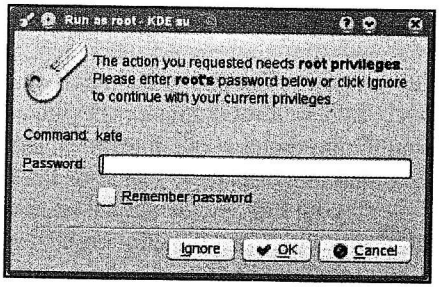


**Figura 1: KDE usa kdesu para preguntar la contraseña de administrador.**

tino. *env_reset* suele ir mano a mano con el parámetro *env_keep*, que contiene una lista de variables de entorno a mantener. Por defecto, lo que se mantiene usualmente está relacionado con diversas variables de lenguajes y ubicaciones. (Esta es en realidad una funcionalidad de seguridad, ya que el usuario podría configurar su ruta de manera que se buscasen primero directorios no estándar, y si la variable de ruta no se resetea, tendríamos la posibilidad potencial de ejecutar otros comandos como usuario destino. Por ejemplo, un script Shell iniciado mediante *sudo*, que use *find*, podría ser modificado para que iniciara */tmp/find* con consecuencias potencialmente peligrosas).

Por defecto, *sudo* se ejecutará como administrador. Podemos cambiar este comportamiento especificándole *-u*. Muchos usuarios creen erróneamente que *sudo* significa que el comando se ejecutará como administrador. Aunque esta es probablemente la manera más habitual en la que se usa *sudo*, no siempre es necesario o incluso deseable. Por ejemplo, a menudo, una cuenta de usuario tiene unas variables de entorno que el administrador no tiene.

Otra opción de seguridad es *targetpw*, la cual le indica a *sudo* que pregunte por la contraseña del usuario destino. Si no está habilitada, *sudo* preguntará la contraseña del usuario *invocante*.

Tenga presente que estas configuraciones son *defaults*. Podemos configurar comandos específicos (o conjuntos de comandos) de manera que las variables de entorno configuradas en línea de comandos no se eliminen si la configuración por defecto se lo indica.

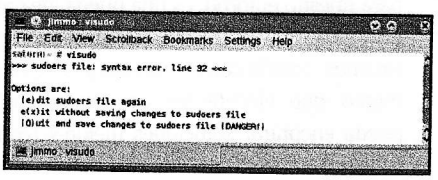


**Figura 2: Visudo informa de un error de sintaxis.**

Puede consultar la página man de *sudoers* para más detalles.

La sintaxis puede ser confusa al principio, pero según mi experiencia, uno se acostumbra rápidamente a ella. El formato general de estas entradas es:

```
user_alias host_alias = ➱
(runas_alias) tag command_alias
```

Tenga en cuenta que *alias* no significa que sólo podamos usar alias, ya que cada entidad es un alias de sí misma. Por ejemplo, podríamos especificar un usuario, comando, etc. sin tener que definir de manera explícita un alias de antemano.

Debido a que la primera entrada es el usuario, se las llama líneas de *user specification*. Es muy probable que encuentre una línea como la siguiente:

```
ALL ALL=(ALL) ALL
```

Tenga cuidado con esto, porque esta línea permite ejecutar *sudo* a todos los usuarios, para todos los usuarios de destino de todos los hosts, y que ejecuten todos los comandos.

Como aún requiere que introduzcamos la contraseña del usuario destino, no es una entrada libre para hacer cualquier cosa en el sistema que queramos. Sin embargo, esto proporciona la seguridad adicional de no tener que iniciar una sesión ilimitada como el usuario destino.

También es muy probable que encontremos una entrada como:

```
root ALL=(ALL) ALL
```

Esta entrada es esencialmente la misma que el comando anterior, pero se aplica sólo al usuario administrador. El comando permite explícitamente al administrador ejecutar todos los comandos como todos los usuarios.

Un mecanismo útil es la capacidad de definir alias, que generalmente se usan con listas de diversos objetos (por ejemplo, una lista de usuarios dentro de una compañía que tienen funciones similares). Las definiciones de los alias tienen todas el mismo formato básico:

```
alias_type space alias_name = ➱
object1, objects2, ...
```

Cuando listamos objetos, no estamos limitados a objetos estándar, como usuarios de */etc/passwd*, o hosts específicos, sino que tenemos un amplio rango de opciones (véase la página man de *sudoers*).

Según mi experiencia, agrupar objetos en alias facilita la administración de configuraciones complejas. Si se permite a diferentes conjuntos de personas ejecutar diferentes conjuntos de comandos, los alias nos ayudan a asegurar que todos tienen acceso a los comandos correctos. Las modificaciones pueden realizarse en un único sitio, en lugar de a lo largo de todo el archivo.

La opción *runas_alias* define el usuario objetivo para una lista de comandos. Al igual que con los restantes objetos, podemos tener múltiples *runas_alias* en la misma línea. Por ejemplo, algunos comandos que querrá ejecutar como administrador, mientras que otros sólo querrá ejecutarlos como un usuario con menos permisos.

Un *command_alias* consiste en nombres, directorios u otros alias de comandos. Hay un par de cosas que debemos destacar en este punto. En primer lugar, el nombre del comando debe ser completamente cualificado (es decir, debe incluir la ruta completa). Si especificamos un comando que no está completamente cualificado, *visudo* lo verá como un error de sintaxis.

Sólo con listar el nombre del comando, podemos especificar cualquier opción o parámetro en línea de comandos. En función de las circunstancias, puede que no sea bueno. Para limitar lo que un usuario concreto puede hacer, debemos definir explícitamente qué opciones están permitidas.

Tenga en cuenta que todos los argumentos deben coincidir *exactamente* a menos que el comando fuese especificado con comodines. Si usamos un asterisco, coincidirá con cualquier argumento (o ninguno) como si no hubiésemos especificado ningún comodín.

Cada entrada puede tener cero o más etiquetas asociadas a ella. La etiqueta más común que he visto es *NOPASSWD*, que indica que los comandos se ejecutan sin pedir previamente la contraseña. Esta etiqueta *NOPASSWD* tiene su par en la etiqueta *PASSWD*, que obliga a proporcionar la contraseña adecuada. (Pedir una contraseña es la configuración por defecto). Al combinar todas estas opciones a la vez, podríamos terminar con algo como lo siguiente:

```
jimmo ALL = (operator) ➱
/bin/more /var/log/messages, ➱
(root) NOPASSWD: /sbin/fdisk -l
```

Aquí, al usuario *jimmo* de todos los hosts se le permite ejecutar *more* como el usuario pero *sólo* sobre el archivo */var/log/messages*. El usuario puede también ejecutar */sbin/fdisk -l*