

Pedro César Tay Dubón

Carné 200915120

# PROYECTO 1

Sistemas Operativos 1

# Índice

## Contenido

Compilación de Kernel de Fedora .....	2
i.    Instalación de KVM.....	2
ii.   Creación de una máquina virtual .....	4
iii.  Instalación de Fedora 27 .....	9
iv.   Preparación para la compilación de un nuevo Kernel .....	13
v.    Compilación del nuevo Kernel .....	19
vi.   Instalación del nuevo kernel .....	21
vii.  Módulos .....	24
a.  Módulo de memoria RAM.....	24
b.  Módulo de CPU .....	29

# Compilación de Kernel de Fedora

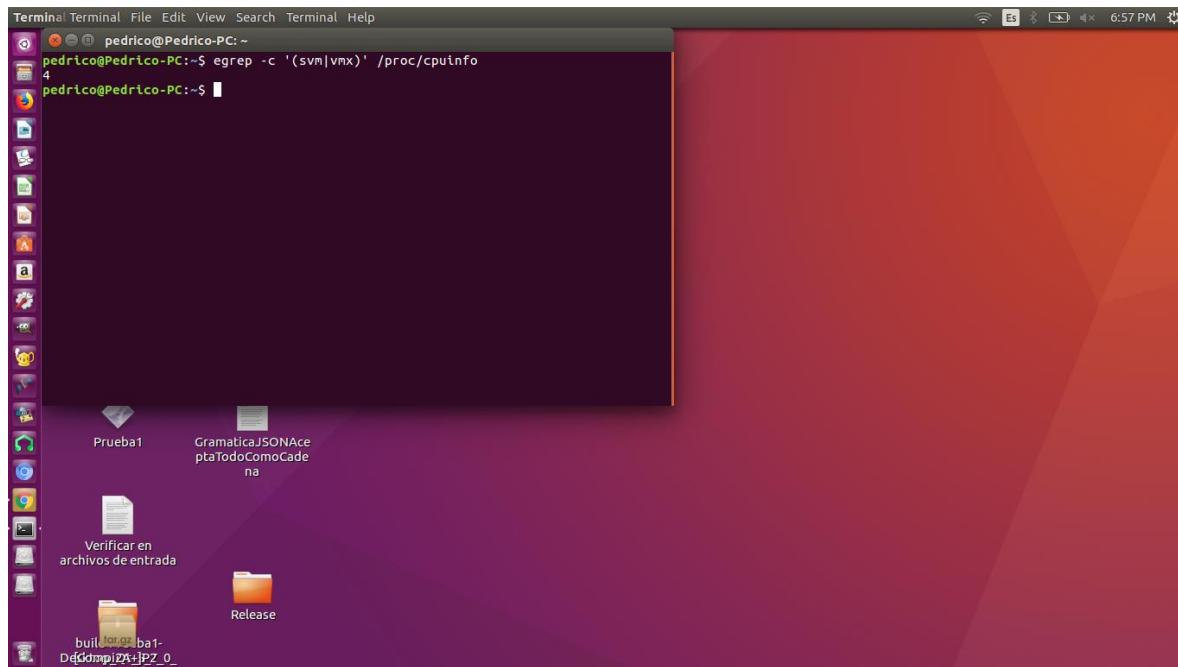
Para esta ocasión se utilizará Fedora 27 como sistema operativo, sobre el cual se compilará un nuevo kernel y posteriormente generar módulos que se instalarán sobre el kernel en tiempo de ejecución.

Para trabajar sobre Fedora 27, primeramente se instalara una versión limpia, y para ello realizaremos una instalación en una máquina virtual de KVM (Kernel-based Virtual Machine). KVM es un hypervisor que nos permite realizar la virtualización de hardware para poder instalar sistemas operativos.

## i. Instalación de KVM

KVM solo funciona en CPU's que tienen virtualización por hardware. Para determinar si nuestro CPU cuenta con esta característica corremos el siguiente comando:

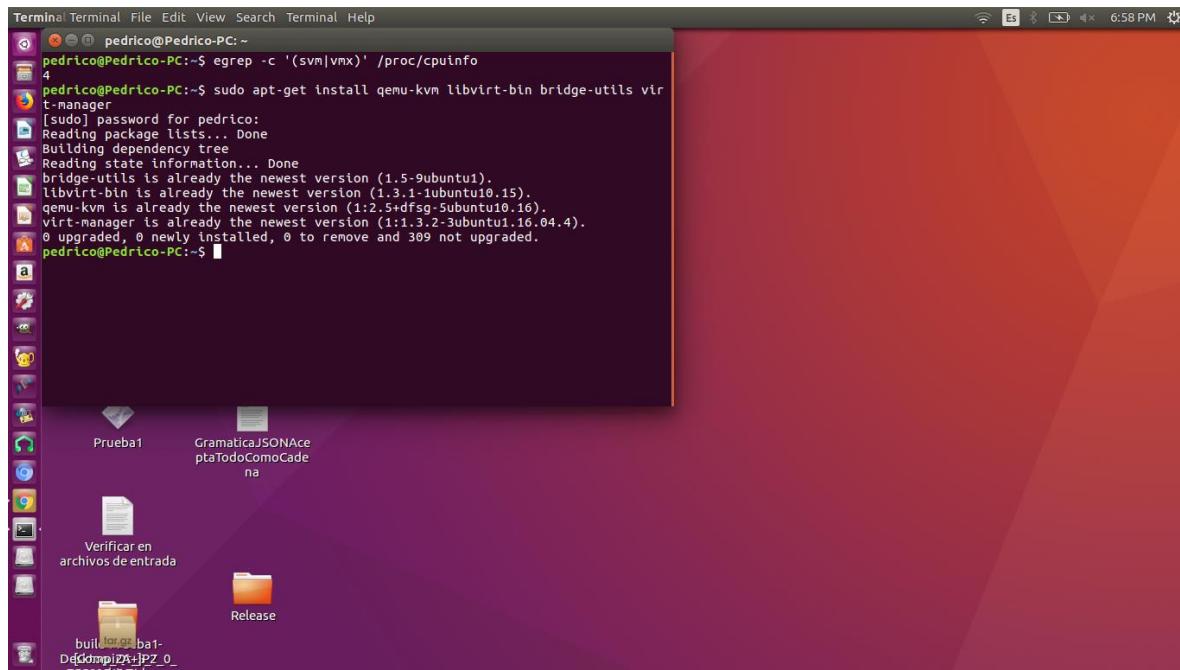
```
egrep -c '(svm|vmx)' /proc/cpuinfo
```



Como se puede observar el resultado es un número, 4 en este caso, lo que indica que contamos con 4 procesadores. Si el resultado fuera 0, no podríamos utilizar KVM para virtualizar.

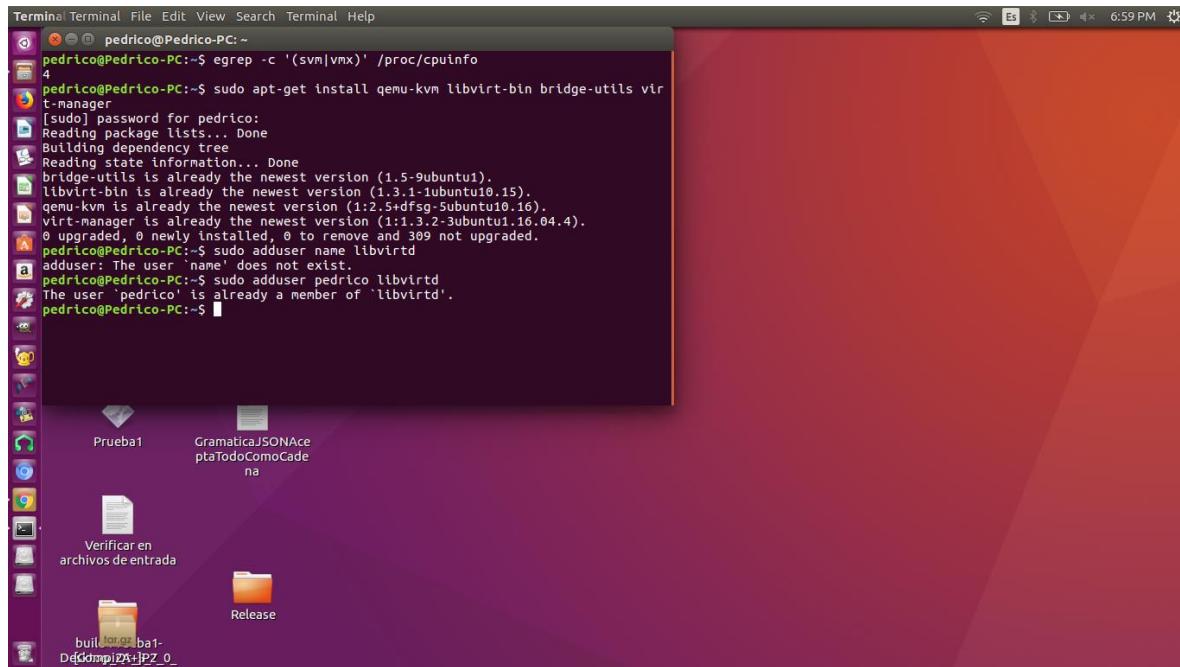
El siguiente comando lo utilizamos para instalar KVM y los paquetes de soporte. Virt-Manager es una aplicación grafica para administrar las máquinas virtuales.

```
sudo apt-get install qemu-kvm libvirt-bin bridge-utils virt-manager
```



Solo el usuario root y los usuarios en el grupo libvirt tienen permiso para usar las maquinas virtuales de KVM. El siguiente comando es para agregar un usuario al grupo libvirt.

```
sudo adduser pedrico libvирtd
```



Después de correr el comando anterior, nos deslogueamos y volvemos a loguearnos. Corremos el siguiente comando y deberíamos de ver una lista vacía de máquinas virtuales. Esto nos indica que todo ha sido instalado correctamente.

```
virsh -c qemu:///system list
```

The screenshot shows a Linux desktop environment with a terminal window open in the foreground. The terminal window title is 'Terminal' and it contains the following command history:

```
Terminal Terminal File Edit View Search Terminal Help
pedrico@Pedrico-PC:~$ egrep -c '(svm|vmx)' /proc/cpuinfo
4
pedrico@Pedrico-PC:~$ sudo apt-get install qemu-kvm libvirt-bin bridge-utils virt-manager
[sudo] password for pedrico:
Reading package lists... Done
Building dependency tree
Reading state information... Done
bridge-utils is already the newest version (1:1.5.9ubuntu1).
libvirt-bin is already the newest version (1:3.1.1-1ubuntu10.15).
qemu-kvm is already the newest version (1:2.5+dfsg-5ubuntu10.16).
virt-manager is already the newest version (1:1.3.2-3ubuntu1.16.04.4).
0 upgraded, 0 newly installed, 0 to remove and 309 not upgraded.
pedrico@Pedrico-PC:~$ sudo adduser name libvirtd
adduser: The user 'name' does not exist.
pedrico@Pedrico-PC:~$ sudo adduser pedrico libvirtd
The user 'pedrico' is already a member of 'libvirtd'.
pedrico@Pedrico-PC:~$ virsh -c qemu:///system list
Id  Name          State
-----
```

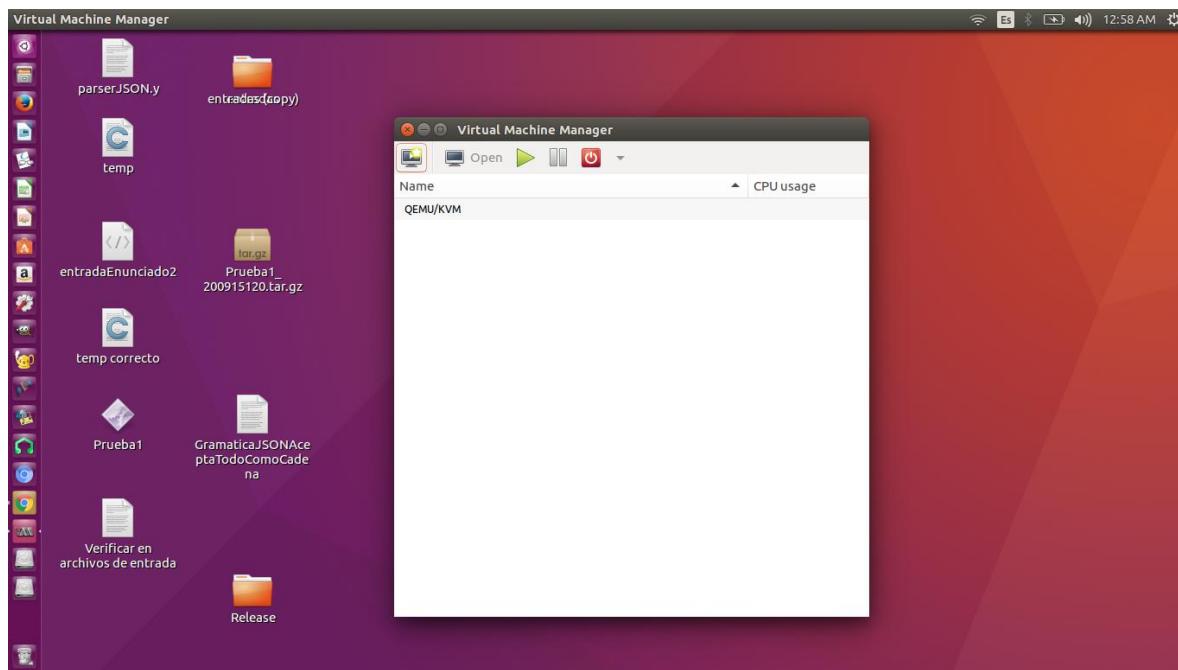
Below the terminal, a file manager window is visible, showing several files and folders on the desktop:

- Prueba1
- GramaticaJSONAcceptorTodoComoCadena
- Verificar en archivos de entrada
- Release
- build/binary1-DedempiaAHPZ\_0

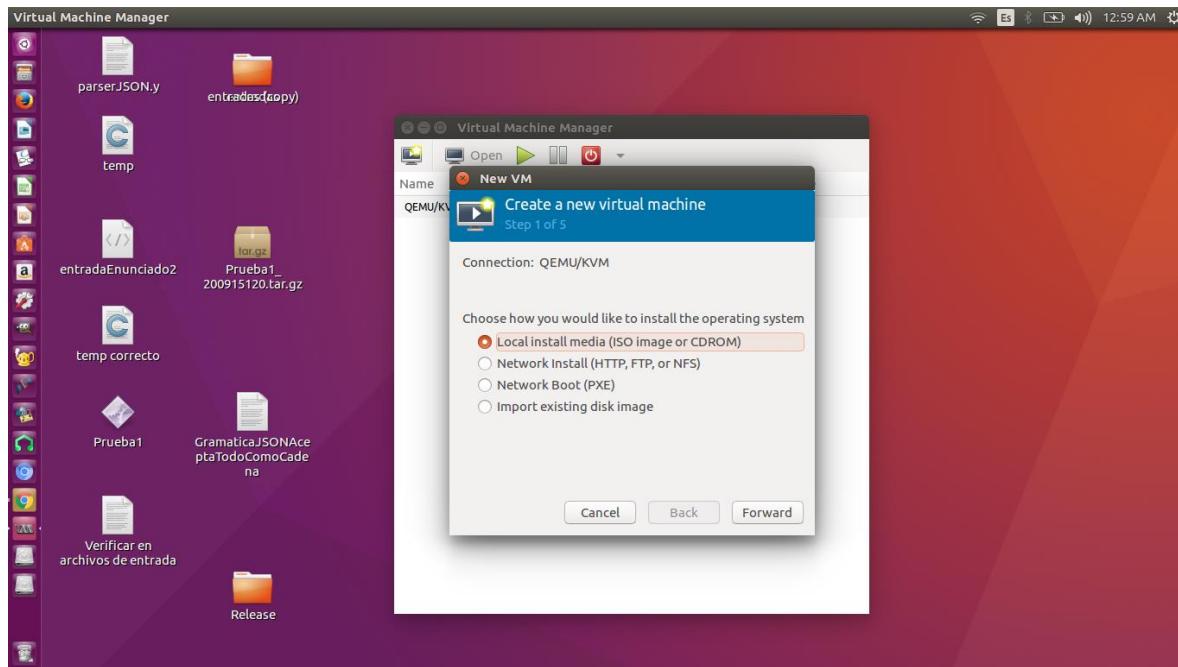
## ii. Creación de una máquina virtual

Tecleamos “virt” en el buscador de aplicaciones e iniciamos Virtual Machine Manager. Una vez abierto seguimos los siguientes pasos.

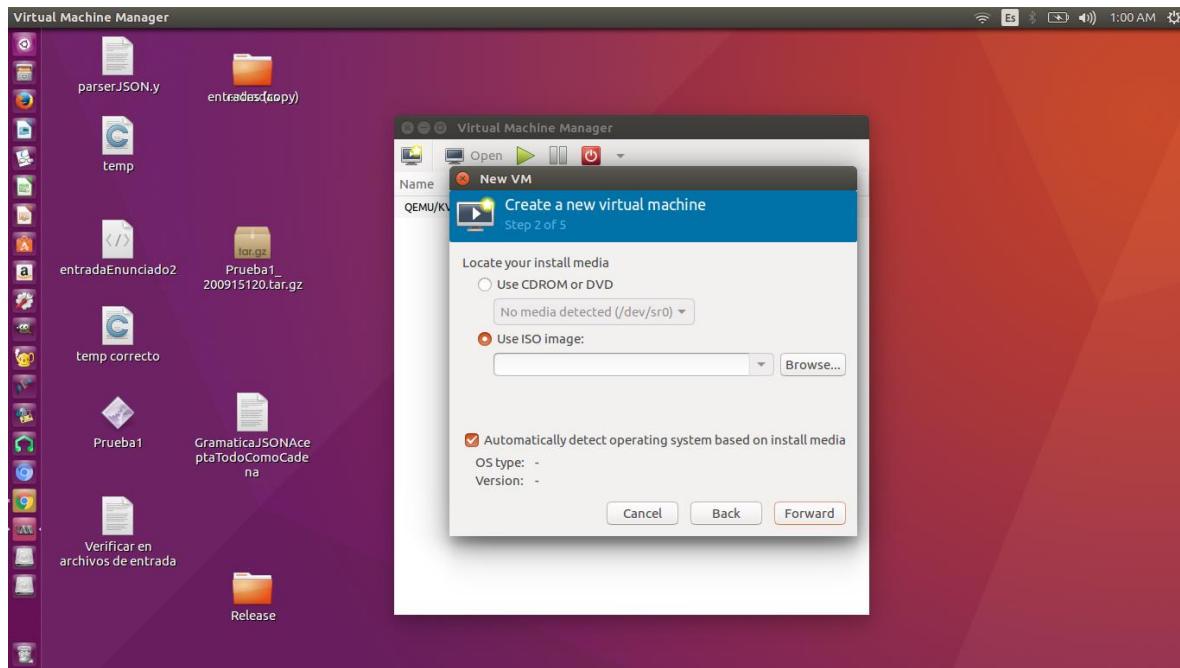
1. Clic en Nueva Máquina Virtual.



