Universidad de Guadalajara

Ingeniería Informática

Hands-on 5: Process & ProcessImage

SEMINARIO DE SOLUCION DE PROBLEMAS DE USO, ADAPTACION, EXPLOTACION DE SISTEMAS OPERATIVOS - 103857 - D03

Alumno: Ramirez Gomez Kevin Ramses

Código: 218218194

Profesor: Jose Antonio Aviña Mendez

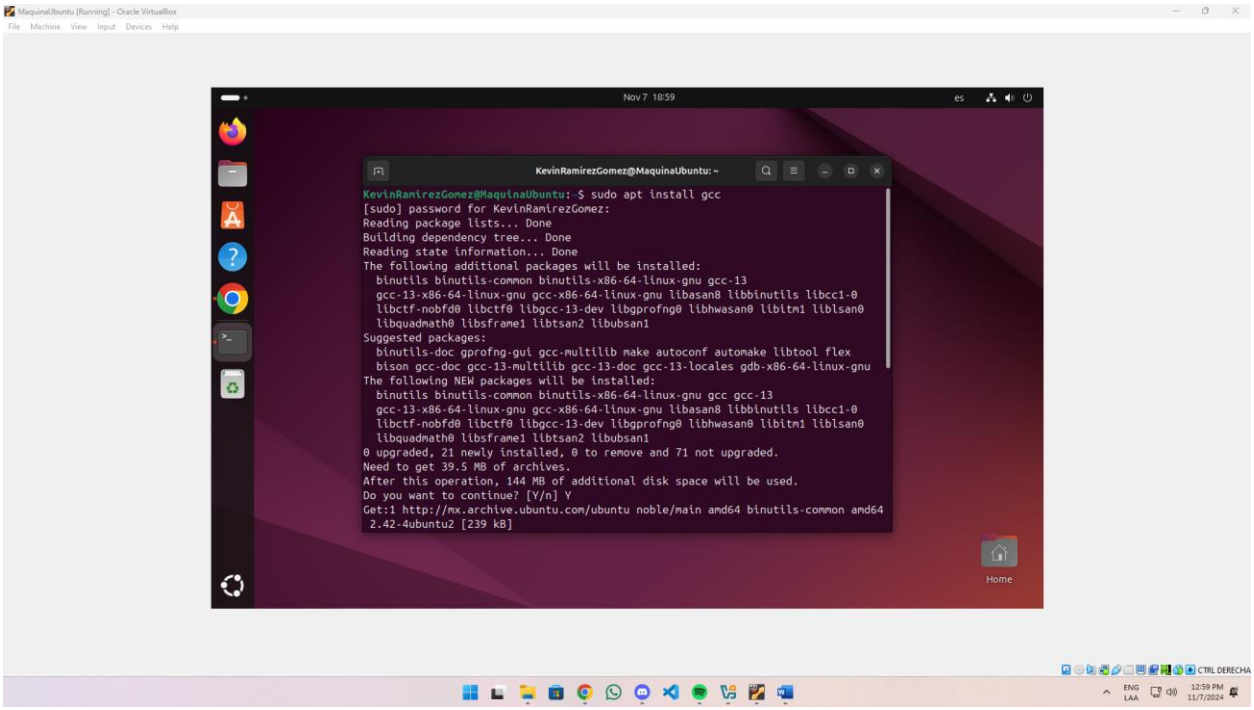
Fecha: 07/11/2024

Índice

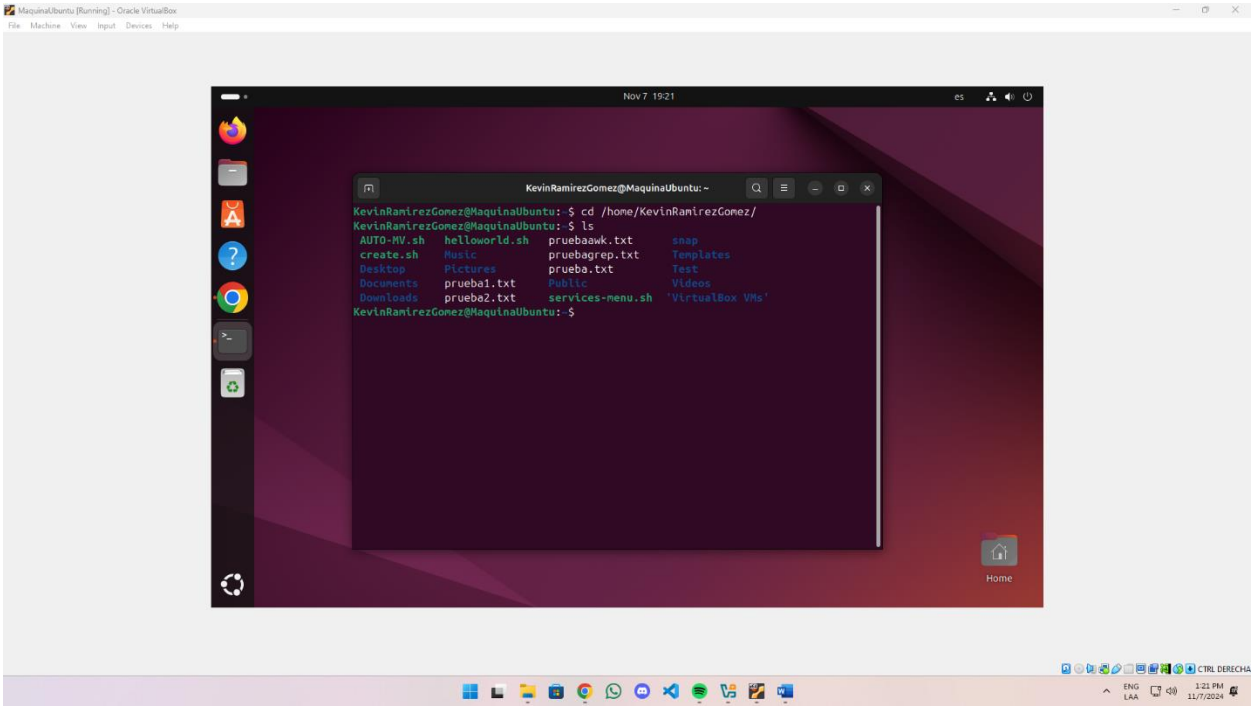
DESARROLLO

INFORMACIÓN DE UN PROCESO

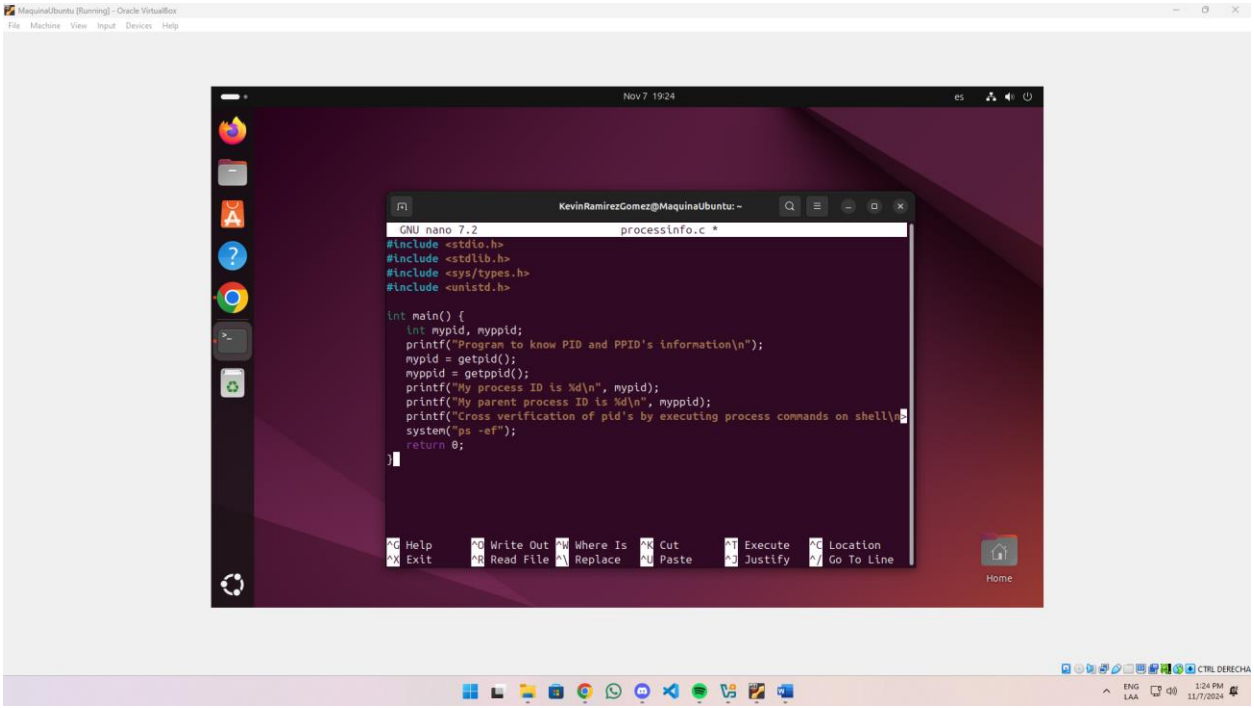
Primero tenemos que ejecutar el siguiente comando para tener el compilador de C presente en el sistema: sudo apt install gcc.



