

Creación de un Módulo Kernel Linux para la lectura de la temperatura de la CPU

MIGUEL MEDINA CANTOS

Mayo 2023



Índice

1. Introducción	3
2. Funcionamiento del Módulo del Kernel	4
3. Ventajas y Desventajas de un Módulo del Kernel	7
3.1. Ventajas	7
3.2. Desventajas	7
4. Elección del sistema de ficheros /proc para el código	8
5. Como funciona la compilación de un Módulo del Kernel	9
6. Instrucciones y/o pasos a seguir	10
6.1. Paso 1: Preparación del entorno.	10
6.2. Paso 2: Creación del módulo del kernel	10
6.3. Paso 3: Compilación del módulo del kernel	10
6.4. Paso 4: Cargar el módulo del kernel	11
6.5. Paso 5: Verificar la carga del módulo	11
6.6. Paso 6: Leer la temperatura de la CPU	11
6.7. Paso 7: Descargar el módulo del kernel	11
6.8. Paso 8: Instalar el módulo del kernel y cargarlo en el sistema para siempre	12
6.9. Paso 9: Comprobar la instalación del módulo	13
7. Módulo del kernel para leer la temperatura de la CPU	14
7.1. Código del módulo	14
8. Explicación del Makefile	15
9. Explicación del código	16
9.1. Inclusión de las bibliotecas y definiciones	16
9.2. Función <code>cpu_temp_show</code>	16
9.3. Función <code>cpu_temp_open</code>	16
9.4. Estructura <code>cpu_temp_fops</code>	17
9.5. Funciones <code>cpu_temp_module_init</code> y <code>cpu_temp_module_exit</code>	17
10. Resolución de los pasos	18
10.1. Solución paso 1: Preparación del entorno	18
10.2. Solución paso 2: Creación del módulo del kernel	19
10.2.1. Editor utilizado y creación del archivo <code>cpu_temp_module.c</code>	20
10.2.2. Creación del archivo Makefile	20
10.3. Solución paso 3: Compilación del módulo del kernel	21
10.3.1. La carpeta después de compilar	21
10.4. Solución paso 4: Cargar el módulo del kernel	22
10.5. Solución paso 5: Verificar la carga del módulo	22
10.6. Solución paso 6: Leer la temperatura de la CPU	23
10.7. Solución paso 7: Descargar el módulo del kernel	23
10.8. Solución paso 8: Instalar el módulo del kernel y cargarlo en el sistema para siempre	24
10.8.1. Instalación del módulo	24
10.8.2. Configuración de la carga automática del módulo	25
10.9. Solución paso 9: Comprobar la instalación del módulo	26
11. Bibliografía	27

1. Introducción

El propósito de este guión de prácticas es proporcionar una guía detallada para la implementación de un módulo del kernel de Linux que permita leer la temperatura de la CPU. La importancia de monitorear la temperatura de la CPU radica en la necesidad de garantizar un funcionamiento óptimo y evitar daños por sobrecalentamiento. A lo largo de este guión, se explorarán los conceptos clave y las técnicas necesarias para crear, compilar y cargar un módulo del kernel que lea la temperatura de la CPU y la muestre en el sistema de archivos *proc*.

Se abordarán los siguientes temas:

- I. Inclusión de bibliotecas y definiciones necesarias.
- II. Implementación de funciones para la lectura de la temperatura y el manejo de archivos.
- III. Creación de un archivo *Makefile* para la compilación del módulo.
- IV. Carga y descarga del módulo en el kernel de Linux.

El guión está diseñado para ser seguido paso a paso, de manera que al finalizar, se obtenga un conocimiento sólido de cómo crear y utilizar módulos del kernel para leer la temperatura de la CPU en un sistema Linux. Además, se espera que este guión sirva como base para futuras investigaciones y desarrollos en la monitorización y control de la temperatura de la CPU.



Figura 1: Un chip CPU!!

2. Funcionamiento del Módulo del Kernel

- I. **Creación del Archivo:** Durante la carga del módulo, se genera un archivo en el directorio `/proc` denominado `cpu_temp`. Este directorio es una representación en el espacio de usuario del estado del sistema en el kernel, permitiendo la comunicación entre ambos. En este caso, el archivo `cpu_temp` se utiliza para exponer la información de la temperatura de la CPU al espacio de usuario. El módulo se encarga de gestionar las operaciones de lectura sobre este archivo.
- II. **Interacción con la Interfaz de Zona Térmica:** El módulo interactúa con la interfaz de zona térmica del kernel de Linux para obtener la temperatura de la CPU. La interfaz de zona térmica es una parte del kernel que proporciona información sobre la temperatura de los componentes del sistema, incluida la CPU. El módulo hace uso de las funciones proporcionadas por esta interfaz para obtener la temperatura actual de la CPU cada vez que es necesario.
- III. **Lectura de la Temperatura:** Cuando se realiza una operación de lectura en el archivo `cpu_temp`, el módulo consulta la interfaz de zona térmica para obtener la temperatura actual de la CPU. Luego, el módulo escribe esta información en el buffer de lectura proporcionado por el sistema. De esta manera, el contenido de este buffer se devuelve al usuario o aplicación que realizó la operación de lectura. Esto permite que los usuarios y aplicaciones puedan obtener la temperatura de la CPU leyendo el archivo `cpu_temp`.

Podemos ver una representación de estos tres puntos en la siguiente imagen:

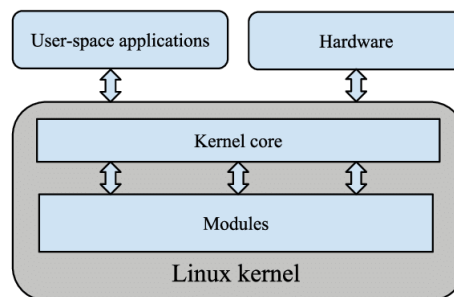


Figura 2: Funcionamiento del Módulo del Kernel

IV. **Funcionamiento de la Interfaz de Zona Térmica:** El sistema de gestión térmica de Linux, conocido como la interfaz de zona térmica, es una parte integral del kernel que se encarga de la monitorización y gestión de la temperatura de los distintos componentes del sistema. Esta interfaz proporciona una manera estandarizada de obtener información detallada sobre las temperaturas de los componentes del sistema, incluida la CPU.

El módulo `cpu_temp` se aprovecha de esta interfaz para obtener la temperatura de la CPU. Cuando se realiza una lectura al archivo `/proc/cpu_temp`, el módulo consulta la interfaz de zona térmica para obtener la temperatura actual de la CPU. La interfaz de zona térmica accede a la información de la CPU a través de sensores térmicos incorporados en la misma, y devuelve la temperatura en grados Celsius.

El uso de la interfaz de zona térmica para obtener la temperatura de la CPU en este módulo ofrece varias ventajas de las cuales hablaremos en la siguiente página:

- I. **Estandarización.**
- II. **Simplicidad.**
- III. **Seguridad.**

- I. **Estandarización:** Al utilizar la interfaz de zona térmica, se está utilizando una API que es consistente y bien mantenida por los desarrolladores del kernel de Linux. Esto significa que es probable que el módulo funcione en una amplia gama de sistemas y configuraciones de hardware.
- II. **Simplicidad:** En lugar de interactuar directamente con el hardware o los sensores de la CPU, el módulo puede obtener la temperatura de la CPU de una manera simple y directa. Esto simplifica el código del módulo y reduce la probabilidad de errores.
- III. **Seguridad:** Al utilizar la interfaz de zona térmica, se evita el riesgo de interferir con el hardware directamente o de causar problemas al sistema al interactuar con los sensores de la CPU de manera incorrecta. La interfaz de zona térmica gestiona el acceso a la información de temperatura de forma segura y controlada, lo que garantiza que el módulo pueda obtener la temperatura de la CPU sin causar problemas en el sistema.

