41132
14

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
CAMPUS ARAGÓN

"DESARROLLO DE UNA METODOLOGÍA PARA CREAR APLICACIONES MULTIPLATAFORMA EMPLEANDO SOFTWARE LIBRE"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A N:

GABRIELA CASTILLO ARIZA

LUCÍA CERVANTES MARTÍNEZ

ASESOR DE TESIS:
M.EN C. MARCELO PÉREZ MEDEL

MÉXICO, 2003.

1

Si los sueños no se hicieran realidad,
la naturaleza no nos haría tenerlos.

Agradecimientos

Le agradezco a la vida por estar aquí.

A mi madre por su apoyo incondicional.

A mi padre donde quiera que se encuentre.

A mis hermanos, Paty, Jesús, Gus, y en especial a Adrián por todo su apoyo y motivación.

A mi tío Genaro por ser como un padre para nosotros.

A Ale por su comprensión y amor.

A Gaby mi amiga de Tésis, por brindarme su amistad, su apoyo y a comprensión.

Y a toda mi familia en general.

Si no hubiera sido por todos ellos, no hubiera sido posible la realización de éste trabajo.

Agradecimientos

A Dios es la razón de mí existir.

A la Virgen de Guadalupe que es mi apoyo y guía.

A mi mamá Lucia que con su amor y confianza me ha apoyado en toda mi vida.

A mi papá Delfino por su comprensión y amor que siempre me ha brindado.

A mis hermanos Lourdes, Aurora, Alfredo, por su apoyo y amor que me demuestran todos los días.

A Ivette y familia por brindarme su amistad y apoyo.

A mi amiga de Tésis, Lucy, *por brindarme su amistad, su apoyo y comprensión.*

A toda mi familia en general por estar conmigo en todo momento.

Agradecimientos

Le agradezco a la vida por estar aquí.

A mi madre por su apoyo incondicional.

A mi padre donde quiera que se encuentre.

A mis hermanos, Paty, Jesús, Gus, y en especial a Adrián por todo su apoyo.

A mi tío Genaro por ser como un padre para nosotros.

A Ale por su comprensión y amor.

Y a toda mi familia en general.

Si no hubiera sido por todos ellos, no hubiera sido posible la realización personal de éste trabajo.

Agradecimientos

A Marcelo por toda su comprensión, paciencia, apoyo y amistad.

A Gerard, por toda su ayuda y amistad.

A mis amigos de la Universidad que siempre están presentes en todo momento.

A mis amigos del UNITEC por brindarme su amistad y apoyo.

A todos los que laboran en el Tecnológico por su ayuda y apoyo.

Al Centro Tecnológico por los servicios prestados para la realización de este trabajo.

Y a mi institución a quien le agradezco la formación académica que me ha dado.

UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO.

ÍNDICE.

Capítulo 2

Aspectos de Portabilidad

Capítulo 3

Herramientas y bibliotecas libres.

Capítulo 4

Metodología para desarrollar aplicaciones multiplataforma.

Capítulo 5

Caso práctico.

Capítulo 6

Conclusiones.

Índice de Figuras y Tablas

Figuras

Tablas

INTRODUCCIÓN

Antes que nada se debe de tener muy claro lo que es una plataforma, muchas veces se confunde con el término arquitectura, se entiende por el segundo es todo lo que conforma la parte física de la computadora, en una palabra el Hardware, mientras que la plataforma en una combinación de ambos, software (sistema operativo) y Hardware interactuando ambos en un mismo tiempo y para un mismo fin.

A través del tiempo se creía que la portabilidad no fuera posible, esto es que una aplicación debiera funcionar en cualquier plataforma sin la necesidad de cambios trascendentales o lo que es peor aún rescribir completamente el código de la aplicación.

A medida que la tecnología fue avanzando aparecieron lenguajes de alto nivel, y esto hizo posible que se puedan migrar las aplicaciones a diferentes arquitecturas tanto de hardware como de software, pero con un alto grado de dificultad.

Este trabajo se hizo pensando en todos aquellos usuarios que desean portar sus aplicaciones a diferentes plataformas, es decir, que un programa con tan sólo unas cuantas modificaciones pueda ser ejecutado en un ambiente diferente para el cual fue diseñado. Por ello en este trabajo lo que se intenta es realizar un manual de referencia para desarrollar aplicaciones que se diseñen en Linux y que se ejecuten en Windows.

Todo el software utilizado en este trabajo, es de origen libre, lo que es llamado Open Source, o sea que el código fuente esta disponible, se distribuye gratuitamente y no sólo eso se tiene la libertad de modificarlo.

Esto por consiguiente trae consigo demasiadas ventajas, puede adecuarse y mejorarse según las necesidades de cada usuario, además de tener un sin número de utilidades y herramientas, como pueden ser sistemas operativos, lenguajes de programación, navegadores de Internet, ambientes gráficos, servidores web, manejadores de bases de datos, paqueterías, programas de diseño, etc.

Para lograr el objetivo se analizaran los diferentes lenguajes de programación y su disponibilidad para portarlos. Que tanto se pude realizar esta operación en ellos. El código fuente de las aplicaciones que se deseen portar desempeñan un papel muy importante para la misma. Es necesario que el código este en C ANSI para su portabilidad.

Para ello se utilizaran diferentes herramientas y bibliotecas como: Cygwin, GCC, GTK +, etc, en los capítulos posteriores se proporcionará una explicación detallada de cada una de éstas.

Actualmente la mayoría de los usuarios usan Windows en alguna de sus versiones, por razones comerciales y no por esto suele ser el mejor, por lo tanto un programa que sólo se ejecute en ambiente Linux no será tan utilizado como uno que es diseñado especialmente para Windows.

CAPÍTULO 1

ANTECEDENTES DE SOFTWARE LIBRE

I. Antecedentes de software libre.

Para la realización de este proyecto fue importante utilizar software libre, por muchas razones, una de ellas es que si se utiliza software comercial se corre el riesgo de que si llegaran a desaparecer estas empresas, se pierden los códigos fuentes de sus aplicaciones junto con éstas, cosa que afectaría demasiado, ya que se perdería todo el trabajo realizado con estas aplicaciones.

1.1 Fundación para el Software Libre (FSF).

La Fundación para el Software Libre es una institución que se dedica a vigilar que exista software que pueda ser copiado, redistribuido, y modificado libremente. Esto es posible gracias a usuarios interesados en estos proyectos, que se dedican a desarrollarlos.

Para administrar adecuadamente los recursos disponibles del proyecto GNU[1], se creó la Free Software Foundation, (Fundación para el Software Libre) representada con las siglas FSF, que es la organización encargada, entre otras cosas, de encausar dicho proyecto.

Puesto que el objetivo principal de la FSF es promocionar, desarrollar y difundir el software libre, una de las principales tareas que se llevaron acabo fue la de asegurarse de que el software GNU fuese siempre de este tipo.

No todas las empresas de software se dedican a desarrollar programas, algunas sólo lo distribuyen, modifican o le dan soporte, etc. La Fundación se dedica única y exclusivamente a desarrollar software libre. Esto con la finalidad de reducir costos, papeleo, y que el software comercial deje de monopolizar el mercado, se busca que haya opciones para los diferentes usuarios, y que con esto se les de la oportunidad de elegir las herramientas que convengan a su beneficio, y que no estén sólo a expensas de algunas empresas que se dedican al comercio con software comercial.

1.2. GNU.

1.2. El sistema GNU.

En los primeros años de la informática moderna, (70´s), los programadores intercambiaban y compartían sus programas e ideas. De esta manera, se fueron creando sociedades de programadores, que al intercambiar sus conocimientos mejoraban sus aplicaciones. En una de estas comunidades, trabajó Richard Stallman, entre otros.

[1] http://www.gnu.org/ La página pertenece a la FSF 18 /mayo/ 2002

Pero en un momento dado las empresas de software comenzaron a privatizar los derechos de los usuarios. Stallman sabía que los usuarios tienen toda la libertad de utilizar su computadora, lo que llamamos hardware, pero no así el software principal, es decir el sistema operativo. Si éste no era libre, ninguna aplicación instalada en cualquier computadora, podría serlo.

En 1984 comenzó el proyecto, en ese momento, era imposible usar cualquier herramienta moderna de software, sin instalar un sistema operativo comercial muy restringido, este poseía una licencia que lo limitaba y además carecía de acceso al código fuente, por lo tanto no se podía hacer ningún tipo de modificación para adaptarlo a las necesidades de los usuarios, todo esto con la intención de no compartir el software libremente con los demás, para así poder lucrar con la venta de licencias de uso.

Richard Stallman comenzó a trabajar en un sistema operativo que fuera libre y gratuito, este proyecto fue llamado GNU. Este sistema fue fundado precisamente para evitar todo esto, su primera meta fue desarrollar un sistema operativo portable compatible con Unix que fuera diseñado totalmente con software libre, de manera que los usuarios fueran capaces de distribuir, modificar y además contribuir con cualquier parte del sistema completo. "El nombre del sistema, GNU, es un acrónimo recursivo que significa GNU no es Unix"[2] como una manera de rendir homenaje a Unix y a la vez indicar que el concepto es algo diferente. Técnicamente, GNU es como Unix, pero éste da a sus usuarios libertad, lo que Unix no hace.

Uno de los primeros programas realizados para este proyecto fué EMACS. El cual es un editor configurable y programable implementado como un intérprete de Lisp. Hoy día, la versión GNU de EMACS es un estándar en la mayoría de las instalaciones de ordenadores.

Otro de los primeros proyectos fue el compilador de C (GCC), desarrollado entre otros por Richard Stallman. Se trata de un compilador multiplataforma y multilenguaje. Soporta múltiples descripciones de arquitecturas y sirve como interfaz único para diferentes lenguajes de programación (actualmente soporta C, C++, Logo, Fortran, Ada, Java, etc).

En 1991 fue desarrollado el Kernel[3] de Linux por Linus Torvalds. GNU hace ahora del sistema GNU/Linux un sistema casi tan práctico de usar como cualquier otro sistema operativo.

[2] http://www.gnu.org/home.es.html. La página pertenece a la FSF 18 /mayo/ 2002

[3] Kernel: : Es el software que constituye el núcleo del sistema operativo, dónde se realizan las funcionalidades básicas como la gestión de procesos, la gestión de memoria y de entrada salida.

Actualmente se desarrollan toda clase de aplicaciones para este sistema operativo, continua el trabajo para cualquier desarrollador que desee tomar el reto de ser parte del sistema GNU.

1.2.2. Software GNU.

Muy poco del software libre se desarrolla internamente, la mayoría de éste, se realiza por fuera con voluntarios que se dedican a desarrollar todo tipo de aplicaciones. Por está razón no todo el software GNU está protegido por las licencias Copyleft o por Copyright (las definiciones de estas licencias se verán con detalle posteriormente en este capítulo) ya que los propios voluntarios deciden cual es más conveniente de acuerdo a sus intereses.

Un software se considera que pertenece al proyecto GNU, siempre y cuando los programadores estén totalmente concientes de las leyes y normas que rigen este tipo de software, y por lo tanto también deben de considerar las responsabilidades que adquieren. Incluyendo estándares de compilación y configuración de los programas. No se es tan estricto en esto de seguir los estándares, pero es importante que se sigan lo mejor posible, éste debe ser libre cumpliendo con todas las características mencionadas anteriormente, mucho de este software está protegido por la licencia Copyleft, pero no todo el software GNU esta bajo esta licencia.

Para continuar con el desarrollo del software GNU los programadores deben de estar concientes, de que jamás deben de utilizar software no libre dentro de sus aplicaciones, ya que simplemente el software perdería sus principales características, por lo cual es llamado de esa forma, debe ser libre para todo aquel que desea utilizarlo.

También la terminología es importante, se deben de seguir los estándares ya que por ejemplo, no se le puede llamar al software libre software gratuito, ya que simplemente no significa lo mismo, esto es porque al software libre en el idioma ingles se le llama freesoftware y free significa gratis o libre. Por ejemplo al sistema operativo Linux, no se le debe llamar simplemente Linux, sino sistema GNU/Linux, o sistema GNU basado en Linux.

1.3. Clasificación de licencias de software.

Hay diferentes tipos de licencias, y a la hora de escoger con cual queremos proteger alguna aplicación que se desarrolle, dependerá de los intereses del programador de cómo quiera que se distribuya su aplicación.

1.3.1. Software libre.

Esta definición no es nueva pero se tratará de ampliar un poco, el software libre es aquel en el cual los usuarios tienen la libertad de usar, copiar, distribuir, estudiar, cambiar y mejorar (si se puede) este tipo de software. Hay ciertas cláusulas que se deben cumplir, para decidir si realmente este software es libre o no. Si éstas de alguna manera no se cumplen entonces el software no se puede considerar como tal.

Las siguientes son las características principales por las cuales este tipo de software se distingue:

- Facultad de usarse con cualquier interés.
- Facultad de investigar el funcionamiento del software y modificarlo si lo considera necesario.
- Facultad de acceder al código fuente, esto es fundamental, ya que es una de las características primordiales de este tipo de software.
- Facultad para distribuir copias a cualquier usuario, que tenga la necesidad o desee utilizar el software
- Facultad de modificar el software, y hacer públicos estos cambios, si son para mejorar el mismo.

Un programa es software libre sólo si los usuarios tienen todas estas libertades. Por esto, todo usuario es libre de redistribuir copias, ya sea con o sin modificaciones, gratis o cobrando una cuota por la distribución a cualquiera y en cualquier lugar. El usuario no tiene que pedir o pagar permisos.

Otro punto importante es que también este tipo de software se puede utilizar en empresas privadas, y se tiene la libertad de hacer modificaciones o utilizarlas ya sea para trabajo, juego, etc., sin mencionar que dichas modificaciones existen. Si se publican los cambios, no se tiene la necesidad de publicar los mismos de una forma en especial esto se hace libremente.

No importa como se adquiera el software libre, si es sin cargo o con un costo, se pueden hacer todo tipo de negocios con él, si se desea hacerlo, se puede vender, cambiar, mejorar, siempre y cuando, se lleven acabo dos cosas, si se modifica compartir los cambios de alguna forma, y la otra es que se publique el código fuente al momento de distribuirlo.
Un ejemplo claro de esto es el propio sistema operativo Linux, el cual algunas empresas o instituciones lo venden, otras sólo le proporcionan soporte, algunas lo desarrollan etc, dependiendo del giro de cada una de estas organizaciones.

1.3.2. Freeware.

Este término no tiene una definición clara, ya que muchas veces se confunde con lo que llamamos software libre, por que la propia traducción al español así lo especifica, pero no es así, hay demasiada diferencia entre ambos tipos de software. Sólo se parecen en que los

dos se distribuyen de manera gratuita, pero la diferencia de éste, es que no se pude modificar ya que no se tiene acceso o no está disponible el código fuente. Comúnmente pueden ser paquetes para alguna aplicación específica, juegos, etc.

Estas aplicaciones se bajan libremente por Internet, esto se hace sin ningún tipo de compensación lucrativa para sus autores o distribuidores. El software de este tipo tiene derechos de autor, por lo tanto no se puede comercializar de ninguna forma, o sea que cualquier persona pude distribuirlo de forma gratuita solamente pero no modificarlo, o cambiarle los derechos de autor. Sólo se pude cobrar el medio por el cual sea distribuido en un caso específico, los discos compactos, o los disquetes, etc.

1.3.3. Shareware.

El término significa literalmente programa compartido e indica que cualquiera puede descargar el programa y empezar a emplearlo sin desembolso previo. Pero ello no significa que sean de libre uso o de empleo gratuito. La licencia de uso indica con claridad en cada caso los términos de empleo, así como la cantidad que debe ser abonada en caso de encontrar de utilidad el programa. El sistema shareware se utiliza a menudo como medio para distribuir versiones de prueba con un costo mínimo.

Las versiones de prueba, en general tienen algún tipo de limitación. En unos casos, algunas funciones no están disponibles; y en otros, el programa sólo admite una cierta cantidad reducida de datos. En su versión más conocida, el programa tiene toda su funcionalidad, pero sólo es operable durante 30 días tras su instalación. Al cabo de este tiempo, algunas aplicaciones de este tipo dejan de funcionar y recuerdan que deben ser desinstalados del ordenador o ser pagados. Algunos programas simplemente recuerdan esto, cada vez que se ejecutan cuando el periodo de prueba ha terminado, pero siguen funcionando.

Como medio para recordar el tipo de software que se activa, la mayoría de programas shareware tienen una pantalla inicial que recuerda los días que quedan de licencia. Esta pantalla inicial y la necesidad de confirmar manualmente la aceptación desaparece en la versión comercial. Por lo que siempre es mejor adquirir esta versión.

Por lo tanto, es recomendable este tipo de software que se adquiere por medio de Internet para usarlo solamente un tiempo de prueba, para observar si en realidad es útil o no, ya que se adquiere gratis por un lapso que la misma licencia te indica, pero si te convence durante ese tiempo, debes de pagarlo totalmente. El término shareware significa compartido esto quiere decir que cualquier usuario puede adquirirlo de Internet y comenzar a usarlo inmediatamente sin ningún costo en el principio de su uso.

Este software no es libre, tampoco es semilibre, estas son las razones por lo cual no lo es:

- Su código fuente no está disponible, por lo tanto no se pueden hacer modificaciones.
- El shareware no se puede seguir utilizando después de un tiempo, sin que esto tenga un costo, la licencia no lo permite, pero como la mayoría de las veces la cuota es para pagarlo de voluntad propia, la gente no lo hace y lo sigue utilizando sin cargos, pero aclaremos que la licencia no lo permite y que además por lo regular la aplicación que se utilice no esta completa.

1.3.4. Software de Dominio Público.

Los programas de dominio público, (public domain) en inglés, son aquellos en los que el autor coloca a disposición del público, no sólo el programa en sí, sino incluso el código fuente, de manera que pueda ser utilizado, y también puede ser modificado, como parte de otros programas.

Este término es un poco confuso ya que también puede pensarse que se refiere a lo que conocemos como software libre, y no es así, esto significa es que el software no está protegido por ninguna de las dos licencias copyright y tampoco esta protegido con copyleft, es decir, que este software no es libre del todo, ya que cualquier usuario que lo distribuya lo puede convertir a software comercial si le conviene a sus intereses.

No puedes definirlo como software libre, disponible o gratis, ya que lo único que tiene de diferencia con el software comercial es que este no hace uso de ninguna licencia.

1.3.5. Software Libre Comercial.

Como se sabe el software comercial se diseña con fines de lucro, pero existe una excepción y es cuando ese software se le da el término de libre, esto es por ejemplo: cuando un

software puede ser modificado y distribuido libremente, pero hay empresas especializadas para darle soporte, o escribir manuales para su uso, este tipo de software suele ser muy complicado y sin éstos es casi imposible utilizarlo, los desarrolladores hacen esto, con toda la intención, para que sea indispensable la adquisición de manuales de referencia.

1.3.6. Software semilibre.

Este tipo de software es aquel que no es libre completamente pero se puede copiar, redistribuir, usar y modificar, sin ningún tipo de lucro, incluyendo las versiones de software modificadas.

La diferencia del porque este software es llamado semilibre teniendo todas las características aparentes de software libre, es que no está protegido por la licencia copyleft la cual está precisamente para proteger que cualquier usuario pueda tener la libertad de modificar, usar, distribuir el software y la principal es tener acceso al código fuente. Si alguna de estas libertades no se cumple el usuario pierde sus derechos y el software deja de ser completamente libre. Por tal motivo este tipo de programas se podría cambiar por software comercial.

A pesar de esto este tipo de software es mejor que el software comercial, ya que no tiene tantas restricciones, pero no se puede usar en sistemas operativos libres, esto es por razones de licencias. Ya que no se permiten las restricciones de este tipo de software. Si se usará un software semilibre en un sistema operativo libre, entonces todo el sistema operativo sería del mismo tipo.

Si esto sucediera no todos los usuarios podrían usarlo, ya que la principal razón de que exista software completamente libre, es que todos los usuarios puedan acceder a él, no sólo las instituciones, o personas interesadas, sino empresas y todo aquel que deseea utilizarlo para cualquier fin que se le ocurra.

1.3.7. Software Comercial.

El software comercial es aquel que más restricciones posee, ya que su fin es totalmente de lucro, y su distribución, uso, modificación está prohibida totalmente, a menos que se pague el derecho a uso. Regularmente su licencia, prohíbe casi todo, y en la mayoría de los casos no se tiene acceso al código fuente.

Según su propia licencia, al usar software sin tomar en cuenta la misma, se está cometiendo en un grave ilícito, y esto implica severas sanciones para la institución o persona que haga uso de este software sin la debida autorización.

Uno de los métodos que utilizan este tipo de empresas las cuales se dedican a monopolizar software, es darle información falsa a los usuarios, para evitar que se utilice el software

libre. Les dicen que el software es de dudosa procedencia, y además está mal diseñado y por lo cual no tiene costo, se dice que no posee un soporte técnico real, puede contener virus, etc. Otro motivo, según estas empresas para no usar otro tipo de software que no sea comercial, se menciona que es incompatible con casi cualquier aplicación, y además también se dice que no se actualiza, y que el con el tiempo es más costoso comprar la versión nueva de dicho software.

