Luis Chuta 1320016

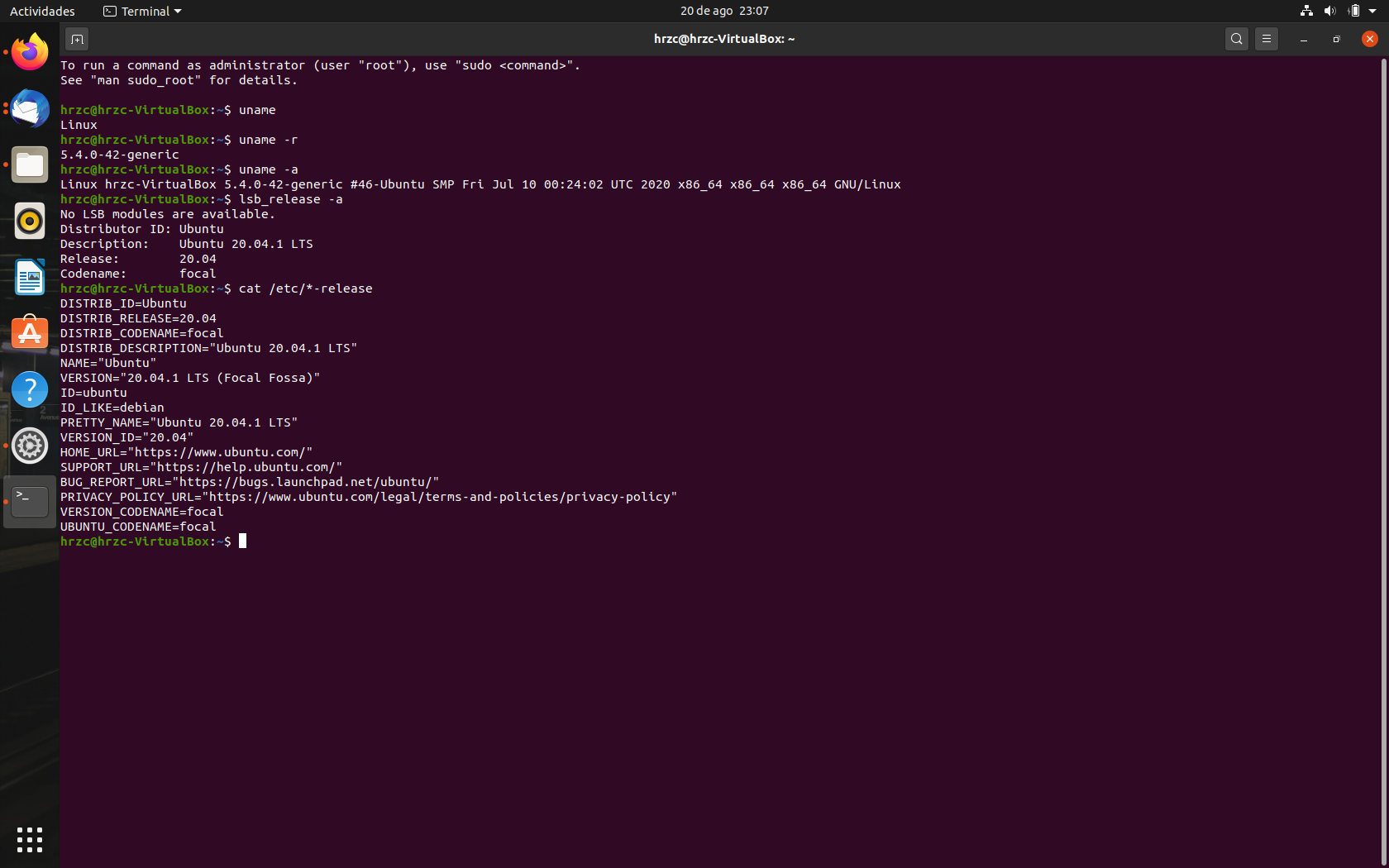
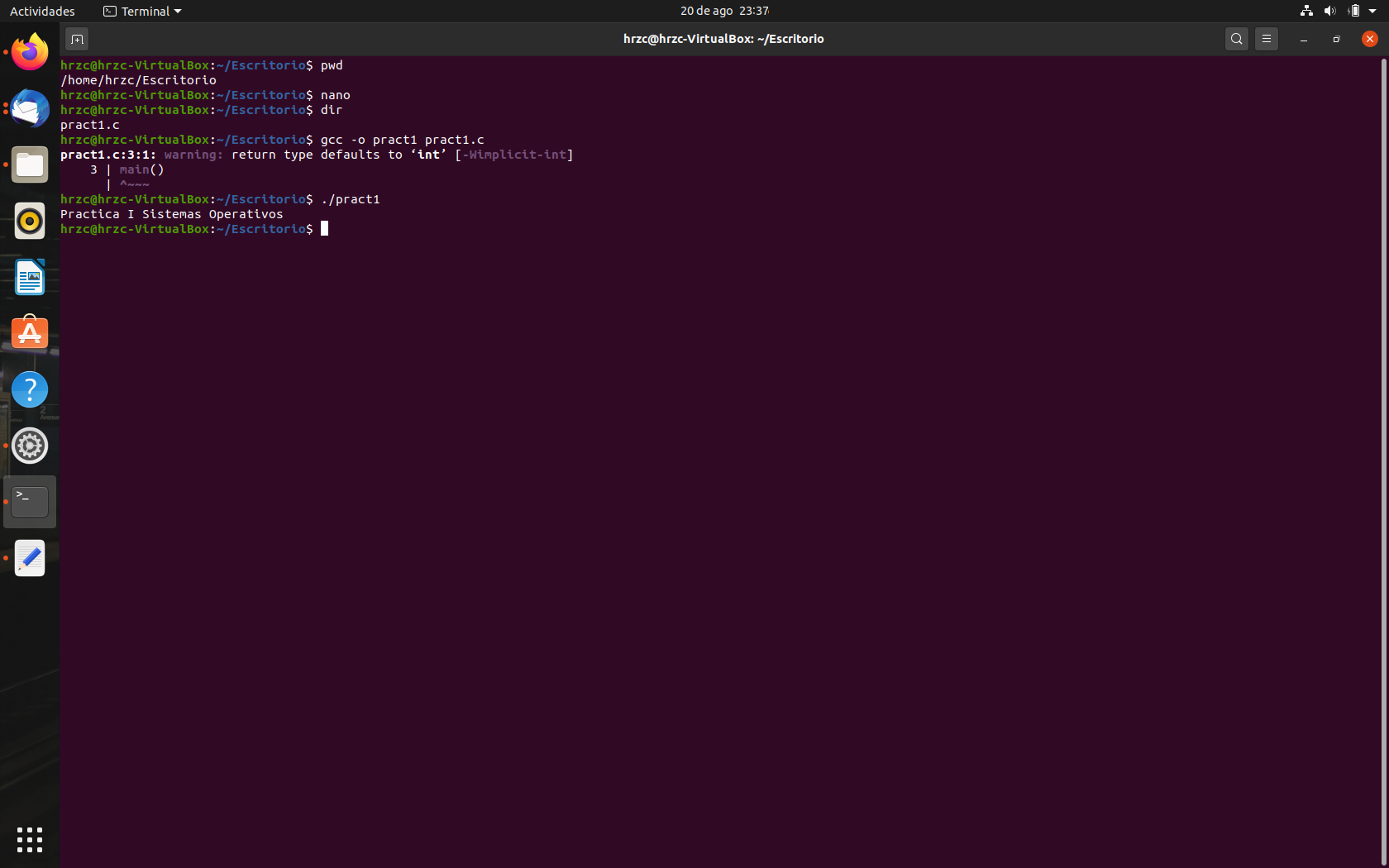
