PRÁCTICA 4 – SISTEMAS EMPOTRADOS

Síntesis de circuitos en Verilog

Lucas Serrano Jiménez

César San Blas Leal

tarea 1

**1. Busca información sobre los principales fabricantes de FPGAs (Xilinx/AMD, Altera/Intel, Microchip y Lattice) y compara las arquitecturas de cada uno de ellos destacando las diferencias en sus arquitecturas (celdas básicas, FPGAs SRAM vs FPGAs flash), ventajas e inconvenientes de cada una.**

**2. Las FPGAs son dispositivos flexibles en los que se puede implementar lógica digital compleja como microcontroladores, puertos Ethernet, etc. Explica qué es una “soft macro” o “soft IP core” y qué es un “hard macro” o “hard IP core”.**

**3. Actualmente hay una interesante comunidad de desarrollo entorno a RISC-V. Busca información sobre este proyecto y explica cómo se están implementando en FPGAs. ¿Se puede implementar con soft y con hard IP cores?**

**4. Los SoCs incorporan módulos diversos como, por ejemplo, microcontroladores (ARM, RISC-V, PowerPC). Busca información sobre las arquitecturas más modernas tipo SoC (System on Chip) identificando su arquitectura, tipo de módulos y periféricos que incorporan.**

tarea 2

**1. Carga el ejemplo *leds* e interpreta el fichero de formas de onda VCD generado por la simulación anterior.**

Este ejemplo, tal y como se explicó durante la clase, simplemente carga en los pines de 4 leds un estado de nivel alto. Al ejecutar la simulación, como era de esperar, muestra un gráfico con un estado on durante todo el tiempo que dura el programa.

A screenshot of a computer

Description automatically generated

**2.1 Explica el funcionamiento de LED\_Blink.v, detallando qué se hace en cada línea de código. ¿Por qué exactamente vemos parpadear el LED con una frecuencia de más o menos 1 segundo?**

PERO NO PARPADEA CADA SEGUNDO NO? ESTE ES EL DE 12 BITS (¿) RARILLO ESTE APARTADO

El led usado en el programa *LED\_Blink.v* toma como entrada la señal de reloj *CLK* y asigna la salida *LED*.

Creo que es porque toma el bit más significativo y ps lo que tarde en llegar al incrementar los 24 bits (MIRAR)

**2.2 Ahora vamos a sintetizar este código y a cargarlo en nuestra placa.**

Tras sintetizar los códigos de LED\_blink y cargarlos en la FPGA se observa el LED encendido constantemente, como se muestra en el vídeo *----------*. Este comportamiento se debe a que el LED tendrá un periodo de encendido y apagado de 212 ciclos en un reloj de 12 MHz por lo que resulta en un tiempo de aproximadamente 5 ns, inapreciable al ojo humano.

**2.3. Sabiendo cómo funciona el código del ejercicio anterior, modifica el código para realizar un contador usando todos los LEDs de su tarjeta FPGA. Se debe poder apreciar el apagado y encendido de los LEDs.**

En este caso, para poder apreciar el apagado y encendido de los LEDS, aumentaremos el número de bits del registro y asignaremos los últimos a los 8 LEDS que posee la FPGA. En este caso se ha aumentado a 24 el número de bits del contador, por lo que ahora el periodo del cambio de estados es de 224 ciclos. Asignando cada uno de los últimos 8 bits del registro a los LEDS de la placa se obtiene una escalera de frecuencias, esta vez apreciable al tratarse de un periodo considerablemente mayor. El comportamiento de este LED se muestra en el siguiente vídeo ------------ .

tarea 3

**3.1 Crea un proyecto de simulación que incluya los ficheros creados y simula tu módulo. Captura las formas de onda y explica qué sucede en la simulación.**

Con la escala adecuada, se puede observar que la señal varía y cambia de estado. Sin embargo, como se explicó en la tarea anterior, a pesar de que el LED sí parpadea, lo hace a una frecuencia tan rápida que es inapreciable.