Como se sabe, todo esto no es verdad, por el contrario, el software libre se mejora constantemente ya que hay una gran cantidad de usuarios interesados en este tipo de proyectos, y día a día se unen más. Se busca la compatibilidad y el mejoramiento del mismo, además de que hay empresas que se dedican única y exclusivamente a darle soporte a este tipo de aplicaciones, claro no todo este software cuenta con el soporte técnico del software comercial, pero se está trabajando en eso, ya que el software libre evoluciona con tanta rapidez, que es casi imposible publicar al mismo tiempo que se diseña, todos los manuales de uso que sirvan para utilizarlo a su máximo potencial. Algunos sólo cuentan con lista de correo, y otros con listas de preguntas frecuentes. En su diseño, la gran mayoría de estas aplicaciones mejoran al software legal, ya que tienen más aplicaciones, es más rápido, ocupa menos espacio en disco duro, etc., son muchas las características por las cuales el software libre supera al software legal.

1.3.8. Características del software libre.

Tipo de software	Tipo de Licencia	Distribución Gratuita	Modificables	Empleo gratuito	Con Código fuente
Freeware	Ninguna	X	X	X	
Shareware	Licencia del autor.	X			
Dominio Público	Ninguna	X	X	X	X
Comercial	Copyright				
Libre Comercial	Ninguna	X	X	X	
Semilibre	Ninguna	X	X	X	X
Software Libre GNU	GPL/GNU	X	X	X	X

1.4. Licencias de software libre.

1.4.1. Concepto de copyleft.

A las personas que trabajan en GNU, les importa mucho que el software sea libre, para todo aquel que desee utilizarlo de acuerdo a sus necesidades y que esto continúe así, además les

gustaría tener más usuarios, pero para ello, tendrían que hacer el software del dominio público, cosa que le quitaría ciertas libertades como ya se mencionó. Para asegurarse de que esto no suceda, lo protegen con una licencia llamada Copyleft. Esta licencia dice que todo aquel que use este software tiene derecho a copiarlo, usarlo, modificarlo y distribuirlo, pero siempre tiene que ir acompañado con su respectivo código fuente y si se le hacen cambios o mejoras, publicarlos.

El proceso que lleva a cabo la licencia copyleft para cubrir un programa es el siguiente:

Para que se proteja un programa con copyleft, primero se reservan los derechos; luego se añaden términos de distribución, los cuales son un instrumento legal en el que le dan a todas las personas los derechos a utilizar, modificar, y redistribuir el código del programa o cualquier programa derivado del mismo, pero sólo si los términos de distribución no son cambiados. Así, el código y las libertades se hacen legalmente inseparables.

Los que se dedican a diseñar software comercial, utilizan copyright para eliminar los derechos a los usuarios, y esto los limita demasiado, todo lo contrario de la licencia copyleft, donde se utilizan estos mismos derechos reservados para diferentes fines, los cuales son los siguientes:

Que los usuarios continúen teniendo todas las libertades antes mencionas. Es por eso que el nombre de la licencia es lo opuesto y en lugar de llamarse copyright se llama copyleft (un juego de palabras).

1.4.2. GPL (Licencia Publica general).

El objetivo principal de la FSF es que el software permanezca libre, para conseguir esto la Fundación desarrolló la Licencia Pública General GNU con las siglas GPL. Ésta es lo contrario a las licencias de los programas comerciales.

La GPL asegura al usuario que un programa que se encuentre protegido con esta licencia, posee la libertad para utilizar y/o personalizar el programa a su agrado. La única obligación que impone la GPL, es cuando un usuario distribuye el programa a otra persona, ésta tiene que otorgar los mismos derechos que tenía el usuario anterior. Es decir, lo que se busca es conservar los derechos de todos los usuarios a que el software sea libre en cualquier momento

Otro aspecto importante que toma en cuenta la GPL es la renuncia de garantía. Para proteger los intereses de los autores de software que ponen sus programas bajo la GPL, se establece que no existe ninguna garantía sobre el programa, y que el usuario es el único responsable de todas las consecuencias que el uso del programa pudiera acarrear. Así también se asegura que si alguien modifica el programa, la reputación del autor original permanece intacta.

No obstante, personas o compañías podrían optar por ofrecer garantía sobre un programa GNU a los usuarios a cambio de un precio. Ésta es una de las maneras de las que las empresas comerciales pueden seguir existiendo y ganando dinero a partir del software libre. Puesto que libre no es lo mismo que gratuito, las empresas tienen derecho a vender el software, y a ofrecer garantías o soporte técnico a cambio de un precio. Naturalmente, no pueden eliminar al usuario sus derechos a intercambiar y modificar el programa.

Cuando en este tema se habla de software libre, no se refiriere a la palabra gratis, se habla de libertad, no de precio. Ya que la palabra utilizada en ingles es freesoftware, esta palabra puede significar dos cosas, libre y gratis, en realidad la mayoría lo toma como gratis, es por eso que la definición es confusa.

Un ejemplo de esto sería, si se comercializan alguno o algunos de estos programas, ya sea con costo o sin él, se debe cumplir con lo que estipula la licencia, es decir, se le debe informar al nuevo usuario los derechos y obligaciones que adquiere junto con el nuevo software. También se debe asegurar que los nuevo usuarios reciban con todo el código fuente de su programa nuevo.

Otra punto importante que es abarcado por la licencia GNU, son las patentes, como el software libre esta teniendo mucho auge en los últimos años, las patentes están a la orden del día, cabe mencionar que si algún software llega a patentarse, perdería todas sus libertades antes mencionadas.

1.4.3. LGPL (Licencia Pública General para Bibliotecas).

Para comenzar se dará una breve explicación del concepto biblioteca o también llamada librería ya que se le conoce como library en el idioma inglés, es un conjunto de funciones o datos que nos sirven para que una aplicación funcione. Hay dos tipos de éstas, las estáticas y las dinámicas, las estáticas son las estandarizadas por el lenguaje de programación, y son llamadas mientras se compila la aplicación, toma las funciones o datos que necesita y los inserta en el programa final, por el contrario las bibliotecas dinámicas o también llamadas dlls son llamadas cuando el programa ya está ejecutándose, al momento de que la aplicación las necesita las llama y las inserta en el programa, una vez que deje de utilizarlas las desecha y queda el espacio libre en memoria, esto es mejor ya que evita la saturación de la memoria de la máquina en el momento de compilación que se llevaría acabo si las bibliotecas fueran estáticas. Ahora se están utilizando con más frecuencia.

Existen otra licencia llamada LGPL es muy parecida a la GPL, la única diferencia entre ambas se encuentra es que la primera se usa para aplicaciones completas, ya sean programas de aplicación o herramientas de desarrollo, y la segunda, es sólo para bibliotecas. Para permitir que las bibliotecas libres pudieran usarse en programas no libres (y así difundir el uso de esas bibliotecas libres) se creó esta licencia, pero otra cosa que la caracteriza es que permite su uso con otros programas sin que estos caigan necesariamente

bajo ninguna licencia GNU. Hasta la fecha, la FSF recomienda no usar la LGPL y poner todos los programas y bibliotecas bajo la GPL, para promocionar definitivamente el software libre.

Es importante que un usuario que utilice la licencia LGPL para bibliotecas, la cambie a la licencia GPL. Por la siguiente razón, quien trabaje desarrollando software comercial, tiene un remuneración monetaria, pero en cambio, quien trabaje desarrollando software libre no recibe nada, es por esta razón que es importante que los desarrolladores de software comercial no puedan utilizar las bibliotecas de software libre. Así sólo los desarrolladores de software libre puedan utilizar sus bibliotecas aunque no tengan remuneración económica (en algunos casos si es posible que obtengan ingresos por desarrollar software libre).

Pero si hay una cantidad considerable de bibliotecas amparadas por GPL, tantas o más que en el software comercial. Entonces se tendrán un conjunto de herramientas útiles que servirán para diseñar programas libres. Esto es lo que promueve el desarrollo de software libre.

La LGPL no da el derecho como usuarios de distribuir el software, con o sin cargo, copiar software de manera gratuita, y entregarle el código fuente en cada copia que se distribuya. La única restricción que propone esta licencia es prohibirle precisamente la negativa de estos derechos.

Para los desarrolladores de software libre las patentes son un gran riesgo constante, ya que estas ponen en peligro los programas, ya que las restricciones tanto para usuarios como para software son muy limitantes. Lo que a ellos les importa es que el software permanezca siendo libre.

1.4.4. Otras licencias.

GNU no es el único proyecto de software libre. A lo largo del tiempo han ido surgiendo otras iniciativas que también tratan de difundir el modelo de software libre. Cada una de ellas ha desarrollado una licencia ligeramente diferente bajo la cual se pueden acomodar sus programas.

Algunas de ellas son la licencia del X11/XFree86, la licencia FreeBSD, y las dos licencias BSD. Todas ellas son muy similares con algunas diferencias en cuanto en su forma de describir lo que desean.

La que se verá con un poco de detalle es la BSD que llevan los programas desarrollados en la Universidad de California en Berkeley. Entre estos programas se encuentran los sistemas operativos BSD, como FreeBSD. Esta licencia establece que cualquiera puede realizar

cualquier acción con el programa, incluido copiarlo en formato fuente o binario y modificarlo. Además, en el caso de que alguien lo modifique, el programa nuevo, no tiene por que ser distribuida bajo una licencia libre, sino que puede ser incluida en programas comerciales. Esta última posibilidad es la que hace que mucha gente no utilice esta licencia, puesto que permite a empresas comerciales de software aprovecharse del trabajo de otros y utilizarlo exclusivamente en su propio beneficio. Además de los sistemas operativos BSD, otros proyectos como el XWindow System se encuentran protegidos por licencias similares a la BSD.

La licencia permite a los usuarios la distribución y modificación del software dentro de ciertos límites, mientras que reserva para el autor una especie de control sobre su obra. Es el autor original el que decide qué dirección debe tomar el desarrollo del programa. El programa más importante desarrollado bajo ésta licencia es el lenguaje Perl, creado por Larry Wall.

Para evitar confusiones con el concepto freesoftware que lo mismo significa, gratuito que libre en el idioma inglés, se ha introducido un nuevo concepto para referirse al término software librea el cual es OpenSource. El software OpenSource es aquel cuyo código fuente está disponibles para todos los usuarios.

Algunas casas de software han desarrollado nuevas licencias OpenSource para sus programas. Entre éstas se cuentan:

- La QPL, desarrollada por Troll Tech para su librería Qt.
- La NPL, desarrollada por Netscape Communications para su proyecto Mozilla.
- La SCSL (Sun Community Software License), bajo la que Sun Microsystems ofrece algunos de sus productos, como StarOffice o Solaris 8.

CAPÍTULO II

ASPECTOS DE PORTABILIDAD

II. Aspectos de Portabilidad.

Cuando desarrollamos un programa para un entorno, al moverlo a un compilador, procesador o sistema operativo diferente, lo ideal sería que no requiriera de ningún cambio, es decir que resulte más sencillo modificar el programa en lugar de reescribirlo en su totalidad, a esto se le llama portabilidad.

2.1. Características de la portabilidad.

Aunque típicamente el software va a ejecutarse en un solo entorno, bajo condiciones especificas, necesitamos tomar en cuenta la portabilidad, ya que cualquier programa exitoso, por lo general, termina siendo empleado en varias plataformas.

Construir software en forma general sin características particulares, tiene como resultado menos mantenimiento y más utilidad durante su ciclo de vida, tomando en cuenta que los entornos cambian. Cuando el compilador, el sistema operativo o el hardware se actualizan, las características también pueden cambiar. Mientras exista menor dependencia de características especiales, por parte del programa existe menos probabilidad de que éste presente fallas y es más fácil que se adapte a las nuevas circunstancias.

Por supuesto que el grado de portabilidad deberá ser acorde con la realidad. No existe un programa que sea absolutamente portable, sólo aquéllos que son extremadamente sencillos y sólo de utilidad didáctica. Pero podemos mantener la portabilidad como una de nuestras metas principales para lograr que un programa se ejecute con cambios mínimos en cualquier plataforma.

Al realizar una aplicación tomando en cuenta aspectos de portabilidad se invierte tiempo adicional, este tiempo se recupera cuando al migrar estas aplicaciones a otra plataforma se realicen en un tiempo menor.

Por último, y lo más importante para que un programa sea más portable debe estar mejor diseñado, mejor construido y probado con mayor detalle, ya que las técnicas de programación portable están muy relacionadas con las de la buena programación en general.

2.2. Lenguajes de programación.

Para escribir código portable se necesita programar en un lenguaje de alto nivel, y dentro de algún estándar. Los archivos binarios no se transportan, pero el código fuente sí. Aun así, la forma en que un compilador traduce un programa a instrucciones de máquina no está definida con precisión, incluso para lenguajes estándar. Pocos de los lenguajes de amplio uso tienen una sola implementación; por lo regular existen múltiples proveedores o

versiones para diferentes sistemas operativos, o versiones que han evolucionado con el tiempo, y varían en como interpretan el código fuente.

A veces el estándar es incompleto y no alcanza a definir el comportamiento cuando las características interactúan o algún aspecto está indefinido en forma deliberada; por ejemplo el tipo char en C y C++ puede ser con o sin signo, y ni siquiera debe tener exactamente 8 bits[1]. Dejar esos asuntos a quien escribe el compilador puede permitir implementaciones más eficientes y evita restringir el hardware en el que se ejecutará el lenguaje, pero dificulta la portabilidad.

Las diversas políticas y los asuntos técnicos de compatibilidad pueden llevar a compromisos que dejen detalles sin especificar. Finalmente, los lenguajes no son del todo compatibles y los compiladores son complejos, por lo que habrá algunas diferencias en la interpretación e implementación.

En ocasiones los lenguajes no están estandarizados. C tiene un estándar oficial ANSI/ISO (American Nacional Standards Institute- Instituto Nacional de Estándares Americanos/ Internacional Organization for Standardization- Organización de Normas Internacionales)[2] emitido en 1988, Java es nuevo y todavía muy lejano de la estandarización. El estándar de un lenguaje normalmente se desarrolla sólo después de que existen varias implementaciones conflictivas a unificar, y cuando es de uso amplio, lo suficiente como para justificar los gastos de estandarización. Por lo tanto, aún hay programas por escribir y múltiples entornos que usar.

La especialización de los lenguajes de alto nivel en tareas o grupos de aplicaciones concretas ha llevado a la existencia de multitud de ellos, cada uno con sus peculiaridades y su forma particular de desarrollar programas.

2.2.1. Diferentes lenguajes de programación.

2.2.1.1. Basic.

Basic (Beginners All-purpose Symbolic Instruction Code – Conjunto de Instrucciones Simbólicas de propósito general para principiantes) es quizás el lenguaje de alto nivel más conocido, se debe gran parte de su difusión a las características de su diseño como el empleo de palabras inglesas corrientes para denominar a las instrucciones y el uso de símbolos matemáticos para las operaciones.

Fundamentalmente se utiliza para resolver problemas científico-técnicos y generales que no presenten demasiada complejidad. Una de sus características principales, además de su sencillez de empleo, es que está pensado para su manejo en forma de diálogo con el

[1] Un entero en C es del tamaño de la palabra del procesador en la maquina, donde se ejecuta.
[2] ANSI/ISO http://www.eskimo.com/~scs/C-faq/s11.html

ordenador, como corresponde al uso de intérprete en lugar de compilador. Cuando se realiza un programa en Basic, el programador o usuario va introduciendo sus instrucciones en el ordenador y el intérprete le va diciendo, una a una, si es correcta o no, para que en este último caso la modifique convenientemente.

La sencillez en el manejo de Basic y el pequeño tamaño de memoria central de los ordenadores que requieren han sido los elementos clave para su incorporación a los equipos de dimensiones reducidas, lo que le ha dado una popularidad y una difusión tremendas. La gran mayoría de las personas que se introducían por primera vez en el mundo de la informática lo hacían a través de Basic.

Actualmente existen en el mercado numerosos compiladores de Basic y algunos de ellos tan completos y estructurados, que no tienen nada que envidiar a otros lenguajes como Pascal. Además se ha mejorado el manejo de archivos de archivos se ha mejorado. Existen también versiones para entornos gráficos que se basan en la programación orientada a eventos, con instrucciones altamente especializadas como es el caso de Visual Basic.

2.2.1.2. Cobol.

A mediados de los años 50's ante el desarrollo de lenguajes de programación de tipo científico, un grupo de representantes gubernamentales, usuarios comerciales y fabricantes de computadoras, se dierón cuenta que era necesario y factible desarrollar un lenguaje de programación para aplicaciones comerciales y administrativas.

En 1959 surgió una versión preliminar de Cobol (Common Business Oriented Language – Lenguaje Orientado a los negocios en general). En 1961, se desarrolló una segunda versión llamada Cobol-61, cuyo diseño sirvió de base para las versiones posteriores. En 1969, se aprobó como versión estándar por el ANSI (American National Standards Institute). La última versión de Cobol ANSI fue aprobada en el año 1974.

Primero surgió Cobol y marcó la pauta de cómo deberían ser los lenguajes comerciales. Posteriormente, surgieron RPG, DB2 y más recientemente dBaseIII+ y todos aquellos lenguajes calificados en común como xBase (como Clipper, Foxbase, Foxpro, dBaseIV, etc.).

La intención original con la creación de lenguajes de programación comerciales, es para ser utilizados por personal no experimentado en la programación de computadoras.

En las primeras versiones el compilador de Cobol era lento y esto trajo como consecuencia la generación de códigos objetos ineficientes. Por esta razón, la tendencia general era utilizar lenguajes simbólicos de bajo nivel que podían ser ensamblados más rápidamente. En la actualidad, se han desarrollado varias versiones de Cobol, las cuales traducen rápidamente y generan un programa objeto muy eficiente.

La complejidad de los lenguajes científicos fundamentalmente era el uso de estructuras de control recursivas, un amplio conjunto de palabras reservadas, dificultad para la administración de archivos de datos (especialmente en la generación de índices), variables locales y globales, etc. Los diseñadores de los lenguajes comerciales trataron de quitar la mayoría de estas características, con el fin de simplificar el lenguaje en cuanto a la seguridad, la consistencia y la regularidad.

Un aporte importante de dichos lenguajes es la facilidad en el uso de archivos de datos (que normalmente utilizan índices) mediante la definición de estructuras de datos y de control necesarias, como por ejemplo, el concepto de registro y las instrucciones de alto nivel para manipular los archivos y sus índices. Los lenguajes científicos carecían de dichas facilidades, pues su orientación no requería normalmente de ellas.

Cobol es un lenguaje de alto nivel que responde a las necesidades del procedimiento de datos administrativos.

Las ventajas que tiene son: la manipulación eficiente y sencilla de archivos de datos y la sencillez del lenguaje le permite a usuarios no especializados en programación, aprenderlo y utilizarlo rápidamente sin tener que lidiar con complejas estructuras de control y de datos, normalmente presentes en los lenguajes de programación científicos.

Las desventajas son: la falta de seguridad, debido al uso de variables globales y modificables por cualquier párrafo de un programa, ya que podría conducir a un comportamiento inesperado del mismo.

Uso de sentencias multifuncionales como el *Perform*, el cual con agregar una cláusula particular se comporta en tres diferentes estructuras de control iterativas.

Uso de saltos indeterminados utilizando la instrucción *Alter*, ya que una vez que se modifica el destino de un *Goto*, es responsabilidad del programador regresarlo al destino original.

Cobol está diseñado para su aplicación en ambientes administrativos y comerciales, ya que trata con facilidad grandes cantidades de datos. La realización de nóminas o la gestión de actividades bancarias son dos ejemplos de actividades, que se pueden realizar con programas escritos en Cobol.

2.2.1.3. Fortran.

Fortran que originalmente significa Sistema de Traducción de Fórmulas Matemáticas pero se ha abreviado a la Formula Translation, es el más antiguo de los lenguajes de alto nivel, fue diseñado por un grupo en IBM durante los años 50 (1950). El lenguaje se hizo tan

popular en los 60s cuando otros vendedores empezaron a producir sus propias versiones y esto llevó a una divergencia creciente de dialectos (en 1963 había 40 compiladores diferentes). Para que Fortran 66 se volviera en el primer lenguaje en ser regularizado oficialmente en 1972. La publicación de la norma para Fortran se llevó a cabo en la forma más general que cualquier otro lenguaje.

A mediados de los años setenta se proporcionaron lenguaje virtualmente cada computadora, mini o mainframe, con un sistema Fortran 66 normal, por lo tanto era posible escribir programas en Fortran en cualquier sistema de forma segura y eficaz.

En 1970 se dio a conocer una nueva norma, ANSI X3.9 y en 1978 fueron publicadas por el Instituto de las Normas Nacional americana. Esta norma era seguida en 1980 por la Organización de Normas Internacionales (ISO) como una Norma Internacional ES 1539. El lenguaje es normalmente conocido como FORTRAN 77 (desde que el proyecto final realmente se completó en 1977) y es ahora la versión del lenguaje en su uso.