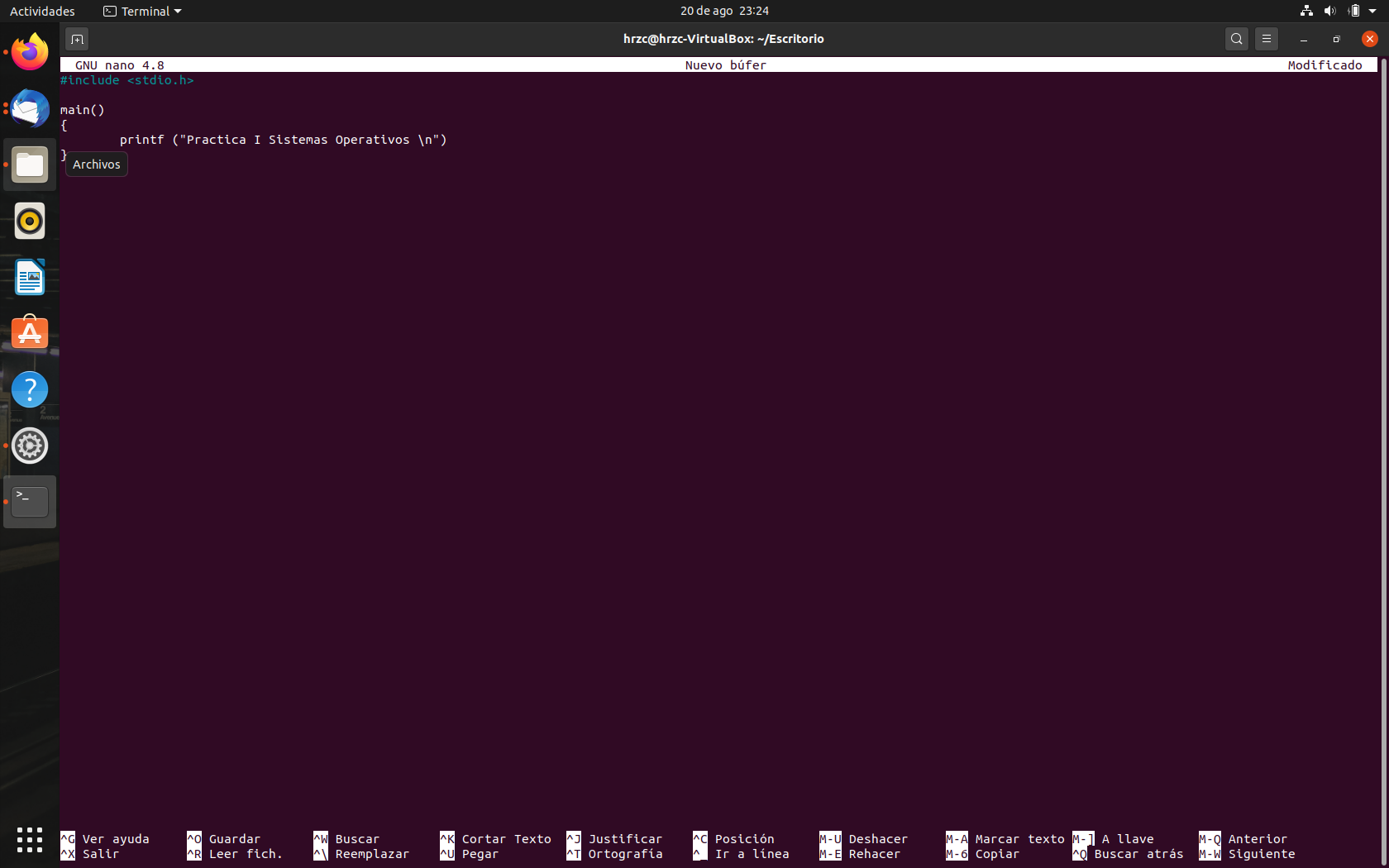
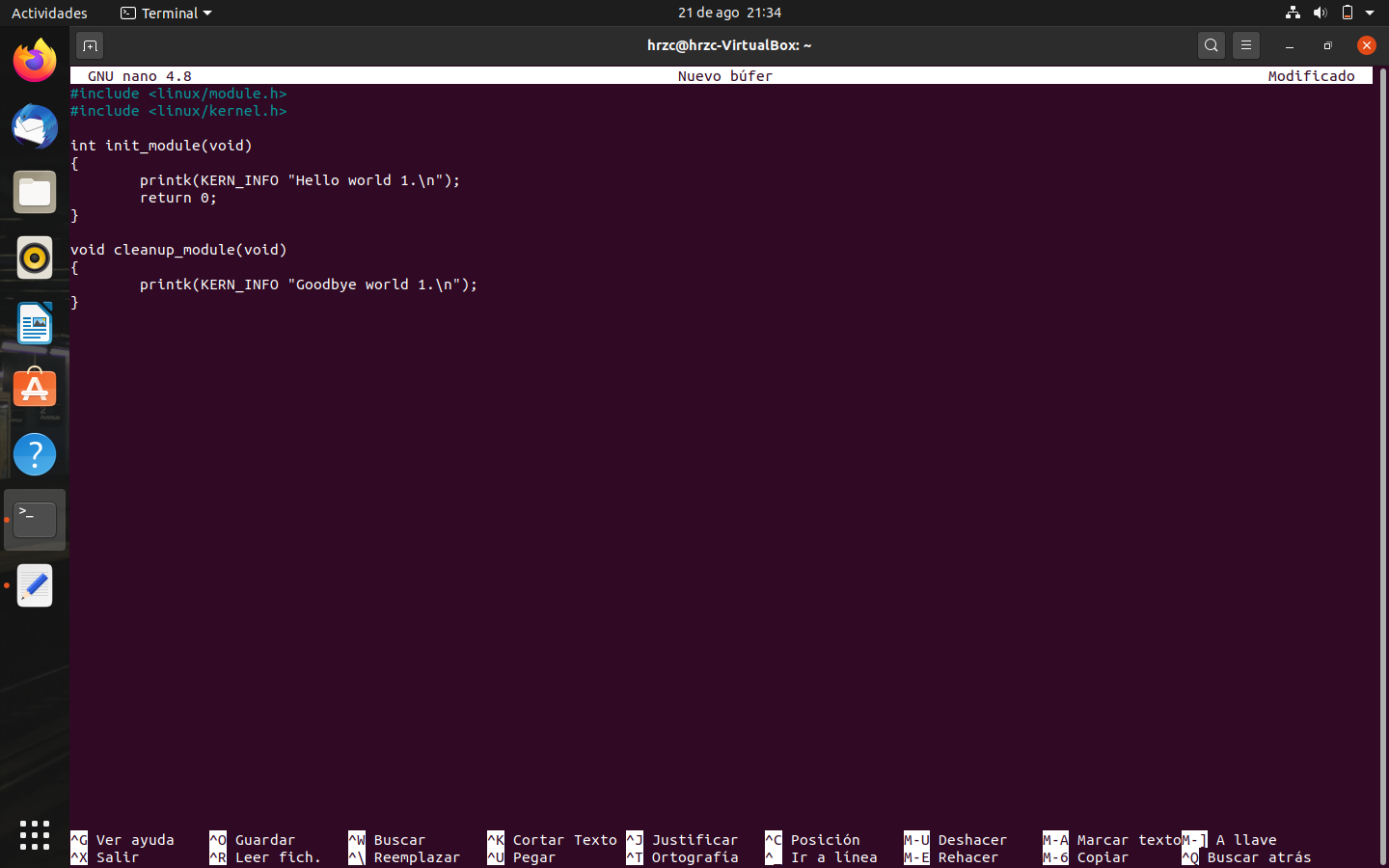
Luis Escobar 1310617

