Universidad Nacional Autónoma de México

Facultad de Contaduría y Administración

Desarrollo de una distribución GNU/Linux, el caso de Beakos.

Tesis

Angel Ruiz Rosas

México , D.F. **2014**

Universidad Nacional Autónoma México

Facultad de Contaduría y Administración

Desarrollo de una distribución GNU/Linux, el caso de Beakos.

Tesis

Que para obtener el título de:

Licenciado en informática

Presenta:

Angel Ruiz Rosas

Asesor:

Dra. Lucía Patricia Carrillo Velázquez

México, D.F. **2014**

INDICE GENERAL

Agradecimientos

La elaboración de este trabajo me ha dado gran satisfacción personal y profesional. Quiero reconocer y dedicar la presente tesis a quienes me han brindado su apoyo, afecto y educación íntegra para lograr ser una mejor persona.

A la doctora Lucía Patricia Carrillo Velázquez, por la confianza, experiencia y profesionalismo brindado durante sus asesorías, para poder desarrollar nuevas habilidades y conocimientos que me condujeron a la elaboración de esta tesis.

A mis Padres, Margarita Rosas Melo y Angel Ruiz Cruz, por su esfuerzo y sacrificio incondicional, su ejemplo de constancia y disciplina, pero sobre todo por la formación integral que me han brindado a lo largo de mi vida.

A Teresa Rosas Melo, por su franqueza, consejos y apoyo en situaciones cruciales de mi vida.

A Javier Rosas Jiménez, por estar siempre pendiente de mi superación personal, dándome apoyo en todo momento y por su confianza depositada en mí, la cual me ha servido para no desistir.

A Mayra Rosas Rosas y Osvaldo Rosas Rosas, por la sinceridad, consejos y cariño que me han brindado siempre.

A Celia Melo, Benigno Rosas y Lidia Cruz, por sus consejos tan acertados, por enseñarme valores como la prudencia, la serenidad y, sobre todo, por la alegría con la que me reciben siempre.

Durante el trayecto de mi formación y desempeño como profesional, debo agradecer a las personas e instituciones que me han permitido aprender y tener grandes experiencias y satisfacciones al colaborar en el logro de importantes retos y proyectos:

Al doctor Inocencio Higuera Ciapara, por su destacado profesionalismo y, sobre todo, su humildad, la cual me mostró el valor del esfuerzo y el trabajo constante, para impulsar el desarrollo tecnológico de este país. Por darme la oportunidad de

colaborar en el desarrollo de sus proyectos y ser mi referente en cuanto a la pasión por la investigación científica en beneficio de la sociedad.

A los Ingenieros Jesús Arriola Villareal, Francisco Sosa Romero y Antonio Moreno Herrera, por impulsarme a desarrollar nuevos conocimientos, por su dedicación en cuanto a la formación profesional de mi persona, su amistad brindada y la picardía durante momentos de tensión, para amenizar el ambiente en el grupo.

Al ingeniero Junior Marín, por aportar los elementos necesarios para consolidar este proyecto, por su amistad y profesionalismo que se contagia.

A las personas que me han brindado su amistad incondicional, misma que me ha permitido tener confianza y creer en que los retos son posibles de alcanzar, les reitero mi gratitud:

A Aldo Muñoz Morales, por los años de amistad y aprecio, elementos que me han ayudado a alcanzar metas, superar las derrotas y aprender de los errores.

A Oliva Zavala Espino, por su afecto incondicional y la confianza depositada en mí, y por ser una persona ejemplar de superación y optimismo ante situaciones adversas.

A Adriana Esparza, por estar siempre en la mejor disposición de ayudarme, por el cariño y grandes momentos de amistad.

A Carlos Roberto Torres, por su compromiso laboral, su gran ingenio y destreza que me sirvieron de experiencia en mis actividades diarias, y por su gran amistad llena de lecciones de vida.

A mis grandes amistades: Alejandra Esparza**,** Rubén Dávila, Iván Rodrigo Hernández Maya, Tania Atehortúa Castrillón, Carolina Ochoa Carmona, Andrea Ospina Álvarez, Luisa Londoño Restrepo, Mónica Ospina, Melissa Londoño Arboleda, Ana Cristina Álvarez, Daniela Sánchez, Víctor Sierra, Esteban Sepúlveda Zuluaga, David Cardona, Juan Pablo Acevedo Álvarez, Carlos Cortés, y a Cristian Paez Forero, por hacerme parte de experiencias inigualables que me dieron ánimo en momentos de incertidumbre y me motivaron a seguir adelante.

INTRODUCCIÓN

Como parte de las reformas que se están presentando en México, referente a las condiciones para la aplicación de las Tecnologías de la Información y las Comunicaciones (TIC), la vinculación entre investigación y desarrollo (I+D) es parte fundamental para la transición del país hacia la consolidación de una sociedad basada en la información y el conocimiento. Dentro de esta serie de reformas que se trabajan de forma conjunta entre diversos instrumentos del Gobierno Federal Mexicano; para las organizaciones es trascendente hacer un uso eficiente de los recursos tecnológicos con los que dispone, además durante su camino hacia la mejora continua y logro de sus objetivos, debe evitar la saturación de infraestructura tecnológica que le represente un desperdicio de recursos y pérdidas económicas; sin embargo, se tiene presente la necesidad de estar a la vanguardia tecnológica para la mejora de servicios y/o productos de calidad.

Este trabajo expone un caso de éxito en el desarrollo de un producto de software, que basado en la aplicación de I+D se adapta a las necesidades concretas que se presentaron al interior de un Centro Público de Investigación (CPI) perteneciente al Consejo Nacional de Ciencia y Tecnología (CONACYT), permitiéndole a este trabajar en la innovación tanto al interior de la organización para optimizar sus recursos como al exterior, brindando nuevos productos y servicios a las empresas, encaminados de igual forma en apoyar el desarrollo tecnológico de la sociedad.

El tema central de esta investigación gira en torno al uso y administración de sistemas operativos GNU/Linux (que son un producto de Software Libre) dentro del CPI. En este sentido México ha formalizado elementos como: el diseño, desarrollo, aplicación y difusión del Software Libre a través distintos instrumentos y organismos como el Congreso Nacional de Seguridad y Software Libre (CONASSOL) y la Asociación Mexicana Empresarial de Software Libre (AMESOL), por lo que en el país se ha formalizado el desarrollo de software bajo este esquema.

A continuación se hace un resumen del contenido desarrollado dentro de los capítulos que conforman este trabajo de investigación.

Capítulo 1.- Se da un panorama general acerca del Centro Público de Investigación perteneciente a CONACYT, describiendo: líneas de desarrollo, servicios, logros, proyectos destacados entre otros aspectos; posteriormente el contenido de este capítulo se va enfocando en los detalles de operación que se presentaron al interior de una de las áreas, encargada de la gestión de sistemas operativos. A partir de esta descripción acerca del CPI y su problemática, se puede comprender el sentido del objetivo principal planteado.

Capítulo 2.- Expone alternativas de solución, se abordan los conceptos teóricos entorno a métodos y metodologías de desarrollo de software, que aportan las bases para la planeación del proyecto y el logro de los objetivos de forma eficiente. En este capítulo se determina la línea de desarrollo a seguir para cumplir con el objetivo inicial planteado.

Capítulo 3.- Se describe el desarrollo de la solución con base en la aplicación de un método y una metodología analizados previamente, derivando en la construcción de un producto de software con tendencia de mejora y crecimiento, que satisface las necesidades iniciales del Centro Público de Investigación y aporta elementos para el cumplimento su Misión y Visión institucional. Sin embargo, no está dentro del alcance de este capítulo exponer a detalle cada uno de los componentes desarrollados (para ello se puede consultar la documentación publicada el sitio web de este proyecto además de otras referencias que se indican dentro del documento). Por otra parte se ejemplifica el proceso de construcción, tomando como referencia la incorporación de una herramienta de software al producto final.

CAPITULO I. MARCO DE REFERENCIA: CENTRO PÚBLICO DE INVESTIGACIÓN (CPI)

1. Acerca del CPI

El Fondo de Información y Documentación para la Industria (INFOTEC) se creó en 1974, como fideicomiso público del Consejo Nacional de Ciencia y Tecnología (CONACYT) y Nacional Financiera (Nafin). Hoy en día es un Centro Público de Investigación, cuyo propósito es realizar tanto investigación aplicada, como innovación y desarrollo en el ámbito de las Tecnologías de la Información y las Comunicaciones (TIC).

INFOTEC tiene la tarea de formar recursos humanos especializados, que generen soluciones para mejorar la competitividad de los sectores público, académico, social y privado, por medio de la investigación, innovación y desarrollo tecnológico, así como modelos y sistemas organizacionales digitales y estratégicos.

- ***Misión:*** *"Hacemos posible que las organizaciones y las personas se desarrollen mediante el apropiamiento de las TIC."*[1]

- ***Visión:*** *"INFOTEC es un Centro Público de Investigación, Innovación y Servicios que hace posible la instrumentación de proyectos clave para acelerar la incursión de México en la Sociedad de la Información y el Conocimiento."*[2]

Dentro de los hitos a lo largo de la historia de este Centro de Investigación, destacan:

- En 1994, se encargó de la administración y evolución de la Red Tecnológica Nacional (RTN), la primera red de internet en México.

[1] Fondo de Información y Documentación para la Industria; *Visión, Misión y Valores*, [en línea], 1 pp., México, Dirección URL: http://www.infotec.com.mx/en_us/infotec/info_mision_vision [consulta: 21 de Marzo de 2014].

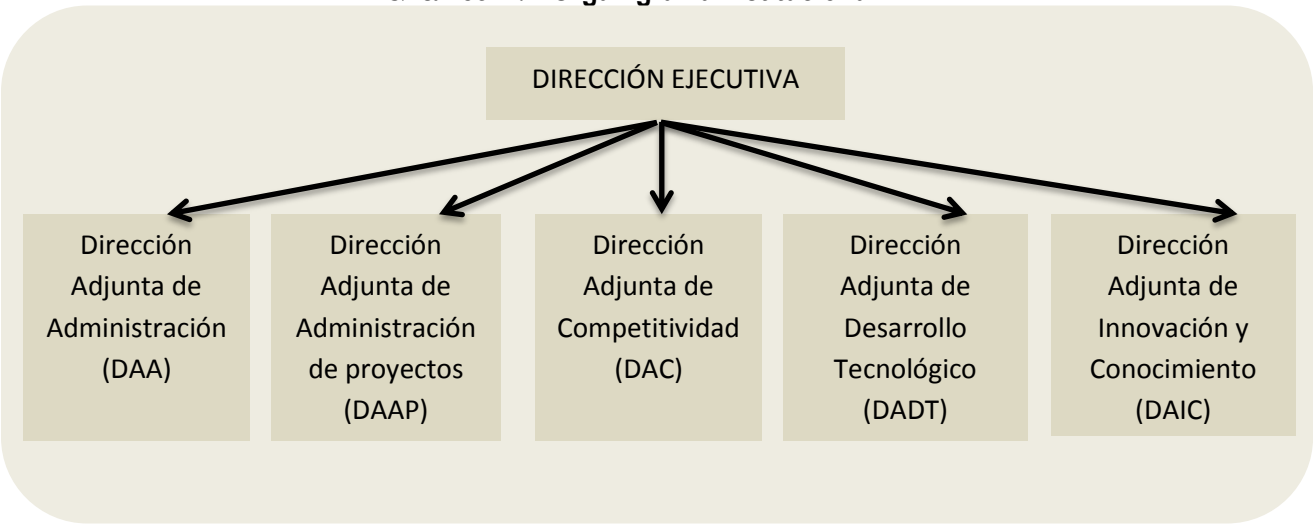
[2] Fondo de Información y Documentación para la Industria; *Visión, Misión y Valores*, [en línea], 1 pp., México, Dirección URL: http://www.infotec.com.mx/en_us/infotec/info_mision_vision [consulta: 21 de Marzo de 2014].

- En 1999, se transformó en Centro de Servicios de Tecnologías de Internet y desarrolló el Sistema ***Compranet***[3], con el que la Secretaría de la Función Pública, recibió el Premio Reto Estocolmo.
- En 2005, el Sistema Estratégico ***Datatur***[4], desarrollado por INFOTEC, fue reconocido con el Premio Innova.
- En 2009, desarrolló y lanzó una nueva herramienta: ***SemanticWebBuilder***, un Sistema de Gestión de Contenidos (*Content Management System*) para desarrollar y administrar portales de internet con tecnología semántica.

2. Estructura Orgánica

INFOTEC está integrado por una Dirección Ejecutiva, Direcciones Adjuntas, así como diferentes gerencias y subgerencias de apoyo.

Gráfico 1.1 **Organigrama Institucional**



[3] Sistema Electrónico de Contrataciones Gubernamentales, simplifica los procesos de contratación de bienes y servicios de las dependencias de la Administración Pública y Federal.
[4] Sistema Nacional de Información Estadística del Sector Turismo.

1. **Dirección Adjunta de Administración (DAA).-** Se encarga de planear, dirigir y administrar los recursos humanos, materiales y financieros, así como las actividades jurídicas de la institución. Es el área responsable de coordinar y vigilar el cumplimiento de las disposiciones de austeridad, racionalidad y disciplina presupuestales, así como las medidas y lineamientos aplicables a la Administración Pública Federal.[5]

2. **Dirección Adjunta de Administración de Proyectos (DAAP).-** Se encarga de dirigir, evaluar y coordinar los proyectos en curso e iniciativas de nuevos proyectos de la institución, asegurando el cumplimiento de los planes y programas. Es el área responsable de diseñar los planes de negocio de la institución; supervisar las actividades y productividad de los proyectos; así como de coordinar las tareas de seguimiento con los clientes.[6]

3. **Dirección Adjunta de Competitividad (DAC).-** Se encarga de la conceptualización y diseño de modelos y soluciones tecnológicas innovadoras, para que las empresas y el sector gubernamental incrementen significativamente su competitividad mediante el uso de las TIC. Es el área responsable de brindar servicios de consultoría estratégica, aplicar metodologías para el desarrollo de proyectos integrales en TIC, apoyar la incubación de proyectos de alta tecnología, analizar procesos y soluciones de negocio, así como diseñar y conceptualizar modelos y sistemas de apoyo para las organizaciones.[7]

[5] Fondo de Información y Documentación para la Industria; Dirección Adjunta de Administración, [en línea], 1 pp., México, Dirección URL: http://www.infotec.com.mx/es_mx/infotec/direccion_adjunta_de_administracion [consulta: 21 de Marzo de 2014].

[6] Fondo de Información y Documentación para la Industria; Dirección Adjunta de Administración de Proyectos, [en línea], 1 pp., México, Dirección URL: http://www.infotec.com.mx/es_mx/infotec/direccion_adjunta_de_administracion_de_proyectos [consulta: 21 de Marzo de 2014].

[7] Fondo de Información y Documentación para la Industria; Dirección Adjunta de Competitividad, [en línea], 1 pp., México, Dirección URL: http://www.infotec.com.mx/es_mx/infotec/direccion_adjunta_de_competitividad [consulta: 21 de Marzo de 2014].

4. **Dirección Adjunta de Desarrollo Tecnológico (DADT**).- Se encarga de dirigir las estrategias de operación, control, infraestructura y mantenimiento de servicios de la institución, así como de contribuir al desarrollo de nuevos productos, mercados y/o negocios, venta e interrelación con otras empresas e instituciones.[8]

5. **Dirección Adjunta de Innovación y Conocimiento (DAIC).-** La Dirección Adjunta de Innovación y Conocimiento se encarga de implementar y conducir proyectos innovadores alineados con la sociedad de la información y el conocimiento, que tienen como base la investigación. Es el área responsable de mantener las relaciones de negocio estratégicas con diferentes instituciones públicas y privadas, así como de planear, diseñar y ejecutar los estudios de posgrado y aprendizaje que ofrece el INFOTEC.[9]

En resumen, la DADT y la DAC se combinan para desarrollar, implementar y operar los sistemas más comunes y con mayor demanda, por su parte la DAIC, dedicada a la investigación, forma la parte metodológica, teórica y académica, que se encarga de la transferencia de información y conocimiento en materia de TIC, hacia la sociedad.

3. Áreas de Especialidad y Líneas de Investigación

La investigación dentro del CPI es parte fundamental para soportar las acciones emprendidas, la cual está orientada hacia la aplicación del conocimiento con el propósito de contribuir al desarrollo e integración de las tecnologías en las organizaciones públicas y privadas.

[8] Fondo de Información y Documentación para la Industria; Dirección Adjunta de Desarrollo Tecnológico, [en línea], 1 pp., México, Dirección URL: http://www.infotec.com.mx/es_mx/infotec/direccion_adjunta_de_desarrollo_tecnológico [consulta: 21 de Marzo de 2014].

[9] Fondo de Información y Documentación para la Industria; Dirección Adjunta de Innovación y Conocimiento, [en línea], 1 pp., México, Dirección URL: http://www.infotec.com.mx/es_mx/infotec/direccion_adjunta_de_innovacion_y_conocimiento [consulta: 21 de Marzo de 2014].

Por tanto el CPI trabaja en el desarrollo de distintas líneas de investigación[10]:

1. Sociedad de la Información y Conocimiento y el apropiamiento social de las TIC,
2. Las TIC y la gestión de la información y el conocimiento,
3. Protección de datos digitales,
4. Seguridad y Ciberdelitos.

El INFOTEC además, tiene presente la necesidad de llevar a cabo estudios prospectivos para identificar problemas, tendencias y escenarios posibles de desarrollo en sus áreas de especialidad. Los resultados de dichos estudios pretenden tener una contribución notable a nivel nacional que le permita mantener presencia en la sociedad de la información y el conocimiento, así mismo, que sus investigaciones apoyen la solución de problemas.

4. Divulgación y apropiación del conocimiento

Para lograr la divulgación del conocimiento, se llevan a cabo diversas actividades las cuales comprenden la publicación y/o difusión de los siguientes apartados:

- Libros,
- Artículos en revistas arbitradas,
- Capítulos de libros,
- Conferencias académicas,
- Seminarios,
- Casos de estudio,
- Presentaciones de libros.

[10] Fondo de Información y Documentación para la Industria; La investigación en Infotec, [en línea], 1 pp., México, Dirección URL: http://www.infotec.com.mx/es_mx/infotec/investigacion_Infotec [consulta: 21 de Marzo de 2014].

5. Investigación + Desarrollo, aplicada a servicios de TIC

Los trabajos en I+D, permiten al INFOTEC ofrecer servicios de calidad en materia de:

- Consultoría,
- Infraestructura tecnológica: física y virtual para recursos de telecomunicaciones, seguridad, administración de sistemas,
- Desarrollo de software echo a la medida,
- Modelado de negocios,
- Dirección de proyectos,
- Estudios de Posgrado: maestrías, diplomados y especialidades.

Para asegurar la correcta implementación de estos servicios, se toman en cuenta estrategias de TI definidas al interior del CPI, las cuales comprenden:

- **Planeación Estratégica de TI**, donde se apoya a los clientes en la definición del plan de negocio para la organización de los recursos tecnológicos, definir objetivos en lapsos de tiempo (corto, mediano y largo plazo), así como emprender las acciones para concretar las necesidades de los clientes.
- **Diagnóstico y Alineación de TI**, diseño de soluciones de infraestructura que se ajusten a las condiciones actuales de las organizaciones, solventando los requerimientos identificados y aportando mejoras.
- **Modelado de soluciones**, donde se apoya a los clientes a estructurar el entendimiento de su entorno, el modelo conceptual de su solución, la estrategia de implementación y el plan del proyecto (alcance, tiempo, recursos e inversión).

6. Servicios profesionales de TIC a cargo de la DADT

La implementación de sistemas de software partiendo de las necesidades de recursos tecnológicos de las organizaciones, debe representarles un costo beneficio que además garantice la operación de sus ambientes productivos de forma segura, continua, estable y escalable; además estas organizaciones demandan que la disponibilidad de sus servicios e información de consulta publicada en la web se mantenga de forma ininterrumpida.

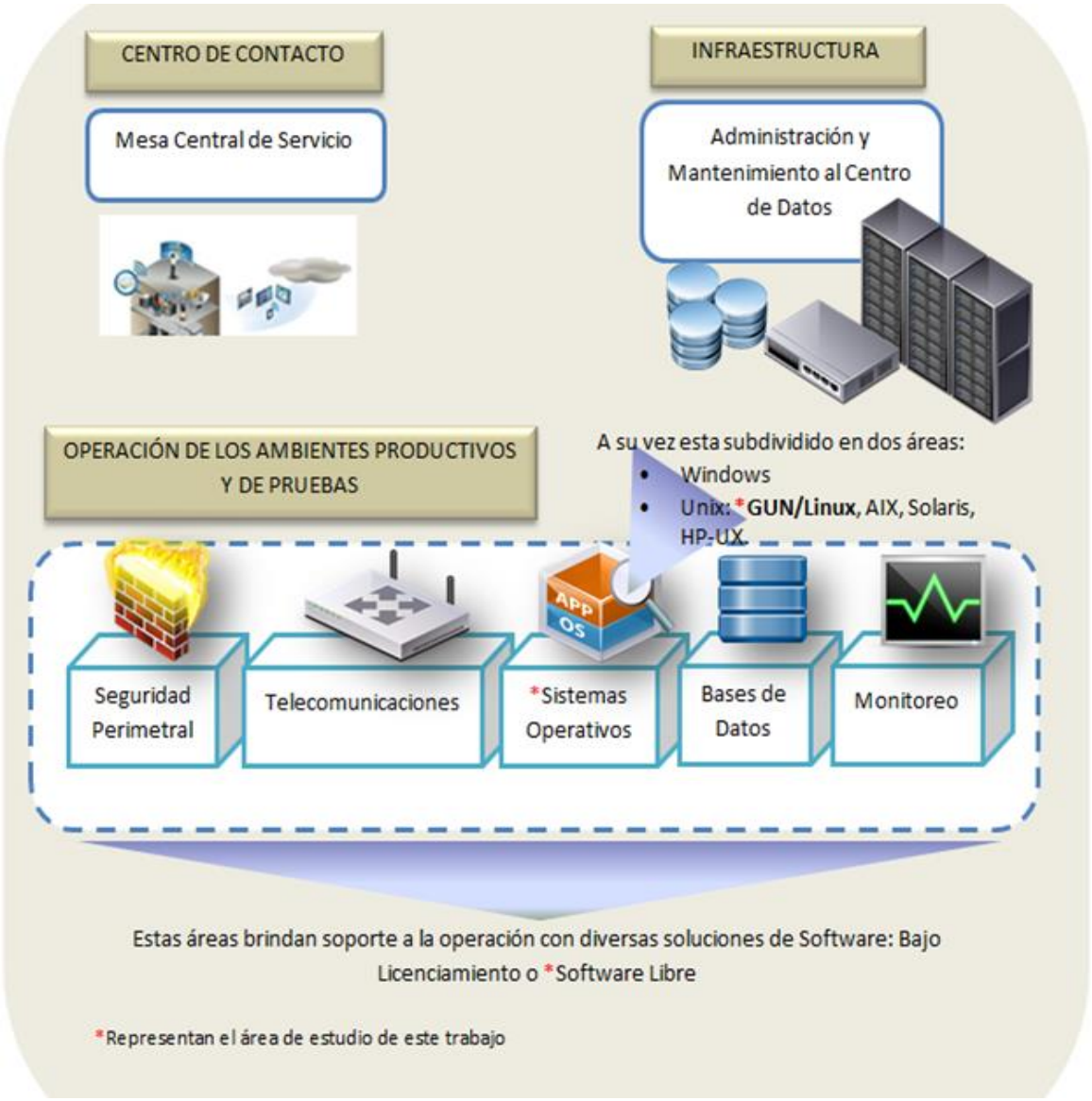
En este sentido la gestión de algunos proyectos para los clientes de INFOTEC involucran la puesta a punto de infraestructura y plataformas que se montan para:

- Sistemas operativos,
- Bases de Datos,
- Telecomunicaciones,
- Seguridad perimetral, entre otras.

