UNIVERSIDAD ANDINA DEL CUSCO

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS

Evaluación de viabilidad del proyecto de Kernel Unificado (Longene) como alternativa para la ejecución de programas de Windows en Linux

---

LÍNEA DE INVESTIGACIÓN Tecnologías de la Comunicación

Tesis presentado por:
Bach. Mehrez Garcia, Amir Fernando Mamdouh

Para optar al Título Profesional de Ingeniero de Sistemas

Asesor:
Ing. Carrasco Poblete, Edwin

CUSCO – PERÚ
2022

DEDICATORIA

Dedico este trabajo a mi madre, la persona más importante en mi vida Ing. Magali Violeta Garcia Mujica que siempre está cuando más la necesito, que siempre me ha apoyado en todas las cosas que hice y que dedico toda su vida a cuidarme. A mis abuelos que en paz descansen Rosa Estela Mujica Miranda y C. Fernando Garcia Chaparro gracias a ellos aprendí los valores que debe tener una persona y suplieron la labor del padre que nunca tuve presente, estoy seguro que les hubiera gustado ver el hombre en el que me convertí. A mi padrino y una de las personas que más admiro Crnl. en situación de retiro Mario Gallegos Polo y la Borda, quien siempre fue un guía para mi desde niño, gracias a él aprendí a autoeducarme, la responsabilidad y el siempre querer crecer como persona en todos los ámbitos. A mi tía Q.E.P.D. Nanci Amalia Garcia Mujica que a pesar de los últimos años tener diferencias siempre confió en mí y me apoyo a su forma mientras estuve estudiando. Y a mis amigos que son personas con las cuales se puede contar, además de aprender de ellos directa o indirectamente por las situaciones que nos suceden.

AGRADECIMIENTOS

Agradezco a mi madre por alentarme, apoyarme económicamente, apoyarme con las situaciones cotidianas e incluso respetar las decisiones que tome para poder realizar este proyecto de investigación. Agradezco a mi hermano que me apoyo las veces que necesitaba. Agradezco a mis amigos y las personas que consulte para tomar la decisión del enfoque que debía darle al proyecto, incluso sus comentarios de que sí ya me graduaba. Agradezco a mi asesor Mgt. Edwin Carrasco Poblete haber aguantado que haya acudido a él luego de largos periodos de espera con muy poco avance, ya que incluso había llegado a un punto muerto del mismo proyecto. Y agradezco a todas y cada una de las personas que ponen su granito de arena para comentar en los foros de discusión y crean contenido sobre ciertas cuestiones específicas, sin haber revisado en tantos hilos no me hubiera sido posible armar el rompecabezas que hice, aunque no me sea posible referenciar a todos y cada uno, gracias a ellos hasta los errores o sus respuestas incorrectas me ayudaron a aprender, aunque nunca haya preguntado, sino solamente fui un espectador.

ÍNDICE GENERAL

ÍNDICE DE TABLA

ÍNDICE DE FIGURA

ÍNDICE DE DIAGRAMA

INTRODUCCIÓN

El presente proyecto de investigación plantea la tesis sobre la evaluación y la viabilidad del proyecto Longene, el cual se compila en diferentes distribuciones Linux permitiendo la ejecución de programas Windows en Linux, específicamente para este proyecto todo se realizará en la distribución Ubuntu 16, obteniendo a raíz de esta investigación la información necesaria para tomar en cuenta el proyecto Longene como una alternativa viable. La importancia de este proyecto se especifica detalladamente en el Capitulo I, tomando en cuenta todos los beneficios y a nivel investigativo el uso además de la comprensión de herramientas que modifican el Núcleo logrando así ejecuciones a bajo nivel de la forma más natural posible.

ABSTRACT

The present research project raises the thesis on the evaluation and viability of the Longene project, which is compiled in different Linux distributions allowing the execution of Windows programs in Linux, specifically for this project everything will be done in the Ubuntu 16 distribution, obtaining from this research the necessary information to take into account the Longene project as a viable alternative. The importance of this project is specified in detail in Chapter I, taking into account all the benefits and at a research level the use and understanding of tools that modify the Kernel, thus achieving low level executions in the most natural way possible.

CAPITULO I: PROBLEMA DE INVESTIGACIÓN

1.1. AMBITO DE INFLUENCIA

1.1.1. AMBITO DE INFLUENCIA TEORICA

Los sistemas operativos son el programa o software más básico de un computador. Son plataformas que facilitan la interacción entre los usuarios, los programas del computador y los dispositivos hardware; sus funciones son administrar los recursos, coordinar con el hardware y organizar todos los archivos existentes. Cada sistema operativo posee sus características que los hacen distintos de otros, ya sea por su forma de ejecutar programas, la interpretación de algoritmos o la asignación de directorios. “El objetivo fundamental de los sistemas informáticos es ejecutar programas y facilitar la resolución de los problemas de los usuarios. El hardware de la computadora se construye hacia este objetivo. Dado que el hardware por sí solo no es particularmente fácil de usar, se desarrollan programas de aplicación. Estos programas requieren ciertas operaciones comunes, como las que controlan los dispositivos I/O. Las funciones comunes de control y asignación de recursos se unen en una sola pieza de software: El Sistema Operativo”. ABRAHAM SILBERSCHATZ, GREG GAGNE, PETER B (2018).

Con el avance de la tecnología, los programas son creados para ciertos entornos y sistemas operativos, así realizar trabajos dedicados y explotar al máximo los recursos en ciertas tareas específicas. Pero existen los usuarios que tienen como principal objetivo el realizar varias tareas con distintos objetivos de distintas áreas al mismo tiempo en lugar de enfocarse en una sola área.

La presente investigación se fundamentara en el análisis y pruebas de un sistema operativo que por interés del proyecto ha sido adaptado con el objetivo de evaluar la viabilidad del proyecto Longene (Tecnología de Kernel Unificado); esta tecnología permite la ejecución de los programas Windows en una distribución Linux,

contiene librerías de múltiples proyectos así como modificaciones en ciertos archivos, la base del proyecto de kerrnel unificado llamado “Longene”, el cual fue financiado por la Universidad de Zhejiang. Longene fue un proyecto de sistema operativo de código abierto y gratuito, cuyo objetivo era el de expandir el kernel de Linux a un kernel compatible que admita tanto las aplicaciones de Linux como las de Windows, así como los controladores de dispositivos Linux como los controladores de dispositivos Windows, de manera tal que se puedan ejecutar aplicaciones de Windows de manera eficiente en el sistema operativo Linux.

1.1.2. AREA DE DOMINIO

Mediante la Resolución N° 373-2018-CFIA-UAC, la Escuela de Ingeniería de sistema de la Facultad de Ingeniería y Arquitectura de la Universidad Andina del Cusco, resuelve aprobar las líneas de investigación de la Escuela Profesional de Ingeniería de Sistemas. El siguiente proyecto que desarrollare pertenece al área de dominio de TECNOLOGÍA DE COMUNICACIÓN; al ser este un proyecto sobre una alternativa de ejecución de programas de Windows en GNU/Linux a partir de modificaciones en el sistema operativo, ente que permite la comunicación entre el hardware y las aplicaciones, se tiene una orientación hacia el área mencionada debido a que esta permite la comunicación a partir de la interpretación hombre-máquina.

1.1.3. LÍNEA DE INVESTIGACIÓN

Como ya se mencionó previamente, el área de dominio será tecnologías de la comunicación, la línea de investigación que voy a seguir será dentro de comunicaciones unificadas y concretamente **infraestructura**; los sistemas operativos son considerados parte de la infraestructura de las tecnologías de la comunicación porque sin estos el hardware no puede realizar sus tareas, ya que este gestiona los recursos y provee los servicios a las aplicaciones.

1.2. PLANTEAMIENTO DEL PROBLEMA

1.2.1. DESCRIPCION DE LA SITUACION ACTUAL DEL LUGAR DE INTERVENCIÓN

El presente proyecto se desarrolla en la Provincia del Cusco y ciudad del Cusco. Esta investigación no está desarrollada en una institución o empresa específica, sino más bien consiste en la evaluación del rendimiento del proyecto Longene, por este motivo el investigador desarrollará este proyecto con el entorno considerado necesario para la instalación y pruebas del proyecto.

El entorno utilizado para hardware es una computadora Alienware 15 R2, ya que esta es una computadora que pertenece al investigador, por la estandarización de estos equipos de la empresa proveedora de esta computadora se presenta en software un Sistema Operativo Windows 10 Pro, para poder crear los escenarios relevantes necesarios para la instalación y pruebas del proyecto Longene se requiere de un entorno Linux para su instalación y pruebas, fue necesario hacer uso de un software que permita crear máquinas virtuales, tomando así el software VMWare debido a que el investigador tiene experiencia con el uso de esta tecnología.

Como se detallará en futuros ítems de la investigación, se hace uso de VMWare 14.1.1 build-7528167, ya que este permite instalar al día de hoy las desde las versiones de Ubuntu 12 de 32 y 64 bits, hasta versiones más nuevas. Así pues, con este software se asigna una memoria de 2 GB de RAM y 25GB de almacenamiento para la maquina que se usara, siendo así que los clones que resulten de las pruebas contaran con las mismas capacidades técnicas.

1.2.2. DESCRIPCIÓN DEL PROBLEMA

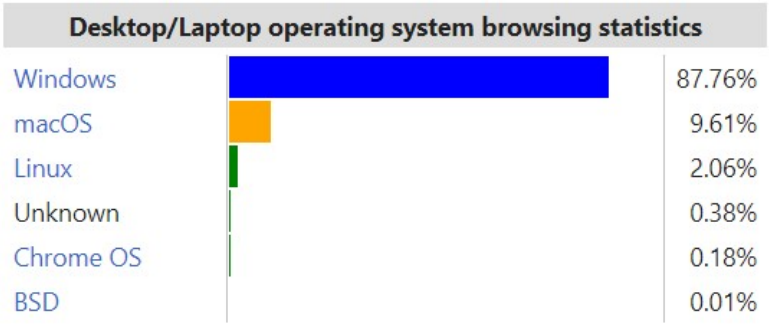
Es de conocimiento general que uno de los sistemas operativos más usados a nivel mundial es el SO (Sistema Operativo) Windows, debido a que fue uno de los primeros en implementar una interfaz amigable logrando así la preferencia de del público en general, esto ocurrió por los años 90's, con lo cual se impuso como líder del mercado. En el año 1995 los sistemas operativos más populares que competían entre ellos eran Macintosh 84 (Apple) y Windows 95 (Microsoft); aunque los dos presentaban una interfaz muy similar, en las siguientes versiones de estos sistemas operativos Apple hizo cambios progresivamente mayores en la interfaz, mientras que Windows se mantuvo con el modelo que creo, logrando la preferencia de los usuarios al mantener su interfaz, porque estos tenían desdén por aprender desde cero a usar un nuevo sistema operativo ofrecido por otras empresas.

Windows sigue siendo el sistema operativo de escritorio dominante, pero el dominio varía según la región y ha perdido gradualmente su participación de mercado en otros sistemas operativos de escritorio (no solo en dispositivos móviles) con el deslizamiento muy notable en los EE. UU., Donde el uso de macOS se ha más que triplicado desde 2009 hasta 2017, con Windows hasta 72.76% y Chrome OS en 3.33%, más Linux tradicional en 1.46%. Hay poca información publicada abiertamente sobre los envíos de dispositivos de computadoras de escritorio y portátiles. Gartner publica estimaciones, pero la forma en que se calculan las estimaciones no se publica abiertamente. Otra fuente de cuota de mercado de varios sistemas operativos es StatCounter, basando su estimación en el uso de la web (aunque esto puede no ser muy preciso). Además, las ventas pueden exagerar el uso. La mayoría de las computadoras se venden con un sistema operativo preinstalado, y algunos usuarios reemplazan ese sistema operativo por uno diferente debido a

preferencias personales, o instalan otro sistema operativo junto con él y usan ambos. Por el contrario, las ventas subestiman el uso al no contar las copias no autorizadas. Por ejemplo, en 2009, aproximadamente el 80% del software vendido en China consistía en copias ilegítimas. En 2007, las estadísticas de una actualización automatizada de IE7 para computadoras Windows registradas diferían con el uso compartido del navegador web observado, lo que llevó a un escritor a estimar que entre el 25 y el 35% de todas las instalaciones de Windows XP no tenían licencia. (Statcounter, 2019)

Figura 1

Porcentaje de Usuarios por SO



Nota. El gráfico representa el porcentaje de Usuarios por Sistema Operativo. (Statcounter, 2019)

La interfaz gráfica que presenta Windows es muy amigable e intuitiva con el usuario, por ende no es extraño que este sistema operativo posea una cantidad enorme de usuarios en el mundo y cada vez se crean más programas orientados exclusivamente a Windows, estos software son muy variados y la lista es inmensa; desde las herramientas más simples de creación de imágenes como Paint hasta programas tan avanzados de simulación estoica o motores de videojuegos, por lo expuesto no sería irreal afirmar que Windows tiene sin duda la mayor cantidad de usuarios a nivel América Latina y el mundo.

Un sistema operativo de entorno Distribuciones GNU/Linux como las distribuciones Debian o Ubuntu no tienen las limitaciones de accesos para configuraciones del sistema operativo debido a que son sistemas operativos de kernel monolítico y de código abierto, mientras que un sistema operativo Windows posee un kernel hibrido y ciertas restricciones de permisos con las cuales se limitan algunas funciones requeridas por los usuarios. Las distribuciones Linux no pueden ejecutar los programas creados exclusivamente para Windows, ni tienen una lista tan inmensa de programas que puedan competir con los del entorno Windows, por lo que se tienen ventajas y desventajas al usar uno u otro sistema operativo.

Debido a la gran cantidad de software que posee Windows, muy útiles para ciertos escenarios, se crearon algunos métodos que permitan la ejecución de programas Windows en algunas distribuciones con un sistema operativo de base diferente.
Las 3 alternativas más usadas para permitir que una computadora Linux pueda compilar programas Windows y así permitir ciertas características de cada plataforma las cuales son virtualización, emulación y multiarranque:

- Virtualización es la técnica empleada sobre las características físicas de computadoras, para ocultar referencias de otros sistemas, aplicaciones o usuarios que interactúen con ellos. Esto implica hacer que un recurso físico un sistema operativo aparezca como si fuera varios recursos lógicos a la vez, o que varios recursos físicos, como servidores o dispositivos de almacenamiento, aparezcan como un único recurso lógico. La Virtualización es un proceso complejo que se basa principalmente en montar un sistema operativo por encima del que usamos normalmente. (VELÁZQUEZ, 2009)

- Wine significa en inglés “Wine is not an emulator” - “Wine no es un emulador” y es un software de línea de comandos de código abierto que es capaz de traducir llamadas API de Windows en llamadas POSIX sobre la marcha para integrar aplicaciones de Windows en tu escritorio Linux/UNIX. Para los usuarios habituales de Linux, lo anterior significa que Wine les permitirá ejecutar aplicaciones que están diseñadas para instalarse solo en sistemas operativos Microsoft Windows. (maslinux, 2018)

- Multiarranque es la capacidad de una computadora para poder tener más de un sistema operativo funcionando en un mismo equipo y arrancar con cualquiera de ellos. Al arrancar la computadora con doble arranque, el sistema preguntará al usuario con cual sistema operativo desea iniciar, gracias a esto se puede tener más de un SO a nuestra disposición, lo cual nos permitirá cambiar entre plataformas con un previo reseteo, después de hacer la selección este procede a cargar el sistema operativo en memoria. Esto permite poder tener más de un sistema operativo funcionando en un mismo disco rígido o equipo y arrancar con cualquiera de ellos. (custer_flux, 2014)

Lamentablemente cada una de las alternativas expuestas posee problemas independientes descritos en la justificación. En la realidad actual los usuarios de entornos GNU/Linux quedaron desplazados como público objetivo de muchas empresas que desarrollan software, a las cuales basándose en estadísticas no les resulta rentable crear la versión para entorno GNU/Linux y optan solamente

por el desarrollo exclusivo orientado a Windows. Aunque el público objetivo sigue existiendo, por el momento no se pueden complacer sus necesidades a cabalidad. A raíz de esto se deben optar por otras posibles soluciones que permitan satisfacer a los usuarios GNU/Linux, a partir de herramientas o configuraciones en el sistema de forma tal que puedan ellos proseguir con sus labores, con el uso de software de Windows en la distribución GNU/Linux de su preferencia.

1.2.3 FORMULACIÓN DEL PROBLEMA

1.2.3.1 FORMULACIÓN INTERROGATIVA DEL PROBLEMA GENERAL

¿Cuán viable es el proyecto de kernel unificado (Longene) como alternativa para la ejecución de programas de Windows en Linux?

1.2.3.2 FORMULACIÓN INTERROGATIVA DE LOS PROBLEMAS ESPECÍFICOS

- ¿Cómo funciona el proyecto Longene?
- ¿Cómo se implementa Longene en Ubuntu 16?
- ¿Cuál es el rendimiento de Longene en Ubuntu 16?

1.2.4. OBJETIVOS DE LA INVESTIGACIÓN

1.2.4.1. OBJETIVO GENERAL

Evaluar la viabilidad del proyecto Longene como alternativa de ejecución de programas Windows en Linux

1.2.4.2. OBJETIVOS ESPECÍFICOS

- Describir el proyecto Longene a partir de la codificación y los archivos importantes que permiten la ejecución del mismo.

- Realizar la implementación de Longene en Ubuntu 16.

- Evaluar el proyecto Longene mediante el rendimiento en pruebas diversas.

1.2.5. JUSTIFICACIÓN

El proyecto Longene presenta muchos beneficios a largo plazo que serán mencionados a lo largo de este título con un previo análisis de lo ya mencionado. Este proyecto fue instaurado con el objetivo de permitir la ejecución de programas Windows en Linux a nivel de kernel. El núcleo o kernel es un software que constituye una parte fundamental del sistema operativo, y se define como la parte que se ejecuta en modo privilegiado. Es el principal responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora o en forma básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema. Las alternativas de ejecución de programas Windows en un sistema anfitrión Linux previamente mencionados poseen algunas debilidades frente al proyecto Longene las cuales serán mencionadas en los siguientes párrafos.

La Virtualización es un proceso complejo que se basa principalmente en montar un sistema operativo por encima del que usamos normalmente. La virtualización previamente mencionada también presenta cierta cantidad de inconvenientes. A diferencia de un servidor físico, una máquina virtual es un conjunto de ficheros que residen físicamente en un almacenamiento compartido. Este tipo de

encapsulación de la máquina virtual ofrece un nuevo tipo de “robo de datos”. Como la máquina virtual es solo un conjunto de ficheros, un servidor entero puede ser copiado a un dispositivo USB o copiado durante un proceso de backup no autorizado a un lugar no protegido por su personal de seguridad o administrador de su entorno virtual, logrando así replicar la máquina de forma sencilla. Menor rendimiento, dado que una máquina virtual corre en una capa intermedia a la del hardware real, siempre tendrá un rendimiento inferior. Teóricamente no podrás utilizar hardware que no esté soportado por el hipervisor de virtualización. Y se desaprovechan los recursos con la creación de máquinas virtuales que no son necesarias. Dentro de esta categoría se encuentran software como VirtualBox, VMWare, Hyper-V, etc. (CONSULTING, 2015)

El Dual-Boot o arranque dual, básicamente se trata de que al iniciar el ordenador la BIOS se encarga de cargar el 'Gestor de Arranque' desde el disco duro y este a su vez llama al sistema alojado para que inicie. La parte 'Dual' viene cuando ese gestor de arranque encuentra más de un sistema operativo y nos ofrece una pantalla previa al arranque del sistema en la que podemos decidir sobre qué entorno queremos trabajar en dicha ocasión. Esto implica hacer que un recurso físico un sistema operativo aparezca como si fuera varios recursos lógicos a la vez, o que varios recursos físicos, como servidores o dispositivos de almacenamiento, aparezcan como un único recurso lógico. El arranque dual puede dar problemas si el núcleo de Linux tiene problemas en las actualizaciones o a veces si sale mal este quedará inutilizable por lo que se requerirá una nueva instalación. Un disco duro se divide entre los dos sistemas operativos existe un problema de que Windows no puede leer la partición de Linux o cualquier otro tipo de particionamiento. Con los BIOS actuales, se tiene UEFI, lo que hace que el arranque dual sea difícil y difícil de instalar grub y si algo sale mal, la partición UEFI se vuelve, entonces será imposible arrancar. Hay algunos problemas

reportados con la actualización de Windows 10 que causaron muchos problemas con el arranque dual. Instalar el arranque dual es más simple que desinstalar. Cuando solo desea recuperar Windows, se tarda mucho tiempo en utilizar las recuperaciones para recuperar un sistema operativo. (A. E. , 2014)

Wine no debe confundirse con una máquina virtual o un emulador. Proporciona compatibilidad binaria, soporte para gráficos, interacción de sonido, así como soporte para módems, redes, escáneres, tabletas, teclados y otros dispositivos. La API del software permite a los desarrolladores integrar Wine en sus proyectos, y como resultado numerosas interfaces gráficas de usuario, tanto gratuitas como comerciales, aparecieron en Internet a lo largo de los años. Wine hace uso de numerosas interfaces gráficas PlayOnLinux, Crossover, Q4Wine, Bordeaux, Pipelight, Swine, WineXS y winetricks son algunos de los front-ends de Wine GUI (interfaz gráfica de usuario) más populares en Linux. Estos generan un entorno modificado que permite la ejecución de programas de Windows en ambientes UNIX como GNU/Linux. Algunos de los errores que presenta el tener una capa superior al kernel son errores de rendimiento más bajo de lo que se tendrían en Windows nativo, una imagen menos fluida, gráficos menos detallados y que todos los juegos que existen para Windows no son soportados. (maslinux, 2018)

Por lo expuesto, la investigación es requerida por muchos usuarios de distribuciones Linux al ejecutar programas Windows; los usuarios que requieren tener un anfitrión Linux y a su vez ejecutar los programas Windows no satisfacen a cabalidad sus necesidades con las alternativas antes mencionadas por ser ineficientes, presentar problemas de rendimiento, problemas de compatibilidad, carecer de la interpretación al instalar drivers, permisos en modos de privilegios, etc. Si lo dicho no fuera requerido las actuales versiones de las alternativas no seguirían actualizando sus versiones y mejorando las

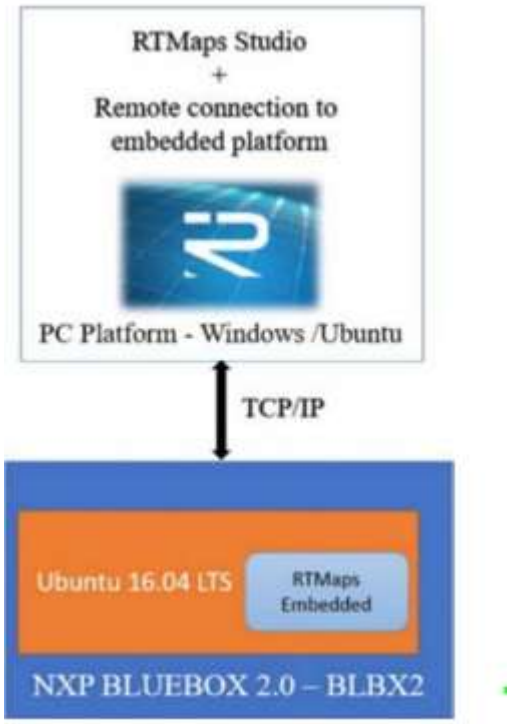
cuestiones mencionadas. Aunque cabe la posibilidad de que estos lleguen a un punto de inflexión en el cual ya no sea posible la mejora, por lo que se deba optar por usar otro camino de diseño.

1.2.6. ALCANCES Y LIMITACIONES

El presente trabajo tiene como objetivo utilizar el proyecto Longene en la distribución de Ubuntu 16.04 para evaluar la viabilidad del proyecto en sí mismo, debido a que la última versión de este fue longene-1.0-rc2, versión la cual fue desarrollada para Ubuntu 12; a partir de esto poder analizar el comportamiento del proyecto para futuras versiones. Aunque este proyecto no posee alguna actualización más desde esta versión se plantea como una alternativa para la ejecución de programas Windows en Linux y el análisis del funcionamiento de ejecución. Incluso en investigaciones recientes el sistema operativo Ubuntu 16.04 sigue siendo relevante como se muestra en el siguiente paper de investigación, con esto se resalta la importancia aún de la tecnología Ubuntu 16.04, con esto se resalta la vigencia tecnológica de la implementación. “Implementación de arquitectura: para implementar un código Pytorch con la ayuda de RTMaps para bluebox2.0, debe constar de tres funciones clave para que funcione en RTMaps. Tres definiciones de función son birth (), core () y death (). La implementación de Pytorch con la ayuda de RTMaps en bluebox2.0 para la arquitectura Shallow SqueezeNext. La conexión entre el estudio RTMaps con conexión remota a la plataforma integrada en una PC y la plataforma en tiempo real con la imagen Ubuntu BSP, bluebox2.0 por NXP se puede acceder a través de TCP/IP, ilustrado en la Figura 2.” (Duggal & El-Sharkawy, 2020)

Figura 2

Conexión entre el estudio RTMaps con conexión remota



Nota. La figura muestra la estructura de un paper, que en el año 2020 continua usando la distribución Ubuntu 16.04 (Duggal & El-Sharkawy, 2020)

Al desarrollar la investigación en un futuro se plantea el retomar el proyecto Longene ya que este es de código abierto conforme a los resultados que se encuentren, así pues, quizás en un futuro con algunas modificaciones a largo plazo esta pueda ser considerada como una alternativa para la ejecución de programas Windows en entorno Linux para algunas instituciones ya sea para realizar investigaciones, experimentar, aprender, etc. Por lo que no se descarta el continuar con el proyecto por parte del investigador o de ponerla en la palestra para que los entusiastas se nutran con los resultados. Un gran ejemplo de la relevancia que se continúa teniendo al evaluar la performance de alternativas que permitan ejecuciones es la siguiente investigación, es “La virtualización de servidores es una innovación tecnológica ampliamente utilizada en las empresas de TI. La virtualización proporciona una plataforma

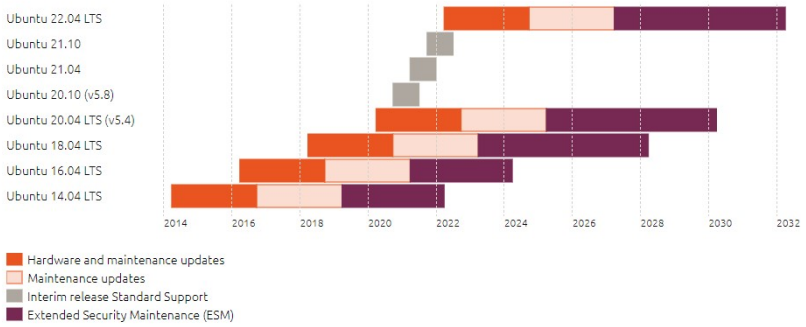
para ejecutar diferentes servicios de sistemas operativos en la nube. Facilita la construcción de múltiples máquinas virtuales en un solo físico básico máquina ya sea en forma de hipervisores o contenedores. Para albergar muchas aplicaciones de microservicios, la tecnología emergente ha introdujo un modelo que consta de diferentes operaciones realizadas por servicios desplegados individuales más pequeños. Así, la demanda para la técnica de virtualización de baja sobrecarga se está desarrollando rápidamente. Hay muchas tecnologías de virtualización ligeras; ventana acoplable es uno entre ellos, que es una plataforma de código abierto. Esta tecnología permite a los desarrolladores y administradores de sistemas construir, crear y ejecutar aplicaciones usando el motor docker. Este documento proporciona la evaluación del rendimiento de los contenedores Docker y las máquinas virtuales.
Utilizando herramientas de referencia estándar como Sysbench, Phoronix y Apache, que incluyen rendimiento de CPU, memoria rendimiento, rendimiento de lectura/escritura de almacenamiento, prueba de carga y medición de velocidad de operación." (Potdara, Narayan, Kengond, & Mulla, 2020)

La propuesta es que con esta alternativa más usuarios de Windows puedan migrar a Linux al no tener el temor de no poder instalar los programas que usan con frecuencia en Windows sin la necesidad de aprender procesos de instalaciones complejas o aprender a manejarse en las distribuciones derivadas de Linux con conocimiento medio o avanzado, de esta forma se podría tener un espectro más amplio de usuarios que no se enfrascan en el entorno Windows. A la vez esto también es beneficioso para usuarios expertos en la materia de Linux ya que podrán hacer uso de herramientas Windows mientras que a la par siguen cumpliendo con las funciones para las que está destinado el entorno Linux y utilizar los programas. Siendo que aún la versión 16.04 de Ubuntu sigue

dentro del espectro de mantenimiento de seguridad extendido hasta el año 2024, representado aún una tecnología útil y utilizada.

Figura 3

Tiempo de Soporte Ubuntu



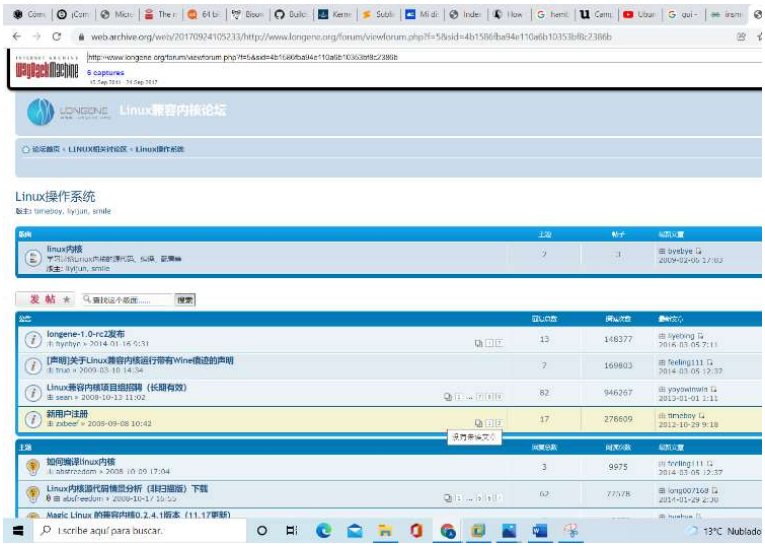
Nota. La figura muestra el tiempo de soporte de cada version de Ubuntu, por distribución. (Griffon, 2021)

La limitación es que el año 2018 se comenzaron a presentar problemas con la web del proyecto Longene; esta web desapareció por 1 mes y luego regreso por una semana para finalmente desaparecer, pero el afán por entender y desarrollar este proyecto fue tan alto que se terminó generando una copia de toda la web, foro, whitebook y la maquina longene-1.0-rc2-ubuntu-12.04.vmdk que tenía el proyecto instalado. La referencia que valida este argumento se demuestra en el punto 2.2.4. Longene. “Longene es un proyecto de sistema operativo autónomo, el propósito es expandir el kernel de Linux en una sola compatibilidad con las aplicaciones de Linux, que admite los controladores de dispositivos Linux y admite las unidades de dispositivo de Windows; Los usuarios pueden ejecutar de manera eficiente las aplicaciones de Windows directamente en el sistema operativo Linux . En 2006, se lanzó su primera versión, pero desde 2014, el desarrollo del proyecto se ha detenido. En mayo de 2018, su sitio web oficial dejo de ser accesible. No hay mucha introducción adicional aquí, a partir del análisis del ángulo de código fuente de Longene de cómo implementar el cargador de PE en el núcleo de Linux.”

(programmerclick, 2020). La web de este proyecto no es accesible desde el 2018, aunque todavía hay información y aún existen registros en la web waybackmachine siendo la última entrada pertinente del foro.

Figura 4

Pantalla de foro del Proyecto Longene

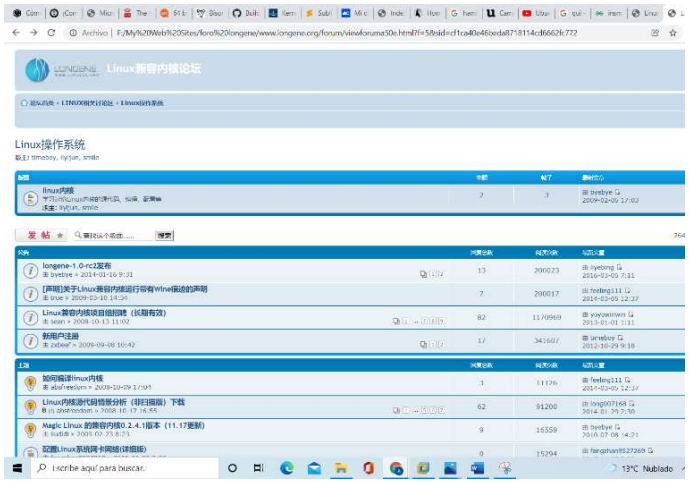


Nota. Esta imagen muestra una captura de pantalla del foro de la página del proyecto Longene. (WaybackMachine, 2022)

La copia que se realizó a la web del proyecto se hizo usando el programa Httrack. Esto se puede visualizar en la Figura 5, tan solo viendo la url.

Figura 5

Pantalla de Foro del proyecto Longene

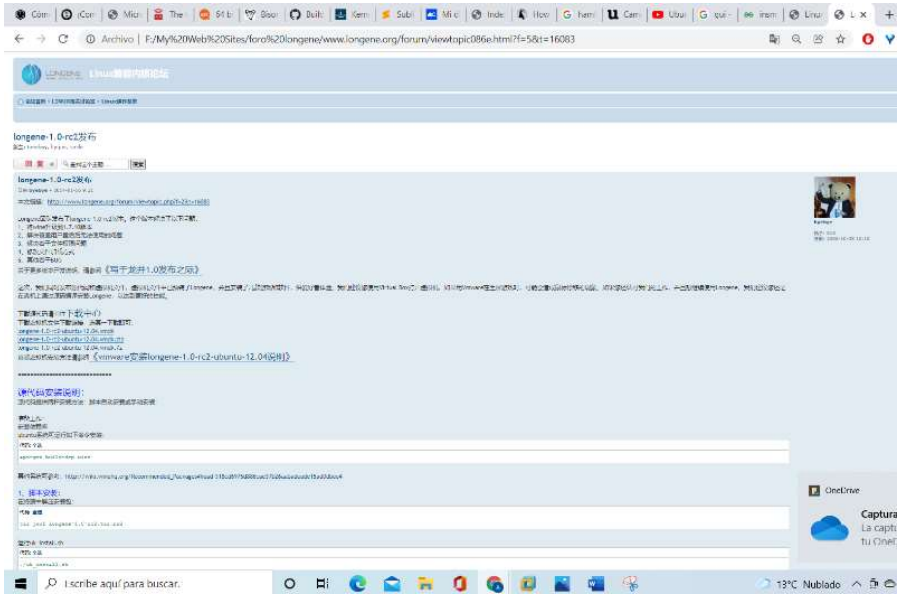


Nota. Esta imagen es una captura del foro del proyecto Longene el cual el investigador tiene clonado. (Diseño Propio)

Dentro del apartado longene-1.0-rc2 se encuentra la lista de comandos que servían para la instalación de Longene en Ubuntu 12. Se procede a mostrar la captura de estos en las figuras 6 y 7.

Figura 6

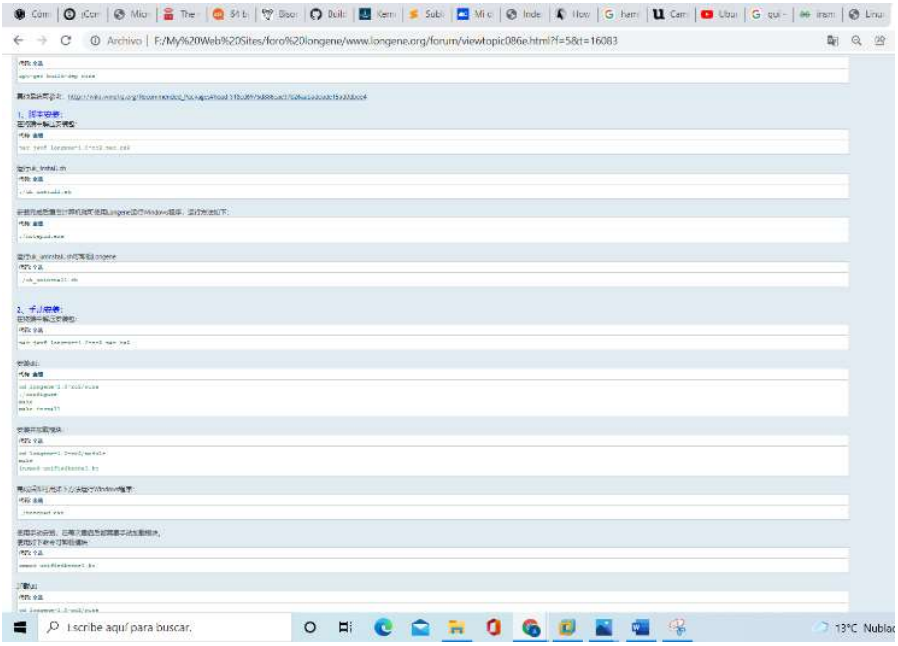
Hilo de Instalación del foro versión 1.0-rc2



Nota. Esta imagen es la pantalla de instalación de Longene que se tenía en el foro de Longene. (Longene, 2014)

Figura 7

Hilo de Instalación del foro versión 1.0-rc2



Nota. Esta imagen es la pantalla de instalación de Longene que se tenía en el foro de Longene. (Longene, 2014)

CAPITULO II: MARCO TEORICO DE LA TESIS

2.1. ANTECEDENTES DE LA TESIS

2.1.1. Antecedentes a Nivel Nacional

1. Tesis “PERSONALIZACIÓN DEL SISTEMA OPERATIVO GNU/LINUX UBUNTU 13 PARA MEJORAR EL ACCESO A INTERNET SATELITAL CON GILAT”, elaborado por (Zapana Mamani, 2014)

“Se menciona que: El acceso a Internet mejoro para los usuarios de GILAT TO HOME PERU, ofreciéndoles un sistema operativo que no requiere el pago de licencias ni contratos con la empresa que desarrolla el producto, pues se dispone de una versión gratuita en el Internet descargable y en modo Live DVD para su instalación y configuración, accediendo a los FOROS o BLOGS, pues estos han contribuido en la unión de personas no expertas en el área de Informática, pero potenciales clientes en los emergentes sitios web de negocio electrónico, así como la banca y la política, esto permite tener una completa democracia electrónica y globalización de la sociedad. Las zonas rurales vienen siendo anexadas lentamente a esta sociedad de la información, una de las empresas que provee el servicio de Internet Satelital es GILAT TO HOME PERU, empresa transnacional con sedes en todos los países de Sudamérica, esta empresa provee el acceso a Internet que si bien no puede ser comparado con el Internet de Banda ancha, ofrece el acceso necesario a zonas geográficamente inaccesibles a la telefonía de fibra óptica. Preocupados por mejorar las Aulas de Innovación Tecnológica, es que la sede en Puno de GILAT, requirió aportar una solución tecnológica y la creación de una distribución con la principal característica de tener preinstaladas una gran cantidad de software educativo, bajo una lista categorizada de acuerdo a las diversas áreas educativas”. (Zapana Mamani, 2014)

Nota. Esta tesis demuestra que existen muchos sitios en el país que requieren el uso de sistemas operativos Linux. Y como en muchos casos es necesario la modificación de los sistemas operativos para lograr el funcionamiento más óptimo para un público especifico.

2. Tesis “DISEÑO DE UN MODELO LOGICO DE VIRTUALIZACION CLIENTE SERVIDOR PARA ORGANIZACIONES EMPRESARIALES”, elaborado por (Tejada Castillo & Torres Quezada, 2014).

“En la actualidad, diariamente las empresas e instituciones están cambiando sus ambientes de trabajo personalizados por la adecuación del sistema de trabajo cliente – servidor, para obtener servicios. La adaptación del sistema cliente - servidor en la mayoría de los casos no se realiza de una forma óptima, la colocación de las computadoras cliente representan una fuerte inversión para empresas e instituciones, considerando que hay que invertir en equipos robustos respecto a “hardware” y en “software” propio para ellos, los equipos cuentan con recursos como procesador y memoria RAM, licencia de sistema operativo, licencia de antivirus y las licencias de programas comerciales de oficina o los que se necesiten para llevar a cabo sus actividades. La virtualización es un concepto realmente interesante, que ha experimentado un notable desarrollo en los últimos años, y que parece que por fin está encontrando sus caminos productivos y no solo experimentales. Si leemos algunos análisis de carga de servidores, millones en todo el mundo, podemos encontrarnos con estadísticas interesantes de cómo, y de forma general, tan solo aprovechamos entre un 20-30% de la capacidad de proceso de estos servidores. Es decir, a cualquier servidor a nivel global le sobran el 70% de sus recursos. Si a esto unimos la proliferación de servidores, dada la reducción de precio que han sufrido, nos encontramos con un parque infrautilizado y con

dificultades de mantenimiento. Con la realización de este proyecto pretendemos:

- Reducir y simplificar la infraestructura de servidores.
- Mejor uso de los recursos de hardware, obtener un máximo uso de la capacidad de procesamiento de un servidor,
- Ahorro de espacio utilizando clientes ligeros (then client) para sustituir los ordenadores que son utilizados como terminales.
- Reducción de costos en el pago de licencias de software privativo.
- Administración global centralizada y simplificada.
- Reducción de los costes de IT gracias al aumento de la eficiencia y la flexibilidad en el uso de recursos.
- Ayudar a la preservación del medio ambiente, brindando índices más bajos de consumo de energía, que se produce cuando varios servidores a baja utilización ejecutando aplicaciones se combinan en una única pieza física de hardware.” (Tejada Castillo & Torres Quezada, 2014)

Nota. Este antecedente propone el uso de virtualización de servidores, debido a que muchas organizaciones nacionales no saben manejar adecuadamente sus recursos, pero para realizar esto se requiere de programas externos los cuales tienen ciertas limitaciones de uso. Mientras que si las organizaciones manejaran un entorno Linux podrían lograr esto con más facilidad hasta considerando “el balanceo de carga”, así pues, de esta forma los usuarios comunes podrían realizar sus tareas en herramientas Windows y a su vez ceder ciertos recursos los cuales no se usan en su totalidad.

3. Tesis “DESARROLLO DE UNA DISTRIBUCION PERSONALIZADA DEL SISTEMA OPERATIVO GNU/LINUX UBUNTU 15 PARA LA UPSC”, elaborado por (Mamani Condori, 2016).

“El centro de Cómputo de la Universidad Privada san Carlos de Puno, cuenta con equipos modernos y está dedicada para el uso de alumnos de un total de cinco carreras profesionales las que hacen uso los cinco días laborables en horarios ininterrumpidos de 7:00am a 10:00pm lo que conlleva a un tráfico de información y descargas con potenciales virus informáticos al estar trabajando sobre el sistema operativo Windows, generando así ocasionales mantenimientos en computadoras por 24 horas. Para mejorar esta situación se presenta una solución de sistema operativo adaptado para el diario uso de los computadores, así como el uso de internet, copiando el archivado de datos evitando sobre todo el problema de virus informáticos que en su mayoría están orientados a dañar el sistema operativo de la competencia. UPSC-Ubuntu San Carlos 2015, viene pre-instalado con aplicaciones de escritorio, compiladores, procesadores de texto, hojas de cálculo y herramientas de edición de audio y video, todo en un solo DVD, se ha agregado soporte de idiomas inglés y Español para un sencillo manejo del mismo”.

Nota. Esta tesis demuestra que existen muchos sitios en el país que requieren el uso de sistemas operativos Linux. Este caso es muy parecido al primer antecedente solo que con el enfoque de una organización como usuario, debido a esto también se tienen que considerar a los trabajadores los cuales están acostumbrados al uso de las herramientas más comunes de entorno Windows y la capacitación de los mismos representa inconvenientes económicos y de tiempo.

2.1.2. Antecedente a Nivel Internacional

1. Paper “Performance Evaluation of Virtualization Technologies for Windows Programs Running on Linux Operating System” elaborado por (Conghui, Jing, Li, & Qiao, 2012)

“System virtual machine, kernel virtualization in kernel space and kernel virtualization in user space are the three virtualization technologies for Windows programs running on Linux operating system. System virtual machine technology can support Linux operating system to install and run Windows operating system and its programs by simulating the complete or a subnet of hardware. However, the other two virtualization technologies respectively emulate the function of Windows kernel in the Linux kernel space and in Linux user space. Because the three virtualization technologies have different principles and methods of realization, the same Windows programs running on them have different performance under the same condition. In this paper, we evaluate the performance of the system virtual machine, kernel virtualization in kernel space and kernel virtualization in user space. Using VMware Workstation, Longene and Wine as the typical representative of the three virtualization technologies, we measure the performance of the three virtualization technologies by the method of benchmark application, and compare them with the test results on the native Windows XP. And the experimental results demonstrate that Wine has the better performance, compared with Windows programs running on the VMware Workstation and Longene”. (Conghui, Jing, Li, & Qiao, 2012)

Nota. Este paper tiene como objetivo la evaluación del rendimiento de virtualización con Wine, VMware y Longene. A su vez se demuestra el uso de virtualización y el modelo de ejecución que

posee cada una de estas alternativas; así como comparativa lógica y cuantitativa.

2. Articulo “Security implications of running windows software on a Linux system using Wine: a malware analysis study” elaborado por (Duncan & Schreuders, 2018)

“Se considera que Linux es menos propenso a los programas maliciosos en comparación con otros sistemas operativos, por lo que los usuarios de Linux rara vez ejecutan programas antimalware. Sin embargo, muchas aplicaciones de software populares publicadas en otras plataformas no pueden ejecutarse de forma nativa en Linux. Wine es una popular capa de compatibilidad para ejecutar programas de Windows en Linux. El nivel de riesgo de seguridad que Wine supone para los usuarios de Linux está en gran medida sin documentar. Este proyecto se llevó a cabo para evaluar las implicaciones de seguridad del uso de Wine, y para determinar si algún tipo específico de malware o comportamiento de malware tiene un efecto significativo en el éxito del malware en Wine. Se aplicó un análisis dinámico (tanto automatizado como manual) a 30 muestras de malware tanto en un entorno Windows como en un entorno Linux con Wine. El comportamiento analizado incluía el acceso al sistema de archivos, al registro y a la red, así como la generación de procesos y servicios. El comportamiento se comparó para determinar el éxito del malware en Wine. Los resultados del estudio demuestran que Wine puede plantear serias implicaciones de seguridad cuando se utiliza para ejecutar software de Windows en un entorno Linux. Se ejecutaron con éxito cinco muestras de malware de Windows a través de Wine en un sistema Linux. No se descubrió ninguna relación significativa entre el éxito del malware y su comportamiento de alto nivel o el tipo de malware. Sin embargo, ciertas llamadas a la API no pudieron recrearse en un entorno Linux y provocaron el fracaso del malware al ejecutarse a través de Wine.

Esto sugiere que determinadas muestras de malware que utilizan estas llamadas a la API nunca se ejecutarán completamente con éxito en un entorno Linux. En consecuencia, el éxito de algunas muestras puede determinarse a partir de la observación de las llamadas a la API cuando se ejecutan en un entorno Windows". (Duncan & Schreuders, 2018)

Nota. Este articulo tiene como objetivo primordial, el analizar los malware en entornos Linux a partir de la ejecución de programas Windows que contengan malwares mediante el uso de Wine. La importancia de un sistema que no posee tanta cantidad de malwares en la red. Así como el uso de Wine como mediador entre los programas Windows en entorno Linux.

3. Paper "Decreasing information technology expenses by using emulators on Windows and Linux platforms" elaborado por (Skendzic, Kovacic, & Jugo, Decreasing information technology expenses by using emulators on Windows and Linux platforms, 2011)

"Los sistemas operativos de propósito general, como Linux, se utilizan cada vez más en los sistemas integrados. Los recursos computacionales suelen ser limitados, y los procesadores integrados suelen tener una cantidad limitada de memoria. Esto hace que el tamaño del código sea especialmente importante. En este artículo se describen técnicas para reducir automáticamente la huella de memoria de los sistemas operativos de propósito general en plataformas empotradas. El problema se complica por el hecho de que el código del kernel tiende a ser bastante diferente del código de la aplicación ordinaria, incluyendo la presencia de una cantidad significativa de código ensamblador escrito a mano, múltiples puntos de entrada, rutas de flujo de control implícitas que involucran a los manejadores de interrupción, y el flujo de control indirecto frecuente

a través de punteros de función. Utilizamos una novedosa técnica de "descompilación aproximada" para aplicar el análisis del programa a nivel de fuente al código ensamblador escrito a mano. Una implementación prototipo de nuestras ideas en una plataforma Intel x86, aplicada a un kernel Linux configurado para excluir el código innecesario, obtiene una reducción del tamaño del código cercana al 24%".(Skendzic, Kovacic, & Jugo, Decreasing information technology expenses by using emulators on Windows and Linux platforms, 2011)

Nota. El objetivo de este paper consiste en describir técnicas para reducir la huella de memoria tomando como base los sistemas operativos Linux que son usados en varias áreas. Así mismo el código del kernel toma en cuenta la ejecución de programas superior en esta capa para lograr una óptima ejecución.

4. Paper "Decreasing information technology expenses by using emulators on Windows and Linux platforms" elaborado por (Skendzic, Kovacic, & Jugo, Decreasing information technology expenses by using emulators on Windows and Linux platforms, 2011)

"Los dos sistemas operativos (también llamados SO) más populares hoy en día son Windows (amplia gama de productos) y Linux (varias distribuciones). Cada sistema operativo tiene una historia de desarrollo diferente, una aplicación dirigida y una implementación de hardware diferente. La popularidad de los sistemas operativos Windows está en un nivel muy alto pero, hoy en día, Linux ha demostrado que el software de alta calidad no tiene que ser necesariamente un software comercial. Los sistemas operativos Linux y Windows y sus aplicaciones no son compatibles debido a las diferencias en el núcleo y los sistemas de archivos y la ejecución del código de los programas. Eso significa que los usuarios tienen que

decidir qué sistema operativo van a utilizar en función de las posibilidades económicas y financieras. El problema de utilizar aplicaciones bajo diferentes sistemas operativos puede resolverse con programas emuladores en ambas plataformas, Windows y Linux. Los emuladores más utilizados actualmente son Wine (en plataformas Linux) y VMware (en ambas plataformas). Los emuladores ofrecen la posibilidad de ejecutar diferentes aplicaciones (incluso la instalación de sistemas operativos) en una plataforma para la que no fueron escritas previamente. Si se eligen sistemas operativos menos costosos (o gratuitos) y emuladores apropiados, los gastos generales en tecnología de la información pueden disminuir considerablemente". (Skendzic, Kovacic, & Jugo, Decreasing information technology expenses by using emulators on Windows and Linux platforms, 2011)

Nota. Este paper tiene como objetivo la comparativa económica que se tiene entre los Sistemas Operativos Windows y Linux, que a su vez dependiendo de las necesidades de los usuarios tienden a escoger una u otra según sus requerimientos de actividades, requerimientos económicos y de uso. Los cuales pueden ser complementados con alternativas de virtualización VMWare y Wine respectivamente.

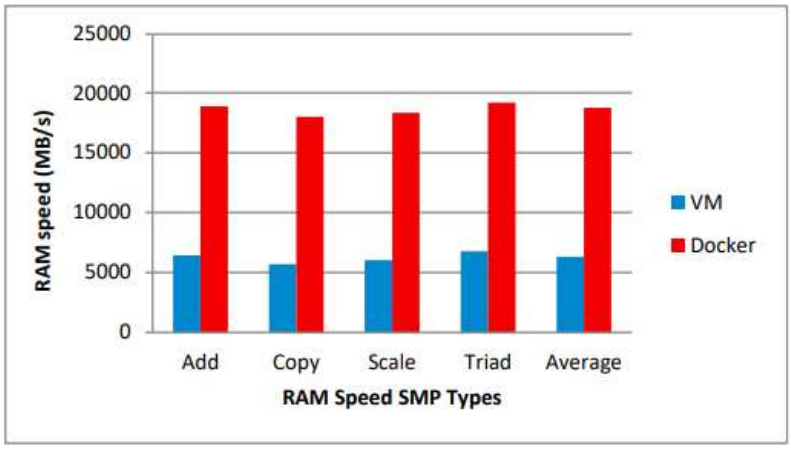
5. Paper "Performance Evaluation of Docker Container and Virtual Machine" elaborado por (Potdara, Narayan, Kengond, & Mulla, 2020)

"Rendimiento de la memoria: RAM speed/SMP (Symmetric Multiprocessing) es una herramienta de evaluación comparativa de la memoria y la caché que se utiliza para medir la velocidad de la memoria RAM de las tecnologías de virtualización, es decir, Docker y Virtual Machine. La Fig. 8. representa la comparación de la velocidad de la RAM entre las tecnologías de virtualización. Los siguientes dos parámetros principales se consideran mientras se

prueba la velocidad de la RAM. Los componentes INTmark y FLOATmark se utilizan en la herramienta de evaluación comparativa RAM Speed SMP, que mide el máximo rendimiento posible de la caché y la memoria mientras se leen y escriben bloques individuales de datos. INTmem y FLOATmem, son simulaciones sintéticas pero equilibradas con el mundo real de la informática. Cada una de ellas consta de cuatro subpruebas (Copy, Scale, Add, Triad) para medir diferentes aspectos del rendimiento de la memoria. La transferencia de datos de una posición de memoria a otra se realiza mediante el comando de copia, es decir, (X = Y). La modificación de los datos antes de escribirlos se multiplica por un determinado valor constante se realiza mediante el comando scale, es decir, (X = n*Y). Los datos se leen de la primera posición de memoria y luego de la segunda cuando se llama al comando ADD. Luego el dato resultante se coloca en el tercer lugar (X = Y + Z). La tríada es una combinación de Add y Scale. Los datos se leen desde la primera ubicación de memoria para escalar y luego se añaden desde el segundo lugar para escribirlos en el tercer lugar (X = n*Y + Z). La Fig. 8. muestra el rendimiento de la memoria con respecto a la prueba RAM Speed SMP". (Potdara, Narayan, Kengond, & Mulla, 2020)

Figura 8

Comparativa de Velocidad de RAM entre tecnologias de Virtualización



Nota. Este grafico representa el resultado de uso de RAM del paper (Potdara, Narayan, Kengond, & Mulla, 2020)

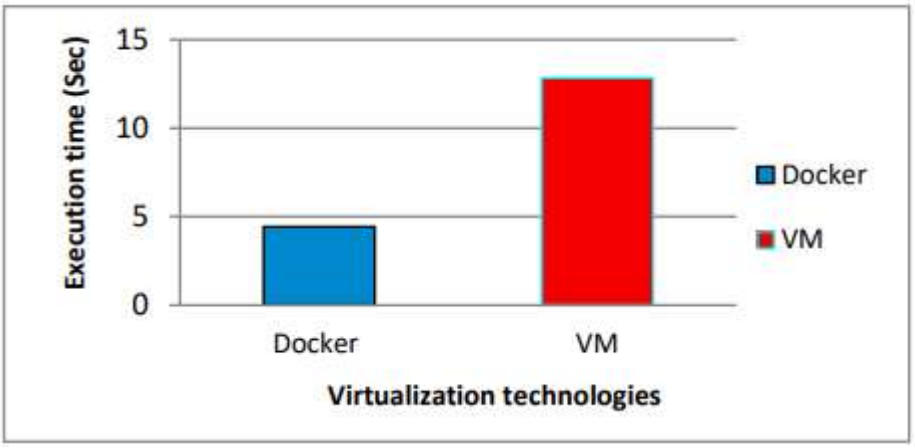
Nota. Este paper tiene como objetivo la comparativa de rendimiento entre la Virtualización y un Docker. La prueba primordial de este punto se centra en la medición de rendimiento de memoria RAM, esto se realiza por un método de adición, copiado y escalado promedio. Aunque es relevante es solo una forma de realizar las pruebas de rendimiento, pero es importante en el fondo ya que muestra que la dimensión se da en base a la velocidad de la RAM en (MB/s).

6. Paper “Performance Evaluation of Docker Container and Virtual Machine” elaborado por (Potdara, Narayan, Kengond, & Mulla, 2020)

   Medición de la velocidad de la operación: El problema de las ocho reinas coloca ocho reinas en un tablero de ajedrez de 8×8 de forma que ninguna de ellas se ataque entre sí. La prueba mide el tiempo que se tarda en resolver el problema. El programa de las ocho reinas está escrito en python y determina el rendimiento computacional del sistema. La Figura 9. muestra el rendimiento computacional tanto de docker como de las máquinas virtuales. Según el tiempo de ejecución, el contenedor docker tarda menos en resolver el problema, mientras que la máquina virtual tarda mucho más.