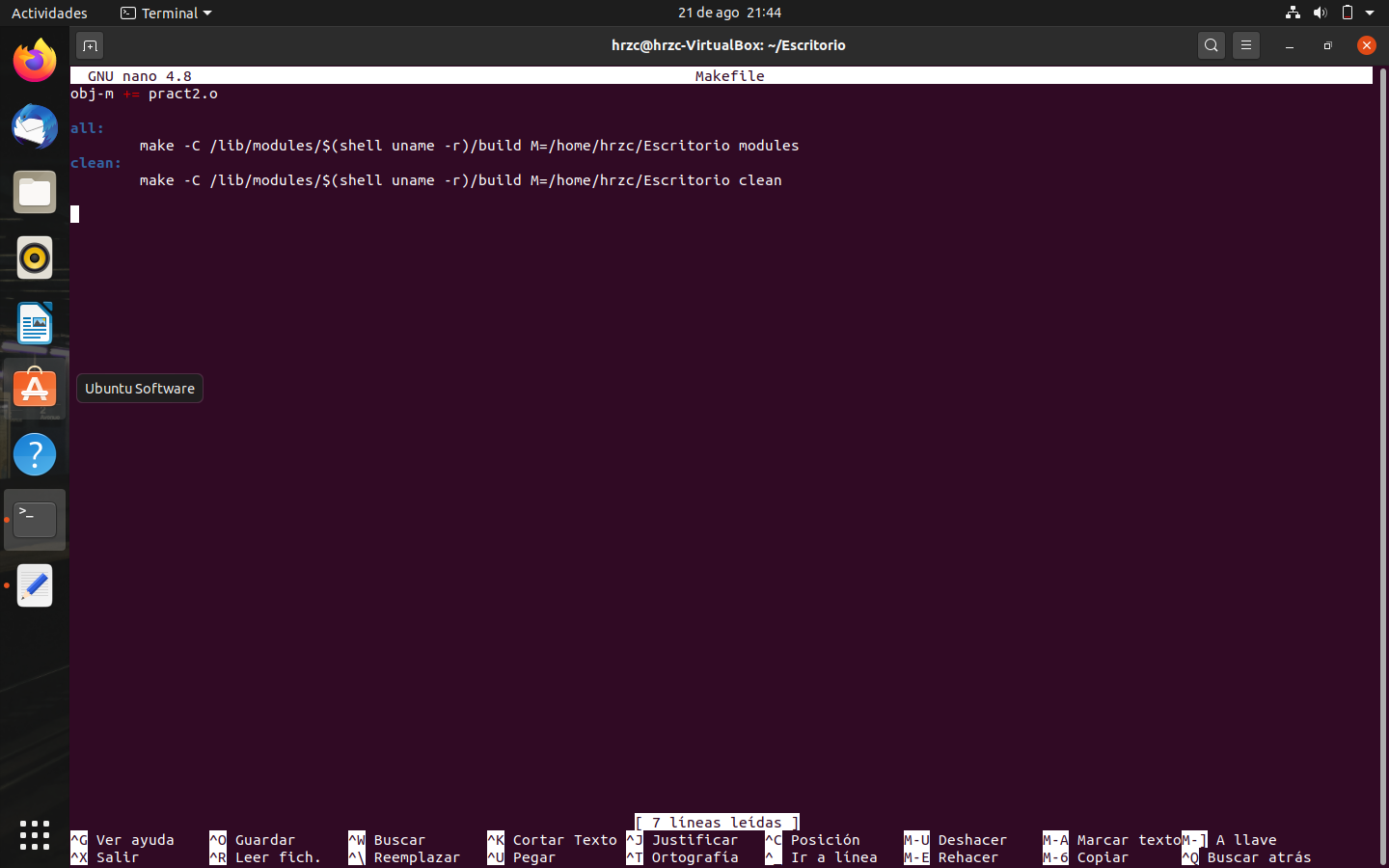
Jose Melendez 1059918

Hector Zetino 1295617

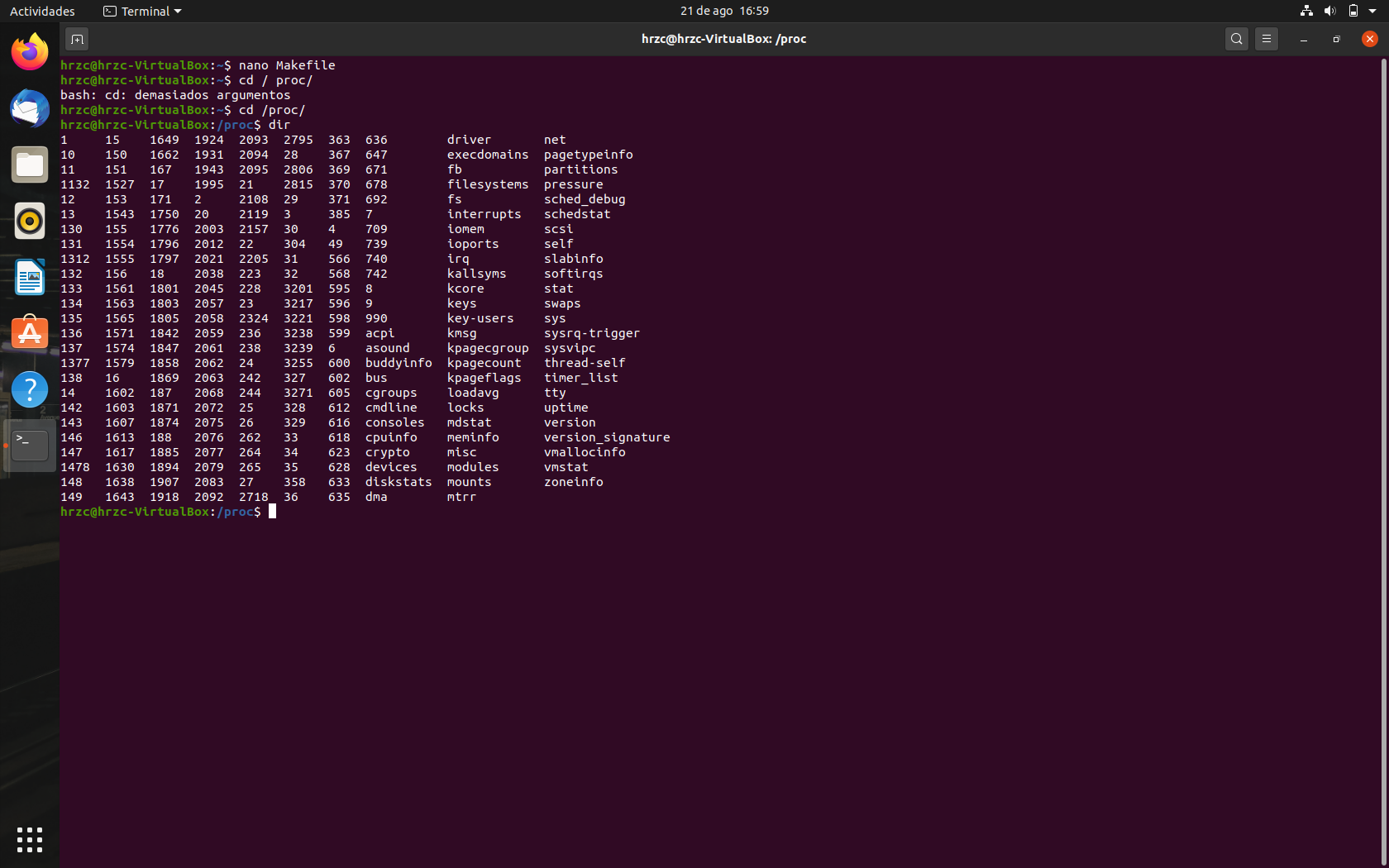
**Practica No. 1(Capturas de pantalla de la práctica)**

