

Práctica C

Creación de una nueva llamada al sistema.

- **Objetivo:** se trata de implementar nuestra propia llamada al sistema, llamémosla **ASOPS**. Para conseguirlo, desde el espacio de usuario se enviará un mensaje al servidor de memoria y desde éste se reenviará a la tarea de sistema. Una vez en la tarea del sistema se podrá implementar la llamada al sistema y enviar la respuesta. Es decir, el camino que se seguirá es el siguiente:

USUARIO (nivel 4) → MM (nivel 3) → SYSTEM (nivel 2-1) → MM (nivel 3) → USUARIO (nivel 4)

- Recuerde que el mensaje que el usuario envía al gestor de memoria (MM) es físicamente gestionado mediante una llamada al sistema (SYSVEC) implementada mediante la función `/usr/src/kernel/proc.c:sys_call()`
- Nosotros pensaremos en ese mensaje como si fuera directamente enviado desde el proceso de usuario al servidor MM
- **PRIMERA FASE: MODIFICACIONES EN EL NÚCLEO.**
 - `/usr/include/minix/callnr.h`: lista de las llamadas al sistema. Añadir la 77: **ASOPS**. No olvide incrementar el número total de llamadas al sistema (NCALLS).
 - **Gestor de memoria:**
 - La función `main()` de `main.c` contiene el bucle infinito de este servidor. Observe la línea
`error = (*call_vec[mm_call])()`;
 - Invoca a una función cuya dirección está en la entrada `mm_call` del vector de punteros a funciones `call_vec[]`
 - La función `call_vec[]` se inicializa en el fichero `table.c`
 - `/usr/src/mm/table.c`: añadir la rutina de servicio de interrupción. Por ejemplo: `do_asops`. Para ello añádala en `table.c` (en el sitio que la corresponda, **muy importante**) y en `/usr/src/mm/proto.h` (sin argumentos, de forma análoga a la declaración de `do_reboot()`)
 - Escriba la nueva función **`do_asops()`**. Puede hacerlo en alguno de los ficheros existentes. Por ejemplo en `/usr/src/mm/utility.c`. Esta función debe enviar el mensaje a la **tarea del sistema**, situada en el nivel 2. Esta tarea se identifica mediante la constante **`SYSTASK`**, ver `/usr/include/minix/com.h` El mensaje que acaba de recibir se encuentra en la variable externa **`mm_in`**, y para reenviarlo hacia la tarea del sistema basta con invocar a la función **`_taskcall()`**. Una vez que se ha recibido la respuesta hay que copiarla al mensaje **`mm_out`**. A continuación se puede ver un ejemplo:

```
PUBLIC int do_asops() {
```

```
int a1, a2, a3;
```

```

a1 = mm_in.m1_i1;

a2 = mm_in.m1_i2;

a3 = mm_in.m1_i3;

printf("MM:do_asops: %d %d %d\n", a1, a2, a3);

_taskcall(SYSTASK, ASOPS, &mm_in); /* Reenvio del mensaje a la
tarea del sistema */

result2 = mm_in.m1_i1; /* result2 es una variable externa que se
copiara en mm_out.m1_i1 */

mm_out.m1_i2 = mm_in.m1_i2;

mm_out.m1_i3 = mm_in.m1_i3;

}

```

- **Tarea de sistema.** En el fichero `/usr/src/kernel/system.c` realizar las siguientes modificaciones:
 - Añadir el prototipo de la función que dará servicio a nuestra llamada al sistema, llamémosla **do_asops()**
 - En la rutina principal, **sys_task()**, añadir nuestra llamada al sistema dentro del switch. Use la constante **ASOPS** añadida en `callnr.h`
 - Escribir la función **do_asops()**. Puede simplemente imprimir los argumentos y después cambiarlos para comprobar que se reciben correctamente en el espacio de usuario.
- **Compile el kernel.** Asegúrese de que ha compilado bien (vea el documento de instalación).

• SEGUNDA FASE: EL PROCESO DE USUARIO

- En un directorio de trabajo aparte cree el fichero **llamasis.c** :
 - Incluya `lib.h`, `sys/types.h` y `unistd.h`
 - Declare una variable de tipo **message** (`/usr/include/minix/type.h`) e inicialice alguno de sus campos
 - Utilice la función de biblioteca **taskcall()** para realizar la llamada al sistema **ASOPS** (enviar el mensaje al gestor de memoria): `_taskcall(MM, ASOPS, &msj)`
 - Imprima los campos del mensaje para comprobar que han sido cambiados desde el núcleo.
 - Compile `llamasis.c`. Reinicie el sistema operativo de forma que se cargue el nuevo núcleo. Ejecute el `a.out`.

• TERCERA FASE: REPASAR TODO LO ANTERIOR Y AÑADIR SU PROPIA LLAMADA AL SISTEMA

- Una vez que le haya funcionado, vuelva a repasar punto por punto todo lo que ha hecho en los pasos anteriores. Documente todas las modificaciones . Asegúrese de entender el por qué de cada línea.
- Añada su propia función que, por ejemplo, proporcione información detallada del proceso actualmente en ejecución (proc_ptr), es decir, del propio invocador. Para eso utilice el primero de los tres argumentos que ha pasado en las dos primeras fases. Dicho argumento deberá seleccionar el tipo de función a realizar por nuestra llamada al sistema ASOPS. Utilice constantes alfanuméricas para identificar dichas funciones.