Figura 9

Pruebas de Programa de Comparativa de Rendimiento 8 Reinas

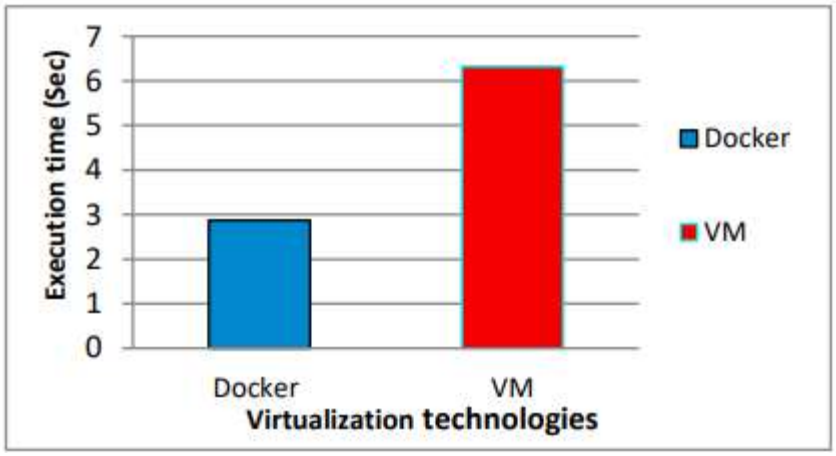


Nota. Este grafico representa el resultado de tiempo de ejecución del paper en 8 Reinas. (Potdara, Narayan, Kengond, & Mulla, 2020)

Prueba de ocho rompecabezas: Se toma un tablero de 4×4 con 8 fichas y un espacio vacío. Utilizando el espacio vacío, la disposición del número de azulejos para que coincida con la configuración final es el objetivo principal. Puede deslizar cuatro operaciones adyacentes (derecha, izquierda, abajo y arriba) azulejos en el espacio vacío la prueba mide cuánto tiempo se tarda en resolver el problema. El programa Eight puzzle está escrito en python y determina el rendimiento computacional del sistema. La Fig. 10. muestra el rendimiento computacional tanto de docker como de las máquinas virtuales. Según el tiempo de ejecución, el contenedor docker tarda menos en resolver el problema, mientras que la máquina virtual tarda mucho más.

Figura 10

Prueba de Comparativa de Rendimiento Ocho Rompecabezas



Nota. Este grafico representa el resultado de tiempo de ejecución del paper en 8 Rompecabezas. (Potdara, Narayan, Kengond, & Mulla, 2020)

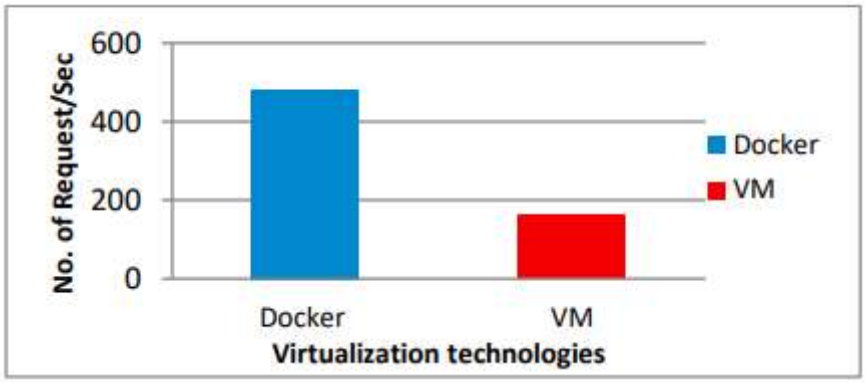
Nota. Este paper tiene como objetivo la comparativa de rendimiento entre la Virtualización y un Docker. En este apartado se mide el tiempo de ejecución de un programa en Python, comparando cada uno de los entornos, el tiempo es medido en segundos y usando un formato de rompecabezas de 4x4.

7. Paper “Performance Evaluation of Docker Container and Virtual Machine” elaborado por (Potdara, Narayan, Kengond, & Mulla, 2020)

Pruebas de carga: Para la comparación del rendimiento de las pruebas de carga se utiliza la herramienta Apache Benchmark, en la que se mide el número de peticiones por segundo que puede tolerar un determinado sistema. Se ejecuta un programa python para probar la carga utilizando la herramienta Apache Benchmarking. La Fig. 11. muestra que el análisis de rendimiento para VM es mucho menor comparado con el de Docker. Esto se debe a la mayor latencia de la red en la máquina virtual que en Docker. El análisis muestra que el contenedor Docker es mejor que la máquina virtual en el manejo del número de peticiones por segundo.

Figura 11

Pruebas de Carga entre Dockers y Máquinas Virtuales



Nota. Este grafico representa el resultado de cantidad de solicitudes por segundo en el paper (Potdara, Narayan, Kengond, & Mulla, 2020)

Este paper tiene como objetivo la comparativa de rendimiento entre la Virtualización y un Docker. La prueba en esta ocasión se basa en la cantidad máxima de solicitudes que puede tolerar por latencia.

2.2. BASES TEÓRICO - CIENTIFICAS

2.2.1. SISTEMA OPERATIVO

"Es difícil definir qué es un sistema operativo aparte de decir que es el software que se ejecuta en modo kernel (además de que esto no siempre es cierto). Parte del problema es que los sistemas operativos realizan dos funciones básicas que no están relacionadas: proporcionar a los programadores de aplicaciones (y a los programas de aplicaciones, naturalmente) un conjunto abstracto de recursos simples, en vez de los complejos conjuntos de hardware; y administrar estos recursos de hardware. Dependiendo de quién se esté hablando, el lector podría escuchar más acerca de una función o de la otra. Por lo que se analizaran ambas.

El sistema operativo como una máquina extendida

La arquitectura (conjunto de instrucciones, organización de memoria, E/S y estructura de bus) de la mayoría de las computadoras a nivel de lenguaje máquina es primitiva y compleja de programar, en especial para la entrada/salida. Para hacer este punto más concreto,

se considera la forma en que se lleva a cabo la E/S de disco flexible mediante los dispositivos controladores (device controllers) compatibles NEC PD765 que se utilizan en la mayoría de las computadoras personales basadas en Intel (a lo largo de este libro utilizaremos los términos “disco flexible” y “diskette” indistintamente). Se utiliza el disco flexible como un ejemplo debido a que, aunque obsoleto, es mucho más simple que un disco duro moderno. El PD765 tiene 16 comandos, cada uno de los cuales se especifica mediante la carga de 1 a 9 bytes en un registro de dispositivo. Estos comandos son para leer y escribir datos, desplazar el brazo del disco y dar formato a las pistas, así como para inicializar, detectar, restablecer y recalibrar el dispositivo controlador y las unidades. Los comandos más básicos son read y write (lectura y escritura), cada uno de los cuales requiere 13 parámetros, empaquetados en 9 bytes. Estos parámetros especifican elementos tales como la dirección del bloque de disco a leer, el número de sectores por pista, el modo de grabación utilizado en el medio físico, el espacio de separación entre sectores y lo que se debe hacer con una marca de dirección de datos eliminados. Si no se comprenden estos tecnicismos, no se preocupe: ése es precisamente el punto, pues se trata de algo bastante oscuro. Cuando la operación se completa, el chip del dispositivo controlador devuelve 23 campos de estado y error, empaquetados en 7 bytes. Como si esto no fuera suficiente, el programador del disco flexible también debe estar constantemente al tanto de si el motor está encendido o apagado. Si el motor está apagado, debe encenderse (con un retraso largo de arranque) para que los datos puedan ser leídos o escritos. El motor no se debe dejar demasiado tiempo encendido porque se desgastará. Por lo tanto, el programador se ve obligado a lidiar con el problema de elegir entre tener retrasos largos de arranque o desgastar los discos flexibles (y llegar a perder los datos). Sin entrar en los detalles reales, debe quedar claro que el programador promedio tal vez no desee involucrarse demasiado con la programación de los discos

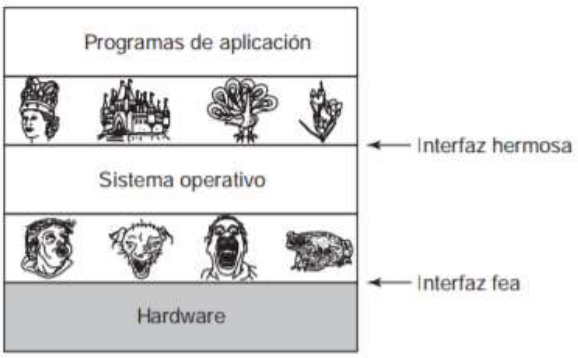
flexibles (o de los discos duros, que son aún más complejos). En vez de ello, lo que desea es una abstracción simple de alto nivel que se encargue de lidiar con el disco. En el caso de los discos, una abstracción común sería que el disco contiene una colección de archivos con nombre. Cada archivo puede ser abierto para lectura o escritura, después puede ser leído o escrito y, por último, cerrado. Los detalles, tales como si la grabación debe utilizar o no la modulación de frecuencia y cuál es el estado del motor en un momento dado, no deben aparecer en la abstracción que se presenta al programador de aplicaciones. La abstracción es la clave para lidiar con la complejidad. Las buenas abstracciones convierten una tarea casi imposible en dos tareas manejables. La primera de éstas es definir e implementar las abstracciones; la segunda, utilizarlas para resolver el problema en cuestión. Una abstracción que casi cualquier usuario de computadora comprende es el archivo: es una pieza útil de información, como una fotografía digital, un mensaje de correo electrónico almacenado o una página Web. Es más fácil lidiar con fotografías, correos electrónicos y páginas Web que con los detalles de los discos, como en el caso del disco flexible descrito. El trabajo del sistema operativo es crear buenas abstracciones para después implementar y administrar los objetos abstractos entonces creados.

Este punto es tan importante que vale la pena repetirlo en distintas palabras. Con el debido respeto a los ingenieros industriales que diseñaron la Macintosh, el hardware es feo. Los procesadores, memorias, discos y otros dispositivos reales son muy complicados y presentan interfaces difíciles, enredadas, muy peculiares e inconsistentes para las personas que tienen que escribir software para utilizarlos. Algunas veces esto se debe a la necesidad de tener compatibilidad con el hardware anterior; otras, a un deseo de ahorrar dinero, y otras más, a que los diseñadores de hardware no tienen idea (o no les importa) qué tan grave es el problema que están ocasionando para el software. Una de las principales tareas del

sistema operativo es ocultar el hardware y presentar a los programas (y a sus programadores) abstracciones agradables, elegantes, simples y consistentes con las que puedan trabajar. Los sistemas operativos ocultan la parte fea con la parte hermosa, como se muestra en la figura 12.

Figura 12

Diferencias entre interfaz de SO y de aplicaciones



Nota. Esta imagen representa la forma en la que se ven los aplicativos a diferencia de los sistemas operativos. (TANENBAUM, 2009)

Hay que recalcar que los verdaderos clientes del sistema operativo son los programas de aplicación (a través de los programadores de aplicaciones, desde luego). Son los que tratan directamente con el sistema operativo y sus abstracciones. En contraste, los usuarios finales tienen que lidiar con las abstracciones que proporciona la interfaz de usuario, ya sea un shell de línea de comandos o una interfaz gráfica. Aunque las abstracciones en la interfaz de usuario pueden ser similares a las que proporciona el sistema operativo, éste no siempre es el caso. Para aclarar este punto, se considera el escritorio normal de Windows y el indicador para comandos orientado a texto. Ambos son programas que se ejecutan en el sistema operativo Windows y utilizan las abstracciones que este sistema proporciona, pero ofrecen interfaces de usuario muy distintas. De manera similar, un usuario de Linux que ejecuta Gnome o KDE ve una interfaz muy distinta a la que ve un usuario de Linux que trabaja directamente encima del Sistema X Window subyacente

(orientado a texto), pero las abstracciones del sistema operativo subyacente son las mismas en ambos casos. En este libro se estudian detalladamente las abstracciones que se proporcionan a los programas de aplicación, pero se trata muy poco acerca de las interfaces de usuario, que es un tema bastante extenso e importante, pero que sólo está relacionado con la periferia de los sistemas operativos.

El sistema operativo como administrador de recursos

El concepto de un sistema operativo cuya función principal es proporcionar abstracciones a los programas de aplicación responde a una perspectiva de arriba hacia abajo. La perspectiva alterna, de abajo hacia arriba, sostiene que el sistema operativo está presente para administrar todas las piezas de un sistema complejo. Las computadoras modernas constan de procesadores, memorias, temporizadores, discos, ratones, interfaces de red, impresoras y una amplia variedad de otros dispositivos.

En la perspectiva alterna, el trabajo del sistema operativo es proporcionar una asignación ordenada y controlada de los procesadores, memorias y dispositivos de E/S, entre los diversos programas que compiten por estos recursos.

Los sistemas operativos modernos permiten la ejecución simultánea de varios programas. Imagine lo que ocurriría si tres programas que se ejecutan en cierta computadora trataran de imprimir sus resultados en forma simultánea en la misma impresora. Las primeras líneas de impresión podrían provenir del programa 1, las siguientes del programa 2, después algunas del programa 3, y así en lo sucesivo: el resultado sería un caos. El sistema operativo puede imponer orden al caos potencial, guardando en búferes en disco toda la salida destinada para la impresora. Cuando termina un programa, el sistema operativo puede entonces copiar su salida, previamente almacenada, del archivo en disco a la impresora, mientras que al mismo tiempo el otro programa puede continuar

generando más salida, ajeno al hecho de que la salida en realidad no se está enviando a la impresora todavía.

Cuando una computadora (o red) tiene varios usuarios, la necesidad de administrar y proteger la memoria, los dispositivos de E/S y otros recursos es cada vez mayor; de lo contrario, los usuarios podrían interferir unos con otros. Además, los usuarios necesitan con frecuencia compartir no sólo el hardware, sino también la información (archivos o bases de datos, por ejemplo). En resumen, esta visión del sistema operativo sostiene que su tarea principal es llevar un registro de qué programa está utilizando qué recursos, de otorgar las peticiones de recursos, de contabilizar su uso y de mediar las peticiones en conflicto provenientes de distintos programas y usuarios. La administración de recursos incluye el multiplexaje (compartir) de recursos en dos formas distintas: en el tiempo y en el espacio. Cuando un recurso se multiplexa en el tiempo, los distintos programas o usuarios toman turnos para utilizarlo: uno de ellos obtiene acceso al recurso, después otro, y así en lo sucesivo. Por ejemplo, con sólo una CPU y varios programas que desean ejecutarse en ella, el sistema operativo primero asigna la CPU a un programa y luego, una vez que se ha ejecutado por el tiempo suficiente, otro programa obtiene acceso a la CPU, después otro, y en un momento dado el primer programa vuelve a obtener acceso al recurso. La tarea de determinar cómo se multiplexa el recurso en el tiempo (quién sigue y durante cuánto tiempo) es responsabilidad del sistema operativo. Otro ejemplo de multiplexaje en el tiempo es la compartición de la impresora. Cuando hay varios trabajos en una cola de impresión, para imprimirlos en una sola impresora, se debe tomar una decisión en cuanto a cuál trabajo debe imprimirse a continuación.

El otro tipo de multiplexaje es en el espacio. En vez de que los clientes tomen turnos, cada uno obtiene una parte del recurso. Por ejemplo, normalmente la memoria principal se divide entre varios programas en ejecución para que cada uno pueda estar residente al

mismo tiempo (por ejemplo, para poder tomar turnos al utilizar la CPU). Suponiendo que hay suficiente memoria como para contener varios programas, es más eficiente contener varios programas en memoria a la vez, en vez de proporcionar a un solo programa toda la memoria, en especial si sólo necesita una pequeña fracción.

Desde luego que esto genera problemas de equidad y protección, por ejemplo, y corresponde al sistema operativo resolverlos. Otro recurso que se multiplexa en espacio es el disco duro. En muchos sistemas, un solo disco puede contener archivos de muchos usuarios al mismo tiempo. Asignar espacio en disco y llevar el registro de quién está utilizando cuáles bloques de disco es una tarea típica de administración de recursos común del sistema operativo.” (TANENBAUM, 2009)

2.2.2. KERNEL LINUX

“Normalmente, cuando se habla de Linux, se suele hacer en relación a distribuciones como Ubuntu, Debian, Mint, etc. Sin embargo, estas distribuciones no son sistemas operativos como tal, sino más bien son «adaptaciones» de un sistema operativo de código abierto: Linux. Y, por lo tanto, a pesar de las peculiaridades de cada una, todas ellas comparten una misma base, lo que conocemos como Kernel Linux.

Linux, como indica su propia documentación, nació como un clon de otro sistema operativo llamado Unix. Concretamente como un sistema alternativo inspirado en Minix (que, a su vez, clonaba los conceptos de Unix). Por sus propiedades, Linux es un sistema operativo real, aunque nadie lo utiliza como tal, sino que recurre a las distribuciones ya que, con ellas, se vuelve mucho más útil y sencillo de utilizar.

Por ello, no se suele decir «voy a instalar Linux» refiriéndose a un sistema operativo concreto, sino que lo que se instala son versiones, o distribuciones, de este núcleo, creadas por empresas o por la comunidad, que comparten una misma base: el Kernel.

El Kernel Linux es el núcleo del sistema operativo. Esta es la parte de software más importante de cualquier sistema operativo. Windows tiene su propio núcleo privado, Apple tiene el suyo (basado en Unix, por cierto), y Linux es el Kernel que utilizan todas las distribuciones. Y su principal función es encargarse de controlar el hardware del ordenador.

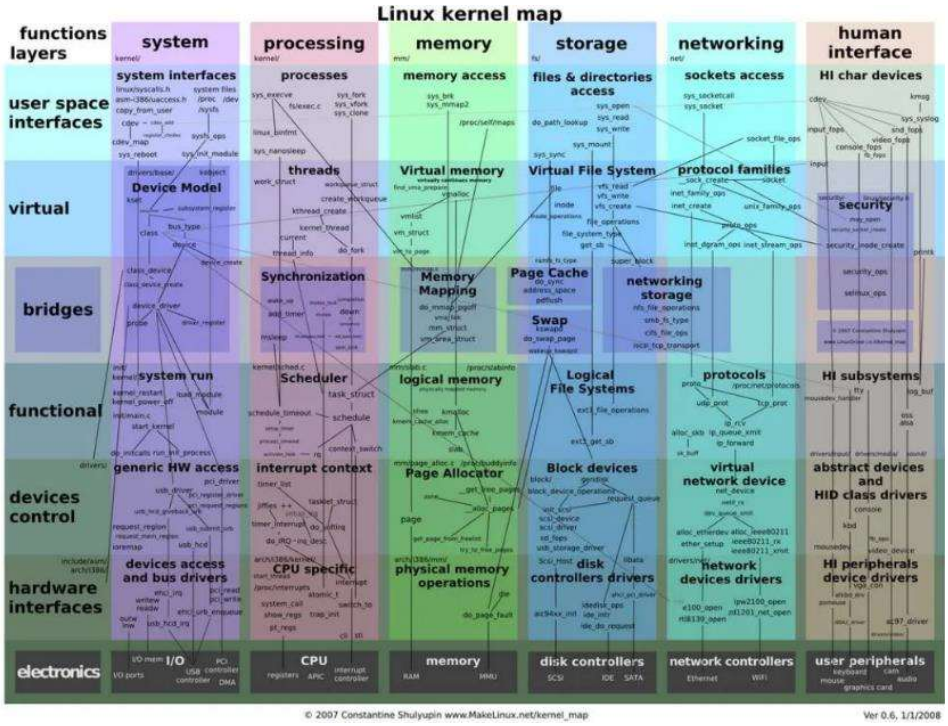
Concretamente, este núcleo es el responsable de gestionar la memoria del sistema, el tiempo de procesos, gestionar todos los procesos, controlar las llamadas del sistema y las conexiones entre procesos y permitir a todo el software tener acceso al hardware, especialmente a los periféricos conectados al ordenador.

Es tal la importancia del Kernel a la hora de controlar el hardware que, de sus más de 28 millones de líneas de código, la mayor parte de él son drivers. Y esto, aunque es bueno para la compatibilidad, empieza a ser un problema para el rendimiento.

Es un núcleo mayormente libre semejante al núcleo de Unix. Linux es uno de los principales ejemplos de software libre y de código abierto. Linux está licenciado bajo la GPL v2 salvo el hecho que tiene blobs binarios no-libres y la mayor parte del software incluido en el paquete que se distribuye en su sitio web es software libre. Está desarrollado por colaboradores de todo el mundo. El desarrollo del día a día tiene lugar en la Linux Kernel Mailing List Archive.

Figura 13

Mapa de Kernel Linux



Nota. Este grafico muestra las funciones y su conexión en el mapa del Kernel Linux. (Velasco, SoftZoneES, 2020)

Es cierto que las versiones actuales del Kernel no tienen nada que ver con las primeras de 1991. Sin embargo, este núcleo está en constante desarrollo. Y cada pocas semanas se suelen ver nuevos lanzamientos. Pero no todos son igual de importantes, ya que depende en gran medida de su numeración.

Las versiones del Kernel Linux pueden tener 4 números que indican la versión: a.b.c.d

a) a indica la versión. Este número es el que menos cambia, ya que solo se suele dar el salto cuando hay cambios extremadamente grandes en el sistema. En toda su historia, solo ha cambiado 5 veces, en 2004, para la versión 1.0, en 2006, para la versión 2.0, en 2011, para la versión 3.0, en

2015, para la versión 4.0, y en 2019 para dar lugar a la versión actual, la 5.0.

b) b indica la subversión. Cuando se lanzan nuevas versiones, pero realmente son actualizaciones menores (nuevos drivers, optimizaciones, correcciones, etc), entonces en lugar de cambiar la versión, se cambia el número de la subversión.
c) c indica el nivel de revisión. Se suele cambiar este número, por ejemplo, cuando se introducen cambios menores, como parches de seguridad, correcciones de errores, etc.
d) d es el último subnivel de la versión. Apenas se utiliza, pero está reservado para que, si se lanza una versión con un fallo muy grave, se lance la nueva versión con este subnivel incluyendo exclusivamente la corrección de dicho fallo grave.

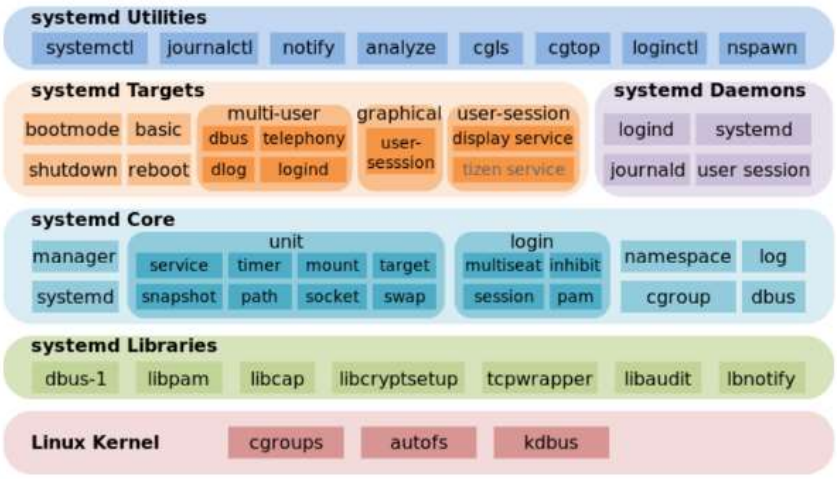
El Kernel es una de las partes más importantes del sistema operativo. Pero no es la única necesaria para poder denominar a Linux, hoy en día, un sistema operativo como tal. Como hemos explicado, este núcleo tiene todos los controladores y todo lo necesario para poder controlar el software y permisos al usuario acceder a él. Pero, para que sea realmente útil, debe tener otros componentes por encima de él antes de llegar al usuario.

- Controlador de demonios. Ya sea Init.d, Systemd, o cualquier otro software similar, es necesario tener un subsistema por encima del núcleo que se encargue de arrancar todos los procesos (daemons) necesarios para que el Kernel empiece a funcionar. Sin él no tenemos más que muchas líneas de código que no se ejecutan.
- Los procesos. Los daemons, demonios, o más conocidos como procesos, son todos los componentes que se quedan cargados en la memoria del sistema (controlados por el kernel) y que permiten a Linux funcionar. El servidor gráfico, por ejemplo, es el demonio que controlará el escritorio.

- Servidor gráfico. Conocido como X, es el encargado de poder ver los gráficos en la pantalla. Si vamos a usar Linux en modo texto, desde terminal, no es necesario. Pero si lo vamos a usar con escritorio, es necesario tener un x-server funcionando.
- Escritorio. Como su nombre indica, el escritorio del ordenador, donde tendremos todos nuestros programas y donde se abrirán las ventanas. Hay muchos escritorios para Linux, como GNOME, KDE o LXDE. Cada uno con unas características, ventajas e inconvenientes propios.
- Programas. Todo lo que ejecutemos desde el escritorio. Ya es la capa más elevada, y el punto a través del cual interactuamos con el ordenador.

Figura 14

Categorías de Bloques de compilación



Nota. Este grafico muestra las categorías de los bloques de compilación. (Velasco, SoftZoneES, 2020)

Cuando el Kernel, y todo lo demás, funciona en concordancia, es cuando podemos hablar de sistema operativo. O lo que es lo mismo, de distribución Linux." (Velasco, SoftZoneES, 2020)

Linux es multiprogramado, dispone de una memoria virtual, de gestión de memoria, conectividad en red y permite bibliotecas

compartidas. Linux es un sistema multiplataforma y es portable a cualquier arquitectura siempre y cuando esta disponga de una versión de GCC compatible.

La parte de un sistema operativo que se ejecuta sin privilegios o en espacio de usuario es la biblioteca del lenguaje C, esta es la que provee el entorno de tiempo de ejecución, y una serie de programas/herramientas los cuales se enfocan en permitir la administración y uso del núcleo, así como proveer servicios al resto de programas en espacio de usuario, formando junto con el núcleo el sistema operativo.

En un sistema con núcleo monolítico como Linux la biblioteca de lenguaje C consiste en una abstracción de acceso al núcleo. Algunas bibliotecas como la biblioteca de GNU proveen funcionalidad adicional para facilitar la vida del programador y usuario o mejorar el rendimiento de los programas.

En un sistema con micronúcleo la biblioteca de lenguaje C puede gestionar sistemas de archivos o controladores además del acceso al núcleo del sistema.

A los sistemas operativos que llevan Linux se les llama de forma genérica distribuciones Linux. Estas consisten en una recopilación de software que incluyen el núcleo Linux y el resto de programas necesarios para completar un sistema operativo. Las distribuciones más comunes son de hecho distribuciones GNU/Linux o distribuciones Android. El hecho de que compartan núcleo no significa que sean compatibles entre sí. Una aplicación hecha para GNU/Linux no es compatible con Android sin la labor adicional necesaria para que sea multiplataforma.

Las distribuciones GNU/Linux usan Linux como núcleo junto con el entorno de tiempo de ejecución del Proyecto GNU y una serie de programas y herramientas del mismo que garantizan un sistema funcional mínimo. La mayoría de distribuciones GNU/Linux incluye software adicional como entornos gráficos o navegadores web así como los programas necesarios para permitirse instalar a sí mismas. Los programas de instalación son aportados por el desarrollador de la distribución. Se les conoce como gestores de paquetes. Los creadores de una distribución también se pueden encargar de añadir configuraciones iniciales de los distintos programas incluidos en la distribución.

Las distribuciones Android incluyen el núcleo Linux junto con el entorno de ejecución y herramientas del proyecto AOSP de Google. Cada fabricante de teléfonos dispone de su propia distribución de Android a la cual modifica, elimina o añade programas extra: interfaces gráficas, tiendas de aplicaciones y clientes de correo electrónico son algunos ejemplos de programas susceptibles de ser añadidos, modificados o eliminados. Además de las distribuciones de los fabricantes de teléfonos existen grupos de programadores independientes que también desarrollan distribuciones de Android. LineageOS y Replicant son dos ejemplos de distribuciones Android independientes.

2.2.3. WINE

"Wine es una reimplementación de la interfaz de programación de aplicaciones de Win16 y Win32 para sistemas operativos basados en Unix. Permite la ejecución de programas diseñados para MS-DOS, y las versiones de Microsoft Windows 3.11, 95, 98, Me, NT, 2000, XP, Vista, 7, 8 y 10. El nombre Wine inicialmente fue un acrónimo para WINdows Emulator. Este significado fue cambiado posteriormente al acrónimo recursivo actual. Wine provee

- Un conjunto de herramientas de desarrollo para portar código fuente de aplicaciones Windows a Unix.
- Un cargador de programas, el cual permite que muchas aplicaciones para Windows 2.0/3.x/9X/ME/NT/2000/XP/Vista/7 y 8 se ejecuten sin modificarse en varios sistemas operativos Unix como macOS, BSD y Unix-like como GNU/Linux, Solaris es un conjunto de herramientas librerías y dependencias (no un emulador) de código abierto diseñado para permitir a los usuarios de Linux ejecutar sin problemas aplicaciones de Windows.

Ha pasado ya un año desde que los responsables lanzaron Wine 3.0, año durante el cual se han estado incluyendo una gran cantidad de cambios mejoras y novedades en esta herramienta para dar forma al nuevo Wine 4.0, la última versión de esta herramienta que llega para hacer mucho más sencillo y eficaz el poder ejecutar cualquier aplicación de Windows que queramos en Linux sin impedimentos.

Wine 4.0

Con Steam Play, Valve está apostando seriamente por hacer compatibles la mayoría de los juegos de Windows en Linux. Steam Play es, a grandes rasgos, una configuración recomendada y óptima

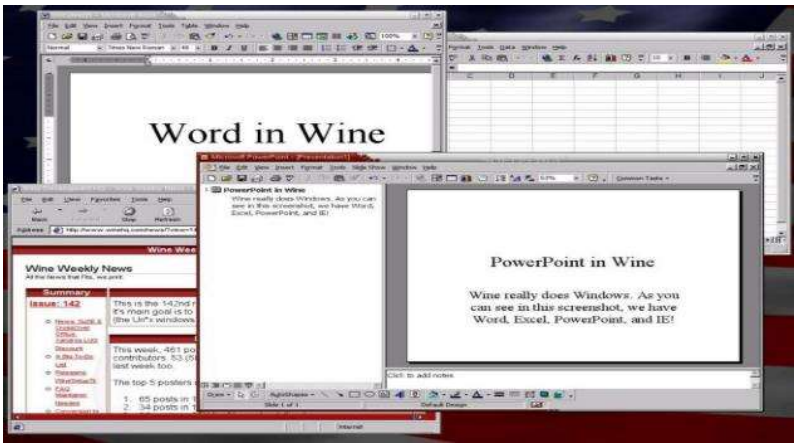
para poder ejecutar juegos concretos de Windows en Linux a través de Wine. Por ello, las principales novedades de esta nueva versión se han centrado principalmente en mejorar esta función.

El nuevo Wine 4.0 ha llegado por todo lo grande trayendo consigo soporte para Direct3D 12 y para la API de Vulkan, así como compatibilidad con las pantallas del tipo HiDPI en caso de ejecutar esta herramienta en Android.

Sin embargo, estos no son los únicos cambios que han llegado con Wine 4.0, y es que esta nueva versión llega con más de 6000 mejoras a nivel interno, mejoras para optimizar el funcionamiento general de las aplicaciones, mejoras al ejecutar contenido Direct3D 10 y 11 y optimizaciones para leer y descodificar cierto tipo de contenido, como MP3 o imágenes PNG. También han llegado una gran cantidad de mejoras y optimizaciones relacionadas con la integración de Wine con los escritorios de los sistemas Linux, criptografía y funciones de red.” (Velasco, https://www.softzone.es/, 2019)

Figura 15

Funcionamiento de Wine



Nota. Esta imagen muestra el funcionamiento de Wine. (Velasco, Wine 4.0, 2019)

2.2.4. LONGENE

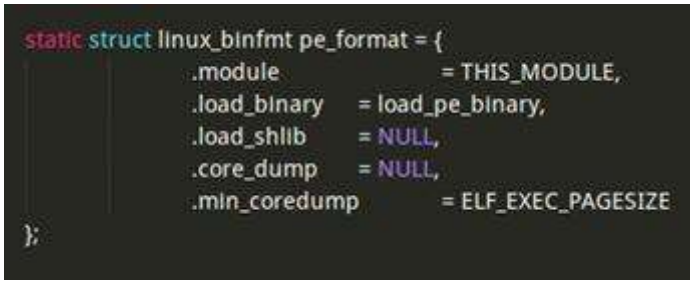
Para esta referencia bibliográfica a pesar de obtenerla en el idioma disponible, parece que del foro del que la obtuve contaba con una traducción automática basada en localización, por este motivo procedo a parafrasear esta base teórica para mantener la estética y un buen uso del lenguaje.

“Longene es un proyecto de sistema operativo autónomo, su propósito es el de expandir el kernel de Linux de una sola compatibilidad con las aplicaciones de Linux a que admita los controladores de dispositivos Linux y admita las unidades de dispositivo de Windows; de esta manera los usuarios pueden ejecutar de manera eficiente las aplicaciones de Windows directamente en el sistema operativo Linux. En 2006, se lanzó su primera versión, pero desde 2014, el desarrollo del proyecto se ha detenido. En mayo de 2018, su sitio web oficial dejo de ser accesible. No hay mucha introducción adicional aquí, a partir del análisis del ángulo de código fuente de Longene, de cómo implementar el cargador de PE en el núcleo de Linux.

La carga del Ejecutable PE en Longene y el proceso de enlaces dinámicos se implementan principalmente en la carpeta /module/ke/. LONGENE utiliza la forma de los módulos de kernel, registra archivos en formato de PE en el kernel. (Anteriormente se analizó cómo extender el formato de archivo soportable, la estructura es PE_Format.

Figura 16

pe_format



Nota. Esta imagen muestra el código del módulo pe_format. (programmerclick, 2020)

Cuando se ejecuta un archivo EXE, por ejemplo, cuando se ajusta ./Notepad.exe En la línea de comandos, el primer kernel de Linux llama a la llamada del sistema EXECVE, finalmente, llega a la función Load_PE_BianRy del módulo de Carga Compatible (esto también se analiza anteriormente).

Primero se va a la función Load_pe_binary, primero se determina si el proceso principal es un programa Win32.

Figura 17

Comprobar win32

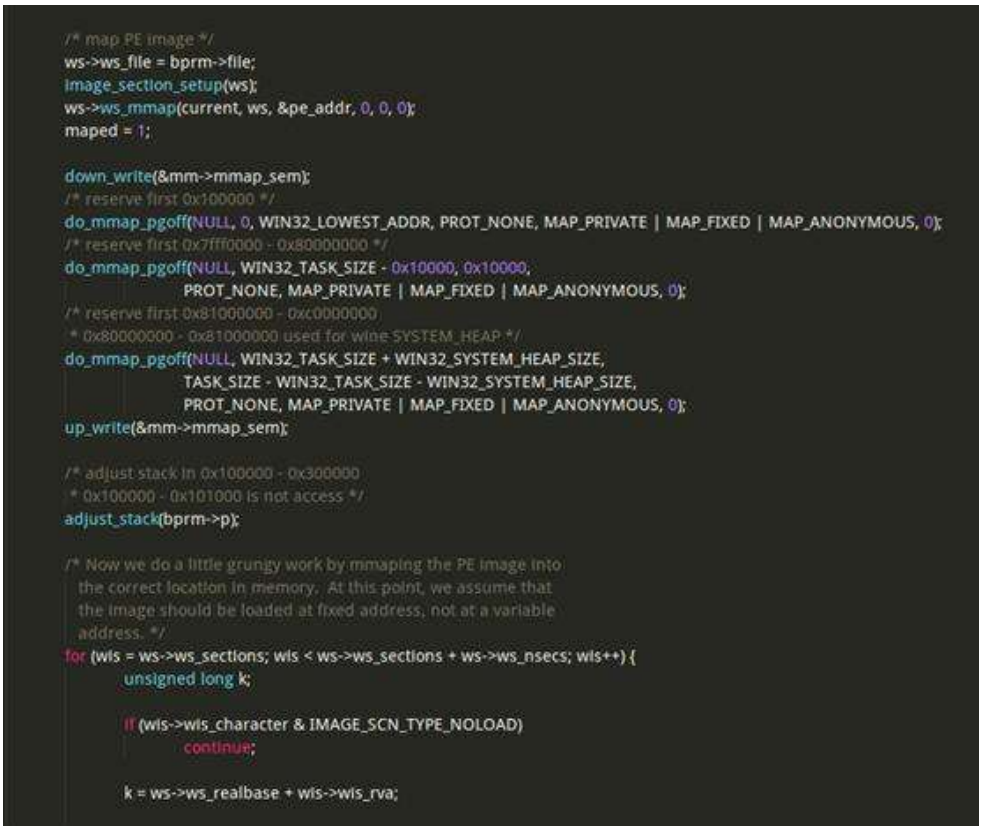
```
if(current->parent->ethread)
{
        is_win32 = TRUE;
        parent_ethread = current->parent->ethread;
        parent_eprocess = parent_ethread->threads_process;
}
```

Nota. Esta imagen muestra la comprobación de si el aplicativo es win32. (programmerclick, 2020)

Lo siguiente es un montón de operaciones regulares, como el espacio de la dirección de proceso reservado, los archivos binarios de mapas a la memoria, ajuste del tamaño de la pila y el espacio de reserva para el intérprete.

Figura 18

Mapeo y Ajuste en Espacios de memoria



```
/* map PE image */
ws->ws_file = bprm->file;
image_section_setup(ws);
ws->ws_mmap(current, ws, &pe_addr, 0, 0, 0);
maped = 1;

down_write(&mm->mmap_sem);
/* reserve first 0x100000 */
do_mmap_pgoff(NULL, 0, WIN32_LOWEST_ADDR, PROT_NONE, MAP_PRIVATE | MAP_FIXED | MAP_ANONYMOUS, 0);
/* reserve first 0x7fff0000 - 0x80000000 */
do_mmap_pgoff(NULL, WIN32_TASK_SIZE - 0x10000, 0x10000,
              PROT_NONE, MAP_PRIVATE | MAP_FIXED | MAP_ANONYMOUS, 0);
/* reserve first 0x81000000 - 0xc0000000
 * 0x80000000 - 0x81000000 used for wine SYSTEM_HEAP */
do_mmap_pgoff(NULL, WIN32_TASK_SIZE + WIN32_SYSTEM_HEAP_SIZE,
              TASK_SIZE - WIN32_TASK_SIZE - WIN32_SYSTEM_HEAP_SIZE,
              PROT_NONE, MAP_PRIVATE | MAP_FIXED | MAP_ANONYMOUS, 0);
up_write(&mm->mmap_sem);

/* adjust stack in 0x100000 - 0x300000
 * 0x100000 - 0x101000 is not access */
adjust_stack(bprm->p);

/* Now we do a little grungy work by mmaping the PE image into
   the correct location in memory.  At this point, we assume that
   the image should be loaded at fixed address, not at a variable
   address. */
for (wis = ws->ws_sections; wis < ws->ws_sections + ws->ws_nsecs; wis++) {
        unsigned long k;

        if (wis->wis_character & IMAGE_SCN_TYPE_NOLOAD)
                continue;

        k = ws->ws_realbase + wis->wis_rva;
```

Nota. Esta imagen muestra el Mapeo y ajuste en espacios de memoria del código de Longene. (programmerclick, 2020)

Lo siguiente es un paso importante para buscar NTDLL.DLL.SO, que es el DLL incorporado generado por el código de WINE. Buscar ntdll.dll.so en $PATH, el valor predeterminado es /usr/local/lib/wine/ntdll.dll.so. NTDLL.DLL contiene todas las posibles funciones que están relacionadas con NT kernel. Puede hacer el siguiente código después de que lo encuentre.

Figura 19

Cargado predefinido de NTDLL.DLL

Nota. Esta imagen muestra el cargado del archivo Ntdll.dll. (programmerclick, 2020)

La función Search_ntdll se muestra en la siguiente figura.

Figura 20

Buscar el archivo NTDLL.DLL

Nota. Esta imagen muestra la búsqueda del archivo Ntdll.dll y la copia del mismo. (programmerclick, 2020)

Llamar a la función Map_System_dll Mapeo ntdll.dll.so. Esta función se define en SYSDLL.C, primera carga NTDLL.DLL.SO, esta parte de la función de la función es básicamente la parte de la parte del archivo ELF cargado en la fuente del kernel de Linux (binfmt_elf.c)

Figura 21

Archivo ELF cargado en la fuente del kernel de Linux

```
eppnt = elf_phdata;
for (i=0; i<interp_elf_ex->e_phnum; i++, eppnt++) {
        if (eppnt->p_type == PT_INTERP && id_name)
                kernel_read(interpreter, eppnt->p_offset, id_name, eppnt->p_filesz);

        if (eppnt->p_type == PT_LOAD) {
                int elf_type = MAP_PRIVATE | MAP_DENYWRITE;
                int elf_prot = 0;
                unsigned long vaddr = 0;
                unsigned long k, map_addr;

                if (eppnt->p_flags & PF_R) elf_prot =  PROT_READ | PROT_WRITE;
                if (eppnt->p_flags & PF_W) elf_prot |= PROT_WRITE;
                if (eppnt->p_flags & PF_X) elf_prot |= PROT_EXEC;
                vaddr = eppnt->p_vaddr;
                if (interp_elf_ex->e_type == ET_EXEC || load_addr_set)
                        elf_type |= MAP_FIXED;

                map_addr = elf_map(tsk, interpreter, load_addr + vaddr, eppnt, elf_prot, elf_type);
                error = map_addr;
                if (map_addr > (unsigned long)TASK_SIZE)
                        goto out_close;

                if (!load_addr_set && interp_elf_ex->e_type == ET_DYN) {
                        load_addr = map_addr - ELF_PAGESTART(vaddr);
                        *interp_load_addr = map_addr - eppnt->p_offset;
                        load_addr_set = 1;
                }
```

Nota. Esta imagen muestra el cargado de la fuente del kernel Linux.(programmerclick, 2020)

Luego regresar a la función MAP_system_dll, después de cargar ntdll.dll.so, llamando a la función UK_FIND_SYMBOL varias veces para obtener la dirección de entrada de cada función en ntdll.dll.so, donde NTDLL_Entry se establece en la dirección de la función LDRIntilalizetethunk, el Función de la función previamente, se utilizó para cargar la DLL para el ejecutable PE; establecer PE_Entry a la dirección de la función ProcessStarTarward, que es una función de reenvío, que es equivalente a llamar a la función BaseProcessStartStart en kernel32.dll.so.

Figura 22

Cargado de sección header de NTDLL.DLL

```
if (tsk == current) {
        int elf_shnum;
        int elf_shsize;
        struct elf_shdr *elf_shdata = NULL;

        /* section header is not mapped to memory, need read it */
        /* load section header list for ntdll */
        elf_shnum = ntdll_elf_ex.e_shnum;
        elf_shsize = elf_shnum * ntdll_elf_ex.e_shentsize;
        elf_shdata = (struct elf_shdr *)kmalloc(elf_shsize, GFP_KERNEL);
        if (!elf_shdata) {
                retval = -ENOMEM;
                goto out_free_ntdll;
        }

        retval = kernel_read(ntdll, ntdll_elf_ex.e_shoff, (void *)elf_shdata, elf_shsize);
        if (retval != elf_shsize) {
                if (retval >= 0)
                        retval = -EIO;
                kfree(elf_shdata);
                goto out_free_ntdll;
        }

        /* LdrInitializeThunk is used to load dll for PE exe file */
        ntdll_entry = uk_find_symbol(elf_shdata, elf_shnum, "LdrInitializeThunk");
        /* when interpreter done, jump to StartThunk */
        start_thunk = uk_find_symbol(elf_shdata, elf_shnum, "StartThunk");
        /* KiUserApcDispatcher is APC Dispatcher */
        apc_dispatcher = uk_find_symbol(elf_shdata, elf_shnum, "KiUserApcDispatcher");
        /* a forward function , will call BaseProcessStart in kernel32.dll.so */
        pe_entry = uk_find_symbol(elf_shdata, elf_shnum, "ProcessStartForward");
        thread_entry = uk_find_symbol(elf_shdata, elf_shnum, "start_thread");
#ifdef EXE_SO
        ntdll_start_thunk = uk_find_symbol(elf_shdata, elf_shnum, "ntdll_start_thunk");
        exeso_start_thunk = uk_find_symbol(elf_shdata, elf_shnum, "exeso_start_thunk");
#endif
```

Nota. Esta imagen muestra el Cargado de sección header del archivo NTDLL.DLL (programmerclick, 2020)

Volver a la función MAP_system_dll para continuar. Llamar a load_elf_interp de nuevo, esta vez es para cargar el intérprete es ld-linux.so, y asignar la dirección a interp_entry.

Figura 23

Asignación de dirección de interprete

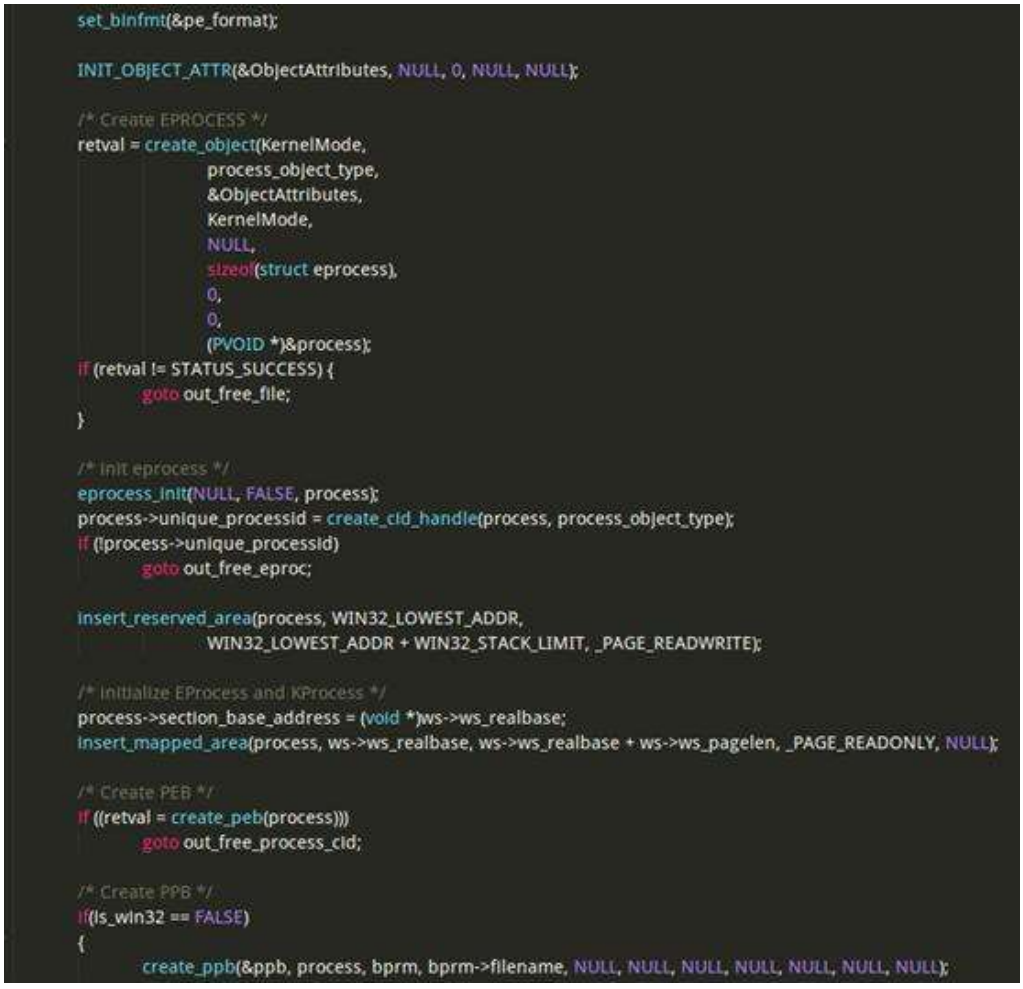
```
/* Get the exec headers */
interp_elf_ex = *((struct elfhdr *)buf);
interp_entry = load_elf_interp(tsk, &interp_elf_ex, interpreter, interp_load_addr, NULL);

allow_write_access(interpreter);
fput(interpreter);
```

Nota. Esta imagen muestra la asignación de la dirección del interprete en Longene. (programmerclick, 2020)

Aquí, se ejecuta Map_system_dll, luego vuelve a la función Load_Pe_Binary. Se crea una estructura de eProcess, inicializar el eProcess y KProcess, crear PEB, PPB, etc. Establecer TEB, inicializar Kthread, etc.

Figura 24

Inicializar Kthread



```
set_binfmt(&pe_format);

INIT_OBJECT_ATTR(&ObjectAttributes, NULL, 0, NULL, NULL);

/* Create EPROCESS */
retval = create_object(KernelMode,
                       process_object_type,
                       &ObjectAttributes,
                       KernelMode,
                       NULL,
                       sizeof(struct eprocess),
                       0,
                       0,
                       (PVOID *)&process);
if (retval != STATUS_SUCCESS) {
        goto out_free_file;
}

/* init eprocess */
eprocess_init(NULL, FALSE, process);
process->unique_processid = create_cid_handle(process, process_object_type);
if (!process->unique_processid)
        goto out_free_eproc;

insert_reserved_area(process, WIN32_LOWEST_ADDR,
                     WIN32_LOWEST_ADDR + WIN32_STACK_LIMIT, _PAGE_READWRITE);

/* initialize EProcess and KProcess */
process->section_base_address = (void *)ws->ws_realbase;
insert_mapped_area(process, ws->ws_realbase, ws->ws_realbase + ws->ws_pagelen, _PAGE_READONLY, NULL);

/* Create PEB */
if ((retval = create_peb(process)))
        goto out_free_process_cid;

/* Create PPB */
if(is_win32 == FALSE)
{
        create_ppb(&ppb, process, bprm, bprm->filename, NULL, NULL, NULL, NULL, NULL, NULL, NULL);
```

Nota. Esta imagen muestra la inicialización del módulo de Kthread. (programmerclick, 2020)

Si el proceso principal es un programa Win32, es IS_WIN32 para verdadero (antes de que se juzgue), primero se espera a que se complete el proceso principal.

Figura 25

IS_WIN32

```
sema_Init(&thread->exec_semaphore,0);
if (is_win32 == TRUE) //parent is a windows process
{
        down(&thread->exec_semaphore);  //wait for the parent

        child_w32process = process->win32process;
        parent_w32process = parent_eprocess->win32process;
        info = child_w32process->startup_info;

        //now parent has finished its work
        if(thread->inherit_all)
        {
                create_handle_table(parent_eprocess, TRUE, process);
                child_w32process = create_w32process(parent_w32process, TRUE, process);
        }
}
```

Nota. Esta imagen muestra el módulo de si el programa es Win32, se crea una tabla de proceso hijo. (programmerclick, 2020)

Entonces es un paso clave. Realice la función LdrinitializeThunk () en ntdll.dll.so como una función APC para completar la conexión de carga y dinámica de la imagen DLL Format DLL.

LONGENE ha realizado el mecanismo de WindowsPC en el kernel de Linux. El APC es una abreviatura de "Llamada de AsyncronusProcedure", que se implementa en MODULE/KE/APC.C, y ve, aquí la implementación se refiere al Código ReactOS. Simplemente, al usar el mecanismo de señal en el sistema Linux, puede implementar la llamada de proceso asíncrona (APC) durante el proceso de creación de procesos, de modo que el mecanismo APC pueda analógico el proceso de carga y conectando dinámicamente WONDONDLL, lo que lo prepara para el proceso. La implementación específica del mecanismo APC no se analiza aquí.

Luego de mirar el código en este punto. Primero se llama al APC_init para inicializar el APC, transmitir NTDLL_ENTRY como un parámetro, es decir, la función Ldrinitializethunk (analizada previamente, NTDLL_Entry asigna la dirección de la función) como función de procesamiento de APC. Luego, llame a INSERT_QUEUE_APC para unirse a la solicitud APC a la cola APC y pase el Interp_Entry como un parámetro.

Figura 26

Iniciar APC para llamar a LdrInitializeThunk

```
/* init apc, to call LdrInitializeThunk */
thread_apc = kmalloc(sizeof(KAPC), GFP_KERNEL);
if (!thread_apc) {
        retval = -ENOMEM;
        goto out_free_thread_cid;
}
apc_init(thread_apc,
                &thread->tcb,
                OriginalApcEnvironment,
                thread_special_apc,
                NULL,
                (PKNORMAL_ROUTINE)ntdll_entry,
                UserMode,
                (void *)(bprm->p + 12));
insert_queue_apc(thread_apc, (void *)interp_entry, (void *)extra_page, IO_NO_INCREMENT);
set_tsk_thread_flag(current, TIF_APC);
```

Nota. Esta imagen muestra como iniciar el APC para la llamada al modulo LdrInitializeThunk. (programmerclick, 2020)

Finalmente, llamar a Start_Thread y regresar al espacio de usuario desde la llamada del sistema. Por supuesto, el proceso realmente ha comenzado a ejecutar, porque el mecanismo APC, el código de la aplicación comienza a ejecutarse después de completar el enlace dinámico.

Figura 27

Llamada a Start_Thread

```
start_thread(regs, pe_entry, bprm->p);
if (unlikely(current->ptrace & PT_PTRACED)) {
        if (current->ptrace & PT_TRACE_EXEC)
                ptrace_notify ((PTRACE_EVENT_EXEC << 8) | SIGTRAP);
        else
                send_sig(SIGTRAP, current, 0);
}

/* save current trap frame */
thread->tcb.trap_frame = (struct ktrap_frame *)regs;
retval = 0;

try_module_get(THIS_MODULE);
/* return from w32syscall_exit, not syscall_exit */
((unsigned long *)regs)[-1] = (unsigned long)w32syscall_exit;
regs->fs = TEB_SELECTOR;
```

Nota. Esta imagen muestra cómo se realiza la llamada para iniciar el strat_thread. (programmerclick, 2020)

Cuando el sistema EXECVE llame a la devolución, volverá de win32syscall_exit. Este es un código de compilación, en el module/ke/win32entry.s.

Figura 28

Volver a win32syscall_exit

```
ENTRY(w32syscall_exit)
        LOCKDEP_SYS_EXIT
        DISABLE_INTERRUPTS(CLBR_ANY)       # make sure we don't miss an interrupt
                                           # setting need_resched or sigpending
                                           # between sampling and the iret
        TRACE_IRQS_OFF
        movl TI_flags(%ebp), %ecx
        testl $_TIF_ALLWORK_MASK, %ecx     # current->work
        jne w32syscall_exit_work
```

Nota. Esta imagen muestra el módulo win32syscall_exit. (programmerclick, 2020)

Funciones assembly de w32syscall_trace_entry y salida del proceso denominada w32syscall_exit_work.

Figura 29

Llamada de ruta w32syscall_trace_entry

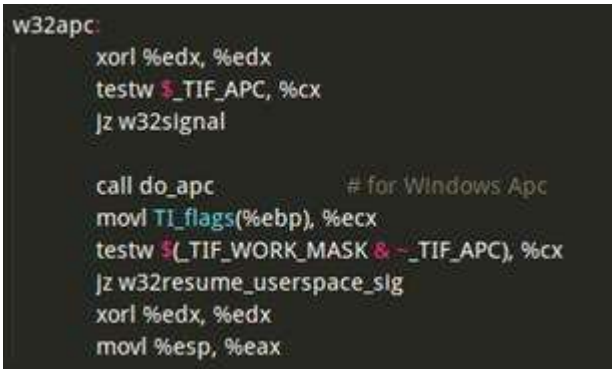
```
w32syscall_trace_entry:
        movl $-ENOSYS,PT_EAX(%esp)
        movl %esp, %eax
        call syscall_trace_enter

        /* back to w32syscall_call */
        movl %eax, %edi
        shrl $8, %edi
        andl $0x10, %edi

        addl $(KeServiceDescriptorTable), %edi

        andl $0x0FFF, %eax
        cmpl 8(%edi), %eax
        jb w32syscall_call

        jmp w32syscall_exit

        # perform syscall exit tracing
        ALIGN
w32syscall_exit_work:
        testl $_TIF_WORK_SYSCALL_EXIT, %ecx
        jz w32work_pending
        TRACE_IRQS_ON
        ENABLE_INTERRUPTS(CLBR_ANY)        # could let syscall_trace_leave() call
                                           # schedule() instead
        movl %esp, %eax
        call syscall_trace_leave
        jmp w32resume_userspace
END(w32syscall_exit_work)
        CFI_ENDPROC
```

Nota. Esta imagen muestra el módulo win32syscall_trace_entry. (programmerclick, 2020)

No hay jerarquía aquí, porque todo el assembly no se introduce específicamente, solo conozca algunos saltos, eventualmente ejecute este código de ensamblaje de Win32APC y llamar a la función DO_APC, como se muestra a continuación.

Figura 30

Código de ensamblaje Win32APC



Nota. Esta imagen muestra el módulo w32apc el cual esta en assembly. (programmerclick, 2020)

Es decir, en realidad se llama do_apc cuando se devuelve la llamada del sistema EXECVE, que está en Module/KE/APC.C.

Figura 31

Procedimiento do_apc

Nota. Esta imagen muestra el procedimiento llamado do_apc. (programmerclick, 2020)

La función DO_APC llama la función de entrega_apc, la función de esta función es eliminar el controlador APC en la cola APC y llama uno por uno. En primer lugar, se procesa el APC del kernel, y luego se procesa el APC del espacio de usuario.

Por ejemplo, el ldrinitializethunk () antes, como una función de APC de usuario agregada a la cola.

Figura 32

Primer proceso de APC

```
/* Do the Kernel APCs first */
while (!list_empty(&thread->tcb.apc_state.apc_list_head[KernelMode])) {
        /* Clear APC Pending */
        thread->tcb.apc_state.kapc_pending = 0;

        /* Get the next Entry */
        apc_listentry = thread->tcb.apc_state.apc_list_head[KernelMode].next;
        apc = list_entry(apc_listentry, struct kapc, apc_list_entry);

        /* Save Parameters so that it's safe to free the Object in Kernel Routine*/
        normal_routine = apc->normal_routine;
        kernel_routine = apc->kernel_routine;
        normal_context = apc->normal_context;
        system_argument1 = apc->system_argument1;
        system_argument2 = apc->system_argument2;

        if (!normal_routine) {
                /* Remove the APC from the list */
                apc->Inserted= 0;
                list_del(apc_listentry);

                /* Enable interrupt and unlock shared resource */
                spin_unlock_irq(&thread->tcb.apc_queue_lock);

                /* Call the Special APC */
                kernel_routine(apc,
                               &normal_routine,
                               &normal_context,
                               &system_argument1,
                               &system_argument2);

                /* Disable interrupt and lock shared resource again */
                spin_lock_irq(&thread->tcb.apc_queue_lock);
        } else {
```

Nota. Esta imagen muestra el proceso de APC. *(programmerclick, 2020)*

Luego manejar el espacio del usuario APC.