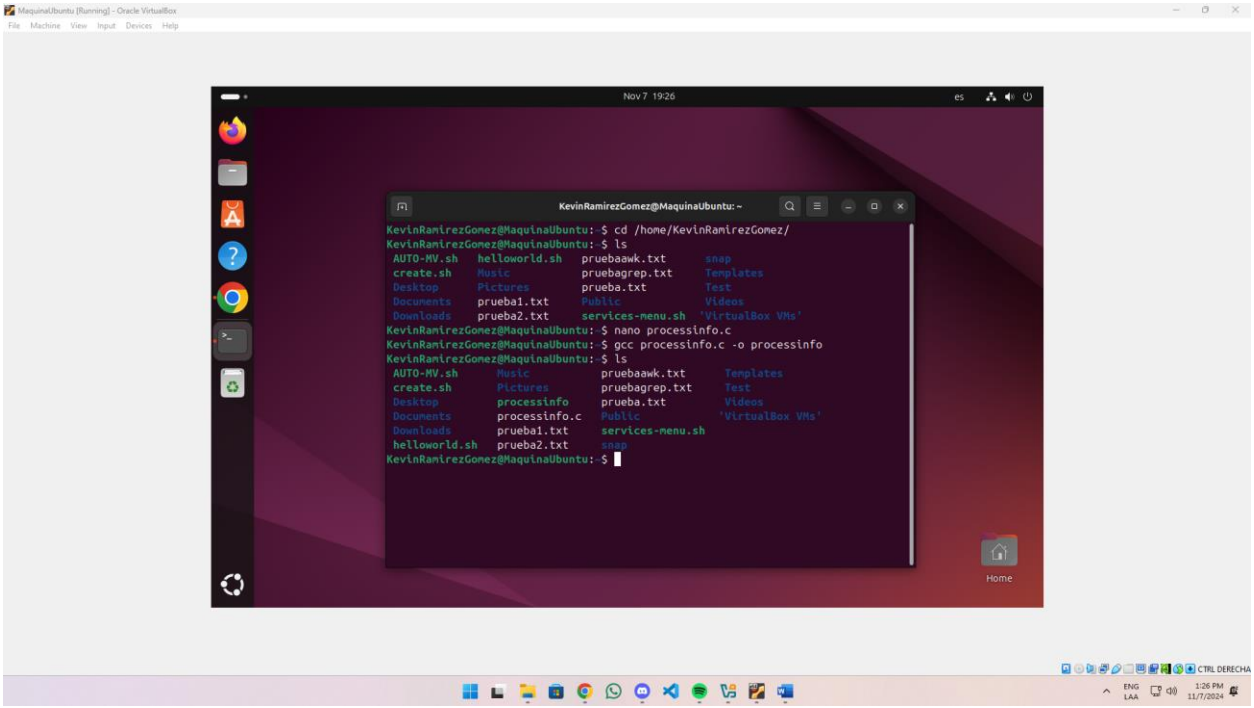
Ya que termine de instalar el compilador luego procederemos a moverlos a la carpeta donde hemos estado trabajando los hand-on anteriores.



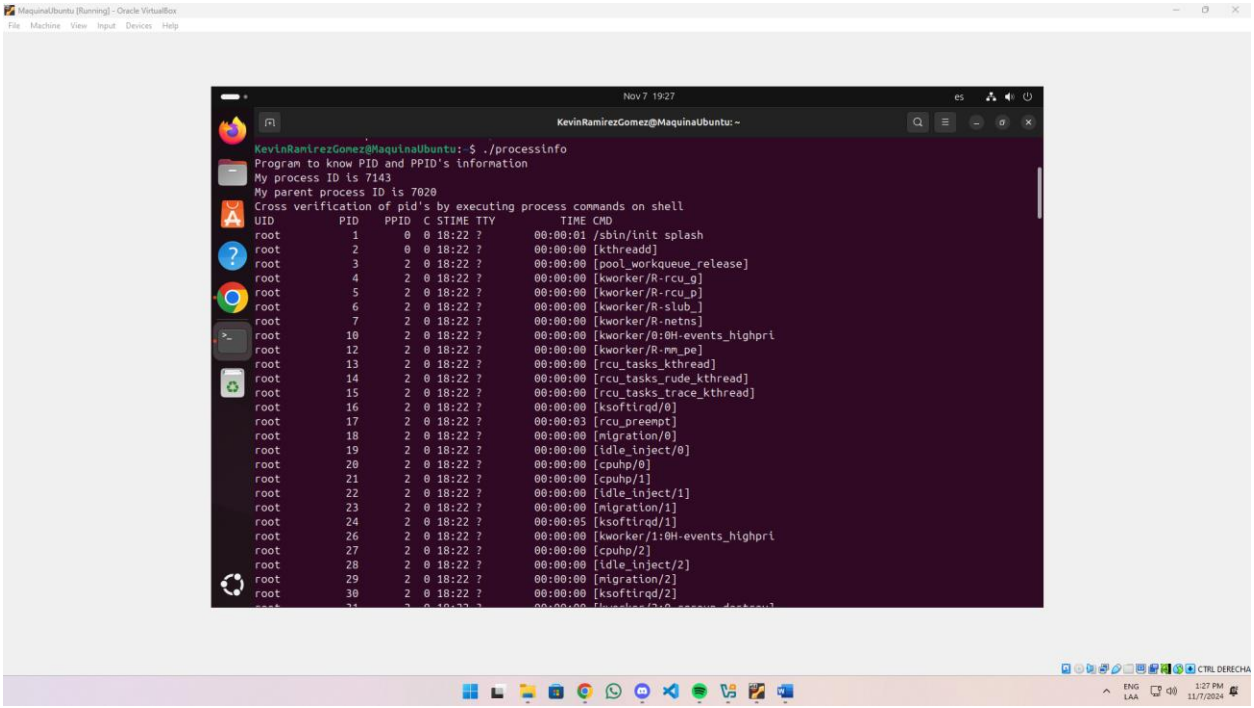
Luego procederemos a crear el archivo con el programa en c para poder convertirlo en un ejecutable.