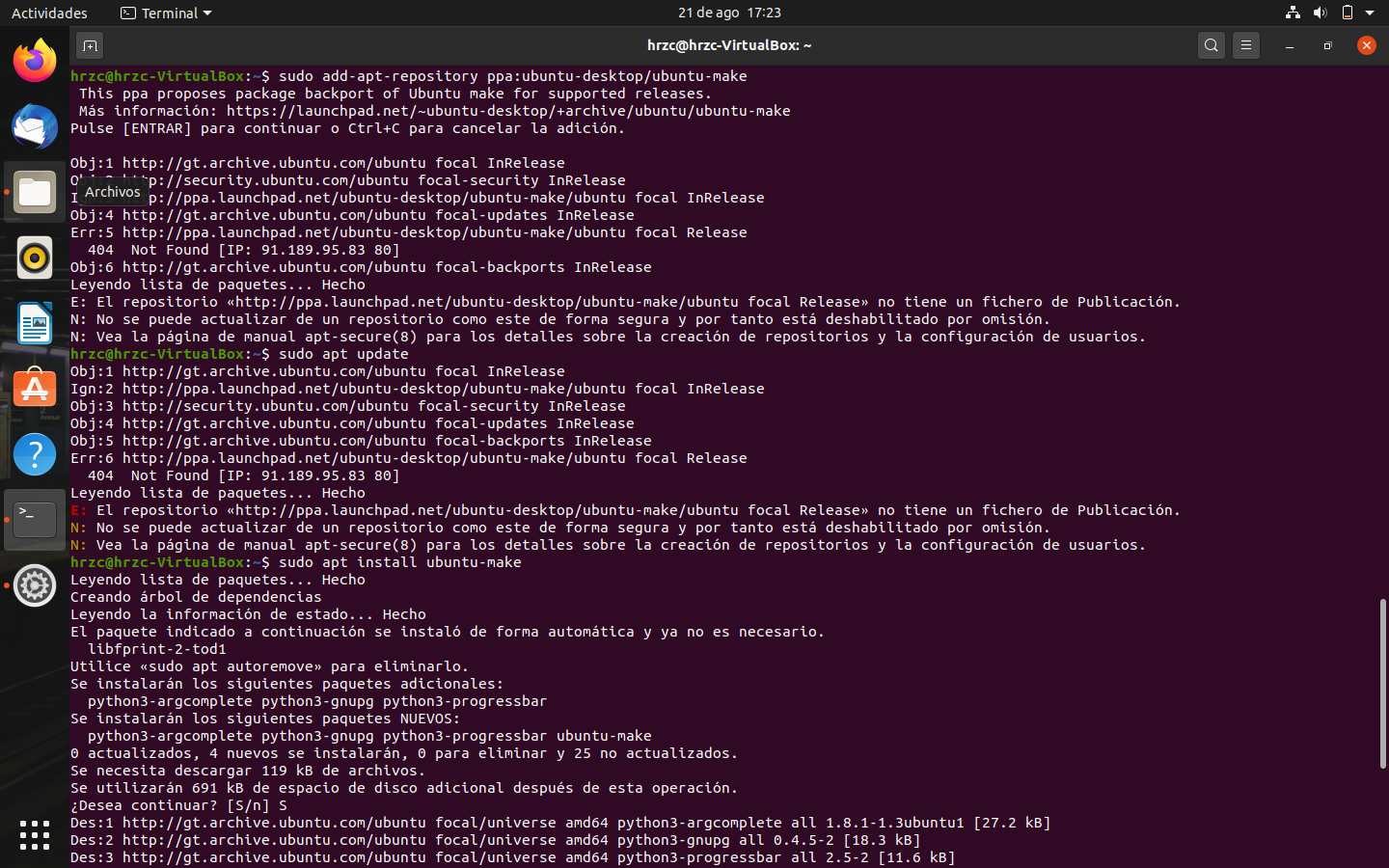
**Ubuntu 20.4**

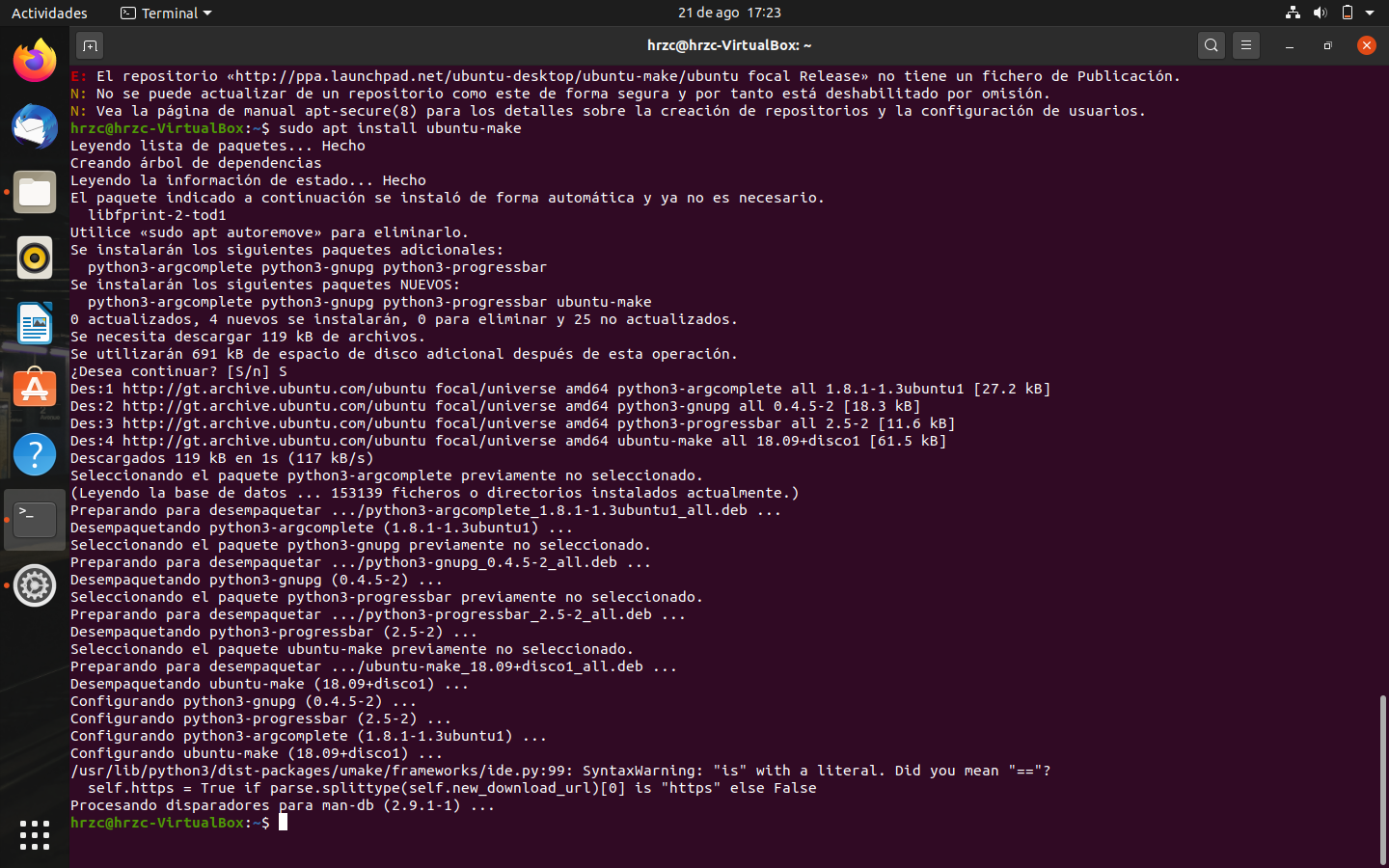
1. Prueba de comandos básicos para uso de la practica
2. ****Creación del archivo con nano, y posteriormente utilizando GCC (compilador integrado para uso C, C++, Fortran) desde la terminal de Ubuntu, además de ejecución del programa que fue escrito en lenguaje C.
3. ****Programa escrito en C, nombrado pract1.c
4. Creación de un módulo básico dentro de la línea de comandos de Ubuntu
5. Creación del archivo Makefile (conjunto de tarea que se desea ejecutar)