Para cada una de estas plataformas, se cuenta con un catálogo de herramientas y soluciones de las cuales la DADT se encarga de su implementación y soporte a la operación; a continuación se describen ejemplos de estas herramientas:

- Instalación y puesta a punto de la plataforma de sistema operativo, que puede ser: Windows, Unix o GNU/Linux (también conocidos como distros o distribuciones GNU/Linux), sobre la cual se montaran aplicativos, bases de datos, repositorios, entre otros recursos.
- Servicios de telecomunicaciones, donde se propone la implementación de infraestructura que contempla: *switches*, *proxys*, *routers,* balanceadores de carga, etc.
- Servicios de seguridad perimetral, que integran una solución de *firewalls*, IPS, filtrado de contenidos, conexiones remotas seguras.
- Servicios de administración de Bases de Datos, soportados por manejadores como *MySQL*, *PostgreSQL, Oracle* entre otros.

Gráfico 1.2. **Áreas Operativas de la DADT**



6.1 Complicaciones en el soporte a la operación de los sistemas operativos GNU/Linux

En el caso de los proyectos desarrollados para los clientes del INFOTEC, la plataforma de sistema operativo propuesta en el diseño de solución es determinante para la administración eficiente de los recursos de hardware destinados a cada uno de los clientes, también es importante en el tema de la compatibilidad de dicho sistema operativo, con los diversos servicios y aplicativos de software que se instalarán.

Al respecto, especialistas de INFOTEC identificaron que cuando era necesario implementar un sistema operativo GNU/Linux (ya sea por requerimiento del cliente o por las necesidades del proyecto), se proponían diversas distribuciones: *CentOS, Red Hat, Fedora, Debian, Ubuntu,* entre otras; esto derivó en que el área encargada de mantener los servidores con sistemas operativos basados en UNIX, tenía toda una gama de sistemas para administrar. Tal variedad de distribuciones era un factor que influía al momento que se requerían realizar diversas tareas como migraciones, actualizaciones, ventanas de mantenimiento y cuando se presentaba algún incidente que afectaba la operación de los ambientes productivos.

Las complicaciones que se presentaron con mayor frecuencia, debido a esta variedad de sistemas fueron:

- Cada distribución GNU/Linux estaba instalada y configurada de manera distinta, por lo que el administrador se llevaba tiempo en obtener un esquema detallado acerca de cómo y dónde se encontraban los servicios que ejecutaba dicha distribución, esto impactaba cuando se presentaba algún incidente y también por ejemplo, era necesario realizar configuraciones complejas en la distribución debido a un requerimiento que el cliente no tenía previsto al inicio de su proyecto.
- Había distribuciones GNU/Linux que estaban sobradas, cuando en algunos proyectos únicamente se requerían herramientas de software básicas; es decir, estas distribuciones incluían paquetería de software adicional que nunca se utilizaba, desde los componentes más básicos del sistema operativo, hasta herramientas de software que únicamente desaprovechaban el consumo de los recursos de procesamiento.

- Las actualizaciones y parches de seguridad que publicaban la mayoría de distribuciones GNU/Linux, en muchas ocasiones tenían que ser instalados a la brevedad posible, sin embargo, se confiaba en que dichas

actualizaciones no fueran a causar conflicto con otras versiones de software ya instalados y que esto no ocasionará inestabilidad en los aplicativos.

- Los requerimientos mínimos de procesamiento para instalar las distribuciones GNU/Linux, no empataban adecuadamente con las características de hardware que se tenían en ese momento dentro del Centro de Datos del INFOTEC, ya que la demanda de procesamiento del software rebasaba las capacidades de la infraestructura.

Estas complicaciones disminuían la eficiencia y oportunidad para atender las necesidades de los clientes en materia de implementación y administración de sistemas GNU/Linux. Dado lo anterior INFOTEC planteó el siguiente objetivo:

"Construir un sistema GNU/Linux ligero para optimizar el desempeño de equipos con hardware limitado, y que le permita al INFOTEC tener control sobre su desarrollo y administración".

El cumplimiento de este objetivo, le permitiría al CPI estandarizar una sola distribución GNU/Linux en la infraestructura limitada que se encontraba en su Centro de Datos, también le permitiría definir para el sistema las formas de implementación y configuración para realizar instalaciones de manera homogénea, lo cual haría más eficiente el soporte a la operación.

CAPÍTULO II. MARCO TÉORICO

1. Gestión de la Función Informática

De acuerdo a lo expuesto en el capítulo anterior, INFOTEC como Centro Público de Investigación y como una entidad prestadora de servicios de TIC, tiene el compromiso de desarrollar soluciones que contribuyan a:

- Cubrir las necesidades y requerimientos de sus clientes en materia de TIC,
- Optimizar el consumo de recursos de su infraestructura de hardware alojada en el Centro de Datos,
- Administrar eficazmente los ambientes productivos hospedados,
- Reducir la brecha digital, además de transmitir el conocimiento obtenido a base de sus investigaciones y desarrollos, es decir, impulsar el desarrollo de la sociedad de la información y el conocimiento.

Por tanto el objetivo planteado por el INFOTEC, pretende cumplir con la problemática expuesta y con los compromisos que tiene como CPI. Dicho lo anterior, se requería de una solución acorde a la situación del INFOTEC, esto derivo en la generación de dos alternativas:

- Adecuación de una distribución GNU/Linux existente, a las necesidades del CPI, para ello era necesario contar con la documentación suficiente y código fuente de la distribución seleccionada.
- Desarrollo a partir de cero de una distribución GNU/Linux propia del INFOTEC.

Para estas alternativas se plantearon los siguientes objetivos estratégicos:

Cuadro 2.1. **Objetivos Estratégicos para el desarrollo del proyecto.**

N°	DESCRIPCIÓN	JUSTIFICACIÓN
1	Integrar un equipo de recursos humanos con las habilidades técnicas acordes al desarrollo del proyecto.	Para disminuir el riesgo de fracaso en el desarrollo del proyecto por falta de experiencia o conocimientos técnicos.
2	Llevar el desarrollo del proyecto dentro del marco Administrativo y de normatividad que requiere el CPI.	Para evitar caer en irregularidades y falta de compromiso por parte de las áreas involucradas.
3	Investigar y retroalimentar los conocimientos teóricos sobre la naturaleza del proyecto.	Para que el equipo de trabajo estuviera en contexto acerca de lo que involucraba el desarrollar un sistema GNU/Linux y aplicarlo durante el proyecto.
4	Definir estrategias para el desarrollo grupal	Para favorecer el desarrollo del proyecto de forma eficiente.
5	Llevar a cabo el desarrollo del sistema GNU/Linux a través de modelos y métodos adecuados a la naturaleza del proyecto.	Para lograr un producto de software de calidad y que cumpla con el objetivo principal planteado.

Cuadro 2.2. **Acciones emprendidas para cumplir con los objetivos planteados**

N°	OBJETIVO ESTRATEGICO	ACCIONES A EMPRENDER
1	Integrar un equipo de recursos humanos con las habilidades técnicas acordes al desarrollo del proyecto.	a) Evaluar los conocimientos técnicos y la experiencia de los integrantes del equipo de desarrollo en manejo de herramientas de software libre a través de: -Trayectoria profesional en la administración y desarrollo de sistemas GNU/Linux, -Cursos y certificaciones obtenidas a fines al área de desarrollo del proyecto.
2	Llevar el desarrollo del proyecto dentro del marco Administrativo y de procesos que requiere el CPI.	**a)** Se integraron las direcciones Adjuntas, cada una con sus compromisos y tareas correspondientes, para la formalización del proyecto.

3	Investigar y retroalimentar los conocimientos teóricos sobre la naturaleza del proyecto.	a) Revisar los conceptos teóricos principales que hacen referencia al Software Libre tales como: -Distribución GNU/Linux, -Código Abierto -Licencias de Software Libre Entre otros temas. b) Analizar proyectos exitosos en cuanto al uso y desarrollo de sistemas GNU/Linux. c) Revisar la documentación de diversas distribuciones existentes.
4	Definir estrategias para el desarrollo grupal	a) Asignar roles de trabajo de acuerdo a las habilidades técnicas de cada uno de los integrantes del equipo de desarrollo. b) Contar con un consultor externo experto en el desarrollo de Software Libre
5	Llevar a cabo el desarrollo del sistema GNU/Linux a través de modelos y métodos adecuados a la naturaleza del proyecto.	a) Elegir un método de desarrollo con base en las necesidades técnicas del sistema GNU/Linux a desarrollar. b) Con base en los recursos, tecnológicos y humanos con que se cuenta para el desarrollo del proyecto, buscar la metodología que mejor se adecue a las condiciones.

2. Definición de conceptos que engloban el desarrollo del proyecto

2.1 Software Libre y Código Abierto

El sentido de libertad que prevalece en el Software Libre va relacionado con las libertades al usuario para utilizar el software y nada que ver con el precio:

- **Libertad 0:** Ejecutar el programa como quiera para cualquier propósito.
- **Libertad 1:** Adaptar el programa, de acuerdo a sus necesidades (para que esta libertad sea efectiva en la práctica, debe tener acceso al código fuente; porque modificar un programa sin disponer del código fuente es extraordinariamente difícil).
- **Libertad 2:** Redistribuir copias, de forma gratuita o con un precio.
- **Libertad 3:** Distribuir versiones modificadas del programa, de modo que la comunidad pueda beneficiarse de sus mejoras.

El proyecto GNU, comprende una variedad de aplicaciones de software que están desarrolladas considerando las cuatro libertades descritas.

El Software Libre cuenta con las características de uso y desarrollo antes expuestas, dichas características aseguran que todo el Software Libre sea de código abierto, sin embargo, esto no aplica a la inversa, es decir, no todo el código abierto es Software Libre, su principal diferencia es que el Código Abierto es una metodología de programación contra el Software Libre que es una filosofía. La intención del Software Libre es brindar un bienestar y beneficio para la sociedad, bajo el lema “compartir y cooperar”.

Por otro lado, a partir del Software Libre se desprende un movimiento social denominado Código Abierto, que se enfoca en resaltar los beneficios de contar con el código fuente de cualquier aplicación, ya que esto permite poner en práctica un modelo de desarrollo que funcione con el aporte de conocimientos de toda una comunidad de usuarios y desarrolladores, con lo que el Código Abierto asegura se obtienen aplicaciones de software estables, seguras, robustas y en constante mejora; además de otras múltiples ventajas para las organizaciones.

Por lo anterior es que el Código Abierto se considera a sí mismo como un promotor del Software Libre.

Cuadro 2.3. **Diferencias entre el Software Libre y el Código Abierto.**

Software Libre	Código Abierto
Es una filosofía de uso y desarrollo de software.	Es una metodología de programación.
Se enfoca en defender las libertades que tienen los usuarios en el uso y desarrollo de los programas de software.	Se enfoca en un beneficio meramente práctico, es decir, en aludir a los beneficios de desarrollar software en conjunto y maximizar la robustez y funcionalidad de los programas.
Garantiza las libertades de los usuarios de programas de software, a través de la GNU GPL.	Su objetivo no es hacer cumplir las libertades de los usuarios, simplemente promueve que el código fuente de cualquier desarrollo esté disponible.
Para que un sistema sea totalmente libre, cada uno de sus componentes debe proporcionar el código fuente por completo, además de respetar las libertades de los usuarios en su totalidad.	Permite la integración de sistemas con software libre y software propietario.
Promueve el desarrollo tecnológico de la sociedad a través del conocimiento compartido.	Promueve el desarrollo de software potente y confiable.

2.2 Kernel Linux

Es un núcleo de sistema operativo, específicamente de los sistemas Unix, sus principales características son:

- Adecuado a los estándares POSIX y *Single Unix Specification* (Especificación Única de Unix).

- Implementa un núcleo monolítico, brindándole funcionalidades como: multitarea, memoria virtual, bibliotecas compartidas, modularidad.
- Es software libre.

Este núcleo es el que incorporan las diversas distribuciones GNU/Linux existentes.

2.3 Distribución GNU/Linux

Una distribución GNU/Linux es un sistema operativo que es Software Libre, integra el núcleo Linux con las aplicaciones de software que pueden ser aquellas que están dentro del proyecto GNU u otras aplicaciones libres que mantienen un desarrollo independiente; estas aplicaciones pueden ser:

- De uso general: procesadores de texto, hojas de cálculo, manejadores de bases de datos, etc.
- De uso específico: aplicaciones contables, Aplicaciones de Punto de Venta, etc.

Existen diversas distribuciones con fines específicos y algunas han marcado una tendencia en cuanto a su desarrollo, que han motivado la creación de toda una gama de sistemas que se desprenden de ellas.

Cuadro 2.4. **Primeras distribuciones GNU/Linux**

Distribución	Descripción
Slackware	Es la distribución más antigua que se sigue manteniendo. Su enfoque principal consiste en la sencillez y fácil uso, es por ello que incorpora un número muy limitado de asistentes de configuración gráficos y por ende no es recomendable para usuarios novatos.
Debian	Se le considera una meta distribución, ha sido el proyecto GNU/Linux sobre el cual se han basado la mayoría de las distribuciones existentes, ya que está desarrollado bajo un esquema muy funcional en varios aspectos técnicos: el

	inicio y ejecución de sus procesos, la administración de sus paquetes de software. Debian incorpora un equilibrio entre sistema operativo fácil de usar y estable.
RedHat	Es una distribución enfocada hacia un ámbito empresarial ya que incorpora paquetes de software que han sido modificados para ofrecer una mayor estabilidad al sistema operativo y a los aplicativos que se ejecutan sobre él.

A estos sistemas GNU/Linux se les conoce como distribuciones “padre” ya que de ellas se han derivado otras distribuciones que siguen su misma línea base de desarrollo, pero que estas distribuciones derivadas, pueden ir desde un enfoque ligeramente distinto al de su distribución “padre” hasta seguir otros objetivos totalmente diferentes.

Hasta el momento existen numerosas distribuciones GNU/Linux que toman la base de su desarrollo a partir de las distribuciones antes mencionadas, cada distribución puede estar también enfocada para cierto tipo de usuarios o para ciertas áreas, por ejemplo: edición de audio y video, ambientes empresariales, seguridad informática, video juegos, etc.

2.4 Licencia GNU GPL

GNU GPL (Licencia Pública General GNU o por sus siglas en inglés *General Public License* GNU), esta licencia se establece para garantizar que se respeten las libertades definidas por el Software Libre: usar, compartir, estudiar, modificar, todo esto dentro de un marco legal. Hasta el momento existen tres versiones de esta licencia, a continuación se presentan las características más destacables de cada una.

Cuadro 2.5. **Historial de versiones de la GNU GPL**

Nombre de la Versión	Principales Características
GPL v1	Garantizar que se proporciona el código fuente completo de los programas de software. Los programas de software bajo esta versión no pueden ser combinados con otro software que estuviera bajo licencias restrictivas.
GPL v2	Agrega una clausula denominada "*Liberty or Death*" la cual define que si una persona se encuentra restringida por leyes para proporcionar únicamente los binarios de un programa de software, esta persona no puede distribuir software libre.
GPL v3	Resuelve ambigüedades y aumenta la compatibilidad de la GPL v3 con otras licencias. Facilita su adaptación a otros países, Incluye cláusulas que evitan el uso indebido de las patentes de software.

3. Análisis de proyectos exitosos en cuanto al uso y desarrollo de sistemas GNU/Linux

Se debían buscar casos de éxito donde las distribuciones GNU/Linux fueran el factor principal de solución, con la finalidad de:

- Analizar los modelos de gestión informática que influyeron en la solución de necesidades,
- Analizar distintos modelos y métodos empleados en el desarrollo de software libre.
- Determinar la viabilidad técnica y presupuestal para adoptar un esquema de desarrollo similar,

- Tener retroalimentación externa a través de conferencias o consultorías por parte de un equipo de trabajo, para conocer las problemáticas o retos a los que se enfrentaron,
- Conjuntar ideas entre las áreas involucradas del INFOTEC, con aportaciones de solución encaminadas a cumplir con el objetivo planteado.

De entre varios proyectos que se documentaron, a continuación se describen los más destacables, por proporcionar un aporte suficiente en diversos temas como:

- Contribución para reducir la brecha digital,
- Innovación,
- Desarrollo de software hecho a la medida,
- Desarrollo basado en modelos,
- Documentación estandarizada,
- Desarrollo de sistemas GNU/Linux como estrategia para la educación y administración pública,

3.1 ARCH Linux

Es una distribución GNU/Linux canadiense desarrollada a partir de cero (desarrollo independiente), es decir, no está basada en alguna otra distribución existente, su desarrollo se destaca por la simplicidad desde el punto de vista técnico, que ofrece al usuario una configuración precisa del sistema operativo, lo cual brinda un desempeño ligero de la distribución. Este proyecto de desarrollo es comparable con los sistemas operativos GNU/Linux que se obtienen a través del método ***Linux From Scratch*** (**LFS**), (el cual se detallará más adelante).

Esta distribución ha ganado gran aceptación por parte de la comunidad de usuarios y desarrolladores ya que ha retomado el objetivo de brindar un sistema operativo que:

- Optimice los recursos de procesamiento.

- Propone un esquema de desarrollo que favorece la integración de toda una comunidad.
- Permite un nivel de personalización por encima de otras distribuciones.
- Incrementa el nivel de conocimiento de sus usuarios en cuanto a la administración y desarrollo de sistemas GNU/Linux, por ende, los usuarios requieren del auto aprendizaje y basta documentación para poder utilizarla.

Dentro de su sitio web se encuentra toda la documentación, de forma organizada en: foros de ayuda, reporte de fallas, historial de versiones, entre otros elementos que están disponibles a los usuarios. Actualmente se encuentra dentro de las diez primeras distribuciones de mayor popularidad publicadas por el sitio web ***DistroWatch*** [11].

3.2 CANAIMA

Es una distribución venezolana, su desarrollo se sustentó bajo la finalidad principal de ser una solución para instalarse en equipos de cómputo de la administración pública nacional. Además se fijaron objetivos tales como:

- Aprovechar las bondades del Software Libre,
- Desarrollar un proyecto socio-tecnológico de forma colaborativa,
- Desarrollar herramientas y modelos productivos apoyados en el Software Libre,
- Construir sistemas operativos de Software Libre para que la nación venezolana tenga la apropiación de sus TIC y generar desarrollo de conocimiento.
- Impulsar diversos proyectos nacionales a nivel público y privado en sectores como: educación, administración pública, telecomunicaciones.

[11] Se encarga de publicar un listado sobre la popularidad que tienen los sistemas operativos como GNU/Linux, UNIX, Solaris, entre otros; para que un sistema operativo sea publicado en este sitio debe cumplir con estándares y modelos de desarrollo.

Canaima GNU/Linux está basada en *Debian* GNU/Linux, por las siguientes razones:

- *Debian* es considerara una metadistribución, es decir, que de ella se pueden desprender otros desarrollos de software con fines específicos.
- Es una distribución que ha madurado en cuanto a su modelo de desarrollo y a la paquetería de software con que cuenta.
- De ella se han desprendido otras distribuciones como: *Ubuntu*, *Knopix*, *Guadelinex*, entre otras.

Canaima GNU/Linux proporciona elementos de soporte técnico de forma organizada a través de:

- Chat IRC,
- Wiki,
- Listas de correo,

Actualmente Canaima GNU/Linux también es una metadistribución, ya que proporciona un sistema GNU/Linux con aplicaciones básicas a partir del cual se pueden instalar o desarrollar nuevas versiones o sabores (como sus desarrolladores lo definen) de Canaima GNU/Linux. Actualmente esta modalidad de desarrollo ha conseguido que la distribución cuente con versiones de sistema operativo que se adaptan a las necesidades de diversos organismos públicos y de recursos tecnológicos, como:

- Administración pública,
- Educación a nivel primaria,
- Investigación forense en sistemas informáticos,
- Optimización del rendimiento del sistema operativo en equipos de cómputo de baja capacidad.

Es decir, existe una versión de Canaima GNU/Linux para cada uno de los puntos mencionados.

El modelo de desarrollo definido y empleado por Canaima GNU/Linux, toma en cuenta los siguientes elementos:

1. Partir de las necesidades y nuevos proyectos propuestos por su comunidad de usuarios y desarrolladores,
2. Definir en concreto cuáles serán los productos y/o mejoras a desarrollar,
3. Conformar grupos de trabajo,
4. Definir un Mapa de Ruta de su próxima versión,
5. Realizar una etapa de Desarrollo a través de un modelo en espiral, el cual comprende la siguientes fases:
    a. Modificación del Código Fuente,
    b. Versionamiento,
    c. Empaquetamiento,
    d. Puesta a prueba.
6. Realizar una etapa de Pruebas empleando las mismas fases de su modelo en espiral para desarrollo. Tanto en la etapa de desarrollo como de pruebas, se generan paquetes de software que son depositados en el repositorio oficial del proyecto.
7. Integrar la versión estable de la distribución,
8. Integrar la versión estable de Canaima Semilla (la metadistribución),
9. Se libera una nueva versión de Canaima GNU/Linux.

Por otra parte se analizó la documentación que proporcionan las diversas distribuciones GNU/Linux, pero se identificó que en las distribuciones revisadas siempre existían elementos de configuración de bajo nivel, de los cuales no se disponía de documentación, esto representaba una limitante en cuanto al control sobre el desarrollo del sistema GNU/Linux planteado por el INFOTEC.

Con base en lo anterior, los ejemplos de proyectos exitosos fueron una adecuada referencia para optar por el desarrollo de una distribución GNU/Linux a partir de cero y que fuera propia del CPI.

4. Análisis de herramientas disponibles para el desarrollo de distribuciones GNU/Linux

Existen diversas herramientas que tienen por objetivo el desarrollo de nuevas distribuciones GNU/Linux, sin embargo, la gran mayoría comienzan su desarrollo a partir de una distribución base y se enfocan en la personalización de la distribución a un nivel básico. A través de una aplicación se va guiando el proceso de creación de la distro, la cual se va a derivar de la distribución base que se haya tomado.

Cuadro 2.6. **Herramientas analizadas para la creación de distribuciones GNU/Linux**

Nombre del Proyecto	Distribución en la que se basa	Características	Ventajas	Desventajas
SUSE Studio	**Open SUSE/ SUSE Linux Enterprise**	Permite generar fácilmente una distribución a través de un constructor de software, basado en una sencilla interfaz de usuario web. El usuario puede armar fácilmente una distribución basada en SUSE Linux Enterprise u openSUSE. Permite seleccionar: • Sistema Operativo Base. • Tipo de arquitectura. • Entorno de escritorio. • Software incluido. Una vez creada la distribución puede ser descargada como una imagen .iso para que posteriormente se pueda descomprimir en algún medio de almacenamiento tal como una USB o un DVD.	La distribución se crea a través de una interfaz de usuario sencilla. No se requiere de conocimientos técnicos avanzados, además de que rápidamente se puede obtener una distribución funcional y estable, su repositorio cuenta con suficiente paquetería de software para instalarse. La distribución puede ser probada por la comunidad de usuarios a través del sitio web de SUSE Studio, donde se puede comenzar a crear una comunidad de desarrolladores que ayuden a mantener la nueva distribución y que al mismo tiempo le aporten nuevas características.	No se posee el control total sobre el desarrollo de la distribución ya que siempre sigue la línea base de desarrollo de Open SUSE o SUSE Linux Enterprise. El nivel de personalización es básico. Elementos como el soporte, la seguridad y versión de los paquetes de software dependen de la distribución padre.