Ahora convertiremos el archivo en un archivo ejecutable para el compilador, con el siguiente comando: gcc processinfo.c -o processinfo



Ahora ejecutamos el programa con el comando ./processinfo.

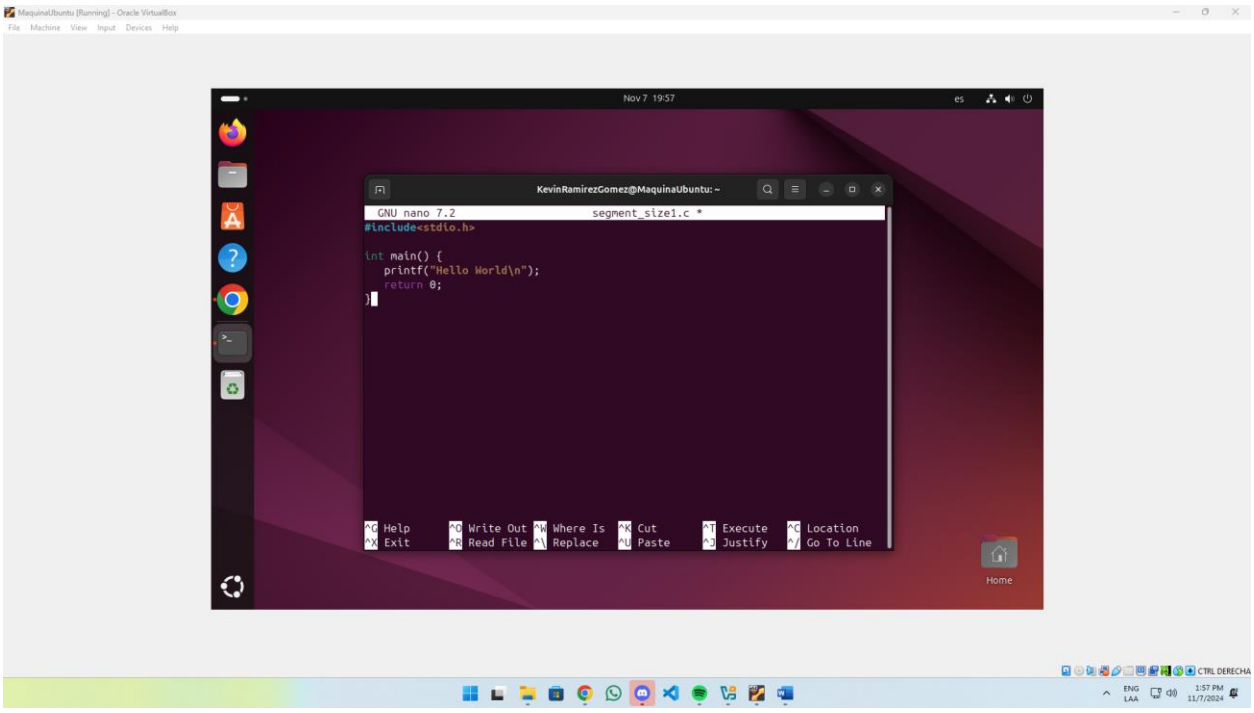


¿Cómo a través de un programa obtener información de un Proceso?

