

# PRÁCTICA 3 – SISTEMAS EMPOTRADOS

## Entrenamiento con ARM: OpenMP y OpenCV

Lucas Serrano Jiménez

César San Blas Leal

### TAREA 1.1

Se deberá estudiar el API de OpenMP y su uso con GNU GCC (gcc, g++), comprobando el correcto funcionamiento de algunos de los ejemplos que hay disponibles en Internet.

La API de OpenMP es una biblioteca de programación que permite a los programadores paralelizar sus aplicaciones (basados en C, C++ o Fortran) de forma explícita; es decir, el programador debe especificar qué partes del código deben ejecutarse en paralelo. El compilador y el sistema de ejecución se encargarán de distribuir el trabajo entre los distintos procesadores.

Esta API no realiza comprobaciones de ningún tipo en el programa, por lo tanto es responsabilidad del programador asegurarse de que la aplicación sea correcta y segura.

GNU GCC es un compilador de código abierto que soporta OpenMP. Para compilar un programa que utiliza OpenMP en GCC, se debe utilizar la opción `-fopenmp`.

Algunas de las directivas y cláusulas más comunes de OpenMP son las siguientes:

- **Directiva parallel:** Declara una región de código que se ejecutará en paralelo.
- **Directiva for:** Declara un bucle que se ejecutará en paralelo.
- **Cláusula private:** Declara que una variable solo será visible para el hilo que la creó.
- **Cláusula shared:** Declara que una variable será visible para todos los hilos.
- **Cláusula reduction:** Declara que una variable será compartida por todos los hilos y que su valor será la suma de los valores individuales.

A continuación, un sencillo programa de ejemplo que muestra el uso de algunas de estas directivas y cláusulas:

```
#include <stdio.h>
#include <omp.h>
```

```
int main() {
    int i, n = 1000;
    float sum = 0.0f;
```

```
    // Sección en paralelo
    #pragma omp parallel
    {
```

```
        // sum` será visible para todos los hilos
        #pragma omp for
```

```

        for (i = 0; i < n; i++) {
            sum += i;
        }
    }

    printf("sum = %f\n", sum);

    return 0;
}

```

Cuyo resultado muestra el valor final de *sum* como suma de todos los bucles realizados en paralelo, aunque cada ejecución termina con un valor distinto. Esto se debe a que cada hilo tiene su propia variable *sum* y al final se juntan todas, produciendo variaciones porque no hay un orden establecido. Si se declarase *sum* como *shared*, escribiendo *#pragma omp for reduction(+:sum)* antes del *for* compartido, el resultado se estabiliza en: *sum = 499500.00*.

## TAREA 1.2

### 1. Explique qué hace el código que se añade a continuación.

El código expuesto en el guión de la práctica suma los valores de los arrays *a* y *b*, previamente inicializados secuencialmente de 0 a *N*-1. Lo característico del código es que utiliza OpenMP para ejecutar las sumas en paralelo, distribuyendo la carga de trabajo entre varios hilos.

### 2. Delimite las distintas regiones OpenMP en las que se está expresando paralelismo.

Las regiones donde se expresa paralelismo en el código implementado son:

- *#pragma omp parallel shared(a, b, c, nthreads, chunk) private(i, tid)*

Esta línea comienza la región paralela con OpenMP. En ella se establecen las variables que se mantendrán inalteradas entre hilos (*i* y *tid*) y las que se compartirán (*a*, *b*, *c*, *nthreads* y *chunk*).

- *#pragma omp for schedule(dynamic,chunk)*

Esta directiva divide el bucle *for* en tareas que se ejecutarán en paralelo entre los hilos.

### 3. Explique con detalle qué hace el modificador de OpenMP *schedule*. ¿Cuáles pueden ser sus argumentos? ¿Qué función tiene la variable *chunk* en el código? ¿A qué afecta?

El operador *shedule* sirve para configurar la distribución de las iteraciones del bucle *for* entre los hilos paralelos.

El valor de *chunk* sirve para especificar cuántas iteraciones se asignan a cada hilo. En este caso, este valor viene definido por la constante *CHUNKSIZE* que vale 10, por lo que por cada bloque atenderá a 10 iteraciones.

