR_somename NNN
```

You need to add a new line, replacing "somename" with your system call's name. Naturally, you must assign a unique number to your system call. Check what the largest value used for an id is and assign that value plus one to your system call.

Next, you must add and entry refering to your call in the system calls table. To do this, modify `sys_call_table.c` file in `arch/um/kernel` and add a line:

```
[ __NR_somename ] = sys_somename,
```

In the same file, add a declaration for your system call in the area where all the other system calls are defined, as shown on the following line:

```
extern syscall_handler_t sys_somename;
```

Finally, you must change the value of "LAST_GENERIC_SYSCALL" so that your new system call's id is considered to be within the allowable range.

```
#define LAST_GENERIC_SYSCALL __NR_somename
```

Having done all these, an attempt to compile the kernel should fail during linking, as the "sys_somename" function must now be implemented.

2 Implementation code

The first step is to create a header file `somename.h` for your system call and place it in `arch/um/include` as shown in listing 1.

Listing 1: somename.h

```
/* somename.h */
/*                    Listing 1: somename.h                    */
#ifndef __UM_SOMENAME_H
#define __UM_SOMENAME_H
extern void sys_somename();
#endif
```

Then, write out the implementation `somename.c` of your system call in `arch/um/kernel` as shown in listing 2.

Listing 2: somename.c

```
/* somename.c */
/*                    Listing 2: somename.c                    */
#include ”linux/kernel.h”
#include ”linux/unistd.h”
#include ”somename.h”
void sys_somename () {
        printk (”Hello, from then new system call !\n”);
        return;
}
```

Finally, modify the respective Makefile in `arch/um/kernel` and add somename.o to the list of build targets.

3 Creating a stub for your system call

Listing 3 shows a program which uses the `_syscall` macro to create a stub for the system call. It then proceeds to call the stub function. When compiling, be sure to specify the -I option so that gcc will look at the modified version of `unistd.h`. In the example, the preprocessor looks for the file in `asm/arch/unistd.h`, so if the UML code is in `/uml-code` you should compile with `-I/uml-code/include`.

Listing 3: testprog.c

```
/* testprog.c */
/*                    Listing 3: testprog.c                    */
#include <stdio.h>
#include <errno.h>
#include ”asm/arch/unistd.h”
_syscall0 (void,somename)
int main (int argc, char *argv []) {
        printf (” calling ...\ n”);
        somename();
}
```