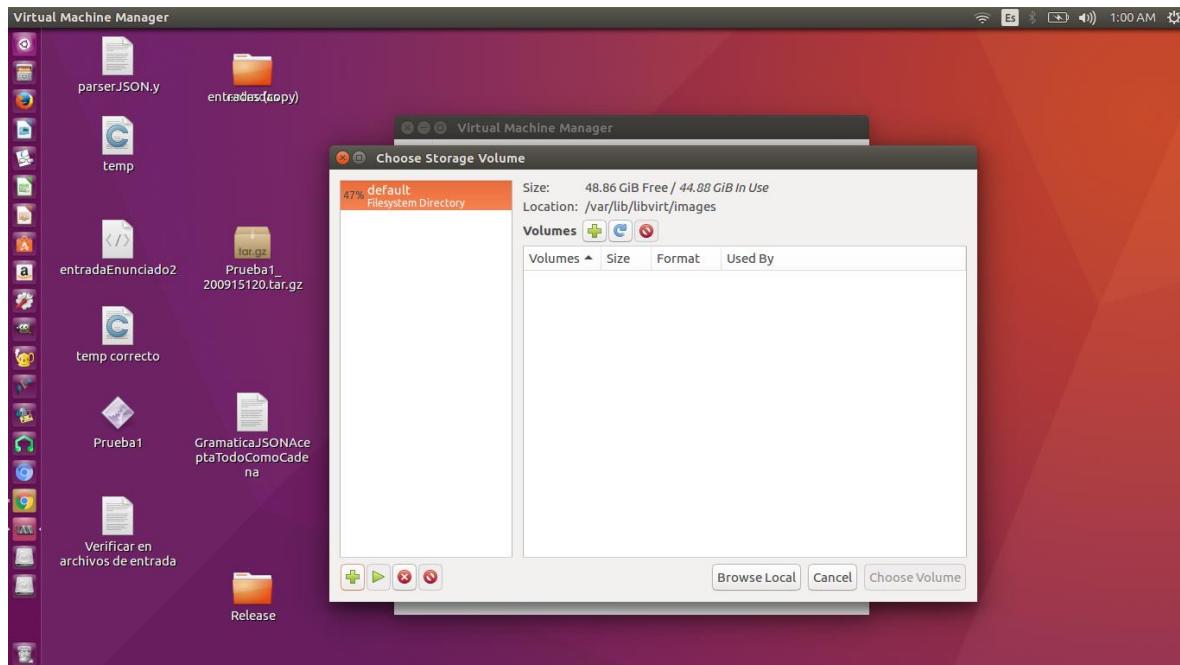
2. En el cuadro de dialogo seleccionamos: Local install media (ISO image or CDROM)



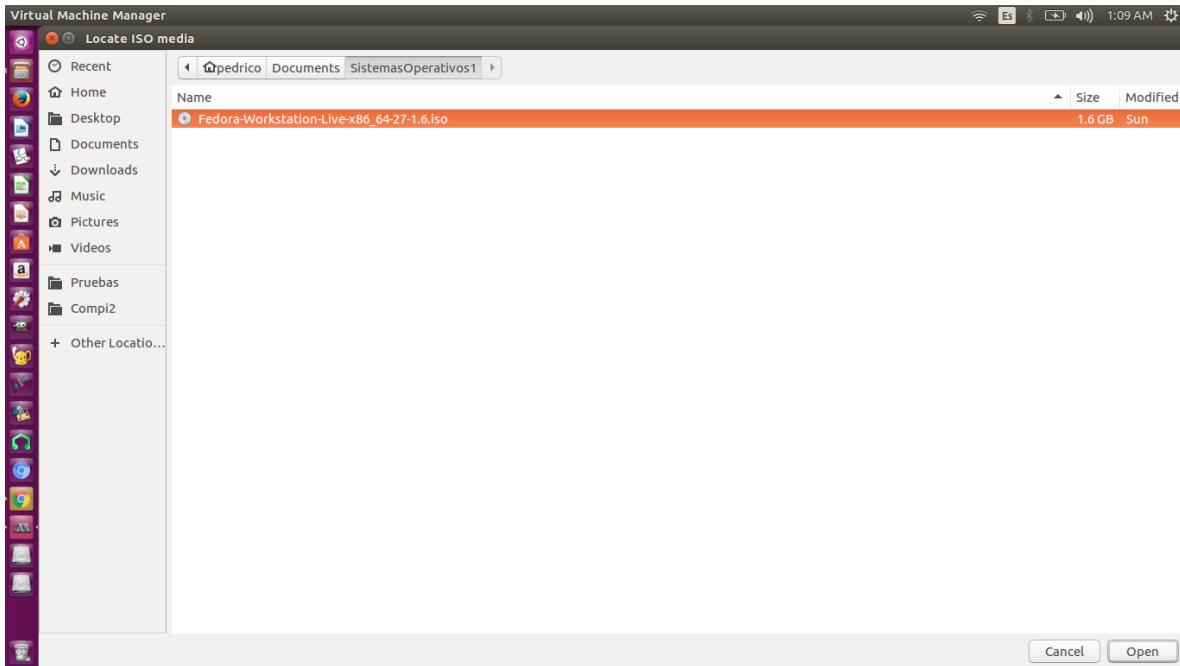
3. Seleccionar: Use ISO image. Click en el botón Browse.



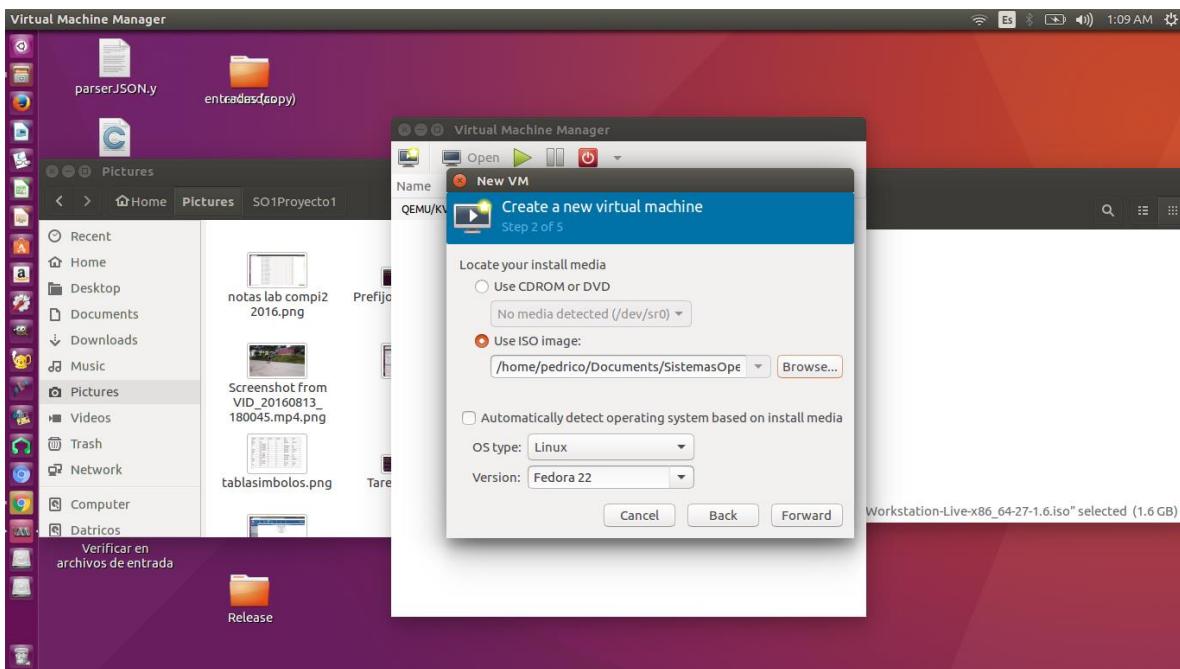
4. En el cuadro de dialogo “Choose Storage Volume” clicamos el botón “Browse Local”.



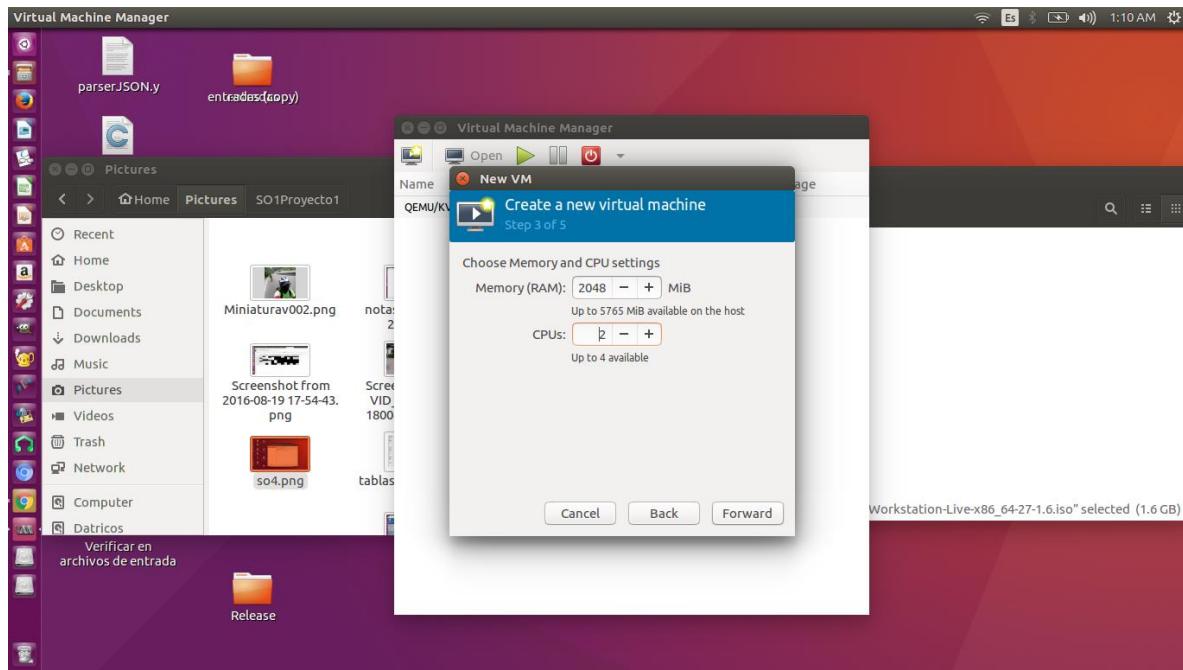
5. Ubicamos el archivo ISO del sistema operativo. En este caso es el ISO de Fedora 27 Workstation y lo seleccionamos.



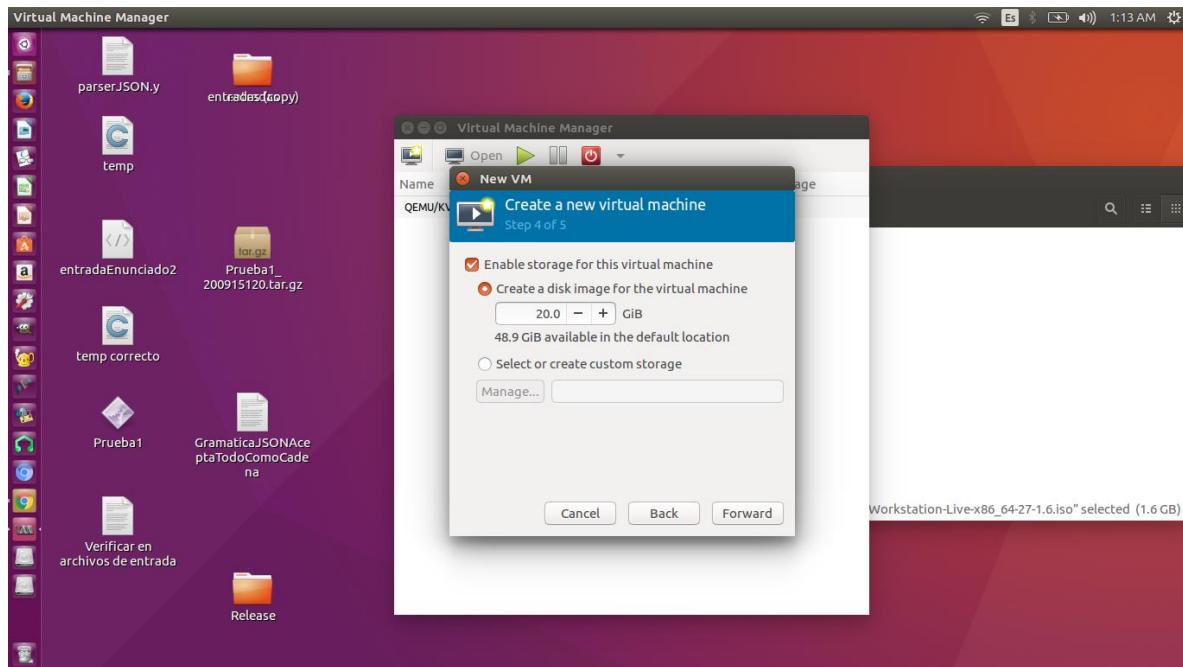
6. Dejamos chequeada la opción “Automatically detect operating system based on install media”: os type: generic, Version: generic. Clicamos el botón “Forward”



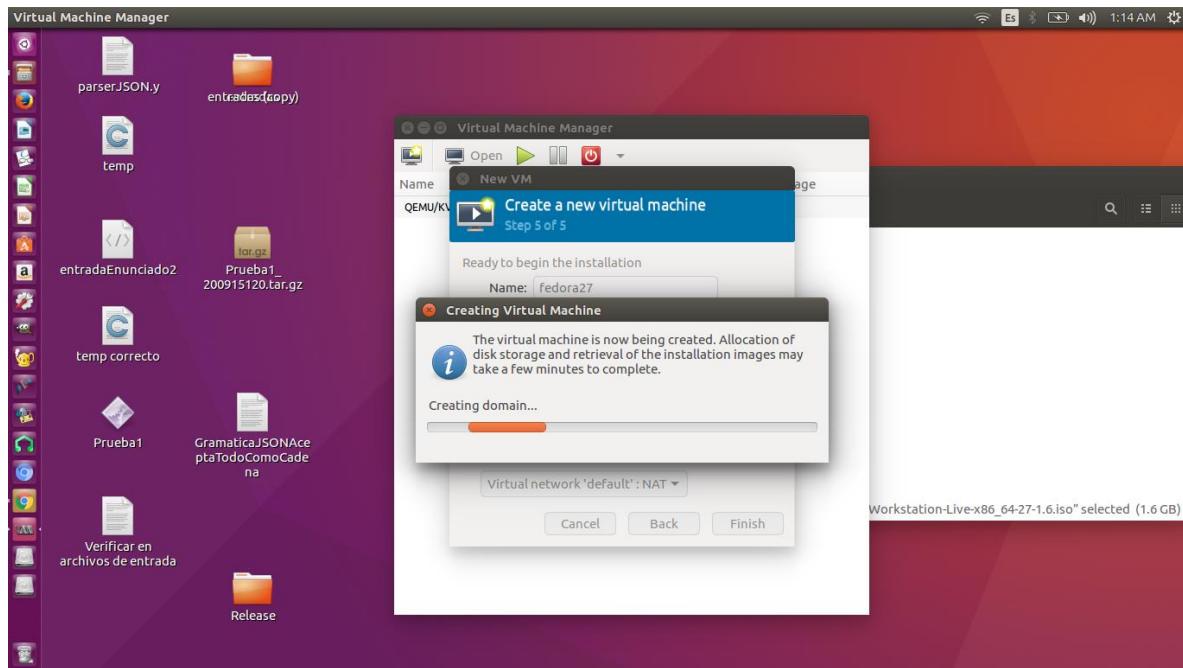
7. Elegimos la cantidad de RAM que vamos a asignarle a la máquina virtual y la cantidad de procesadores. Es conveniente dejarle a la maquina física al menos 1.5 GB de RAM y 2 procesadores. Clicamos el botón “Forward”



8. Asignamos la capacidad de almacenamiento. En este caso le asignaremos 40GB ya que el proceso de compilación de kernel ocupa aproximadamente 15GB.