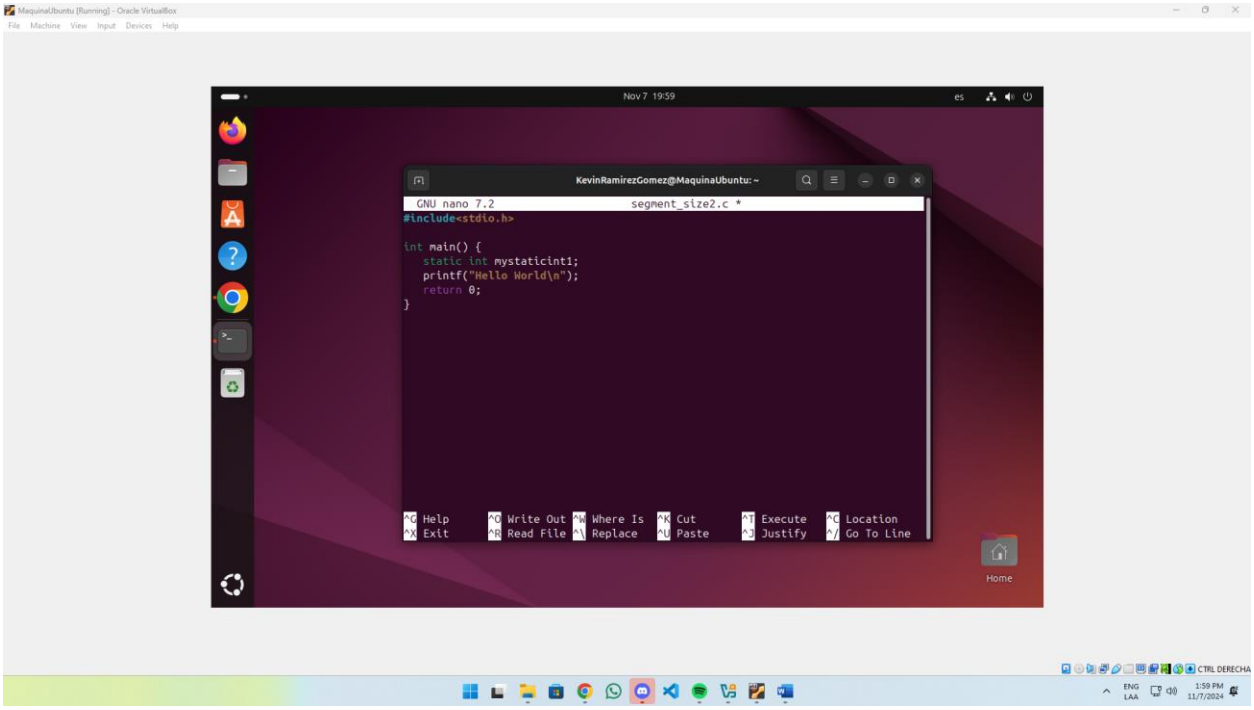
Los procesos en un entorno Linux permite entender de manera profunda cómo los programas interactúan con el sistema operativo. Utilizar funciones como getpid() y getppid() ayuda a identificar el proceso actual y su proceso padre, mientras que comandos como system("ps -ef") permiten verificar y gestionar los procesos en ejecución. Esta parte de la actividad me sirvió para comprender y trabajar con la información de procesos desde un programa en C ofreciendo una perspectiva profunda sobre cómo los programas interactúan con el sistema operativo.

IMAGEN DE UN PROCESO

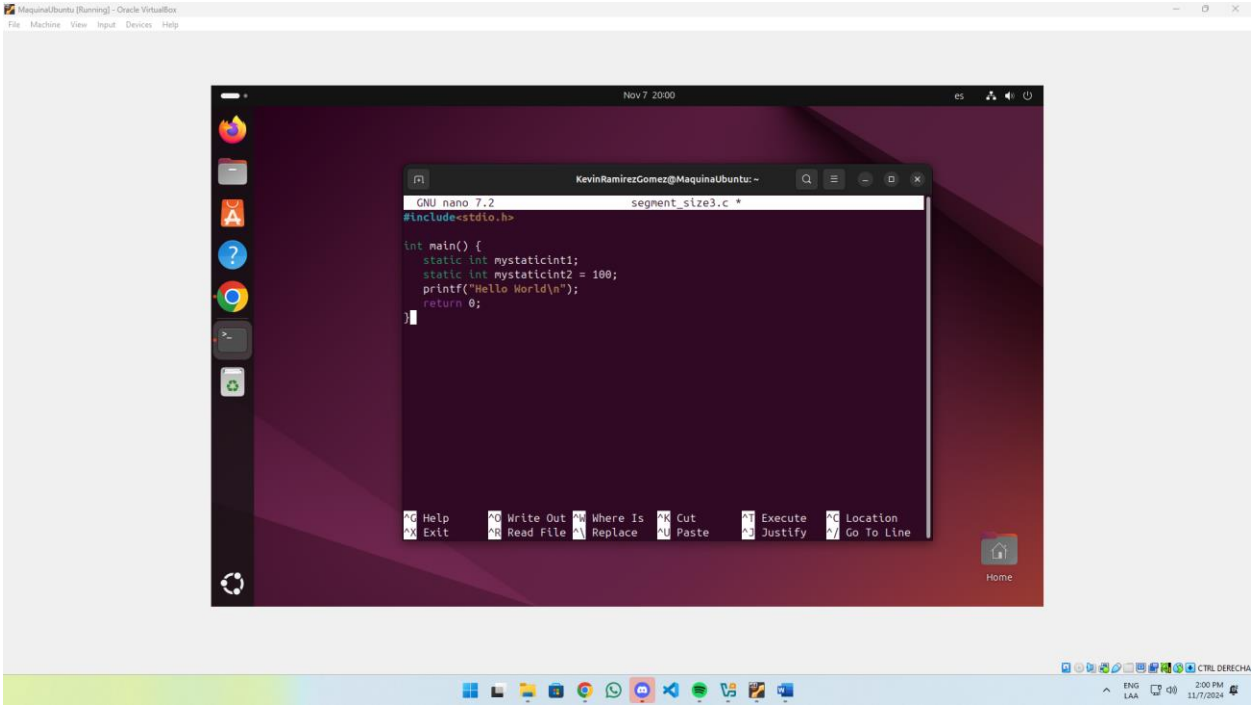
Ahora para esta parte de la actividad crearemos cinco programas en c, donde estos nos muestran los diferentes segmentos que suele contener una imagen de proceso.



