

Practica 2 Sistemas Operativos 1

Byron Jose Lopez Herrera

201222626

Instalacion del hipervisor KVM en Fedora 29

1. Para instalar el hipervisor kvm es necesario verificar si nuestro cpu tiene la virtualizacion habilitada con el siguiente comando.

```
$ cat /proc/cpuinfo | egrep "vmx|svm"
```

2. Instalar paquetes para virtualizar.

```
$ sudo dnf -y install bridge-utils libvirt virt-install qemu-kvm
```

3. Habilitar e iniciar el proceso de KVM.

```
$ sudo sudo systemctl start libvirtd  
$ sudo systemctl enable libvirtd
```

4. Instalar el gestor grafico para KVM

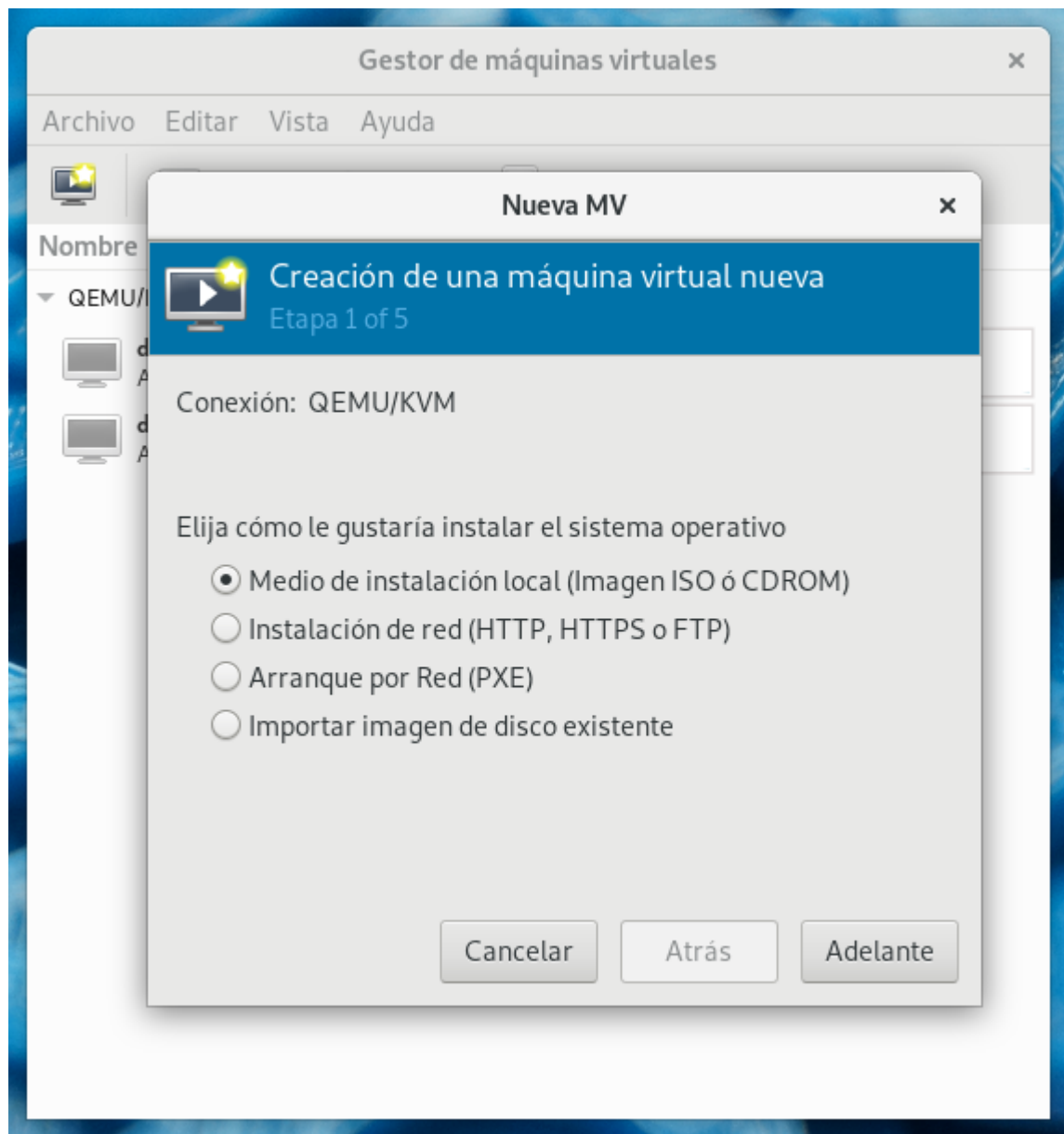
```
$ sudo dnf -y install virt-manager
```

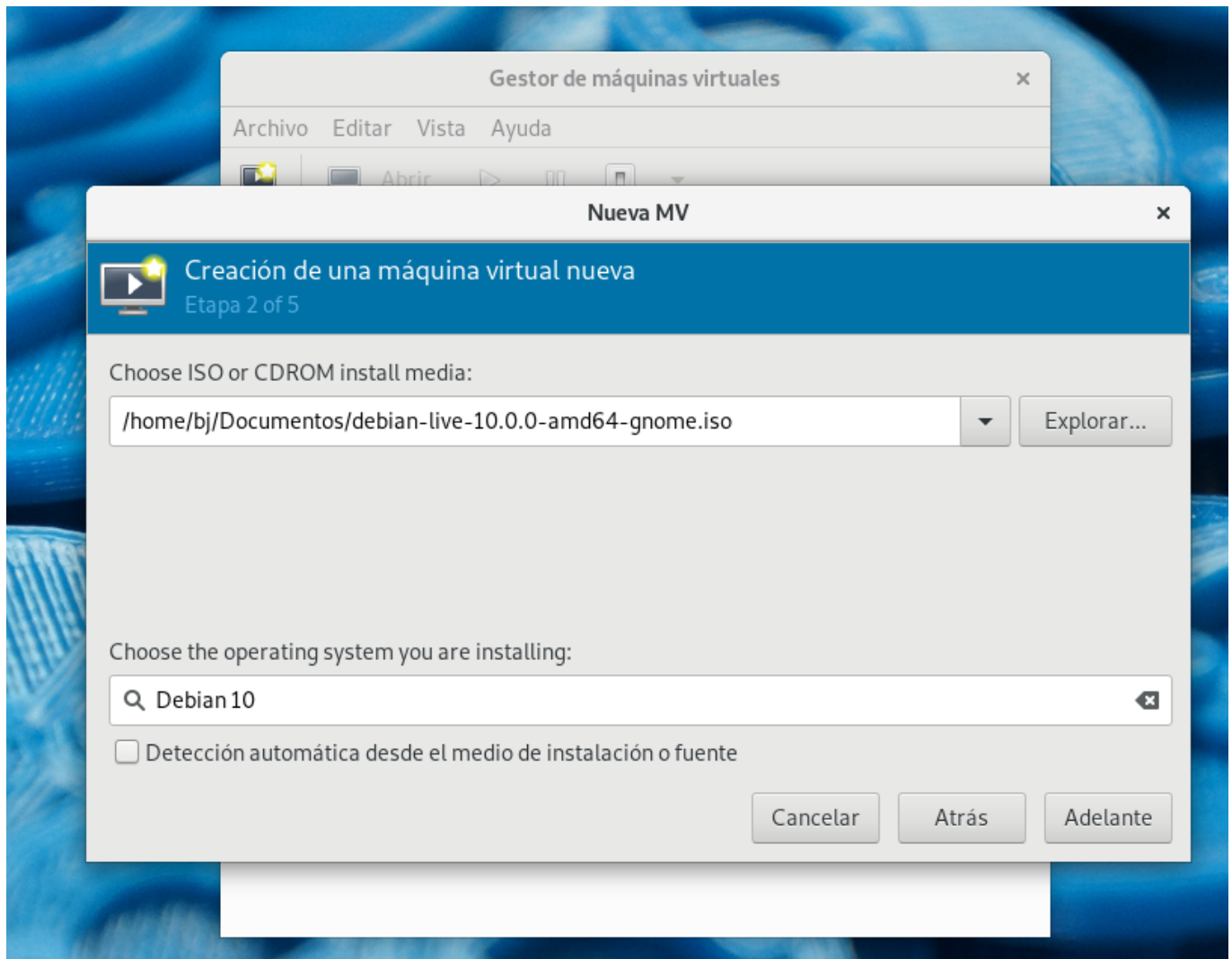
Instalacion de Debian 10

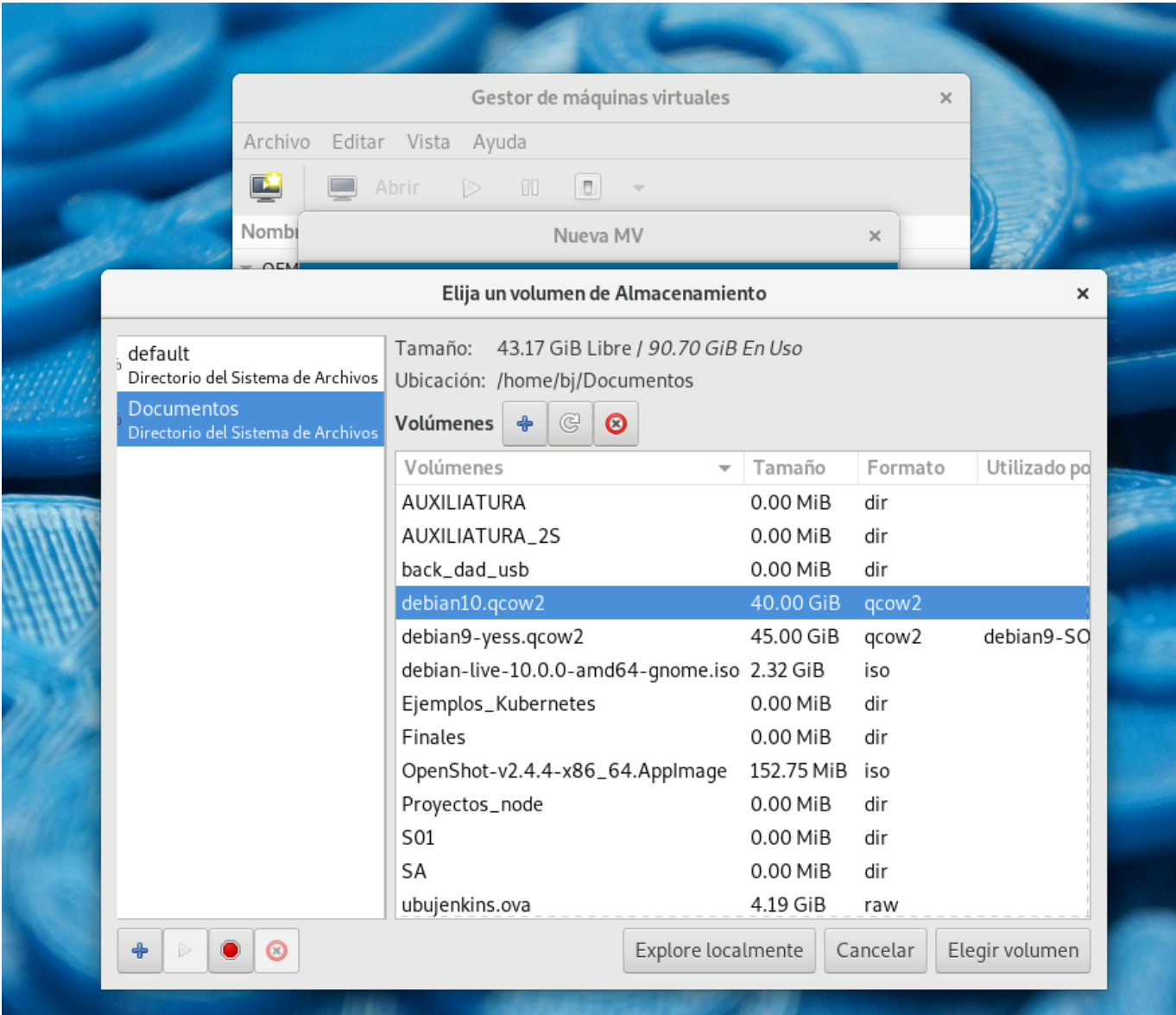
La instalacion de Debian 10 es bastante facil e intuitiva. A continuacion se muestran imagenes de algunos de los pasos que se siguen. Algunas cosas a tomar en cuenta para la instancia para poder compilar el kernel sin ningun problema.

