UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE QUÍMICA

"Implementación de software libre en la ingeniería química"

TRABAJO QUE PARA OBTENER EL TITULO DE INGENIERO QUÍMICO PRESENTA:

Juárez Contreras Alejandro

México, D.F. junio de 2011

Jurado asignado:

Presidente: I.Q. Eugenio León Fautsch Tapia

Vocal: M.C. Caritino Moreno Padilla

Secretario: I.Q. Antonio Francisco Díaz García

1er suplente: I.Q. José Agustín Texta Mena

2do suplente: I.Q. Ernesto José Calderón Castillo

Facultad de Química

Asesor de tesis

_____

Eugenio León Fautsch Tapia

Sustentante

_____

Alejandro Juárez Contreras

Dedicatoria

Que el 3,1415926535... con todo $\rho=a(1+\cos\theta)$.

A mis padres quienes me han apoyado siempre y que han sido bendición a mi vida.

A mis hermanos que me han enseñado lo hermoso de la vida.

A mis amigos quienes de una u otra forma me han acompañado siempre en mis tristezas y alegrías.

A la UNAM por ser mi segundo hogar.

A mis queridos maestros y a los que simplemente fueron mis profesores.

Por supuesto que a RMS gran hacker[1] de sombrero blanco, por su gran aportación a la humanidad.

A los hermanos argentinos de Tuxinfo.

A toda la gente involucrada con el movimiento del software libre y que han aportado a la humanidad su amor y compromiso.

A todos aquellos que a través de compartir su tiempo y conocimientos hicieron posible esta tesis.

A la memoria del prof. Roldan Parrodi y del prof. Rosas.

A la tía “Caro”.

1 Por “hacker” se refiere a programadores fervientemente dedicados, por hobby, a explotar sus ordenadores al máximo, con resultados útiles para otras personas. Este concepto es contrario al habitualmente aceptado, que dice que un “hacker” es un pirata informático.

Tomado del libro

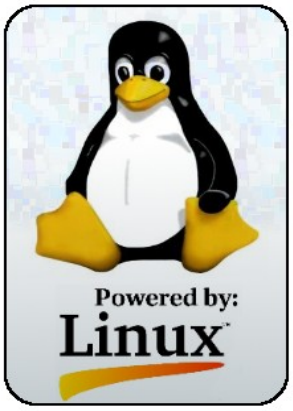
Linux: Instalación y Primeros Pasos Copyright Oc 1992-1996 Matt Welsh (Traducción: Proyecto LuCAS) Versión 2.2.2 - En castellano ver1.0, 8 Agosto de 1996. Fecha de montaje: 10 de noviembre de 1996

Así pues con esta tesis se espera poder ayudar a lograr un primer acercamiento hacia el uso de las herramientas de software libre; por lo cual no se pretende en ningún momento el profundizar por completo en cada uno de los puntos tocados dentro de la misma. Más bien se trata de lograr que quien lea este trabajo, se sienta atraído al uso de las mismas, a la participación, aprendizaje y difusión de esta filosofía en su vida, en su carrera y en el aporte científico de nuestro país y de la UNAM.

INDICE

Introducción..... 1
Observaciones acerca de este material.....4
Objetivos..... 5
Justificación.....6
Unas notas sobre UNIX.....7
Antecedentes..... 10
¿Qué es GNU y la Free Software Foundation?.....11
Filosofía del software libre.....11
Clasificación de personas relacionadas con el mundo del software.....12
Tipos de sombreros..... 13
Tipos de licencias..... 14
¿Para qué conocer las implicaciones de las licencias?.....15
Casos conocidos de mala utilización de software comercial..... 16
Virus,casos y cosas 18
Clasificación de virus y otros peligros..... 19
Notas importantes acerca del sistema GNU/Linux..... 21
Uso básico de la terminal en GNU/Linux..... 22
Algoritmos 23
Diagramas de flujo..... 23
Algunos pequeños consejos antes de comenzar..... 25
Estructuras de programación 25
Estructuras selectivas 26
Estructuras anidadas 28
Estructura repetitivas 29
Errores 37
Algunas precauciones para no cometer errores 38
Tipos de variables 39
Gfortran..... 40
Como compilar 42
Unas notas acerca de Gfortran 43
Bucles en programación Gfortran 46
Gpc..... 51
Gcc..... 63
“bc”(basic calculator) 72
Python.....79
Geany.....82
Hoja de cálculo de OpenOffice..... 85
Algunas notas sobre LibreOffice.....89
Octave.....90
Maxima 95
R..... 97
Scilab.....99
Aplicaciones del software libre en ingeniería química..... 101
Conversiones..... 103
Trigonometría..... 103

Dibujo de gráficas de gases reales.....103
Solución numérica de ecuaciones diferenciales.....111
Tanques de mezclado.....114
Columnas de destilación.....120
Flash adiabático.....123
Ejemplos de funciones avanzadas de OpenOffice 127
Solver en OpenOffice 127
Programación en Calc con OpenOffice 128
Conclusiones.....130
Propuesta plan de estudio para la materia optativa software libre en ingeniería química.....132
Método de aproximaciones sucesivas 135
Método de Newton-Rapshon 136
Palabras reservadas.....137
Tabla Hechos importantes en la historia de UNIX.....139
Historia gráfica del desarrollo UNIX.....140
Notas finales 141
Bibliografía.....148



Introducción

Esta tesis pretende dar una introducción a la filosofía e implementación del software libre en ingeniería química por medio del uso de diversas “tecnologías libres” y a través del código libre para la resolución de problemas; de hecho, casi toda esta tesis está integrada del mismo para su estudio; y es por medio de este punto de vista que se pretende el facilitar aún más la incursión por parte de un ingeniero químico u otra persona a cualquier otro lenguaje de programación.

Esta tesis está dirigida hacia los ingenieros químicos que cuenten con poco o nada de conocimiento acerca del tema de programación; sin que por ello se dejen de lado a las personas que quieran estudiar por su propia cuenta acerca de este tema tan útil.

Tampoco se deja de lado el uso de diagramas de flujo para el desarrollo de programas pues este es un paso necesario para la utilización correcta de todas estas herramientas.

Más que una simple exposición de ideas esta tesis pretende ser una herramienta de trabajo tanto en el aula como en el quehacer diario de la vida del Ingeniero Químico que se prepara en la UNAM. Y para todas aquellas personas que quieran aprender el manejo de un buen numero de programas libres.

Se tocan puntos tan diversos como la filosofía e historia del software libre, el uso de la terminal en Linux; así como la necesidad de libertad en la creación de software libre para los países en vía de desarrollo.

Por otro lado no se pretende en ningún momento hacer un sinnúmero de aplicaciones enfocadas a la Ingeniería química, pues seguramente ese trabajo llevaría tanto tiempo que tal vez nunca se podría abarcar tanto material; más bien se trata de demostrar que el uso de estas herramientas es tan válido como las de tipo comercial y que se trata de una lástima que las mismas pasen desapercibidas por muchos de los Ingenieros químicos en el país.

Seguramente podremos ver en los años venideros no con cierta pena que muchos de los países que están apostando a este tipo de tecnologías gozarán de adelantos tecnológicos muy importantes, más libertad en cuanto a sus propias herramientas, y de mayor independencia a nivel internacional, mientras que los que no lo hicieron así tendrán que seguir dependiendo de terceros para seguir “compitiendo” a nivel mundial.

¿Para qué sirven y cómo se pueden implementar cada una de estas herramientas? La respuesta es simple. si se conocen las bases de su utilización, tan solo la imaginación e ingenio, podrán ser el limite; además de que servirían para algo aún mucho más importante, el de lograr ser más competitivos, en el mundo laboral a nivel nacional y mundial, hecho que redundaría en beneficio de toda la sociedad.

Por otro lado la ingeniería química nació cuando la herramienta científica de cálculo más avanzada era la regla de cálculo y fue esta la que en sus inicios le ayudo en la resolución de problemas; posteriormente con la calculadora electrónica los tiempos de la ejecución de las operaciones matemáticas disminuyeron pero fue necesario desarrollar procedimientos para el control del error que se podía tener al utilizar la herramienta.

Con la aparición de la computadora indudablemente la rapidez en el cálculo aumento, lo que hizo posible la solución de mayor cantidad de problemas en este arte-ciencia que es la ingeniería química. Hay mucha seguridad de que este tipo de soluciones seguirá evolucionando de una forma notable en los países que se involucren tanto en el desarrollo de software libre, como en el de hardware, debido a que estos no dependerán exclusivamente de una solución impuesta y si de las que puedan desarrollar e implementar por ellos mismos.

Hay que reconocer que las computadoras producen lo que podemos llamar una sinergia; por así decirlo, de la cual un conjunto de circuitos electrónicos, de sistemas operativos, y de programas producen una serie de resultados muy significativos en la solución de problemas.

Aunque no es posible el dar una fecha exacta en la que UNIX[2] se volvió un software de tipo privado; este hecho se hizo más tangible a partir de 1981 (si bien antes de este hecho la versión 6 ya se comercializaba por medio de una licencia durante los años 1974 y 1975); a partir del desarrollo del UNIX System III por parte de AT&T, el cual se baso en la versión 7 de UNIX. Éste código antes de su restricción de uso, fue utilizado para impartir clases de programación e implementación de sistemas operativos.

Con la aparición de restricciones en el código UNIX, este dejo de utilizarse en muchas de las universidades del mundo que lo empleaban; lo cual origino por parte de Andrew Tanenbaum la creación del sistema operativo llamado MINIX, mismo que en buena parte substituyo a UNIX; este sistema contaba con muchas similitudes en la estructura y operación del sistema UNIX, pero tenía el inconveniente de ciertas limitaciones técnicas y otras tantas en cuanto a la libertad absoluta para obtener su licencia que solo se reducía a la parte académica.

Fue en el año de 1984 que Richard Stallman hizo su aparición publica para buscar una solución al problema con las limitaciones de los derechos de autor (copyright), problemática que se tratará más adelante. Años después,en 1991 Linus Torvalds comenzó a escribir en base al curso sobre MINIX que tomo bajo la dirección de Andrew Tanenbaum un nuevo sistema operativo que no contuviera nada de software restringido[3], y que además fuera un clon de UNIX, todo esto escrito en lenguaje C. Esta idea inicialmente fue redactada como proyecto de tesis, poco después fue liberada bajo licencia GNU (GNU NOT UNIX), la cual se basa en las cuatro premisas del código libre que había creado Richard Stallman.

Al quedar liberado el código a la comunidad este sufrió grandes y muy rápidos cambios, mismos que han seguido hasta nuestros días, agregándole nuevas funcionalidades y mejoras, todo esto se ha compartido con el resto del mundo. Otro hecho notable es que la llegada de la Internet marcó el nacimiento de GNU/Linux y el comienzo de una gran cooperación a nivel internacional, sin la cual el sistema del “pingüino” no sería para nada lo que significa hoy.

2 Esta palabra proviene de UNICS, acrónimo de **Un**iplexed **I**nformation and **C**omputing **S**ystem la cual se le atribuye a Brian Kernighan, un hack de MULTICS (Multiplexed Information and Computing Service. Dado que a UNICS se le consideraba un sistema MULTICS pero castrado, debido a la palabra homófona eunuchs; se le cambió el nombre a UNIX, para evitar esa ironía. http://es.wikipedia.org/wiki/Debian

3 También denominado privado, aunque Richard Stallman lo denomina privativo. Términos que se utilizaran a lo largo de esta obra. http://www.youtube.com/watch?v=7UKksU4EIRY video 1 , http://www.youtube.com/watch?v=KefkyDZxdlQ&feature=related video 2 ; http://www.youtube.com/watch?v=uW_haEgy-4E&feature=related video 3

Por supuesto que los sistemas operativos actuales requieren miles de horas-hombre para poderse crear y soportar; Además suponen una tarea ardua de corregir los problemas que se generen; es obvio que nadie en este mundo tiene la capacidad de revisar a conciencia los miles de millones de líneas que se han generado para la creación de estos, a menos que todo este trabajo se distribuya entre los miles o millones (tal vez), de programadores de todo el mundo; claro si esto no infringiera el “copyright”.

Pero esto genera también una disyuntiva al estarse usando sistemas con códigos cerrados, estos pueden contener “códigos maliciosos” u otros problemas como “puertas traseras” diseñadas por sus mismos programadores.

Una vez dicho esto, existe la posibilidad de ver a las computadoras como hermosas “cajas negras” en las cuales podemos “confiar” para realizar nuestros cálculos. Hemos llegado a depender mucho de estas “cajas negras” en la cuales simplemente metemos números esperando obtener resultados, sin siquiera saber cómo fueron diseñadas. Si bien esta es una forma práctica y sencilla de ver la vida, no es conveniente del todo. Es como si simplemente esperásemos que los demás hicieran nuestro trabajo sin siquiera saber qué es lo que están haciendo y como lo están realizando. El efecto de ser práctico debe de llevar una buena porción de desconfianza.

La sociedad requiere de quien desarrolla este tipo de cajas negras posea una ética profesional muy grande, que permita con un grado de seguridad suficiente confiar en que sus datos e información estén a salvo, y por ende se eviten todas las violaciones a la privacidad. En la actualidad se hace uso de una gran cantidad de máquinas, las cuales contienen en su interior software y sistemas operativos de tipo privativo, del cual se desconoce su código.

Un sistema operativo según Andrew S. Tanenbaum (creador de MINIX), en su libro “Sistemas operativos, diseño e implementación” es el que controla todos los recursos de la computadora y ofrece la base sobre la cual pueden escribirse los programas de aplicación.

Entre los sistemas operativos actuales se puede optar por utilizar a GNU/Linux y propiamente DEBIAN[4], como base para hacer una distribución dirigida a estudiantes de ingeniería química. Entre las razones más importantes para optar por este sistema es que se trata de un sistema operativo muy sencillo de instalar, aprender, difundir y estudiar. Además que este se puede descargar en forma gratuita de la red y es un clon de los sistemas operativos de tipo UNIX.

4 La palabra DEBIAN proviene de la conjunción de los nombres de su fundador Ian Murdock y el de su ahora exesposa Deborah; esta distribución fue fundada en el año de 1993. Es una de las distribuciones GNU/Linux por excelencia. http://www.srbyte.com/2008/08/feliz-cumpleaos-debian.html

La filosofía de esta distribución está basada en las siguientes cinco premisas de su manifiesto que a su vez se derivan de las cuatro premisas de Stallman:

1. El sistema “DEBIAN” será absolutamente libre, tanto por su contenido, como por su temporalidad.

2. El objetivo será el de contribuir a la comunidad del software libre, por lo que DEBIAN por su parte liberará las herramientas que desarrolle, difundirá al máximo su uso, comunicara los cambios, solucionará errores y propondrá mejoras a los creadores de software libre.

3. Los usuarios se beneficiaran del uso, pero tendrán la posibilidad de contribuir como desarrolladores del mismo sistema DEBIAN, el cual mantendrá una base de datos de acceso público donde se dé cuenta de los errores que reporten los usuarios, No ocultándose ninguna clase de problema para que sean inmediatamente conocibles por el resto de los usuarios.

4. Las prioridades son los usuarios y el crecimiento del software libre.

5. DEBIAN no impedirá en su sistema el uso de software no libre.

Además de que al estar los códigos abiertos al público en general, se puede tener la posibilidad de generar implementaciones propias bajo la entera certeza de hacerlo de forma segura y transparente.

Observaciones acerca de este material

Una observación muy importante acerca de la codificación de los programas de ejemplo que se encuentran en esta tesis, en que en estos se podrán notar que existen algunas supuestas “faltas ortográficas”, pero estas son justificadas en razón de que no existen ciertos símbolos en los compiladores, como la ñ y los acentos, lo cual se debe a que se utilizaron en su mayoría traducciones inglesas para su creación e implementación.

Mucha de esta información ha nacido de la lectura de manuales y libros que con mucho esfuerzo han ido aportando diversas personas de muchas nacionalidades, estratos sociales, y distintos niveles educativos, los cuales han perseguido un mismo fin en común, el de ayudar a sus semejantes con su trabajo.

Esta tesis en alguna forma es una invitación para que se aporten nuevas ideas a los proyectos de software libre con la finalidad de mejorar las condiciones de la Universidad y nuestro país.

No se puede dejar que se pierda la libertad de crear, estudiar, modificar y aun ejecutar implementaciones propias de software y eso sólo se puede lograr usando software libre. Por lo tanto no se puede depender completamente de software de tipo privativo, es una obligación como ingenieros el aprender a implementar soluciones que apoyen a todas las premisas GNU.

Cada uno de los programas realizados para esta tesis fueron codificados y revisados cuidadosamente para su correcta ejecución dentro del sistema GNU/Linux; utilizándose como base código libre sin haberlo copiado tal cual, para lo que se requirió de muchas horas de arduo trabajo e investigación. Se dice que “no hay nada nuevo debajo del sol”[5]; pero se pretende mostrar la utilidad y validez de este tipo de herramientas para el ingeniero químico.

5 Eclesiastes 1:9 La santa biblia, sociedades bíblicas unidas antigua versión de Casiodoro de Reina revisada por Cipriano de Valera, revisión de 1960

Objetivos

El trabajo que se desarrolla está encaminado a:

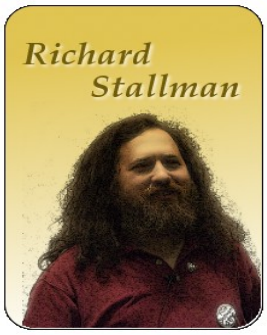
Mostrar un panorama de la filosofía del software libre, realizando un análisis crítico de sus beneficios y sus perjuicios, así como sus límites y alcances.

Promover el uso del software libre para su uso en el ejercicio de todas las profesiones en especial en la carrera de ingeniería química.

Difundir su uso, mediante la presentación de ejemplos claros de algunos programas de software libre, para ayudar al entendimiento e implementación de soluciones computarizadas.

Contribuir en forma práctica a la discusión sobre el diseño o reestructuración de los planes de estudio de la carrera de ingeniería química, incorporando contenidos relacionados con el software libre

Apoyar con este material a usuarios principiantes y autodidactas para que puedan acceder a estas herramientas.



Justificación

El mundo se ha vuelto muy dependiente de las soluciones integradas en “cajas negras”. Si bien es cierto que muchas de las compañías que las producen se han posicionado con los años y la experiencia en las implementaciones informáticas, las cuales las han vuelto “indispensables”, es importante que se tome una política de estado y social sobre la investigación e implementación de estas herramientas tecnológicas. A los Ingenieros químicos compete también el comenzar a indagar sobre estos recursos con el fin de utilizarlos dentro de su quehacer científico y tecnológico.

GNU/Linux es una buena propuesta para ser utilizado como base, porque ya desde hace mucho tiempo se vienen manejando sistemas de cómputo de tipo UNIX, en diversos países con gran eficiencia en el cálculo de implementaciones científicas y tecnológicas.

Por otro lado al utilizar los sistemas operativos de código abierto:

a) Se pueden revisar los códigos, pudiendo darse cuenta de la posibilidad de código malicioso.

b) Toda la comunidad se ve beneficiada con los avances en cuestión de mejoramiento del código.

c) Se poseen herramientas eficientes para la supervisión y mantenimiento evitándose al máximo posibles incursiones al sistema.

d) Se pueden utilizar códigos de encriptación para evitar robo de información.

e) La Universidad puede hacer uso legítimo de proyectos de Ingeniería química usando software libre.

f) Surge la creación de nuevas oportunidades para los egresados; con la posibilidad de crear nuevas ramas de ingeniería química.

Alrededor del mundo se están haciendo verdaderos esfuerzos por desarrollar y mejorar este tipo de herramientas. Es así como en muchos países se esta implementando este modelo de software para muchas de sus actividades y no conformes con eso, se están enfocando muchos recursos hacia el lado educativo, basta decir que China, Venezuela, Brasil, Cuba, España, etc. Han incursionado con magníficos resultados en ahorro monetario y en creación de nuevas tecnologías que los han retribuido en su avances tecnológicos, educativos y de seguridad como países.

La facultad de química no debería de perder de vista este tipo de desarrollos, dentro de sus planes de estudio, debido a que en algún futuro se encontraría en desventaja tecnológica con respecto a sus homologas de otras universidades alrededor del mundo.

Unas notas sobre UNIX

Al hablar de GNU/Linux es necesario reconocer que mucha de la filosofía que lo hace único nació de las primeras distribuciones de UNIX; dado que el movimiento hacker de hoy no seria el mismo sin las aportaciones de este.

Al igual que UNIX, el sistema operativo GNU/Linux tiene como base de su creación al poderoso lenguaje C.

Los dos comparten las posibilidades de un kernel limpio y portable; es decir, este tiene la posibilidad de implementarse en cualquier tipo de procesador o arquitectura de computo; además de ser ideal para poder ser estudiado y mejorado.

Conocer parte de la historia de este sistema operativo es muy importante debido a todas las aportaciones y gran influencia que dejo marcada en GNU/Linux. Es en la época de los 70 cuando hace aparición UNIX, como un desarrollo de tipo experimental y académico, lo cual era lo hacia de uso común en la mayoría de las universidades también debido a que se podía modificar libremente, ya que su código se encontraba libre y abierto a todo el mundo.

Es de hacer notar que en estos años debido a que este podía modificarse libremente aparecieron una gran diversidad de variantes de UNIX, las cuales provocaban problemas de compatibilidad y control (Hoy en día existe la posibilidad de que ocurra lo mismo, si no llega a existir un criterio de normatividad entre los estándares de GNU/Linux). Pero no paso mucho tiempo antes de que a UNIX se le viera como un verdadero negocio y entonces se le catalogara como secreto comercial, convirtiendo en privativo cada uno de las mejoras hechas por muchos de los hackers. Es por esto y haciendo otra vez mención a lo ocurrido en el pasado, la verdadera gran ventaja que tienen los sistemas de tipo GNU/Linux, estriba en la copyleft (concepto valido y legal); y por supuesto cualquier otro sistema operativo que aparezca y se acoja a la misma, evitando que este núcleo o cualquier otro y las distintas herramientas que lo compongan se vuelvan de tipo privativo como desgraciadamente ocurrió con UNIX.

El futuro de UNIX se veía muy prometedor como un sistema operativo altamente efectivo, y capaz de ser utilizado en cualquier tipo de tarea y procesador, pero por razones de tipo comercial en su versión 7 se decidió cerrar su código fuente, hecho que provocó que muchas universidades dejaran de impartir cursos basados con este sistema operativo, puesto que mucha de las mejoras que obtuvo fue a través de la comunidad de hackers, quienes ahora veían su propio trabajo como algo cerrado.

Al pensar con cuidado sobre la aparición del copyleft fue verdaderamente una idea genial, el que Richard Stallman propusiera crear un clon de UNIX; que contara con un sistema compatible con las implementaciones de tipo UNIX, que tuviera la posibilidad de ser multitarea y el cual además heredara la filosofía primera de los hackers. Al aparecer GNU/Linux muchos de los antiguos programadores de UNIX encontraron todo lo anterior; es decir, algo así como un oasis en el cual satisfacer todas sus necesidades de programador.

Otro hecho importante fue la aparición de MINIX, el cual es un sistema operativo compatible con UNIX versión 7, mismo que comparte el hecho de ser escrito en Lenguaje C, con kernel de tipo modular, este posee además un código fuente completamente detallado para su estudio, modificación y redistribución, con fines educativos y de investigación, solo que tenia algunas restricciones de tipo comercial. Resultara excelente darle un vistazo al libro escrito por Tanenbaum en el que explica a MINIX paso a paso y como fue desarrollado.

Poco tiempo después de la creación de MINIX Linus Torvalds toma clases con Andrew Tanenbaum creador de MINIX, lo cual le motiva a crear su propio sistema operativo, el cual fuera similar al de su profesor; primero como su proyecto de tesis y después como un proyecto con fines comerciales llamado FREAX; fue su amigo y colaborador Ari Lemmke quien lo convenció de llamarlo Linux; este sistema era capaz de correr aplicaciones del proyecto GNU; poco tiempo después el mismo Ari Lemmke lo convenció de usar la filosofía GNU y luego de esto Linus opto por liberar su núcleo bajo la licencia GNU/GPL, momento mismo en que nace GNU/Linux, y así es cuando los antiguos hackers que no querían perder su trabajo nuevamente como un secreto de tipo comercial se vieron cubiertos nuevamente. En este instante mientras me encuentro escribiendo esta tesis, hay millones de personas trabajando y beneficiándose por el trabajo en conjunto de hackers de todo el mundo.

A manera de pequeña reflexión si Andrew Tanenbaum hubiese apostado a MINIX por una licencia de tipo GNU/GPL seguramente hoy en día este estaría ocupando el lugar de GNU/Linux y estaríamos hablando de GNU/MINIX en esta tesis.

Hace unos cuantos años se dio un paso muy importante para la vida del núcleo GNU/Linux la apertura de parte del código de UNIX por parte de la empresa SUN bajo el nombre de OpenSolaris; así esta empresa además de beneficiarse del trabajo de desarrolladores de todo el mundo, dio lugar a que estos se beneficiaran con parte de las implementaciones creadas por la misma en beneficio de toda la comunidad.

Aunque este excelente modelo puede cambiar hoy en debido a que ORACLE compro la compañía SUN y parece tener la idea de volver a cerrar el código OPenSolaris, hecho por el cual desarrolladores de todo el mundo están haciendo esfuerzos para quitar por completo todas las partes privativas de este OpenSolaris y al igual que hace muchos años este UNIX se vuelva una opción viable entre los distintos sistemas operativos. Uno de los hechos mas positivos y que se deben de observar es que casi todas las implementaciones creadas durante este tiempo bajo este proyecto fueron de código libre y por tanto le pertenecen a la misma comunidad que las creó, sin que otros puedan llegar a quitárselas volviéndolas privativas nuevamente.

Se espera que ORACLE no deje de aportar cosas importantes a la comunidad, pues dejaría de crecer en gran parte debido a que son los programadores de código libre los que han aportado mucho de lo que en realidad era SUN y sin este apoyo los proyectos se verían retrasados por varios años de estudio; puede además que aunque se contraten a nuevos y buenos programadores, pero se vean seriamente limitados todos sus proyectos sin el aporte de la comunidad. Tal vez se una apreciación un tanto aventurada la de augurar lo siguiente, pero si ORACLE no da un vistazo amigable a la comunidad, esta empresa perderá mucho mas de lo que se ha imaginado en los siguientes años.

Hoy crece un nuevo proyecto nacido de la iniciativa de la comunidad bajo el nombre de ILLUMOS, tomando el código de OpenSolaris y depurando cada una de las partes privativas que este pueda contener, este seguirá beneficiando y ayudando a desarrollar aún más a GNU/Linux y viceversa. Este proyecto no se trata de un sistema operativo en sí mismo; si no más bien de la creación de un nuevo núcleo del cual se tienen ya varias distribuciones hijas, entre ellas: Nexenta, Schillix, etc. Al nacer del OpenSolaris de SUN este seguramente seguirá siendo uno de los UNIX mas avanzados del mundo.

Para el tiempo en que se esta acabando de corregir esta tesis la empresa ORACLE, debido al gran empuje de la comunidad en lograr un proyecto libre de código privado, y a que muchas de las distribuciones GNU/Linux han respaldado esta decisión, optando por otras soluciones ofimáticas tales como la de LibreOfice, están por liberarse por completo los códigos de OpenOfice, ya que este poseía algunas partes de tipo privativo y ORACLE al parecer pretendía dejar sin actualizaciones a esta herramienta.

Esta al igual que otras cosas es prueba de que la comunidad alrededor del mundo puede lograr alcanzar metas importantes para todos y cada uno de los miembros que la componen.

Antecedentes

Aproximadamente hace unas 3 décadas atrás se encontraba trabajando, compartiendo y colaborando en programas informáticos con sus compañeros en el laboratorio de Inteligencia Artificial del MIT, un científico llamado Richard Stallman (en esta época de la historia, era muy común el compartir los códigos generados entre los científicos para mejorarlos o bien utilizarlos en sus propios proyectos).

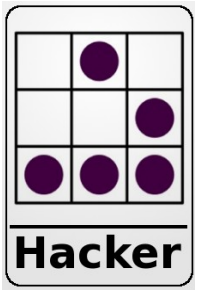
En una ocasión Richard Stallman se encontraba en problemas al querer hacer funcionar correctamente una impresora que se hallaba dentro de este laboratorio, la cual no tenia una señal para hacer saber que el papel se había atascado y al generar muchos problemas, Stallman se propuso solucionar este inconveniente para lo cual se encargo de contactar a los fabricantes, pidiéndoles que le facilitaran el código fuente de la misma para corregir los problemas que hubiera en este, sin recibir ningún dinero a cambio; los fabricantes por supuesto se negaron a dar tal información, posteriormente en un nuevo intento por solucionar sus problemas, Stallman pidió que se le facilitaran las especificaciones de la impresora, para hacer sus propios controladores, otra vez se negaron a darle tales especificaciones. Hecho que realmente enfureció a Stallman haciéndole concebir la idea de software libre.

Aparte de este inconveniente sus compañeros de laboratorio formaron una empresa llamada Synaptics, con la finalidad de substituir todo el software libre que se utilizaba dentro del laboratorio, por software de tipo privativo. Richard se opuso rotundamente a que se volvieran cerrados los códigos, pero sus compañeros intentaron convencerlo dándole la posibilidad de firmar un acuerdo substancioso, para que este aceptara mantener en secreto muchos de los códigos generados en el laboratorio, hecho que lo molesto profundamente, por lo cual el mismo se puso a programar el doble de tiempo haciendo un esfuerzo mayor para evitar que esto pasara.