### 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

Repita 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.

```
Thread 7 starting...
Thread 7: c[70]= 140.000000
Thread 7: c[71]= 142.000000
Thread 7: c[72]= 144.000000
Thread 7: c[73]= 146.000000
Thread 7: c[74]= 148.000000
Thread 7: c[75]= 150.000000
Thread 7: c[76]= 152.000000
Thread 7: c[77]= 154.000000
Thread 7: c[78]= 156.000000
Thread 7: c[79]= 158.000000
Thread 4 starting...
Thread 4: c[40]= 80.000000
Thread 4: c[41]= 82.000000
Thread 4: c[42]= 84.000000
Thread 4: c[43]= 86.000000
Thread 4: c[44]= 88.000000
Thread 4: c[45]= 90.000000
Thread 4: c[46]= 92.000000
Thread 4: c[47]= 94.000000
Thread 4: c[48]= 96.000000
Thread 4: c[49]= 98.000000
Thread 1 starting...
Thread 1: c[10]= 20.000000
Thread 1: c[11]= 22.000000
```

## 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 (*N*×*N*) usadas en la operación, como parámetro para observar la variación del tiempo de ejecución y el

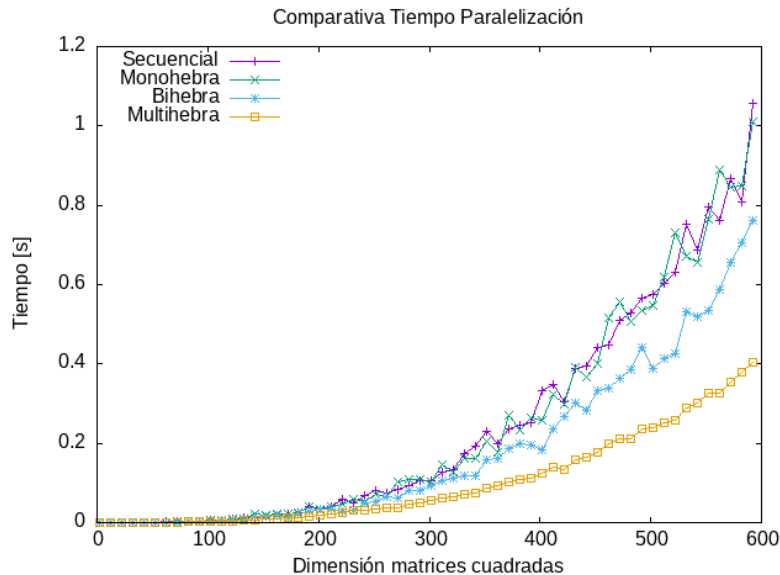
comportamiento según el tipo de paralelismo. Puesto que el código desarrollado contiene directivas de OpenMP para el paralelismo, la compilación requiere del argumento `-fopenmp`, como ya se advirtió en la tarea 1.1. En este ejercicio se ha ido incrementando el valor de `N` de 10 en 10 hasta llegar a 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.

```
(base) cesar@cesar-virtual-machine:~/Desktop$ g++ -o tarea1_3 tarea1_3.cpp -fopenmp
(base) cesar@cesar-virtual-machine:~/Desktop$ ./tarea1_3
# N (matrices NxN) T_Secuencial T_Monohebra T_Bihebra T_Multihebra Ganancia(1H) Ganancia(2H) Ganancia(4H) Eficiencia(1H) Eficiencia(2H) Eficiencia(4H)
2 2.8027e-05 5.2423e-05 7.5562e-05 0.000114644 0.534632 0.24447 0.370914 0.534632 0.122235 0.0927285
12 4.3643e-05 4.512e-05 3.959e-05 8.898e-05 0.967265 0.490481 1.10237 0.967265 0.245241 0.275594
22 0.000131007 0.000124224 0.000123857 0.000155718 1.05943 0.845162 1.06257 1.05943 0.422581 0.265643
32 0.000293173 0.000296051 0.000251171 0.000240286 0.990279 1.2291 1.16722 0.990279 0.61005 0.291806
42 0.000574954 0.000558862 0.000449512 0.000478189 1.02879 1.26236 1.27986 1.02879 0.601179 0.319766
52 0.000967137 0.00105001 0.00079444 0.000862111 0.921072 1.12182 1.21738 0.921072 0.560912 0.304346
62 0.00257425 0.00299081 0.00147695 0.00105487 1.23122 2.44035 1.74295 1.23122 1.22018 0.435736
72 0.00229512 0.00227661 0.00189969 0.00132875 1.00813 1.72727 1.20816 1.00813 0.863037 0.302039
82 0.00331694 0.00327227 0.00257831 0.00174113 1.01365 1.98505 1.28648 1.01365 0.92524 0.32162
92 0.0047608 0.00469424 0.00358181 0.00245186 1.0153 1.94386 1.33063 1.0153 0.971931 0.332659
102 0.00618994 0.0059281 0.00488295 0.00248243 1.04417 2.4935 1.26766 1.04417 1.24675 0.310916
112 0.00629666 0.00620755 0.005759 0.00596781 1.01436 1.0551 1.09336 1.01436 0.527552 0.27334
122 0.0100295 0.0082065 0.00748845 0.00764023 1.22214 1.31272 1.35379 1.22214 0.65636 0.338448
132 0.013515 0.0093377 0.0085509 0.0089720 1.2762 1.41662 1.37321 1.2762 0.72035 0.31032
```

