APAGADO AUTOMÁTICO DE UN EQUIPO DE CÓMPUTO

Gloria Castro Muñoz
Erick Flores Enríquez
Job Norberto Roque Ramos
Hugo Cesar Vázquez Martínez
gdea@ece.buap.mx

RESUMEN

Actualmente en la mayoría de los trabajos se usan los equipos de cómputo como una herramienta para controlar procesos o almacenar información que es de suma importancia para los empresarios, siendo necesario protegerlos de apagones inesperados que pudieran provocar problemas y así reflejarse en perdidas de dinero.

Como una solución a este problema, se plantea el diseño del apagado automático de un sistema de computo vía software, basado en la comunicación de éste con un UPS, aprovechando el tiempo de suministro energético proporcionado después de ocurrir algún problema con el suministro eléctrico.

INTRODUCCIÓN

En estudios realizados en años recientes por empresas de mantenimiento de computadoras, se obtuvo como resultado que el promedio de las computadoras, están sujetas a 289 disturbios de energía potencialmente dañinos por año.

En el mejor de los casos, un problema de energía puede derivar en apagados (shutdowns) inesperados y dañar los equipos. En el peor, un mal suministro eléctrico puede corromper archivos críticos de datos ó dañar permanentemente el equipo.

Algunos de los problemas como cortes de energía en computadoras pueden ser eliminados por medio de estabilizadores de tensión como los no_break (UPS), brindando una protección temporal y el sistema debe ser apagado en forma segura y ordenada por alguna persona. Sin embargo en ocasiones las computadoras están trabajando sin alguien que las esté vigilando lo cual podría provocar que la energía suministrada por el no_break se agote y la computadora quede expuesta nuevamente a los disturbios del suministro eléctrico. En este trabajo se desarrolla un apagado automático del equipo de cómputo mediante una interfaz con el dispositivo de alimentación (no_break), así como el software requerido por dicha interfaz.

Cuando existe falta de suministro eléctrico para el UPS, este tiene un tiempo limitado de energía que este puede proporcionarle a una computadora, por lo cual si no es apagada correctamente en ese intervalo de tiempo se apagara de forma errónea y quedara expuesta a otros posibles cortes de energía y provocar serios problemas cuando se encienda. Esto se puede evitar realizando un apagado de forma automática y correcta por parte de la propia computadora. Para que pueda llevarse acabo es necesario que la computadora tenga una forma de comunicación con el UPS tal que este ultimo le indique cuando existe falta de suministro eléctrico y que solamente queda un tiempo determinado para poder apagarla. Tal comunicación se establece mediante la utilización del puerto serie de la computadora y un interruptor interno del UPS, realizando la conexión como se muestra en el diagrama de la figura 1.Cabe mencionar que solo basta con activar el interruptor para proceder a apagar la computadora.

Como parte de la comunicación entre los dos aparatos, la función del UPS es la de activar el interruptor cuando se presente una falta de energía o cuando no se pueda seguir regulando las sobrecargas y por tanto el suministro de energía, para la computadora, estará variando. La función de la computadora es la de enviar continuamente pulsos por una salida del puerto serie y simultáneamente estar monitorizando la posible recepción de los pulsos en otra terminal del puerto serie; cuando exista la recepción de los pulsos será indicativo de que el interruptor a sido cerrado por lo que se tiene que apagar la computadora.

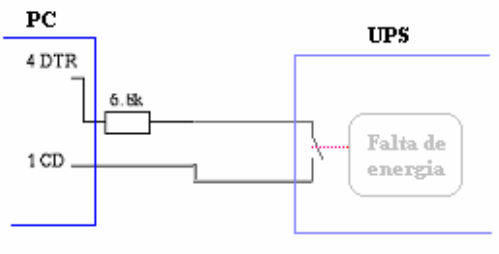

Figura 1. Diagrama de interconexión entre el UPS y la computadora

2. CUERPO DEL TEXTO

El trabajo se realiza bajo ambiente LINUX, implicando que para poder implementar la solución es indispensable que el equipo de cómputo cuente con este software. Para establecer un monitoreo constante y transparente para el usuario, es decir siempre que la maquina sea encendida el monitoreo se active automáticamente, lo que hacemos es convertir el programa en un proceso daemon, es decir que independientemente de los procesos que el usuario inicie y trabaje, este programa estará activado y solamente el usuario root podrá detener este proceso.