Figura 33

Manejo del espacio de usuario APC

```
/* Now we do the User APCs */
if ((!list_empty(&thread->tcb.apc_state.apc_list_head[UserMode])) &&
                (thread->tcb.apc_state.uapc_pending)) {                 /* && (DeliveryMode == UserMode) */

        /* Get the APC Object */
        apc_listentry = thread->tcb.apc_state.apc_list_head[UserMode].next;
        apc = list_entry(apc_listentry, struct kapc, apc_list_entry);

        /* Save Parameters so that it's safe to free the Object in Kernel Routine*/
        normal_routine = apc->normal_routine;
        kernel_routine = apc->kernel_routine;
        normal_context = apc->normal_context;
        system_argument1 = apc->system_argument1;
        system_argument2 = apc->system_argument2;

        /* Remove the APC from Queue, call the APC */
        list_del(apc_listentry);
        apc->inserted = 0;

        /* Enable interrupt and unlock shared resource */
        spin_unlock_irq(&thread->tcb.apc_queue_lock);

        kernel_routine(apc,
                       &normal_routine,
                       &normal_context,
                       &system_argument1,
                       &system_argument2);

        if (!normal_routine) {
                test_alert_thread(UserMode);  /* Unimplemented */
        } else {
                /* Set up the Trap Frame and prepare for Execution in NTDLL.DLL */
                init_user_apc(Reserved,
                              TrapFrame,
                              normal_routine,
                              normal_context,
                              system_argument1,
                              system_argument2);
```

Nota. Esta imagen muestra el proceso manejo de espacio de usuario en el APC. (programmerclick, 2020)

El APC que maneja el espacio de usuario primero se llama la función init_user_apc.

Figura 34

Manejo de espacio init_user_apc

```
VOID
STDCALL
init_user_apc(IN PVOID Reserved,
              IN PKTRAP_FRAME TrapFrame,
              IN PKNORMAL_ROUTINE NormalRoutine,
              IN PVOID NormalContext,
              IN PVOID SystemArgument1,
              IN PVOID SystemArgument2)
{
        PContext context;
        PULONG esp;

        ktrace("ESP 0x%lx\n", TrapFrame->sp);
        /*
         * Save the thread's current context (in other words the registers
         * that will be restored when it returns to user mode) so the
         * APC dispatcher can restore them later
         */
        context = (PContext)(((PUCHAR)TrapFrame->sp) - sizeof(*context));
        memcpy(context, TrapFrame, sizeof(*context));

        /*
         * Setup the trap frame so the thread will start executing at the
         * APC Dispatcher when it returns to user-mode
         */
        esp = (PULONG)(((PUCHAR)TrapFrame->sp) - (sizeof(CONTEXT) + (6 * sizeof(ULONG))));
        esp[0] = 0xdeadbeef;
        esp[1] = (ULONG)NormalRoutine;
        esp[2] = (ULONG)NormalContext;
        esp[3] = (ULONG)SystemArgument1;
        esp[4] = (ULONG)SystemArgument2;
        esp[5] = (ULONG)context;
        TrapFrame->Ip = get_apc_dispatcher();
        TrapFrame->sp = (ULONG)esp;
} /* end init_user_apc */
```

Nota. Esta imagen muestra el manejo de espacio de usuario mediante función init_user_apc (programmerclick, 2020)

Esta función se diagnostica en la segunda línea de TrapFrame de código-> IP = get_apc_dispatcher ();

El rol es que al regresar al espacio de usuario, la primera función que se ejecuta por primera vez será KiUSerApcDispatcher, y cada APC llamará a esta función. Esta función está en WINE /dlls/ntdll/init.c.

Es necesario prestar especial atención a los parámetros de esta función, que se establecen en load_pe_binary. Entre ellos, ApcRoutine se establece en NTDLL en NTDLL; APCContext es el puntero de la pila de procesos de Linux, que no está incluida para PE y APC; iOSB es la dirección de entrada del intérprete LIBLD-linux.SO; reservado es el primero de un inactivo La dirección de

memoria de la página, esta memoria inactiva se conserva como un contenido de pila que PE y APC se adhieren a PE y APC; el contexto es nulo.

Figura 35

KiUserApcDispatcher

```
void __attribute__((stdcall, no_instrument_function))
KiUserApcDispatcher(PIO_APC_ROUTINE ApcRoutine, void *ApcContext,
    void *Iosb, unsigned long Reserved, void *Context)
{
  if (Reserved) {
    extra_page = Reserved;
    StartInterp(ApcRoutine, ApcContext, Iosb, Reserved, Context);
  }
  ApcRoutine(ApcContext, Iosb, Reserved);
  /* switch back to the interrupted context */
  NtContinue((PCONTEXT)Context, 1);
}
```

Nota. Esta imagen muestra la Llamada a la función StartInterP. (programmerclick, 2020)

La función StartInterp también está en init.c, que consiste en un código de ensamblaje. El papel principal es ingresar al intérprete de Linux, es decir, ld-linux.so.

Figura 36

StartInterp

```
__asm__ (
    ".globl StartInterp\n"
    "StartInterp:\n\t"
    "pusha\n\t"
    "mov 0x28(%esp), %ecx\n\t"        /* stack top used for linux arg */
    "sub %esp, %ecx\n\t"              /* stack size need backup */
    "mov %esp, %esi\n\t"
    "mov 0x30(%esp), %edi\n\t"
    "mov %ecx, (%edi)\n\t"            /* backup the size */
    "add $0x4, %edi\n\t"
    "shr $2, %ecx\n\t"
    "rep movsl\n\t"
    "mov 0x28(%esp), %ecx\n\t"
    "mov 0x2c(%esp), %esi\n\t"        /* Iosb, here in interpreter */
    "mov %ecx, %esp\n\t"
    "jmp *%esi\n"                             /* _start in interpreter */
    /* finally jmp to AT_ENTRY */
```

Nota. Esta imagen muestra la funcion StartInterp en init.c. (programmerclick, 2020)

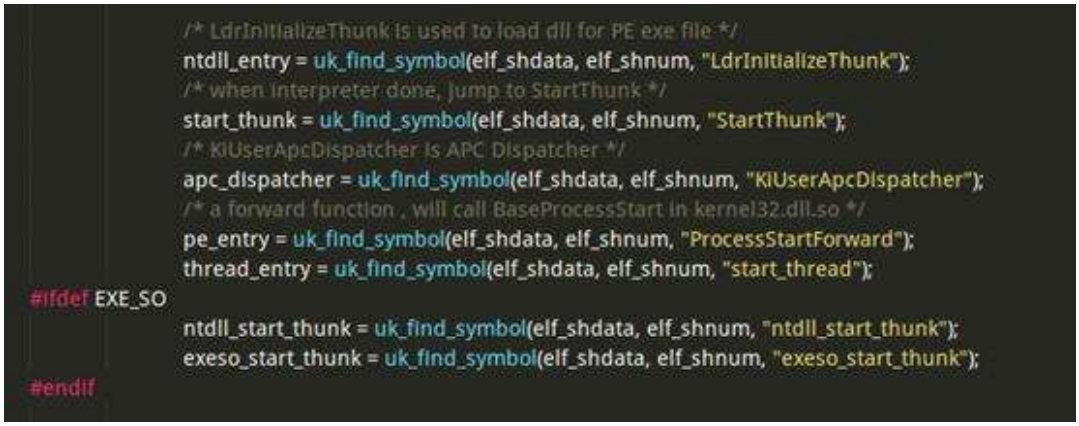
"MOV 0x2C (% ESP),% ESI \ N \ T" Esta oración establece ESI a la entrada a la interpretación /lib/ld-linux.SO.

Después de ingresar al InterPreter LD-linux.SO, el intérprete ELF completa la conexión de la conexión NTDLL de enlace, buscará el valor del elemento AT_Entry, y su dirección de destino de salto es AT_Entry, que se establece en la dirección de funciones de inicio Thunk en load_pe_binary, por lo que Se ejecutará para empezar a trabajar.

Se establece AT_Entry después del camino para buscar NTDLL.DLL.SO, se llamará la función MAP_SYSTEM_DLL, que en SYSDLL.C, asigna la dirección de la función START THENK a START_THUNK, luego cree_f_tables grabados.

Figura 37

AT_Entry



Nota. Esta imagen muestra a AT_Entry para busequeda (programmerclick, 2020)

Create_f_tables El último parámetro es la entrada, asigna la entrada a ATRY en esta función.

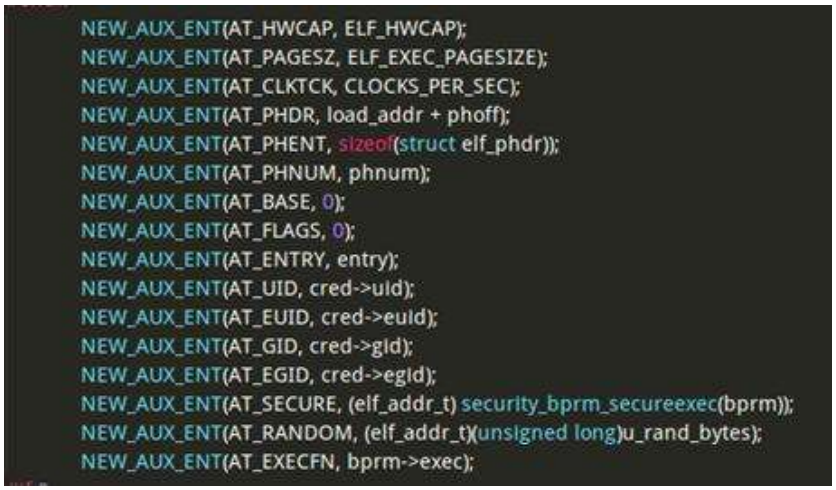
Figura 38

Creacíon de elf_tables

Nota. Esta imagen muestra la creacíon de elf_tables dentro de NTDLL_SO. (programmerclick, 2020)

Figura 39

Datos Auxiliares de entrada



Nota. Esta imagen muestra las llamadas a datos Auxiliares de entrada. (programmerclick, 2020)

La función StartThunk también está compuesta de ensamblaje, como se muestra a continuación.

Figura 40

Función StartThunk

Nota. Esta imagen muestra la función StartThunk.(programmerclick, 2020)

Esta función se refiere principalmente a la función _start en el LIBC, el propósito principal es llamar a la preparación para prepararse para la función de APC verdadero LdrIntializethunk. Después de eso, debe recuperar la pila preparada para PE y APC, de lo contrario, no podrá regresar de la llamada APC, no puede ingresar la entrada del archivo PE. Dado que POPA restaura el entorno en el que se presiona PUSTA en StartInter, el último RET volverá a KiUSerApcDispatcher, y luego el LDRIntilalizetethunk ingresará el enlace del archivo PE. La función preparada se muestra en la siguiente figura.

Figura 41

Registro de wine_dll y la llamada a ntdll.dll

```
LOG(LOG_FILE, 0, 0, "PrepareThunk(), init=%p\n", init);
if (__builtin_expect (rtld_fini != NULL, 1))
  __cxa_atexit ((void (*) (void *)) rtld_fini, NULL, NULL);

p = evp = argv + argc + 1;

while (*p++);
auxvec = (ElfW(auxv_t) *)p;
auxvec_len = (unsigned long)argv[0] - (unsigned long)p;

bin_dir = get_wine_bindir();
wine_path = malloc(strlen(bin_dir) + sizeof(wine));
strcpy(wine_path, bin_dir);
strcat(wine_path, wine);
free(bin_dir);

wine_init_argv0_path(wine_path);
build_dll_path();
__wine_main_argc = argc;
__wine_main_argv = argv;
__wine_main_environ = evp;
free(wine_path);

init_for_load();

/* Call the initializer of the program, if any.  */
if (init)
    {
  (*init)();
    }

/* _init in ntdll.dll.so */
_init();  //__wine_dll_register() is called here for ntdll.dll
NtCurrentTeb()->Peb->ProcessParameters->Environment = NULL;
return extra_page;
```

Nota. Esta imagen muestra la función preparada para iniciar el registro de wine_dll y la llamada a ntdll.dll.(programmerclick, 2020)

Después de la ejecución, vuelva a la función KiUSerApCDispatcher para comenzar a ejecutar LDRIntializethunk, donde no se analiza este código de función. Debido a que el Código de WINE analizado previamente, el código aquí se implementa en esta función en Wine.

El paso final de la función KiUSerAPCDispatcher es llamar a la función NTContinue.

Figura 42

Función NtContinue

```
NTSTATUS
SERVICECALL
NtContinue(IN PContext Context,
           IN BOOLEAN TestAlert)
{
    PKTRAP_FRAME trap_frame = (PKTRAP_FRAME)current->ethread->tcb.trap_frame;

    ktrace("\n");
    /*
     * Copy the supplied context over the register information that was saved
     * on entry to kernel mode, it will then be restored on exit
     * FIXME: Validate the context
     */
    memcpy(trap_frame, Context, sizeof(*trap_frame));

    /* FIXME
     * Copy floating point context into the thread's FX_SAVE_AREA
     */

    __asm__(
                "andl %%esp, %%ecx;\n\t"
                "movl %%ecx, %%ebp;\n\t"
                "movl %%ebx, %%esp;\n\t"
                "jmp w32syscall_exit\n\t"
                :
                : "b" (trap_frame), "c" (-THREAD_SIZE));

    /* This doesn't actually happen b/c KeRosTrapReturn() won't return */
    return STATUS_SUCCESS;
} /* NtContinue */
EXPORT_SYMBOL(NtContinue);
```

Nota. Esta imagen muestra la función NtContinue (programmerclick, 2020)

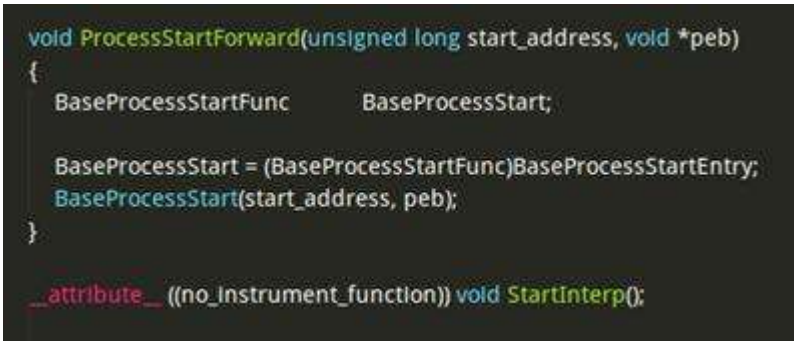
La función es restaurar el contexto de PE y volver al espacio del kernel. Si no hay otra función de APC que se procesará, W32SSYSYSIPL_EXIT comenzará a ejecutar START_THEAD (REGS, PE_ENTRY, BPRM-> P)

Anteriormente, se mencionó que PE_Entry se estableció en la dirección de la función ProcessStartPorward, que en

Wine/DLS/NTDLL/init.c es una función de reenvío, llamando a la función BaseProcessStart.

Figura 43

Función BaseProcessStart

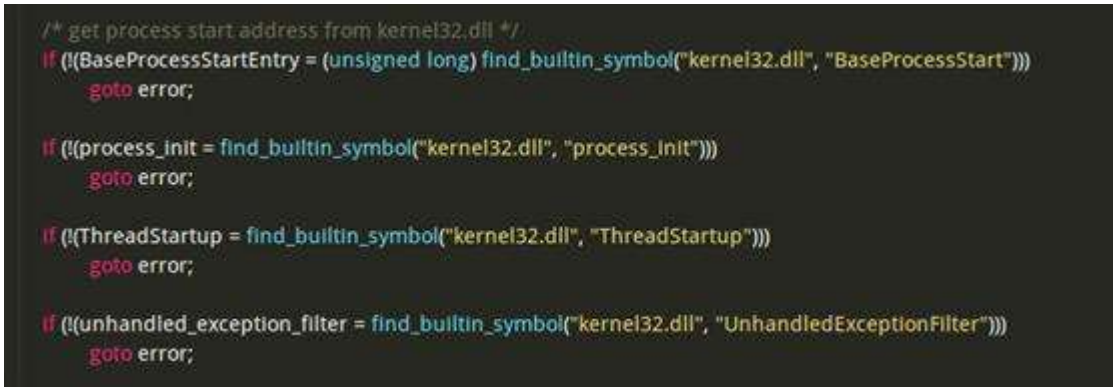


Nota. Esta imagen muestra función ProcessStartForward. (programmerclick, 2020)

La función BaseProcessStart obtiene la dirección de la función a través de BaseProcessStarry, y se asigna de base en LdrIntializethunk. Como se mencionó anteriormente, el mecanismo de APC llama a la función LdrInitializethunk, que llama la función Find_Builtin_Symbol varias veces en esta función para obtener la dirección de la función BaseProcessStart en Kernel32.dll.

Figura 44

Funcion BaseProcessStart



Nota. Esta imagen muestra la Funcion BaseProcessStart para obtener la dirección de Kernel32.dll (programmerclick, 2020)

Ir a la función BaseProcessStart, ingresar la entrada de PE a través de la entrada, luego se inicia oficialmente el archivo PE.

Figura 45

Ingreso de entrada PE

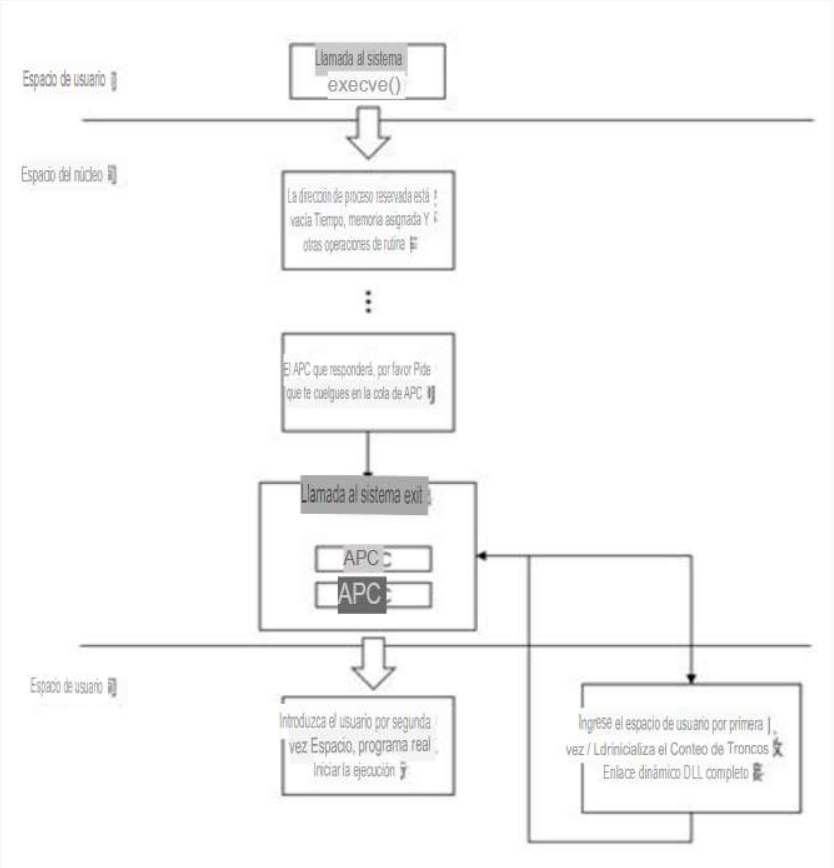
```
void BaseProcessStart(unsigned long start_address, void *param)
{
  unsigned long  exit_code;
  LPTHREAD_START_ROUTINE  entry;

  SERVER_START_REQ( new_thread )
  {
                int ret=0;
                ret = wine_server_call_err( req );
                if(ret)
                {
                        fprintf(stderr,"ERROR:new_thread\n");
                        return;
                }
  }
  SERVER_END_REQ;

  {
    PEB *peb = NtCurrentTeb()->Peb;
    LOG(LOG_FILE,0,0, "%04x:Starting process %s \n",\
            GetCurrentThreadId(), debugstr_w(peb->ProcessParameters->ImagePathName.Buffer));
  }

  __TRY
  {
    entry = (LPTHREAD_START_ROUTINE)start_address;
    exit_code = entry(param);
  }
  __EXCEPT(UnhandledExceptionFilter)
  {
    exit_code = GetExceptionCode();
  }
  __ENDTRY;

  ExitProcess(exit_code);
}
```

Nota. En esta imagen se muestra como el PE ingresa en BaseProcessStart. (programmerclick, 2020)

Con el análisis del par de código anterior, veremos el proceso de ejecución de todo el archivo PE desde una perspectiva macro, como se muestra en la figura 46.” (programmerclick, 2020)

Figura 46

Perspectiva Macro de Ejecución



Nota. Esta imagen muestra el proceso de ejecución de un programa con el proyecto Longene. (programmerclick, 2020)

CAPITULO III: DESARROLLO, IMPLEMENTACIÓN O TRANSFERENCIA DE TECNOLOGÍA

3.1. ESTUDIO DE FACTIBILIDAD

A) Factibilidad técnica

Para esta factibilidad se realizó el análisis de los requerimientos necesarios para las pruebas e instalación del proyecto Longene en máquinas virtuales, esta se realizo mediante los recursos existentes en el caso de hardware y software requeridos según se detalla explícitamente, según sea pertinente en esta investigación.

Tabla 1

Factibilidad técnica recurso humano

Factibilidad técnica			
Tipo de recursos	**Nombre**	**Experiencia**	**Conocimientos**
Humano	Amir Fernando Mamdouh Mehrez Garcia	Análisis, despliegue y diseño de sistemas multiplataforma. Conocimientos scripting Bash. Administración de servidores en Linux.	Lenguajes de programación C, Bash (Shell Linux), PHP, etc. Conocimientos de Instalación de librerías y despliegue de repositorios Linux.

Nota. Esta tabla muestra la factibilidad técnica requerida basada en recursos humanos. (Diseño Propio)

Tabla 2

Factibilidad técnica recurso hardware

Factibilidad técnica			
Tipo de recursos	**Nombre**	**Cantidad**	**Conocimientos**
Hardware	Laptop	1 und	-Memoria RAM 8 GB - Procesador - Intel core i5 – 6300 2.4GHz - Disco duro 1TB

Nota. Esta tabla muestra la factibilidad técnica requerida basada en recursos de hardware. (Diseño Propio)

Tabla 3

Factibilidad técnica recurso software

Factibilidad técnica			
Tipo de recursos	**Nombre**	**Cantidad**	**Conocimientos**
Software	Windows 10 Pro	1 und	Versión del sistema operativo base de la maquina anfitriona con las actualizaciones correspondientes.
	Ubuntu 16.04 (16 y 32 bits)	1 und	Versión de Sistema operativo utilizada para el despliegue de Wine y Longene
	VMWare	1 und	Versión de prueba del HiperVisor VMWare para la

			creación de máquinas para pruebas
	Librerías Varias de Ubuntu	1 und	Librerías de Ubuntu de repositorios gratuitos
	Wine	1 und	Reimplementación de interfaz para aplicaciones de Win32 y/o Win16 para sistemas operativos basados en Unix.
	Longene	1 und	Proyecto de Kernel Unificado para la modificación de Wine en el Kernel
	Notepad++	1 und	Aplicación de software para las pruebas de instalación y rendimiento
	Vecino Infernal	1 und	Juego de Win32 para las pruebas de instalación y rendimiento

Nota. Esta tabla muestra la factibilidad técnica requerida basada en recursos de software. (Diseño Propio)

B) Factibilidad Económica

En el estudio de factibilidad económica permitió realizar la evaluación de la inversión necesaria para desarrollar esta investigación. Esta factibilidad permite considerar los costos del

proyecto en relación a la instalación y pruebas propuestas; para pergeñar el costo respectivo en futuros escenarios de investigación.

Costo de recursos humanos

Al ser un proyecto de titulación aplicada enfocada en la evaluación de la viabilidad de un proyecto, el investigador no tuvo recursos humanos adicionales para la instalación y pruebas del proyecto.

Tabla 4

Factibilidad económica recurso humano

Factibilidad económica			
Nombre	**Cantidad**	**Costo Unitario**	**Costo Total**
Investigador	1	0.00	0.00

Nota. Esta tabla muestra la factibilidad económica requerida basada en recursos humanos. (Diseño Propio)

Costo de Hardware

La factibilidad técnica va en relación a la considerada para el investigador para las pruebas del proyecto. Ya que el investigador cuenta con el equipo necesario y no hay la necesidad de adquirir adicionales, este es el único tomado en cuenta.

Tabla 5

Factibilidad técnica recurso hardware

Factibilidad económica			
Hardware			
Nombre	**Cantidad**	**Costo Unitario**	**Costo Total**
Laptop	1	3500	3500

Nota. Esta tabla muestra la factibilidad económica requerida basada en recursos de hardware. (Diseño Propio)

Costos del software

Para la instalación y pruebas del proyecto Longene, se poseen en su mayoría software libres o versiones gratuitas de los mismo; por este

motivo la consideración se hará por paquetes o de forma detallada según se requiera.

Tabla 6

Factibilidad económica recurso software

Factibilidad económica			
Software			
Nombre	Cantidad	Costo Unitario	Costo Total
Windows 10 Pro	1 und	100.00	0.00
Ubuntu 16.04 (16 y 32 bits)	1 und	0.00	0.00
VMWare	1 und	0.00	0.00
Librerías Varias de Ubuntu	1 und	0.00	0.00
Wine	1 und	0.00	0.00
Longene	1 und	0.00	0.00
Notepad++	1 und	0.00	0.00
Vecino Infernal	1 und	0.00	0.00

Nota. Esta tabla muestra la factibilidad económica requerida basada en recursos de software. (Diseño Propio)

El costo total del proyecto fue de 3600 por parte del investigador, esto permitió poder realizar la investigación pertinente para evaluar así la viabilidad del proyecto Longene, ya que el proyecto es económicamente factible considerando que solo se realiza una inversión únicamente considerada en el hardware, con la consigna adicional de que los softwares son gratuitos y siendo a su vez el hardware el equipo personal del investigador, equipo que fue usado

para la instalación y pruebas, siendo este referente para que se realice las mismas configuraciones en equipos futuros.

C) Factibilidad operativa

El presente proyecto esta diseñado para evaluar la viabilidad del Proyecto de kernel unificado Longene como alternativa para la ejecución de programas Windows en Linux, por este motivo se plantea la ruta de instalación, forma de uso y pruebas necesarias consideradas en este investigación; por lo expuesto, la factibilidad operativa depende de la experiencia de los usuarios al momento de usar un entorno Linux, ya que este es un entorno modificado para ser una alternativa a un uso habitual de un entorno Windows, ya que Windows es más común para los usuarios finales habituales, no se considera que exista inconveniente por parte de usuarios que están acostumbrados a usar Windows frente a la alternativa presentada en esta investigación.

Además, la operatividad dependerá de los usuarios que usen esta alternativa, por lo que no se puede plantear de forma explicita o medir la operatividad requerida.

Así pues, por lo dicho y necesario para esta investigación, lo ideal consiste en evaluar en primeria instancia si es viable con los recursos considerados. Por este motivo se procede con el desarrollo de la evaluación de viabilidad en los ítems posteriores.

3.2. INSTALACIÓN LONGENE

Los únicos comandos que proporcionaba Longene en su foro para la instalación son las siguientes líneas de código (traducidas al español).

Código 1

Código de Instalación de Longene

```
1.  ///Lanzamiento de longene-1.0-rc2
2.  ///bybyebye » 2014-01-16 9:31
3.  ///Enlace a este artículo:
    http://www.logene.org/forum/viewtopic.php?f=2&t=16083
4.  ///El equipo de Longene lanzó la versión longene-1.0-rc2. Esta versión
    resuelve los ///siguientes problemas:
5.  ///1. Actualice Wine a la versión 1.7.10
6.  ///2. Resuelva el problema que los usuarios comunes no pueden usar
    después de reiniciar
7.  ///3. Resuelva algunos problemas de permisos de archivos
8.  ///4. Modifique el método de carga de archivos
9.  ///5. Otros errores sobre más versiones Para instrucciones de
    desarrollo, consulte ///"Escrito con motivo del lanzamiento de Longjing
    1.0 ".
10. ///Esta vez, lanzamos el código fuente y los archivos de la máquina
    virtual al mismo tiempo. Longene ha sido preinstalado en los archivos
    de la máquina virtual y varios juegos y se ha instalado software para
    que los fans experimenten. Recomendamos que use Virtual Box para abrir
    la máquina virtual. Si usa Vmware para jugar en pantalla completa,
    puede causar la deriva del mouse. Si aún reconoce nuestro trabajo y
    desea continuar usando Longene, le recomendamos que compile e instale
    Longene desde el código fuente en una máquina real para lograr un mejor
    rendimiento.
11.
12. ///Para descargar el código fuente, vaya al centro de descargas para
    descargar el enlace de descarga del archivo de la máquina virtual,
    elija uno para descargar:
13. ///longene-1.0-rc2-ubuntu-12.04.vmdk
14. ///longene-1.0-rc2-ubuntu-12.04.vmdk.zip
15. ///longene-1.0-rc2 -ubuntu-12.04.vmdk.7z
16. ///Para ver el método de instalación de la máquina virtual, consulte
    las "instrucciones de instalación de vmware longene-1.0-rc2-ubuntu-
    12.04"
17.
18. ============ =========== =========
19.
20. ///Instrucciones de instalación del código fuente:
21. ///el código fuente proporciona dos métodos de instalación: instalación
    automática de ///secuencias de comandos o instalación manual:
22.
23. ///Preparación:
24. ///Instale la biblioteca de dependencias El
25. ///sistema Ubuntu puede ejecutar el siguiente comando para instalar:
26. ///CÓDIGO: SELECCIONAR TODO
27. apt-get build-dep wine
28.
29. ///Para otros sistemas, consulte:
    http://wiki.winehq.org/Recommended_Packages#head-
    ///318cd6975d886cae37b26aabadeade15ad0dbee4
30.
31. ///1. Instalación del script:
32. ///Descomprima el paquete de instalación en la terminal:
33. ///CÓDIGO: SELECCIONAR TODO
34. tar jxvf longene-1.0-rc2.tar.bz2
35. ///ejecutar uk_install.sh
```

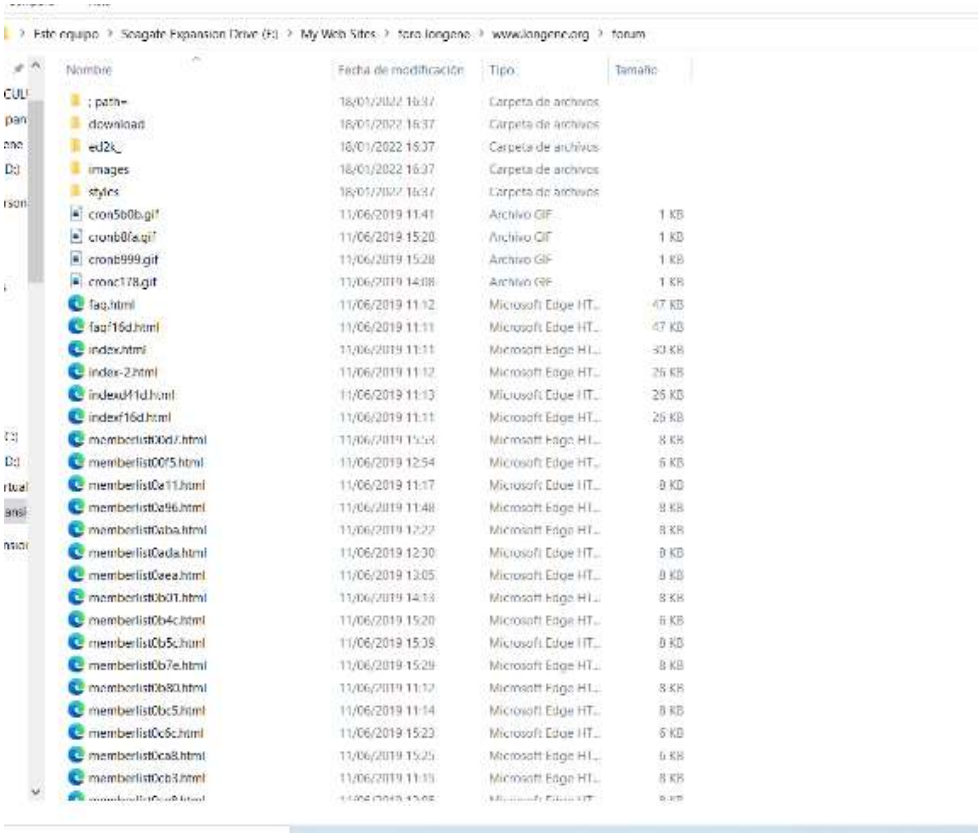
```
///CÓDIGO: SELECCIONAR TODO
./uk_install.sh
///Una vez completada la instalación, reinicie la computadora y use Longene para ejecutar ///los programas de Windows. El método de operación es el siguiente:
///CÓDIGO: SELECCIONAR TODO
./notepad.exe

///Ejecute uk_uninstall.sh para desinstalar Longene
///CÓDIGO: SELECCIONAR TODO
./uk_uninstall.sh
///2. Instalación manual:
///Descomprima el paquete de instalación en la terminal:
///CÓDIGO: SELECCIONAR TODO
tar jxvf longene-1.0-rc2.tar.bz2
///Instale la dll:
///CÓDIGO: SELECCIONAR TODO
cd longene-1.0-rc2/wine
./configure
make
make install
///Instale y cargue el módulo:
///CÓDIGO: SELECCIONAR TODO
cd longene-1.0-rc2/module
make
insmod unifiedkernel.ko
///Una vez finalizado, puede utilizar los siguientes métodos para ejecutar programas de Windows:
///CÓDIGO: SELECCIONAR TODO
./notepad.exe
///Con la instalación manual, el módulo debe cargarse manualmente después de cada reinicio.
///Use el siguiente comando para desinstalar el módulo:
///CÓDIGO: SELECCIONAR TODO
rmmod unifiedkernel.ko
///desinstalar dll
///CÓDIGO: SELECCIONAR TODO
cd longene-1.0-rc2/wine
make uninstall
///Posibles problemas:
///si el programa no se puede ejecutar usando ./notepad.exe en la terminal, la terminal muestra:
///CÓDIGO: SELECCIONAR TODO
unable to find an interpreter for ./notepad.exe

///Ejecute el siguiente comando antes de ejecutar:
///CÓDIGO: SELECCIONAR TODO
mount -t binfmt_misc none /proc/sys/fs/binfmt_misc
echo ':DOSWin:M::MZ::/usr/local/bin/wine:' > /proc/sys/fs/binfmt_misc/register

```

Nota. Este código muestra los pasos para la instalación del proyecto Longene de distintas formas. (Longene, 2014)

Actualmente se poseen la copia de toda la web y la maquina con Longene instalado la cual se pudo rescatar de un repositorio en Baidu que es el buscador chino más usado; lamentablemente no había información para instalar Longene en versiones superiores a Ubuntu 12.04.

Figura 47

Carpeta de foro Longene



Nota. Esta imagen muestra la carpeta de archivos del sitio web de Longene descargado por el investigador. (Diseño Propio)

Para lograr la instalación de Longene en Ubuntu 16, se necesitaron hacer muchos intentos los cuales fueron realizados aplicando prueba y error en la instalación; con lo que finalmente se logró a instalar de forma adecuada; como ya se mencionó no existían guías en todo internet para lograr este fin, por lo que se recurrió a la habilidad adquirida durante varios años de estudio llegando finalmente a este cometido, para describir como se logró esto, se tomaran los errores "**complejos**" aquellos que determinen generar un nuevo título. Se definió como error "**complejo**" aquel error que consista en eliminar muchas librerías anexadas, ejecuciones que ya se estén dando en el núcleo o que se tengan que hacer muchos cambios para regresar a un estado anterior. Esto se visualizará en el punto 3.4. Pruebas de Instalación.

3.2.1. ENTENDIENDO LONGENE

Se tomo la libertad de crear este apartado para tratar de explicar el proyecto, debido a que es preciso el detallar de forma resumida el proyecto Longene y definir ciertos criterios que fueron dejados al aire por falta de la información de otros referentes.

Ya que el proyecto en si era demasiado complejo y con muchos requisitos los cuales son volátiles debido a que Linux usa repositorios que están desarrollados y van cambiando constantemente; ya sea que estos repositorios cambian por actualizaciones o que se eliminen, se hizo un poco cuestionable la realización de este proyecto, pero finalmente se pudo lograr la instalación, recalcando que fue con incontables intentos, los cuales se podrán ver desde el apartado 3.4. Pruebas de Instalación.

Primero se desea explicar si es viable la modificación del proyecto o por que este no fue inmerso en un escándalo por parte de Windows. Al igual que Wine, Longene trabaja bajo una licencia la cual es LGLPv2: "*Las licencias para la mayoría del software están diseñadas para quitarle su libertad para compartirlo y cambiarlo. Por el contrario, el público general de GNU las licencias están destinadas a garantizar su libertad para compartir y cambiar software gratuito: para asegurarse de que el software sea gratuito para todos sus usuarios. Esta licencia, la Licencia pública general menor, se aplica a algunos paquetes de software especialmente designados--típicamente bibliotecas del Free Software Foundation y otros autores que decidan utilizarlo. Ustedes también pueden usarlo, pero le sugerimos que primero piense detenidamente si esta licencia o la Licencia Pública General ordinaria es mejor estrategia a utilizar en cualquier caso particular, con base en las explicaciones a continuación. Cuando hablamos de software libre, nos referimos a la libertad de uso, no precio Nuestras Licencias Públicas Generales están diseñadas para asegurar que usted tiene la libertad de*

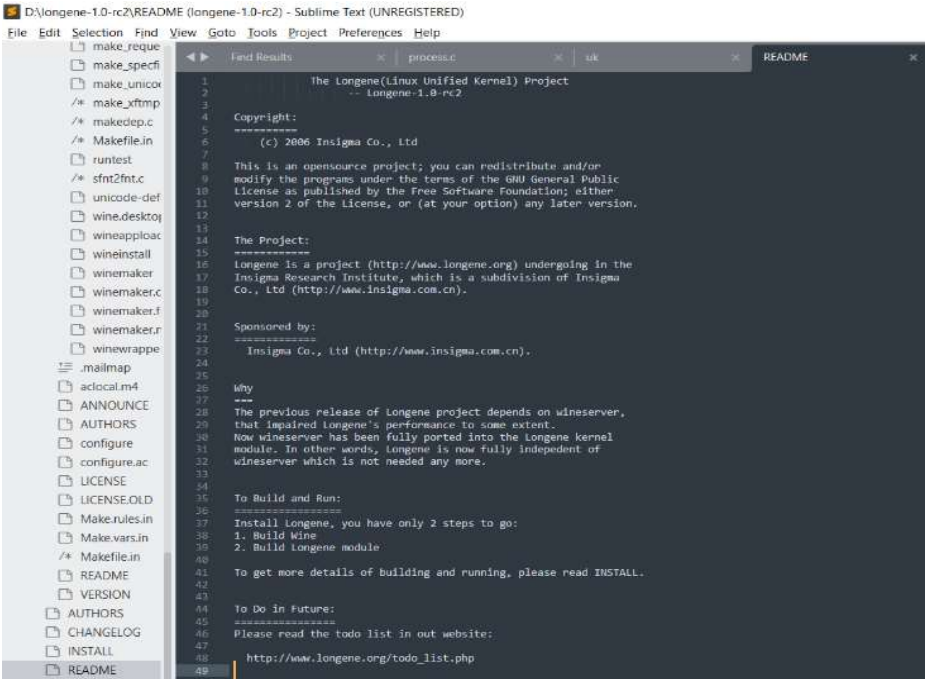
distribuir copias de software libre (y cobrar para este servicio si lo desea); que recibe el código fuente o puede obtener si lo quieres; que puede cambiar el software y usar piezas de en nuevos programas gratuitos; y que se le informe que puede hacer estas cosas.
Para proteger sus derechos, necesitamos hacer restricciones que prohíban distribuidores negarle estos derechos o pedirle que renuncie a estos derechos. Estas restricciones se traducen en ciertas responsabilidades para usted si distribuye copias de la biblioteca o si la modifica.

Por ejemplo, si distribuye copias de la biblioteca, ya sea gratis o por una tarifa, debe otorgar a los destinatarios todos los derechos que le otorgamos Uds. Debe asegurarse de que ellos también reciban o puedan obtener la fuente código. Si vincula otro código con la biblioteca, debe proporcionar archivos de objetos completos a los destinatarios, para que puedan volver a vincularlos con la biblioteca después de realizar cambios en la biblioteca y volver a compilar eso. Y debe mostrarles estos términos para que conozcan sus derechos." (Code Weavers, 2018).

Si se tuvieran dudas de la legitimidad de las imágenes presentes respecto a información como la que se muestra dentro de la carpeta longene-1.0-rc2, se recomienda verificar el siguiente enlace https://github.com/qqzhang/uk-1.0, este repositorio no ha sido modificado desde el año 2014 y además de lo ya mencionado en el apartado 2.2.4. Longene del marco teórico se abala que no existe un artículo público, sitio web público o guía publica en la que se haya implementado longene-1.0-rc2, esta adaptación se realizó entre comillas "a ciegas", por la falta de guías adicionales de instalación que hayan sido dejados o brindados por los desarrolladores del proyecto. La imagen figura 48 muestra el README inicial que tiene cualquier proyecto estructurado, ya que son los primeros pasos.

Figura 48

Readme longene-1.0-rc2

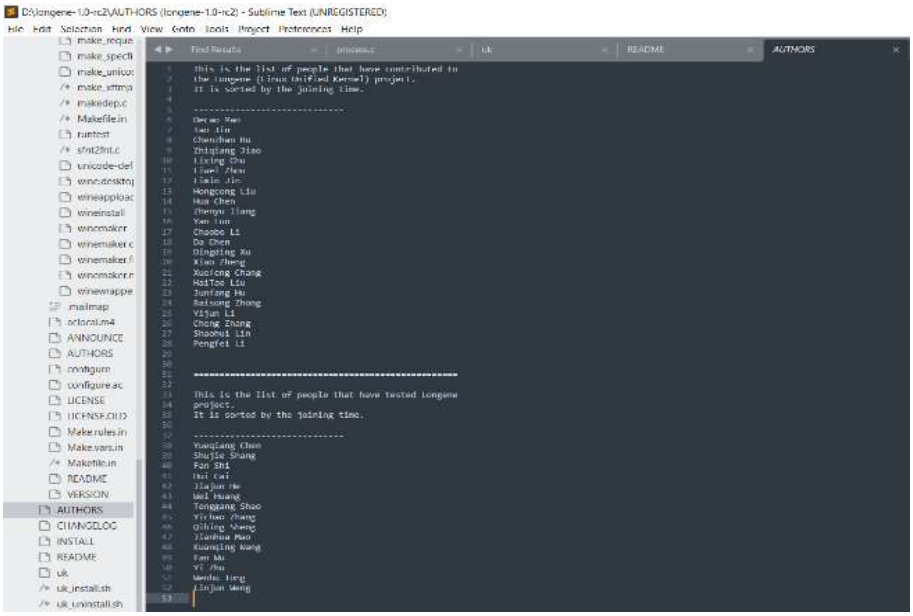


Nota. Esta imagen muestra una foto tomada del archivo Readme del proyecto longene-1.0-rc2. (Diseño Propio)

En la figura 49 se tiene la lista de los autores desarrolladores del proyecto.

Figura 49

Authors longene-1.0-rc2

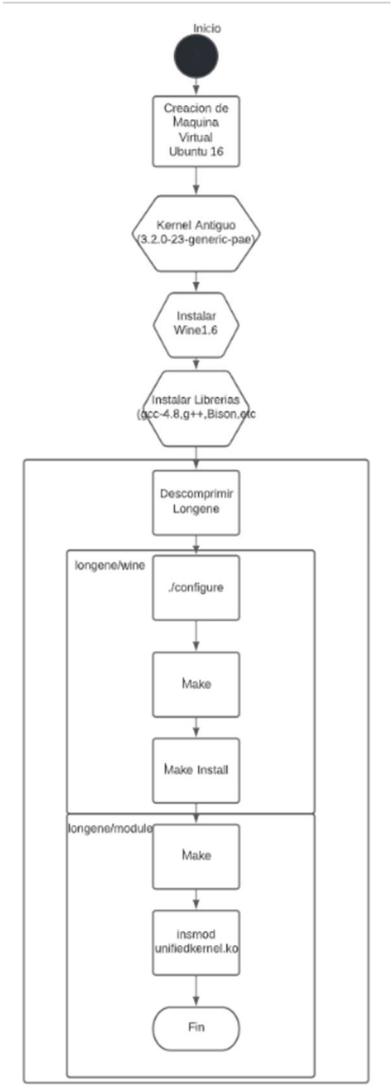


Nota. Esta imagen muestra una foto tomada del archivo Authors del archivo longene-1.0-rc2. (Diseño Propio)

Después de un pequeño descargo de responsabilidades para explicar ciertos criterios éticos y legales en suma necesarios, se procede a mostrar un diagrama de la instalación, viendo este de una forma más dinámica, a pesar de que no sea la mejor opción en este caso se generó un diagrama de flujo el cual permite explicar de forma más didáctica el desarrollo del proceso de instalación correcto del proyecto Longene, a este diagrama llegó y se explicara de forma detallada en los siguientes apartados.

Diagrama 1

Diagrama de Flujo de la instalación de Longene



Nota. Este diagrama muestra el proceso de instalación correcto del proyecto Longene versión longene-1.0-rc2 en Ubuntu 16.04. (Diseño Propio)

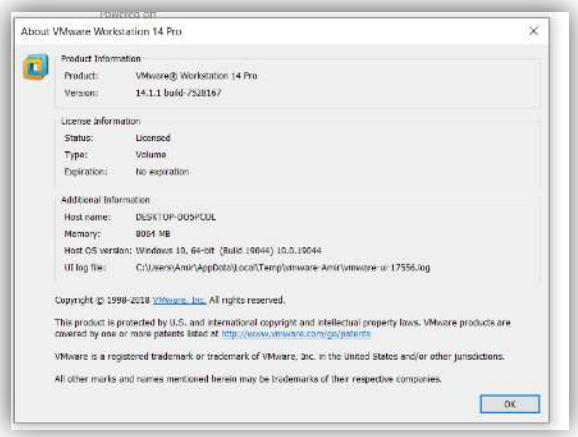
3.3. CREACIÓN DE ENTORNO

3.3.1. CREACIÓN DE MAQUINA VIRTUAL EN VMWARE

Creación de máquina virtual para las pruebas del funcionamiento del proyecto Longene, para esto se usará el programa VMware Workstation; este programa es un hipervisor el cual permite crear máquinas virtuales, admite conexiones de red, uso compartido de unidades de disco físico y acceder a los dispositivos de la computadora. VMware contiene su propio driver de controladores de dispositivo llamado VMware Tools mejorando el rendimiento de almacenamiento, redes, gráficos y sonidos, la versión que se usa es la **14.1.1 build-7528167**, la ventaja de esta versión es que aún se pueden crear máquinas virtuales de 32-bits y se permiten instalar las versiones de Ubuntu 12 y el Ubuntu 16.

Figura 50

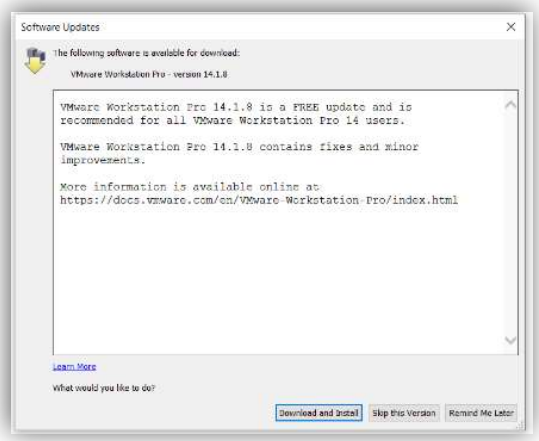
Versión de VMware Workstation 14 Pro



Nota. Esta imagen muestra el apartado acerca de, del VMWare usado para la investigación. (Diseño Propio)

Figura 51

Sugerencia de actualizaciones de VMware

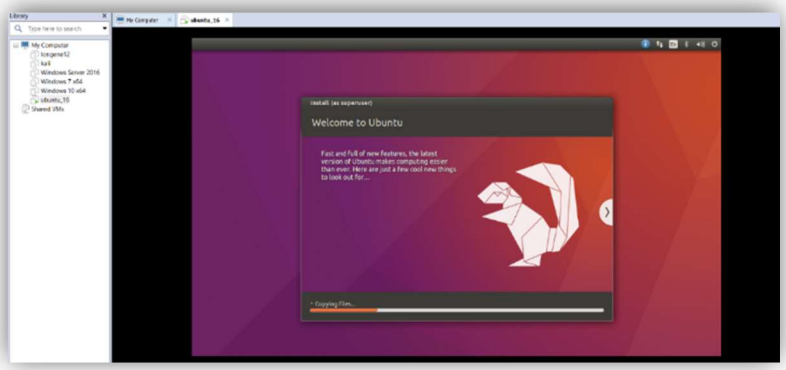


Nota. Esta imagen muestra una Notificación de posibles actualizaciones de VMware Workstation (Diseño Propio)

Al crear máquinas virtuales en VMware se define al inicio de la creación las configuraciones pertinentes, en el caso de Windows por ejemplo solicita la key del registro y en el caso de máquinas Linux (para el caso actual Ubuntu concretamente) se define el nombre del usuario y la contraseña, el nombre de usuario de las maquinas será **amir** y la contraseña ***** será siempre la misma, pero por cuestiones de las pruebas el usuario que será más usado es el super usuario también definido como **root**.

Figura 52

Pantalla de instalación de Ubuntu

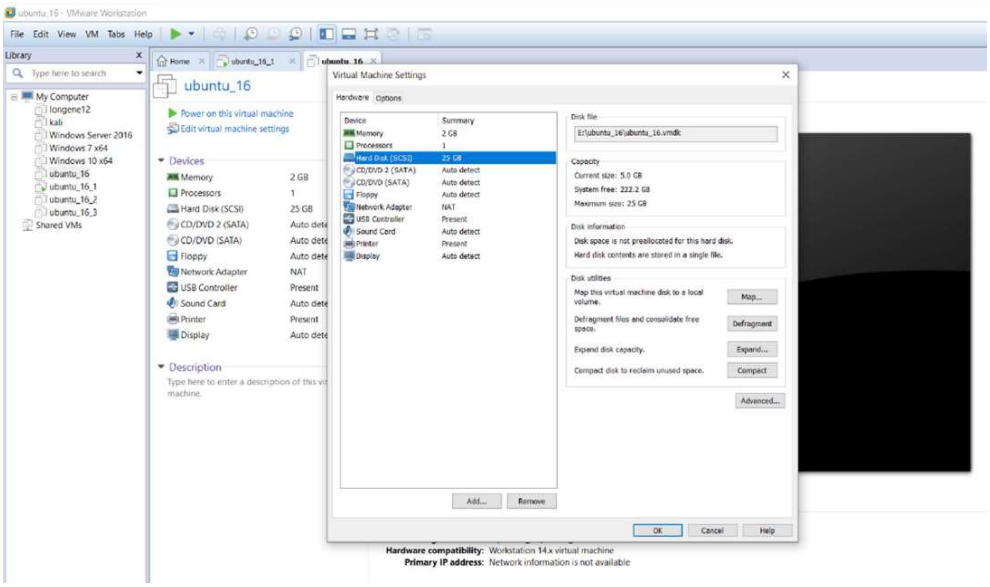


Nota. Esta imagen muestra la pantalla de instalación de Ubuntu. (Diseño Propio)

Para la configuración estándar que tendrán todas las máquinas virtuales que se usaran se asignó la siguiente cantidad de recursos 2GB RAM (actualmente la maquina tiene problemas de RAM por lo que es la cantidad máxima que se le pudo dar), 1 núcleo, adaptador en modo NAT (el modo NAT permite conectar la máquina virtual al exterior definiendo esta como una red privada interna que sale a un router externo siendo este router la maquina anfitriona por un puerto definido para así no tener que hacer configuraciones adicionales para acceder internet) y una carpeta compartida llamada D:\carpetalongene (en esta carpeta se encuentra la última versión estable del proyecto Longene, los comandos de ejecución y un instalador de Notepad++ para Windows xp de 32-bits) véanse las figura 53, 54 y 55.

Figura 53

Configuraciones de máquina virtual

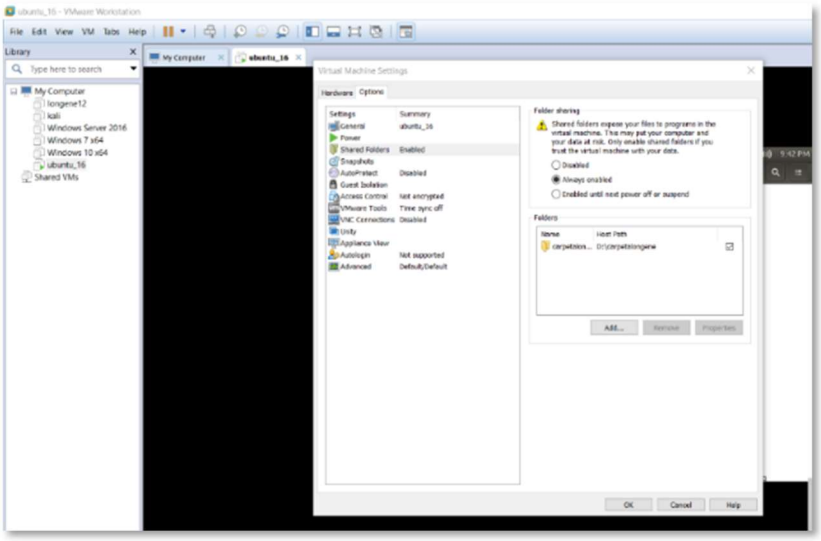


Nota. Esta imagen muestra las configuraciones de hardware de la máquina virtual de Ubuntu. (Diseño Propio)

Se tiene la carpeta compartida llamada D:\carpetalongene, en esta carpeta se encuentra la última versión estable del proyecto y archivos adicionales para la instalación y pruebas del proyecto.

Figura 54

Carpeta de Recursos para Instalación

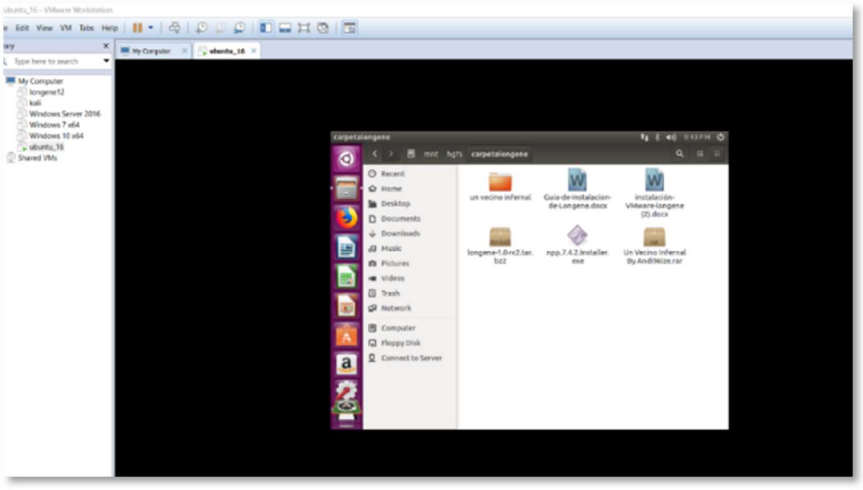


Nota. Esta imagen muestra la carpeta compartida de recursos para las instalaciones respectivas. (Diseño Propio)

Dentro de la carpetalongene se tiene un archivo con los comandos de instalación, el comprimido bz2 del proyecto Longene, el juego “Un Vecino Infernal” y el instalador de Notepad++ 7.4.2 para Windows.