Fortran 90 es un desarrollo mayor del lenguaje pero no obstante incluye todos los de Fortran 77 como un subconjunto estricto y para que cualquier Fortran utilizando normalmente como el programa del 77 continuará siendo un programa válido en Fortran 90.

Además de las viejas estructuras de Fortran 77, Fortran 90 permite expresar los programas de manera que satisfacen más al ambiente de la informática moderna y han quedado obsoletos muchos de los mecanismos que eran apropiados en Fortran 77.

En Fortran 90 algunos rasgos de Fortran 77 han sido reemplazados por otros más seguros y eficaces, ya que muchos de éstos fueron quitados de la revisión interina del lenguaje Fortran 95.

En los últimos años el lenguaje basado en Fortran 90 conocido como High Performance Fortran (HPF) se ha desarrollado. Este lenguaje contiene todo lo correspondiente con Fortran 90 y también incluye otras extensiones que son muy útiles. Fortran 95 incluirá muchos de los nuevos rasgos de HPF.

Fortran es un lenguaje muy potente para el tipo de tareas para el que está diseñado, pero su evolución a lo largo de los últimos años ha sido muy poca. Así, por ejemplo, no se le han agregado posibilidades para aplicaciones gráficas o el desarrollo de juegos. El resultado de todo ello es que Fortran ha seguido en su sitio, pero sin ganar nuevos campos de aplicación.

2.2.1.4. Pascal.

Pascal fue diseñado por Nicolas Wirth en la década de los sesenta, el cual facilita la formación de alumnos en temas de programación informática. Su principal característica es que es un lenguaje que permite la programación estructurada, por lo que los programas

escritos con él son muy compactos, presentan una estructura muy simple y tienen un diseño lógico coherente, es probablemente, después de Basic, el lenguaje de programación más difundido en el terreno de los microprocesadores.

Aunque el Pascal original fue diseñado estrictamente para la enseñanza, tuvo tantos seguidores, que eventualmente hizo su participación dentro de la programación comercial, y acaparó la atención cuando salió el Turbo Pascal de Borland para la PC IBM. Teniendo el editor integrado, un compilador muy rápido, y un precio bajo fueron la combinación perfecta, para que se convirtiera en el lenguaje preferido para escribir pequeños programas para MS-DOS.

Los compiladores de C se hicieron más rápidos y tuvieron buenos editores integrados y depuradores. A principios de 1990 cuando Windows acaparó casi todo, y Borland ignoró a Pascal en favor de C++ para escribir aplicaciones para Windows. Turbo Pascal fue relegado.

Finalmente, en 1996, Borland lanzó su "Visual Basic Killer" (algo como "El asesino del Visual Basic"). Delphi es un compilador rápido de Pascal acoplado con una sencilla interfaz de usuario. A pesar de muchas adversidades, ganó muchos usuarios.

En general, Pascal es más simple que C, aunque la sintaxis es similar, carece de muchas operaciones rápidas que tiene C. Por lo tanto es más difícil escribir código incrustable "inteligente", y hacer operaciones de bajo nivel como manipulación de bits.

Las ventajas son: que es fácil de aprender y las implementaciones en plataformas específicas (Delphi) tienen una buena presentación.

Algunas desventajas son: que los lenguajes sucesores orientados a objetos, de Pascal (Modula, Oberon) no fueron exitosos y los fabricantes de compiladores no están apegados a los estándares.

2.2.1.5. Lenguaje C

El Lenguaje C es uno de los lenguajes que más reconocimiento tiene a nivel profesional. Este ofrece un juego de instrucciones muy reducido, con una gran cantidad de funciones de librerías las cuales las podremos ir enriqueciendo con nuevas funciones escritas a la medida de nuestras necesidades. Es necesario destacar el elevado grado de portabilidad que posee. Es un lenguaje estructurado de alto nivel utilizado para la implementación de Sistemas Operativos y Lenguajes de alto nivel, como para la realización de Utilidades y Programas de Aplicación. Por otra parte posee características relativamente de bajo nivel (manejo de direcciones de memoria, acceso a funciones de entrada / salida, etc.).

Las ventajas son: que es un buen lenguaje para escribir programas pequeños muy rápidos y fáciles de interconectar con lenguaje ensamblador, también es muy estandarizado, así como las versiones en otras plataformas que son similares.

Algunas desventajas son: que es difícil de implementar técnicas de programación orientada a objetos y la sintaxis puede ser difícil de asimilar.

Portabilidad: es lo principal del lenguaje y las llamadas a la función ANSI, el flujo de control, administración de memoria, y simple manejo de archivos, son muy portables. El resto es dependiente de la plataforma. Hacer un programa que sea portable entre Windows y Mac, por ejemplo, requiere que la interfaz de usuario se haga con llamadas específicas a funciones del sistema para cada plataforma. Esto significa, por lo general, escribir la interfaz de usuario dos veces. Existen librerías, sin embargo, que hacen el proceso de manera más fácil.

2.2.1.6. Java.

El lenguaje de programación Java, fue diseñado por la compañía Sun Microsystems Inc. con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes de computadoras formadas por más de un tipo de computadora, ya sean PC, MAC's, estaciones de trabajo, etc.), y que fuera independiente de la plataforma en la que se va a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma.

Java es un lenguaje de programación expresamente diseñado para usarse en el entorno distribuido de Internet, pero más sencillo de usar que el lenguaje C++, en la programación orientada a objetos. Java puede usarse para crear aplicaciones completas que corran en un solo ordenador o se distribuyan entre servidores y clientes de una red.

También puede usarse para construir pequeños módulos de aplicación o applets[1] para utilizarlos como parte de una página web. Los applets hacen posible que el usuario de una página web interactúe con ella.

2.2.1.6.1. Las principales características de Java.

Lenguaje simple: Java posee una curva de aprendizaje muy rápida. Resulta relativamente sencillo escribir applets interesantes desde el principio. Todos aquellos familiarizados con C++ encontrarán que Java es más sencillo, ya que se han eliminado ciertas características, como los punteros. Debido a su semejanza con C y C++, y dado que la mayoría de la gente los conoce aunque sea de forma elemental, resulta muy fácil aprender Java. Los programadores experimentados en C++ pueden migrar muy rápidamente a Java y ser productivos en poco tiempo.

[1] Applet es un pequeño programa en Java que se ejecuta dentro de las páginas Web.

Orientado a objetos: Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.

Distribuido: Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets[4] y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

Interpretado y compilado a la vez: Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador.

Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time).

Robusto: Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo de los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria, pero consume recursos.

Seguro: Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia. A nadie le gustaría ejecutar en su ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

Indiferente a la arquitectura: Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows NT, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. El compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñado para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java.

La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores

[4] Sockets permiten la comunicación entre procesos que se están ejecutando en máquinas remotas, pero integradas en una red. Los sockets corresponden al cuarto nivel del modelo OSI. (capa de transporte).

aritméticos, de manera que los programas son iguales en todas las plataformas. Estas dos últimas características se conocen como la Máquina Virtual Java (JVM).

2.2.1.6.2. Java con respecto a su portabilidad.

La portabilidad permite que los usuarios se puedan mover fácilmente entre sistemas operativos y plataformas diferentes. También permite que los programadores puedan transferir fácilmente sus trabajos a distintas máquinas con diferentes sistemas.

La mejor forma que tienen los programadores para aportar beneficios a la portabilidad es rechazar herramientas que no sean estándar.

- No usar J++ o VisualJ++, utilizar entornos de desarrollo de otros proveedores.
- No utilizar el Internet Explorer, utilizar Netscape Communicator.
- . No usar la Máquina Virtual Java de Microsoft, utilizar la de GNU, Kaffe o Sun.

Si los programadores permitimos que Microsoft acabe con la portabilidad de Java para su propio beneficio financiero, estaremos contribuyendo a que la industria de los ordenadores vuelva a los tiempos donde nada se podía ejecutar fuera de su plataforma nativa.

2.2.1.7. Otros Lenguajes.

Entre otros lenguajes de alto nivel que se pueden encontrar en el mercado, conviene no olvidarse de Logo basado en lenguajes para inteligencia artificial y que incorpora numerosas posibilidades para que los niños aprendan a razonar

Podemos mencionar a Lisp y Prolog, que son lenguajes diseñados específicamente para la realización de programas que siguen los principios de la inteligencia artificial; y **Ada** que es un lenguaje especializado en la programación de sistemas informáticos controlados por microprocesador y que se utiliza bastante en aplicaciones militares.

Con las posibilidades de la transmisión de datos, los ordenadores se pueden comunicar entre sí a grandes distancias, permitiendo la aparición de nuevos servicios y prestaciones.

2.3. Internacionalización

Es el término que indica que un programa puede ejecutarse sin suposiciones sobre su entorno cultural. Los problemas son desde conjuntos de caracteres hasta la interpretación de iconos en interfaces.

Los conjuntos de caracteres de los diferentes alfabetos del mundo son más extensos que el código ASCII.

La mayoría de los países europeos aumentan la codificación ASCII, que sólo define valores hasta 0x7F (7 bits), con caracteres extra para representar las letras de su lenguaje. La codificación Latin-1, comúnmente usada en Europa Occidental, es un superconjunto de ASCII que especifica valores de bytes entre 80 y FF para símbolos y caracteres acentuados; E7, por ejemplo, representa la letra ç. La palabra en inglés boy se representa en ASCII (o en Latin-1) con tres bytes con valores hexadecimales 62 6F 79, mientras que la palabra francesa garçon se representa en Latin –1 con los bytes 67 61 72 E7 6F 6E . Otros lenguajes definen otros símbolos pero no todos pueden caber en los 128 valores no utilizados en ASCII , por lo que hay varios estándares conflictivos para los caracteres asignados a los bytes 80 a FF[5].

Algunos lenguajes simplemente no caben en 8 bits; los lenguajes importantes de Asia tienen miles de caracteres. Las codificaciones empleadas en China, Japón y Corea usan 16 bits por carácter .Para leer un documento escrito en un lenguaje en una computadora preparada para otro es un grave problema de portabilidad. Suponiendo que los caracteres lleguen intactos, leer un documento en chino en una computadora estadounidense involucra, como mínimo, software y fuentes especiales.

El conjunto de caracteres Unicode es un intento de aminorar esta situación mediante el ofrecimiento de una sola codificación para todos los lenguajes en el mundo. Unicode es compatible con el subconjunto de 16 bits del estándar ISO 10646, emplea 16 bits por carácter, con los valores 00FF correspondientes a Latin-1.

Todos los lenguajes más conocidos y los no tan conocidos, están representados en Unicode, por lo que es la codificación a usar para transferir documentos entre países o para almacenar texto multilingue.

Sin embargo, Unicode tiene un inconveniente debido a que los caracteres ya no caben en un byte, por lo que el texto de Unicode sufre de la confusión del orden de bytes.

Para evitar esta situación, los documentos de Unicode se traducen usualmente a una codificación de flujo de bytes conocida como UTF-8 antes de enviarlos entre programas o a una red. Cada carácter de 16 bits se codifica como una secuencia de 1, 2 0 3 bytes para su transmisión. El conjunto de caracteres ASCII emplea los valores 00 a 7F, que caben en un solo byte con UTF-8, por lo que este es compatible hacia atrás con ASCII.
Los valores entre 80 y 7FF se representan con dos bytes, y los valores 800 y superiores se representan con tres bytes.

[5] Todos los valores aquí mencionados están en hexadecimal

La compatibilidad hacia atrás de UTF-8 y ASCII es un beneficio, porque permite a los programas que tratan texto como un flujo ininterrumpido de bytes para trabajar con texto Unicode en cualquier lenguaje.

Para las personas que crean interfaces deben considerar que los diversos lenguajes en ocasiones requieren cantidades muy diferentes de caracteres para decir lo mismo, por lo que debe haber suficiente lugar en pantalla y arreglos.

Para evitar errores deben de estar libres de localismos y frases únicamente con significado para un cierto segmento de población y escribirlos con lenguaje sencillo.
Una técnica común es reunir los textos de todos los mensajes en un solo punto para poderlos reemplazar fácilmente con traducciones a otros lenguajes.

Hay muchas dependencias de lo cultural, como el formato de fecha mm/dd/aa que sólo se emplea en Norteamérica. Si hay alguna probabilidad de que el software se vaya a utilizar en otro país, este tipo de dependencias debe evitarse o minimizarse. Los iconos de las interfaces graficas muchas veces son dependientes de la cultura; muchos son indescifrables para quienes no son originarios del mismo entorno.

Una de las características más interesantes de los proyectos de software libre es el soporte para múltiples lenguajes. Todos estos proyectos usan la biblioteca gettext[6], que es un software GNU que extrae las cadenas de texto de los fuentes de un programa, y los guarda en un fichero que luego cualquiera puede tomar y traducir. Es algo muy sencillo y útil, el cual ha permitido, que los proyectos como GNOME estén disponibles en multitud de lenguajes.

Una limitación de gettext es que sólo funciona con ficheros fuente (.c, etc), por lo que en el proyecto GNOME se ha desarrollado un complemento a gettext que permite incluir en este sistema de traducción otro tipo de ficheros, específicamente ficheros XML. Esto se debe a que, cada vez más y más, se usa XML como formato para muchos ficheros dentro del proyecto GNOME, como por ejemplo los ficheros .oaf, que describen un componente instalado en el sistema, o los .glade, que son los ficheros en los que Glade, que es el diseñador de interfaces de usuario, almacena las ventanas diseñadas.

2.4. Bibliotecas

2.4.1. GTK.

GTK+ es una librería inicialmente creada como parte del proyecto GIMP, es la base de la interfaz gráfica en GNOME que se encarga de los "widgets", como son botones, ventanas, cajas de texto, etc.

[6] gettext: Es una biblioteca que extrae cadenas de texto la cual es un software GNU.

En esta nueva versión, las novedades en cuanto a "widgets" son la sustitución de algunos de ellos que ya no se utilizan y que en su momento fueron utilizados en las versiones anteriores.

Una de estas "adaptaciones" es el sistema de objetos, que originalmente estaba en GTK, ahora se ha pasado a glib. Sin embargo, debido a la gran cantidad de programas que ya usaban el sistema de objetos de GTK (GtkObject), en GTK 2.0 se ha mantenido el API de este sistema para así evitar tener que cambiar miles de líneas de código. Por supuesto, la implementación real que se usa es la de glib, simplemente se ha eliminado la implementación de GTK, pero se han mantenido todas las funciones y tipos de datos.

Otra adaptación, aunque esta vez bastante más completa, es el bucle de ejecución de la aplicación. La parte de glib, incluye facilidades para ejecutar un bucle en nuestra aplicación, aunque como es de esperar, no tiene soporte para eventos relacionados con el sistema gráfico, es decir, movimientos del ratón, pulsaciones de teclas, etc. Así, GTK incluye su propio bucle de ejecución (gtk_main()), que aparte de permitirnos el uso de todas las funcionalidades ofrecidas por GMainLoop, añade soporte para capturar todos los eventos relacionados con la interfaz gráfica, así como para integrar los eventos de glib (alarmas, E/S, etc) en el bucle de GTK.

2.4.1.1. Servicios

GTK ofrece una serie de servicios de alto nivel para las aplicaciones, algunos de los cuales provienen de versiones anteriores de gnome-libs.

Uno de los más sobresaliente y reciente que se ha agregado, es el servicio de portapapeles, algo de lo que, desgraciadamente, el sistema X Window siempre ha carecido (sólo existía el concepto de "selecciones"). Pero con GTK 2.0, se ha agregado una nueva clase, Gtk Clipboard, que permite el intercambio de datos en distintos formatos.

2.4.1.2. Manejo de imágenes.

Con la llegada de GTK 2.0, se agrega a esta librería otra más que, a pesar de existir desde hace tiempo, no era parte de la plataforma estable de GNOME. Esta librería es gdk-pixbuf, que fue creada para sustituir a imlib, que es una librería de manejo de imágenes que resultó estar limitada, y que ha sido incluida como parte de GTK para la versión 2.0, lo cual significa que GTK hace uso directo de esta librería, olvidándose ya por fin de imlib.

gdk-pixbuf es una librería que permite cargar y mostrar imágenes en aplicaciones GNOME. Incluye un conjunto de funciones para las operaciones más comunes en el tratamiento de imágenes gráficas.

2.4.2. GLIB.

GLib es una librería de funciones para programación en C, que contiene multitud de utilidades para facilitar la programación en este lenguaje.

Incluye cosas tan básicas como un conjunto de tipos de datos portables entre arquitecturas, así como la gestión de estructuras de datos, listas enlazadas, arrays dinámicos, tablas de búsqueda, árboles binarios, o la gestión de Entrada/Salida asíncrona, hasta cosas más complejas como el sistema de objetos, que permite simular la programación orientada a objetos en C.

Para que dicha librería sea lo más completa posible, para esta versión se ha agregado el sistema de objetos de GTK a GLib, de forma que no sea necesario enlazar con las librerías GTK (que dependen de X Windows) para usar este potente sistema de objetos, que permite simular, de una forma bastante transparente a los programadores de C, la programación orientada a objetos, en el cual se basan todas las librerías que componen la plataforma GNOME.

2.4.2.1. Tipos de datos portables.

Una de las tareas básicas de glib es el servir de enlace para las diferencias que existen entre los distintos sistemas operativos soportados por esta librería las cuales incluyen un conjunto de tipos de datos portables.

2.4.2.2. Gestión de memoria.

Glib nos ayuda con la gestión de memoria, facilitándonos la escritura de programas legibles y sin interminables revisiones para comprobar si se pudo asignar memoria o no.

También hace uso optimizado de la memoria, liberando sólo cuando es realmente necesario, las funciones de asignación de memoria de glib (g_malloc* y g_new*), si después de varios intentos no se pudo asignar memoria, estas funciones terminan el programa, y evitan las comprobaciones por punteros nulos en nuestro código, ya que tenemos la seguridad de que glib siempre nos devolverá un puntero válido.

2.4.2.3. Estructuras de datos.

Una de las tareas más complejas en el momento de programar en C es el manejo de estructuras complejas de datos. Por ello, glib nos ayuda en el manejo de distintas estructuras complejas de datos, como las listas enlazadas (glist.h), los arrays dinámicos (garray.h), colas (gqueue.h y gasyncqueue.h), "diccionarios" (gquark.h), tablas de hash (ghash.h), etc.

2.4.2.4. Bucle de ejecución.

Una de las características más útiles de glib son los bucles de ejecución. Estos son estructuras opacas que nos permiten ejecutar un bucle que permanece a la espera de diversos eventos, y que envía "señales" a distintas funciones que se registran interesadas en determinados eventos.
Además nos permite, con este sistema, la posibilidad de realizar programas asíncronos (no bloqueantes), tenemos a nuestra disposición distintas características bastante útiles, como las alarmas (eventos que se notifican cada determinado tiempo), o funciones que responden a eventos de dispositivos de E/S, o a tiempos de inactividad en el programa.

2.4.2.5. Entrada/Salida.

Considerando el concepto de portabilidad, glib también ofrece un conjunto de funciones que permiten al programador abstraerse de las diferencias entre plataformas, y centrarse única y exclusivamente en la funcionalidad de su aplicación. En el caso de la entrada/salida, glib ofrece los GIOChanel's, que son una abstracción del acceso a distintos medios de entrada/salida (sockets, ficheros, tuberías, etc).

2.4.2.6. Accesibilidad.

Otros aspectos que se han hecho a GTK para la versión 2.0 es el soporte para accesibilidad (ATK, Accessibility Toolkit – Conjunto de Herramientas de Accesibilidad), que va a permitir el desarrollo de aplicaciones para personas con discapacidades físicas. Este "toolkit" está siendo desarrollado por Sun™, como parte de su compromiso con el proyecto GNOME (han adoptado GNOME como escritorio estándar de Solaris) y de la normativa a punto de ponerse en marcha en los Estados Unidos que exigirá que todo el software que se use en las administraciones públicas debe ser usado por personas con discapacidades físicas.

Así, ATK es un conjunto de clases abstractas que se integran perfectamente con las clases ya existentes en GTK. La razón de que sean clases abstractas es porque ATK simplemente tiene como objetivo la inclusión de opciones de accesibilidad en GTK, y no la implementación del soporte. Dicha implementación se debe realizar en módulos de más alto nivel (es decir, independientes de GTK), y eso es precisamente a lo que se dedican los módulos de más alto nivel como es Gail (GNOME Accessibility Implementation Library), el cual aporta una implementación real de "widgets" con soporte para accesibilidad. Estos incluyen ventanas, botones, etiquetas, orientación a objetos de GTK/glib, y reutilizan la funcionalidad ofrecida por GtkAccessible, que es una clase abstracta existente en ATK.

2.4.3. Pango.

Pango es un completo sistema para la representación de texto en diversos alfabetos. Con la integración de este sistema en GTK, tenemos a nuestro alcance un soporte multi- lenguaje realmente impresionante.

Una de las cosas que cabe destacar en Pango es que, aun formando parte del proyecto GNOME, no es específico para este entorno, por lo que puede ser utilizado en aplicaciones totalmente independientes de GNOME. Aunque se mencione junto a GTK, se distribuye por separado.