Stallman cuenta así su propia decisión:

"La elección fácil era unirme al mundo del software propietario, firmar los acuerdos de no revelar y prometer que no iría en ayuda de mi amigo hacker. Es muy probable que desarrollara software que se entregaría bajo acuerdos de no revelar y de esa manera incrementara también las presiones sobre otra gente para que traicionaran a sus compañeros. Podría haber hecho dinero de esta manera, y tal vez me hubiese divertido escribiendo código. Pero sabía que al final de mi carrera al mirar atrás a los años construyendo paredes para dividir a la gente, sentiría que usé mi vida para empeorar el mundo". (http://www.gnu.org/thegnuproject.es.html)

Indignado por este tipo de actos decidió comenzar el proyecto GNU y un año después funda la Free Software Foundation (FSF)



¿Qué es GNU y la Free Software Foundation?

Richard Stallman tomo la iniciativa en 1984 de hacer un sistema operativo compatible con UNIX, escrito desde cero y que utilizara herramientas de tipo software libre, para lo cual acuño la palabra GNU la cual es un acrónimo el cual significa “GNU's Not UNIX” traducido al español es “GNU no es UNIX”.

La Free Software Foundation nació en 1985 para dar cobertura legal al proyecto y canalizar las ayudas económicas. En la actualidad la FSF promueve el desarrollo y uso del software libre, particularmente del sistema operativo GNU, y defiende los derechos de los usuarios a copiar, estudiar, modificar y redistribuir los programas informáticos.

Filosofía del software libre

El software libre es una cuestión de la libertad de los individuos para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Más precisamente, significa que los usuarios de este tipo de programas tienen como protección y normatividad a la GPL.

La Licencia Publica General (GPL); la cual contempla las siguientes cuatro libertades esenciales:

Libertad 0

- La libertad de ejecutar un programa con cualquier propósito.

Libertad 1

- La libertad de estudiar cómo trabaja el programa, y cambiarlo para que haga lo que usted quiera. El acceso al código fuente es una condición necesaria para ello.

Libertad 2

- La libertad de redistribuir copias para que pueda ayudar al prójimo.

Libertad 3

- La libertad de distribuir copias de sus versiones modificadas a terceros. Si lo hace, puede dar a toda la comunidad una oportunidad de beneficiarse de sus cambios. El acceso al código fuente es una condición necesaria para ello.

Pero no tan solo eso fue todo lo que se logro, esta filosofía ayudo también a revolucionar mucho del pensamiento de la sociedad actual en otros campos del saber y el arte; pues se trata de una hermosa filosofía, la de poder compartir los conocimientos generados con otros para el enriquecimiento de todos; es decir, hablar de libertad, es hablar de libertad de información, de libertad de saber que es lo que estamos utilizando y para que lo utilizamos, de los compromisos con todos los demás, es precisamente en este punto que el software libre ha logrado su mayor crecimiento en la filosofía de “ayudar a los otros, para crecer juntos”.

Esta es la verdadera filosofía del “hacker”.

Clasificación de personas relacionadas con el mundo del software.

Antes que nada para poder entender un poco más acerca de la filosofía del software libre comenzare por hacer uso de la definición que hace el finés Pekka Himanen[6] acerca del hacker, en su obra la ética del hacker y el espíritu de de la era de la información (la cual contiene un prologo de Linus Torvalds y un epilogo de Manuel Castells), este deberá de tener las siguientes características:

Pasión

Libertad

Conciencia Social

Verdad

Antifascismo

Anticorrupción

libre Acceso a la información

Valor Social

Por lo tanto podemos decir que un verdadero hacker es una persona interesada en el desarrollo, implementación, investigación y experimentación dentro del mundo del software.

Por otro lado el Cracker se trata de un pirata informático, vándalo o delincuente con altos conocimientos en informática, y poca conciencia ética.

Se han escrito un sinnúmero de artículos acerca de estos temas los cuales abordan desde muchos puntos de vista la ética que debe de tener un hacker e incluso uno de los grandes promotores del software libre Erick Raymond escribió el "Hacker Howto[7]" en el cual se explican todas las bases para convertirse en uno de ellos, desde que leer hasta que estilo de vida llevar.

En la actualidad se ha dado un fenómeno tremendo en el cual han aparecido nombres poco comunes para las personas que no estén muy familiarizadas con estos temas. No tiene mucho sentido enumerar y definir a cada uno de estos individuos pues no es la finalidad de esta tesis, sin embargo se menciona una clasificación que es peculiarmente interesante y cómoda para quien apenas se introduce en estos temas.

6 www.tallerh.com.ar/sitio/textos/hack.pdf
7 http://www.catb.org/~esr/faqs/hacker-howto.html

Tipos de sombreros

Para la clasificación que sigue se utilizan parte de las ideas definidas por Edward de Bono en su obra "Seis sombreros para pensar"[7]. En ella se plantea que se pueden tener visiones diferentes del papel que toca una persona y las asocia con el color del sombrero.

Con estas ideas, en la cultura de los ciberespacios se han acuñando diversos tipos de terminaciones en cuanto a los intereses y motivaciones que los hackers tienen; la siguiente es una forma de diferenciarlos en cuanto a su visión ética:

Sombrero blanco

Este tipo de hackers rompen las redes de seguridad con intenciones altruistas, estos se encargan por lo regular de trabajar con las empresas para encontrar debilidades en la seguridad de sus sistemas de computo o servidores. Estos trabajan con el fin de diseñar sistemas más seguros.

Sombrero gris

Este tipo de hacker tiene una ética ambigua, tanto puede usar sus conocimientos para ayudar como para hacer destrozos.

Sombrero azul

Es aquel que es contratado por las empresas creadoras de software con el fin de encontrar deficiencias en cuanto a la seguridad antes de su lanzamiento comercial y poder corregirlas a tiempo.

Sombrero negro

Un hacker de sombrero negro es aquel que se mete sin autorización dentro de una computadora o sistema computacional, por medio de un ordenador personal ya sea por una Intranet o vía internet con el fin de causar destrucción, terrorismo o tomar el control de la computadora desde un lugar remoto.

7 Aunque estos conceptos son muy semejantes en realidad difieren en poco de los sombreros definidos por Edward de Bono que son los siguientes:

Sombrero Blanco, blanco, virgen, hechos puros, números e información.
Sombrero Rojo, emociones y sentimientos, también presentimiento e intuición.
Sombrero Negro, abogado del diablo, enjuiciamiento negativo, razón por la que no resultará.
Sombrero Amarillo, luz del sol, brillo y optimismo, positivo, constructivo, oportunidad.
Sombrero Verde, fertilidad, creatividad, plantas brotando de las semillas, movimiento, provocación.
Sombrero Azul, moderación y control, director de orquesta, pensar en el pensamiento.

Tomado del libro

Seis sombreros para pensar Edward de Bono Granica Ediciones Página 93

Tipos de licencias

Existen muchos tipos de licencias que apoyan tanto el tipo de visión de un grupo como el de otro; por así decirlo existen tantos tipos de licencias como formas de software existe; algunas de estas licencias son:

Software libre o free software

Este software dispone de las cuatro libertades que son poder: modificarlo, utilizarlo, copiarlo y redistribuirlo, además pone a disposición de cualquiera el código fuente con que fue creado éste.

Copyleft.

El Copyleft, es una modificación del Copyrigth; de hecho esta tipo de licencia aunque es un caso serio, en el fondo se trata de una burla, por la falta de libertades que la Copyrigth no otorga. Debido a que el Copyrigth protege al autor de un programa de las posibles modificaciones y redistribuciones, sin su consentimiento las cuales son prohibidas por la normatividad internacional.

GPL. (General Public License)

La GPL es soportada por GNU. Esta licencia se encarga de cuidar que los programas creados bajo su protección no sean tomados por software de tipo propietario y se pierdan las libertades ganadas por la comunidad. Debido a que en otros años al integrar en el software propietario el software libre, y al no existir este tipo de seguridad simplemente, lo reclamaron como parte de su propios desarrollos, negando la posibilidad de volverlo a utilizar como software libre.

DEBIAN.

La licencia Debian es parte del contrato realizado entre Debian y la comunidad de usuarios de software libre, y se denomina DEBIAN Free Software Guidelines (DFSG). En esencia, esta licencia contiene criterios para la distribución que incluyen, además de la exigencia de publicación del código fuente: (a) la redistribución libre ; (b) el código fuente debe ser incluido y debe poder ser redistribuido; (c) todo trabajo derivado debe poder ser redistribuido bajo la misma licencia del original; (d) puede haber restricciones en cuanto a la redistribución del código fuente, si el original fue modificado; (e) la licencia no puede discriminar a ninguna persona o grupo de personas, así como tampoco ninguna forma de utilización del software; (f) los derechos otorgados no dependen del sitio en el que el software se encuentra; y (g) la licencia no puede “contaminar” a otro software.

Open Source.

La licencia de Open Source Initiative deriva de la licencia DEBIAN.

BSD.

La licencia BSD (Berkeley Software Distribution), es de tipo permisiva debido a que tiene muy pocas restricciones , ya que permite que el software creado bajo la misma pueda ser vendido, pudiendo o no contener el código fuente, una buena cualidad de la misma, es que esta licencia cuida del crédito de los autores del software; el gran problema reside en que no procura que el software o sus modificaciones sigan siendo libres y por lo tanto se pueden perder éstas; si así, lo dispone el que modifica su código.

Software propietario.

Como su nombre lo indica es aquel cuya copia, redistribución o modificación están, en alguna medida, prohibidos por su propietario. Para usar, copiar o redistribuir, se debe solicitar permiso al propietario y por ello pagar alguna cantidad.

Software comercial.

El software comercial es el software desarrollado por una empresa con el objetivo de lucrar con su utilización. Nótese que "comercial" y "propietario" no son lo mismo. La mayor parte del software comercial es propietario, pero existe software libre que es comercial, y existe software no-libre que no es comercial. Ambos casos son perfectamente válidos y tienen sus propias ventajas y desventajas.

Aunque es muy fácil caer en la confusión entre gratuidad y libertad, en realidad existen programas libres que son pago, y en contrapartida existen programas gratuitos que no pueden ni ser modificados, distribuidos y mucho menos existe siquiera la posibilidad de ver su código fuente.

¿Para qué conocer las implicaciones de las licencias?

Es necesario el analizar con cuidado muchas otras distintas formas de licencia y que se han mencionado debido a que no es el fin de esta tesis; pero sí, es en extremo importante saber aunque sea un poco acerca de algunas de ella. La mejor opción siempre será aquella que contenga las 4 libertades del software libre y que además procure que nuestra obra continué libre y siga siendo siempre así.

Este tema es crucial para que se pueda continuar con una filosofía de libertad; para lo cual es necesario estar bien informados acerca de todo lo que ocurre a nuestro alrededor, con el fin de ser críticos y autocríticos; hecho que servirá para apoyar y dar mayor difusión a esta filosofía y sus causas .

Al tratarse de un tema complejo y muy basto, se propone involucrarse un poco más haciendose uso de la extensa literatura que acerca de esta índole se encuentra esparcida por toda la red.

No se puede dar el lujo de estar desinformado en esta era de la “comunicación” acerca de estos rubros; Hay tantas cosas que se deberían de saber y tantas cosas que sólo sirven para hacernos más egoístas en nuestro estilo de vida como sociedad.

En forma análoga a lo que dijo Jesús “y conoceréis la verdad y la verdad os hará libres”(Juan 8:32), en el mundo del software libre la verdadera libertad se basa en conocer y entender tanto el código, como el movimiento y la filosofía que ha sido creada por este, ya que sin estas bases simplemente desaparecería todo lo ganado por los hacker a través de los años.

Casos conocidos de mala utilización de software comercial:

Con el siguiente relato se espera poner de manifiesto algunas de las causas que hacen necesario el cambio hacia un software libre.

Imaginemos por un momento que nos encontramos enfrente de nuestro ordenador personal, trabajando tranquilamente y queremos ver las notas tomadas en nuestro programa recién instalado, mismas que nos servirían para hacer contacto con un cliente muy importante. Al encenderlo buscamos nuestra aplicación recientemente instalada y nos encontramos con la gran sorpresa de que ya no existe, y junto con ella nuestros datos, nuestros contactos y la posibilidad de obtener un buen negocio...

A nuestra mente pueden llegar varias posibilidades de semejante perdida de datos según sea el caso:

Algún niño curioso se puso a jugar con nuestro equipo; pero un momento éste se encuentra en nuestra oficina y el acceso esta restringido a menores de edad.

A lo mejor nos equivocamos y accidentalmente desinstalamos nuestra valiosa aplicación, pero si hubiera sido así nuestro sistema operativo nos hubiera informado antes de tomar esa terrible decisión a sabiendas que perderíamos nuestro trabajo y nuestra información.

Tal vez sea !un virus¡ pero no puede ser, tenemos nuestra máquina con un buen antivirus que hemos comprado y lo hemos actualizado en forma constante.

Quizá fue una falla en el suministro eléctrico, pero no puede ser; nuestro equipo tiene una fuente de energía de emergencia.

Volvemos a nuestras ideas y decimos fue culpa de un “hacker”....

Todo lo anterior suena lógico dentro de nuestra forma de razonar, pero lo cierto es que esta orden se llevo a cabo desde las oficinas centrales a las cuales hemos favorecido comprando nuestro equipo y nuestro software. Este tipo de implementaciones tecnológicas se llaman “switch killer”. ¿Pero hasta donde es permitido el que puedan cometer este tipo de actos en contra de nuestra intimidad o la de cualesquier otro individuo?

Puede que en el fondo tener una herramienta de este tipo sea algo muy útil a la hora de que nuestro equipo fuera sustraído en forma ilegal por otra persona. Pero esta decisión también debería de ser tomada por el susodicho comprador y no por decisión de alguna empresa en particular.

Por otro lado, aunque la piratería hace grandes estragos monetarios en la industria del software, paradojicamente esta provoca que los grandes consorcios se vean beneficiados debido a que todo “mundo” hace uso de sus productos y no hay otro tipo de software o implementaciones que vengan a sustituir a éstas, ya que se constituyen en un estándar a seguir; El verdadero problema con ello reside en que mucho del software pirata viene con código malicioso. Otra posibilidad reside en que las políticas antipiratería se vean cada vez más reforzadas, pero ante la gran cantidad de personas, empresas, instituciones, etc. que hacen uso de estos productos ilegales, es prácticamente imposible frenar este tipo de practicas comerciales y es así como cada vez se vuelve mayor el problema.

Peor aún para el usuario final el software del cual no se cuenta el código fuente disponible puede ser modificado sin nuestro consentimiento, ya sea por la empresa que nos lo vendió o por una persona que pueda tener acceso al mismo y como resultado, no podemos darnos cuenta de los peligros potenciales que este encierra y mucho menos, defendernos de estos ataques.

Y al final de todo, que podemos hacer en contra de una compañía cualquiera; si en nuestro caso hemos hecho una firma digital, con la cual hemos aceptado ciertas condiciones de uso y sus implicaciones legales como la de ceder ciertos derechos a ésta; y aún suponiendo que nos hallamos tomado realmente el tiempo suficiente para haber leído con la debida atención y paciencia cada una de las clausulas que ésta encierra como para razonar acerca de las implicaciones que tiene y sus consecuencias. Y aún más suponiendo que después de meditarlo fríamente y sopesar todo lo anterior ¿qué es lo que podemos hacer? pues si no estamos de acuerdo con todos los puntos del contrato digital; y no damos clic en el botón de aceptar simplemente, no podemos hacer uso de “nuestro” programa.

Desgraciadamente muchas empresas serias y algunas piratas han tomado parte de los códigos libres y los han utilizado en la realización de sus propias implementaciones a las cuales les han añadido en algunos casos puertas traseras, programas encargados de enviar información de lo que escribe quien lo ha puesto en su computadora, etc. De esta forma se pueden dar casos en los que al utilizar un sistema operativo dentro de teléfonos o móviles este no sea 100% libre y seguro.

Es altamente recomendable el utilizar programas que provengan de repositorios seguros creados por la misma comunidad y de los cuales se tenga además su código fuente disponible.

Otro problema es cuando se toma como base un kernel o núcleo libre y se escribe sobre este un sistema de tipo privado, es difícil de predecir que pasará con los términos legales que amparan a la parte privada, y de los puntos que se tendrán que aceptar y firmar digitalmente por parte del usuario final.

Es preferible escoger un sistema operativo que sea basado en las cuatro premisas del software libre con la finalidad de no tener que atenerse a alguna de las sorpresas anteriormente citadas.

Virus, casos y cosas

¿Qué pasa con los virus en el mundo GNU/Linux? ¿Existen o son sólo un mito?

La respuesta por “desgracia” es un simple sí; también existen ciertas “amenazas”, pero estas tienen un fin bien distinto, pues sirven para probar la seguridad del sistema y dar así oportunidad a encontrar fallos del mismo. Estas “amenazas” en caso de ser creadas con otros fines, al ser detectadas por la comunidad debido a algún fallo o problema causado por este, encontrara la solución por lo regular en cuestión de días, no tratándose de un simple parche si no más bien de una serie de cambios y mejoras para evitar en un futuro otro ataque similar. Además por su forma de desarrollo en GNU/Linux, los virus no pueden llegar a ser tan peligrosos como en otros entornos operativos, pues se tienen diversas capas de seguridad propias del sistema. Por otro lado algunos pueden alegar que debido al hecho de que no se trata de un sistema operativo común, éste no es tan utilizado; lo cual hace que este no sea ocupado en cosas serias y por ende nadie lo quiera atacar.

Debido a todo lo anterior muchos de los detractores de este tipo de software alegan que “nadie” realmente importante lo utiliza, opinan además que este es un hecho más que suficiente para que los crackers no le presten atención alguna; Aunque la realidad es que en muchos países alrededor del mundo y sobre todo en la Comunidad Europea, se lo está implementando como una solución real, tanto para fines académicos, científicos, financieros y de gobierno, logrando mejorar sus servicios, y seguridad nacional, reduciendo además en miles de millones de euros sus gastos en compra de software privativo.

Cabe notar el hecho de que el laboratorio Fermi tiene su propia distribución y la utiliza para manejar un cañón de hadrones de miles de millones de euros, la NASA lo utiliza en proyectos de tecnología avanzada para el desarrollo de viajes espaciales, etc.

Seguramente la creación de virus u otro monstruo informático con fines académicos y de mejoramiento es un hecho de suma importancia, para encontrar más deficiencias y lograr hacer más eficiente este sistema operativo (siempre que sea usado en forma ética por supuesto).

Se debe tener muy en cuenta que hasta ahora ningún sistema operativo en la tierra esta libre de errores o debilidades, pero lo cierto es que, el uso de un código abierto admite miles de posibilidades a la implementación de soluciones por parte de cualquier miembro de la comunidad y su difusión a nivel mundial.

Otro punto importante es que este entorno operativo está diseñado para pedir muy poco al hardware en el que está implementado, lo cual lo hace prácticamente ideal para evitar el retraso tecnológico de algunos países, e incluso les abre las puertas a la posibilidad de utilizar sus propios recursos tanto para la mejoría como la producción del mismo. Ejemplos claros de esto son China, India, Francia, Cuba, Venezuela, Argentina, etc.

Al referirse a “virus” en este trabajo se lo hace en forma general, ya que existen exploits, troyanos, y diversa variedad que se puede encontrar dentro del mundo informático. Este tema es muy importante si se quiere adentrar al tema de seguridad en sistemas operativos

Cabe aclarar que existen excelentes artículos en los que se dan suficientes razones para decir que nunca podrán existir virus en GNU/Linux además de las ya citadas antes. Tomando en cuenta que GNU/Linux se encuentra en constante evolución y mejoramiento.

Clasificación de virus y otros peligros

Existen muchas clasificaciones para tratar de explicar y de darle coherencia a este tipo de software que en la mayoría de los casos crean grandes daños a instituciones, proyectos, estudiantes, etc. Los virus, troyanos, gusanos y demás fauna existente y que seguirá inventándose a lo largo de la historia de las computadoras es sin duda tema de controversias, discusiones, estudio y análisis detallados que van desde conspiraciones a temas tan triviales como charlas de café.

En esta tesis se expone la teoría de que nada más acertado que las analogías con el caballo de Troya, gusanos, exploits, etc. para darse cuenta de la peligrosidad de estos monstruos tecnológicos y del alcance que pueden llegar a tener dentro de un sistema operativo que permita por su mismo diseño el que estos puedan tomar control del mismo ya sea cambiando, borrando, sustrayendo información, etc. A menos que exista un programa extra que este detectando este tipo de invasiones al trabajo creado por un usuario cualquiera, suponiendo que este programa vigilante se encuentre siempre actualizado. En GNU/Linux no ocurre lo mismo, debido a que como se decía en renglones anteriores se eliminan estas posibilidades al quitar las opciones o fallos de seguridad en el sistema, por otro lado para que pueda ejecutarse un programa dentro de GNU/Linux es necesario que el superusuario tome control de la maquina y de la orden directa de ejecución; al utilizarse una sesión como usuario un ataque solo podría tomar control

Los virus actúan y se clasifican principalmente por la zona que atacan:

a) virus de sector de arranque, o área cero los cuales toman el control del equipo desde el arranque del mismo.

b) los virus de archivo o ejecutables que simplemente insertan su código sobre los archivos ejecutables con extensiones .com, .exe, .ovl, .dll, etc. que se están ejecutando dentro de la sesión en que se encuentren activos.

c) virus de macro, que se ejecutan dentro del lenguaje de macros tipo basic incorporado dentro de algunas suites de tipo ofimáticas y que debido a esto se vuelven de tipo multiplataforma.

d) virus de hardware que acaban destruyéndolo, alterándolo; por ejemplo alterando la velocidad de los ventiladores del procesador logrando que este se queme, haciendo que las cabezas lectoras del disco duro vibren hasta romperlas, etc.

A últimas fechas con la aparición de virus fantasmas; es decir, que no pueden ser detectados por los antivirus, virus de microcódigo; es decir, y por la falta de información acerca de la existencia de los mismos, y la eminente existencia de puertas traseras, etc. se puede llegar a crear una bomba de tiempo dentro de ciertos sistemas operativos que no implementen normas de seguridad más altas. Por lo tanto es necesario usar un sistema operativo con una mayor norma de seguridad tal como lo es GNU/Linux para evitar todos estos problemas.

Existen también “virus benignos” que una vez terminada la labor de encontrar fallos en el sistema se autodestruyen, pero esto depende de la ética de quien los desarrolla.

Por la forma de ataque:

Caballos de Troya estos aparentan ser aplicaciones, juegos, actualizaciones, etc. Siendo el operador del sistema quien ingenuamente les permite en forma literal ser instalados, con el fin de infectar la computadora, tomando el control de la misma, robando información, etc.

Bombas de tiempo este tipo de amenaza se encuentra latente hasta que llegada una fecha, una cierta cantidad de información reunida, un porcentaje de capacidad del disco duro, este procede a destruir ya sea la información o el hardware.

Gusanos al igual que en una fruta estos se reproducen dentro de la misma infectando y destruyéndolo todo a su paso.

Mutantes estos se encuentran cambiando su propio código logrando ocultarse y engañando a los antivirus.

Macrovirus estos logran reproducirse al abrirse algún archivo infectado por medio de macroinstrucciones.

De correo electrónico o de internet los cuales se propagan por este medio, por el simple hecho de abrir algún mensaje o archivo infectado.

Exploit* palabra inglesa que significa explotar o aprovechar y nada mejor para describir el proceso por el cual se va escalando privilegios explotando un fallo u error (denominados bug), dentro de un sistema operativo, con el fin de adueñarse de este o la de atacar a un tercer sistema operativo.

* http://es.wikipedia.org/wiki/Exploit

Notas importantes acerca del sistema GNU/Linux

Cuando recién se comienza a utilizar una distribución de tipo GNU/Linux, se puede notar rápidamente de que se tiene ante uno, un sistema altamente diseñado para la seguridad de los documentos, desde la misma entrada a el sistema que pedirá según sea el caso el nombre del usuario y una contraseña para poder ingresar; esta contraseña por obvias razones nunca deberá de ser una palabra sencilla del diccionario, deberá de tener mayúsculas y minúsculas, además de números y símbolos, con el fin de evitar al máximo posibles invasiones.

La seguridad del sistema es un tema de muchísimo estudio en el diseño de las nuevas distribuciones GNU/Linux, esto para el usuario más experimentado es una gran ventaja, pero para el novato puede ser un gran problema, ya que a veces se quiere experimentar con otras distribuciones GNU/Linux, y este se encuentra con la gran sorpresa de que todas las carpetas creadas en su distribución antigua, no pueden ser ya abiertas debido a que se toman como una invasión a la privacidad de sus propios archivos; esto lamentablemente puede dar lugar a la posible pérdida de los documentos. Una forma segura de migrar las carpetas y documentos es la de comprimir los mismos y grabarlos ya sea en un disco o una unidad USB, para después poderlos traspasar esta información a la nueva distribución, debido a que no siempre una distribución puede satisfacer todas las necesidades especificas del usuario final, o bien por querer seguir experimentando, para saber como se comporta un equipo con otro sabor de GNU/Linux.

Se pueden hacer uso de programas de encriptación dentro de GNU/Linux, para darle mayor seguridad a los proyectos, pero siempre se tienen que tener copias de respaldo. Todo sistema operativo es susceptible de fallar, y también puede darse el caso de que los equipos queden dañados por alguna falla eléctrica, golpe, caída, café, agua, robo, desastre natural, etc.

Es muy importante el reconocer que GNU/Linux se trata de un sistema en constante evolución y que por otra parte hay también otros núcleos que se encuentran en experimentación; en teoría si se diera el caso de que este núcleo se volviera privativo, la comunidad se cambiaría por completo a otro que si cumpla con las cuatro libertades del software libre.

Siempre se debe cuidar que la distribución sea lo más estable posible, no es muy aconsejable usar el último grito de la moda en programas, debido a que estos sirven para el estudio y verificación por parte de los desarrolladores del mismo; el que sea estable asegurará que no se tendrán amargos sabores de boca por un fallo o pérdida de valiosa información.

Se debe de tomar la precaución de no utilizar cualquier programa que acabe de salir como novedad, a menos que se este muy seguro de que éste se encuentra respaldado por la comunidad, o la Free Software Foundation.

Siempre se debe de pensar en este sistema como un forma amena de aprender algo nuevo cada día; y porque no, en algún momento como la posibilidad de hacer aportaciones valiosas a los demás.

Uso básico de la terminal en GNU/Linux

Muchas de las veces no se utilizan los programas de software libre debido a que no se tiene la suficiente información de como utilizarlos debidamente; por esta razón se empezará por dar las bases para que sean de utilidad para los futuros ingenieros químicos que pretendan tener un conocimiento más profundo y que además no se sientan apabullados por no saber en dónde comenzar.

Una de las principales herramientas de GNU/Linux y corazón mismo de este sistema es la terminal, ésta se encuentra incluida en todas y cada una de las distintas distribuciones con ligeras variantes.

Por lo general la terminal la podemos encontrar en el menú accesorios > terminal.

La terminal de GNU/Linux diferencía las letras mayúsculas de las minúsculas en la sintaxis, por lo tanto debe de asegurarse de poner correctamente los nombres de los archivos o directorios que se van a crear o a borrar.

Es de suma importancia al nombrar las carpetas con nombres continuos para poder cambiar fácilmente de carpetas.

nombre_del_archivo

<clear> sirve para limpiar la pantalla de la terminal.

<ls> muestra los archivos que se encuentran en el directorio actual.

<cd> cambio de directorio. Cuando no existan nombres continuos se ponen entre comillas, ejemplo:

cd “programación básica”

<mkdir> sirve para crear un directorio nuevo.

<mv> permite mover o renombrar archivos o directorios.

<rm> elimina archivos y directorios.

<touch> permite crear archivos nuevos.

Con la ayuda de estos comandos se puede comenzar a usar las diversas opciones que se tienen para programar y debido a que GNU/Linux presenta poderosas herramientas gráficas que pueden facilitar aún más el trabajo. Para lograr una mayor comprensión del manejo de la terminal existe una amplia variedad de libros que pueden dar más información acerca de otros comandos, y de esta forma poder aprender aún más del uso de la terminal en GNU/Linux. En todas las distribuciones GNU/Linux se pueden encontrar al menos los anteriores comandos básicos.

Cuando se comience a crear un primer programa se retomará este mismo tema con ayuda de ejemplos claros.

Algoritmos

Este es el momento indicado para incluir un nuevo concepto, el de algoritmo debido a que las máquinas no poseen inteligencia, se debe de partir de la idea de que se ha de “enseñarles” a través de una serie de pasos como lograr una función o trabajo en especifico.

Un algoritmo es pues una serie de pasos u operaciones ha realizar para lograr un proceso específico que realizará un trabajo deseado.

Cuando se es capaz de plasmar el problema en una serie de pasos sencillos, se está justo en el momento de hacer uso de un diagrama de flujo para representar el algoritmo.

Con ayuda de estas herramientas (algoritmo y diagrama de flujo), se puede construir un pseudocódigo que sirve como base para ser traducido a cualquier lenguaje de programación, tan sólo con seguir sus propias reglas de codificación.

Un ejemplo de algoritmo es el que detalla los pasos que han de seguirse para resolver una ecuación, pero también lo es una receta de cocina, un paso de baile, etc.

Es un excelente ejercicio el desarrollar habilidad para crear algoritmos; lograr descomponer en una serie de pasos sencillos diversas tareas cotidianas o problemas matemáticos, pues de esta forma se van haciendo a los verdaderos programadores y se tornara más sencillo desarrollar el trabajo.

Diagramas de flujo

Cuando recién se comienza con el trabajo de programar, se percibe que esta tarea no es tan fácil (claro se trata de aprender a utilizar otro lenguaje). Por lo cual una forma de llegar a hacer programas que además de funcionales sean fáciles de revisar para cualquier otro; es la de utilizar diagramas de flujo. Previamente se requiere haber aprendido a hacer algoritmos, aunque después de un corto periodo de tiempo, ambos conceptos quedaran interrelacionados entre sí.

Para los seres humanos no siempre es tan fácil usar ideas abstractas, siempre es mejor tener algo con que poder relacionar las cosas con las cuales se aprende tales como imágenes, olores,etc. Es por eso que los diagramas de flujo ayudaran a tener una visión mejor y más clara de los algoritmos, además de que estos facilitan tremendamente nuestra traducción a lenguajes de programación.