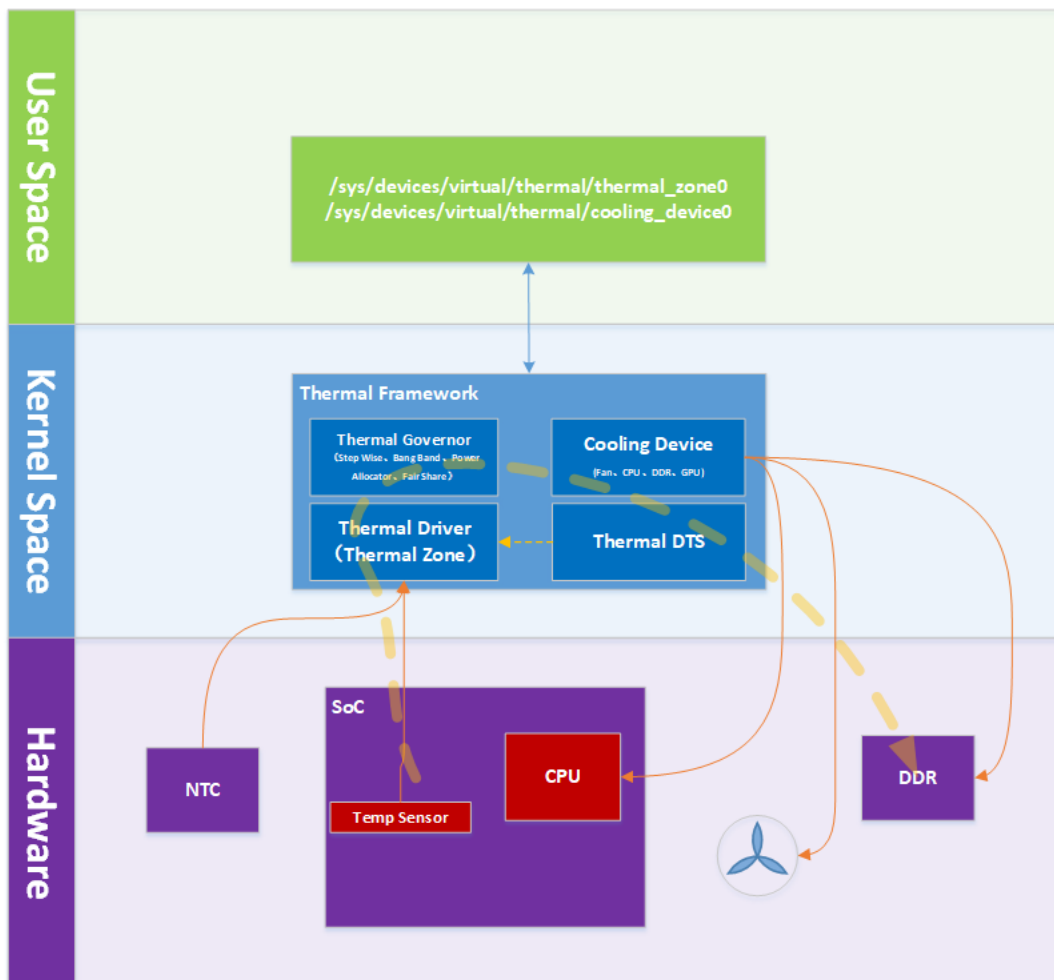


Figura 3: **Funcionamiento de la Interfaz de Zona Térmica.**

3. Ventajas y Desventajas de un Módulo del Kernel

3.1. Ventajas

- I. **Flexibilidad:** Los módulos del kernel, como el módulo `cpu_temp_module`, permiten a los usuarios añadir o quitar funcionalidades del sistema operativo sin necesidad de reiniciar.
- II. **Eficiencia:** Los módulos sólo se cargan cuando se necesitan, liberando memoria del sistema para otros usos cuando no están en uso.
- III. **Desarrollo y Pruebas:** Los módulos facilitan el desarrollo y la depuración de nuevas funcionalidades del kernel. Los desarrolladores pueden cargar y descargar su código rápidamente para probarlo.
- IV. **Información del sistema:** El módulo `cpu_temp_module` proporciona información útil sobre la temperatura de la CPU, lo que puede ser útil para el monitoreo del sistema y la prevención de problemas de sobrecalentamiento.

3.2. Desventajas

- I. **Estabilidad:** Si un módulo del kernel contiene errores, puede causar problemas de estabilidad en el sistema o incluso bloquearlo completamente.
- II. **Seguridad:** Los módulos del kernel se ejecutan con privilegios elevados. Si un módulo es vulnerable, podría ser explotado para obtener control total sobre el sistema.
- III. **Compatibilidad:** Los módulos del kernel deben ser compilados para la versión específica del kernel que se está utilizando. Si se actualiza el kernel, los módulos también deben ser recompilados.
- IV. **Complejidad:** El desarrollo de módulos del kernel requiere un entendimiento profundo del sistema operativo y del lenguaje C, lo que puede ser un desafío para los principiantes.

4. Elección del sistema de ficheros `/proc` para el código

El sistema de ficheros `/proc` es una interfaz estándar en sistemas Linux que permite acceder a información y configuraciones del kernel de manera estructurada y organizada. Esta interfaz proporciona un acceso fácil y rápido a la información del kernel, lo que permite a los usuarios y aplicaciones obtener datos relevantes sin necesidad de utilizar APIs complicadas o específicas del kernel.

Al exponer datos y configuraciones del kernel al espacio de usuario, `/proc` facilita la comunicación entre el kernel y las aplicaciones o usuarios que necesiten acceder a dicha información. Utilizar `/proc` permite a los usuarios leer y escribir información del kernel a través de archivos de texto, lo que simplifica el proceso y hace más fácil la manipulación de dichos datos.

Elegir el sistema de ficheros `/proc` asegura la compatibilidad con una amplia variedad de herramientas y aplicaciones existentes que utilizan este sistema para obtener y modificar información del kernel. De esta manera, al implementar el módulo creado para leer la temperatura de la CPU en el sistema de ficheros `/proc`, se garantiza una mayor compatibilidad y facilidad de uso en entornos Linux.



Figura 4: Sistema de ficheros `proc`.

5. Como funciona la compilación de un Módulo del Kernel

La compilación de un módulo del kernel implica la traducción del código fuente, escrito en el lenguaje de programación C, en un formato que el kernel de Linux pueda entender y ejecutar. Este proceso se realiza utilizando una herramienta de compilación, normalmente el compilador GCC (GNU Compiler Collection).

El primer paso en la compilación de un módulo del kernel es escribir el código del módulo en C. Este código debe seguir ciertas convenciones y utilizar ciertas funciones y estructuras proporcionadas por el kernel de Linux.

Una vez que el código fuente está completo, se puede proceder a la compilación del módulo. Para ello, se utiliza un archivo Makefile, que es un script que dirige la herramienta de compilación para convertir el código fuente en un módulo del kernel. Este archivo Makefile especifica las reglas para la compilación del módulo, incluyendo la ubicación del código fuente, las opciones de compilación y las dependencias entre diferentes partes del código.

Cuando se ejecuta el comando `make`, la herramienta de compilación lee el archivo Makefile y compila el módulo del kernel según las reglas especificadas. El resultado es un archivo con extensión `.ko` (kernel object), que es el módulo del kernel compilado y listo para ser cargado en el kernel de Linux.