GENTOO	**GENTOO**	El usuario tiene la opción de instalar un sistema base, a partir de ahí puede ir compilando los demás paquetes que requiera, hasta obtener su propia distribución personalizada, basada en Gentoo.	Durante la construcción de la distribución, el usuario va adquiriendo conocimiento acerca de la funcionalidad de cada uno de los componentes que va incorporando a su sistema. Permite desarrollar una distribución realmente ligera.	Se requiere de conocimientos técnicos para poder desarrollar la distribución. El desarrollo de la distribución es completamente a través de línea de comandos, por lo que no es muy agradable a usuarios acostumbrados a los asistentes gráficos.
UBUNTU Custumization Kit	**UBUNTU**	Es una herramienta de software que permite crear una distribución personalizada de Ubuntu o de sus distribuciones derivadas tomando una imagen .iso del sistema operativo. La herramienta consiste en un asistente gráfico que guía paso a paso la personalización de la distribución, lo que se obtiene con esta herramienta es una distribución personalizada de Ubuntu, con el entorno de escritorio, idioma y paquetería de software seleccionada por el usuario.	Si el usuario está familiarizado con Ubuntu o con alguna de sus distribuciones derivadas, se le facilita personalizar la distribución de acuerdo a su preferencia para poder cubrir sus necesidades. Enfocada para usuarios que requieren de modificaciones mínimas y básicas al sistema operativo para poder cubrir sus requerimientos y sacar mayor provecho de una distribución Ubuntu. La nueva distribución creada deriva de una de las distribuciones con mayor impacto en la comunidad de desarrolladores y usuarios, lo que la convierte en una distribución estable y con basto soporte de software y hardware.	El nivel de adaptación de la distribución es muy limitado en cuanto a requerimientos técnicos muy específicos, enfocándose únicamente en la modificación de la interfaz gráfica (fondo de pantalla, temas de los íconos, color de ventanas), paquetería de software incluida en el .iso y edición de algunos scripts de arranque del sistema. A pesar de que el usuario puede personalizar la distribución para incluir solo la paquetería de software que requiera tener instalada, el consumo de recursos de procesamiento sigue siendo elevado.

De las herramientas consultadas se concluye que tienen por objetivos:

- Crear distribuciones de uso específico y que en algunos casos, representen un menor consumo de recursos de procesamiento.
- Seleccionar el entorno de escritorio (íconos, fondos de pantalla, apariencia de las ventanas, etc.) acorde a las preferencias y necesidades del usuario.
- Modificar los scripts de arranque del sistema, para que el usuario defina que aplicaciones se deben ejecutar cuando inicie sesión.
- Instalación de paquetería específica, el usuario puede escoger que paquetería básica se incluye dentro de la distribución: ofimática, navegadores web, herramientas para administración de red, servidores (Web, Correo, Base de Datos, FTP) juegos, etc; además de que se tiene la opción de instalar paquetería adicional desde el servidor de repositorios.

Cabe aclarar que las herramientas descritas con anterioridad, pueden dar como resultado la creación de una nueva distribución, tomando en cuenta que dichas distribuciones tienen un nivel básico de desarrollo, pero que al final gracias a estas modificaciones básicas la nueva distribución puede tomar un enfoque distinto del que tiene su distribución padre.

5. Método de Desarrollo

Como se acaba de describir en el apartado anterior, la mayoría de las herramientas que ofrecen la creación de una nueva distribución GNU/Linux, permiten únicamente una personalización del sistema y en algunos casos un nivel de desarrollo básico, es decir, estas herramientas van enfocadas a usuarios finales; es por ello que la mayoría son de fácil uso, se enfocan en la apariencia gráfica y no requieren de conocimientos técnicos avanzados. Sin embargo, para que el proyecto de INFOTEC pueda cumplir con su objetivo planteado, este demanda un mayor alcance de desarrollo sobre el sistema operativo. Es decir, que el método de desarrollo a utilizar debe ser capaz de guiar la construcción de sistemas GNU/Linux desde cero.

5.1 Acerca de Linux From Scratch (LFS)

LFS es una colección de notas sobre cómo construir un sistema Linux[12] base, incorpora una serie de pasos y configuraciones que se deben hacer para que al final se obtenga un sistema con los componentes mínimos para operar un sistema Linux.

El objetivo del LFS es ser el primer punto de referencia para iniciar el proceso de construcción de una distribución GNU/Linux, partiendo desde la compilación de un *kernel* de Linux junto con sus módulos, controladores, y herramientas de desarrollo; además de la compilación, configuración de programas y herramientas de software (pertenecientes al proyecto GNU) que se deseen incorporar a la distribución.

Por ejemplo, con LFS se puede desarrollar una distribución GNU/Linux que se encargue de soportar los principales servicios de seguridad perimetral integrando herramientas como IPS, *firewall*, *iptables*, *sniffer* entre otras, para poder auditar y asegurar los sistemas informáticos, otro ejemplo es desarrollar una distribución enfocada a la edición de audio y video, o una distribución enfocada al diseño gráfico; entre otras más opciones para la creación de distribuciones especializadas.

La estructura de LFS se apega a los principales estándares que siguen los sistemas operativos basados en UNIX:

1. POSIX.1-2008.
2. File system Hierarchy Standard (FHS).
3. Linux Standard Base (LSB) Specifications.
4. Single UNIX Specification.

Dentro de las principales ventajas que ofrece el LFS se encuentran:

- Permite que los usuarios/desarrolladores conozcan cómo se integra y trabaja un sistema Linux y cuáles son los principales programas GNU que

[12] Aunque se ha definido que Linux como tal, se refiere al núcleo del sistema operativo, un sistema Linux además del núcleo incorpora un conjunto de herramientas y comandos básicos para la ejecución de tareas.

se deben incorporar para cualquier distribución GNU/Linux con componentes mínimos.

- Desarrollar distribuciones compactas, que aprovechen los recursos de hardware donde son instaladas. Por ejemplo, una distribución configurada para montar un servidor web, probablemente utilizará paquetes como Apache, PHP, MySQL, Java, etc, y dada la naturaleza de su operación paquetes como suite de ofimática, interfaz gráfica, juegos, servicio de impresión, cliente/servidor de correo, cliente de mensajería instantánea etc, estarán de sobra y solo consumirán recursos de procesamiento. *Linux From Scratch* implica la construcción de una distribución desde cero, permitiendo que se integré únicamente el software necesario para cumplir con un fin determinado.
- Incorpora flexibilidad, ya que ofrece la construcción de un sistema GNU/Linux base, el o los desarrolladores pueden construir paquetería de software adicional, personalizar *scripts* para la ejecución de tareas automáticas como: respaldos, rotación de logs, registro de accesos al sistema, etc.
- Incorpora seguridad, debido a que los programas de software son compilados desde el código fuente, además de que se puede verificar su integridad (es decir, comprobar que el código no ha sido alterado por terceros) a través del algoritmo criptográfico *MD5*, comúnmente el más utilizado en las distribuciones GNU/Linux.

Al concluir los pasos del LFS el sistema es apto para usuarios competentes y no representa una opción para un ambiente de producción, sin embargo, como ya se ha dicho es el primer paso para comenzar con el desarrollo una distribución GNU/Linux personalizada y optimizada para soportar servicios en específico. El LFS concluye al entregar un sistema Linux básico (que se ejecute en un solo equipo de cómputo), que incluya componentes mínimos, pero garantizando que dicho sistema sea modular y escalable. Posterior a ello se encuentran otros

métodos de desarrollo que guían y complementan la construcción de una distribución GNU/Linux como las que se conocen para usuarios finales.

- ***Beyond Linux From Scratch* (BLFS)**: Es la continuación al LFS, se enfoca en la construcción y configuración de paquetes de software que son incorporados al sistema base construido desde LFS para obtener un sistema GNU/Linux funcional. Su principal característica es que guía a los desarrolladores en la construcción de un sistema GNU/Linux distribuible.
- ***Automated Linux FormScratch* (ALFS)**: Proporciona herramientas para la automatización y gestión de la construcción de un sistema LFS y BLFS.
- ***Cross Linux FormScratch* (CLFS)**: Proporciona los medios para realizar una compilación cruzada de un sistema LFS en distintos tipos de sistemas.
- ***Hardened Linux From Scratch* (HLFS)**: Se centra en la construcción de un sistema LFS incorporando mayor seguridad.
- **Hints**: Es una colección de documentos que explican cómo mejorar el sistema LFS con instrucciones que no se incluyen en los manuales del LFS ó BLFS.
- **LiveCD**: Proyecto el cual se enfoca en incorporar la paquetería necesaria para proporcionar un CD de rescate.
- **Patches**: Subproyecto conformado por un conjunto de servidores con la finalidad de ser un repositorio central para todos los parches disponibles del LFS.

5.2 Pasos de Linux From Scratch

A continuación se describen brevemente y de forma general las instalaciones y configuraciones principales que se realiza en cada paso del LFS.

I. INTRODUCCIÓN

1. **Introducción.-** Se exponen los objetivos del LFS, además se hacen recomendaciones sobre los temas y conocimientos técnicos adecuados para una mejor compresión de la documentación.

II. PREPARAR EL AMBIENTE DE DESARROLLO

2. **Preparar una nueva partición.-** Indica cómo crear una partición de Linux, un sistema de archivos y la ruta donde se compilará e instalará el sistema LFS.
3. **Paquetes y Parches.-** Se explica que paquetes y parches de software son necesarios descargar para construir un sistema LFS y cómo aplicarlos dentro del sistema de archivos.
4. **Preparaciones finales.-** Se realizan configuraciones como la declaración de variables de entorno globales, creación de directorios, configuración de una cuenta de usuario, para dejar listo el ambiente de desarrollo.
5. **Construcción de un sistema Linux Primario.-** Se realiza la instalación de la paquetería básica para integrar una plataforma de desarrollo sobre la cual se construye el sistema LFS.

III. CONSTRUCCIÓN DEL SISTEMA LFS

6. **Instalación del software básico del sistema.-** Se detalla la instalación de paquetes necesarios para resolver las dependencias recursivas, por ejemplo, al momento de construir un compilador se requiere de otro compilador.
7. **Creación de los scripts de arranque del sistema.-** Explica cuáles son los archivos que se deben generar para arrancar los principales servicios del sistema.
8. **Hacer el sistema LFS arrancable.-** Se configura un cargador de arranque (GRUB) que permite seleccionar entre otras opciones con que núcleo arrancará el sistema LFS.

9. **Ajustes finales.-** Se detallan algunas configuraciones necesarias para poder continuar con el desarrollo de la distribución GNU/Linux.

6. Metodología de Desarrollo

Para determinar qué metodología emplear en la construcción del sistema GNU/Linux, esta se debía adecuar a las siguientes condiciones del equipo de trabajo:

- Reducido número de desarrolladores,
- Infraestructura de desarrollo limitada en cantidad y capacidad de procesamiento,
- Cambios inesperados en los requerimientos del sistema,
- Desarrollar una primer versión del sistema GNU/Linux en un lapso corto de tiempo,
- Favorecer el autoaprendizaje y la retroalimentación de conocimiento entre los desarrolladores.

Un proyecto de Software Libre enfoca la mayoría de sus recursos a la programación, además dispone de la línea desarrollo de sus iniciadores más el desarrollo a voluntad de una comunidad que pueden ser: instituciones educativas, personas particulares, fundaciones, etc; con lo cual se requiere implementar un modelo dinámico que permita el desarrollo de un sistema tomando en cuenta los factores anteriores.

Se decidió implementar los Métodos Ágiles de Desarrollo, específicamente Programación Extrema (XP) porque se adecuan con el entorno del equipo de trabajo y la naturaleza del desarrollo de Software Libre.

6.1 Metodologías Ágiles

Estas metodologías se basan en cuatro premisas, conocidas como Manifiesto Ágil, que definen las formas de trabajo, las cuales se listan a continuación:

1. Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. Es más importante construir un buen equipo que construir el entorno.
2. Desarrollar software que funciona más que conseguir una buena documentación. No producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante.
3. La colaboración con el cliente más que la negociación de un contrato. Interacción constante entre el cliente y equipo de desarrollo.
4. Responder a los cambios más que seguir estrictamente un plan. La planificación no debe ser estricta sino flexible y abierta.

6.2 Programación Extrema (XP)

Se enfoca a todos aquellos proyectos de desarrollo de software, donde los requerimientos no se encuentran bien establecidos o constantemente sufren cambios. Se lleva a cabo mediante conjunto de reglas y prácticas que se realizan en ciclos cortos de desarrollo: análisis, diseño, desarrollo, pruebas, implantación.

Prácticas

1. El juego de la planeación: Consiste en delimitar los alcances de cada ciclo de desarrollo (conocido como iteración). En esta práctica se involucran dos tipos de actores:
    a. Equipo de negocios,
    b. El equipo de desarrollo,
2. Pequeñas entregas: Ciclos cortos de desarrollo con las funcionalidades más valiosas, lo que conlleva a realizar las etapas de: análisis, diseño, desarrollo, pruebas, implantación. Al final de cada iteración se entrega un avance al cliente, y que este pueda retroalimentar para la mejora del sistema.
3. Metáfora: Elaboración de una idea general del sistema que prescinda de tecnicismos. Una historia para definir, estimar y encargar a algún miembro

el desarrollo de un problema. Presentar una solución y reescribir la historia para ir detallando más los problemas y sus soluciones.

4. Diseño Simple, el sistema debe contener solo aquellas funcionalidades necesarias.
5. Pruebas, (*Testing*).
6. Refabricación, busca la eliminación de funciones obsoletas.
7. Programación por pares.
8. Propiedad Colectiva: Establece que cualquier integrante del equipo puede modificar el código.
9. Integración Continua.
10. Cuarenta horas semanales.
11. Cliente en el sito de desarrollo.

7. Estrategia para el Desarrollo Grupal

La organización del equipo de desarrollo se apoyó en las prácticas de la Programación Extrema, que se adecuaban a las capacidades y habilidades técnicas, además de que propiciaba una retroalimentación de conocimiento de manera directa entre los integrantes del equipo; lo cual colaboraba para un entorno adecuado durante el desarrollo del proyecto.

Dentro de los roles que se designaron entre el grupo de desarrollo se encuentran:

- Líder de desarrollo.- Dentro de sus funciones se encuentran la de definir, delimitar y dirigir la línea de desarrollo para la construcción del sistema GNU/Linux. Además de la programación avanzada de componentes de software complejos como el núcleo del sistema, controladores del hardware para arquitecturas a 32 y 64 bits, la construcción de la biblioteca *glibc*, entre otras. Revisa los avances del equipo de desarrollo de forma continua.

- Desarrollador senior.- Coordina las funciones de los desarrolladores *junior*, *tester* y documentador, a su vez se encarga de la construcción de aplicaciones complejas para la integración de elementos como interfaces gráficas de usuario, protocolos de red, herramientas de seguridad y virtualización en el sistema GNU/Linux. Tiene la responsabilidad de la construcción de las aplicaciones críticas para la operación del sistema GNU/Linux en ambientes productivos.
- Desarrollador junior.- Apoyan la construcción de las aplicaciones del desarrollador Senior, además de que se encargan de la construcción de bibliotecas básicas, paquetería de software que complementa al sistema GNU/Linux, tales como: ofimática, multimedia, diseño gráfico, clientes de correo, etc.
- Tester.- Conjunta los nuevos paquetes de software compilados por el equipo de desarrollo, actualiza el servidor de repositorios del sistema GNU/Linux, realiza la instalación de nuevos paquetes en un ambiente de pruebas, notifica los resultados de las pruebas, depura los archivos de configuración de los paquetes de software.
- Documentador.- Define los elementos pertinentes de documentación que emplean otras distribuciones GNU/Linux tales como: manuales de instalación y configuración (tanto del sistema como de la paquetería de software disponible), wiki, listas de correo y foros de ayuda principalmente. Genera y mantiene actualizada la documentación disponible en el sitio web oficial de la distribución.

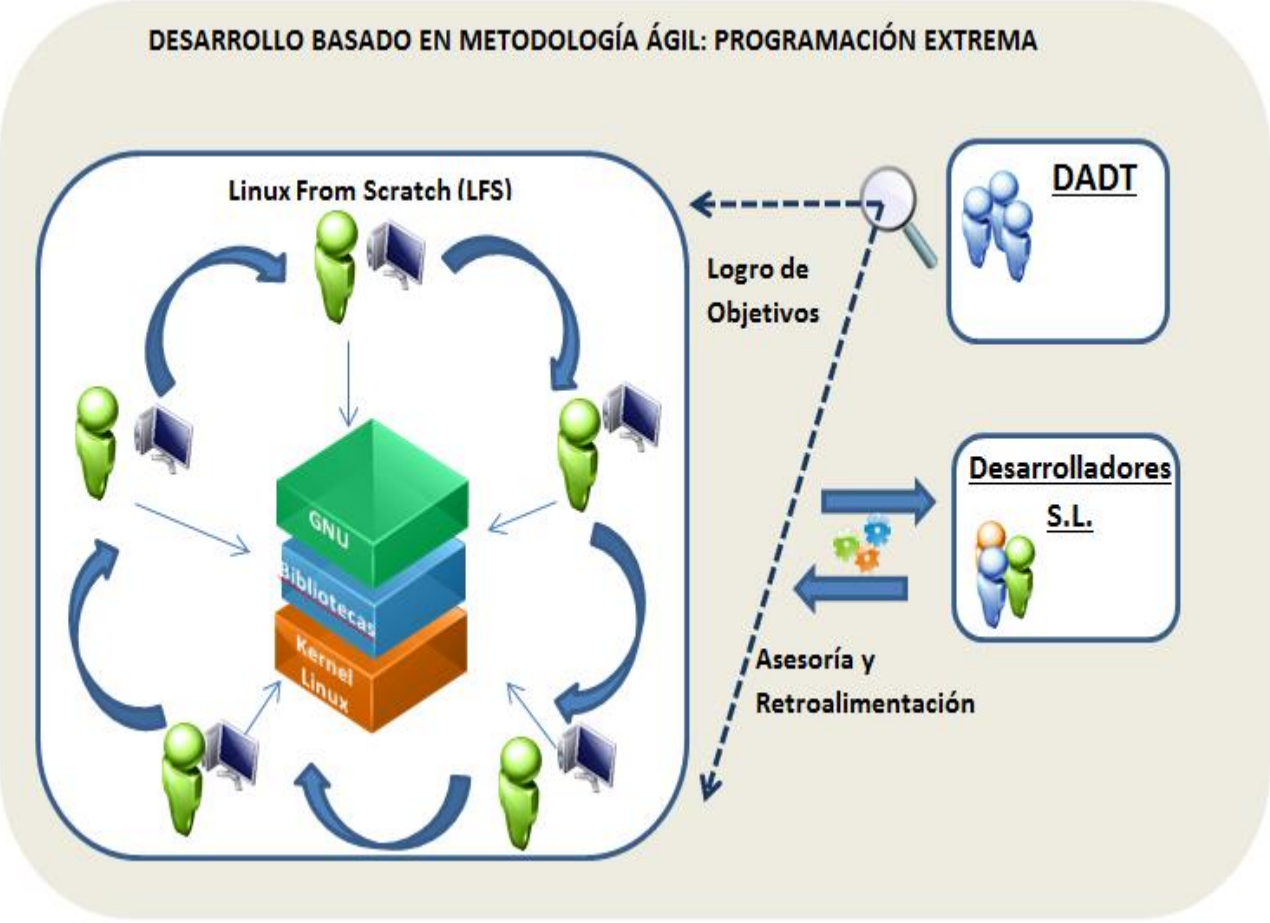
Como parte de la estrategia para el desarrollo grupal, se tuvo consultoría externa por parte del Investigador *Octavio Rosell* de Venezuela, quien es experto en Software Libre, funge como un articulador de organizaciones sociales de base tecnológica en el área del Software Libre y GNU/Linux. Dentro del ámbito profesional ofrece servicios de consultoría en procesos de migración e implementación del Software Libre en su país, además de que posee dominio sobre licencias libres, manejo de lenguajes de programación, configuración y

administración de sistemas GNU/Linux y dominio de diversas herramientas de Software Libre.

Durante su estancia en INFOTEC *Octavio Rosell* colaboró con:

- Ciclo de conferencias, con la finalidad de informar a la comunidad acerca de las bondades del Software Libre.
- Asesoramiento dentro de la planeación en la parte técnica para definir la línea de desarrollo que INFOTEC implementaría para la construcción de su propia distribución GNU/Linux.
- Impartir cursos de desarrollo de Software Libre para la capacitación de los desarrolladores.

Gráfico 2.1. **Desarrollo Grupal**



CAPITULO III. DESARROLLO DE LA SOLUCIÓN

1. Delimitación de la distribución: Beakos GNU/Linux

Beakos GNU/Linux es el nombre que se decidió dar al desarrollo de INFOTEC, su logotipo está representado por un pato con los emblemas y colores institucionales:

Gráfico 3.1. **Logotipo de Beakos GNU/Linux**

1.1 Tipo de licenciamiento

Beakos GNU/Linux es distribuido bajo la **GNU GPL** (Licencia Pública General GNU o por sus siglas en inglés *General Public License GNU*) **versión 3**, es decir, respeta las libertades de los usuarios para modificar, distribuir, y tener acceso al código fuente del desarrollo de la distribución, esto permite que todas las actualizaciones o modificaciones a la distribución sean liberadas bajo las mismas condiciones. Se optó por utilizar esta licencia ya que la línea de desarrollo de Beakos GNU/Linux está enfocada a respetar las libertades de los usuarios con respecto al uso del software. La GNU GPL v3 también conducirá a que se obtenga un desarrollo de Beakos GNU/Linux 100% libre, es decir, no incorporar firmware, ni cualquier otro software que no proporcione su código fuente.

La *Free Software Foundation* (FSF) promueve el uso de la GNU GPL versión 3, ya que esta encamina los desarrollos de software a respetar las cuatro libertades de los usuarios y no permite que las empresas puedan beneficiarse lucrativamente con desarrollos de programas tomados del software libre, que posteriormente son

licenciados bajo condiciones restrictivas; es decir, esta tercer versión de la GNU GPL, corrige ciertos detalles que se pasaron por alto en la versión anterior (GNU GPL versión 2).

De acuerdo a lo mencionado anteriormente, otro aspecto distintivo entre *Free Software* y *Open Source*, se denota en el tipo de licencia que utilizan, es decir, el *Free Software* invita a los desarrolladores a migrar hacia la versión 3 de la GNU GPL, mientras que el *Open Source* se queda con la GNU GPL versión 2, debido a que esta conviene más a los intereses del propio Código Abierto.

1.2 Protocolos de Operación Grupal

Tal como se expuso en el capítulo anterior*, Linux From Scratch* es el método que más se adecua al objetivo de desarrollo planteado por el INFOTEC.

Toda vez que se había decidido desarrollar una distribución desde cero, al final, esta debía incorporar ciertos controladores (*drivers*), bibliotecas y herramientas de software que los desarrolladores decidieran incluir al momento de liberar una versión de Beakos GNU/Linux; conforme se fueran compilando cada uno de los programas desde su código fuente, estos se instalarían y configurarían mediante *scripts* en la infraestructura montada para el desarrollo. El alcance del LFS no es entregar un sistema GNU/Linux distribuible, sin embargo, para la construcción de Beakos GNU/Linux, se tomó en cuenta desde un inicio definir e implementar formas estandarizadas para poder gestionar y distribuir los programas incorporados (compilados) al sistema, de lo contrario, los programas de software únicamente podrían ser ejecutados en el hardware donde fueron compilados, lo cual no formaría parte de un desarrollo viable.

Por otro lado, tampoco es viable que una sistema GNU/Linux incorpore toda su variedad de software en cada una de sus versiones liberadas ya que tardaría demasiado tiempo en instalarse; también para distribuirse se necesitaría uno o varios medios con capacidad de almacenamiento suficiente para guardar un volumen de aplicaciones de software considerable; este esquema sería muy tedioso, complicado de administrar y se desecharía rápidamente.

1.2.1 *Elección de un formato de empaquetamiento, para definir un estándar en la paquetería de Beakos GNU/Linux*

Un **formato de empaquetamiento** es un concepto empleado para referirse a los paquetes de software con determinada extensión de archivo, que contienen un conjunto de ficheros binarios, *scripts* y manuales de un programa en específico. Estos paquetes son generados a partir de herramientas de software como: archivadores y compresores. A continuación se mencionan algunos ejemplos de estas herramientas:

- Para la agrupación de archivos: **GNU *tar***, ***ar***.
- Para la compresión de archivos: ***gzip***, ***bzip***, ***bzip2***, ***xz*** (que reducen el tamaño del paquete con la finalidad de optimizar su distribución).

Cada formato de empaquetamiento se distingue por una extensión de archivo, de entre las más conocidas se encuentran: ***pkg.tar.xz***, ***.deb***, ***.rmp***, ***.tgz***, ***.tar.gz***.

Desde el punto de vista de desarrollo, este esquema permite contar con un sistema GNU/Linux distribuible, además los desarrolladores puede llevar un registro cuantificable de los programas construidos para el sistema GNU/Linux, concentrarlos en un repositorio y disponer de ellos.