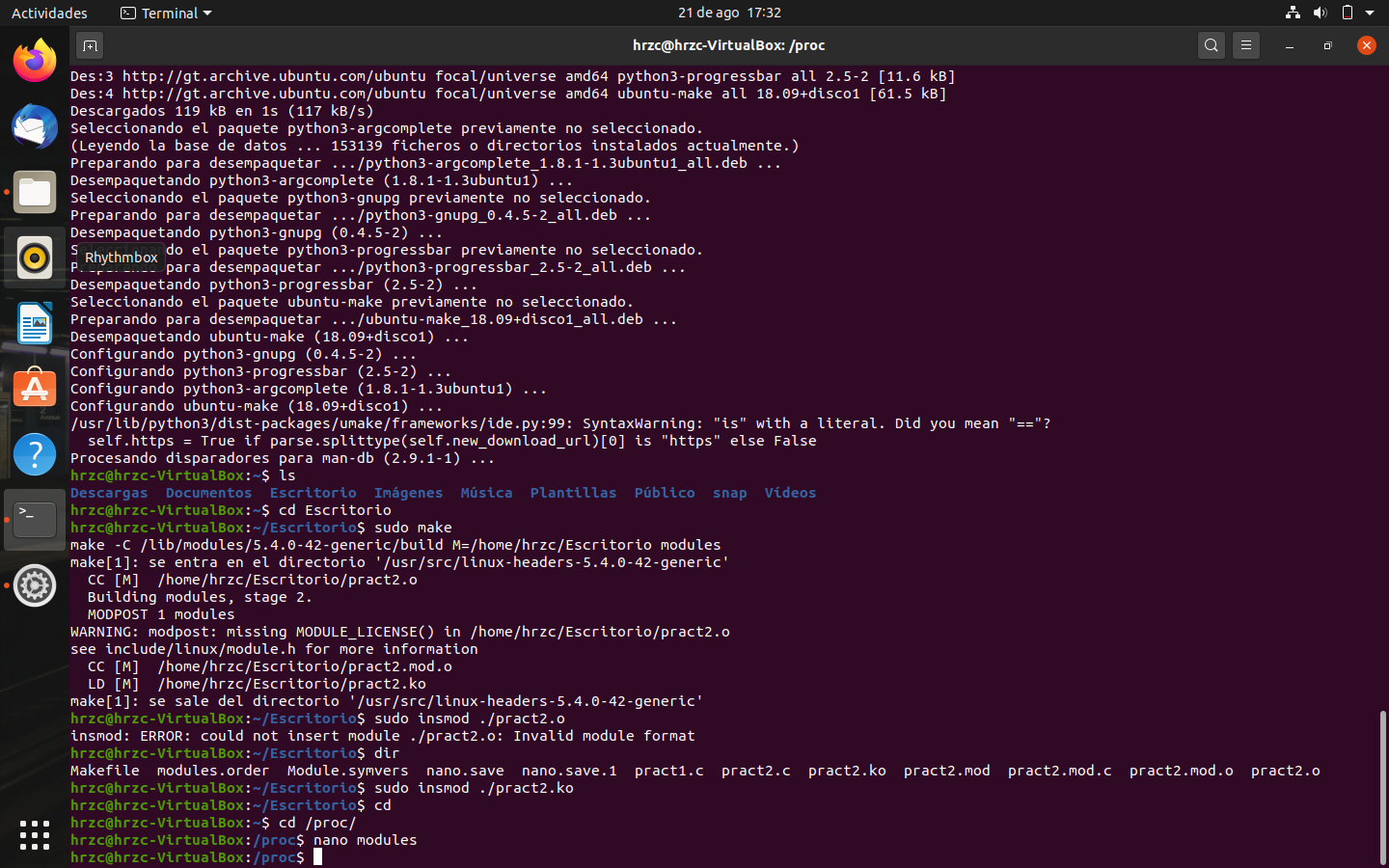


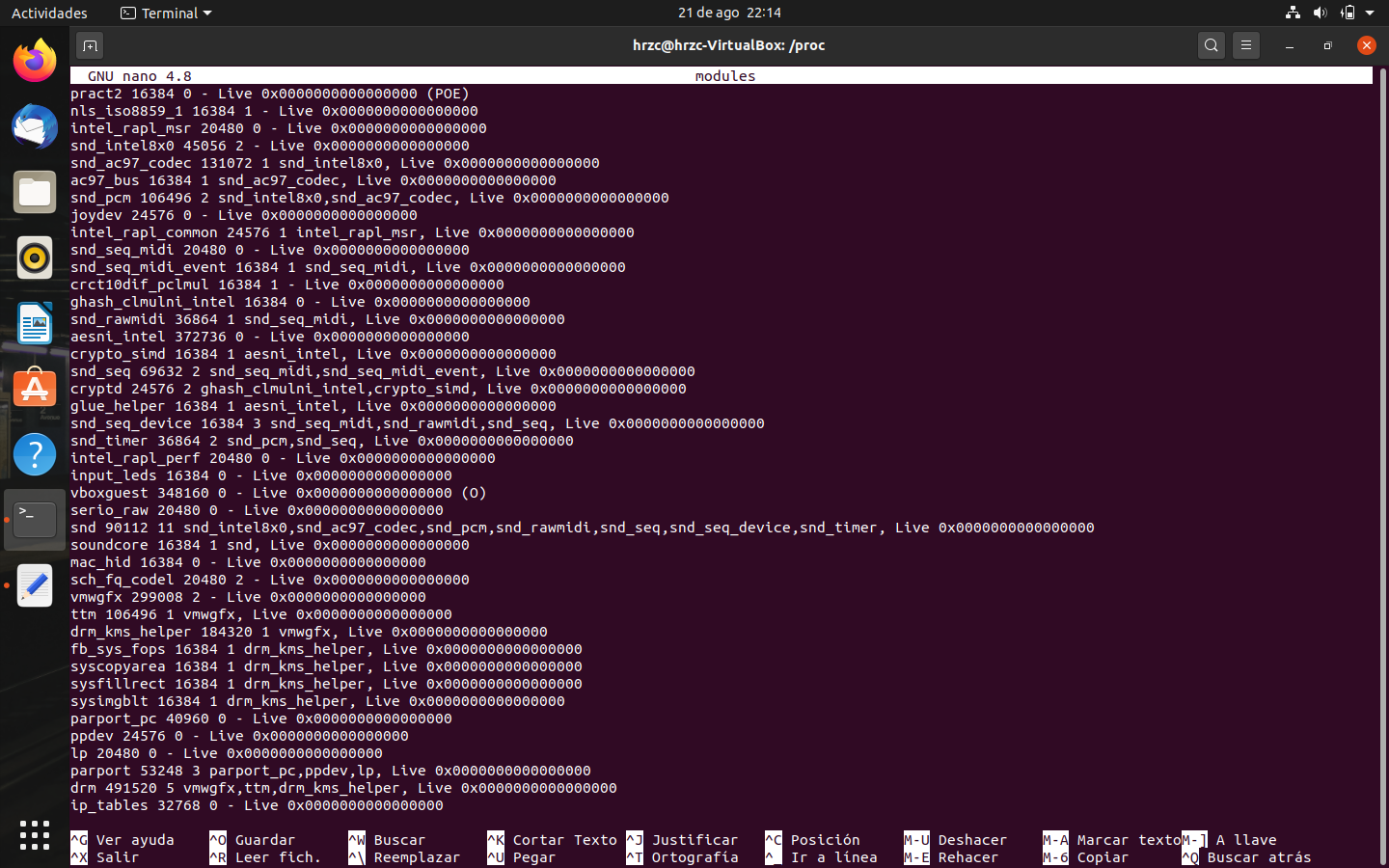
1. Listado de módulos presentes dentro del kernel