Es importante tener en cuenta que el módulo del kernel debe ser compilado para el mismo kernel y la misma arquitectura de hardware para la que se va a cargar. Si se compila el módulo para un kernel o una arquitectura de hardware diferente, no será posible cargar el módulo.

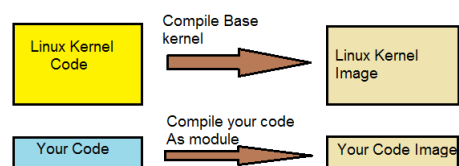


Figura 5: **Compilación de un Módulo del Kernel.**

6. Instrucciones y/o pasos a seguir

6.1. Paso 1: Preparación del entorno.

- I. Asegúrate de que estés utilizando una distribución de Linux compatible con el código del módulo (el módulo asume que estás utilizando un procesador Intel). La explicación del código se encuentra en la sección [9](#). Una distribución Linux compatible sería [Ubuntu](#) .
- II. Instala las herramientas y dependencias necesarias para compilar módulos del kernel en tu sistema. En distribuciones basadas en Debian/Ubuntu, utiliza el siguiente comando:

```
1 sudo apt-get install build-essential linux-headers-$(uname -r)
```

6.2. Paso 2: Creación del módulo del kernel

- I. Cree un nuevo archivo llamado `cpu_temp_module.c` y copie el código del módulo de la sección [7.1 para seguir con los pasos](#). La explicación del código se encuentra en la sección [9](#).
- II. Crea un archivo llamado Makefile en el mismo directorio que `cpu_temp_module.c` y copia el contenido del Makefile de la sección [8.Explicación del Makefile](#).

6.3. Paso 3: Compilación del módulo del kernel

- I. Abra una terminal y navegue al directorio donde se encuentran `cpu_temp_module.c` y `Makefile`.
- II. Compile el módulo ejecutando el siguiente comando:

```
1 make
```

6.4. Paso 4: Cargar el módulo del kernel

Una vez compilado el módulo, debes cargarlo en el sistema. Utiliza el siguiente comando para cargar el módulo en el sistema:

```
1 sudo insmod cpu_temp_module.ko
```

Para cargar el módulo, necesitas usar el comando anterior con privilegios de administrador.

6.5. Paso 5: Verificar la carga del módulo

Para asegurarte de que el módulo se ha cargado correctamente, puedes utilizar el comando:

```
1 lsmod | grep cpu_temp_module
```

Si el módulo se ha cargado correctamente, deberías ver `cpu_temp_module` en la salida.

6.6. Paso 6: Leer la temperatura de la CPU

Ahora que el módulo está cargado, puedes leer la temperatura de la CPU. Navega a `/proc/cpu_temp` y utiliza el comando:

```
1 cat /proc/cpu_temp
```

para leer la temperatura.

6.7. Paso 7: Descargar el módulo del kernel

Cuando hayas terminado, puedes descargar el módulo del kernel utilizando el comando:

```
1 sudo rmmod cpu_temp_module
```

6.8. Paso 8: Instalar el módulo del kernel y cargarlo en el sistema para siempre

Si deseas que el módulo se cargue automáticamente en el arranque, puedes instalarlo. Ejecuta el comando:

```
1 sudo make install
```

Para cargar automáticamente el módulo del kernel al inicio, debes agregar el nombre del módulo a un archivo de configuración en `/etc/modules-load.d/`. Aquí te explico cómo hacerlo:

1. Abre un terminal.
2. Crea un nuevo archivo de configuración con el nombre de tu módulo, por ejemplo, `cpu_temp_module.conf`:

```
1 sudo touch /etc/modules-load.d/cpu_temp_module.conf
```

3. Abre el archivo creado con un editor de texto, como `nano`:

```
1 sudo nano /etc/modules-load.d/cpu_temp_module.conf
```

4. Agrega el nombre de tu módulo (sin la extensión `.ko`) al archivo. En este caso, sería `cpu_temp_module`. Asegúrate de que el nombre del módulo esté en una nueva línea.

```
1 cpu_temp_module
```

5. Guarda y cierra el archivo. En el caso de `nano`, presiona `Ctrl + X`, luego `Y` para confirmar y finalmente `Enter` para salir.

Después de estos pasos, el módulo se cargará automáticamente al inicio cada vez que reinicies el sistema. Recuerda que tu módulo debe estar instalado correctamente en `/lib/modules/$(uname -r)/extra/` o en otra ubicación reconocida por el sistema para que esto funcione.

6.9. Paso 9: Comprobar la instalación del módulo

Para verificar que el módulo se ha instalado correctamente, puedes reiniciar tu máquina y luego utilizar el comando:

```
1 lsmod | grep cpu_temp_module
```

Si el módulo se ha instalado correctamente, deberías ver `cpu_temp_module` en la salida.

7. Módulo del kernel para leer la temperatura de la CPU

7.1. Código del módulo

Módulo del kernel para leer la temperatura de la CPU

```

1 #include <linux/init.h>
2 #include <linux/module.h>
3 #include <linux/kernel.h>
4 #include <linux/proc_fs.h>
5 #include <linux/seq_file.h>
6 #include <linux/hwmon.h>
7 #include <linux/thermal.h>
8
9 #define PROC_FILENAME "cpu_temp"
10
11 static int cpu_temp_show(struct seq_file *m, void *v)
12 {
13     struct thermal_zone_device *tz;
14     int temperature;
15
16     tz = thermal_zone_get_zone_by_name("x86_pkg_temp");
17     if (tz) {
18         if (thermal_zone_get_temp(tz, &temperature) == 0) {
19             seq_printf(m, "%d grados celsius\n", temperature / 1000);
20         } else {
21             seq_puts(m, "Error reading temperature\n");
22         }
23     } else {
24         seq_puts(m, "Thermal zone not found\n");
25     }
26
27     return 0;
28 }
29
30 static int cpu_temp_open(struct inode *inode, struct file *file)
31 {
32     return single_open(file, cpu_temp_show, NULL);
33 }
34
35 static const struct proc_ops cpu_temp_fops = {
36     .proc_open = cpu_temp_open,
37     .proc_read = seq_read,
38     .proc_lseek = seq_lseek,
39     .proc_release = single_release,
40 };
41
42 static int __init cpu_temp_module_init(void)
43 {
44     if (!proc_create(PROC_FILENAME, 0, NULL, &cpu_temp_fops)) {
45         return -ENOMEM;
46     }
47
48     return 0;
49 }
50
51 static void __exit cpu_temp_module_exit(void)
52 {
53     remove_proc_entry(PROC_FILENAME, NULL);
54 }
55
56 module_init(cpu_temp_module_init);
57 module_exit(cpu_temp_module_exit);
58
59 MODULE_LICENSE("GPL");
60 MODULE_AUTHOR("Miguel Medina Cantos");
61 MODULE_DESCRIPTION("Modulo del kernel simple para leer la temperatura de la CPU");

```

8. Explicación del Makefile

Makefile para compilar y cargar e instalar el módulo del kernel

```

1 obj-m += cpu_temp_module.o
2
3 KDIR := /lib/modules/$(shell uname -r)/build
4 PWD := $(shell pwd)
5
6 default:
7     $(MAKE) -C $(KDIR) M=$(PWD) modules
8
9 install:
10    $(MAKE) -C $(KDIR) M=$(PWD) modules_install
11    depmod -A
12
13 clean:
14    $(MAKE) -C $(KDIR) M=$(PWD) clean
15
16 uninstall:
17    rm /lib/modules/$(shell uname -r)/extra/cpu_temp.ko
18    depmod -A

```

Este Makefile se utiliza para compilar el módulo del kernel y cargarlo e instalarlo en el sistema. A continuación, se describen las diferentes partes del archivo:

- I. `obj-m += cpu_temp_module.o`: Define el archivo objeto que se generará a partir del código fuente del módulo.
- II. `KDIR := /lib/modules/$(shell uname -r)/build`: Define la ubicación del directorio de construcción del kernel.
- III. `PWD := $(shell pwd)`: Define la ubicación del directorio actual.
- IV. `default`: Esta regla compila el módulo del kernel utilizando el directorio de construcción del kernel y el directorio actual.
- V. `install`: Esta regla instala el módulo del kernel en el directorio adecuado y actualiza las dependencias de los módulos.
- VI. `clean`: Esta regla limpia los archivos generados durante la compilación.
- VII. `uninstall`: Esta regla desinstala el módulo del kernel y actualiza las dependencias de los módulos.