Para realizar el programa, se dividieron las tareas en distintas funciones, como lo son: el manejar los posibles errores que se puedan presentar, en el inicio y durante su ejecución; el control del puerto serial, para establecer la comunicación con el UPS; el programa principal, que hace uso de todas las funciones y se encarga de convertirse a si mismo en un daemon para finalmente apagar la computadora por software cuando se presente la señal de activación.

Para activar el programa se hace uso de un scrip [2] el cual se encarga ya sea de iniciar, detener, parar o reactivar el proceso según desee el usuario, y para esto se pide que el usuario (root) proporcione la ubicación y el nombre del descriptor de archivo del puerto serial, y el número correspondiente a la opción. Una vez elegida una opción, el scrip ejecuta el programa principal 'apagado'[3] pasándole como parámetros: El número de argumentos proporcionados por el usuario y el descriptor de archivo si es que lo proporciona. El programa 'apagado' verifica que hallan sido pasados solamente dos argumentos puesto que de lo contrario no puede ejecutare el programa, lo que sigue es ejecutar la función *daemon_start* mediante la cual verificamos la existencia del descriptor de archivo proporcionado por el usuario y se convierte el proceso en daemon, posteriormente se abre el descriptor de archivo en modo de lectura y escritura en caso de fallar esta instrucción, el error es canalizado mediante *err_sys*. Después se establece que si existe la presencia de la señal de interrupción (ctrl.+z) el programa debe ejecutar *offandexit*, o la señal de termino de proceso mandada por el sistema cuando el usuario apaga la computadora. Una vez realizado todos los pasos anteriores se procede a estar verificando la activación del interruptor (del UPS) lo cual se efectúa primero colocando en la salida del puerto serial un 0, puesto que si ya existía un 1, no se interprete como una activación del interruptor. Y después establecer un ciclo en el cual se envía un 1 mediante *'mandarseñal'* y se verifica la condición de activación mediante *'señalactiva'* y si la señal condición es activada entonces se procede a apagar el sistema mediante *'runshutdown'* saliendo del ciclo y finalizando el programa.

Dentro del programa 'apagado' se definen: *'SHUTDOWN_CMD'* que representa la instrucción del sistema (/sbin/shutdown –t2 –h now) que realiza el apagado automático; la función *'offanexit'* que manda un mensaje de que el proceso será finalizado ya sea por una señal del sistema cuando se apaga el sistema de computo y una por interrupción del usuario (ctrl+z) y asigna a la salida del puerto serial 0 para finalmente ejecutar *daemon_shutdown* ; y la función *'runshutdown'* que se encarga de efectuar el apagado ejecutando *'SHUTDOWN_CMD'* y avisando a todos los usuarios activos que se esta ejecutando el cierre del sistema por tanto se apagara el sistema de cómputo. En caso de no poder realizar el apagado, canaliza esto como un error mediante *err_ret*.

El programa DAEMON.C contiene las siguientes funciones para crear el proceso demonio [1].

*Función *void daemon_start* la cual inicializa el proceso demonio y consta de las siguientes partes:

1. Se ejecuta un fork para crear un nuevo proceso (proceso hijo) y terminar el proceso padre.
2. Creamos una nueva sesión al proceso hijo por medio de una llamada a setsid para que el proceso hijo no tenga una terminal de control.
3. Hacemos que el directorio de trabajo sea el directorio raíz.
4. Establecemos la umask a cero para tener libertad de crear archivos y directorios.
5. Cerramos los descriptores de archivo que el proceso hijo haya heredado y que no necesite.

En caso de no poder realizar su objetivo, es interpretado como un error, administrandose como *error_sys* cuando no se puede ejecutar la parte 1 y como *error_ret* en todas las demas partes.