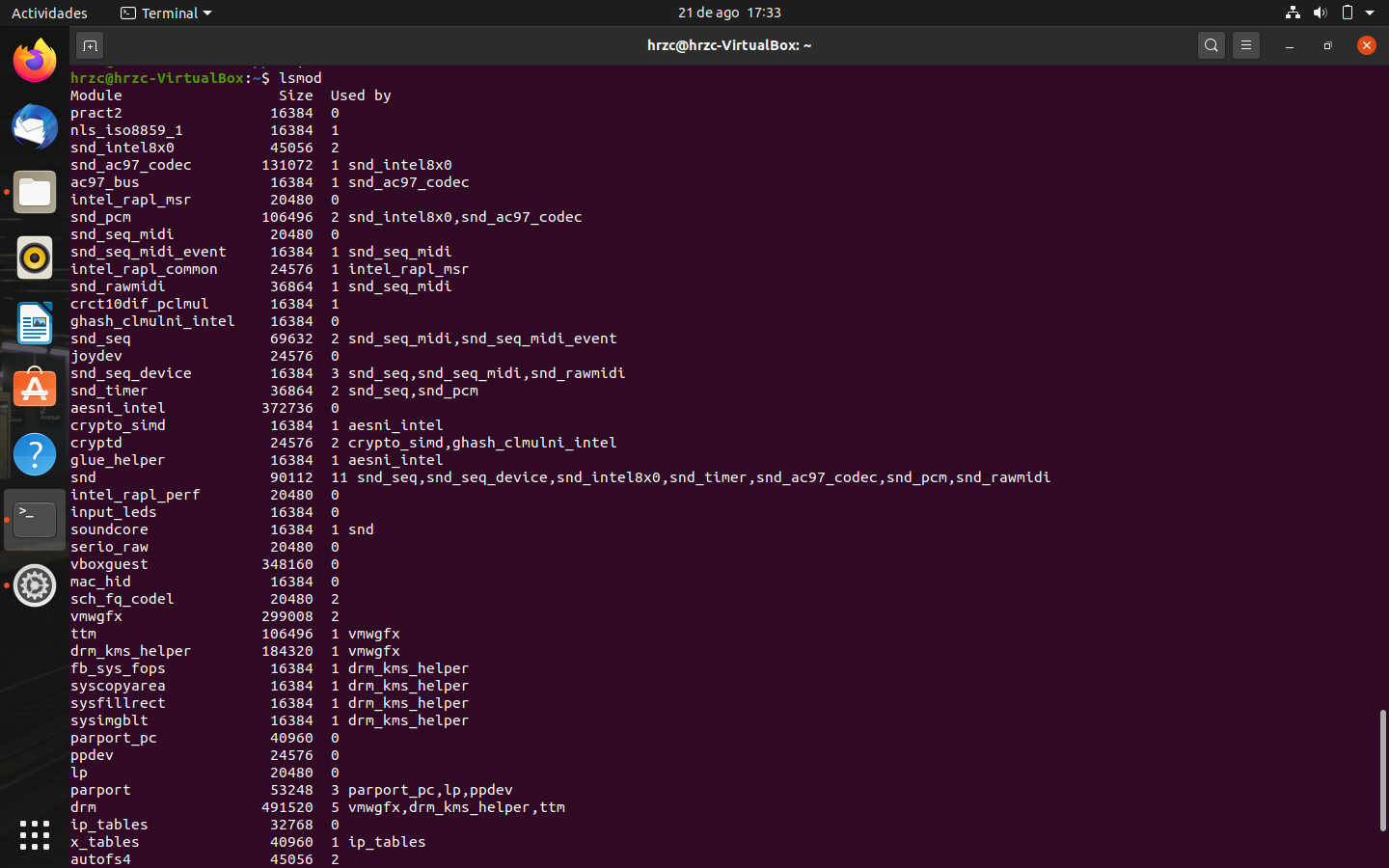


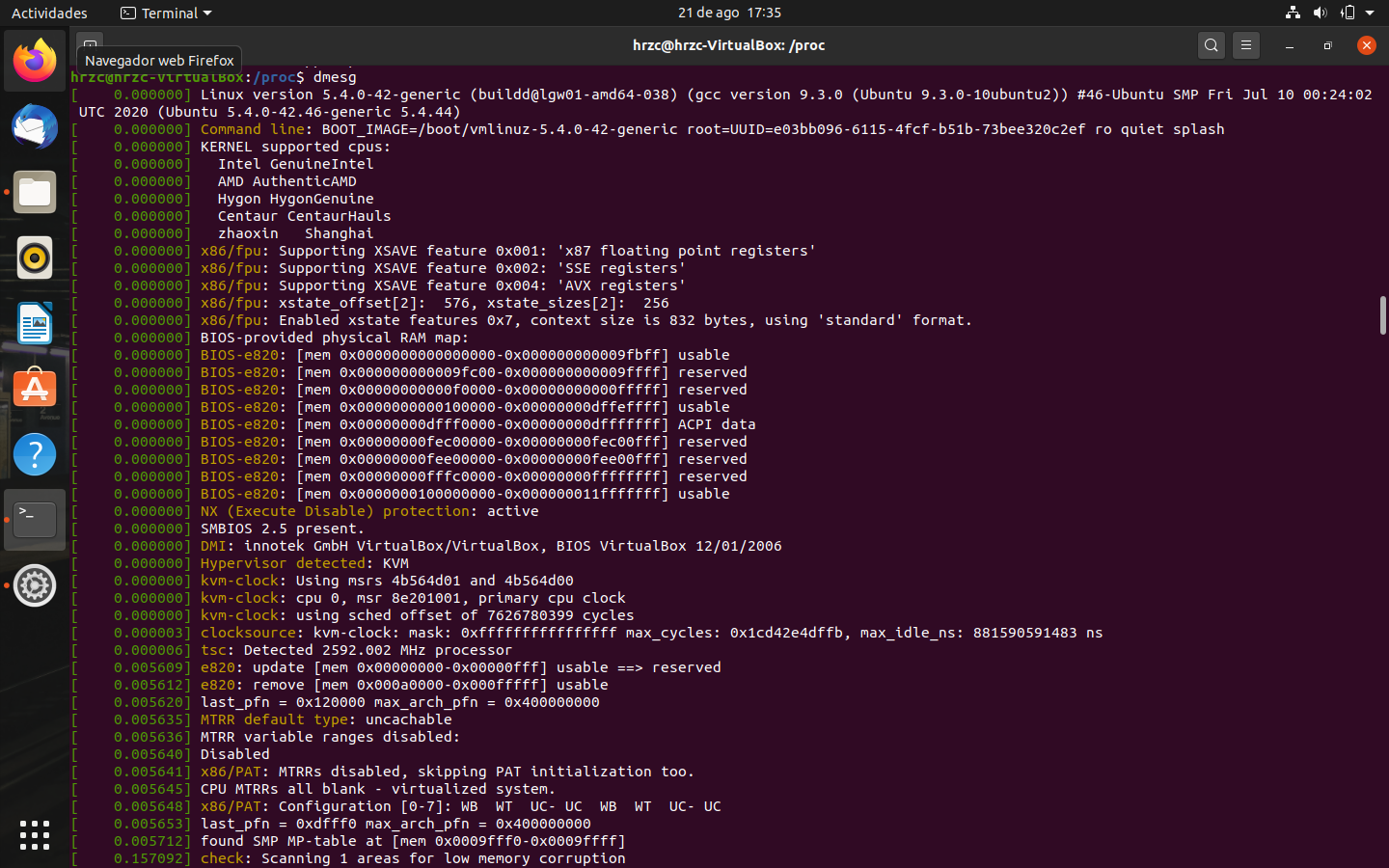
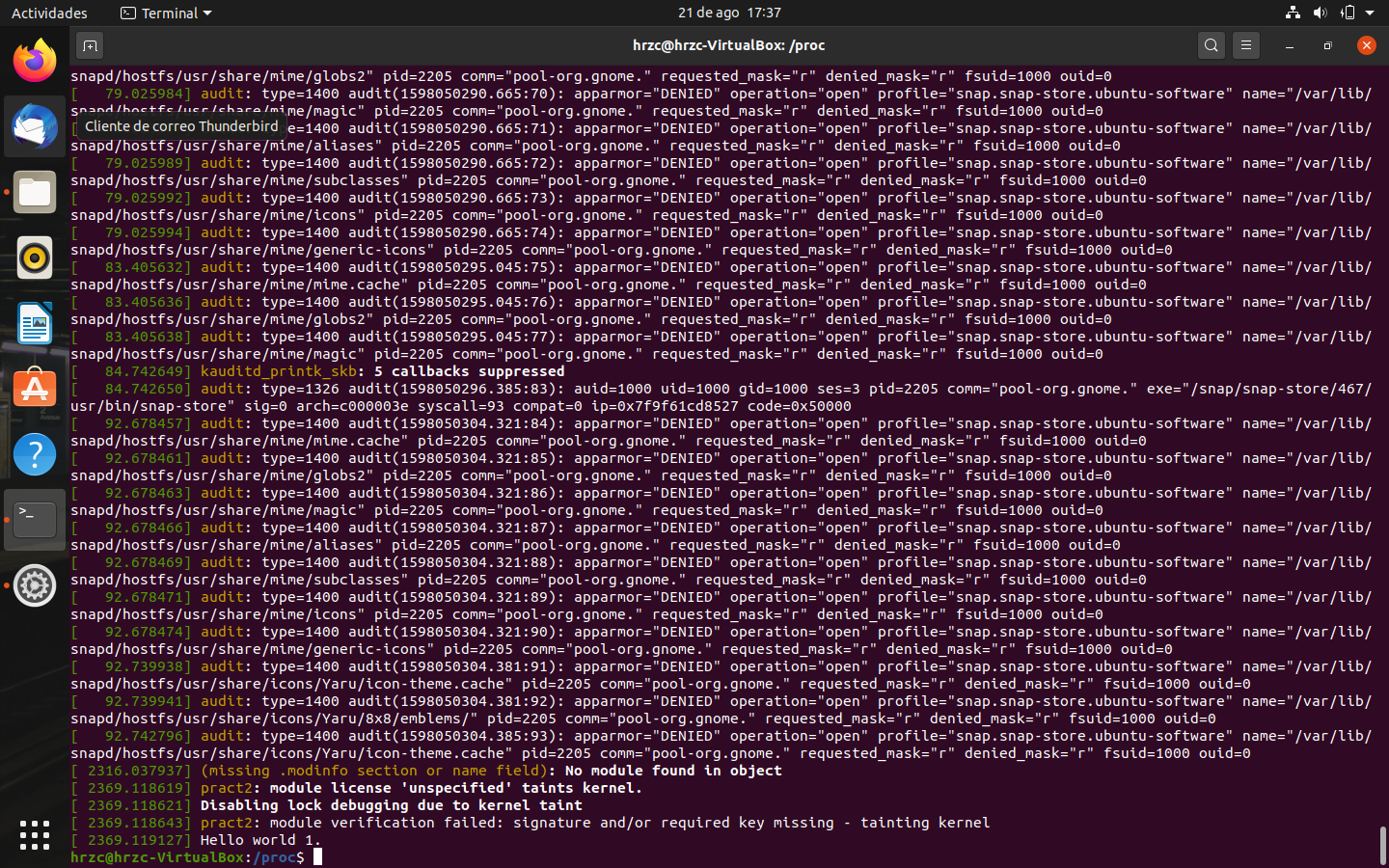
1. Instalando paquete Make (este permite compilar de forma sencilla nuestros programas).