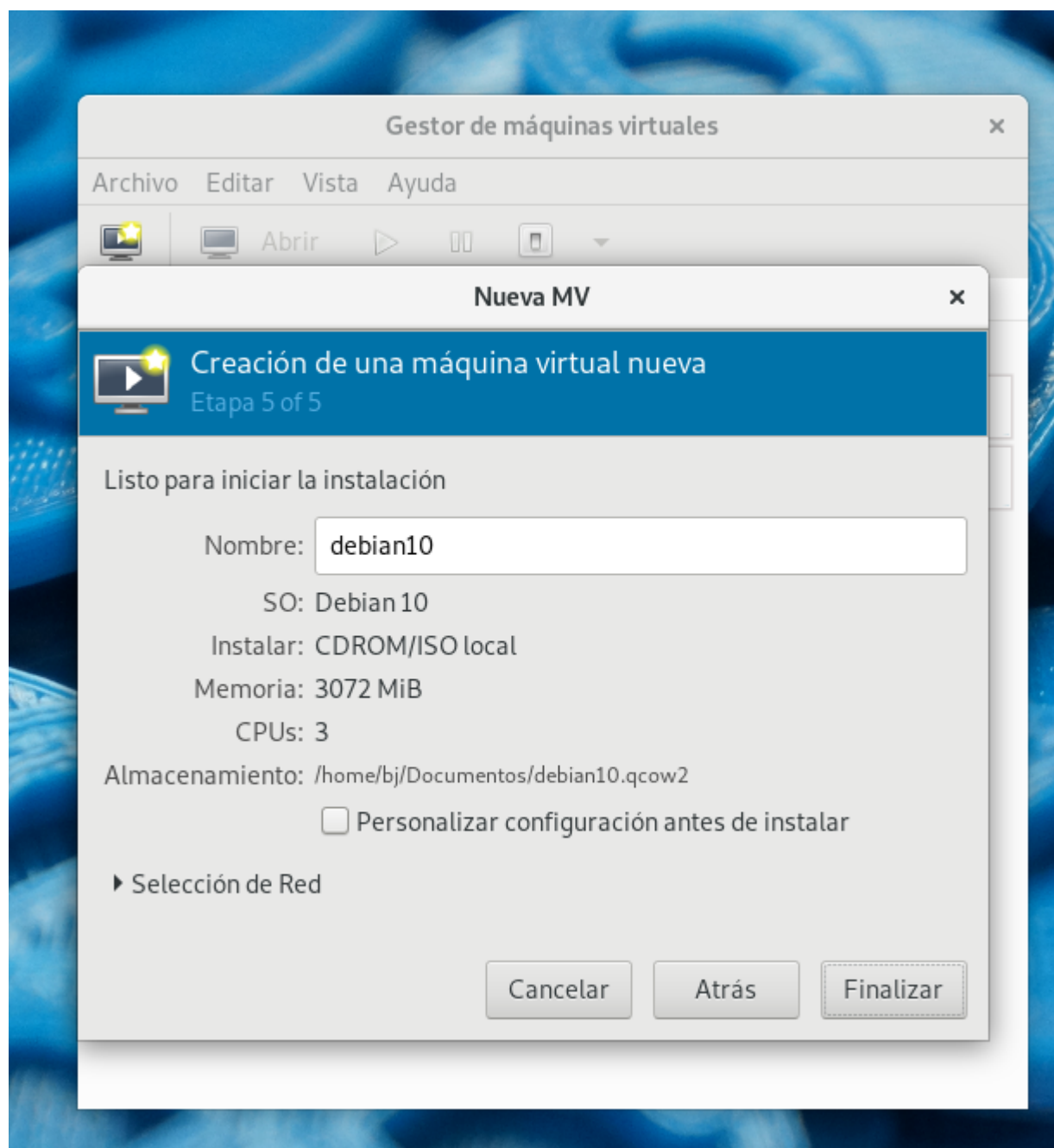
- Darle por lo menos a la maquina unos 40 GB de espacio.
- Si es posible darle la mayoria de hilos de ejecucion en mi caso son 3 nucleos.

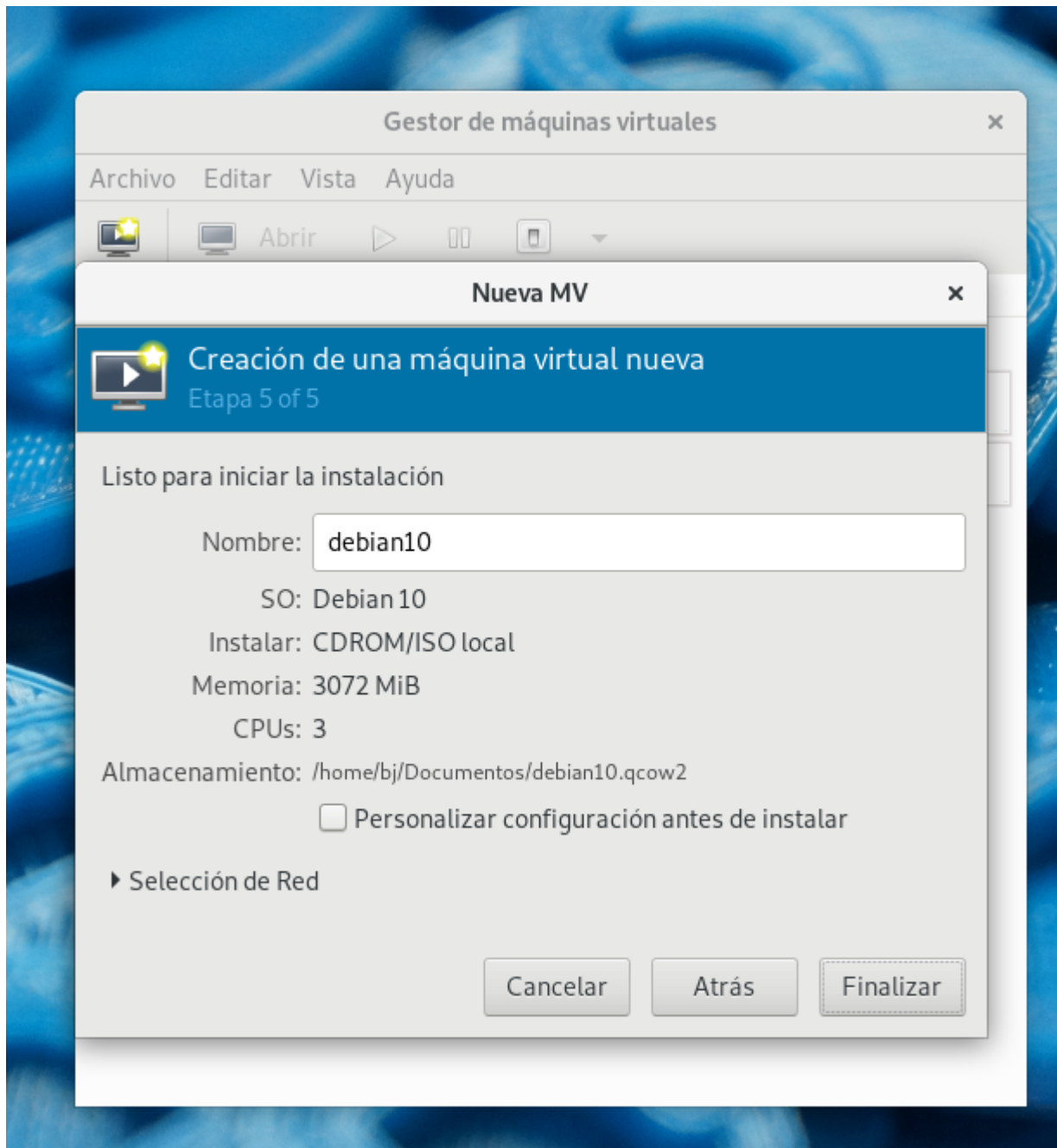
Imagenes











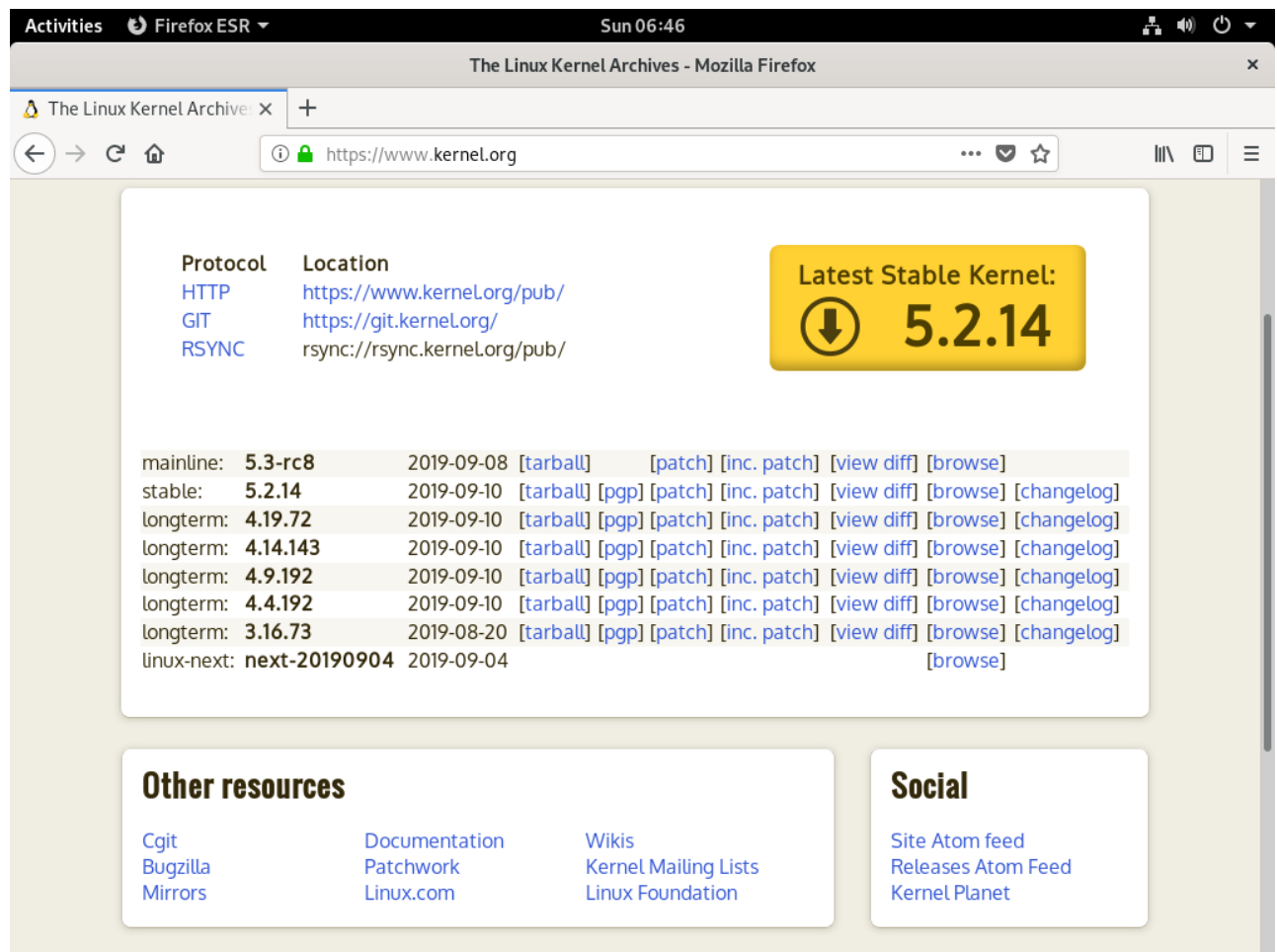
Compilacion del kernel

1. Instalar las librerías de GCC y otras herramientas para la compilacion del kernel.

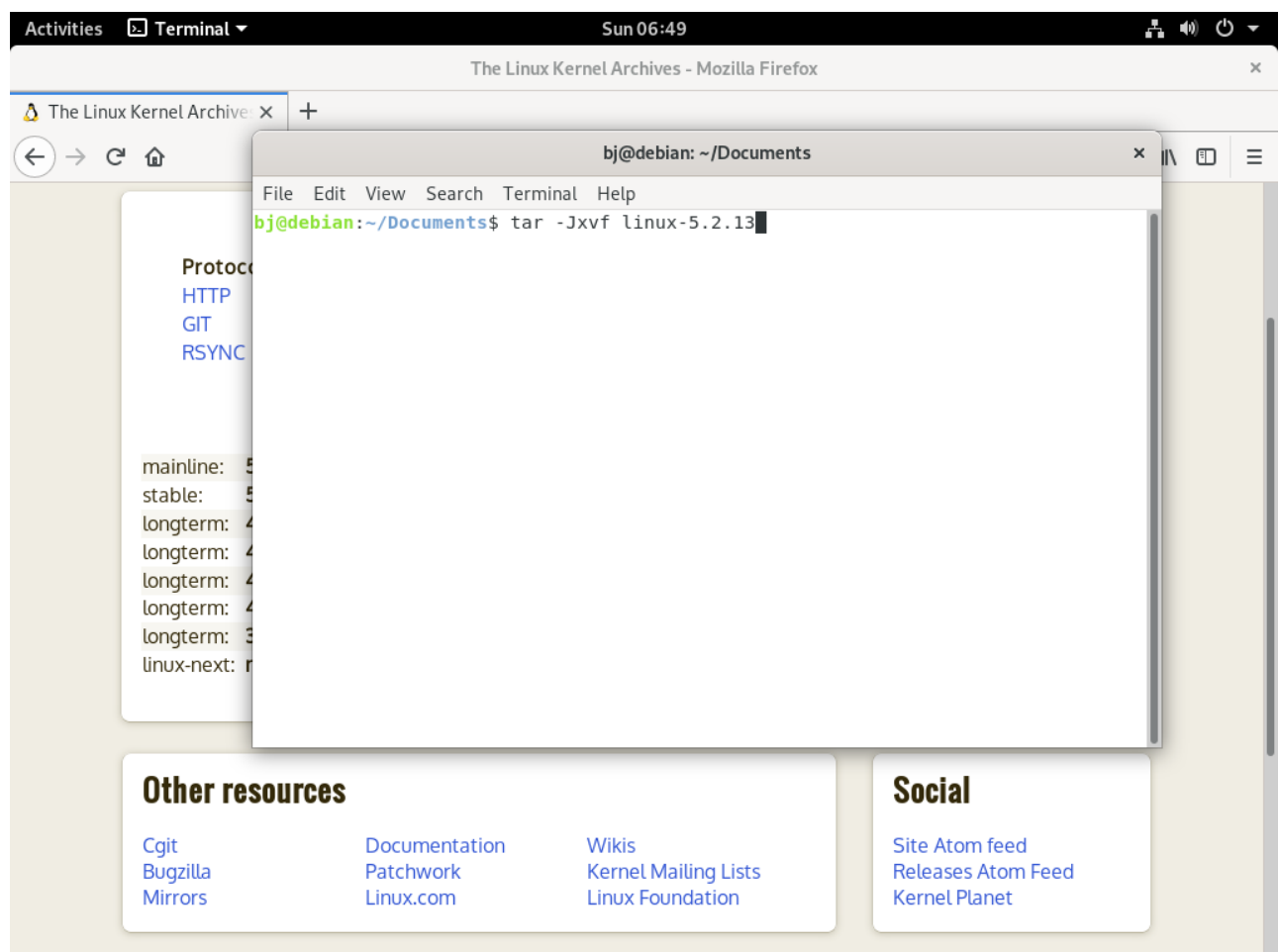
```
$ sudo apt-get install build-essential libncurses-dev bison flex  
libssl-dev libelf-dev
```

