Semáforos en Minix

M. Besio P. García H.Rajchert

22 de Noviembre de 2006

Resumen

En este trabajo expondremos la información y los resultados obtenidos al realizar una modificación al sistema operativo Minix.

1. Enunciado: Modificaciones a Minix

1.1. Objetivo

El proyecto a realizar consiste en agregar al Sistema Operativo Minix 2.0.0, una característica o herramienta, que dispongan los Sistemas Operativos más elaborados y que Minix no tenga. El proyecto está centrado en la investigación del funcionamiento de Minix, manejo de pasaje de mensajes, servidores, task (drivers), manejo de system calls e interrupciones. La primera parte consiste en el estudio de las posibles implementaciones a realizar, para luego comenzar con las modificaciones necesarias.

1.2. Temas propuestos

- Archivos ocultos: Se deben poder ocultar archivos, según el dueño, grupos o otros, una estructura similar a lectura-escritura-ejecución.
- Cola de mensajes: Debe cumplir con el estándar de POSIX.
- Semáforos: Debe cumplir con el estándar de POSIX.
- Mount de FAT 12 (read): Se debe poder hacer un mount de un diskete con FAT 12 y ver su contenido, pero no modificarlo.

Se aceptará incluir otros temas además de los aquí presentados. La propuesta de los alumnos debe ser aprobada por los integrantes de la cátedra en función de la complejidad del tema elegido y de la cantidad de alumnos que integren el grupo.

1.3. Contenido

1.3.1. Script de instalación y desinstalación

El grupo debe desarrollar un script que realice lo siguiente:
En la instalación:

- Backup de archivos fuentes que se van a modificar.
- Copiar archivos fuentes modificados.
- Compilar el kernel.

En la desinstalación:

- Restaurar los arhivos fuentes resguardados.
- Compilar el kernel.

El objetivo del script es no modificar la instalación de Minix existente.

1.3.2. Programas de prueba

Se deberán desarrollar y documentar debidamente los programas de testeo utilizados por el grupo, explicando en cada uno su función.

1.3.3. Páginas de manuales

Cada grupo deberá incluir en el sistema los manuales (programa man) de las nuevas funcionalidades que hayan implementado.

1.3.4. Informe

El informe debe contener las siguientes secciones como mínimo:

- Carátula.
- Enunciado del TP.
- Tema elegido.
- Listado de los fuentes de Minix modificados y los fuentes nuevos, comentando brevemente en cada caso la modificación realizada y el objetivo de la misma. (En el informe impreso, en tipo de letra "Bold" o "Negrita").
- Manual del usuario. Comandos. Forma de compilar.
- Programas de test. Explicación de su funcionamiento.
- Funcionamiento interno de Minix analizado.

Todos estos ítem serán evaluados.

1.4. Consideraciones

- Es posible utilizar en el trabajo rutinas o librerías ya desarrolladas consultando previamente al docente a cargo la inclusión de las mismas. No está permitido incluir rutinas o librerías desarrolladas por integrantes de otros grupos.
- El tipo de diseño y la forma de implementación serán discutidos entre el grupo y la cátedra durante las clases de laboratorio, dejando la posibilidad de modificar éste enunciado escrito, previo acuerdo entre el docente y los integrantes del grupo.
- Para la evaluación se tendrá en cuenta no sólo el funcionamiento del programa sino también el informe y la investigación realizada sobre el funcionamiento interno del Sistema operativo Minix.

1.5. Material a entregar

Cada grupo deberá entregar:

- 1 Diskette con los archivos fuentes y el informe del trabajo realizado (FAT 12).
- Informe impreso.

1.6. Integrantes del grupo

El trabajo debe ser realizado en grupos de 3 integrantes o menos. La nota definitiva se compone de un coloquio oral y demostración del trabajo funcionando en el que deben estar presentes, sin excepciones, todos los integrantes del grupo para su defensa el día de la entrega.

1.7. Fecha de entrega y defensa del trabajo final

Miércoles 22 de Noviembre a las 18 hs en el Laboratorio LI1.

2. Tema elegido

El tema que elegimos es semáforos, dado que Minix no tiene un soporte nativo de los mismos y proporcionan ventajas significativas cuando se trata con problemas de concurrencia. Lo implementamos bajo el estándar POSIX.

3. Cambios en los fuentes de Minix

Se agregó una biblioteca con las funciones que indica el estándar de POSIX para implementar semáforos. Puede obtener una lista de las mismas en la próxima sección. La misma fue incorporada a Minix como una biblioteca estándar. Se modificó el servidor de FS para agregarle un system call que invoca a estas funciones. Para lograrlo se agregó el número de la nueva system call en `callnr.h` y en `table.c` de `fs` se la implementó, mientras que en el de `mm` se colocó la contrapartida de la system call indicando que no se use.

Los archivos de Minix modificados o agregados son los siguientes:

- `fs/Makefile` - Cambiado para compilar.
- `fs/misc.c` - Agregada una funcion que captura el system call.
- `fs/proto.h` - Agregados los prototipos de la funcion de misc.c.
- `fs/main.c` - Agregado el init para malloc y la inicializacion de las funciones del semaforo.
- `fs/table.c` - Agregado el system call.
- `mm/table.c` - Agregado el `no_sys`.
- `fs/my_param.h` - Archivo con macros para envolver mensajes.
- `fs/semaphore.c` - Implementacion de la biblioteca.