Cabe aclarar que todos los seres humanos tienen una curva de aprendizaje particular y nadie nace sabiéndolo todo. Se debe de procurar tener siempre en mente que todos los programas son susceptibles de mejorarse o bien corregirse y que cada día que se programe se podrá mejorar en el trabajo. No se puede exigir perfección al programar, aun los mejores programas que existen se van corrigiendo con el paso del tiempo; sin embargo se debe de procurar realizar bien el trabajo con el fin de que éste sea de utilidad. Se debe de recordar que uno de los atributos de los grandes hackers es el de ser humildes; tanto para enseñar como para aprender.

Para aprender las reglas de este nuevo lenguaje, se empieza por aprender a usar algo equivalente al abecedario en la creación de diagramas de flujo. Los siguientes símbolos que utilizarán satisfacen las recomendaciones de la International Organization for Standardization (norma ISO 5807:1985) y el American Standards Institute (ANSI).

Símbolos usados para diagramas de flujo

Marca el principio o fin del programa.

Se utiliza para indicar el momento en el que deben de proporcionarse los valores de las variables; es decir, el de lectura de datos.

Marca una operación o acción realizada en el programa. En alguna literatura se conoce como asignación, pues a una variable se le debe de dar un nuevo valor

Significa una toma de decisión.

Muestra una salida de datos o información.

Este símbolo puede ser utilizado para decisiones múltiples.

Observe con cuidado los anteriores símbolos con el fin de aprenderlos y usarlos en la tarea de programar, aunque existen otro tipo de notaciones se ha optado por la aceptada ANSI e ISO, debido a que es la más extendida, además de ser fácil de entender y que además da la opción de hacer el pseudocódigo de una forma más simple.

Todos estos símbolos han sido creados con ayuda del programa Dia, que cumple con la licencia GNU de la Free Software Foundation, la cual es una excelente herramienta para hacer diagramas de flujo de una forma profesional, bien si desea hacerlos públicos a través de la red, imprimirlos o proyectarlos para la explicación de los mismos ante los alumnos, amigos u otros programadores.

Una excelente idea es la de tener siempre a la mano una plantilla con este tipo de simbología (la cual se puede comprar en algunas papelerías especializadas), o en su caso hacer una con ayuda de una regla de cartón y por último siempre cabe la posibilidad de hacer los dibujos a mano.

Una observación más se debe de recordar que para poder llegar a programar de forma limpia, ordenada y profesional es necesario practicar e intercambiar nuestros códigos en forma libre, procurando hacer uso de las cuatro premisas del software libre para mejorar nuestros programas y contribuir con el desarrollo de los demás y el nuestro.

Se procurará hacer lo más fácil posible con ayuda de los diagramas de flujo la enseñanza del pseudocódigo para la creación de los programas, no pretendiendo tampoco poner en cada uno de los ejemplos de esta tesis su diagrama de flujo.

Los siguientes ejemplos pueden servir como base general para la creación de programas cada vez más complejos. Se debe por otro lado el evitar pensar que un programa entre mas complejo sera mejor; las cosas más increíbles hechas en este mundo han sido creadas a partir de muchas muy simples. Un programa por muy simple que sea, es importante siempre y cuando sea funcional.

Algunos pequeños consejos antes de comenzar

Las siguientes estructuras son las más utilizadas en los distintos lenguajes de programación, observándolas y estudiándolas cuidadosamente, no será nada difícil aprender a programar y/o adecuarlas a otros casos, según sean las necesidades. Para entenderlos simplemente se deben de seguir las flechas y pensar paso a paso.

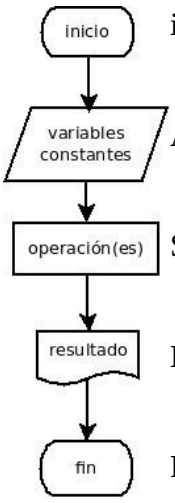
En los siguientes ejemplos invita a realizar las propias anotaciones en los espacios en blanco que se encuentran al lado de cada una de las siguientes figuras con el fin de analizarlas con cuidado y obtener así una mejor compresión de las mismas.

No se puede tomar éste trabajo como aburrido para lo cual se requiere que se propongan estructuras propias; pues solo con paciencia y mucha practica, se logrará conseguir aprender a programar correctamente.

Estructuras de programación

Lineal simple

Esta estructura sirve cuando queremos crear un programa sencillo que podamos utilizar “pocas” veces, como podemos observar en la siguiente figura se da en una serie de pasos sencillos.



inicio de programa.

Aquí se declaran las variables o constantes del programa.

Sirve para declarar la función que se usen en el programa*.

Mostrará el resultado del programa.

Fin de programa.

* Y asignar a una variable el valor que le corresponda, señalando además el valor de la variable que corresponde al resultado del algoritmo.

Aclaración de suma importancia:

Es muy necesario el notar desde aquí dos errores muy comunes al programar entre los novatos, el primero olvidar hacer los diagramas de flujo y el segundo dejar de pensar paso a paso. Este último es muy frecuente debido a que se desea hacer lo mas pronto un programa y no perder mucho tiempo en él. Todo programa es perfectible y por lo regular es necesario hacer varias correcciones antes de dar por hecho que éste funcionara correctamente.

Estructuras selectivas

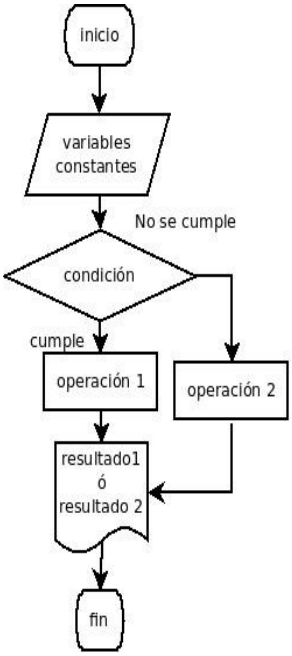
if (if then)

Esta sencilla estructura da la posibilidad de hacer un programa que dado el caso de cumplirse una condición determinada, procede a realizar la operación que se le indique y en caso contrario simplemente terminara el proceso o bien tomará otra alternativa.

Además de que esta es muy conveniente para repetir un proceso sencillo "n" veces, hasta que se le indique cuando terminar al computador; para lo cual se habrá definido el valor de término anteriormente.

if else (if then else)

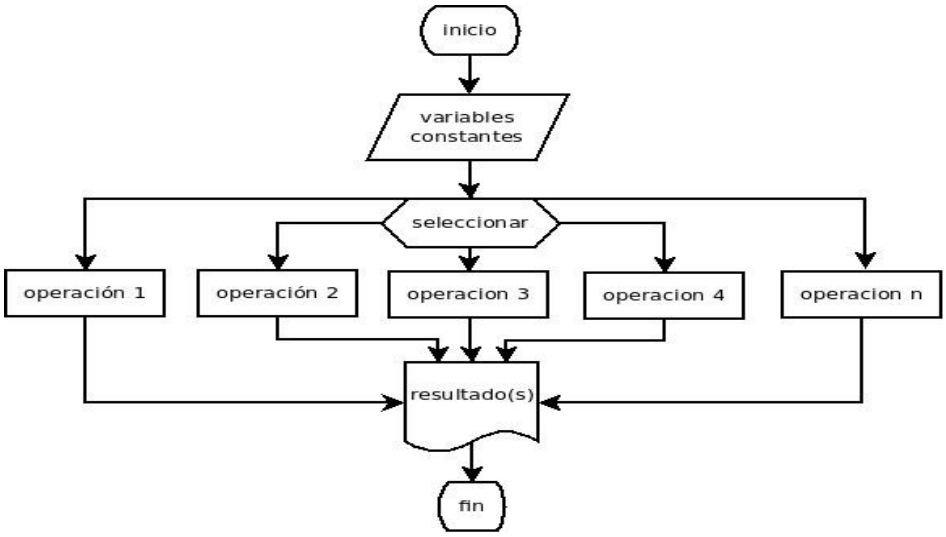
Es un caso muy similar al anterior sólo que la diferencia reside en que si no se cumple la condición para realizar la operación principal entonces se realizara otra por defecto.



estructura selectiva múltiple

switch (case)

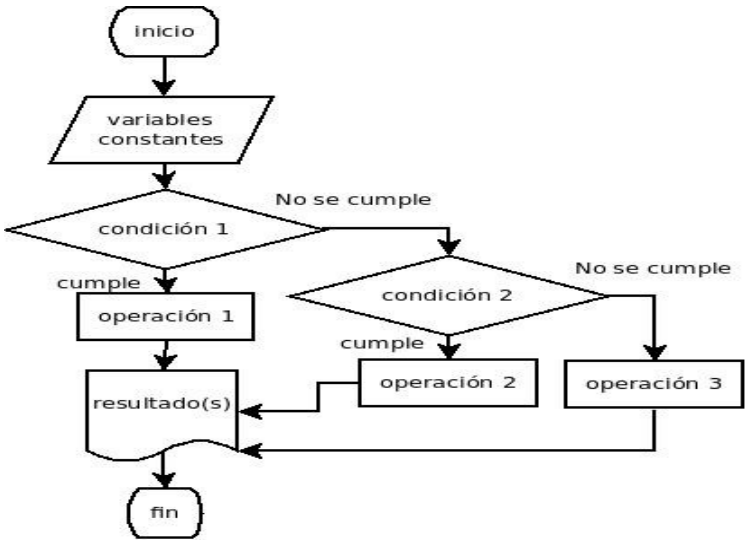
Cuando tenemos más de dos posibilidades para realizar las actividades de un algoritmo frente a una condición dada, se puede descomponer la condición en varias condiciones de dos alternativas, pero también se puede utilizar una estructura como la siguiente:



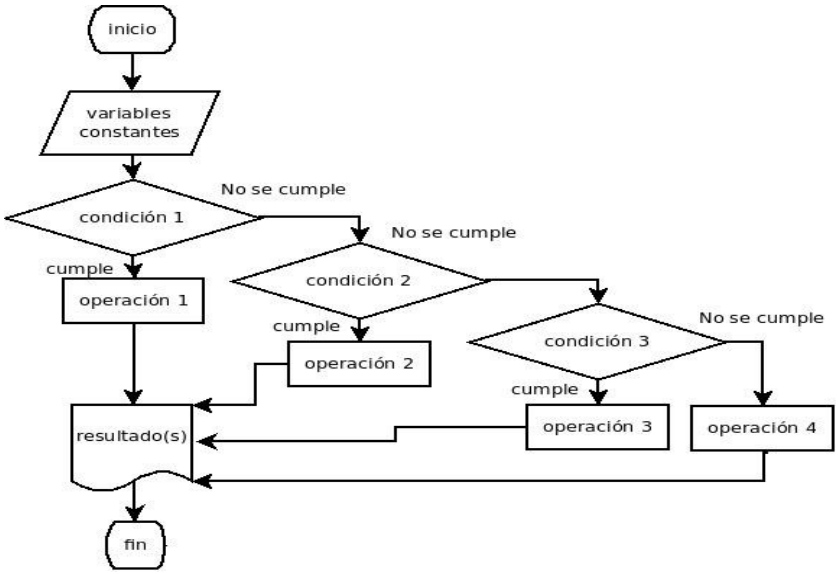
Estructuras anidadas

Se les denomina a las que nacen de varias bifurcaciones de estructuras de selección simple como las siguientes

ejemplo 1



ejemplo 2



Según sea el caso se puede hacerlas tan complejas como se requiera, siempre y cuando se tenga el cuidado de seguir un orden lógico en la selección de condiciones, para no crear errores graves durante la ejecución del programa.

Estructura repetitivas

Para poder comprender claramente de que se tratan las estructuras de repetición que ofrecen los lenguajes de programación, primero se deberá de imaginar por ejemplo que se necesita realizar una sencilla suma un par de veces, se considera que no hay problema en hacer estas operaciones; y que tal vez se hará relativamente fácil dependiendo de las cantidades que se deben adicionar. Pero que tal si se necesita realizar estos mismos cálculos para miles o millones de datos, seguramente sería algo tortuoso el pensarlo siquiera y humanamente algo imposible de hacer. (!inhumano trabajo¡).

La iteración según el diccionario enciclopédico estudiantil Océano significa simplemente repetir y en este caso se puede definir como el proceso en el cual se repite una operación u orden en un ciclo definido por el propio usuario. A estos ciclos que se presentan en el algoritmo se les conoce también como ciclos de calculo, iteraciones o en los textos de autores españoles como bucles.

Por ejemplo: Para realizar un programa que ordene a la computadora a contar hasta cuatro y luego detenerse, se tendrá que pensar en descomponerlo en una serie de pasos para poderlo entender.

Para facilitar la asimilación del problema, se utilizara una analogía con una caja y canicas

1er paso imaginar una caja vacía; es decir, se tienen 0 canicas en su interior.

2do introducir una canica; es decir, se tiene 1 canica.

3er paso introducir otra canica; es decir, se tienen 2 canicas

4to paso introducir otra canica; es decir, se tienen 3 canicas

.
.
.

Y así se seguirán estos pasos hasta el infinito mientras no se disponga de una orden que le de final a este proceso. Para lograr detenerlo se necesita de una sentencia de control; es decir, algo que sirva como forma de comparación y sentencia de paro.

Para poder entenderlo correctamente vuelva a ver el problema anterior pero con la condición de que al llegar a cuatro canicas se detenga el proceso.

1er paso imaginar una caja vacía; es decir, se tienen 0 canicas en su interior y menos de cuatro por lo tanto prosigue.

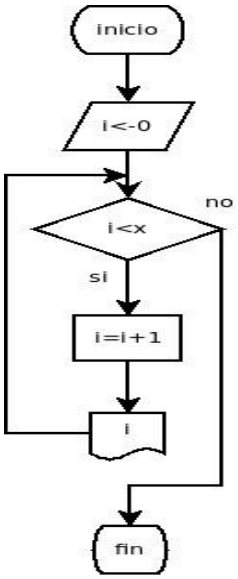
2do introducir una canica; es decir, se tiene 1 canica y menos de cuatro por lo tanto prosigue.

3er paso introducir otra canica; es decir, se tienen 2 canicas y menos de cuatro por lo tanto prosigue.

4to paso introducir una canica; es decir, se tienen 3 canicas y menos de cuatro por lo tanto prosigue.

5to paso introducir una canica; es decir, se tienen 4 canicas y es igual a cuatro por lo tanto se detiene el proceso.

El siguiente diagrama de flujo es una representación del anterior proceso.



Este diagrama de flujo puede ser interpretado de la manera siguiente, siempre teniendo en mente el anterior ejemplo de la caja y las canicas.

Primero se le asigna el valor de 0 a i, lo cual queda representado por medio de $0 \leftarrow i$; en los lenguajes de programación por lo regular queda como i=0. Pero en el caso de las variables se estará diciendo que se le asigne este valor inicial, no es que ya lo tenga implícito durante todo el programa. Recuerde que hay una gran diferencia entre variables y constantes, esto deberá de estar muy presente si se quiere aprender a programar sin muchas dificultades.

Luego se define una sentencia que detenga el proceso de iteración (ciclo), en donde x toma un valor mayor al que i tenga en ese momento.

En el siguiente paso se puede ver la siguiente operación i=i+1; la que indica que se le esta adicionando un 1 al valor que i tenga en ese momento.

El próximo paso sería mostrar el valor obtenido en esa operación, para después volver a la sentencia de control hasta cumplirla y entonces detener el programa.

Paso a paso con ayuda de números se vería de la siguiente manera, tomando para x=4 como sentencia de control:

Otra forma de representar este problema se podrá observar en la siguiente tabla:

Resultado	***Acción***
i ← 0	primero se asigna valor igual a cero
i ← i+1	se carga en memoria la operación a realizar
i=0+1	se asigna el valor a la variable
1	se obtiene un resultado
1<4	se compara con la sentencia de control
i=1+1	este nuevo valor se toma
2	se obtiene un resultado
2<4	se compara con la sentencia de control
i=2+1	este nuevo valor se toma
3	se obtiene un resultado
3<4	se compara con la sentencia de control
i=3+1	este nuevo valor se toma
4	se obtiene un resultado
4<4	se compara con la sentencia de control
i=4+1	este nuevo valor se toma
5	se obtiene un resultado
5<4	Se detiene el proceso no cumple la sentencia de control

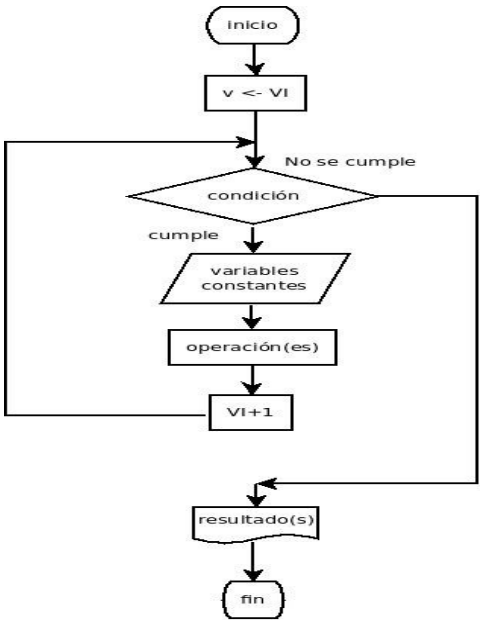
Para poder entender todo el proceso; se deben de leer las celdas a la par del anterior diagrama de flujo.

Este mismo ejercicio se puede implementar con cualesquier otro diagrama que se tenga a la mano.

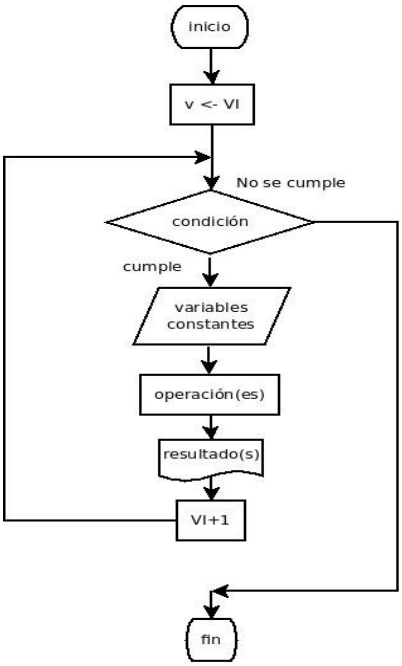
for

El ciclo se repetirá hasta cumplir con la condición de pasos previamente impuesta por el usuario y justo en ese momento procederá a salir del ciclo.

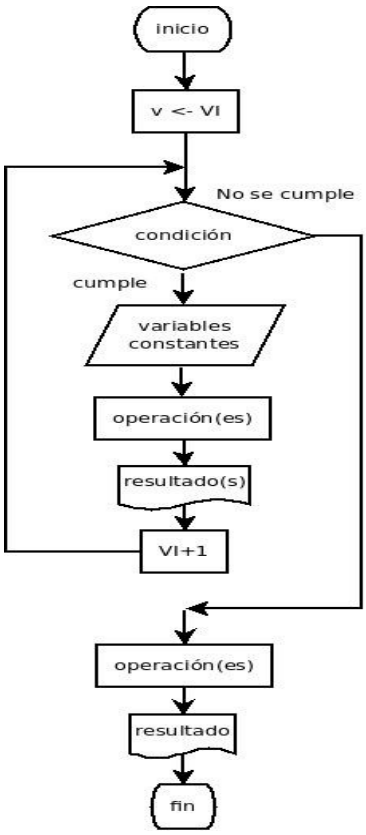
En forma análoga es como si se le pidiera a alguien que caminara 5 pasos y entonces levantara una bandera.



En el siguiente caso mostrara cada uno de los resultados y luego de cumplirse con la condición establecida el programa se detendrá.



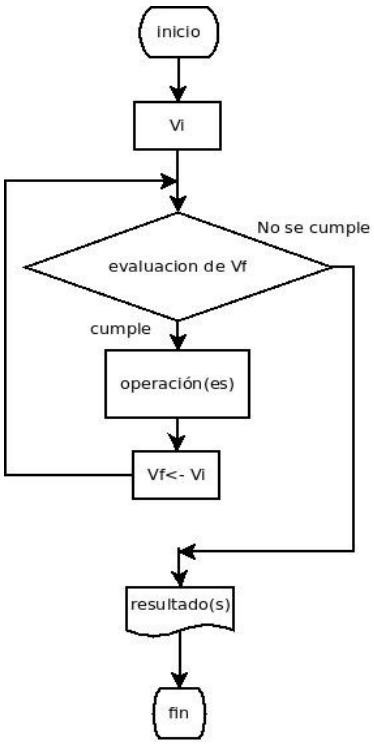
En el siguiente ejemplo podremos notar como se van imprimiendo los resultados de cada iteración y tras cumplirse la condición necesaria se procede a realizar una operación y mostrar el resultado obtenido con el valor final del ciclo anterior.



While

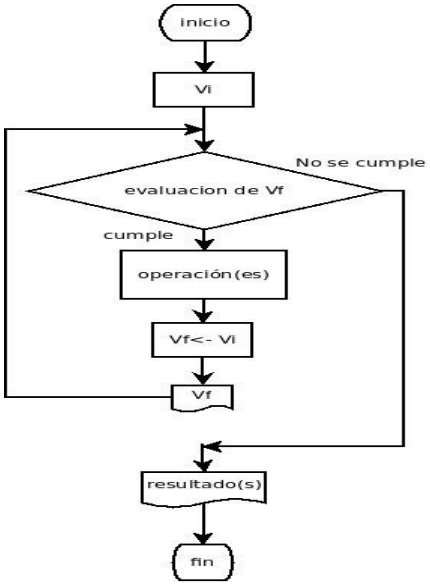
Es muy parecida a la anterior estructura sólo que en esta no definimos la cantidad de veces que ha de repetirse una operación o proceso, pero a cambio si definimos una condición de término de ciclo.

Es como si le pidiésemos a alguien que caminara hasta encontrar una piedra y en ese momento levantara una bandera; se debe notar que en esta proposición no se sabe de antemano la cantidad de pasos que se necesitan para lograr la condición antes impuesta.



En este diagrama de flujo se ha definido con Vi el valor inicial y con Vf el valor final, como se puede ver se tiene un bloque de decisión en el cual se compara el valor final que ha sido generado a partir del inicial. Este se repetirá sin necesidad de un contador, y lo hará así hasta que la sentencia se cumpla.

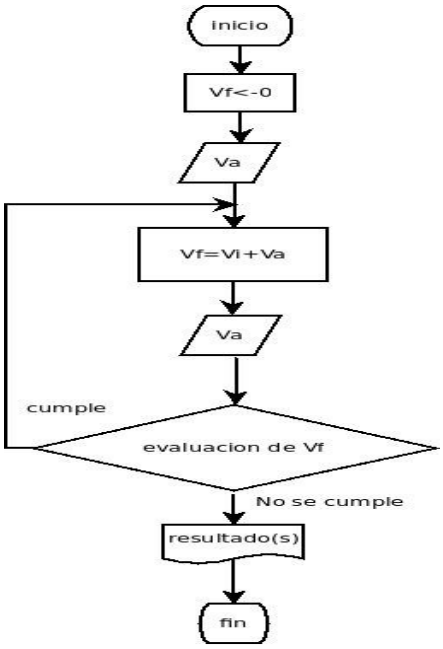
Como en los ejemplos anteriores se puede modificar este diagrama de flujo para que de información cada vez que se cumpla con una iteración, como por ejemplo si quiere hacer un diagrama con los resultados obtenidos



si se presta atención esta estructura
puede servir para usarla como si se tuviera un contador, con la ventaja de que podemos utilizar números decimales para indicarle cada paso iterativo. En el capitulo dedicado a “bc”se muestra un ejemplo.

do-while (repeat)

Esta estructura asegura que el programa correrá al menos una vez antes de detenerse en caso de que no se cumpla con la condición preestablecida.



Se puede crear un pseudocódigo con ayuda del último diagrama de flujo, que muestra la serie de pasos para contar hasta 4. (véase el problema ejemplo de la página 29)
En este ejemplo se supone que el valor de Va siempre es igual a 1.

Resultado		***Acción***
Vf ← 0		primero se asigna valor igual a cero a Vf
Va ← ?	1	Se pide el valor de Va
Vf = Vf + Va	1 = 0 + 1	se obtiene el valor de la variable Vf
Va ← ?	1	Se pide otro valor de Va
Vf < 4	1 < 4	se compara con la sentencia de control
Vf = Vf + Va	2 = 1 +1	Se suma el valor anterior de Va al nuevo ve Vf
Va ← ?	1	Se pide otro valor de Va
Vf < 4	2 < 4	se compara con la sentencia de control
Vf = Vf + Va	3 = 2 + 1	Se suma el valor anterior de Va al nuevo ve Vf
Va ← ?	1	Se pide otro valor de Va
Vf < 4	3 < 4	se compara con la sentencia de control
Vf = Vf + Va	4 = 3 + 1	Se suma el valor anterior de Va al nuevo ve Vf
Va ← ?	1	Se pide otro valor de Va
Vf < 4	4 < 4	se compara con la sentencia de control
Vf = Vf + Va	5 = 4 +1	Se suma el valor anterior de Va al nuevo ve Vf
Va ← ?	1	Se pide otro valor de Va
Vf < 4	5 < 4	Se detiene el proceso; no se cumple la sentencia de control

Ahora se puede ver con claridad el pseudocódigo:

asigna el valor de cero a	Vf ← 0
hacer (do)	
pide el valor de Va	Va
Suma Va + Vf	Va + Vf
asigna este valor a	Vf = Vf + Va
compara hasta que (while)	Vf < 4

Como siempre y a lo largo de esta tesis se seguirá insistiendo en que de ser posible se mantenga la sana costumbre de poner la mayor cantidad de comentarios a todos los programas que se desarrollen, con la finalidad de hacerlos más útiles tanto para revisiones posteriores como para que los demás aprendan de ellos. De hecho es ésta la manera en como se hicieron los ejemplos a lo largo de esta tesis, con la esperanza de que sean base para la creación de nuevos programas, además de ayudar a dar los primeros pasos en el mundo de la programación.

Errores

En computo existen los siguientes tipos de errores que tienen causas directas con las limitaciones de la arquitectura de las maquinas y se conocen como:

Overflow (desbordamiento por capacidad máxima): Este se da debido a que los cálculos en la computadora dan un número tan grande que no puede ser manejado por la capacidad de ésta.

Underflow (desbordamiento por capacidad mínima): Al igual que el anterior error éste se debe a que la computadora de como resultado un numero tan pequeño que no puede ser manejado por la capacidad de cómputo de la misma.

División entre un número muy pequeño puede hacer que nuestra maquina obtenga un overflow. Ahora bien si se hace entre un numero muy grande y uno muy pequeño, el error puede darse por causa de redondeo.

División entre cero

Otras causas indirectas de errores son:

Errores de salida: estos se dan por no escoger bien el formato con que se presentaran los resultados que se han obtenido; es decir, la cantidad de decimales que se han de presentar en pantalla, lo cual dará errores de truncamiento o redondeo.

Error de discretización: éste se debe a que no se puede convertir exactamente un número decimal en un número discreto y esto conlleva un pequeño error siempre.

Errores debido a un mal diseño del microprocesador o el coprocesador matemático, en algunos de los primeros microprocesadores INTEL, AMD y CYRIX se dio este problema, aunque cada vez son menos estos problemas puede darse el caso en pequeños PIC, además de algunos microcontroladores o procesadores tipo ARM que no han sido diseñados con el fin de tener un alto rendimiento en operaciones de punto flotante.

En todo proceso humano existirán siempre la presencia de errores y en los cálculos realizados por medio de computadoras se les puede clasificar también por los siguientes tipos tal como se muestra en el cuadro siguiente:

Tipo	***Explicación o cálculo***
Absoluto	Es la diferencia entre el valor verdadero y una aproximación a éste.
Relativo	Es el cociente del error absoluto entre el valor verdadero.
Por ciento	Es el error relativo multiplicado por 100
Inherente	Éste se da por equivocación o por la naturaleza aproximada de la representación.
Truncamiento	Son los errores debidos a los términos decimales omitidos.
Redondeo	Se debe a la cantidad máxima de dígitos que se le puede asignar a una variable.

El concepto de valor verdadero es muy relativo bien sea porque que se conoce realmente o se utilice una buena aproximación. Pero se debe de respetar este valor para los cálculos.

De hecho los errores inherentes se encuentran implícitos en los datos con lo que efectúa algún cálculo la computadora. Se debe de recordar que los errores al utilizar medios computacionales básicamente se dan por dos cuestiones bien particulares que son el método utilizado y por otro lado el debido a la capacidad de la o procesador. Cuando se va a hacer uso de un método en particular esta decisión depende del modelo matemático y para que se utilizará.

Siempre que se desarrolle un nuevo programa se debe de asegurar de que este corre perfectamente bien; para lo cual se necesita hacer antes unos cuantos cálculos manuales, además nos servirá para darse cuenta perfectamente si la precisión que se necesita es la suficiente y hacer los arreglos pertinentes.

Algunas precauciones para no cometer errores

Al desarrollar los programas se debe evitar lo siguiente para minimizar al máximo errores en el diseño de los mismos:

El sumar números de muy distintas magnitudes entre sí. Por ejemplo

3 500 000+0.000 143 = 3 500 000. 000 143 nos arrojaría un valor de $3.5*10^6$, con lo cual perderíamos unas cuantas cifras significativas.

El restarse dos números casi iguales en forma análoga al ejemplo anterior este daría algo igualmente ilógico.