2. Descargar desde la pagina oficial del kernel linux los binarios.

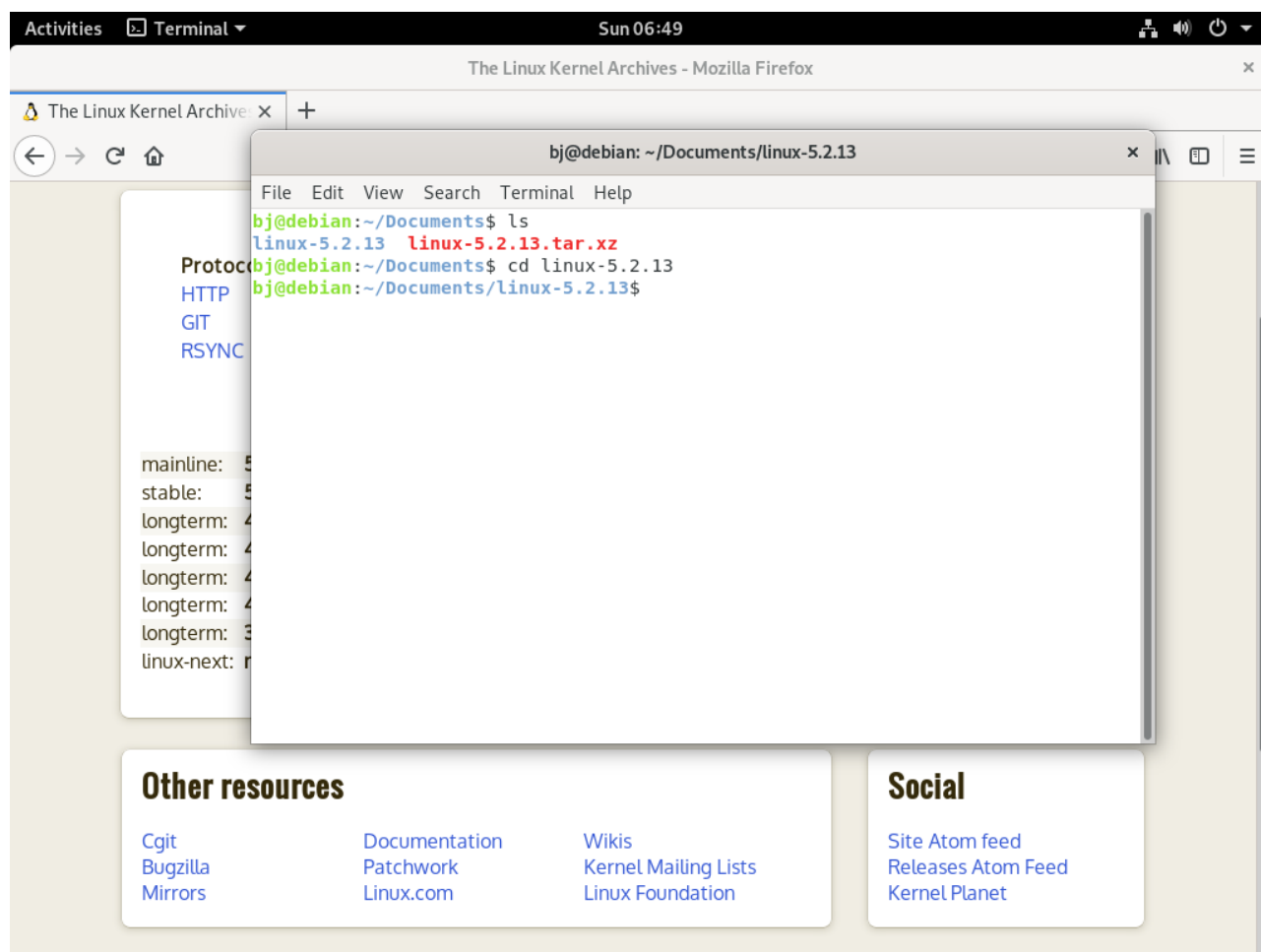
```
$ tar -Jxvf linux-5.2.13
```



3. Extraer los archivos en cualquier directorio de preferencia.

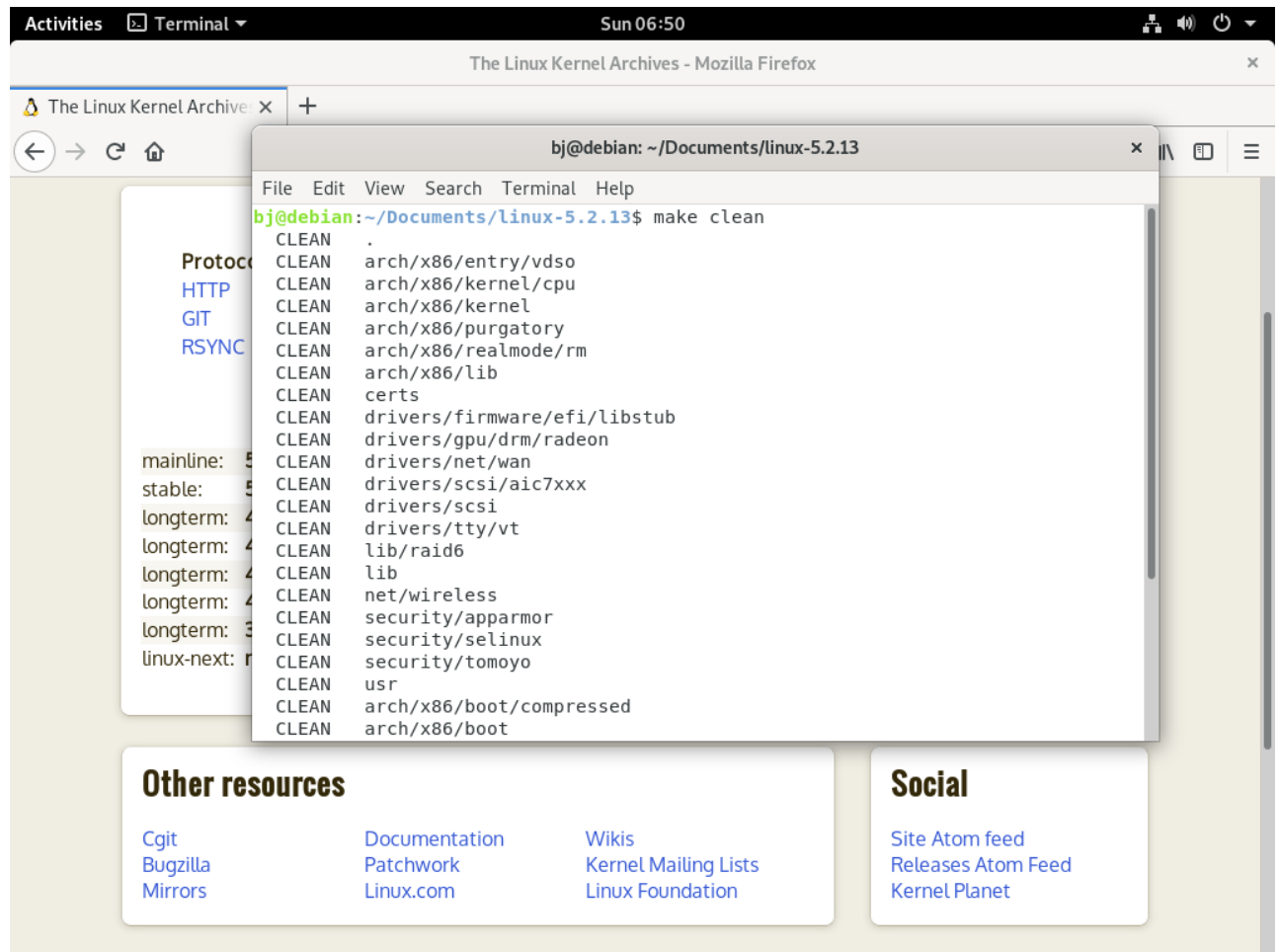


- Ingreso a la carpeta que se descomprimió con los archivos del kernel.



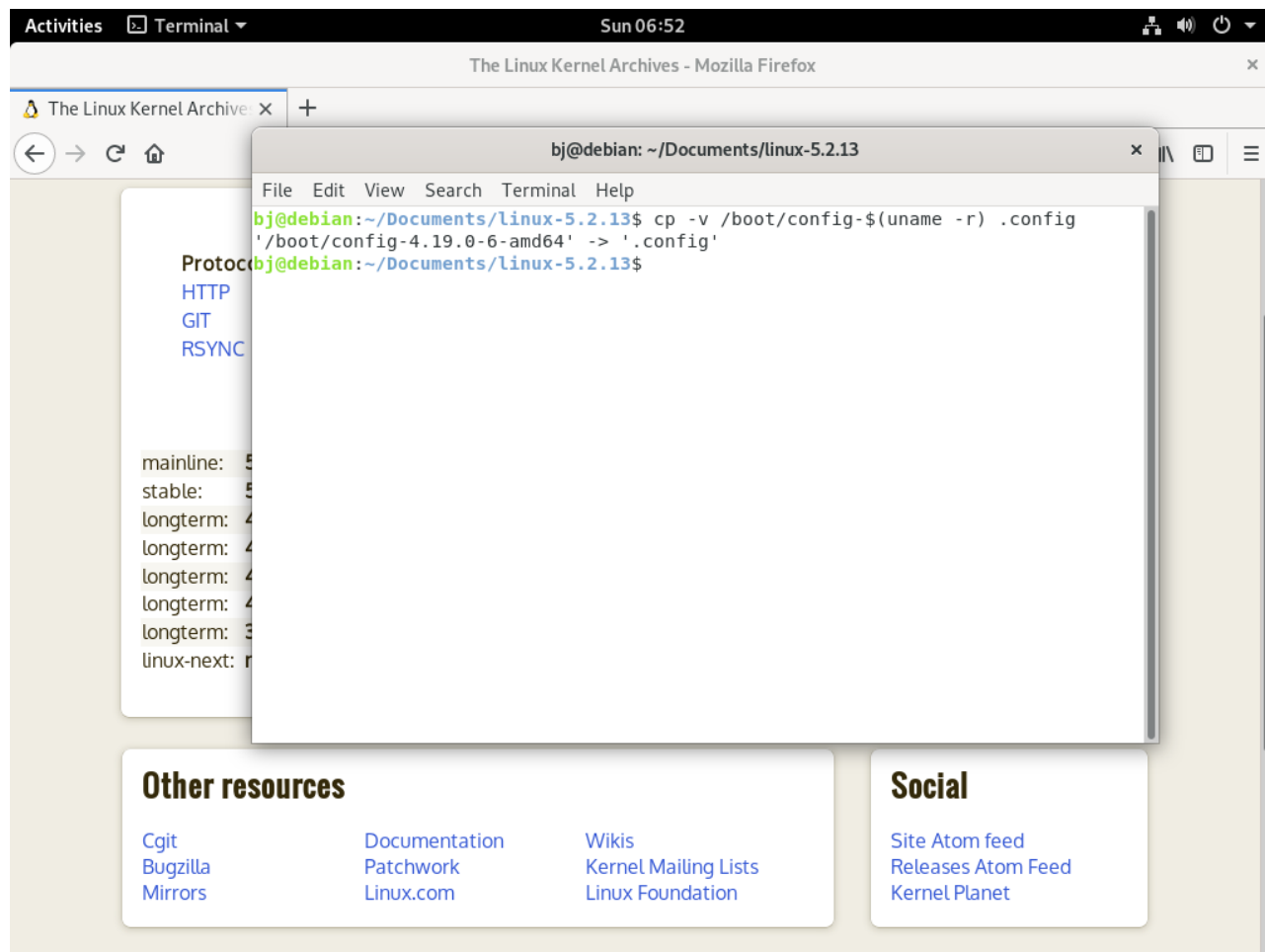
- Utilizar el siguiente comando para limpiar el cache de make.

```
$ make clean
```