Pango usa Unicode (un estándar para el manejo de caracteres de distintos lenguajes/alfabetos) para todo el manejo de texto. Su forma básica de trabajo es que, a partir de una cadena de texto con caracteres Unicode, produce una representación de esa cadena en el alfabeto especificado, soportando incluso la representación de texto no sólo en distintos alfabetos (occidental, hebreo, chino, cirílico, etc), sino que además se adapta a las peculiaridades de cada uno de ellos, como la dirección de escritura (de izquierda a derecha, de derecha a izquierda, de arriba a abajo, etc), la separación/unión entre los distintos caracteres, etc.

El Pango soporta casi todos los lenguajes que pueden existir en el mundo, y puede funcionar a la cabeza de múltiples sistemas de ventanas, incluyendo las tradicionales X fonts, o las fuentes OpenType. Pango se usa en todos los programas que están siendo migrados para la versión 2.0 de Gnome que utilizan el toolkit de GTK.

Estás librerías tienen en cuenta cualquier peculiaridad de esos alfabetos que son un tanto distintos a los nuestros por ejemplo el árabe o el chino, como la dirección de texto como el caso de la escritura de derecha a izquierda y las escrituras tales como Tamil, separación existente entre los caracteres.

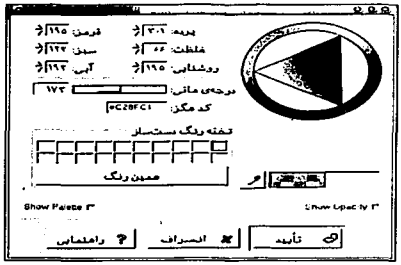
Pango funciona con las librerías GTK+, pero lo hace de forma totalmente transparente al programador, quien sólo debe preocuparse de utilizar el estándar de caracteres unicode.

GTK hace un uso extensivo de Pango, añadiendo por tanto todas estas características a las aplicaciones que se desarrollan para el proyecto GNOME.

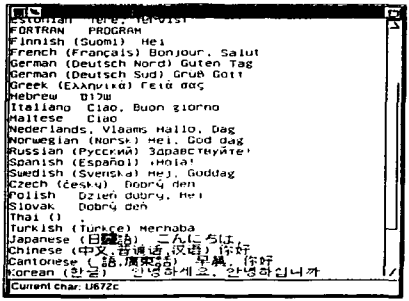
Esta integración de Pango en GTK+ y Gnome, aportará nuevos usuarios al mundo GNU/Linux y consolidará a quienes no estaban del todo conformes por causas del lenguaje.

Pango utiliza los caracteres de Unicode internamente (representado usando Utf-8), y las interfaces de Pango también utilizan Utf-8. Otros pueden ser apoyados usando una biblioteca de la traducción tal como GNU iconv para convertir el texto a Utf-8 antes de procesar.

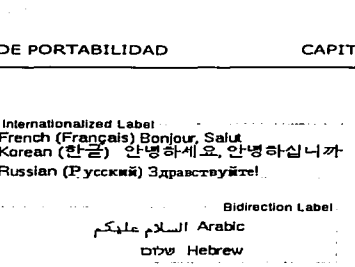
A continuación se muestra una combinación de prototipos de las versiones de Pango con GTK.



En la Figura 2.5 a) Se muestra el selector del color de GTK+.



En la Figura 2.5 b) Se muestran varios lenguajes.



En la Figura 2.5 c) Se muestran las etiquetas de GTK+ en varios lenguajes.

El uso de Pango en GTK+ es obligatorio, pero totalmente transparente al programador, que sólo tiene que preocuparse de usar caracteres Unicode (un estándar para la representación de caracteres en múltiples alfabetos) para todo el texto que quiera representar en pantalla, GTK+ y Pango se encargaran de lo demás.

Pango viene a cubrir las necesidades que muchos usuarios de países con alfabetos distintos al occidental para que puedan usar el sistema X Window. Con Pango, todos estos usuarios podrán usar sus escritorios GNOME en su lenguaje nativo.

Por último, cabe mencionar que la versión de GTK+ para Windows, que existe desde hace más de dos años, forma ya parte de la distribución oficial de GTK+, lo cual significa que Windows es otra de las plataformas soportadas oficialmente por GTK+. Por lo tanto ya hay aplicaciones tales como GIMP o Glade que tienen sus correspondientes versiones en Windows.

CAPÍTULO III

HERRAMIENTAS Y BIBLIOTECAS LIBRES

III. Herramientas y bibliotecas libres.

3.1. Herramientas utilizadas.

3.1.1. GCC.

Las siglas GCC significan GNU Compiler Collection o también en español Colección de compiladores GNU. Antes de nombrarse así se le conocía en el idioma inglés como, GNU C Compiler o Compilador C de GNU. Pero el nombre GCC es el nombre más común para este compilador, es como su nombre lo dice, una colección de compiladores y admite diversos lenguajes: C, C++, Objetive C, Ada, Fortran, y Java. Y por lo tanto este compilador puede procesar los programas fuente de cualquiera de los lenguajes antes mencionados.

El compilador se distribuye bajo la Licencia GPL (General Public License) por lo cual es de libre distribución: se pueden hacer copias de él y regalarlas o venderlas siempre y cuando se incluya el código fuente (o se indique cómo conseguirlo) y se mantenga la licencia.

GCC por definición, es el compilador de C para Linux. Incorpora soporte para ANSI C[1], K&R C[2], C++ y OBJETIVE C[3].

Este programa incorpora entre otras cosas, las siguientes opciones: comprobación del código C a más niveles que sus competidores, creación de información de depurado, optimiza de diferentes maneras el código objeto resultante e incorpora los compiladores antes mencionados (C++ y Objective C). Aun no existen opciones para trabajar con Pascal u otros lenguajes, porque hoy en día no se han desarrollado.

La meta principal de GCC es que sea un compilador rápido y que sea portable, se puede decir que es una de las principales características del lenguaje C, la portabilidad como se vio en el capítulo anterior. Otra razón es porque la mayoría de los programadores desarrollan en C o en C++. El núcleo mismo de Linux está escrito en C. El compilador de base en Linux es GCC. Éste ha sido escogido como compilador de C por la Agencia Espacial Europea a causa de su estabilidad y del hecho que por ser un Software Libre, la continuidad de su existencia está asegurada, por la razón de que en el desarrollo de este compilador participan cientos de voluntarios de todo el mundo que lo siguen depurando.

---

[1] ANSI C: The American National Standars Institute (ANSI) versión de C" surgio de la necesidad de estandarizar a C y surgió esta versión después de muchas incompatibilidades entre las versiones del propio lenguaje

2 K&R C: Lenguaje programación que lleva las iniciales de sus creadores (Brian W. Kernighan y Dennis M. Ritchie) y cuando salió se convirtió en lo conocemos como lenguaje C.

[3] Objetive C: Lenguaje orientado a objetos

Existen versiones para prácticamente todos los sistemas operativos. Viene incluido en la mayoría (sino en todas) las distribuciones de GNU/Linux. La versión DOS de este compilador es el DJGPP el cual se mencionara en este capítulo más adelante.

El GCC se puede conseguir en la página oficial del GCC: gcc.gnu.org. Para más información sobre la versión DOS/Windows (DJGPP) puedes consultar en diversas paginas en la bibliografía a las cuales se dan de referencia o si lo prefieres en el capítulo siguiente.

3.1.2. Cygwin.

A continuación se da una breve explicación de la razón por la cual se hizo uso del Cygwin, en lugar de las herramientas que se menciona en este capítulo más adelante. Es importante que se tenga en cuenta que en este trabajo no se obliga a nadie a utilizarlo, dejando el hecho a la libre elección de cada usuario.

3.1.2.1. Motivos por el cual se va a utilizar Cygwin.

En algunas ocasiones se necesita tener dos sistemas operativos al mismo tiempo, ya que ciertas aplicaciones sólo puede funcionar en uno o en otro entorno, pero no en ambos, y como se sabe aunque el sistemas operativo más usado es Windows, no por esto suele ser el mejor. El Cygwin surgió a partir de una necesidad de disponer de aplicaciones que se ejecuten en dos o más plataformas sin la necesidad de hacerles a las mismas demasiados cambios, el tener dos sistemas operativos en una misma máquina, consume muchos recursos de la misma, entre ellos el tiempo, además teniendo en cuenta que el espacio en disco duro que necesitan ambos sistemas aun en su instalación mínima, aunque existe la posibilidad de instalar dos sistemas en la misma máquina. Una opción conveniente fue la de diseñar un emulador Unix/Linux, aunque no sólo es eso, sino que viene con la mayoría de las funcionalidades de Unix.

Aunque Linux últimamente ya no es tan complejo de instalar, ahora se puede decir que es tan fácil en su instalación como el propio Windows, ya que posee una gran diversidad de controladores. Pero no siempre es factible que los usuarios cambien de plataforma, por varias razones. Una de ella puede ser, que hay poca información que los usuarios tienen sobre el sistema.

La configuración del Cygwin no tiene mayor problema al momento de bajar la aplicación, sólo tienes que entender las instrucciones que va dando la misma herramienta, y si surge una duda, en el capítulo siguiente se te irá guiando paso a paso por toda la instalación con una explicación detallada de cada uno de las aplicaciones que contiene el programa completo, pero si estas totalmente familiarizado con Linux, esto no te dará el menor problema.

Claro que depende lo que vayamos a utilizar de este entorno al momento de instalarlo en alguna máquina ya que como se verá a detalle en el capítulo siguiente, en el instante de que se realice la instalación, da una serie de opciones que se deben considerar como necesarias o no, dependiendo de los usos del programa.

El Cygwin nos trae una gran ventaja, que es la compatibilidad de hardware al momento de instalarlo, ya que sistemas operativos causan problemas en ese aspecto tan importante, con esta herramienta esos problemas se podría decir que están resueltos.

3.1.2.2. Antecedentes de Cygwin.

Su nombre proviene del arreglo de los siguientes nombre, cygnus + gnu + Windows = Cygwin. Este es un entorno Unix desarrollado para Red Hat, para Windows. Éste consiste de dos partes:

- ✓ En una biblioteca DLL[4] (Cygwin.dll) la cual actúa como un emulador Unix/Linux el cual provee funcionalidades API (Application Programming Interface, Interfaz para Programación de Aplicaciones).
- ✓ Una colección de herramientas para Unix, la cual provee un aspecto Unix/Linux

El Cygwin es una de las herramientas más populares que posee el proyecto de desarrollo GNU para Windows NT ó 9x. Funcionan con el uso de la biblioteca que proporcionan las llamadas y el ambiente del sistema UNIX que estos programas requieren.

Con las herramientas instaladas, los programadores pueden escribir en la consola de Win32[5] o las aplicaciones gráficas de la GUI que hagan uso de la API estándar Win32 Microsoft y/o la API del Cygwin. Consecuentemente, es posible portar fácilmente muchos programas significativos de UNIX, portarlos sin la necesidad de cambios extensos al código de fuente. Esto incluye las configuraciones de desarrollo y la construcción de la mayoría del software disponible de GNU (herramientas de desarrollo incluidas con las distribuciones de Cygwin). Incluso si las herramientas del compilador son menos usadas, se

[4] Dynamic Link Library (Biblioteca de vínculos dinámicos) es un archivo que contiene funciones que se pueden llamar desde aplicaciones u otras Dll. Los desarrolladores utilizan las Dll para poder reciclar el código y aislar las diferentes tareas. Las Dll no pueden ejecutarse directamente, es necesario llamarlas desde un código externo. Se ahorra memoria cuando se ejecuta una aplicación.

[5] Consola de Win32: Es el sistema MS-DOS, o en su defecto en versiones recientes la consola (modo texto) que lo simula.

puede tener interés en las muchas utilidades estándares de UNIX. Pueden ser utilizados por ambos el shell bash[6] (proporcionado) o el command.com.

El Cygwin es una herramienta de que pertenece a la familia del software libre, esto quiere decir, que no se debe de pagar ningún costo por adquirirlo o utilizarlo, además de que se debe proporcionar el código fuente del mismo. Este software está protegido por la Licencia GPL.

3.1.2.3. Expectativas para los programadores de Unix/Linux.

Los desarrolladores que vienen de un trabajar con UNIX encontrarán un sistema de utilidades que ellos ya conocen y están totalmente familiarizados con su uso, incluyendo el shell[7] de Unix, las herramientas del compilador son los estándares del compilador GNU, sólo cuando son portadas al host Windows encontraran alguna diferencia . Los programadores que desean portar el software de Unix/Linux hacia Windows NT ó a 9x encontrarán que la biblioteca de Cygwin proporciona una manera fácil de portar muchos paquetes de UNIX/linux con sólo un mínimo de cambios en el código fuente.

3.1.2.4. Expectativas para los programadores de Windows.

Los programadores que vienen de un ambiente Windows encontrarán un sistema de herramientas capaces de escribir en la consola o en los ejecutables de la GUI que dependen de la API Win32 de Microsoft. Las utilidades del linker y del dlltool quizás fueron utilizadas para escribir la Biblioteca de vínculos dinámicos (Dynamic Link Library, DLL) de Windows. Todas las herramientas se pueden ser utilizadas desde las líneas de comandos de Microsoft, con el soporte completo de las rutas normales de Windows.

3.1.2.5. Funcionamiento de Cygwin.

Cuando un binario ejecutable vinculado con la biblioteca es ejecutado, la DLL del Cygwin es cargada dentro del segmento de texto de la aplicación, pero si se requiere emular el Kernel de Unix/Linux quizás se necesita el acceso a todos los procesos que corren bajo Linux, la primera biblioteca que se ejecuta es la DLL de Cygwin, ésta crea áreas de memoria compartidas que otros procesos usando instancias separadas, de las DLL que pueden tener acceso. Esto es usado para conservar los rastros de los descriptores de los archivos abiertos y auxiliar en las llamadas al sistema *fork* y *exec*. Esto se agrega a las áreas de memoria compartidas. Por cada proceso también hay una estructura de pre-proceso que

[6] Shell bash: Interprete de comandos más popular.
7 Shell: Es una capa intermedia entre el Sistema Operativo y el usuario

contiene información tal como identificador de proceso, identificador de usuario, mascaras de señales, y otras estructuras similares a los procesos de información específicos.

En las primeras etapas del proceso de desarrollo se toman importantes decisiones de diseño que no tendrán que estar estrictamente apegadas al estándar existente de Unix como POSIX[8], si esto no fuera posible o si disminuyera significativamente el uso de las herramientas en la plataforma win32.

3.1.2.6. Soporte tanto a Windows NT como 9x.

Mientras el Windows 95 y el Windows 98 son similares uno del otro seguramente podemos ignorar su diferencia cuando se instale el Cygwin, pero Windows NT es extremadamente diferente. Por esta razón, cuando la biblioteca DLL es cargada verifica que sistema operativo está activo para actuar en consecuencia.

En algunos casos, el APIWin32 sólo es diferente por razones históricas. En esta situación, la misma funcionalidades básica está disponible dentro Windows 9x y NT, pero los método usados para obtener esta funcionalidad difieren.

3.1.2.7. Acceso a Archivos.

Cygwin soporta tanto a las rutas tipo Win32 como POSIX, usando ambas diagonales invertidas (//) como delimitadores del directorio los archivos en la DLL son trasladados de Win32 a POSIX cuando es necesario. Como resultado la biblioteca permite que los archivos del sistema sean completados a POSIX, interpretando las rutas nuevamente a Win32 para lo que vuelve a llamar a la función de la API win32. Empezando con dos diagonales // que también son soportadas.

El esquema de POSIX examina el espacio del sistema de archivos de Windows y éste es almacenado en el registro de Windows, mientras la diagonal de raíz (/) señala los directorios de la partición del sistema se da por defecto, esto es fácil de cambiar con la utilidad de montaje del Cygwin, adicionalmente se podrá seleccionar la partición, esto permite montar arbitrariamente rutas win32 en el espacio del sistema de archivos del POSIX muchas personas usan la utilidad para montar cada letra de unidad dentro de la partición raíz. (e.g. C:\ to /c, D:\ to /d, etc...).

La biblioteca exporta varias funciones específicas de Cygwin que pueden ser usadas por programas externos para convertir rutas o listas de rutas de win32 a POSIX o viceversa.

[8] POSIX: Portable Operating System Interface Es un estándar con una serie de normas definidas para permitir la portabilidad entre diferentes sistemas UNIX. GNU/Linux cumple con este estándar.

Los scripts de los shells no pueden llamar a estas funciones directamente pero pueden hacer la misma conversión ejecutando la utilería cygpath que provee Cygwin con el sistema de archivos win32. En la práctica Cygwin no hace distinción entre mayúsculas y minúsculas, Mientras que algunos programas de Unix en la actualidad si lo hacen, esto puede dar dificultadas para las aplicaciones de tipo no Cygwin que operan sobre estos archivos.

3.1.2.8. Creación de Proceso.

Las llamadas fork son particularmente interesantes porque no hacen búsquedas en la parte superior de la API de win32. Esto lo hace muy difícil de implementar correctamente. Actualmente las llamadas fork, del Cygwin son implementaciones en las cuales no se puede escribir o copiar.

La primera cosa que sucede cuando un proceso padre fork (bifurca) a un proceso hijo es que el padre inicializa un espacio en la tabla de procesos de Cygwin para el hijo. Entonces crea un proceso hijo suspendido usando le llamada CrearProceso de Win32. Posteriormente, el proceso padre llama a setjmp para guardar su propio contexto y poner un puntero en una área de memoria compartida de Cygwin (memoria entre todas las tareas de Cygwin). Entonces llena las secciones de copiado del .data y .bss del nodo hijo desde su propia dirección dentro de la dirección del hijo. Después la dirección del hijo es inicializada, el hijo se ejecuta mientras el padre espera en estado del objeto de exclusión mutua.

Mientras que se tienen algunas ideas acerca de cómo acelerar esta implementación de la bifurcación reduciendo el número de interruptores del contexto entre el proceso padre y proceso hijo, éste por lo regular es ineficiente bajo Win32. Afortunadamente, en muchas circunstancias la familia de llamadas de bifurcaciones provista por Cygwin puede ser sustituida por la pareja bifurcación/ejecución con sólo un poco de esfuerzo. Estas llamadas limpian la parte superior del mapa de memoria de la API win32, esto es más eficiente.

Por otra parte las bifurcaciones y las ejecuciones presentan sus propias dificultades porque no hay un camino para hacer una ejecución, Cygwin a inventado sus propios identificadores de proceso (PIDs), como resultado, cuando un proceso ejecuta varias llamadas tiene múltiples PIDs de Windows asociados con un solo PID de Cygwin.

3.1.2.9. Señales.

Cuando un proceso de Cygwin inicia, la biblioteca comienza un segundo enlace a un proceso para utilizar la señal. Este proceso espera a que los eventos de Windows sean usados para pasarle las señales a otros procesos. Cuando un proceso detecta una señal, éste busca la mascara de bits de las señales y el manejador apropiado.

Surgen demasiadas complicaciones en la implementación por el hecho de que el manejador de señales opera en la misma dirección de espacio que el programa se está ejecutando. La consecuencia inmediata es que las funciones del sistema Cygwin son interrumpidas, a menos que se tenga un especial cuidado para prevenir todo esto, se pueden hacer algunas cosas para evitar el envío de estas señales de interrupción en el caso de que un proceso envíe una señal a otro proceso.

3.2. Las bibliotecas GNOME.

El proyecto GNOME es software libre el cual se compone de un ambiente de escritorio muy fácil y vistoso para el usuario, aun más que el propio Windows. Y para el programador es una poderosa herramienta de trabajo, ya que en él se pueden diseñar aplicaciones muy sencillas o complicadas según sea el caso. GNOME forma parte del proyecto GNU y es software libre.

3.2.1 GTK +.

GTK+ es una biblioteca escrita en C, que reúne un conjunto de widgets (elementos de interfaz gráfica) orientados hacia la programación de aplicaciones gráficas en X Window. Está altamente orientada a objetos y se acopla con los lenguajes más populares, como C++, Objective C, Perl, TOM, Guile, Python, etc. GTK+ además usa GLib, que es una biblioteca en C muy útil, incluye ayuda para portar los programas a diferentes plataformas y contenedores como listas enlazadas o tablas de claves.

GTK+ es una herramienta multiplataforma para crear interfaces gráficas de usuario ofreciendo un completo paquete de widgets, también es adecuado para crear desde pequeños proyectos, hasta aplicaciones completas

GTK+ es software libre y pertenece al proyecto GNU está protegido por la licencia LGPL, por lo tanto los programadores que utilicen software restringido, comercial, etc., pueden utilizar también está biblioteca, claro sin quitarle ninguno de sus características de software libre que le da la licencia LGLP.