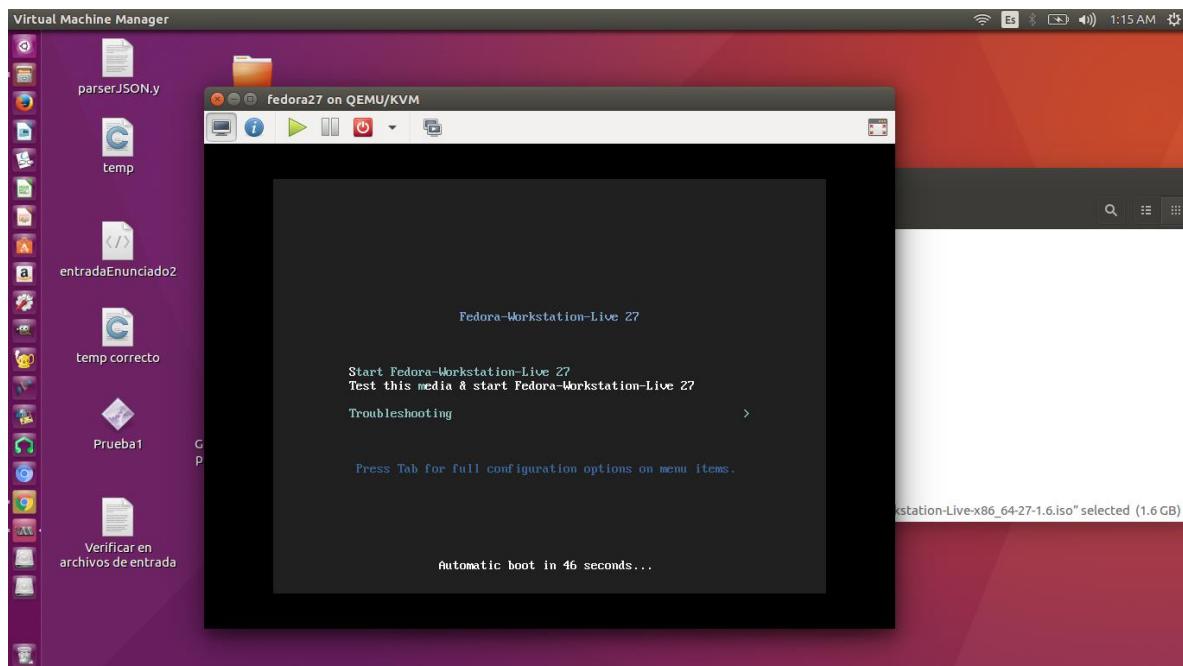


9. Se creará la máquina virtual.

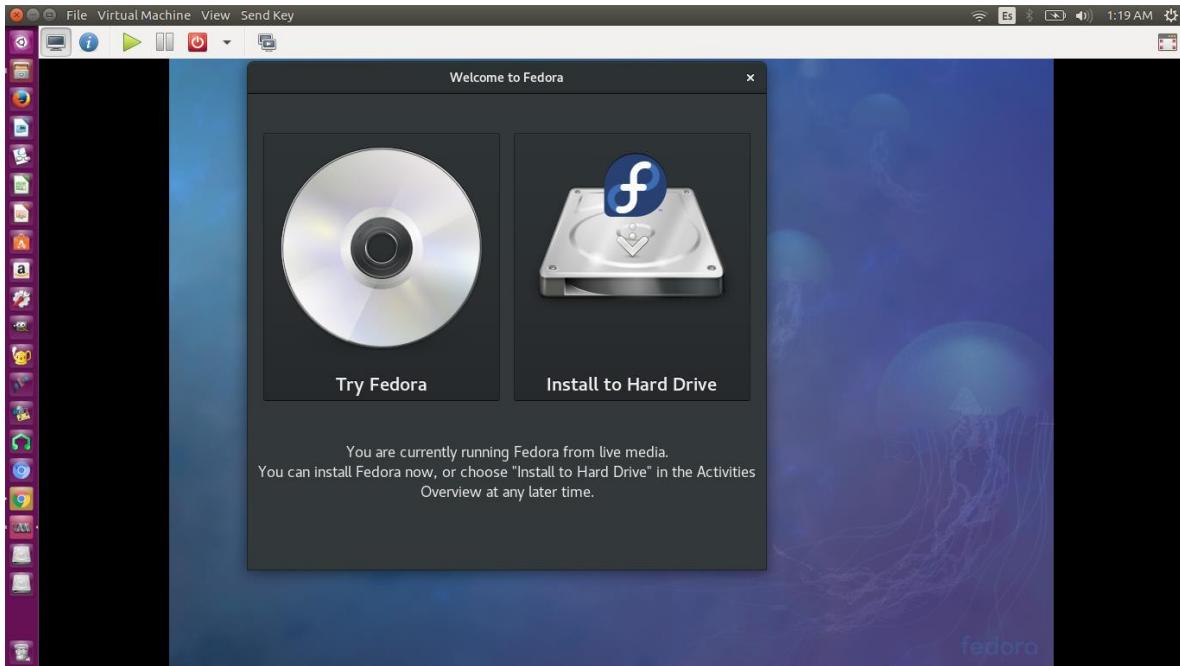


### iii. Instalación de Fedora 27

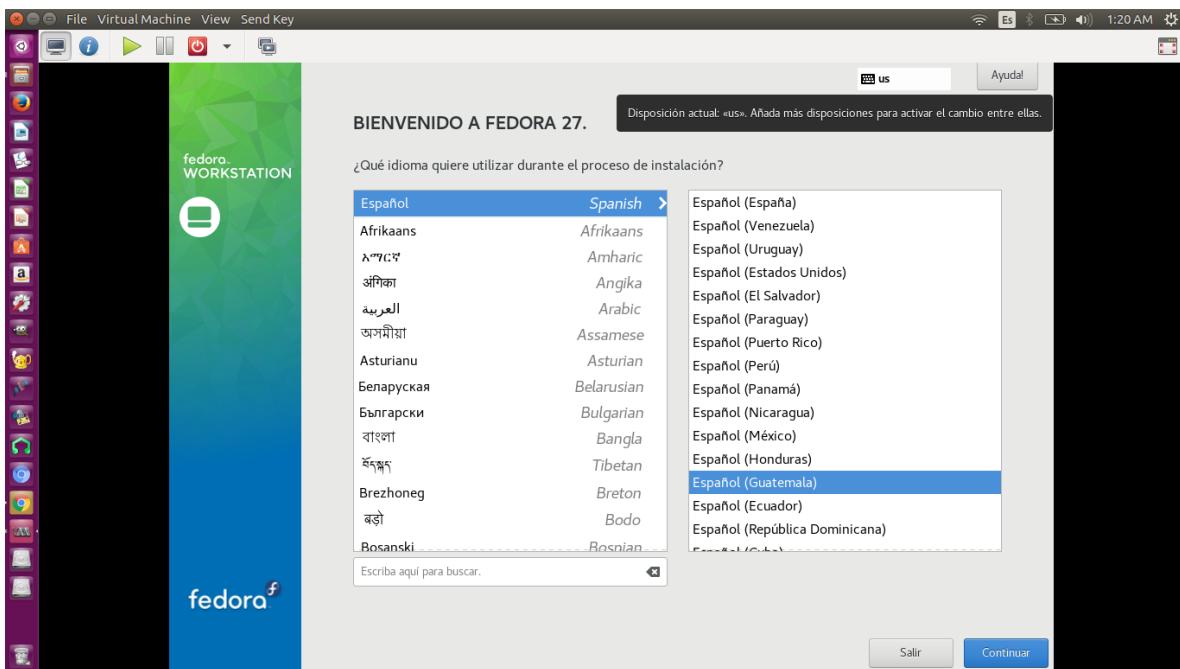
1. Se selecciona la instalación al momento de bootejar con el ISO: "Start Fedora-Workstation-live 27".



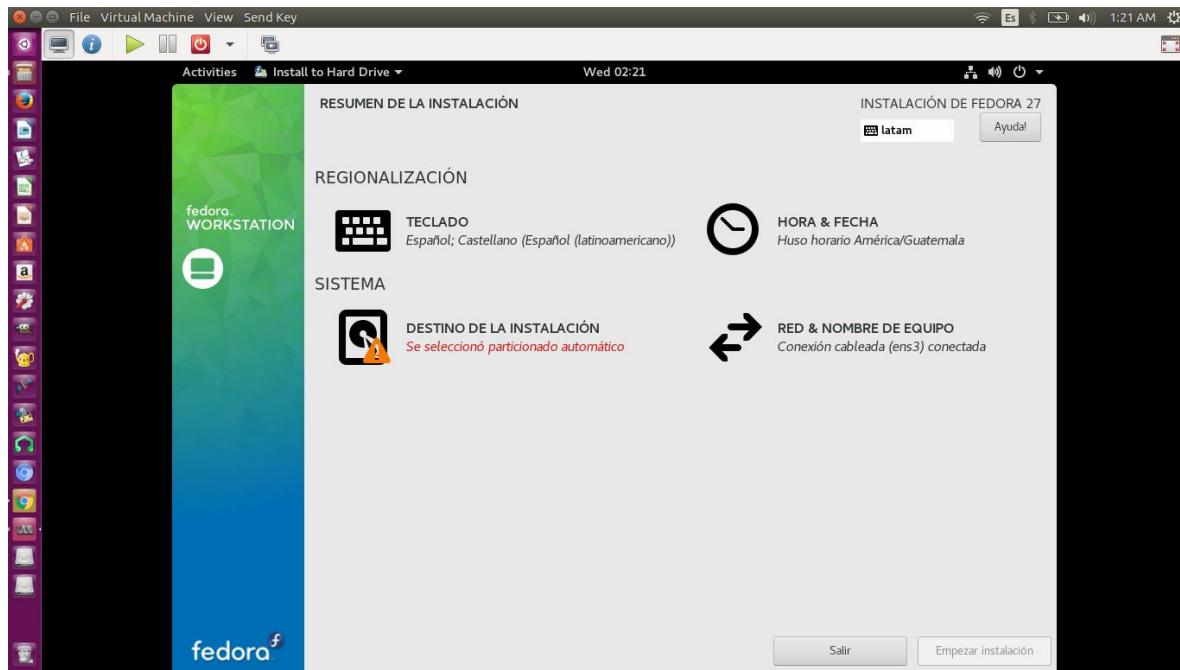
2. Seleccionamos la opción: "Install to hard drive" para comenzar con la instalación en la máquina virtual.



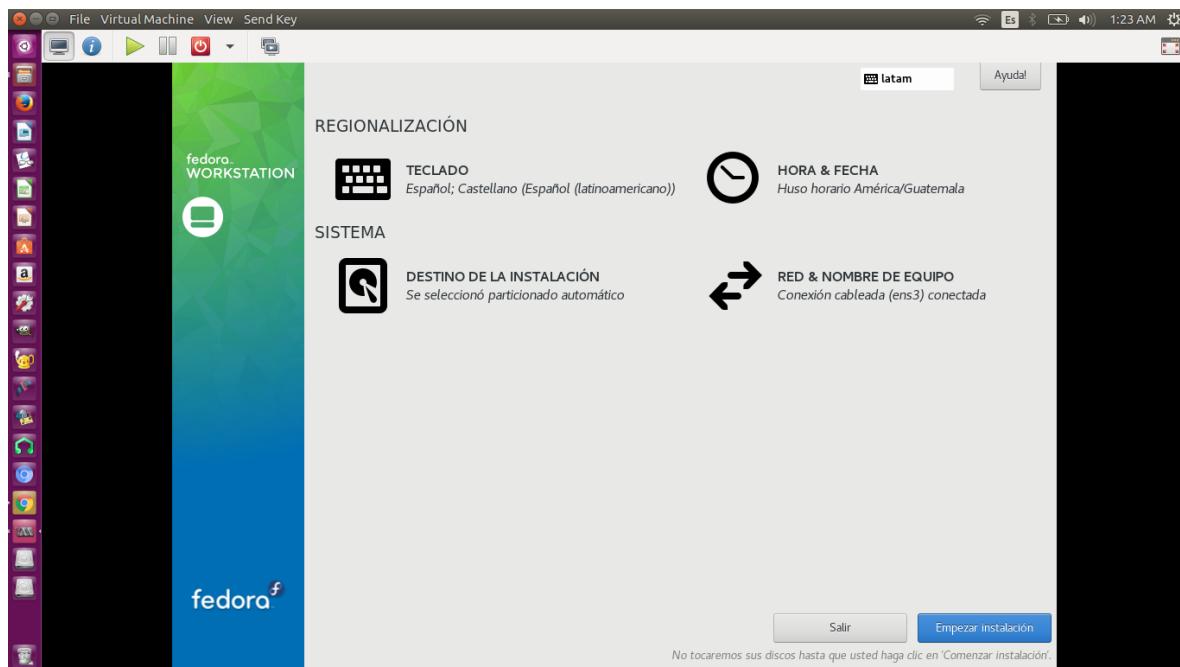
3. Seleccionamos el idioma de preferencia. El instalador detecta automáticamente el idioma dependiendo de nuestra posición geográfica.



4. Debemos entrar en la opción: “Destino de la instalación”, para aceptar los settings por defecto y si es necesario, modificarlos. En este caso no realizamos ninguna modificación.



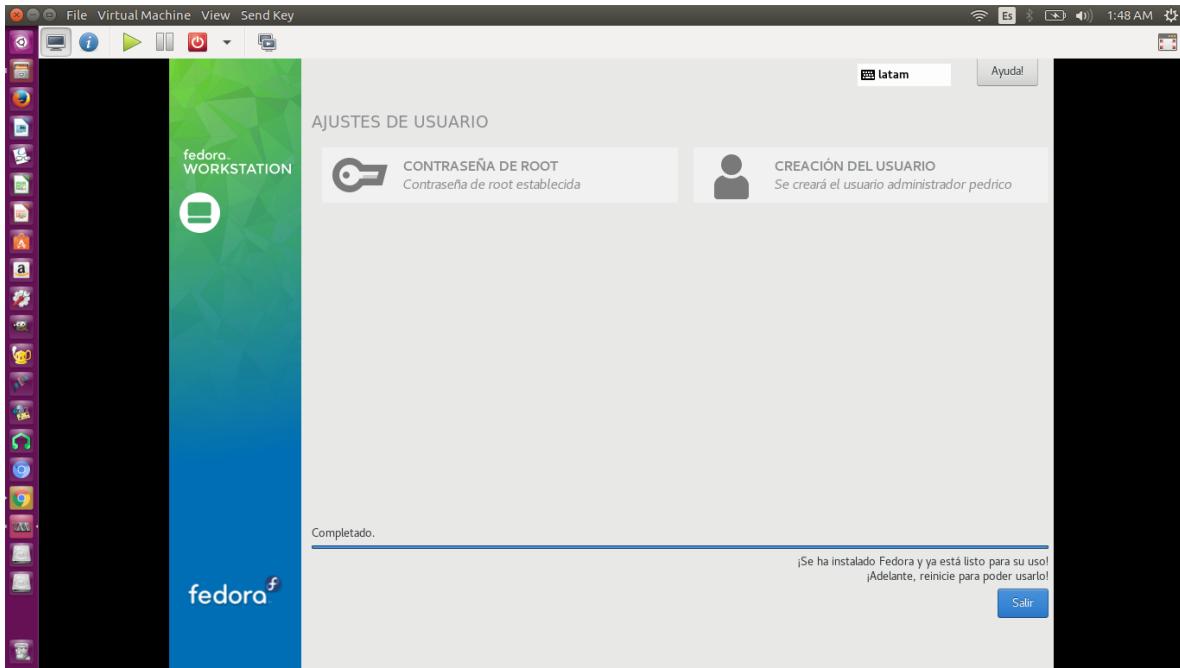
5. Presionamos el botón “Empezar Instalación”



6. Mientras se está realizando la instalación del SO, podemos configurar el usuario root y crear un usuario. Podemos asignar contraseñas o realizar configuraciones avanzadas. En este caso únicamente asignamos una contraseña a root y creamos el usuario “pedrico”.



7. Una vez finalizado el proceso de instalación, podremos presionar el botón “Salir”.

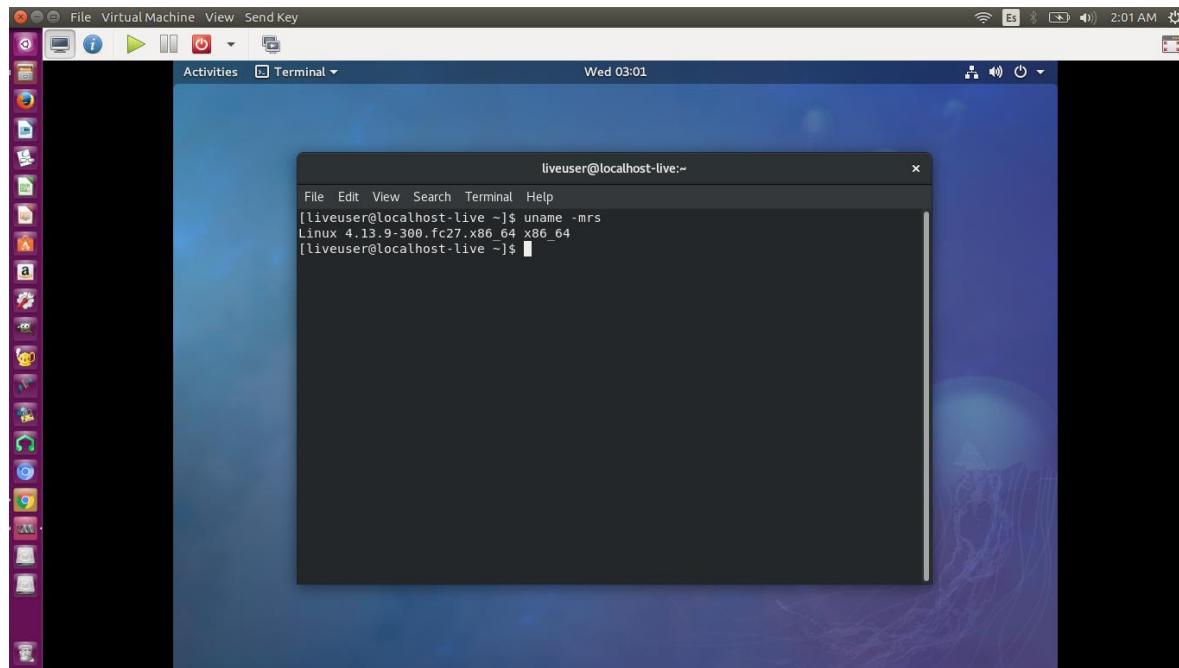


#### iv. Preparación para la compilación de un nuevo Kernel

Una vez tenemos instalado el SO Fedora 27 proseguiremos con la compilación del nuevo kernel para nuestro sistema operativo.

1. Primero verificaremos la versión actual ingresando el siguiente comando en la terminal.

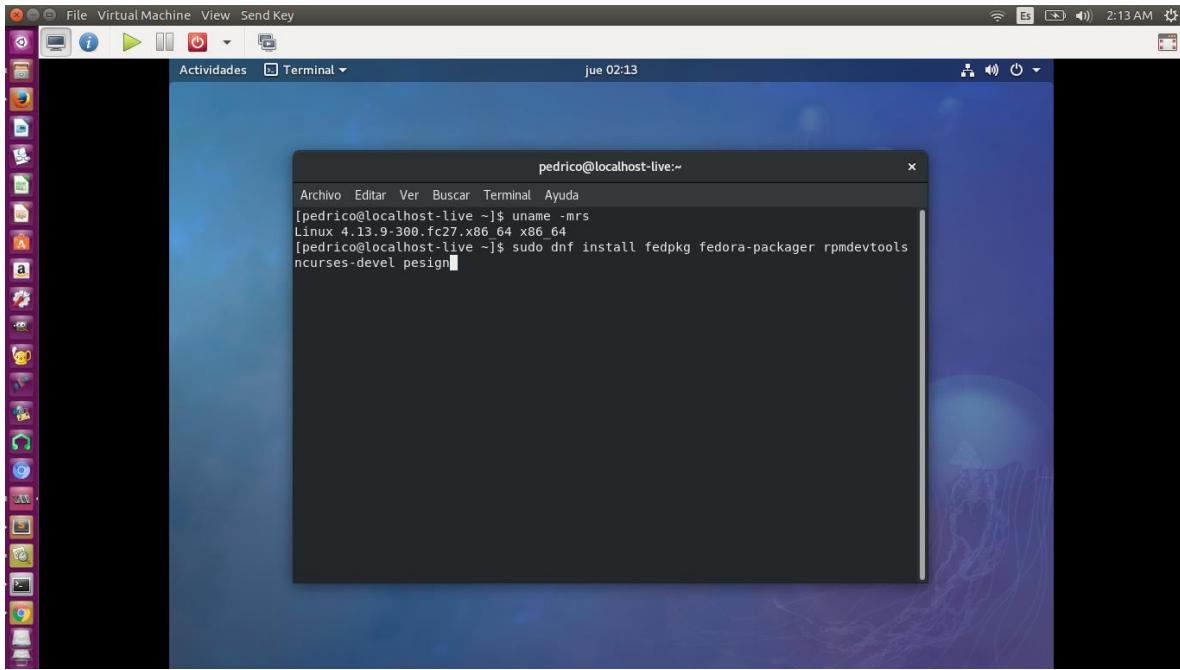
`uname -mrs`



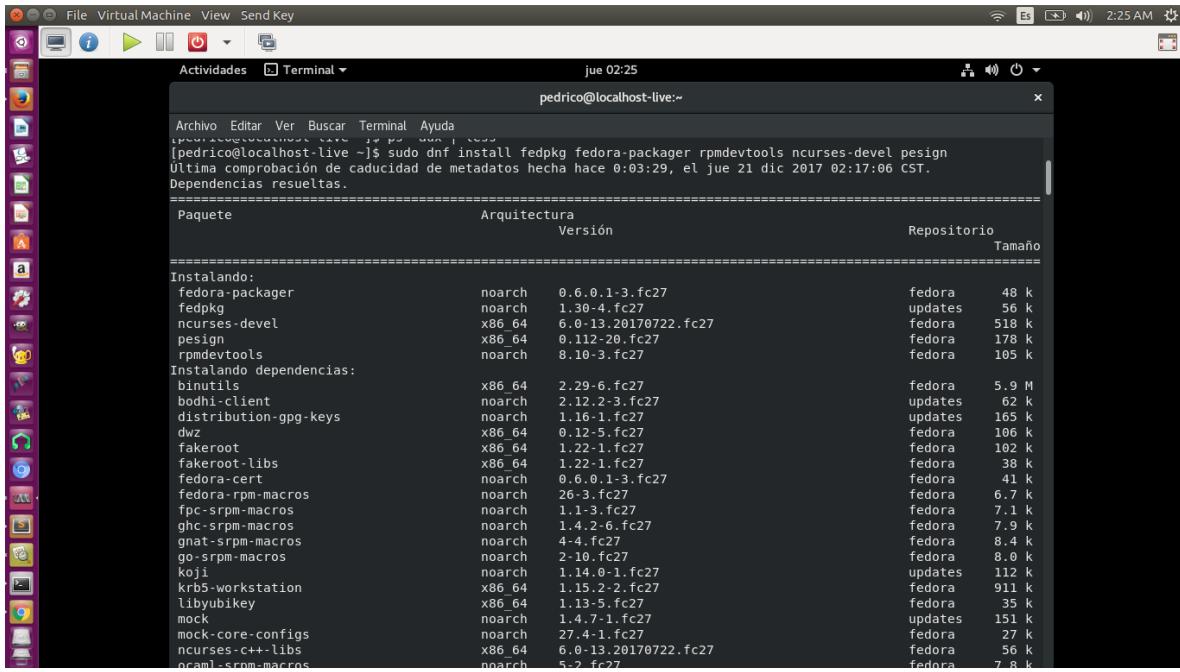
En nuestro caso obtenemos la versión Linux 4.13.9.300.fc27.x86\_64. Este número de versión nos servirá más adelante para poder comprobar la instalación del nuevo kernel.