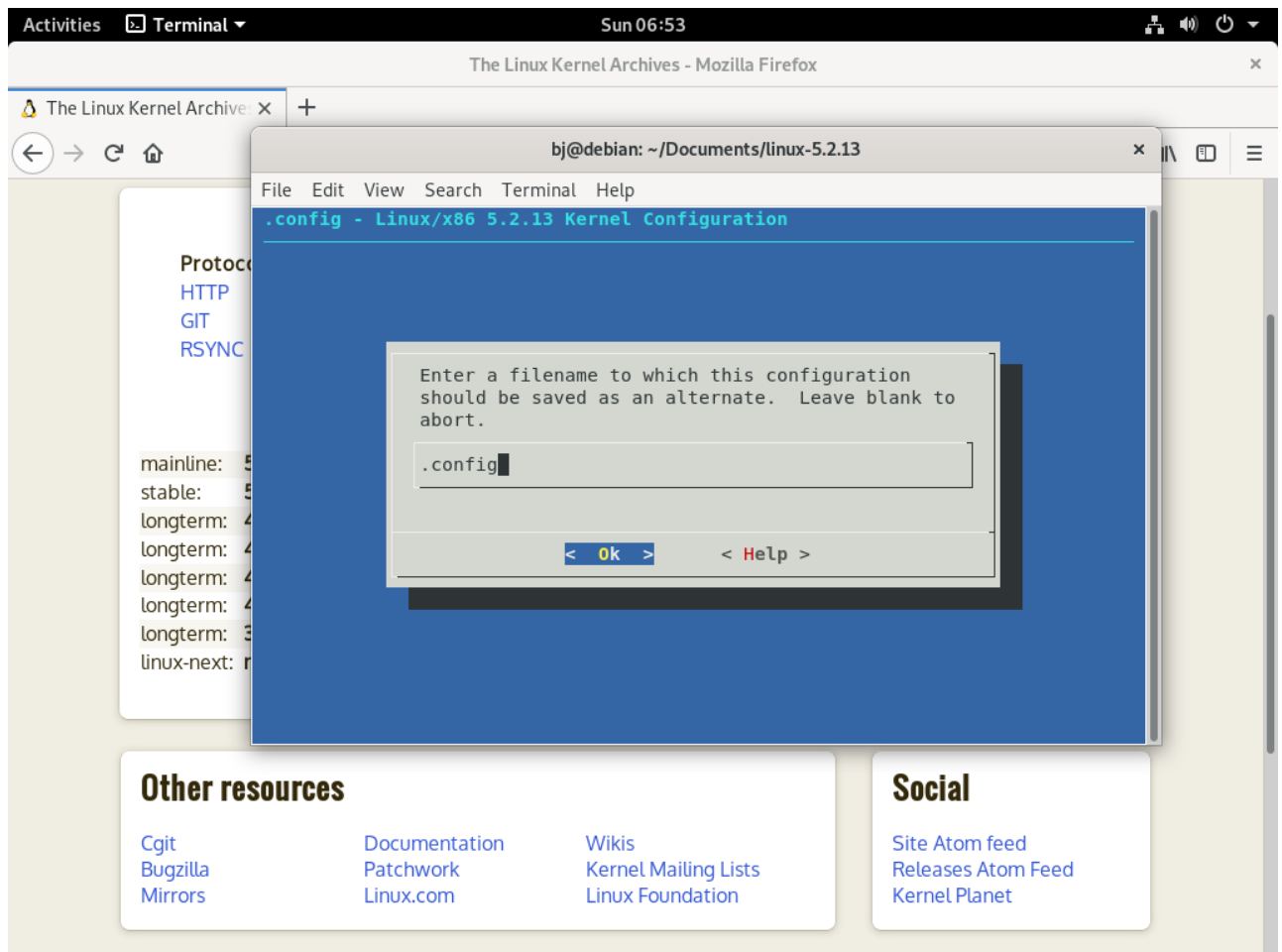
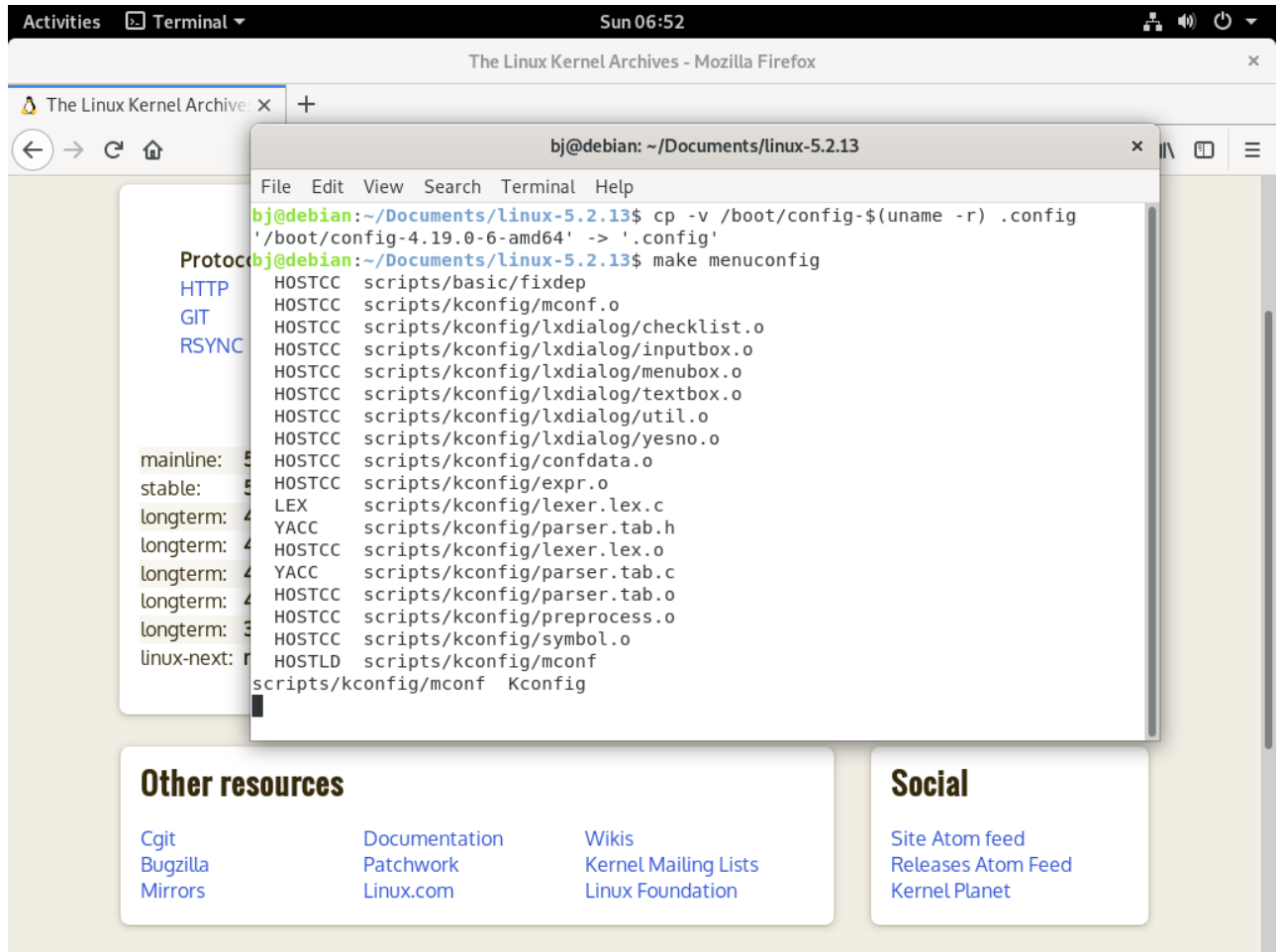
6. Utilizar el siguiente comando copiar las configuraciones del kernel.

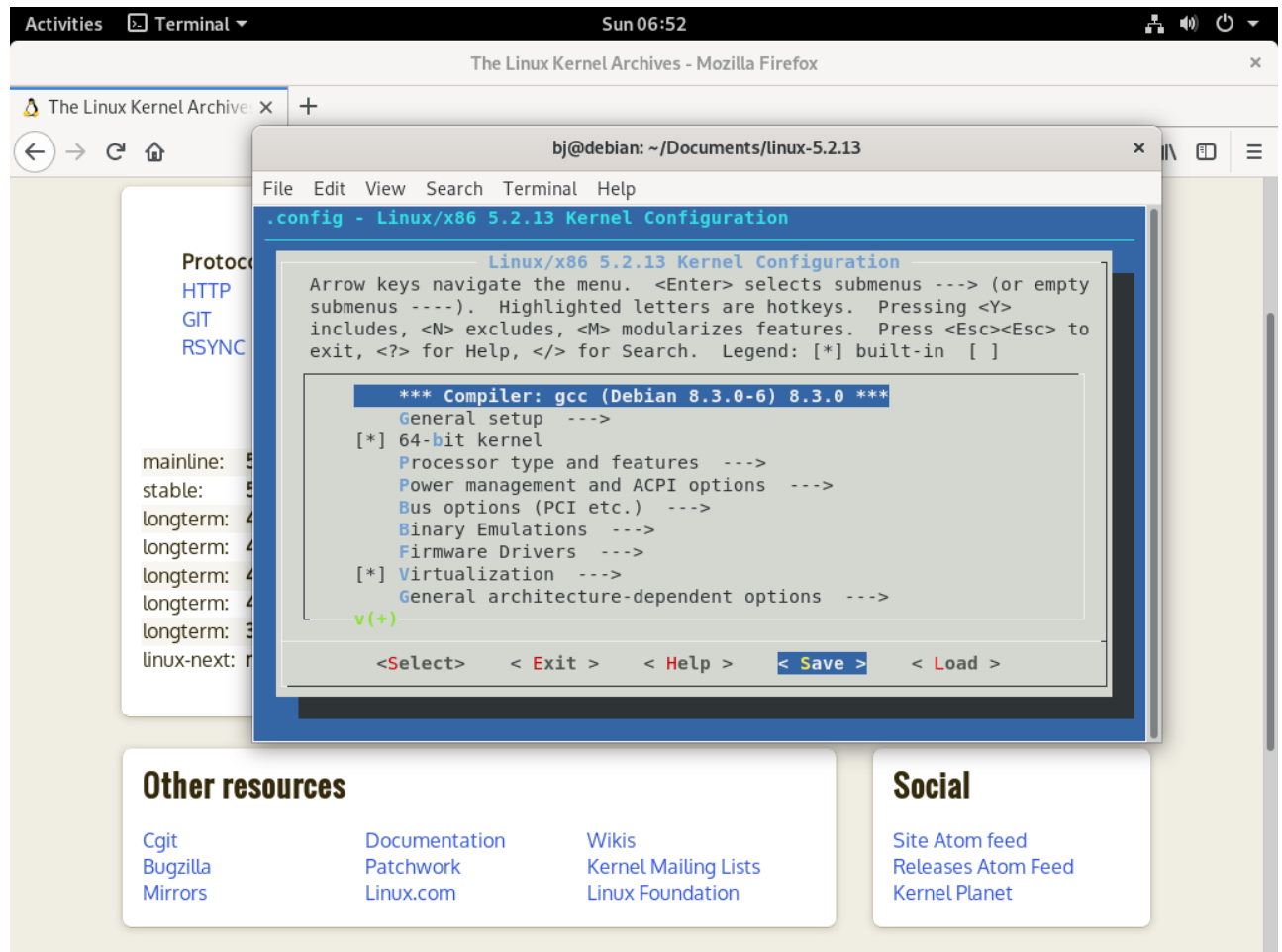
```
$ cp -v /boot/config-$(uname -r) .config
```



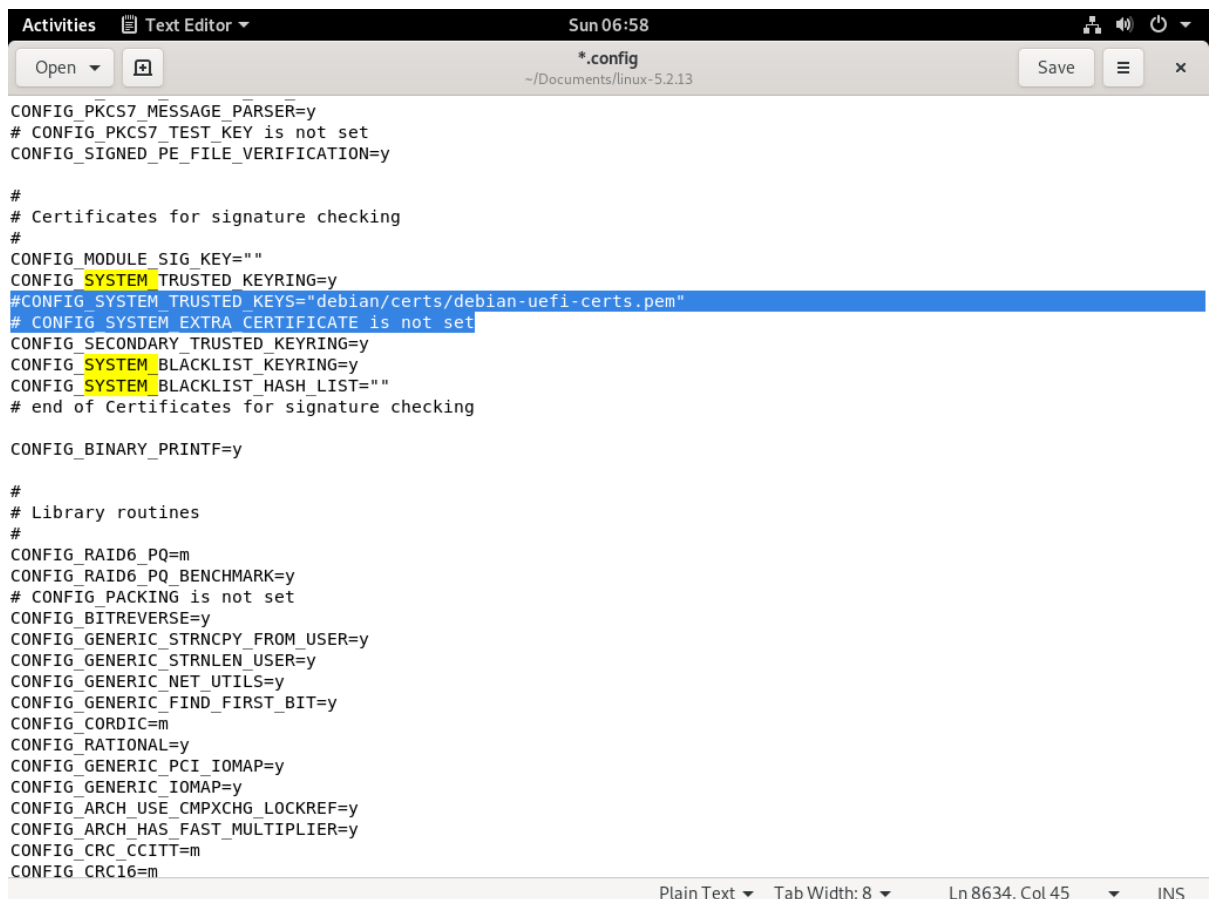
7. Utilizar el siguiente comando para seleccionar los modulos para instalar del kernel. Para nuestro caso instalaremos todo, lo que no se configura nada.

```
$ make menuconfig
```