9. Explicación del código

El código del módulo del kernel se divide en varios bloques, que se describen a continuación:

9.1. Inclusión de las bibliotecas y definiciones

Inclusión de las bibliotecas y definiciones

```
1 #include <linux/init.h>
2 #include <linux/module.h>
3 #include <linux/kernel.h>
4 #include <linux/proc_fs.h>
5 #include <linux/seq_file.h>
6 #include <linux/hwmon.h>
7 #include <linux/thermal.h>
8
9 #define PROC_FILENAME "cpu_temp"
```

Este bloque incluye las bibliotecas necesarias para crear un módulo del kernel, acceder al sistema de archivos proc, leer datos de sensores de temperatura y realizar operaciones con archivos secuenciales. Además, se define el nombre del archivo que se creará en el sistema de archivos proc.

9.2. Función cpu_temp_show

Función cpu_temp_show

```
1 static int cpu_temp_show(struct seq_file *m, void *v)
2 {
3     struct thermal_zone_device *tz;
4     int temperature;
5
6     tz = thermal_zone_get_zone_by_name("x86_pkg_temp");
7     if (tz) {
8         if (thermal_zone_get_temp(tz, &temperature) == 0) {
9             seq_printf(m, "%d grados celsius\n", temperature / 1000);
10        } else {
11            seq_puts(m, "Error reading temperature\n");
12        }
13    } else {
14        seq_puts(m, "Thermal zone not found\n");
15    }
16
17    return 0;
18 }
```

Esta función se encarga de leer la temperatura de la CPU y escribir la información en el archivo secuencial m. Primero, busca la zona térmica llamada "x86_pkg_temp" y, si la encuentra, intenta leer la temperatura. Si la lectura es exitosa, escribe la temperatura en el archivo secuencial; de lo contrario, escribe un mensaje de error. Hay que decir que la zona térmica corresponde con los procesadores intel así que con otro procesador no funcionaría.

9.3. Función cpu_temp_open

Función cpu_temp_open

```
1 static int cpu_temp_open(struct inode *inode, struct file *file)
2 {
3     return single_open(file, cpu_temp_show, NULL);
4 }
```

Esta función se utiliza para abrir el archivo en el sistema de archivos proc y asociar la función cpu_temp_show para manejar las operaciones de lectura.

9.4. Estructura `cpu_temp_fops`

Estructura `cpu_temp_fops`

```

1 static const struct proc_ops cpu_temp_fops = {
2     .proc_open = cpu_temp_open,
3     .proc_read = seq_read,
4     .proc_lseek = seq_lseek,
5     .proc_release = single_release,
6 };

```

Esta estructura define las operaciones permitidas en el archivo del sistema de archivos `proc`. Se asigna la función `cpu_temp_open` para la apertura del archivo, y las funciones `seq_read`, `seq_lseek` y `single_release` para la lectura, búsqueda y liberación del archivo, respectivamente.

9.5. Funciones `cpu_temp_module_init` y `cpu_temp_module_exit`

Función `cpu_temp_module_init`

```

1 static int __init cpu_temp_module_init(void)
2 {
3     if (!proc_create(PROC_FILENAME, 0, NULL, &cpu_temp_fops)) {
4         return -ENOMEM;
5     }
6
7     return 0;
8 }

```

Función `cpu_temp_module_exit`

```

1 static void __exit cpu_temp_module_exit(void)
2 {
3     remove_proc_entry(PROC_FILENAME, NULL);
4 }

```

Estas funciones se utilizan para inicializar y finalizar el módulo del kernel, respectivamente. La función `cpu_temp_module_init` crea el archivo en el sistema de archivos `proc` utilizando las operaciones definidas en `cpu_temp_fops`. La función `cpu_temp_module_exit` elimina el archivo del sistema de archivos `proc` al descargar el módulo.

10. Resolución de los pasos

10.1. Solución paso 1: Preparación del entorno

Como se ha explicado en la [sección 6.1](#).

Se utilizará el comando dado en la terminal:

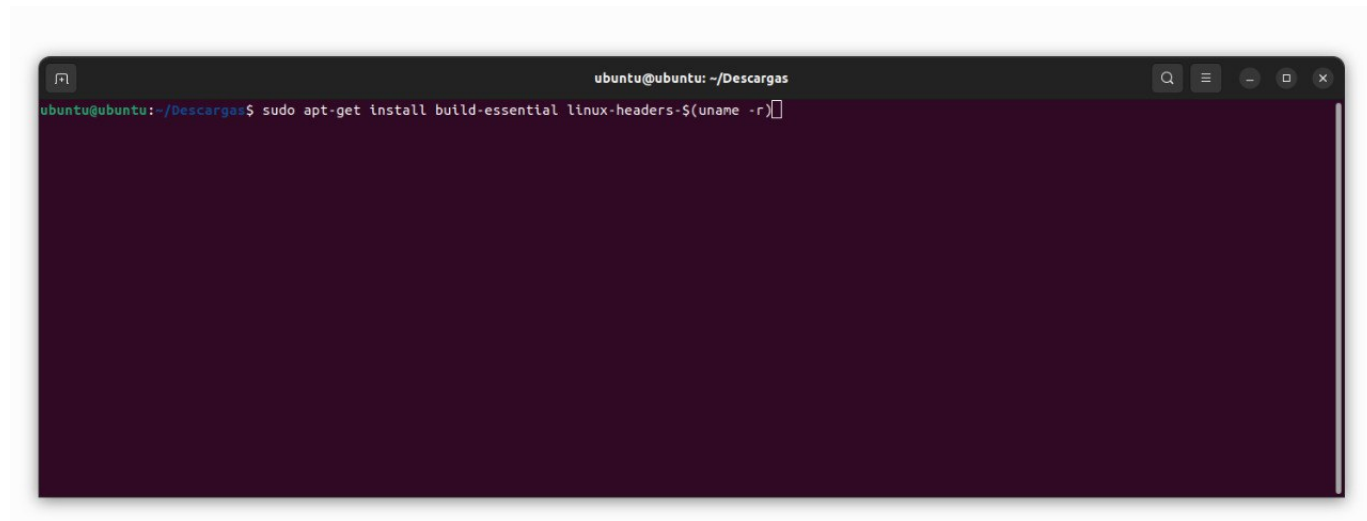


Figura 6: Terminal Ubuntu

Y con ello obtendremos las dependencias y herramientas necesarias para compilar el kernel del guión.

10.2. Solución paso 2: Creación del módulo del kernel

Para comenzar con la creación del módulo del kernel, se inicia por abrir un editor de texto de preferencia, por ejemplo, Gedit, Nano o Vim. En este nuevo archivo de texto, se copia y pega el código del módulo proporcionado anteriormente. Se guarda este archivo con el nombre `cpu_temp_module.c` en la ubicación deseada.