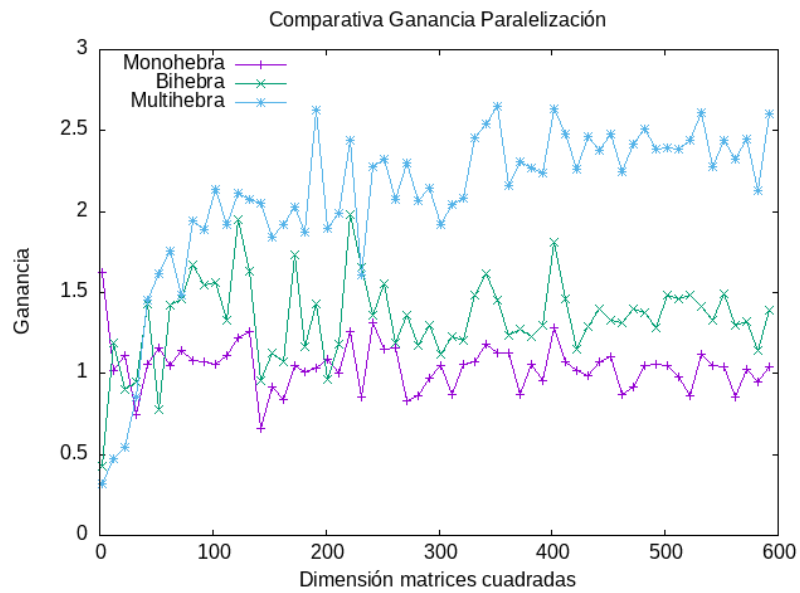
Los resultados obtenidos, para graficarlos posteriormente, se han guardado en el fichero `tarea1_3.time` empleado una tubería. Después, con el código `tarea1_3.gpi` implementado y mediante la herramienta Gnuplot se han graficado estos valores.

```
(base) cesar@cesar-virtual-machine:~/Desktop$ ./tarea1_3 > tarea1_3.time
(base) cesar@cesar-virtual-machine:~/Desktop$ gnuplot tarea1_3.gpi
```

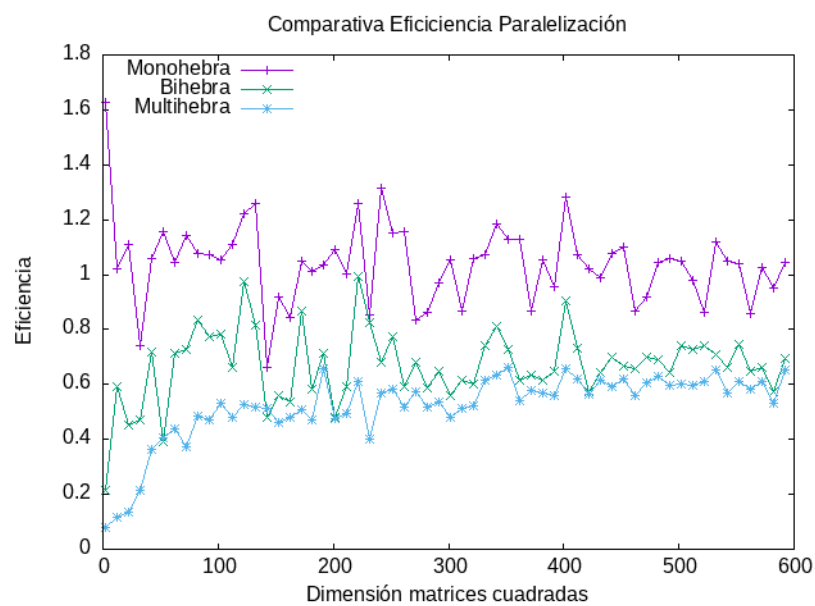
La primera gráfica recoge los tiempos medidos para cada caso de ejecución. Como es de esperar, la ejecución con 4 hebras presenta un tiempo de ejecución menor que en el resto de casos. Además, la ejecución secuencial y la monohebra no muestra diferencia alguna, indicando que el procedimiento es el mismo.