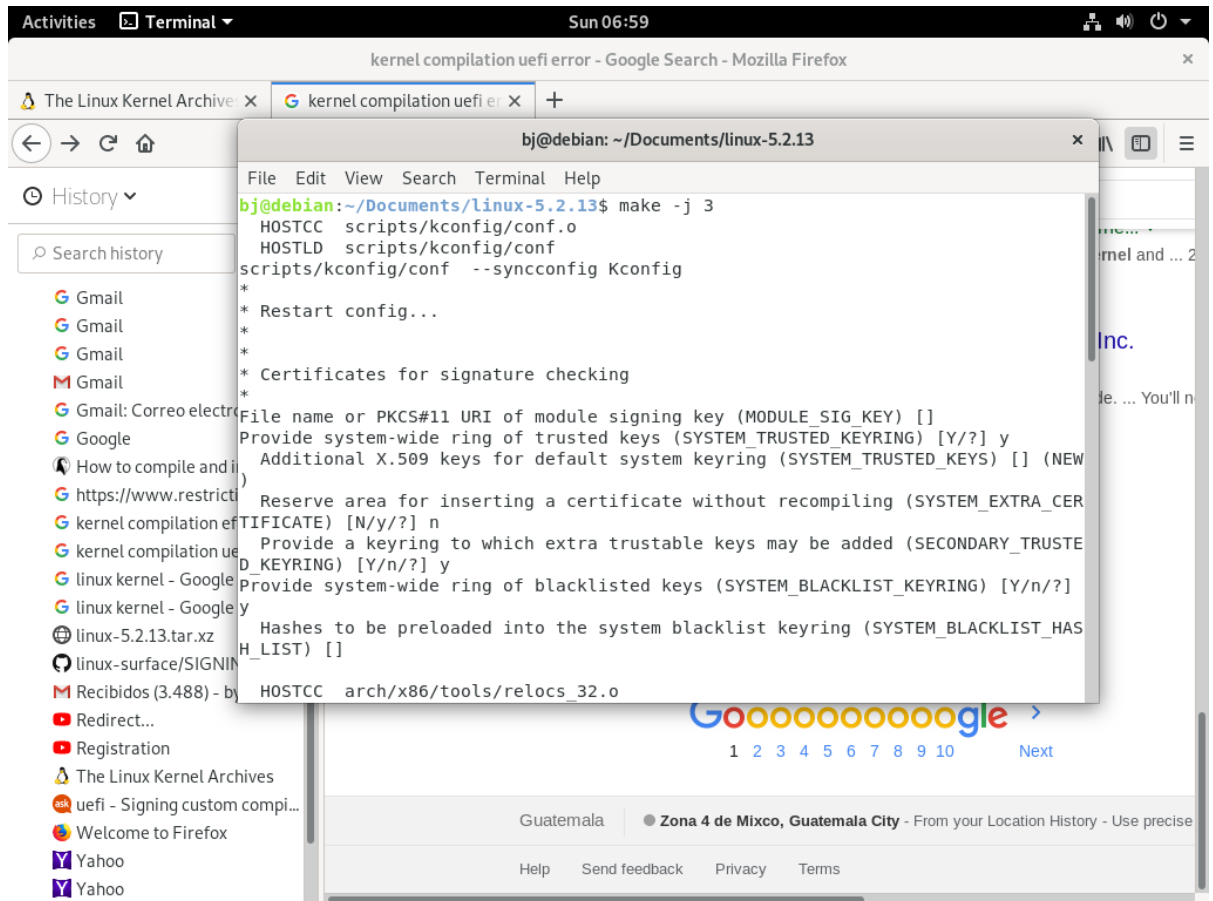


7. Si tu sistema utiliza Uefi es necesario comentar las siguientes lineas para que no falle la compilacion del kernel.

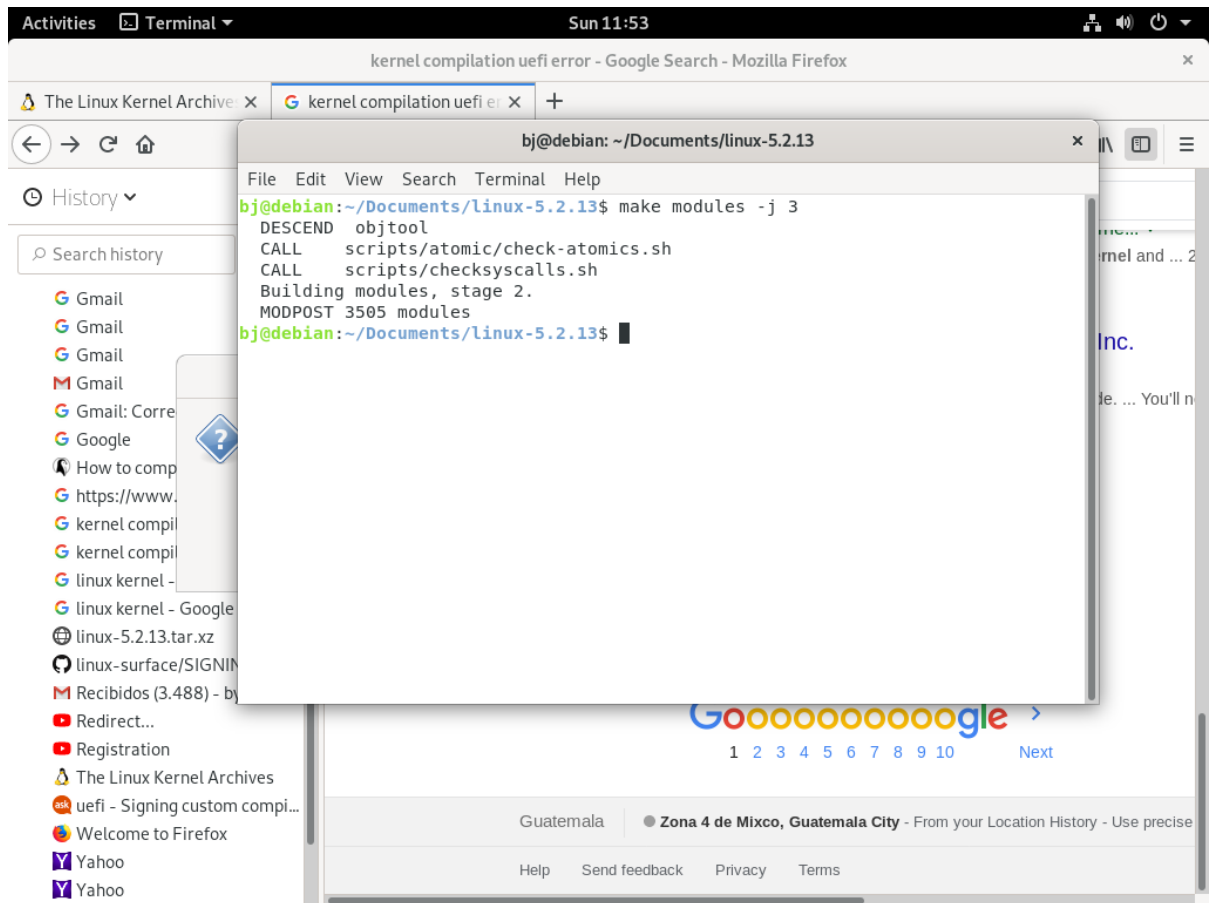


8. Utilizar los siguiente comando para inicializar la compilacion.El numero 3 es numero maximo de hilos que tiene capacidad mi VM.