La ubicación en este caso será la carpeta descargas:

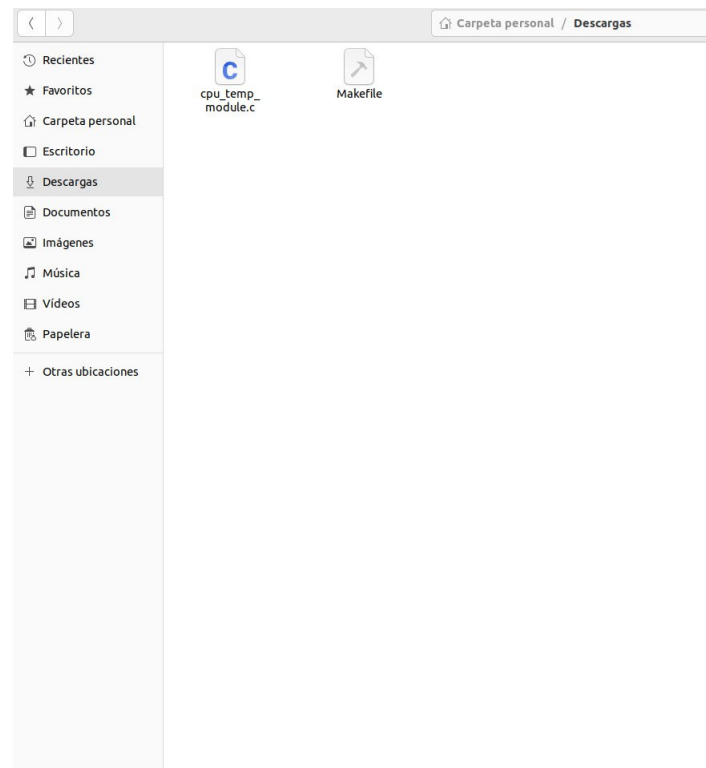
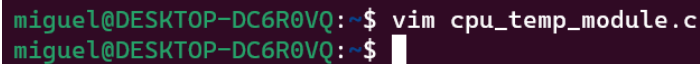


Figura 7: Carpeta Descargas

10.2.1. Editor utilizado y creación del archivo `cpu_temp_module.c`

En este caso se utilizará VIM como editor para copiar el código. Ponemos el comando en terminal para crear el archivo `cpu_temp_module.c`:

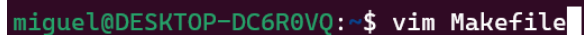
A terminal window with a dark purple background. The prompt is 'miguel@DESKTOP-DC6R0VQ:~\$'. The command 'vim cpu_temp_module.c' has been entered, and the prompt is now 'miguel@DESKTOP-DC6R0VQ:~\$' followed by a cursor.

```
miguel@DESKTOP-DC6R0VQ:~$ vim cpu_temp_module.c
miguel@DESKTOP-DC6R0VQ:~$
```

Figura 8: creación del archivo `cpu_temp_module.c`

10.2.2. Creación del archivo Makefile

Luego, aún en el mismo directorio que `cpu_temp_module.c`, se abre otro nuevo archivo de texto en el editor de texto elegido. En este segundo archivo, se copia y pega el contenido del Makefile que se proporcionó anteriormente. Se guarda este archivo como `Makefile`.

A terminal window with a dark purple background. The prompt is 'miguel@DESKTOP-DC6R0VQ:~\$'. The command 'vim Makefile' has been entered, and the prompt is now 'miguel@DESKTOP-DC6R0VQ:~\$' followed by a cursor.

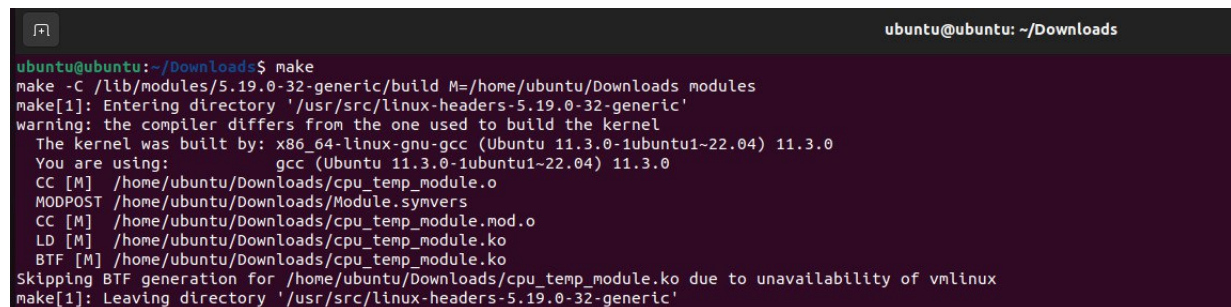
```
miguel@DESKTOP-DC6R0VQ:~$ vim Makefile
```

Figura 9: creación del archivo `Makefile`

Es importante asegurarse de que ambos archivos, `cpu_temp_module.c` y `Makefile`, estén en el mismo directorio. Al final de este proceso, se tendrán dos archivos en el directorio seleccionado: `cpu_temp_module.c`, que contiene el código del módulo del kernel, y `Makefile`, que tiene las instrucciones para compilar el módulo. La creación del módulo del kernel está completa y listo para el próximo paso, que es la compilación del módulo.

10.3. Solución paso 3: Compilación del módulo del kernel

Para compilar el módulo del kernel, se abre una terminal. Luego, se navega al directorio donde se han guardado los archivos `cpu_temp_module.c` y `Makefile`. Una vez en el directorio correcto, se ejecuta el comando `make`. Este comando genera el archivo `cpu_temp_module.ko`, que es el módulo del kernel compilado. Para confirmar que la compilación fue exitosa, se puede tomar una captura de pantalla de la terminal después de ejecutar el comando y guardarla como una imagen (por ejemplo, `paso3.png`).



```

ubuntu@ubuntu: ~/Downloads
ubuntu@ubuntu:~/Downloads$ make
make -C /lib/modules/5.19.0-32-generic/build M=/home/ubuntu/Downloads modules
make[1]: Entering directory '/usr/src/linux-headers-5.19.0-32-generic'
warning: the compiler differs from the one used to build the kernel
The kernel was built by: x86_64-linux-gnu-gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0
You are using: gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0
CC [M] /home/ubuntu/Downloads/cpu_temp_module.o
MODPOST /home/ubuntu/Downloads/Module.symvers
CC [M] /home/ubuntu/Downloads/cpu_temp_module.mod.o
LD [M] /home/ubuntu/Downloads/cpu_temp_module.ko
BTF [M] /home/ubuntu/Downloads/cpu_temp_module.ko
Skipping BTF generation for /home/ubuntu/Downloads/cpu_temp_module.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-5.19.0-32-generic'

```

Figura 10: Compilación del módulo del kernel.