AÑADIR FOTO

**3.2 Realiza un testbench en Verilog para simular el diseño del apartado 2.3 y aporta las formas de onda.**

En este caso al asignar cada LED a un bit del registro, se puede apreciar la escalera de frecuencias que generan las señales. Debido a la naturaleza binaria del registro usado para controlar el estado de los LEDs, cada uno tendrá el doble de periodo que el anterior.

AÑADIR FOTO

BITÁCORA CUTRE: 27-11

En clase hemos terminado la Tarea 2 y realizado entera la 3.

**4. ¿Qué función tiene el modificador de OpenMP dynamic en el código?**

Establece que las iteraciones se asignan a medida que los hilos terminan su trabajo anterior, lo que ayuda a equilibrar la carga de trabajo de manera más eficiente si algunas iteraciones toman más tiempo que otras y a evitar que hayan hilos inactivos.

**5. Investigue qué pasa si no declara como privadas las variables *i* y *tid*.**

Si la variable *i* no fuese privada, se modificaría simultáneamente por todos los hilos que estén trabajando con ella por lo que el bucle for no funcionaría correctamente. En el caso de *tid,* si dentro del hilo el valor pudiera ser alterado mientras se ejecuta, al mostrar por pantalla el resultado de la suma el identificador del hilo sería erróneo.

RETO JEDI 1

A screen shot of a computer program

Description automatically generatedRepita la tarea 1.2 empleando un planificador estático y no dinámico. Puede hacerlo “a mano” o usando algún modificador de OpenMP.

El programa difiere respecto al código del guón de la práctica en que la asignación de iteraciones del bucle for a los hilos se hace de manera estática en lugar de dinámica. Esto se consigue sustituyendo el argumento *dynamic* de *schedule* por *static,* lo que implica que cada hilo se encarga exclusivamente del número de iteraciones establecido en la variable chunk. A diferencia del modo dinámico, donde las iteraciones se asignaban a medida que los hilos iban terminando su trabajo, para este modo cada hilo realizará únicamente el número de iteraciones que dicte chunk. En este caso la variable vale 10, por lo que cada hilo realizará 10 iteraciones, como se muestra en la figura.

tarea 1.3

**Diseñe un programa en C/C++ para multiplicar dos matrices cuadradas con elementos de tipo double (punto flotante de 64 bits) entre 0 y 1. Después multiplique la matriz resultante por otra matriz de enteros entre 0 y 255 elemento a elemento.**

El código implementado se encuentra en el archivo *tarea1\_3.cpp*.

**Muestre en una serie de gráficas cómo varía el tiempo paralelo de ejecución en función de algún parámetro que modifique la carga computacional del problema. Luego, paralelice el código con OpenMP y tome los siguientes tiempos de ejecución: código secuencial original (sin paralelizar), código paralelo ejecutado en mono mono-hebra y código con dos o más hebras. Apunte las ganancias en velocidad que se observan.**

En este caso se ha empleado N, que define las dimensiones de las matrices cuadradas (NxN) usadas en la operación, como parámetro para observar la variación del tiempo de ejecución y el comportamiento según el tipo de paralelismo. Puesto que el código desarrollado contiene directivas de OpenMP para el paralelismo, la compilación requiere del argumento -fopenmap, como ya se advirtió en la tarea 1.1. En este ejercicio se ha ido incrementando el valor de N de 10 en 10 hasta llegar a 600, obteniendo un número significativo de muestras, y midiendo el tiempo, la ganancia y la eficiencia de funciones con paralelismo monohebra, bihebra y multihebra (cuatro en este caso). A continuación se muestran los resultados obtenidos.

A group of text on a screen

Description automatically generated with medium confidence

Los resultados obtenidos, para graficarlos posteriormente, se han guardado en el fichero *tarea1\_3.time* empleado una tubería*.* Después, con el código *tarea1\_3.gpi* implementado y mediante la herramienta Gnuplot se han graficado estos valores.