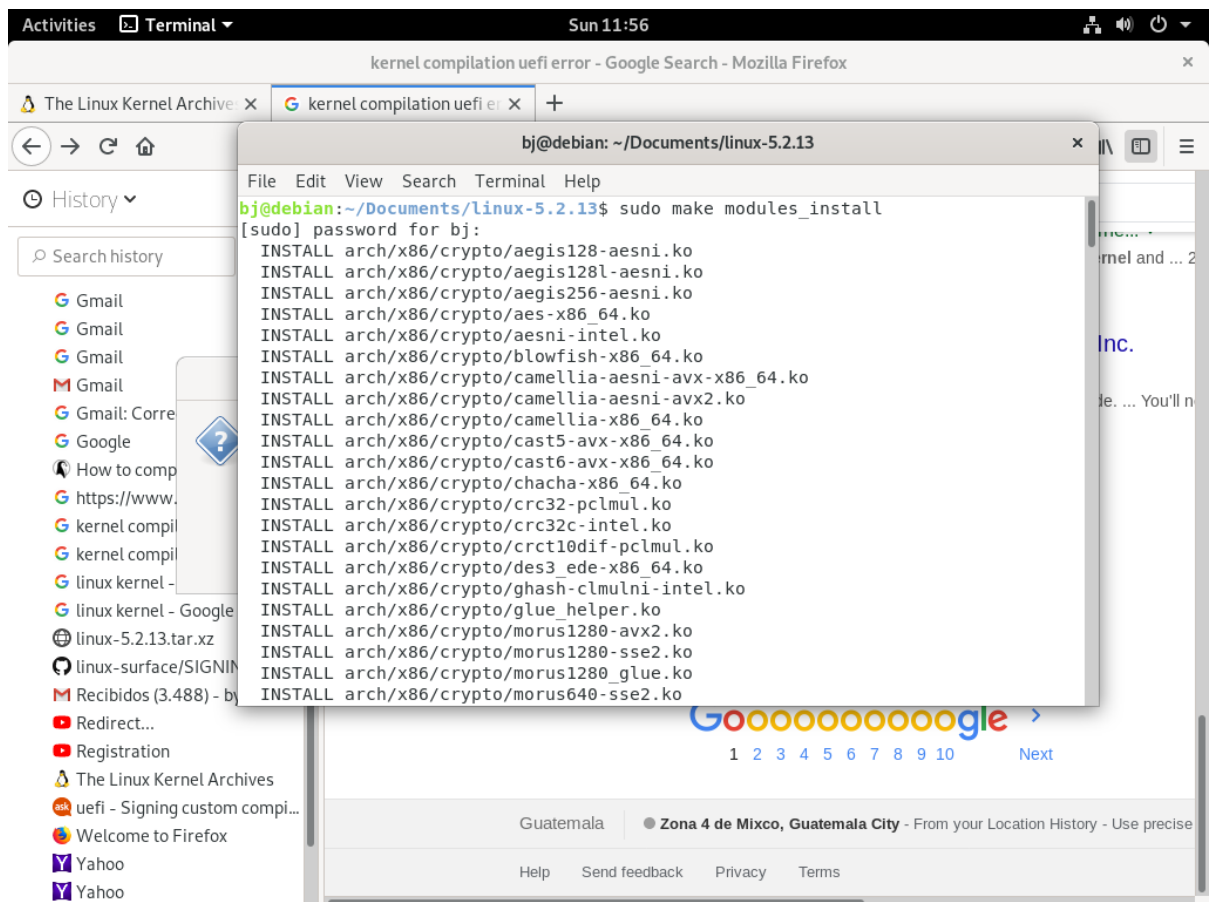
```
$ make -j 3
```



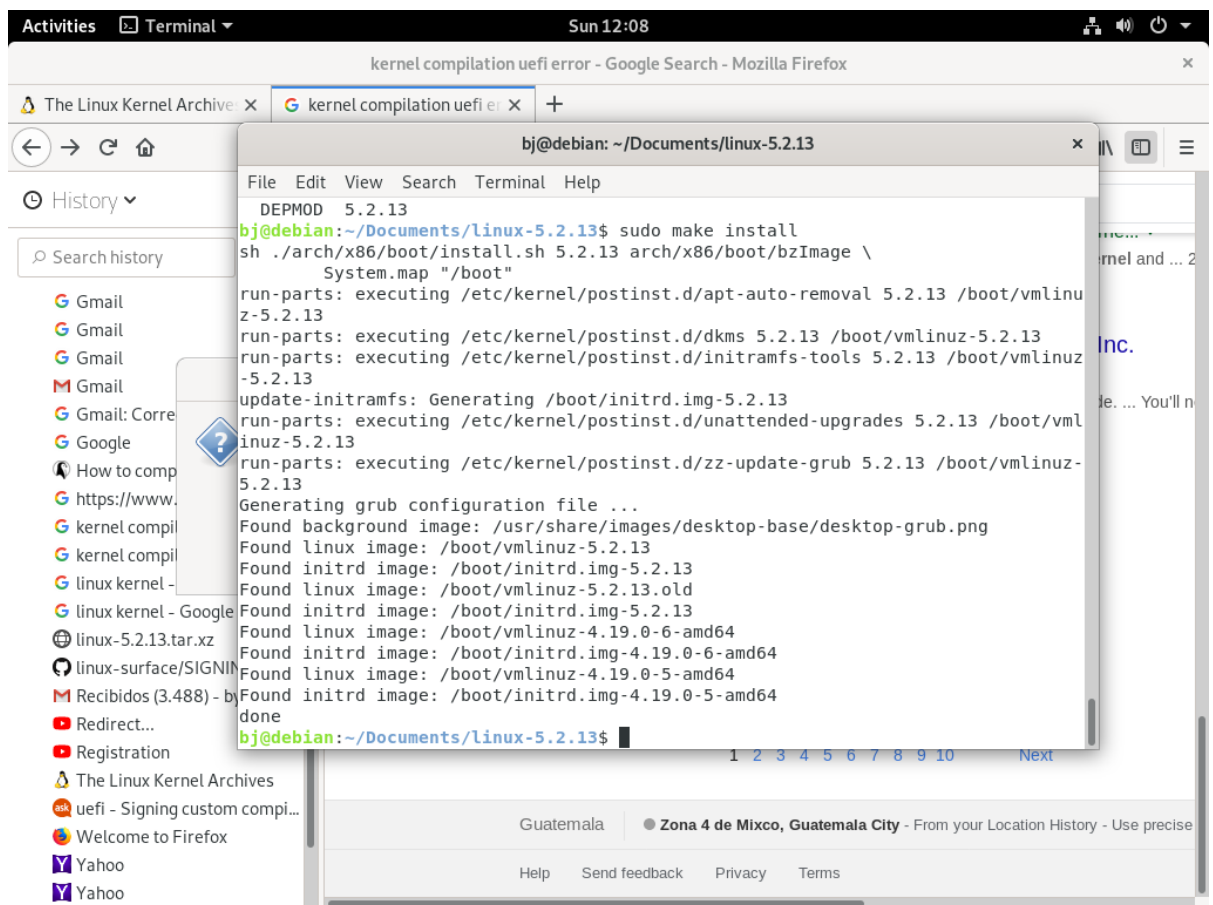
```
$ make modules -j 3
```



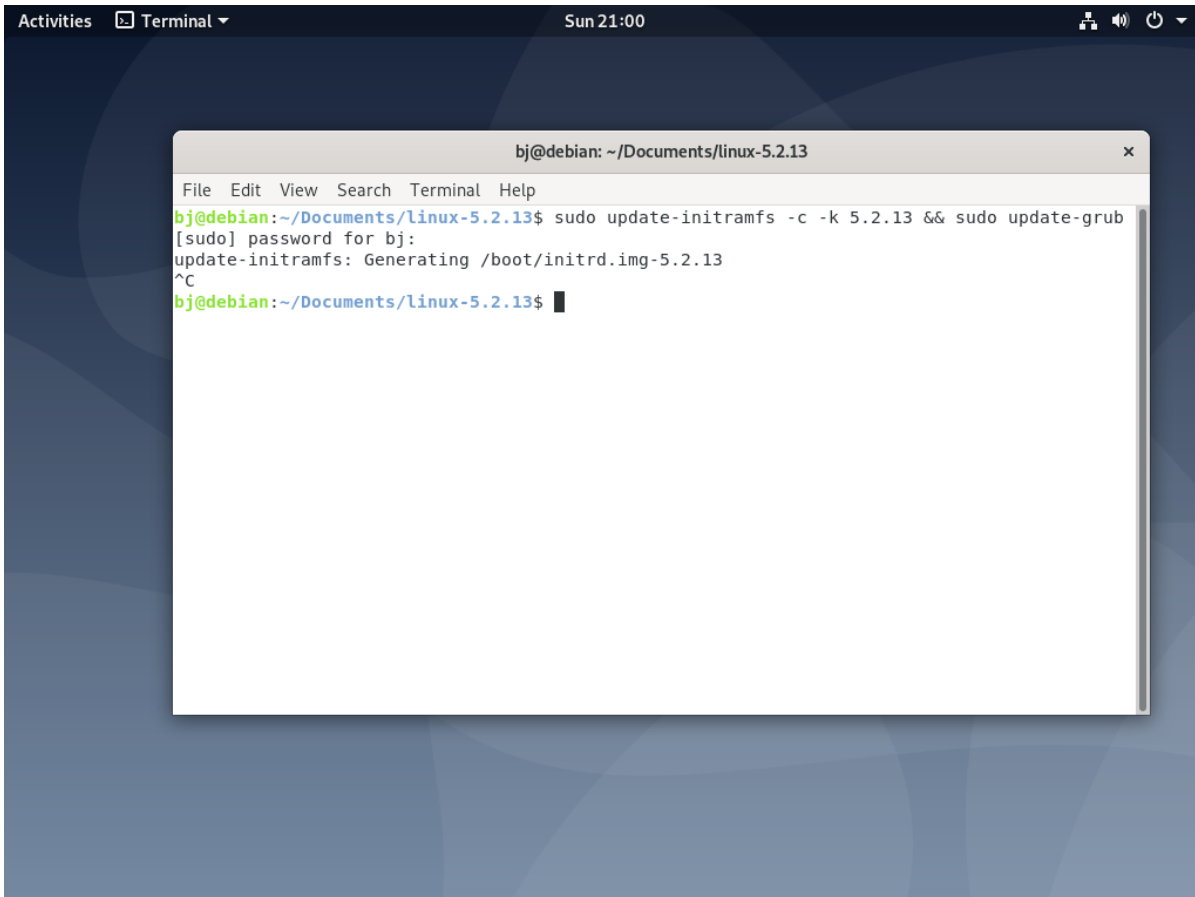
```
$ sudo make modules_install
```



```
$ sudo make install
```



```
$ sudo update-initramfs -c -k 5.2.13 && sudo update-grub
```



A terminal window titled "bj@debian: ~/Documents/linux-5.2.13" is shown. The terminal output is as follows:

```
bj@debian:~/Documents/linux-5.2.13$ sudo update-initramfs -c -k 5.2.13 && sudo update-grub
[sudo] password for bj:
update-initramfs: Generating /boot/initrd.img-5.2.13
^C
bj@debian:~/Documents/linux-5.2.13$
```

9. Reiniciar la VM y se podra ver en el GRUB el nuevo kernel.



The GRUB boot menu is displayed, showing the following entries:

```
GNU GRUB version 2.02+dfsg1-20

*Debian GNU/Linux, with Linux 5.2.13
Debian GNU/Linux, with Linux 5.2.13 (recovery mode)
Debian GNU/Linux, with Linux 4.19.0-6-amd64
Debian GNU/Linux, with Linux 4.19.0-6-amd64 (recovery mode)
Debian GNU/Linux, with Linux 4.19.0-5-amd64
Debian GNU/Linux, with Linux 4.19.0-5-amd64 (recovery mode)
```

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line. ESC to return previous
menu.

Modulos para el kernel

Por ultimo se desarrollaron dos modulos para poder montarlos en el kernel. Para poder crear modulos se necesitan de los Headers de linux, por lo que se descargan de la siguiente forma.

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

Modulo CPU

Este modulo sobreescribira un archivo en /proc con mi informacion y informacion de los procesos del sistema. El codigo del modulo es el siguiente.

```
#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/proc_fs.h>
#include <linux/utsname.h>
#include <linux/seq_file.h>
#include <asm/uaccess.h>
#include <linux/sched/signal.h>
#include <linux/sched.h>

#define PROCFS_NAME "cpu_201222626"
#define EJECUCION "\tEjecutandose\n"
#define DURMIENDO "\tDurmiendo\n"
#define PARADO "\tParado\n"
#define MUERTO "\tMuerto\n"
#define OTRO "\tOtro\n"
#define ZOMBI "\tZombie\n"
#define IDLE "\tIdle\n"
static void estado(struct seq_file *m, u32 est);

static int cpumod_show(struct seq_file *m, void *v){
    seq_printf(m, "Carné: 201222626\n");
    seq_printf(m, "Nombre: Byron Jose Lopez Herrera\n");
    seq_printf(m, "Sistema Operativo: %s %s %s\n",
                utsname()->sysname, utsname()->release, utsname()->version);
    seq_printf(m, "PID\tNombre\tEstado\n");

    struct task_struct *task;
    rcu_read_lock();

    for_each_process(task){
        task_lock(task);
        seq_printf(m, "%i\t%s", task->pid, task->comm);
        estado (m, task->state);
        task_unlock(task);
    }
    rcu_read_unlock();
    return 0;
}
```

```
static void estado(struct seq_file *m, u32 est){
    switch(est){
    case TASK_RUNNING:
        seq_printf(m, EJECUCION);
        break;
    case TASK_INTERRUPTIBLE:
    case TASK_UNINTERRUPTIBLE:
        seq_printf(m, DURMIENDO);
        break;
    case __TASK_STOPPED:
    case __TASK_TRACED:
    case TASK_STOPPED:
        seq_printf(m, PARADO);
        break;
    case EXIT_ZOMBIE:
        seq_printf(m, ZOMBI);
        break;
    case TASK_DEAD:
        seq_printf(m, MUERTO);
        break;
    case TASK_IDLE:
        seq_printf(m, IDLE);
        break;
    default:
        seq_printf(m, OTRO);
        break;
    }
}

static int cpumod_open(struct inode *inode, struct file *file){
    return single_open(file, cpumod_show, NULL);
}

static const struct file_operations cpu_fops= {
    .owner = THIS_MODULE,
    .open = cpumod_open,
    .read = seq_read,
    .llseek = seq_lseek,
    .release = single_release,
};

static int __init cpumod_init(void){
    printk(KERN_INFO "Byron Jose Lopez Herrera\n");
    proc_create(PROCFS_NAME, 0, NULL, &cpu_fops);
    printk(KERN_INFO "Completado. Proceso: /proc/%s.\n", PROCFS_NAME);

    return 0;
}

static void __exit cpumod_exit(void){
    remove_proc_entry(PROCFS_NAME, NULL);
    printk(KERN_INFO "Sistemas Operativos 1\n");
}
```

```
    printk(KERN_INFO "Modulo deshabilitado\n");
}

module_init(cpumod_init);
module_exit(cpumod_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Dragonsor");
MODULE_DESCRIPTION("Modulo realizado como practica de sistemas operativos1");
```

El codigo del archivo make es el siguiente.

```
obj-m += cpu_201222626.o
all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Modulo de memoria

Este modulo desplegara informacion personal y informacion de la memoria ram.El codigo del modulo es el siguiente.

```
#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/proc_fs.h>
#include <linux/seq_file.h>
#include <asm/uaccess.h>
#include <linux/hugetlb.h>
#include <linux/mm.h>
#include <linux/mman.h>
#include <linux/mmzone.h>
#include <linux/quicklist.h>
#include <linux/swap.h>
#include <linux/vmstat.h>
#include <linux/atomic.h>
#include <asm/page.h>
#include <asm/pgtable.h>
#include <linux/utsname.h>
#include <linux/vmalloc.h>
#include <linux/percpu.h>
#ifdef CONFIG_CMA
#include <linux/cma.h>
```

```

#endif

#define PROCFS_NAME "mem_201222626"

static int memoria_show(struct seq_file *m, void *v){
    struct sysinfo i;

    unsigned long pages[NR_LRU_LISTS];

    int lru;

#define K(x) ((x)<<(PAGE_SHIFT - 10))
    si_meminfo(&i);

    for(lru = LRU_BASE; lru < NR_LRU_LISTS; lru++){
        pages[lru] = global_node_page_state(NR_LRU_BASE + lru);
    }

    unsigned long total;
    unsigned long libre;
    total = K(i.totalram);
    libre = K(i.freeram);
    unsigned long prc ;
    unsigned long usado = total - libre;
    prc = usado * 10000 /total;
    unsigned long prc2 = usado * 100 / total;
    unsigned long dec = (prc - prc2 * 100);
    /**
     * total - 100
     * usado - x
     * */

    seq_printf(m, "Carné: 201222626\n");
    seq_printf(m, "Nombre: Byron Jose Lopez Herrera\n");
    seq_printf(m, "Sistema Operativo:%s %s %s\n",utsname()-
>sysname,utsname()->release, utsname()->version);
    seq_printf(m, "MemoriaTotal: %8lu kB\n", total);
    seq_printf(m, "MemoriaLibre: %8lu kB\n", libre);
    seq_printf(m, "MemoriaUsada: %lu.%lu %%\n", prc2, dec);

    return 0;
}

static int memoria_open(struct inode *inode, struct file *file){
    return single_open(file, memoria_show, NULL);
}

static const struct file_operations memoria_fops = {
    .owner = THIS_MODULE,
    .open = memoria_open,
    .read = seq_read,
    .llseek = seq_lseek,

```

```

    .release = single_release,
};

static int __init memoria_init(void){
    printk(KERN_INFO "201222626\n");
    proc_create(PROCFS_NAME, 0, NULL, &memoria_fops);
    printk(KERN_INFO "Completado. Proceso: /proc/%s.\n", PROCFS_NAME);
    return 0;
}

static void __exit memoria_exit(void){
    remove_proc_entry(PROCFS_NAME, NULL);
    printk(KERN_INFO "Sistemas Operativos 1\n");
    printk(KERN_INFO "Modulo deshabilitado\n");
}

module_init(memoria_init);
module_exit(memoria_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("dragonsor");
MODULE_DESCRIPTION("Modulo realizado como practica dos de sistemas
operativos1");

```

El codigo del archivo make es el siguiente.

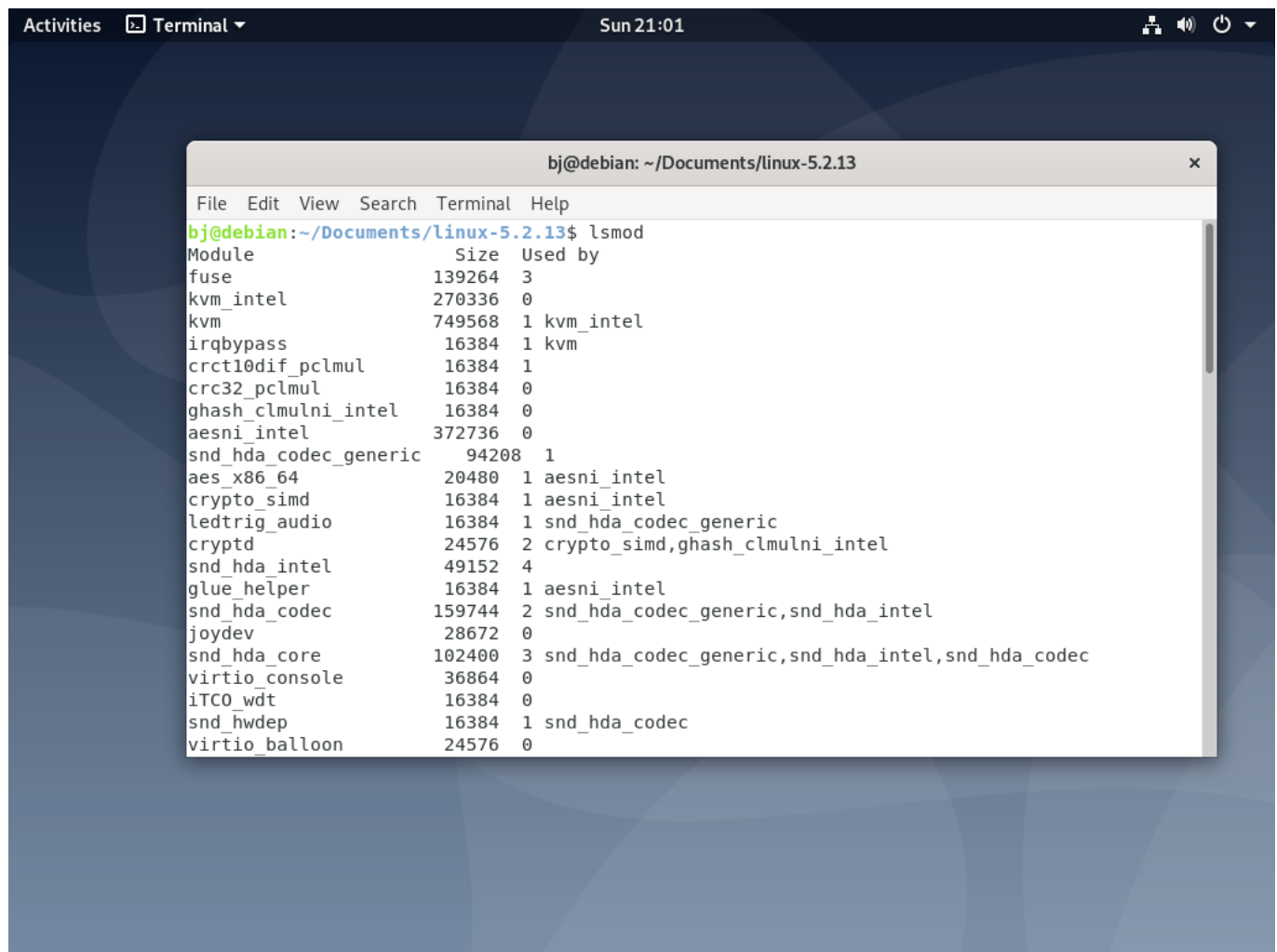
```

