Arquitectura de lkmonitor

Arquitectura de lkmonitor

1.- Introducción

Lkmonitor es una aplicación para el escritorio de Gnome. Esto significa que lkmonitor utiliza las funciones proporcionadas por las bibliotecas Glib y Gtk para dar apoyo a la mayoría de sus funciones.

Lkmonitor muestra información acerca de distintas características del sistema, como cpu o uso de memoria. Esta información es recogida del pseudo-sistema de ficheros */proc*. La actividad principal de lkmonitor es leer información de estos ficheros, parsearla y extraer de ellos la información relevante. Desde este punto de vista, lkmonitor actúa como un simple lector de ficheros (Ilustración 1).

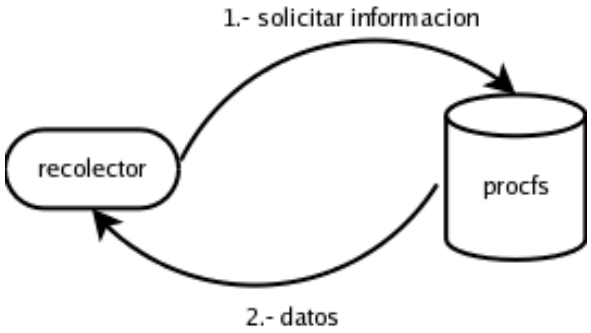

Ilustración 1 Ciclo normal de lkmonitor

Aunque este documento no pretende entrar en el funcionamiento interno del *procfs*, se explicarán algunos conceptos.

2.- Acerca del *procfs*

Procfs es un pseudo-sistema de ficheros, lo que significa que los ficheros bajo */proc*, no existen realmente en el disco duro, sino que la información que continen es calculada bajo demanda. Como el resto de sistemas de ficheros en Linux, el *procfs* está soportado por el Sistema de Ficheros Virtual o VFS. El VFS es una capa del kernel que proporciona una abstracción cuando trabajamos con sistemas de ficheros, de forma que gestiona las diferencias entre dichos sistemas, mientras ofrece un interfaz de acceso uniforme.

Aunque otros sistemas tipo UNIX disponen también de un sistema */proc* (FreeBSD, por ejemplo), el formato es diferente entre ellos. Mientras que Linux ofrece un formato basado en texto para la mayor parte de sus ficheros, FreeBSD y otros sistemas ofrecen poca información en este formato y más en formato binario.

Mientras que la primera aproximación podría ser más conveniente a la hora de trabajar con *shell scripts*, la segunda es quizá más cercana a la programación.

Bajo */proc* podemos encontrar información general o información específica para determinados procesos. Linux distingue los distintos tipos de información mediante el número de i-nodo. Un número de i-nodo en Linux es representado como un número de 32 bits, mientras que un PID (identificador de

proceso) es representado como un número de 16 bits. Con este esquema, Linux divide el número de i-nodo en dos mitades de 16 bits. La parte alta es interpretada como el identificador de proceso sobre el que dar información y la parte baja como el tipo de información que dar. Puesto que un PID de cero no es válido, este se utiliza para marcar aquellos ficheros que darán información general del sistema.

Lo que el kernel hace cuando realizamos por ejemplo un *cat /proc/cpuinfo* se muestra en la Ilustración 2. En primer lugar, el proceso creado por el *shell*, solicita información mediante la lectura del fichero. El VFS captura la solicitud y establece que el tipo de fichero a leer es un pseudo fichero del *procfs*.

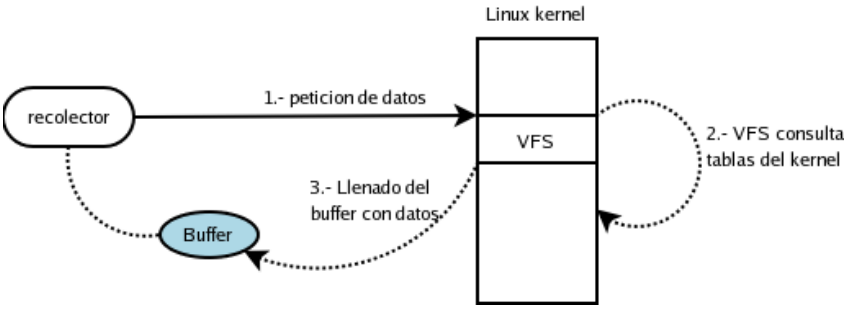

Ilustración 2

El susbsistema *procfs*, consulta las tablas del kernel para encontrar la información requerida por el proceso.

Las estructuras de datos del kernel consultadas, dependen del tipo de información que se quiera obtener (global, específica, acerca de la cpu, acerca de un proceso concreto, etc.). Una vez que se han recogido los datos, sólo queda depositarlos en el buffer del proceso.

El aspecto más importante es que el proceso de recogida de los datos es completamente transparente desde el punto de vista externo.

3.- Arquitectura de lkmonitor

Lkmonitor tiene dos componentes diferenciados implementados como hilos. Uno de ellos (el hilo principal) controla el bucle de eventos y el otro (el hilo recolector) se encargar de recoger la información actualizada. Los dos hilos comparten algunas estructuras de datos, pero las más importantes son los objetos de Gtk. La biblioteca Gtk no proporciona un acceso seguro por parte de múltiples hilos, por eso necesitamos sincronización y exclusión mutua para poder acceder de forma segura a objetos como etiquetas, paneles o botones.
Afortunadamente, Gtk ofrece dos primitivas para bloquear hilos e implementar secciones críticas. Estas funciones son: *gdk_threads_enter()* y *gdk_threads_leave()*. De este modo evitamos que el hilo principal acceda a los objetos mientras el hilo recolector está depositando en ellos la información actualizada. El comportamiento del hilo recolector se muestra en la Ilustración 3.