- `fs/semaphore2.c` - Contenido de la parte del semaforo que interactua con Minix.
- `/usr/include/minix/callnr.h` - Agregado el numero de system call.
- `/usr/include/semaphore.h` - Prototipos del semaforo.

4. Manual del usuario

Para instalar esta modificación en Minix Ud. deberá ejecutar el script `tpminix.sh` seguido de las palabras clave `install` o `uninstall`. Con esto se modificará el sistema agregando la funcionalidad del trabajo.

Una vez instalada la modificación, usted podrá acceder al manual online de la biblioteca y cada una de las funciones implementadas por medio del comando `man`. Las funciones corresponden al estándar POSIX y son las siguientes:

```
sem_close
sem_destroy
sem_getvalue
sem_init
sem_open
sem_post
sem_trywait
sem_unlink
sem_wait
```

5. Programas de prueba

El problema que resuelve la aplicación `prod_cons` es el clásico de consumidor-productor. Para ello se requieren dos o más procesos, al menos uno productor y otro consumidor. Los primeros producen mensajes y los segundos los consumen.

Dado que la implementación de la system call fue a destiempo, las funciones de la biblioteca en sí no podían ser probadas durante su producción misma (en Linux). Por lo tanto, se ha desarrollado una aplicación en Linux entera cliente-servidor TCP/IP que emula la IPC de Minix con pasaje de mensajes por sockets.

Para compilarla se debe acceder a `src/tests/linux` y ejecutar el script `compilar_ejemplos.sh`. Una vez completada la compilación se podrá iniciar el productor con `prod_cons p` y el consumidor con `prod_cons c`. Se pueden lanzar varios consumidores y varios productores a la vez (debe lanzarse desde varias terminales dado que el programa se bloquea esperando teclas). Una vez dentro, por cada caracter que se ingrese (incluido ENTER) se le estará indicando que consuma (o produzca, según sea el caso) un elemento. Por ejemplo, si se ingresa `aaa` en el consumidor, se consumirán (o se intentará consumir) 4

elementos (3 letras + ENTER). De esta manera podrá apreciarse cómo es la dinámica de los semáforos detrás de esta lógica, y probar su funcionamiento en Linux.

En Minix, el mismo programa se ejecuta nativamente y se encuentra en `/usr/ast/sem/`.

También se incluye otro programa (`prod_cons_init`) que funciona de la misma manera que el anterior, sólo que no realiza `sem_open` y `sem_close` sino que realiza `sem_init` y `sem_destroy`.

6. Funcionamiento interno de Minix

Minix tiene una estructura interna dividida en cuatro niveles. En su base se encuentra el administrador de procesos, seguido de las tareas (drivers), los servidores y finalmente los procesos de usuario. La capa de administración de procesos se encarga de atrapar todas las interrupciones, alternar entre tareas y proveer al sistema del pasaje de mensajes entre procesos como forma de intercomunicación. La siguiente capa contiene los procesos que manejan la comunicación con los dispositivos de distintos tipos. Dentro de ellos se encuentra el System Task, que si bien no interactúa con ningún dispositivo, existe para proveer servicios como copiar partes entre diferentes regiones de memoria. La capa 3, la de servicios, permite al nivel inferior de la cadena tener la asistencia necesaria para su correcto funcionamiento. Por ejemplo, aquí se ubica el MM (Memory Manager) que proporciona llamadas al sistema tan importantes como fork, exec, etc. En la última capa se ubican los procesos de usuario, como el shell, editores de texto, compiladores y programas escritos por el usuario.

Particularmente, en nuestro interés cayó el tema de IPCs en Minix. Este cuenta con tres primitivas para enviar y recibir mensajes. Ellas son:

```
\item send(dest, &message)
\item receive(source, &message)
\item send_rec(src_dst, &message)
```

Cuando un mensaje es enviado a un proceso que no está esperando por uno, el proceso que lo envía se bloquea. Fue con la función `send_rec` que hemos implementado el bloqueo de los procesos, en combinación con la variable global `dont_reply` que selecciona a qué procesos responder (al no responder, se bloquean).

Con respecto a nuestras pruebas, el desarrollo de la aplicación cliente-servidor no sólo resultó provechoso para detectar bugs de antemano, sino que sirvió para dejar muy en claro la arquitectura interna de microkernel de Minix. El mismo Tanenbaum aclara que la estructura de Minix puede pensarse como una aplicación de este tipo, dado que el pasaje de mensajes es análogo a una transacción en TCP/IP.

Cabe destacar que la biblioteca implementada no soporta la reentrancia. Si esta misma biblioteca se hubiera implementado para Linux debería haberse creado con soporte para la reentrancia, dado que allí

se pueden producir interrupciones a la ejecución de la misma. En cambio, Minix entre niveles distintos no presenta problema dado que los procesos se comunican mediante colas de mensajes, y hasta que el nivel superior no termine de atender el pedido, no se pasará a otro.

Referencias

[1] "Sistemas Operativos: Diseño e implementación, Segunda Edición", Tanenbaum & Woodhull, Ed. Pearson, 1998, New Jersey.

[2] http://www.opengroup.org/onlinepubs/000095399/ basedefs/semaphore.h.html

[3] http://www.midnightsoret.com.ar/personales/ alejandrovaldez/minix/minixSystemCall.html

Índice

Made with LaTeX