GTK + es llamado paquete de herramientas de Gimp[9] porque fue originalmente escrito para desarrollar el programa de manipulación de la imagen de (GNU) GIMP, ahora es diferente, ya que hoy en día, se utiliza en diversas aplicaciones y proyectos como pueden ser: el modelo objeto de ambiente de red de GNOME. Éste se construye encima de GDK, (Herramientas de dibujo de Gimp) que es básicamente una envoltura alrededor de las funciones básicas para acceder a funciones fundamentales del windowing (Xlib en el caso del sistema de las ventanas de X), y del gdk-pixbuf, una biblioteca para la manipulación de la imagen del cliente-servidor. Como se muestra en la figura 3.1.

[9] Gimp: Programa para la creación y retoque de imágenes fotográficas.

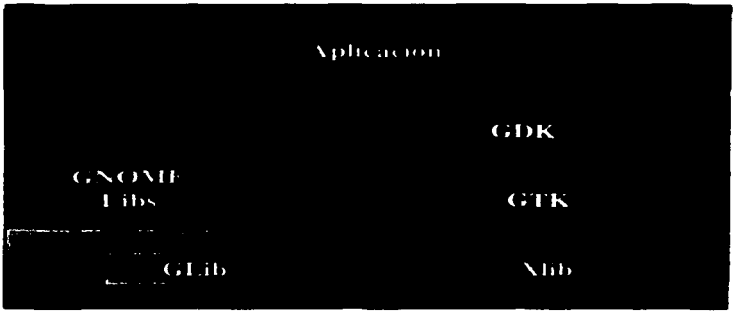


Figura 3.1.

GTK+ es esencialmente una programación de interfaces orientadas a objetos. Aunque está escrito totalmente en C, se ejecuta usando la idea de las clases y de las funciones del servicio repetido (indicadores a las funciones).

Recientemente a GTK+ le realizaron mejoras y le anexaron algunas herramientas, a continuación se mencionan y se da una breve explicación de las mismas, éste está intregrado por las siguientes bibliotecas.

1. Glib
2. GDK
3. GdkPixbuf
4. Pango
5. ATK

3.2.2. Glib.

Es la biblioteca de bajo nivel que forma la base de GTK+ y GNOME. Proporciona los datos estructurados manipulados por C.

Contiene reemplazos, para ciertas llamadas estándares, así como otras funciones adicionales que manejan listas encadenadas. Las funciones del reemplazo se utilizan para

aumentar la portabilidad de GTK, como algunas de las funciones puestas en ejecución, no son estándares en otros sistemas Unix.

3.2.3. GDK.

GDK (Graphics Drawing Kit. Conjunto de Herramientas de Dibujo Gráfico) es una biblioteca que forma parte de una capa de bajo nivel para dibujo, situada entre GTK+ y la interfaz para programación de aplicaciones API. GDK, está basada sobre Xlib.

GDK proporciona la capacidad de dibujar hasta el nivel de pixel y proporciona funciones de bajo nivel para la creación y manipulación de ventanas.

El utilizar rutinas de GDK para escribir una aplicación no es más fácil que utilizar Xlib directamente. Pero, para esta biblioteca GTK+ proporciona un widget que puede emplearse para crear aplicaciones que necesiten realizar algo de dibujo manual, este widget se llama GtkDrawingArea.

Todas las funciones están escritas para tener una manera de acceder a las funciones Xlib de una manera más fácil y automática Además, ya que GDK usa la biblioteca GLib, será más portable y seguro de usar en múltiples plataformas.

Es la única parte de GTK+ que tiene que ser reescrita para soportar otra plataforma. Es por esta razón por la que GTK+ ya ha sido portada a varios sistemas operativos (X Window, MS Windows, QNX, BeOS, etc.)

3.2.4. Gdk-Pixbuf.

En una biblioteca que sirve para manipular imágenes, se pueden cargar, visualizar, grabar, etc. Es una de las mejoras de la versión 2 de GTK, en la versión anterior (GTK 1.2) de GTK ya se había utilizado una biblioteca parecida, sólo que esta era menos eficiente, por lo tanto ya quedo totalmente obsoleta, esta se llamaba Gdk-imlib que también se utilizaba para la manipulación de imágenes.

Sólo con la biblioteca Gdk-Pixbuf se pueden realizar aumento o disminución del mapa de bits (imagen) lo que se le conoce como Zoom, además de guardar proporcionalidad con respecto a otra imagen (escalable). Puede utilizar distintos formatos (png, gif, jpeg, etc.).

3.2.5. Pango.

Una de las mejoras más importantes de la nueva versión de GTK 2 es Pango, una biblioteca de software libre, que facilita la representación y visualización del texto internacional en la pantalla. Se puede decir que trabaja con casi todos los lenguajes que pueden existir en el

mundo, y puede funcionar en múltiples sistemas de ventanas incluyendo las tradicionales X fonts, o las fuentes OpenType.

Como en el mundo hay demasiados sistemas de codificación tan sólo en un continente y aunque existe el código ASCII este no es suficiente, ya que para algunos usuarios son muy importantes símbolos o letras que otros no utilizan o incluso desconocen, por ejemplo la @ tan importante para el manejo de correo electrónicos pero éste no es símbolo común en todos los idiomas existentes. Además de que algunos idiomas son muy complicados de escribir como los orientales. Aun en el idioma inglés que se presume es un idioma universal, no había un único sistema de codificación que se adecuara a todas las letras, signos de puntuación y símbolos técnicos de uso común.

Además, los sistemas de codificación presentan problemas entre ellos. Es decir, dos sistemas de codificación pueden utilizar el mismo número para dos caracteres distintos o bien utilizar números distintos para el mismo carácter. Por estas razones Pango utiliza unicode el cual proporciona un número único para cada carácter, sin importar la plataforma, el programa, o el idioma.

3.2.6. ATK.

Como se menciona en el capítulo anterior, esta biblioteca: ATK contiene un juego de interfaces de accesibilidad, este juego de herramientas puede ser usados en aplicaciones tales como lectores de pantalla, amplificadores, y los dispositivos de la entrada alternativos.

Esta es una mejora en la versión de GTK 2 su siglas significan conjunto de herramientas de accesibilidad que el cual permite el desarrollo de aplicaciones para personas discapacitadas físicamente.

GTK+ y bibliotecas asociadas (GLib, Pango, Atk) están disponibles en el sitio:

ftp://ftp.gtk.org/pub/gtk/v1.3/

3.3. Alternativas Existentes.

Para desarrollar una metodología general, para crear aplicaciones multiplataforma hay varias opciones, claro que no todas se desarrollan con software libre, sino por el contrario estas herramientas que podrían ayudar para este propósito tienen un costo, por eso se mencionan, pero no se tomaron en cuenta para el desarrollo de este proyecto. Ya que la finalidad del mismo es la de realizar una metodología con software libre.

3.3.1. Biblioteca Qt.

Qt es la primera de muchas bibliotecas gráficas para el desarrollo de programación orientada a objetos, se pude decir que compite con GTK y se ha usado para el desarrollo de KDE, el cual es un entorno con diversas aplicaciones y una administración de ventanas disponible para cualquier aplicación y que está muy por encima de la interfaz gráfica de Windows.

Como se mencionó ésta permite el desarrollo de interfaces gráficas muy sencillas y vistosas. El lenguaje de programación que utiliza es C++. Qt es una biblioteca de C++ que proporciona un toolkit (juego de herramientas) para el desarrollo de interfaces graficas multiplataforma. Qt esta totalmente orientado a objetos.

El inconveniente de Qt es que su versión para Windows no es libre. Una de las principales características de Qt, es sin duda, su orientación multiplataforma. De esta manera, podemos escribir aplicaciones portables a MS/Windows, Unix/X11, Mac y sistemas embebidos.

Qt es un producto de la compañía noruega trolltech y se distribuye en diferentes versiones. Estas son:

- ✓ Qt Enterprise
- ✓ Qt Professional
- ✓ Qt Free
- ✓ Qt/Embedded Free.

Las dos primeras, permiten la construcción de software comercial, e incluyen actualizaciones y soporte técnico. La tercera, Qt Free, es la versión de Qt para Unix/X11 para el desarrollo de software gratuito y abierto. Esta versión reside bajo licencia GPL. Ésta última permite el desarrollo de software libre.

Qt es soportado por las siguientes plataformas:

- ✓ MS/Windows -- 95, 98, NT 4.0, Millenium y 2000
- ✓ Unix/X11 -- Linux, Sun Solaris, HP-UX, Compaq Tru64 UNIX, IBM AIX, SGI IRIX.
- ✓ Macintosh -- Mac OS X
- ✓ Embebido -- Plataforma Linux con soporte para segmentes de memoria.

Las dos herramientas principales de productos de Qt son:

La QT/Escritorio: Herramientas para aplicaciones para desarrolladores profesionales en Mac Ms Windows, Linux y Unix.

El Qt /Embebido: Herramientas para desarrolladores y aplicaciones para dispositivos de internet y computadoras palmtop.

3.3.1.1. La biblioteca QT/Escritorio.

Qt proporciona una API independiente de la plataforma: GUI, (Interfaz de usuario gráfica), acceso a bases de datos, conectando a redes de computadoras, manejo de archivos, etc. Las bibliotecas Qt encapsulan las diferentes APIs,o diferentes sistemas operativos, proveyendo las aplicaciones con una sola API en común para todas las operaciones del sistema. La API nativa de C está encapsulada en un conjunto bien diseñado, con las clases de C++, totalmente orientadas a objetos.

Se recomienda a los usuarios, sólo en casos en los cuales la aplicación redituć el costo de esta herramienta, ya que como es bastante aceptable, reduce el tiempo cuando se porta una aplicación de una plataforma a otra, y simplemente se puede compilar el programa.

Con una herramienta así, se pude decidir posteriormente en que plataforma se desea instalar cualquier aplicación diseñada. Reduciendo tiempo de diseño al momento de decidir cual plataforma es más conveniente. Sin temor a una equivocación costosa.

El desempeño de Qt no está basado en emular un sistema operativo. Éstas herramientas son aplicaciones nativas del lenguaje C++, compiladas en cada plataforma.

Las herramientas Qt para escritorios contienen los siguientes productos:

- ✓ Qt/Windows está diseñado para MS Windows 95/98/ME, NT4, 2000 y XP.
- ✓ Qt/X11 está diseñado para Linux, Solaris, HP-UX, IRIX, AIX, y muchos otras variantes de Unix.
- ✓ Qt/Mac está diseñado para Apple Mac OS X.

Una versión no-comercial de la versión Qt/x11, es la estándar del juego de herramientas para aplicaciones de GUI en Linux. El escritorio KDE está basado en QT/X11.

3.3.1.2. La biblioteca Qt/Embebido.

Es una versión de Qt diseñada para recursos restringidos de sistemas embebidos. Esto proporciona funcionalidad completa en GUI sin requerir de X11[10] o Motif[11]. Lo cual reduce substancialmente la memoria y el CPU demandado por el software embebido.

Qt/Embebido proporciona lo mismo que las herramientas QT/Escritorio, aunque puede ser configurando nuevamente, según lo requerido por el sistema. De esta manera, el consumo de memoria del software incluido, logra reducirse más allá. Qt/Embebido es totalmente compatible con las versiones del escritorio de Qt. Logra migrarse entre los escritorios y los sistemas embebidos con una simple compilación. Qt/Embebido está actualmente disponible para los sistemas de Linux.

Qtopia es un ambiente de ventanas y aplicaciones de serie, diseñadas para PDAs, (computadoras palmtop), aplicaciones de Internet, y dispositivos similares. Esto es completamente basado en Qt/Embebido.

Qtopia incluye un juego completo de Manejo de Información Personal (PIM). Contiene aplicaciones como, calendarios, libreta de direcciones, listas de tareas, etc. También contiene correo para clientes, juegos, utilerías de configuración y más.

3.3.2. Kylix.

El proyecto Kylix es la versión de Delphi y C++Builder para el sistema operativo Linux. El cual es de una herramienta de desarrollo rápido (RAD) ésta permitirá diseñar aplicaciones para el sistema operativo Linux muy fáciles y rápidas. Un ejemplo de esto es que se pueden desarrollar aplicaciones de bases de datos, que funcionen sobre Internet. El entorno de desarrollo será completamente visual, el lenguaje sobre el que se apoya es el Object Pascal y C++, la biblioteca de clases es la CLX (nueva biblioteca de Borland Inprise para Delphi y C++ Builder), la cual se apoya sobre la biblioteca Qt para ofrecer una total esencia del sistema operativo.

Como se menciona Kylix no es libre, aunque este funcione sobre Linux el cual sí lo es, como se sabe no todo el software que se utiliza en esta plataforma es de este tipo. Con esta nueva herramienta se propone una nueva opción, como se sabe, Linux es más estable que cualquier versión de Windows y por lo tanto, se dio pie a esta nueva alternativa, como se sabe Linux está basado el la tecnología de Unix de ahí su eficiencia. Kylix reconoce el lenguaje es el Object Pascal en el caso de Delphi para Linux, y C++ en el de C++ Builder para Windows.

[10] Motif: Biblioteca de funciones para el desarrollo de aplicaciones gráficas.
[11] X11: Biblioteca utilizada para el entorno de ventanas de Unís (X Windows)

Kylix tiene un ambiente de desarrollo muy similar al actual de Delphi y C++Builder. El ambiente de desarrollo está realizado con el propio Kylix. Esté posee un compilador de alto rendimiento con un ambiente visual y herramientas de desarrollo, que principalmente sirven para habilitar las herramientas de aplicaciones de desarrollo. Kylix permite el desarrollo rápido de comercio electrónico con Servicios Web en el sistema operativo Linux. Se pueden construir aplicaciones gráficas GUI amigables para el usuario final con la facilidad de un ambiente de desarrollo drag-and-drop (arrastrar y soltar).

Debido a que Kylix y Delphi comparten una biblioteca de componentes común, el código fuente desarrollado con Kylix puede ser compilado en Delphi (y viceversa), permitiendo un uso eficiente en implementaciones de aplicaciones miltiplataforma, para usuarios convencidos de diseñar en aplicaciones en ambas sistemas operativos, Linux y Windows.

3.3.2.1. Aplicaciones de libre distribución en Linux.

Linux está siendo utilizado cada vez por más usuarios que confían en él, éstos una vez que hacen uso de él, se convencen de su consistencia, escalabilidad y bajo costo, Linux es el estándar más factible para servidores Web y rápidamente comienza a ser una alternativa a Windows en los computadoras. El desarrollo de aplicaciones para Linux con las herramientas tradicionales resulta difícil. Las herramientas de desarrollo tradicionales de Linux son difíciles de aprender, y no proporcionan la productividad que espera un desarrollador habituado a los rápidos ciclos de desarrollo de las herramientas para Windows. Borland Kylix cambia esta idea.

Con el entorno de diseño visual de Kylix se podrán crear sofisticadas aplicaciones Linux más rápidamente que otra herramienta de desarrollo de su tipo. Todo esto gracias a que posee una combinación única de herramientas de diseño visual, compilador y depurador de código nativo.

3.3.2.2. Requerimientos Mínimos de Sistema:

- ✓ Intel Pentium 200 MHz (PII a 400 MHz recomendado)
- ✓ 64 MB RAM (128 MB RAM recomendado)
- ✓ CD-ROM
- ✓ 175 MB espacio de disco duro para instalación completa
- ✓ Monitor VGA o de mayor resolución
- ✓ Ratón

3.3.2.3. Biblioteca de clases.

Kylix utiliza dos bibliotecas diferentes una estática y otra dinámica, estas son la CLX y si se necesita hacer uso de funciones del núcleo (kernel) del sistema operativo, podemos importarlas desde los Shared Objects que es lo mismo que las DLL de Windows.

3.3.2.4. Biblioteca CLX.

Kylix no tendrá la Biblioteca de Componentes Visuales (Visual Components Library, VCL), ya que es muy utilizada por el API de Windows (Win32 API). En su lugar, incluye Biblioteca de Componentes para de Desarrollo Multiplataforma, CLX (Component Library for Cross-platform Development), muy similar a la VCL, con la diferencia de que la implementación utiliza la biblioteca de componentes Qt de la compañía Troll. Qt está disponible tanto para Linux como para Windows. CLX proporciona una paleta mejorada de unos 165 componentes reutilizables, que se pueden ampliar y personalizar.

La biblioteca de QT es libre sólo si se desarrolla software libre: Si no es así este debe de ser comprada. Si el software que se desarrolle es comercial se puede prescindir de esta herramienta.

Esta biblioteca es totalmente nativa para Linux y Windows, también maneja una arquitectura orientada a objetos, CLX simplifica el desarrollo de interfaces graficas del usuario de las aplicaciones con acceso a bases de datos y aplicaciones Web, con la arquitectura de componentes multiplataforma basada en la librería visual de componentes (VCL)

3.3.2.5. Bibliotecas de enlace dinámico para Linux (Shared Objects).

En Kylix también se pueden crear bibliotecas de enlace dinámico (son los llamados archivos .DLL en Windows), que en Linux también existen con la extensión .so (Shared Objects), y la importación de procedimientos y funciones que estén declarados en las mismas.

3.3.2.6. Motor gráfico que utiliza Kylix.

La biblioteca CLX de Kylix utiliza Qt para generar controles visuales. Qt funciona muy bien sobre ambos escritorios de Linux tanto en el de KDE como sobre el escritorio GNOME. A su vez esta biblioteca hace llamadas al gestor de ventanas que se esté utilizando, el cual pasará las llamadas al servidor gráfico XWindows. Utilizando la capa de abstracción que es Qt, se puede migrar fácilmente una aplicación a Windows, cuando

Delphi soporte la biblioteca CLX, ya que Qt para Windows hace llamadas al GDI[12] de Win32. Es Qt quien se encarga de resolver las llamadas de CLX y conducirlas dependiendo del sistema operativo que se esté usando.

3.3.3. DJGPP.

DJGPP es un compilador para DOS de lenguajes C y C++ aunque también es válido para algunos otros lenguajes. Fue desarrollado por DJ Deloire. El código que produce está diseñado para microprocesadores de 32 bits y permite acceder de forma directa a toda la memoria, pero hubo un problema DOS funciona con 16 bits, por lo que se necesito un programa para que pudieran ser ejecutado 32 bits bajo el propio DOS. Para ello se hace uso del estándar llamado DPMI. (DOS Protected Mode Interface) Interfase de Modo protegido, DOS-Extender. Debido a que DOS no posee servicios de DPMI (Win3.1, Win95 y OS/2 si) también se tuvo que desarrollar un servidor de DPMI.

Actualmente DJGPP es un entorno completo de programación el cual posee casi todas las herramientas creadas para GNU.

Como pertenece al proyecto GNU está protegido por la licencia GPL Dado que es de libre distribución se puede copiar y usar libremente, además incluye el código fuente. Este compilador está siendo constantemente desarrollado por personas a través de Internet, de manera parecida al sistema operativo GNU/Linux. No existe un servicio técnico, pero sin embargo existe una página FAQ (Frecuent Ask Question) de preguntas más frecuentes y una lista de correo en las cuales los usuarios resuelven sus dudas .

3.3.3.1. Interfaz de usuario.

Uno cosa que se debe de tomar muy en cuenta en cualquier programa, es la interfaz con el usuario. Ya que dependiendo que tan amigable sea para con el mismo, es como va a ser frecuente su uso, si una aplicación tiene una interfaz atractiva, pero no posee la calidad de otros programas que posean una interfaz menos amigable, ya no es tan atractivo. Esto es uno de los motivos por lo que Unix no es tan usado como el propio Windows. Aunque se ha tratado de evitar esta situación con X-Windows. La interfaz de usuario del DJGPP es muy similar a la que utiliza Unix, y por lo tanto es algo complicado, por esta causa se creo un Entorno de Desarrollo Integrado IDE (Integrated Development Environment). Un IDE es un aplicación que integra el editor de texto para escribir un programa, el sistema de ayuda, el compilador, el linker, el debugger y otras herramientas necesarias para el proceso

[12] Interfaz de Dispositivo Gráfico. Proporciona capacidades para manejar gráficos vectoriales bidimensionales, proyección de imágenes y posibilidades para caracteres tipográficos de manera optimizada.

de programación. Se decidió programar esta aplicación en C/C++ porque DOS es muy compatible con este lenguaje.

Al principio este compilador tenia algunas fallas: No tenía ayuda y además poseía un editor de textos difícil de operar, carecía de una herramienta para buscar errores en los programas, herramienta llamada Debugger.

Ahora posee herramientas mejoradas, se creo un editor mucho más amigable, resaltará en color la sintaxis del programa capturado, grabación de teclas, para no volver a escribirlas, seleccionar, borrar y copiar bloques de texto, etc.