2. Necesitaremos instalar las siguientes dependencias que nos proporcionan las herramientas para compilar los rpm's. Para ello tecleamos el siguiente comando:

`sudo dnf install fedpkg fedora-packager rpmdevtools ncurses-devel pesign`

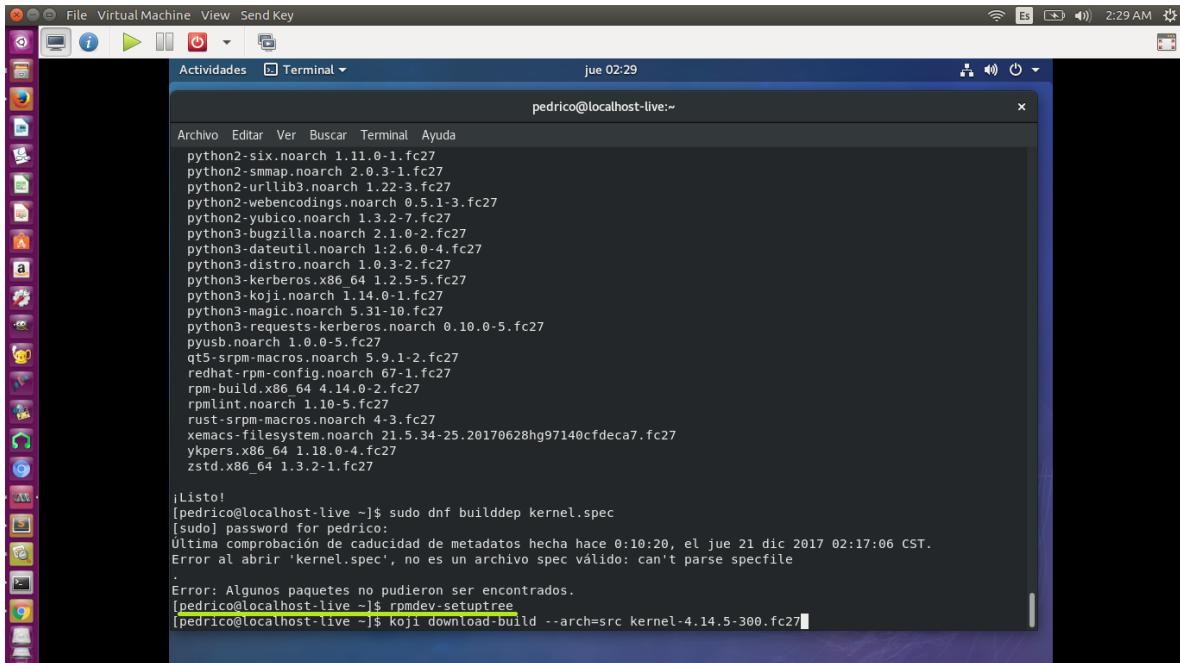


Nos instalará varios paquetes.



3. El siguiente comando construye la estructura de carpetas para la compilación del nuevo kernel en la carpeta /home/pedrico/

\$rpmpdev-setuptree

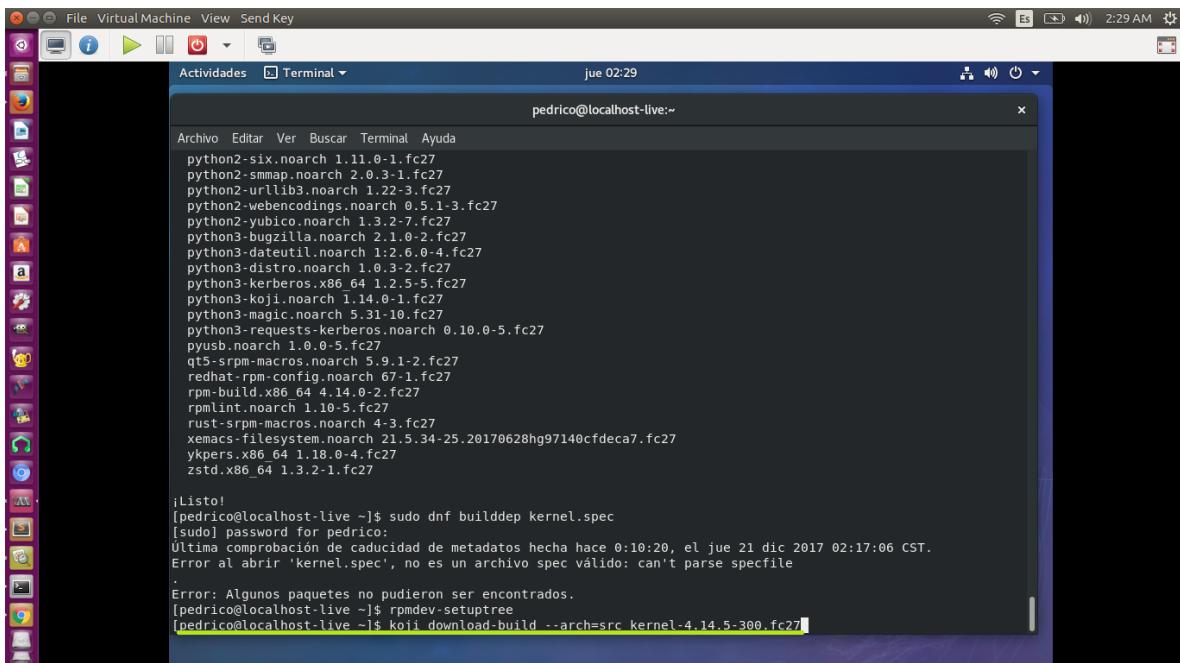


```
Archivo Editar Ver Buscar Terminal Ayuda
python2-six.noarch 1.11.0-1.fc27
python2-smmmap.noarch 2.0.3-1.fc27
python2-urllib3.noarch 1.22-3.fc27
python2-webencodings.noarch 0.5.1-3.fc27
python2-yubico.noarch 1.3.2-7.fc27
python3-bugzilla.noarch 2.1.0-2.fc27
python3-dateutil.noarch 1:2.6.0-4.fc27
python3-distro.noarch 1.0.3-2.fc27
python3-kerberos.x86_64 1.2.5-5.fc27
python3-koji.noarch 1.14.0-1.fc27
python3-magic.noarch 5.31-10.fc27
python3-requests-kerberos.noarch 0.10.0-5.fc27
pyusb.noarch 1.0.0-5.fc27
qt5-srpm-macros.noarch 5.9.1-2.fc27
redhat-rpm-config.noarch 67-1.fc27
rpm-build.x86_64 4.14.0-2.fc27
rpmlint.noarch 1.10-5.fc27
rust-srpm-macros.noarch 4-3.fc27
xemacs-filesystem.noarch 21.5.34-25.20170628hg97140cfdeca7.fc27
ykpers.x86_64 1.18.0-4.fc27
zstd.x86_64 1.3.2-1.fc27

;Listo!
[pedrico@localhost-live ~]$ sudo dnf builddep kernel.spec
[sudo] password for pedrico:
Última comprobación de caducidad de metadatos hecha hace 0:10:20, el jue 21 dic 2017 02:17:06 CST.
Error al abrir 'kernel.spec', no es un archivo spec válido: can't parse specfile
.
Error: Algunos paquetes no pudieron ser encontrados.
[pedrico@localhost-live ~]$ rpmpdev-setuptree
[pedrico@localhost-live ~]$ koji download-build --arch=src kernel-4.14.5-300.fc27
```

4. El siguiente comando descargara el kernel kernel-4.14.5-300.fc27 de la pagina de versiones de fedora, Koji. Koji es un proyecto (<https://koji.fedoraproject.org/koji/>) cuyo objetivo es gestionar las fuentes y compilados de paquetes para fedora. Allí podemos encontrar varias versiones de los kernels de Fedora. En la caja de búsqueda de la página podemos teclear la palabra “kernel” y nos filtrará los resultados de los kernels de fedora registrados en koji.

[koji download-build --arch=src kernel-4.14.5-300.fc27](#)

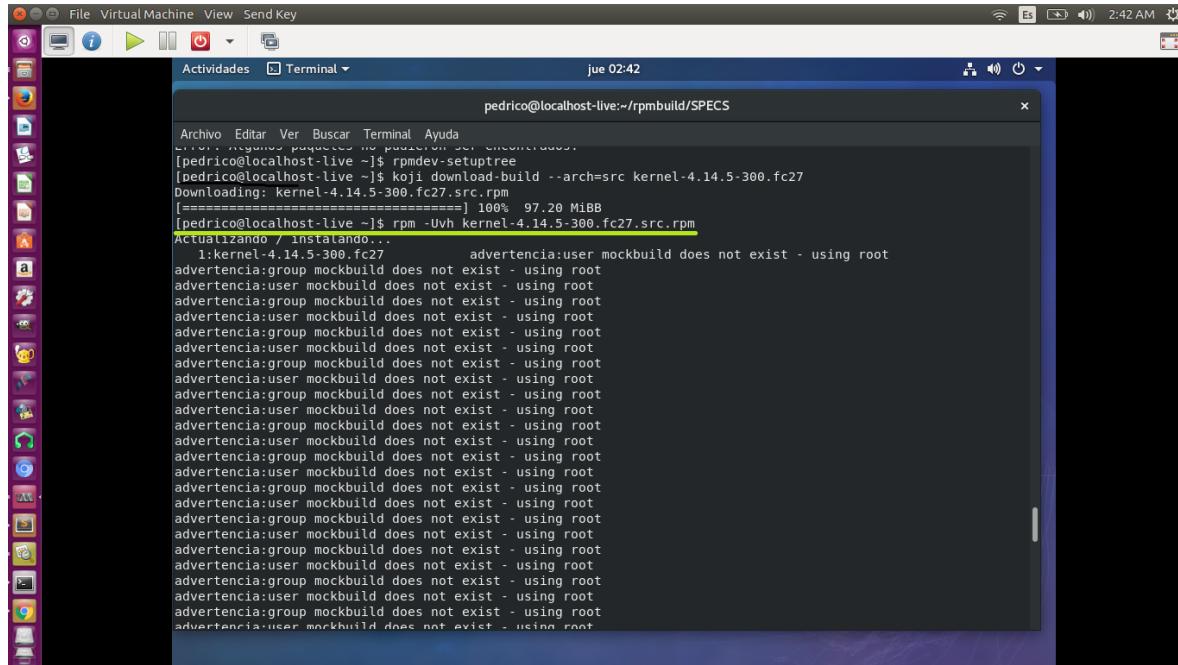


```
Archivo Editar Ver Buscar Terminal Ayuda
python2-six.noarch 1.11.0-1.fc27
python2-smmmap.noarch 2.0.3-1.fc27
python2-urllib3.noarch 1.22-3.fc27
python2-webencodings.noarch 0.5.1-3.fc27
python2-yubico.noarch 1.3.2-7.fc27
python3-bugzilla.noarch 2.1.0-2.fc27
python3-dateutil.noarch 1:2.6.0-4.fc27
python3-distro.noarch 1.0.3-2.fc27
python3-kerberos.x86_64 1.2.5-5.fc27
python3-koji.noarch 1.14.0-1.fc27
python3-magic.noarch 5.31-10.fc27
python3-requests-kerberos.noarch 0.10.0-5.fc27
pyusb.noarch 1.0.0-5.fc27
qt5-srpm-macros.noarch 5.9.1-2.fc27
redhat-rpm-config.noarch 67-1.fc27
rpm-build.x86_64 4.14.0-2.fc27
rpmlint.noarch 1.10-5.fc27
rust-srpm-macros.noarch 4-3.fc27
xemacs-filesystem.noarch 21.5.34-25.20170628hg97140cfdeca7.fc27
ykpers.x86_64 1.18.0-4.fc27
zstd.x86_64 1.3.2-1.fc27

;Listo!
[pedrico@localhost-live ~]$ sudo dnf builddep kernel.spec
[sudo] password for pedrico:
Última comprobación de caducidad de metadatos hecha hace 0:10:20, el jue 21 dic 2017 02:17:06 CST.
Error al abrir 'kernel.spec', no es un archivo spec válido: can't parse specfile
.
Error: Algunos paquetes no pudieron ser encontrados.
[pedrico@localhost-live ~]$ rpmpdev-setuptree
[pedrico@localhost-live ~]$ koji download-build --arch=src kernel-4.14.5-300.fc27
```

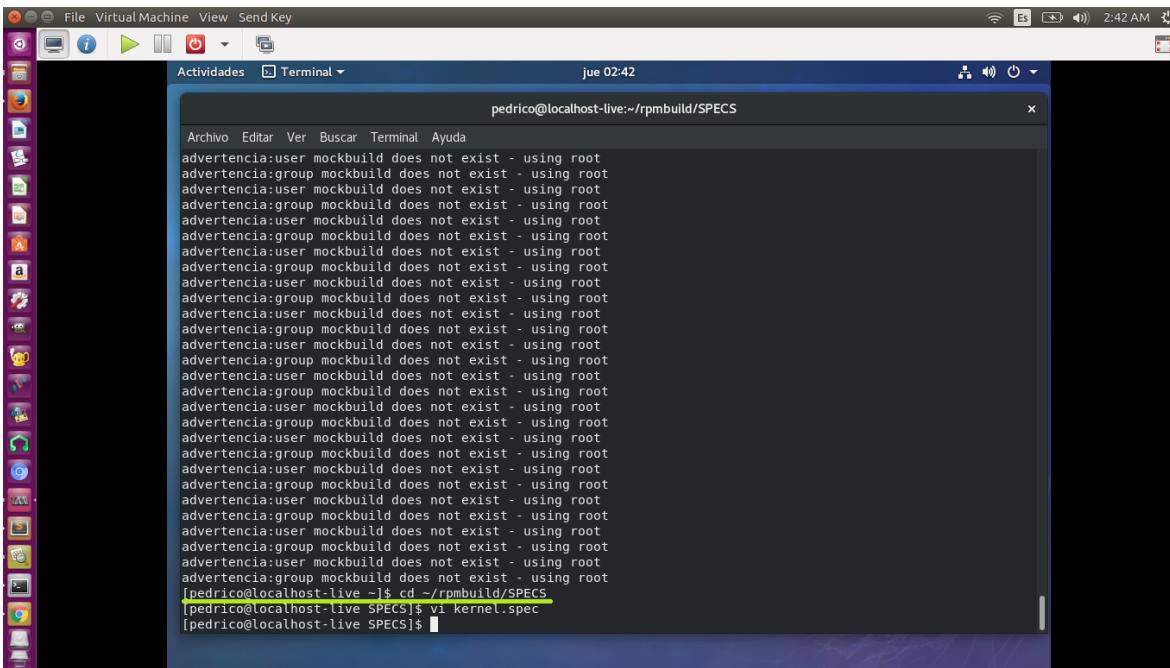
5. Instalamos el kernel recién descargado con el siguiente comando. Este comando escribe el directorio /home/pedrico/rpmbuild/SOURCES y el archivo /home/pedrico/rpmbuild/SPECS. Los mensajes con la siguiente estructura pueden ser ignorados: warning: user kojibuilder does not exist - using root.

```
rpm -Uvh kernel-4.14.5-300.fc27.src.rpm
```



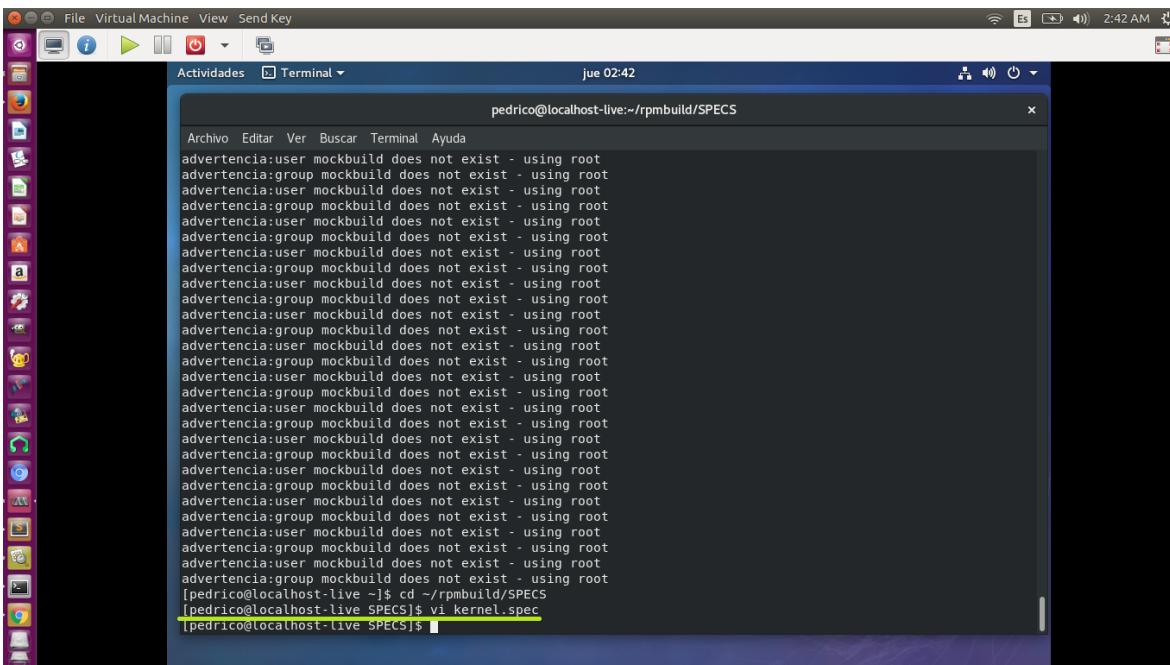
6. Cambiamos de directorio:

```
cd ~/rpmbuild/SPECS
```



7. Abrimos el archivo `kernel.spec`. En este caso utilizaremos el editor `vi`, ya que necesitamos modificarlo.

vi kernel.spec



- ## 8. Nos posicionaremos en la línea:

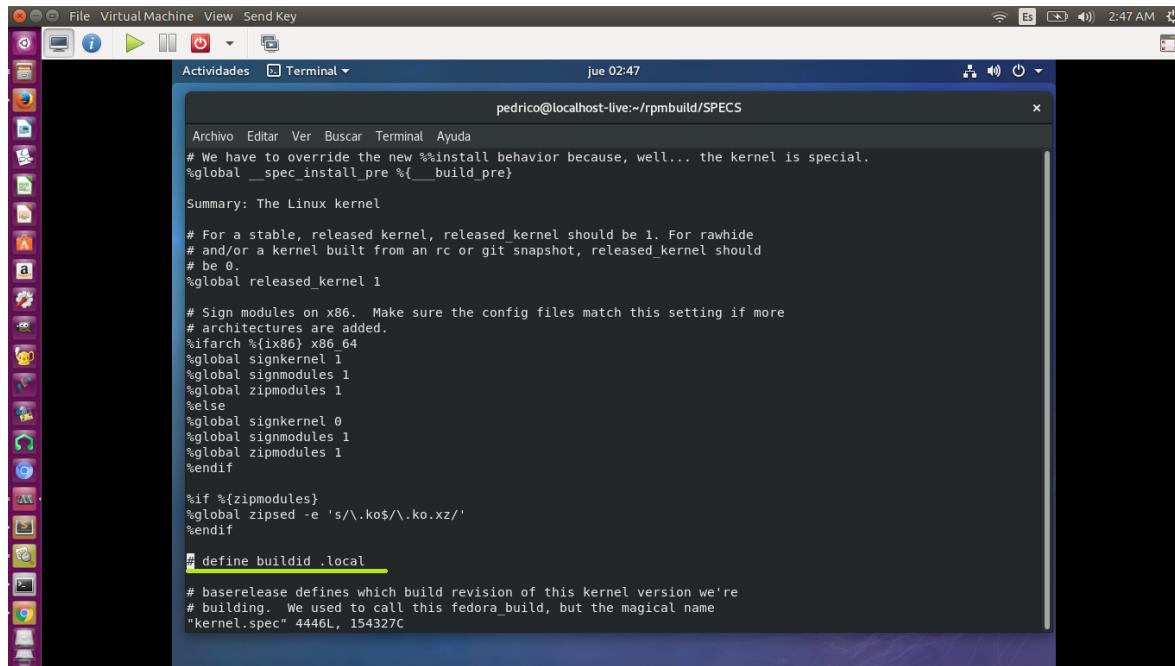
```
# define buildid .local
```

y la cambiaremos por:

```
%define buildid <custom_text>
```

Donde <custom\_text> es un identificador para nuestro nuevo kernel. En este caso colocaremos

```
%define buildid .pedrico
```



```
# We have to override the new %%install behavior because, well... the kernel is special.
%global __spec_install_pre %{__build_pre}

Summary: The Linux kernel

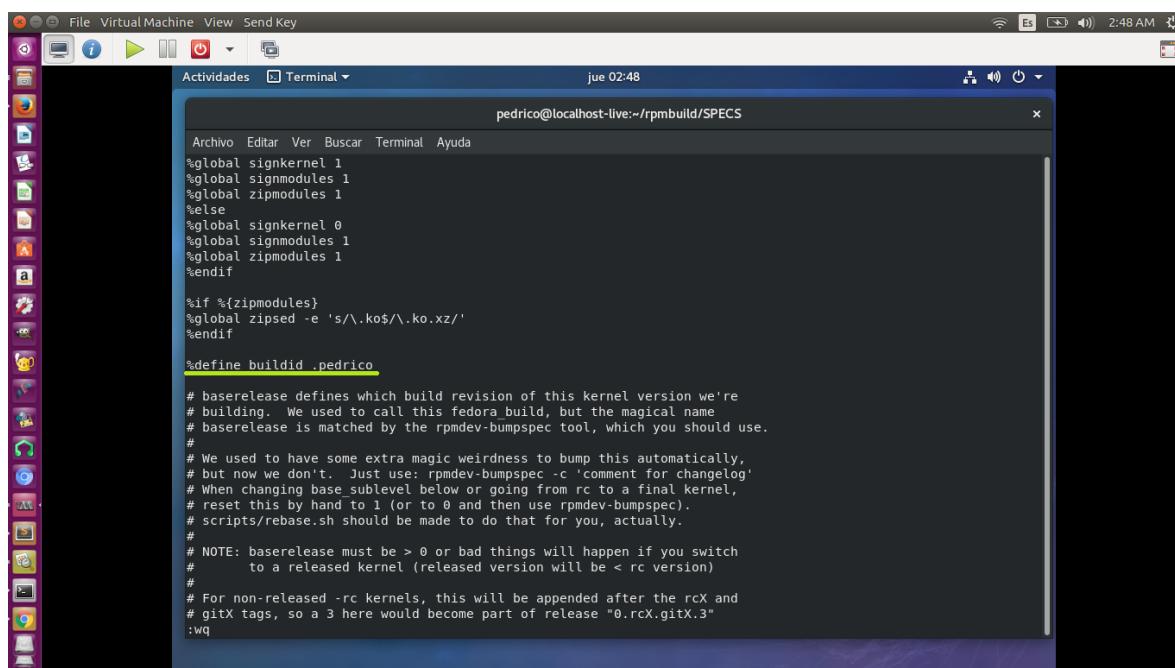
# For a stable, released kernel, released_kernel should be 1. For rawhide
# and/or a kernel built from an rc or git snapshot, released_kernel should
# be 0.
%global released_kernel 1

# Sign modules on x86. Make sure the config files match this setting if more
# architectures are added.
%ifarch %ix86 x86_64
%global signkernel 1
%global signmodules 1
%global zipmodules 1
%else
%global signkernel 0
%global signmodules 1
%global zipmodules 1
%endif

%if %{zipmodules}
%global zipsed -e 's/\.ko$/\.ko.xz/'
%endif

# define buildid .local

# baserelease defines which build revision of this kernel version we're
# building. We used to call this fedora_build, but the magical name
# "kernel.spec" 4446L, 154327C
```



```
%global signkernel 1
%global signmodules 1
%global zipmodules 1
%else
%global signkernel 0
%global signmodules 1
%global zipmodules 1
%endif

%if %{zipmodules}
%global zipsed -e 's/\.ko$/\.ko.xz/'
%endif

#define buildid .pedrico

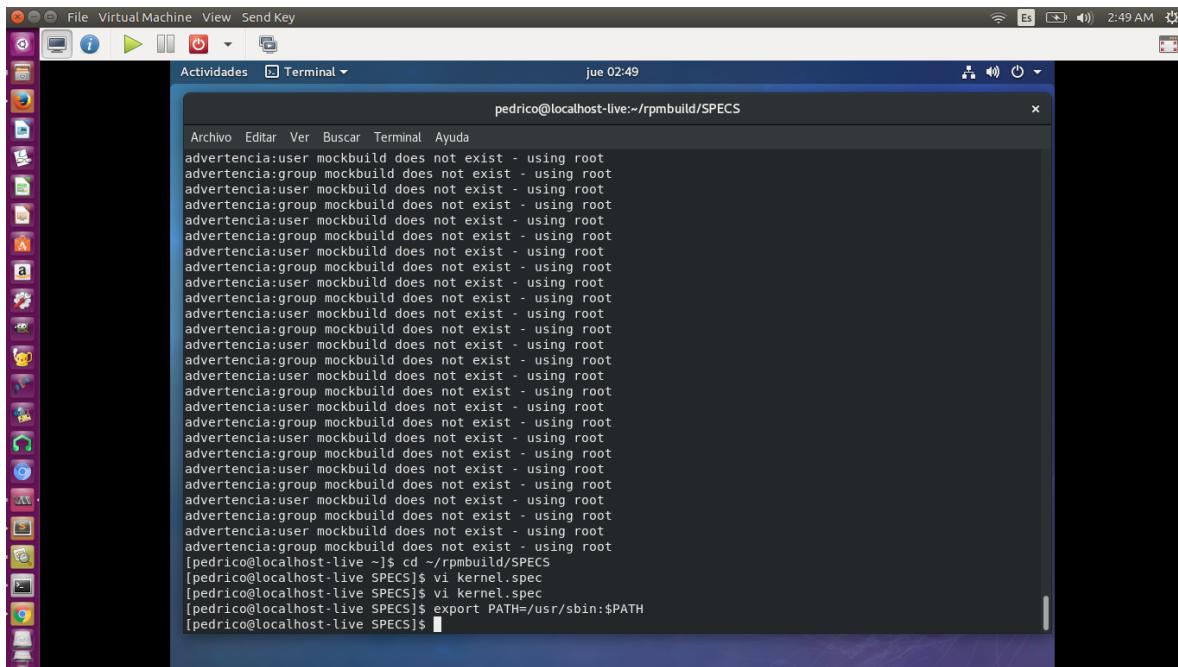
# baserelease defines which build revision of this kernel version we're
# building. We used to call this fedora_build, but the magical name
# baserelease is matched by the rpmdv-bumpspec tool, which you should use.
#
# We used to have some extra magic weirdness to bump this automatically,
# but now we don't. Just use: rpmdv-bumpspec -c 'comment for changelog'
# When changing base sublevel below or going from rc to a final kernel,
# reset this by hand to 1 (or to 0 and then use rpmdv-bumpspec).
# scripts/rebase.sh should be made to do that for you, actually.
#
# NOTE: baserelease must be > 0 or bad things will happen if you switch
#       to a released kernel (released version will be < rc version)
#
# For non-released -rc kernels, this will be appended after the rcX and
# gitX tags, so a 3 here would become part of release "0.rcX.gitX.3"
:wq
```

Salimos guardando los cambios, presionamos ESC e ingresamos “:wq”.

## V. Compilación del nuevo Kernel

1. Nos aseguramos que /usr/sbin es nuestro path:

```
export PATH=/usr/sbin:$PATH
```

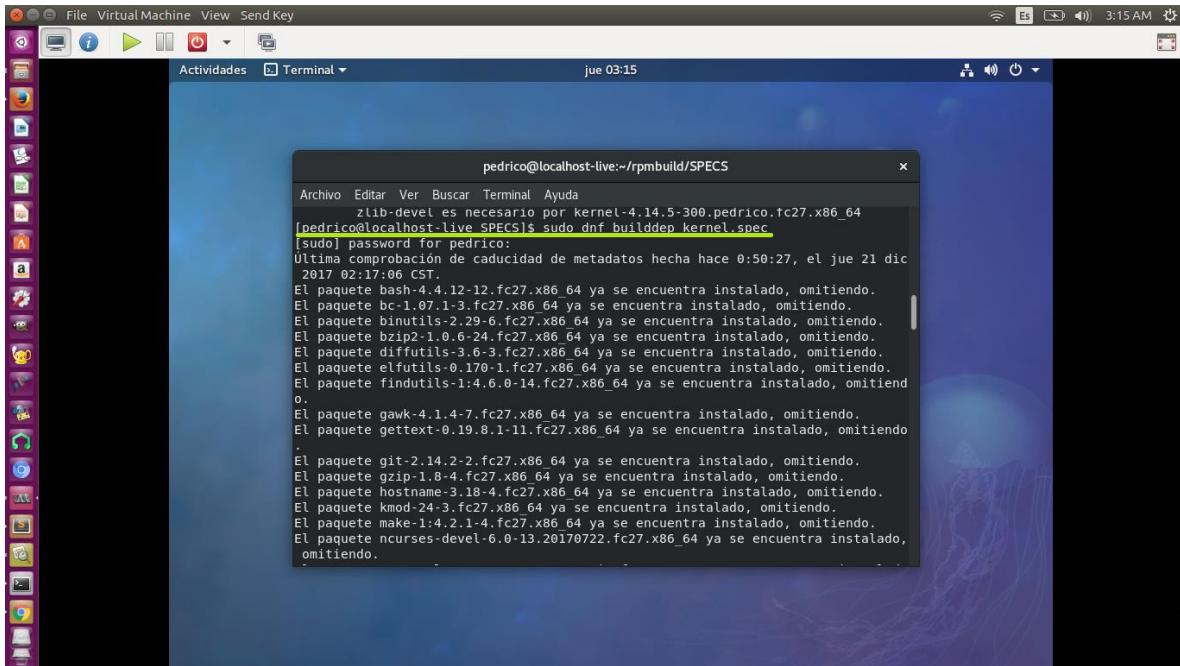


A screenshot of a Linux desktop environment showing a terminal window. The terminal window title is "pedrico@localhost-live:~/rpmbuild/SPECS". The terminal content shows the user running the command "export PATH=/usr/sbin:\$PATH". The terminal window is part of a desktop interface with a menu bar at the top and a docked application bar below it. The desktop background is visible behind the window.

```
Archivo Editar Ver Buscar Terminal Ayuda
advertencia:user mockbuild does not exist - using root
advertencia:group mockbuild does not exist - using root
advertencia:user mockbuild does not exist - using root
advertencia:group mockbuild does not exist - using root
advertencia:user mockbuild does not exist - using root
advertencia:group mockbuild does not exist - using root
advertencia:user mockbuild does not exist - using root
advertencia:group mockbuild does not exist - using root
advertencia:user mockbuild does not exist - using root
advertencia:group mockbuild does not exist - using root
advertencia:user mockbuild does not exist - using root
advertencia:group mockbuild does not exist - using root
advertencia:user mockbuild does not exist - using root
advertencia:group mockbuild does not exist - using root
advertencia:user mockbuild does not exist - using root
advertencia:group mockbuild does not exist - using root
advertencia:user mockbuild does not exist - using root
advertencia:group mockbuild does not exist - using root
advertencia:user mockbuild does not exist - using root
advertencia:group mockbuild does not exist - using root
advertencia:user mockbuild does not exist - using root
advertencia:group mockbuild does not exist - using root
advertencia:user mockbuild does not exist - using root
advertencia:group mockbuild does not exist - using root
advertencia:user mockbuild does not exist - using root
advertencia:group mockbuild does not exist - using root
[pedrico@localhost-live ~]$ cd ~/rpmbuild/SPECS
[pedrico@localhost-live SPECS]$ vi kernel.spec
[pedrico@localhost-live SPECS]$ vi kernel.spec
[pedrico@localhost-live SPECS]$ export PATH=/usr/sbin:$PATH
[pedrico@localhost-live SPECS]$
```

2. Instalamos las dependencias que nos ayudan a compilar

```
sudo dnf builddep kernel.spec
```

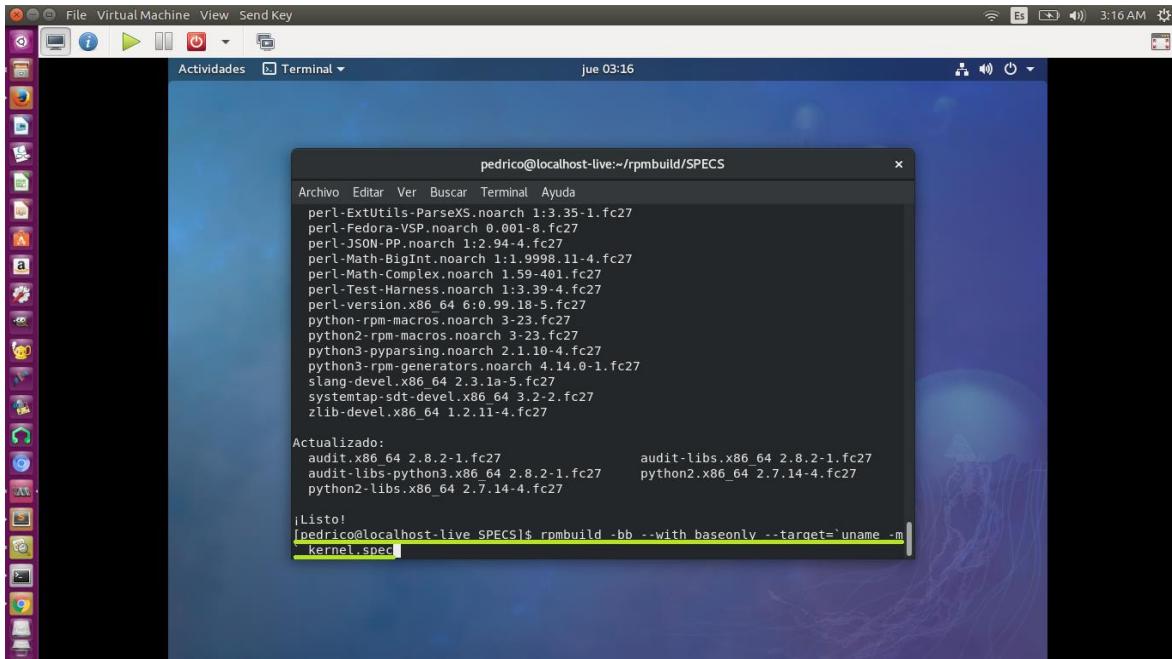


3. Ejecutamos el comando de compilación con los siguientes parámetros. Este proceso puede tardar varias horas dependiendo de los recursos asignados a la máquina virtual. En mi caso la compilación duro entre 4 y 5 horas.

```
rpmbuild -bb --with baseonly --without debuginfo --target=`uname -m` kernel.spec
```

-bb --with baseonly      Especifica que solo un kernel específico debe ser construido.

--without debuginfo      Quita algún código de debugueo del kernel



```
pedrico@localhost-live:~/rpmbuild/SPECS
Archivo Editar Ver Buscar Terminal Ayuda
perl-ExtUtils-ParserXS.noarch 1:3.35-1.fc27
perl-Fedora-VSP.noarch 0.001-8.fc27
perl-JSON-PP.noarch 1:2.94-4.fc27
perl-Math-BigInt.noarch 1:1.9998.11-4.fc27
perl-Math-Complex.noarch 1.59-401.fc27
perl-Test-Harness.noarch 1:3.39-4.fc27
perl-version.x86_64 6:0.99.18-5.fc27
python-rpm-macros.noarch 3-23.fc27
python2-rpm-macros.noarch 3-23.fc27
python3-pyparsing.noarch 2.1.10-4.fc27
python3-rpm-generators.noarch 4.14.0-1.fc27
slang-devel.x86_64 2.3.14-5.fc27
systemtap-sdt-devel.x86_64 3.2-2.fc27
zlib-devel.x86_64 1.2.11-4.fc27

Actualizado:
audit.x86_64 2.8.2-1.fc27          audit-libs.x86_64 2.8.2-1.fc27
audit-libs-python3.x86_64 2.8.2-1.fc27    python2.x86_64 2.7.14-4.fc27
python2-libs.x86_64 2.7.14-4.fc27

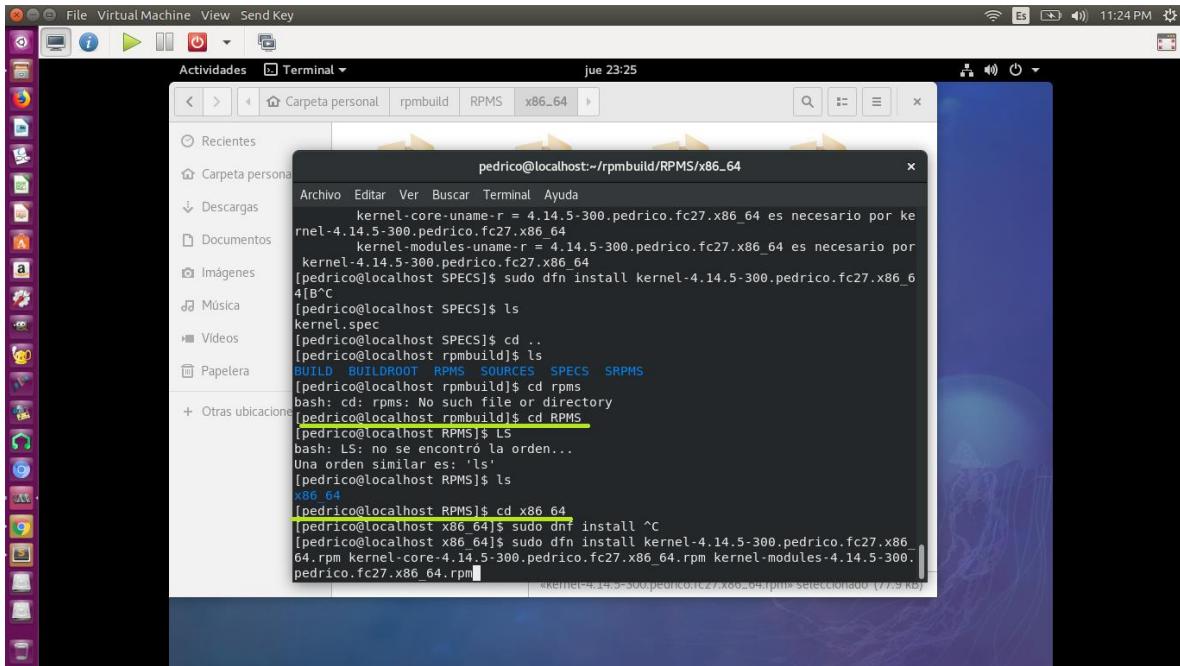
iListo!
[pedrico@localhost-live SPECS]$ rpmbuild -bb --with baseonly --target='uname -m'
kernel.spec
```

## vi. Instalación del nuevo kernel

Una vez finalizado el proceso de compilación debemos de instalar el nuevo kernel generado. Para ello seguimos los siguientes pasos:

1. Nos colocamos en el directorio /home/pedrico/rpmbuild/RPMS/x86\_64

```
cd rpmbuild/RPMS/x86_64
```

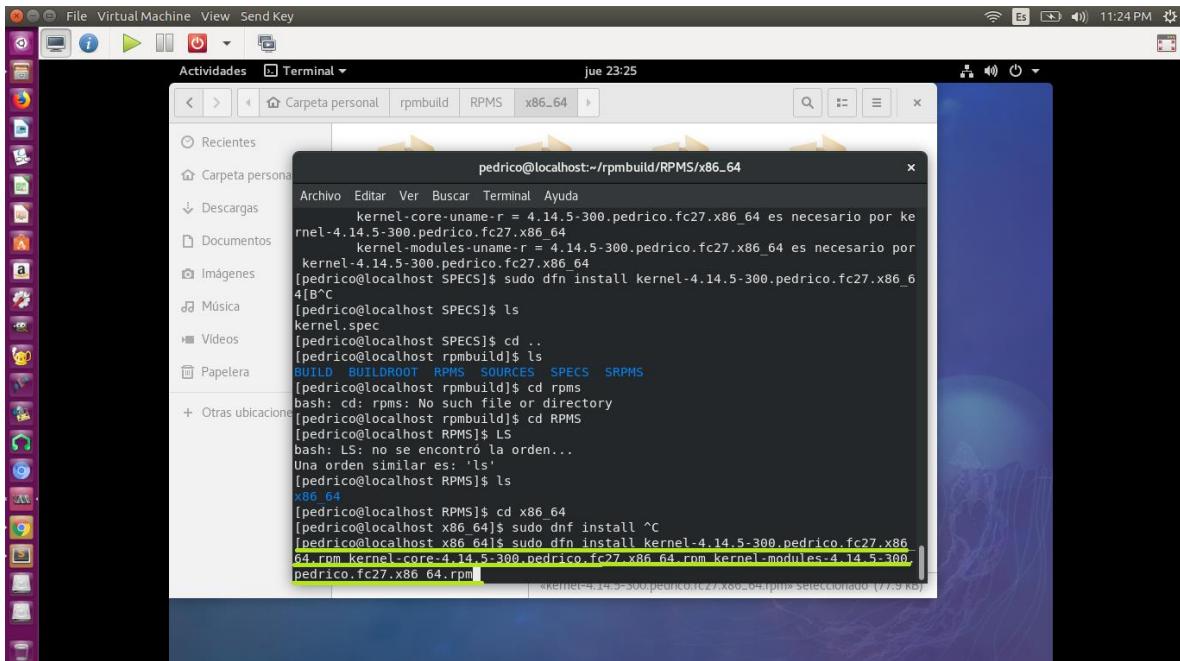


The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "pedrico@localhost:~/rpmbuild/RPMS/x86\_64". The terminal displays the following command-line session:

```
kernel-core-uname-r = 4.14.5-300.pedrico.fc27.x86_64 es necesario por kernel-4.14.5-300.pedrico.fc27.x86_64
kernel-modules-uname-r = 4.14.5-300.pedrico.fc27.x86_64 es necesario por kernel-4.14.5-300.pedrico.fc27.x86_64
[pedrico@localhost SPECS]$ sudo dnf install kernel-4.14.5-300.pedrico.fc27.x86_64[8C]
[pedrico@localhost SPECS]$ ls
kernel.spec
[pedrico@localhost SPECS]$ cd ..
[pedrico@localhost rpmbuild]$ ls
BUILD BUILDROOT RPMS SOURCES SPECS SRPMS
[pedrico@localhost rpmbuild]$ cd rpms
bash: cd: rpms: No such file or directory
[pedrico@localhost rpmbuild]$ cd RPMS
[pedrico@localhost RPMS]$ LS
bash: LS: no se encontró la orden...
Una orden similar es: 'ls'
[pedrico@localhost RPMS]$ ls
x86_64
[pedrico@localhost RPMS]$ cd x86_64
[pedrico@localhost x86_64]$ sudo dnf install ^
[pedrico@localhost x86_64]$ sudo dnf install kernel-4.14.5-300.pedrico.fc27.x86_64.rpm kernel-core-4.14.5-300.pedrico.fc27.x86_64.rpm kernel-modules-4.14.5-300.pedrico.fc27.x86_64.rpm
```

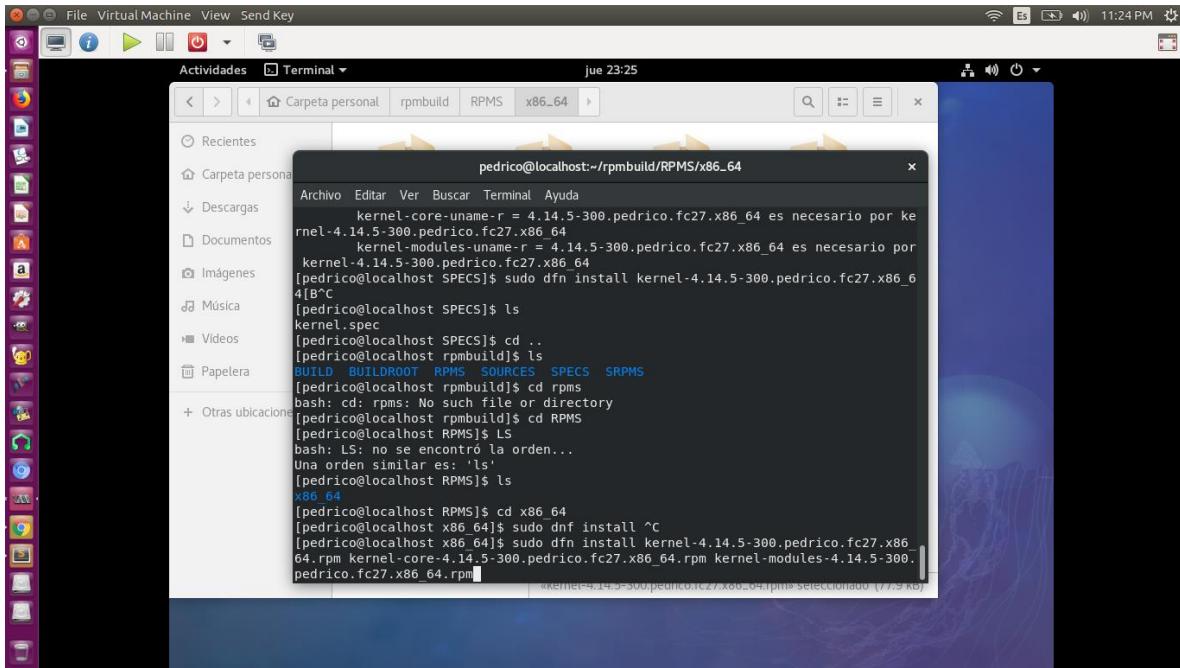
## 2. Realizamos la instalación ejecutando el siguiente comando:

```
sudo dnf install kernel-4.14.5-300.pedrico.fc27.x86_64.rpm kernel-core-4.14.5-300.pedrico.fc27.x86_64.rpm kernel-modules-4.14.5-300.pedrico.fc27.x86_64.rpm
```



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "pedrico@localhost:~/rpmbuild/RPMS/x86\_64". The terminal displays the following command-line session, identical to the one in the previous screenshot:

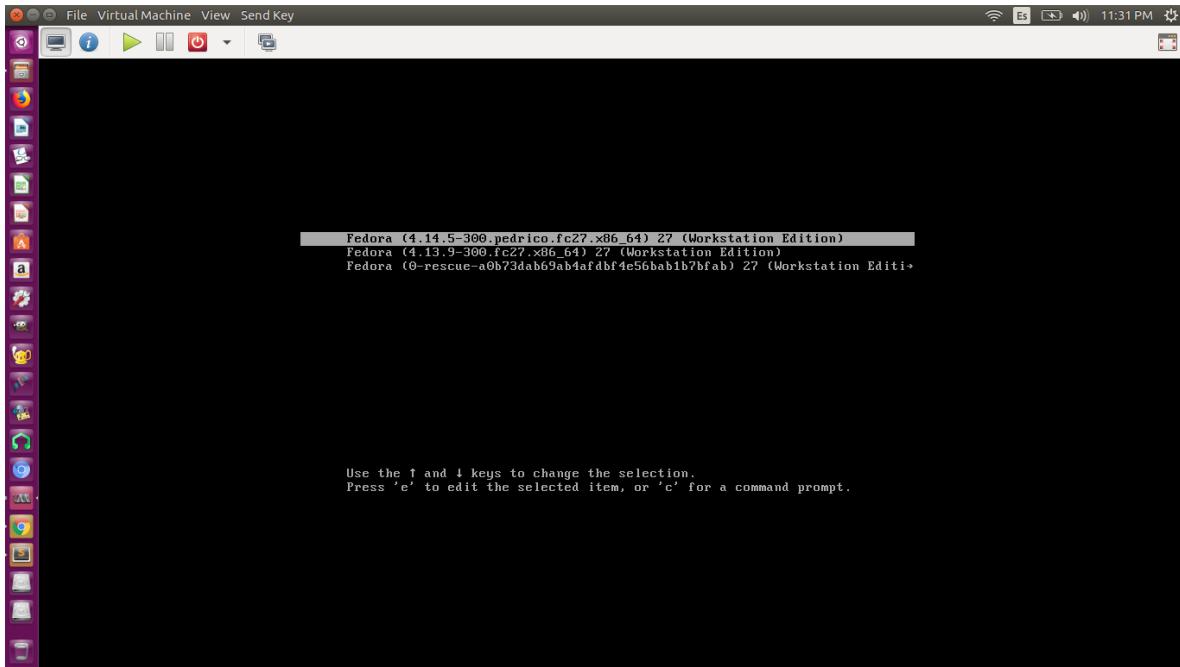
```
kernel-core-uname-r = 4.14.5-300.pedrico.fc27.x86_64 es necesario por kernel-4.14.5-300.pedrico.fc27.x86_64
kernel-modules-uname-r = 4.14.5-300.pedrico.fc27.x86_64 es necesario por kernel-4.14.5-300.pedrico.fc27.x86_64
[pedrico@localhost SPECS]$ sudo dnf install kernel-4.14.5-300.pedrico.fc27.x86_64[8C]
[pedrico@localhost SPECS]$ ls
kernel.spec
[pedrico@localhost SPECS]$ cd ..
[pedrico@localhost rpmbuild]$ ls
BUILD BUILDROOT RPMS SOURCES SPECS SRPMS
[pedrico@localhost rpmbuild]$ cd rpms
bash: cd: rpms: No such file or directory
[pedrico@localhost rpmbuild]$ cd RPMS
[pedrico@localhost RPMS]$ LS
bash: LS: no se encontró la orden...
Una orden similar es: 'ls'
[pedrico@localhost RPMS]$ ls
x86_64
[pedrico@localhost RPMS]$ cd x86_64
[pedrico@localhost x86_64]$ sudo dnf install ^
[pedrico@localhost x86_64]$ sudo dnf install kernel-4.14.5-300.pedrico.fc27.x86_64.rpm kernel-core-4.14.5-300.pedrico.fc27.x86_64.rpm kernel-modules-4.14.5-300.pedrico.fc27.x86_64.rpm
```



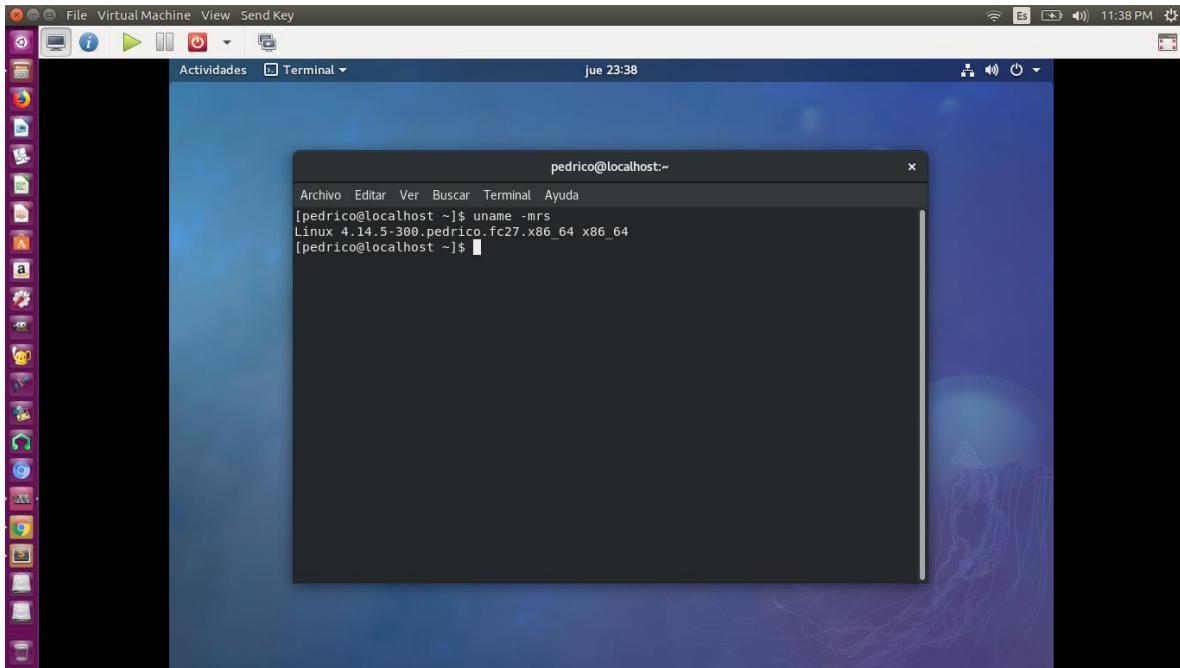
A screenshot of a Linux desktop environment. On the left is a vertical dock with icons for various applications like a browser, file manager, and system tools. In the center, there's a window titled "Actividades" (Activities) showing a file browser with a sidebar containing "Recientes" (Recent), "Carpetas personales" (Personal Folders), "Descargas" (Downloads), "Documentos" (Documents), "Imágenes" (Images), "Música" (Music), "Videos" (Videos), and "Papelería" (Paper). Overlaid on the desktop is a terminal window with a dark background. The terminal title bar says "pedrico@localhost:~/rpmbuild/RPMS/x86\_64". The terminal window contains the following command-line session:

```
jue 23:25
kernel-core-uname-r = 4.14.5-300.pedrico.fc27.x86_64 es necesario por kernel-4.14.5-300.pedrico.fc27.x86_64
kernel-modules-uname-r = 4.14.5-300.pedrico.fc27.x86_64 es necesario por kernel-4.14.5-300.pedrico.fc27.x86_64
[pedrico@localhost SPECS]$ sudo dnf install kernel-4.14.5-300.pedrico.fc27.x86_64[B+C]
[pedrico@localhost SPECS]$ ls
kernel.spec
[pedrico@localhost SPECS]$ cd ..
[pedrico@localhost rpmbuild]$ ls
BUILD BUILDROOT RPMS SOURCES SPECS SRPMs
[pedrico@localhost rpmbuild]$ cd rpms
bash: cd: rpms: No such file or directory
[pedrico@localhost RPMS]$ LS
bash: LS: no se encontró la orden...
Una orden similar es: 'ls'
[pedrico@localhost RPMS]$ ls
x86_64
[pedrico@localhost RPMS]$ cd x86_64
[pedrico@localhost x86_64]$ sudo dnf install ^
[pedrico@localhost x86_64]$ sudo dnf install kernel-4.14.5-300.pedrico.fc27.x86_64.rpm kernel-core-4.14.5-300.pedrico.fc27.x86_64.rpm kernel-modules-4.14.5-300.pedrico.fc27.x86_64.rpm
```

3. Reiniciamos la máquina y podremos elegir el nuevo kernel instalado 4.14.5-300.pedrico.fc27.x86\_64. Podemos ver que el identificador colocado en el archivo kernel.spec aparece en el listado de kernels a elegir.



4. Verificamos la versión de kernel instalado con el siguiente comando:



## vii. Módulos

Los módulos del núcleo (kernel) son fragmentos de código que pueden ser cargados y eliminados del núcleo bajo demanda. Extienden la funcionalidad del núcleo sin necesidad de reiniciar el sistema. A continuación generaremos dos módulos informativos. El primero nos brindará información de la memoria RAM y el segundo información del CPU.

### a. Módulo de memoria RAM

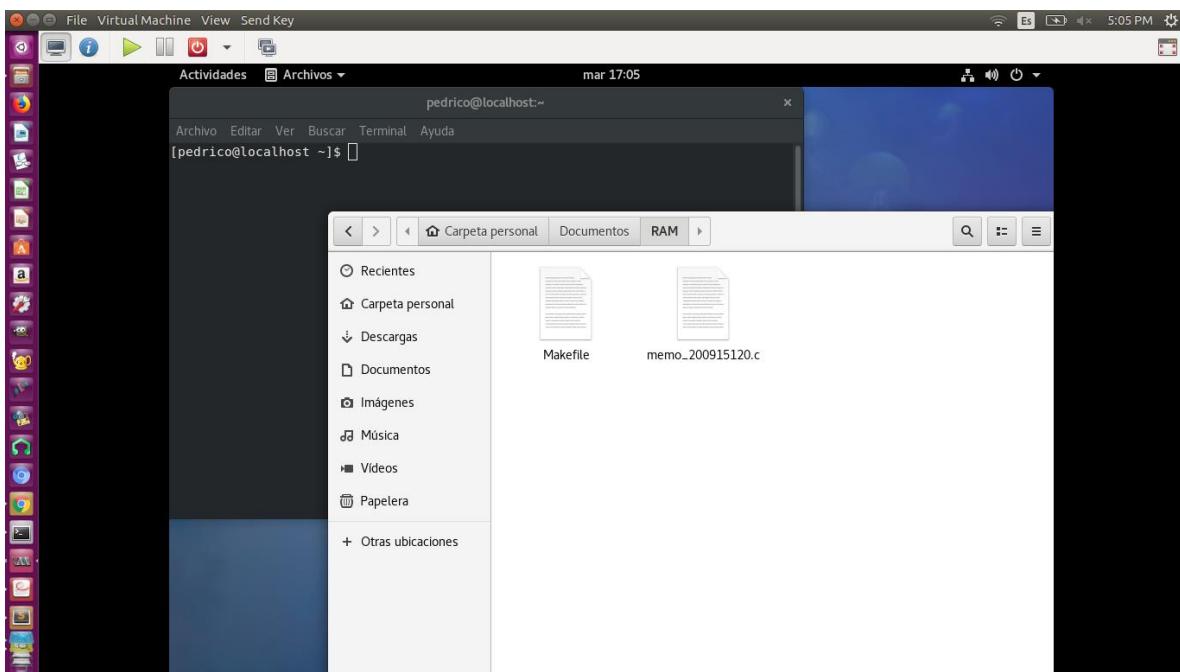
En este módulo se sobrescribe un archivo en el directorio /proc con la siguiente información:

- Carné
- Nombre
- Sistema operativo
- Memoria total
- Memoria libre
- % de memoria utilizada

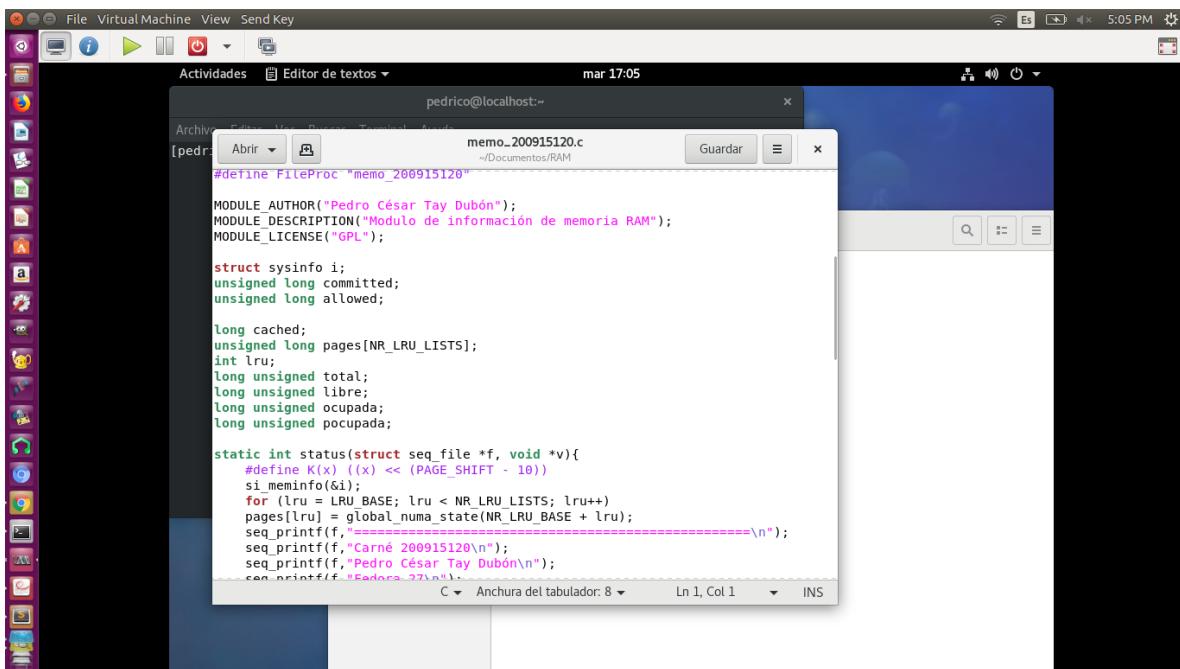
Al cargar el módulo se registrará el número de carnet (200915120) y al descargar el módulo se registra el nombre del curso (Sistemas Operativos 1).