En la pantalla de la computadora los números muy grandes o pequeños serán representados en muchas ocasiones por medio de una letra “E”, ésta letra representa a potencias de base 10; al usarse la letra “E” se evita el tener que representarla por medio de varios caracteres, de tal suerte que se ocupa menos memoria por parte de la computadora. El manejo de ésta notación se muestra en los siguientes ejemplos:

3.5E6 = $3.5*10^6$, 4.3E4 = $4.3*10^4$

Se puede notar claramente en los ejemplos anteriores que con esta notación solo se usa un carácter con la letra “E”, y no tres como lo sería con *10. Puesto que cada carácter ocupa más memoria dentro del computador, es muy importante no ocupar muchos de estos, puesto que lo volverían más lento en el manejo de los cálculos. Este tema fue muy importante en los años en que las memorias RAM eran extremadamente caras y de muy poca capacidad

Tipos de variables

En todos los lenguajes de programación se hace uso de varios tipos de variables por lo cual es necesario el asignarles las propiedades inherentes a cada una de ellas. Esto tiene como finalidad la de asignarle los suficientes recursos sin la necesidad de que se desperdicien estos. Es muy importante debido a que antiguamente las computadoras tenían recursos muy limitados y en la actualidad debido a que se pueden programar pequeños circuitos programables, éste es un tema muy importante; siempre hay que pensar bien antes de asignar el tipo al cual pertenecerán.

Tipo	Datos que se almacenan en memoria
Enteros	Estas solo pueden almacenar números de tipo entero.
Reales	Estas pueden almacenar números decimales. (punto flotante)
Imaginarios	Tienen tanto una parte real como una imaginaria.
Caracteres	Esta almacena un carácter.
Cadena de caracteres	Sirve para almacenar una secuencia de caracteres
Lógico	Solo pueden almacenar valores de falso o verdadero o cero y uno

Además de esto la precisión para las variables de tipo real se puede dividir en dos tipos:

Precisión sencilla: ésta es la que por defecto hace uso el programa y se debe de usar si los cálculos no requieren de mucha precisión.

Doble precisión: se utiliza cuando se necesita para hacer cálculos más finos; es decir cuando se quiere minimizar el error o bien los cálculos así lo requieren. El único problema es que este método requiere de más uso de memoria por parte de un programa.

Aparte de lo anterior, es necesario tener mucho cuidado de no revolver los diversos tipos de variables para no crear errores innecesarios; por ejemplo:

Una variable que sirva como contador tiene que ser de tipo entero.

Para una división entre un numero decimal y un entero, deberán usarse en ambas variables las de tipo real, para evitar errores; y aún en caso de que el programa corra, éste dará un resultado que puede no ser correcto llegando a perder mucho tiempo valioso de diseño antes de poder dar cuenta de éste error.

Nota importante:

A partir de aquí en adelante, empezará la programación en diferentes tipos de lenguajes; algunos puntos necesarios para su correcto uso se trataran más adelante en los apéndices correspondientes.

Los siguientes ejemplos tienen como finalidad apoyarlo en sus inicios en la programación y no de substituir por completo el estudio y práctica necesarias para que usted logre desarrollar aplicaciones aún más complejas. Se espera que divertirse y mejorar cada día aunque sea un poco en todo lo que se ama, se vuelvan una de las metas principales.

Gfortran (g77, f77)

Comenzaremos por hacer un programa en Gfortran con ayuda de GNU/Linux, debido a que es un lenguaje muy sencillo para aprender correctamente las bases de la programación en el entorno GNU/Linux. Es muy importante analizar los pasos siguientes pues estos servirán para los demás lenguajes de programación.

Primero: se verá la construcción de un programa con ayuda de cualquier editor de textos incluido en la distribución de GNU/Linux, el cual puede ser OpenOffice, gedit, vim, vi, emacs, etc.

Segundo: se guarda el documento con la extensión que se necesita. Por ejemplo *.f para programas de tipo Fortran 77.

A continuación la sintaxis para comenzar a programar en Fortran

```
c primer.f
c
c  los comentarios comienzan con una letra c
c  Este programa sirve para enseñar las bases de programación
c  en Fortran
c  escrito por: Juárez Contreras Alejandro
c   sábado 17 de julio de 2010
c
c es conveniente poner una c y luego numerar hasta el 7
c para escribir correctamente usando la norma de Fortran 77
c234567----------------------------------------
          print*,
          print*, ' hola, éste es mi primer programa en software libre'
          print*, 'programar es fácil y divertido'
c Note como se dejan espacios en este lenguaje para mayor claridad
c al momento de hacerlos funcionar, utilice su imaginación para
c hacerlos mas amigables a quien los utilice.
          print*,
          print*, ' *****     *****'
          print*, '   ----         ----  '
          print*, '         .......         '
c la palabra stop permite detener lo que aparece en pantalla para poderlo ver y
c la palabra end da por terminado el programa.
          stop
          end
```

tercero: Se crea una carpeta en donde se guardaran los programas.

Cuarto: se hará uso de la terminal de la siguiente manera:

Al hacer uso del comando “ls” se verán tanto las carpetas y archivos del directorio en el que se encuentre.

```
usuario@máquina:~$ ls
Documentos          Imagenes
Descargas           Música
Escritorio
```

Con el comando “cd” se podrá cambiar el directorio de trabajo, recordando que es necesario poner el nombre de forma correcta al respetar tanto las mayúsculas como las minúsculas del mismo.

```
usuario@máquina:~$ cd Documentos
```

Se hace uso nuevamente del comando “ls”.

```
usuario@máquina:~/Documentos$ ls
programas_fortran        programas_python
Tutoriales
```

Otra vez se usa el comando cd

```
usuario@máquina:~/Documentos$ cd programas_fortran
usuario@máquina:~/Documentos/programas_fortran$  ls
primer.f
```

Una vez localizado el archivo *.f se procede a crear un ejecutable con la siguiente sintaxis.

```
usuario@máquina:~/Documentos/programas_fortran$gfortran primer.f -o primer.exe
```

Se procede a correr el programa; es decir, se ejecuta.

```
usuario@máquina:~/Documentos/programas_fortran$./primer.exe

hola éste es mi primer programa en software libre
programar es fácil y divertido

  *****          *****
    ----                    ----
          ......................
```

Con estos pasos se puede programar en otros lenguajes, bajo plataformas GNU/Linux, UNIX, Solaris, Open Solaris e incluso Mac.

Nota: también se puede hacer uso de la siguiente sintaxis para compilar el programa.

```
usuario@máquina:~/Documentos/programas_fortran$gfortran -c primer.f
```

En GNU/Linux los programas ejecutables no tienen extensión .exe y además es necesario el dar ciertos permisos para que estos puedan ser ejecutados.

Además en GNU/Linux existen muchas formas de hacer lo mismo y de diferentes formas, por ejemplo también se puede simplemente haber escrito en la terminal:

usuario@máquina:~/Documentos/programas_fortran$gfortran primer.f -o primer
usuario@máquina:~/Documentos/programas_fortran$./primer

Como compilar

Se habla mucho acerca de compilar en el mundo GNU/Linux, pero básicamente ¿qué es esto? y ¿para qué sirve?

Según el Diccionario Enciclopédico Estudiantil Océano se trata de reunir en una sola obra extractos de otros libros o documentos.

Aplicado éste concepto a la máquina, el compilar asegura que cada una de las partes del programa se encuentren lo mejor configuradas, logrando hacer su trabajo en forma óptima según sean las características de nuestro equipo y procesador. En GNU/Linux se pueden compilar todos nuestros programas e incluso el núcleo utilizado.

Se puede aprender a hacerlo con un sencillo programa escrito en Gfortran. Para esto se puede hacer uso del comando make de la siguiente forma:

usuario@máquina:~/Documentos/programas_fortran$make gfortran primer.f -o primer

para ejecutarlo simplemente escribimos:

usuario@máquina:~/Documentos/programas_fortran$run ./primer

obteniendo:

hola éste es mi primer programa en software libre
programar es fácil y divertido

```
*****          *****
 ----                    ----
       ......................
```

Después de esto para limpiar lo hecho en la máquina simplemente se escribe:

usuario@máquina:~/Documentos/programas_fortran$make clean

Unas notas acerca de Gfortran

Siempre se debe poner los comentarios suficientes para que otro programador entienda nuestro trabajo.

La primera columna sirve para denotar si se trata de un comentario con ayuda de una “c” como ya se observo desde un principio.

La columna marcada con el número 6 en Gfortran sirve para concatenar cadenas muy largas de caracteres, basta poner cualquier carácter en la misma. Existe una característica heredada de las antiguas máquinas lectoras de tarjetas perforadas que sólo admitían un número máximo de caracteres (a saber 72 por línea). En cualquier otra columna esto arrojará un error. Las otras 5 tienen otras utilidades, pero esto será más que suficiente por el momento.

Estructuras de selección

Una vez que ya se conoce el entorno de la terminal de GNU/Linux para compilar y ejecutar los programas, se puede empezar por hacer uso de las sentencias selectivas existentes para Gfortran. Se revisa cuidadosamente el diagrama de flujo correspondiente a esta estructura selectiva y se hacen las anotaciones correspondientes dentro del mismo y del código.

if

```
c if1.f
c Éste programa sirve para aprender a usar la instrucción if
c escrito por: Juárez Contreras Alejandro
c234567-----------------------------
      print*, 'Éste programa muestra como utilizar if then'
      print*, 'Introduzca el valor de a'
      read * , a
      print*, 'Introduzca el valor de b'
      read * , b

c Evaluación de la variable de control
      print*, ' si el valor de la resta de a y b es negativo se '
      print*, 'multiplica por -1'
      c = a - b
      print*, ' el valor de la resta es: ', c

c Esta es la condición de evaluación
      if(c.le.0.0)  c = -1.0 * c

c Esto es con el fin de obtener la raíz cuadrada del mismo.
c si se cumple entonces se procede a realizar los siguientes cálculos

       d = sqrt(c)
      print *, 'Los valores obtenidos son resta y para la raíz:'
      print*, c,d
      stop
        end
```

En el siguiente ejemplo podemos ver claramente como la sentencia “if then” es muy semejante a la anterior “if simple”. Pero esta tiene ciertos puntos importantes que no se pueden obviar pues de lo contrario simplemente el programa nunca podrá ser compilado correctamente.

if then

```
c if2.f
c Éste programa enseña el uso de la sentencia if then
c escrito por: Juárez Contreras Alejandro
c234567--------------------------------
      print *, 'Introduzca el valor de  a'
      read *, a
      print *, ' Introduzca el valor de b'
      read *, b
c realiza el siguiente cálculo
c234567
      print*, ' Restamos a-b'
      c = a - b

      if(c.lt.0.0) then

c Si se cumple la condición anterior entonces
c234567
      print*, 'El resultado es:', c
      print *, 'al ser negativo da una'
      print *, ' raíz compleja'

c finalizamos con endif  esta parte del bloque y damos por finalizado el programa
       endif

c En caso de que no se cumpla la condición necesaria
c se procede a realizar las siguientes operaciones.
       d = sqrt(c)

       print *, 'Como el valor fue positivo'
       print * ,'El valor de la resta es:', c
       print *, 'y el de la raíz es:', d

c Para finalizar correctamente el programa necesitaremos agregar las siguientes líneas
       stop
       end
```

Éste ejemplo se puede visualizar de una forma más sencilla, si se imagina por un momento que se tiene enfrente un camino que se bifurca en otros dos para llegar a un poblado; se puede elegir entre una y otra opción pero no las dos al mismo tiempo, al fin y al cabo se tendrá que finalizar el recorrido en algún momento.

if then else

```
c if3.f
c instrucción if
c escrito por: Juárez Contreras Alejandro
c234567----------------------------
          print*, ' Introduzca el valor de a '
          read *, a
          print*, ' Introduzca el valor de b '
          read *, b

c realizada la siguiente operación
          print*, ' Hacemos la resta de a-b '
          c = a - b
          print*, ' El resultado es:', c

         if (c.gt.0.0) then

c si se cumple la condición solicitada, proceda a realizar lo siguiente
         print*, ' El resultado es positivo '
         d = sqrt(c)

          print * , ' Por lo tanto la raíz de este número es:', d

c En caso de no ser así realice lo siguiente

            else

            print*, ' El resultado es negativo', c
            print*, ' Para obtener una raíz lo multiplicamos'
            print*, ' por -1'

           d = sqrt(-1.0 * c)

            print * , ' de lo contrario obtendríamos un '
            print *, ' numero complejo '
            print *, ' siendo ya positivo éste número la raíz es ', d
              endif

             stop
              end
```

Bucles en programación Gfortran

Para hacer bucles en Gfortran se puede utilizar la sentencia DO

```
c  Éste programa esta diseñado para enseñar
c  a hacer bucles con do en Gfortran
c
c  escrito por  Juárez Contreras  Alejandro  2010
c
c
c234567---------------------------------------------

          print *, 'introduzca el valor inicial a sumar'

c  la sentencia read le asigna un valor a la variable  a utilizar.
          read*, h
c se define la cantidad de veces que se desea repetir la operación

          do i=1, 10

c define cual es la función que se desea repetir
           h=h+1
c una vez terminado el proceso, el ciclo termina

           enddo
c si no  se pueden observar los valores obtenidos, el trabajo seria inútil, obsérvese pues como
c se pone el valor de la variable al final del aviso.

           print*, 'esta es una serie', h
           stop
           end
```

Si al programa anterior le hacemos un pequeño cambio se puede ver con claridad cada uno de los números que se obtienen en cada una de las operaciones, lo cual hace ver más claramente el potencial de esta orden.

```
c Éste programa esta diseñado para enseñar
c a hacer bucles con do en Gfortran
c
c escrito por Juárez C. Alejandro  2010
c
c
c234567-----------------------------------------------
      print *, 'introduzca el primer valor a sumar'
      read*, h
      do i=1, 10
       h=h+1
       print*,' sumando 1 a 1',  h
       enddo
       print*, 'esta es una serie', h
       stop
       end
```

Se tiene pues el resultado final de la suma o bien cada uno de los pasos que se necesitaron para llegar hasta ella.

Otra forma de arreglar el programa de tal forma que pida el número de veces a repetir la función sería de la siguiente manera:

```
c
c
c Escrito por Juárez Contreras Alejandro
c
c234567-----------------------------------------------
      print*, ' número de veces a repetir la operación:'
      read*, n
      print*, 'introduzca el valor inicial'
      read*, r

      do i= 2, n
      r = r+(1.0/i)
      enddo
      print*, ' términos:', n,
     ~ 'serie: ' , r
c Observe como el símbolo ~ sirve para concatenar la línea; aunque puede ser cualquier otro
c símbolo o letra, debido a que se encuentra en la columna 6
      stop
      end
```

Pruebe con otros símbolos en la columna 6, a ver que sucede.

Ahora es el tiempo de empezar a utilizar matrices.

```
c Éste programa arroja los valores de la suma
c de pares de datos introducidos.
c escrito por Juárez Contreras Alejandro
c
c234567----------
c Esta matriz puede contener hasta 10 * 10
c el número de columnas queda definido por la cantidad
c de contenedores propuestos, en éste caso x, y, z

      dimension x(10), y(10), z(10)

c  especifica el número de renglones de la matriz
      print*, 'especifique el número de renglones de la matriz'
      read*, n
c una vez especificado el número de renglones, se pide que trabaje hasta cierto número
c de iteraciones
      do i=1, n

c introduce los números que sean específicos de la matriz
      print*, ' introduzca los valores x, y, z '
      read*, x(i), y(i), z(i)
      print*, x(i), y(i), z(i)
      enddo
c una vez terminado de  introducir los números correspondientes
c se procede a sumarlos

c se pone el valor de sx = 0, sy=0, sz=0 para evitar que se vicie con algún número
c anterior en la memoria

      sx = 0.0
      sy = 0.0
      sz = 0.0
      do i=1,n
      sx = sx + x(i)
      sy = sy + y(i)
      sz = sz + z(i)
c se procede a obtener en forma secuencial la suma de los números introducidos

       enddo
c Al escribir print fuera del bucle do éste dará solo los valores finales
c copie la línea siguiente antes de enddo y observe que pasa
       print*, 'los valores de la suma de valores x, y, z es:', sx , sy, sz

       stop
       end

c Juegue con el anterior programa hasta entenderlo, borre y cambie variables.
```

Funciones y subprogramas

Estas son de las herramientas más útiles y necesarias en la programación; motivo por el cual siempre hace falta una buena explicación para poder hacer un uso correcto de la mismas. La verdadera ventaja de utilizar funciones reside en que se puede ir creando bibliotecas propias del programador que a su vez pueden ser aprovechadas una y otra vez según convengan.

```
C suma2.f
C Función suma 2 números
C234567----------------------------------------

          print *, '  Éste programa suma dos valores por medio de una función'
           rint *, '  introduzca el valor de a  '
          read *, a
          print *, '  introduzca el valor de  b '
          read *,  b
c Aquí se invocando a la función suma y se sale del programa
c principal momentáneamente
          c = suma(a, b)
c Aparecen en pantalla los valores obtenidos gracias a la función.
          print * , a, b, c
c Aquí se vuelve a invocar a la función suma
          d = suma (a, c)
c Aparecen en pantalla los valores obtenidos nuevamente gracias a la función.
          print  *, a, c, d

          stop
          end

c234567
c A partir de aquí se a define la función
c que se llamara función suma:
c  suma = r1 + r2
c  con function se define el nombre de la función en éste caso suma
c  y con  (r1,r2)  se definirán las variables de la nueva función
c las cuales son r1 y r2
c234567
          function suma(r1,r2)

c A continuación se define la operación que se necesita
          suma = r1 + r2

          return
c con return  se regresamos al programa principal.
          end
```

Como se puede ver en el ejemplo anterior las variables de la función son independientes de las del programa principal y éstas toman momentáneamente los valores de las que tiene el programa principal.

Otra forma de atacar un problema es subdividiéndolo en diversas partes, con el fin de hacerlo más sencillo, para lo cual se hace uso de subprogramas; a continuación se muestra como lograr esto a través de un programa muy sencillo. Éste puede ser fácilmente modificado para ser usado en cualquier otra situación.

Un consejo importante es que lo repita con cuidado en su computador y realice su propio diagrama de flujo, teniendo mucha calma hasta entenderlo perfectamente.

```
c suma1.f
c Éste programa sirve para entender la función subroutine
c escrito por Juárez Contreras Alejandro
c234567-----------------------------

      read *, a, b

      call suma(a, b, c)
c Aquí se esta invocando al subprograma suma y se sale del programa
c principal momentáneamente

      print * , a, b, c
c Se obtienen los valores que arroja el subprograma
      stop
      end
c 234567-----------------------------
c El subprograma suma es:  r3 = r1 + r2
c 234567-----------------------------

c  con subroutine se define el nombre del subprograma en éste caso suma
c  y con  (r1, r2, r3)  se definen las variables del mismo
c las cuales son r1, r2 y r3

       subroutine suma(r1,r2,r3)

      r3 = r1 + r2

      return
      end
```

Es muy importante recordar que el programar de una forma limpia y ordenada dentro del software libre ayudara a mejorar e implementar soluciones a la medida para no depender del software con licencia. Y que de esta manera siempre se dispondrá siempre del código fuente para mejorarlo, compartirlo, estudiarlo y nuevamente poder ser utilizado en otros programas ahorrando tiempo, dinero y esfuerzo.

Gpc

Vamos a comenzar por escribir nuestro primer programa. Aunque al usar Gpc no es indispensable el uso de las líneas de programación “uses crt;” ni “ClrScr;” estas sirven para lograr una mejor integración con el lenguaje Pascal de tipo comercial; sin embargo, es necesario tomar muy en cuenta estas líneas en caso de querer incluirlas para posteriormente hacer un uso correcto dentro de una IDE de tipo comercial. En GNU Pascal Compiler los programas utilizaran la terminación (.pas).

```
(* Escrito por Juárez Contreras Alejandro *)
(* éste programa muestra las bases para programar en Pascal *)
(* note la forma de hacer comentarios en Pascal *)

program primer_programa;
uses crt;
begin

ClrScr;   (*note que éste comando usa letras mayúsculas y minúsculas y deben de respetarse*)
 (* Si bien es opcional esta línea sirve para trabajar con una pantalla limpia *)

writeln( 'Éste es mi primer programa en lenguaje gpc')

end.
```

El anterior programa también funcionara correctamente para el compilador Gpc si se escribe de la siguiente manera:

```
(* Escrito por Juárez Contreras Alejandro *)
(* éste programa muestra las bases para programar en Pascal *)
(* note la forma de hacer comentarios en Pascal *)

program primer_programa;

begin

writeln( 'Éste es mi primer programa en lenguaje gpc')

end
```

De aquí en adelante se seguirá haciendo uso de estas líneas con el fin de acostumbrarse a esta manera de programar, pero teniendo muy en cuenta todas las precauciones antes observadas.

La forma más sencilla de volver interactivo un programa, con objeto de poder dar a la maquina información (el valor de una variable por lo regular), se logra con la palabra reservada read( ); tal cual se muestra en el programa siguiente:

```
(* Escrito por Juárez Contreras Alejandro *)
(* éste programa enseña a introducir datos en Pascal*)

program primer_programa;    (* La  instrucción program es una declaración del nombre del programa*)


var a ,b ,c : real;                 (*Esta instrucción declara que se utilizaron las variables a,b, y c*)
                                    (*y que además estas serán del tipo real *)


begin                               (*Esta instrucción declara donde comienza el programa*)


writeln('escriba los números a y b para sumar');        (*Esta instrucción hace que la máquina *)
                                                        (*envié el mensaje:*)
                                                        (* “escriba los números a y b para sumar” *)


read(a,b);                          (* se pueden definir tantas variables como se quiera; *)
                                    (*solo se deben separar con una coma (,)*)
                                    (*Esta instrucción hace que posterior a su ejecución la maquina *)
                                    (*espere que se le suministren los valores de las variables a y b*)


writeln( 'ya sabemos como interactuar con el programa');    (*Esta instrucción hace que la*)
                                                            (* máquina envié el mensaje:*)
                                                    (* “ya sabe como interactuar con el programa” *)


c:=a+b;                             (* esta instrucción hace que  c tome el valor de a+b*)


writeln('el resultado es:',c:10:3);         (* Esta declaración hace que el resultado se de*)
                                            (* hasta con 10 números significativos y 3 decimales *)


readln;             (*Esta declaración hace que el programa espere a que se presione*)
                    (*cualquier tecla para terminar el programa con la declaración end.*)
end.
```

Nota importante: Los comentarios deberán de ponerse al lado del programa en forma preferentemente para facilitar la lectura del mismo y ayudar a comprender bien cada una de las líneas que lo componen, siempre que esto sea posible.

Por cierto que en Pascal se carece de una orden directa para hacer directamente las potencias de números y para ello se pueden utilizar a los logaritmos como se puede ver en el siguiente ejemplo

Nota: se debe de recordar que 2^4 es igual a exp(4*ln2)

```
(* Escrito por Juárez Contreras Alejandro *)
program potencia;
var x, y, z : real;

begin

writeln(' Éste programa eleva un numero x  a una potencia y');

writeln;                          (*   Esta declaración deja un espacio en blanco para mejorar *)
                                  (* la lectura dentro de la terminal *)

writeln(' escriba un número' x);

read(x);
writeln('escriba el número y');

read(y);

z:=y*ln(x);
z:=exp(z);

writeln('el resultado es:', z:10:3);
readln;
end.
```

Nota importante: Durante la compilación de los programas dentro de Gpc muchos de los errores mostrados en la pantalla de la terminal se deben por lo regular a la falta de algún (;) dentro de la sintaxis del programa.

Sentencias

if-then-else

El lenguaje Gpc es muy rico en la variedad en que podemos hacer uso de esta instrucción. Lo que la hace muy útil.

```
(* Escrito por Juárez Contreras Alejandro *)
program potencia;
uses crt;
var x, y, z : real;

begin

ClrScr;

writeln(' Éste programa sirve para sumar dos números, el primero deberá ser positivo');
writeln(' en caso de ser negativo el primero de ellos ');
writeln(' imprime en pantalla adiós ');
writeln(' Introduzca un número');

read(x);
writeln('sumado a');

read(y);

if x>0 then                    (* Si se cumple la condición entonces realizamos lo siguiente *)


begin

z:=x+y;
writeln('el resultado es:', z:10:3);
readln

end                            (* Da por terminada esta sección *)


else                           (* En caso de no cumplirse la condición, realizamos lo siguiente *)


begin                                                    (*Esta es la otra sección*)
writeln;
writeln('El primer número fue negativo, Adiós ');
writeln;
end (* Da por terminada esta sección *)



end. (* Da por terminado el programa*)
```

Las instrucciones begin y end; tiene aquí una función muy parecida a la de un paréntesis. La máquina toma como una sola instrucción lo que esta entre el begin y el end; pero realiza todas las ordenes.

La siguiente sintaxis muestra la gran versatilidad de esta sentencia; dado que puede servir para hacer hasta juegos de preguntas muy sencillos. Como el siguiente:

```
(* Escrito por Juárez Contreras Alejandro *)
program falsoverdadero1;
uses crt;
var lugar, x : integer;

begin
ClrScr;

writeln(' En donde nació Francisco Villa el “Centauro del Norte” ');

writeln(' Introduzca un número');

writeln('1= Chihuahua; 2= Yucatán; 3= Parral; 4= Paracho ');
read(x);

lugar:= x;

if (mes = 1)  then
writeln(' falso ');


if (mes = 2 ) then
writeln(' falso ');


if (mes = 3)  then
writeln(' verdadero, nació en Parral en el año de 1878  ');


if (mes = 4)  then
writeln(' falso ');


end. (* Da por terminado el programa*)
```

Tenemos otras posibilidades de hacer sentencias if anidadas, como a continuación se puede observar, hasta aquí es necesario que se repitan los ejemplos dentro de un compilador y asegurarse de haberlos entendido. Cada uno de los comentarios son muy importantes. Basta con borrar algunas partes del programa y sustituirlas por operaciones propias, claro que siempre respetando la sintaxis correcta. Es muy fácil observar cuales son las partes indispensables del programa.

Otra forma de hacer el anterior programa

```
(* Escrito por Juárez Contreras Alejandro *)
program falsoverdadero2;
uses crt;
var mes, x : integer;

begin
ClrScr;

writeln(' En donde nació Francisco Villa el “Centauro del Norte” ');

writeln(' Introduzca un número');

writeln('1= Chihuahua; 2= Yucatán; 3= Parral; 4= Paracho ');

writeln('La opción que usted escogió fue la número:');
read(x);

mes:= x;

if (mes = 1)  then
writeln(' falso ')

(* Observe como no existe punto y coma entre cada una de las siguientes líneas *)
(* si usted agrega el punto y coma esto arrojara errores al ser compilado *)
(* éste programa *)
else

if (mes = 2) then
writeln(' falso ')

else

if (mes = 3)  then
writeln(' verdadero, nació en Parral en el año de 1878  ')

else

if (mes = 4)  then
writeln(' falso ')

end. (* Da por terminado el programa*)
```

Case

Para lograr lo anterior sin hacer uso de muchas sentencias de tipo también se dispone de otra herramienta más sencilla dentro de la programación en Gpc que es la sentencia case

```
(* Escrito por Juárez Contreras Alejandro *)
program falsoverdadero;
uses crt;
var lugar, x : char;

begin
ClrScr;

writeln(' En donde nació Francisco Villa el “Centauro del Norte” ');

writeln(' Introduzca una letra');

writeln('a= Chihuahua; b= Yucatán; c= Parral; d= Paracho ');
read(x);

lugar:= x;

case lugar of

(* Observe como están en entrecomillados sencillos las variables a utilizar *)
(* Los dos puntos concatenan la respuesta; no se olvide de ponerlos *)

'a' :  writeln(' falso ');

'b' :  writeln(' falso ');

'c' : writeln(' verdadero, nació en Parral en el año de 1878  ');

'd' : writeln(' falso ');

end; (* sin esta línea arrojara un error al intentar compilarlo *)

end. (* Da por terminado el programa*)
```

Para usar esta instrucción se recomienda usar siempre variables de tipo char; pues las de tipo integer o real arrojan errores al momento de ser compilado el programa. De hecho las de tipo real jamás podrían ser utilizadas por esta sentencia. Aquí se debe observar nuevamente que las instrucciones begin y end funcionan como un paréntesis nuevamente.

while-do

```
(* Escrito por Juárez Contreras Alejandro *)
(***********************************)
program unoauno;

uses crt;                    (*ver la nota al final de la página*)

var num: integer;

begin

ClrScr;

(* Inicializamos la variable num con cero*)
num:=0;

while (num <= 10) do

begin

 num:= num+1;

 writeln (' resultado ', num);

end; (* Da por terminado el ciclo *)

end. (* Da por terminado el programa*)
```

Esta estructura es muy útil debido a que permite hacer uso de variables de control de tipo integer, real y char.

Nota importante: La unidad crt (catode ray tube) es un modulo independiente del programa que permite utilizar una serie de funciones especiales que aplican al monitor como CLrScr o gotoxy (columna, renglón) que sirve para ubicar el texto en la pantalla en un punto especifico con lo que se puede mejorar también el diseño de salida de los datos a través de la terminal.

repeat-until

```
(* Escrito por Juárez Contreras Alejandro *)
(**********************************)
program unoauno2;
uses crt;
var num: integer;

begin
ClrScr;

(* Inicializamos la variable num con cero *)
num:=0;

(* Observe cuidadosamente que la sentencia repeat se ubica después de begin *)

begin

 repeat

(* Aquí se definen las operaciones o funciones a realizar *)

 num:= num+1;

 writeln (' resultado ', num);

(************************************)

(* la instrucción until ayuda a definir la condición a cumplir, para seguir iterando*)

 until (num<= 10);

end;

end. (* Da por terminado el programa*)
```

Aunque es un poco más compleja que la anterior sentencia, ésta no deja de ser también muy sencilla de aplicar a los problemas debido a que al igual que la anterior, admite variables de control tipo integer, char y real.

for

```
(* Escrito por Juárez Contreras Alejandro *)
(* éste programa sirve para imprimir en pantalla *)
(* 9 veces un saludo, contando cada una de ellas *)
(*****************************************)
program fro2;
uses crt;

var contador: integer;

(* Comienza con un begin *)

begin
ClrScr;

(* Se puede notar claramente la inclusión de un begin *)
(* después de la palabra do *)

for contador:= 0 to 8 do begin

writeln ('hola amigo');
writeln(contador);

(* Con el fin hacer más claro el programa se agrega un renglón en blanco *)
writeln;  (* Esta instrucción hace que el cursor salte una línea en la pantalla del monitor *)

(* Nunca debe de faltar la palabra end; *)
end;
end.
```

En esta sentecia la variable que sirve para hacer el conteo estrictamente debe de ser entera.