La primera gráfica recoge los tiempos medidos para cada caso de ejecución. Como es de esperar, la ejecución con 4 hebras presenta un tiempo de ejecución menor que en el resto de casos. Además, la ejecución secuencia y la monohebra no muestra diferencia alguna, indicando que el procedimiento es el mismo.

A graph of different colored lines

Description automatically generated

La siguiente gráfica muestra la ganancia obtenida por parte de los diferentes modos de paralelización respecto a la función secuencial. Como se observó en la anterior gráfica, el modo monohebra no encuentra mejora respecto al secuencial, por lo que su ganancia se encuentra en valores próximos al 1. Por otro lado, la función con más hebras muestra una ganancia superior al resto, proporcional a la mejora en el tiempo de ejecución.

A graph of different colored lines

Description automatically generated

La última gráfica presenta los resultados de la eficiencia en función del número de hebras. En este caso, a diferencia de las dos gráficas anteriores, la paralelización que domina es la de una hebra (equivalente a secuencial). Destaca la poca eficiencia que presenta el modelo de paralelización de cuatro hebras. Esto indica que la ganancia en velocidad no es proporcional al número de hebras.

A graph of different colored lines

Description automatically generated

RETO JEDI 2

Si la matriz B es transpuesta, el código que produce el compilador para multiplicar las dos matrices A y B producirá menos fallos de caché, dado que maximizamos la localidad espacial de los datos en la cachés. Intente demostrar y medir si este hecho afecta a su problema. Nota: puede que tenga que escalar el problema para que el efecto sea apreciable.

Se ha añadido una nueva función *operacionMatrizTraspuesta,* que realiza la misma operación requerida en la tarea anterior pero trasponiendo la matriz B antes de realizar la multiplicación. Los resultados temporales se muestran en la siguiente gráfica donde la línea lila representa el código base secuencial y el verde el que añade la matriz traspuesta. A pesar de haber escalado considerablemente el tamaño del problema (nótese que se ha ampliado hasta matrices de 800x800), no se detecta ninguna diferencia con respecto al programa que no traspone la matriz.

A graph of a number of points

Description automatically generated with medium confidence

tarea 2.1

**Identifique y muestre algunas características de las interfaces de red de las que dispone la tarjeta (apunte el tipo de interfaz, nombre en el sistema, MAC, ...):**

Texto

Descripción generada automáticamente

El sistema muestra información de las tres interfaces de red que tiene el sistema:

* enp3s0: Corresponde a la interfaz de conexión Ethernet, el 0 del final de su nombre indica que es la primera. Si el equipo tuviera más serían *enp3s1, enp3s2…*
* lo: Corresponde a la interfaz *loopback*; es decir, la interfaz mediante la que el sistema se comunica consigo mismo.
* wlp2s0: Corresponde a la interfaz de conexión inalámbrica. Antiguamente se llamaba *wlan* y sigue el mismo criterio de numeración que la interfaz de Ethernet, de modo que las demás serían *wlp2s1, wlp2s2…*

Además, el comando *ifconfig* proporciona información extra de cada interfaz como sus direcciones IPv6, por ejemplo.

**1. ¿Qué es la MAC? ¿Para qué sirve?**

MAC es una dirección única de cada tarjeta de red asignada por su fabricante. Consiste en una secuencia de 48 bit, representados normalmente en hexadecimal. Un dispositivo puede tener varias direcciones MAC siempre que tenga varias tarjetas de red.

Esta información es muy útil a la hora de gestionar conexiones a redes ya que facilitan la identificación de cada dispositivo conectado, pudiendo permitir o denegar su acceso.