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.



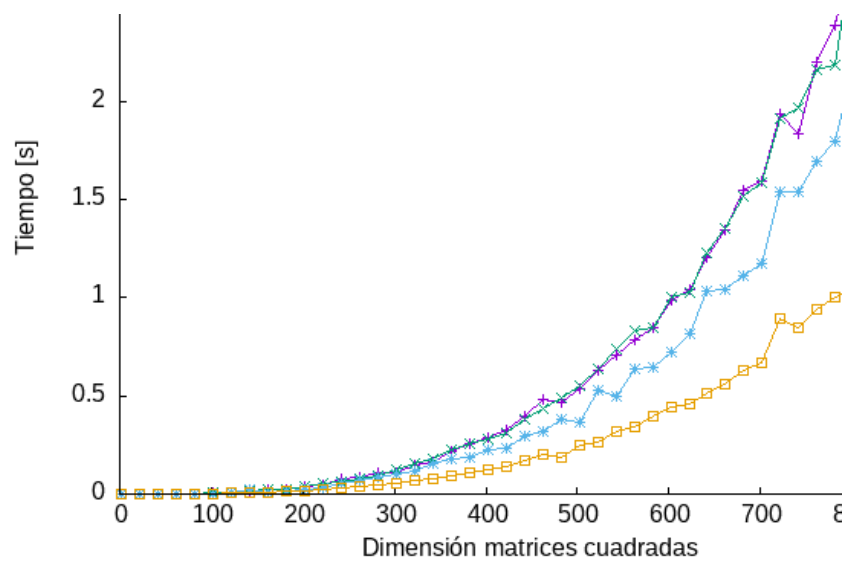
La última gráfica presenta los resultados de la eficiencia en función del número de hebras. En este caso, a diferencia de las dos gráficas anteriores, la paralelización que domina es la de una hebra (equivalente a secuencial). Destaca la poca eficiencia que presenta el modelo de paralelización de cuatro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



## RETO JEDI 2

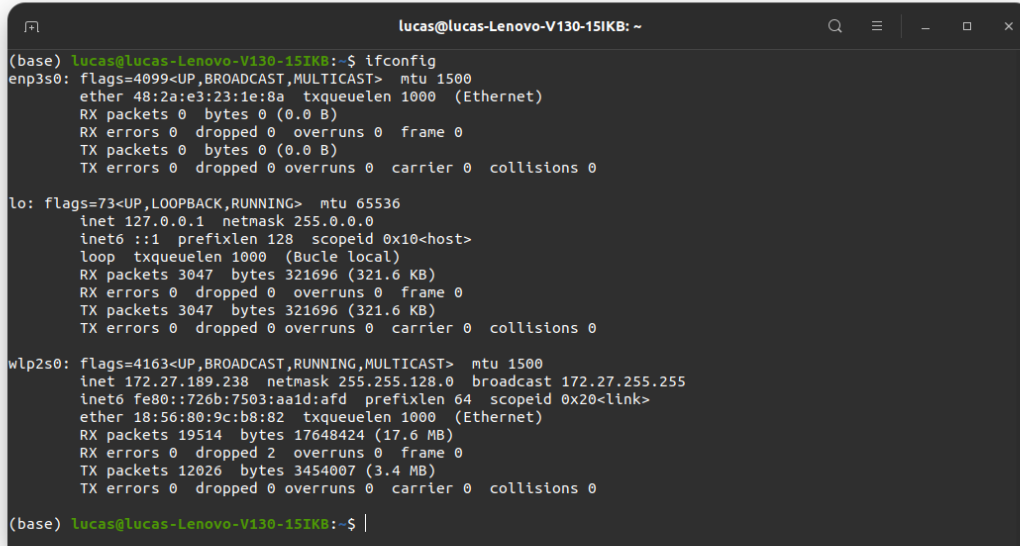
Si la matriz B es transpuesta, el código que produce el compilador para multiplicar las dos matrices A y B producirá menos fallos de caché, dado que maximizamos la localidad espacial de los datos en la cachés. Intente demostrar y medir si este hecho afecta a su problema. Nota: puede que tenga que escalar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



## 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, ...):



```
(base) lucas@lucas-Lenovo-V130-15IKB:~$ ifconfig
enp3s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
        ether 48:2a:e3:23:1e:8a txqueuelen 1000 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Bucle local)
        RX packets 3047 bytes 321696 (321.6 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 3047 bytes 321696 (321.6 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 172.27.189.238 netmask 255.255.128.0 broadcast 172.27.255.255
        inet6 fe80::726b:7503:aa1d:afd prefixlen 64 scopeid 0x20<link>
        ether 18:56:80:9c:b8:82 txqueuelen 1000 (Ethernet)
        RX packets 19514 bytes 17648424 (17.6 MB)
        RX errors 0 dropped 2 overruns 0 frame 0
        TX packets 12026 bytes 3454007 (3.4 MB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(base) lucas@lucas-Lenovo-V130-15IKB:~$ |
```

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.