En lo personal no puedo asegurar el buen funcionamiento de esta sentencia debido a que tuve varios problemas al momento de programar con ella dentro de la IDE Geany y de su propia IDE fpc; sin embargo, sería de mucha utilidad, si alguien quiere hacer algún comentario o anotaciones acerca de la misma para una posterior edición de esta tesis. Además de que arroja errores en algunas ocasiones si se le agrega la línea ClrScr, en la sentencia for, pero las otras funciones while-do y repeat-until, funcionan perfectamente bien y sin problemas.

El siguiente ejemplo que se aborda es muy interesante,ya que al estudiarlo detenidamente se puede llegar a comprender muchos de los errores que se cometen a la hora de programar con éste tipo de sentencia. O a lo menos ofrece una solución aceptable en el caso de insistir al uso de la misma.

```
(* Escrito por Juárez Contreras Alejandro *)
(* éste programa sirve para imprimir en pantalla *)
(* las tablas de multiplicar *)
(*****************************************)
program multiplica2;

var  numero1, numero2, x: integer;

begin
(* si se escribe justo aquí una linea ClrScr; arrojara un error, ¡evite éste! *)

writeln('Éste programa imprime las tablas de multiplicar ');
writeln('desde el 1 hasta la que usted decida ');
writeln('por favor introduzca un numero');
read(x);

for numero1:= 1 to x do

for numero2 := 1 to 10 do

writeln( 'multiplicación _el:',  numero1, '_ por_ '  numero1, ' _da_',  numero1 * numero2);

end.
```

También tiene la posibilidad de usar un contador hacia atrás en gpc.

```
(* Escrito por Juárez Contreras Alejandro *)
(* éste programa sirve para imprimir en pantalla *)
(* 9 veces un saludo, contando al revés cada una de ellas *)
(*****************************************)
program alrevés;
uses crt;

var contador: integer;

(* Comienza con un begin *)

begin
ClrScr;

for contador:=8 downto 0 do begin     (* Se puede notar claramente la inclusión de la palabra downto *)

writeln ('hola amigo');
writeln (contador)

end;
end.
```

funciones

```
(* Escrito por Juárez Contreras Alejandro *)
(* éste programa sirve para aprender a *)
(* utilizar funciones *)

(**********************************)
program funcion1;
uses crt;
(*Aquí definimos  las variables globales*)
var potencia, x, y: real;

(**********************************)
(* En esta sección definimos nuestra función*)
(* como podemos observar la palabra function sirve para definir *)
(* el tipo de variables que esta utilizara y además de que tipo de variable *)
(* sera el resultado que hemos de obtener *)

function poten(x, y:real) : real;

begin

poten:= exp(y*ln(x));

end;

(**********************************)

begin
ClrScr;

writeln(' Éste programa eleva un numero x  a una potencia y');
writeln(' además se hace uso de la función potencia');
writeln('escrita por el usuario');
writeln(' escriba un número');
read(x);
writeln('elevado a la potencia');
read(y);

(* Note con cuidado como se hace uso de nuestra función ya creada*)
(* imagine las posibilidades  de desarrollo para problemas mayores *)

potencia:= poten(x, y);

writeln('el resultado es:', potencia:10:3);
readln;

end.
```

Gcc

Éste lenguaje es por excelencia el que fue y ha sido el más utilizado dentro de la estructura de los GNU/Linux y de UNIX por lo que resulta interesante aprender sus fundamentos. Para guardar los programas escritos en lenguaje C se usa la extensión *.c, para compilarlos y ejecutarlos se hace uso de gcc. Los comentarios dentro de la sintaxis comienzan con /* y terminan con */ y estos además no pueden ser anidados.

Se debe de observar con cuidado el siguiente programa.

```
/* primer.c
Éste programa sirve para enseñar las bases de programación
 en lenguaje C
escrito por : Juárez Contreras Alejandro
sábado 17 de julio de 2010 */

#include <stdio.h> /* esta cabecera sirve para definir los parámetros de entrada y salida*/

int main( )          /* esta sentencia siempre deberá de estar presente dentro de un programa en C*/

{                    /* puede decirse que es la forma análoga de un begin en Pascal*/
        printf ("Programar es muy fácil y divertido"); /*muestra en pantalla éste mensaje */
        return 0;        /* si no incluimos esta línea, mandará un mensaje de error*/
}               /* esta llave es análoga al end de Pascal */
```

En el anterior ejemplo se puede observar la instrucción return 0; si no se incluye dentro de la sintaxis del programa, éste correrá normalmente, pero al momento de compilarlo arrojara un mensaje de error, mismo que puede hacer pensar que existe otro problema y propicie el invertir mucho tiempo valioso antes de conseguir encontrarlo. Esto se debe a que esta sentencia le dice al compilador que el programa a terminado con éxito y le devuelve el control al sistema operativo.

Otra forma correcta de escribir el anterior programa seria la siguiente:

```
/* primer.c
Éste programa sirve para enseñar las bases de programación
 en lenguaje C
escrito por : Juárez Contreras Alejandro
sábado 17 de julio de 2010 */
#include <stdio.h>

int main(int argc, char** argv)

{
        printf ("Programar es muy fácil y divertido");
        return 0;
}
```

Cada uno de los argumentos que se encuentran escritos dentro de los paréntesis sirven para decirle al programa con que clase de variables va a trabajar.

Debido a a que el lenguaje de programación C es muy importante cabe hacer algunas aclaraciones muy importantes acerca de su sintaxis.

La cabecera stdio.h (standar input output) sirve para cargar el modulo que permitirá como su nombre dar ordenes de salida y entrada del programa.

Para hacer uso de las funciones matemáticas incluidas dentro de éste modulo se indica otra cabecera de la siguiente manera # include <math.h>, justo debajo de la cabecera #include <stdio.h>.

No todos los compiladores de lenguaje C arrojaran un mensaje de error por carecer de la sentencia return 0, pero sin embargo es muy aconsejable el siempre incluirla dentro del programa que se éste escribiendo, con el fin de mantener la compatibilidad con las demás versiones.

La palabra reservada int indica que se utilizaran variables enteras; mientras que char indica la función para carácter.

Las sentencias argc número de argumentos de línea, es la información que se teclea después del nombre cuando se ejecute éste y argv puntero de carácter.

Según el Glosario de informática de Guido J. Pace año 1970–2003 ; El puntero es un tipo de variable, en el cual se almacena la dirección de un dato y permite manejar direcciones “apuntando” a un elemento determinado.

Un ingeniero químico necesita hacer operaciones matemáticas en forma continua, por lo cual se puede ver con el siguiente ejemplo de una suma que esto es muy fácil.

```
/* suma.c
Éste programa sirve para enseñar las bases de programación
 en lenguaje C
escrito por : Juárez Contreras Alejandro
sábado 17 de julio de 2010 */

#include <stdio.h>

int main( )

{
        printf ("la suma de 6.255+7.25 ");

        float x=6.255, z=7.25, v=0.0; /* aquí se define los valores de las variables utilizadas*/
        v=x+z;                        /* se realiza el cálculo */

        printf("\n  es:%.2f", v );
        return 0;
}
```

Para que el programa tenga mayor legibilidad se puede hacer que éste deje el espacio de un renglón mediante la instrucción \n, pero es muy importante que la introduzca entre las comillas y dentro de la instrucción printf. Observe con cuidado el siguiente ejemplo:

```
/* renglones.c  Escrito por Juárez Contreras Alejandro */
#include <stdio.h>

int main( )
{
        printf ("Como dejar renglones");

        printf("\n ");

printf("funciona así \n ");

printf("\n ");

printf("\n o también así ");

        return 0;
}
```

Éste tema es de suma importancia debido a que un programa deberá de ser siempre claro, para que éste pueda ser utilizado con la mayor facilidad y limpieza siempre. Antes de continuar con los temas siguientes, se debe experimentar con el programa anterior modificándolo hasta que se entienda perfectamente como utilizar esta importante opción.

Para lograr hacer que los programas sean interactivos se hace uso de la instrucción scanf; la cual permite que poder utilizar el programa con otras cantidades que se vayan necesitando a lo largo del trabajo.

```
/* suma2.c
Éste programa sirve para enseñar las bases de programación
 en lenguaje C
escrito por : Juárez Contreras Alejandro
sábado 17 de julio de 2010 */
#include <stdio.h>

int main( )
{
        printf ("Introduzca dos números a sumar");

        float x, z, v;                  /* Se definen de que tipo y cuales son las variables*/

            scanf("%f %f", &x ,&z);     /* El programa espera a que se le entregue una cantidad*/

v=x+z;                                  /* Se realiza la operación indicada*/

        printf("\n la suma es:%.2f",v); /* Define con cuantos decimales se dará el resultado*/

        return 0;
}
```

Sentencia if sencilla

```
/* calificación.c
Éste programa sirve para enseñar la sentencia if
escrito por : Juárez Contreras Alejandro */
#include <stdio.h>

int main( )
{
        float cal;                          /* Esta variable es de tipo punto flotante */
        printf( "ingrese la calificación ");
        printf("\n");
        scanf("%f.3", &cal);

        if (cal >= 6)                       /* esta es la variable de control */
        printf ("n aprobado");              /* cualquier valor que no cumpla con ella */
                                            /* no devolverá ningún mensaje */
        return 0;
}
```

if ... else

```
/* califica2.c
Éste programa sirve para enseñar la sentencia if
escrito por : Juárez Contreras Alejandro */
#include <stdio.h>

int main( )
{
        float cal;
        printf( "ingrese la calificación");
        scanf("%f.3", &cal);

        if (cal >= 6)                   /* esta es la variable de control */
        printf ("aprobado");

        else printf("reprobado");    /* en caso de no cumplirse da éste mensaje */
        return 0;
}
```

Las sentencias if anidadas son muy sencillas de hacer en C y por ende son de muchísima ayuda, como se puede observar en el siguiente ejemplo:

```
/* califica3.c
Éste programa sirve para enseñar la sentencia if
escrito por : Juárez Contreras Alejandro */
#include <stdio.h>

int main( )
{
        float cal;
        printf( "ingrese la calificación");
        scanf("%f.3", &cal);

        if (cal >= 8)                   /* esta es la variable de control */
        printf ("muy bien");

        else                            /* se vuelve a tomar otra decisión */

        if  (cal >=6 )                  /* esta es la nueva variable de control */

        printf("bien");                 /* en caso de cumplirse da éste mensaje */

        else

        printf("suficiente");           /* y en caso de no hacerlo da éste mensaje */

        return 0;
}
```

switch-case

Para poder hacer uso de múltiples decisiones se utiliza la sentencia switch-case reduciendo así aun más la posibilidad de incurrir en errores al tener que escribir una menor cantidad de líneas de programación.

```
/* decisiones.c
escrito por Juárez Contreras Alejandro */
#include <stdio.h>

int main( )

{
  int n;          /* para evitar que el programa se vuelva difícil de entender aun para*/
                  /* el usuario comienza por definir el tipo de variables que usará éste */
 printf ("introduzca un número");
 printf( "éste aparecerá escrito con letras" );
 printf("recuerde que éste solo funciona hasta el cuatro ")

    scanf("%i", &n);          /* aquí se lee la  variable de control */

        switch(n)             /* y es aquí en donde se toman las siguientes decisiones */
                              /* según el numero que se escoja */
         {

        case 1 : printf ("uno \n"); break;

        case 2: printf ("dos \n"); break;

        case 3 : printf ("tres \n"); break;

        case 4 : printf ("cuatro \n"); break;

   default:printf ("adiós, ya no sé más números \n"); break;      /* en caso de no cumplirse  ninguna*/

}

return 0;               /* sin esta línea mandara errores al compilarse el programa*/

}
```

Mantenga el orden y la limpieza en cada uno de sus programas; pues de esta forma logrará encontrar más fácilmente posibles errores en la sintaxis y éste será mas útil para los demás.

Aunque siempre se trata de una experiencia muy motivadora el poder lograr correr un programa tan sencillo como el anterior; siempre será necesario el escribir uno que tenga mayor poder de aplicación. Más adelante dentro de la sección dedicada a los ejemplos de aplicaciones en ingeniería química se retomará un ejemplo aún más completo.

for

```
/* cuenta.c
escrito por Juárez Contreras Alejandro */

#include <stdio.h>

int main( )

{

  int conta;
/* valor inicial; variable de control; incremento o decremento */
  for(conta=0; conta <=10; conta++)  /* note la forma peculiar en que se define  el incremento */
                                     /* de 1 en 1 como ++ de la variable conta (vea anexo) */
{
          printf("\n estoy contando %i", conta);

          }
return 0;
}
```

Note la gran potencia de esta sentencia en el ejemplo siguiente. Escriba el siguiente programa proceda a compilarlo desde una terminal ¿que nota? ¿Porqué sucede?

```
/* cuenta2.c
escrito por Juárez Contreras Alejandro */

#include <stdio.h>

int main( )

{

  int conta, sum;

  sum=5;
  for(conta=0; conta <=10; conta = conta+1)          /*ponga atención aquí*/
  {
          printf("\n estoy contando %i", conta);
          sum = sum +1;                              /* y aquí */

          printf("\n el siguiente %i", sum);
          }
return 0;
}
```

while

```
/* contar.c
muestra el uso de la sentencia while
escrito por Juárez Contreras Alejandro */

#include<stdio.h>
int main( )

{

int i;
printf("introduzca un numero");
scanf("%d",&i);

while( i<=10 )              /* nunca introduzca “;” en éste lugar, vea la nota importante */

{

printf("\ni=%d",i);

i=i+1;

}

return 0;
}
```

Como un ejercicio adicional después de haber logrado correr por primera vez correctamente el programa anterior se deberá de modificar, hasta que se éste seguro de como funciona y pueda comenzar a escribir el suyo correctamente. Respete el esqueleto del mismo para evitar al máximo posibles errores.

¡nota importante! Si se llegase a escribir “ while(i<=10); ” al compilar el programa se mandara un mensaje de que éste a logrado tener éxito, pero la triste realidad es que jamas se obtendrá ni una sola señal de haber logrado algo y lo más seguro es que se perderá mucho tiempo antes de poder encontrar éste error. Y por ende es muy posible que quien éste haciendo éste tipo de programas se decepcionaría tanto de su trabajo que probablemente no le quedarán ganas de volver a programar.

do while

Éste programa se ha hecho a propósito en base al anterior y aunque se puede decir que se trata de una copia del mismo, tiene la finalidad de que se pueda apreciar con mayor facilidad la semejanza que tienen ambas sintaxis. Esta sentencia realiza al menos una vez un cálculo y en caso de que no se cumpla la condición propuesta éste se detiene.

Para un mayor entendimiento se deberá copiar la sintaxis del mismo en un papel, después se hará una comparación entre ambos línea por línea, realizándose los respectivos diagramas de flujo

```
/* contar2.c
muestra el uso de la sentencia do-while
escrito por Juárez Contreras Alejandro */

#include<stdio.h>
int main( )

{

        int i;

        printf("introduzca un numero \n");
       scanf("%d",&i);

do                          /* se debe de poner mucha atención aquí */

{

       printf("\ni=%d",i);
       i=i+1;

}

while( i<=10 );             /* y aquí */

return 0;

}
```

bc (basic calculator)

GNU/Linux dispone también de una herramienta poco conocida, la cual es ideal en caso de no tener un entorno gráfico y una computadora con pocos recursos de procesamiento, pues apenas si consume estos (104 kB); se puede invocar desde terminal con simplemente escribir bc. Se trata afortunadamente de una aplicación que la mayoría de las distribuciones traen por defecto (si es que no todas).

Esta aplicación es ideal para correrse en terminal, aunque también se puede hacer uso de Geany. Si se dispone de un entorno gráfico para facilitar aún más la tarea.

operación	resultado
expresión + expresión	suma
expresión - expresión	resta
expresión * expresión	multiplicación
expresión / expresión	división
expresión ^ expresión	potencia
sqrt(expresión)	Raíz cuadrada
expresión % expresión	Resto de la división
++ expresión	Suma uno a la expresión
-- expresión	Resta uno a la expresión
expresión ++	Suma uno a la expresión en la siguiente sentencia
expresión --	Resta uno a la expresión en la siguiente sentencia

Nota: En bc no se puede definir la potencia de un número por medio de ** siempre tendrá que ser ^

bc posee también las siguientes funciones básicas*:

Función	Se obtiene:
s(x)	seno de x*
c(x)	coseno de x*
a(x)	arcotangente de x*
l(x)	logaritmo natural de x
e(x)	logaritmo neperiano de x
j(n,x)	Función de Bessel *j* de orden *n* de *x*

* Los datos obtenidos de las funciones trigonométricas se dan en radianes

Para definir cuantos números decimales se necesitan en el resultado se usa el comando:

scale = n en donde “n” denota la cantidad de números decimales definidos.

A pesar de ser una aplicación muy modesta se tiene la opción de crear programas propios (con extensión .b). Una gran ventaja de esta aplicación reside en que tiene una gran semejanza con la sintaxis de C; realmente es una lástima que apenas exista documentación o referencias acerca de la misma. (Se puede contribuir con la creación de alguna documentación).

Como ya es costumbre se comenzará por crear un primer programa para aprender a hacer uso de su sintaxis:

```
/* Éste programa sirve para enseñar a utilizar bc de forma sencilla
como podemos observar es importante el tener cuidado con nuestra sintaxis
para no crear errores.
México  UNAM 2010
Juárez Contreras Alejandro*/

print "\n"                                  /*salto de línea*/
print "programar con bc es muy fácil"  /*imprime el mensaje*/
print "\n"
print "\n"
print " la suma de 4+2 es:"
4+2                                        /*imprime el resultado de la suma*/
print "\n"
quit                                       /*finaliza el programa*/
```

Para poder hacer uso de los programas creados para bc simplemente se escribe en la terminal que proporciona el entorno de Geany:

usuario@maquina:~/ bc nombre_del_programa.b

se tienen las siguientes opciones:

usuario@maquina:~/ bc -l nombre_del_programa.b

El poner -l sirve para invocar la librería de funciones trigonométricas y de logaritmos.

usuario@maquina:~/ bc -q nombre_del_programa.b

El agregar -q Evita que aparezca el mensaje de bienvenida del programa.

usuario@maquina:~/ bc -lq nombre_del_programa.b

En éste caso se le estará indicando al programa que haga uso de las funciones trigonométricas y además se evitará también que aparezca el mensaje de bienvenida. Éste programa posee otras opciones pero éstas son las más utilizadas.

Ahora se muestra como se declaran las variables en éste noble y sencillo programa, con ayuda del siguiente ejemplo:

```
/* Éste programa sirve para enseñar a declarar variables en bc de forma sencilla
como se puede observar es muy importante el tener cuidado con la sintaxis
para no crear errores.
UNAM México  2010
Juárez Contreras Alejandro*/

x=45; /*Aquí se esta declarando cada una de las constantes */
y=3;
z=2;

x+y;  /* Estas son las operaciones que sean realizadas usando las constantes declaradas anteriormente*/
x/y;
y^z;

quit
```

Pero como se necesita disponer de un entorno interactivo a continuación se muestra la forma en que se puede lograr.

```
/* Éste programa sirve para enseñar de una forma muy sencilla las  bases para
hacer un programa interactivo con ayuda de bc
Juárez Contreras Alejandro
UNAM México 2010 */

print "escriba 2 números a sumar"
print "\n"
print " primer número: "; as = read ( )  /* Espera a que sea introducido el valor necesario */
print "segundo número: "; bs = read ( )
as+bs
quit
```

Éste lenguaje es tan sencillo que es excelente para divertirse experimentando con ensayo y error, cambiando cosas, con la idea de ver que sucede y luego utilizarlo en otros problemas.

Bucles

if

```
/* El siguiente programa enseña como utilizar la condición if
Juárez Contreras Alejandro
UNAM México 2010*/

print " El siguiente programa resta 3 a un número si éste es mayor a 5 \n""

print "escriba un número: \n";x=read ( );


if (x>5) {   /* Aquí se puede ver claramente la condición a cumplir */

print "el resultado obtenido es: \n"

x-3          /* si se cumple, se produce un resultado  */

}             /* Tanto la llave de inicio como la del final son muy importantes */
quit       /* En caso de no ser cierta simplemente se acaba el programa */
```

El ejemplo siguiente es ideal en caso de tener varias operaciones o procesos a resolver, dentro de un mismo problema.

```
/* El siguiente programa enseña como utilizar la condición if */
print " El siguiente programa resta 3, multiplica por 2 \n"
print"  y eleva un número al cuadrado solo si éste es mayor a 5 \n"
print "escriba un número: \n";x=read ( );


if (x>5) {   /* Aquí se puede ver claramente la condición a cumplir */
print "los resultados obtenidos son: \n"

x-3          /* si se cumple, se produce un resultado  */
x*2          /* nótese que no es necesario el ";" entre las funciones en éste caso */
x^2

}
quit       /* En caso de no ser cierta simplemente se termina el programa */
```

while

```
/* El siguiente programa enseña a usar la sentencia while
Juárez Contreras Alejandro
UNAM México 2010 */

x=1

while (a<5){
print “hola \n”

}
quit
```

Intente hacer el mismo programa sin “\n” y luego cambie el valor de x por cualesquier otro.

Para realizar un conjunto de operaciones simplemente se realiza la siguiente sintaxis, de esta forma se puede dar cuenta claramente, que esta sentencia puede ser usada como si se tratase de un simple for.

```
/* El siguiente programa realiza ”n” veces
   la operación con que fue programada
Juárez Contreras Alejandro
UNAM México 2010*/

print "escriba el número de iteraciones: \n"; a= read ( );
print “escriba un 1 por favor”;
print "escriba un número : \n"; x= read ( );

while (a<10) {        /* Aquí se puede ver claramente la condición a cumplir */
                      /* En éste caso se repetirá 10 veces */
print "el resultado obtenido es: \n"

a=a+1                 /* si se cumple, se produce un resultado  */

  x=x+3               /* esta es la operación a repetir en forma de bucle */

 x                    /* muestra los valores obtenidos durante cada iteración */

}

quit
```

¿Qué sucede si cambiamos el 1 por otro número? y ¿qué pasa cuando se pone otro valor distinto a 3 en la operación? Y si se usan números decimales ¿qué pasa?

for

Como ejemplo para la sentencia for se puede realizar el siguiente programa:

```
/* El siguiente programa enseña la forma de usar la
sentencia for
Juárez Contreras Alejandro
UNAM México 2010*/

for (i=0; i<10; i++)
print "hola"

quit
```

Como se puede observar es mucho más cómodo poder utilizar esta sentencia y no la while en caso de usar valores enteros. Bien se puede antojar usar i+1 (en vez de i++, aunque también se puede hacer uso de ++i); esto produce un error y el programa se desborda, por lo cual no hay que preocuparse simplemente se cierra Geany y se vuelve a abrir sin ningún tipo de problemas.

Se debe de recordar que la sentencia for es ideal para problemas en los que se conoce la cantidad de veces que se habrá de iterar, y que además de esto no se pueden utilizar números fraccionarios para realizar los pasos dentro del programa. En ese caso se podrá utilizar la sentencia while.

Nota importante:

La instrucción break sirve para abortar o finalizar la ejecución de cualquier bucle for o while desde cualquier punto del programa una vez concluida la condición de alto.

Asignaciones*

Dentro de muchos lenguajes de programación incluido el C, bc, Python, etc. se hace uso de i++ y ++i como una forma simplificada para indicar un incremento de uno en uno aunque en apariencia son exactamente lo mismo dentro de la sintaxis para un incremento. Esta se divide en:

a) notación postfija por ejemplo a= i++ lo que significa que primero se deberá de evaluar la expresión y posteriormente realizar la operación de incremento o decremento (a=i--), según sea el caso; es decir,

```
a=i;
i=i+1
```

b) notación prefija por ejemplo a= i++ lo que significa que primero se deberá de realizar la operación de incremento o decremento (a=i--) y después evaluar la expresión; es decir,

```
i=i+1;
a=i
```


*Nueva enciclopedia de la microcomputación teoría y práctica edición 1997 Ediciones EUROMEXICO Paginas 225-229

Funciones

Es posible definir funciones dentro de bc con el fin de que se puedan hacer uso de ellas en cualquier otro momento. Una muy buena idea es la de ir creando una librería propia de funciones, para lo cual bc dispone de la herramienta necesaria para hacerlo.

Lo cual es relativamente sencillo como a continuación se explica con el siguiente ejemplo:

```
define convck(x){     /* Las variables van entre paréntesis */
auto a                /* con esto se borran los valores obtenidos después de */
a=x+273.15            /*  de ser usados */
return (a)            /* una función retorna valores, por medio de la palabra return */
}
```

Una vez realizado todo esto, se guarda con el nombre de “convCK.b”. Siempre que se a creado una nueva función se deberá de comprobar su buen funcionamiento, lo cual se logra ingresando en bc y escribiendo en línea de la siguiente manera:

```
n = conversorck(23)
n
296.15
```

Ahora simplemente se guardamos el programa en algún lugar seguro para ir creando nuestra librería de funciones, para lo cual siempre se deberá de usar un nombre que sea fácil de asociar con el trabajo. A continuación se procederá a hacer uso de la misma dentro de un programa, aunque ésta puede ser utilizada en cualquier otro.

```
/* Éste programa sirve para convertir grados centígrados
a Kelvin, usando una función previamente definida  */

print "Éste programa sirve para convertir grados centígrados \n";
print " a kelvin \n";
print " introduzca la temperatura en centígrados \n"; y=read ( );

define convck(x){
auto a
a=x+273.15
return (a)
}

b=convck(y)
print “ equivale en grados Kelvin:\n ”
b
```

Éste programa se guarda con el nombre de conversorck.b y se hace funcionar con los procedimientos ya descritos anteriormente.

Python

Se trata de un lenguaje de programación muy sencillo y que tiene una muy alta portabilidad y que a últimas fechas se le a dado un gran seguimiento, lo que a la larga a hecho que existan muchísimas librerías y aplicaciones del mismo.

Se comenzará con el primer programa en python, como se puede ver se trata claramente de un lenguaje de programación muy sencillo. Los comentarios se escriben poniendo simplemente un # al principio.

Los programas en python se nombran con terminación .py

Si no corren las aplicaciones, no hay de que preocuparse, simplemente puede ser que se haya puesto algún acento dentro de la sintaxis o en los comentarios. Si es así hay que borrarlo cuanto antes y así se habrá encontrado la solución. ¡Python es un programa que odia los acentos! No se debe olvidar esto si es que no se quiere perder la cordura o muchas horas de sueño reparador.

```
#  primer_programa
#  hecho en python para comenzar a enseñar las bases de este lenguaje
#  escrito por Alejandro Juarez Contreras
#  como se puede ver se utilizaron mas lineas tan solo para el comentario
#
################################

print "Programar es divertido y muy facil"
```

A continuación se muestra un segundo programa tan sencillo como el anterior; pero no hay, que dejarse engañar fácilmente, la sencillez tiene un costo y es que además de volverse adictiva ésta puede llevar a caer en malas costumbres de programación. Aunque Python evita todo esto manteniendo ciertas reglas y obligando a que se cumplan de una forma inteligente.

```
#  segundo_programa
#  hecho en Python para comenzar a enseñar las bases de este lenguaje
#  escrito por Alejandro Juarez Contreras
#  como podemos ver utilizamos mas lineas tan solo para el comentario
################################

r=2**3
print r
```

Al igual que en C, se pueden utilizar “ \n” para dar una mayor claridad a los programas.

Como se podrá ver en ejemplo que se da a continuación, en Python existe una forma muy peculiar para volver a un programa interactivo.

```
#  potencia_programa
#  hecho en Python para comenzar a enseñar las bases de este lenguaje
#  escrito por Alejandro Juarez Contreras
#  como se puede ver se utilizan mas lineas tan solo para el comentario
################################

print "este programa eleva a una potencia un numero"
x = float(raw_input("Ingresa un numero, por favor: "))
y = float(raw_input("Ingresa otro numero, por favor: "))
z=x**y
print "la potencia es:", z

# hay que definir los tipos de numeros a utilizar
# float para numeros de punto flotante
# int para los enteros
# complex para los de tipo complejo
```

Python es sencillo con respecto a su sintaxis, lo cual se comprueba a continuación.

If

```
#  si_programa
#  hecho en Python para comenzar a enseñar las bases de este lenguaje
#  escrito por Alejandro Juarez Contreras
#  como se puede ver se han utilizado mas lineas tan solo para el comentario
################################

guess = int(input('Ingresa un numero por favor: \n '))
if cal >= 6 :
        print "aprobado"
else :
        print "reprobado"

#  si2_programa
#  hecho en Python para comenzar a enseñar las bases de este lenguaje
#  escrito por Alejandro Juarez Contreras
#  como se puede ver se han utilizado mas lineas tan solo para el comentario
################################
cal=input("ingrese una calificacion \n")
if cal >=8:
        print " Muy bien"
elif cal >= 6:
        print "Bien"
elif cal <=5:
        print "reprobado"
else:
        print "Regular"
```

```
#  cuenta.py
#  hecho en python para aprender a usar la sentencia for
#  escrito por Alejandro Juarez Contreras
#  como se podra observar se utilizan mas lineas tan solo para el comentario
#
#################################
for i in range(1, 6):
        i=i+1
        print(i)
else:
        print('el ciclo for a terminado por fin')
```

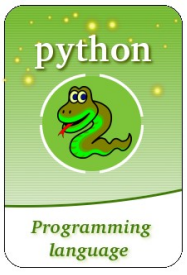
Aunque eliminemos las dos últimas líneas del programa éste seguirá funcionando perfectamente bien.

while

```
#  cuando.py
#  hecho en python para aprender a usar la sentencia while
#  escrito por Alejandro Juarez Contreras
#  como se puede ver se han utilizado mas lineas tan solo para el comentario
#
#################################
a = int(input('Introduzca un numero : '))
while a < 10 :
        print a
        a=a+1
```

Y aunque parezca increíble estas tres hojas pueden ayudar a aprender las bases de Python, con la ventaja de que se podrá encontrar programando de una forma más o menos regular en cuestión de unas cuantas horas, claro que esto será aún mayor dependiendo de cuánta práctica previa se halla tenido en otros lenguajes de programación. No hay que olvidar que se deberá de seguir estudiando éste lenguaje, puesto que existen muchos otros recursos que no se tratan en éste trabajo y que serán mencionados en un ejemplo de aplicación.