obj-m += mem_201222626.o
all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean

```

Pasos para compilar los modulos,cargarlos al kernel y removerlos del kernel.

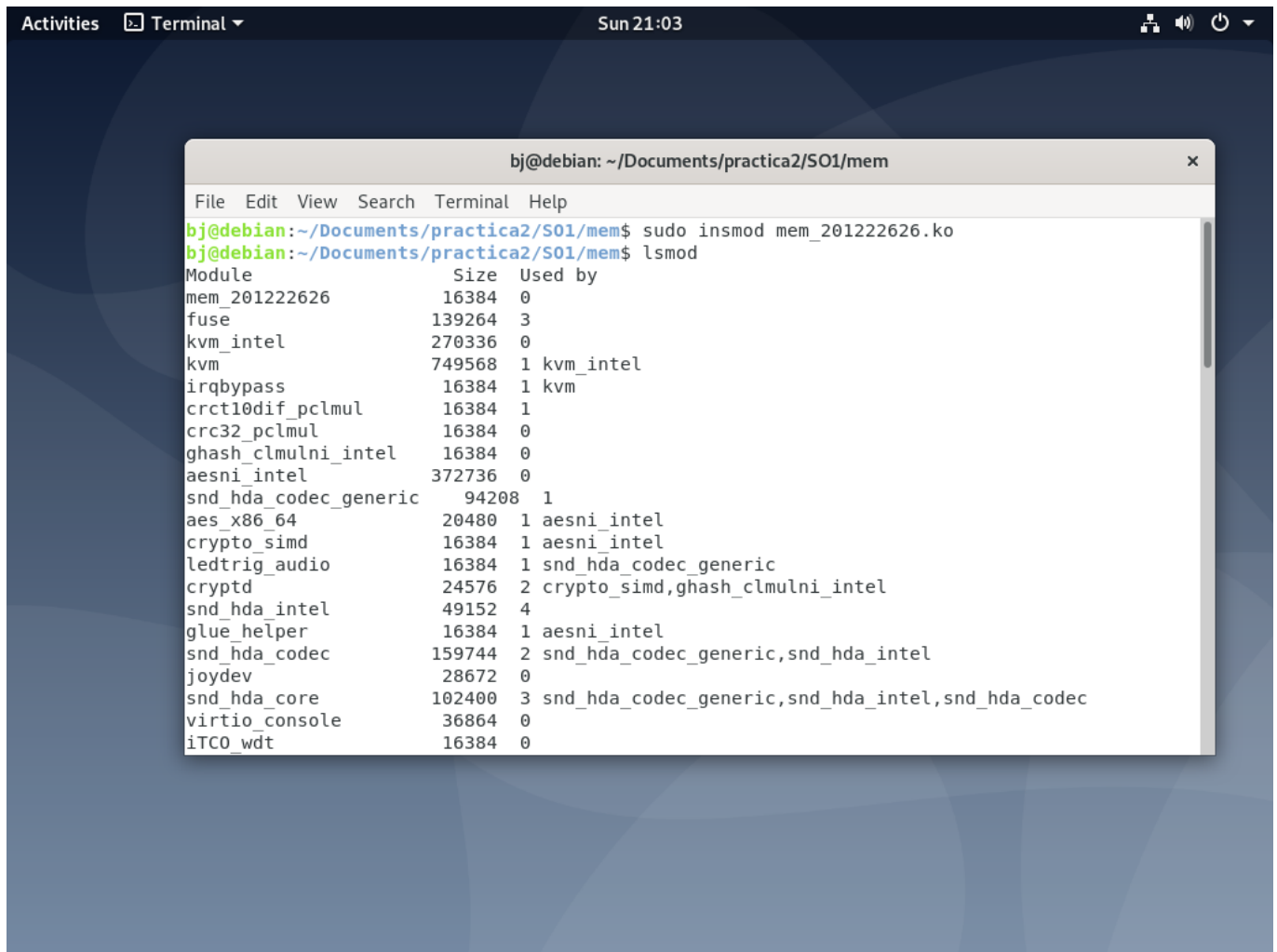
1. Para compilar cada modulo solo basta con ejecutar el archivo make lo que generara los respectivos archivos de salida. En la siguiente imagen se observa que los modulos haun no se ahan cargado.



```
bj@debian: ~/Documents/linux-5.2.13$ lsmod
Module                  Size  Used by
fuse                    139264  3
kvm_intel               270336  0
kvm                     749568  1 kvm_intel
irqbypass              16384  1 kvm
crct10dif_pclmul       16384  1
crc32_pclmul           16384  0
ghash_clmulni_intel    16384  0
aesni_intel            372736  0
snd_hda_codec_generic  94208  1
aes_x86_64             20480  1 aesni_intel
crypto_simd            16384  1 aesni_intel
ledtrig_audio          16384  1 snd_hda_codec_generic
cryptd                 24576  2 crypto_simd,ghash_clmulni_intel
snd_hda_intel          49152  4
glue_helper            16384  1 aesni_intel
snd_hda_codec          159744  2 snd_hda_codec_generic,snd_hda_intel
joydev                 28672  0
snd_hda_core           102400  3 snd_hda_codec_generic,snd_hda_intel,snd_hda_codec
virtio_console         36864  0
iTCO_wdt               16384  0
snd_hwdep              16384  1 snd_hda_codec
virtio_balloon         24576  0
```

2. Para cargar el modulo al kernel se utiliza el siguiente comando.

```
$ sudo insmod mem_201222626.ko
$ sudo insmod cpu_201222626.ko
```

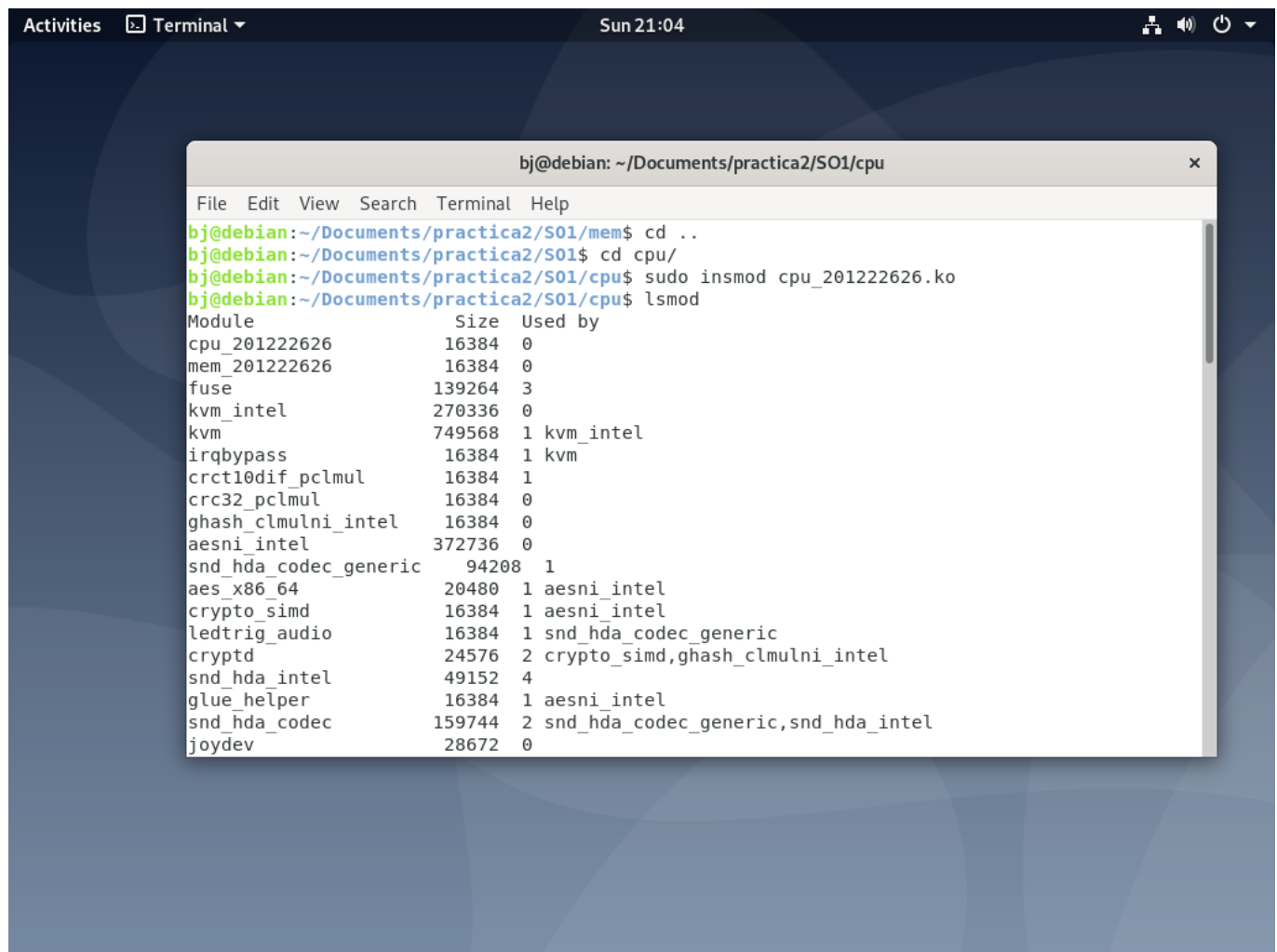


The screenshot shows a terminal window titled "bj@debian: ~/Documents/practica2/S01/mem". The user has executed the command `sudo insmod mem_201222626.ko` and then `lsmod`. The output of `lsmod` is a table listing loaded kernel modules, their sizes, and the modules they are used by.

Module	Size	Used by
mem_201222626	16384	0
fuse	139264	3
kvm_intel	270336	0
kvm	749568	1 kvm_intel
irqbypass	16384	1 kvm
crct10dif_pclmul	16384	1
crc32_pclmul	16384	0
ghash_clmulni_intel	16384	0
aesni_intel	372736	0
snd_hda_codec_generic	94208	1
aes_x86_64	20480	1 aesni_intel
crypto_simd	16384	1 aesni_intel
ledtrig_audio	16384	1 snd_hda_codec_generic
cryptd	24576	2 crypto_simd,ghash_clmulni_intel
snd_hda_intel	49152	4
glue_helper	16384	1 aesni_intel
snd_hda_codec	159744	2 snd_hda_codec_generic,snd_hda_intel
joydev	28672	0
snd_hda_core	102400	3 snd_hda_codec_generic,snd_hda_intel,snd_hda_codec
virtio_console	36864	0
itco_wdt	16384	0

3. Luego de cargarlos ya se pueden observar en la lista de modulos y la salida cuando se cargaron.

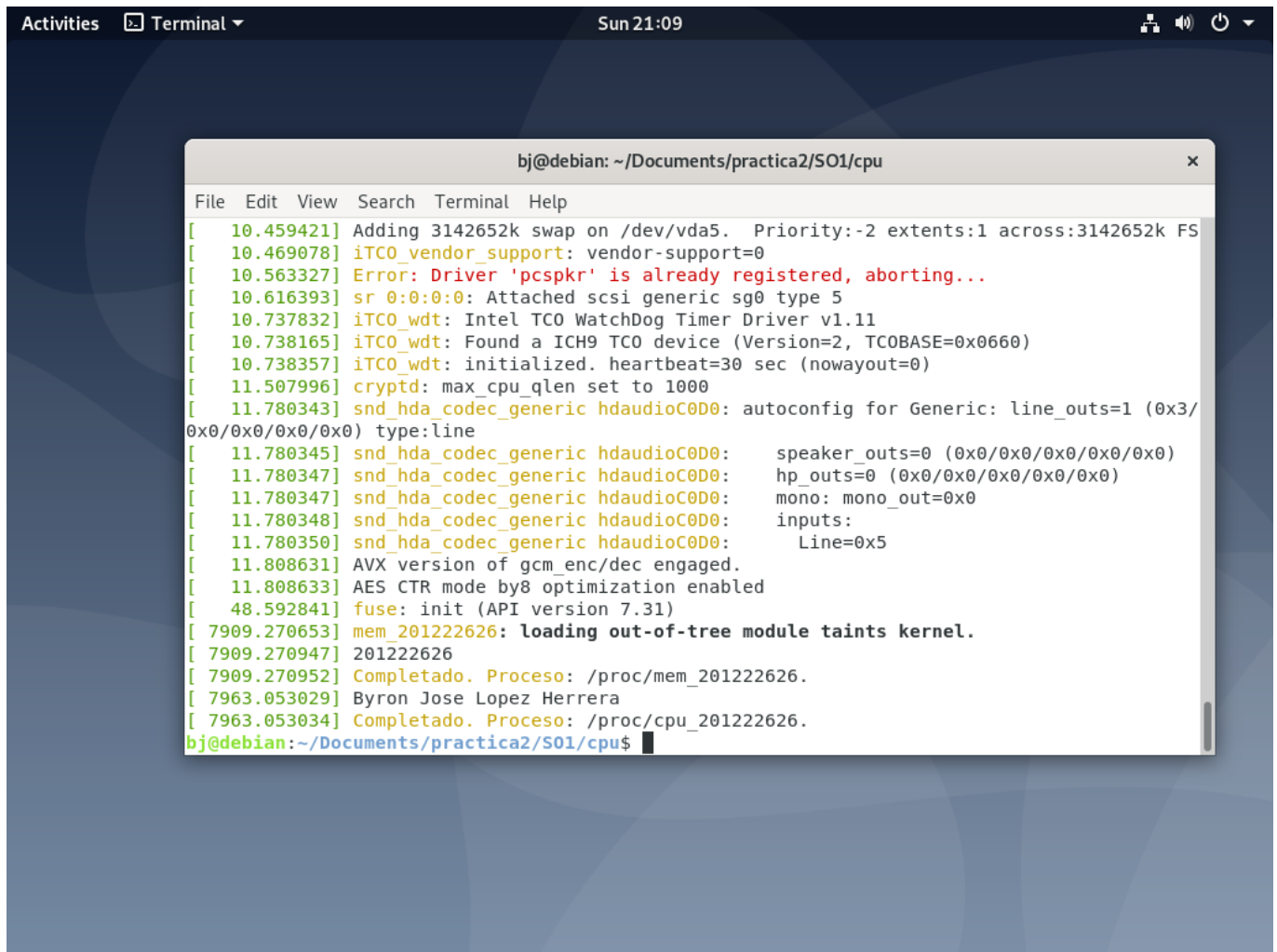
```
$ sudo lsmod
$ sudo insmod cpu_201222626.ko
```



The screenshot shows a terminal window titled "bj@debian: ~/Documents/practica2/SO1/cpu". The user navigates from a parent directory to the "cpu" subdirectory and then uses `sudo insmod` to load the `cpu_201222626.ko` module. Finally, the `lsmod` command is executed, displaying a list of currently loaded kernel modules, their sizes, and the modules they are used by.