**2. ¿Qué es y para qué sirve el parámetro MTU de cada interfaz de red? ¿Se puede**

**cambiar? ¿Cómo nos afectaría?**

El parámetro MTU (por sus siglas en inglés, *Maximum Transfer Unit*) indica el tamaño máximo de paquete que se puede transmitir por esa interfaz. Se puede cambiar de manera temporal como indica la siguiente imagen:

Texto

Descripción generada automáticamente

Se muestra que el MTU de la interfaz enp3s0 comienza en 1500 (su valor por defecto) y se cambia a 1000 manualmente. Sin embargo, después de reiniciar el equipo volvería a su valor por defecto de nuevo. Existe una manera de cambiarlo de forma permanente modificando archivos del sistema.

Este valor puede afectar al sistema de diferentes maneras. Si tiene un tamaño muy pequeño, la información debe fragmentarse y puede llegar a perderse parte de ella (por ejemplo, si la comunicación es mediante protocolos que no permiten esa fragmentación). Además, aumentaría notablemente los tiempos de espera ya que no podría aceptar toda la información al mismo tiempo.

tarea 2.2

**(a) ¿Qué es el propio directorio /proc? ¿Qué información nos da? ¿Quién nos la da?**

Es un directorio que contiene información del sistema como los procesos que están ocurriendo o la configuración del mismo. Los ficheros que dan esta información son virtuales; es decir, no ocupan espacio en disco y el sistema los crea temporalmente mientras el usuario los consulte.

**(b) Características completas del microprocesador del que dispone según la información que encuentre en /proc. Elabore además una lista con todos los conjuntos de extensiones al repertorio ISA base de su microprocesador y encuentre qué tipo de instrucciones aportan (defina qué hacen estos conjuntos de instrucciones). ¿Dispone de extensiones vectoriales? ¿Cuáles? ¿De cuántos cores dispone?**

Captura de pantalla de un celular

Descripción generada automáticamente Esta imagen muestra que el equipo tiene 3 procesadores, /proc proporciona esta información al respecto:

Captura de pantalla de un celular

Descripción generada automáticamente Captura de pantalla de un celular

Descripción generada automáticamente Captura de pantalla de un celular

Descripción generada automáticamente

**(c) ¿Puede realizar su procesador la función de un CRC de 32 bits (código de redundancia cíclico) por hardware? ¿Puede encontrar algún ejemplo de, en su caso, dicha instrucción o instrucciones?**

Sí se puede, existe un conjunto de instrucciones que le da esa capacidad.

**(d) Tiempo exacto que lleva encendida la tarjeta/equipo**

**Pantalla negra con letras blancas

Descripción generada automáticamente con confianza media** El primer número corresponde a los segundos que el sistema lleva encendido (1 hora, 21 minutos y 48 segundos aproximadamente) y el segundo número son los segundos de inactividad que ha registrado el sistema acumulado por cada CPU (4 horas, 45 minutos y 14 segundos en total; 1 hora, 11 minutos y 18 segundos por CPU).