La información es obtenida por medio del struct “sysinfo” y se actualiza cada vez que se abre el archivo generado.

## 1. Archivos fuente.



2. Archivo memo\_200915120. Contiene los procedimientos para generar el módulo y la lógica que este contendrá al abrir el archivo asociado.



3. Makefile. Archivo que contiene las instrucciones para compilar el archivo memo\_200915120.

A screenshot of a Linux desktop environment. On the left is a vertical application menu with icons for various applications like a web browser, file manager, and system tools. In the center, there is a terminal window titled "pedrico@localhost:~" with the command "makefile" running. The terminal shows a Makefile with the following content:

```
obj-m := memo_200915120.o
KDIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)

default:
    $(MAKE) -C $(KDIR) M=$(PWD) modules
```

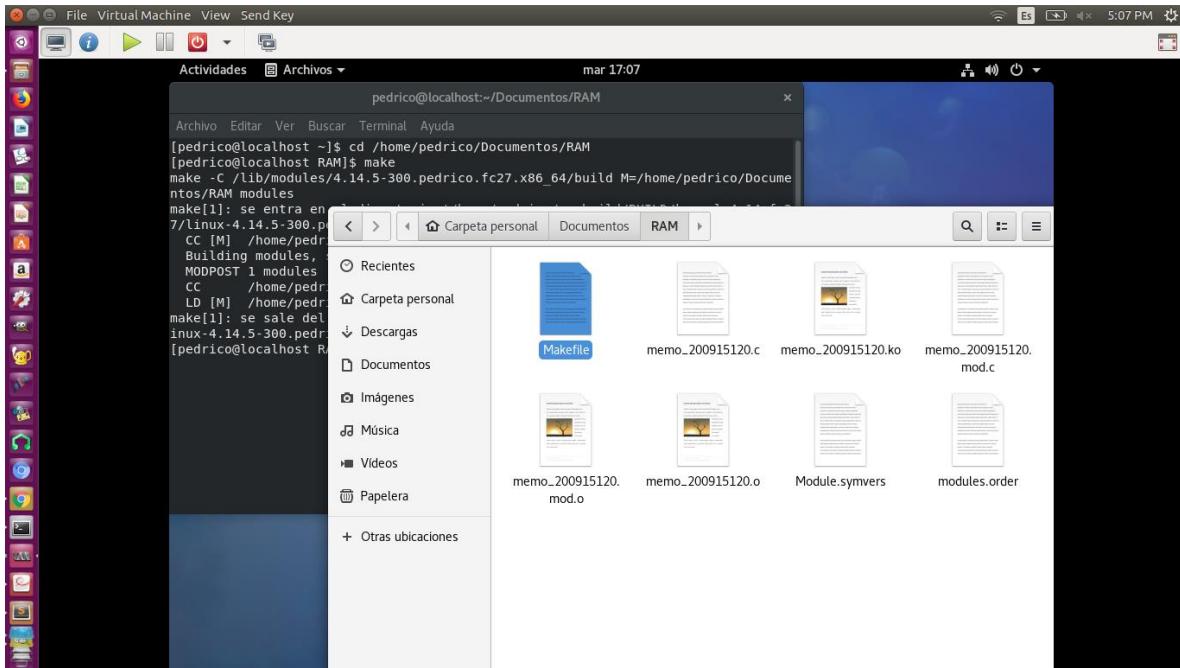
The terminal also displays the current date and time: "mar 17:05". Below the terminal, a file browser window is open, showing a directory structure with files like "memo\_200915120.ko", "memo\_200915120.mod.c", "module.symvers", and "modules.order".

4. Nos posicionamos en el directorio de los archivos fuente y ejecutamos el comando "make". Este nos generará varios archivos de salida.

A screenshot of a Linux desktop environment. On the left is a vertical application menu with icons for various applications like a web browser, file manager, and system tools. In the center, there is a terminal window titled "pedrico@localhost:~/Documentos/RAM" with the command "make" running. The terminal shows the output of the make command, which includes:

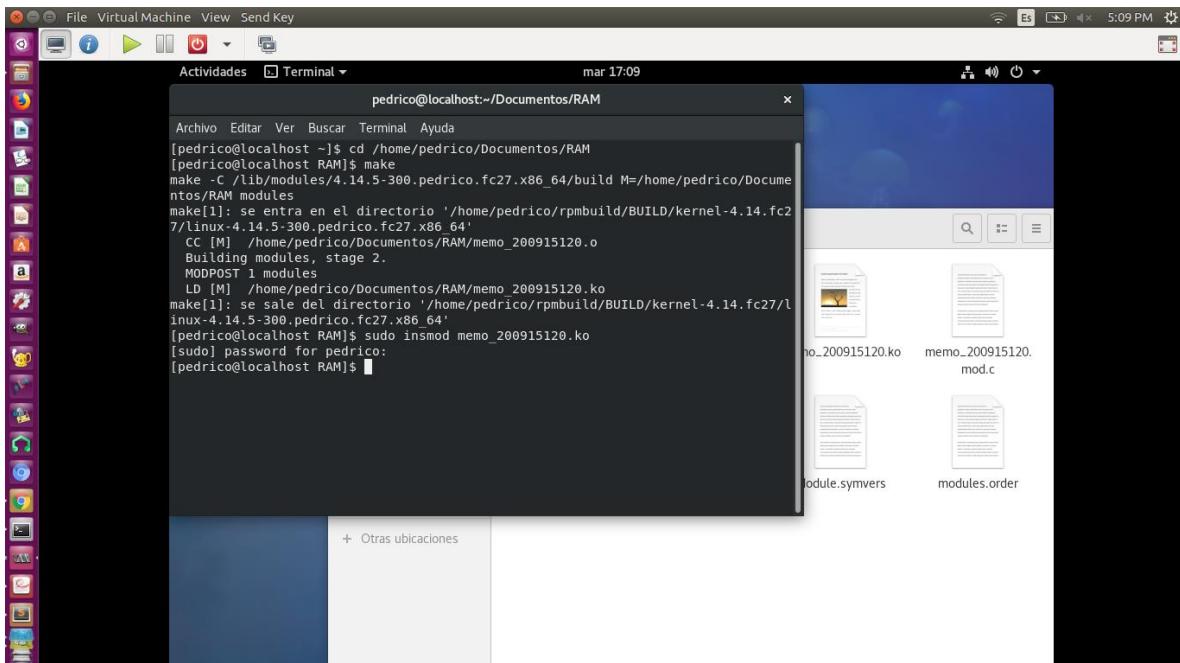
```
[pedrico@localhost ~]$ cd /home/pedrico/Documentos/RAM
[pedrico@localhost RAM]$ make
make -C /lib/modules/4.14.5-300.pedrico.fc27.x86_64/build M=/home/pedrico/Documentos/RAM modules
make[1]: se entra en el directorio '/home/pedrico/rpmbuild/BUILD/kernel-4.14.fc27/linux-4.14.5-300.pedrico.fc27.x86_64'
  CC [M]  /home/pedrico/Documentos/RAM/memo_200915120.o
  Building modules, stage 2.
  MODPOST 1 modules
  LD [M]  /home/pedrico/Documentos/RAM/memo_200915120.ko
make[1]: se sale del directorio '/home/pedrico/rpmbuild/BUILD/kernel-4.14.fc27/linux-4.14.5-300.pedrico.fc27.x86_64'
[pedrico@localhost RAM]$ sudo insmod memo_200915120.ko
[sudo] password for pedrico:
[pedrico@localhost RAM]$
```

The terminal also displays the current date and time: "mar 17:09". To the right of the terminal, a file browser window is open, showing a directory structure with files like "memo\_200915120.ko", "memo\_200915120.mod.c", "module.symvers", and "modules.order".



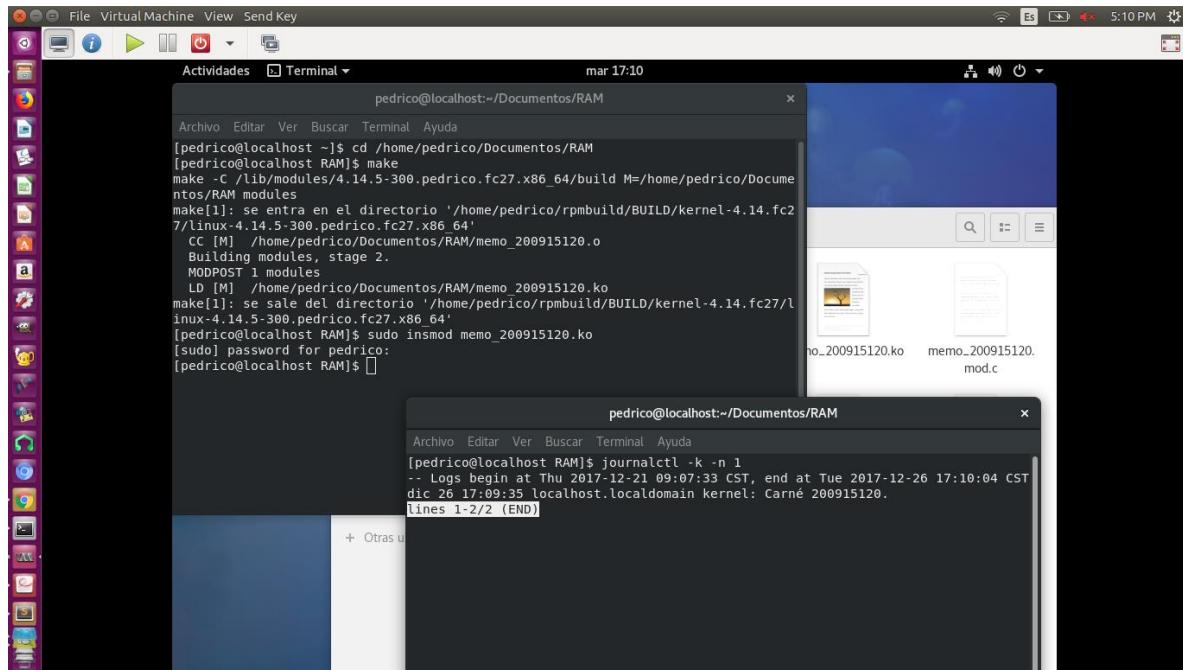
## 5. Instalamos el modulo ejecutando:

```
sudo insmod memo_200915120.ko
```



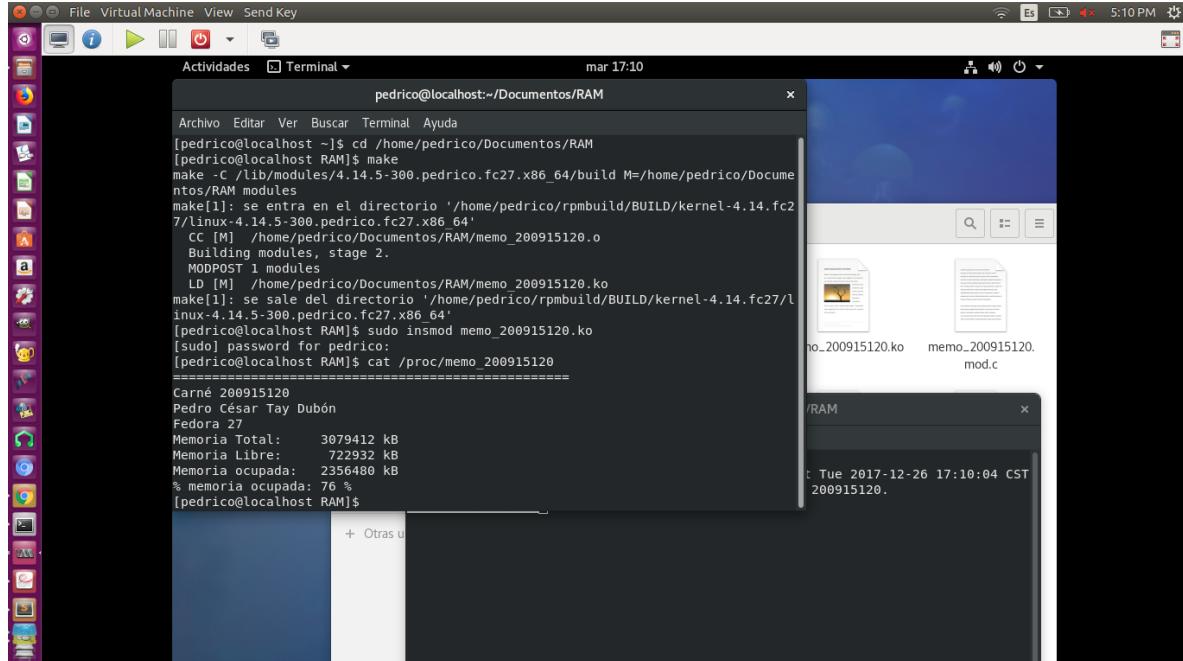
Verificamos en el archivo de log el registro que genera éste módulo al ser insertado: "Carné 200915120" ejecutando:

```
journalctl -k -n 1
```



6. Abrimos el archivo asociado al módulo. Para ello ejecutamos:

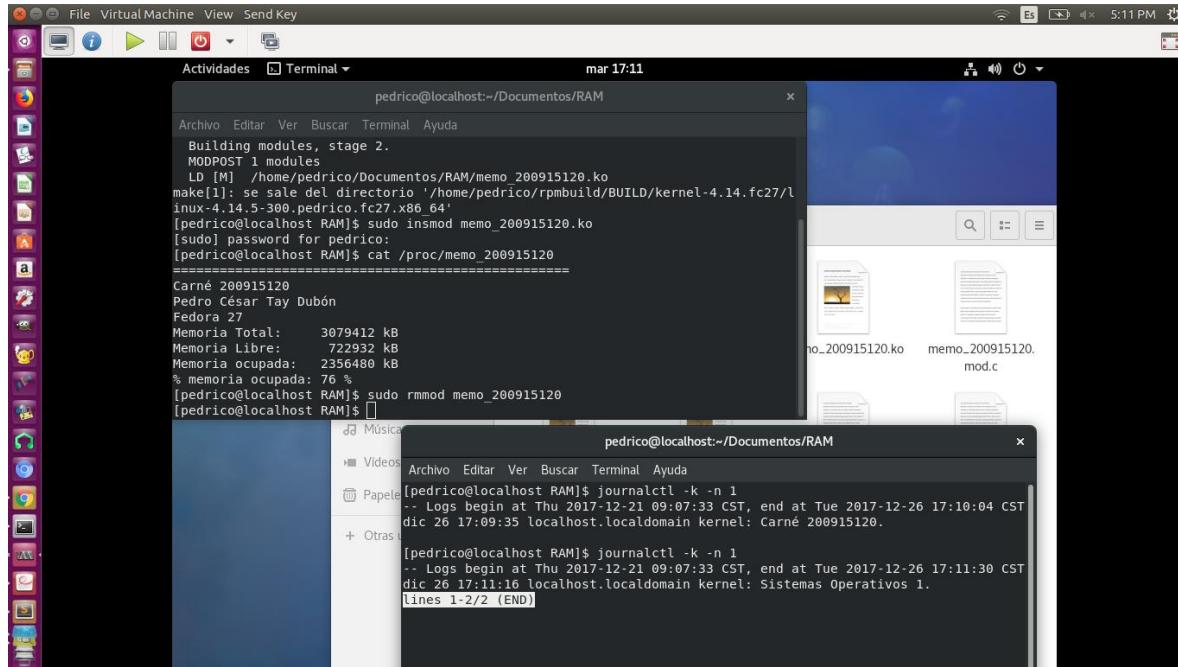
```
cat /proc/meminfo
```



Podemos observar la información generada por el archivo memo\_200915120

## 7. Desinstalamos el modulo ejecutando:

```
sudo rmmod memo_200915120
```



Ejecutamos el siguiente comando para verificar el registro generado al desinstalar el modulo “Sistemas Operativos 1”:

```
journalctl -k -n 1
```

### b. Módulo de CPU

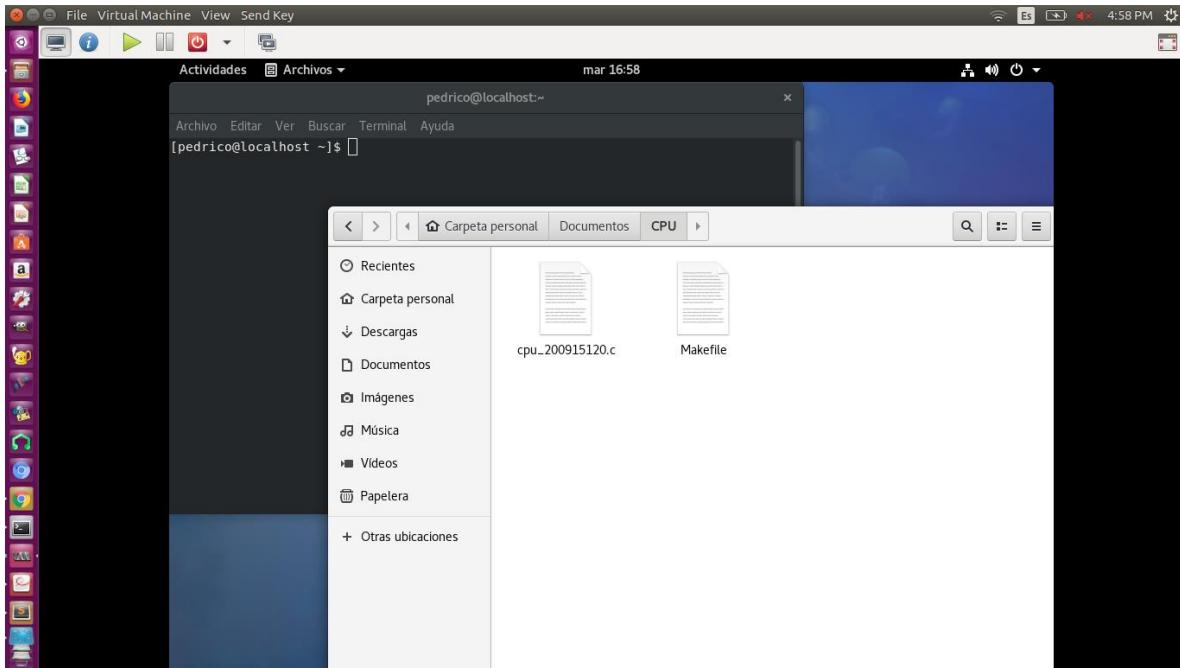
En este módulo se sobrescribe un archivo en el directorio /proc (cpu\_200915120) con la siguiente información:

- Carné
- Nombre
- Sistema operativo
- Lista de procesos (PID, nombre y estado)

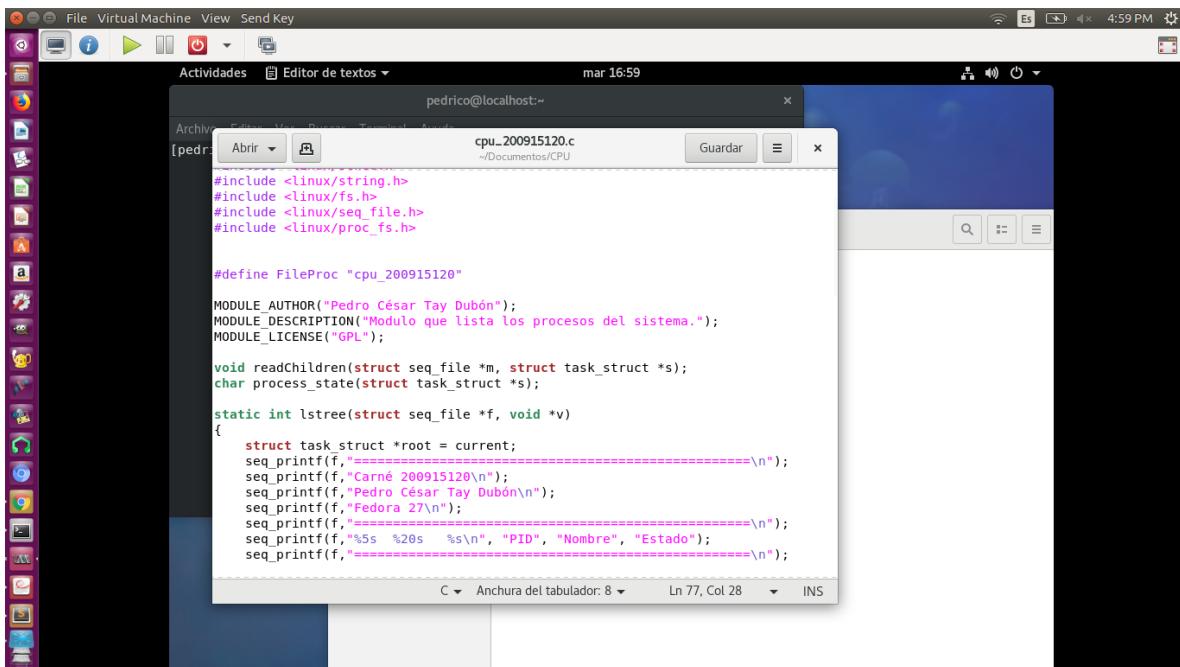
Al cargar el módulo se registrará mi nombre (Pedro César Tay Dubón) y al descargar el módulo se registra el nombre del curso (Sistemas Operativos 1).