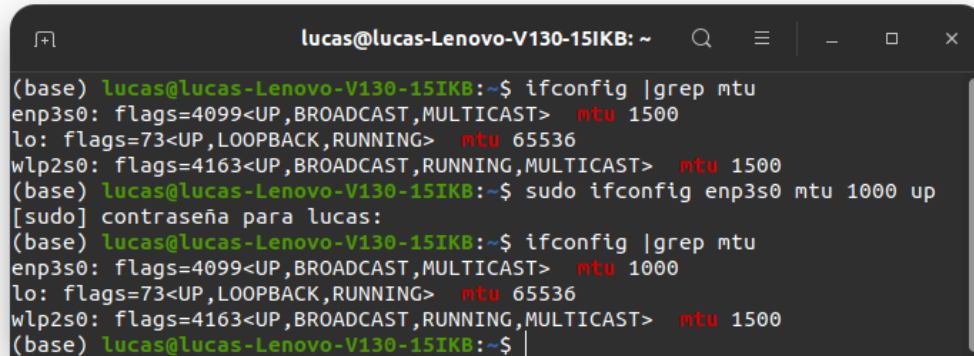
### 1. ¿Qué es la MAC? ¿Para qué sirve?

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:

A terminal window titled 'lucas@lucas-Lenovo-V130-15IKB: ~' showing the process of changing the MTU of the network interface 'enp3s0'. The user runs 'ifconfig | grep mtu' and sees 'mtu 1500' for 'enp3s0'. Then they run 'sudo ifconfig enp3s0 mtu 1000 up' and are prompted for a password. After running 'ifconfig | grep mtu' again, they see 'mtu 1000' for 'enp3s0'.

```
(base) lucas@lucas-Lenovo-V130-15IKB:~$ ifconfig |grep mtu
enp3s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
(base) lucas@lucas-Lenovo-V130-15IKB:~$ sudo ifconfig enp3s0 mtu 1000 up
[sudo] contraseña para lucas:
(base) lucas@lucas-Lenovo-V130-15IKB:~$ ifconfig |grep mtu
enp3s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1000
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
(base) lucas@lucas-Lenovo-V130-15IKB:~$
```

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?**



```
lucas@lucas-Lenovo-V130-15IKB: /proc
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$ cat cpuinfo | grep processor
processor      : 0
processor      : 1
processor      : 2
processor      : 3
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$
```

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

```
lucas@lucas-Lenovo-V130-15IKB: /proc
processor      : 1
vendor_id     : GenuineIntel
cpu family    : 6
model         : 142
model name    : Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
stepping      : 9
microcode     : 0xf4
cpu MHz       : 2259.830
cache size    : 3072 KB
physical id   : 0
siblings      : 4
core id       : 1
cpu cores     : 2
apicid        : 2
initial apicid : 2
fpu           : yes
fpu_exception : yes
cpuid level   : 22
```

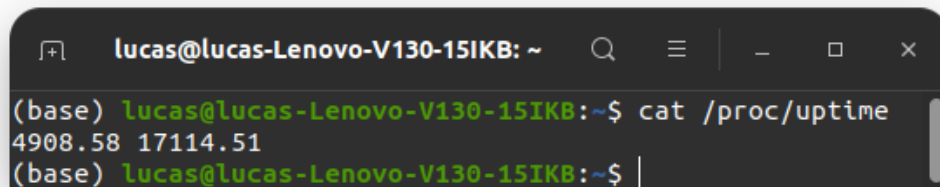
```
lucas@lucas-Lenovo-V130-15IKB: /proc
processor      : 2
vendor_id     : GenuineIntel
cpu family    : 6
model         : 142
model name    : Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
stepping      : 9
microcode     : 0xf4
cpu MHz       : 2700.000
cache size    : 3072 KB
physical id   : 0
siblings      : 4
core id       : 0
cpu cores     : 2
apicid        : 1
initial apicid : 1
fpu           : yes
fpu_exception : yes
cpuid level   : 22
```

```
lucas@lucas-Lenovo-V130-15IKB: /proc
processor      : 3
vendor_id     : GenuineIntel
cpu family    : 6
model         : 142
model name    : Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
stepping      : 9
microcode     : 0xf4
cpu MHz       : 2700.000
cache size    : 3072 KB
physical id   : 0
siblings      : 4
core id       : 1
cpu cores     : 2
apicid        : 3
initial apicid : 3
fpu           : yes
fpu_exception : yes
cpuid level   : 22
```

(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.