Para el caso Beakos GNU/Linux se decidió utilizar **.tgz** por ser un formato sencillo que cumple con su objetivo de agrupar y comprimir, además de que sus herramientas de gestión son sencillas de compilar. Las herramientas o gestores, permiten la instalación/desinstalación, configuración y actualización de los paquetes.

1.2.2 *Elección de un gestor de paquetes, para definir un estándar en la paquetería de Beakos GNU/Linux*

Una vez que los programas compilados ya se encuentren empaquetados con la extensión de archivo **.tgz**, va a ser necesario utilizar una **herramienta de administración de paquetes** o un **gestor de paquetes** que permita a los desarrolladores instalar, desinstalar, actualizar, configurar, listar, agrupar etc., los paquetes de software de Beakos GNU/Linux, todas estas funcionalidades son

características de las **herramientas de administración de paquetes**, sin embargo, la desventaja de estas es que no resuelven de forma automática, las dependencias previas (otros programas) que requiriere un programa de software para ser instalado; por ello se recurre a un **gestor de paquetes**. Dentro de sus principales características se encuentran:

- Trabaja sobre una estructura de directorios válida, esto de acuerdo al formato de empaquetamiento que se esté utilizando.
- Gestiona la manipulación de los paquetes de software.
- La mayoría reconoce únicamente una extensión de formato de empaquetamiento en específico.
- Mantiene actualizado el servidor de repositorios.
- Permite realizar una actualización entre las versiones de paquetes que se encuentran en el servidor de repositorios, con la versión de los paquetes instalados en un sistema operativo GNU/Linux (cliente).
- Resuelve las dependencias, ya que instala los paquetes de software necesarios para la correcta instalación de un programa.

Es decir, que existe una estrecha relación entre el **formato de empaquetamiento** y el **gestor de paquetes**, ya que dependiendo de la extensión (formato) de los paquetes, se determinará el gestor que los manipulará.

Se decidió incorporar ***swaret*** como el **gestor de paquetes** para Beakos GNU/Linux por las siguientes razones:

- Es un gestor sencillo y ligero, ofrece funcionalidades importantes para la gestión de paquetes en servidores.
- Resuelve dependencias.
- Se cuenta con suficiente documentación acerca de la herramienta, lo cual permite su compilación y posterior administración.

- Tiene un desarrollo independiente, es decir, que no está estrechamente ligado una distribución en particular, como es el caso de ***apt-get*** y ***aptitude*** con **Debian** o ***yum*** con **Red Hat**. En este caso dichos gestores siguen la línea de desarrollo de la distribución que los mantiene lo cual influye en el desarrollo de sus distribuciones derivadas.

Posteriormente se trabajó en el desarrollo de un *Front-End* para la gestión de paquetes a través de ***swaret*** en la versión de Escritorio de Beakos GNU/Linux. Tres elementos como son: el **formato de empaquetamiento,** las **herramientas de administración de paquetes** y el **gestor de paquetes** conforman el **Sistema de Empaquetamiento** de Beakos GNU/Linux.

Cuadro 3.1. **Sistemas de Empaquetamiento para distribuciones GNU/Linux**

Formato de Empaquetamiento	Herramienta de Administración de Paquetes	Gestor de paquetes compatible	Distribuciones GNU/Linux que los implementan
.tgz	Installpkg	Swaret	Beakos, Slackware
.deb	Dpkg	apt-get , aptitude	Debian , Ubuntu, Mint y sus derivados
.rpm	Rpm	yast, yum, urpmi	Open SuSe, SuSe Enterprise Linux, Red Hat Enterprise Linux, CentOS, Fedora, Madriva
.pkg.tar.xz	installpkg	Pacman	Arch Linux

1.2.3 *Acerca del repositorios oficial de la Distribución*

Desde la construcción del sistema base de Beakos GNU/Linux, fue indispensable contar con un servidor de repositorio para ir colocando los nuevos paquetes de software compilados, debido a que cada desarrollador se enfocaba en construir paquetería de software para un servicio en específico: seguridad, redes, gráficos, *kernel*, etc., y a fin de evitar que más de un desarrollador compilara la misma herramienta o utilería de software, se subían los paquetes nuevos hacia el servidor de repositorio, posteriormente a través del gestor de paquetes se buscaba si una herramienta o utilería necesaria ya se encontraba disponible.

Esto además de concentrar el desarrollo de nuevos paquetes de software, permitía ir llevando un control sobre las versiones de paquetes disponibles para Beakos GNU/Linux.

1.3.2 *Clasificación de la paquetería de software de Beakos GNU/Linux*

Una vez que se tuvieran los programas de software compilados y bajo un sistema de empaquetamiento, estos pueden ser colocados en el servidor de repositorio para que posteriormente sean administrados con ***swaret***. Dentro del repositorio es necesario clasificar los paquetes de software de acuerdo al tipo de aplicación al que pertenecen. Para Beakos GNU/Linux se designó la siguiente clasificación de paquetes dentro del repositorio:

- **Aplicaciones.-** Contiene utilerías y herramientas de software variadas: editores de texto, IDE de desarrollo, emuladores, y herramientas complementarias para el sistema operativo.
- **Compiz.-** Contiene todos los paquetes necesarios para la instalación de este manejador de gráficos.
- **Databases.-** Contiene los manejadores de Base de Datos soportados por la distribución y las bibliotecas correspondientes a cada manejador, para su correcta instalación.
- **Developer.-** Contiene herramientas de software para desarrollo: compilador, utilerías de software necesarias en la instalación de otras aplicaciones desarrollo para (dependencias), además de bibliotecas para la construcción de cualquier paquete relacionado con gráficos, audio, ofimática, seguridad, actualización de *kernel*, entre otras.
- **Gnome-desktop.-** Contiene a su vez una clasificación de sub carpeteas, donde se encuentra la paquetería necesaria para la instalación del ambiente gráfico de *GNOME*.
- **Games.-** Contiene juegos destinados a la versión de escritorio de Beakos GNU/Linux.

- **Libs.-** Contiene las bibliotecas que integran el sistema base de Beakos/GNU Linux.
- **Network.-** Agrupa paquetes entre bibliotecas y utilerías que trabajan sobre los protocolos de comunicación para la correcta instalación y configuración de programas destinados a la gestión de redes. También contiene *drivers* para el reconocimiento y funcionamiento de tarjetas de red.
- **Security.-** Contiene bibliotecas básicas para poder configurar servicios y aplicaciones de seguridad como *firewalls*, *snifers*, encriptado de datos, firma cifrada.
- **Servers.-** Contiene los paquetes de software que actúan sobre los servicios tipo cliente-servidor más comunes: *httpd, nfs, samba, ftp, vsftpd, openldap, postfix, openssh, sendmail, webmind,* etc.
- **Xorg-system.-** Contiene los paquetes que conforman el sistema de gráficos **X11** de código abierto, es la base para construir e instalar cualquier sistema gestor de ventanas o un entorno de escritorio.

Es importante esta clasificación ya que así se tiene un mayor control sobre la paquetería disponible y para el caso de desarrollo, esta agrupación de paquetes facilita la ubicación de los programas necesarios para la construcción de un software en específico, por ejemplo, para la construcción de cualquier entorno de escritorio, es necesario instalar el grupo de paquetes que se encuentra en ***Xorg-system***.

2. Desarrollo de Beakos GNU/Linux siguiendo el método de LFS

2.1 Introducción

Dentro del desarrollo del sistema se tomaron en consideración las notas de este apartado las cuales se resumen en decir que ***Linux From Scratch*** es un método de construcción de distribuciones GNU/Linux desde cero, a pesar de que guía paso a paso la construcción de una distribución GNU/Linux, asume que el usuario

cuenta con los conocimientos necesarios para comprender las instrucciones y comandos ejecutados durante el proceso de desarrollo. Para la construcción de Beakos GNU/Linux se decidió utilizar el **LFS versión 6.4** por ser una versión estable, y durante los temas posteriores se explicará de forma general cuales fueron las configuraciones necesarias que aplican para esta versión.

Para un mayor detalle durante este capítulo, es conveniente definir los siguientes conceptos que se emplearan subsecuentemente:

Sistema GNU/Linux físico.- Es la distribución que se encuentra instalada en el equipo de cómputo sobre la cual se está trabajando para montar el ambiente de desarrollo y posteriormente construir el nuevo sistema LFS.

Sistema LFS.- Es el ambiente donde se prepara y desarrolla el nuevo sistema Beakos GNU/Linux, el cual se encuentra contenido en una partición de disco independiente dentro del Sistema GNU/Linux físico.

Por otra parte para facilitar la lectura de la documentación técnica empleada en este capítulo, es conveniente definir el estilo de documentación utilizado, que detalla los comandos, líneas de código y en algunos casos los resultados obtenidos en pantalla.

a) La documentación técnica está estructurada en el siguiente orden:

```
<comandos>
…
<líneas de código>
…
<resultados en pantalla>
…
```

b) La ejecución de cualquier comando empleado en este apartado se realiza utilizando el usuario ***root***, el cual siempre se va a denotar tal y como se mostraría en una terminal o intérprete de comandos de Beakos GNU/Linux:

```
root[~]# <comando empleado>
```

c) Se especificarán las opciones del comando empleado durante los procesos de construcción y configuración de Beakos GNU/Linux:

```
root[~]# ./configure --prefix=usr/local
```

d) En los casos donde sea pertinente, se resaltarán las líneas de código de algunos scripts o los resultados en pantalla arrojados por determinado comando; se copiaran únicamente las líneas convenientes, la denotación de tres puntos seguidos (…) hace referencia a que existen más líneas código que se omitieron en esta documentación.

```
root[~]# ./configure --prefix=/usr/local
…
config.status: creating Makefile
config.status: creating libpng.pc
config.status: creating libpng-config
config.status: creating config.h
config.status: executing depfiles commands
config.status: executing libtool commands
```

2.2 Preparar el ambiente de Desarrollo

Para la construcción de Beakos GNU/Linux, se instaló sobre otra distribución GNU/Linux todo el ambiente de desarrollo que permitiera la compilación del código fuente de cada programa necesario, es decir, un sistema GNU/Linux permite incorporar las herramientas de software iniciales para preparar un ambiente de desarrollo:

- *Kernel* de *Linux*,
- Intérprete de comandos o *Shell*,
- Herramientas para la gestión de discos,
- Herramientas de compresión,
- Monitor de procesos,
- Compiladores,
- Editores de texto.

Dicha distribución es de manera indistinta y no implica en el desarrollo de la nueva distribución, siempre y cuando cuente con dichos elementos. Ya que únicamente se toman las herramientas generales que incorpora cualquier sistema GNU/Linux para poder generar un ambiente de desarrollo.

En este paso se configuró e instaló lo siguiente:

a) Se designó una nueva partición de 3 GB, en el sistema GNU/Linux físico, que es donde se instalarán todos los programas del nuevo sistema LFS.- Es necesario utilizar una herramienta de software para la gestión de discos como ***fdisk***.

b) Se creó el sistema de archivos sobre la nueva partición.- Es necesario utilizar la herramienta de software ***mkfs.ext3*** para dar formato a la nueva partición.

c) Se montó la nueva partición con permisos de lectura y escritura.- Es necesario utilizar la herramienta ***mount*** para declarar en el sistema GNU/Linux físico la nueva partición creada y poder acceder a ella:

- /mnt/lfs

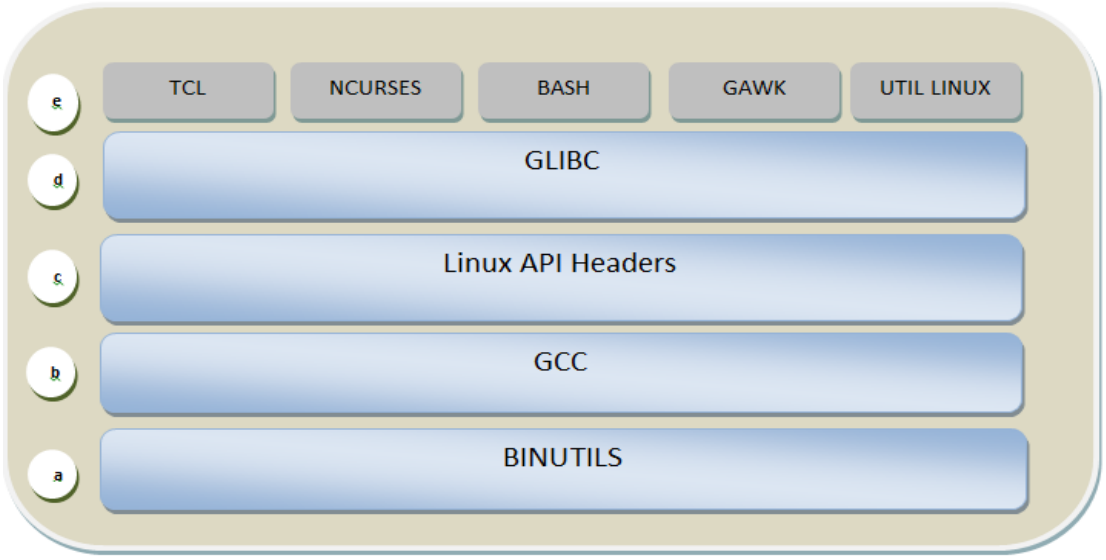
Posteriormente, una vez realizadas las configuraciones anteriores, se creó un usuario dentro del sistema GNU/Linux físico y se realizaron configuraciones en el archivo ***fstab*** para mantener dichas configuraciones de forma permanente.

2.2.1 *Construcción de un Sistema Linux Primario (temporal)*

Una vez generada la partición destinada al sistema LFS y echas las configuraciones pertinentes, se preparó un sistema Linux con características mínimas, sobre el cual se desarrolló posteriormente el sistema LFS por completo. Esto es necesario, ya que crear este primer sistema Linux de forma temporal permite realizar configuraciones e instalaciones de las herramientas de software que se utilizarán a lo largo del desarrollo de Beakos GNU/Linux (o de cualquier otra distribución GNU/Linux).

Además de que así lo indica el método del LFS, de esta forma garantiza que la construcción del sistema se lleve a cabo de forma correcta.

Gráfico 3.2 **Programas de software indispensables, que se compilaron para obtener el sistema Linux primario**



a) Construcción de las ***Binutils***, es un paquete que contiene un enlazador, ensamblador y otras herramientas para la manipulación de archivos y objetos.

b) Construcción de ***GCC*** (*GNU C Compiler*), es una colección de compiladores del proyecto GNU, que incluye un compilador de **C** y **C++**.

c) Construcción de las ***Linux API Headers***, que permiten implementar los estándares de los sistemas tipo UNIX, expuestos anteriormente: POSIX, LSB, etc. A través de un API de desarrollo que el Kernel de Linux pueda interpretar y con el cual se pueda comunicar para la gestionar los componentes de *hardware*.

d) Construcción de ***Glibc***, contiene la biblioteca principal del lenguaje C y es la biblioteca portable perteneciente al proyecto GNU utilizada para proporcionar todas las rutinas básicas y llamadas al sistema (*open*, *read*, *write*, *close*, *kill*, entre otras), que permitan la interacción entre componentes de *hardware* como el CPU y su arquitectura. ***Glibc*** está apegado a los estándares de UNIX tales como: ***POSIX*** e ***ISO C***.

e) Construcción de programas de **software adicionales**, se compila una variedad de programas que integran al **sistema Linux Primario**; el listado

completo de los programas de software se encuentra en los anexos de este documento.

Es importante que se compilen los paquetes en el mismo orden que se indican en el gráfico 3.2 ya que van cubriendo dependencias que son necesarias para los paquetes posteriores; además es necesaria la aplicación de los parches adecuados que indica el LFS 6.4.

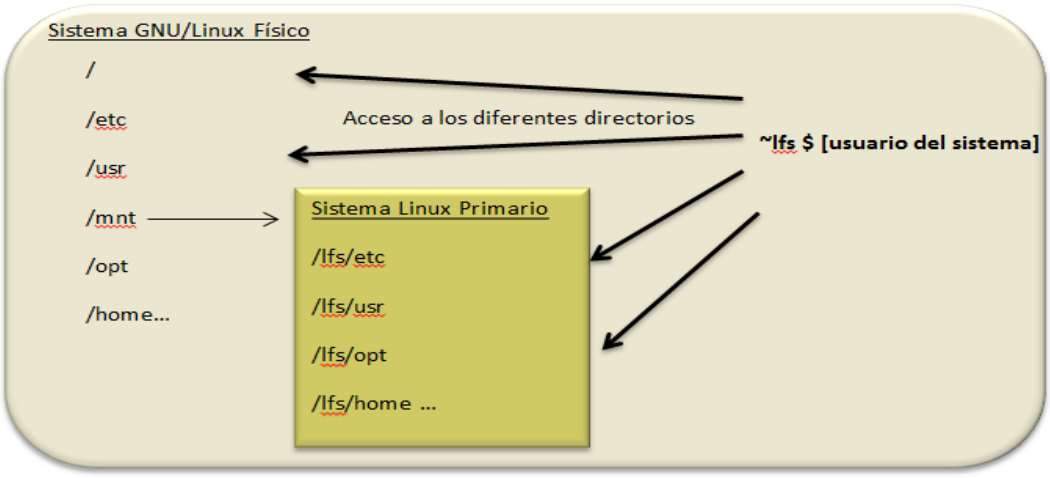
El método del LFS indica las opciones con las que se debe configurar cada programa de software en el ***configure***, sin embargo, para la construcción de Beakos GNU/Linux se optó por incluir o quitar ciertas opciones para algunos paquetes, es aquí donde interviene la decisión de los desarrolladores de la distribución para personalizar la construcción de su sistema GNU/Linux.

Hasta este momento ya se tenía el **sistema Linux Primario** con los componentes mínimos y dentro de dicho sistema comienza la construcción del sistema LFS, que posteriormente incluiría paquetes de software adicionales para la configuración y personalización de la primera versión de Beakos GNU/Linux.

2.2.2 *Construcción del sistema LFS sobre un sistema Linux Primario (chroot)*

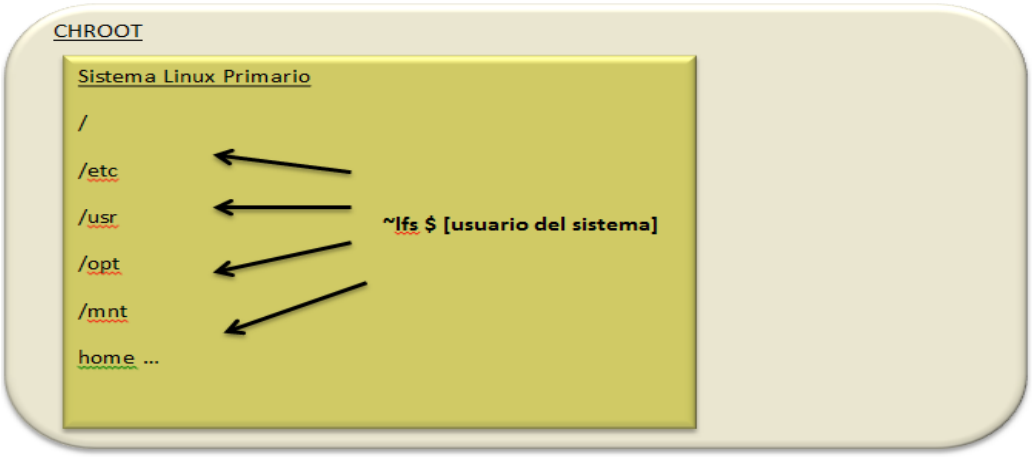
Para la construcción del sistema LFS, se trabajó dentro de la partición **/mnt/lfs**, donde se había construido y configurado el entorno Linux con características mínimas; se utilizó un usuario del sistema para ejecutar la herramienta ***chroot*** y "*enjaular"* al usuario dentro del sistema Linux Primario, es decir, ***chroot*** permite colocar un usuario del sistema GNU/Linux Físico dentro del sistema LFS instalado en la partición **/mnt/lfs** haciendo que esta partición sea vista como la **/ raíz** de un sistema Linux, por lo tanto el usuario ya no puede escalar hacia los directorios del sistema GNU/Linux Físico.

Gráfico 3.3. **Estructura de directorios del Sistema Linux Primario**



El Sistema Linux primario contiene una estructura de sistema de archivos como cualquier sistema Linux. Aquí el usuario del sistema GNU/Linux Físico, puede acceder a todos los directorios y subdirectorios existentes, sin embargo el siguiente diagrama muestra lo que sucede cuando se aplica ***chroot***:

Gráfico 3.4**. Estructura de directorios con chroot en Sistema Linux Primario**



2.3 Construcción del Sistema LFS

Con esto, se pudieron aplicar las configuraciones al **Sistema Linux Primario** siguiendo la documentación del LFS; posteriormente se continuó con la construcción de toda la paquetería que conformaría al **Sistema LFS** el cual a su vez integraría el **sistema base de Beakos GNU/Linux**.

Como se mencionó al inicio de este trabajo, queda fuera del alcance de este capítulo replicar las configuraciones y comandos aplicados, mismos que se

encuentran dentro de la documentación del LFS versión 6.4; sin embargo, hasta el momento se ha estructurado de forma general los elementos y configuraciones principales para obtener un Sistema LFS funcional, el cual representa un hito en el desarrollo del proyecto.

2.3.1 *Creación de los Scripts de arranque del sistema*

Posterior a la construcción de toda la paquetería del **sistema LFS**, se realizó la configuración de los scripts del sistema, estos permiten de manera general definir qué servicios o procesos se van a ejecutar cuando arranque el **sistema LFS**, también permiten definir configuraciones en el sistema de forma permanente, por ejemplo, la configuración de red, puntos de montaje para cada partición del disco duro, ejecución de tareas programadas, etc.

A continuación se describen los principales scripts que se configuraron para el **sistema LFS** de Beakos GNU/Linuix:

- **ifdown.-** Interactúa con el script de *network* para dar de baja las interfaces de red.
- **ifup.-** Interactúa con el script de *network* para dar de alta las interfaces de red.
- **network.-**Contiene la configuración de la interfaces de red que reconoce el sistema, aquí se especifican los parámetros para que el sistema esté dentro de una red determinada.
- **reboot.-** Realiza un reinicio del sistema, dando de baja los servicios en el orden configurado.
- **swap.-** Contiene el detalle de la partición de intercambio de memoria RAM, aquí se especifica la partición y su tamaño que el sistema reconoce para tomar la memoria de intercambio, también a través de este *script* se activa o desactiva dicha partición.
- **mountfs.-** Monta el sistema de archivos, a excepción de los que se especifiquen como montaje manual.

- **mountkernfs.-** Monta los sistemas de archivos virtuales como **/proc**.
- **modules.**- Carga los módulos del *kernel* que son listados en el archivos **/etc/sysconfig/modules**.
- **sysklogd.-** Arranca o detiene los servicios encargados de registrar los ***logs*** tanto del *kernel* como del sistema en general.

Sin estos scripts el usuario tendría que realizar cada una de las configuraciones necesarias al momento que inicie su **sistema LFS**, estas configuraciones se guardarían de manera temporal y se perderían al momento de apagar o reiniciar el sistema.

Cada uno de estos scripts se ejecuta de acuerdo a los niveles de ejecución donde son colocados, Linux implementa los siguientes niveles:

Nivel 0: Sistema Apagado

Nivel 1: Sistema en Mono usuario.- El sistema arranca con súper usuario (*root*).

Nivel 2: Sistema en Multiusuario.- El sistema arranca para usuarios múltiples, pero sin levantar los servicios de red.

Nivel 3: Sistema en Multiusuario con servicios de red. El sistema arranca para usuarios múltiples y levanta los servicios de red.

Nivel 4: No utilizado [reservado].

Nivel 5: Sistema en ambiente gráfico. El sistema arranca con todos los servicios anteriores además del ambiente gráfico.

Nivel 6: Reinicio del Sistema.