Figura 55

Archivos de carpeta de recursos “carpetalongene”



Nota. Esta imagen muestra los archivos dentro de la carpeta compartida de recursos para las instalaciones respectivas. (Diseño Propio)

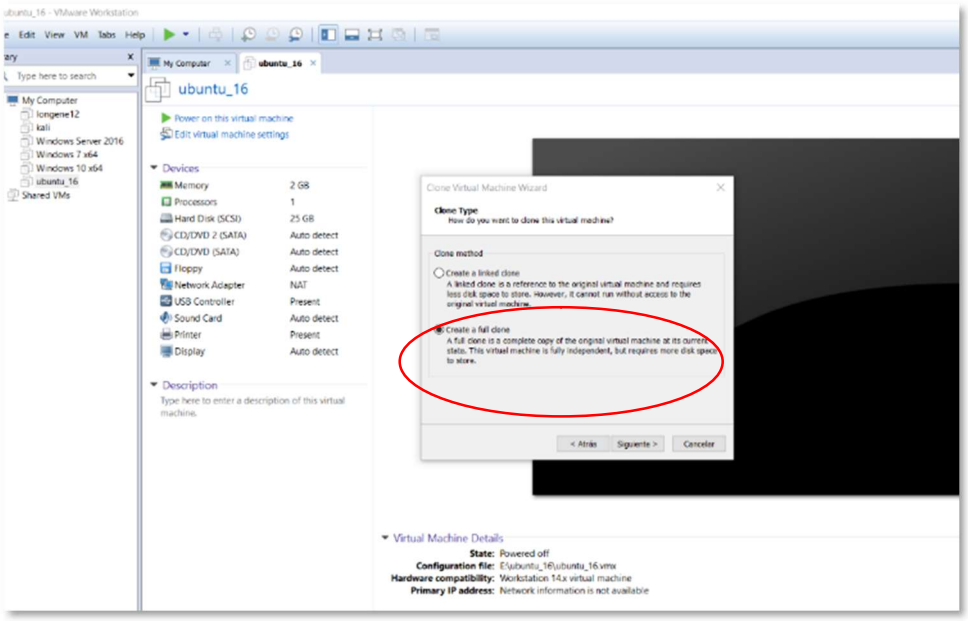
3.3.2. CLONACIÓN DE LA MAQUINA VIRTUAL

Para poder realizar pruebas diversas de la instalación y el cargado del modulo Longene, se procedió a clonar la máquina virtual para tener las mismas características, pero el tipo de clonación escogida es full clone**.** "*Un full clone es una VM hijo independiente que no comparte nada con la VM padre después de la operación de clonación. La operación en curso de un clon completo es totalmente independiente del VM padre*". (VMware, 2021)

Esto significa que es un sistema operativo totalmente nuevo con las características instaladas del sistema operativo existente, por ende, estos dos no comparten memoria virtual entre sí mismos. Elección de clone full, en la pantalla de clonación de VMware

Figura 56

Elección de clone full

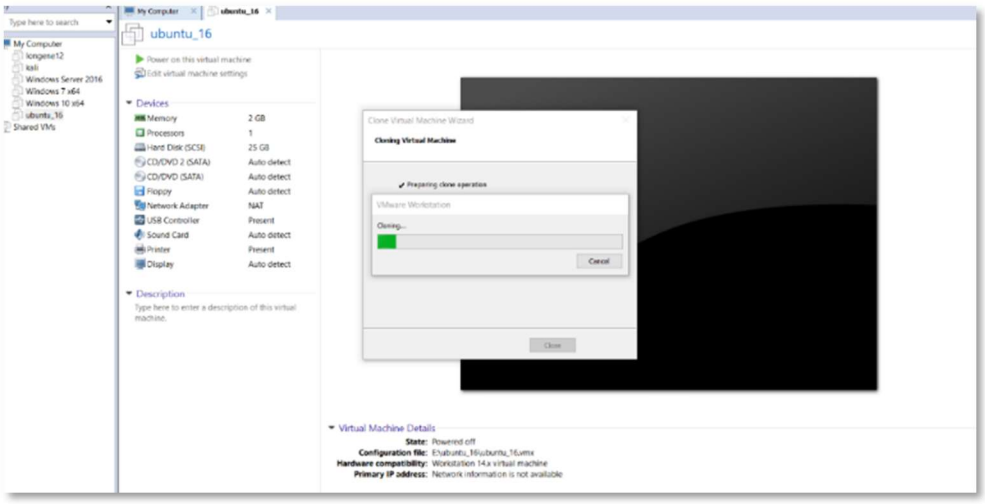


Nota. Esta imagen muestra la opción para crear un clone completo con VMWare. (Diseño Propio)

Se puede observar la pantalla de carga de clonación de la máquina, que se está realizando.

Figura 57

Pantalla de Carga de Clonación Full

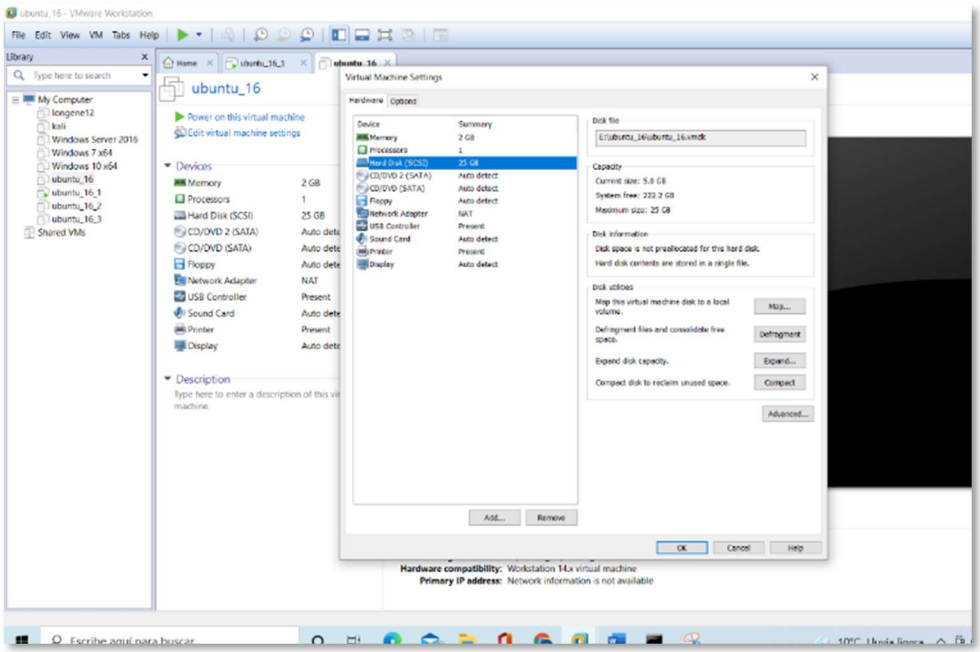


Nota. Esta imagen muestra la pantalla de carga de un clonado full. (Diseño Propio)

En la siguiente imagen se muestra, la configuración de recursos de máquinas virtuales futuras a clonar.

Figura 58

Configuraciones de máquina virtual clonada



Nota. Esta imagen muestra las configuraciones de una máquina virtual clonada. (Diseño Propio)

3.4. PRUEBAS DE INSTALACIÓN

3.4.1 LONGENE EN UBUNTU DE 64 BITS

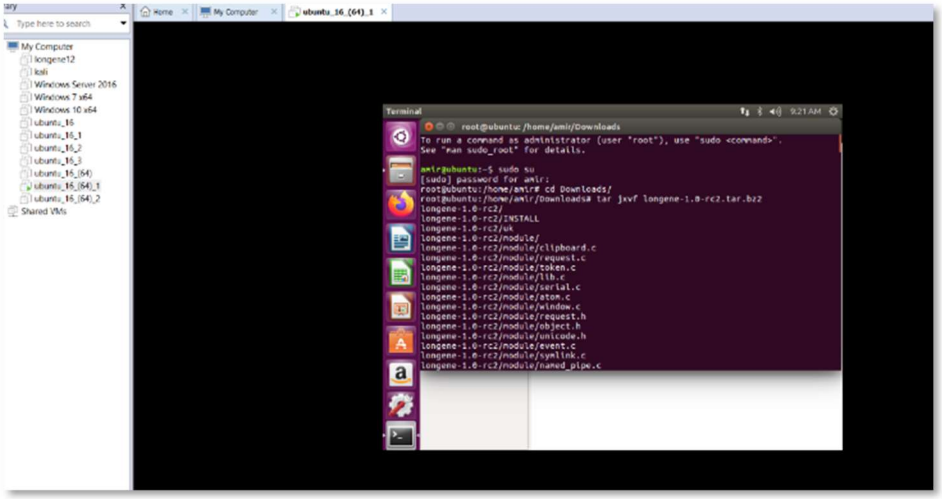
Para todos los casos de la prueba se copiaron los archivos de la carpeta compartida ubicada en el directorio /var/mnt/carpetalongene a la carpeta home/amir($user)/Downloads; luego de esto se descomprime el archivo longene-1.0-rc2.tar.bz2 que fue la última versión estable sacada por Longene, esto se hace mediante el comando tar y las opciones jxvf, tar es uno de los comandos más usados para la compresión de archivos en el entorno Ubuntu:

j: compresión bzip 2 x: Extraer archivo

v: Mostrar descripción detallada f: Nombre de archivo

Figura 59

Descompresión de proyecto Longene

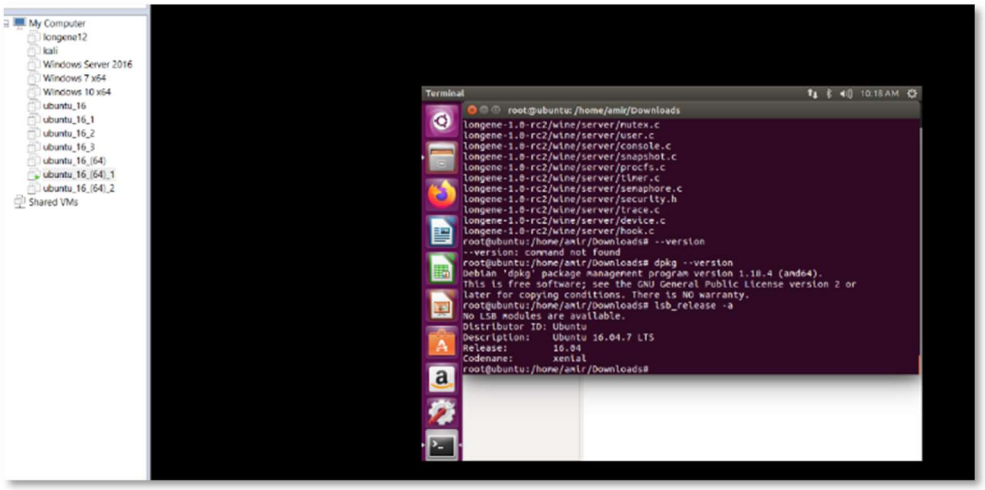


Nota. Esta imagen muestra la descompresión de proyecto Longene. (Diseño Propio)

Este es el resultado de la descompresión del archivo tar mencionado antes por lo que para verificar el entorno donde se está haciendo la prueba verifico la versión de Linux que estoy usando, con el comando dpkg -version. Para el caso de estas pruebas de Ubuntu 16 de 64 bits se usará la versión 16.04.7 y se verifica en la figura 60 que el tipo de sistema operativo es de 64 bits.

Figura 60

Visualizar versión de Ubuntu por comando

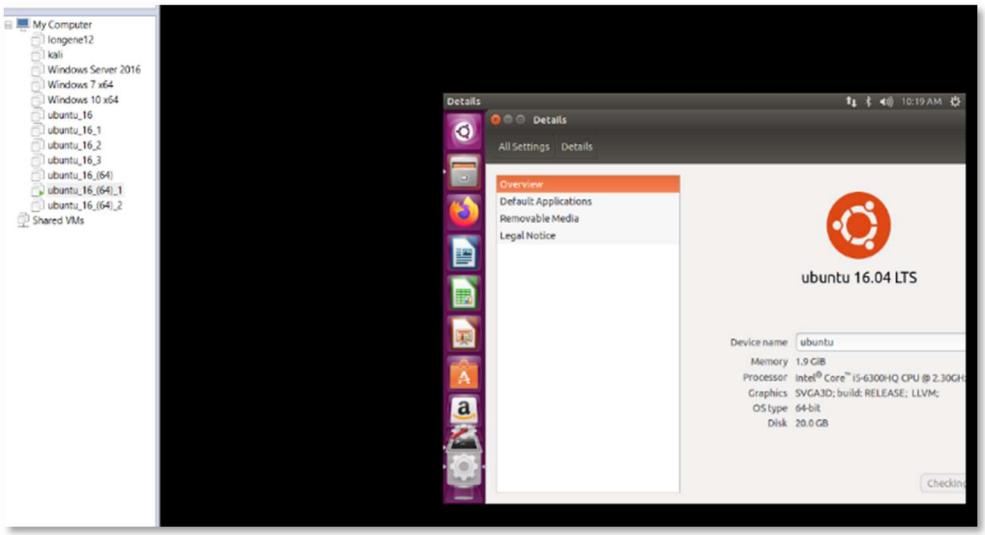


Nota. Esta imagen muestra la visualización de la versión de Ubuntu por comando. (Diseño Propio)

Adicionalmente existe la posibilidad de corroborar la versión de Ubuntu de forma visual, entrando por la opción de herramientas.

Figura 61

Visualizar la versión de Ubuntu por interfaz



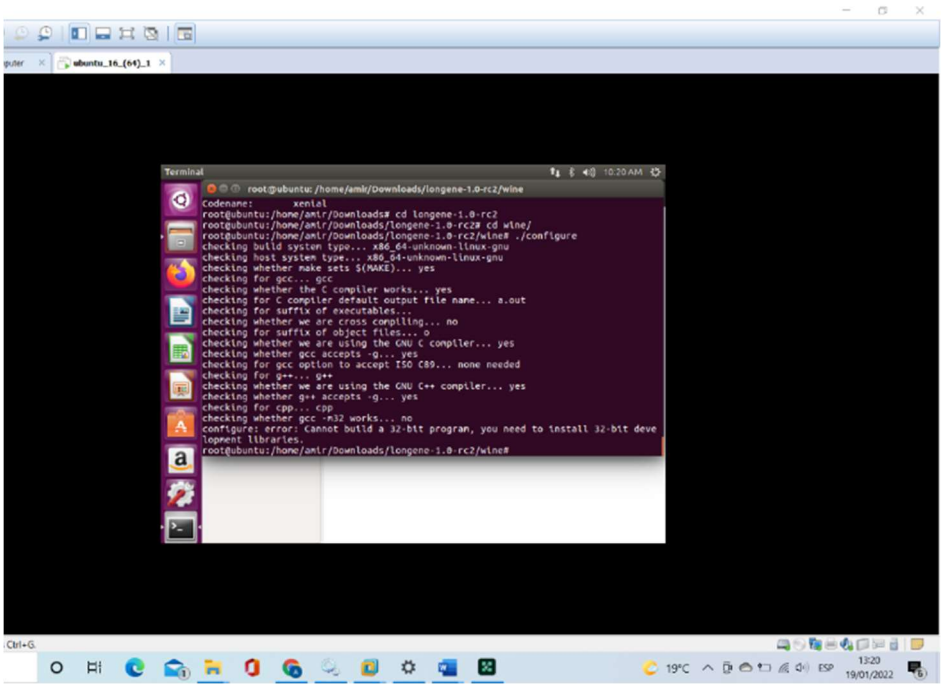
Nota. Esta imagen muestra la visualización de la versión de Ubuntu por interfaz gráfica. (Diseño Propio)

Dentro de la carpeta /home/amir($user)/Downloads/longene-1.0-rc2/wine se procede a realizar la instalación requerida para la ejecución de los programas Windows en Linux, para esto se

comienza con la ejecución del archivo ./configure, configura es un script que comprueba las dependencias requeridas para que el proceso de compilación e instalación estén disponibles, normalmente los script de tipo configura están hechos en C por lo que es necesario un compilador de este tipo.

Figura 62

Ejecución de comando ./configure



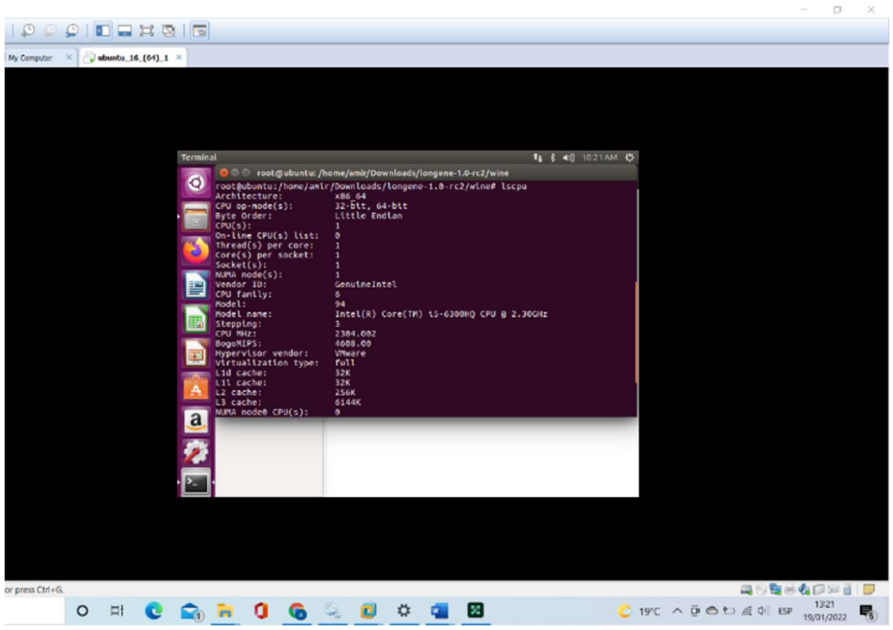
Nota. Esta imagen muestra el resultado de la ejecución del comando ./configure. (Diseño Propio)

El primer error que salta de la ejecución de configura es que el programa no puede ser compilado porque longene-1.0-rc2 usa las librerías de un programa de 32 bits.

Dada la confirmación hecha previamente procedo con la revisión la versión de la versión de Ubuntu en la que se encuentra, esto es realizado con el comando lscpu, el cual muestra información del procesador, la arquitectura, frecuencias, etc.

Figura 63

Ejecución de comando lscpu

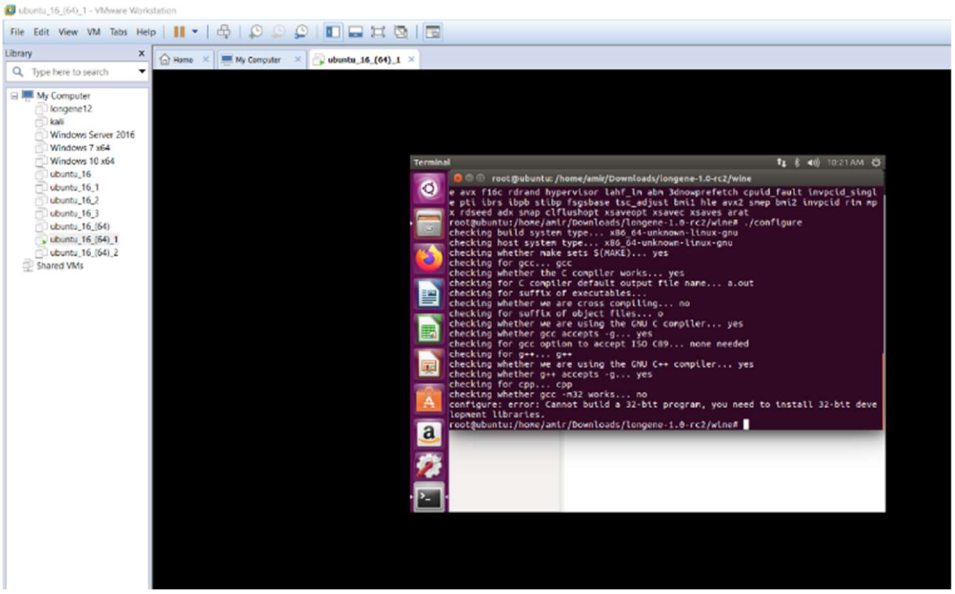


Nota. Esta imagen muestra el resultado de la ejecución del comando lscpu. (Diseño Propio)

Aunque el cpu cuenta con los 2 modos de ejecución, 32-bits y 64-bits, Ubuntu no puede compilar el proyecto Longene porque no se tienen las librerías de 32-bits implementadas dentro de la máquina.

Figura 64

Error ejecutar ./configura en 64 bits

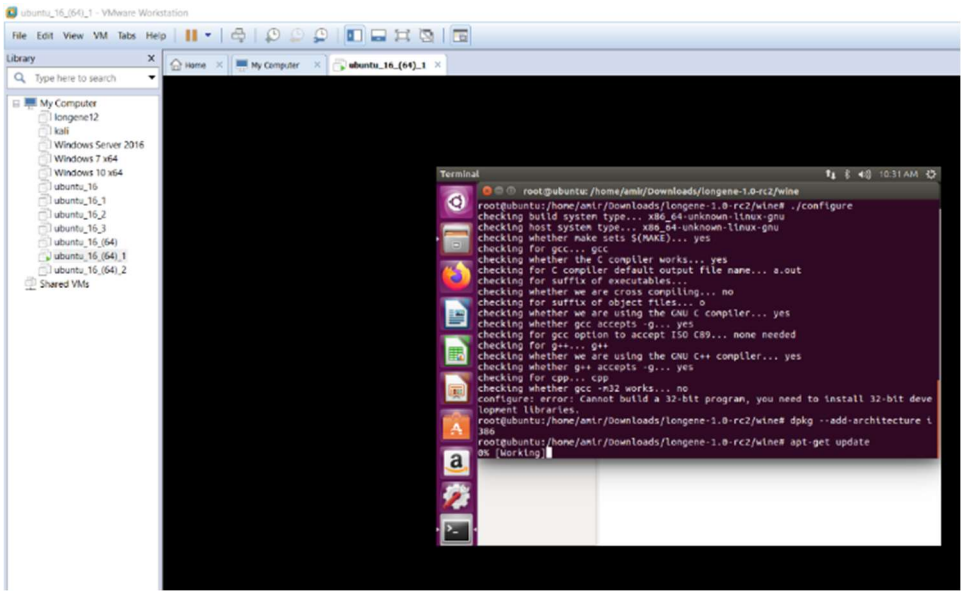


Nota. Esta imagen muestra el mensaje de error al ejecutar ./configura en 64 bits. (Diseño Propio)

Para permitir la ejecución de un programa de 32-bits se procede a agregar las librerías de arquitectura de 32-bits con el comando dpkg --add-architecture i386 (en Ubuntu la denominación de un programa de 32 bits esta denominada como i386)

Figura 65

Agregar librerías arquitectura 32 bits

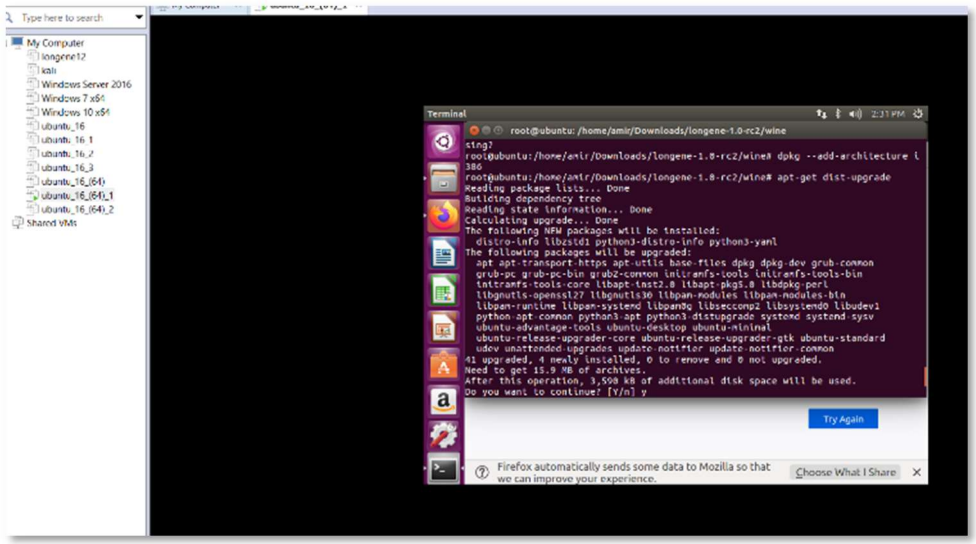


Nota. Esta imagen muestra la ejecución de comandos para ejecutar en 32 bits en 64 bits. (Diseño Propio)

Lo que sigue es agregar la arquitectura para proceder con la instalación de la librería de 32 bits de la distribución, esto permitirá tener los recursos necesarios para lograr la ejecución de programas en 32-bits, para esto se usara el comando **dpkg --add-architecture i386**, este comando permite agregar la arquitectura adicional y después se solicitaran las librerías necesarias para la instalación multi-arquitectura ya sea i386 para 32 bits y amd64 para 64 bits. Luego se hace una actualización de la distribución para instalar las librerías adicionales necesarias para un funcionamiento correcto al momento de actualizar la distribución.

Figura 66

Upgrade de la distribución

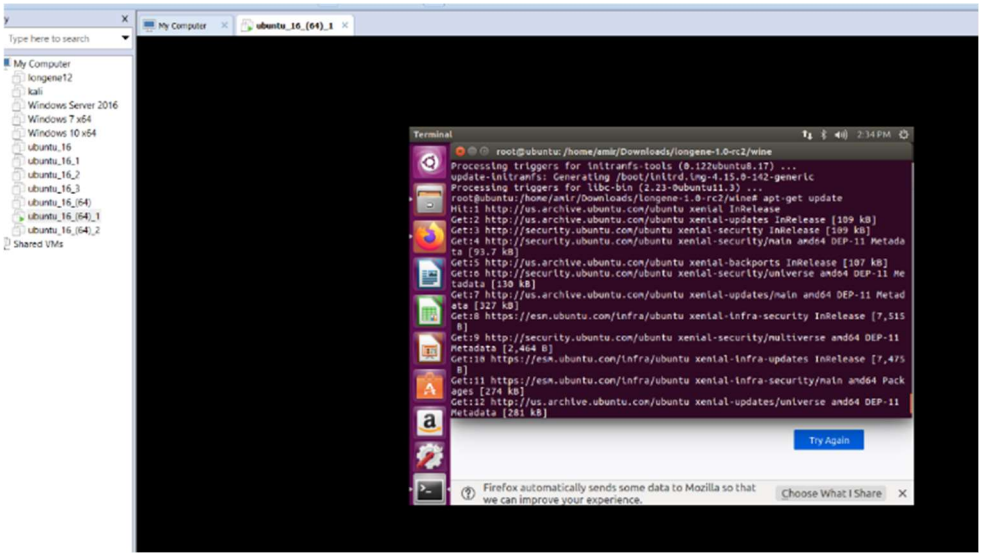


Nota. Esta imagen muestra la ejecución de upgrade de la distribución. (Diseño Propio)

Después de actualizar la distribución se necesitan actualizar todos los recursos de la source.list de Ubuntu, esto se realiza con el comando **apt-get update**. Este comando sirve para actualizar la lista de paquetes disponibles y sus versiones definidas en la source.list.

Figura 67

Actualización de paquetes disponibles

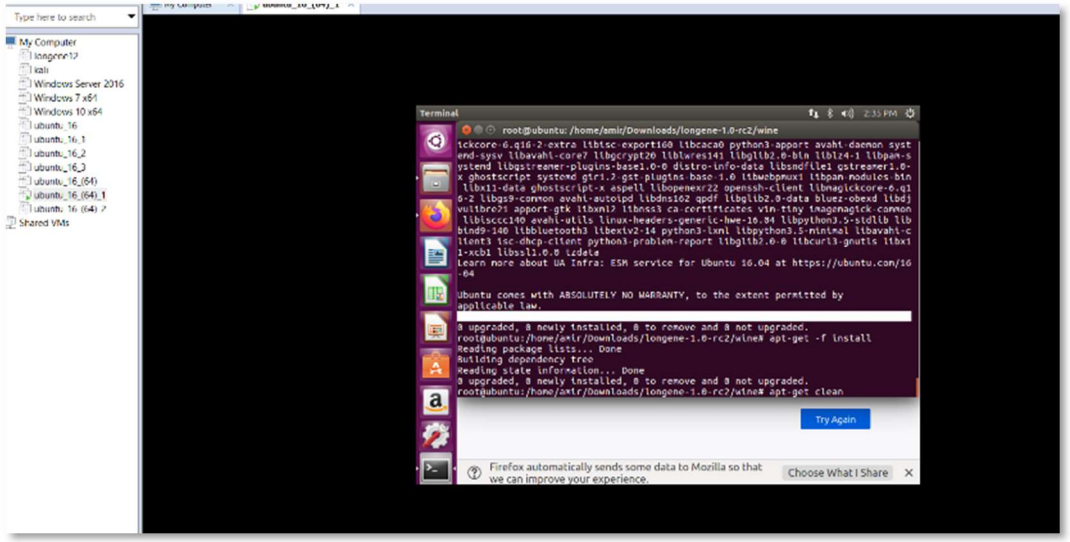


Nota. Esta imagen muestra la ejecución de actualización de paquetes disponibles (Diseño Propio)

Lo último que queda para permitir la instalación de wine se usara el comando apt-get -f install (-f, --fix-broken es un comando para intentar corregir las dependencias rotas cuando se usa para instalar o eliminar las librerías necesarias para la acción que se quiera realizar). Después de eso la ejecución del comando apt-get clean el cual limpia el registro de los archivos temporales del repositorio local de archivos de paquetes recuperados, así se remueve la duplicidad o falta de referencias.

Figura 68

Forzar la instalación e instrucciones de clean

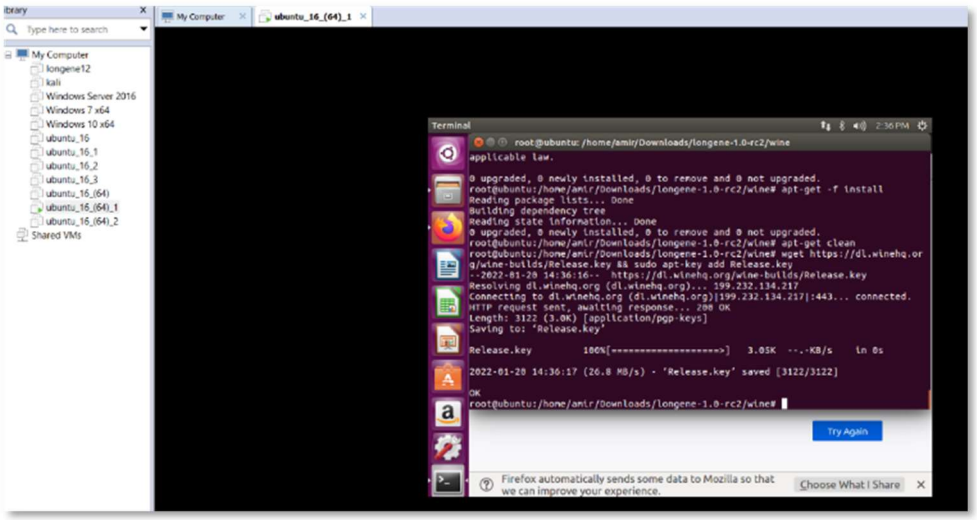


Nota. Esta imagen muestra la ejecución de instrucciones de clean y forzar la instalación (Diseño Propio)

Para instalar wine en la maquina es brindar el link donde se encuentra el servidor repositorio, esto se logra con el comando wget el cual se usa para recuperar el contenido y archivos de un servidor web y después agregar el key. En este caso wget https://dl.winehq.org/wine-builds/Release.key && sudo apt-key add Release.key

Figura 69

Agregar el Key Wine

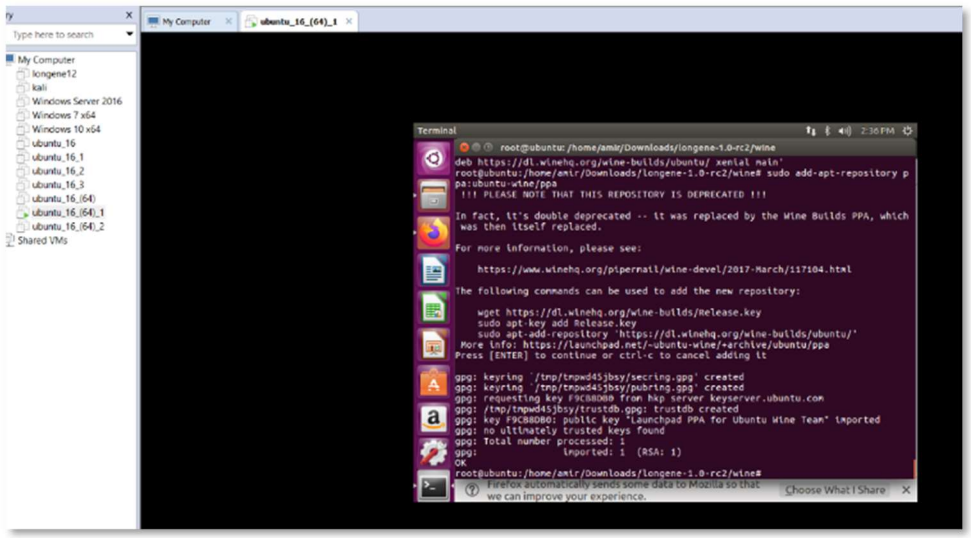


Nota. Esta imagen muestra el agregar la key de Wine.(Diseño Propio)

Se procede a agregar el repositorio al archivo source.list, esto se realiza con el comando apt-add-repository '$extension $link $versionubuntu', ya que el Ubuntu que se esta usando es el 16 se debe hacer referencia a la máquina de tipo xenial main.

Figura 70

Respuesta de adicionado de repositorio

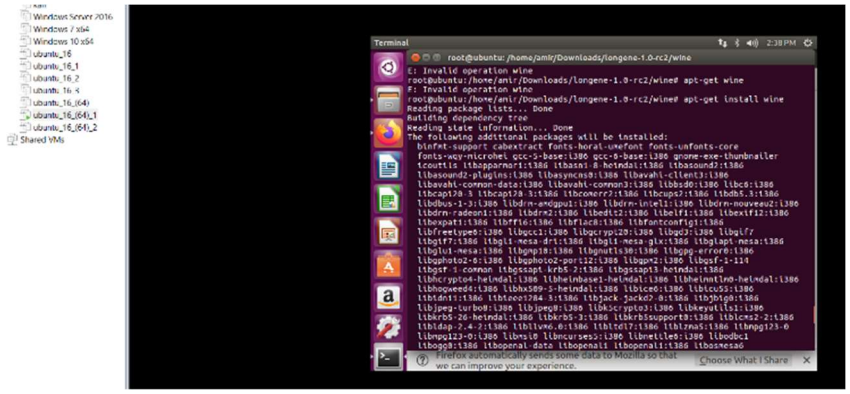


Nota. Esta imagen muestra la respuesta de adicionar el repositorio. (Diseño Propio)

Con el repositorio agregado lo que toca es la instalación de wine, esto se realizara con el comando apt-get install wine, al dia 20/01/2022 la versión por defecto al instalar wine del repositorio de xenial main, es la versión 1.6 de Wine. El comando “apt-get install” se usa para descargar la última versión de la aplicación que se desea instalar desde un repositorio de software en línea que apunta al archivo de configuración source.list**.**

Figura 71

apt-get install Wine

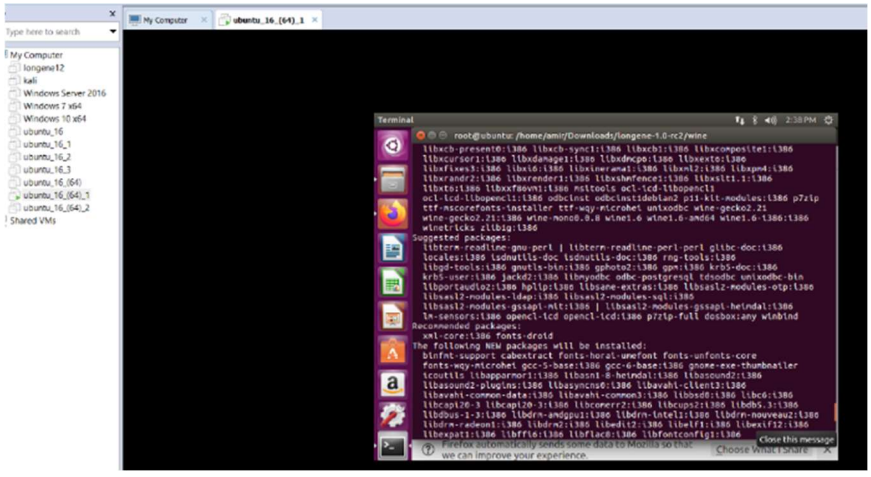


Nota. Esta imagen muestra la Instalación de Wine con el comando apt-get install Wine. (Diseño Propio)

Se puede observar la sugerencia de paquetes al solicitar la instalación de Wine.

Figura 72

Sugerencia Paquetes Wine

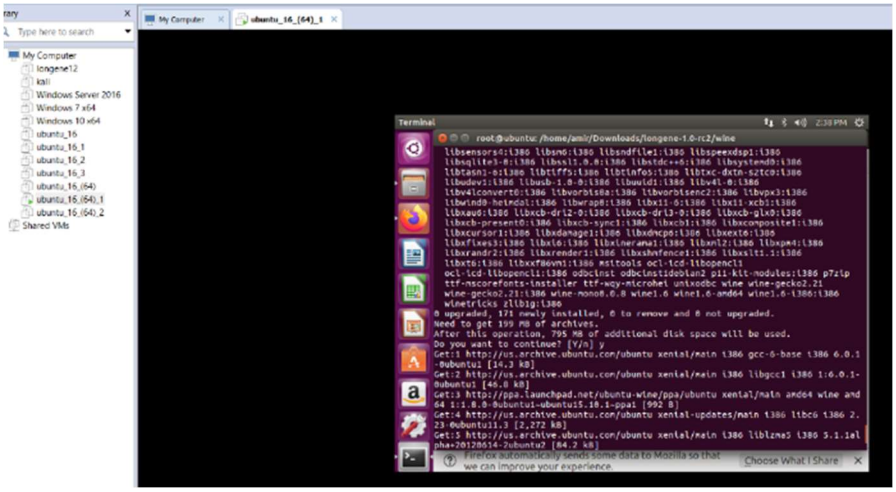


Nota. Esta imagen muestra la sugerencia de paquetes recomendados para instalar Wine. (Diseño Propio)

Al intentar instalar la versión se observa que se instalara wine1.6 por defecto.

Figura 73

Tamaño y adicionales de instalación de Wine

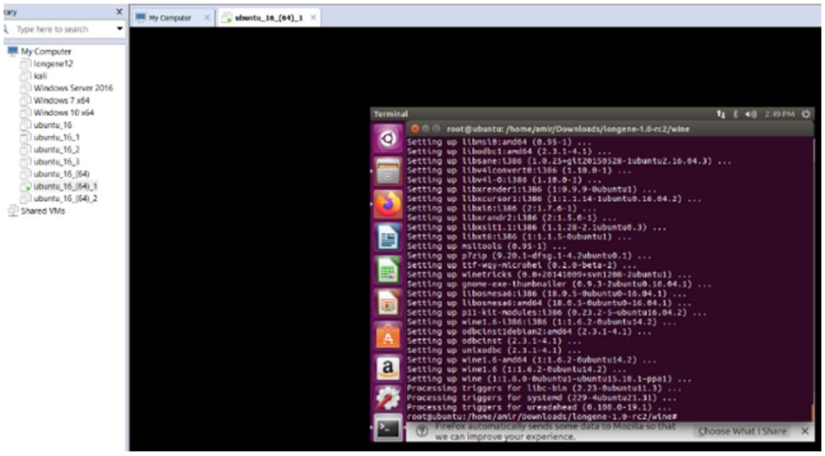


Nota. Esta imagen muestra el Tamaño y archivos adicionales de instalación de Wine. (Diseño Propio)

El resultado final de instalar wine de esta forma muestra que no hubo errores durante la instalación.

Figura 74

Resultado Final instalación Wine



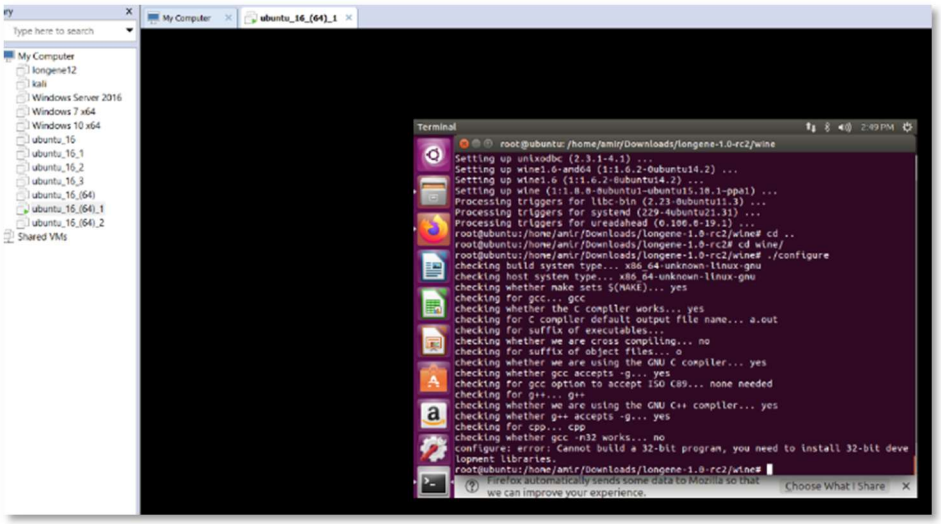
Nota. Esta imagen muestra el resultado final de la instalación de Wine (Diseño Propio)

Después de observar que la instalación fue realizada con éxito en la figura 74, se procede con el intento de instalación del proyecto Longene. Para esto se accede a la carpeta

/home/amir/Downloads/longene-1.0-rc2/wine**.** Dentro de esta carpeta se utiliza el comando ./configure, este comando ejecuta el script contenido dentro de configura para reparar y localizar la distribución fuente para que se compile y cargue en el sistema Linux, esto asegura de que todas las dependencias necesarias para el resto del proceso de compilación e instalación estén disponibles y revela todo lo que se necesita saber para usar esas dependencias.

Figura 75

Error ./configure Longene



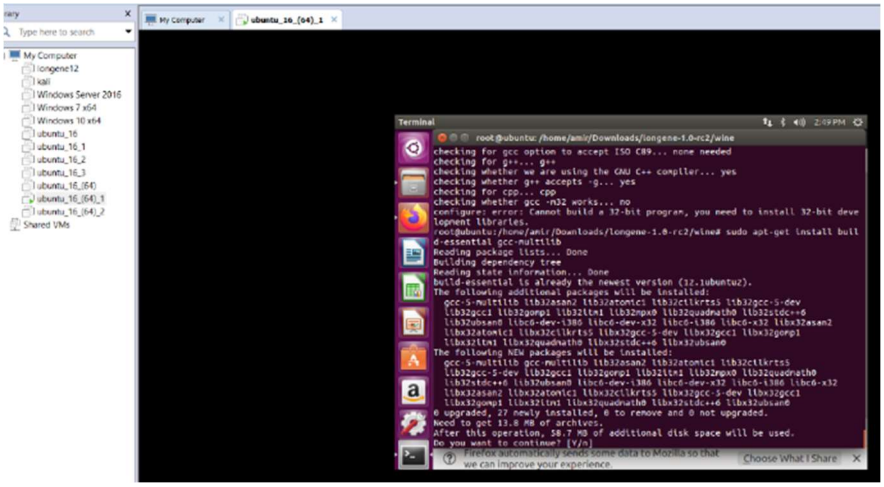
Nota. Esta imagen muestra el resultado con error al intentar ejecutar ./configure en Longene (Diseño Propio)

Al intentar la ejecución de la configuración nuevamente sigue apareciendo el error de que no se tienen instaladas las versiones de 32-bits de las librerías. Debido a esto se procede a instalar las librerías build-essential y gcc-multilib. “Los paquetes build-essentials son metapaquetes que son necesarios para compilar software. Incluyen el depurador GNU, la colección de compiladores g++/GNU y algunas herramientas y bibliotecas más que se requieren para compilar un programa. Por ejemplo, si necesita trabajar en un compilador C/C++, debe instalar metapaquetes esenciales en su sistema antes de iniciar la instalación del compilador C. Al instalar los paquetes build-essential, algunos otros paquetes como G++, dpkg-dev, GCC y make, etc.” (Búzdar, 2021)

En el caso de gcc-multilib, este se utiliza para realizar una compilación cruzada, compilar en una arquitectura de procesador diferente. Si se está usando un Ubuntu de 64 bits se usa esto para ejecutar un en Ubuntu de 32 bits.

Figura 76

Instalación de gcc-multilib

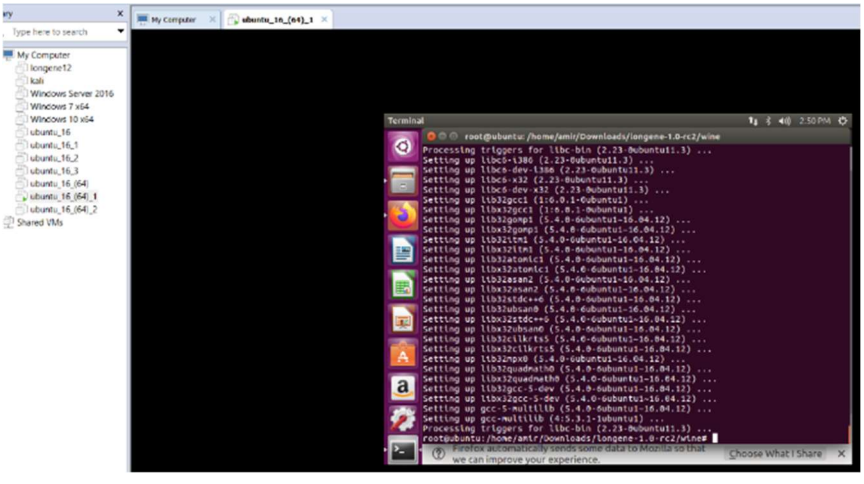


Nota. Esta imagen muestra la pantalla al instalar gcc-multilib. (Diseño Propio)

Como se observa en la imagen anterior, no se encontró ningún error al intentar instalar gcc-multilib.

Figura 77

Resultado instalar gcc-multilib

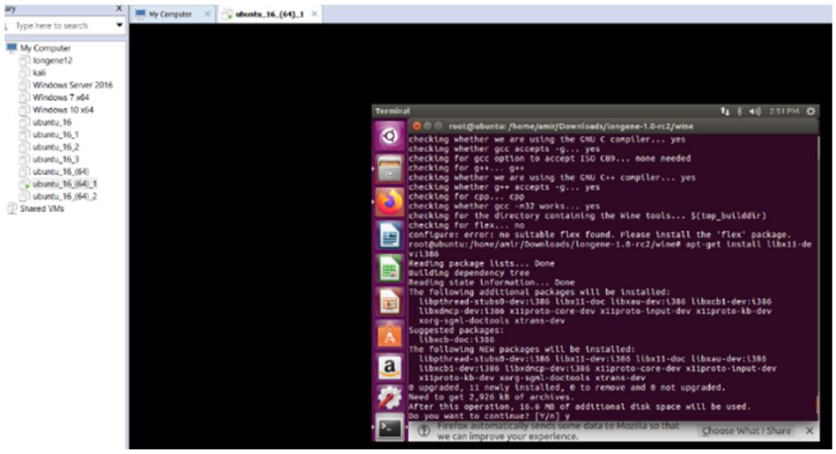


Nota. Esta imagen muestra la pantalla resultante al instalar gcc-multilib. (Diseño Propio)

Al intentar ./configura nuevamente se permite ejecutar el código pero aparece un error de que flex no fue encontrado por lo que para que esto no suceda se instala la librería flex con el comando apt-get install libx11-dev:i386, para instalarla en las librerías de 32-bits se usa :i86.

Figura 78

Instalación de libx11:i386 para flex

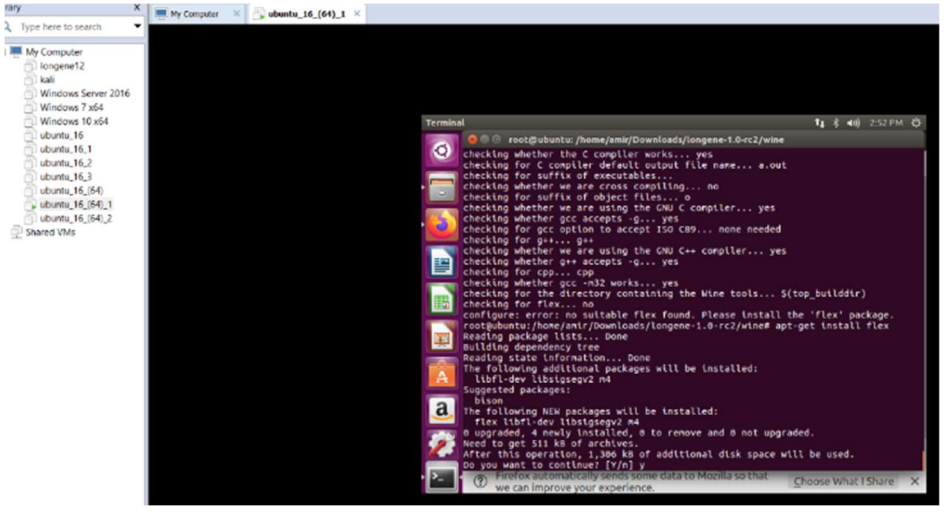


Nota. Esta imagen muestra la pantalla al instalar de libx11:i386 para flex. (Diseño Propio)

Se procede con la instalación de la librería flex necesaria para seguir con la instalación de Longene.

Figura 79

Instalación de flex

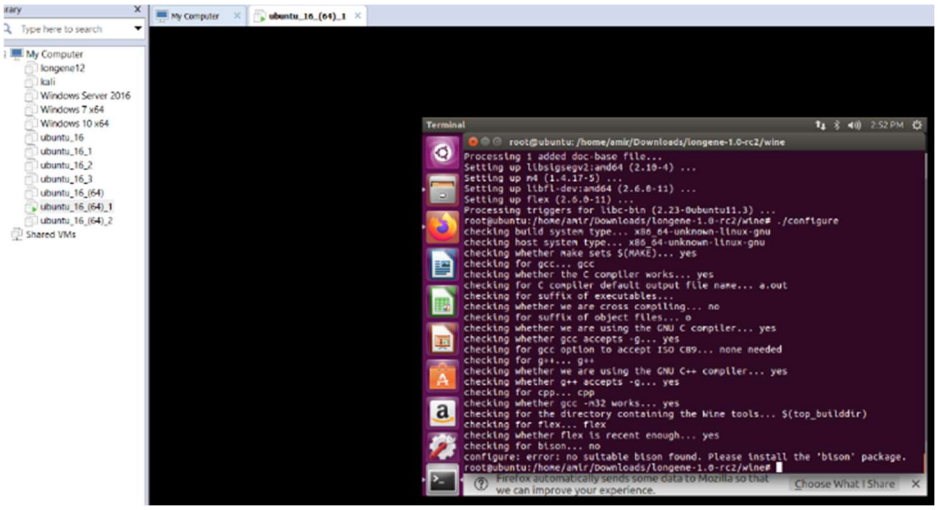


Nota. Esta imagen muestra la pantalla al instalar flex. (Diseño Propio)

Después de haber instalado la librería flex, sin instalar bison, se procede con a la ejecución del comando ./configura pero este presenta un error de que no se encuentra el paquete bison.

Figura 80

Error requiere Bison



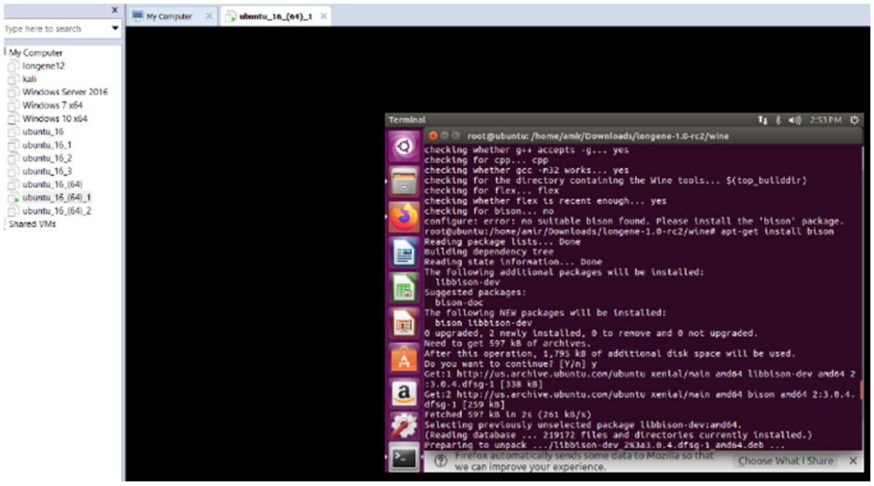
Nota. Esta imagen muestra la pantalla de error al intentar realizar ./configure, se pide que requiere bison, porque este no se encuentra. (Diseño Propio)

Ubuntu sugiere que para la instalación de la librería flex se instale la librería bison. “Bison es un generador de analizadores de propósito general que convierte una gramática libre de contexto anotada en un analizador determinista LR o LR generalizado (GLR) que emplea tablas de analizador LALR (1). Como función experimental, Bison también puede generar tablas de analizador IELR (1) o LR (1) canónicas. Una vez que domine Bison, puede usarlo para desarrollar una amplia gama de analizadores de lenguaje, desde los que se usan en simples calculadoras de escritorio hasta lenguajes de programación complejos”. (System, 2014)

Debido al error suscitado se procede con la instalación de la librería bison con el comando apt-get install bison**.**

Figura 81

Instalación de Bison por comando

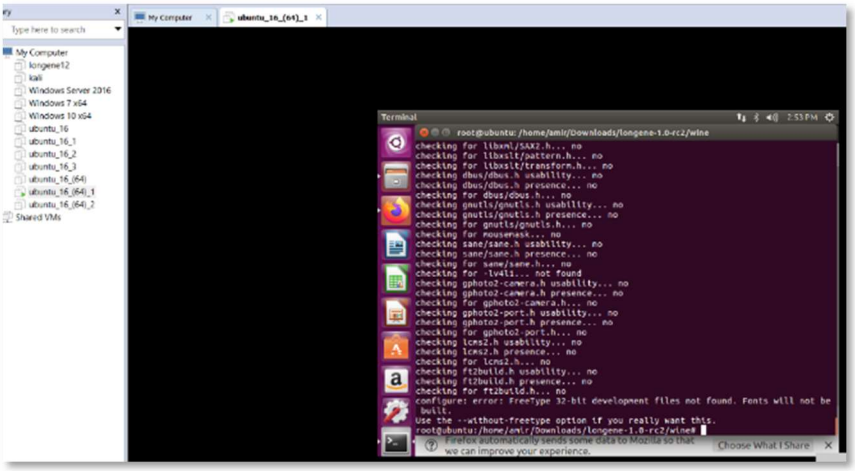


Nota. Esta imagen muestra la pantalla al instalar bison. (Diseño Propio)

Se intenta nuevamente con la ejecución del comando ./configura pero aparece un error de que no se encuentra la librería freetype por lo que se procede a la instalación de esta librería con el comando apt-get install libfreetype6-deb:i386.

Figura 82

Error de freetype

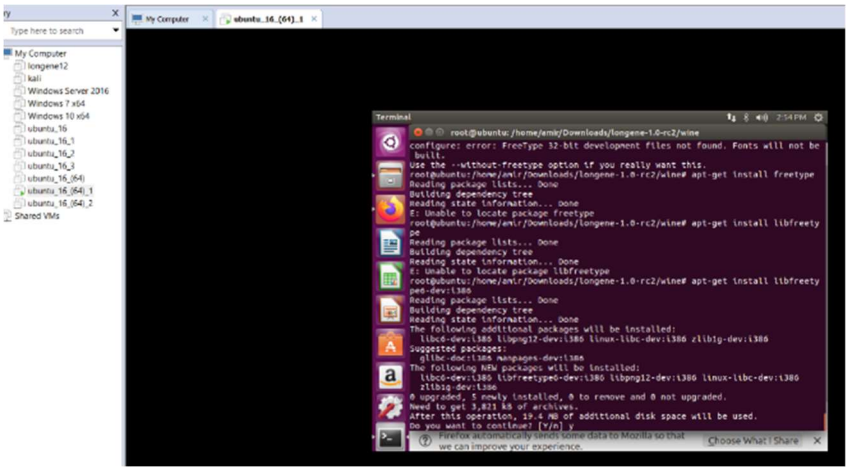


Nota. Esta imagen muestra la pantalla al intentar ./configure, falta la librería freetype. (Diseño Propio)

Confirmación e instalación de paquetes sugeridos para freetype .

Figura 83

Instalación de freetype

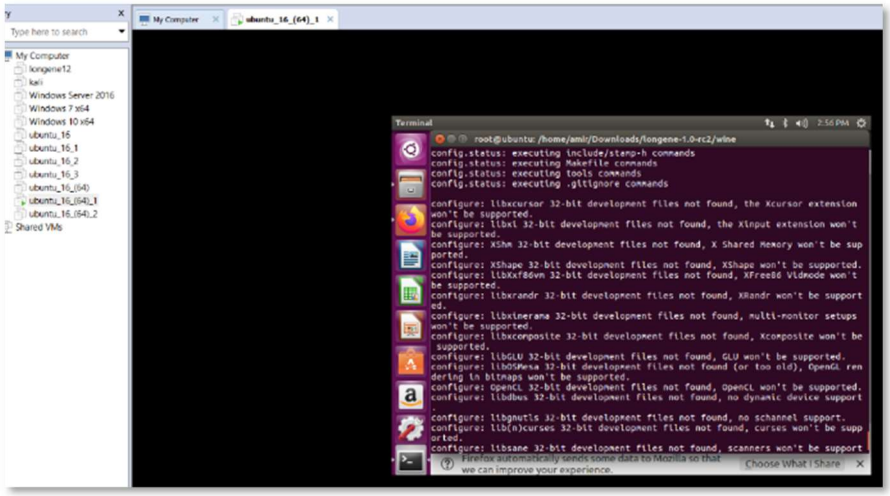


Nota. Esta imagen muestra la pantalla al instalar la librería freetype. (Diseño Propio)

Se intenta una vez más la ejecución de ./configura de Longene con el último paquete de freetype instalado, pero se encontraron con errores aún.