```
bj@debian:~/Documents/practica2/SO1/mem$ cd ..
bj@debian:~/Documents/practica2/SO1$ cd cpu/
bj@debian:~/Documents/practica2/SO1/cpu$ sudo insmod cpu_201222626.ko
bj@debian:~/Documents/practica2/SO1/cpu$ lsmod
```

Module	Size	Used by
cpu_201222626	16384	0
mem_201222626	16384	0
fuse	139264	3
kvm_intel	270336	0
kvm	749568	1 kvm_intel
irqbypass	16384	1 kvm
crct10dif_pclmul	16384	1
crc32_pclmul	16384	0
ghash_clmulni_intel	16384	0
aesni_intel	372736	0
snd_hda_codec_generic	94208	1
aes_x86_64	20480	1 aesni_intel
crypto_simd	16384	1 aesni_intel
ledtrig_audio	16384	1 snd_hda_codec_generic
cryptd	24576	2 crypto_simd,ghash_clmulni_intel
snd_hda_intel	49152	4
glue_helper	16384	1 aesni_intel
snd_hda_codec	159744	2 snd_hda_codec_generic,snd_hda_intel
joydev	28672	0

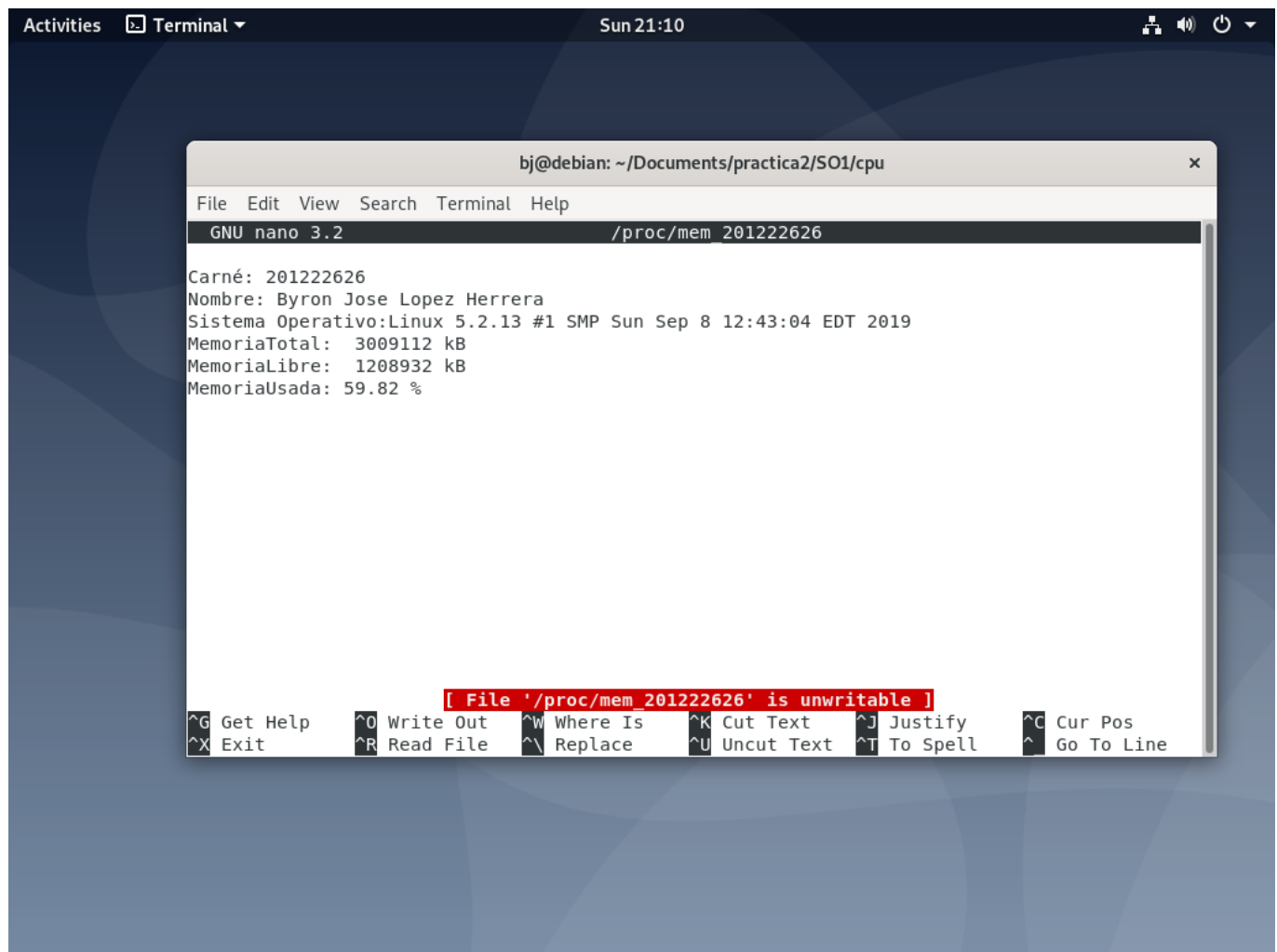


```
bj@debian: ~/Documents/practica2/SO1/cpu
File Edit View Search Terminal Help
[ 10.459421] Adding 3142652k swap on /dev/vda5. Priority:-2 extents:1 across:3142652k FS
[ 10.469078] iTCO_vendor_support: vendor-support=0
[ 10.563327] Error: Driver 'pcspkr' is already registered, aborting...
[ 10.616393] sr 0:0:0:0: Attached scsi generic sg0 type 5
[ 10.737832] iTCO_wdt: Intel TCO WatchDog Timer Driver v1.11
[ 10.738165] iTCO_wdt: Found a ICH9 TCO device (Version=2, TCOBASE=0x0660)
[ 10.738357] iTCO_wdt: initialized. heartbeat=30 sec (nowayout=0)
[ 11.507996] cryptd: max_cpu_glen set to 1000
[ 11.780343] snd_hda_codec_generic hdaudioC0D0: autoconfig for Generic: line_outs=1 (0x3/
0x0/0x0/0x0/0x0) type:line
[ 11.780345] snd_hda_codec_generic hdaudioC0D0: speaker_outs=0 (0x0/0x0/0x0/0x0/0x0)
[ 11.780347] snd_hda_codec_generic hdaudioC0D0: hp_outs=0 (0x0/0x0/0x0/0x0/0x0)
[ 11.780347] snd_hda_codec_generic hdaudioC0D0: mono: mono_out=0x0
[ 11.780348] snd_hda_codec_generic hdaudioC0D0: inputs:
[ 11.780350] snd_hda_codec_generic hdaudioC0D0: Line=0x5
[ 11.808631] AVX version of gcm_enc/dec engaged.
[ 11.808633] AES CTR mode by8 optimization enabled
[ 48.592841] fuse: init (API version 7.31)
[ 7909.270653] mem_201222626: loading out-of-tree module taints kernel.
[ 7909.270947] 201222626
[ 7909.270952] Completado. Proceso: /proc/mem_201222626.
[ 7963.053029] Byron Jose Lopez Herrera
[ 7963.053034] Completado. Proceso: /proc/cpu_201222626.
bj@debian:~/Documents/practica2/SO1/cpu$
```

4. Para remover un modulo es el siguiente comando.

```
$ sudo rmmod mem_201222626
```

5. La salidas de los modulos son los siguientes.



The screenshot shows a terminal window with a nano editor open. The editor's title bar indicates the file path is `bj@debian: ~/Documents/practica2/SO1/cpu`. The editor's menu bar includes `File Edit View Search Terminal Help`. The status bar at the top of the editor shows `GNU nano 3.2 /proc/mem_201222626`. The file content is as follows:

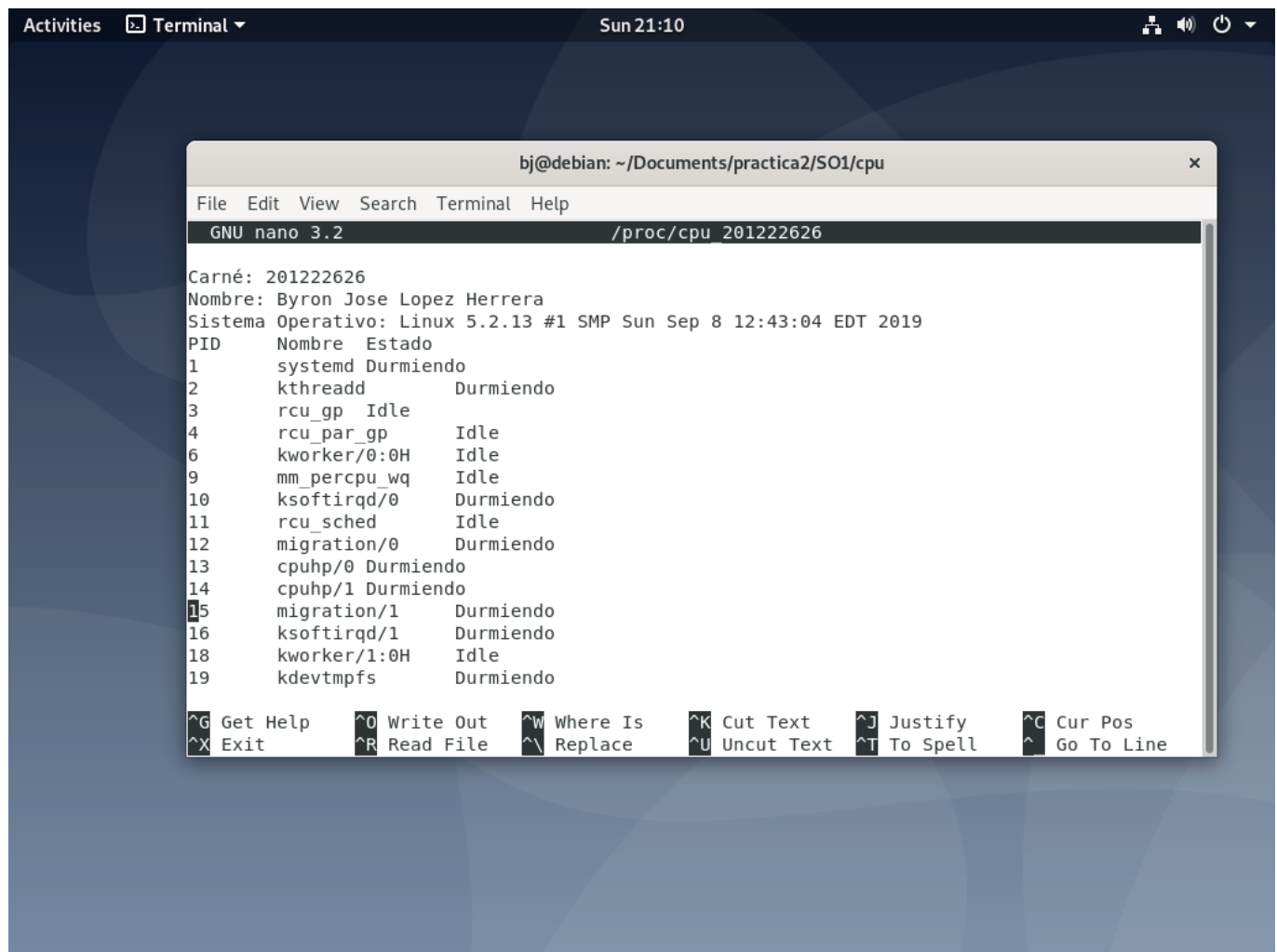
```
Carné: 201222626
Nombre: Byron Jose Lopez Herrera
Sistema Operativo:Linux 5.2.13 #1 SMP Sun Sep 8 12:43:04 EDT 2019
MemoriaTotal: 3009112 kB
MemoriaLibre: 1208932 kB
MemoriaUsada: 59.82 %
```

A red error message is displayed in the center of the editor:

```
[ File '/proc/mem_201222626' is unwritable ]
```

The bottom of the editor shows a grid of keyboard shortcuts:

<code>^G</code> Get Help	<code>^O</code> Write Out	<code>^W</code> Where Is	<code>^K</code> Cut Text	<code>^J</code> Justify	<code>^C</code> Cur Pos
<code>^X</code> Exit	<code>^R</code> Read File	<code>^N</code> Replace	<code>^U</code> Uncut Text	<code>^T</code> To Spell	<code>^_</code> Go To Line



The screenshot shows a terminal window titled "bj@debian: ~/Documents/practica2/SO1/cpu". Inside the terminal, the GNU nano 3.2 editor is open, displaying the contents of the file `/proc/cpu 201222626`. The file contains system information and a list of processes.

```
Carné: 201222626
Nombre: Byron Jose Lopez Herrera
Sistema Operativo: Linux 5.2.13 #1 SMP Sun Sep 8 12:43:04 EDT 2019
PID      Nombre  Estado
1        systemd Durmiendo
2        kthreadd      Durmiendo
3        rcu_gp      Idle
4        rcu_par_gp   Idle
6        kworker/0:0H Idle
9        mm_percpu_wq Idle
10       ksoftirqd/0   Durmiendo
11       rcu_sched    Idle
12       migration/0 Durmiendo
13       cpuhp/0      Durmiendo
14       cpuhp/1      Durmiendo
15       migration/1 Durmiendo
16       ksoftirqd/1   Durmiendo
18       kworker/1:0H Idle
19       kdevtmpfs     Durmiendo
```

The bottom of the nano editor shows the command palette with various shortcuts:

- `^G` Get Help
- `^O` Write Out
- `^W` Where Is
- `^K` Cut Text
- `^J` Justify
- `^C` Cur Pos
- `^X` Exit
- `^R` Read File
- `^N` Replace
- `^U` Uncut Text
- `^T` To Spell
- `^_` Go To Line