10.3.1. La carpeta después de compilar

La carpeta después de compilar quedará tal cual después de compilar:

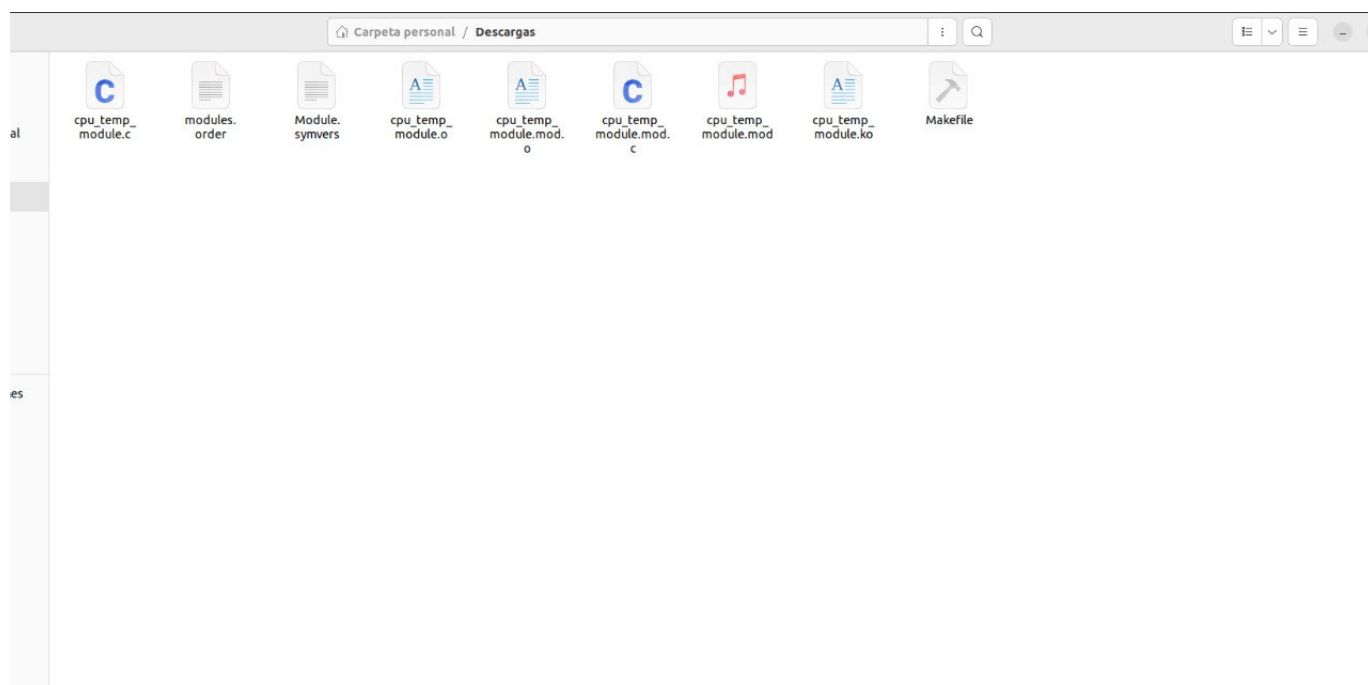
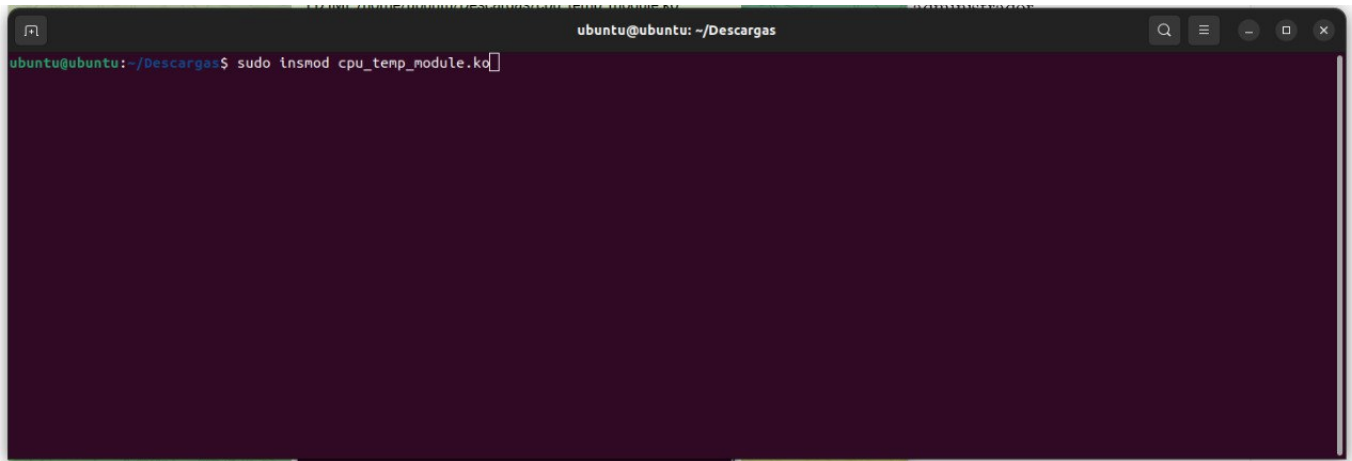


Figura 11: Carpeta descargas después de compilar.

10.4. Solución paso 4: Cargar el módulo del kernel

En el Paso 4, se procede a cargar el módulo del kernel en el sistema con el comando dado:

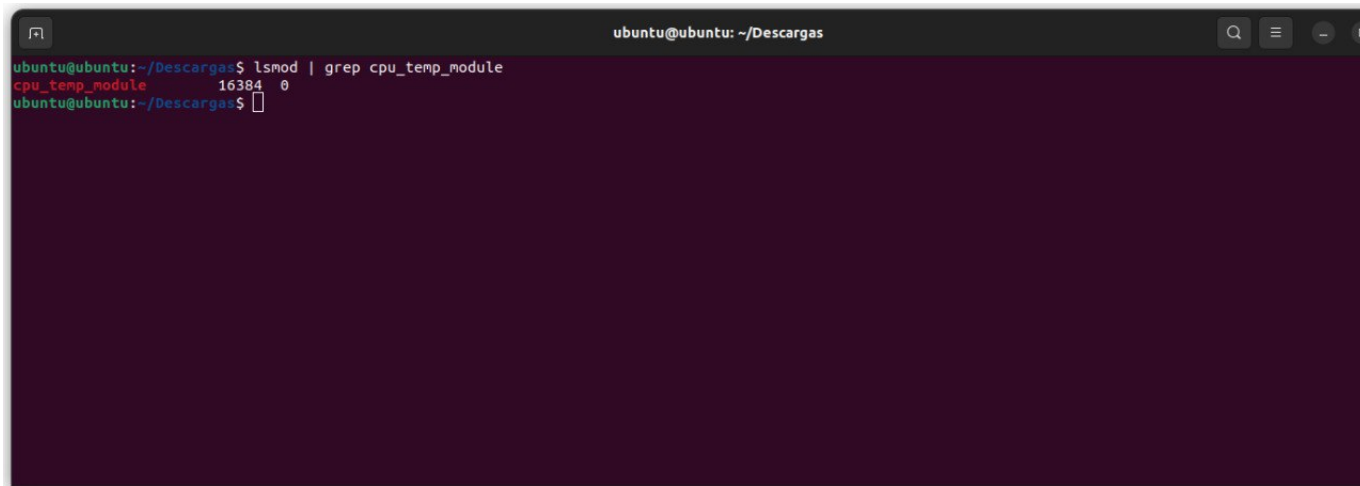
A terminal window titled 'ubuntu@ubuntu: ~/Descargas' with a search icon, menu icon, and window controls. The prompt is 'ubuntu@ubuntu:~/Descargas\$' and the command 'sudo insmod cpu_temp_module.ko' is entered at the end of the line.

```
ubuntu@ubuntu:~/Descargas$ sudo insmod cpu_temp_module.ko
```

Figura 12: Carga del módulo del kernel.