Figura 84

Error librerías faltantes



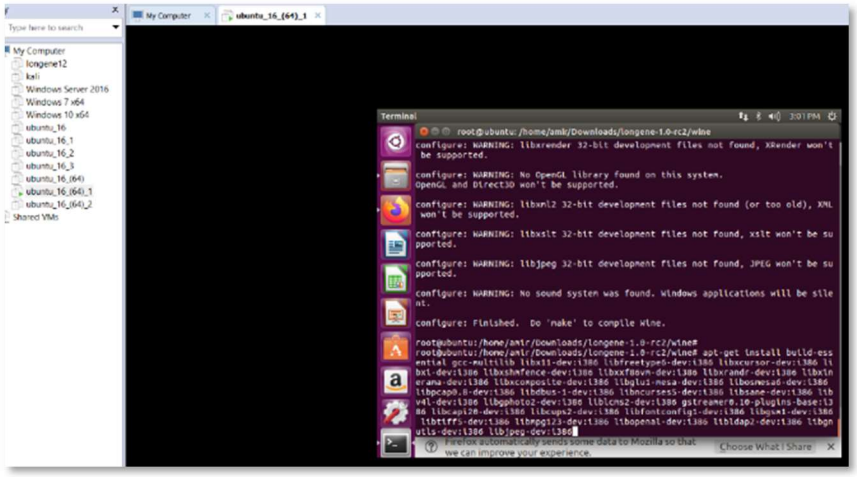
Nota. Esta imagen muestra la pantalla al intentar ./configure nuevamente, se requiere la instalación de librerías adicionales. (Diseño Propio)

Después de varias pruebas e intentos se creó el siguiente comando de ejecución para la instalación de las librerías necesarias.

sudo apt-get install build-essential gcc-multilib libx11-dev:i386 libfreetype6-dev:i386 flex bison libxcursor-dev:i386 libxi-dev:i386 libxshmfence-dev:i386 libxrandr-dev:i386 libxcomposite-dev:i386 libosmesa6-dev:i386 libpcap0.8-dev:i386 libdbus-1-dev:i386 libncurses5-dev:i386 libsane-dev:i386 libv4l-dev:i386 libgphoto2-dev:i386 liblcms2-dev:i386 gstreamer0.10-plugins-base:i386 libcapi20-dev:i386 libcups2-dev:i386 libfontconfig1-dev:i386 libgsm1-dev:i386 libtiff5-dev:i386 libmpg123-dev:i386 libopenal-dev:i386 libldap2-dev:i386 libgnutls-dev:i386 libjpeg-dev:i386 libglu1-mesa-dev:i386 libxinerama-dev:i386 libxxf86vm-dev:i386

Figura 85

Instalación librerías varias

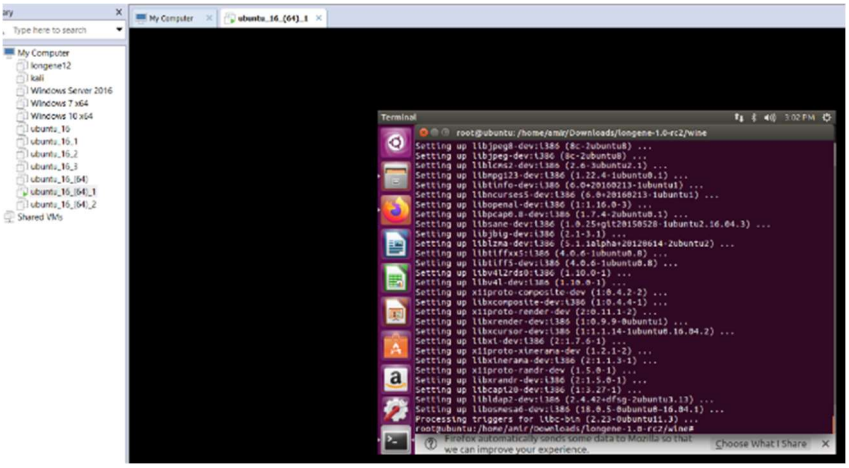


Nota. Esta imagen muestra la pantalla para instalar librerías varias. (Diseño Propio)

Lo siguiente fue obtener los resultados de instalación de librerias adicionales.

Figura 86

Resultado instalación de librerías varias

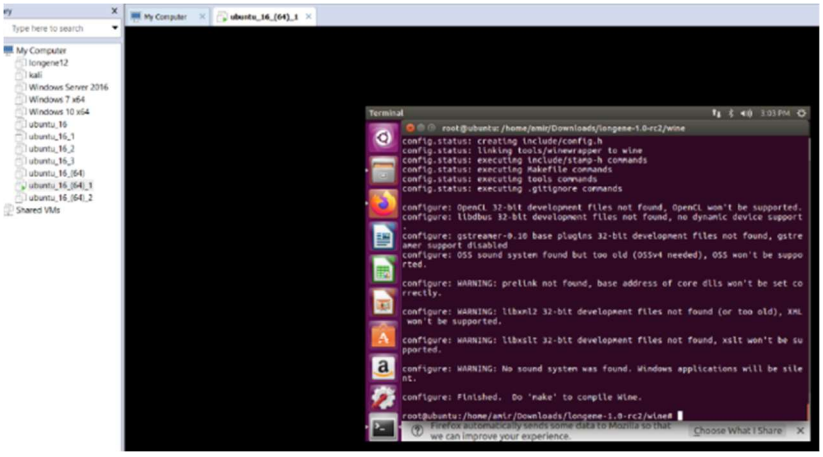


Nota. Esta imagen muestra la pantalla resultante al instalar librerías varias. (Diseño Propio)

Luego de haber instalado las librerías mencionadas, aún se encontraron advertencias por no encontrar la librería especifica. Como se muestra en la figura 87.

Figura 87

Resultado ejecución ./configure con librerías

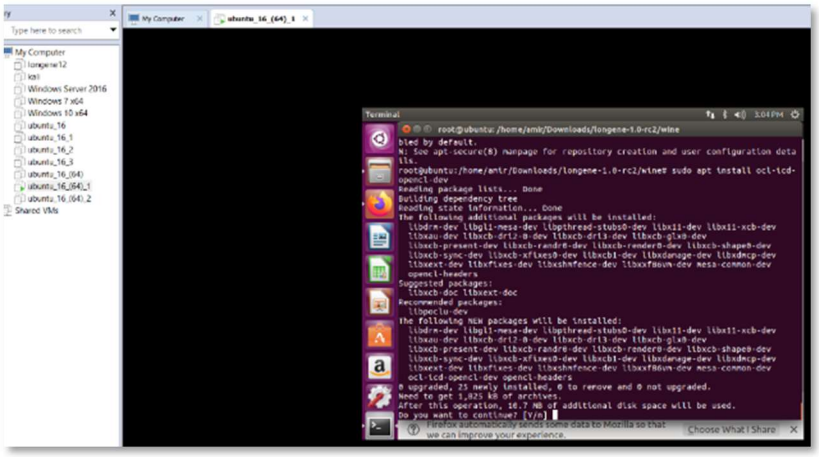


Nota. Esta imagen muestra la pantalla resultante al ejecutar ./configure luego de instalación de librerías. (Diseño Propio)

A continuación, se procede a instalar la librería opencl solicitada.

Figura 88

Instalación de OpenCL

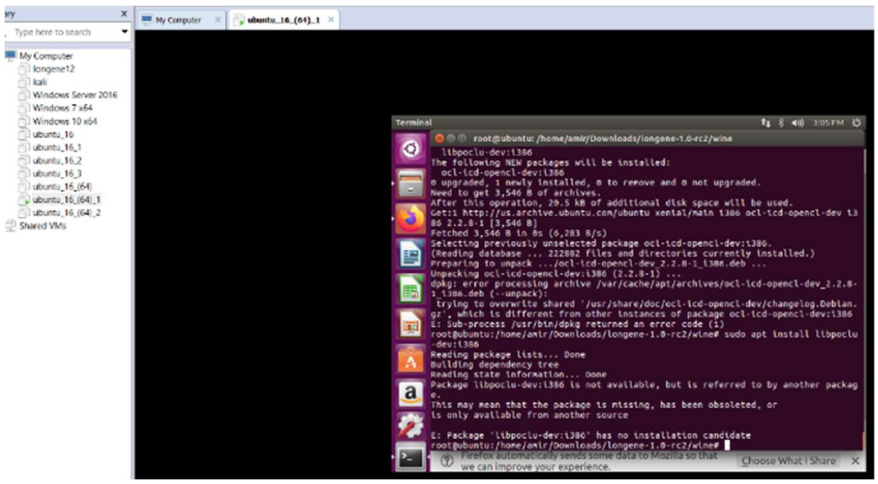


Nota. Esta imagen muestra la pantalla al instalar OpenCL. (Diseño Propio)

El intento de instalar la librería resulto en un error. Como se muestra en la figura 89

Figura 89

Error libpoclu no es candidato

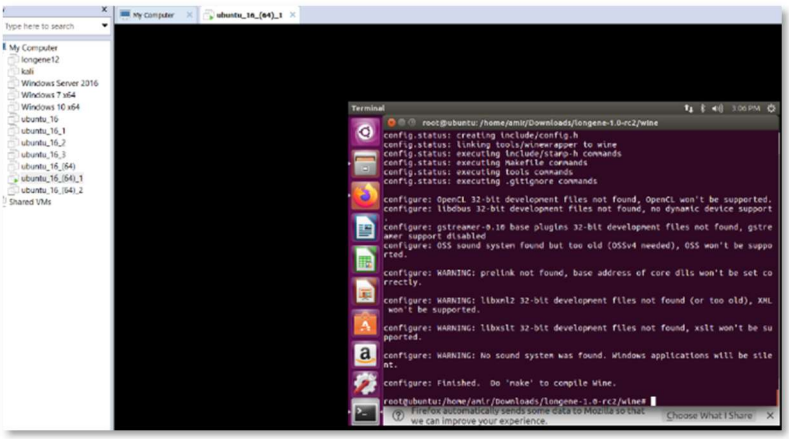


Nota. Esta imagen muestra la pantalla al error al instalar libpoclu por no ser candidato. (Diseño Propio)

A pesar de la gran cantidad de librerías instaladas aún existen librerías que fueron instaladas por el nombre o no son candidatas para instalar en Ubuntu 16 por lo que se procede a intentar el comando ./configura otra vez.

Figura 90

Nuevo intento ./configure



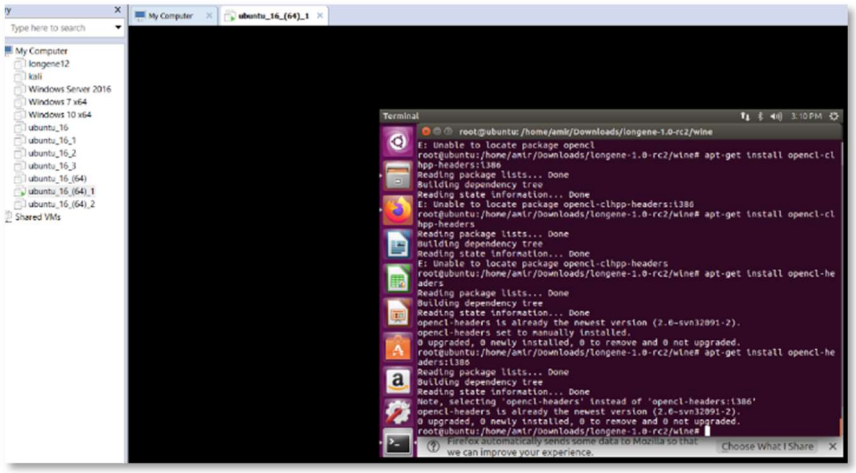
Nota. Esta imagen muestra la pantalla al intentar nuevamente la ejecución de ./configure. (Diseño Propio)

Como se observa en la figura 90 aún existen errores de librerías de 32 bits no encontradas, estas son OpenCL, gstreamer-0.10, libxml2 y libxslt. Aunque se intentó instalar la librería opencl-headers:i386 esta aparece actualizada pero aun así la librería no es candidata para la compilación en ./configure. A pesar de estos errores se permite la instalación del make de la carpeta /home/amir/Downloads/longene-1.0-rc2/wine dentro de longene.

Se visualiza que la librería opencl se encuentra actualizada.

Figura 91

Resultado de intentar instalar opencl

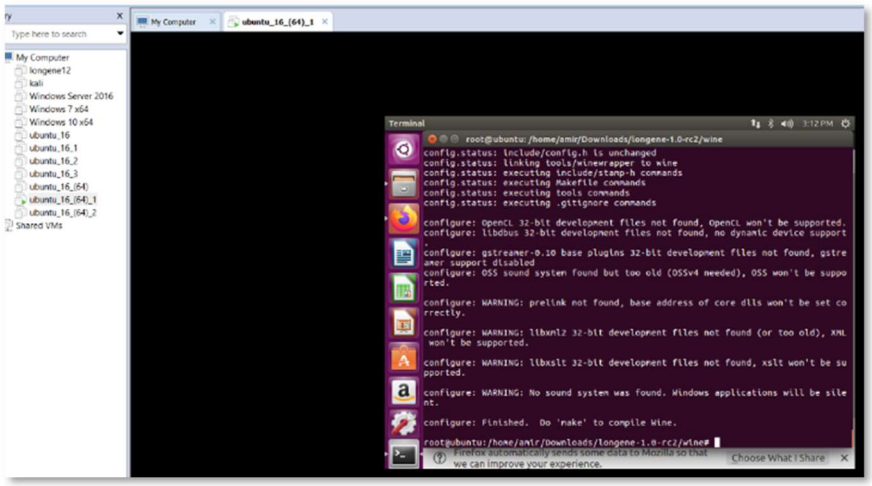


Nota. Esta imagen muestra la pantalla al intentar instalar opencl. (Diseño Propio)

Por lo que se intenta nuevamente el comando ./configura resultando en advertencias pero con permiso de make.

Figura 92

Resultado de intento de ./configure



Nota. Esta imagen muestra la pantalla resultante al intentar ./configure, permitiendo en esta ocasión make. (Diseño Propio)

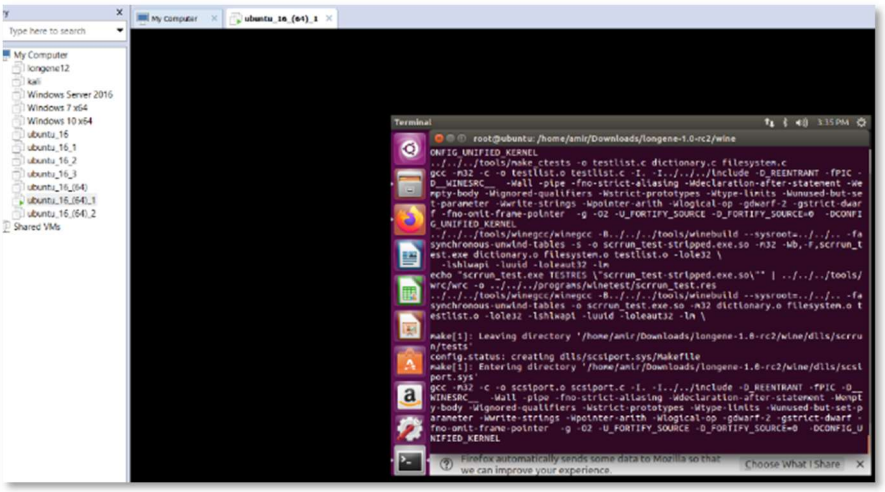
Teniendo el permiso de ejecución de make. Make es un comando que sirve para construir el software previamente desplegado con el comando ./configure, este ejecuta una serie de tareas definidas en

el archivo Makefile para construir el programa terminado a partir de su código fuente.

Por esto se procede con la ejecución del comando make.

Figura 93

Ejecución de comando make

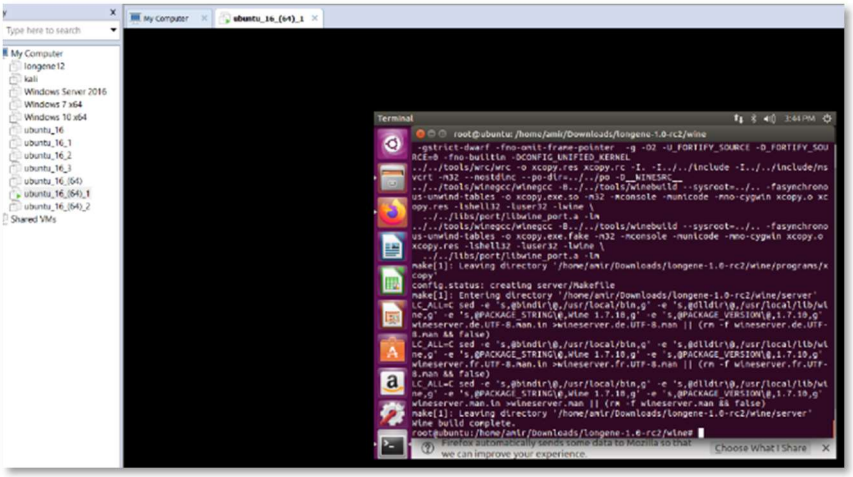


Nota. Esta imagen muestra la pantalla al intentar make. (Diseño Propio)

Después de esperar unos minutos, finalmente se obtiene el resultado de la ejecución de make. Reportando que la construcción de Wine esta completa.

Figura 94

Resultado comando make

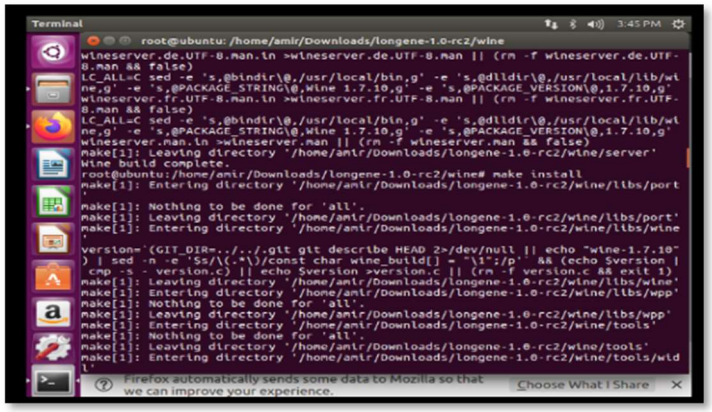


Nota. Esta imagen muestra la pantalla resultada de la ejecución de comando make. (Diseño Propio)

Al haber tenido una ejecución del comando **make** de forma normal y no reportar que se interrumpa la ejecución de este comando. Se procede con el comando make install. Make install es un comando copiará el programa construido, y sus bibliotecas y documentación, en las ubicaciones correctas.

Figura 95

Ejecución de make install

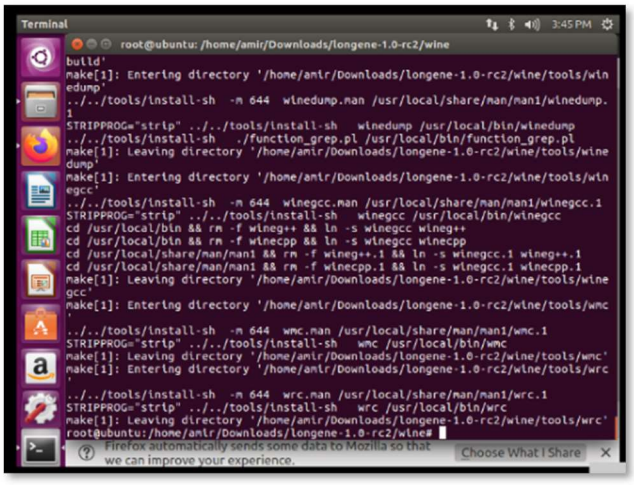


Nota. Esta imagen muestra la pantalla al ejecutar el comando de make install (Diseño Propio)

El resultado de la ejecución de make install es que este funciono sin problemas, por lo que no se paró la ejecución, al menos en los últimos comandos mostrados.

Figura 96

Resultado de ejecutar make install

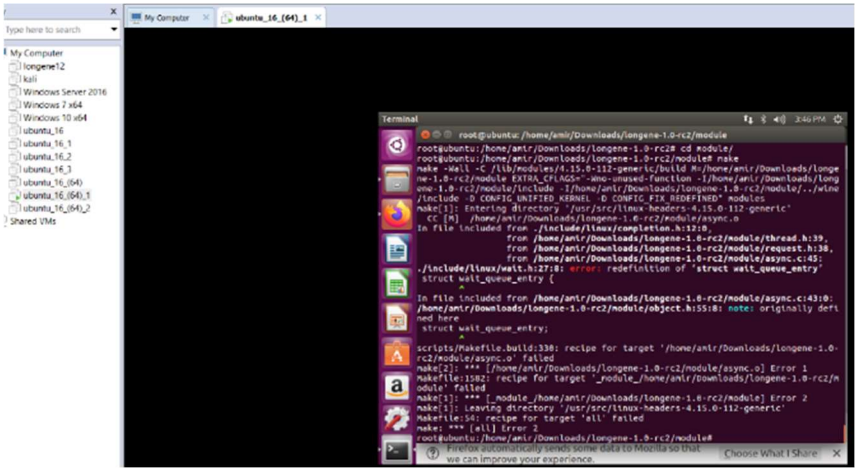


Nota. Esta imagen muestra la pantalla resultado de ejecución de comando de make install. (Diseño Propio)

Luego se procede con la siguiente parte de la instalación la cual se da en la carpeta /home/amir/Downloads/longene-1.0-rc2/module, usando el comando make.

Figura 97

Ejecutar make dentro de module



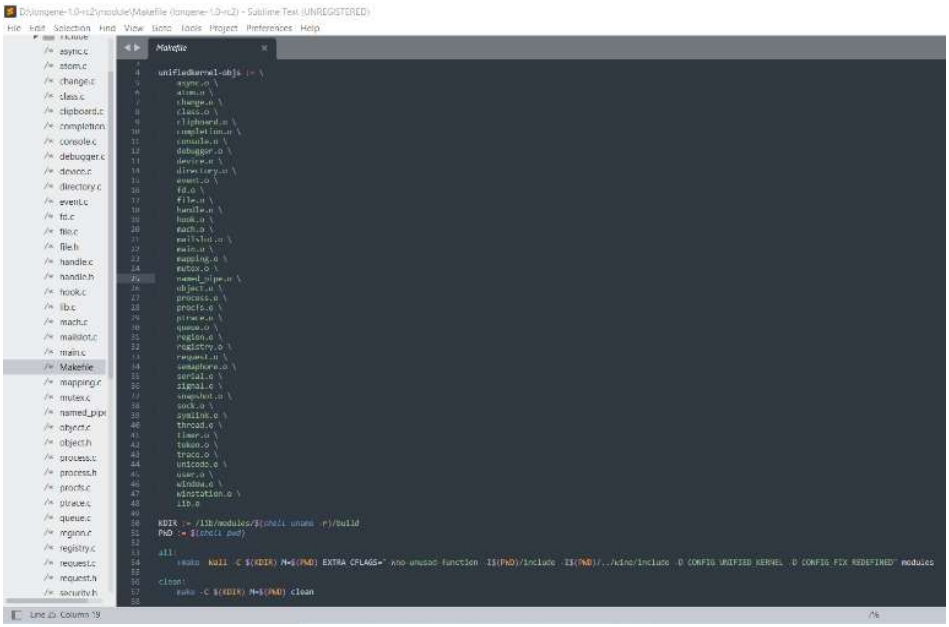
Nota. Esta imagen muestra la pantalla resultado de ejecución de comando de make en carpeta module. (Diseño Propio)

Como se aprecia en el paso anterior se tiene un error de la definición de la instrucción 'struct wait_queue_entry'. La localización de este archivo es rastreada hasta el directorio /usr/src/linux-headers-4.15.0-12-generic/include/wait.h. Después de la investigación sobre este directorio se encontró la siguiente información: "Debido a la variedad de hardware existente es posible que en tu distro Linux no funcione tal o cual cosa, por eso a veces es necesario compilar manualmente algún módulo del kernel o software diverso. A la hora de realizar estas tareas debemos estar preparados, y para eso necesitaremos tener instaladas algunas herramientas. *NOTA: Esto no es necesario si tu equipo está enfocado en un uso doméstico y todo te funciona bien (si funciona no lo toques) pero sí es interesante tenerlo en ciertas situaciones*. Los headers o cabeceras de kernel son necesarios, por ejemplo, para compilar drivers para luego activarlos en el kernel. Algunas distribuciones pueden traerlos instalados por defecto según el tipo de instalación que hayamos realizado." (Yoyo, 2019)

Por lo que se procedió a intentar sustituir los archivos los cuales generaban este error como por ejemplo wait.h, se tuvieron errores a pesar de que se saltaba esta línea, pero ocurrieron errores en todos y cada uno de los archivos dentro de la carpeta mencionada, por lo que esta opción fue descartada. Luego de esto se intentó instalar un modulo de kernel antiguo dentro de la carpeta /usr/src/, pero como se muestran en las figura 98 y figura 99 se hace referencia al comando uname -r. El comando uname -r muestra la versión que se esta ejecutando o release del kernel.

Figura 98

Script Makefile de Longene

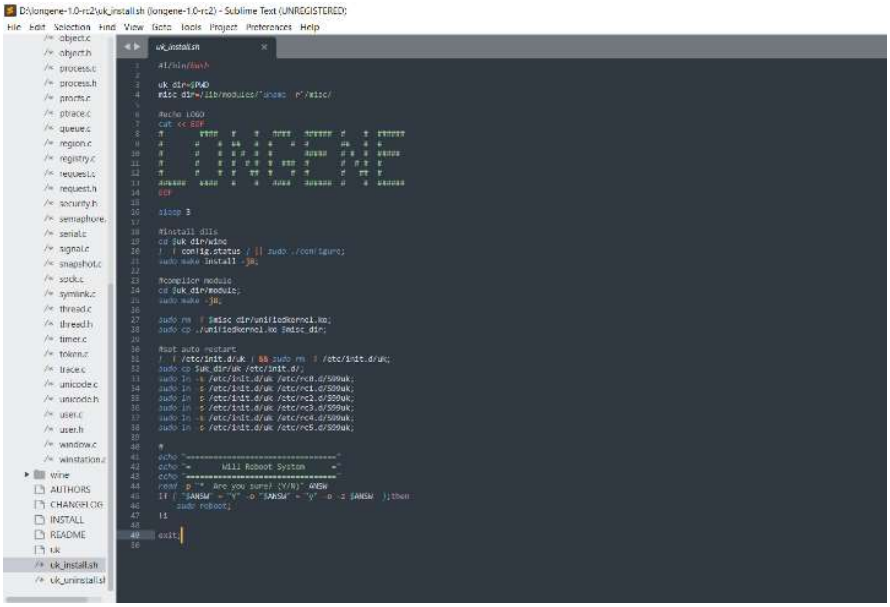


Nota. Esta imagen muestra la pantalla del script Makefile de Longene. (Diseño Propio)

A continuación, se ven comandos que se siguen para la instalación de Longene, siendo finalmente el cargar el módulo de Longene.

Figura 99

Script de autoinstalación de Longene



Nota. Esta imagen muestra script de autoinstalación de Longene. (Diseño Propio)

Después de tratar de referenciar al módulo antiguo instalado y que se pueda ejecutar la instrucción de make, se tuvo un error final el cual indicaba que la versión de gcc con la que se estaba tratando de ejecutar era gcc-5 y por eso no se podían completar las instrucciones. Ya que se tuvieron estos errores estas pruebas no fueron documentadas porque se consideraron irrelevantes debido a que no aportaban solución al problema presentado, por lo que estos errores implicaron que se reconsideren las opciones previamente probadas, y se decidió que se creara un nuevo entorno en el que se realizara la instalación de Longene teniendo presente que el entorno más optimo era un Ubuntu de 32 bits y se debía utilizar un kernel antiguo el cual tenga las estructuras previamente diseñadas donde puede transcurrir este proyecto de forma más natural.

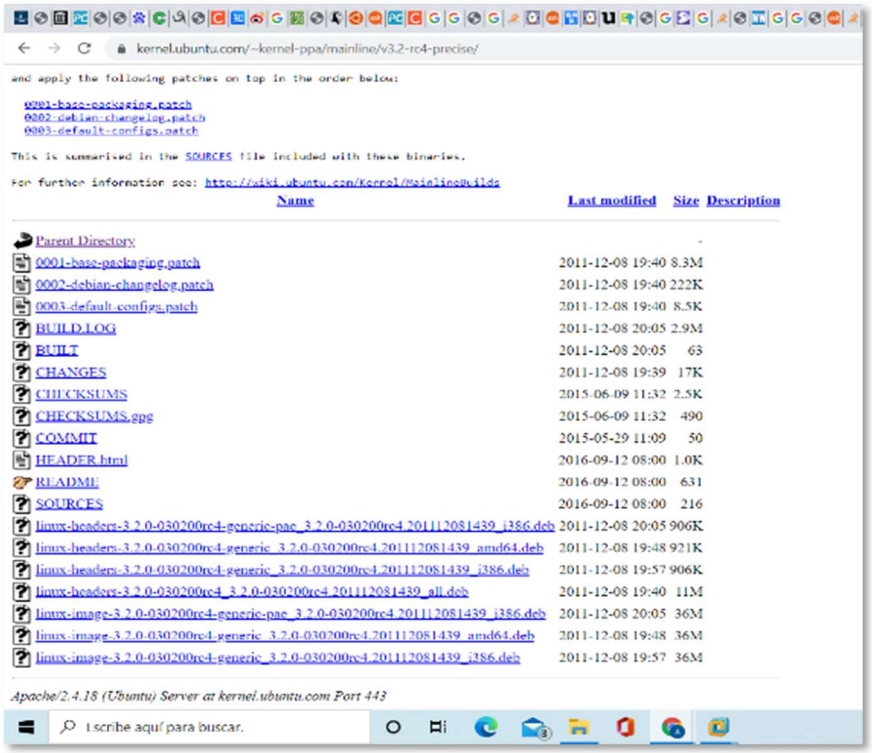
Para que se ejecute el proyecto Longene se considerara comenzar desde 0 con primero la instalación previa de un kernel antiguo haciendo referencia a los headers.

3.4.2. PRUEBA INSTALACIÓN DE LONGENE UBUNTU 32 BITS

Se encontró un servidor que contiene versiones antiguas del kernel de Linux el link de este repositorio es: https://kernel.ubuntu.com/~kernel-ppa/mainline/ . Y se decidió usar la versión 3.2-rc4-precise ya que es la versión más cercana a la versión por defecto de Ubuntu 12.04.

Figura 100

Imagen repositorio de headers

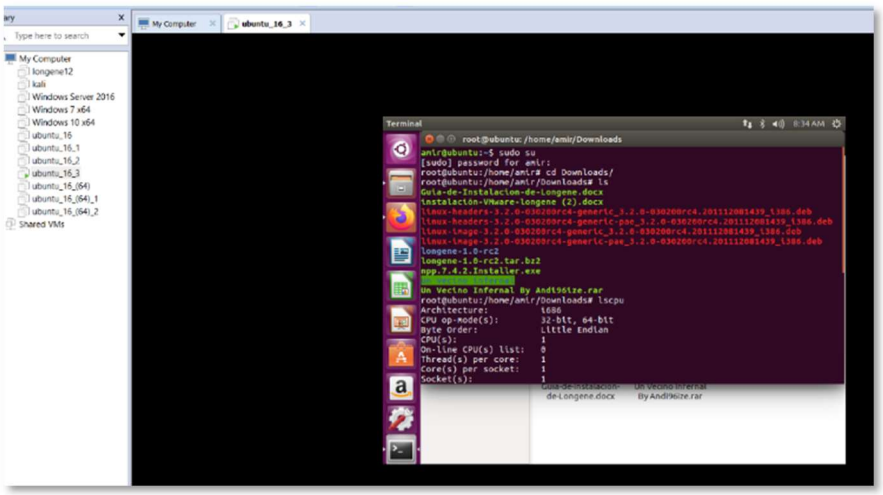


Nota. Esta imagen muestra la lista de archivos tomada del repositorio de headers. (Diseño Propio)

Para verificar que se está en la versión Ubuntu 16.04 de 32 bits se verificara con el comando lscpu, apareciendo primero la arquitectura i686 correspondiente a la versión en la que se implementara en esta ocasión.

Figura 101

Iscpu de la nueva máquina de pruebas

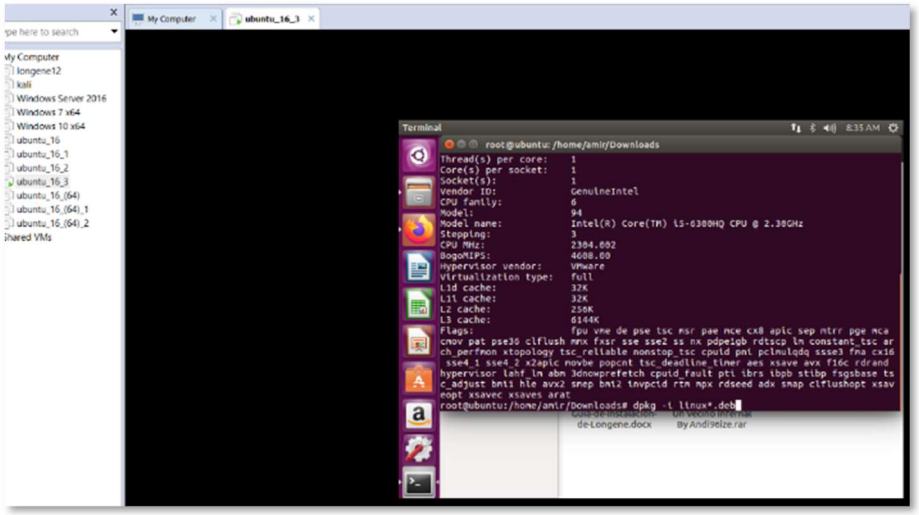


Nota. Esta imagen muestra lscpu de nueva máquina. (Diseño Propio)

Segunda parte de la ejecución de el comando lscpu.

Figura 102

Iscpu nueva máquina parte 2

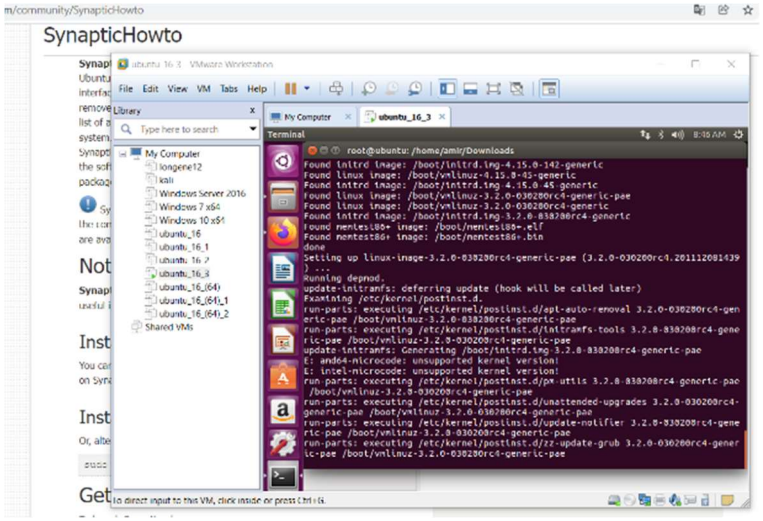


Nota. Esta imagen muestra la parte 2 de la pantalla al hacer lscpu de la nueva máquina de pruebas. (Diseño Propio)

Se procede con la descarga de los paquetes necesarios para la instalación de este kernel antiguo. Primero se descargaron los archivos y luego se instalaron con el comando dpkg -i linux-* este comando sirve para instalar todos los archivos .deb que comiencen los caracteres “linux-”.

Figura 103

Instalación de archivos .deb

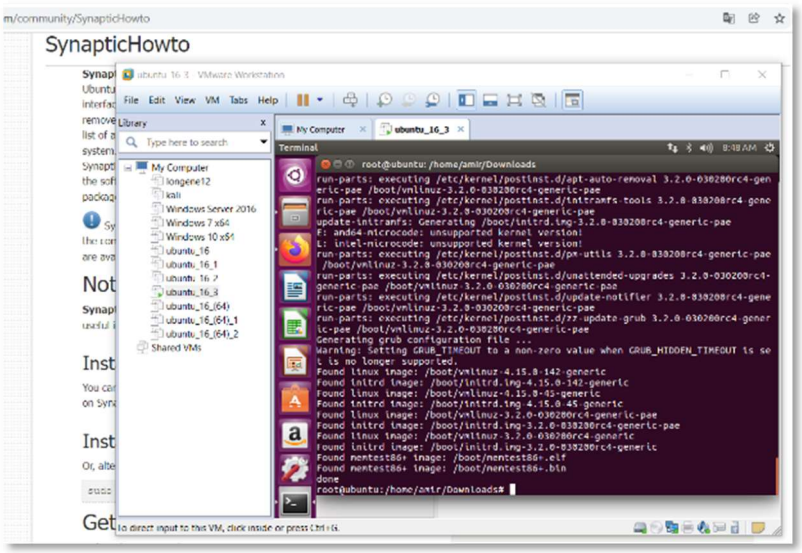


Nota. Esta imagen muestra la instalación de archivos .deb. (Diseño Propio)

Después de la instalación de los archivos con dpkg descargados del repositorio, se encontraron algunos errores al momento de instalar, además de una alerta de sobre el grub.

Figura 104

Resultado de instalación de dpkg



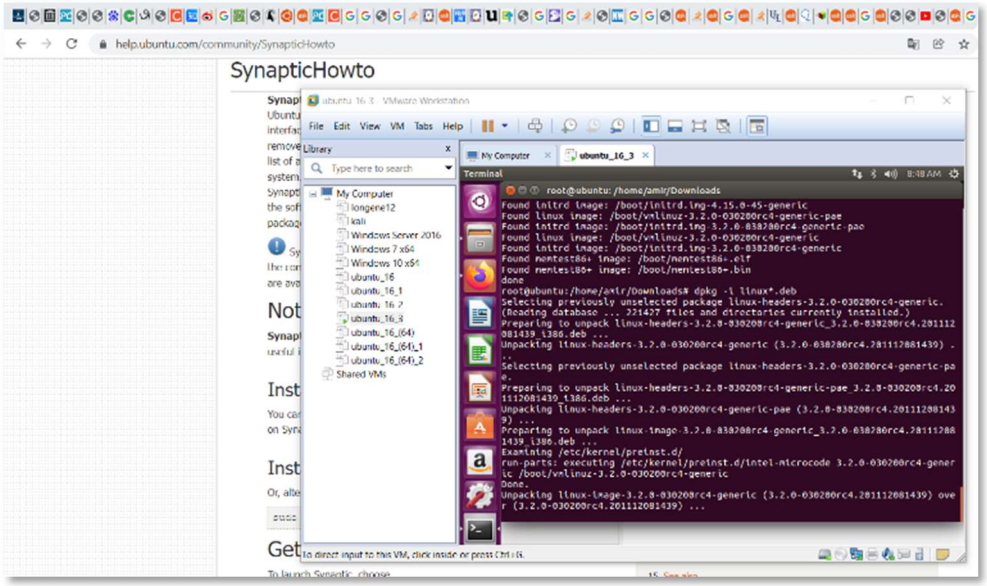
Nota. Esta imagen muestra el resultado de instalación de dpkg. (Diseño Propio)

Ya que se encontraron algunos errores de la inexistencia de dependencias solicitadas al solo descargar la imagen y header de

una versión especifica por no descargar todos los paquetes asociados a i386, por esta razón se descargaron todos los headers generic/generic-pae, images generic/generic-pae y también el paquete linux-headers-3.2.0-030200rc4_3.2.0-030200rc4.201112081439_all.deb como se muestra en la figura 103, este paquete permite crear los modulos de makefile dentro de la carpeta /usr/include/$version ya que hace recibe la referencia donde se ejecutarán las instrucciones venidas de la selección de kernel compilado uname -r con los respectivos argumentos.

Figura 105

Instalación dpkg con pae

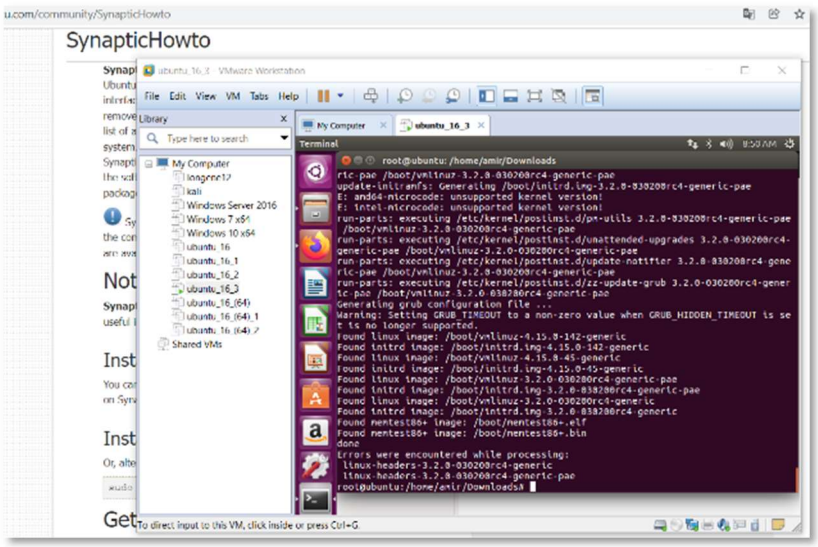


Nota. Esta imagen muestra la instalación de dpkg con pae (Diseño Propio)

El resultado de la instalación, resulto en un error porque no se podían procesar los headers específicos.

Figura 106

Resultado instalacion dpkg

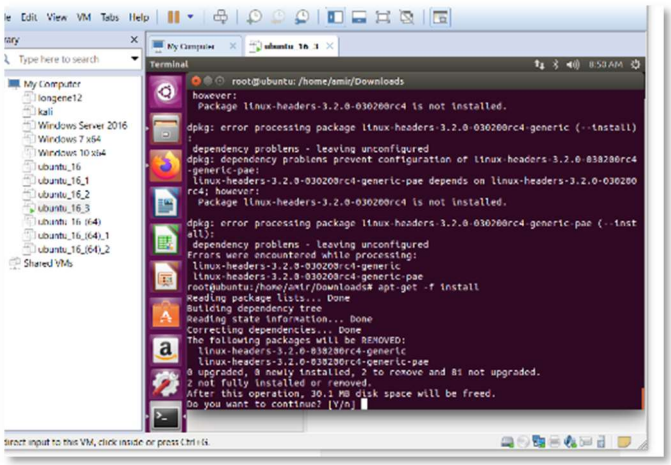


Nota. Esta imagen muestra el resultado instalación de dpkg. (Diseño Propio)

Ya que aún existían errores en la instalación de los paquetes se usó el comando apt-get -f install que permite forzar la instalación instalando las dependencias que soliciten los paquetes.

Figura 107

apt-get -f install

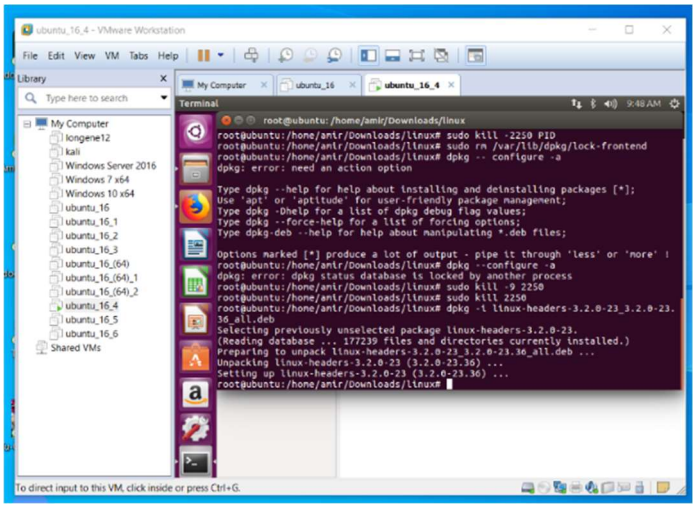


Nota. Esta imagen muestra la ejecución de apt-get -f install. (Diseño Propio)

Después de forzar la instalación de los paquetes, se puede instalar el paquete linux-headers-3.2.0-030200rc4_3.2.0-030200rc4.201112081439_all.deb se visualiza que se puede realizar la instalación normalmente como se muestra en la figura 108.

Figura 108

Instalación de paquete all_deb

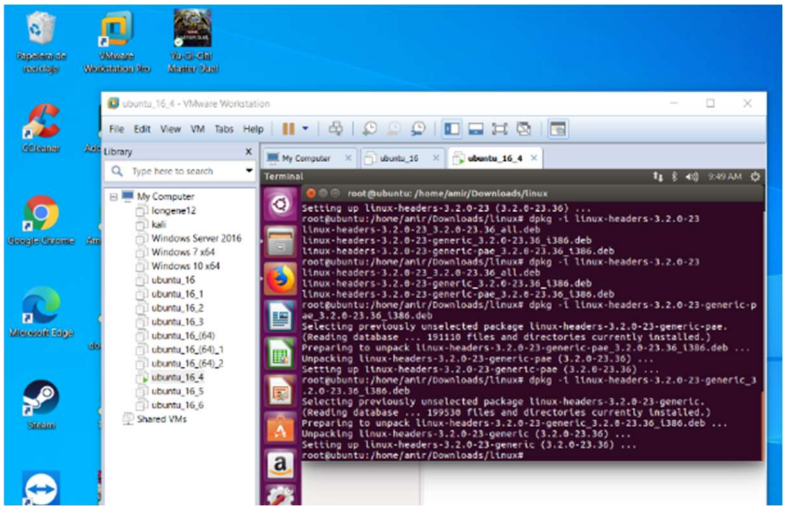


Nota. Esta imagen muestra la instalación de paquetes all_deb. (Diseño Propio)

Después de la instalación de all_deb se proceden a instalar los paquetes dpkg uno por uno.

Figura 109

Instalación paquetes dpkg

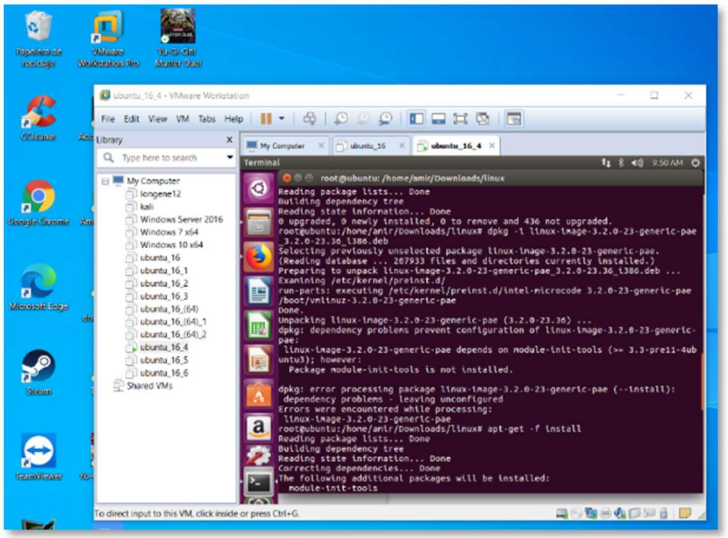


Nota. Esta imagen muestra la instalación paquetes dpkg. (Diseño Propio)

Usando el comando apt-get -f install dentro del directorio /home/amir/Downloads/linux donde se movieron las descargas de los paquetes referentes al kernel antiguo se nos recomienda instalar el módulo module-init-tools, por lo que se aceptó esta instalación de este paquete.

Figura 110

Solicitud de paquete module-init

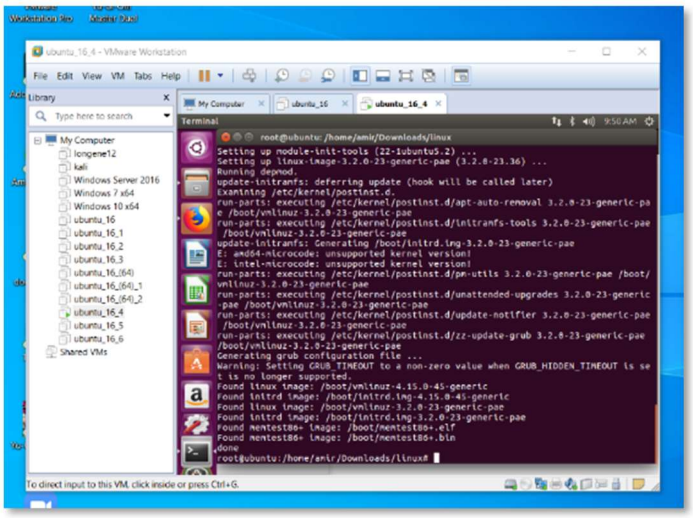


Nota. Esta imagen muestra la solicitud de paquete module-init. (Diseño Propio)

Se visualiza que la instalación fue realizada correctamente como se muestra en la ilustración 111, por lo que se procede a revisar las carpetas del kernel antiguo instalado, esto se ve en el directorio /usr/include.

Figura 111

Resultado instalación paquete module

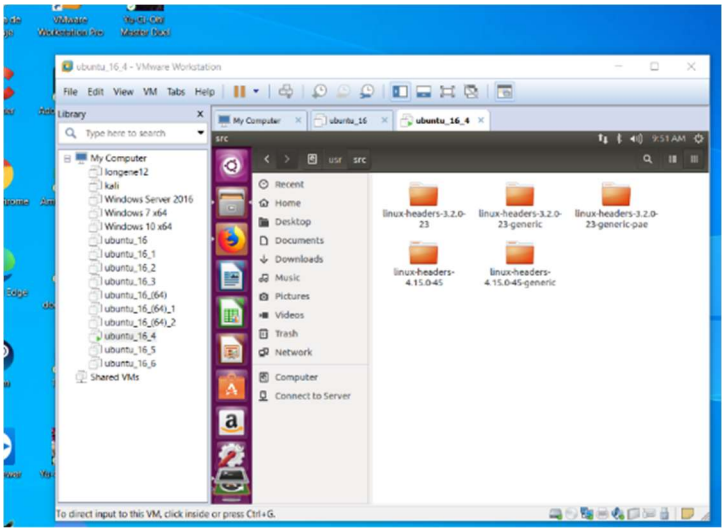


Nota. Esta imagen muestra el resultado de instalación del paquete module. (Diseño Propio)

Se verifican las carpetas de la ruta /usr/src, donde se visualiza la existencia de la versión 3.2.0

Figura 112

Carpeta /usr/src



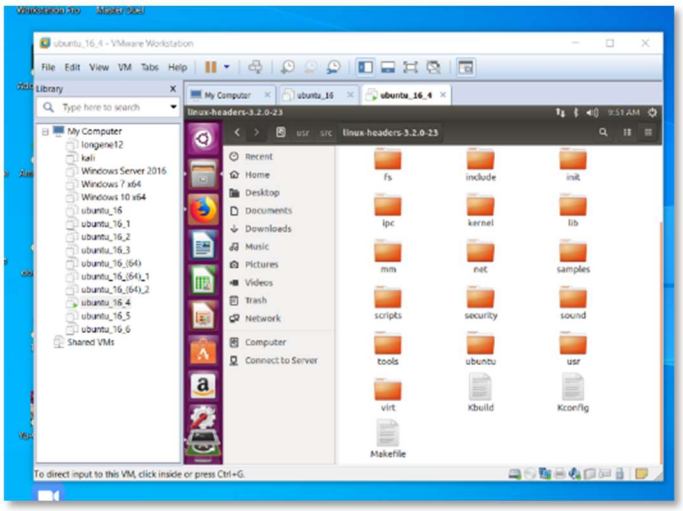
Nota. Esta imagen muestra la nueva lista de archivos de carpeta /usr/src. (Diseño Propio)

Dentro de la carpeta linux-headers-$version, se encuentran las carpetas pertinentes y archivos como Makefile, kbuild y kconfig, los

cuales permiten la ejecución de algún programa en este kernel antiguo véase figura 113.

Figura 113

Carpeta /usrc/src/linux-headers-3.2.0-23

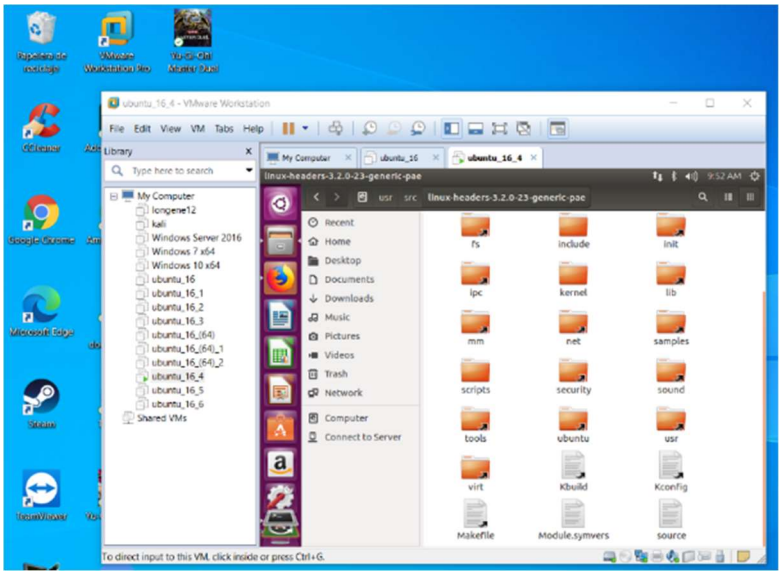


Nota. Esta imagen muestra los archivos de /usrc/src/linux-headers-3.2.0-23. (Diseño Propio)

Se puede observar que en el caso del kernel el cual se cargara en el grub se tienen también los archivos Makefile, kbuild y kconfig, pero estos archivos no son archivos existentes, simplemente se trata de vinculaciones o conocidos comúnmente como links. “Un enlace simbólico (también conocido como enlace suave o enlace simbólico) consiste en un tipo especial de archivo que sirve como referencia a otro archivo o directorio. Los sistemas operativos similares a Unix/Linux a menudo usan enlaces simbólicos. Esta guía explica cómo usar el comando ln para crear enlaces simbólicos/suaves.” (Gite, 2020). Estos archivos link no existirían si no se hubiera instalado el modulo *_all.deb mencionado antes.

Figura 114

Archivos carpeta linux-headers--generic-pae*

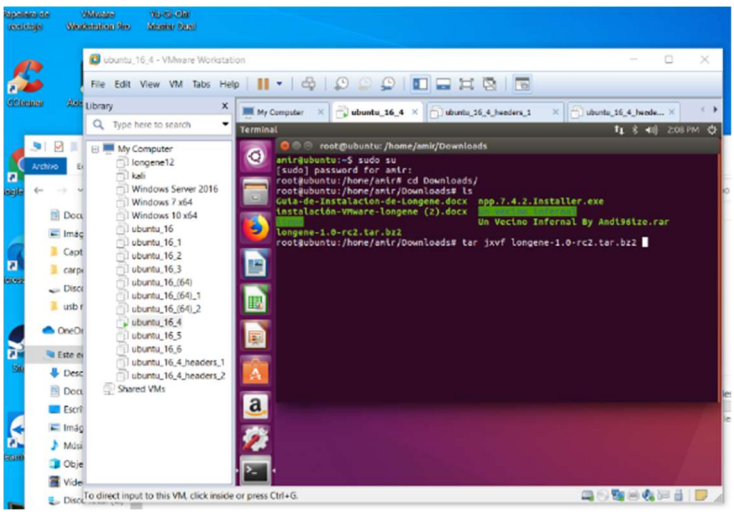


Nota. Esta imagen muestra los archivos de la carpeta linux-headers-*-generic-pae. (Diseño Propio)

Se procede con la descompresión del tar de Longene en la carpeta /home/amir/Downloads véase figura 115.

Figura 115

Descompresion Longene

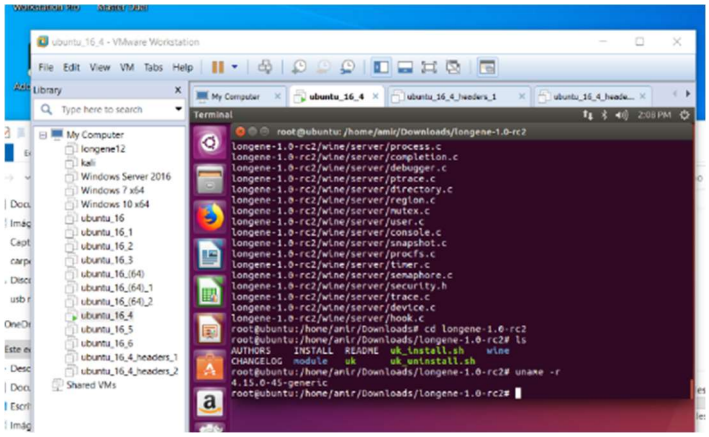


Nota. Esta imagen muestra el comando de descompresión de Longene. (Diseño Propio)

Después de la descompresión se verifica la versión actual del kernel que se tiene montada con el comando uname-r, pero aún se visualiza que la versión actual es la 4.15.0-45-generic véase figura 116.

Figura 116

Verificar versión con uname -r

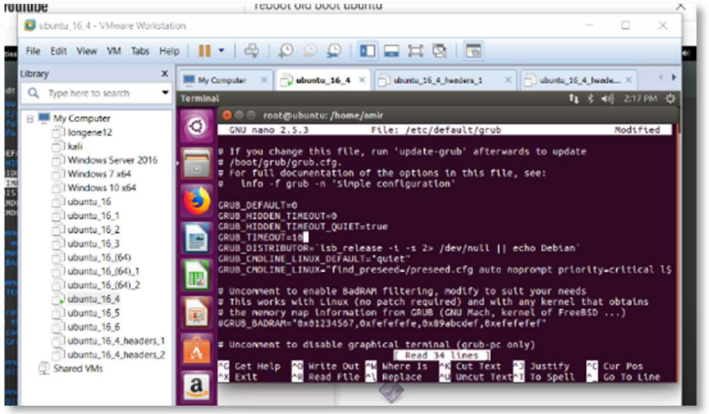


Nota. Esta imagen muestra el verificar la versión con comando uname -r (Diseño Propio)

Como ha sido visto previamente visto en la figura 116, por lo que al tratar de configurar **/etc/default/grub** y darle un update sigue apareciendo un error, muestra un error de Warning: Setting GRUB_TIMEOUT que HIDDEN_TIMEOUT no es soportado véase figura 117, por lo que se tiene que hacer una configuración más.