Dentro de Beakos GNU/Linux el archivo de configuración para cada uno de estos niveles de ejecución se encuentra dentro de:

/etc/rc.d/

2.3.2 *Hacer el sistema LFS arrancable*

Para lograr que el sistema pueda arrancar al momento de encender el equipo de cómputo donde sea instalado, es necesario generar los scripts adecuados e

instalar ciertas herramientas siguiendo aún la base del LFS, hay tres elementos que se deben considerar para lograr que el sistema pueda arrancar:

- Crear el archivo de configuración **/etc/fstab**.
- Construir un *kernel* destinado al nuevo **sistema LFS** (hay que recordar que el *kernel* que se tiene corresponde al que se construyó para el **sistema Linux Primario** y este cuenta con componentes básicos.)
- Instalar un cargador de arranque, conocido como ***grub***.

2.3.2.1 Crear el archivo /etc/fstab

El archivo ***fstab*** es utilizado por algunos programas para determinar donde se montarán los sistemas de archivos al arranque del sistema operativo; permite definir un orden secuencial de montaje, así como revisar la integridad de los datos de sistema de archivos que se montará, en busca de errores.

Con un editor de texto como ***vi*** se creó el archivo ***fstab*** para Beakos GNU/Linux de la siguiente forma:

Cuadro3.2. **Creación del archivo fstab para el sistema Beakos GNU/Linux**

```
root[~]# cat >/etc/fstab << "EOF"

# Begin /etc/fstab
# file system   mount-point    type          options          dump    fsck
# order
/dev/sda1             /          ext4          defaults         1       1
/dev/sda2             swap       swap          defaults         0       0
proc                  /proc      proc          defaults         0       0
sysfs                 /sys       sysfs         defaults         0       0
devpts                /dev/pts   devpts    gid=5,mode=620       0       0
tmpfs                 /run       tmpfs         defaults         0       0
# End /etc/fstab
EOF
```

Detalle del archivo creado o modificado:

File System: En esta columna va el nombre del sistema de archivos que se va a montar.

Mount-point: En esta columna va la ruta absoluta donde se montará el sistema de archivos.

Type: Aquí se define el tipo al que pertenece el sistema de archivos, para mayor referencia acerca de los diversos tipos de sistemas de archivos, se recomienda consultar la documentación propia de ***fstab***.

Options: En esta columna se definen parámetros de opciones con los que se montaran los sistemas de archivos, comúnmente para las particiones declaradas por el usuario dentro de este *script* suele dejarse la opción "***defaults***". Pero existen múltiples opciones de montaje por lo que se recomienda consultar la documentación de ***fstab***, para mayor referencia.

Dump: Esta opción se habilita en caso de que se requieran realizar copias de seguridad de los sistemas de archivos.

Fsck: En esta columna se ponen parámetros únicamente de 0 y 1, cero indica que no se va a realizar un chequeo al sistema de archivos y uno para indicar que se revisará la partición en busca de posibles errores en la integridad de los datos antes de montarse.

Tanto ***sda1*** como ***sda2*** son particiones del disco que se crearon con anterioridad de forma manual, el resto de los sistemas de archivos son generados por el sistema Linux.

2.3.2.2 Construir un kernel destinado al Nuevo sistema LFS

Debido a que la construcción de un núcleo de sistema operativo queda fuera del alcance de este trabajo, se expondrá de forma general el procedimiento a seguir para la construcción de un **nuevo *kernel*** que se empleó para el **sistema LFS**.

Para la construcción del núcleo comprenden los siguientes pasos:

1. Configuración,
2. Compilación,
3. Instalación.

El siguiente comando prepara el ambiente de construcción del núcleo:

```
root[~]# make mrproper
```

Con esto se asegura que el árbol del núcleo se encuentra limpio de errores que pudieron presentarse al momento de su descarga y desempaquetado.

Posteriormente se debe continuar con el proceso de configuración a través de cualquiera de las siguientes interfaces de usuario:

a) **make menuconfig:** Es una aplicación basada en ***ncurses*** (sencillas ventanas gráficas) el cual despliega un árbol de opciones a escoger, lo que representa el menú de configuraciones que se pueden realizar.

b) **make xconfig:** Es una interfaz de usuario la cual proporciona un entorno con más elementos gráficos, permite la navegación por el menú de opciones desde el cursor del *mouse*, a diferencia de ***menuconfig*** donde la navegación es con las flechas del teclado.

c) **make oldconfig:** Es una interfaz de línea de comando que copia la configuración del archivo ***.config*** de un *kernel* que ya se encuentra compilado migrando los datos del archivo ***.config*** a la configuración del *kernel* en construcción.

Para el caso particular de Beakos GNU/Linux se utilizó ***menuconfig***; una vez realizadas las configuraciones, hay que considerar los módulos que serán habilitados para el *kernel*. Comúnmente las distribuciones GNU/Linux enfocadas a usuarios finales incorporan suficientes módulos garantizando que el sistema pueda reconocer gran variedad de dispositivos de hardware instalados en el amplio catálogo de equipos de cómputo, un ejemplo de esto es *Ubuntu*. Si bien es una ventaja el incorporar mayor variedad de módulos al *kernel*, también implica un

mayor consumo de recursos de hardware desde el arranque del sistema operativo y durante su ejecución, lo cual demanda tener suficientes recursos de procesamiento para que la experiencia del usuario con la distribución sea de forma fluida y óptima.

La ventaja que le representa a Beakos GNU/Linux en este sentido, es que su instalación se efectúa de forma rápida, al igual que el arranque del sistema, comparado con otras distribuciones como: *Ubuntu*, *Redhat*, *Fedora*, etc. Esta ventaja es notable en ambientes productivos cuando se debe realizar una instalación depurada y personalizada en equipos de hardware que requieran aprovechar más sus recursos para la ejecución de sus aplicativos que para la ejecución del propio sistema operativo, también al momento de una caída del sistema es importante que este se pueda reestablecer a la brevedad posible.

Como se mencionó a inicios de este trabajo, uno de los objetivos de Beakos GNU/Linux es ser una distribución ligera, que cuente únicamente con los componentes necesarios para un ambiente productivo, por lo que se incorporaron únicamente los módulos de *kernel* indispensables, como los que se describen en el siguiente cuadro.

Cuadro 3.3. **Módulos básicos de *kernel* en el sistema base de Beakos GNU/Linux**

Módulo	Utilizado para
VGA, Nvidia	Gráficos/Video
Snd_pcm_oss	Tarjetas de audio genéricas
ipv4, ipv6	Protocolos de comunicación de red
Iptables, netfilter	Firewall a nivel lógico

El listado completo de módulos incorporados en Beakos GNU/Linux se encuentra en los anexos de este trabajo.

Una vez que se incorporaron los módulos necesarios, hay que realizar la configuración para las distintas secciones del *kernel*, las cuales se clasifican en:

a) **Configuración general:** Ofrece la configuración de opciones generales al *kernel*.

b) **Habilitar el soporte para la carga de módulos:** Permite la carga de módulos adicionales una vez que el sistema se encuentra instalado y ejecutando.

c) **Activar la capa de bloque:** Esto permite que se puedan montar unidades de disco en el sistema.

d) **Tipo de procesador y sus características:** Permite definir qué tipo de procesadores soportará el *kernel* (**x86** y/o **x64**).

e) **Opciones de gestión de energía:** Para el control de los dispositivos de hardware encargados de administrar el consumo y ahorro de batería, esta configuración aplica para los dispositivos portátiles.

f) **Soporte para red:** Aquí es donde se habilitan las funciones de red incluidas las inalámbricas, las opciones de firewall también se definen en esta sección.

g) **Controladores de dispositivos:** Aquí se definen que controladores se van a incorporar al *kernel* (video, audio, red, usb, etc.)

h) **Sistemas de archivos:** Permite definir qué sistemas de archivos estarán disponibles: ***ext3***, ***ext4***, ***jfs***, ***xfs***, soporte para ***automounter***, ***nfs***.

i) **Virtualización:** Permite definir si el *kernel* tendrá soporte para trabajar con hipervisores que generen máquinas virtuales dentro del sistema operativo.

La construcción de un núcleo implica una toda una serie de pasos y configuraciones con diversas opciones, las cuales son empleadas por cada desarrollador de acuerdo a sus propósitos de uso. Para mayor referencia en cuanto a la construcción detallada del núcleo para un sistema base se puede consultar la documentación de ***Linux From Scratch*** *(LFS)*.

Sin embargo, en los siguientes apartados se exponen las características del sistema Beakos GNU/Linux, como resultado final de todo el proceso de construcción del LSF incluyendo la compilación del nuevo núcleo, también dentro de los anexos de este documento se encuentra el listado de toda la paquetería que comprende al sistema base de Beakos GNU/Linux.

2.3.2.3 Instalar un cargador de arranque, GRUB

Una vez que se tuvo preparado el *kernel* para el sistema base, se continuó con la instalación de un gestor de arranque múltiple, mejor conocido como **GRUB** (***GNU Grand Unified Bootloader***). Sin este programa la ejecución de un sistema Linux no se podría realizar de forma automática, además de que se carecería de opciones para distintos tipos de arranque, sin embargo, si se podría arrancar un sistema Linux ya instalado, pero implicaría una serie pasos técnicos además de que es poco funcional.

Grub utiliza una nomenclatura de la siguiente forma, para nombrar a los discos instalados:

(hdn, m)

dónde:

- **hd**, hace referencia a ***hard disk*** (disco duro),
- **n**, se refiere al número del disco, cero para el primer disco duro reconocido,
- **m**, se refiere al número de la partición del disco duro.

El cargador de arranque escribe sobre la primera pista física del disco duro, esta sección se encuentra fuera de cualquier sistema de archivos configurado, antes de iniciarse el sistema operativo accede a la información del cargador de arranque la cual se encuentra en la ruta:

- /boot/grub/

Se recomienda que la instalación del ***grub*** se efectúe dentro del directorio ***/boot*** , el cual es montado en una partición distinta a la partición donde se encuentra el sistema de archivos ***/raíz***, para Beakos GNU/Linux se generó una nueva partición de 100 MB (de igual forma es el tamaño recomendado para la partición que alojará los datos del directorio y subdirectorios de ***/boot***), las instrucciones y los pasos a seguir para la creación de una nueva partición en el disco se detallan en los anexos de este documento. Una vez que se tiene la partición destinada al cargador de arranque se continúa con su instalación:

- root[~] # grub-install /dev/sda1

Posteriormente con un editor de texto, se genera el *script* de configuración propio del cargador.

Cuadro3.4. **Instalación del Cargador de Arranque**

Entorno de Desarrollo: ***Sistema LFS***
Comando ejecutado: **cat > /boot/grub/grub.cfg << "EOF"**
Posición actual donde se ejecuta el comando: ***/root***
Ruta absoluta del archivo creado o modificado: ***/boot/grub/grub.cfg***
Código:

```
root[~]# cat > /boot/grub/grub.cfg << "EOF"
# Begin /boot/grub/grub.cfg
set default=0
set timeout=5
title           beakos GNU/Linux, kernel 2.6.34
root            (hd0,1)
kernel          /vmlinuz-2.6.34 root=UUID=f8b1af89-5a75-48a7-
8680-6ee56269c3f0 ro vga=788
initrd          /initrd-2.6.34
### END BEAKOS AUTOMAGIC KERNELS LIST
EOF
```

Con esto se concluyó la construcción de un **sistema LFS**, posteriormente se hizo un reinicio del sistema para comprobar el arranque sin errores y con todos los componentes compilados.

2.3.2.4 Pruebas del Sistema

Se realizaron pruebas para validar la correcta funcionalidad del sistema:

- En el Reinicio del sistema se validó,
  - Arranque del ***grub***,
  - Reconocimiento del kernel construido (2.6.34),
  - Montaje de las unidades de disco instaladas,
  - Inicio en automático de los principales servicios configurados: red, seguridad, correo electrónico, impresión, conexiones remotas, bitácora de actividades (logs).

Una vez iniciado el sistema base de Beakos GNU/Linux sin errores, se continuó con el proceso de validación:

- Dentro del Sistema de archivos se validó:
  - Reconocimiento de las particiones generadas en el disco,
  - Tamaño correcto destinado a cada una de las particiones,
  - Montaje correcto para las unidades: ***/* (raíz)**, /***boot*** y swap en las particiones del disco definidas en el archivo ***fstab***.

- Para la memoria física (RAM) del sistema, se validó:
  - Reconocimiento total de la memoria RAM instalada,
  - Reconocimiento total del espacio destinado a la memoria swap,

- Para los módulos del *kernel* se validó:
  - Que se tuvieran instalados los principales módulos: video, audio, red, seguridad, disco.
  - Reconocimiento de los dispositivos de hardware instalados.

Posterior a la validación general del sistema base de Beakos GNU/Linux, se contaba con un sistema sobre el cual se podían construir nuevos desarrollos; en el siguiente apartado se expondrán las características y alcances de dicho sistema.

3. Principales características del Sistema Base de Beakos GNU/Linux

Una vez realizadas las configuraciones anteriores y validado la funcionalidad del sistema en este momento ya se tiene un sistema funcional basado en LFS, con comandos y herramientas para su administración, el alcance de dicho sistema hasta ese momento era:

- Instalación del sistema LFS soportado en arquitecturas x86 a 32 bits,
- *Kernel* de Linux 2.6.34 con los módulos especificados en el anexo de este documento,

- Gestor de arranque que además de cargar el sistema LFS, permite seleccionar el sistema operativo de inicio en caso de tener más de un sistema instalado en un equipo de cómputo,
- Permite la instalación del sistema de forma manual,
- Intérprete de comandos ***bash***,
- Editor de texto ***vim***,
- Soporte para distintos idiomas y caracteres,
- Scripts de arranque y configuración del sistema, **/etc/init.d**.
- Incorpora las herramientas de desarrollo (***gcc***, ***gnu make***, ***make install***) y bibliotecas para compilar programas de software adicionales.

De forma general este es el alcance de todo sistema *LFS*, sin embargo, para el desarrollo del "**sistema base**" de Beakos GNU/Linux, se consideraron configuraciones y desarrollos adicionales para conformar una distribución aún más funcional. Si bien durante la construcción del sistema *LFS* se fueron adecuando ciertas configuraciones tanto en *scritps* como en módulos del *kernel*, lo cual ya daba una orientación hacia una distribución personalizada, la verdadera distribución funcional de Beakos GNU/Linux surgió al momento de incorporar los siguientes desarrollos y que como ya se ha dicho, estos están fuera del alcance del *LFS*:

- *LFS* entrega un sistema el cual debe ser instalado de forma manual ya que no cuenta con un instalador. Sin embargo, para el sistema base de Beakos GNU/Linux se desarrolló un instalador basado en el instalador de ***slackware*** que emplea ***ncurses***.
- Incorpora el sistema de empaquetamiento con formato ***tgz***.
- Incorpora el gestor de paquetes ***swaret***.
- Permite la comunicación con dos servidores de repositorios, uno contiene los paquetes de software que vienen únicamente con los binarios para la instalación y configuración de las aplicaciones, por otra parte está el

repositorio de estos mismos paquetes pero que incluyen el código fuente, bibliotecas entre otros archivos que fungen como herramientas necesarias para la compilación o desarrollo.

- El *Kernel* de Linux 2.6.34 incorpora módulos adicionales para soportar:
    - Drivers de tarjetas inalámbricas de red de la serie Broadcom (BCM43 y BCM43 legacy),
    - Drivers de video de la familia de tarjetas: VGA, ATI RADEON,
    - Drivers de audio.

Hasta aquí se tenía un Sistema Base de Beakos GNU/Linux, a partir del cual se derivó la versión enfocada a cumplir con el objetivo inicial planteado durante el desarrollo del proyecto:

Construir un sistema GNU/Linux ligero para optimizar el desempeño de equipos con hardware limitado, y que le permita al INFOTEC tener control sobre su desarrollo y administración.

En resumen esto se pudo lograr dadas las siguientes justificaciones:

Elementos que integran el objetivo principal	Resultados obtenidos
Construir un sistema ligero para optimizar recursos de desempeño en equipos con hardware limitado.	Una versión de Beakos GNU/Linux para servidores con los siguientes requerimientos mínimos para su instalación: • 256 MB de RAM • 4 GB de espacio en disco duro • Procesador Pentium 4 o superior Dichos recursos son menores en comparación con otras distribuciones GNU/Linux que se tenían instaladas.
Que le permita al INFOTEC tener control sobre su desarrollo y administración.	La construcción basada en LFS permite que el sistema Beakos GNU/Linux siga la línea de desarrollo que mejor se adecue a las necesidades del INFOTEC.

4. Caso práctico: Ejemplo en la construcción de paquetes para incorporar un gestor ventanas sobre el Sistema Base de Beakos GNU/Linux

En este apartado se ejemplifica el proceso de construcción de algunos paquetes de software requeridos para instalar un gestor de ventanas en un sistema base de Beakos GNU/Linux.

4.1 Xorg

Es la implementación libre del sistema de ventanas distribuido ***X Window*** versión 11 y es la base para poder dotar de una Interfaz Gráfica de Usuario (GUI) a través de un gestor de ventanas o un entorno de escritorio, a los sistemas GNU/Linux. ***Xorg*** se ejecuta de manera independiente a:

- Dispositivo de hardware gráfico,
- Arquitectura del equipo de cómputo,
- Sistema Operativo,
- Red o protocolo de conexión,

4.2 Actividades Previas

Cuando se concluyó la construcción del sistema ***Xorg*** por completo fue necesario depurar los paquetes de software para evitar conflicto en las búsquedas a través del gestor de paquetes ***swaret***. Posteriormente los paquetes de ***Xorg*** se subieron al servidor de repositorios y estos fueron clasificados en subdirectorios de la siguiente forma:

Xorg-system/

- **app/:** Se encuentran diversos programas que permiten la interacción de componentes de hardware con el servidor X, por ejemplo compiladores de código fuente, gestores de autorización, gestores de elementos gráficos.
- **dbus/:** Se encuentran los programas que integran a *D-Bus*, que es un sistema para la interacción entre el sistema operativo y las sesiones de usuario abiertas.

- **driver/:** Aquí se colocaron los paquetes relacionados con los controladores para distintos tipos de tarjetas gráficas.
- **font/:** Aquí se colocaron los paquetes relacionados a los tipos de fuentes estándar que implementa *Xorg*.
- **fontconfig/:** Se encuentra la biblioteca para instalar/eliminar las fuentes tipográficas, así como su configuración en las aplicaciones del sistema operativo.
- **hal/:** Aquí se encuentra el programa que pude ser considerado como el driver de la tarjeta madre. Proporciona una capa de abstracción del hardware, a través de la comunicación con el *kernel* del sistema operativo y los componentes físicos del dispositivo de cómputo.
- **lib/:** Aquí se colocaron las bibliotecas que *Xorg* implementa para la comunicación con otras aplicaciones y elementos gráficos de software.
- **libxau/:** Aquí se encuentra una biblioteca la cual es una implementación del Protocolo de Autorización X11 y que es útil para restringir el acceso del usuario a los componentes de la pantalla.
- **mesalib/:** Aquí se encuentra el paquete que es la implementación de software libre de la especificación *OpenGL*, el cual es un sistema para el despliegue de gráficas en 3D.
- **xbitmaps/:** Aquí se encuentra el paquete que proporciona imágenes en mapa de bits, utilizadas por múltiples aplicaciones que se construyen para la instalación de *Xorg*.
- **xcursor-themes/:** Aquí se colocó el paquete que contiene temas para visualizar distintos tipos del ícono desplegado en pantalla para el cursor del mouse.
- **xkeyboardconfig/:** Aquí se encuentran los paquetes que proporcionan una configuración estándar, bien estructurada del teclado gráfico que proporciona el sistema *Xorg*.

- **xorg-server/:** Aquí se encuentra el servidor X.

Esta es la clasificación de los paquetes de ***Xorg*** que se encuentra en el servidor de repositorios de Beakos GNU/Linux.

4.2.1 *Instalación de Herramientas de Desarrollo*

A partir de un sistema base Beakos GNU/Linux el cual incorpora las herramientas y programas de software ya mencionados con anterioridad y que de forma general es un sistema administrable únicamente vía línea de comando, el usuario puede incorporar paquetería adicional para la personalización de la distribución (el uso de las principales opciones de ***swaret*** se describe dentro de los anexos de este documento). En los siguientes apartados de este documento se trabajará sobre la construcción de un gestor de ventanas con aplicaciones de uso general.

En la configuración por defecto dentro del sistema base de Beakos GNU/Linux, ***swaret*** únicamente lee el servidor de repositorios que contiene los paquetes ordinarios los cuales contienen archivos: ejecutables, manuales y algunas bibliotecas, es decir, aquí no se encuentran paquetes con su código fuente, sin embargo, para iniciar con la construcción de un entorno gráfico para Beakos GNU/Linux, (o la compilación de cualquier otro programa) hay que habilitar para ***swaret*** la lectura del servidor de repositorios que contiene los paquetes con su código fuente, es decir los paquetes que en Beakos se identifican por llevar ***devel*** en el nombre del paquete.

Por ejemplo, si se busca el siguiente paquete con el servidor de repositorio que trae habilitado por defecto el sistema base:

```
root[~]# swaret --search make

swaret 1.6.3-2

make-3.81-i386-1 (300 kB) (BEAKOS_PACKAGES) [Status: NOT INSTALLED]
```

Únicamente se muestra el paquete ordinario de ***make-3.81-i386-1***; para habilitar el repositorio de desarrollo hay que modificar el archivo de configuración ***swaret.conf*** que se encuentra dentro de /etc.

Cuadro3.5. **Archivo de configuración swaret.conf**

```
root[~]#vi /etc/swaret.conf

##############################################################
#                                                            #
# swaret.conf, Version: 1.6.3                                #
#                                                            #
### /etc/swaret.conf - SWARET EXAMPLE CONFIGURATION FILE###
...

ROOT=http://www.beakos.com.mx/repo-balam/beakos-1.7

#Repository Development Officer:
#ROOT=http://www.beakos.com.mx/repodevel-balam/beakos-1.7
…

REPOS_ROOT=BEAKOS_PACKAGES%http://www.beakos.com.mx/repo-
balam/beakos-        1.7

#Repository Development Officer
#REPOS_ROOT=BEAKOS_PACKAGES_DEV%http://www.beakos.com.mx/repode
vel-balam/beakos-1.7
…

DEP_ROOT=http://www.beakos.com.mx/repo-balam/beakos-1.7

#DEP_ROOT=http://www.beakos.com.mx/repodevel-balam/beakos-1.7
```

Es necesario quitar el comentario únicamente a las líneas que hacen referencia al servidor de repositorio de desarrollo (***devel***) quitando el símbolo de #:

- `ROOT=http://www.beakos.com.mx/repodevel-balam/beakos-1.7`
- `REPOS_ROOT=BEAKOS_PACKAGES_DEV%http://www.beakos.com.mx/repodevel-balam/beakos-1.7`
- `DEP_ROOT=http://www.beakos.com.mx/repodevel-balam/beakos-1.7`

Hay que guardar los cambios realizados, posteriormente se debe actualizar el gestor de paquetes (***swaret***):

```
root[~]# swaret --update
```

Una vez habilitado el servidor de repositorio con los paquetes para desarrollo (devel), nuevamente buscamos el paquete ***make***:

```
root[~]# swaret --search make

swaret 1.6.3-2

make-3.81-i386-1 (300 kB) (BEAKOS_PACKAGES) [Status: NOT INSTALLED]

make-devel-3.81-i386-1 (247 kB) (BEAKOS_PACKAGES_DEV) [Status: NOT INSTALLED]
```

Con esto vemos que ***swaret*** ya arroja los resultados de la búsqueda incluyendo los paquetes que se encuentran en el repositorio ***devel***: ***make-devel-3.81-i386-1***.

Por último, es necesario instalar las herramientas de desarrollo básicas para la compilación y construcción de cualquier paquete de software en Beakos GNU/Linux, dichas herramientas se encuentran en los siguientes paquetes y se instalan desde ***swaret***:

1) ***build-essentials***, este paquete contiene una lista genérica de programas necesarios para la construcción de otros paquetes de software:
    a. make-3.81: Programa para compilar paquetes de código fuente.
    b. mpfr-2.3.2: Contiene funciones de precisión múltiple.
    c. gmp-4.2.4: Contiene bibliotecas matemáticas.
    d. glibc-devel-2.8 : Contiene un lenguaje de procesamiento numérico.
    e. linux-headers-2.6.34: Headers del Kernel 2.6.34.
    f. binutils-2.18: Contiene un enlazador, un ensamblador y otras herramientas para manejar archivos object.
    g. gcc-devel-4.3.2: Colección de compiladores GNU, C y C++.
2) ***automake-devel-1.10.1-i386-1*:** Es una herramienta para generar automáticamente archivos ***Makefile.in*** compatibles con los estándares de codificación GNU.