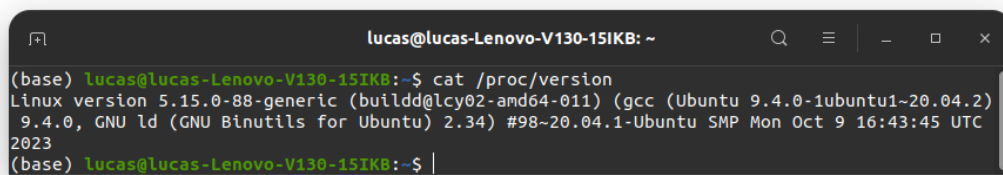
(d) Tiempo exacto que lleva encendida la tarjeta/equipo



```
lucas@lucas-Lenovo-V130-15IKB: ~  
(base) lucas@lucas-Lenovo-V130-15IKB:~$ cat /proc/uptime  
4908.58 17114.51  
(base) lucas@lucas-Lenovo-V130-15IKB:~$
```

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).

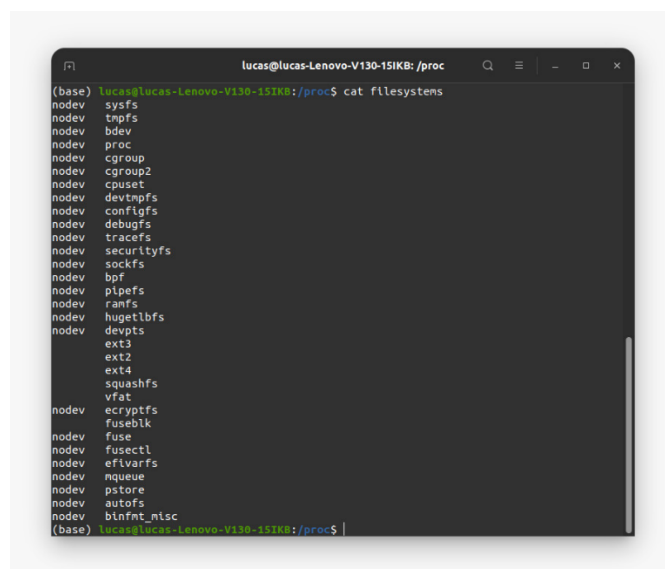
(e) Versión de Linux y versión del compilador con la que fue compilado.



```
lucas@lucas-Lenovo-V130-15IKB: ~  
(base) lucas@lucas-Lenovo-V130-15IKB:~$ cat /proc/version  
Linux version 5.15.0-88-generic (build@lcy02-amd64-011) (gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0, GNU ld (GNU Binutils for Ubuntu) 2.34) #98-20.04.1-Ubuntu SMP Mon Oct 9 16:43:45 UTC 2023  
(base) lucas@lucas-Lenovo-V130-15IKB:~$
```

(f) ¿Qué sistemas de archivos (filesystems) se soportan? Comente las características de, al menos, 5 de ellos.

Soporta los siguientes sistemas:



```
lucas@lucas-Lenovo-V130-15IKB: /proc  
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$ cat filesystems  
nodev sysfs  
nodev tmpfs  
nodev bdev  
nodev proc  
nodev cgroup  
nodev cgroup2  
nodev cpuset  
nodev devtmpfs  
nodev configfs  
nodev debugfs  
nodev tracefs  
nodev securityfs  
nodev sockfs  
nodev bpf  
nodev pipefs  
nodev ramfs  
nodev hugetlbfs  
nodev devpts  
nodev ext3  
nodev ext2  
nodev ext4  
nodev squashfs  
nodev vfat  
nodev ecryptfs  
nodev fuseblk  
nodev fuse  
nodev fusectl  
nodev efivarfs  
nodev nqueue  
nodev pstore  
nodev autofs  
nodev binfmt_misc  
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$
```

**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 configuración del 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?