3.3.3.2.Cuadro Comparativo

Caracteristicas de Kylix	Caracteristicas de Free Pascal/GNU Pascal	Caracteristicas de wxWindows	Caracteristicas de Java
✓ Software comercial ✓ Entorno de desarrollo visual ✓ Desarrollo para el servidor web ✓ Aplicaciones compiladas de forma nativa ✓ Biblioteca de componentes para desarrollo multiplataforma(CLX) ✓ Desarrollo Delphi para Linux ✓ Depurador completamente integrado ✓ Acceso a las API del sistema operativo	✓ Software libre. ✓ Es un compilador para DOS de 32 bits C/C++. ✓ No existe soporte. ✓ Tiene una FAQ para consultar y una línea de correo para dudas. ✓ Permite crear aplicaciones Windows directamente. ✓ Se desarrolla de manera similar al sistema GNU. ✓ No es un emulador de Linux	✓ Es Software libre. ✓ Biblioteca de componentes para desarrollos multiplataforma (Glib) ✓ Adquisición fácil ✓ Esta siendo mejorada constantemente ✓ No tiene secretos ya que el código fuente esta disponible. ✓ Para aplicaciones multiplataforma junto con el emulador Cygwin es como trabajar el Linux/Unix directamente. ✓ Está orientada a objetos	✓ Software comercial. Por lo tanto restringido en algunas cosas. ✓ Reduce el tiempo de la aplicación a desarrollar ✓ Bibliotecas de desarrollo multiplataforma ✓ Plataforma independiente API ✓ Esta totalmente orientada a objetos. ✓ No emula sistemas operativos. Se compila en la plataforma en la cual se trabaje.

CAPÍTULO IV

METODOLOGIA PARA DESARROLLAR APLICACIONES MULTIPLATAFORMA

IV. Metodología para desarrollar aplicaciones multiplataforma.

En este trabajo veremos como caso particular una aplicación que se ejecute en los sistemas operativos Linux a Windows, considerando que el emulador de Linux llamado Cygwin puede instalarse en otras plataformas, de tal forma que se cumpla el objetivo de portabilidad.

4.1. Metodología para migrar una aplicación de Linux a Windows.

A partir de la necesidad de migrar una aplicación de Linux a Windows, se tiene que instalar y configurar el emulador de Linux bajo Windows, llamado Cygwin, el cual puede ser descargado de la página oficial del software (http://www.cygwin.com), dicho emulador es un ambiente Unix desarrollado por Red Hat para Windows el cual incluye varias bibliotecas necesarias para la migración de nuestra aplicación.

4.1.1. Descarga del emulador de Linux.

Como primer paso se debe de ingresar a la página http://www.cygwin.com/, después se da un click sobre el link "*Install Cygwin Now*". Ver figura 4.1.a

Figura 4.1

A continuación aparecerá un cuadro de diálogo el cual solicitara el destino para guardar el ejecutable en donde se guarda el programa de instalación. Ver figura 4.2.

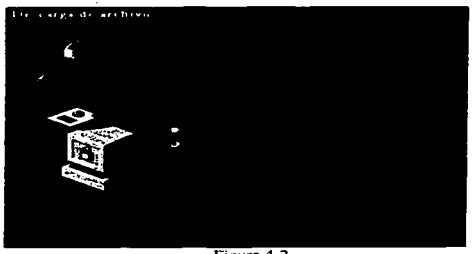


Figura 4.2

Posteriormente se presenta otro cuadro de dialogo, donde se selecciona la ruta donde se desea que se guarde el archivo ejecutable. Ver figura 4.3.

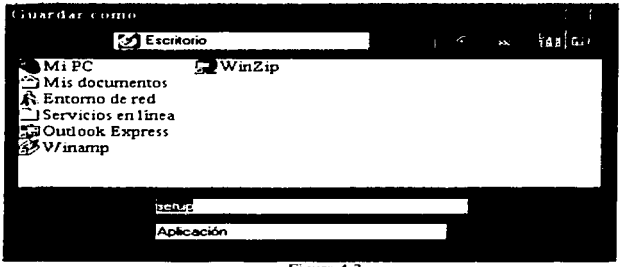


Figura 4.3

Enseguida presenta un cuadro dialogo donde muestra como se lleva acabo la descarga del archivo ejecutable. Ver figura 4.4.

Figura 4.4.

Después en el siguiente cuadro de dialogo, se presentan tres opciones, las cuales con:

- Instalarlo de Internet.
- Descargarlo de Internet

- Instalarlo desde un directorio local.

La primera opción se refiere cuando se instala desde Internet, y por lo tanto no existen archivos instalados en la máquina.

La segunda opción se refiere cuando se descarga por primera vez, y por lo tanto no existen archivos referentes a Cygwin, los cuales se necesitan descargar de Internet.

La tercera opción se refiere cuando ya se tienen los archivos previamente descargados y guardados en la máquina .

En este caso se utilizara la segunda opción ya que no se cuenta con ningún archivo descargado en la máquina y se requiere de ellos para la instalación. Ver en la figura 4.5.

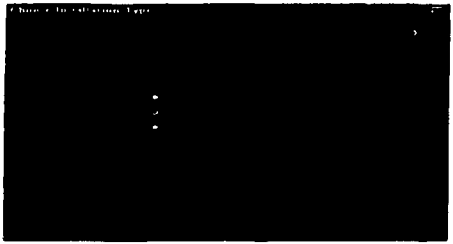


Figura 4.5.

4.1.2. Seleccionar el directorio para descargar los paquetes.

Posteriormente se le asigna la ruta del directorio en este caso C:\cyginstall en donde se descargan los paquetes y damos un click en "next" para continuar con el proceso, si no se desea ese directorio en particular el programa te da la opción desde *" el browser"* para que elijas el directorio que sea más conveniente para el usuario. Ver la figura 4.6.

Figura 4.6.

En el cuadro que sigue muestra tres opciones, las cuales son:

- ✓ Conexión directa
- ✓ Usando la configuración de Internet Explorer5.
- ✓ Usando una red local.

En este caso elegiremos la primera opción debido a que es mas rápida. Ver figura 4.7.

Figura 4.7.

Esta imagen muestra el progreso de la una conexión que se eligió en la imagen anterior. Ver figura 4.8.

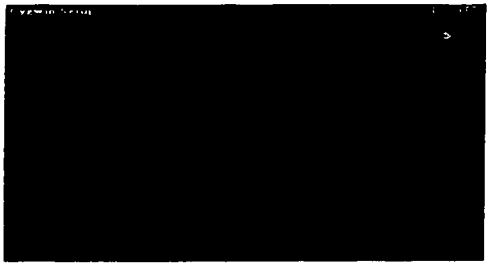


Figura 4.8.

4.1.3. Elegir el sitio para bajar el programa.

Enseguida aparece una ventana (ver figura 4.9) en la cual se puede elegir una dirección para bajar el programa, que sea conveniente, ya que algunas son más rápidas debido a que se encuentran en servidores cercanos, por lo cual elegimos:

ftp://mirrors.rcn.net.

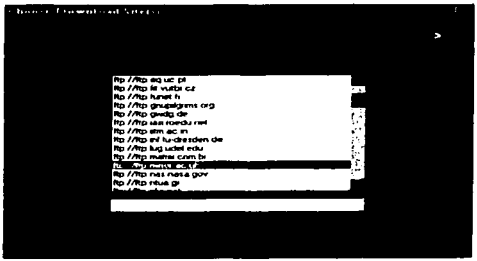


Figura 4.9.

4.1.4. Seleccionar los paquetes para descargarlos.

Después da la opción para elegir los paquetes que necesitamos descargar, es decir permite realizar en forma personalizada la instalación, dependiendo de las necesidades que se requieran, por lo tanto utilizaremos las herramientas para desarrolladores, e inicia el proceso para descargarlos desde la dirección que se elige en la ventana anterior. Ver figura 4.10.

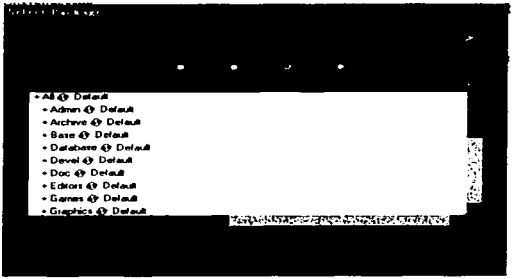


Figura 4.10.

Como se observa en esta figura una vez que se le de un click al signo de mas se despliegan una lista de los paquetes que se deseen bajar según las necesidades de cada usuario.

Basta con seleccionar el paquete deseado y automáticamente lo detecta colocando una flecha en el cuadro elegido.

Nota: es importante elegir solo las paquetes necesarios, ya que esto disminuirá en mucho el tiempo de descarga. Ver figura 4.11.

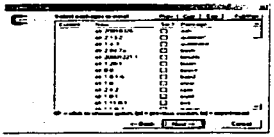


Figura 4.11.

4.1.5. Descarga de los paquetes.

A continuación inicia el proceso para descargar los paquetes que se van a utilizar y esperar que se finalice el proceso satisfactoriamente. Ver figura 4.12.

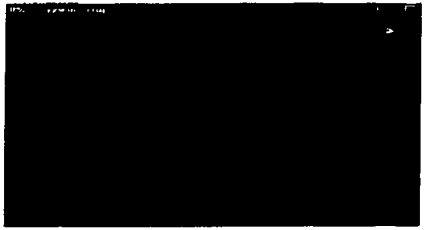


Figura 4.1 2

4.1.6. Descarga completa.

El proceso ha finalizado totalmente para continuar con la siguiente fase que es la instalación. Ver figura 4.13.

Figura 4.13.

4.2. Procedimiento para instalar el programa Cygwin.

Para instalarlo se necesita crear una carpeta con el nombre de Cygwin (el nombre es opcional), cuando termina de descargarlo nos crea el setup.exe, que es un icono, que al activarlo inicia el proceso de instalación. Ver figura 4.14

Figura 4.14

Una vez descargado el Cygwin se necesita instalarlo, lo primero que se debe hacer es darle un click al icono de instalación. Ver figura anterior.

Posteriormente aparece un cuadro de dialogo, el cual es la primera pantalla del proceso de instalación. Se selecciona la opción de siguiente y se continua con el proceso. Ver figura 4.15.

Figura 4.15

Como se muestra en la figura 4.16 el cuadro de dialogo es el mismo que en la figura 4.7. sólo que ahora la opción que se debe seleccionar es diferente, ya que en lugar de elegir la segunda opción la cual es descargarlo de Internet, se debe seleccionar la tercera, ya que los archivos ahora se encuentran en el disco duro.

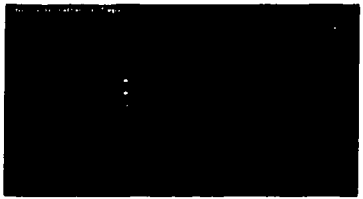


Figura 4.16.

Después de se selecciona el directorio donde se han descargado los paquetes que se encuentran en C:\cyginstall (como ya se mencionó el nombre es opcional), también se seleccionan las opciones "*all user*" y el sistema operativo que se va a instalar en este caso "*DOS*", a continuación se da click en "siguiente" Ver figura 4.17.

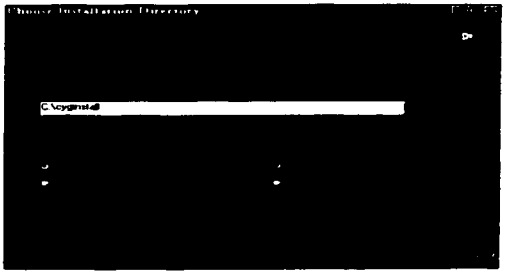


Figura 4.17.

En este cuadro de dialogo se asigna la ruta donde se encuentran los archivos que se descargaron por medio de Internet. Ver figura 4.18.

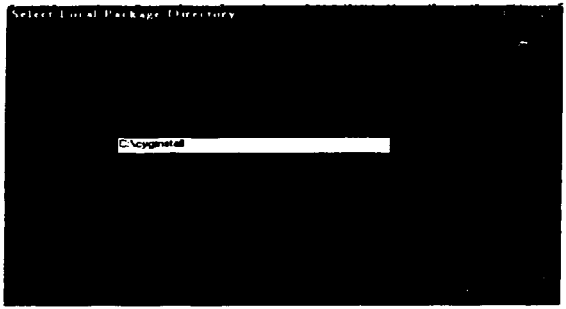


Figura 4.18.

4.2.1. Instalación de paquetes.

A continuación nos presenta nuevamente los paquetes que contiene el Cygwin para instalarlos, e inicia la instalación de cada uno de ellos los cuales deben ser descompactados y colocados de acuerdo a la ruta que tienen asignada. Ver figura 4.19.

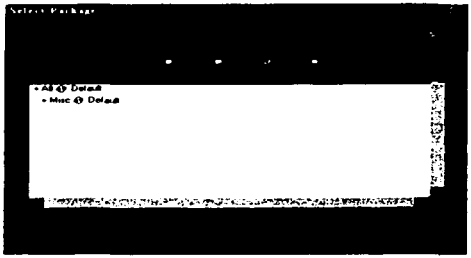


Figura 4.19.

Después se despliega la lista de los paquetes que se desean instalar dando un clic en el signo de mas si se desea ver a detalle esta lista, se da un clic en la opción *view* del menú del cuadro de dialogo. Ver figura 4.20.

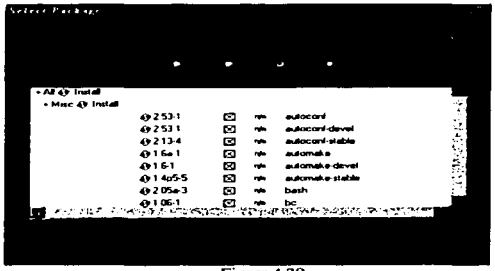


Figura 4.20

FALTA LAS PAGINAS

71 A 72

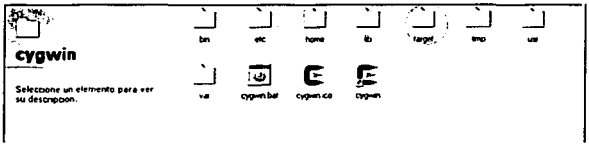


Figura 4.25.

Estas se encuentran compactadas, así que se deben descomprimir cada una de ellas generando varios archivos como se muestra a continuación: Ver figura 4.26.

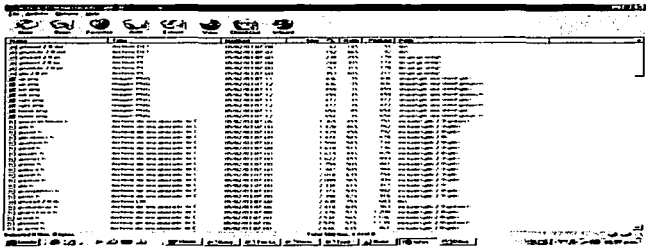


Figura 4.26

Para conocer la ruta de cada una de las bibliotecas se debe observar la dirección que muestra en el Path para colocarlas correctamente. Ver figura 4. 27.

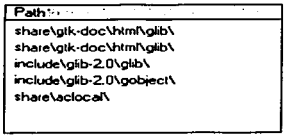


Figura 4.27.

Posteriormente se les asigna la dirección elegida a la cual van a ser descargados. Ver figura 4.28.

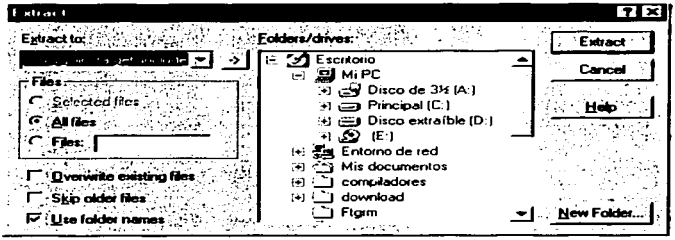


Figura 4.28.

Por lo tanto la carpeta target la cual se ha creado para este propósito queda de la siguiente manera. Ver figura 4.29.

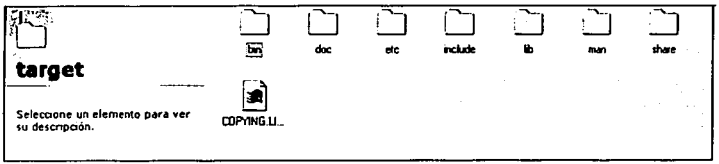


Figura 4.29.

4.2.3. Importancia de las bibliotecas GTK.

La biblioteca GTK también se emplea para desarrollar aplicaciones en otras plataformas en este caso utilizaremos la versión GTK+-2.0.

Para generar el GTK es necesario descargar primero Glib, ya que facilita la portabilidad hacia otras plataformas y permite que los tipos de datos cambien sin necesidad de rescribir la aplicación (se elige la opción al momento de descargar el emulador de Linux).

La biblioteca Glib es una colección de funciones comunes, ampliamente utilizadas por GTK.

Glib proporciona algunos tipos predefinidos para facilitar la portabilidad, simplificar y clarificar el código para mantener cierta consistencia.

Después de generar e instalar Glib, puede generar GTK.

GTK esta escrito en C; pero a pesar de ello, esta orientado a objetos y hay disponibles acoplamientos para utilizar GTK desde muchos lenguajes incluyendo C++.

Al igual que la mayor parte de las herramientas modernas de interfaces graficas de usuario, GTK+ es una herramienta conducida por sucesos.

Teniendo en cuenta que el Cygwin ya tiene instalado las bibliotecas Gtk y Glib, necesitamos bajar las demás bibliotecas que complementan y actualizan a Gtk.

4.2.4. Descargar e instalar las bibliotecas GTK.

Para cumplir el objetivo establecido se necesitan agregar las bibliotecas GTK, por lo tanto necesitamos descargarlas de la siguiente dirección:

http://www.gtk.org/.

El GTK es una biblioteca orientada a objetos escrita en C que puede utilizarse en aplicaciones en diversos lenguajes.

El archivo GTK+-2.0.tar.gz es un archivo comprimido debido a las extensiones tar y gzip.

Primero será necesario descompactar el archivo utilizando Winzip. El archivo descompactado será ahora un archivo de tipo tar.

Para construir una aplicación GTK, el código fuente debe incluir la biblioteca gtk/gtk.h, la cual es la biblioteca principal del propio Gtk.

Para ejecutar la aplicación se utilizará el compilador Gcc, el cual es una colección de compiladores libres para C, C++, Objective C y otros lenguajes.

4.2.5. Ejecución de banderas.

Las banderas se encuentran en la ruta C: \cygwin \usr\include\lib\pkgconfig. Éstas son indicadores de comandos permite mostrar los parámetros exactos que se pasan al compilador gcc.

Por lo tanto se anota en la consola del emulador:

$ pkg-config -- cflags gtk+-2.0

Lo cual da como resultado lo siguiente:

-I/usr/include/gtk-2.0 -I/usr/lib/gtk-2.0/include - I/usr/include/glib-2.0

- I/usr/lib/glib-2.0/include - I/usr/include/pango-1.0 - I/usr/X11R6/include

-I/usr /include/freetype2 -I/usr/include/atk-1.0

Y después se ejecuta la siguiente bandera para pkg-config --libs, la instrucción es la que se muestra a continuación:

$pkg-config --libs gtk +-2.0

-L/usr/lib –L/usr/X11R6/lib –lgtk-x11-2.0 –lgdk-x11-2.0 – lXi –lgdk

Si ambas se ejecutan correctamente es decir que despliegue los parametros anteriores, esto significa que la instalación está completa y que Cygwin esta listo para ejecutar una aplicación.

FALTA

PAGINA

77

V. Caso práctico

5.1. Antecedentes.

Como caso práctico para la demostración del Objetivo del presente trabajo de tesis, se realizará la migración de *Material 3D* (software diseñado para obtener el grado de Maestría por el Ing. Marcelo Pérez Medel) de Linux a Windows.

Material 3D, es una aplicación programada bajo Linux en Lenguaje C, auxiliado de bibliotecas GTK. Dicha aplicación sirve para el modelado y diseño de objetos en tres dimensiones.

A continuación describiremos brevemente algunas características y comportamiento del programa en su ambiente de origen (Linux).

La figura 5.1 muestra el área de trabajo de *Material 3D*. La cual, como en la mayoría de aplicaciones en modo gráfico, tiene un menú y una barra de herramientas en la parte superior de la ventana como ayuda para realizar diversas funciones.

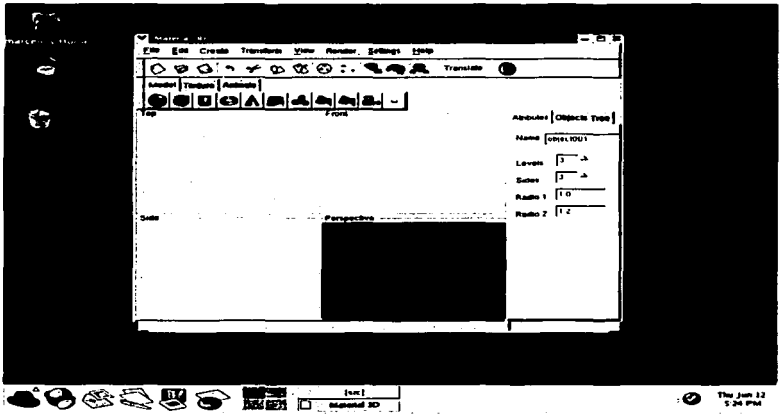


Figura 5.1.

CAPÍTULO V
CASO PRÁCTICO

Entre las funciones que realiza la aplicación, está la de generar una imagen, esto es, a partir de un objeto ya modelado, presionamos el botón "generar imagen".