3) ***cmake-devel-2.8.2-i386-1*:** Es una familia de herramientas para construir, probar y empaquetar software. Usado para controlar el proceso de compilación de software.

4) ***linux-source-2.6.34-i386-1*:** Paquete que contiene los fuentes del *Kernel* de Linux

Una vez instalada la paquetería de desarrollo, es necesario reinstalar *ggc-devel-4.3.2*, para evitar el despliegue de errores al momento de la compilación de aplicaciones:

```
root[~]# swaret --reinstall gcc-devel-4.3.2
```

4.3 Paquetes necesarios para la gestión de gráficos

Antes de construir y/o instalar un gestor de ventanas o entorno de escritorio sobre el sistema Beakos GNU/Linux (y en general para cualquier distribución) es necesario tener instalada toda la paquetería que comprende ***Xorg***, dicha paquetería se instaló de la siguiente forma en Beakos GNU/Linux:

```
root[~]# swaret --install xorg –a
```

4.4 Gestores de Ventanas y Entornos de Escritorio

Existen diversas GUI para poder interactuar y manipular elementos gráficos como: ventanas, botones, íconos, es decir, todos los elementos gráficos de un sistema operativo. Dentro de la clasificación para interfaces gráficas de usuario en distribuciones GNU/Linux existen:

a) Entornos de Escritorio

b) Gestores de Ventanas

Los Entornos de Escritorio, incorporan componentes (bibliotecas gráficas) que permiten desplegar elementos visuales con mayor detalle de diseño gráfico, una apariencia gráfica totalmente enfocada hacia usuarios finales que están acostumbrados a interactuar con elementos como:

- Cursor del mouse,
- Opciones de ventanas,
- Barras de herramientas,
- Barra de tareas,
- Íconos de acceso directo

Todo esto de forma similar a un ambiente de Windows.

Por otro lado también existen los gestores de ventanas que a diferencia de los Entornos de Escritorio, incorporan librerías gráficas que permiten un despliegue de componentes con una apariencia visual más sencilla, pero al mismo tiempo reducen el consumo de recursos. Los gestores de ventanas se caracterizan por brindar un manejo equilibrado entre la administración del sistema desde consola (línea de comando) y desde sencillas ventanas.

4.5 ¿Por qué Fluxbox para la distribución?

Porqué es un gestor de ventanas ligero además de que cubre necesidades importantes. Después de que se liberó la primera versión del sistema base de Beakos GNU/Linux, se decidió incorporar una GUI, con la intención de que el desarrollo de la distribución fuera creciendo, además de que incorporar una interfaz gráfica al sistema atraería a un mayor número de usuarios para probar la distribución, adicional a ello se exponen las siguientes razones funcionales para la implementación de un gestor de ventanas:

- El usuario puede iniciar la construcción de software libre a partir de un gestor de ventanas sencillo que no requiere de configuraciones complejas ni depende de un gran número de paquetes instalados.
- Optimiza el consumo de recursos de hardware.
- Su configuración implica al usuario un auto aprendizaje en distribuciones GNU/Linux.

- Es una manera rápida y funcional de comprobar si la construcción de ***Xorg*** es correcta.

4.6 Dependencias para la construcción de Flux Box 1.3.1

Para construir Flux Box en el sistema base de Beakos GNU/Linux, se requiere del paquete:

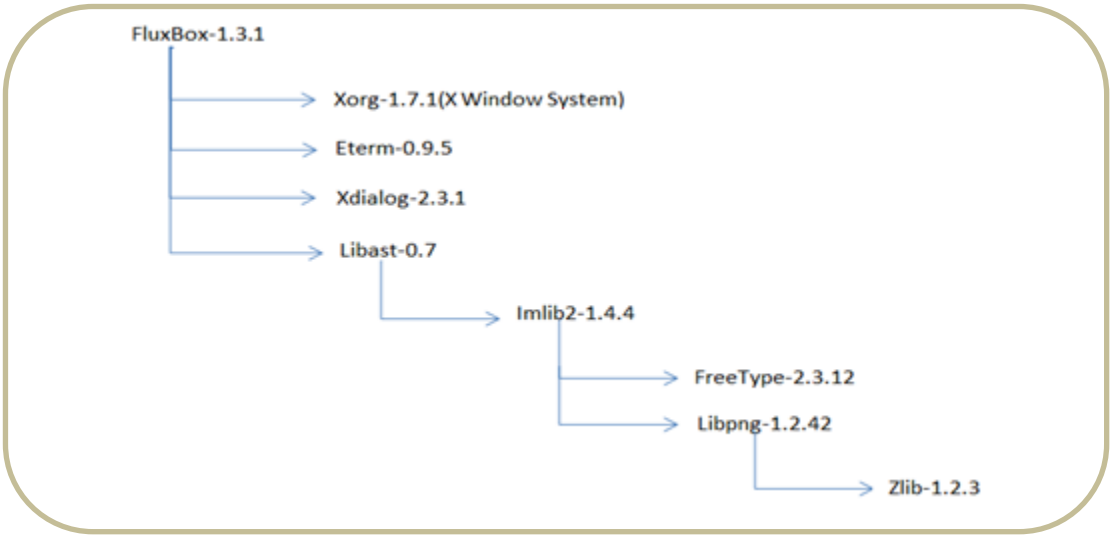
- Xorg-1.7.1

Los paquetes que se listan a continuación no son obligatorios para la instalación del gestor de ventanas, sin embargo, brindan funcionalidades y aplicaciones útiles, por lo que de igual forma se construyeron para la versión de Beakos GNU/Linux con entorno gráfico:

- Eterm-0.9.5
- Xdialog-2.3.1
- Libast-0.7

El código fuente de los paquetes se descargó de la página de ***Linux From Scratch*** o de la página oficial de cada proyecto. Una vez que el equipo de desarrollo construyó ***Xorg-1.7.1*** se comenzó a trabajar en la construcción de las dependencias para la compilación de ***Flux Box 1.3.1***. El siguiente diagrama muestra el árbol de dependencias y subdependecias que se construyeron:

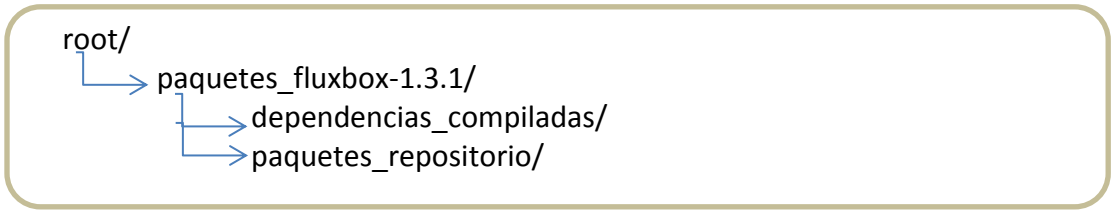
Gráfico 3.5 **Dependencias para la construcción de FluxBox**



Como se observa ***Zlib-1.2.3*** es el último paquete de las subdependecias para construir *Libast-0.7*, que a su vez forma parte de las dependencias para la construcción de ***FluxBox***. Sin embargo, dentro de la documentación de ***Linux From Scratch***, no se indica que se requiera del paquete ***Zlib-1.2.3***, dicho paquete se identificó hasta comenzar con la construcción de ***libpng-1.2.42***.

Es importante dar a conocer el árbol del directorio o directorios de trabajo que se emplearon durante la construcción de las dependencias de *FluxBox* para hacer más específico el trabajo de construcción.

Gráfico 3.6 **Directorios de trabajo para la construcción de FluxBox**



paquetes_fluxbox-1.3.1: Dentro de este directorio se colocaran los programas descargados que contienen el código fuente y que son las dependencias para la construcción de ***FluxBox***.

dependencias_compiladas: Dentro de este directorio se colocaran los paquetes de las dependencias de software que se vayan construyendo de forma correcta.

paquetes_repositorio: Dentro de este directorio se colocaran los paquetes de software que ya se encuentren depurados y listos para subir al servidor de repositorios.

Esta distribución de directorios de trabajo se empleó para la construcción del gestor de ventanas y es la que se recomienda para la construcción de paquetes de software, sin embargo, el desarrollador puede optar por una distribución distinta. A continuación se detalla el proceso de construcción del gestor de ventanas para Beakos GNU/Linux.

4.6.1 *Construcción de libpng-1.2.42*

Entorno de Desarrollo: ***Sistema base Beakos GNU/Linux***
*Directorio de Trabajo:****/root/paquetes_fluxbox1.3.1/***
Paquete compilado: ***libpng-1.2.42***
Código:

```
root[~]# tar –zxvf libpng-1.2.42.tar.gz; cd libpng-1.2.42
root[~]# ./configure --prefix=/usr/local
…
checking if libraries can be versioned... yes
checking for symbol prefix...
configure: pkgconfig directory is ${libdir}/pkgconfig
configure: creating ./config.status
config.status: creating Makefile
config.status: creating libpng.pc
config.status: creating libpng-config
config.status: creating config.h
config.status: executing depfiles commands
config.status: executing libtool commands
```

Al final del proceso de *configure,* no se obtuvo error alguno, por lo cual se genera el archivo ***makefile*** y otros archivos de configuración. De acuerdo con el proceso de construcción de paquetes, ahora correspondía ejecutar ***make****:*

```
root [~] # make
png.h:506:18: error: zlib.h: No such file or directory
...
png.c:804: error: 'png_struct' has no member named 'zstream'
png.c: In function 'png_check_IHDR':
png.c:980: error: 'png_struct' has no member named
'user_width_max'
png.c:990: error: 'png_struct' has no member named
'user_height_max'
png.c:1066: error: 'png_struct' has no member named
'mng_features_permitted'
png.c:1071: error: 'png_struct' has no member named
'mng_features_permitted'
make[1]: *** [libpng12_la-png.lo] Error 1
make[1]: Leaving directory `/root/paquetes_fluxbox1.3.1/libpng-
1.2.42'
make: *** [all] Error 2
```

El proceso de ***make*** arroja un error durante la ejecución y revisando el log se identifica en la línea 1 que no se encontró el archivo ***zlib.h***, por lo que las líneas siguientes arrogan errores al no encontrar los parámetros adecuados para las funciones dentro del programa ***png.c***, el cual es proporcionado por ***zlib.h***.

Dado lo anterior se identificó que libpng-1.6.4 requería de ciertas dependencias. El paquete de **Zlib** se compilo durante el proceso de construcción del sistema base de Beakos GNU/Linux, por lo que dicho paquete ya se encontraba disponible en el servidor de repositorios, listo para ser descargado e instalado:

```
root [~]# swaret --search zlib
swaret 1.6.3-2

Listing available Packages matching Keyword: zlib...
zlib-1.2.3-i386-1 (124 kB) (BEAKOS_PACKAGES) [Status:
INSTALLED]
zlib-devel-1.2.3-i386-1 (90 kB) (BEAKOS_PACKAGES_DEV) [Status:
NOT INSTALLED]

root [~]# swaret --install zlib-devel
swaret 1.6.3-2

zlib-devel

Listing available Packages matching Keyword: zlib-devel...
zlib-devel-1.2.3-i386-1 (90 kB) (BEAKOS_PACKAGES_DEV)

Install zlib-devel-1.2.3-i386-1 (BEAKOS_PACKAGES_DEV)?
(y/n/A/Q): [y]
```

Como se observa en el resultado de la búsqueda que arrojo el gestor de paquetes, se indica que el paquete ***zlib-1.2.3-i386-1*** ya se encuentra instalado; sin embargo, hay que recordar que para la construcción de nuevos paquetes se requiere también tener instaladas las cabeceras (***headeres****)* del paquete o los paquetes que conforman las dependencias del nuevo programa de software que se requiere compilar.

Una vez instalado el paquete de desarrollo de ***zlib*** se ejecuta nuevamente ***make*** para continuar con la compilación de ***libpng-1.6.4:***

```
root [~] # make
…
libtool: link: ranlib .libs/libpng.a
libtool: link: ( cd ".libs" && rm -f "libpng.la" && ln -s
"../libpng.la" "libpng.la")
cp libpng-config libpng12-config
cp libpng.pc libpng12.pc
make[1]: Leaving directory `/root/paquetes_fluxbox1.3.1/libpng-
1.2.42'
```

En esta ejecución el proceso de ***make*** se realiza sin errores y lo podemos identificar por la última línea del log donde el script de ***make*** sale del directorio donde se encuentra ***libpng-1.2.42***.

Una vez efectuado ***make***, se continúa con el proceso de generar el paquete con el formato o la extensión de paquete manejado en Beakos GNU/Linux (**.tgz**); para ello se ejecuta el comando ***checkinstall***, el cual sirve para poder generar paquetes de software en tres formatos distintos: **rpm**, **deb** y **tgz**. A continuación se detalla dicho proceso:

Entorno de Desarrollo: ***Sistema base GNU/Linux***
*Directorio de Trabajo:****/root/paquetes_fluxbox1.3.1/libpng-1.2.42***
Paquete compilado: ***libpng-1.2.42***
Código:

```
root [~]# checkinstall

checkinstall 1.5.3, Copyright 2001 Felipe Eduardo Sanchez Diaz Duran
           This software is released under the GNU GPL.

The package documentation directory ./doc-pak does not exist.
Should I create a default set of package docs?  [y]:
make[2]: Entering directory `/root/paquetes_fluxbox1.3.1/libpng-
1.2.42'
cd /usr/include; rm -f png.h pngconf.h
cd /usr/include; ln -s libpng12/png.h png.h
cd /usr/include; ln -s libpng12/pngconf.h pngconf.h
cd /usr/lib/pkgconfig; rm -f libpng.pc
cd /usr/lib/pkgconfig; ln -s libpng12.pc libpng.pc
make[2]: Leaving directory `/root/paquetes_fluxbox1.3.1/libpng-
1.2.42'
```

```
make[1]: Leaving directory `/root/paquetes_fluxbox1.3.1/libpng-
1.2.42'

Building file list...OK

Please choose the packaging method you want to use.
Slackware [S], RPM [R] or Debian [D]? S
```

Como se ha comentado en este trabajo el formato de empaquetamiento utilizado para la distribución de Beakos GNU/Linux es **.tgz** (el cual fue en un principio el formato utilizado únicamente por la distribución de ***Slackware***, hoy en día este formato es un proyecto de desarrollo independiente de la distribución). Es por ello que se selecciona la opción "S". A continuación se tiene que escribir una descripción acerca del programa de software contenido en el paquete:

```
Please write a description for the package. Remember that pkgtool
shows only the first one when listing packages so make that one
descriptive.
End your description with an empty line or EOF.

>> Contiene bibliotecas utilizadas por otros programas para para
lectura y escritura de archivos PNG.
```

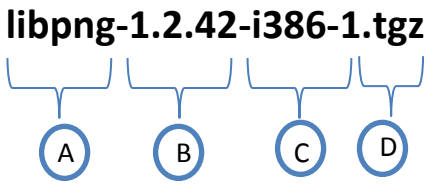
Una vez introducida la descripción de paquete ***checkinstall*** despliega un resumen para que el desarrollador valide los datos generales del nuevo paquete:

```
This package will be built according to these values:

1 -  Summary: [Contiene bibliotecas usadas por otros programas
para la lectura y escritura de archivos PNG ]
2 -  Name:    [ libpng ]
3 -  Version: [ 1.2.42 ]
4 -  Release: [ 1 ]
5 -  License: [ GPL ]
6 -  Group:   [ Applications/System ]
7 -  Architecture: [ i386 ]
8 -  Source location: [ libpng-1.2.42 ]
9 -  Alternate source location: [  ]

Enter a number to change any of them or press ENTER to
continue:
```

En caso de ser necesario, se pueden modificar los campos desplegados de lo contrario se continúa con el proceso, el cual consiste en la instalación de programa de software construido y la generación del archivo **.tgz**:

```
            ******************************************
            **** Slackware package creation selected ***
            ******************************************

1 -  Summary: [ Contiene librerias usadas por otros programas
para la lectura y escritura de archivos PNG ]
2 -  Name:    [ libpng ]
Preparing Slackware install directory...
3 -  Version: [ 1.2.42 ]
4 -  Release: [ 1 ]
5 -  License: [ GPL ]
OK

Writing package description...
6 -  Group:   [ Applications/System ]
7 -  Architecture: [ i386 ]
8 -  Source location: [ libpng-1.2.42 ]
9 -  Alternate source location: [  ]
OK
Writing Slackware install script...OK
Creating package libpng-1.2.42-i386-1...OK
Installing package...OK

************************************************************
 Done. The new package has been installed and saved to
 /root/paquetes/libpng-1.2.42/libpng-1.2.42-i386-1.tgz

 You can remove it from your system anytime using:
      removepkg libpng-1.2.42-i386-1
************************************************************
```

Como resultado de este proceso ***checkinstall*** nos da ***libpng*** empaquetado como:



a) Nombre del programa,

b) Versión del programa,

c) Arquitectura para la cual fue construido el programa,

d) Extensión del paquete (formato de empaquetamiento).

4.6.2 *Clasificación de los paquetes binarios y de desarrollo*

Dado que ***checkinstall*** empaqueta todos los archivos tanto binarios como bibliotecas de desarrollo y manuales en un solo paquete, por buenas prácticas siempre se clasifican los archivos de desarrollo en otro paquete identificado con ***devel*** en el nombre, esto se lleva a cabo en las distribuciones GNU/Linux por varias razones:

1. Evita que se instalen archivos de configuración innecesarios dentro de un sistema Linux que no es precisamente para desarrollo.
2. Reduce el tamaño de los paquetes que se encuentran en el repositorio de software, lo cual ahorra tiempo de descarga y espacio en el disco duro del equipo donde son instalados.
3. Evita que usuarios comunes o novatos, tengan la posibilidad de modificar las configuraciones del sistema involuntariamente y por ende volverlo inestable.
4. Evita que los sistemas GNU/Linux sean vulnerables ante la ejecución de scripts que tengan por objetivo compilar programas de software maliciosos.

Es necesario separar los archivos de desarrollo (bibliotecas .h) en un paquete ***libpng-devel***, dejando únicamente los archivos ordinarios (ejecutables y algunas bibliotecas) y demás scripts de instalación en un paquete de ***libpng*** depurado.

Cuadro 3.6 **Árbol de directorios de libpng-1.2.42-i386-1.tgz**

libpng-1.2.42-i386-1.tgz después de *checkinstall*

```
.
|-- install
|   |-- description
|   `-- doinst.sh
`-- usr
    |-- bin
    |   `-- libpng12-config
    |-- doc
    |   `-- libpng-1.2.42
    |       |-- ANNOUNCE
    |       |-- CHANGES
    |       |-- INSTALL
    |       |-- LICENSE
    |       |-- README
    |       `-- TODO
    |-- include
    |   `-- libpng12
    |       |-- png.h
    |       `-- pngconf.h
    |-- lib
    |   |-- libpng.so.3.42.0
    |   |-- libpng12.a
    |   |-- libpng12.la
    |   |-- libpng12.so.0.42.0
    |   `-- pkgconfig
    |       `-- libpng12.pc
    `-- share
        `-- man
            |-- man3
            |   |-- libpng.3.gz
            |   `-- libpngpf.3.gz
            `-- man5
                `-- png.5.gz

13 directories, 19 files
```

libpng-1.2.42-i386-1.tgz depurado

```
.
|-- install
|   |-- description
|   `-- doinst.sh
`-- usr
    |-- bin
    |-- lib
    |   |-- libpng.so -> libpng.so.3.42.0
    |   |-- libpng.so.3 -> libpng.so.3.42.0
    |   |-- libpng.so.3.42.0
    |   |-- libpng12.so -> libpng12.so.0.42.0
    |   |-- libpng12.so.0 -> libpng12.so.0.42.0
    |   `-- libpng12.so.0.42.0
    `-- share

5 directories, 8 files
```

libpng-devel-1.2.42-i386-1.tgz (para desarrollo)

```
.
|-- install
|   |-- description
|   `-- doinst.sh
`-- usr
    |-- bin
    |   `-- libpng12-config
    |-- doc
    |   `-- libpng-1.2.42
    |       |-- ANNOUNCE
    |       |-- CHANGES
    |       |-- INSTALL
    |       |-- LICENSE
    |       |-- README
    |       `-- TODO
    |-- include
    |   `-- libpng12
    |       |-- png.h
    |       `-- pngconf.h
    |-- lib
    |   |-- libpng12.a
    |   |-- libpng12.la
    |   `-- pkgconfig
    |       `-- libpng12.pc
    `-- share
        `-- man
            |-- man3
            |   |-- libpng.3.gz
            |   `-- libpngpf.3.gz
            `-- man5
                `-- png.5.gz
```

El cuadro muestra un comparativo del contenido entre el ***libpng-1.2.42*** que se empaquetó con ***checkinstall*** y los paquetes ***libpng*** **depurado** y ***libpng*** para **desarrollo**.

Partiendo de la estructura de directorios propuesta para la construcción de ***FluxBox***, dentro de ***dependencias_compiladas/*** tenemos el paquete **libpng-1.2.42-i386-1.tgz**, el cual se tiene que volver a descomprimir para poder clasificar sus archivos y obtener los dos paquetes: depurado y para desarrollo de ***libpng*** mostrados en el cuadro anterior; antes hay que preparar una estructura de directorios nuevamente en subcarpetas contenidas dentro de ***dependencias_compiladas/*** para poder identificar cuáles son los archivos de los que se compone ***libpng*** y no mezclar archivos de los demás paquetes que se vayan depositando.

```
root[~]# mkdir libpng-1.2.42_descomprimido
root[~]# cd libpng-1.2.42_descomprimido
root[~]# mv ../libpng-1.2.42-i386.tgz ./
root[~]# tar –zxvf libpng-1.2.42-i386-1.tgz
./
./usr/
./usr/doc/
./usr/doc/libpng-1.2.42/
./usr/doc/libpng-1.2.42/TODO
./usr/doc/libpng-1.2.42/README
./usr/doc/libpng-1.2.42/LICENSE
./usr/doc/libpng-1.2.42/INSTALL
./usr/doc/libpng-1.2.42/CHANGES
./usr/doc/libpng-1.2.42/ANNOUNCE
…
```

Una vez que se descomprimió **libpng-1.2.42-i386-1.tgz** dentro del directorio que se creó, tenemos los siguientes directorios:

- **install/**
- **usr/**

Dentro del directorio **install/** es necesario depurar el script ***doninst.sh***, a continuación se muestra el contenido de dicho script:

```
1) #
2) # doinst.sh, auto-generated by checkinstall-1.5.3
3) #
4) echo
5) cat /install/description
6) sed '/PACKAGE LOCATION/r /install/description' < "/var/log/packages/libpng-1.2.42-i386-
   1" > description.tmp
7) mv description.tmp "/var/log/packages/libpng-1.2.42-i386-1"
8) rm /install/description
9) ( cd . ; rm -rf onfig )
10) ( cd . ; ln -sf libpng12-config onfig )
11) ( cd . ; rm -rf onf.h )
12) ( cd . ; ln -sf libpng12/pngconf.h onf.h )
13) ( cd . ; rm -rf h )
14) ( cd . ; ln -sf libpng12/png.h h )
15) ( cd . ; rm -rf o.3 )
16) ( cd . ; ln -sf libpng.so.3.42.0 o.3 )
17) ( cd . ; rm -rf a )
18) ( cd . ; ln -sf libpng12.la a )
19) ( cd . ; rm -rf .so )
20) ( cd . ; ln -sf libpng12.so.0.42.0 .so )
21) ( cd g ; rm -rf libpng.pc )
22) ( cd g ; ln -sf libpng12.pc libpng.pc )
23) ( cd . ; rm -rf o )
24) ( cd . ; ln -sf libpng12.so o )
25) ( cd  ; rm -rf  )
26) ( cd  ; ln -sf libpng12.a  )
27) ( cd . ; rm -rf .so.0 )
28) ( cd . ; ln -sf libpng12.so.0.42.0 .so.0 )
```

Líneas de código eliminadas

Tal como se observa, es necesario eliminar las líneas de código 9 a la 28, ya que estas líneas contienen instrucciones que hacen referencia hacia rutas inexistentes por lo cual son quitadas del *script*, sin embargo, utilizando un editor de texto como ***vi*** se deben agregar las siguientes líneas de código las cuales se refieren a rutas y enlaces simbólicos correctos:

- `( cd /usr/include; rm -f png.h pngconf.h )`
- `( cd /usr/include; ln -s libpng12/png.h png.h )`
- `( cd /usr/include; ln -s libpng12/pngconf.h pngconf.h )`
- `( cd /usr/lib/pkgconfig; rm -f libpng.pc )`
- `( cd /usr/lib/pkgconfig; ln -s libpng12.pc libpng.pc )`

Estas líneas se agregaron después de observar que las líneas depuradas contenían instrucciones que no empataban con la estructura de directorios particular de Beakos GNU/Linux, por lo cual se agregaron las nuevas líneas, las cuales funcionan específicamente para la distribución.