Figura 117

Configuración /etc/default/grub

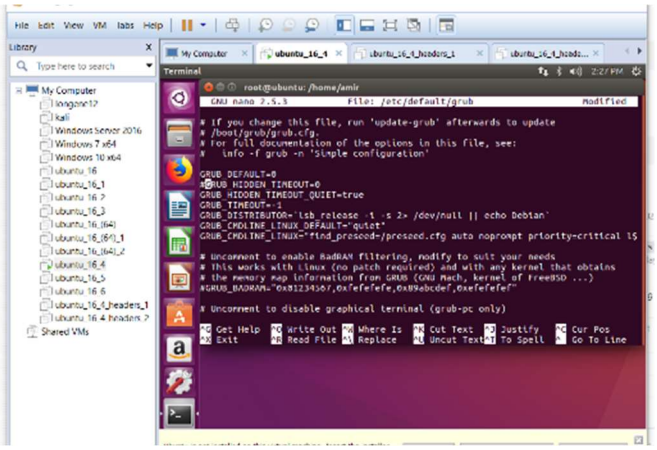


Nota. Esta imagen muestra la configuración del /etc/default/grub. (Diseño Propio)

Para esto accedo al archivo **/etc/default/grub**, en este archivo declaro como comentario GRUB_HIDDEN_TIMEOUT y a GRUB_TIMEOUT le doy un valor de -1 para que espere la indicación de con que kernel arrancar véase figura 118. (ATOM, Ubuntu Linux How to set older kernel as default in GRUB, 2017). Luego de esto se realiza un sudo upgrade-grub.

Figura 118

Configuración Grub

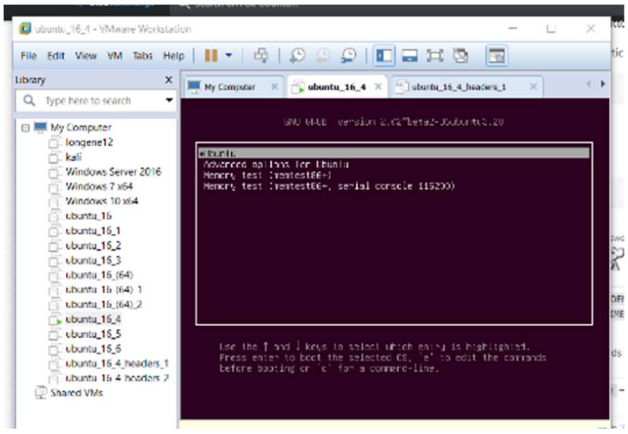


Nota. Esta imagen muestra la configuración del Grub. (Diseño Propio)

Al momento de reiniciar la maquina ya aparece la pantalla del grub, esta pantalla nos permite elegir con que versión de kernel queremos compilar el sistema operativo. Véanse la figura 119.

Figura 119

Primera pantalla carga de Grub

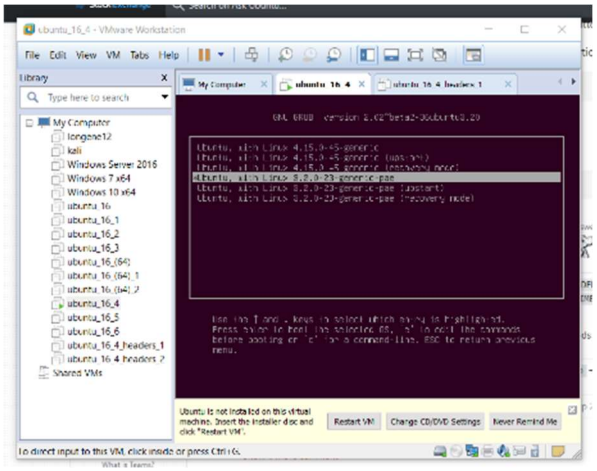


Nota. Esta imagen muestra la primera pantalla de Grub. (Diseño Propio)

La siguiente pantalla que nos muestra son las opciones de kernel de grub que deseamos cargar por lo que elegimos cargar el modulo 3.2.0-23-generic-pae.

Figura 120

Pantalla de elección de kernel grub

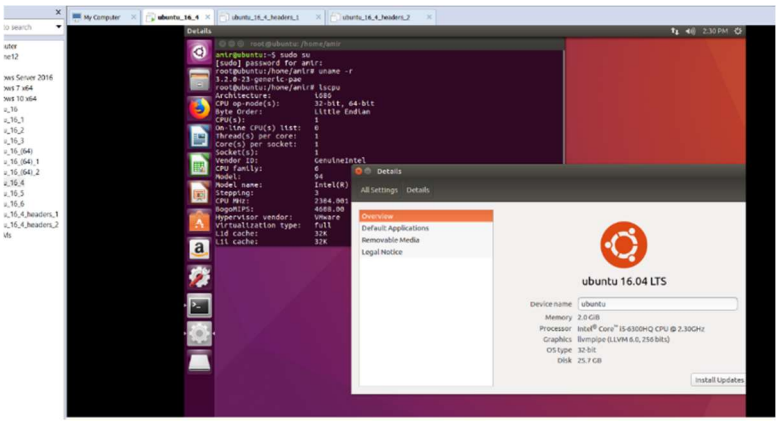


Nota. Esta imagen muestra pantalla de elección de kernel grub. (Diseño Propio)

A primera vista se visualiza que la vista del VMware se hizo más grande, esto es debido a que no esta funcionando con un kernel nativo de la versión actual del sistema operativo, pero aun así se está ejecutando el GUI (Graphical user interface/Interfaz gráfica de usuario) correspondiente. Y se mantiene la versión de Ubuntu 16.04 y el tipo de sistema es de 32 bits.

Figura 121

Vista ejecución con nuevo grub

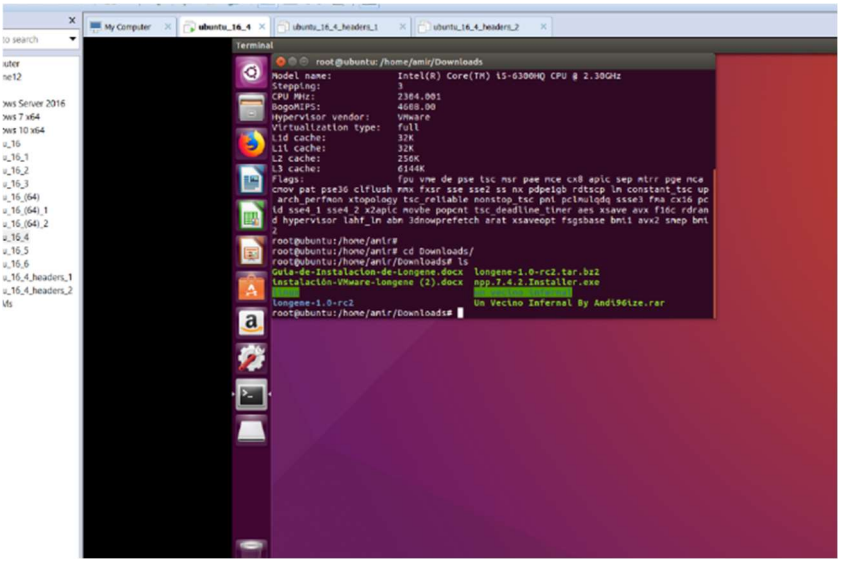


Nota. Esta imagen muestra ejecución con nuevo grub. (Diseño Propio)

A continuación, se muestra la información faltante del comando lscpu y los archivos que se siguen teniendo.

Figura 122

Segunda pantalla de lscpu y archivos de Downloads

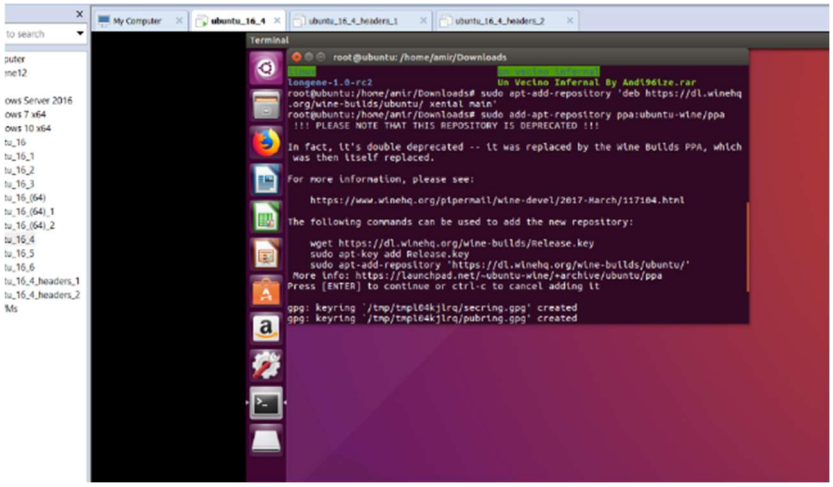


Nota. Esta imagen muestra archivos de carpeta downloads y segunda pantalla de lscpu. (Diseño Propio)

Para continuar con este nuevo entorno generado se procede con los pasos anteriores mencionados. Como el agregar el repositorio de Wine.

Figura 123

Agregar repositorio ppa:ubuntu

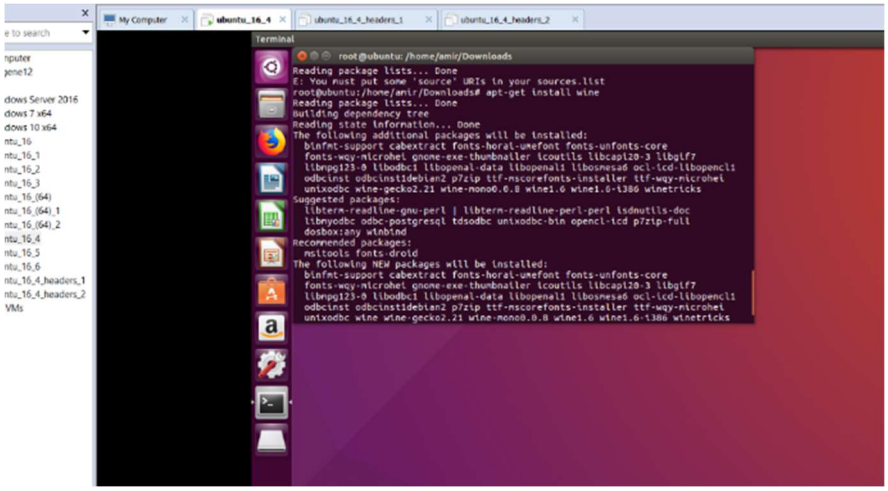


Nota. Esta imagen muestra el agregar repositorio ppa:ubuntu. (Diseño Propio)

Después de ejecutar un apt-update se procede con la instalación de la versión Wine 1.6

Figura 124

Instalación wine 1.6 por defecto

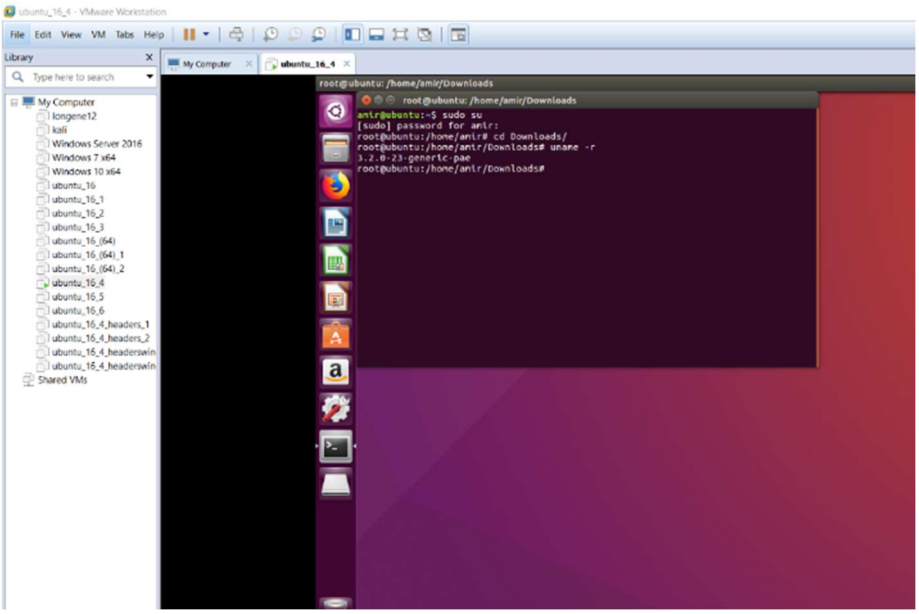


Nota. Esta imagen muestra la instalación de wine 1.6 de repositorio por defecto. (Diseño Propio)

Para comprobar la versión de kernel que se está ejecutando se confirma con el comando uname-r, por lo que se muestra que efectivamente se usa el kernel 3.2.0-23-generic-pae.

Figura 125

Ejecución uname -r

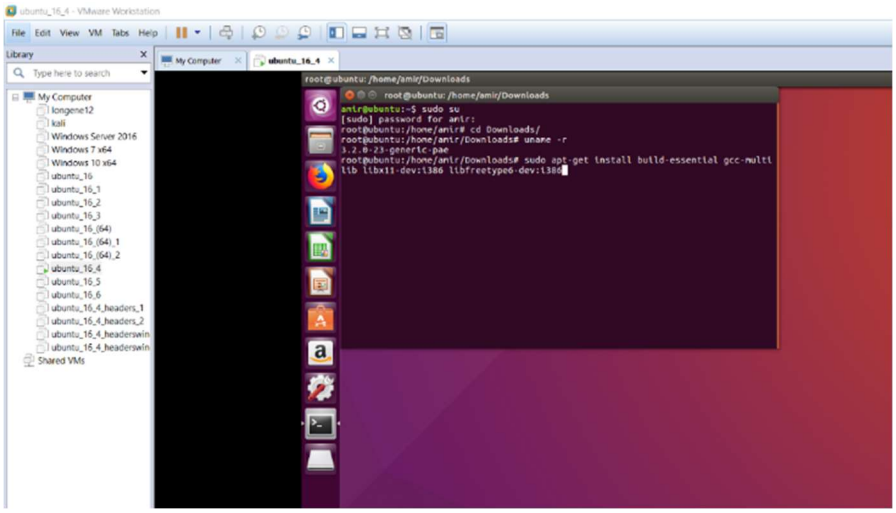


Nota. Esta imagen muestra la ejecución uname -r, comando para ver el kernel de uso actual. (Diseño Propio)

Se procede a la instalación de las primeras 4 librerías documentadas y la librería bison, ya que muchas de las anteriores no eran candidatas porque estaban obsoletas o ya no existían sus repositorios.

Figura 126

Instalación de librerías consideradas indispensables



Nota. Esta imagen se muestra instalación de librerías consideradas indispensables. (Diseño Propio)

Para realizar la nueva instalación de Longene en esta máquina con el grub modificado esta vez se usará una definición en el comando bastante interesante, presentados en la siguiente imagen.

Figura 127

Comandos de ejecución en archivos txt

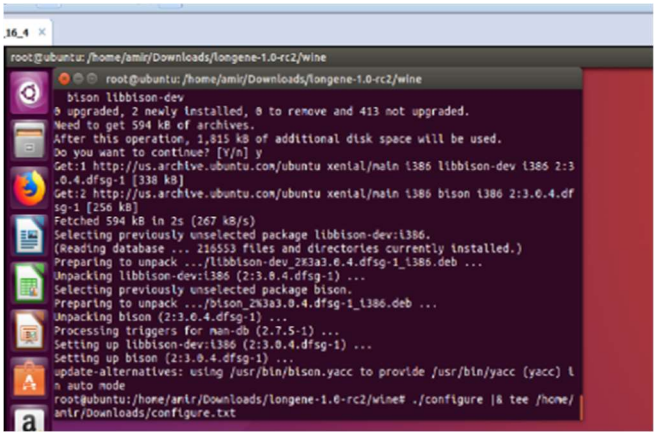
Syntax	visible in terminal: StdOut	visible in terminal: StdErr	visible in file: StdOut	visible in file: StdErr	existing file
>	no	yes	yes	no	overwrite
>>	no	yes	yes	no	append
2>	yes	no	no	yes	overwrite
2>>	yes	no	no	yes	append
&>	no	no	yes	yes	overwrite
&>>	no	no	yes	yes	append
| tee	yes	yes	yes	no	overwrite
| tee -a	yes	yes	yes	no	append
n.e. (*)	yes	yes	no	yes	overwrite
n.e. (*)	yes	yes	no	yes	append
|& tee	yes	yes	yes	yes	overwrite
|& tee -a	yes	yes	yes	yes	append

Nota. Esta imagen se muestra comandos para guardar la ejecución en archivos txt. (Commander♦, 2016)

Esta tabla muestra los comandos usados para hacer un output de la ejecución de la terminal en un archivo que nosotros definamos y en una ubicación que elijamos, por lo que usare "**|& tee**" con esto se puede ver la ejecución del programa en consola y almacenar todas las instrucciones generadas en un archivo .txt en la ruta /home/amir/Downloads como se visualiza en la figura 128, 129 y 130. En esta ocasión para una mejor visualización pondré la imagen del comando y al lado derecho la salida de ejecución para en la imagen debajo de estas 2 mostrar el resultado en el archivo creado automáticamente y la salida de la terminal.

Figura 128

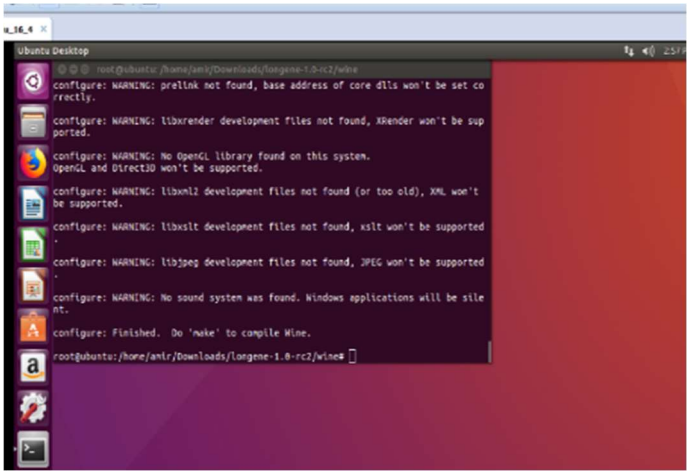
./configure con |& tee



Nota. Ejecución ./configure con comando con |& tee. (Diseño Propio)

Figura 129

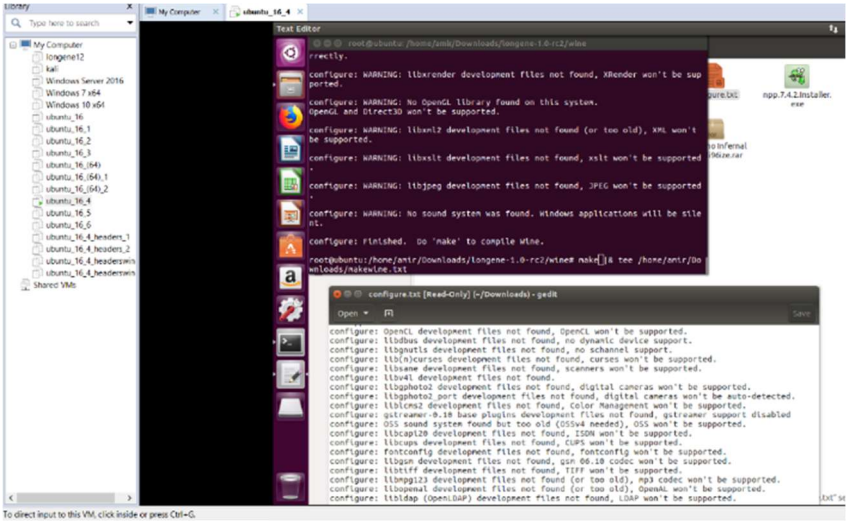
Resultado de comando con |& tee



Nota. Resultado ./configure con |& tee. (Diseño Propio)

Figura 130

Archivo resultado ./configure con |& tee

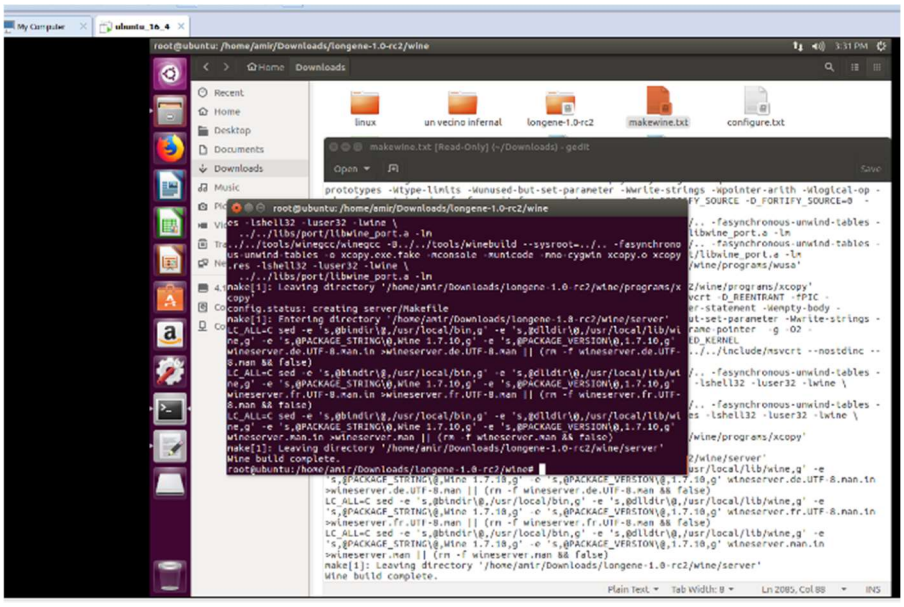


Nota. Esta imagen es el archivo resultado de ./configure con |& tee. (Diseño Propio)

Para el siguiente caso se realiza lo mismo pero esta vez haciendo el comando make dentro de la carpeta /home/amir/Downloads/longene-1.0-rc2/wine.

Figura 131

Ejecución de comando make

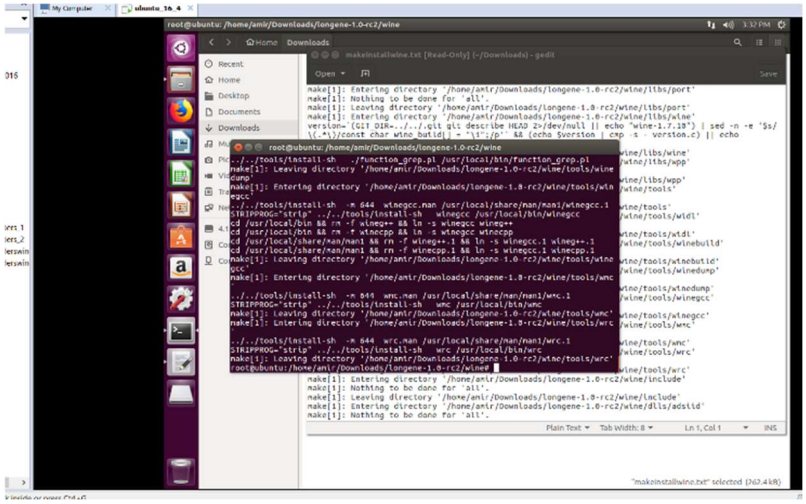


Nota. Esta imagen es el resultado de la ejecución de comando make. (Diseño Propio)

Como en los casos anteriores se realiza la instalación con la misma salida, pero esta vez el comando es **make install.**

Figura 132

Ejecución de make install

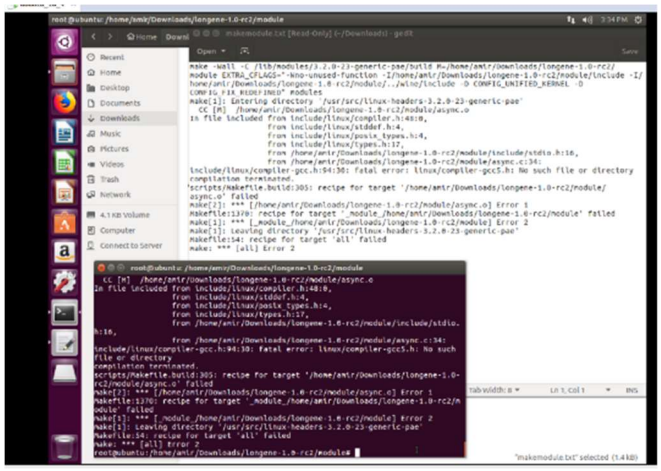


Nota. Esta imagen es el resultado de la ejecución de comando make install. (Diseño Propio)

Como en los casos anteriores se realiza la instalación con la misma salida, pero esta vez el comando es **module.** En esta ocasión se tiene un error, esto es debido a que al instalar las bibliotecas se instalaron build-essential y gcc-multilib, pero al ser Ubuntu 16 la librería candidata por defecto en estos casos es gcc-5. El error indica que gcc-5 no puede compilar.

Figura 133

Error por gcc-5



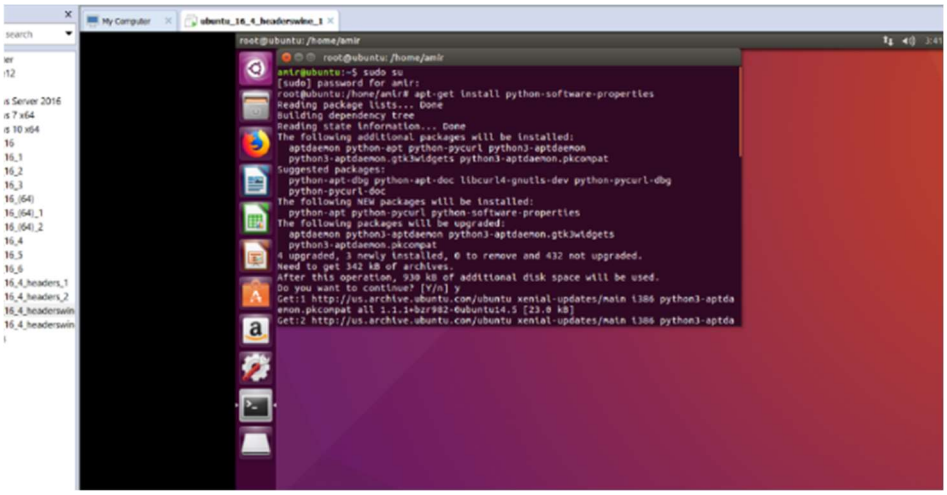
Nota. Esta imagen muestra error por gcc-5. (Diseño Propio)

3.4.3. PRUEBA DE INSTALACIÓN DE LONGENE EN UBUNTU DE 32 BITS CON GCC-4

Debido a que la solución anterior no fue optima se procede a usar una maquina virtual clonada de Ubuntu 16.04 de 32 bits, que ya tenía el header configurado, en esta ocasión se forzara la instalación del gcc-4. Para esto primero se instala apt-get install python-software-properties (Bowen, 2017).

Figura 134

Instalación de python-software-properties

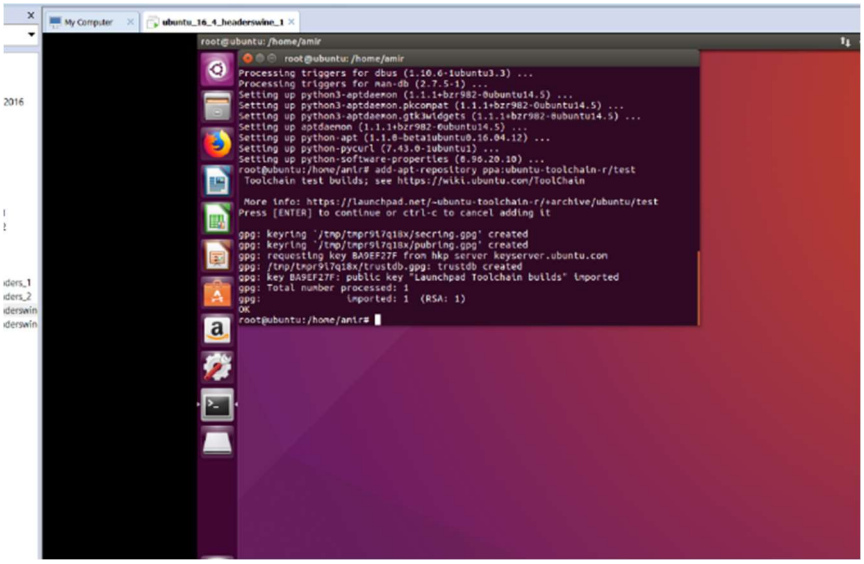


Nota. Esta imagen es la instalación de python-software-properties. (Diseño Propio)

Se agrega el repositorio con el comando add-apt-repository ppa:ubuntu-toolchain-r/test.

Figura 135

Agregar el repositorio add-apt-repository ppa:ubuntu-toolchain-r/test

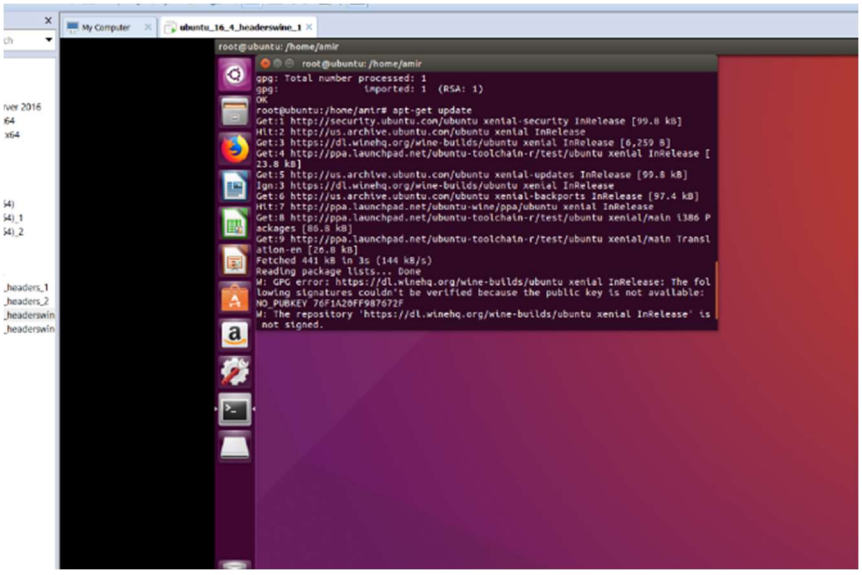


Nota. Esta imagen es agregar el repositorio add-apt-repository ppa:ubuntu-toolchain-r/test. (Diseño Propio)

Se realiza una actualización de los repositorios en source.list.

Figura 136

Actualización de source.list

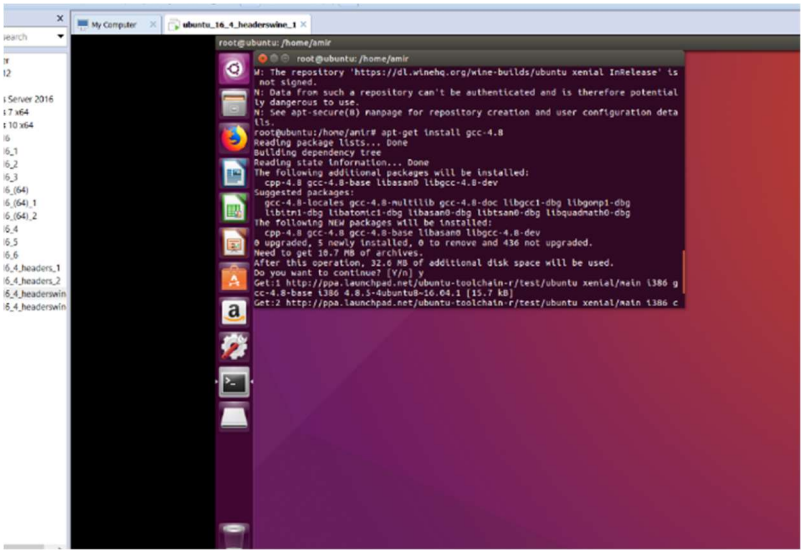


Nota. Esta imagen es la actualización de source.list. (Diseño Propio)

Se procede con la instalación de la versión gcc-4.8 correspondiente a la versión 4, ya que fueron agregados los repositorios a donde se manda la referencia.

Figura 137

Instalación de gcc-4.8

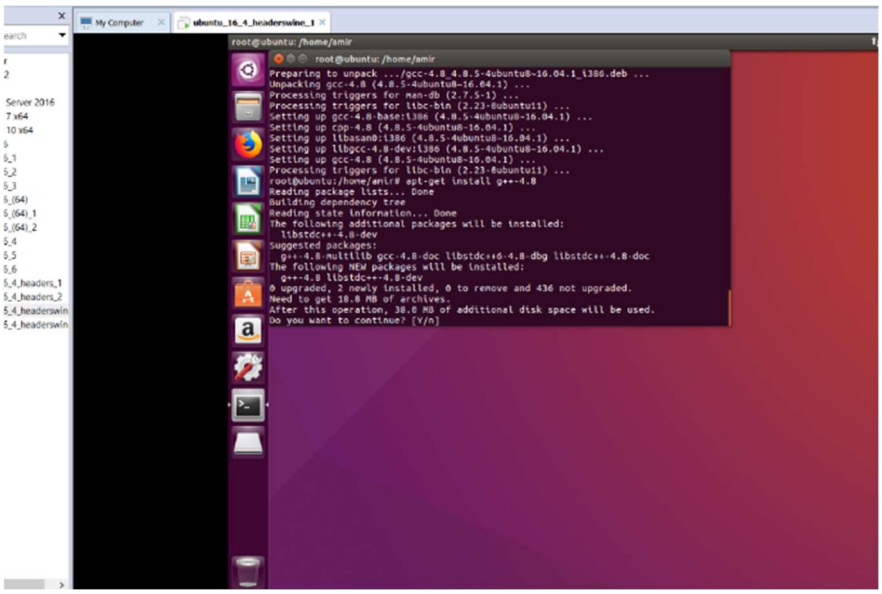


Nota. Esta imagen es la instalación de gcc-4.8. (Diseño Propio)

Se instala la librería g++-4.8 correspondiente a la versión 4, ya que fueron agregados los repositorios a donde se manda la referencia.

Figura 138

Instalación de g++ 4.8



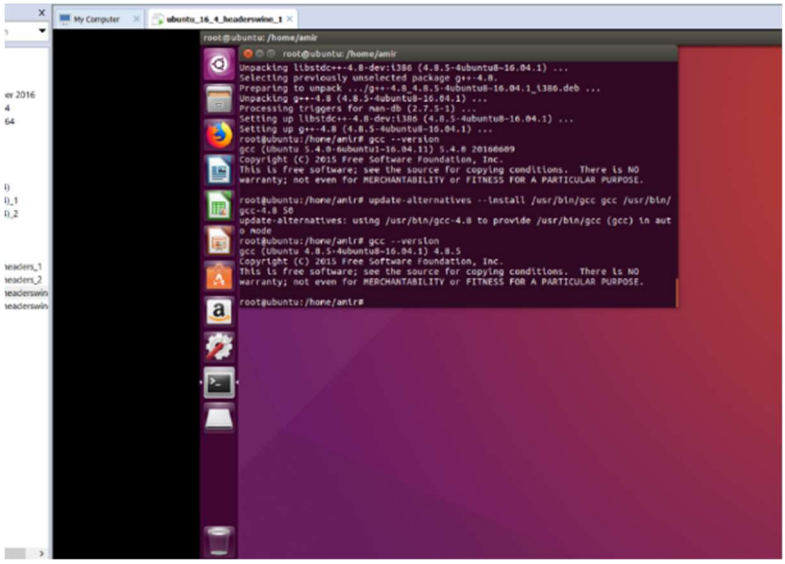
Nota. Esta imagen es la instalación de g++4.8. (Diseño Propio)

Después de terminar con la instalación de la versión 4 se debe referenciar la carpeta gcc-4.8 como versión principal del gcc, esto se logra con el comando update-alternatives --install /usr/bin/gcc gcc

/usr/bin/4-8 50. Se debe revisar la versión de gcc por lo que se ejecuta gcc --version, se muestra que la versión 4.8 aparece como la principal pero que ubuntu no garantiza este software ya que como es una versión antigua solo debe ser usada para propósitos particulares.

Figura 139

Verificación de version de gcc

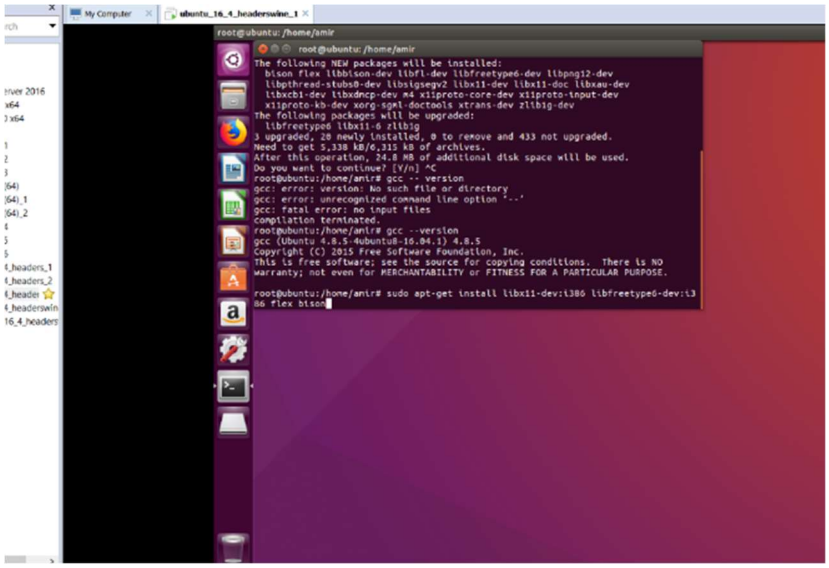


Nota. Esta imagen es la verificación de versión de gcc. (Diseño Propio)

Se procede a instalar las librerías como en el punto 3.4.2..

Figura 140

Instalación de librerías esenciales y bison



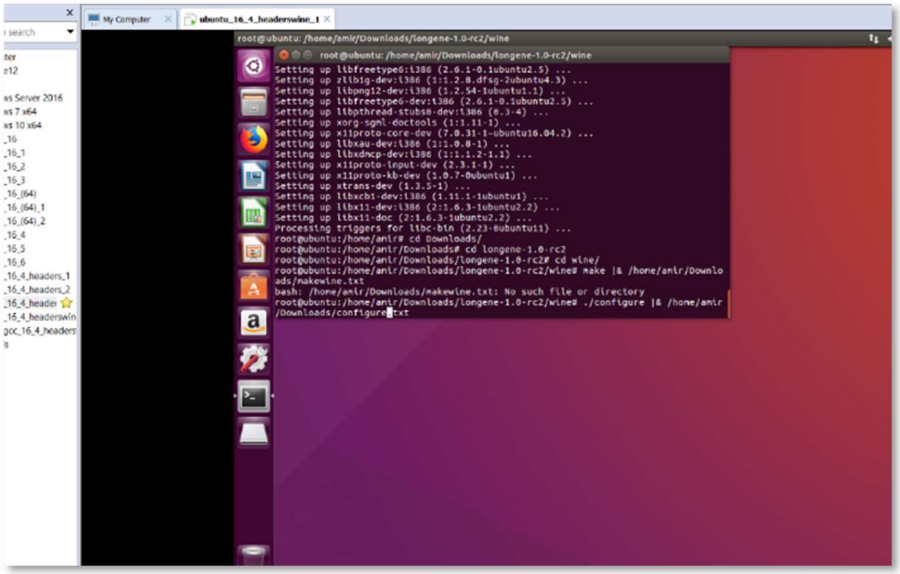
Nota. Esta imagen es la instalación de librerías esenciales y Bison. (Diseño Propio)

Luego de que no se mandaran fallos se procede con la instalación de Longene, siguiendo los pasos ya mencionados con anterioridad.

Se comienza con el comando ./configure.

Figura 141

Ejecución de configure gcc 4.8

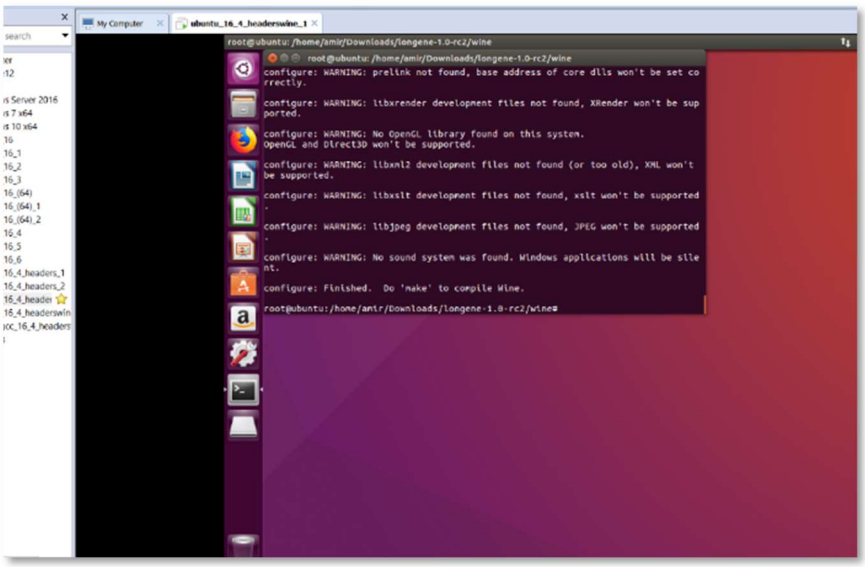


Nota. Esta imagen es la ejecución de ./configure con gcc 4.8 (Diseño Propio)

En la pantalla de resultados se obtuvieron las mismas alertas mencionadas antes, pero de igual forma se permite proseguir.

Figura 142

Resultado ./configure con gcc 4.8

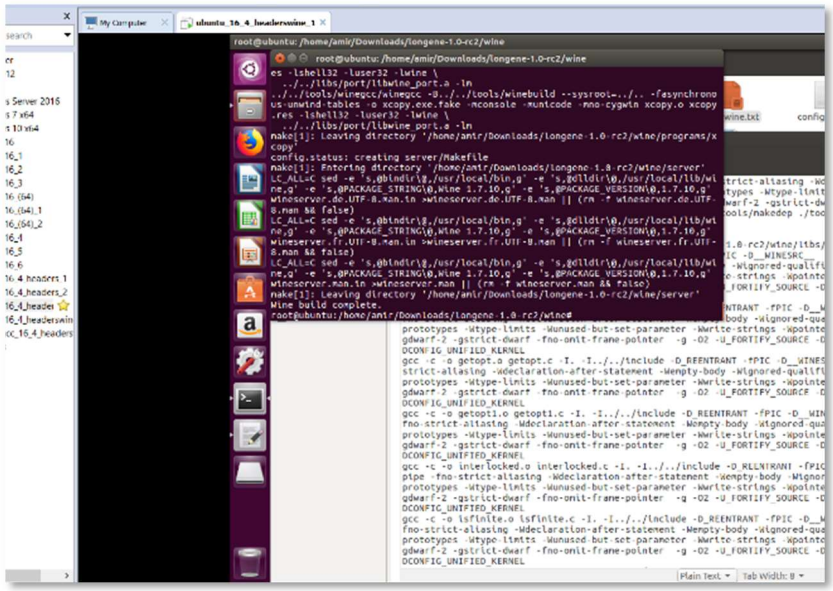


Nota. Esta imagen es el resultado ./configure gcc 4.8 (Diseño Propio)

Se procede con la ejecución de make, obteniendo de resultado ningún tipo de error dentro de la ejecución.

Figura 143

Resultado ejecución make gcc4.8

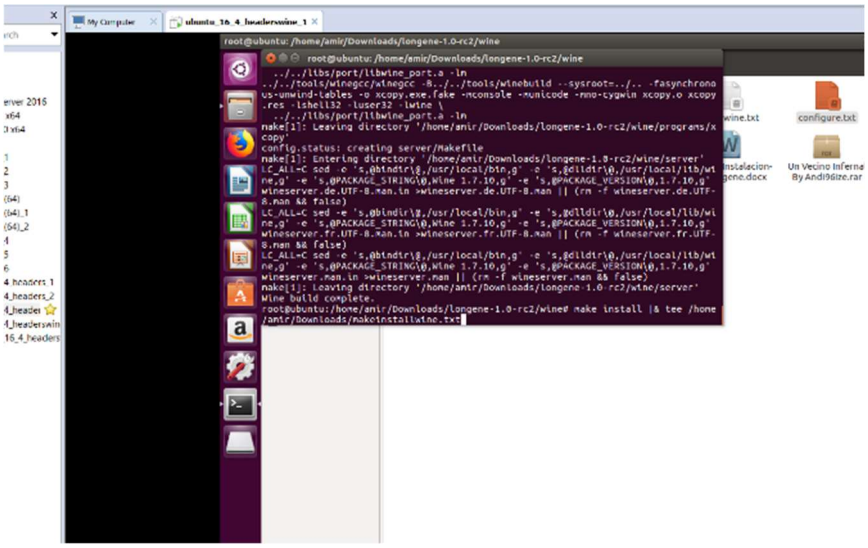


Nota. Esta imagen es el resultado ejecución make gcc4.8. (Diseño Propio)

Luego se procede con la ejecución de make install.

Figura 144

Ejecución make install gcc4.8

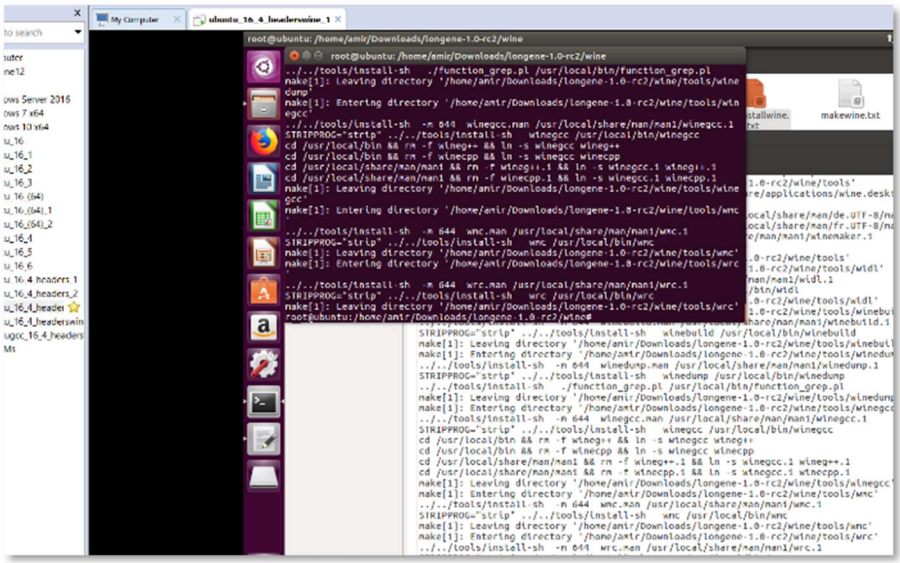


Nota. Esta imagen es resultado de make install gcc4.8. (Diseño Propio)

No se obtuvo ningún tipo de error al ejecutar el comando make install.

Figura 145

Resultado ejecución make install gcc4.8

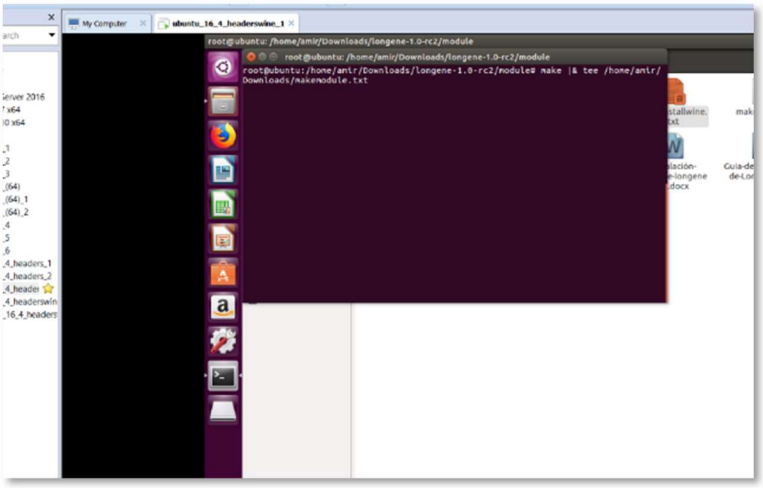


Nota. Esta imagen es resultado de la ejecución de make install con gcc4.8. (Diseño Propio)

Ya que se instalaron normalmente los pasos anteriores, se procede al paso donde se tuvo un error la última vez, hacer el make dentro de la carpeta /home/amir/Downloads/longene-1.0-rc2/modules.

Figura 146

Ejecución de make en module gcc 4.8

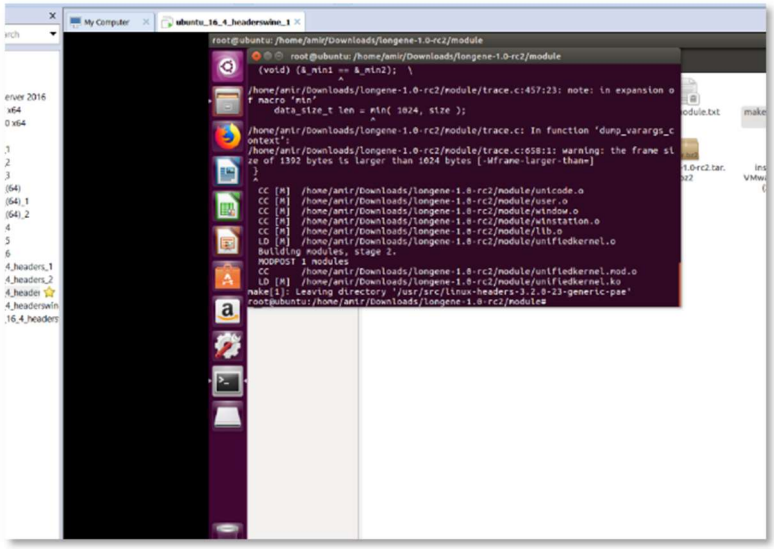


Nota. Esta imagen es ejecución de make en module gcc 4.8 (Diseño Propio)

En esta ocasión no se produjeron errores que hayan parado con la instalación.

Figura 147

Resultado make module gcc 4.8

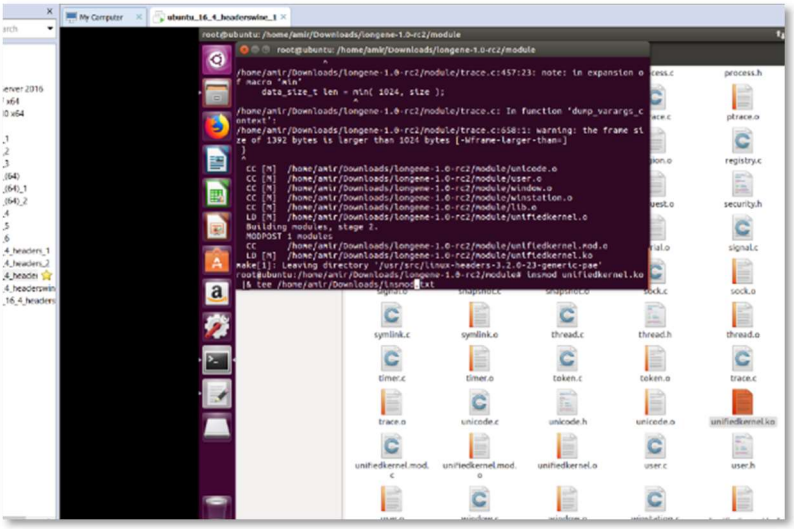


Nota. Esta imagen es resultado de ejecutar make en module gcc 4.8. (Diseño Propio)

Como se visualiza en la figura 147, finalmente Longene ha sido instalado con éxito, por lo que lo único que queda es cargar el módulo que se generó en la carpeta /home/amir/Downloads/longene-1.0-rc2/module/ llamado unifiedkernel.ko. Esto se realizará con el comando insmod. "*El comando insmod en los sistemas Linux se usa para insertar módulos en el kernel. Linux es un sistema operativo que permite al usuario cargar módulos del kernel en tiempo de ejecución para ampliar las funcionalidades del kernel. Los LKM (módulos de kernel cargables) generalmente se usan para agregar soporte para nuevo hardware (como controladores de dispositivos) y/o sistemas de archivos, o para agregar llamadas al sistema. Este comando aquí inserta el archivo de objeto del kernel (.ko) en el kernel con/sin argumentos, junto con algunas opciones adicionales*." (MukkeshMckenzie, 2019). Esto se puede visualizar en la figura 148.

Figura 148

Ejecución de insmod unifiedkernel

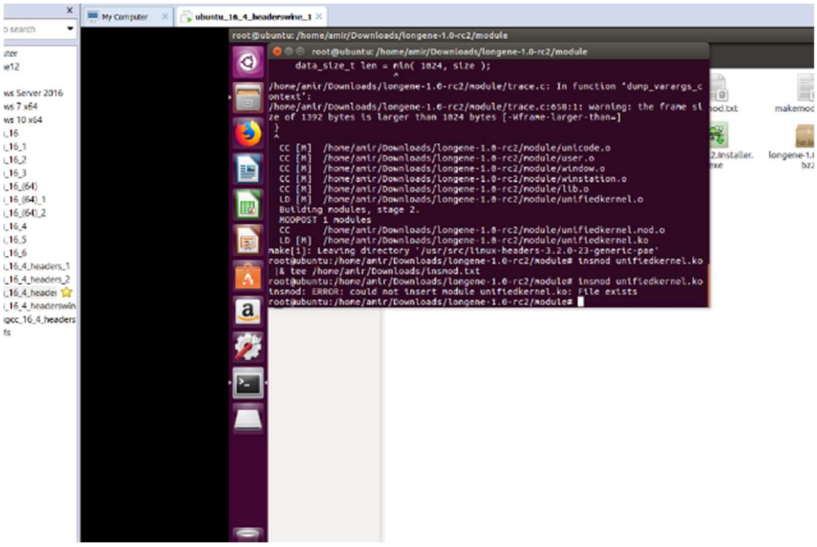


Nota. Esta imagen es resultado de ejecutar insmod unifiedkernel con gcc4.8. (Diseño Propio)

Si es la primera vez que se carga el módulo la ejecución no bota ningún mensaje, pero si se intenta una vez más se menciona que el archivo ya existe.

Figura 149

Resultado segunda ejecución de insmod



Nota. Esta imagen es resultado de segunda ejecución de insmod. (Diseño Propio)

Habiendo terminado el proceso de instalación con éxito se procede a las pruebas de los programas.

CAPITULO IV: RESULTADOS

4.1. COMPROBACIÓN DE LA PROSPECTIVA

4.1.1. ENTORNO DE PRUEBAS

Para proceder con los resultados se dará una breve reseña de los puntos a considerar para las pruebas realizadas. Esto debido a que se considera necesario justificar las variables que se tomaron en cuenta mostrando los resultados obtenidos.

Por este motivo se consideraron 3 características de la investigación las cuales son Tipo de Investigación, Nivel de Investigación y Método de Investigación; ya que se consideró relevante para la evaluación de viabilidad del proyecto, en este punto se limitara a mencionarlos según la clasificación considerada por el investigador:

- Tipo de Investigación: Aplicativa
- Nivel de Investigación: Explicativa
- Método de Investigación: No experimental

Para dilucidar estos puntos se dará una explicación somera de cada uno para tomarlo en cuenta.

- Una investigación aplicativa se basa en aplicar los conocimientos se caracteriza porque busca la aplicación o utilización de los conocimientos que se adquieren. La investigación aplicada se encuentra estrechamente vinculada con la investigación básica, pues depende de los resultados y avances.
- Se enfoca en explicar por qué ocurre un fenómeno y en qué condiciones se manifiesta, o por qué se relacionan dos o más variables, además proporciona un sentido de entendimiento al fenómeno.
- La investigación no experimental es sistemática y empírica en la que las variables independientes no se manipulan porque ya han sucedido. Las inferencias sobre las relaciones entre variables se realizan sin intervención o influencia directa.

Es por esto que para la realización de las pruebas de la evaluación de la viabilidad se consideró “Realizar pruebas a partir de los conocimientos adquiridos durante la educación universitaria para proporcionar el entendimiento de cómo funciona el proyecto, evaluando el mismo mediante las variables que fueron consideradas en base a un antecedente internacional enfocado en comparativas de rendimiento*”.*

Así pues, para continuar con la investigación se procede a crear el cuadro informativo de las variables que se tomaron consideración para evaluar la viabilidad de Longene, en base al marco teórico de la tesis y explicado dentro del siguiente título.

Tabla 7

Tabla de Variables e Indicadores

Variables dependientes	**Indicadores de variables dependientes**	**Variables independientes**	**Indicadores de variables independientes**
Evaluación de Rendimiento	Tiempo de ejecución	Viabilidad de Longene	Cantidad de instrucciones de ejecuciones realizadas (strace)
	Uso de CPU		Cantidad de memoria de CPU usada al momento de la ejecución
	Uso de memoria		•Tamaño de archivo de ejecución

Nota. Esta tabla contiene las variables e indicadores para la evaluación de la viabilidad de Longene. (Diseño Propio)

Para justificar la razón de usar estas variables se procedo abordar estos puntos detalladamente, tomando en consideración referencias bibliográficas de otros proyectos de investigación.