**(e) Versión de Linux y versión del compilador con la que fue compilado.**

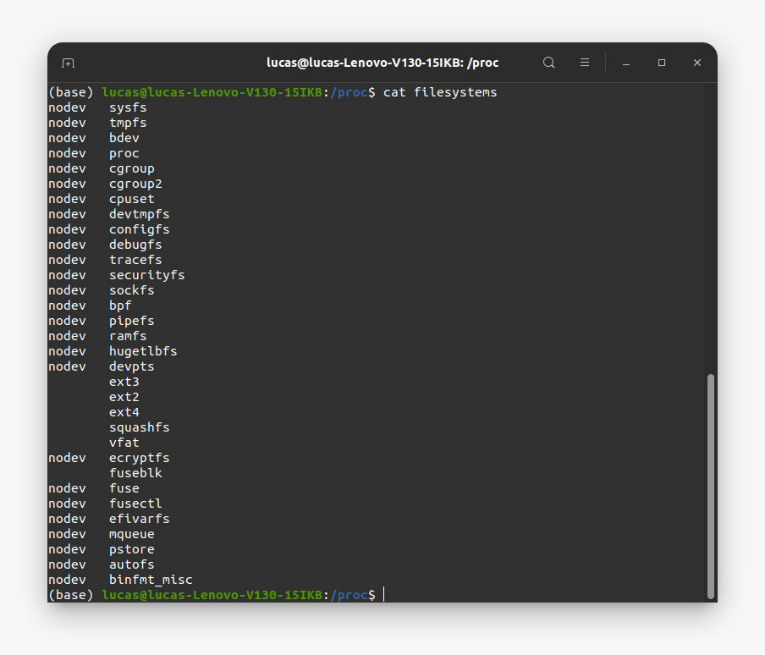
Pantalla de computadora

Descripción generada automáticamente con confianza media

**(f) ¿Qué sistemas de archivos (filesystems) se soportan? Comente las características de, al**

**menos, 5 de ellos.**

Soporta los siguientes sistemas:



**Ext4:** Sistema de archivos estándar de Linux. Es un sistema de archivos que mantiene un registro de todos los cambios realizados en los archivos, lo que permite deshacerlos en caso de fallo o error.

**Sysfs:** Sistema de archivos virtual que contiene información sobre el hardware del sistema. Es el sistema del directorio /proc.

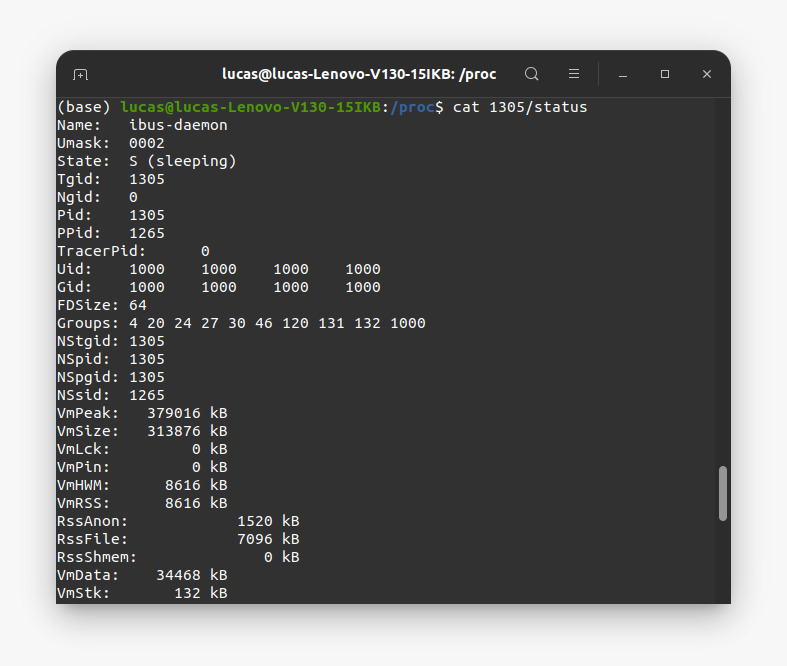
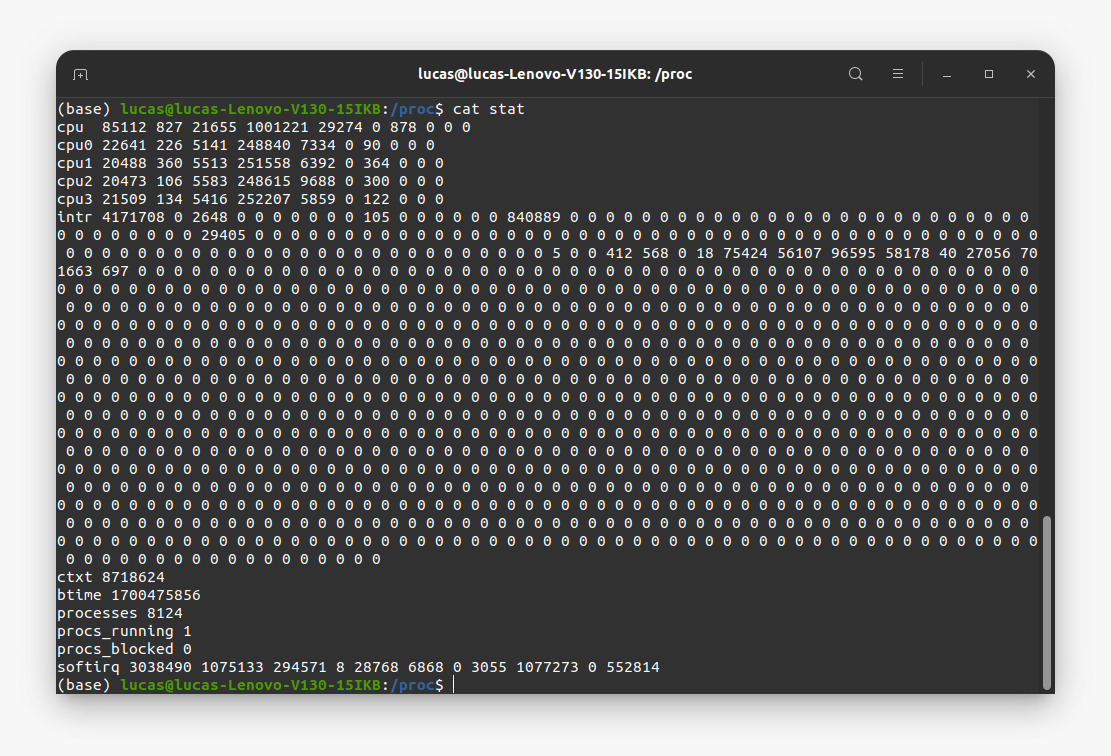
**Tmpfs:** Sistema de archivos temporal que se almacena en la memoria.