Una vez modificado el script ***doinst.sh*** se continúa con la clasificación de los archivos. Se crea un directorio temporal dentro del directorio ***dependencias_compiladas/***, que será el paquete de ***libpng*** depurado (sin archivos fuente):

```
root[~] # mkdir -v libpng_1.2.42; cd libpng_1.2.42
```

Posteriormente dentro del nuevo directorio se genera la estructura de directorios propia del paquete de software:

```
root[~] # mkdir –v install usr
```

Posteriormente estando dentro del nuevo directorio **libpng_1.2.42** creado, se tienen que copiar los siguientes archivos tomados de **libpng-1.2.42_descomprimido** en los directorios correspondientes:

```
root[~] # cp -v ../libpng-1.2.42_descomprimido/install/* ./install
```

A continuación, dentro del directorio ***install*** de ***libpng_1.2.42*** hay que modificar los privilegios de cada uno de los archivos copiados, esto es importante ya que si no se otorgan los privilegios de ejecución adecuados, al momento de la instalación pueden surgir errores de lectura y escritura.

```
root[~]# ls -l
-rw-r--r-- 1 root root 111 Oct 26 15:45 description
-rw-r--r-- 1 root root 877 Oct 11 00:34 doinst.sh
```

Ahora asignamos los nuevos:

```
root[~]#  chmod 755 description
root[~]#  chmod 755 doninst.sh
root[~]#  ls –l
-rwxr-xr-x 1 root root 111 Oct 26 15:45 description
-rwxr-xr-x 1 root root 877 Oct 11 00:34 doinst.sh
```

Tanto el propietario como el grupo al que pertenecen los archivos quedan como ***root***.

Ahora se tienen que colocar las bibliotecas básicas dentro de la carpeta **usr/**, cabe recordar que dentro de **libpng_1.2.42** se generaron los directorios ***install/*** y ***usr/*** ya trabajamos sobre el primer directorio, por lo cual resta trabajar sobre **usr/**, dentro de esta carpeta se generaron las siguientes subcarpetas:

```
root[~] # mkdir -v bin lib share
```

Tomando como referencia el cuadro 3.6, se movieron los siguientes archivos del paquete **libpng-1.2.42** generado con ***checkisntall*** hacia la subcarpeta ***lib/*** *(contenida en **usr/**)* que se acaba de generar:

- **libpng.so.3.42.0**
- **libpng12.so.0.42.0**

A continuación se muestran los comandos empleados con las rutas absolutas, para mayor referencia:

```
root[~] # mv
/root/paquetes_fluxbox1.3.1/dependencias_compiladas/libpng-
1.2.42_descomprimido/usr/lib/libpng.so.3.42.0
/root/paquetes_fluxbox1.3.1/dependencias_compiladas/libpng_1.2.42/
usr/lib/
```

```
root[~] # mv
/root/paquetes_fluxbox1.3.1/dependencias_compiladas/libpng-
1.2.42_descomprimido/usr/lib/libpng12.so.0.42.0
/root/paquetes_fluxbox1.3.1/dependencias_compiladas/libpng_1.2.42/
usr/lib/
```

Una vez que ya se encuentran las dos bibliotecas de **libpng** dentro de la subcarpeta **lib/** de **usr/**, se generaron los enlaces simbólicos necesarios.

```
root[~] # ln -sf libpng.so.3.42.0 libpng.so
root[~] # ln -sf libpng.so.3.42.0 libpng.so.3
root[~] # ln -sf libpng12.so.0.42.0 libpng12.so
root[~] # ln -sf libpng12.so.0.42.0 libpng12.so.0
root[~] # ls –l
lrwxrwxrwx 1 root root     16 Oct 27 16:07 libpng.so -> libpng.so.3.42.0
lrwxrwxrwx 1 root root     16 Oct 27 16:08 libpng.so.3 -> libpng.so.3.42.0
-rwxr-xr-x 1 root root 151548 Oct 11 00:29 libpng.so.3.42.0
lrwxrwxrwx 1 root root     18 Oct 27 16:10 libpng12.so -> libpng12.so.0.42.0
lrwxrwxrwx 1 root root     18 Oct 27 16:10 libpng12.so.0 -> libpng12.so.0.42.0
-rwxr-xr-x 1 root root 141764 Oct 11 00:29 libpng12.so.0.42.0
```

Los enlaces simbólicos son de gran importancia ya que las demás aplicaciones que requieran de estas dos bibliotecas las llegan a solicitar por el nombre de cualquiera de los enlaces simbólicos que se generaron, que en este caso sirven como “alias” para cada una de las bibliotecas de **libpng**; por ejemplo, una aplicación cualquiera dentro de su *scritp* de configuración de parámetros puede requerir **libpng.so**, sin embargo, el nombre con el que se tiene instalada esa biblioteca dentro del sistema es **libpng.so.3.42.0** por lo que al no existir un alias de esta como **libpng.so** la aplicación que requiere de esta biblioteca lanzará un mensaje de error indicando que no se ha encontrado el archivo, siendo que sí está instalado pero con un nombre diferente. Es por ello que de forma general las bibliotecas siempre deben llevar enlaces simbólicos que les permitan nombrarlas con sus respetivos “alias”.

Una vez que se tuvo el paquete ordinario de **libpng** y depurado, este se empaquetó nuevamente de la siguiente forma:

Directorio de trabajo: ***/root/paquetes_fluxbox1.3.1/dependencias_compiladas/libpng_1.2.42***

```
root[~] # tar -zcvf ../libpng-1.2.42-i386-1.tgz .
```

Una vez generado el paquete, se tienen los siguientes archivos:

- **libpng-1.2.42-i386-1.tgz**
- **libpng-1.2.42_descomprimido**
- **libpng_1.2.42**

Con esto se generó el paquete de **libpng-1.2.42** ordinario, por último es necesario modificar los permisos del paquete generado:

```
root[~] # chmod 664 libpng-1.2.42-i386-1.tgz
```

```
root[~] # mv libpng-1.2.42-i386-1.tgz
/root/paquetes_fluxbox1.3.1/paquetes_repositorio
```

El nuevo paquete es colocado en la carpeta **paquetes_repositorio/** y está listo para ser subido al servidor de repos de Beakos GNU/Linux.

En el siguiente árbol de directorios se puede observar el contenido de trabajo elaborado hasta el momento.

```
root [ ~/paquetes_fluxbox1.3.1 ]# tree
.
|-- dependencias_compiladas
|   |-- libpng-1.2.42-i386-1.tgz
|   |-- libpng-1.2.42_descomprimido
|   |   |-- install
|   |   `-- usr
|   |       |-- bin
|   |       |   `-- libpng12-config
|   |       |-- doc
|   |       |   `-- libpng-1.2.42
|   |       |       |-- ANNOUNCE
|   |       |       |-- CHANGES
|   |       |       |-- INSTALL
|   |       |       |-- LICENSE
|   |       |       |-- README
|   |       |       `-- TODO
|   |       |-- include
|   |       |   `-- libpng12
|   |       |       |-- png.h
|   |       |       `-- pngconf.h
|   |       |-- lib
|   |       |   |-- libpng12.a
|   |       |   |-- libpng12.la
|   |       |   `-- pkgconfig
|   |       |       `-- libpng12.pc
|   |       `-- share
|   |           `-- man
|   |               |-- man3
|   |               |   |-- libpng.3.gz
|   |               |   `-- libpngpf.3.gz
|   |               `-- man5
|   |                   `-- png.5.gz
|   `-- libpng_1.2.42
|       |-- install
|       |   |-- description
|       |   `-- doinst.sh
|       `-- usr
|           |-- bin
|           |-- lib
|           |   |-- libpng.so -> libpng.so.3.42.0
|           |   |-- libpng.so.3 -> libpng.so.3.42.0
|           |   |-- libpng.so.3.42.0
|           |   |-- libpng12.so -> libpng12.so.0.42.0
|           |   |-- libpng12.so.0 -> libpng12.so.0.42.0
|           |   `-- libpng12.so.0.42.0
|           `-- share
|--libpng-1.2.42
|-- libpng-1.2.42.tar.gz
`--paquetes_repositorio
    `-- libpng-1.2.42-i386-1.tgz

22 directories, 26 files
```

3. Paquete generado con checkinstall

4. Directorio descomprimido para depurar

5. Directorio depurado

2. Directorio descomprimido donde se realiza la compilación

1. Paquete descargado que contiene el código fuente, posteriormente se descomprime.

6. Paquete depurado y listo para subir al repositorio

Para la construcción de un solo programa de software se desprenden varios directorios y subdirectorios, además de la ejecución de los diversos comandos que esto implica. Observando el árbol de directorios anterior se puede identificar la diferencia de contenido entre uno y otro paquete; al finalizar el proceso de empaquetado del **libpng** depurado se tenían dos paquetes con el mismo nombre:

A. `/root/paquetes_fluxbox1.3.1/dependencias_compiladas/libpng-1.2.42-i386-1.tgz`
   Que corresponde al paquete generado con ***checkinstall***.

B. `/root/paquetes_fluxbox1.3.1/paquetes_repositorio/libpng-1.2.42-i386-1.tgz`

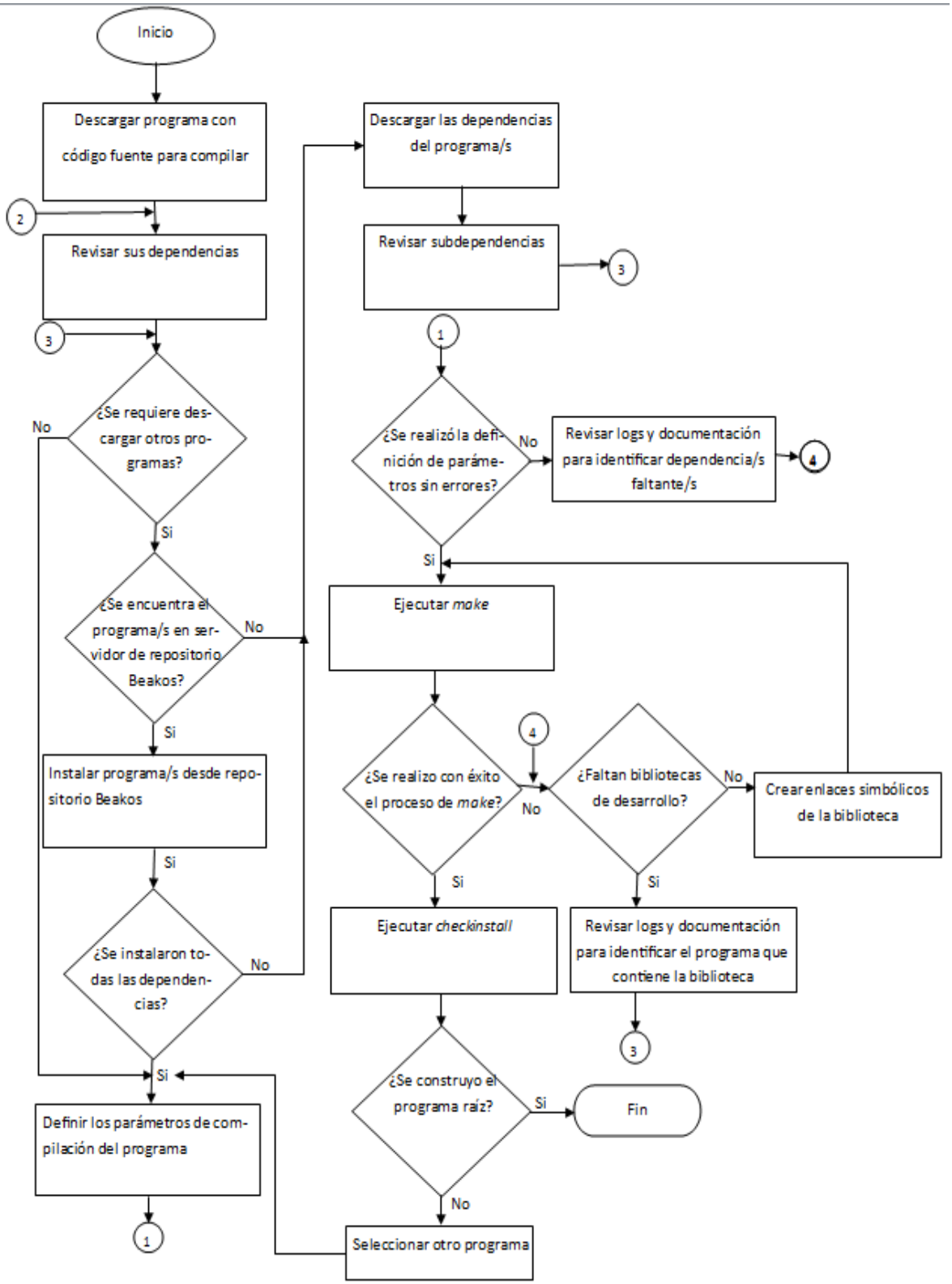
   Que corresponde al paquete depurado.

Por ello es importante mantener una estructura sobre los paquetes que va construyendo para así evitar subir paquetes erróneos al servidor de repositorios, además de que el tener clasificado cada paquete construido llega a facilitar un proceso de revisión de los paquetes cuando se requiere aplicar algún parche o modificación a su contenido.

Una vez que ya se tiene un esquema detallado del estatus actual de construcción de **libpng**, así como de los directorios de trabajo y su contenido, se continuó con la creación del paquete **libpng-devel**, el cual contendrá los paquetes restantes de **libpng-1.2.42_descomprimido**.

Similar a este proceso de construcción es como se fueron compilando las demás dependencias hasta construir el gestor de ventanas FluxBox para Beakos GNU/Linux, tomando en cuenta que cada aplicación se compila con sus parámetros de configuración adecuados. El siguiente cuadro muestra de forma general como se puede realizar la construcción de paquetería de software:

Gráfico 3.7 **Diagrama de flujo para la construcción de los paquetes de software de FluxBox**



5. Resultados obtenidos

5.1 Reconocimiento como una distribución 100% mexicana a nivel internacional

El sistema **Beakos GNU/Linux** se ha colocado dentro de las diversas distribuciones GNU/Linux, como un desarrollo nacional avalado por ***DistroWatch***.

DistroWatch es el sitio web que además de publicar el nivel de popularidad de sistemas GNU/Linux, reconoce a las distribuciones GNU/Linux que tienen una base de desarrollo estandarizada, para ello evalúan diversos elementos que debe tener una distribución para poder ser anunciada en su sitio. Dentro de los elementos que se evalúan, se encuentran los siguientes:

- Contar con un sitio web oficial, el cual se encuentre publicado bajo un nombre de dominio propio.
- Tener la documentación del sistema GNU/Linux disponible, actualizada y bien estructurada, siguiendo el estilo de distribuciones como ***Debian***, ***Ubuntu***, ***Slackware***, etc. Esta documentación comúnmente se encuentra contenida en wikis que sean un subdominio del sitio web oficial de la distribución.
- Contar con listas de correo, foros y otros medios de discusión acerca de la distribución, estos deben estar como un subdominio del sitio web oficial.
- Proporcionar las versiones de los paquetes que incorpora la distribución, por ejemplo, versión del núcleo, compiladores, bibliotecas, paquetes de desarrollo, etc.
- Tener como mínimo una versión estable de la distribución.
- Mantener actualizada la documentación, así como ir liberando nuevas mejoras o versiones de la distribución.

La primera versión de **Beakos GNU/Linux**, ha pasado por un proceso de madurez y mejora que le ha permitido ser aceptada dentro de ***DistroWatch*** y de forma oficial es calificada como la primera distribución GNU/Linux mexicana. Este

reconocimiento, sirvió para demostrar la calidad en el desarrollo de Beakos GNU/Linux. A continuación se deja la siguiente liga de consulta:

- http://distrowatch.com/table.php?distribution=beakos

5.2 Vinculación con la comunidad de usuarios

Desde que se liberó la primera versión de Beakos GNU/Linux y que esta fue puesta a disposición de los usuarios, se comenzó a fomentar la conformación de una comunidad que participara en el desarrollo de nuevos productos de software.

INFOTEC tiene la iniciativa de captar una comunidad de investigadores y desarrolladores, interesada en aprender acerca del uso y desarrollo de software.

Se han generado convenios con universidades, para las cuales se encuentra diseñando el modelo de aprendizaje e integración que sea atractivo a los alumnos y a las propias instituciones, a continuación se menciona acerca de este esquema:

- Impartir cursos y talleres en instituciones educativas, con valor académico, que estén respaldados bajo una certificación.
- Impartir cátedras para la formación de nuevos investigadores.
- Realizar estadías dentro del INFOTEC para aquellos alumnos que deseen realizar sus prácticas profesionales o servicio social.

CONCLUSIONES

Este trabajo de investigación, en un **primer nivel**, realizó el análisis de las problemáticas experimentadas dentro de un área de operación en el Centro Público de Información INFOTEC, este análisis explora y describe:

1. **Los antecedentes, sus principales funciones y servicios del INFOTEC,** donde el CPI se encarga de atender necesidades de interés nacional en materia de TIC, a través de la implementación de I+D, bajo esquemas de negoció que le permitan brindar solución a empresas públicas y privadas con base en el aprovechamiento de recursos de TIC. Sus estrategias van encaminadas en busca del desarrollo de la sociedad de la información y el conocimiento.
2. **Con base en la estructura orgánica del CPI, se identificaron las áreas técnicas que se involucran en el desarrollo e implementación de nuevos proyectos**, cada una de estas áreas: DADT, DAC y DACI logran interactuar a través de la definición de objetivos que se encaminan al cumplimiento de la Visión y Misión institucional. Cada una de estas direcciones converge en procesos de entrada y salida de información, dando pie a la retroalimentación entre todas las Direcciones Adjuntas y como resultado el planteamiento de nuevos proyectos internos o externos; en el caso de este proyecto expuesto, la DADT fungió como un cliente interno del propio CPI al requerir alternativas de solución ante sus problemáticas operativas, que a su vez le permitieran garantizar adecuados niveles de servicio hacia los clientes del INFOTEC.
3. **La problemática que afectaba la operación eficiente de los ambientes productivos instalados en el Centro de Datos del CPI**, donde se concluye que la deficiente administración en la operación de sistemas operativos GNU/Linux daban pie a otros factores que repercutían en la calidad del servicio que se brindaba a los clientes, así también, al interior de la organización representaba un desaprovechamiento en parte de la infraestructura de hardware que se tenía en ese momento.

Como resultado del análisis realizado anteriormente, en un **segundo nivel** se aunó en la exploración de una solución a través de elementos formales y estandarizados que atendieran las problemáticas planteadas, destacando lo siguiente:

1. **Métodos formales para el desarrollo de la solución,** donde se analizaron diversas alternativas para el desarrollo de sistemas GNU/Linux, determinando como la solución más factible el método del LFS por permitir un nivel de desarrollo considerable acorde a lo que el CPI requería como parte de su solución.
2. **Dinámicas para la integración grupal y desarrollo del proyecto de forma ordenada y efectiva,** se determinó que las metodologías ágiles permiten la incorporación de nuevas formas de interacción y trabajo grupal que se adaptaban a las condiciones de desarrollos con cambios emergentes y no previstos durante los proyectos, además de que las dinámicas de trabajo permiten el desarrollo de productos de software funcionales de manera pronta.

Derivado de la fundamentación teórica plasmada en este trabajo de investigación se llevó a cabo en un **tercer nivel** el desarrollo y la implementación del proyecto donde se aplicó:

1. **Un método de desarrollo de software acorde a las necesidades actuales de la DADT,** la adecuada aplicación del método Linux From Scratch resultó eficaz para el desarrollo de un sistema GNU/Linux propio del INFOTEC, formando parte de la solución planteada, ya que no bastó con implementar este método, si bien aporta grandes elementos de desarrollo, se requirió poner en práctica desarrollos a base de prueba y error para llegar establecer configuraciones de componentes complejos como el núcleo del sistema. Las omisiones de cierta documentación dentro del LFS resultaron ser uno de los retos superados durante el desarrollo de la solución final: Beakos GNU/Linux, aunado al detalle de la compatibilidad entre ciertos paquetes que se decidieron construir con versiones más recientes a las especificadas en el LFS 6.4, (esto

debido a que las nuevas versiones ya incluían mejoras identificadas por los desarrolladores durante su administración en otros sistemas GNU/Linux).

2. **Una metodología efectiva para la aplicación del método LFS y acorde a las necesidades del proyecto**, la definición de las formas de trabajo grupal fueron de los hitos importantes durante el proyecto ya que el objetivo era copiar las dinámicas de otras distribuciones GNU/Linux destacadas, al implementar herramientas estándar a lo largo de su desarrollo, estas herramientas pueden ser: el servidor de repositorios, sistema de empaquetamiento, clasificación de las aplicaciones de software, documentación, sitio web, foros de ayuda, entre otros. Las prácticas de la programación extrema, como la programación por pares, el acondicionamiento de un entorno de trabajo que favoreciera el desarrollo de software y la retroalimentación, permitieron establecer las formas adecuadas para la interacción y dar solución a problemas de desarrollo que surgían a lo largo del proyecto. Una vez que se obtenía la construcción de algún componente de software su documentación servía como base de conocimiento para futuros desarrollos, incluso para considerar diseñar un método de construcción derivado del LFS para futuras versiones de Beakos GNU/Linux o de otros desarrollos de software.

En cuanto a los resultados obtenidos, se desarrolló un producto de software basado en estándares, que cumplió con los aspectos técnicos y funcionales para el logro del objetivo planteado por el INFOTEC, con ello el desarrollo de Beakos GNU/Linux aporta elementos para elevar la competitividad del CPI, bajo propuestas sustentadas en I+D y la gestión informática que garantizaron el uso eficiente de los recursos de TIC con que disponía el INFOTEC.

Finalmente se reconoce la necesidad dentro de aspectos técnicos y de crecimiento del proyecto:

1. **La definición de un método** que resulte eficiente en la gestión de las versiones de paquetes de software del sistema, para tener un seguimiento de las aplicaciones en sus diferentes estados: inestables, de pruebas y estables. Ya que hasta el momento únicamente se han llevado a un estado estable las

aplicaciones que integran el sistema Beakos GNU/Linux en su versión para servidores.

2. **Complementar la documentación** del sistema, para ello es importante actualizar los manuales técnicos, y generar nuevos tutoriales para la capacitación de los usuarios dentro y fuera del INFOTEC.
3. **Dar continuidad a la línea de desarrollo** de Beakos GNU/Linux, donde se tiene planeado motivar aún más la participación de la comunidad que se está formando alrededor de este proyecto, para contribuir en cuanto a propuestas y desarrollos de nuevas características que se vayan incorporando al sistema.
4. Con base en la continuidad que se pretende para la línea de desarrollo y tomando como referencia investigaciones consultadas en torno al desarrollo de Software Libre; es conveniente trabajar en el **diseño de una metodología** (resultado de la combinación de metodologías existentes como: RUP, SCRUM, XP, etc.) que permita al INFOTEC contar con procesos y desarrollos que se adapten a su entorno y necesidades al interior, pero que también esta metodología integre a los procesos de desarrollo y madurez del proyecto a colaboradores externos (comunidad de desarrolladores) sin perder de vista el control y la calidad del producto final.