Advertencia: Python es un lenguaje altamente indentado*, por lo cual es necesario tener cuidado de entender este termino y respetar los espacios asignados en las estructuras de los programas tal cual están redactados en los ejemplos expuestos hasta aquí.



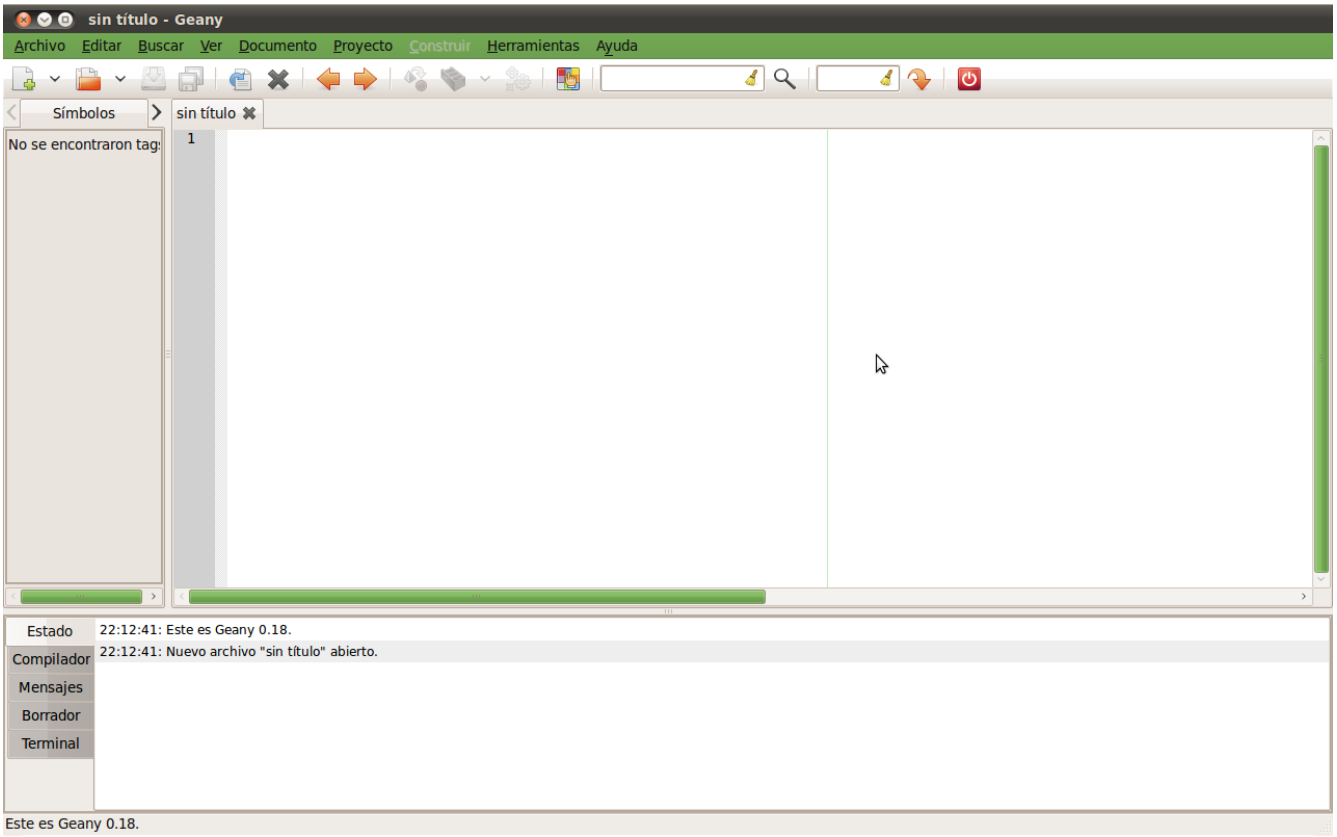
*_Indentación:_ Al igual que sangría o tipado es un termino y un habito de trabajo que se utiliza en programación para darle coherencia y limpieza a un programa evitando que éste pierda su facilidad de lectura, corrección o estudio por parte de cualquier programador.

Geany

Se trata de un entorno gráfico de programación con la ventaja de ser muy ligero, el cual consume muy pocos recursos, además de ser muy intuitivo en la creación de programas bajo diversos lenguajes de programación; también conocido como un IDE (Integrated Development Enviroment o dicho de otro modo se trata de un Entorno de Desarrollo Integrado); Una vez instalado dentro del Sistema Operativo se puede encontrar en el menú de programación, y con tan sólo un clic se podrá invocar en la pantalla, como se puede observar se trata de un entorno con un gran parecido a un block de notas pero con varias herramientas incluidas entre ellas estado, compilador, mensajes, borrador y terminal.

Si bien existen otros IDES, éste es particularmente ligero y puede ser utilizado por gente novata en máquinas con muy pocos recursos de memoria y procesador.

Se explicara a continuación la manera de utilizarlo con un ejemplo sencillo en Gfortran, esta secuencia es valida para cualesquier otro lenguaje de programación, siempre y cuando se tengamos su compilador incluido en la distribución.



Como se puede observar en la figura anterior al invocar a éste IDE ya se tiene listo el entorno para comenzar a plasmar un primer programa, es conveniente que se empiece por nombrar el programa con su respectiva terminación y guardarlo en su respectiva carpeta. Esto tiene dos objetivos, el primero que las palabras reservadas aparezcan con distinto color y el segundo se comience a ver la estructura del código mismo, con el fin de corregir con mayor facilidad los errores que se cometan.

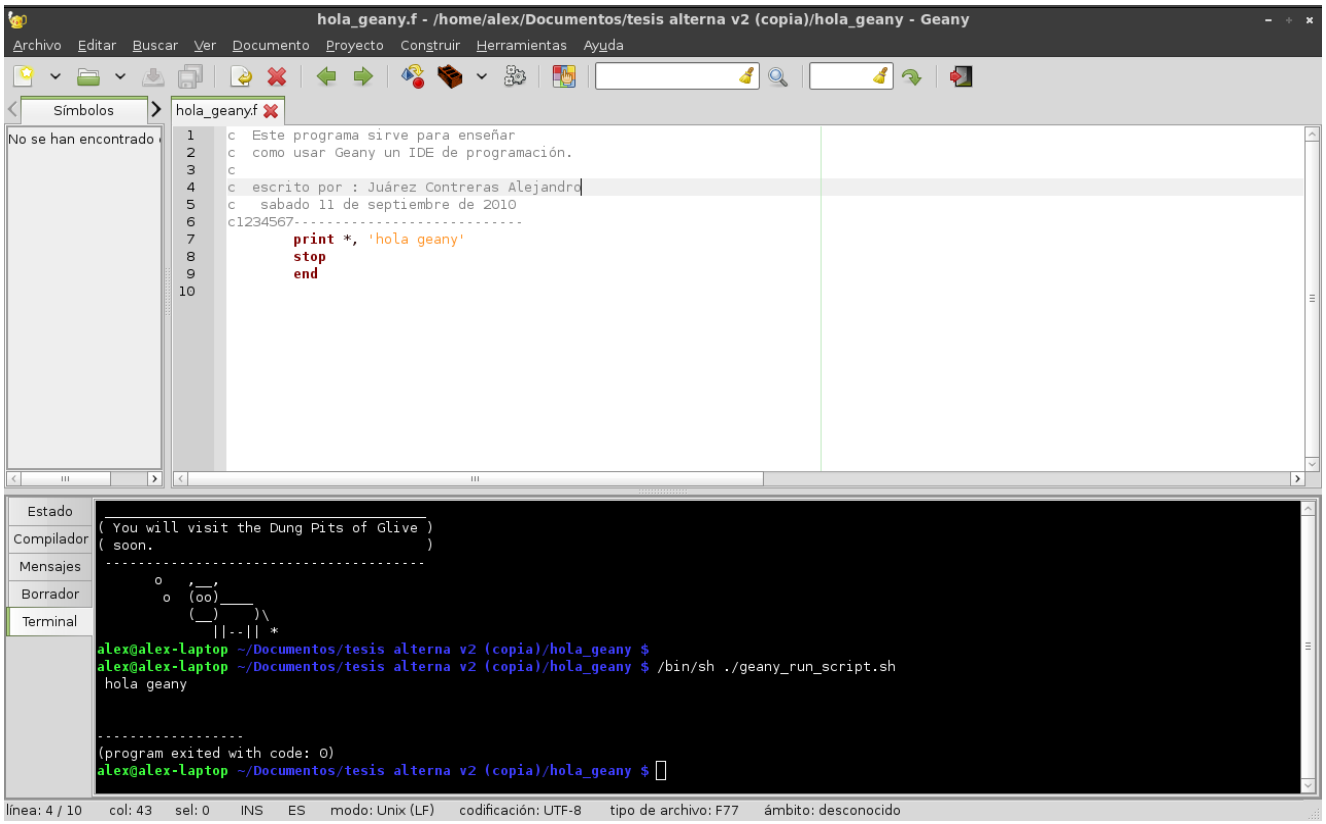
Para poder crear un primer programa en éste caso hola_geany.pas (es muy importante el no olvidarse de la terminación correcta del programa; si no éste nunca correrá correctamente); lo cual se puede hacer fácilmente desde el menú Archivo -->Guardar como: se elige la opción buscar otras carpetas, se desplaza por las carpetas hasta el lugar en donde se crea más conveniente el tenerla, y con la opción crear carpeta nueva, se nombra la carpeta que lo contendrá en éste caso hola_geany.

Después simplemente se escribe el código correspondiente y en el menú del IDE se elige el botón según la secuencia correspondiente.

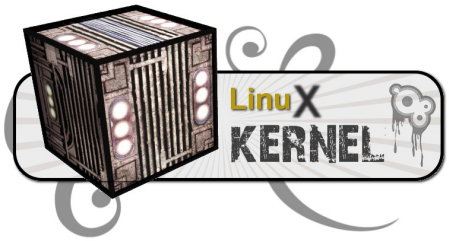
figura 1	figura 2	figura 3
Compilar el archivo actual	Construir el archivo actual	Ejecutar o ver el archivo actual
		

Una vez hecho esto se podrá ver a través de las pestañas del IDE diversos mensajes, lo cual facilitará la corrección de los programas.

Por último se puede ver como en la pestaña de terminal comienza a correr nuestro pequeño programa y según sea el caso éste pedirá más datos o simplemente desplegara algún letrero en pantalla; Éste IDE sirve en realidad para encontrar errores de forma muy práctica y así evitar algunos de los errores más comunes al intentar localizar una carpeta determinada. Sobre todo no se debe de olvidar que programar es divertido.



Una característica muy importante de Geany es que cuando se quiere hacer un programa a partir de las plantillas integradas que éste tiene, pues aparece en forma automática la licencia GPL; esta ayuda sirve para proteger nuestra obra intelectual con el fin de que pueda continuarse distribuyendo de forma libre y legal, con la ventaja de que la comunidad podrá modificarlo, distribuirlo o estudiarlo para mejorarlo sin que pueda perderse después por algún mal manejo por parte de un programa de tipo privado.



Hoja de cálculo OpenOffice

Ofrece una poderosa hoja de cálculo con la cual se pueden hacer iteraciones, con la ventaja de que se trata de una aplicación gratuita, sin tener problemas de piratería y pudiéndose usarla sin ningún tipo de restricción.

Para lo cual se usara como ejemplo un problema resuelto por el método de aproximaciones sucesivas

siendo F(x)=0; en la cual x=a es una raíz de ella; es decir, un valor "a" que la vuelva igual a 0.

F(a)=0.

Éste método se explica con más detalle en el anexo sobre aproximaciones sucesivas.

El método de aproximaciones sucesivas consiste en sustituir x_0 (el cual es un valor aproximado u estimado), obteniéndose un nuevo valor x_1; es decir, una nueva aproximación a la raíz.

Se sigue haciendo esto hasta obtener el valor u aproximación requerida por nuestros fines particulares.

Se procederá a encontrar una raíz de x= $\sqrt{0.6}$

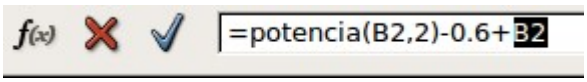
Se comienza por despejar x de ésta formula lo cual dará $x^2=0.6$, luego se iguala a 0 la ecuación quedando $x^2-0.6=0$; sumando x a ambos lados de la ecuación dará $x^2-0.6+x=x$

Se puede usar la hoja de cálculo de OpenOffice, Gnumeric, LibreOfice, etc.

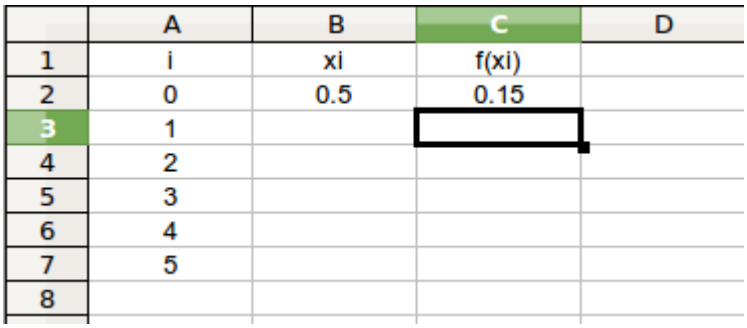
Primero se nombraran cada una de las celdas, la primera con la letra i servirá para saber el número de iteraciones que se van a hacer, la siguiente con la etiqueta x_i que define el valor inicial aproximado o supuesto, y justo después $f(x_i)$ que dará el valor resultante de cada una de las iteraciones.

	A	B	C	D
1	i	xi	f(xi)	
2	0			
3	1			
4	2			
5	3			
6	4			
7	5			
8				

Segundo se Introduce la formula en la barra de funciones (si se trata de una formula muy compleja, se puede ir resolviendo en partes para evitar errores, y facilitar su manejo); utilizando el primer valor de x_i, se puede suponer como valor inicial 0.5.



Se da Enter obteniéndose los primeros valores,

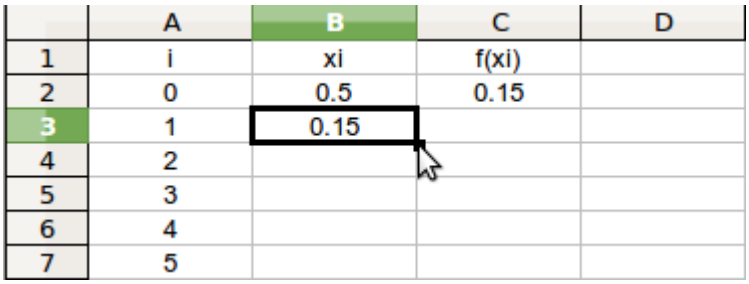


	A	B	C	D
1	i	xi	f(xi)	
2	0	0.5	0.15	
3	1			
4	2			
5	3			
6	4			
7	5			
8				

!Mucha Atención¡

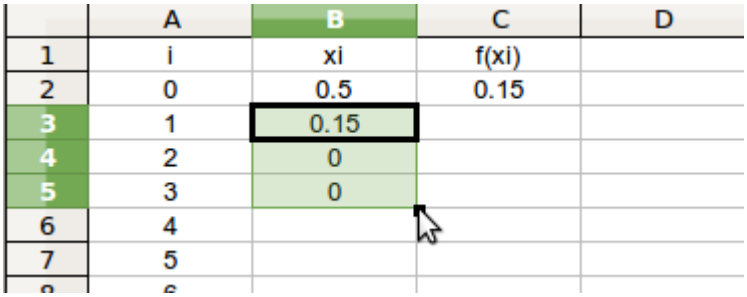
Una vez obtenido el primer valor se escribe C2 justo en la celda de abajo en éste caso B3 dando Enter, con lo cual se logra que el cálculo sea recurrente. Es de suma relevancia hacer correctamente éste paso, de lo contrario nunca funcionara esta técnica.

Luego se jala hacia abajo a partir de la celda B3 en donde aparece la flecha de la figura siguiente.



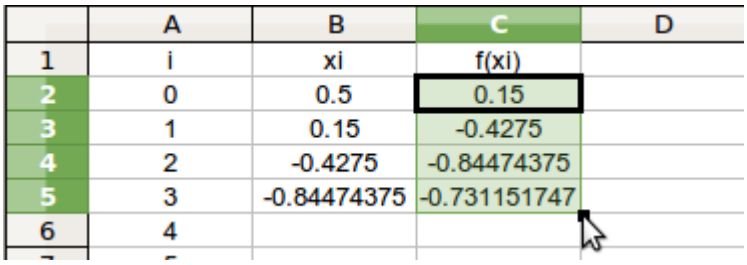
	A	B	C	D
1	i	xi	f(xi)	
2	0	0.5	0.15	
3	1	0.15		
4	2			
5	3			
6	4			
7	5			

Aparecerán varios ceros pero esto es normal, no hay que preocuparse por ello .



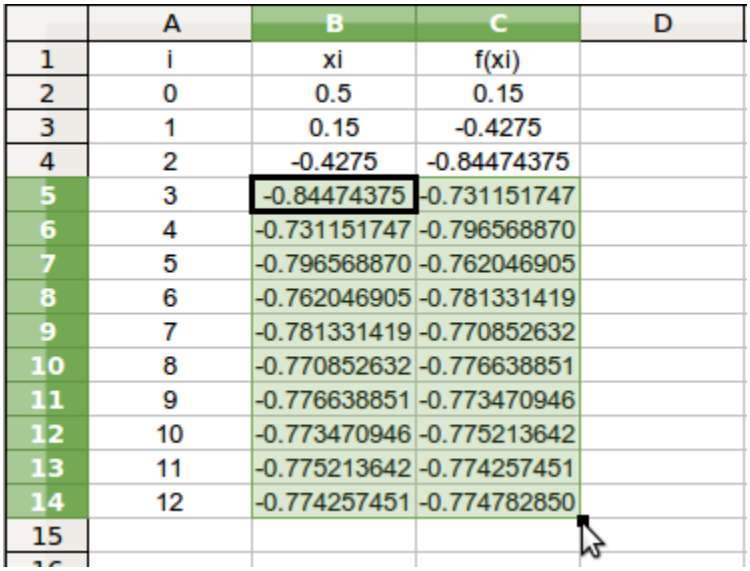
	A	B	C	D
1	i	xi	f(xi)	
2	0	0.5	0.15	
3	1	0.15		
4	2	0		
5	3	0		
6	4			
7	5			

Ahora se jala hacia abajo la celda C2 a partir del cursor en forma de cruz, obteniendo nuevos valores para la iteración.



	A	B	C	D
1	i	xi	f(xi)	
2	0	0.5	0.15	
3	1	0.15	-0.4275	
4	2	-0.4275	-0.84474375	
5	3	-0.84474375	-0.731151747	
6	4			

Después se resaltarán los renglones siguientes y se jalarán hacia abajo hasta obtener el valor deseado o hasta obtener la precisión requerida.



	A	B	C	D
1	i	xi	f(xi)	
2	0	0.5	0.15	
3	1	0.15	-0.4275	
4	2	-0.4275	-0.84474375	
5	3	-0.84474375	-0.731151747	
6	4	-0.731151747	-0.796568870	
7	5	-0.796568870	-0.762046905	
8	6	-0.762046905	-0.781331419	
9	7	-0.781331419	-0.770852632	
10	8	-0.770852632	-0.776638851	
11	9	-0.776638851	-0.773470946	
12	10	-0.773470946	-0.775213642	
13	11	-0.775213642	-0.774257451	
14	12	-0.774257451	-0.774782850	
15				

Éste valor se puede comprobar con ayuda de la calculadora que viene en el menú de accesorios simplemente al elevar al cuadrado el valor obtenido. Es deseable hacer más ejercicios que tengan solución conocida, con el fin de lograr además de un buen manejo de las técnicas aquí presentadas, aumentar la confianza en nuestros cálculos.

Como se puede ver se trata de una herramienta muy eficiente. Esta secuencia de pasos es valida para resolver problemas aún más complejos; por lo tanto es algo muy razonable el repetir éste ejemplo hasta poder realizarlo correctamente.

Esta poderosa herramienta permite además hacer uso de matrices. Justo en la ventana de ayuda se pueden encontrar instrucciones sencillas para hacer uso de los comandos pertinentes para el manejo de las mismas. Sin embargo cuando se es novato en el uso de esta hoja de cálculo se hará difícil su correcto manejo.

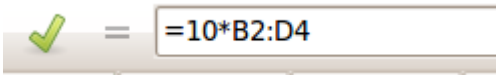
A continuación se muestra a través de algunos ejemplos como utilizar los comandos en forma efectiva.

Primero se empieza por definir una matriz tal cual se pudiera ver en una hoja de cálculo; ésta matriz de 3*3 quedará de la siguiente forma:

	A	B	C	D	E
1					
2		4	3	4	
3		3	4	3	
4		2	2	1	
5					

y será representada como: B2:D4; se nota como claramente el primer miembro de la matriz es representado por B2 y el último por D4. Con lo cual queda plenamente definida la matriz.

Para multiplicarla por un escalar, damos clic sobre cualquier celda que se encuentre libre acto seguido se escribe como sigue =10*B2:D4, en la barra de función



Se aprieta crtl+shift y después damos Enter obteniendo el resultado en forma de una matriz nueva.

	A	B	C	D	E
8		40	30	40	
9		30	40	30	
10		20	20	10	
11					

En éste ejemplo se puede observar claramente que se eligió la celda B8, la cual se encontraba vacía obteniéndose la matriz correspondiente.

Para una matriz m*n se puede seguir la misma secuencia de pasos.

!Mucha Atención¡

Es muy importante que al hacer cualquier operación que tenga que ver con matrices hay que apretar al mismo tiempo las teclas ctrl+shift y posteriormente dar un Enter para que el programa tome en cuenta la orden dada como una matriz. De lo contrario éste arrojara un error, o bien el mensaje #value.

Algunas notas sobre libreOffice

Para el momento en que se encuentra escribiéndose esta tesis, la empresa ORACLE, acababa de comprar a SUN microsystems y como es lógico los códigos que permanecían abiertos corrían el riesgo de verse afectados en cuanto a desarrollo, debido a las políticas de esta corporación, por lo cual la comunidad abrió la puerta a un proyecto alternativo llamado LibreOffice.

Es una particularidad muy interesante dentro del mundo de los proyectos de software libre que cuando acaba uno de estos, otro equipo lo retoma y prosigue con su desarrollo, asegurando la continuidad del mismo y su mejoramiento ya sea bajo el mismo nombre que lo vio nacer o con otro; esto asegura que todo ese trabajo seguirá siendo viable, o que también llegado el caso el usuario pueda seguir adecuando estas herramientas debido a que se tiene en manos del usuario el código fuente y la libertad para modificarlo.

Los pasos que anteriormente fueron descritos en el capitulo dedicado a OpenOffice sirven perfectamente para resolver problemas con libreOffice, Gnumeric y cualquier otra hoja de calculo que se utilice.

Puede esperarse que el proyecto OpenOffice siga avanzando y mejorando por parte de ORACLE y la comunidad de software libre y que éste no se quede estancado; además de que libreOffice siga dando facilidades a la comunidad para mantener el código abierto y disponible para todo el mundo.

Nota importante:

En las siguientes páginas se encontrará información relacionada con algunas de las muchas herramientas de software libre de muy alta calidad y que compiten directamente con otras de tipo privativo; como se ha expresado casi al principio de esta tesis no se pretende el profundizar por completo en el uso de cada una de ellas.

Octave

Aunque existen varias interfaces, para el manejo de éste entorno de programación dentro de los sistemas GNU/Linux, éste a sido primordialmente diseñado para ser usado a nivel terminal. Para poder invocarlo simplemente se escribe octave desde terminal y aparecerá en línea *octave:1>* ; lo cual indica que ya se esta dentro del entorno de trabajo.

Los operadores fundamentales suma +, resta -,multiplicación *, división /, potenciación ** ó ^, como sucede siempre en todo lenguaje de programación en éste también se respeta la jerarquía de operaciones y de paréntesis.

Para salir de éste programa simplemente se escribe exit ó quit y se pulsa Enter.

Siempre es agradable el trabajar en una terminal limpia lo cual se logra al escribir el comando clc.

Ejemplos:

Se debe de recordar que en octave se puede asignarle valor a las variables de la siguiente manera; por ejemplo a=3, b= 9 y si se escribe en la terminal a+b aparecerá:

```
octave:4> a=3
a =  3                    lo cual indica que se le asigno en memoria el valor de 3 a la variable b
octave:5> b=9
b =  9
octave:6> a+b
ans =  12
```

nota: estos valores permanecerán en memoria durante toda la sesión.

Matrices

Para poder introducir matrices o vectores dentro de éste entorno se hace lo siguiente, por ejemplo sea el caso de dos vectores a=(4,2,1) y b=(2,3,1) a sumar

```
octave:1> a=[4 2 1]
a =
   4   2   1
octave:2> b=[2 3 1]
b =
   2   3   1
octave:3> a+b
ans =
   6   5   2
```

Para el manejo de matrices se puede proceder de la siguiente manera:

```
octave:1> a=[3 2 1 4; 2 5 6 2; 5 6 4 0]
a =

   3   2   1   4
   2   5   6   2
   5   6   4   0

octave:2> b=[2 3 4 1; 0 9 8 1; 6 5 3 7]
b =

   2   3   4   1
   0   9   8   1
   6   5   3   7

octave:3> a+b
ans =

    5    5    5    5
    2   14   14    3
   11   11    7    7
```

Como se pudo observar en los ejemplos anteriores es muy fácil hacer uso de las matrices desde una terminal con ayuda de éste entorno de programación.

Para poder programar dentro de Octave es necesario el saber que sus programas tienen extensión .m y que los comentarios siempre comienzan con el signo #

```
# hola.m
# Éste programa muestra las bases de programación en Octave
# Escrito por Juárez Contreras Alejandro
##############################
disp(' programar es divertido');
printf(' y muy fácil ' );

# note las dos formas de imprimir en pantalla un texto
# como siempre se usará un programa interactivo
# ponga atención en la siguiente línea
################################

x = input( "\n ; introduzca su edad " );
y=x*12;
printf('Entonces usted tiene %i meses de edad', y);

# justo en donde puso el símbolo %i aparecerá el valor del cálculo
# Esto se hereda del lenguaje C
```

if

```
# califica.m
# muestra la forma de utilizar la sentencia if
#Escrito por Juárez Contreras Alejandro
#######

disp("introduzca una calificación\n");

x = input('a evaluar: ');

if (x>=8 )
printf ("muy bien\n");

elseif (x>= 6)
printf ("suficiente\n");

else
printf ("reprobado\n");

endif
```

Nota importante:

En caso de querer utilizar el símbolo “\n” para hacer más limpia la salida de instrucciones en pantalla, se debe recordar que éste no funcionará si la pone en entrecomillados simples; es decir, '\n' . Por otro lado hay que asegurarse de poner comillas dobles de cada lado de los mensajes de salida "*mensaje*" y no los siguientes símbolos “*mensaje*”, pues nunca funcionará el programa.

for

```
# cuenta.m
# muestra la forma de utilizar la sentencia for
#Escrito por Juárez Contreras Alejandro
#######

for i = 0:10

i=i;

printf ("cuento uno a uno hasta el diez%i\n",i);

endfor
```

La notación 0:10 indica que se incremente de uno en uno, desde el cero hasta el diez. ¿Cómo se lograría que contara hasta el numero 11 el anterior programa? Proponga al menos 2 maneras distintas.

switch-case

```
# decisiones.m
# muestra la forma de utilizar la sentencia switch-case
#Escrito por Juárez Contreras Alejandro
#######

printf("Éste programa le dice hasta 4 \n");
printf("en letras \n");

x=input('introduzca una número:');

switch x

case (x=1)
printf("uno\n");

case (x=2)
printf("dos\n");

case (x=3)
printf("tres\n");

case (x=4)
printf("cuatro\n");

otherwise
printf("adiós\n");

endswitch
```

while

```
# cuenta2.m
# muestra la forma de utilizar la sentencia while
#Escrito por Juárez Contreras Alejandro
#######

while (i <= 10)

i=i+1;

printf("los números del 1 al 10 son:%i\n", i)

endwhile
```

do-until

```
# cuenta3.m
# muestra la forma de utilizar la sentencia do-until
#Escrito por Juárez Contreras Alejandro
#######

i = 0;

do

i=i+1;

printf("\n los números del 1 al 7 son:%i", i);

until (i == 7)
```

Aunque estos programas ya no tienen casi comentarios, se puede notar que si se ha seguido una secuencia lógica al estudiar y programar varios de los ejemplos anteriores el insertarlos ya no representaran ningún problema debido a la sencillez con la que se puede realizar y estudiar su sintaxis.

Se debe de pensar muy bien antes de volverse un programador serio; procurando que sea divertido programar ya que no se trata de un trabajo sencillo.