**Configfs:** Sistema de archivos virtual que se utiliza para almacenar la configuracióndel sistema. Incluye datos sobre la configuración del hardware, la configuración del sistema operativo…

**Pipefs:** Sistema de archivos virtual que se utiliza para crear tuberías entre procesos.

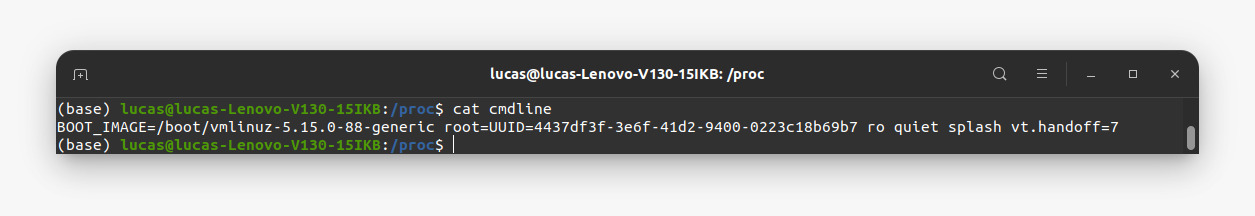
**(g) ¿Qué muestra el fichero /proc/stat ? ¿Hay alguna orden que de la misma información pero de forma más legible?**

Muestra el estado de los procesos desde el último arranque del sistema. Cada proceso tiene una carpeta propia que muestra su información de una manera más clara.