Estas nuevas áreas para la investigación y desarrollo permitirán llevar el sistema Beakos GNU/Linux hacía una tendencia adecuada de crecimiento y madurez que sea solvente de acuerdo a nuevos requerimientos funcionales y técnicos en la operación de sistemas GNU/Linux dentro del INFOTEC, pero que además sirva como referente para desprender nuevas soluciones de TIC que favorezcan a la sociedad en general.

GLOSARIO

API.- Application Programming Interface, es un conjunto de métodos o funciones de programación de uso general que permiten la comunicación entre componentes de software, brindan la posibilidad de reutilizar código para la generación de elementos como ventanas o íconos en una aplicación.

Biblioteca.- En informática una biblioteca es un conjunto de subprogramas que facilitan el desarrollo de software, estas contienen código que es utilizado por otros programas independientes, esto representa parte de la modularidad en los programas.

CDE.- *Common Desktop Environment*, es una interfaz gráfica de usuario para entornos de escritorio de código abierto, permite la gestión de datos, archivos y aplicaciones. Proporciona portabilidad, facilidad de uso y un diseño distribuido del entorno gráfico.

Compilador.- Es un programa de software que se encarga de traducir un código fuente escrito en lenguaje de alto nivel a código objeto (código binario) ejecutable. Para ello realiza una serie de fases o etapas las cuales comprenden: análisis léxico, análisis sintáctico, análisis semántico, generación de código intermedio, optimización independiente de la arquitectura, generación de código, optimización dependiente de la arquitectura.

Creative Commons.- Es una organización sin fines de lucro que permite el intercambio y uso compartido del conocimiento y creatividad a través de instrumentos legales, dichos instrumentos son un conjunto de licencias que se combinan con el copyright para que un autor posea total o parcialmente los derechos sobre su obra, a fin de tener una licencia más flexible, con los términos de derecho de autor que más se adecuen a sus necesidades.

CUPS.- Sistema para la administración de impresoras dentro de una red, trabaja bajo la arquitectura cliente/servidor cuenta con una interfaz gráfica que facilita la gestión al Administrador, entre sus funcionalidades están la administración de

usuarios, control de las colas de impresión, generación de reportes, agregar/eliminar impresoras, etc. También puede ser administrado desde línea de comandos para usuarios más avanzados.

Emacs.- Es un editor de texto extensible, desarrollado por Richard Stallman, sin embargo, Emacs aporta múltiples herramientas de software a través de sus módulos, tales como un cliente de correo, interprete de comandos, navegador web, cliente de RSS, cliente de IRC, compiladores para distintos lenguajes de programación como C, C++, Bison, etc. Como su creador lo describe, “desde Emacs puedes hacer toda tu informática”.

FDISK.- Es una herramienta de software que permite realizar la gestión de discos de almacenamiento,

FHS.- *File Hierachi System*, es una norma que consta de un conjunto de requisitos y lineamientos para los archivos y directorios en cuanto a su ubicación bajo sistemas operativos UNIX. Su objetivo es ser un estándar para apoyar la interoperabilidad en la administración de aplicaciones y herramientas de desarrollo, además de que la estructura de directorios y archivos definida permite al usuario predecir la ubicación de configuraciones y aplicaciones instaladas en el sistema operativo.

Firmware.- Es un conjunto de instrucciones de bajo nivel, que se encuentran directamente vinculadas con los componentes de hardware, para manipular circuitos electrónicos que permitan el correcto funcionamiento de los dispositivos de cómputo (tarjetas de red, tarjetas de video, puertos de conexión, etc). Comúnmente este tipo de software va incorporado en una memoria de tipo ROM, que acompaña al dispositivo de hardware para que este pueda operar.

FreeBSD.- Es la implementación libre del sistema operativo BSD, (*Berkeley Software Distribution*) el cual es una versión de UNIX. Como parte de la filosofía del software libre FreeBSD es mantenido por una comunidad de desarrolladores a nivel mundial.

GCC.- *GNU C COMPILER*, es un compilador para lenguaje C, desarrollado por Richard Stallman durante la creación de su sistema GNU.

Headers.- Son archivos de cabecera con extensión .h, contienen las declaraciones de funciones para el lenguaje de programación C, ya sea que estos archivos sean escritos por el programador o que ya se incluyan en el compilador.

Host.- Es cualquier tipo de computadora mono usuario o multiusuario, conectada en red, para proveer servicios y/o hacer uso de ellos: base de datos, contenido HTTP, correo electrónico, repositorio.

Iptables.- Políticas de seguridad para complementar el firewall de los sistemas tipo UNIX.

Installpkg.- Es un instalador de paquetes *.tgz diseñando en un principio para Slackware. Los paquetes precompilados de Beakos están armados bajo la estructura tgz, por lo que es posible instalarlos desde

LDAP.- Es un servicio de directorios distribuidos para almacenar diversos tipos de información, común mente utilizad para la Administración de Identidades dentro de una organización, es decir, proporciona servicios de autenticación y autorización para usuarios.

MD5.- Es un algoritmo criptográfico, los sistemas GNU/Linux lo emplean para garantizar que los paquetes de software alojados en la web, (comúnmente un repositorio) conserven su integridad y no puedan ser alterados por terceros sin que un usuario final se pueda percatar. Este mecanismo de cifrado ayuda a tener certeza de que el software descargado e instalado está libre de código malicioso.

MOTIF.- Es un estándar para interfaces gráficas de usuario, definido en la IEEE 1295, tiene como finalidad normalizar la presentación de una aplicación en diversas plataformas; se aplica principalmente para entornos UNIX. Proporciona una biblioteca y un conjunto de herramientas que facilitan el desarrollo de aplicaciones gráficas, también es la base para las toolkits de CDE.

Ncurses.- Es una biblioteca que permite la programación de interfaces basadas en texto.

NFS.- Network File System, es un espacio de almacenamiento o sistema de archivos compartido, accesible desde la red, (comúnmente LAN), trabaja bajo el modelo cliente-servidor, donde un *host* servidor comparte un sistema de archivos a diversos *hosts* clientes con permisos de lectura y/o escritura.

OpenSSH.- Es una aplicación desarrollada por OpenBSD para realizar conexiones seguras tipo cliente/servidor entre equipos de una red. Utiliza el protocolo SSH debido a que la información viaja encriptada. OpenSSH está licenciado bajo la GUN GPL.

Posix.- *Portable Operating System Interface*, es un conjunto de estándares para la interfaz de sistemas operativos, definido por la IEEE basado en el sistema UNIX, tiene el objetivo de garantizar la portabilidad de las aplicaciones entre diferentes plataformas de sistema operativo. Aplicable para UNIX, Linux y Windows NT. Surge de los estándares de SUS (Single Unix Specification) conocido como IEEE 1003, a diferencia de SUS (que es libre), la IEEE cobra por el acceso a la documentación de POSIX.

Removepkg.- Es un programa similar a installpkg a diferencia de que installpk es para instalar paquetes .tgz y removepkg para desinstalarlos.

Samba.- Es una herramienta de software que permite a los sistemas basados en UNIX, como GNU/Linux, comunicarse con otros sistemas operativos, principalmente Windows. Es una implementación de código abierto del conjunto de protocolos SMB (*Server Message Block*) y CIFS (*Common Internet File System*) ambos protocolos de red desarrollados por Microsoft.

Script.- Es un conjunto de instrucciones o comandos reconocidos por el sistema operativo, plasmados en un archivo de texto plano, para la ejecución de tareas rutinarias específicas. Dichos scripts son leídos por un intérprete de comandos como bash.

Repositorio.- Es un servidor de paquetería de software, trabaja bajo el modelo cliente-servidor, se utiliza comúnmente en las distribuciones GNU/Linux para que los host clientes puedan descargar, instalar/actualizar paquetería adicional.

SFTP.- Es la implementación segura del protocolo FTP, que corre bajo SSH, proporciona una conexión encriptada hacia un servidor para la transferencia y manipulación de archivos, se conecta por el puerto 22.

Shell.- Intérprete de comandos que permite la interacción entre el sistema operativo y el usuario, también es utilizado como un lenguaje de programación para la creación de guiones o scripts que son instrucciones para automatizar o programar ciertas tareas. Dentro de UNIX y los sistemas similares a este existen dos grupos de intérpretes de comandos los basados en Bourne: BSH, KSH o BASH, y los basados en el interpréte C: CSH ó TCSH.

Single UNIX Specification.- Es el conjunto de estándares libres que definen las características a las que se debe apegar un sistema operativo para ser considerado de tipo UNIX.

Snort.- Es una herramienta de software (*sniffer*) que permite inspeccionar los paquetes que viajan a través de una red y permite la detección de intrusos.

SSH.- Secure Shell es un sistema de inicio de sesión encriptado, que es una alternativa a conexiones inseguras como telnet, rlogin, rcp, ftp, etc. Se conecta a otros equipos a través de una red, para ejecutar comandos y copiar archivos entre máquinas remotas. Existen actualmente dos versiones, la primera implementa un conjunto de algoritmos de cifrado que son vulnerables a contener agujeros de seguridad, sin embargo, la versión dos corrige las vulnerabilidades al utilizar un algoritmo de intercambio de llaves mejorado.

Unix.- Es un sistema operativo multiusuario desarrollado en 1969 por Ken Thompson, Dennis Ritchie y Joseph Osanna, para los Laboratorios Bell, sus principals características desde sus inicios fueron la portabilidad, facilidad de uso y eficiente administración de los recursos de hardware. Debido a su funcionalidad fue proporcionado a las universidades bajo una licencia de uso. Tiempo después debido a la popularidad que había ganado UNIX, Richard Stallaman desarrolló su propio sistema operativo libre basado en UNIX.

Wireshark.- Es una herramienta de software para la “escucha” de paquetes en la red, es decir, muestra de forma detallada el tráfico de paquetes que corre sobre una red de datos. Requiere de una tarjeta de red que pueda ser configurada en modo “promiscuo”.

Sistema X Window.- También conocido como X11 (el 11 se refiere a la última versión de este sistema), es una aplicación o gestor de ventanas que permite la interacción gráfica en red entre el usuario y uno o más host destino, es decir, trabaja sobre el modelo cliente-servidor. Esta desarrollado bajo los estándares CDE y Motif, dándole portabilidad al sistema. La comunicación entre un cliente y el servidor de X se realiza a través del protocolo llamado Xprotocol y este a su vez utiliza una biblioteca llamada Xlib que realiza las operaciones binarias necesarias para que el usuario pueda hacer uso de Xprotocol.

ANEXOS

1. CREAR UNA NUEVA PARTICIÓN UTILIZANDO FDISK

```
root[~] # fdisk /dev/sda

Orden (m para obtener ayuda): n
Acción de la orden
e   Partición extendida
p   Partición primaria (1-4)
p
Se ha seleccionado la partición 4
Primer cilindro (327-652, valor predeterminado 327):
Se está utilizando el valor predeterminado 327
Last cilindro, +cilindros or +size{K,M,G
} (327-652, valor predeterminado 652): +2G

Orden (m para obtener ayuda): p

Disco /dev/sdb: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders
Units = cilindros of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x131b4a38

/dev/sdb4             327         588     2100646   83  Linux

Orden (m para obtener ayuda): w
¡Se ha modificado la tabla de particiones!
```

Después de crear una nueva partición de disco es necesario darle un formato (ext3):

```
root[~]# mkfs.ext3 /dev/sdb4
```

1.1 Declarar la partición en el archivo fstab

```
root[~]# blkid /dev/sdb4
/dev/sdb4: UUID="26d99e69-8b43-47d8-aa36-00611df57169"
SEC_TYPE="ext2" TYPE="ext3"
```

```
root[~]# vi /etc/fstab

UUID=be8331b2-f6bd-4722-b00d-1031d61194ec /mnt/lfs ext3
defaults   1 2
```

1.2 Montar la partición

```
#mkdir /mnt/lfs
#mount -a
```

2. USO DE LAS PRINCIPALES OPCIONES DE SWARET

Sintaxis:

```
swaret --accion [palabra_clave] [opción]
```

Ejemplos:

root[~]# swaret --search ssh

Busca en el repositorio las coincidencias del nombre de aplicación ingresado.

root[~]# swaret --install openssh -a

Instala desde el repositorio, la aplicación/es de *software* que coincida/n con la palabra ingresada, la opción **–a** es para hacer la instalación de forma automática, es decir, sin pedir confirmación al usuario.

root[~]# swarte --reinstall openssh

Reinstala todas las aplicaciones que coincidan con la palabra ingresada.

root[~]# swarte --remove openssh

Desinstala del sistema, la aplicación/es que coincida/n con la palabra ingresada. En este caso si no se hace uso de la opción **–a**, el sistema pedirá la confirmación al usuario para cada aplicación se vaya a desinstalar.

root[~]# swaret --get openssh

Descarga en el sistema (sin instalar) el paquete/s que coincida/n con la palabra ingresada.

```
root[~]# man swaret
```

Muestra la documentación detallada del gestor de paquetes, para una mayor referencia sobre su uso.

3. PROGRAMAS DE SOFTWARE QUE INTEGRAN AL SISTEMA LINUX PRIMARIO BASADOS EN LA VERSIÓN 6.4 DEL LFS

1.	Binutils-2.20.1 - Pass 1	8. Tcl-8.5.8	15. Diffutils-3.0	22. M4-1.4.14
2.	GCC-4.5.1 - Pass 1	9. Expect-5.44.1.15	16. File-5.04	23. Make-3.82
3.	Linux-2.6.35.4 API Headers	10. DejaGNU-1.4.4	17. Findutils-4.4.2	24. Patch-2.6.1
4.	Glibc-2.12.1	11. Ncurses-5.7	18. Gawk-3.1.8	25. Perl-5.12.1
5.	Adjusting the Toolchain	12. Bash-4.1	19. Gettext-0.18.1.1	26. Sed-4.2.1
6.	Binutils-2.20.1 - Pass 2	13. Bzip2-1.0.5	20. Grep-2.6.3	27. Tar-1.23
7.	GCC-4.5.1 - Pass 2	14. Coreutils-8.5	21. Gzip-1.4	28. Texinfo-4.13a

La instalación de la paquetería debe seguir el mismo orden en el que se encuentra listada.

4. MÓDULOS DEL KERNEL HABILITADOS PARA BEAKOS GNU/LINUX

Módulo	Tamaño	Requerido por
ipv6	201541	34
snd_pcm_oss	28083	0
snd_mixer_oss	10218	2 snd_pcm_oss
iptable_mangle	904	0
ipt_REJECT	1505	0
ipt_LOG	3883	0
ipt_MASQUERADE	1126	0
nf_nat_ftp	1056	0
iptable_nat	2690	0
nf_nat	10487	3 ipt_MASQUERADE,nf_nat_ftp,iptable_nat
xt_state	903	1
nf_conntrack	4109	1 nf_nat_ftp

_ftp		
nf_conntrack_ipv4	7331	4 iptable_nat,nf_nat
nf_conntrack	38613	7 ipt_MASQUERADE,nf_nat_ftp,iptable_nat,nf_nat,xt_state,nf_conntrack_ftp,nf_conntrack_ipv4
nf_defrag_ipv4	779	1 nf_conntrack_ipv4
iptable_filter	812	1
ip_tables	7723	3 iptable_mangle,iptable_nat,iptable_filter
x_tables	8444	8 iptable_mangle,ipt_REJECT,ipt_LOG,ipt_MASQUERADE,iptable_nat,xt_state,iptable_filter,ip_tables
btusb	7980	0
ppdev	4114	0
parport_pc	15955	0
bluetooth	39347	1 btusb
snd_ens1371	13242	1
gameport	5885	1 snd_ens1371
snd_rawmidi	12143	1 snd_ens1371
snd_seq_device	3713	1 snd_rawmidi
parport	21519	2 ppdev,parport_pc
snd_ac97_codec	78772	1 snd_ens1371
rfkill	10384	1 bluetooth
joydev	6696	0
ac97_bus	706	1 snd_ac97_codec
snd_pcm	48254	3 snd_pcm_oss,snd_ens1371,snd_ac97_codec
snd_timer	12406	1 snd_pcm
snd	32456	8 snd_pcm_oss,snd_mixer_oss,snd_ens1371,snd_rawmidi,snd_seq_device,snd_ac97_codec,snd_pcm,snd_timer
soundcore	3399	2 snd
snd_page_alloc	4969	1 snd_pcm
serio_raw	2912	0
vmware_balloon	3030	0
psmouse	28083	0
intel_agp	20064	1
agpgart	19468	1 intel_agp
evdev	5601	7
mac_hid	2157	0
shpchp	21055	0

pci_hotplug	8034	1 shpchp
pcspkr	1211	0
i2c_piix4	7052	0
ext3	93224	2
jbd	31843	1 ext3
mbcache	3802	1 ext3
dm_mirror	10621	0
dm_region_hash	5604	1 dm_mirror
dm_log	6532	2 dm_mirror,dm_region_hash
dm_snapshot	23980	0
dm_mod	45519	3 dm_mirror,dm_log,dm_snapshot
sg	19257	0
sd_mod	22664	4
mptspi	9543	3
mptscsih	14056	1 mptspi
floppy	41607	0
mptbase	48285	2 mptspi,mptscsih
scsi_transport_spi	15301	1 mptspi
ehci_hcd	29221	0
ide_cd_mod	20381	0
cdrom	24815	1 ide_cd_mod
scsi_mod	105899	5 sg,sd_mod,mptspi,mptscsih,scsi_transport_spi

5. PAQUETERÍA DEL SISTEMA BASE DE BEAKOS GNU/LINUX

Paquete	Versión
linux Kernel	2.6.34
MesaLib	7.6
Mysql	5.1.47
Nautilus	2.30.1
Nvidia	--

Openssh	4.6p1
Openssl	0.9.8g
Perl	5.10.0
Postfix	2.5.1
Python	2.5.2
Samba	3.5.4
Vim	7.2
Xfdesktop	4.8.0
Xine-lib	1.1.15
Xorg-server	1.7.1
Gcc	4.3.2
Automake	1.10.1
Cmake	2.8.2
Make	3.81
Cups	1.4.5

6. SERVICIOS DEL SISTEMA QUE SE EJECUTAN AL ARRANCAR UN SISTEMA BASE DE BEAKOS GNU/LINUX:

Lvm-tools: Utilerías requeridas para las funciones de gestión de particiones lógicas en el disco.

Sysklogd: Utilerías de registro del Sistema, este servicio captura en una serie de archivos la bitácora de actividades y mensajes que arroja un sistema Linux durante su ejecución.

Portmap: Es el servicio que se encarga de la asignación de puertos para las llamadas de **RPC** (***Remote Procedure Call***); portmap utiliza el demonio llamado rpcbind para la gestión de sus tareas.

Iptables: Este servicio, mantiene la ejecución del cortafuegos de Linux.

Random: Este servicio permite la generación de números de forma aleatoria

Netfs: Se encarga del montaje y desmontaje de los sistemas de archivo en red, tales como: **NFS** (***Network Filesystem***), **samba**, **NCP**. Trabaja a través del script: **/etc/init.d/netfs**.

Nas: Por sus siglas en ingles se refiere a ***Network Audio System***, es una aplicación con arquitectura cliente-servidor para la gestión de audio.

Fcron: Servicio que monitorea la ejecución de tareas programadas.

Cups: Este servicio mantiene disponible la gestión de dispositivos de impresión conectados en red, así como, la impresión de documentos.

D-Bus: Es un Sistema de comunicación entre procesos, ayuda a coordinar el ciclo de vida de un proceso, proporciona un demonio para el sistema y otro por cada usuario conectado, media la ejecución algunos servicios y demonios bajo demanda.

Acpid: Es un demonio de los sistemas Linux, para administrar los eventos **ACPI** (***Advanced Configuration and Power Interface****);* este demonio trabaja sobre /proc/acpi/event para identificar (escuchar) los eventos de **ACPI** y los manipula a través de la ejecución de determinadas aplicaciones. De entre los eventos más

comunes que son administrados por este demonio se encuentran: presionar teclas especiales (apagado/suspendido/hibernación), al cerrar la computadora portátil, al conectar algún dispositivo periférico, etc.

Avahidaemon: Servicio para encontrar otros nodos (equipos de cómputo o impresoras) que se encuentren dentro de la red local a la que se está conectado, para establecer una comunicación.

Haldaemon: Este demonio se encarga de gestionar los eventos del programa **HAL** (***Hardware Abstraction Layer***), a través del monitoreo de cambios en el *hardware* de los equipos de cómputo.

Samba: Aplicación tipo cliente-servidor, para compartir archivos entre sistemas **GNU/Linux** y **Windows**

Sshd: Este servicio permite realizar conexiones remotas seguras a través del protocolo **ssh**.

Network-manager: Este servicio mantiene la ejecución del administrador de redes, es decir, agregar, eliminar o modificar una conexión de red. Se representa a través una interfaz de usuario.

Gdm: ***Gnome Display Manager***, este servicio gestiona el inicio de sesiones y la carga de perfiles en el sistema, se comunica con otros programas para validar la autenticación de los usuarios.

BIBLIOGRAFÍA

[1] Antonio Ariño Villarroya *El movimiento open: la creación de un dominio público en la era digital*, Valencia, Universidad de Valencia, 2009.

[2] José María Sainz de Vicuña Ancín, *El plan estratégico en la práctica*, México, Alfa Omega Grupo Editor, 2012, Segunda Edición Revisada y Actualizada.

[3] Óscar Díaz Fouces, Marta García González, *Traducir (con) software libre*, Granada, Editorial Comares, 2009.

[4] Pressman, Roger S., *Ingeniería del software: un enfoque práctico*, traducción, Jesús Elmer Murrieta Murrieta, Eloy Pineda Rojas, Víctor Campos Olguín, México, McGraw-Hill, 2005, Sexta Edición.

[5] Ricardo Hernández Jiménez, *Administración de la función informática; una nueva profesión*, México, Limusa, 2003, Primera Edición.

[6] Roberto Feltrero Oreja, *El software libre y la construcción ética de la sociedad del conocimiento*, Barcelona, Icaria, 2007.

[7] S/autor, *Edición especial linux máxima seguridad*, Madrid, Pearson Educación, 2000.

[8] Steve Shah; Wale Soyinka, *Manual de administración de linux*, México, Mc Graw Hill, 2007, Cuarta Edición.

Fuentes de información electrónicas

[1] Free Software Foundation, *¿Qué es el software libre?*, [en línea], Dirección URL: http://www.gnu.org/philosophy/free-sw.html, consultada el 4 de Septiembre de 2013.

[2] Free Software Foundation, *Categorías de software libre y software que no es libre*, [en línea], Dirección URL:

http://www.gnu.org/philosophy/categories.html, consultada el 8 de Septiembre de 2013.

[3] Free Software Foundation, *Software libre y educación,* [en línea], Dirección URL: http://www.gnu.org/education/education.html, consultada 9 de Septiembre de 2013.

[4] Gerard Beekmans, *Linux from scratch versión 6.4*, [en línea], Linux From Scratch, 2008, Dirección URL:

http://www.linuxfromscratch.org/lfs/view/6.4/index.html, consultada el 4 de Junio de 2013.

[5] Iván Jiménez Alcantar, *La importancia de las metodologías ágiles*, [en línea], Software Gurú, México, 2012, Dirección URL: http://sg.com.mx/buzz/la-importancia-las-metodolog%C3%ADas-%C3%A1giles#.U4Jo7vl5Nf1, consultada el 17 de Noviembre de 2013.

[6] Jorge Heras, *El rol del manager en equipos ágiles*, [en línea], Software Gurú, México, 2014, Dirección URL:

http://sg.com.mx/sgvirtual/6/sessions/el-rol-del-manager-equipos-agiles#.U4JbY_l5Nf0, consultada el 25 de Febrero de 2014.

[7] Vidyasagar Potdar, Elizabeth Chang, *Open source and closed source software development Methodologies* [en línea], Curtin University of Technology, Australia, Dirección URL: http://www.researchgate.net/publication/42428079_Open_Source_and_Closed_Source_Software_Development_Methodologies/file/9c96052b28b8e8192e.pdf, consultada el 19 de Abril de 2014.