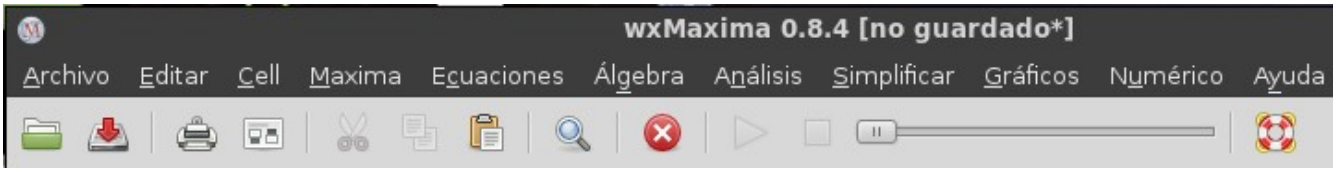
Maxima

Se trata de un excelente sistema de cálculo simbólico el cual es publicado bajo licencia GNU/GPL. Pero éste programa al funcionar bajo el entorno gráfico LXDE, produce un pequeño problema al hacer los cálculos, pues simplemente no los realiza de forma automática, en éste caso se utiliza el comando ***evaluar celda***, que se encuentra en el menú ***cell***.

Debido a que Maxima esta diseñado para ser usado a través de la terminal, se usara el entorno gráfico wxMaxima que le sirve perfectamente de complemento gráfico a Maxima para introducirlo en su manejo; prácticamente éste entorno hace que uno se enamore de la sencillez con que se pueden realizar muchísimas operaciones en forma casi automática.

El siguiente ejemplo gráfico de como usar matrices en wxMaxima basta para darse cuenta de la sencillez de su uso, justo para que después uno pueda adentrase por sí mismo a éste pizarrón electrónico.

Dentro del menú ***Álgebra*** se escoge la opción ***introducir matriz*** y en el cuadro de diálogo que aparece en pantalla se introduce el nombre, numero de filas y columnas que se le han asignado a ésta para poder realizar el cálculo correspondiente.



En éste caso le pondremos la letra “a” en nombre y al darle clic en aceptar, justo aparece el siguiente cuadro de diálogo para crear una matriz de 3*3.

Se procede a llenar los recuadros con la matriz que se necesite trabajar y una vez hecho esto, al dar clic en aceptar aparece en pantalla lo siguiente:

```
(%i11) a: matrix(
         [4,5,8],
         [5,3,6],
         [8,2,7]
       );
                [ 4  5  8 ]
                [         ]
(%o11)          [ 5  3  6 ]
                [         ]
                [ 8  2  7 ]
```

Otra vez se vuelve a escoger el menú de Álgebra y se selecciona trasponer matriz,

```
(%i12) transpose(%);
                [ 4  5  8 ]
                [         ]
(%o12)          [ 5  3  2 ]
                [         ]
                [ 8  6  7 ]
```

en éste mismo menú la opción determinante arrojara:

```
(%i13) determinant(%);
(%o13) -11
```

para multiplicar esta matriz por 2, se escribe en la pantalla la letra a*2, después se resalta la línea izquierda la cual se tiene que poner en verde

```
       -->  a*2
```

en el menú en ***cell*** se escoge la opción ***evaluar celda*** y se obtiene:

```
(%i14) a*2;
                [  8  10  16 ]
                [            ]
(%o14)          [ 10   6  12 ]
                [            ]
                [ 16   4  14 ]
```

Con wxMaxima se puede graficar, integrar, derivar, simplificar, encontrar soluciones a polinomios, hacer fracciones simples y un largo etc. Todo esto de una forma semejante a la que se observó en el ejemplo anterior. Con la ventaja de que se trata de un programa muy liviano, poderoso, de código libre, gratuito, que se puede modificar, estudiar, distribuir libremente y un largo etc.

Ésta hermosa herramienta es muy intuitiva y seguramente el tomarse toda una tarde para explorarla, jugar y entenderla redundará en grandes beneficios.

```
(%i29) integrate(x³+3, x);
                 4
(%o29)          x  + 3 x
                ---
                 4
```

R

El paquete electrónico estadístico por excelencia dentro del proyecto GNU/Linux, aunque sus propios diseñadores dicen que no lo es en sí, más bien éste tiene muchas funciones estadísticas (si no es que todas) y el hecho que su programación sea orientada a objetos lo hace un gran entorno de trabajo y programación.

Para comenzar a utilizar éste entorno de programación; primero se debe invocar el entorno de trabajo con simplemente escribir en la terminal R y dando un enter. Para salir escribe q( ) y se da enter.

Por ejemplo se tiene una lista de números que representan una serie medidas de temperaturas tomadas en el laboratorio y se quiere conocer su media, el valor mínimo, el máximo, etc.

Temperaturas 34.5, 33.6, 37, 39.4, 35.6, 32.2, 34.5, 32.9

para poder ser utilizadas correctamente se les asigna a un vector de la siguiente manera:

temperatura<-c(34.5, 33.6, 37, 39.4, 35.6, 32.2, 34.5, 32.9)

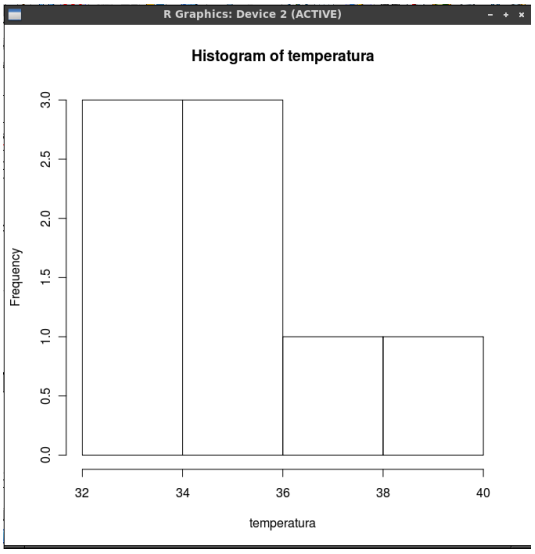
Se debe de notar claramente que se trata de un signo menor (<) y un signo menos (-) sin espacios entre ellos seguidos de una letra c

Si ahora se escribe max(temperatura), en terminal aparecerá el valor de 39.4, para otras series de valores se puede hacer uso del siguiente cuadro de comandos, éste entorno tiene muchas más posibilidades y realmente es mucho más fácil y rápido de manejar que una hoja de cálculo.

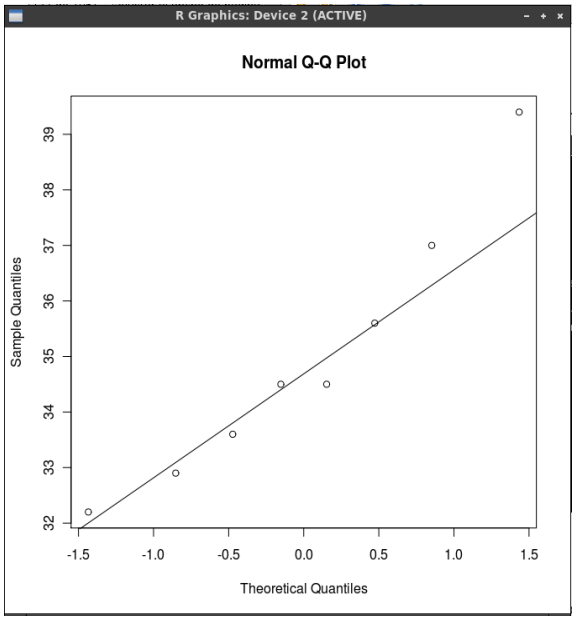
Función	***Operación***
max(x)	valor máximo en el vector x
min(x)	valor mínimo en el vector x
sum(x)	sumatoria de valores en x
mean(x)	media aritmética de x
median(x)	mediana de x
range(x)	rango de x (valor menor y valor mayor)
round(x)	valores redondeados de x
var(x)	varianza de x
sd(s)	desviación estándar de x
sort(x)	el vector x ordenado de menor a mayor
summary(x)	cuartiles, media, valor mínimo y máximo
sample(x, y)	muestra aleatoria de tamaño y ≤ n de x

Se tiene la posibilidad también de hacer hermosas gráficas con tan solo usar comandos de línea. A continuación se darán algunos ejemplos:

Al escribir en terminal hist(temperatura) se obtiene el histograma correspondiente:



Con la orden qqline(temperatura) se obtiene una recta que pasa por donde los cuartiles de la distribución de los datos :



Como se puede observar las gráficas son de alta calidad y mucho más rápidas de realizar que en una hoja de cálculo. Éste programa posee muchísimas funciones y se trata además de un entorno muy agradable y extremadamente rápido para el manejo de grandes cantidades de datos.

Scilab

Éste es un producto desarrollado por el INRIA (Institut Nationale de Recherche en Informatique et en Automatique) y el ENPC (Ecole Nationale des Ponts et Chaussées) de Francia el cual se decidió liberar bajo la CeCILL License.

Esta licencia dá permiso de utilizar, copiar y distribuir el software en forma legal; éste entorno se sigue mejorando día con día con la finalidad de ser implementado tanto en la industria, educación e investigación; siempre y cuando se haga referencia a los autores de este programa.

Se puede disponer del código fuente para su estudio y mejoramiento. Una de las finalidades de sus desarrolladores es la mantener esta aplicación con los últimos avances científicos en materia de cálculo numérico.

Para comenzar con este breve recorrido por el manejo de esta herramienta; se recomienda mantener siempre la pantalla de trabajo limpia a través del comando ***clc.*** Aunque también se puede ir directamente al menú ***preferences*** y elegir la opción ***Clear Console***.

Matrices

El lenguaje de Scilab esta enfocado particularmente al manejo de matrices, lo cual lo hace un instrumento muy poderoso y fácil de utilizar.

Para introducir una matriz, se procede de la siguiente manera a través de la línea de terminal:

-->a=[1 2 3 ; 2 4 6; 3 4 5] se da Enter y se obtiene:

a =

1. 2. 3.
2. 4. 6.
3. 4. 5.

Ahora se puede introducir otra matriz por ejemplo -->b=[2 4 3; 1 2 4; 2 3 1]

b =

2. 4. 3.
1. 2. 4.
2. 3. 1.

Teniendo estas dos matrices en memoria se pueden sumar con la siguiente línea
-->a+b
ans =

3. 6. 6.
3. 6. 10.
5. 7. 6.

El comando inv( ) es muy útil debido a que permite obtener la inversa de una matriz. como ejemplo se va a utilizar la matriz b que ya se tiene en memoria.

```
-->inv(b)
ans  =

 - 2.     1.     2.
   1.4  - 0.8  - 1.
 - 0.2    0.4    0.
```

Complejos

Los números complejos se pueden utilizar, de la siguiente manera:

```
-->a=5+4*%i
 a  =

   5. + 4.i

-->b=4+5*%i
 b  =

   4. + 5.i

-->a+b
 ans  =

   9. + 9.i
```

El ejemplo anterior de números complejos puede ser utilizado tanto en la resta como en la multiplicación y división de los mismos. Estos también se pueden utilizar en las operaciones de matrices. Como se puede ver en el siguiente ejemplo:

```
-->a=[2+3*%i 4+2*%i ; 5+3*%i 7-2*%i ]
 a  =

   2. + 3.i    4. + 2.i
   5. + 3.i    7. - 2.i
```

Con lo anteriormente visto será más que suficiente para poder hacer algunos cálculos importantes dentro de la materia de álgebra superior que se imparte dentro de la carrera de ingeniería química de la Facultad de Química de la UNAM. Esta aplicación tiene su propia sintaxis de programación, pero como no es el fin de esta tesis el profundizar en la misma se dará por terminada esta sección.

Aplicaciones en ingeniería química

Los ingenieros químicos hacen uso de una gran cantidad de fórmulas a lo largo de toda su carrera y vida laboral, y en muchas de las ocasiones se tienen que hacer estas operaciones para grandes cantidades de información, por lo cual es conveniente hacer uso de alguno de los paquetes que hemos ido aprendiendo a utilizar a lo largo de esta tesis. Si bien es cierto que ahora existen calculadoras con muchas funciones integradas, los reportes y balances se entregan en forma general a través de archivos informáticos que sirven para su posterior uso, estudio o implementación. Estos también tienen la cualidad de poder ser enviados a otra parte del mundo.

Aunque no es nada nuevo el uso de computadoras para este tipo de tareas, si lo es la posibilidad de hacer uso de software libre y las ventajas de no depender por completo de alguna empresa en particular.

Conforme se dan pasos a lo largo de toda una vida profesional, se pueden ir creando librerías propias para su uso personal, o bien para compartirlas con los demás.

Para los siguientes ejemplos se utilizará indistintamente cualesquier lenguaje ya estudiado con anterioridad.

El lector tiene la entera libertad de hacer paso a paso un estudio de los mismos y experimentar con sus propias versiones, haciendo uso del método de ensayo y error, con lo cual adquirirá destreza, confianza y experiencia en el uso de la terminal, de la IDE y del entorno operativo que se este utilizando. Es de suma importancia que se desarrolle la costumbre de hacer diagramas de flujo para entender la mecánica de cada uno de los programas que antes y a continuación se presentan.

Por esta misma razón se insiste en la necesidad de hacer los suficientes comentarios para hacer más fácil tanto la enseñanza de los mismos como su posible corrección.

Aplicaciones sencillas

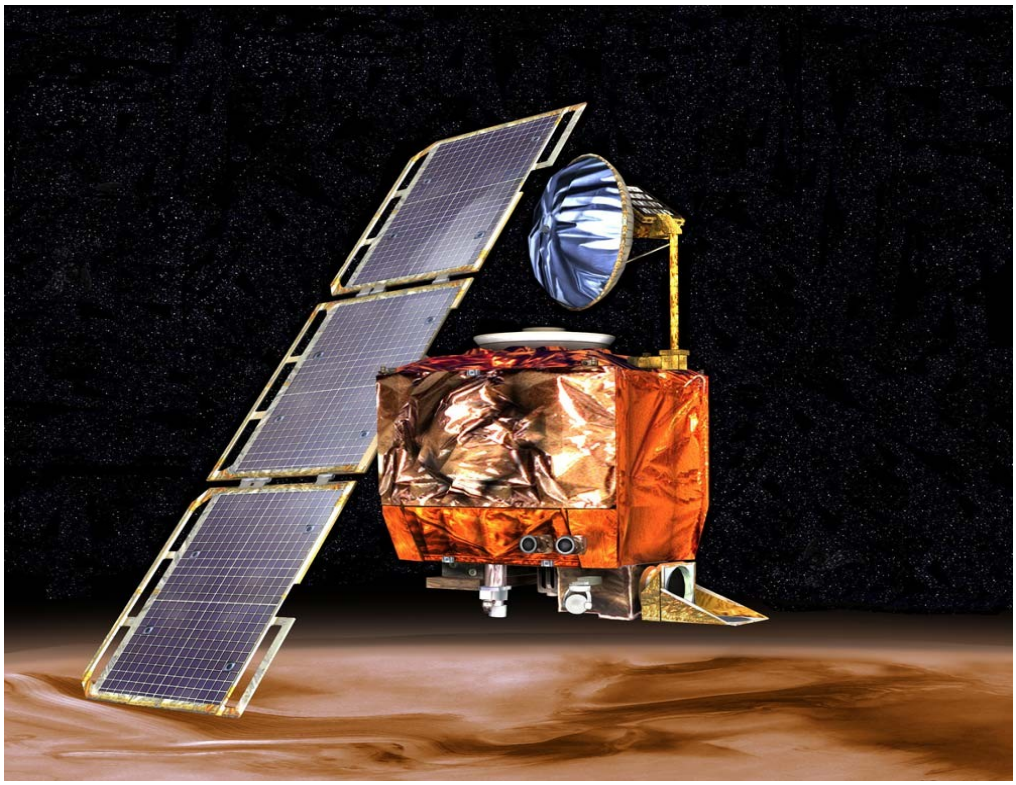
Se hace un énfasis especial en que lo sencillo no necesariamente deja de ser útil; en muchas de las ocasiones una sencilla aplicación puede ser muy importante para el manejo de grandes cantidades de datos. Una aplicación sencilla, si esta bien programada puede ser fácilmente transportada hacia un subprograma que en conjunto con otros subprogramas pueden llegar a crear un programa sumamente complejo. Las cosas más maravillosas de la naturaleza están hechas por otros miles o millones de cosas sencillas.

Muchos problemas grandes y difíciles se pueden descomponer en otros más pequeños y fáciles, asegurándose así de su correcta solución.

Siempre que se habla de hacer un programa que resuelva problemas sencillos se tiende a despreciar su verdadera importancia, antes de hacer esto deberíamos de tener en mente lo que ocurrió durante el año de 1999, cuando fue trasmitida la noticia de que la sonda Mars Climate Observer enviada por la NASA terminó por estrellarse dentro de la atmósfera marciana quedando ésta por completo destruida; este “simple” error condenó a la misión a una pérdida de 125 millones de dólares, además de la pérdida de varios años de investigación y planeación.

¿Pero cómo ocurrió este accidente? todo comenzó cuando fueron contratadas dos compañías para trabajar en conjunto en la construcción de la sonda marciana, la Lockheed Martin Astronautics, de Denver que hacía uso del sistema de unidades inglesas en todos sus diseños y la Jet Propulsion Laboratory de Pasadena que utilizaba el Sistema Internacional de Unidades para sus desarrollos. La primera empresa envío los datos de los cálculos que realizó pero sin hacer especificación alguna sobre que tipo de unidades de medida utilizaba; mientras que la segunda empresa hizo su parte tomando en cuenta estos cálculos como si se trataran de las utilizadas por el Sistema Internacional de Unidades; simplemente se trato de un problema de comunicación entre las dos empresas en las cuales jugaron dos factores, el primero fue que se minimizó la importancia de confirmar los datos y sus unidades y la segunda fue la de no tomar en cuenta la posibilidad de crear un programa emergente, que bien hubiese podido ser creado con software libre y que pudiera corregir los datos introducidos en la computadora de abordo.

Del anterior problema se puede deducir que dar por sentado las cosas, la falta de comunicación, no tener un programa de respaldo y sobre todo la falta de comentarios, notas u observaciones en el diseño, planificación ya sea de un proyecto o diseño de programas puede ser fatal, si se hubieran revisado las unidades de diseño, un simple programa hecho con cualquiera de las implementaciones de software libre anteriormente vistas hubiese bastado para evitar todo el desastre anterior.



Conversiones

En todas las áreas de la ciencia y en la ingeniería química se vuelven imprescindibles este tipo de programas, que aunque sencillos ayudan a evitar más fácilmente las equivocaciones cuando se realizan muchas operaciones en forma repetitiva además que permite el ir crear bibliotecas propias que pueden llegar a compartirse, mejorar, vender, etc. Todo gracias a las libertades que otorgan el uso de las cuatro libertades básicas del software libre.

Una aplicación muy útil en muy particular en ingeniería y matemáticas es la siguiente es la de convertir los radianes a grados

Trigonometría

1 radianes = 57.2958 grados

con ayuda de bc el programa quedaría como sigue:

```
/* Este programa sirve para convertir radianes a grados
Juárez Contreras Alejandro 2010 */
scale = 2                                   /* decimales utilizados */
print "\n"                                      /*salto de línea*/
print "introduzca el valor en grados:"; xg = read ( ) /*imprime el mensaje*/
print "\n"
print "\n"
print " el valor en radianes es: "
xg/57.2958                                    /*imprime el resultado de la operación*/
print "\n"
quit                                          /*finaliza el programa*/
```

Dibujo de gráficas de gases reales

Para encontrar la solución a el problema de los volúmenes molares de gases reales, existen diversas ecuaciones que sirven para modelar su comportamiento; una de ellas es la ecuación de Van der Walls:

$$(P+\frac{a*n}{V^2})(V-n*b)=RT$$

en donde:

n= número de moles

P= presión en atm

V= volumen molar del gas en L/gmol

a= constante particular de un gas

b= constante particular de un gas

T= temperatura del gas en grados K

R= constante de los gases (0.08205 L*atm/K*gmol)

suponiendo que se hacen los cálculos sobre una base de 1 mol se tiene que la ecuación se reduce a

$(P+\frac{a}{V^2})(V-b)=RT$ al desarrollar este polinomio se obtiene:

$PV-bP+\frac{a}{V}-\frac{ab}{V^2}=RT$; que al multiplicarlo por V^2 $V^2*(PV-bP+\frac{a}{V}-\frac{ab}{V^2}=RT)$ se tiene:

$PV^3-bPV^2+aV-ab=RTV^2$ esta ecuación puede ser rescrita a su vez de la siguiente manera:

$PV^3-bPV^2-RTV^2+aV-ab=0$ y luego reordenando $PV^3-(bP+RT)V^2+aV-ab=0$

Tal y como se puede observar, se trata de un polinomio cúbico con respecto a V, el cual se puede reescribir de la forma siguiente:

$f(V)=PV^3-(bP+RT)V^2+aV-ab=0$

Esta ecuación puede ser resuelta por medio del método de Newton-Rapshon, con solo substituir los resultados correspondientes:

$x_1=x_0-\frac{f(x_0)}{f'(x_0)}$

$f(V)=PV^3-(bP+RT)V^2+aV-ab=0$

$f'(V)=3PV^2-2(bP+T)V+a=0$

quedando de la siguiente manera:

$V_1=V_0-\frac{PV^3-(bP+RT)V^2+aV-ab}{3PV^2-2(bP+RT)V+a}$

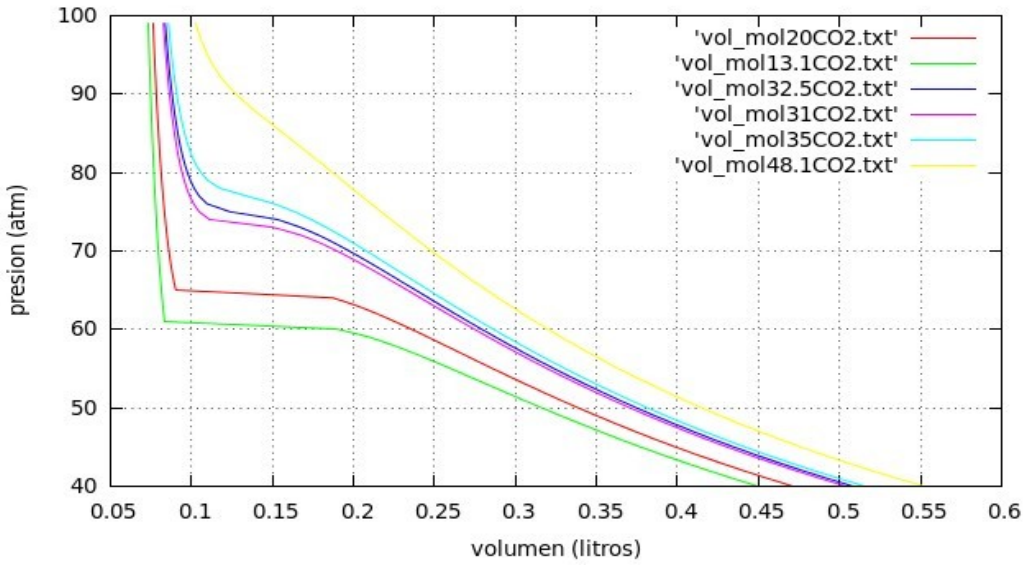
Con ayuda del siguiente programa se pueden dibujar las isotermas correspondientes a cualquier gas con solo tener los valores correspondientes de a y b, a una determinada temperatura en grados centígrados.

Para crear la gráfica correspondiente a los cálculos de este programa se utilizo gnuplot, un manual básico, recomendable y que puede ser descargado en formato pdf desde la red es el escrito por Juan José García Rojo Herramientas en GNU/Linux para estudiantes universitarios GNUPLOT: herramienta para gráficos de funciones y datos.

```
# Escrito por Alejandro Juarez Contreras
# UNAM Mexico
####################

print "Este programa sirve para calcular"
print "el volumen molar"
print "por medio de la ecuacion de Van der Walls"
print "con solo ingresar los valores de las constantes y la temperatura"
print "en grados centigrados"

a = float(raw_input("Ingresa el valor de a: "))

b = float(raw_input("Ingresa el valor de b: "))

t = float(raw_input("Ingresa el valor de T, centigrados por favor: "))

V=1

R=0.08205

T=t+273.15

#  La orden f = open('vol_mol.txt', 'w'), sirve para abrir un archivo con extension *.txt, llamado
#  vol_mol.txt y que sera guardado en la misma carpeta en que fue creado el programa; la 'w'
#  indica que se escribiran en forma automatica los datos obtenidos por este mismo.
#

f = open('vol_mol.txt', 'w')

# La orden for P in range(40, 100): sirve para hacer los calculos en un rango de 40 a 100 de uno en uno
# para la variable P

for P in range(40, 100):

 print 'El valor del volumen molar es' ,V


#  La sentencia while True junto con la if error < 0.0001: aseguran que se logre hacer la iteracion para
#  cada una de las presiones y temperaturas correspondientes.
#

 while True:

  x0=(P*(V**3))-((V**2)*((b*P)+(R*T)))+a*V-a*b

  x1=(3*P*(V**2))-(2*V*((b*P)+(R*T)))+a

  V1 = V-(x0/x1)
```

```
# este paso asegura que la condicion de error sea cumplida para una presion y temperaturas dadas

  error=abs(V1-V)

  V=V1

  if error < 0.0001:

# una vez cumplida la condicion de error se detiene el programa por medio de la sentencia break y se
# prosigue con el siguiente calculo

     break

#  la sentencia f.write(str(V) + ' ' + str(P) + '\n') sirve para escribir dentro del archivo *.txt los valores
#  obtenidos por medio del programa en forma de columnas, siendo la primera la del volumen y
#  la segunda la de la presion.
#

 f.write(str(V) + ' ' + str(P) + '\n')

#  por ultimo al acabar el programa cerramos el archivo *.txt con la orden f.close( )
#

f.close( )
```

Se recomienda hacer el diagrama de flujo de este programa para que posteriormente al correrlo y compararlo con el código este sea mucho más fácil de entender. Debido a que Python tiene una muy alta indentación y para evitar que se incurran en errores se vuelve a repetir a continuación todo el código de este programa sin comentarios.



Gráfica 1

```
# Escrito por Alejandro Juarez Contreras
# UNAM Mexico
####################

print "Este programa sirve para calcular"
print "el volumen molar"
print "por medio de la ecuacion de Van der Walls"
print "con solo ingresar los valores de las constantes y la temperatura"
print "en grados centigrados"

a = float(raw_input("Ingresa el valor de a: "))

b = float(raw_input("Ingresa el valor de b: "))

t = float(raw_input("Ingresa el valor de T, centigrados por favor: "))

V=1

R=0.08205

T=t+273.15

f = open('vol_mol.txt', 'w')

for P in range(40, 100):

 print 'El valor del volumen molar es' ,V

 while True:

  x0=(P*(V**3))-((V**2)*((b*P)+(R*T)))+a*V-a*b

  x1=(3*P*(V**2))-(2*V*((b*P)+(R*T)))+a

  V1 = V-(x0/x1)

  error=abs(V1-V)

  V=V1

  if error < 0.0001:

     break

 f.write(str(V) + ' ' + str(P) + '\n')

f.close()
```

Una versión más completa del anterior programa es la siguiente:

```
print "            Este programa sirve para calcular"
print "                    el volumen molar"
print "        por medio de la ecuacion de Van der Walls"
print
print "con solo ingresar el valor de las constantes y la temperatura"
print "                  en grados centigrados"
print
print "  o bien puedes elegir de que gas se trata y su temperatura"
print
print "     Si Conoces el valor de las constantes a y b marca 0"
print
print "CO2            marca 1        H2O            marca 5"
print "hidrogeno    marca 2        etano          marca 6"
print "metano       marca 3        nitrogeno      marca 7"
print "oxigeno      marca 4        argon          marca 8"
print

Num = float(raw_input("  tu opcion es: "))
print

if Num == 0:

   print "escogiste valores de a y b"

   a = float(raw_input("Ingresa el valor de a: "))

   b = float(raw_input("Ingresa el valor de b: "))

elif Num == 1:
    print "escogiste el CO2"
    a=3.59
    b=0.0427

elif Num == 2:
    print "escogiste el hidrogeno"
    a=0.244
    b=0.0266

elif Num == 3:
    print "escogiste el metano"
    a=2.25
    b=0.0428

elif Num == 4:
    print "escogiste el oxigeno"
    a=1.36
```

```
    b=0.0318

elif Num == 5:
    print "escogiste el H2O"
    a=5.46
    b=0.0305

elif Num == 6:
    print "escogiste el etano"
    a=5.49
    b=0.0638

elif Num == 7:
    print "escogiste el nitrogeno"
    a=1.39
    b=0.0391

elif Num == 8:
    print "escogiste el argon"
    a=1.35
    b=0.0322

else:
    print 'Vuelve a intentarlo'

print

t = float(raw_input("  Ingresa el valor de T, centigrados por favor: "))

V=1

R=0.08205

T=t+273.15

f = open('vol_mol.txt', 'w')

for P in range(40, 100):

 print 'El valor del volumen molar es' ,V

 while True:

  x0=(P*(V**3))-((V**2)*((b*P)+(R*T)))+a*V-a*b

  x1=(3*P*(V**2))-(2*V*((b*P)+(R*T)))+a

  V1 = V-(x0/x1)
```

```
  error=abs(V1-V)

  V=V1

  if error < 0.0001:

     break

 f.write(str(V) + ' ' + str(P) + '\n')

f.close()

print
print "Los calculos se realizaron a una temperatura (centigrados) de:",t
```

En este programa se utilizo la estructura if-elif-else

```
if condición 1
   acción
elif condición 2
   acción
elif condición 3
   acción
         .
         .
         .
else condición_por_defecto
   acción
```

Los programas anteriores devuelven un archivo con extensión *.txt, tal como se puede ver en el ejemplo siguiente.