Lkmonitor es una aplicación que muestra información y la actualiza continuamente. El ciclo del recolector es simple: determinar si la información es dinámica (en el caso del panel de la cpu es estática), recoger la información actualizada y mostrarla en la ventana. Luego, el ciclo comienza de nuevo.

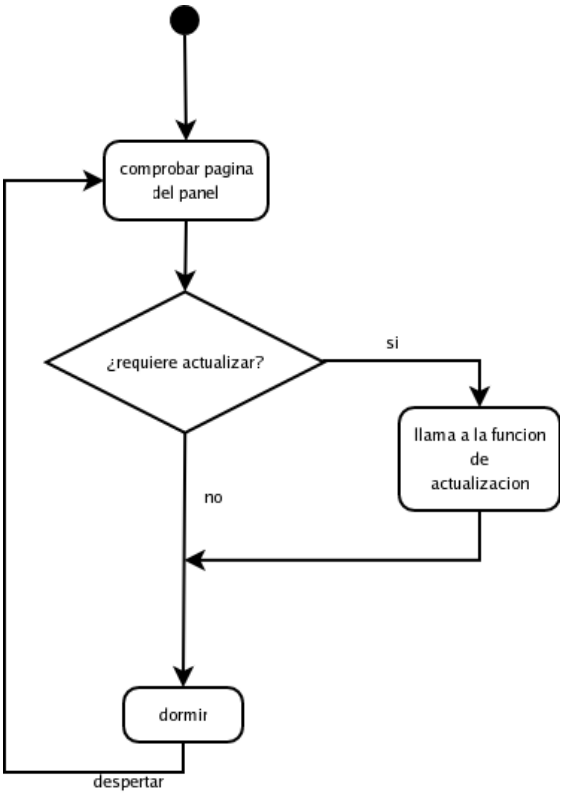

Ilustración 3

Como podemos ver en la Ilustración 4, la aplicación comienza creando algunas estructuras de datos donde se almacenará la información recogida. Después inicializa el subsistema de hilos, de forma que se puedan utilizar las primitivas de bloqueo de hilos. Después de esto, crea la ventana principal, lanza el hilo recolector y por último ejecuta el bucle de control de eventos.

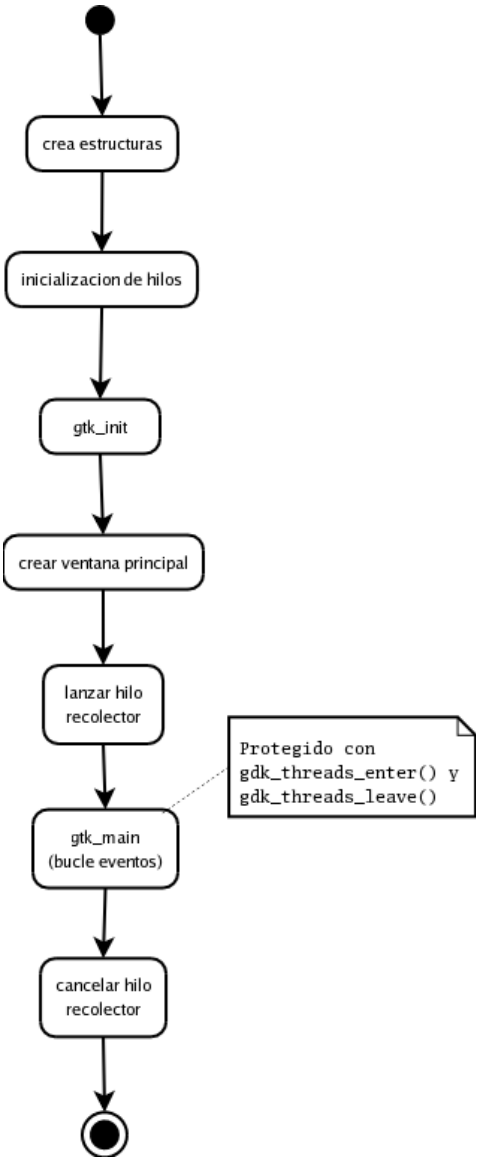

Ilustración 4

El hilo recolector, llama a la familia de funciones *update* para actualizar los datos *(update_cpu, update_mem)*. Estas funciones, en primer lugar, realizan la recogida de datos mediante las llamadas a las funciones *get_*_info*. Estas funciones son las que realizan el trabajo real de leer y parsear el fichero para extraer la información. Cuando este trabajo está hecho, las funciones *update*, impiden el acceso a los objetos de Gtk bloqueando otros hilos mediante *gdk_threads_enter()*. Cuando la ventana está actualizada, liberan la sección crítica.

El comportamiento de las funciones de actualización, se muestra en la Ilustración 5.

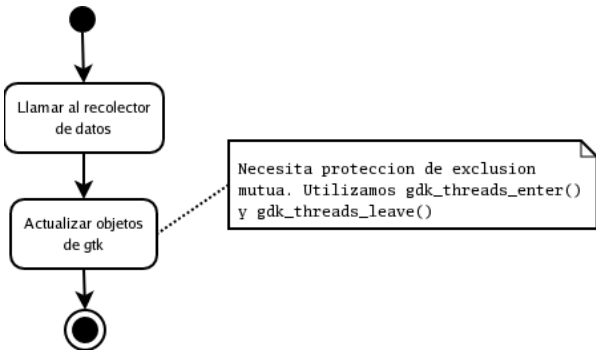

Ilustración 5