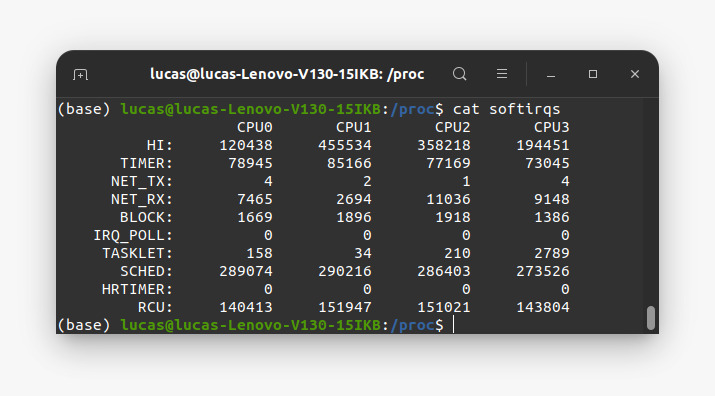
**(h) ¿Qué muestra el fichero /proc/cmdline ?**

Indica los parámetros que recibió el kernel en el momento de su arranque.



**(i) ¿Qué muestra el fichero /proc/softirqs ?**

Muestra información sobre los diferentes procesadores del sistema.

****

**(j) ¿Qué hay en los directorios dentro /proc que empiezan con un número? ¿Qué información se muestra? Ponga algún ejemplo representativo.**

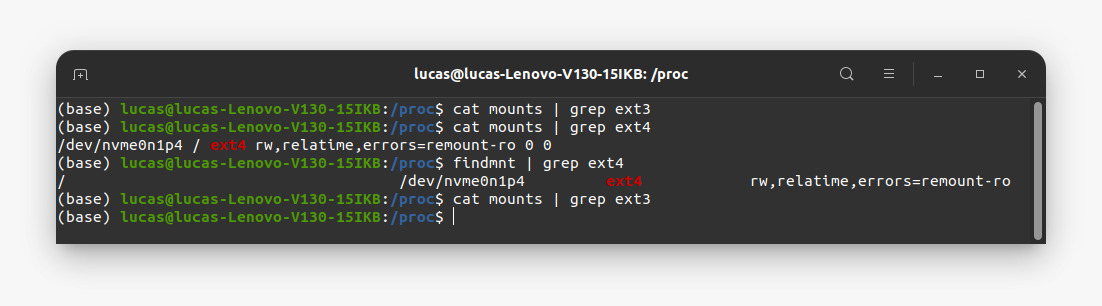
Son procesos en ejecución, cada uno tiene un directorio interno que muestra detalles sobre el mismo. Como se ha visto en la tarea g:

Captura de pantalla de computadora

Descripción generada automáticamente

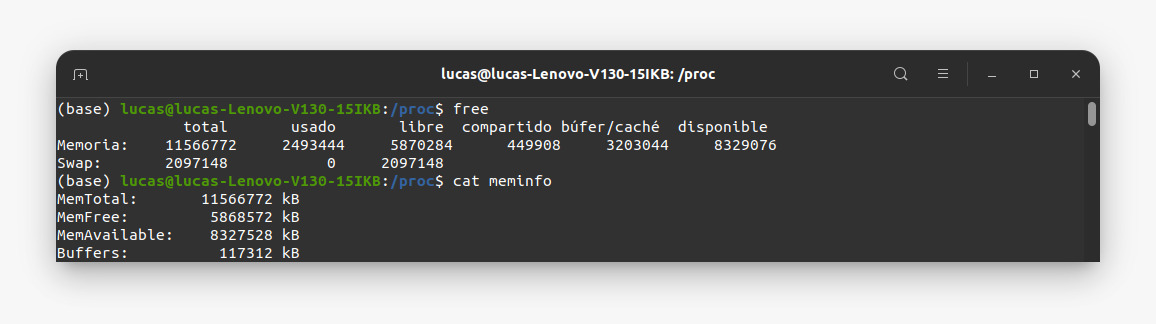
**(k) ¿Cuántos sistemas de archivos de tipo ext3 o ext4 están montados? ¿Dónde? ¿Con qué archivo u orden puede acceder a esta información?**

Del tipo ext3 no hay ninguno y del tipo ext4 hay uno solo. Se encuentra en *dev/nvme0n1p4*. Se ha comprobado esta información con los dos métodos que se muestran en la imagen:

****

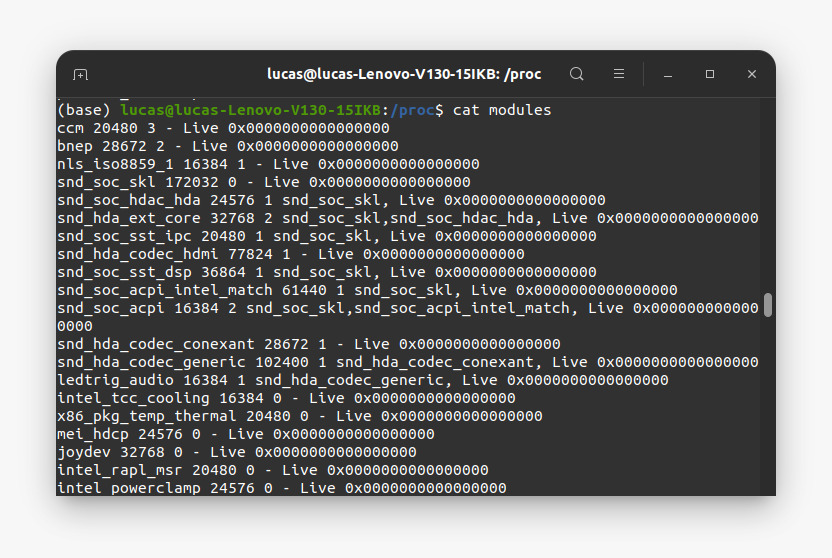
**(l) ¿Cuánta memoria tiene libre? Compruebe que encuentra la misma información que la orden free**

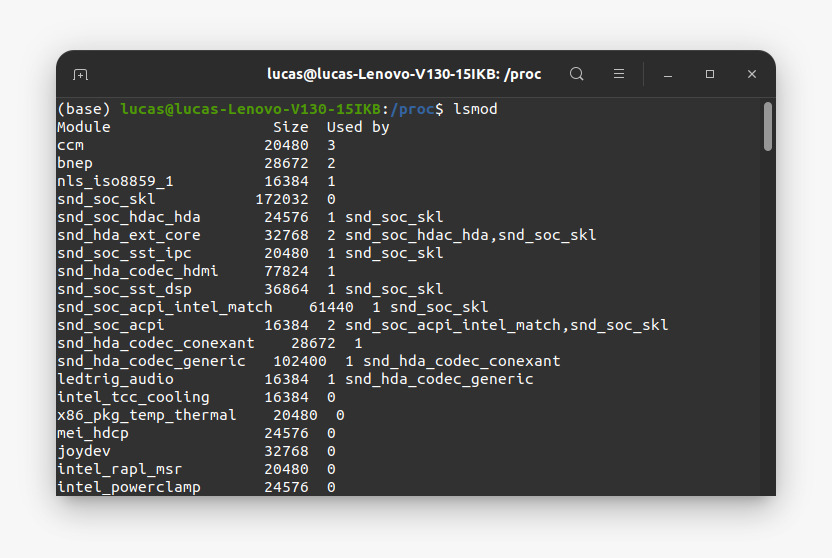
Mediante meminfo se imprime información sobre la memoria del sistema y, efectivamente, coincide con el comando free; aunque no al 100% ya que puede variar con cada ejecución del comando.

****

**(m) ¿Cuáles son los módulos que tiene instalado el kernel? ¿Qué orden (comando) puede acceder a esta misma información de forma más legible?**

Mediante el archivo modules se puede comprobar los módulos instalados en el kernel; sin embargo, la orden lsmod devuelve la información de ese mismo fichero de manera más legible.





**(n) ¿Qué módulos usan el módulo de bluetooth?**

Hay que filtrar la información dada por lsmod. Se comprueba que el módulo de bluetooth es empleado por los módulos: btrtl, btinel, btbcm, bnep y btusb.