10.5. Solución paso 5: Verificar la carga del módulo

Después de ejecutar el comando del paso anterior, se verifica que el módulo se haya cargado correctamente:

A terminal window titled 'ubuntu@ubuntu: ~/Descargas' with a search icon, menu icon, and window controls. The prompt is 'ubuntu@ubuntu:~/Descargas\$' and the command 'lsmod | grep cpu_temp_module' is entered. The output shows 'cpu_temp_module' in red, followed by '16384' and '0' in white.

```
ubuntu@ubuntu:~/Descargas$ lsmod | grep cpu_temp_module
cpu_temp_module      16384  0
ubuntu@ubuntu:~/Descargas$
```

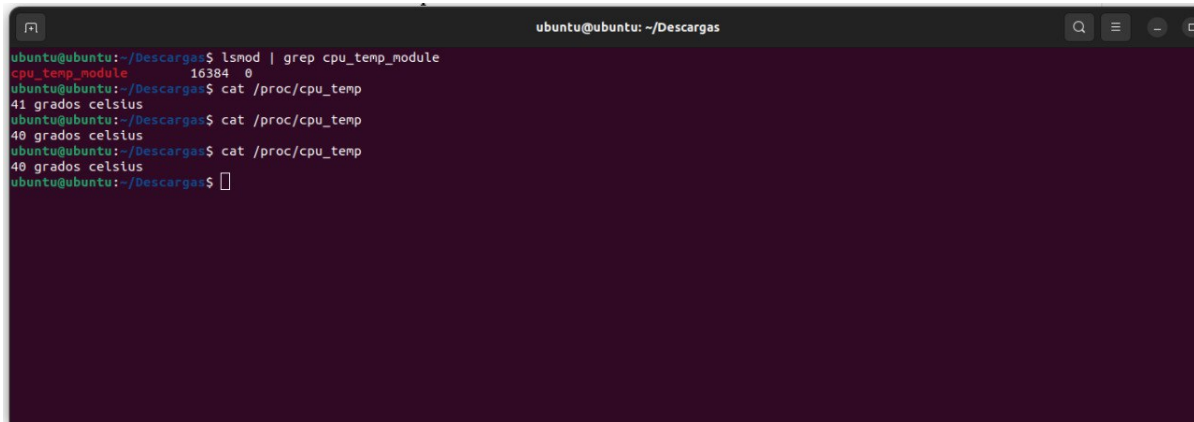
Figura 13: Verificación de la carga del módulo.

Este comando debería mostrar una línea que contenga el nombre del módulo `cpu_temp_module` junto con otra información relevante, lo que confirma que el módulo se ha cargado correctamente.

10.6. Solución paso 6: Leer la temperatura de la CPU

Finalmente, se puede verificar la funcionalidad del módulo consultando la temperatura de la CPU. Para hacerlo, se accede al archivo creado en el sistema de archivos proc con el comando dado

El resultado de este comando debería ser la temperatura actual de la CPU en grados Celsius:

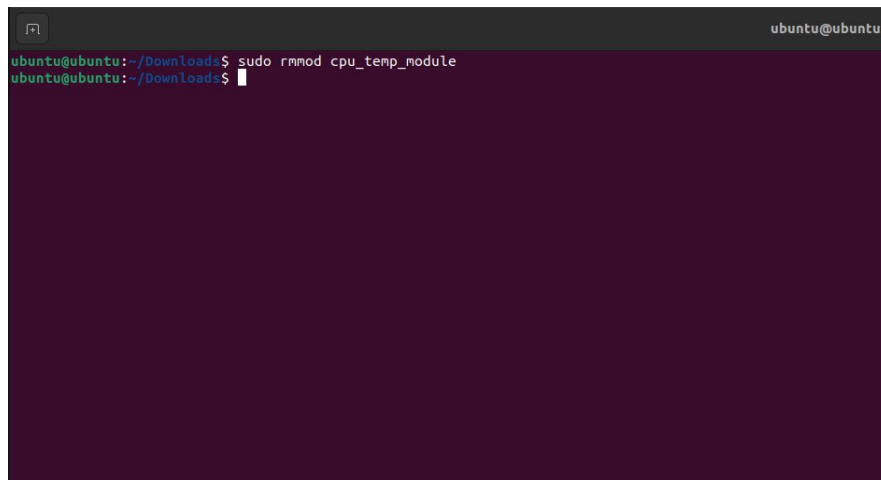
A terminal window titled 'ubuntu@ubuntu: ~/Descargas' showing the following commands and output:

```
ubuntu@ubuntu:~/Descargas$ lsmod | grep cpu_temp_module
cpu_temp_module      16384  0
ubuntu@ubuntu:~/Descargas$ cat /proc/cpu_temp
41 grados celsius
ubuntu@ubuntu:~/Descargas$ cat /proc/cpu_temp
40 grados celsius
ubuntu@ubuntu:~/Descargas$ cat /proc/cpu_temp
40 grados celsius
ubuntu@ubuntu:~/Descargas$
```

Figura 14: Leer la temperatura de la CPU desde el fichero `cpu_temp`.

10.7. Solución paso 7: Descargar el módulo del kernel

Se abre la terminal y se pone el comando dado en terminal consiguiendo con esto descargar el módulo del sistema.

A terminal window titled 'ubuntu@ubuntu: ~' showing the following command and prompt:

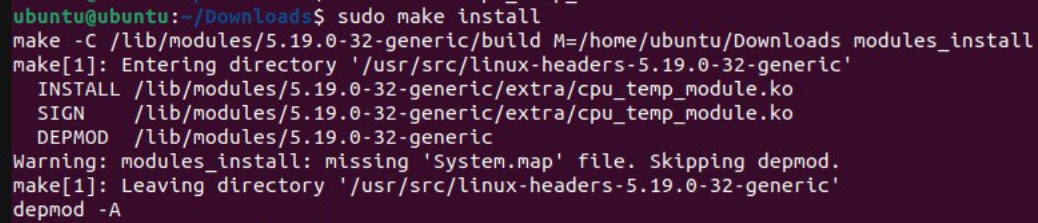
```
ubuntu@ubuntu:~/Downloads$ sudo rmmod cpu_temp_module
ubuntu@ubuntu:~/Downloads$
```

Figura 15: Descarga del módulo del kernel

10.8. Solución paso 8: Instalar el módulo del kernel y cargarlo en el sistema para siempre

10.8.1. Instalación del módulo

Se abre un terminal y se ejecuta el comando indicado en el paso 8 de la guía para instalar el módulo del kernel.