En la figura 5.1 a) podemos observar un ejemplo del modelado de un objeto. La interfaz del software nos ayuda a ver el objeto en diversas perspectivas (frontal, lateral y superior).

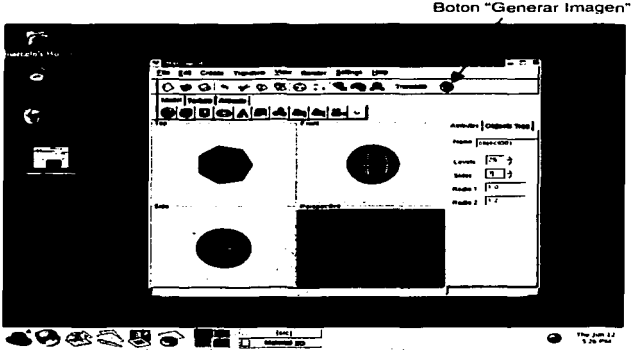


Figura 5.1a).

La imagen resultante de dicho modelado tiene el siguiente aspecto: (Fig. 5.1b)

Figura 5.1b)

Para que una aplicación, desarrollada bajo Linux sea portable (al otro sistema operativo) debe hacer uso de la biblioteca Gtk+, ya que por el momento las bibliotecas GNOME no dan ésta posibilidad. Si desarrollamos un programa en Glade, podemos agregar o utilizar la biblioteca Gtk+ de dos formas: La primer opción es seleccionar la biblioteca Gtk+ al crear un nuevo proyecto en Glade (Fig. 5.2). La segunda forma de agregar la biblioteca es en forma manual, es decir, programando la utilización de las funciones Gtk+.

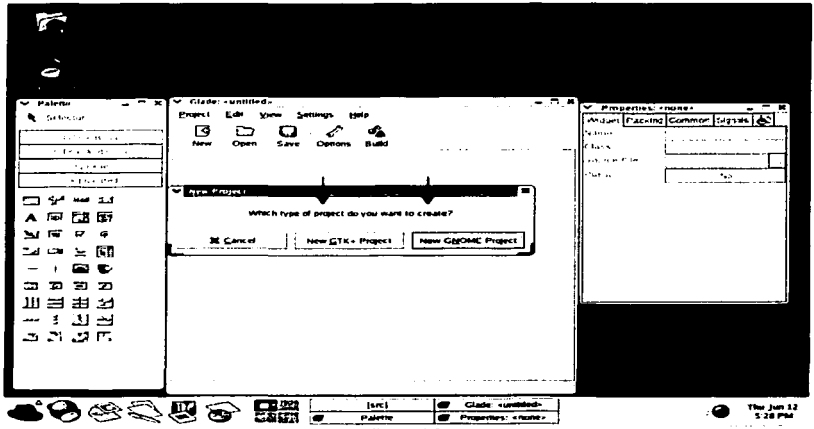


Figura 5.2.

5.2. Metodología.

Una vez teniendo lista nuestra aplicación a migrar (con funciones Gtk+), debemos localizar todos los archivos que hacen que ésta funcione correctamente (bajo Linux). A continuación mostramos los archivos de *Material 3D* (Fig. 5.3)

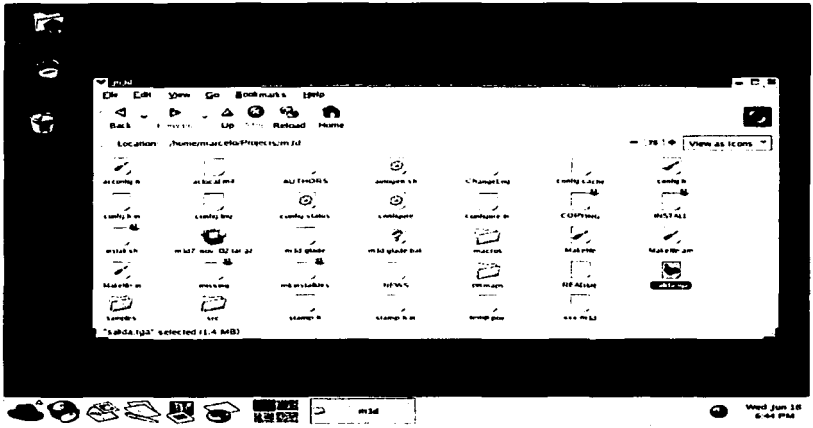


Figura 5.3

Cada carpeta tiene una función: por ejemplo la carpeta[1] Pixmaps, contiene los archivos para el diseño de iconos y la carpeta SRC, contiene los archivos fuentes *.c y *.h, (ver figura 5.3.) de los cuales se hablara con detalle más adelante.

Una vez que la aplicación funcione correctamente en Linux, y se tengan todos los directorios y archivos correspondientes, debe seguir el siguiente procedimiento, para que pueda hacer lo mismo bajo Windows.

5.2.1. Transferencia de archivos de un sistema operativo a otro.

Paso 1) Estas carpetas y archivos se transfieren en la unidad de almacenamiento portátil,(floppy) para poder manipular la información en el sistema operativo diferente en el cual se programó la aplicación.

Paso 2) Se busca la carpeta del Cygwin, en donde se encuentra instalado.

[1] Carpeta o directorio se manejan por igual en este trabajo.

Paso 3) Se abre la carpeta correspondiente al Cygwin, donde se encuentra previamente instalado en la máquina. Ver Figura 5.4 .

Figura 5.4.

Paso 4) Una vez dentro de la carpeta principal del Cygwin se abre una subcarpeta llamada home, (ver figura 5.5), donde se guardan los programas y aplicaciones hechas por los diferentes usuarios, este sistema es muy parecido al que utiliza Linux, y como Cygwin trabaja exactamente bajo ese mismo ambiente, por lo tanto, esté posee un árbol de directorios parecido.

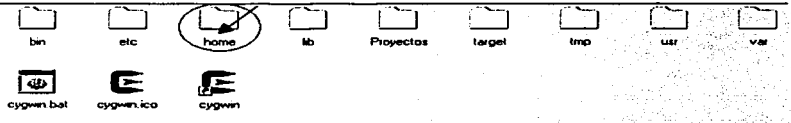


Figura 5.5.

Paso 5) Dentro del directorio home se crea una subcarpeta donde se trasfieren todos los archivos y directorios de la aplicación a migrar. En este caso, la carpeta se llama Proyectos. Ver figura 5.6.

Figura 5.6.

Nota: aunque el Cygwin no hace diferencia entre mayúsculas y minúsculas se respetará el nombre para no perder la costumbre cuando se utilice en Linux.

5.2.2. En caso de que los archivos se encuentren compactados.

En el caso de que los archivos están compactados, Ver figura 5.7.

Figura 5.7.

Paso1) Se utiliza el programa que tiene instalado Windows, que se llama Winzip, para descompactar los archivos. Ver figura 5.8.

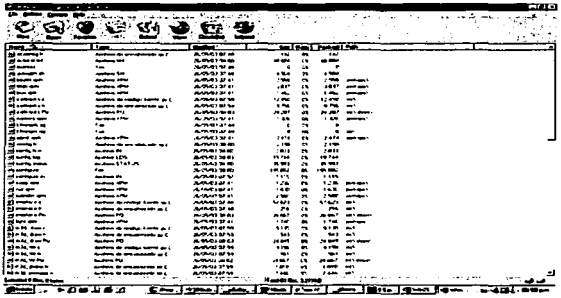


Figura 5.8.

Paso 2) Después nos da la opción para extraer los archivos. Ver figura 5.9.

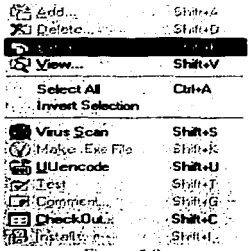


Figura 5.9.

Paso 3) Finalmente nos da la opción para asignar la ruta, en las cuales se van copiando en sus respectivas carpetas. Ver figura 5.10.

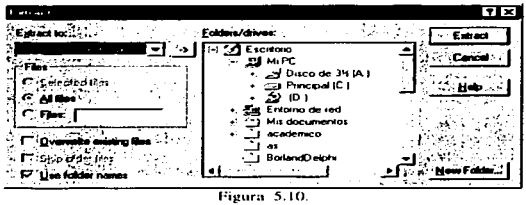


Figura 5.10.

5.2.3. Ejecución de Cygwin.

Paso 4) Se ejecuta el Cygwin, presionando el icono correspondiente o buscando en el menú de inicio de Windows el nombre del programa. Ver figura 5.11.

Figura 5.11.

Paso 5) Una vez hecho lo anterior se despliega la ventana del Cygwin, en la consola de Win32, como se muestra en la figura 5.12.,que es un ambiente totalmente parecido a la consola de Linux.

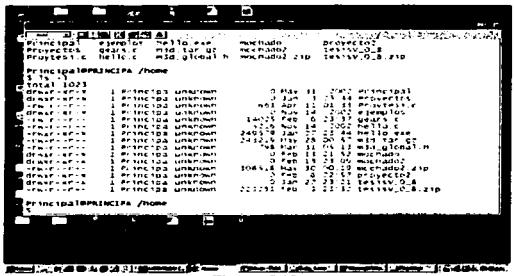


Figura 5.12.

5.2.4. Transferencia de archivos en Cygwin.

Paso 8) En la carpeta home, donde se creó una subcarpeta llamada Proyectos, que a su vez se encuentra dentro de ella la carpeta SRC, contiene los archivos fuentes de la aplicación. Ver figura 5.13.

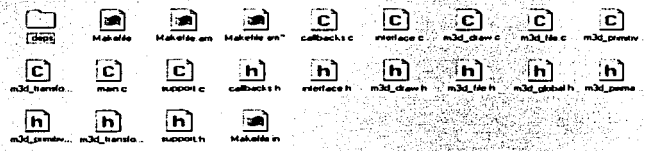


Figura 5.13.

La manera que nos muestra el Cygwin, los archivos que se encuentran en la carpeta SRC, se visualizan de la siguiente forma, esto es con el fin de familiarizarnos con cualquiera de los dos entornos. Ver la figura 5.13a) .

```
Principal@PRINCIPA /Proyectos/src
$ ls
Makefile        callbacks.o  m3d_draw.o        m3d_primitives.h  pixmaps
Makefile.am     interface.c  m3d_file.c        m3d_primitives.o  support.c
Makefile.am~    interface.h  m3d_file.h        m3d_transform.c   support.h
Makefile.in     interface.o  m3d_file.o        m3d_transform.h   support.o
SALIDA.TGA      m3d.exe      m3d_global.h      m3d_transform.o   temp.pov
callbacks.c     m3d_draw.c   m3d_pixmaps.h     main.c
callbacks.h     m3d_draw.h   m3d_primitives.c  main.o
```

Figura 5.13a).

Para poder ejecutar la aplicación se debe ingresar a la ruta correspondiente donde se encuentran los archivos ejecutables. La cual será la siguiente para este caso: home/Proyectos/src. Ver Figura 5.14

```
Principal@PRINCIPA /Proyectos/src
$ _
```

Figura 5.14

5.2.5. Algunas instrucciones para compilar y ejecutar programas.

A continuación se explicará como funciona cada una de las instrucciones que se utilizarán más adelante.

- ✓ Para compilar se utiliza el nombre del compilador que es gcc –c y el nombre del programa con la extensión .c

  gcc –c main.c (Salida: main.o)

- ✓ Para crear el ejecutable utilizamos nuevamente el nombre del compilador gcc y el nombre del programa con la extensión .o –o y el nombre de salida del programa

  gcc main.o –o programa

- ✓ Para ejecutarlo únicamente utilizamos ./ y el nombre del programa.

  ./programa

- ✓ Los pasos 1 y 2 se pueden integrar en un solo paso, con el nombre del compilador gcc, el nombre del programa.c –o y el nombre de salida del programa.

  gcc main.c –o programa

Debido a que el compilador nos muestra advertencias en lo que se refiere al código, sin embargo dicha información nos puede ocultar algunas fuentes de error en tiempo de ejecución, o bien errores que podrían manifestarse al migrar nuestro código en otras plataformas.

Para evitar errores de manera anticipada, necesitamos utilizar banderas las cuales nos van a servir para corregirlos, y en este caso utilizaremos las siguientes:

- ✓ Se usa la bandera –Wall donde –W especifica los tipos de advertencias que nos interesan y all muestra todas en general.

  gcc –Wall main.c –o programa

Las banderas de optimización permiten un código más compacto y/o más veloz.

La optimización suele ser conservativa (hasta cierto punto), manteniendo el comportamiento original del programa previo a este paso.

- ✓ Se usa la bandera –O seguido de un número que indica el nivel de optimización

  gcc –O2 main.c –o programa

5.2.6. Instrucciones para ejecutar y compilar varios programas.

Hay que estructurar el código lo máximo posible, usando distintos archivos fuentes.

- ✓ Para compilarlos y enlazarlos, se usa la siguiente instrucción:

gcc main.c rutinas.c protocolos.c –o programa[2]

5.2.7. Compilando un programa en Cygwin.

Para hacer la compilación de la aplicación se va utilizar el compilador Gcc, el cual es una colección de compiladores libres para C, C++, Objective C entre otros lenguajes.

La compilación del programa hola utilizando gcc en Cygwin, es de la siguiente manera:

gcc – wall –g hola.c -o hola –fnative -struct `pkg –config --cflags --libs gtk+-2.0´

5.2.8. Instrucciones para compilar un programa en Linux.

Cuando compilamos la aplicación en Linux se necesitan las siguientes instrucciones:

gcc –Wall -g hola.c -o hola ` pkg – config --cflags gtk+-2.0´ `pkg-config --libs gtk+-2.0´

Para compilarlo y crear el ejecutable nos queda de la siguiente manera:

gcc hola.c –o hola

$ ./HOLA.

Y cuando ejecutamos el programa hola.c se ve así en la figura 5.15.

Figura 5.15

[2] Nota: No importa el orden en el cual se coloquen los diversos programas.

Paso 9) Una vez que se encuentra en la ruta correcta, se escriben las siguientes instrucciones de compilación, que sólo son válidas para Cygwin, ya que para compilar en Linux existen algunos cambios que veremos posteriormente.

```
S gcc -Wall -g programa. -o archivo `pkg-config --cflags gtk+-2.0` -mno-cygwin -fnati
ve-struct `pkg-config --libs gtk+-2.0`
```

Lo cual se vería así en Cygwin. Ver Figura 5.16.

```
Principal@PRINCIPA /Proyectos/src
$ gcc -Wall -g programa.c archivo -o `pkg-config --cflags gtk+-2.0` -mno-cygwin
 -fnative-struct `pkg-config --libs gtk+-2.0`
```

Figura 5.16.

La instrucción anterior se utiliza cuando se compila una aplicación sencilla de un solo programa y con ello se obtiene un ejecutable, pero cuando la aplicación es más compleja, posee varios archivos como es el caso de esta aplicación. En la carpeta SRC se tienen los archivos extensión .c y .h, (interface.c, interface.h, callbacks.c, callbacks.h, etc.). Ver figura 5.17.

```
Principal@PRINCIPA /home/Proyectos/src
$ ls
Makefile        callbacks.h  m3d_file.c        m3d_primitives.h  support.h
Makefile.am     interface.c  m3d_file.h        m3d_transform.c
Makefile.am~    interface.h  m3d_global.h      m3d_transform.h
Makefile.in     m3d_draw.c   m3d_pixmaps.h     main.c
callbacks.c     m3d_draw.h   m3d_primitives.c  support.c
```

Figura 5.17.

Como se sabe los archivos con extensión .c son los que poseen el código fuente, en los cuales se pueden hacer correcciones en el momento que marquen un error, mientras que los archivos .h, contienen las bibliotecas a utilizar. Pero lo que se necesita en este caso son los .o, que son el intermedio entre el ejecutable y el código fuente. Estos se compilan uno por uno, utilizando la siguiente instrucción. Ver figura 5.18.

```
Principal@PRINCIPA /home/Proyectos/src
$ gcc -Wall -g m3d_transform.c -c `pkg-config --cflags gtk+-2.0` -mno-cygwin -f
native-struct `pkg-config --libs gtk+-2.0`
```

Figura 5.18.

Esta instrucción se repite nuevamente con cada programa que tiene la extensión .c que se encuentre en la aplicación. Ver Figura 5.19.

```
Principal@PRINCIPA /home/Proyectos/src
$ gcc -Wall -g m3d_file.c -c `pkg-config --cflags gtk+-2.0` -mno-cygwin -fnativ
e-struct `pkg-config --libs gtk+-2.0`
```

Figura 5.19.

Si se realiza en forma correcta cada una de las instrucciones para cada uno de los programas, cuando se compila el archivo objeto, en la pantalla del Cygwin se muestra de la siguiente manera. Ver figura 5.20.

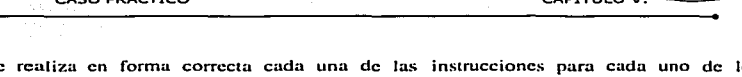


Figura 5.20

Como se muestra en figura 5.16 y la figura 5.18 la diferencia esta en cambiar la -c en lugar de -o, para obtener los programas objetos, los cuales nos serán de gran utilidad posteriormente.

Una vez que se obtengan los archivos objetos (es decir los archivos con extensión .o) y todos los archivos .c (código fuente) que se tienen en la aplicación.Ver figura 5.21.

```
Principal@PRINCIPA /home/Proyectos/src
$ ls -l *.o
-rw-r--r--    1 Principa unknown    243607 Jun  5 21:50 callbacks.o
-rw-r--r--    1 Principa unknown    298500 Jun  5 21:52 interface.o
-rw-r--r--    1 Principa unknown    240198 May 29 23:02 m3d_draw.o
-rw-r--r--    1 Principa unknown    230272 Jun  5 21:32 m3d_file.o
-rw-r--r--    1 Principa unknown    261764 Jun  5 21:50 m3d_primitives.o
-rw-r--r--    1 Principa unknown    228081 Jun  5 21:42 m3d_transform.o
-rw-r--r--    1 Principa unknown    226676 May 29 22:48 main.o
-rw-r--r--    1 Principa unknown    233560 Jun  5 21:51 support.o
```

Figura 5.21

5.2.9. Ejecución de varios programas en Cygwin.

Cuando se tienen varios programas, se enlazan una vez que se tengan todos los programas objeto o extensión .o. Para esto serán enlazados con la misma instrucción, donde cambia nuevamente la letra -c por la vocal -o .Ver figura 5.22

```
Principal@PRINCIPA /home/Proyectos/src
$ gcc -Wall -o m3d m3d_draw.o interface.o m3d_file.o m3d_transform.o callbac
ks.o main.o support.o m3d_primitives.o `pkg-config --cflags gtk+-2.0` -mno-cygw
in -fnative-struct `pkg-config --libs gtk+-2.0`

Principal@PRINCIPA /home/Proyectos/src
$
```

Figura 5.22.

Para que todos se enlacen en uno solo archivo ejecutable, se le asigna un nombre, con el cual se va a ejecutar junto con los demás programas, en este caso se llama **m3d**. Ver Figura 5.23.

```
Principal@PRINCIPA /home/Proyectos/src
$ gcc -Wall -o m3d/m3d m3d_draw.o interface.o m3d_file.o m3d_transform.o callbac
ks.o main.o support.o m3d_primitives.o `pkg-config --cflags gtk+-2.0` -mno-cygw
in -fnative-struct `pkg-config --libs gtk+-2.0`

Principal@PRINCIPA /home/Proyectos/src
$
```

Figura 5.23.

Esto se hace con un programa ligador que busca todos los componentes necesarios para crear el archivo que buscamos, es decir el ejecutable de todos los archivo que se tienen. Una vez ligados los archivos .o, el ejecutable se crea en la carpeta m3d para este caso.

Para terminar se ejecuta se la instrucción siguiente : /m3d. Lo que se vería de la siguiente forma en la consola del Cygwin. Ver figura 5.24.

```
Principal@PRINCIPA /home/Proyectos/src
$ ./m3d_
```

Figura 5.24.

5.2.10. Despliegue de la aplicación.

Una vez hecho lo anterior, se ejecuta la aplicación automáticamente. Y ésta se despliega como se muestra a continuación. Ver figura 5.25.

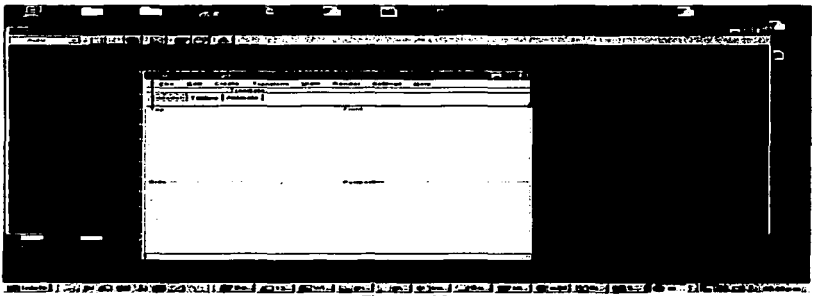


Figura 5.25.