La información es obtenida por medio de struct de procesos y se actualiza cada vez que se abre el archivo generado.

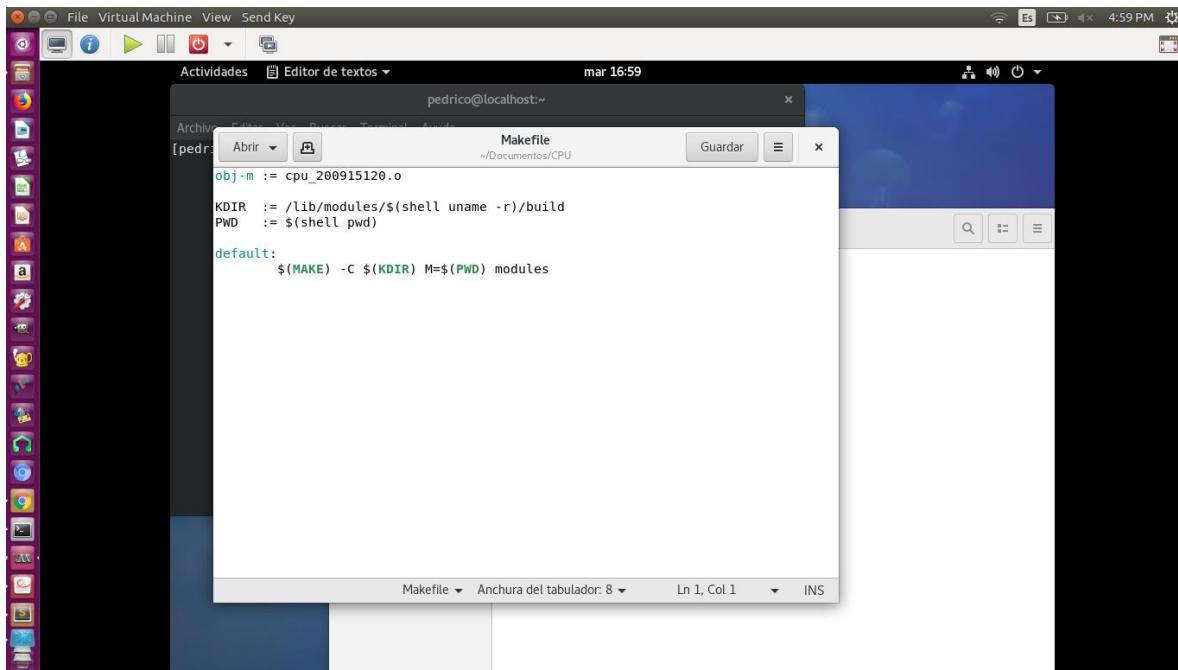
## 8. Archivos fuente.



9. Archivo `cpu_200915120`. Contiene los procedimientos para generar el módulo y la lógica que este contendrá al abrir el archivo asociado.



10. Makefile. Archivo que contiene las instrucciones para compilar el archivo cpu\_200915120.



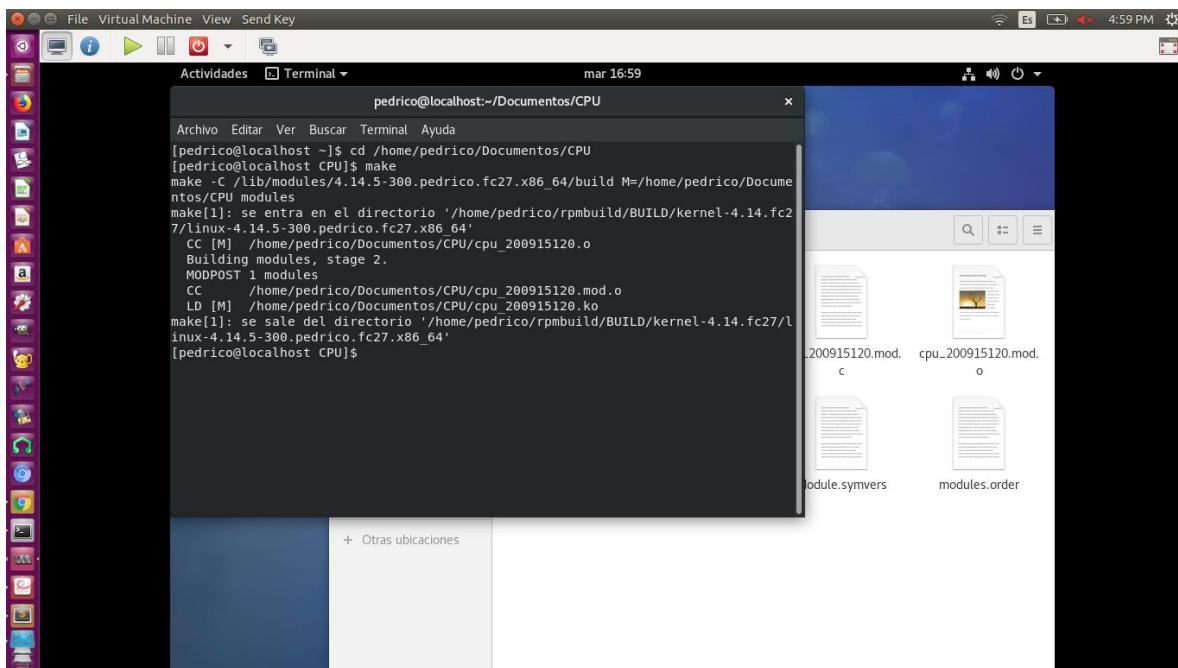
A screenshot of a Linux desktop environment. On the left is a vertical application menu bar. In the center, there is a terminal window titled "pedrico@localhost:~" with the command "makefile" running, displaying the content of a Makefile. To the right of the terminal is a file browser window showing files related to the kernel build, including "cpu\_200915120.mod.c", "cpu\_200915120.mod.o", "module.symvers", and "modules.order".

```
obj-m := cpu_200915120.o

KDIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)

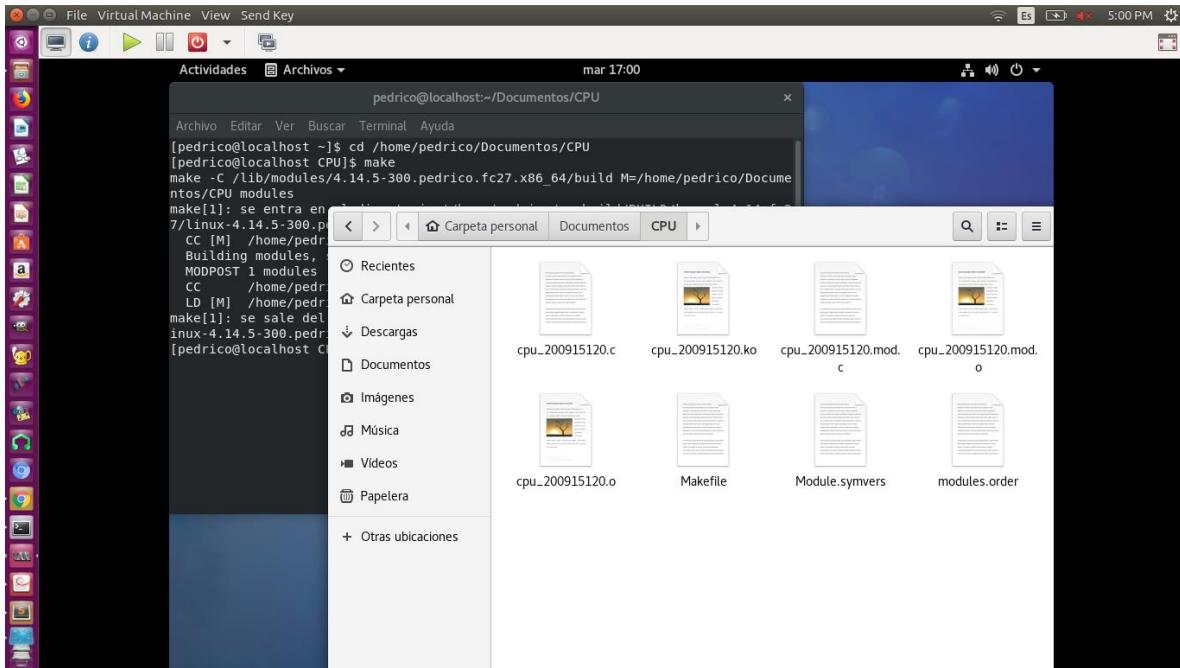
default:
    $(MAKE) -C $(KDIR) M=$(PWD) modules
```

11. Nos posicionamos en el directorio de los archivos fuente y ejecutamos el comando "make". Este nos generará varios archivos de salida.



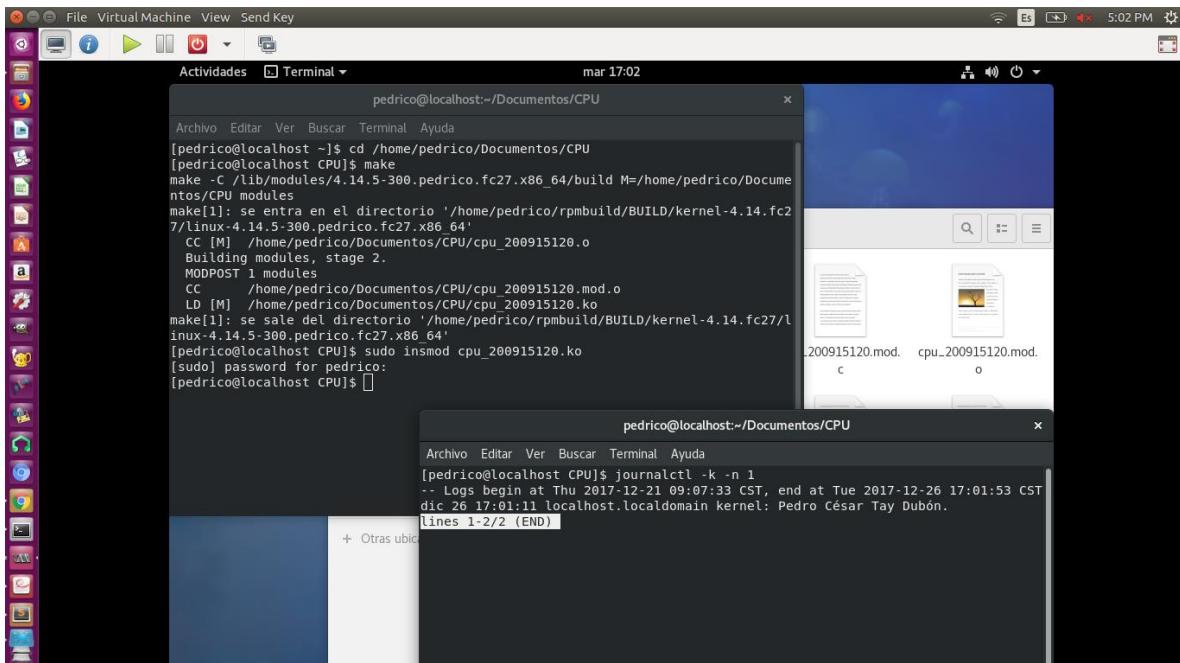
A screenshot of a Linux desktop environment. On the left is a vertical application menu bar. In the center, there is a terminal window titled "pedrico@localhost:~/Documentos/CPU" with the command "make" running, showing the output of the kernel module compilation. To the right of the terminal is a file browser window showing the generated kernel module files: "cpu\_200915120.mod.c", "cpu\_200915120.mod.o", "module.symvers", and "modules.order".

```
[pedrico@localhost ~]$ cd /home/pedrico/Documentos/CPU
[pedrico@localhost CPU]$ make
make -C /lib/modules/4.14.5-300.pedrico.fc27.x86_64/build M=/home/pedrico/Documentos/CPU modules
make[1]: se entra en el directorio '/home/pedrico/rpmbuild/BUILD/kernel-4.14.fc27/linux-4.14.5-300.pedrico.fc27.x86_64'
CC [M] /home/pedrico/Documentos/CPU/cpu_200915120.o
Building modules, stage 2.
MODPOST 1 modules
CC      /home/pedrico/Documentos/CPU/cpu_200915120.mod.o
LD [M]  /home/pedrico/Documentos/CPU/cpu_200915120.ko
make[1]: se sale del directorio '/home/pedrico/rpmbuild/BUILD/kernel-4.14.fc27/linux-4.14.5-300.pedrico.fc27.x86_64'
[pedrico@localhost CPU]$
```



## 12. Instalamos el modulo ejecutando:

```
sudo insmod cpu_200915120.ko
```

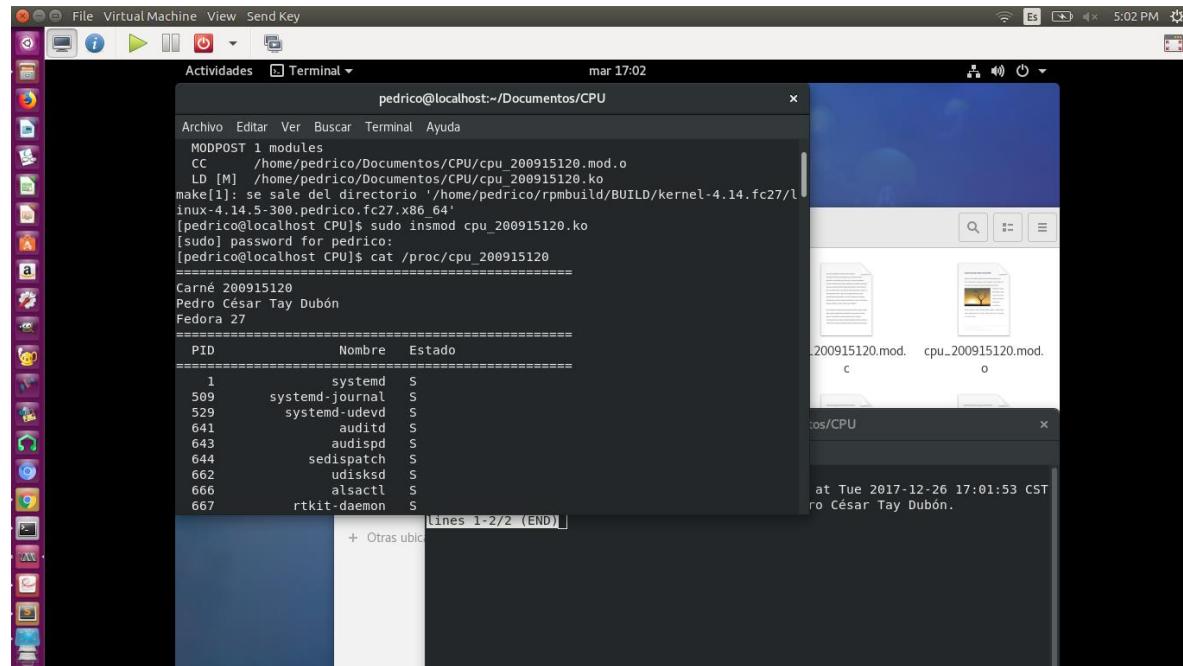


Verificamos en el archivo de log el registro que genera éste módulo al ser insertado: "Pedro César Tay Dubón" ejecutando:

```
journalctl -k -n 1
```

13. Abrimos el archivo asociado al módulo. Para ello ejecutamos:

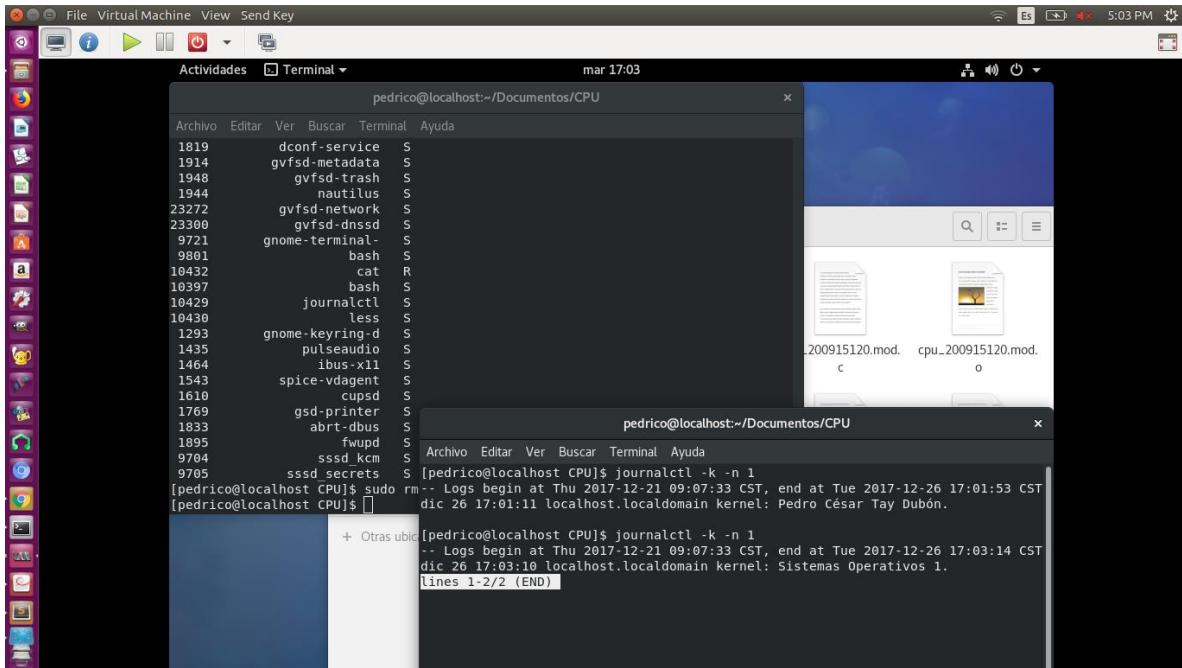
```
cat /proc/cpu_200915120
```



Podemos observar la información generada por el archivo cpu\_200915120.

14. Desinstalamos el modulo ejecutando:

```
sudo rmmod cpu_200915120
```



Ejecutamos el siguiente comando para verificar el registro generado al desinstalar el modulo “Sistemas Operativos 1”:

[journalctl -k -n 1](#)