Este programa o tiene variables estáticas ni globales, por lo que el segmento de datos y BSS no se ven afectados.

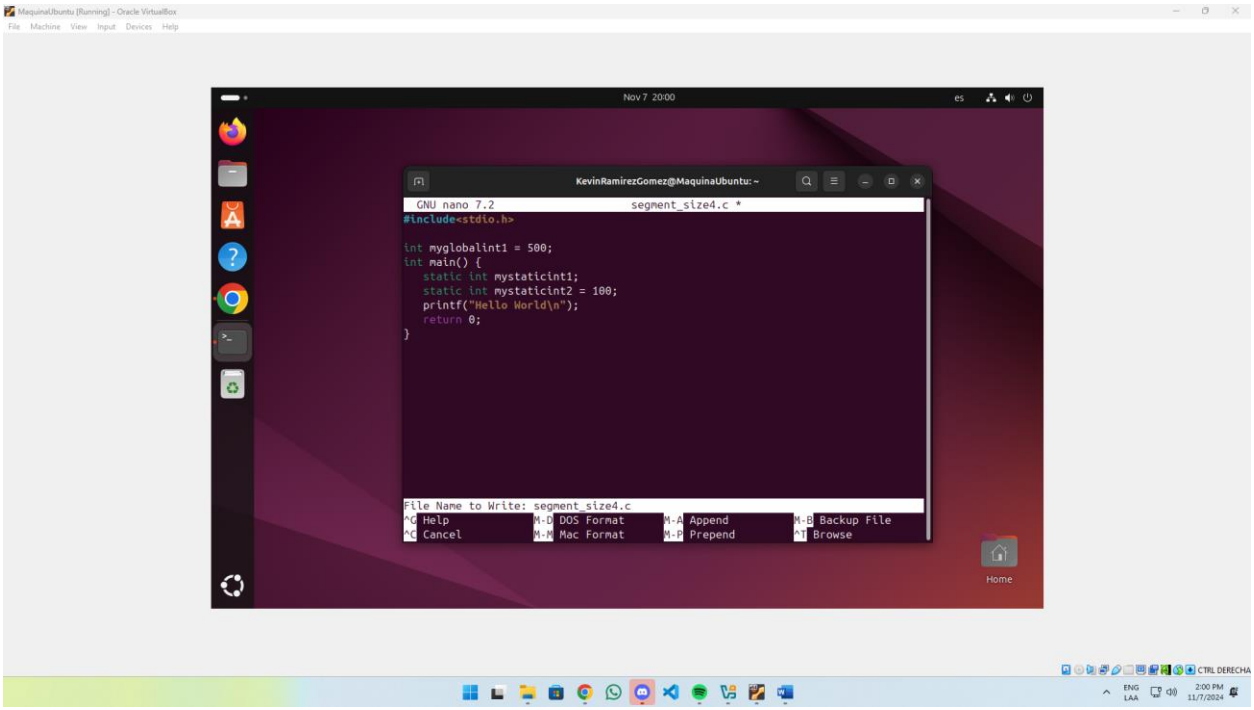


Este codigo introduce una variable estática no inicializada, static int mystaticint1. Esta variable se aloja en el segmento BSS, aumentando su tamaño en 4 bytes.