4.1.1.1. REFERENCIAS PARA CRITERIOS DE ENTORNOS DE PRUEBA

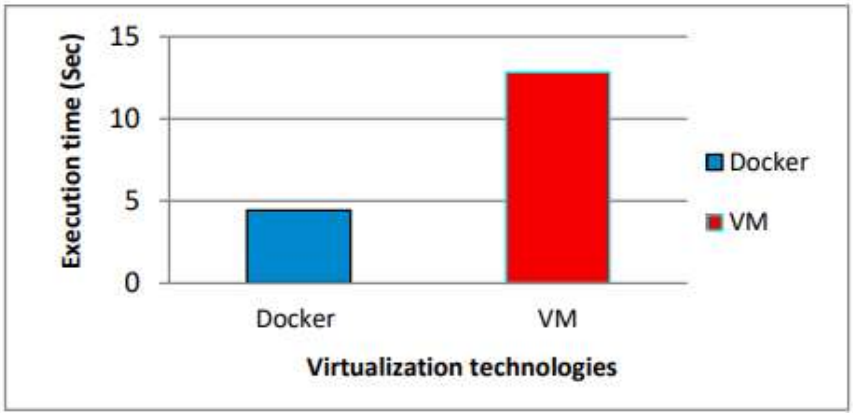
Prueba de Tiempo de Ejecución

Apartado 2.1.2 Antecedentes a Nivel internacional Inciso 6. Paper "Performance Evaluation of Docker Container and Virtual Machine" elaborado por (Potdara, Narayan, Kengond, & Mulla, 2020)

"La prueba mide cuánto tiempo lleva resolver el problema. El programa Eight queen está escrito en python y determina el rendimiento computacional del sistema. La figura 150 muestra el rendimiento computacional tanto de la ventana acoplable como de las máquinas virtuales. Según el tiempo de ejecución, el contenedor docker tarda menos en resolver el problema, mientras que la máquina virtual tarda mucho más.

Figura 150

Eight Queen Program Performance Comparison



Nota. Este grafico muestra la comparativa de tiempo de ejecución entre Docker y VM. (Potdara, Narayan, Kengond, & Mulla, 2020)

Prueba de ocho rompecabezas: tomar un tablero de 4 × 4 con 8 fichas y un espacio vacío. Utilizando el espacio vacío, el objetivo principal es la disposición del número de mosaicos para que coincida con la configuración final. Puede deslizar cuatro mosaicos de operaciones adyacentes (derecha, izquierda, abajo y arriba) en el

espacio vacío; la prueba mide cuánto tiempo lleva resolver el problema. El programa Eight puzzle está escrito en python y determina el rendimiento computacional del sistema. Se muestra el rendimiento computacional tanto de la ventana acoplable como de las máquinas virtuales. Según el tiempo de ejecución, el contenedor docker tarda menos en resolver el problema, mientras que la máquina virtual tarda mucho más."

Para realizar esta prueba en la investigación se realizarán tomas de tiempo al ejecutar dos programas y verificar la comparativa entre estos.

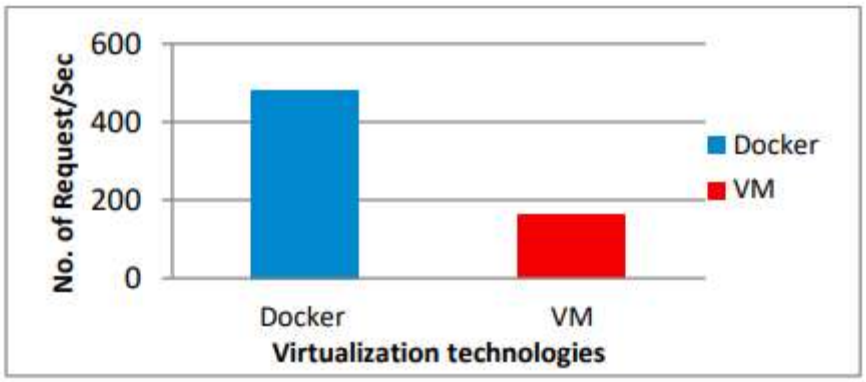
Prueba Uso de CPU

Apartado 2.1.2 Antecedentes a Nivel internacional Inciso 7. Paper "Performance Evaluation of Docker Container and Virtual Machine" elaborado por (Potdara, Narayan, Kengond, & Mulla, 2020)

"Pruebas de carga: Para la comparación del rendimiento de las pruebas de carga se utiliza la herramienta Apache Benchmark, en la que mide la cantidad de solicitudes por segundo que un sistema determinado puede tolerar. Se ejecuta un programa de python para probar la carga utilizando la herramienta Apache Benchmarking. La figura 151 muestra que el análisis de rendimiento para VM es mucho menor en comparación con el de Docker. Esto se debe a una mayor latencia de red en la máquina virtual que en Docker. El análisis muestra que el contenedor Docker es mejor que la máquina virtual en el manejo de la cantidad de solicitudes por segundo. ".

Figura 151

Load test comparison between Docker and Virtual Machine



Nota. Este grafico muestra la comparativa de cantidad de solicitudes por segundo entre Docker y VM. (Potdara, Narayan, Kengond, & Mulla, 2020)

Para realizar esta prueba en la investigación se tomaran como referentes la cantidad de solicitudes realizadas al CPU, las respuestas recibidas y el monitoreo del CPU al momento de ejecutar las solicitudes.

Prueba Uso de Memoria

Para las pruebas de uso de memoria no es necesario realizar las referencias debido a que esto recae en conjunto entre las 2 pruebas previamente mencionadas, por este motivo, ya tengo se tiene el suficiente contexto para la propuesta en este ámbito por lo que se consideraran como uso de memoria la cantidad de solicitudes realizadas para la ejecución de los 2 programas y los monitoreos de HTOP.

4.1.2. COMANDOS Y TÉCNICAS DE PRUEBAS

Bash es una interfaz de comandos de Linux, en esta herramienta se pueden ingresar comandos script para realizar órdenes y ejecutar ciertos programas desde la misma, esta herramienta es la principal y más usada de las distribuciones Linux, por esta razón se dará una breve descripción de ciertos criterios que contiene:

- Comandos: Intérprete de lenguaje de comandos que ejecuta comandos leídos desde la entrada estándar o desde un archivo. bash también incorpora características útiles de los shells Korn y C (ksh y csh). bash pretende ser una implementación conforme de la parte de Shell y Utilidades de la especificación IEEE POSIX (IEEE Standard 1003.1). bash se puede configurar para que sea compatible con POSIX de forma predeterminada. (Hope, 2021)
- Opciones: Puede hacer esas dos cosas usando las opciones de la línea de comandos. Encuentro que incluso los programas Bash simples deberían tener algún tipo de función de ayuda, incluso si es bastante rudimentario. Muchos de los programas de shell de Bash que escribo se usan con la frecuencia suficiente como para olvidar la sintaxis exacta del comando que necesito ejecutar. Algunos son tan complejos que necesito revisar las opciones y los argumentos necesarios, aunque los uso con frecuencia. Tener una función de ayuda incorporada le permite ver esas cosas sin tener que recurrir a inspeccionar el código en sí. Una buena y completa instalación de ayuda es también una parte de la documentación del programa. (Ambos, 2021)
- La capacidad de usar parámetros posicionales, también conocidos como argumentos, para especificar los datos que se usarán como valores para las variables en los scripts es un método para lograr esto. Otro es el uso de opciones y argumentos de opción. Este artículo explora estos dos métodos para obtener datos en el script y controlar la ruta de ejecución del script. En los parámetros posicionales Bash usa una herramienta llamada parámetros posicionales para proporcionar un medio para ingresar datos en un programa Bash cuando se invoca desde la línea de comando. Hay diez parámetros posicionales que van desde $0 hasta $9, aunque hay formas de sortear ese límite. (Ambos, 2021)

Así pues, por lo expuesto y sustentado se usarán 2 máquinas con las mismas características y recursos, una donde se instale solamente Wine y otra con Wine con Longene y se consideraran los siguientes puntos:

- El **tiempo de ejecución** debe ser el tiempo en el que se abre una aplicación, para este aspecto se considera la cantidad de llamadas de instrucciones para ejecutar un programa. Con el comando "strace" se pueden visualizar las llamadas al sistema realizadas por la ejecución de un proceso, así como las señales recibidas. "*strace es una potente herramienta de línea de comandos para depurar y solucionar problemas de programas en sistemas operativos similares a Unix, como Linux. Captura y registra todas las llamadas al sistema realizadas por un proceso y las señales recibidas por el proceso."* (Kili, 2017). Concretamente usare las opciones "-df"
  strace -df , d: todas las salidas de ejecución stderr y f para seguir las bifurcaciones.
  Output usage: strace [-CdffhiqrtttTvVxxy] [-I n] [-e expr]...
  [-a column] [-o file] [-s strsize] [-P path]...
  -p pid... / [-D] [-E var=val]... [-u username] PROG [ARGS]
  or: strace -c[df] [-I n] [-e expr]... [-O overhead] [-S sortby]
  -p pid... / [-D] [-E var=val]... [-u username] PROG [ARGS]
  -c -- count time, calls, and errors for each syscall and report summary
  -C -- like -c but also print regular output
  -d -- enable debug output to stderr
  -D -- run tracer process as a detached grandchild, not as parent
  -f -- follow forks, -ff -- with output into separate files

-i -- print instruction pointer at time of syscall https://conpilar.es/como-rastrear-la-ejecucion-del-programa-usando-el-comando-strace-de-linux/#:~:text=Strace%20es%20una%20poderosa%20herramienta,se%C3%B1ales%20recibidas%20por%20un%20proceso.

- **Uso de CPU** para este criterio existen muchas formas de hacerlo de igual forma que el punto anterior, pero en este caso se usaran dos comandos por separado, uno de estos es **ps aux**. *"El comando ps (estado del proceso) es uno de los comandos más utilizados en Linux. Por lo general, se utiliza para obtener información más detallada sobre un proceso específico o todos los procesos. Por ejemplo, se usa para saber si un proceso en particular se está ejecutando o no, quién está ejecutando qué proceso en el sistema, qué proceso está usando más memoria o CPU, cuánto tiempo se está ejecutando un proceso, etc. a:- Esta opción imprime los procesos en ejecución de todos los usuarios. u:- Esta opción muestra la columna de usuario o propietario en la salida. x:- Esta opción imprime los procesos que no se han ejecutado desde la terminal."* (ComputerNetworkingNotes, 2019), esto lo hace la opción más usada por administradores. Adicionalmente usare la librería Htop, *"Esta es una utilidad para monitorizar y gestionar los procesos del sistema, que se ejecuta en la terminal."* (A. D. , s.f.), a diferencia de los tres comandos anteriores mencionados esta librería si debe ser instalada y se realizara con el comando **apt-get install htop**, después de esto para usarla se usa el comando htop. Así mismo se usara la salida de output a file para ps aux y mostraran las imágenes de lo generado, en el caso de htop es más complejo por lo que simplemente se mostraran las evidencias. También se registraran la cantidad de solicitudes obtenidas al realizar el strace -df anteriormente mencionado.

- **Uso de Memoria** se refiere a la cantidad de memoria usada por un programa desde el espacio que ocupa al momento de ejecutarse o la cantidad que ocupa en disco, para poder visualizar esta cantidad necesito llevar a cabo una monitorización del uso de recursos, por esta razón se plantea el uso de htop y adicionalmente tomar en consideración el tamaño final del archivo generado por strace -df ya que de esta forma se pueden registrar cuantas cosas tuvo que hacer para poder ejecutarse.

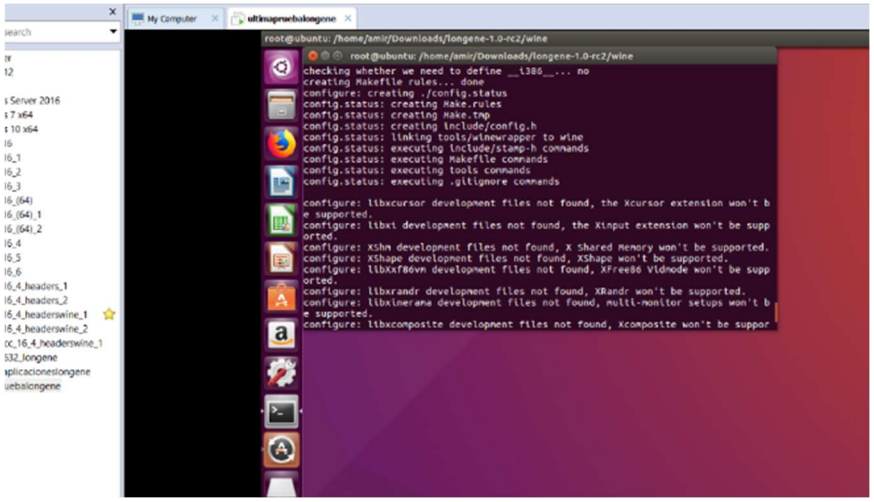
4.1.3. PRUEBAS

4.1.3.1. CONFIGURACIONES ADICIONALES EN PRUEBA

Después de todo lo que fue mencionado, para las pruebas de aplicación se consideraron 3 programas los cuales permitan observar la diferencia del comportamiento de Wine y Wine con Longene. Para estas pruebas necesito usar software de 32 bits, por lo que plantee usar Office 16 de 32 bits, “Un Vecino Infernal” (el juego) y Notepad++. Antes de proceder con la documentación de las pruebas de estos programas surgió un inconveniente. El inconveniente suscitó con el juego Un Vecino Infernal, al ser un juego usa bastante el GUI, el error era que la pantalla se volvía negra cuando se ejecutaba el juego y no se mostraba nada, a pesar de haber instalado Longene sin errores que detuvieran la instalación se tuvo que hacer una nueva máquina virtual y añadir algunas librerías que se dejaron afuera antes del comando ./configure.

Figura 152

Instalación de librerías adicionales

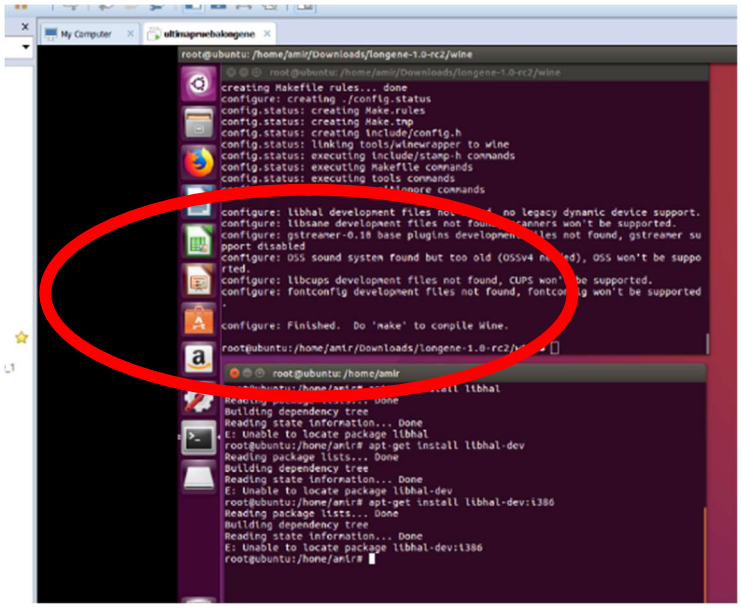


Nota. Esta imagen muestra la instalación de librerías adicionales. (Diseño Propio)

La lista fue bastante extensa ya que aparentemente wine 1.6 necesitaba la librería ia32-libs pero no existen candidatos óptimos, así que se tuvo que instalar una por una las dependencias, las que se considerarían más relevantes según los resultados fueron libosmesa6:i386 libosmesa-dev libjpeg-turbo8-dev:i386 qt4-qmake libx*(cursor, randr, etc) pre-link open-cl. Al final hubo algunas librerías que no pudieron ser instaladas, ya que es más fácil decir cuales no fueron instaladas y cuales si se mostraran en la imagen de las únicas librerías que no pudieron ser instaladas.

Figura 153

Librerías no instaladas



Nota. Esta imagen muestra las librerías adicionales que no pudieron ser instaladas. (Diseño Propio)

Esto se debió a distintas razones, que la librería no existe ya como candidata o que se instaló, pero la versión no es reconocida, por lo que lo ideal sería buscar una versión antigua, lamentablemente no hay certeza de encontrarlas. Aun así y a pesar de todo se logró el cometido de probar “Un Vecino Infernal” en Longene y funciono, por lo que se tomara en cuenta esta última versión mejorada de instalación para las pruebas.

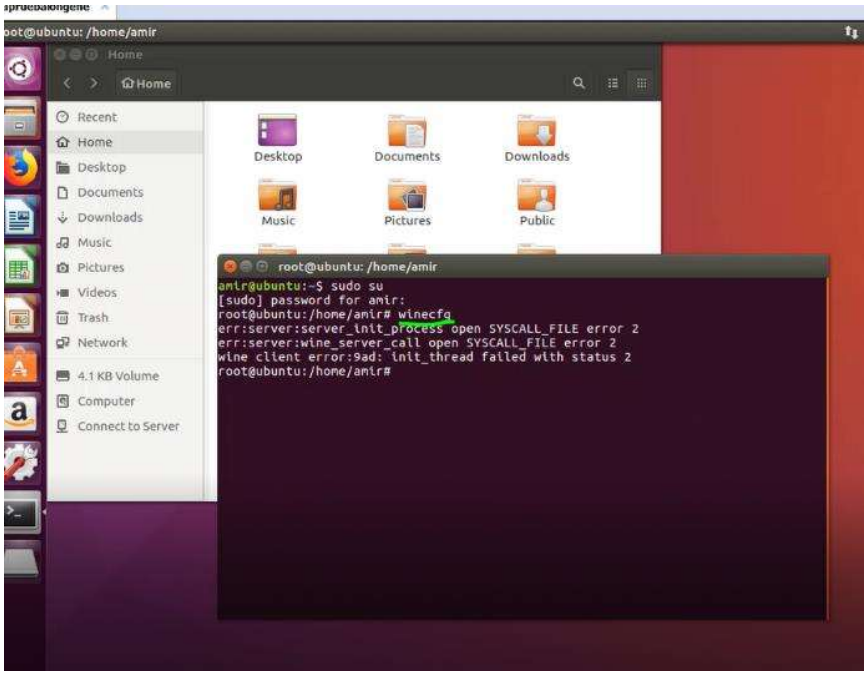
4.1.3.2. INTENTO DE PRUEBAS DE WORD

Para instalar algún software Office en Wine el proceso es realmente complejo ya que se deben usar environment, esto es crear adicionar archivos dlls y configuraciones especificas en una carpeta donde se redirigiría la instalación de office. Se encontró la forma de hacer esto en un blog una web China “在 Ubuntu 18.04 上使用 Wine 安装 Office 2016 ProPlus” (日出1.1, 2019). Se siguieron todos los pasos y se pude instalar Office en Ubuntu 18 y 16, pero para esto se necesitaba el winetricks y el winecfg. Winetricks es una herramienta adicional de WINE, lamentablemente no fue posible la instalación de esta

herramienta dentro del entorno de winecfg, ya que el unifiedkernel cortaba toda la configuración adicional que se podría realizar esto se muestra en la figura 154. Por este motivo se descartó la opción de realizar pruebas en algún software OFFICE y solo se tomaron en consideración las otras dos aplicaciones previamente mencionadas las cuales son NotePad++ y “Un vecino Infernal”.

Figura 154

Ejecutar WINECFG en Longene



Nota. Esta imagen muestra la pantalla al intentar de ejecutar WINECFG (Diseño Propio)

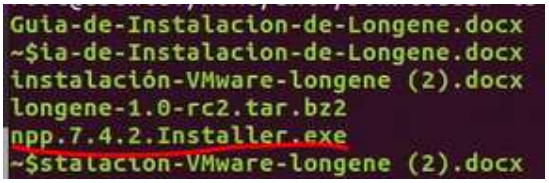
4.1.3.3 PRUEBAS EN LONGENE

4.1.3.3.1. NOTEPAD++

Para la instalación y ejecución de Notepad++ no hubo ningún inconveniente, por este motivo se mostrarán las pantallas de instalación y ejecución de este programa.

Figura 155

Instalador Notepad++



Nota. Esta imagen muestra el instalador Notepad++. (Diseño Propio)

Se realizo un strace a la instalación, aunque solamente bastaba usar la sintaxis "./ejecutable.exe", el strace se usó para tener constancia de la instalación.

Figura 156

Instalación de notepad++

Nota. Esta imagen muestra la sentencia para la instalación de notepad++. (Diseño Propio)

Luego de ejecutar la orden apareció la pantalla del mismo instalador de Notepad++.

Figura 157

Notepad menú idioma

Nota. Esta imagen muestra el menú de selección de idioma de la instalación de Notepad++. (Diseño Propio)

A continuación, luego de seleccionar el idioma, apareció la pantalla principal de instalación de Notepad++.

Figura 158

Pantalla principal de instalación de Notepad++

Nota. Esta imagen muestra la pantalla principal de la instalación de Notepad++. (Diseño Propio)

Lo siguiente que aparecía al instalar Notepad++ es elegir la ruta de instalación, la ruta C: es exclusiva de Windows, pero Wine crea una ruta virtual de esta haciendo referencia a una subruta y permitiendo la instalación de los programas Windows.

Figura 159

Elección de ruta instalación Notepad++

Nota. Esta imagen muestra la elección de ruta donde se instalará Notepad++. (Diseño Propio)

Luego se muestra la pantalla de selección de componentes adicionales de Notepad++.

Figura 160

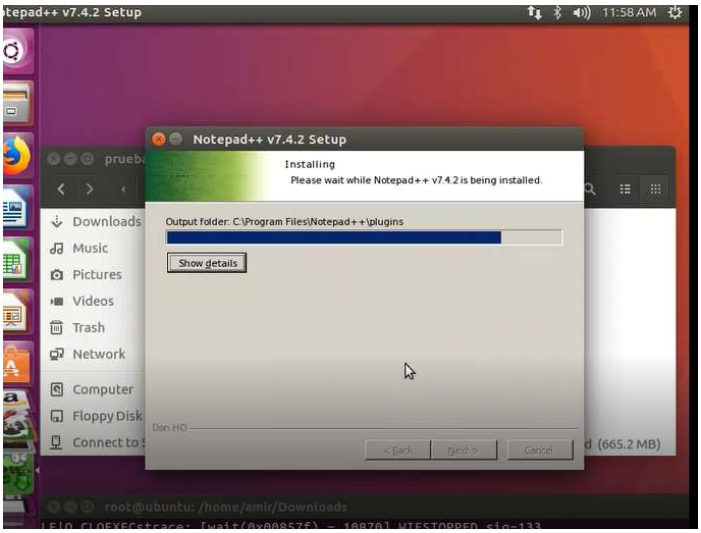
Elección de componentes Notepad++

Nota. Esta imagen muestra la elección de componentes de instalación Notepad++. (Diseño Propio)

Finalmente se muestra la pantalla de carga al instalar el programa Notepad++, esto resulto sin ningún tipo de incidencia.

Figura 161

Pantalla de carga de instalación de Notepad++



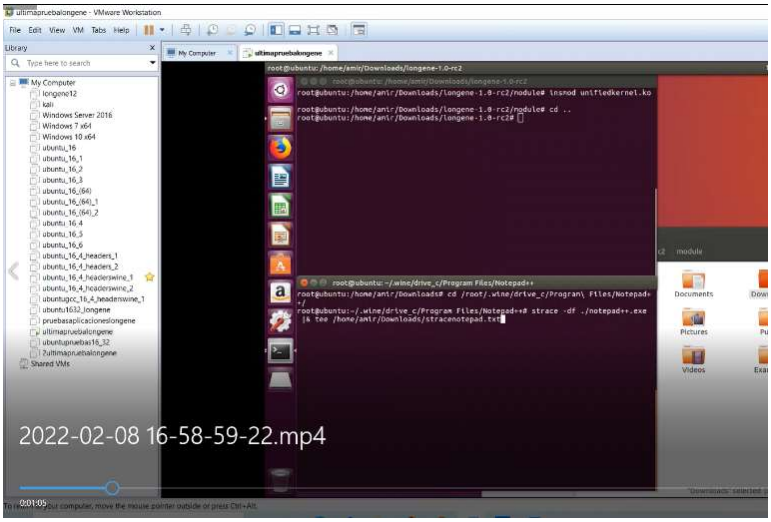
Nota. Esta imagen muestra la pantalla de carga de la instalación de Notepad++. (Diseño Propio)

Como es un programa instalado en el entorno de Wine, se debe acceder a la carpeta que este crea, esta carpeta se encuentra

localizada en “~/.wine/drive_c/Program Files/NotePad++”, el tiempo para la ejecución del programa incluyendo el guardado de instrucciones en un archivo denominado “stracenotepad.txt” con el comando strace se realizo en un total de 44 segundos, como se muestra en las figura 162 y 163

Figura 162

Ejecución de Notepad++ en Longene

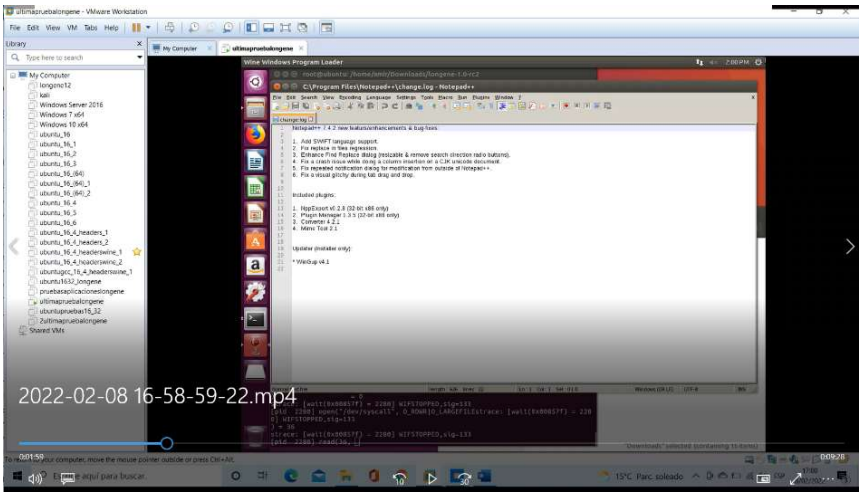


Nota. Esta imagen muestra el comando de Ejecución de Notepad++ en Longene. (Diseño Propio)

En la siguiente imagen se observa el software ejecutándose en el entorno Linux configurado con Wine y Longene.

Figura 163

Notepad++ Longene



Nota. Esta imagen muestra el Notepad++ funcionando en Longene. (Diseño Propio)

Ya que el resultado de la opción ps aux es demasiado extensa para ser mostrada completa en el documento en cada caso, se consideró que esta será la primera y última vez que se mostrara el resultado total de ps aux, siendo que en esta investigación el objetivo está en demostrar la veracidad de lo mencionado, además de que no se considera optimo e imprescindible llegado a este punto.

Código 2

Código ps aux

```
1.  USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
    COMMAND
2.  root          1  0.0  0.1  25024  3528 ?        Ss   13:22   0:02
    /sbin/init ro auto noprompt quiet
3.  root          2  0.0  0.0      0     0 ?        S    13:22   0:00
    [kthreadd]
4.  root          3  0.0  0.0      0     0 ?        S    13:22   0:00
    [ksoftirqd/0]
5.  root          6  0.0  0.0      0     0 ?        S    13:22   0:00
    [migration/0]
6.  root          7  0.0  0.0      0     0 ?        S    13:22   0:00
    [watchdog/0]
7.  root          8  0.0  0.0      0     0 ?        S<   13:22   0:00
    [cpuset]
8.  root          9  0.0  0.0      0     0 ?        S<   13:22   0:00
    [khelper]
9.  root         10  0.0  0.0      0     0 ?        S    13:22   0:00
    [kdevtmpfs]
```

```
10. root        11  0.0  0.0      0     0 ?        S<   13:22   0:00
    [netns]
11. root        12  0.0  0.0      0     0 ?        S    13:22   0:00
    [sync_supers]
12. root        13  0.0  0.0      0     0 ?        S    13:22   0:00 [bdi-
    default]
13. root        14  0.0  0.0      0     0 ?        S<   13:22   0:00
    [kintegrityd]
14. root        15  0.0  0.0      0     0 ?        S<   13:22   0:00
    [kblockd]
15. root        16  0.0  0.0      0     0 ?        S<   13:22   0:00
    [ata_sff]
16. root        17  0.0  0.0      0     0 ?        S    13:22   0:00
    [khubd]
17. root        18  0.0  0.0      0     0 ?        S<   13:22   0:00 [md]
18. root        21  0.0  0.0      0     0 ?        S    13:22   0:00
    [khungtaskd]
19.
20. root        22  0.0  0.0      0     0 ?        S    13:22   0:00
    [kswapd0]
21. root        23  0.0  0.0      0     0 ?        SN   13:22   0:00 [ksmd]
22. root        24  0.0  0.0      0     0 ?        SN   13:22   0:00
    [khugepaged]
23. root        25  0.0  0.0      0     0 ?        S    13:22   0:00
    [fsnotify_mark]
24. root        26  0.0  0.0      0     0 ?        S    13:22   0:00
    [ecryptfs-kthrea]
25. root        27  0.0  0.0      0     0 ?        S<   13:22   0:00
    [crypto]
26. root        35  0.0  0.0      0     0 ?        S<   13:22   0:00
    [kthrotld]
27. root        36  0.1  0.0      0     0 ?        R    13:22   0:03
    [kworker/0:1]
28. root        38  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_0]
29. root        39  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_1]
30. root        40  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_2]
31. root        41  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_3]
32. root        42  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_4]
33. root        43  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_5]
34. root        44  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_6]
35. root        45  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_7]
36. root        46  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_8]
37. root        47  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_9]
38. root        48  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_10]
39. root        49  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_11]
40. root        50  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_12]
41. root        51  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_13]
42. root        52  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_14]
43. root        53  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_15]
44. root        54  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_16]
```

```
45. root        55  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_17]
46. root        56  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_18]
47. root        57  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_19]
48. root        58  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_20]
49. root        59  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_21]
50. root        60  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_22]
51. root        61  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_23]
52. root        62  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_24]
53. root        63  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_25]
54. root        64  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_26]
55. root        65  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_27]
56. root        66  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_28]
57. root        67  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_29]
58. root        68  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_30]
59. root        71  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_31]
60. root        98  0.0  0.0      0     0 ?        S    13:22   0:00
    [kworker/u:31]
61. root        99  0.0  0.0      0     0 ?        S    13:22   0:01
    [kworker/0:2]
62. root       100  0.0  0.0      0     0 ?        S    13:22   0:00
    [kworker/u:32]
63. root       120  0.0  0.0      0     0 ?        S<   13:22   0:00
    [devfreq_wq]
64. root       179  0.0  0.0      0     0 ?        S<   13:22   0:00
    [mpt_poll_0]
65. root       180  0.0  0.0      0     0 ?        S<   13:22   0:00
    [mpt/0]
66. root       182  0.0  0.0      0     0 ?        S<   13:22   0:00
    [kpsmoused]
67. root       183  0.0  0.0      0     0 ?        S    13:22   0:00
    [scsi_eh_32]
68. root       221  0.0  0.0      0     0 ?        S    13:22   0:00
    [jbd2/sda1-8]
69. root       222  0.0  0.0      0     0 ?        S<   13:22   0:00 [ext4-
    dio-unwrit]
70. root       253  0.0  0.1   7164  2144 ?        Ss   13:22   0:00
    /lib/systemd/systemd-journald
71. root       255  0.0  0.0      0     0 ?        S    13:22   0:00
    [kauditd]
72. root       287  0.0  0.1  14328  3032 ?        Ss   13:22   0:00
    /lib/systemd/systemd-udevd
73. systemd+   323  0.0  0.0  12608  1056 ?        Ssl  13:22   0:00
    /lib/systemd/systemd-timesyncd
74. root       343  0.0  0.0  21292   392 ?        Ssl  13:22   0:00
    vmware-vmblock-fuse /run/vmblock-fuse -o rw,subtype=vmware-
    vmblock,default_permissions,allow_other,dev,suid
75. root       346  0.1  0.2  52004  5056 ?        Ssl  13:22   0:04
    /usr/bin/vmtoolsd
76. root       484  0.0  0.0      0     0 ?        S<   13:22   0:00 [hci0]
77. message+   576  0.0  0.1   6864  2716 ?        Ss   13:22   0:00
    /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile -
    -systemd-activation
```

```
78. root       584  0.0  0.1  37672  3328 ?        Ssl  13:22   0:01
    /usr/lib/accountsservice/accounts-daemon
79. syslog     585  0.0  0.0  30724  2056 ?        Ssl  13:22   0:00
    /usr/sbin/rsyslogd -n
80. root       586  0.0  0.1  14192  3400 ?        Ss   13:22   0:00
    /usr/sbin/cupsd -l
81. root       587  0.0  0.0   5572  1236 ?        Ss   13:22   0:00
    /usr/sbin/cron -f
82. root       588  0.0  0.1  39240  4116 ?        Ss   13:22   0:00
    /usr/bin/VGAuthService
83. root       590  0.0  0.0   4104  1704 ?        Ss   13:22   0:00
    /lib/systemd/systemd-logind
84. avahi      596  0.0  0.0   5916  1616 ?        Ss   13:22   0:00 avahi-
    daemon: running [ubuntu.local]
85. root       618  0.0  0.0   2240   684 ?        Ss   13:22   0:00
    /usr/sbin/acpid
86. root       627  0.0  0.4  82288  8612 ?        Ssl  13:22   0:01
    /usr/sbin/NetworkManager --no-daemon
87. root       636  0.0  0.3 845660  7884 ?        Ssl  13:22   0:00
    /usr/lib/snapd/snapd
88. root       656  0.0  0.0   4744   704 tty1     Ss+  13:22   0:00
    /sbin/agetty --noclear tty1 linux
89. avahi      679  0.0  0.0   5916   496 ?        S    13:22   0:00 avahi-
    daemon: chroot helper
90. root       684  0.0  0.0      0     0 ?        S    13:22   0:00
    [flush-8:0]
91. root       685  0.0  0.2  37432  4536 ?        Ssl  13:22   0:00
    /usr/sbin/cups-browsed
92. root       690  0.0  0.1  43428  3476 ?        SLsl 13:22   0:00
    /usr/sbin/lightdm
93. root       719  0.0  0.2  39104  4944 ?        Ssl  13:22   0:00
    /usr/lib/policykit-1/polkitd --no-debug
94. root       745  1.4  2.9 205524 61444 tty7     Ssl+ 13:22   0:36
    /usr/lib/xorg/Xorg -core :0 -seat seat0 -auth /var/run/lightdm/root/:0
    -nolisten tcp vt7 -novtswitch
95. root       786  0.0  0.0      0     0 ?        S<   13:22   0:00
    [loop0]
96. lp         824  0.0  0.1  11232  2180 ?        S    13:22   0:00
    /usr/lib/cups/notifier/dbus dbus://
97. lp         825  0.0  0.1  11232  2180 ?        S    13:22   0:00
    /usr/lib/cups/notifier/dbus dbus://
98. lp         826  0.0  0.1  11232  2184 ?        S    13:22   0:00
    /usr/lib/cups/notifier/dbus dbus://
99. lp         827  0.0  0.1  11232  2184 ?        S    13:22   0:00
    /usr/lib/cups/notifier/dbus dbus://
100.  lp         828  0.0  0.1  11232  2180 ?        S    13:22   0:00
    /usr/lib/cups/notifier/dbus dbus://
101.  root       841  0.0  0.0   6008  1848 ?        S    13:22   0:00
    /sbin/dhclient -d -q -sf /usr/lib/NetworkManager/nm-dhcp-helper -pf
    /var/run/dhclient-enp2s1.pid -lf /var/lib/NetworkManager/dhclient-
    3fb52880-a759-3ec5-ae0b-ed2f5bf557a5-enp2s1.lease -cf
    /var/lib/NetworkManager/dhclient-enp2s1.conf enp2s1
102.  nobody     853  0.0  0.0   9116  1756 ?        S    13:22   0:00
    /usr/sbin/dnsmasq --no-resolv --keep-in-foreground --no-hosts --bind-
    interfaces --pid-file=/var/run/NetworkManager/dnsmasq.pid --listen-
    address=127.0.1.1 --cache-size=0 --conf-file=/dev/null --proxy-dnssec -
    -enable-dbus=org.freedesktop.NetworkManager.dnsmasq --conf-
    dir=/etc/NetworkManager/dnsmasq.d
103.  whoopsie   948  0.0  0.3  58304  6296 ?        Ssl  13:22   0:00
    /usr/bin/whoopsie -f
104.  root      1014  0.0  0.1  27964  3336 ?        Sl   13:23   0:00
    lightdm --session-child 12 19
105.  rtkit     1102  0.0  0.0  23784  1320 ?        SNsl 13:23   0:00
    /usr/lib/rtkit/rtkit-daemon
106.  root      1114  0.0  0.2  79756  4900 ?        Ssl  13:23   0:00
    /usr/lib/upower/upowerd
107.  colord    1134  0.0  0.3  41784  6440 ?        Ssl  13:23   0:00
    /usr/lib/colord/colord
```

```
108. amir      1151  0.0  0.1   6384  2496 ?        Ss   13:23   0:00
    /lib/systemd/systemd --user
109. amir      1152  0.0  0.0   7600  1296 ?        S    13:23   0:00 (sd-
    pam)
110. amir      1228  0.0  0.1  38644  3332 ?        Sl   13:23   0:00
    /usr/bin/gnome-keyring-daemon --daemonize --login
111. amir      1234  0.0  0.1   8508  2424 ?        Ss   13:23   0:00
    /sbin/upstart --user
112. amir      1317  0.0  0.0   7412   692 ?        S    13:23   0:00
    upstart-udev-bridge --daemon --user
113. amir      1321  0.0  0.0   6696  2056 ?        Ss   13:23   0:00
    dbus-daemon --fork --session --address=unix:abstract=/tmp/dbus-
    Ctte4YV9L2
114. amir      1333  0.0  0.2  45808  4396 ?        Ss   13:23   0:00
    /usr/lib/i386-linux-gnu/hud/window-stack-bridge
115. amir      1357  0.1  0.1  45256  3308 ?        Ssl  13:23   0:02
    /usr/bin/ibus-daemon --daemonize --xim --address unix:tmpdir=/tmp/ibus
116. amir      1362  0.0  0.1  38256  3096 ?        Sl   13:23   0:00
    /usr/lib/gvfs/gvfsd
117. amir      1370  0.0  0.1  50680  2560 ?        Sl   13:23   0:00
    /usr/lib/gvfs/gvfsd-fuse /run/user/1000/gvfs -f -o big_writes
118. amir      1375  0.0  0.1  36448  2976 ?        Sl   13:23   0:00
    /usr/lib/ibus/ibus-dconf
119. amir      1384  0.0  0.0   7336   440 ?        S    13:23   0:00
    upstart-dbus-bridge --daemon --system --user --bus-name system
120. amir      1386  0.0  0.0   7336   432 ?        S    13:23   0:00
    upstart-dbus-bridge --daemon --session --user --bus-name session
121. amir      1391  0.0  0.7  95576 15300 ?        Sl   13:23   0:00
    /usr/lib/ibus/ibus-ui-gtk3
122. amir      1395  0.0  0.4  62600 10248 ?        Sl   13:23   0:00
    /usr/lib/ibus/ibus-x11 --kill-daemon
123. amir      1400  0.0  0.0   7656   504 ?        S    13:23   0:00
    upstart-file-bridge --daemon --user
124. amir      1423  0.0  0.6  73744 14112 ?        Ssl  13:23   0:00
    /usr/lib/i386-linux-gnu/bamf/bamfdaemon
125. amir      1428  0.0  0.1  43476  3224 ?        Sl   13:23   0:00
    /usr/lib/at-spi2-core/at-spi-bus-launcher
126. amir      1435  0.0  0.0   6060  1928 ?        S    13:23   0:00
    /usr/bin/dbus-daemon --config-file=/etc/at-spi2/accessibility.conf --
    nofork --print-address 3
127. amir      1441  0.0  0.0  24100   924 ?        Ss   13:23   0:00 gpg-
    agent --homedir /home/amir/.gnupg --use-standard-socket --daemon
128. amir      1443  0.0  0.1  29168  3260 ?        Sl   13:23   0:00
    /usr/lib/at-spi2-core/at-spi2-registryd --use-gnome-session
129. amir      1446  0.0  0.1  28164  2860 ?        Sl   13:23   0:00
    /usr/lib/ibus/ibus-engine-simple
130. amir      1461  0.0  1.0 133212 21128 ?        Ssl  13:23   0:00
    /usr/lib/i386-linux-gnu/hud/hud-service
131. amir      1463  0.0  0.7 256212 15704 ?        Ssl  13:23   0:00
    /usr/lib/unity-settings-daemon/unity-settings-daemon
132. amir      1477  0.0  0.3  75808  7436 ?        Ssl  13:23   0:00
    /usr/lib/gnome-session/gnome-session-binary --session=ubuntu
133. amir      1484  0.0  0.7 103084 15452 ?        Ssl  13:23   0:00
    /usr/lib/i386-linux-gnu/unity/unity-panel-service
134. amir      1508  0.0  0.1  46648  3464 ?        Ssl  13:23   0:00
    /usr/lib/i386-linux-gnu/indicator-messages/indicator-messages-service
135. amir      1513  0.0  0.1  52296  2488 ?        Ssl  13:23   0:00
    /usr/lib/i386-linux-gnu/indicator-bluetooth/indicator-bluetooth-service
136. amir      1521  0.0  0.1  71192  3944 ?        Ssl  13:23   0:00
    /usr/lib/i386-linux-gnu/indicator-power/indicator-power-service
137. amir      1524  0.0  0.4 133152  8912 ?        Ssl  13:23   0:00
    /usr/lib/i386-linux-gnu/indicator-datetime/indicator-datetime-service
138. amir      1525  0.0  0.5 126608 11104 ?        Ssl  13:23   0:00
    /usr/lib/i386-linux-gnu/indicator-keyboard/indicator-keyboard-service -
    -use-gtk
139. amir      1528  0.0  0.2 322052  5048 ?        Ssl  13:23   0:00
    /usr/lib/i386-linux-gnu/indicator-sound/indicator-sound-service
```

```
140. amir      1531  0.0  0.5  75812 11364 ?        Ssl  13:23   0:00
    /usr/lib/i386-linux-gnu/indicator-printers/indicator-printers-service
141. amir      1534  0.0  0.1  69552  3432 ?        Ssl  13:23   0:00
    /usr/lib/i386-linux-gnu/indicator-session/indicator-session-service
142. amir      1555  0.0  0.3  66184  6336 ?        Ssl  13:23   0:00
    /usr/lib/i386-linux-gnu/indicator-application/indicator-application-
    service
143. amir      1559  0.0  0.1  25268  2504 ?        Sl   13:23   0:00
    /usr/lib/dconf/dconf-service
144. amir      1573  0.0  0.3 211196  7620 ?        S<l  13:23   0:00
    /usr/bin/pulseaudio --start --log-target=syslog
145. amir      1582  0.0  0.5 123620 11752 ?        Sl   13:23   0:00
    /usr/lib/evolution/evolution-source-registry
146. amir      1632  0.0  1.7 163524 35372 ?        Sl   13:23   0:00
    /usr/lib/evolution/evolution-calendar-factory
147. amir      1646  0.1  1.5 183540 31720 ?        Sl   13:23   0:03
    nautilus -n
148. amir      1649  0.0  0.4  62064 10124 ?        Sl   13:23   0:00
    /usr/lib/policykit-1-gnome/polkit-gnome-authentication-agent-1
149. amir      1650  0.0  0.5  78476 10372 ?        Sl   13:23   0:00
    /usr/lib/unity-settings-daemon/unity-fallback-mount-helper
150. amir      1655  0.0  0.1  38984  4112 ?        Sl   13:23   0:00
    /usr/lib/gvfs/gvfs-udisks2-volume-monitor
151. amir      1656  0.0  2.2 149540 47368 ?        SLl  13:23   0:01
    /usr/bin/gnome-software --gapplication-service
152. amir      1668  0.0  0.8 117992 17260 ?        Sl   13:23   0:00 nm-
    applet
153. amir      1669  0.1  1.3 113172 27956 ?        Sl   13:23   0:04
    /usr/bin/vmtoolsd -n vmusr --blockFd 3
154. root      1670  0.0  0.2  54296  5188 ?        Ssl  13:23   0:00
    /usr/lib/udisks2/udisksd --no-debug
155. amir      1690  0.0  0.1  37104  2832 ?        Sl   13:23   0:00
    /usr/lib/gvfs/gvfs-gphoto2-volume-monitor
156. amir      1697  0.0  0.1  35856  2356 ?        Sl   13:23   0:00
    /usr/lib/gvfs/gvfs-goa-volume-monitor
157. amir      1705  0.0  0.1  35872  2652 ?        Sl   13:23   0:00
    /usr/lib/gvfs/gvfs-mtp-volume-monitor
158. amir      1712  0.0  0.1  78832  3792 ?        Sl   13:23   0:00
    /usr/lib/gvfs/gvfs-afc-volume-monitor
159. amir      1714  4.2  8.2 385060 170224 ?       Rsl  13:23   1:41
    compiz
160. amir      1734  0.0  1.4 148972 30620 ?        Sl   13:23   0:00
    /usr/lib/evolution/evolution-calendar-factory-subprocess --factory
    contacts --bus-name
    org.gnome.evolution.dataserver.Subprocess.Backend.Calendarx1632x2 --
    own-path
    /org/gnome/evolution/dataserver/Subprocess/Backend/Calendar/1632/2
161. amir      1743  0.0  1.4 147616 30456 ?        Sl   13:23   0:00
    /usr/lib/evolution/evolution-calendar-factory-subprocess --factory
    local --bus-name
    org.gnome.evolution.dataserver.Subprocess.Backend.Calendarx1632x3 --
    own-path
    /org/gnome/evolution/dataserver/Subprocess/Backend/Calendar/1632/3
162. amir      1745  0.0  0.1  46700  3244 ?        Sl   13:23   0:00
    /usr/lib/gvfs/gvfsd-trash --spawner :1.3 /org/gtk/gvfs/exec_spaw/0
163. amir      1748  0.0  0.4 117572  8776 ?        Sl   13:23   0:00
    /usr/lib/evolution/evolution-addressbook-factory
164. amir      1761  0.0  0.4 124704  9028 ?        Sl   13:23   0:00
    /usr/lib/evolution/evolution-addressbook-factory-subprocess --factory
    local --bus-name
    org.gnome.evolution.dataserver.Subprocess.Backend.AddressBookx1748x2 --
    own-path
    /org/gnome/evolution/dataserver/Subprocess/Backend/AddressBook/1748/2
165. amir      1812  0.0  0.3  70900  8112 ?        Sl   13:23   0:00
    zeitgeist-datahub
166. amir      1819  0.0  0.0   2364   556 ?        S    13:23   0:00
    /bin/sh -c /usr/lib/i386-linux-gnu/zeitgeist/zeitgeist-maybe-vacuum;
    /usr/bin/zeitgeist-daemon
```

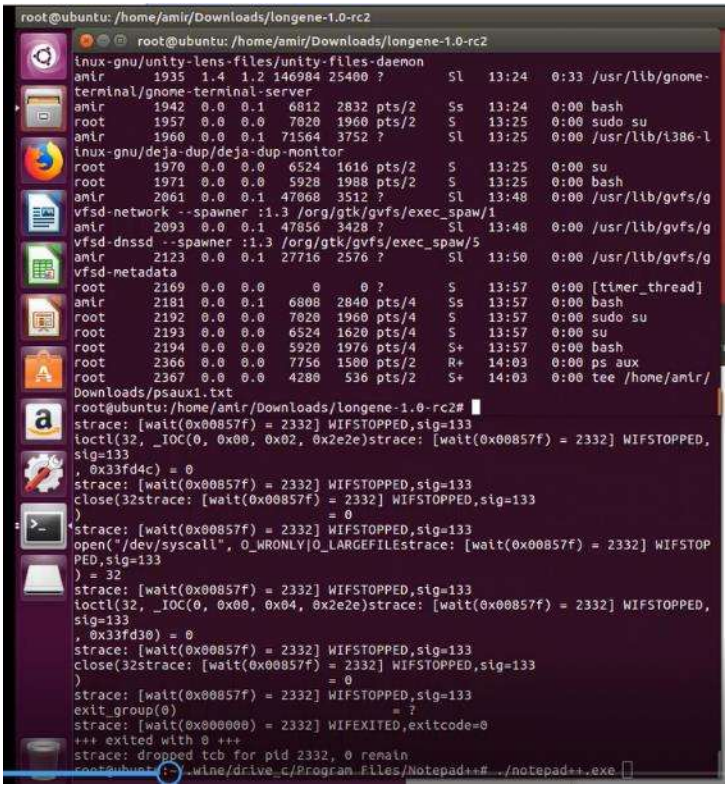
```
167.  amir      1823  0.0  0.2  52328  4480 ?        Sl   13:23   0:00
    /usr/bin/zeitgeist-daemon
168.  amir      1831  0.0  0.4  70264  8428 ?        Sl   13:23   0:00
    /usr/lib/i386-linux-gnu/zeitgeist-fts
169.  amir      1839  0.0  0.0   4428   276 ?        S    13:23   0:00
    /bin/cat
170.  amir      1863  0.0  0.6  80944 13528 ?        Sl   13:24   0:00
    update-notifier
171.  amir      1889  0.0  0.4 108720  9152 ?        Sl   13:24   0:00
    /usr/lib/i386-linux-gnu/unity-scope-home/unity-scope-home
172.  amir      1901  0.0  0.6 104808 13236 ?        Sl   13:24   0:00
    /usr/bin/unity-scope-loader applications/applications.scope
    applications/scopes.scope commands.scope
173.  amir      1902  0.0  0.3  97108  7284 ?        Sl   13:24   0:00
    /usr/lib/i386-linux-gnu/unity-lens-files/unity-files-daemon
174.  amir      1935  1.4  1.2 146984 25400 ?        Sl   13:24   0:33
    /usr/lib/gnome-terminal/gnome-terminal-server
175.  amir      1942  0.0  0.1   6812  2832 pts/2    Ss   13:24   0:00 bash
176.  root      1957  0.0  0.0   7020  1960 pts/2    S    13:25   0:00 sudo
    su
177.  amir      1960  0.0  0.1  71564  3752 ?        Sl   13:25   0:00
    /usr/lib/i386-linux-gnu/deja-dup/deja-dup-monitor
178.  root      1970  0.0  0.0   6524  1616 pts/2    S    13:25   0:00 su
179.  root      1971  0.0  0.0   5928  1988 pts/2    S    13:25   0:00 bash
180.  amir      2061  0.0  0.1  47068  3512 ?        Sl   13:48   0:00
    /usr/lib/gvfs/gvfsd-network --spawner :1.3 /org/gtk/gvfs/exec_spaw/1
181.  amir      2093  0.0  0.1  47856  3428 ?        Sl   13:48   0:00
    /usr/lib/gvfs/gvfsd-dnssd --spawner :1.3 /org/gtk/gvfs/exec_spaw/5
182.  amir      2123  0.0  0.1  27716  2576 ?        Sl   13:50   0:00
    /usr/lib/gvfs/gvfsd-metadata
183.  root      2169  0.0  0.0      0     0 ?        S    13:57   0:00
    [timer_thread]
184.  amir      2181  0.0  0.1   6808  2840 pts/4    Ss   13:57   0:00 bash
185.  root      2192  0.0  0.0   7020  1960 pts/4    S    13:57   0:00 sudo
    su
186.  root      2193  0.0  0.0   6524  1620 pts/4    S    13:57   0:00 su
187.  root      2194  0.0  0.0   5920  1976 pts/4    S+   13:57   0:00 bash
188.  root      2366  0.0  0.0   7756  1500 pts/2    R+   14:03   0:00 ps
    aux
189.  root      2367  0.0  0.0   4280   536 pts/2    S+   14:03   0:00 tee
    /home/amir/Downloads/psaux1.txt
190.
```

Nota. Este código muestra los resultados al ejecutar ps aux mientras está funcionando Notepad++ en Longene. (Diseño Propio)

Las siguientes líneas de texto mostradas anteriormente fueron el resultado del guardado del archivo ps aux. Y la veracidad de esta información se da en la figura 164 donde se muestra la misma información obtenida.

Figura 164

Resultado de ps aux

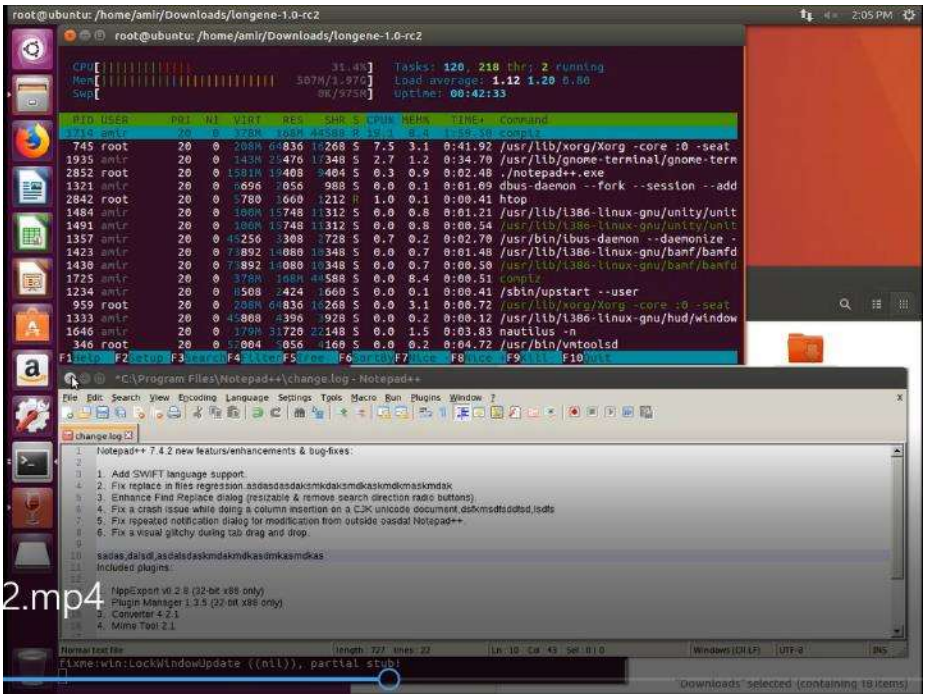


Nota. Esta imagen muestra el resultado usar ps aux. (Diseño Propio)

Lo siguiente es mostrar la monitorización de notepad++ en HTOP.

Figura 165

HTOP Notepad++ Longene



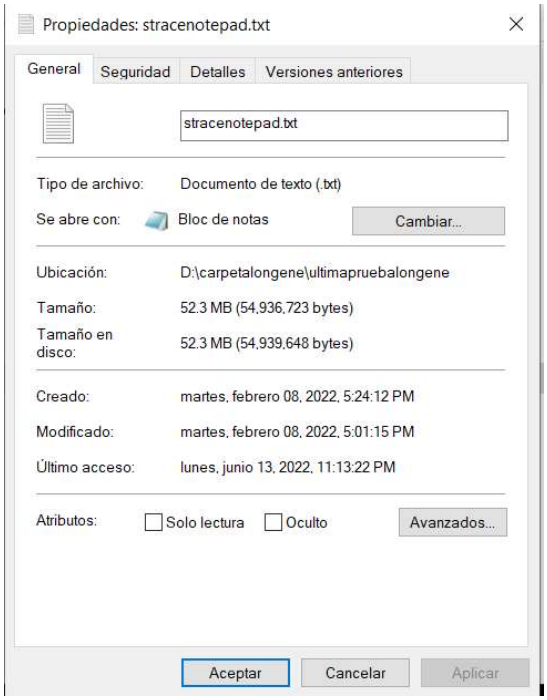
Nota. Esta imagen muestra el resultado de ejecutar HTOP cuando Notepad++ se está ejecutando en Longene. (Diseño Propio)

A diferencia de la figura 164 donde se usa ps aux, donde no se encuentra el proceso de NotePad++ o aparece con otra denominación, en la figura 165 se muestra la lectura de HTOP y el programa ./notepad++.exe esta usando 0.3% del CPU y 0.9% de la Memoria.

El archivo resultante al ejecutar strace en Notepad++ se muestra en la figura 166

Figura 166

Archivo strace de Notepad++ Longene

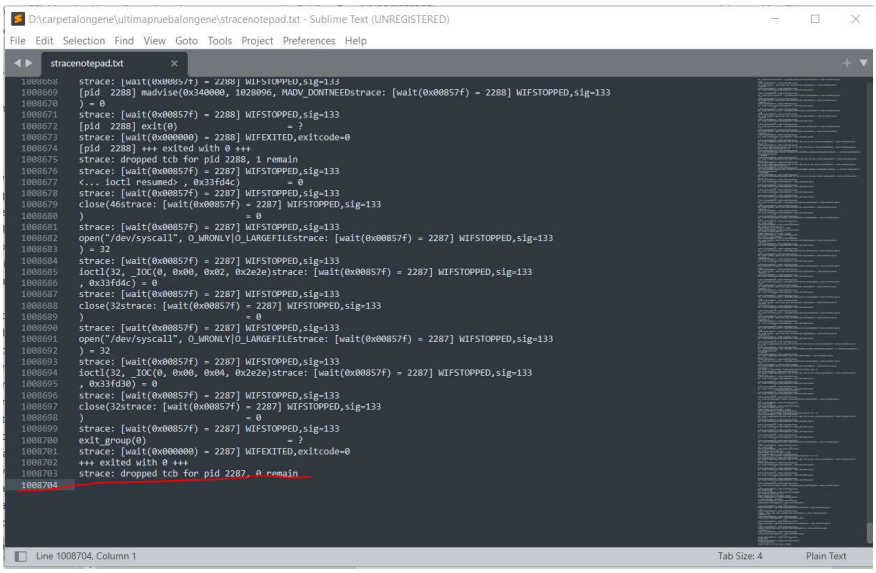


Nota. Esta imagen muestra el archivo resultante de hacer strace al ejecutar Notepad++ en Longene. (Diseño Propio)

Finalmente se muestra el archivo total obtenido por strace y la cantidad de líneas de código que este generó.

