

# Implementación de un System Call en Minix 3

*Felipe Caballero*

*Universidad Adolfo Ibáñez, Santiago de Chile  
Sistemas Operativos 2006-Segundo Semestre*

## Introducción

Los System Calls (SC) son “una invocación de una rutina del Sistema Operativo (OS). Los OS tienen varias rutinas (los SC) para realizar varias operaciones de bajo nivel”<sup>1</sup>. La implementación de SC no es algo muy complejo de realizar pero requiere conocer con cierto grado de profundidad como funciona el OS.

## Implementación de un System Call

En Minix 3, el OS está basado en servidores, existen dos servidores importantes el FS y el PM, entre otros. El FS (File System) se encarga del manejo de archivos (creación, borrado, etc.) y el PM (Process Manager) se encarga de todo lo referente a los procesos del sistema.

Se desea implementar un System Call (SC) en Minix 3 que retorne el PID(Process ID, o identificador del proceso) y el PID del padre del proceso (PPID) que utiliza el SC. Para esto, se debe crear una función en el sistema que lleve a cabo la tarea deseada (adentro del archivo `getset.c`), además hay que declarar el prototipo de esta. Luego, para que esta función pueda ser utilizada por programas creados por el usuario, tiene que existir algún tipo de librería que este pueda utilizar, por lo que hay que crear una (`getpids.h`). Además hay que llevar a cabo varios ajustes en el OS (todos los arreglos e implementaciones de funciones se pueden ver detalladamente en el archivo `log.rtf`).

Una vez implementado el SC, tiene que crearse una forma de verificar el funcionamiento de este. Para esto, se crea el programa `sc_test.c` que realiza una comparación entre las funciones nativas del sistema que obtienen el PID y el PPID utilizando la función `assert(comparacion a evaluar);`<sup>2</sup>

---

1 Información sacada de <http://www.angelfire.com/anime3/internet/opersys.htm>

2 La función `sc_test.c` fue obtenida de <http://matteo.vaccari.name/so/minix>

# Bitácora

Se desea implementar un System Call (SC) en Minix 3 que retorne el PID(Process ID, o identificador del proceso) y el PID del padre del proceso que utiliza el SC.

Para esto, se llevan a cabo los siguiente pasos:

1-Editar el archivo `/usr/src/include/minix/callnr.h`:

- incrementar del numero máximo de SC (de 95 a 96)
- crear la linea `"#define GETALLPID 95"` al final del archivo

2-Modificar el archivo `/usr/src/servers/pm/proto.h`:

- Debajo de `"/* getset.c */"` se crea un prototipo del SC que se desea crear:  
`"_PROTOTYPE( int do_getpids, (void) );"`

3-En el archivo `/usr/src/servers/pm/table.c`:

- insertar la linea `"do_getpids, /* 95 = get parent pid and own pid */"` (tiene que estar en la posición adecuada, en este caso la posición 95, igual a lo especificado en el paso 1)

4-En el archivo `/usr/src/servers/fs/table.c`

- insertar la linea `"no_sys, /*95 = get parent pid and own pid*/"` (tiene que estar en la posición adecuada, en este caso la posición 95, igual a lo especificado en el paso 1)

5-Editar el archivo `/usr/src/servers/pm/getset.c`:

- insertar la siguiente funcion al final:

```
/*=====*
 *                               do_getpids                               *
 *=====*/
PUBLIC int do_getallpid()
{
/* This method gets the process' pid and its parents'.
*/
    register struct mproc *rmp = mp;
    /* mp es un puntero al proceso actual */
    int proc;

    if(pm_isokendpt(m_in.endpt, &proc) == OK && proc >= 0)
    {
        rmp->mp_reply.reply_res3 = mproc[who_p].mp_pid;
        /* mproc[who_p] representa el proceso actual */

        rmp->mp_reply.reply_res2 = mproc[rmp->mp_parent].mp_pid;
        /* mproc[rmp->mp_parent] representa el proceso padre */

        return 0;
        /* Si todo funciona adecuadamente el retorno es 0 */
    }
    else
    return 1;
    /* Si hubo algun error, el retorno es 1 */
}
}
```

6-Crear el archivo `/usr/src/include/getpids.h` con lo siguiente:

```

#include <lib.h>

PUBLIC int getpids(pid_t *current, pid_t *parent)
{
    message m;

    *current = 999;
    /* Se le asigna al puntero del pid del proceso actual un valor arbitrario inicialmente */
    *parent = 999;
    /* Se le asigna al puntero del pid del proceso padre un valor arbitrario inicialmente */

    _syscall(MM, GETALLPID, &m);
    /* Se ejecuta el system call */

    *current = m.m2_i2;
    /* Se le asigna el pid del proceso actual al puntero del pid del proceso actual */

    *parent = m.m2_i1;
    /* Se le asigna el pid del proceso padre al puntero del pid del proceso padre */

    return 0;
}

```

### 7-Crear las librerías necesarias (para incluir a getpids.h en /usr/include/)

```

cd /usr/src/tools
make includes

```

### 8-Recompilar el kernel

```

cd /usr/src/tools/
make hdboot /* Esta función hace que la imagen compilada se inicie por defecto */
/* En el menu inicial de Minix3 hay que seleccionar la opción 3 (custom) */

```

### 10-Crear el archivo sc\_test.c:

```

#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <getpids.h>

#include <assert.h>
#include <stdio.h>

#define N 4

void assert_getpids_works(void) {
    pid_t expected_me = getpid();
    pid_t expected_parent = getppid();
    pid_t actual_me, actual_parent;

    int result = getpids(&actual_me, &actual_parent);
    assert(0 == result);
    assert(actual_me == expected_me);
    assert(actual_parent == expected_parent);
}

main(void) {
    int i;

    assert_getpids_works();

    for (i=0; i<N; i++) {
        if (0 == fork()) {
            assert_getpids_works();
            return 0;
        }
    }
}

```

```
    } else {  
        int status;  
        wait(&status);  
    }  
}  
printf("OK\n");  
return 0;  
}
```

## Conclusion

La creación de System Calls en Minix 3 es una tarea fácil de llevar a cabo pero que requiere un amplio conocimiento de la estructura del Sistema Operativo.

## Bibliografía

Istruzioni per Minix - <http://matteo.vaccari.name/so/minix>

Adding System Calls in Minix - <http://wwwcsif.cs.ucdavis.edu/~engle/ta/ecs150-f03/syscall.html>

Agregar un System Call a Minix - <http://www.midnightsoret.com.ar/personales/alejandrovaldez/minix/minixSystemCall.html>

Grupo de Minix en Google - <http://groups.google.com/group/comp.os.minix?lnk=li>