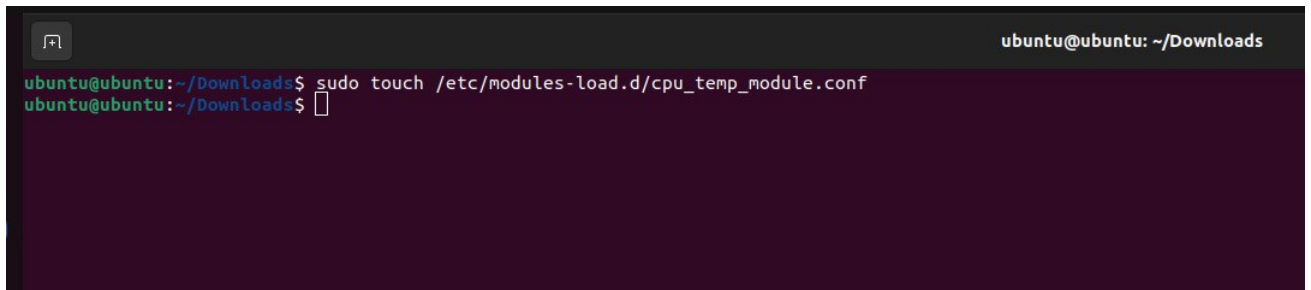
A terminal window with a dark background and light-colored text. The prompt is 'ubuntu@ubuntu:~/Downloads\$'. The command 'sudo make install' has been executed. The output shows 'make' running in the directory '/lib/modules/5.19.0-32-generic/build' with 'M=/home/ubuntu/Downloads/modules_install'. It then enters the directory '/usr/src/linux-headers-5.19.0-32-generic' and performs 'INSTALL' of '/lib/modules/5.19.0-32-generic/extra/cpu_temp_module.ko', 'SIGN' of the same file, and 'DEPMOD' of '/lib/modules/5.19.0-32-generic'. A warning message states: 'Warning: modules_install: missing 'System.map' file. Skipping depmod.' The process then leaves the directory and runs 'depmod -A'.

```
ubuntu@ubuntu:~/Downloads$ sudo make install
make -C /lib/modules/5.19.0-32-generic/build M=/home/ubuntu/Downloads modules_install
make[1]: Entering directory '/usr/src/linux-headers-5.19.0-32-generic'
  INSTALL /lib/modules/5.19.0-32-generic/extra/cpu_temp_module.ko
   SIGN   /lib/modules/5.19.0-32-generic/extra/cpu_temp_module.ko
  DEPMOD  /lib/modules/5.19.0-32-generic
Warning: modules_install: missing 'System.map' file. Skipping depmod.
make[1]: Leaving directory '/usr/src/linux-headers-5.19.0-32-generic'
depmod -A
```

Figura 16: Instalar el módulo del kernel

10.8.2. Configuración de la carga automática del módulo

- I. Se abre una terminal.
- II. Para seguir con los pasos se necesita crear un nuevo archivo de configuración. Para ello pondremos el comando dado en terminal:

A terminal window with a dark background. The prompt is 'ubuntu@ubuntu: ~/Downloads'. The command 'sudo touch /etc/modules-load.d/cpu_temp_module.conf' has been entered and executed. The prompt is now 'ubuntu@ubuntu: ~/Downloads\$' with a cursor.

```
ubuntu@ubuntu: ~/Downloads$ sudo touch /etc/modules-load.d/cpu_temp_module.conf
ubuntu@ubuntu: ~/Downloads$
```

Figura 17: Creación del archivo de configuración.

- III. Se edita el archivo de configuración en este caso con **nano** como en la guía y agregamos la línea dada por la guía y guardamos el fichero:

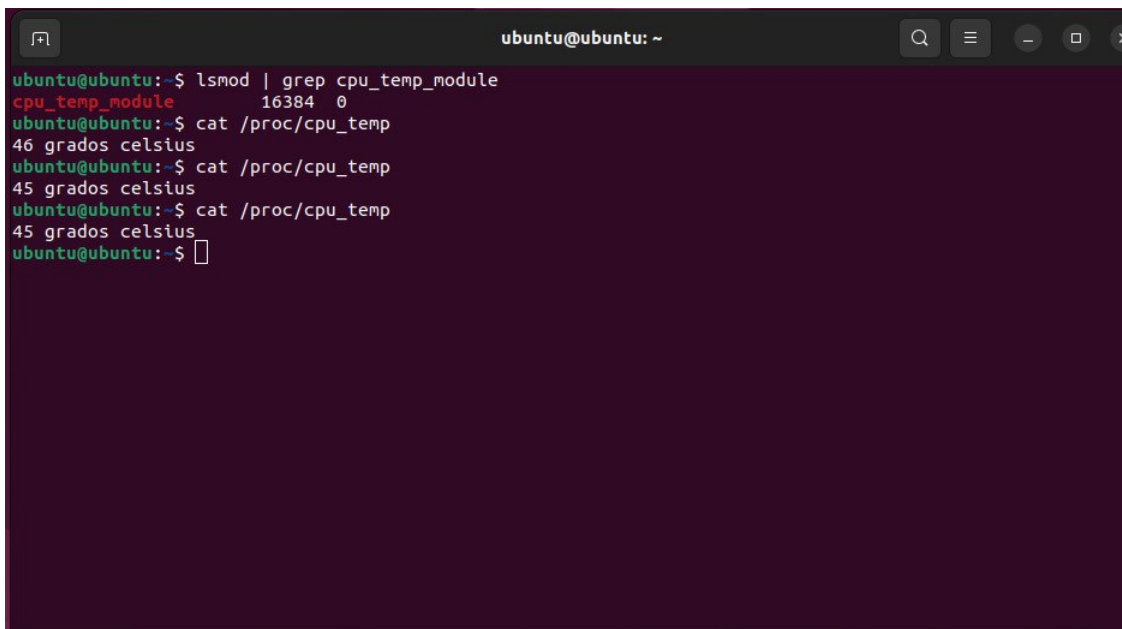
A terminal window showing the nano text editor. The top bar indicates 'GNU nano 6.2' and the file path '/etc/modules-load.d/cpu_temp_module.conf'. The first line of the file is 'cpu_temp_module'. The cursor is at the end of the first line.

```
GNU nano 6.2 /etc/modules-load.d/cpu_temp_module.conf
cpu_temp_module

```

Figura 18: Añadido de la línea al archivo de configuración.

iv. Reiniciamos el sistema y vemos que en este caso todo ha ido bien:

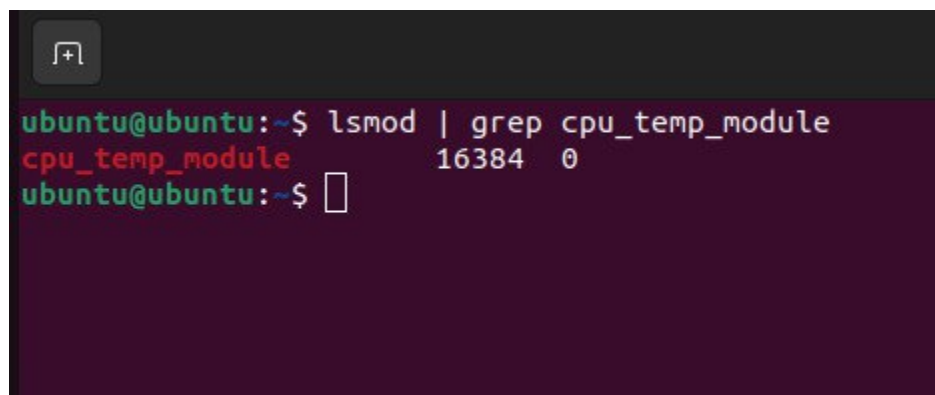


```
ubuntu@ubuntu: ~  
ubuntu@ubuntu:~$ lsmod | grep cpu_temp_module  
cpu_temp_module      16384  0  
ubuntu@ubuntu:~$ cat /proc/cpu_temp  
46 grados celsius  
ubuntu@ubuntu:~$ cat /proc/cpu_temp  
45 grados celsius  
ubuntu@ubuntu:~$ cat /proc/cpu_temp  
45 grados celsius  
ubuntu@ubuntu:~$
```

Figura 19: Instalación correcta del módulo cada vez que se inicia el sistema.

10.9. Solución paso 9: Comprobar la instalación del módulo

Simplemente introducimos el comando dado en terminal:



```
ubuntu@ubuntu:~$ lsmod | grep cpu_temp_module  
cpu_temp_module      16384  0  
ubuntu@ubuntu:~$
```

Figura 20: Comprobación de la instalación del módulo.

Y se puede observar que el modulo está bien cargado e instalado.

11. Bibliografía

1. [Sistema de ficheros proc.](#)
2. [Kernel linux con makefiles.](#)
3. ["The Linux Kernel Module Programming Guide"](#)