En esta figura se observa, que la aplicación no esta del todo completa, debido a que no se muestran los iconos al momento de ejecutarse, para realizar las diferentes tareas con respecto a las imágenes que se van a trabajar, el cual dificulta su utilización para las imágenes en tres dimensiones, por lo tanto necesita que la carpeta pixmaps visualice los iconos que contiene. Ver figura 5.26.

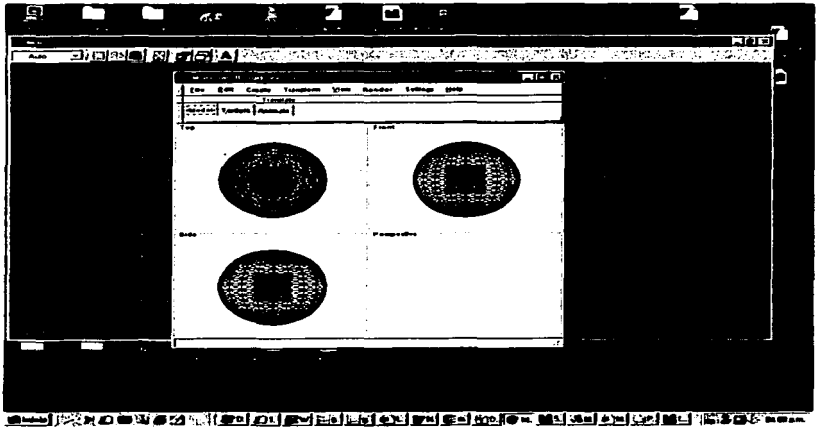


Figura 5.26.

5.2.11. Corrección de errores de la aplicación.

Al momento de compilar la aplicación y que ésta no funciona de manera correcta quiere decir que está teniendo conflicto con alguna instrucción o bien con alguna biblioteca, en este caso nosotros debemos depurar nuestro programa para corregir los errores que puedan suceder nuevamente esto es con la finalidad de evitar con anticipación los posibles errores que puedan ocurrir en la aplicación.

En este caso en particular para resolver el problema de la visualización de los iconos, se realizaron algunas correcciones, se cambio de lugar la carpeta pixmaps, para que así encontrara los archivos que necesitaba desde donde el emulador los estaba buscando. Ver figura 5.27.

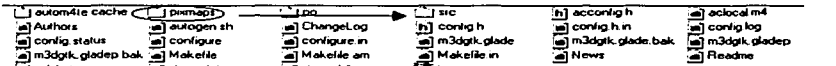


Figura 5.27.

Se movió del lugar donde se encontraba (carpeta Proyectos) a la carpeta src, y al realizar esto los iconos fueron visualizados en la aplicación al ejecutarse. Ver figura 5.28.

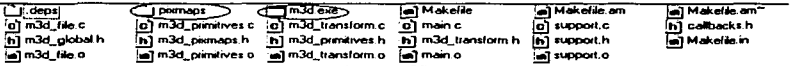


Figura 5.28.

Como se muestra en esta figura 5.28, la carpeta src no ha cambiado mucho, solo ha creado el archivo ejecutable y ahora la carpeta pixmaps esta dentro de ella.

Con este cambio se observara que la aplicación se encuentra en forma completa, ya que los iconos se muestran completamente, para así, utilizarla sin problemas, ya que sólo su diseñador, sabría donde se encuentra cada una de las herramientas sino se muestran los iconos correspondientes. Ver figura 5.29.

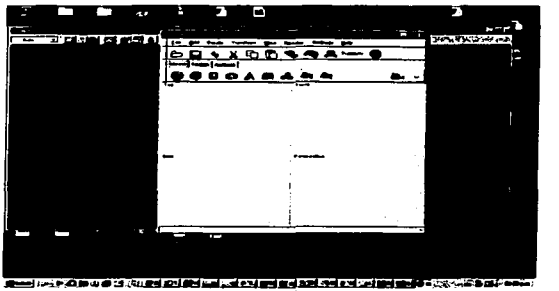


Figura 5.29.

Como se muestra en la imagen la aplicación esta casi completa y se ve el cambio cuando presenta los iconos, debido a que cada icono muestra la función que realiza cada uno de ellos y que anteriormente se encontraban en blanco. Ver figura 5.30.

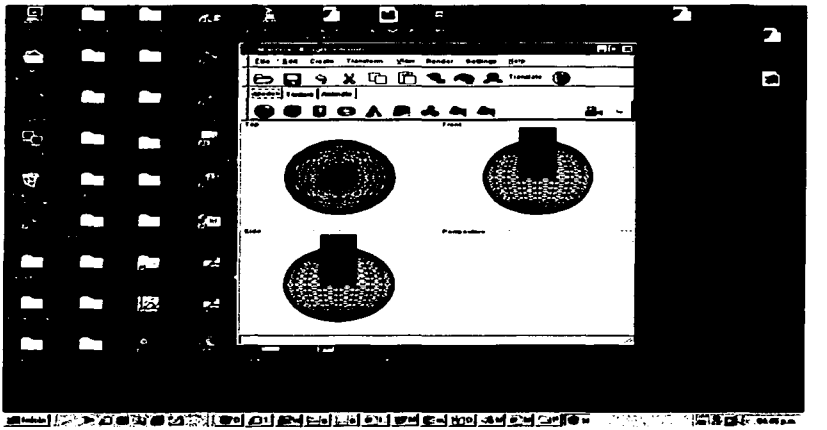


Figura 5.30.

Ahora sólo falta una cosa, que proporcione una imagen de salida al momento de presionar el icono que sirve para generar la imagen que se ha creado.

5.2.12. Generación de la imagen creada en pantalla.

Una vez qe se ha generado la imagen, esta crea dos archivos de salida. (Como se muestra en la figura 5.30.) Ambos archivos se crean en la carpeta SCR, uno llamado **temp.pov,** el cual se crea en el programa de diseño en tres dimensiones PovRay[3], en este se modelan y

[3] PovRay: Es una herramienta para crear gráficos tridimencionales. para plataformas Mac, Linux, Windows, etc. http://www.povray.org/

animan objetos en tres dimensiones y el otro llamado salida.tga el cual se puede exhibir en el programa LviewPro[4] el cual funciona para modelar, desplegar y filtrar imágenes (que puede cambiar de un formato a otro, ejemplo, de *.gif a *.bmp, o a *.jpg, etc.). Ver figura 5.31.

```
Principal@PRINCIPA /home/Proyectos/src
$ ls
Makefile        callbacks.o   m3d_draw.o          m3d_primitives.h   pixmaps
Makefile.am     interface.c   m3d_file.c          m3d_primitives.o   support.c
Makefile.am~    interface.h   m3d_file.h          m3d_transform.c    support.h
Makefile.in     interface.o   m3d_file.o          m3d_transform.h    support.o
SALIDA.TGA      m3d.exe       m3d_global.h        m3d_transform.o    temp.pov
callbacks.c     m3d_draw.c    m3d_pixmaps.h       main.c
callbacks.h     m3d_draw.h    m3d_primitives.c    main.o
```

Figura 5.31.

Al ejecutar estos dos archivos, el primero ejecuta la imagen para ser visualizada. Ver figura 5.32, el segundo la presenta paso a paso al momento de generar la imagen, en PovRay. Ver figura 5.32a.

Figura 5.32.

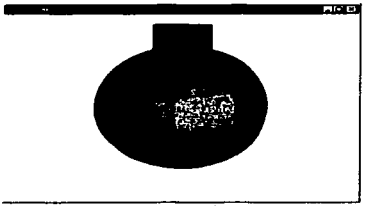


Figura 5.32a.

[4] LviewPro: Programa para diseñar, visualizar y cambiarle el formato a las imágenes. http://www.lview.com/index1024.htm

Con esto se da por concluida la aplicación, ya que se demostró el Cygwin funciono completamente para transportar una aplicación en un sistema operativo totalmente diferente y como el objetivo planteado sin muchos cambios en el código fuente.

CAPÍTULO VI

CONCLUSIONES

VI CONCLUSIONES.

Como se ha visto las aplicaciones multiplataforma, son de gran utilidad debido a que en estos tiempos ayudan al programador a reutilizar código de los programas y esto resulta más sencillo, modificarlos en lugar de reescribrilos en su totalidad, es decir, mientras exista menor número de características particulares en el programa, se tiene menos probabilidad de que éste presente fallas a futuro y sea más fácil que se adapte a las circunstancias para las que sea utilizado, con el fin de que las programas se ejecuten con un mínimo de cambios en cualquier plataforma.

Para escribir código portable, se necesita programar en un lenguaje de alto nivel y un grado superior de portabilidad dentro del estándar oficial, como es el caso del lenguaje C, en el cual se puede acceder a su código fuente y a su vez utilizando bibliotecas GTK que lo hacen aun más portable. Con la ayuda del compilador GCC se pueden procesar los programas fuentes de cualquiera de los lenguajes como el propio C, que es lo que interesa en este trabajo.

Otra herramienta a utilizar en este trabajo es el emulador Cygwin, debido a la necesidad de ejecutar una aplicación en dos plataformas en este caso Linux a Windows, para este trabajo se porto una aplicación de material 3d, que sirve para modelar y diseñar objetos en tres dimensiones.

Tomando en cuenta que se requiere utilizar dos entornos en un solo sistema operativo y con ello ejecutar aplicaciones para que puedan ser implementadas en cualquiera de los dos sistemas, empleando software libre, lo cual tiene muchas ventajas, algunas de ellas son:

- ✓ Permite el acceso a su código fuente, y que se realicen de esta manera modificaciones a los programas.
- ✓ Con el punto anterior se tiene la seguridad de permanencia de la aplicación, ya que si se utiliza software comercial, y a la empresa que lo desarrolla desaparece (por diversas razones) con ésta, también desaparecen los fuentes de la aplicación. Se debe de programar la aplicación nuevamente con otro software o simplemente olvidarse de ésta por siempre.
- ✓ Se puede modificar, y mejor aun, dejar que más usuarios la perfeccionen. Con la seguridad de la libre distribución.
- ✓ Se pueden obtener beneficios con la aplicación de todo tipo sin que por eso pierda la licencia que la hace libre.

El objetivo principal de este trabajo fue cumplido satisfactoriamente ya que se demostró que es posible migrar una aplicación que fue diseñada especialmente para el sistema operativo Linux, que a su vez se logro ejecutar en Windows, con cambios mínimos en su código fuente, y debido a la investigación que se realizó en este trabajo, podemos darnos cuenta que se puede llevar a cabo de una forma fácil y práctica.

Cygwin funciono de forma correcta demostrando ser un emulador que lo mismo puede soportar aplicaciones sencillas, es decir sin interfaz gráfica, que aplicaciones completas y con todo un entorno gráfico complejo, considerando que la aplicación que se porto modela y anima objetos 3d.

GLOSARIO

Glosario:

ANSI/ISO
(American Nacional Standards Institute / Internacional Organization for Standardization)
Instituto Nacional de Estándares Americanos / Organización de Normas Internacionales
.

API
Application Program Interface (Interfaz para programas de aplicación). Conjunto de convenciones de programación que definen cómo se invoca un servicio desde un programa.

ATK
Accessibility Toolkit (Conjunto de herramientas de accesibilidad). Permite el desarrollo de aplicaciones para personas discapacidades físicas.

DLL
Dynamic Link Library ("Biblioteca de vínculos dinámicos") es un archivo que contiene funciones que se pueden llamar desde aplicaciones u otras Dll. Los desarrolladores utilizan las Dll para poder reciclar el código y aislar las diferentes tareas. Las Dll no pueden ejecutarse directamente, es necesario llamarlas desde un código externo.

EMACS
Editor de texto. Aunque es su principal función, Emacs es hoy en dia un programa muy extenso y con muchas utilidades, gracias a su soporte de plug-ins en lenguaje LISP. Desde Emacs podrás contestar el correo, leer las noticias de USENET, compilar programas, jugar al tetrix... Requiere un periodo de aprendizaje largo.

FAQ
(Frecuent Ask Question) Lista de preguntas frecuente. Página donde los especialista en la materia hacen una lista de las posibles dudas que los usuarios pudieran tener con respecto a su aplicación.

FSF
Free Software Foundation. Fundación que pretende el desarrollo de un sistema operativo libre tipo UNIX. Fundada por Richard Stallman, empezó creando las herramientas necesarias para su propósito, de modo que no tuviera que depender de ninguna compañía comercial. Después vino la creación del núcleo, que todavía se encuentra en desarrollo.

GLIB
Es una biblioteca de funciones para programación en C, que contiene multitud de utilidades para facilitar la programación en este lenguaje.

GNU
Gnu is Not Unix. Proyecto de la FSF para crear un sistema UNIX libre.

GPL
General Public License. Una de las mejores aportaciones de la FSF. Es una licencia que protege la creación y distribución de software libre.

GTK
Es una herramienta multiplataforma para crear interfaces gráficas de usuario ofreciendo un completo paquete de widgets, también es adecuado para crear desde pequeños proyectos, hasta aplicaciones completas.

GUI
Interfaz Graphics User (Interfaz grafica de usuario).

KDE
K Desktop Environment. Entorno de escritorio que integra gestor de ventanas propio y una barra de tareas y que al igual que GNOME permite la interacción entre sus aplicaciones. Programado en C++ y con la base de librerías QT+ ha sido victima de criticas por parte de la comunidad GNU/Linux, ya que estas bibliotecas eran propiedad de una empresa comercial.

KERNEL
Es el software que constituye el núcleo del sistema operativo, dónde se realizan las funcionalidades básicas como la gestión de procesos, la gestión de memoria y de entrada salida.

MOTIF
Biblioteca de funciones para el desarrollo de aplicaciones gráficas.

PANGO
Es un completo sistema para la representación de texto en diversos alfabetos.

PATH
Variable del entorno, cuyo valor contiene los directorios donde el sistema buscara cuando intente encontrar un comando o aplicación.

PLATAFORMA
Es un término de carácter genérico que designa normalmente una arquitectura de hardware, aunque también se usa a veces para sistemas operativos o para el conjunto de ambos.

POSIX
Portable Operating System Interface Es un estándar con una serie de normas definidas para permitir la portabilidad entre diferentes sistemas UNIX. GNU/Linux cumple con este estándar.

SEÑALES
Las señales son eventos que se hacen llegar a un proceso en ejecución para su tratamiento por este. Las señales las podemos mandar nosotros u otros programas a otros programas. Tienen diferentes valores, y en función a esos valores el proceso que las recibe actúa de una manera u otra.

SISTEMA OPERATIVO
Es un intermediario entre los programas de usuario y el hardware, es decir el software encargado de proporcionar el entorno necesario dónde será posible ejecutar otros programas.

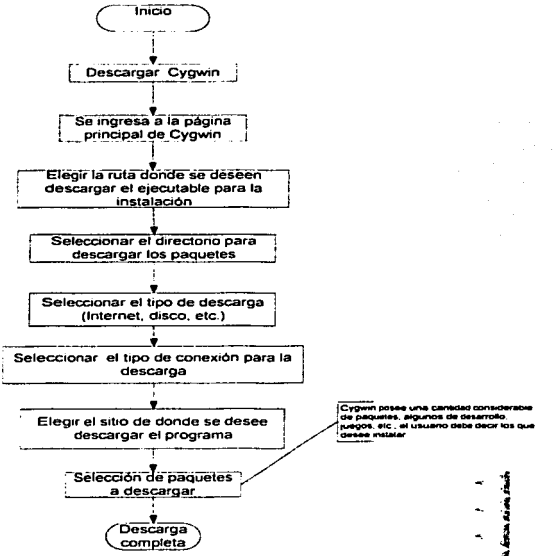
WIDGETS
Son elementos de la interfaz grafica de usuarios, como pueden ser, menús desplegables, iconos, barras de desplazamiento, ventanas, etc.

X11R6
Ultima versión utilizada del sistema de ventanas Xwindow.

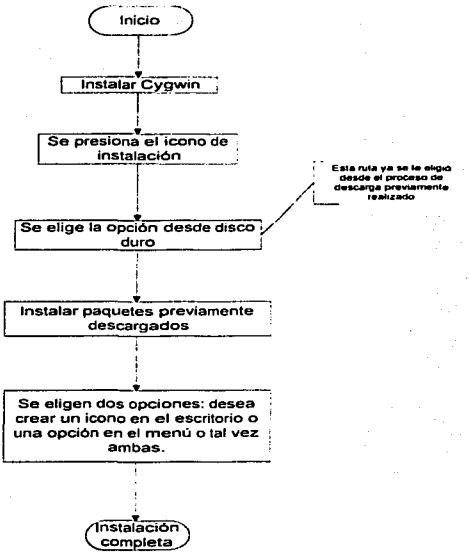
ANEXO

Anexo: Diagramas de la metodología que se lleva acabo en la ejecución de una aplicación.

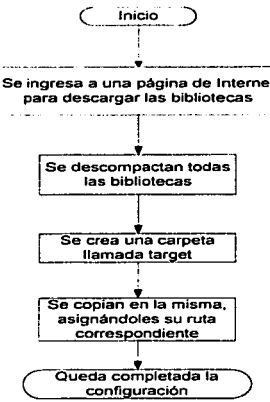
Descarga de Cygwin.



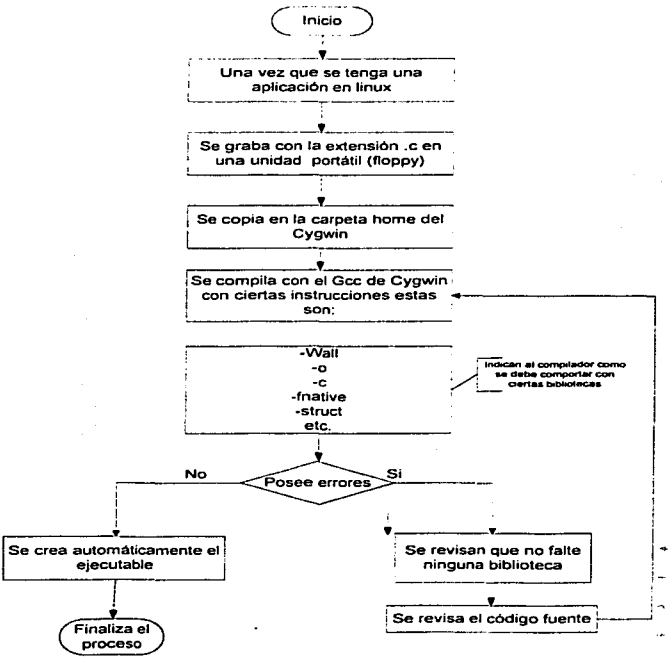
Instalación de Cygwin.



Descarga e instalación de bibliotecas



Ejecución de una aplicación.



BIBLIOGRAFÍA

BIBLIOGRAFÍA

Tenenbaum S. Andrew.
"Sistemas Operativos: Diseño e implementación".
Traduce Juan Carlos Vega Fongaoga.
México, Ed. Prentice Hall, 1988.
Páginas 741.

Harlow Eric
"Guía Avanzada: Desarrollo de Aplicaciones Linux con Gtk+ y Gdk".
Traducción: Vuela Pluma.
España, Ed. Prentice Hall, 1999.
Páginas 496.

W. Kernigghan Brian, Pike Rob.
"La Práctica de la Programación".
Traducción: Vuela Pluma.
México, Ed. Pearson Educación, 2000.
Páginas 280.

Ceballos Sierra Fco. Javier.
"Visual C++ 6. Programación avanzada en Win32".
España, Ed. Alfaomega, 1999.
Páginas 576.

Llanos Ferraris Diego Rafael.
"Curso de C bajo Linux".
España, Ed. Publicaciones Valladolid e Intercambio Científico Universitario, 1998.
Páginas 355.

Lalani "Sam" Suleiman, Kris Jamás Press.
"Java Programer's Library".
U.S.A., Ed. Jamsa Press, 1996.
Páginas 555.

Revistas consultadas:

Mundo Linux.
Ignacio Martín Bragado.
De Linux a Windows Portabilidad (I).
Mensual, México, No 40, Año 2002, Pag: 54-59

Mundo Linux.
Ignacio Martín Bragado.
De Linux a Windows Portabilidad (II).
Mensual, México, No 40, Año 2002, Pag: 62-67

Páginas de Internet consultadas:

Página de consulta de las bibliotecas Qt:

http://www.trolltech.com/

Página principal del software GNU:

http://www.gnu.org

Página principal donde se encuentra todo lo relacionado con la FSF

http://www.gnu.org/fsf/fsf.html.

En esta página se explica todo lo necesario con respecto al tipo de licencias y las diferencias entre las mismas.

http://www.gnu.org/copyleft/copyleft.es.html.

En esta página se informa al usuario sobre los derechos reservados de la licencia copyright

http://www.loc.gov/copyright.

Definiciones sobre los diferentes clases de software que existen:

http://freeware.iespana.es/freeware/defshareware.htm.
http://www.gnu.org/philosophy/categories.es.html#PublicDomainSoftware.

Información de ambas licencias tanto LGPL y GPL:

http://www.gnu.org/copyleft/gpl.es.html.
http://www.gnu.org/copyleft/lgpl.es.html.

Página principal del emulador Cygwin:

http://www.cygwin.org.