Figura 167

Archivo stracenotepad



Nota. Esta imagen muestra la cantidad de líneas del archivo resultante stracenotepad. (Diseño Propio)

En cuanto a la salida que dio el strace, teniendo de nombre el archivo “stracenotepad.txt” fue un archivo de 52.3MB y con un total de 1,008,704 líneas.

Tabla 8

Notepad++ Longene

Notepad++	
Tiempo de ejecución	44 segs
Uso de CPU	0.3%
Uso de Memoria	0.9%
Tamaño Archivo	52.3MB
Cantidad de Líneas de salida	1008704

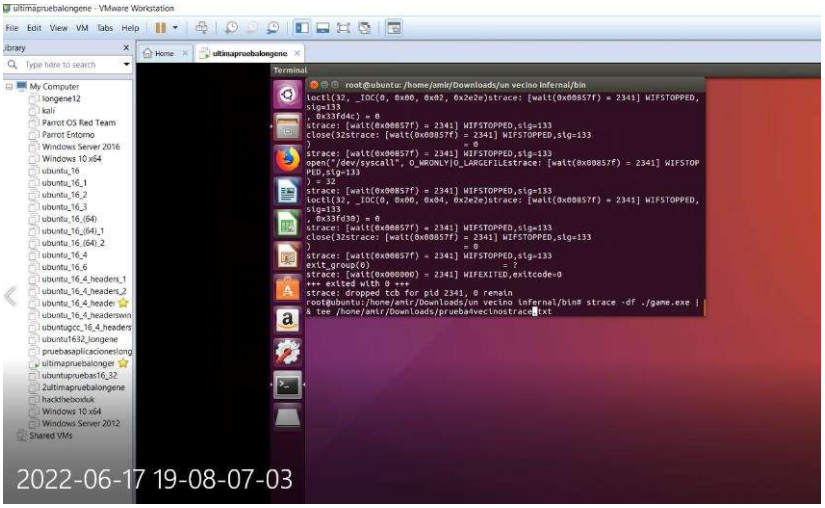
Nota. Esta tabla muestra los valores de ejecutar Notepad++ en Longene. (Diseño Propio)

4.1.3.3.2. UN VECINO INFERNAL

En el caso de este juego, no es necesaria la instalación y solamente necesita ser ejecutado, por este motivo no es necesario que se ingrese a carpetas creadas por wine para el entorno predeterminado. Por lo que solamente se necesita ejecutar el programa y se usara el comando strace -df ./game.exe |& tee /home/amir/Downloads/prueba4vecinostrace.txt

Figura 168

Ejecución Vecino Infernal Longene



Nota. Esta imagen es la ejecución Vecino Infernal(Diseño Propio)

Como se muestra en la figura 169 se puede ver el programa funcionando en Ubuntu.

Figura 169

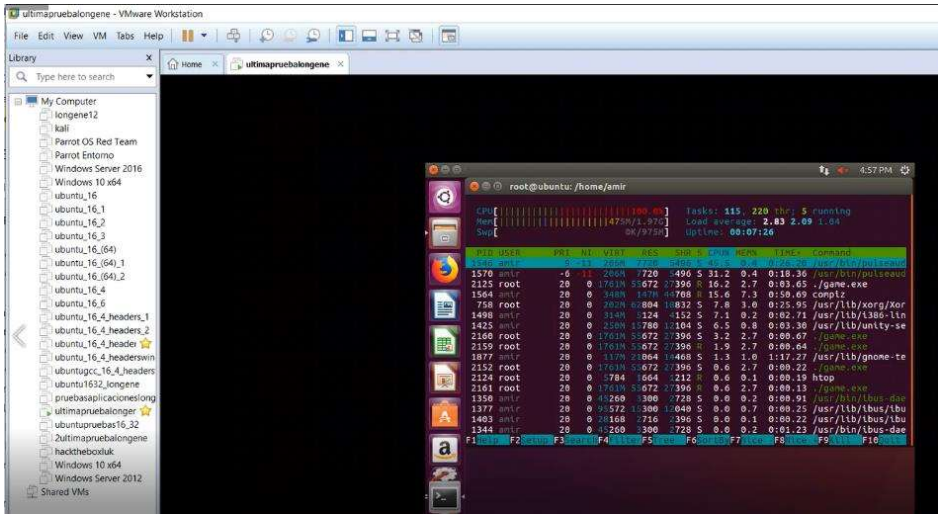
Vecino Infernal ejecutándose Longene

Nota. Esta imagen muestra Un vecino Infernal ejecutándose en Longene. (Diseño Propio)

La ejecución con el comando strace duro desde el segundo 0:15, hasta cerrar el juego en el minuto 1:34 segundos, tomando así 79 segundos para la ejecución y cierre del programa.

Figura 170

HTOP Vecino Infernal Longene

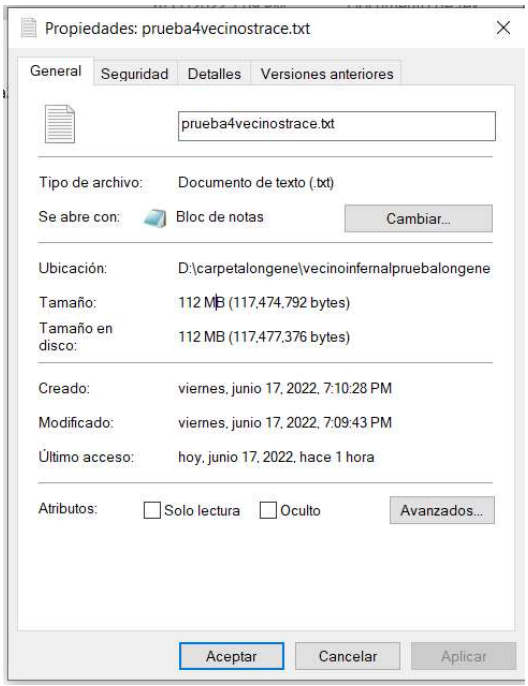


Nota. Esta imagen muestra el resultado de ejecutar HTOP cuando Vecino Infernal se está ejecutando en Longene. (Diseño Propio)

En la figura 170 se puede visualizar con la herramienta HTOP que el uso de memoria es de 16.2% en uso de CPU y 2.7% de uso de Memoria. El archivo resultante del strace es llamado prueba4vecinostrace.txt

Figura 171

Archivo strace Vecino Infernal Longene

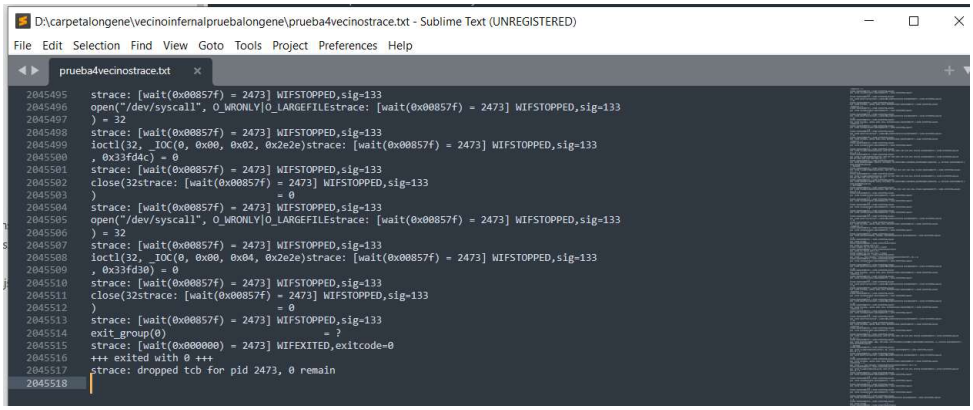


Nota. Esta imagen muestra el archivo resultante de hacer strace al ejecutar Vecino Infernal en Longene. (Diseño Propio)

Finalmente, al abrir el archivo strace generado por un vecino infernal en Longene se tienen 2045518 líneas de salida.

Figura 172

Archivo prueba4vecinostrace



Nota. Esta imagen muestra la cantidad de líneas del archivo resultante prueba4vecinostrace. (Diseño Propio)

Como se muestra en la figura 171 el peso del archivo strace es de 112 MB y en la ilustración 134 se observa que hay un total de 2,045,517 líneas.

Tabla 9

Vecino Infernal Longene

Un Vecino Infernal	
Tiempo de ejecución	75 segs
Uso de CPU	16.2%
Uso de Memoria	2.7%
Tamaño Archivo	112MB
Tiempo de ejecución	75 segs

Nota. Esta tabla muestra los valores de ejecutar Vecino Infernal en Longene. (Diseño Propio)

4.1.3.4. PRUEBAS SIN LONGENE

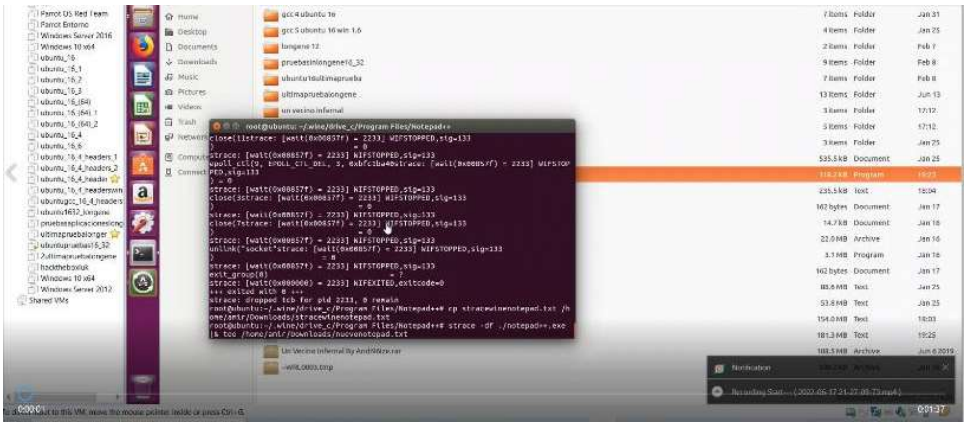
En este apartado se procederá a registrar las pruebas realizadas anteriormente, pero sin instalar el proyecto Longene, esto solo funcionará con Wine.

4.1.3.4.1. NOTEPAD++

Se comenzará con Notepad++ en el cual se realizó la instalación de la misma forma que con el apartado de 4.3.3.1 Notepad++ de Pruebas en Longene

Figura 173

Ejecución de Notepad++ sin Longene

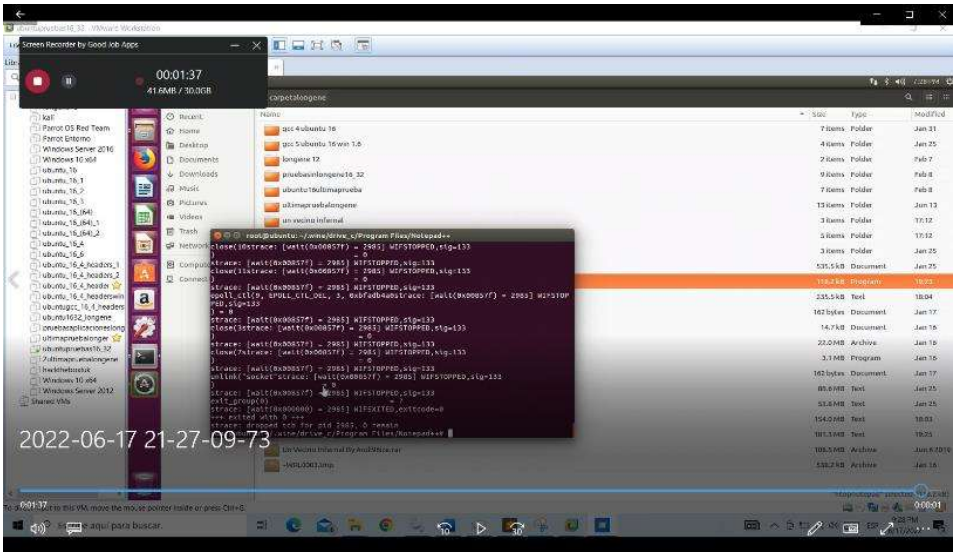


Nota. Esta imagen muestra el comando de ejecución strace ./notepad++ en la maquina sin Longene (Diseño Propio)

El programa fue cerrado luego de la ejecución en el minuto 1:37, se puede observar en la figura 174 que no se forzó el cierre del programa de ninguna forma ya que incluso aparecen mensajes de salida “exited with 0”.

Figura 174

Resultado de cierre Notepad++

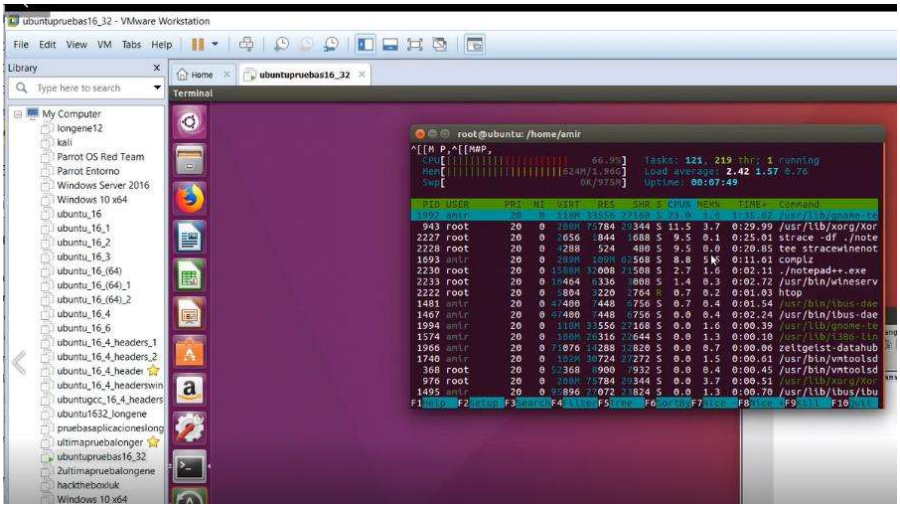


Nota. Esta imagen muestra el resultado de cierre Notepad++ en la maquina sin Longene. (Diseño Propio)

La ejecución con el comando strace duro desde el segundo 0:01, hasta cerrar el programa en el minuto 1:36 segundos, tomando así 95 segundos para la ejecución y cierre del programa.

Figura 175

HTOP Notepad++ sin Longene



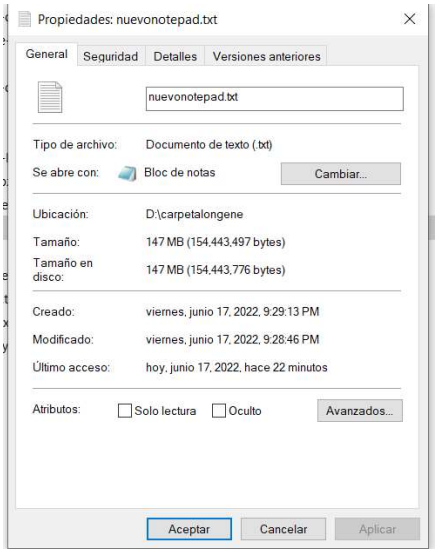
Nota. Esta imagen muestra el resultado de ejecutar HTOP cuando Notepad++ se está ejecutando sin Longene (Diseño Propio)

En la figura 175 se muestra el HTOP ejecutándose en Wine sin Longene con el Notepad++ al costado derecho con id 2230.

En la figura 176 se puede visualizar con la herramienta HTOP que el uso de memoria es de 2.7% en uso de CPU y 1.6% de uso de Memoria.

Figura 176

Archivo strace Notepad++ sin Longene

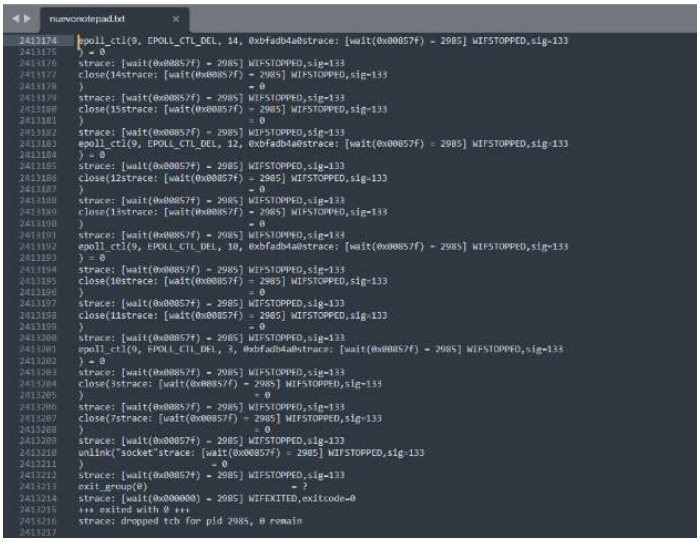


Nota. Esta imagen muestra el archivo nuevonotepad sin Longene. (Diseño Propio)

Así se puede mostrar el archivo del strace al ejecutar Notepad++

Figura 177

Archivo nuevonotepad



Nota. Esta imagen muestra las líneas de ejecutar Notepad++ en la maquina sin Longene. (Diseño Propio)

Como se muestra en la figura 176 el peso del archivo strace es de 147 MB y en la ilustración 177 se observa que hay un total de 2,413,216 líneas.

Tabla 10

Notepad++ sin Longene

Notepad++ sin Longene	
Tiempo de ejecución	95 segs
Uso de CPU	2.7%
Uso de Memoria	1.6%
Tamaño Archivo	147MB
Cantidad de Líneas de salida	2413216

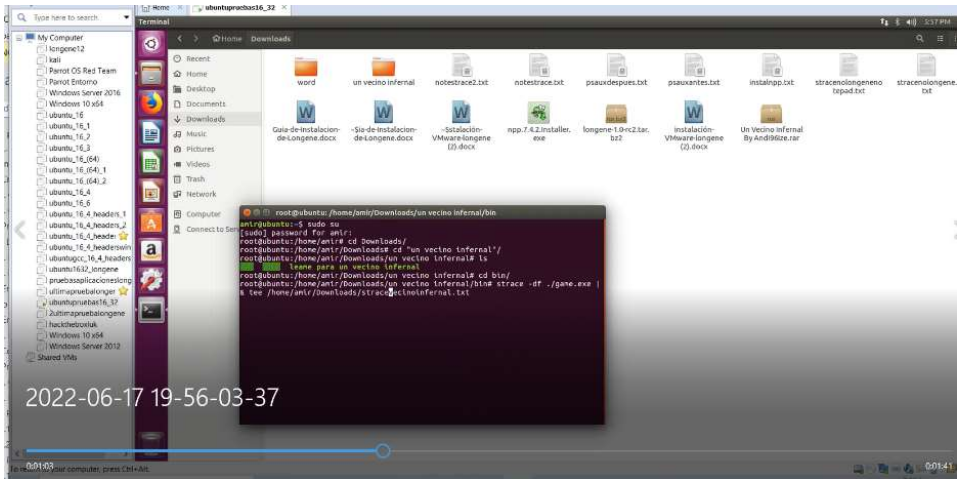
Nota. Esta tabla muestra los valores de ejecutar Notepad++ en la maquina sin Longene. (Diseño Propio)

4.1.3.4.2. UN VECINO INFERNAL

En el caso de este juego, no es necesaria la instalación y solamente necesita ser ejecutado, por este motivo no es necesario que se ingrese a las carpetas creadas por wine para el entorno predeterminado. Y en esta ocasión ejecutare desde el entorno sin Longene.

Figura 178

Ejecución Vecino infernal sin Longene

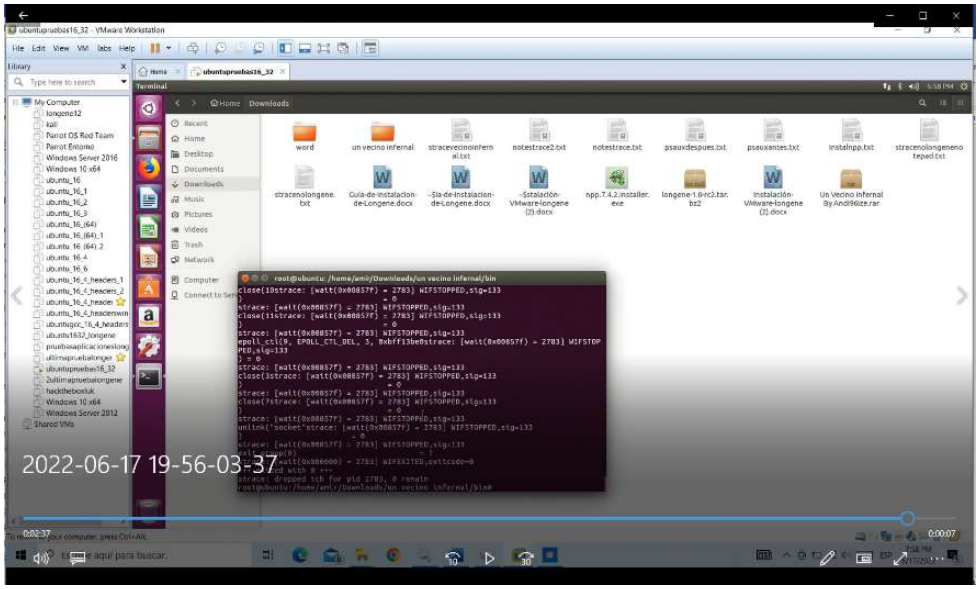


Nota. Esta imagen es la ejecución Vecino Infernal en la maquina de pruebas sin Longene. (Diseño Propio)

Finalmente, para abrir el programa y cerrarlo de forma natural se llego a esto en un total de 94 segundos.

Figura 179

Resultado cierre vecino infernal sin Longene

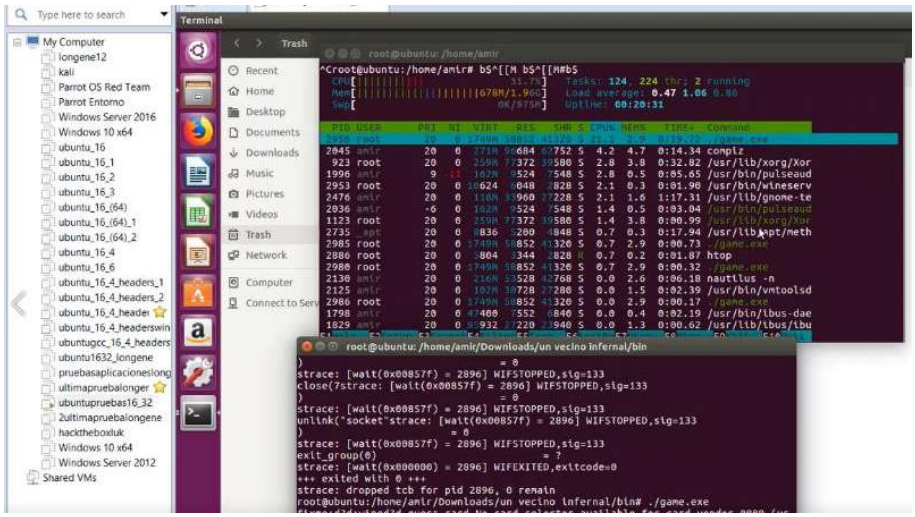


Nota. Esta imagen muestra el resultado de cierre de un vecino infernal sin Longene. (Diseño Propio)

La ejecución con el comando strace duro desde el segundo 1:03, hasta cerrar el juego en el minuto 2:37 segundos, tomando así 94 segundos para la ejecución y cierre del programa.

Figura 180

HTOP Vecino infernal sin Longene

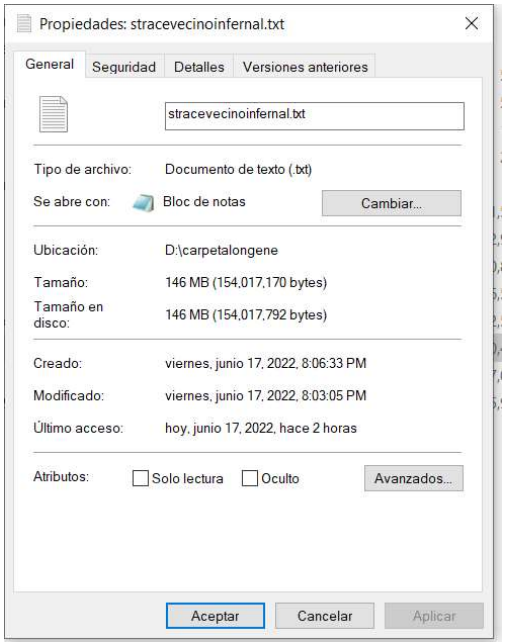


Nota. Esta imagen muestra el resultado de ejecutar HTOP cuando Vecino Infernal se está ejecutando sin Longene. (Diseño Propio)

En la figura 180 se puede visualizar con la herramienta HTOP que el uso de memoria es de 21.1% en uso de CPU y 2.9% de uso de Memoria. El archivo obtenido del strace de un vecino infernal en Wine sin Longene fue nombrado stracevecinoinfernal.txt.

Figura 181

Archivo strace Vecino infernal sin Longene

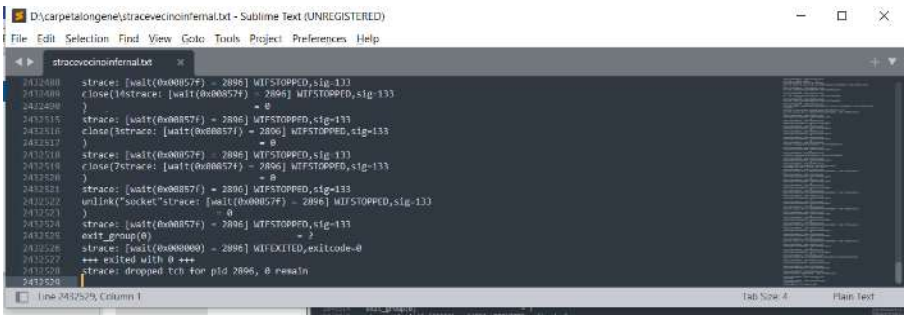


Nota. Esta imagen muestra el archivo resultante de hacer strace al ejecutar Vecino Infernal sin Longene. (Diseño Propio)

Al abrir el archivo stracevecinoinfernal.txt se obtiene un total de 243529 lineas.

Figura 182

Archivo stracevecinoinferal sin Longene



Nota. Esta imagen muestra la cantidad de líneas del archivo resultante stracevecinoinferal. (Diseño Propio)

Como se muestra en la figura 181 el peso del archivo strace es de 146 MB y en la figura 182 se observa que hay un total de 2,432,529 líneas.

Tabla 11

Vecino Infernal sin Longene

Un Vecino Infernal sin Longene	
Tiempo de ejecución	94 segs
Uso de CPU	21.1%
Uso de Memoria	2.9%
Tamaño Archivo	146MB
Cantidad de Líneas de salida	2432529

Nota. Esta tabla muestra los valores de ejecutar Vecino Infernal en Longene. (Diseño Propio)

4.1.4. CONTRASTE DE RESULTADOS

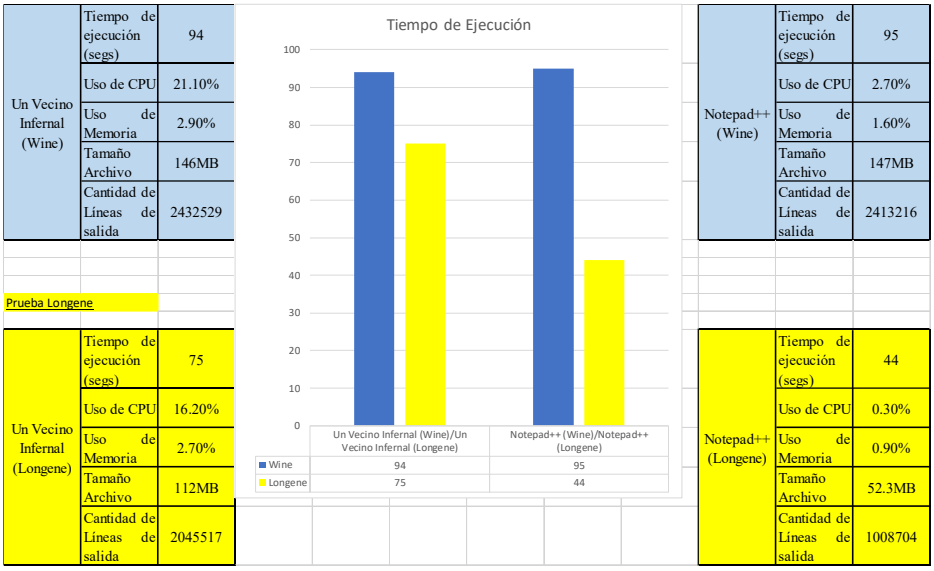
Para los contrastes de resultados se generaron gráficos con los datos obtenidos mediante las pruebas, por este motivo se dividirán los resultados según lo que se requiera para justificar la hipótesis contemplada en el proyecto de investigación.

Prueba de Tiempo de Ejecución

El tiempo de ejecución es la cantidad de tiempo que toma al sistema operativo para ejecutar el programa y después cerrarlo. El resultado obtenido en este punto se encuentra en el siguiente gráfico.

Figura 183

Pruebas de Tiempo de Ejecución



Un Vecino Infernal (Wine)	Tiempo de ejecución (segs)	94
	Uso de CPU	21.10%
	Uso de Memoria	2.90%
	Tamaño Archivo	146MB
	Cantidad de Líneas de salida	2432529
Notepad++ (Wine)	Tiempo de ejecución (segs)	95
	Uso de CPU	2.70%
	Uso de Memoria	1.60%
	Tamaño Archivo	147MB
	Cantidad de Líneas de salida	2413216

Prueba Longene

Un Vecino Infernal (Longene)	Tiempo de ejecución (segs)	75
	Uso de CPU	16.20%
	Uso de Memoria	2.70%
	Tamaño Archivo	112MB
	Cantidad de Líneas de salida	2045517
Notepad++ (Longene)	Tiempo de ejecución (segs)	44
	Uso de CPU	0.30%
	Uso de Memoria	0.90%
	Tamaño Archivo	52.3MB
	Cantidad de Líneas de salida	1008704

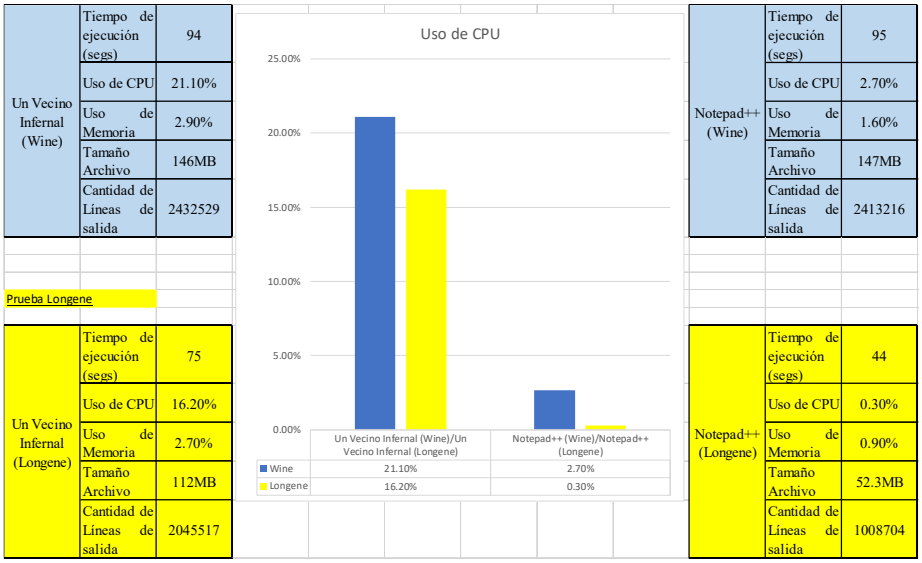
Nota. Este grafico muestra los resultados de las pruebas basadas en Tiempo de Ejecución. (Diseño Propio)

Prueba Uso de CPU

Para la prueba de “Uso de CPU”, se considera el valor uso de CPU y el tamaño del archivo generado al ejecutar cada programa. Los resultados obtenidos en este punto se encuentran en los siguientes gráficos.

Figura 184

Pruebas de Uso de CPU



Un Vecino Infernal (Wine)	Tiempo de ejecución (segs)	94
	Uso de CPU	21.10%
	Uso de Memoria	2.90%
	Tamaño Archivo	146MB
	Cantidad de Líneas de salida	2432529
Notepad++ (Wine)	Tiempo de ejecución (segs)	95
	Uso de CPU	2.70%
	Uso de Memoria	1.60%
	Tamaño Archivo	147MB
	Cantidad de Líneas de salida	2413216

Prueba Longene

Un Vecino Infernal (Longene)	Tiempo de ejecución (segs)	75
	Uso de CPU	16.20%
	Uso de Memoria	2.70%
	Tamaño Archivo	112MB
	Cantidad de Líneas de salida	2045517
Notepad++ (Longene)	Tiempo de ejecución (segs)	44
	Uso de CPU	0.30%
	Uso de Memoria	0.90%
	Tamaño Archivo	52.3MB
	Cantidad de Líneas de salida	1008704

Nota. Este grafico muestra los resultados de las pruebas basadas en Uso de CPU. (Diseño Propio)

El tamaño de archivo hace referencia a los resultados de strace al ejecutar cada programa, desde que se abren hasta cerrarlos.

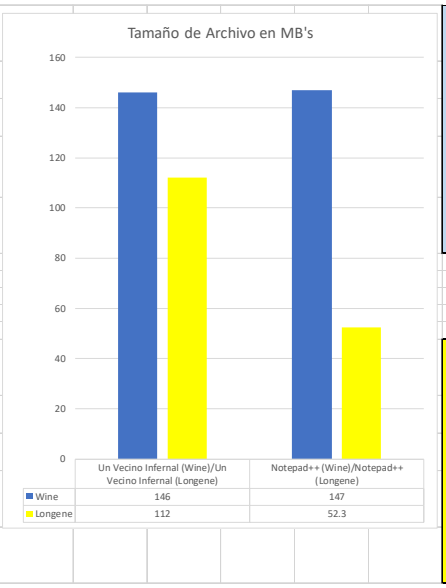
Figura 185

Pruebas de Tamaño de Archivo

Un Vecino Infernal (Wine)	Tiempo de ejecución (segs)	94
	Uso de CPU	21.10%
	Uso de Memoria	2.90%
	Tamaño Archivo (MB's)	146
	Cantidad de Líneas de salida	2432529
Notepad++ (Wine)	Tiempo de ejecución (segs)	95
	Uso de CPU	2.70%
	Uso de Memoria	1.60%
	Tamaño Archivo (MB's)	147
	Cantidad de Líneas de salida	2413216

Prueba Longene

Un Vecino Infernal (Longene)	Tiempo de ejecución (segs)	75
	Uso de CPU	16.20%
	Uso de Memoria	2.70%
	Tamaño Archivo (MB's)	112
	Cantidad de Líneas de salida	2045517
Notepad++ (Longene)	Tiempo de ejecución (segs)	44
	Uso de CPU	0.30%
	Uso de Memoria	0.90%
	Tamaño Archivo (MB's)	52.3
	Cantidad de Líneas de salida	1008704



Nota. Este grafico muestra los resultados de las pruebas basadas en Tamaño de Archivo. (Diseño Propio)

Prueba Uso de Memoria

Para la prueba de “Uso de Memoria”, se considera el valor uso de Memoria y la cantidad de líneas de salida obtenidas en por el archivo generado al ejecutar cada programa. Los resultados obtenidos en este punto se encuentran en los siguientes gráficos.

Figura 186

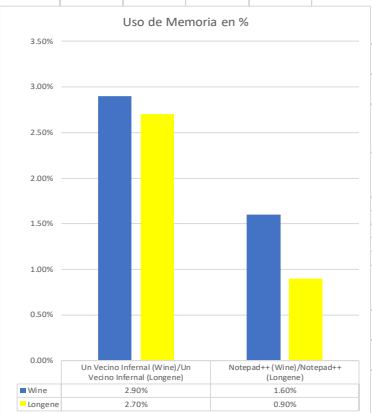
Pruebas de Uso de Memoria

Prueba Wine sin Longene

Un Vecino Infernal (Wine)	Tiempo de ejecución (segs)	94
	Uso de CPU	21.10%
	Uso de Memoria	2.90%
	Tamaño Archivo (MB's)	146
	Cantidad de Líneas de salida	2432529
Notepad++ (Wine)	Tiempo de ejecución (segs)	95
	Uso de CPU	2.70%
	Uso de Memoria	1.60%
	Tamaño Archivo (MB's)	147
	Cantidad de Líneas de salida	2413216

Prueba Longene

Un Vecino Infernal (Longene)	Tiempo de ejecución (segs)	75
	Uso de CPU	16.20%
	Uso de Memoria	2.70%
	Tamaño Archivo (MB's)	112
	Cantidad de Líneas de salida	2045517
Notepad++ (Longene)	Tiempo de ejecución (segs)	44
	Uso de CPU	0.30%
	Uso de Memoria	0.90%
	Tamaño Archivo (MB's)	52.3
	Cantidad de Líneas de salida	1008704

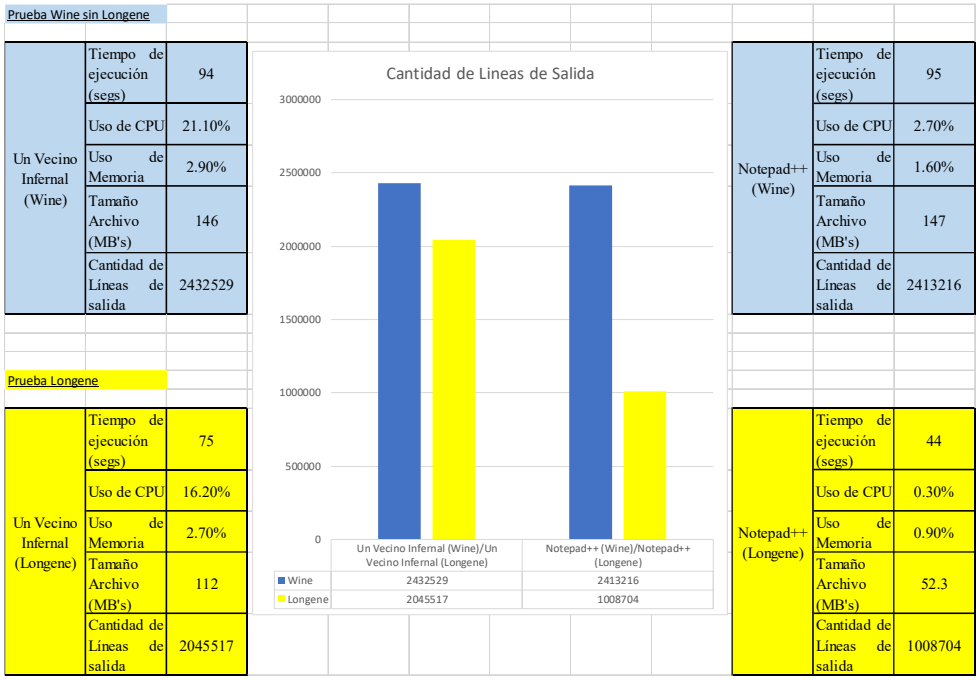


Nota. Este grafico muestra los resultados de las pruebas basadas en Uso de Memoria (Diseño Propio)

La cantidad de líneas de salida son el total de líneas de salida que tiene cada archivo generado por strace.

Figura 187

Pruebas Cantidad de Lineas de Salida



Prueba Wine sin Longene		
Un Vecino Infernal (Wine)	Tiempo de ejecución (segs)	94
	Uso de CPU	21.10%
	Uso de Memoria	2.90%
	Tamaño Archivo (MB's)	146
	Cantidad de Líneas de salida	2432529
Notepad++ (Wine)	Tiempo de ejecución (segs)	95
	Uso de CPU	2.70%
	Uso de Memoria	1.60%
	Tamaño Archivo (MB's)	147
	Cantidad de Líneas de salida	2413216

Prueba Longene		
Un Vecino Infernal (Longene)	Tiempo de ejecución (segs)	75
	Uso de CPU	16.20%
	Uso de Memoria	2.70%
	Tamaño Archivo (MB's)	112
	Cantidad de Líneas de salida	2045517
Notepad++ (Longene)	Tiempo de ejecución (segs)	44
	Uso de CPU	0.30%
	Uso de Memoria	0.90%
	Tamaño Archivo (MB's)	52.3
	Cantidad de Líneas de salida	1008704

Nota. Este grafico muestra los resultados de las pruebas basadas en Cantidad de Líneas de Salida. (Diseño Propio)

4.2. CUMPLIMIENTO DE OBJETIVOS

Como se demostró a lo largo de esta investigación se desarrollaron los puntos propuestos:

- Se llego a explicar cómo funciona el proyecto Longene y la razón por la cual este fue desarrollado. Siendo esta una alternativa la cual permita la ejecución de programas Windows en Linux usando un modulo que se carga directamente en el kernel de Linux
- Se realizo la implementación en la versión 16.04 de Ubuntu, siendo esta versión superior a la versión para la cual se desarrolló este proyecto. A lo largo de la realización de esta implementación se encontraron muchos problemas e

inconvenientes que no pudieron ser obtenidos directamente de la fuente de los que realizaron este proyecto, aun así, valiéndose de la experticia se logran sortear los inconvenientes y finalmente cargar el módulo de la forma más natural posible.

- Finalmente se pudo evaluar desde distintas pruebas las variables consideradas para este proyecto. Haciendo las comparativas mediante herramientas las cuales se tienen directamente en Linux como el comando strace o usando la herramienta HTOP la cual simplemente necesita descargarse e instalarla, además de esto se pudo justificar las variables consideradas tomando de referencia otros proyectos de investigación los cuales tienen como objetivo central la comparativa de rendimiento.

4.3. CONTRIBUCIÓN

A pesar de tener muchas complicaciones durante el desarrollo de un proyecto de esta magnitud se pudieron sortear estos inconvenientes demostrando así que con la base de la enseñanza de Sistemas Operativos la cual fue obtenida durante la educación universitaria, además de un sin número de investigaciones de distintas fuentes y experiencias en otras áreas, se pudo lograr el objetivo propuesto para el desarrollo de la tesis.

El tener una herramienta como esta permite de distintas formas poder continuar con investigaciones sobre los procesos de ejecución y la interpretación de comandos dentro del Kernel de Linux, gracias a esto se pueden desarrollar muchas investigaciones con distintos enfoques; desde la perspectiva del proyecto se podrían realizar desde pruebas ofensivas de hackeo como la vulneración de aplicaciones de Windows específicas hasta realizar sobrecargas de buffer o alternativas mucho menos convencionales.

Otras alternativas que ha considerar bastante interesantes es usar esto en entornos académicos para que los alumnos se acostumbren a usar software libre basado en Linux y poder ejecutar sus programas Windows de forma tal que pierdan el miedo de usar este tipo de sistema operativo y poco a poco les cueste menos el trabajar con comandos de forma natural.

CONCLUSIONES

- El proyecto Longene sin duda fue un proyecto ambicioso que se centró en las llamadas al sistema y la interpretación de las mismas para permitir la ejecución de programas Windows en Linux. en esta tesis se observó que para que ejecute un programa debe comprobar si este es win32, este tipo de archivos tiene la extensión .exe, primero se debe comprobar el programa, luego se debe cargar en el módulo module/ke/$path siendo este el modulo principal que debe ser cargado con insmod al momento de instalarlo todo esto con una estructura de formato PE_format, después se llama a la función de sistema EXECVE que comprueba si es un binario ejecutable y se envia load_pe binary para la comprobación de win32 ya mencionada, luego se deben realizar ciertas instrucciones incluyendo el mapeo de pila y el espacio de memoria que debe ser asignado para la ejecución de ese programa en específico, después de muchas interacciones dentro de los dlls se obtiene un syscallexit(), para luego llegar a la funcion de WINE KiUSerApcDispatcher, a partir de esto se llega al interpretador ld-linux.so. agregando los enlaces necesarios para la ejecución de comandos, haciendo después de bastantes lineas una llamada al sistema mediante APC, demorando un poco más la ejecución la primera vez para establecer el usuario el cual ejecutara el programa.
- Para desarrollar una investigación de esta índole se debe usar una metodología de prueba y error, ya que no existía la suficiente información necesaria, debido a que el proyecto fue abandonado por sus creadores, además se obtuvo que para poder implementar Longene en Ubuntu16 se debe instalar un kernel antiguo al por defecto, en este caso el 3.2.0-23-generic-pae, después instalar la versión 1.6 de Longene ya que era la única accesible mediante repositorio a la fecha de

esta investigación, después instalar las librerías gcc, g++, bison,etc; para descomprimir Longene, instalarlo y finalmente cargar el módulo de instalación denominado unifiedkernel.ko

- Para la evaluación del proyecto se concluyó que se debían tener en consideración las variables que fueron mencionadas en otros artículos enfocados en la medición de rendimiento, tomando estos valores como importancia de pruebas se obtuvo que Longene es superior en todas y cada una de las pruebas propuestas usando de guía estos artículos, así pues, Longene es superior frente al uso directo de Wine.

RECOMENDACIONES

- Para futuros proyectos los cuales se realicen a partir de esta investigación, se recomienda el aprender el funcionamiento básico del modelo de ejecución de Longene, de esta forma poder incluso mejorar el código base que es obtenible en un repositorio existente a la actualidad en GitHub. Puede existir la posibilidad de crear un proyecto propio a base de Longene, por sobretodo recomendaría revisar el archivo de carga de programas.
- Debido a que el proceso de poder instalar Longene en Ubuntu 16 fue tan exhaustivo, se recomienda seguir el proceso de instalación al pie de la letra para evitar futuros errores o búsquedas innecesarias de información, siendo el más relevante el cargar un kernel (header) antiguo.
- Se recomienda para futuros proyectos basados en este el utilizar pruebas con otros programas que usen verificación de computadora, para así tener un contexto más amplio, se considera que una de las travas importantes es el no poder cargar el winecfg para la configuración de entorno, ya que si no estuviera deshabilitado al instalar Longene se podría usar este modo en otra instalación de aplicaciones como Office.

GLOSARIO DE TÉRMINOS

SO (SISTEMA OPERATIVO)

Los sistemas operativos consisten en interfaces gráficas, entornos de escritorio o gestores de ventanas, que brindan al usuario una representación gráfica de los procesos en marcha. También puede ser una línea de comandos, es decir, un conjunto de instrucciones ordenado en base a su prioridad y que funciona en base a comandos y órdenes introducidos por el usuario.

Las primeras versiones de las computadoras no poseían sistemas operativos. En la década de los sesenta los ordenadores usaban procesamientos por lotes. Fue durante estos años cuando comenzaron a desarrollarse los sistemas operativos. Si bien a partir de los ochenta ya habían comenzado a surgir algunos muy conocidos, fue a partir de los noventa cuando estos programas comenzaron a ser más flexibles y fuertes. Uno de los grandes hitos fue el lanzamiento de Windows 95.

WINE

Wine (acrónimo recursivo en inglés para Wine Is Not an Emulator, que significa «Wine no es un emulador») es una reimplementación de la interfaz de programación de aplicaciones de Win16 y Win32 para sistemas operativos basados en Unix. Permite la ejecución de

programas diseñados para MS-DOS, y las versiones de Microsoft Windows 3.11, 95, 98, Me, NT, 2000, XP, Vista, 7, 8 y 10. El nombre Wine inicialmente fue un acrónimo para WINdows Emulator.1 Este significado fue cambiado posteriormente al acrónimo recursivo actual.

Wine provee de:

- Un conjunto de herramientas de desarrollo para portar código fuente de aplicaciones Windows a Unix.
- Un cargador de programas, el cual permite que muchas aplicaciones para Windows 2.0/3.x/9X/ME/NT/2000/XP/Vista/7 y 8 se ejecuten sin modificarse en varios sistemas operativos Unix como macOS, BSD y Unix-like como GNU/Linux, Solaris

EMULADOR

En informática, un emulador es un software que permite ejecutar programas o videojuegos en una plataforma (sea una arquitectura de hardware o un sistema operativo) diferente de aquella para la cual fueron escritos originalmente. A diferencia de un simulador, que solo trata de reproducir el comportamiento del programa, un emulador trata de modelar de forma precisa el dispositivo de manera que este funcione como si estuviese siendo usado en el aparato original.

KERNEL

En informática, un núcleo o kernel (de la raíz germánica Kern, núcleo, hueso) es un software que constituye una parte fundamental del sistema operativo, y se define como la parte que se ejecuta en modo privilegiado (conocido también como modo núcleo).

LENGUAJE MAQUINA

El lenguaje de máquina o código máquina es el sistema de códigos directamente interpretable por un circuito microprogramable, como el microprocesador de una computadora o el microcontrolador de un autómata. Este lenguaje está compuesto por un conjunto de instrucciones que determinan acciones a ser tomadas por la máquina. Un programa consiste en una cadena de estas instrucciones más un conjunto de datos sobre el cual se trabaja. Estas instrucciones son normalmente ejecutadas en secuencia, con eventuales cambios de flujo causados por el propio programa o eventos externos. El lenguaje de máquina es específico de la arquitectura de la máquina, aunque el conjunto de instrucciones disponibles pueda ser similar entre arquitecturas distintas.

GUI (GRAPHIC USER INTERFACE)

Una GUI es una interfaz de usuario gráfica que utiliza el conjunto de imágenes y objetos gráficos para representar la información y acciones en la interfaz, en lugar de una visualizar texto en la computadora. El término se fue considerado porque las primeras interfaces de usuario interactivas a las computadoras no eran gráficas; eran pantallas las cuales mostraban texto y teclado, además de que solamente se tenían comandos los cuales venían en el manual de instrucciones además de que las respuestas que daba la computadora que eran infamemente breves. Un ejemplo es la interfaz de comandos del sistema operativo DOS antes de la llegada las GUI. Un paso intermedio en las interfaces de usuario entre la interfaz de línea de comandos y la interfaz gráfica de usuario era la interfaz basada en menús no gráfica, que le permitía interactuar utilizando un ratón en lugar de tener que escribir comandos del teclado.

BIBLIOGRAFÍA

A., D. (s.f.). *Htop, cómo instalarlo y controlar los procesos en Ubuntu 17.10*. Obtenido de https://ubunlog.com/htop-controlar-procesos-ubuntu/

A., E. (9 de Junio de 2014). *https://openwebinars.net/.* Obtenido de https://openwebinars.net/blog/que-es-dual-boot-y-por-que-usarlo/

Ambos, D. (19 de Mayo de 2021). *Adding arguments and options to your Bash scripts*. Obtenido de https://www.redhat.com/sysadmin/arguments-options-bash-scripts

ATOM. (3 de Enero de 2017). *Ubuntu Linux How to set older kernel as default in GRUB*. Obtenido de Youtube: https://www.youtube.com/watch?v=1cCLa4qusFo&ab_channel=ATOM

ATOM. (3 de Enero de 2017). *Ubuntu Linux How to set older kernel as default in GRUB*. Obtenido de https://www.youtube.com/watch?v=1cCLa4qusFo&ab_channel=ATOM

Batool, W. (Abril de 2021). *Linux df Command*. Obtenido de https://linuxhint.com/use-linux-df-command/

Bowen, K. (13 de Abril de 2017). *How to install gcc-4.8*. Obtenido de https://askubuntu.com/questions/271388/how-to-install-gcc-4-8

Búzdar, P. (Febrero de 2021). *What is build-essential Ubuntu, how to install and use it?* Obtenido de https://linuxhint.com/install-build-essential-ubuntu/

Code Weavers. (25 de Abril de 2018). *Licensing*. Obtenido de https://wiki.winehq.org/Licensing

Commander♦, B. (08 de Febrero de 2016). *How do I save terminal output to a file?* Obtenido de https://askubuntu.com/questions/420981/how-do-i-save-terminal-output-to-a-file

ComputerNetworkingNotes. (21 de Abril de 2019). *ps aux command and ps command explained*. Obtenido de https://www.computernetworkingnotes.com/linux-tutorials/ps-aux-command-and-ps-command-explained.html

Conghui, H., Jing, C., Li, Z., & Qiao, L. (2012). Performance Evaluation of Virtualization Technologies for Windows Programs Running on Linux Operating System. *International Conference on Network Computing and Information Security* (págs. 759-766). Berlin: Springer, Berlin, Heidelberg.

CONSULTING, J. V. (20 de Diciembre de 2015). *https://www.jmgvirtualconsulting.com/.* Obtenido de https://www.jmgvirtualconsulting.com/servicios-virtualizacion/los-5-mayores-problemas-de-la-virtualizacion-de-servidores/

custer_flux. (9 de Junio de 2014). *https://openwebinars.net/.* Obtenido de https://openwebinars.net/blog/que-es-dual-boot-y-por-que-usarlo/

DAINTITH, J. (01 de Octubre de 2020). *Encyclopedia.com*. Obtenido de Compatibility: https://www.encyclopedia.com/computing/dictionaries-thesauruses-pictures-and-press-releases/compatibility

Duggal, J. K., & El-Sharkawy, M. (2020). Shallow SqueezeNext: Real Time Deployment on Bluebox2.0 with 272KB Model Size. *Journal of Electrical and Electronic Engineering*, 127-136.

Duncan, R., & Schreuders, Z. C. (2018). Security implications of running windows software on a Linux system using Wine: a malware analysis study. *Journal of Computer Virology and Hacking Techniques volume*, 39-60.

Gite, V. (22 de Abril de 2020). *How to: Linux / UNIX create soft link with ln command*. Obtenido de https://www.cyberciti.biz/faq/creating-soft-link-or-symbolic-link/

Griffon. (21 de March de 2021). *Ubuntu – Ubuntu 16.04 End of Standard Support in April 2021*. Obtenido de http://c-nergy.be/blog/?p=16584

Hope, C. (11 de Junio de 2021). *The bash shell*. Obtenido de https://www.computerhope.com/unix/ubash.htm

Kili, A. (25 de Octubre de 2017). *10 Strace Commands for Troubleshooting and Debugging Linux Processes*. Obtenido de https://www.tecmint.com/strace-commands-for-troubleshooting-and-debugging-linux/

Mamani Condori, E. W. (2016). Desarrollo de una distribución personalizada del sistema operativo gnu/linux ubuntu 15 para la UPSC. *(Titulacion de Licenciatura)*. UPSC, Puno, Puno, Perú.

maslinux. (15 de Mayo de 2018). *https://maslinux.es/.* Obtenido de https://maslinux.es/que-es-eso-de-wine/

MukkeshMckenzie. (22 de Mayo de 2019). *insmod command in Linux with examples*. Obtenido de https://www.geeksforgeeks.org/insmod-command-in-linux-with-examples/

Potdara, A., Narayan, Kengond, S., & Mulla, M. (2020). Performance Evaluation of Docker Container and Virtual Machine. *Third International Conference on Computing and Network Communications (CoCoNet'19)* (págs. 1419-1428). Kerala, India: ScienceDirect.

programmerclick. (2020). *Linux Kernel Implement PE loader - Análisis de código fuente de LONGENE*. Obtenido de https://programmerclick.com/article/88572140869/

Simplilearn. (14 de Setiembre de 2020). *Simplilearn*. Obtenido de https://www.simplilearn.com/feasibility-study-article

Skendzic, A., Kovacic, B., & Jugo, I. (2011). Decreasing information technology expenses by using emulators on Windows and Linux platforms. *2011 Proceedings of the 34th International Convention MIPRO*.

Skendzic, A., Kovacic, B., & Jugo, I. (2011). Decreasing information technology expenses by using emulators on Windows and Linux platforms. *MIPRO, Proceedings of the International Convention.* Opatija, Croatia: IEEE.

Statcounter. (19 de Diciembre de 2019). *https://gs.statcounter.com/.* Obtenido de https://gs.statcounter.com/os-market-share/desktop/united-states-of-america/#monthly-200901-201905

System, G. O. (06 de Agosto de 2014). *GNU Bison*. Obtenido de https://www.gnu.org/software/bison/

TANENBAUM, A. S. (2009). *SISTEMAS OPERATIVOS MODERNOS* (Tercera ed.). (A. V. Romero Elizondo, Trad.) Amsterdam: Pearson Educacion.

Tejada Castillo, L. M., & Torres Quezada, A. M. (2014). Diseño de un modelo lógico de virtualización cliente servidor para organizaciones empresariales. *(Tesis de Licenciatura).* UNITRU, Trujillo, La Libertad, Perú.

Velasco, R. (23 de Enero de 2019). *https://www.softzone.es/.* Obtenido de https://www.softzone.es/2019/01/23/wine-4-disponible/

Velasco, R. (13 de Agosto de 2020). *SoftZoneES*. Obtenido de https://www.softzone.es/programas/linux/kernel-nucleo-linux/

VELÁZQUEZ, E. (10 de Enero de 2009). *https://www.pymesyautonomos.com/.* Obtenido de https://www.pymesyautonomos.com/tecnologia/que-es-la-virtualizacion

VMware. (27 de Mayo de 2021). *Understanding Clones in VMware vSphere 7*. Obtenido de https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/performance/cloning-vSphere7-perf.pdf

Yadav, G. (03 de Abril de 2019). *strace: See what system calls kernel is executing.* Obtenido de

https://www.learnsteps.com/strace-see-what-system-calls-kernal-is-executing/

Yoyo. (23 de Agosto de 2019). *Kernel Headers y build-essential en Debian Ubuntu y basadas*. Obtenido de https://salmorejogeek.com/2019/08/23/kernel-headers-y-build-essential-en-debian-ubuntu-y-basadas/

Zapana Mamani, N. Y. (2014). Personalización del sistema operativo gnu/linux ubuntu 13 para mejorar el acceso a internet satelital con Gilat. *(Tesis de Titulacion).* UNAP, Juliaca, Puno, Perú.

日出1.1. (18 de Febrero de 2019). *在 Ubuntu 18.04 上使用 Wine 安装 Office 2016 ProPlus*. Obtenido de https://blog.sunriseydy.top/technology/linux/wine-install-office2016/