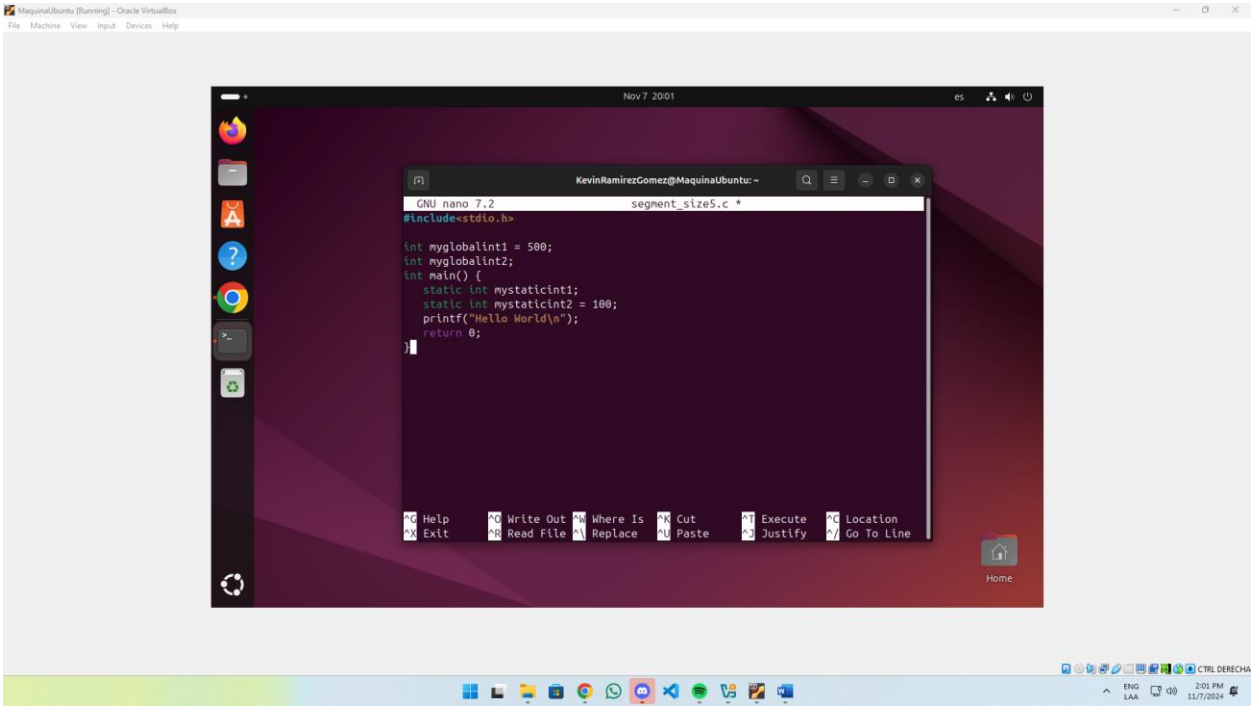


Este codigo añade una variable estática inicializada static int mystaticint2 = 100. Esta variable se coloca

en el segmento de datos, que aumenta en 4 bytes, mientras que mystaticint1 sigue en el BSS.



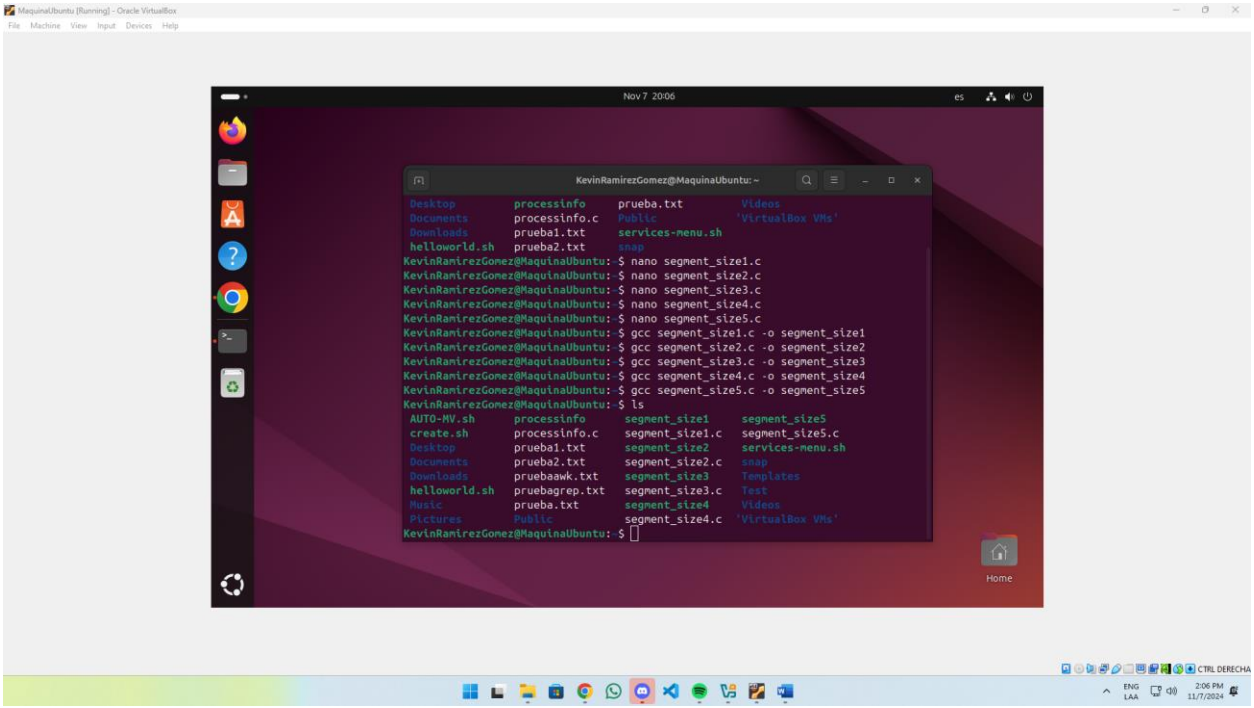
Este codigo declara una variable global inicializada int myglobalint1 = 500. Esta variable incrementa el tamaño del segmento de datos en 4 bytes, además de las variables estáticas.