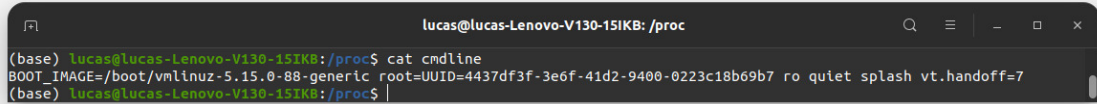
Muestra el estado de los procesos desde el último arranque del sistema. Cada proceso tiene una carpeta propia que muestra su información de una manera más clara.

[illegible]

```
lucas@lucas-Lenovo-V130-15IKB: /proc
(base) lucas@lucas-Lenovo-V130-15IKB: /proc$ cat 1305/status
Name: ibus-daemon
Umask: 0002
State: S (sleeping)
Tgid: 1305
Ngid: 0
Pid: 1305
PPid: 1265
TracerPid: 0
Uid: 1000 1000 1000 1000
Gid: 1000 1000 1000 1000
FDSize: 64
Groups: 4 20 24 27 30 46 120 131 132 1000
NSTgid: 1305
NSpid: 1305
NSpgid: 1305
NSsid: 1265
VmPeak: 379016 kB
VmSize: 313876 kB
VmCck: 0 kB
VmPtn: 0 kB
VmHWM: 8616 kB
VmRSS: 8616 kB
RssAnon: 1520 kB
RssFile: 7096 kB
RssShmem: 0 kB
VmData: 34468 kB
VmStk: 132 kB
```

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

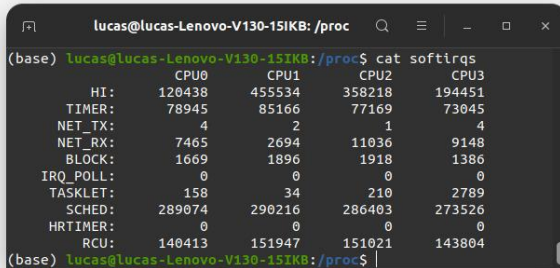
Indica los parámetros que recibió el kernel en el momento de su arranque.



```
lucas@lucas-Lenovo-V130-15IKB: /proc
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$ cat cmdline
BOOT_IMAGE=/boot/vmlinuz-5.15.0-88-generic root=UUID=4437df3f-3e6f-41d2-9400-0223c18b69b7 ro quiet splash vt.handoff=7
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$
```

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

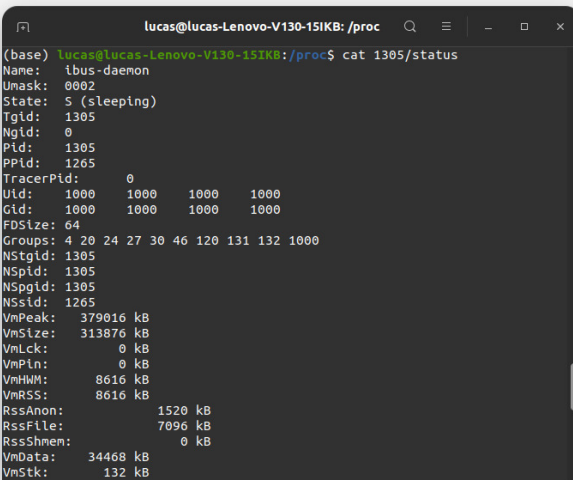
Muestra información sobre los diferentes procesadores del sistema.



```
lucas@lucas-Lenovo-V130-15IKB: /proc
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$ cat softirqs
          CPU0      CPU1      CPU2      CPU3
HI:       120438    455534    358218    194451
TIMER:    78945    85166    77169    73045
NET_TX:     4        2        1        4
NET_RX:    7465    2694    11036    9148
BLOCK:     1669    1896    1918    1386
IRQ_POLL:   0        0        0        0
TASKLET:    158     34     210    2789
SCHED:    289074   290216   286403   273526
HRTIMER:   0        0        0        0
RCU:      140413   151947   151021   143804
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$
```

**(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:



```
lucas@lucas-Lenovo-V130-15IKB: /proc
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$ cat 1305/status
Name:   tbus-daemon
Umask:  0002
State:  S (sleeping)
Tgid:   1305
Ngid:   0
Pid:    1305
PPid:   1265
TracerPid: 0
Uld:    1000 1000 1000 1000
Gid:    1000 1000 1000 1000
FDSize: 64
Groups: 4 20 24 27 30 46 120 131 132 1000
NStgid: 1305
NSpid:  1305
NSpgid: 1305
NSSid:  1265
VmPeak: 379016 kB
VmSize: 313876 kB
VmLck:  0 kB
VmPIn:  0 kB
VmHWM:  8616 kB
VmRSS:  8616 kB
RssAnon:        1520 kB
RssFile:        7096 kB
RssShmem:        0 kB
VmData: 34468 kB
VmStk:  132 kB
```