1. ****Creación del módulo con la ayuda del comando Make, además agregando el modulo junto a los otros módulos dentro del kernel

**I.**  Despliegue de módulos dentro del sistema, en donde se encontrará pract2

****

**J.** Mostrando mensajes que se obtienen a través del Kernel, solicitado mediante el comando dmesg

**Preguntas de la práctica**

**1. ¿Las instrucciones para impresión en pantalla son las mismas para la compilación en C de aplicación y la compilación para hacer un módulo? Explique por qué lo son o no.**

Para la impresión en pantalla de un archivo .c se utiliza “printf” para imprimir un texto, y para la impresión de un modulo que se agregara al kernel es “printk” que se utiliza para imprimir mensajes dentro del kernel.

**2. ¿Qué son las extensiones .h y para qué sirven?**

Son denominados archivos de cabecera en el lenguaje C, son archivos compartidos dentro de cualquier código escrito en C, en este se pueden almacenar funciones, variables, constantes, estas instrucciones pueden ser referenciadas en un archivo .C para obtener un valor que devuelve una función dentro del archivo .h

**3. ¿Por qué se deben de usar 2 funciones al hacer código para hacer un módulo del Kernel? ¿Cuál es su función?**

La primera función es “int init\_module” la cual es un constructor devuelve un valor int, y esto lo hace para confirmar que el archivo funciono correctamente o no por tener errores, la siguiente funcion se utiliza al eliminar el módulo del kernel “cleanup\_module”.

**4. ¿Qué diferencia existe entre la compilación con GCC y la que necesita un Makefile?**

GCC, es denominado un compilador que se encuentra dentro del proyecto GNU que es utilizado para lenguajes de bajo nivel (C, C++, Objetive C, entre otros), la función de este es la generación de un programa que sea ejecutable de formato binario dentro de la maquina donde se correrá el programa.

Existen diferentes sintaxis para el uso de gcc, se recalca que todas las opciones van precedidas de un guion.

1. **Gcc PracEj.c** = este compila el archivo “PracEj.c”, este genera un archivo que ya es ejecutable dentro del sistema “a.out”.
2. **Gcc -o PracEj PracEj.c** = este compila el archivo “PracEj.c” y genera un archivo que ya es ejecutable “PracEj”.
3. **Gcc -c PracEj.c** = a diferencia de las demás opciones esta no genera un archivo ejecutable, únicamente el código objeto dentro del archivo “PracEj.o”

Makefile, un archivo Makefile tiene una estructura donde este tiene cuatro diferentes tipos de declaraciones básica:

1. **Variables**.
2. **Comentarios.**
3. **Reglas Explicitas.**
4. **Reglas implícitas.**

Un archivo Makefile tiene como función el almacenamiento de comandos de compilación y dentro de esto con todos los parámetros necesarios para encontrar los archivos .h, llamados ficheros de cabecera.

Esto reduce el trabajo de escritura de código y se hace más sencillo su utilización.

**5. ¿Al crear un módulo de Kernel, se están creando System Calls? ¿Existe algo que lo diferencie?**

1. No ya que es una función que se utiliza para agregar un módulo dentro del kernel, y para una system call es únicamente una interacción con el API y no directamente con el kernel.
2. Las system calls son llamados directos al kernel que sirven para la ejecución de funciones especificas para controlar diferentes dispositivos , o bien la ejecución de una instrucción privilegiada en el sistema.