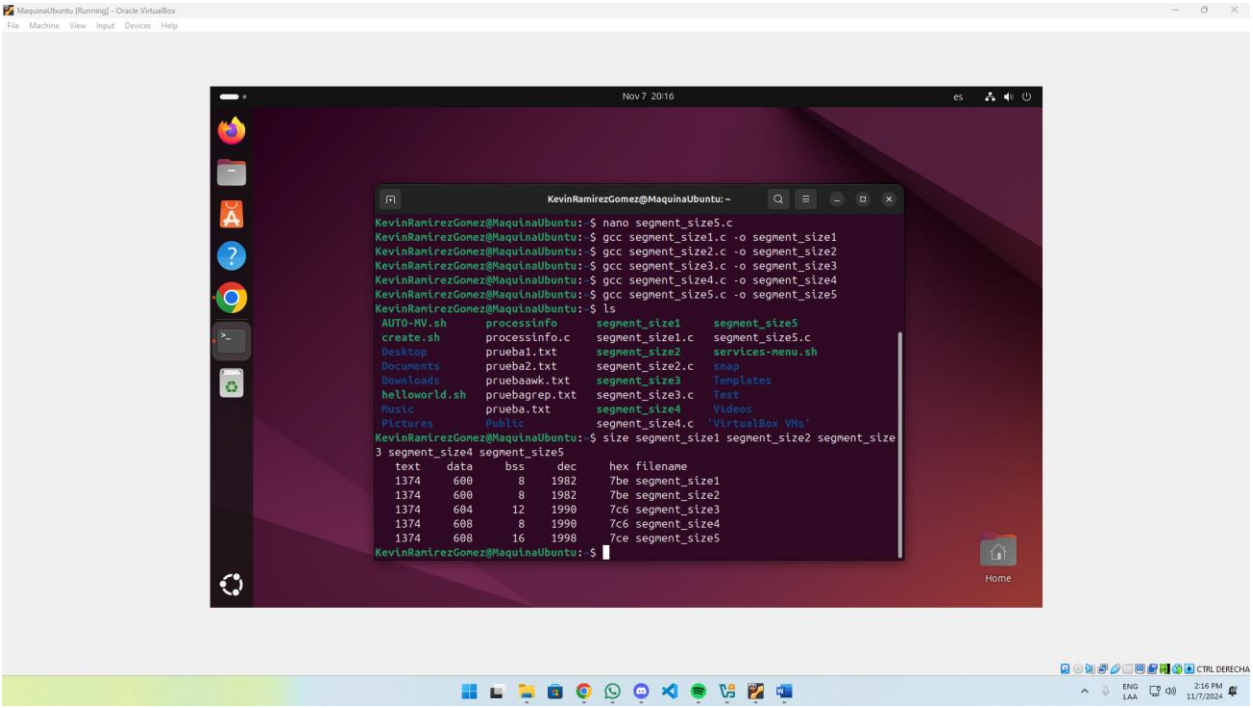
Este código incluye una variable global no inicializada int myglobalint2. Esta variable se aloja en el

segmento BSS, aumentando su tamaño en 4 bytes. Además, las variables myglobalint1 y mystaticint2 ocupan espacio en el segmento de datos.

Ahora procederemos a convertirlos en archivos ejecutables.



Ahora podemos observar como el segmento BSS y DATA cambian según las variables estaticas o globales, inicializadas/no inicializadas en el programa.



¿Cómo a través de un programa obtener información sobre la Imagen de un Proceso?

Al analizar cómo las variables estáticas y globales afectan la segmentación de memoria usando herramientas como size. Al agregar diferentes tipos de variables en los codigos, los segmentos BSS y DATA cambiarán. El comando size nos permite ver cómo se distribuyen los tamaños de estos segmentos y cómo varían según el programa compilado, dándonos los siguientes resultados: text: instrucciones ejecutables, data: variables inicializadas, bss: variables no inicializadas, dec: Tamaño total en decimal, hex: Tamaño total en hexadecimal y filename: Nombre del archivo ejecutable.