**(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:

```

lucas@lucas-Lenovo-V130-15IKB: /proc
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$ cat mounts | grep ext3
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$ cat mounts | grep ext4
/dev/nvme0n1p4 / ext4 rw,relatime,errors=remount-ro 0 0
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$ findmnt | grep ext4
/ /dev/nvme0n1p4 ext4 rw,relatime,errors=remount-ro
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$ cat mounts | grep ext3
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$

```

**(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.

```

lucas@lucas-Lenovo-V130-15IKB: /proc
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$ free
              total        usado        libre compartido búfer/caché disponible
Memoria:    11566772    2493444    5870284    449908    3203044    8329076
Swap:        2097148         0    2097148
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$ cat meminfo
MemTotal:    11566772 kB
MemFree:     5868572 kB
MemAvailable: 8327528 kB
Buffers:     117312 kB

```

**(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.

```

lucas@lucas-Lenovo-V130-15IKB: /proc
(base) lucas@lucas-Lenovo-V130-15IKB:/proc$ cat modules
ccm 20480 3 - Live 0x0000000000000000
bnep 28672 2 - Live 0x0000000000000000
nls_iso8859_1 16384 1 - Live 0x0000000000000000
snd_soc_skl 172032 0 - Live 0x0000000000000000
snd_soc_hdac_hda 24576 1 snd_soc_skl, Live 0x0000000000000000
snd_hda_ext_core 32768 2 snd_soc_skl,snd_soc_hdac_hda, Live 0x0000000000000000
snd_soc_sst_ipc 20480 1 snd_soc_skl, Live 0x0000000000000000
snd_hda_codec_hdmi 77824 1 - Live 0x0000000000000000
snd_soc_sst_dsp 36864 1 snd_soc_skl, Live 0x0000000000000000
snd_soc_acpi_intel_match 61440 1 snd_soc_skl, Live 0x0000000000000000
snd_soc_acpi 16384 2 snd_soc_skl,snd_soc_acpi_intel_match, Live 0x0000000000000000
snd_hda_codec_conexant 28672 1 - Live 0x0000000000000000
snd_hda_codec_generic 102400 1 snd_hda_codec_conexant, Live 0x0000000000000000
ledtrig_audio 16384 1 snd_hda_codec_generic, Live 0x0000000000000000
intel_tcc_cooling 16384 0 - Live 0x0000000000000000
x86_pkg_temp_thermal 20480 0 - Live 0x0000000000000000
mei_hdcp 24576 0 - Live 0x0000000000000000
joydev 32768 0 - Live 0x0000000000000000
intel_rapl_msr 20480 0 - Live 0x0000000000000000
intel_powerclamp 24576 0 - Live 0x0000000000000000

```

```

lucas@lucas-Lenovo-V130-15IKB: /proc$ lsmod
Module                  Size  Used by
ccm                     20480  3
bnep                    20672  2
nls_iso8859_1          16384  1
snd_soc_skl            172032  0
snd_soc_hdac_hda       24576  1 snd_soc_skl
snd_hda_ext_core       32768  2 snd_soc_hdac_hda,snd_soc_skl
snd_soc_sst_ipc        20480  1 snd_soc_skl
snd_hda_codec_hdmi     77824  1
snd_soc_sst_dsp        36864  1 snd_soc_skl
snd_soc_acpi_intel_match 61440  1 snd_soc_skl
snd_soc_acpi           16384  2 snd_soc_acpi_intel_match,snd_soc_skl
snd_hda_codec_conexant  28672  1
snd_hda_codec_generic 102400  1 snd_hda_codec_conexant
ledtrig_audio          16384  1 snd_hda_codec_generic
intel_tcc_cooling       16384  0
x86_pkg_temp_thermal   20480  0
mei_hdcp               24576  0
joydev                 32768  0
intel_rapl_msr         20480  0
intel_powerclamp       24576  0

```

#### (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, btintel, btbcm, bnep y btusb.

```

lucas@lucas-Lenovo-V130-15IKB: /proc$ lsmod | grep bluetooth
bluetooth             688128  11 btrtl,btintel,btbcm,bnep,btusb

```