*Función *void daemon_shutdown* que tiene a cargo las siguientes tareas:

1. Escribe al administrador de ingreso al sistema el mensaje de error.
2. Cierra la conexión con el administrador de ingreso al sistema
3. Cierra todos los archivos abiertos.

El programa 'dispositivo' es de suma importancia puesto que en el se definen las funciones mediante las que se realiza la comunicación con el UPS que es lo que

determina la acción a realizar. Dichas funciones se describen a continuación:

'Ioctl' se encarga de realizar el acceso al puerto serial, ya sea para enviar o leer un dato; mediante un descriptor de archivo el cual contiene la información sobre el periférico que deseamos utilizar. Cuando la petición para utilizar el puerto serie ha fallado es indicativo de que no puede realizarse la comunicación con el UPS y la acción a tomar es la de abortar la ejecución del proceso y avisar de que es un error grabe, lo cual se realiza mediante la utilización de la función 'err_sys'.

'señalactiva' es utilizada para leer la información que llegue al puerto serial, específicamente por el pin 1 (CD), y cuando se lea un 1 entonces será indicativo de que el interruptor se ha activado y entonces la función regresa 1, de lo contrario regresa 0.

'mandarseñal' la finalidad es mandar un 1 por el pin 4 (DTR). Que es lo que se recibe en el caso de que el UPS active el interruptor.

Cuando se está ejecutando un proceso pueden ocurrir distintos problemas [1], algunos de ellos son de carácter fatal, tal que será imposible seguir ejecutando el proceso demonio. Otros no serán tan fatales pero es necesario informar al usuario que han sucedido. Por esta razón fue necesario crear un archivo que administrara los errores clasificándolos como errores fatales, errores no fatales, errores relacionados con el sistema y errores no relacionados con el sistema.

err_ret es un una función de error no fatal relacionado con llamada al sistema la cual imprimirá el mensaje que le fue pasado así como la información que envía el sistema debido al error ocurrido y regresará a la línea del programa de donde fue llamada.

err_msg ésta es una función de error de carácter informativo ya que no es fatal y no está relacionada con llamada al sistema, imprimirá el mensaje que le fue pasado y regresará a la línea del programa de donde fue llamada.

err_quit ésta es una función de error fatal no relacionado con llamada al sistema que imprimirá el mensaje que le fue pasado y terminará el programa.

err_sys ésta es una función de error fatal relacionado con la llamada al sistema que imprimirá el mensaje que le fue pasado así como el que envía el sistema y terminará la ejecución del programa.

err_dump es una función de error fatal relacionado con la llama al sistema que imprimirá el mensaje que le fue pasado así como el que envía el sistema y a diferencia de la anterior descargará el núcleo y terminará la ejecución del programa.

3. CONCLUSION

Este prototipo tiene la ventaja de ser utilizado desde un simple usuario hasta aquellas empresas que hagan uso de servidores, además el tipo de ambiente (LINUX) en el que trabaja el presente proyecto puede ser adquirido gratuitamente.

Como limitación se puede mencionar que este proyecto no es viable bajo otro tipo de sistemas operativos, y es necesario contar con un banco de baterías que nos brinden el tiempo suficiente para realizar el correcto apagado del equipo de cómputo.

En el futro se pretende lograr mayor eficiencia en el desempeño del programa mediante la canalización de un posible error para solucionarlo a través de software, y de esta forma hacerse independiente de un usuario en caso de presentarse algún error.

4. AGRADECIMIENTOS

Agradecemos la colaboración del MC. Manuel Rendón Marín por habernos brindado su apoyo en la elaboración de éste proyecto.

REFERENCIAS

[1] Wall Kart, Programación en Linux con ejemplos, Prentice Hall, Buenos Aires, (2000) 191-200.

[2] Facundo Arena Héctor, Linux Avanzado guía del administrador, Manuales Compumagazine. Buenos Aires, (2000) ,173-185.

[3]http://www.linuxfocus.org/English/January2001/article186.shtml