Se abre la terminal y se crea una nueva carpeta conteniendo el programa, el cual deberá de correrse bien por medio de terminal o de Geany, como ejemplo para el CO2 a 13.1 °C con a=3.59 y b=0.0427 se obtienen los siguientes resultados correspondientes al volumen molar y la presión.

```
0.448986122137   40
0.433434052866   41
0.418507996151   42
0.404154995303   43
0.390325576049   44
          .
          .
          .
          .
0.0739664012381  95
0.0738088549897  96
0.073654473943   97
0.0735031348828  98
0.0733547217469  99
```

En la carpeta correspondiente se cambia el nombre del archivo vol_mol.txt por el de vol_mol13.1CO2.txt y así sucesivamente con diferentes corridas de datos, obteniéndose una biblioteca de datos que pueden ser utilizados tanto para ser graficados como para poder ser estudiados. Con el siguiente script de gnuplot se puede obtener la gráfica correspondiente.

plot 'vol_mol20CO2.txt' with lines,'vol_mol13.1CO2.txt' with lines,'vol_mol32.5CO2.txt' with lines, 'vol_mol31CO2.txt' with lines , 'vol_mol35CO2.txt' with lines,'vol_mol48.1CO2.txt' with lines; set xlabel 'volumen'; set ylabel 'presion'; set grid; set autoscale

Para lograr tener una imagen en pdf, se puede optar por las opción que tiene integrada gnuplot para poder ser impreso y utilizado en cualquier otro lugar o bien se pueden seguir los siguientes pasos primero utilizar una captura de pantalla guardándola en formato jpg; luego recortar la imagen con shotwell(programa GNU/Linux GPL2), tercero escribir en terminal la siguiente línea de comando convert vanderwalls.jpg vanderwalls.pdf cuidando de hacerlo dentro de la carpeta correspondiente. Esto funciona si se tiene instalado el programa imagemagick(GNU/Linux GPL2); obteniéndose la gráfica 1.

Solución numérica de ecuaciones diferenciales

Una de las herramientas más utilizadas por los ingenieros químicos son las ecuaciones diferenciales con valor inicial, que pueden llegar a ser resueltas por medio de la programación de las distintas opciones que ofrece el código abierto. Justo antes de proceder a programar para resolver este tipo de problemas se debe de optar por escoger el método numérico que más convenga; En el siguiente ejemplo se muestra el uso del método de Runge-Kutta:

Para lograr resolver un problema de valor inicial. Primero se debe de dividir el intervalo que va de x_0 a x_1 en n subintervalos de ancho h

$$h=\frac{x_f-x_0}{n}$$

de esta manera se obtiene un conjunto de (n+1) puntos tales que $x_i=x_0+ih, 0\leqslant i\leqslant n$

Método de Runge-Kutta

$$y_{i+1}=y_i+\frac{h}{6}(K_1+2K_2+2K_3+K_4)$$

en donde:

$$K_1=f(x_i,y_i)$$

$$K_2=f(x_i+\frac{h}{2},y_i+\frac{hK_1}{2})$$

$$K_3=f(x_i+\frac{h}{2},y_i+\frac{hK_2}{2})$$

$$K_4=f(x_i+h,y_i+hK_3)$$

A continuación se procede a resolver el siguiente problema con valor inicial, por medio del método de Runge-Kutta.

$$PVI\begin{Bmatrix}\frac{dy}{dx}=(x-y)\\ y(0)=2\\ y(1)=?\end{Bmatrix}$$

Este problema se puede resolver con un método semimanual en el cual se puede observar claramente del desarrollo del mismo.

Este problema será subdividido en 5 subintervalos iguales obteniéndose el valor de:

$$h=\frac{1-0}{5}=0.2$$

Para obtener el resultado de este problema se va a hacer uso del lenguaje python de GNU/Linux; recordando que este lenguaje no acepta acentos dentro su sintaxis.

```
# Escrito por Alejandro Juarez Contreras
# UNAM Mexico
####################

print "Este programa sirve para calcular por medio de Runge-Kutta"

x0 = float(raw_input("Ingresa el valor de x0, por favor: "))

y0 = float(raw_input("Ingresa el valor de y0, por favor: "))

h  = float(raw_input("Ingresa el valor de h, por favor: "))

k1=x0-y0

k2=(x0+(h/2))-(y0+(h/2)*(k1))

k3=(x0+(h/2))-(y0+(h/2)*(k2))

k4=(x0+(h))-(y0+(h)*(k3))

y1=y0+ (h/6)*(k1+2*k2+2*k3+k4)

print "El valor de y n+1 es:", y1
```

Al correr el programa e ingresar los valores que este pide en forma secuencial se van obteniendo los valores para continuar con la iteración tal como se muestra en la tabla siguiente. Se debe de observar que el valor de h permanece constante a lo largo de la solución de todo el problema.

x	y	h
0.0	2.00000	0.2
0.2	1.65620	0.2
0.4	1.44722	0.2
0.6	1.31238	0.2
0.8	1.23823	0.2
1.0	1.21378	0.2

El anterior ejercicio fue resuelto por Antonio Nieves Francisco C. Domínguez en su libro métodos numéricos aplicados a la ingeniería 2da edición por medio del método Euler, página 462

¿Qué pasa si se modifica el valor de h?

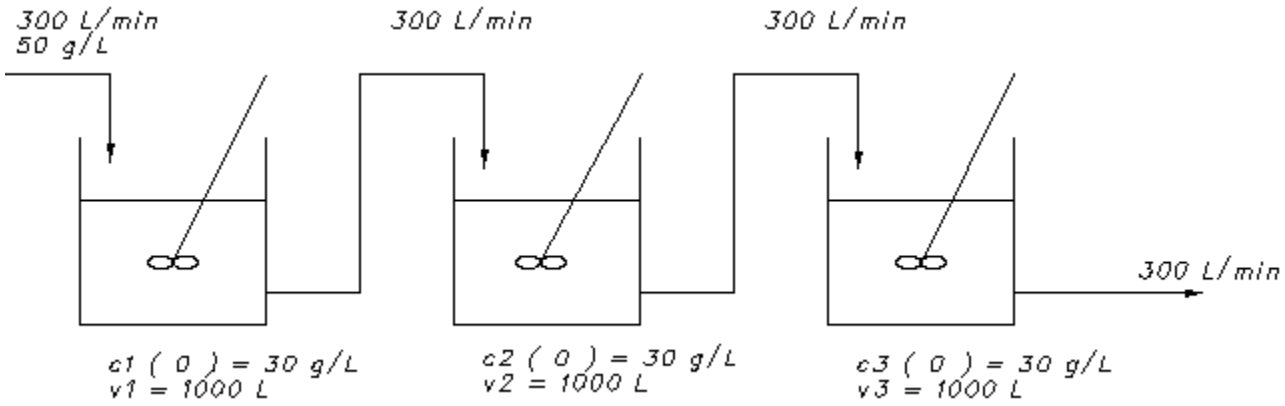
¿Cuál de los dos métodos es más preciso? compare ambos valores con el de la solución analítica.

¿Cómo se puede modificar el programa para que este continué hasta encontrar la solución final?

Tanques de mezclado

Uno de los principales problemas con los que se enfrenta un ingeniero químico es el diseño de tanques de mezclado, reactores químicos, etc. Como en el ejemplo que a continuación se muestra:

Se tienen 3 tanques de 1000 litros de capacidad cada uno, perfectamente agitados. Los tres recipientes están completamente llenos con una solución cuya concentración es de 30 g/L. A partir de cierto momento se alimenta al primer tanque una solución que contiene 50 g/L con un gasto de 300 L/min (Hay un arreglo entre los tres recipientes tal que al haber un gasto al primero , la misma cantidad fluye de éste al segundo, del segundo al tercero y de éste afuera del sistema, con lo cual se mantiene constante el volumen en todos ellos).



Calcule la concentración en cada tanque después de 10 minutos de haber empezado a agregar solución al primero.

Para lograr encontrar la solución a este problema, se debe de hacer un balance de soluto en el primer tanque

Acumulación = Entrada – salida

$$\frac{dC_1V_1}{dt}=300*(50)-300C_1$$

como V1 = 1000 L y permanece constante

$$\frac{dC_1}{dt}=15-0.3C_1\ y\ C_1(0)=30$$

Balance de soluto en el segundo tanque

$$V_2\frac{dC_2}{dt}=300*C_1-300C_2$$

como V2 = 1000 L y permanece constante

$$\frac{dC_2}{dt}=0.3*(C_1-C_2)\, y\, C_2(0)=30$$

Balance de soluto en el tercer tanque

$$V_3\frac{dC_3}{dt}=300*C_2-300\,C_3$$

como

v3=1000 L

$$\frac{dC_3}{dt}=0.3(C_2-C_3)\, y\, C_3(0)=30$$

Las ecuaciones 1 a 3, con sus respectivas condiciones iniciales, constituyen un sistema cuya solución representa los valores buscados, esto es:

$$PVI\left\{\begin{array}{c}\frac{dC_1}{dt}=15-0.3\,C_1\\ \frac{dC_2}{dt}0.3(C_1-C_2)\\ \frac{dC_3}{dt}=0.3\,(C_2-C_3)\\ C_1(0)=30\\ C_2(0)=30\\ C_3(0)=30\\ C_1(10)=?\\ C_2(10)=?\\ C_3(10)=?\end{array}\right.$$

Para resolver este sistema de ecuaciones diferenciales se puede usar el modelo de Runge-Kutta. Una gran dificultad muy frecuente para el alumno con este tipo de problemas es que al depender de una sola variable no pueden observar que el modelo se reduce de la siguiente manera:

$$K_1=f\,(y_i)$$

$$K_2=f\,(y_i+\frac{hK_1}{2})$$

$$K_3=f\,(y_i+\frac{hK_2}{2})$$

$$K_4 = f(y_i + h K_3)$$

aunque se debe de notar que nuestra formula original sigue siendo la misma.

$$y_{i+1} = y_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

Al ser una serie de constantes que dependen una de otra se proceden a resolver en cascada las ecuaciones; es decir, la solución del primero generara la del segundo y así en consecuencia cuando se logren obtener los primeros valores se vuelve a repetir el ciclo.

Este tipo de métodos numéricos no son exactos pero a cambio son más rápidos de hacer y tienen una gran precisión, lo que los vuelve muy útiles para el diseño real de tanques de agitación.

Para la primera iteración:

para las k1

$$k_1 c_1 = 15 - 0.3 * c_1$$

$$k_1 c_2 = 0.3(c_1 - c_2)$$

$$k_1 c_3 = 0.3(c_2 - c_3)$$

para las k2

$$k_2 c_1 = 15 - 0.3(c_1 + \frac{h * k_1 c_1}{2})$$

$$k_2 c_2 = 0.3((c_1 + \frac{h * k_1 c_1}{2}) - (c_2 + \frac{h * k_1 c_2}{2}))$$

$$k_2 c_3 = 0.3((c_2 + \frac{h * k_1 c_2}{2}) - (c_3 + \frac{h * k_1 c_3}{2}))$$

para las k3

$$k_3 c_1 = 15 - 0.3(c_1 + \frac{h * k_2 c_1}{2})$$

$$k_3 c_2 = 0.3((c_1 + \frac{h * k_2 c_1}{2}) - (c_2 + \frac{h * k_2 c_2}{2}))$$

$$k_3 c_3 = 0.3((c_2 + \frac{h * k_2 c_2}{2}) - (c_3 + \frac{h * k_2 c_3}{2}))$$

para las k4

$$k_4 c_1 = 15 - 0.3(c_1 + h * k_3 c_1)$$

$$k_4 c_2 = 0.3((c_1 + h * k_3 c_1) - (c_2 + h * k_3 c_2))$$

$$k_4 c_3 = 0.3((c_2 + h * k_3 c_2) - (c_3 + h * k_3 c_3))$$

Las nuevas concentraciones a buscar se calculan de la siguiente manera:

$$y_1 = c_1 + \frac{h}{6}(K_1 c_1 + 2K_2 c_1 + 2K_3 c_1 + K_4 c_1)$$

$$y_2 = c_2 + \frac{h}{6}(K_1 c_2 + 2K_2 c_2 + 2K_3 c_2 + K_4 c_2)$$

$$y_3 = c_3 + \frac{h}{6}(K_1 c_3 + 2K_2 c_3 + 2K_3 c_3 + K_4 c_3)$$

Con ayuda del programa de la página siguiente el cual hace uso del algoritmo de RK-4 y utilizando un paso de integración de 1 min para el sistema de ecuaciones diferenciales propuestas; se obtienen los siguientes valores:

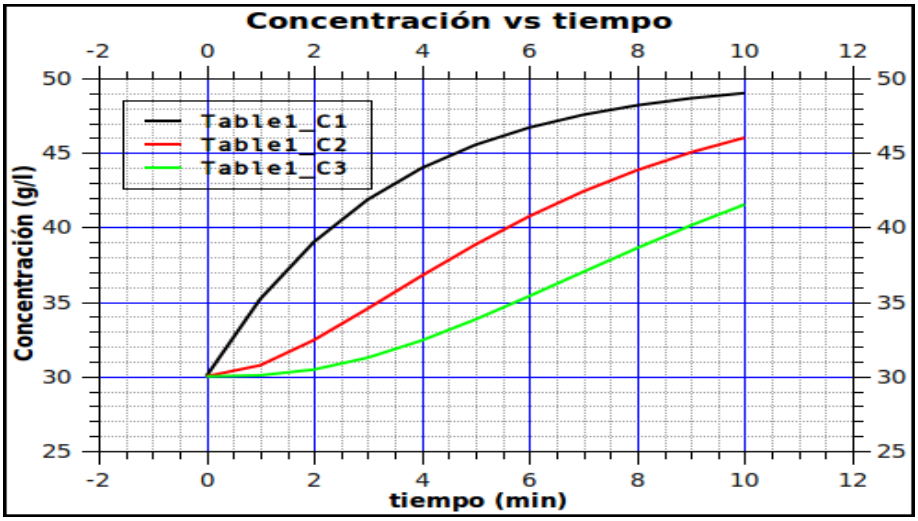
Tiempo (min)	C1	C2	C3
0	30.0000	30.0000	30.0000
1	35.1833	30.7403	30.0698
2	39.0232	32.4402	30.4597
3	41.8679	34.5525	31.2549
4	43.9754	36.7493	32.4086
5	45.5367	38.8450	33.8220
6	46.6934	40.7444	35.3870
7	47.5503	42.4085	37.0071
8	48.1851	43.8317	38.6058
9	48.6554	45.0271	40.1276
10	49.0039	46.0172	41.5363

```
# Escrito por Alejandro Juarez Contreras
# UNAM Mexico
####################
# ¡Recuerde Python no acepta acentos ni siquiera dentro de los comentarios!
#
print "Este programa sirve para calcular por medio de Runge-Kutta"
print "las concentraciones de tres tanques perfectamente agitados"

c1 = float(raw_input("Ingresa el valor de c1, por favor: "))
c2 = float(raw_input("Ingresa el valor de c2, por favor: "))
c3 = float(raw_input("Ingresa el valor de c3, por favor: "))
h  = float(raw_input("Ingresa el valor de h, por favor: "))

#  Esta parte calcula las k1 de cada de las concentraciones
################

k1c1 = 15-(0.3*c1)

k1c2 = 0.3*(c1-c2)

k1c3 = 0.3*(c2-c3)

# Esta parte calcula las k2
#################

k2c1 = 15 - 0.3*( c1 + ((h*k1c1)/2))

k2c2 = 0.3* (( c1 +((h*k1c1))/2)-( c2 +((h*k1c2))/2))

k2c3 = 0.3*(( c2 +((h*k1c2))/2)-( c3 +((h*k1c3))/2))

# Esta parte calcula las k3
######################

k3c1 = 15 - 0.3*(c1 +((h*k2c1))/2)

k3c2 = 0.3*((c1 +((h*k2c1))/2)-(c2 +((h*k2c2))/2))

k3c3 = 0.3*((c2 +((h*k2c2))/2)-(c3 +((h*k2c3))/2))

# Esta parte calcula las k4
################################

k4c1  =  15  -  y = x^3 + 2x^2 + 10x - 20   0.3*(c1 +(h*k3c1))

k4c2 = 0.3*((c1+(h*k3c1))- (c2 +(h*k3c2)))

k4c3 = 0.3*((c2 +(h*k3c2))-(c3 +(h*k3c3)))
```

```
# Esta parte calcula las nuevas concentraciones
#######################################
y1 = c1 + (h/6)*(k1c1+2*k2c1+2*k3c1+k4c1)

y2 = c2 + (h/6)*(k1c2+2*k2c2+2*k3c2+k4c2)

y3 = c3 + (h/6)*(k1c3+2*k2c3+2*k3c3+k4c3)

# imprimir valores
########

print "La nueva concentracion de c1 es:", y1

print "La nueva concentracion de c2 es:", y2

print "La nueva concentracion de c3 es:", y3
```

Con ayuda del programa QtiPlot y los resultados anteriormente obtenidos se llega a la siguiente gráfica

Gráfica de las distintas concentraciones contra el tiempo



¿Cómo influye el cambio de concentraciones en cada uno de los tanques?

¿Qué pasaría si cambiamos h? ¿se podría llegar al mismo resultado?

¿cambiarían las tendencias de nuestras gráficas?

¿Cómo se podría mejorar el programa anterior?

¿Se Podría utilizar el anterior código para ser utilizado en otro problema similar?

El anterior ejercicio fue resuelto por Antonio Nieves Francisco C. Domínguez en su libro métodos numéricos aplicados a la ingeniería 2da edición por medio del método Euler, páginas 503-505.

Columnas de destilación

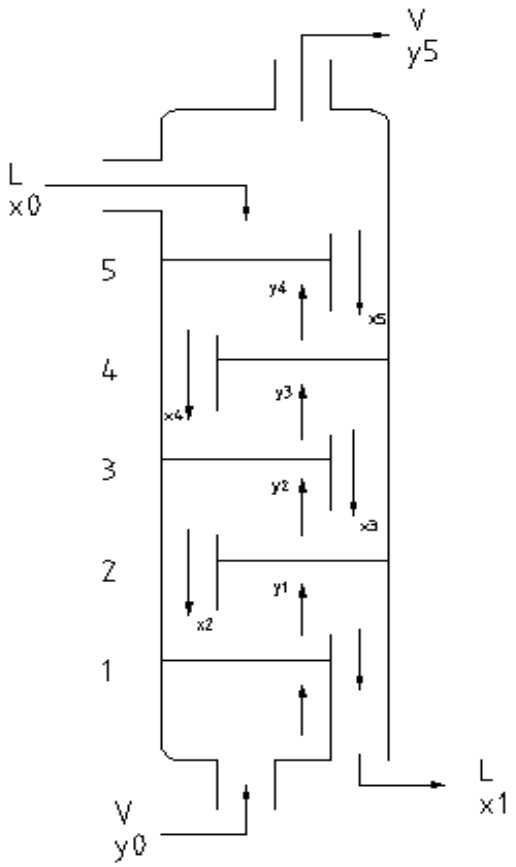
A través de la hoja de cálculo de OpenOffice se pueden resolver con facilidad problemas de ingeniería química como el siguiente:

En una columna de cinco platos, se requiere absorber benceno contenido en una corriente de gas V, con un aceite L que circula a contracorriente del gas. Considérese que el benceno transferido no altera substancialmente el numero de moles de V y L fluyendo a contracorriente, que la relación de equilibrio esta dada por la ley de Henry (y=mx) y que la columna opera a régimen permanente. Calcule la composición del benceno en cada plato

Datos:

V = 100 moles/min; L = 500 moles/min
y_0 = 0.09 fracción molar del benceno en V
x_0 = 0.00 fracción molar de benceno en L(esto se debe a que el aceite entra por el domo sin benceno)
m = 0.12

Para poder lograr resolver este problema primero se deben de realizar los balances de materia dentro de la torre para cada uno de los platos que ésta tiene. En la siguiente figura las flechas hacia abajo representan el líquido que desciende por la columna por acción del burbujeo que se presenta en cada uno de los platos, a su vez las flechas hacia arriba muestran el vapor más la corriente de benceno que se alimenta a la columna.



Estos balances se pueden obtener muy fácilmente al analizar la figura anterior y así para cada plato se tiene que:

Número de plato	Balance de benceno
5	L(x0-x5)+V(y4-y5)=0
4	L(x5-x4)+V(y3-y4)=0
3	L(x4-x3)+V(y2-y3)=0
2	L(x3-x2)+V(y1-y2)=0
1	L(x2-x1)+V(y0-y1)=0

Una vez logrados los balances, se puede llegar al siguiente sistema de ecuaciones lineales, previamente al introducir los valores de los datos conocidos.

$$512x5 - 12x4 = 0$$
$$500x5 – 512x4 + 12x3 = 0$$
$$500x4 – 512x3 +12x2 = 0$$
$$500x3 – 512x2 + 12x1 = 0$$
$$-500x2 + 512x1 = 9$$

el cual da como resultado la siguiente matriz aumentada

512	-12	0	0	0	0
500	-512	12	0	0	0
0	500	-512	12	0	0
0	0	500	-512	12	0
0	0	0	-500	512	9

Como se puede observar los valores de x serán las concentraciones de benceno correspondientes en cada plato de la torre. Para conocerlas se debe de resolver la matriz correspondiente. Lo cual se hace con ayuda de OpenOffice tal y como se describe a través de las siguientes figuras.

Se debe de saber que para encontrar el valor de un sistema de ecuaciones se puede hacer uso de la siguiente fórmula del álgebra matricial AX=b; de la cual se despeja X obteniéndose la fórmula requerida para estos propósitos $X=A^{-1}b$.

Para hacerlo con OpenOffice primero se abre una hoja en Calc y después se procede a escribir los valores dentro de la matriz a partir de la celda A1

	A	B	C	D	E
1	512	-12	0	0	0
2	500	-512	12	0	0
3	0	500	-512	12	0
4	0	0	500	-512	12
5	0	0	0	-500	512

La anterior figura es la matriz sin aumentar. La matriz aumentada; es decir b, se escribe a partir de G1 hasta la G5. Esto tiene el fin de que el lector se de cuenta más fácilmente de cada una de las operaciones correspondientes, con el tiempo este podrá realizar sus propias matrices de la forma que más le convenga.

G
0
0
0
0
9

Regresando a el problema se elige la celda A7 y en el cuadro de función se escribe =MINVERSA(A1:E5) para posteriormente apretar las teclas ctrl + shift y por último al dar Enter obtener la inversa de la matriz

7	0.002000000	-0.000048000	-0.000001152	-0.000000028	0.000000001
8	0.001999999	-0.002047999	-0.000049151	-0.000001179	0.000000028
9	0.001999972	-0.002047972	-0.002049124	-0.000049151	0.000001152
10	0.001998848	-0.002046820	-0.002047972	-0.002047999	0.000048000
11	0.001952000	-0.001998848	-0.001999972	-0.001999999	0.002000000

Una vez que se encontró la matriz inversa; se la multiplica por la matriz aumentada (matriz b), para lo cual se escoge la celda G7 y simplemente al escribir en la barra de funciones lo siguiente =MMULT(A7:E11,G1:G5) después oprimiendo ctrl+shift y luego Enter, se llega a los valores correspondientes a cada una de las variables y por ende el de la composición del benceno en el líquido de cada plato de la torre.

X5 =	0.000000006
X4 =	0.000000249
X3 =	0.000010368
X2 =	0.000432000
X1 =	0.018000000

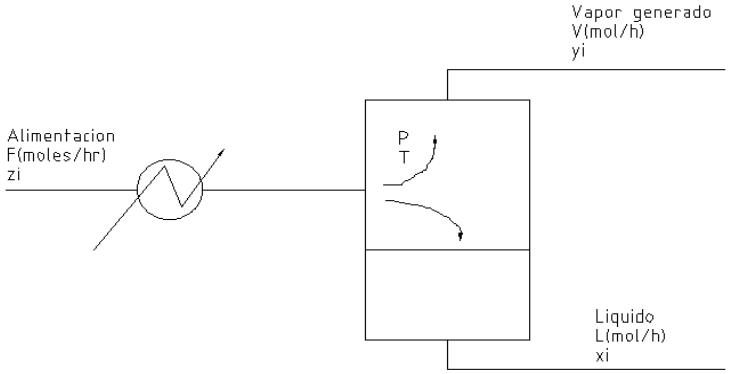
No se debe de preocupar por no conocer cada uno de los comandos de los que dispone OpenOffice para el manejo de matrices. Para salvar este pequeño problema simplemente se elige al ***Asistente de función*** y luego en ***Categoría,*** se da clic en ***Matriz***; se leen con cuidado las indicaciones y se experimenta con cada una de ellos. OpenOffice puede llegar a ser muy intuitivo después de haberse realizado todos los anteriores problemas.

IMPORTANTE: Recuerde que sólo con la práctica continua lograra mejorar sus propios proyectos

Flash adiabático

A continuación se calcula un flash adiabático con ayuda de la hoja de cálculo para esto se muestra un método semi-manual; este mismo método sirve para el diseño correcto de otros problemas de complejidad semejante y aún mayor.

Determine la cantidad de vapor V (moles/hr) y la cantidad de liquido L (moles/hr) que se generan en una vaporización instantánea continua a una presión de 1600 psia y una temperatura de 120 °F de la siguiente mezcla



componente	Composición z_i	$K_i=y_i/X_i$
CO_2	0.0046	1.65
CH_4	0.8345	1.80
C_2H_6	0.0381	0.94
C_3H_8	0.0163	0.55
i-C_4H_{10}	0.0050	0.40
n-C_4H_{10}	0.0074	0.38
C_5H_{12}	0.0287	0.22
C_6H_{14}	0.2200	0.14
C_7H_{16}	0.0434	0.09

Como se puede observar a partir de la figura que antecede se puede lograr un balance total de materia a partir de F = L + V lo cual conduce a que el balance de materia por componente sea:

$Fzi = Lxi + Vyi \quad i= 1, 2, ..., n$

Las relaciones de equilibrio líquido-vapor establecen que:

$Ki = \frac{yi}{xi}$ en donde i = 1,2, ..., n

al despejar $yi = Ki\,xi$ e introducirla en la ecuación de balance por componente se obtiene:

$$Fzi = Lxi + Vkixi \text{ ó bien } Fzi = xi(L+VKi)$$

y al despejar $xi = \frac{Fzi}{L+VKi}$; al retomar de la ecuación del balance total se puede obtener L=F-V y al substituir nuevamente $xi = \frac{Fzi}{F-V+VKi}$ y al simplificar da: $xi = \frac{Fzi}{F+V(Ki-1)}$

Por otra parte se sabe que la suma de las fracciones molares en la mezcla arroja las siguientes restricciones $\sum_{i=1}^{n} xi = 1$; $\sum_{i=1}^{n} yi = 1$ por lo que se puede escribir $\sum_{i=1}^{n} yi - \sum_{i=1}^{n} xi = 0$ ó simplemente

$\sum_{i=1}^{n} Kixi - \sum_{i=1}^{n} xi = 0$ que simplificado da $\sum_{i=1}^{n} xi(Ki-1) = 0$ al sustituir $xi = \frac{Fzi}{F+V(Ki-1)}$ llegando a la siguiente ecuación $\sum_{i=1}^{n} \frac{Fzi(Ki-1)}{F+V(Ki-1)} = 0$

Se puede observar que el valor de V que satisface la ecuación esta comprendido en el intervalo $0 \le V \le F$, lo que dificulta el lograr una estimación inicial de V, ya que el valor de F puede ser muy grande. Para disminuir este problema se normaliza el valor de V; lo que se puede lograr al dividir el numerador y el denominador entre F de la última ecuación:

$f(\Psi) = \sum_{i=1}^{n} \frac{zi(Ki-1)}{1+\Psi(Ki-1)} = 0$ en donde $\Psi = \frac{V}{F}$

Como se puede observar esta ecuación ésta expresada en términos de una nueva variable Ψ cuyos límites son: $0 \le \Psi \le 1$ (es decir se a normalizado el valor de la misma); la ecuación es no lineal en una variable, por lo que puede llegar a ser resuelta por medio del método de Newton-Rapshon

$f'(\Psi) = \sum_{i=1}^{n} \frac{-zi(Ki-1)}{[1+\Psi(Ki-1)]^2} = 0$ recuerde que el método de Newton-Rapshon dice que:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

y como en este caso se utiliza la variable Ψ en vez de la variable x se puede redefinir como:

$$\Psi_1 = \Psi_0 - \frac{f(\Psi_0)}{f'(\Psi_0)}$$

Con toda la anterior explicación es muy fácil comenzar a abordar el problema, tal como se puede ver a continuación. En caso de no haberse entendido bien alguna parte vuelva a revisarse y a analizar muy bien antes de continuar.

Para poder lograr que la hoja itere de una forma eficiente; se tiene que dividir antes en varias partes el problema, tal como se muestra en la siguiente tabla:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2	**componente**	**Z**	**Ki**		**Z*(K-1)**	**1+Ψ*(K-1)**	**F (Ψ)**		**-Z*(K-1)²**	**(1+Ψ*(K-1))²**	**F '(Ψ)**			
3	CO2	0.0046	1.65		0.002990	1.578082732	0.00189470421		-0.0019435	2.490345108563	-0.000780414		**Ψ**	**nueva Ψ**
4	CH4	0.8345	1.80		0.667600	1.711486439	0.39007028318		-0.53408	2.929185831566	-0.182330528		0.889358049	0.889358049
5	C2H6	0.0381	0.94		-0.002286	0.946638517	-0.0024148605		-0.00013716	0.896124481982	-0.000153059			
6	C3H8	0.0163	0.55		-0.007335	0.599788878	-0.0122293031		-0.00330075	0.359746698113	-0.009175206			
7	i-C4H10	0.0050	0.40		-0.003000	0.466385171	-0.0064324515		-0.0018	0.217515127356	-0.008275287		**Flujo (moles/hr)**	
8	n-C4H10	0.0074	0.38		-0.004588	0.448598010	-0.0102274194		-0.00284456	0.201240174235	-0.014135150		1	
9	C5H12	0.0287	0.22		-0.022386	0.306300722	-0.0730850384		-0.01746108	0.093820132163	-0.186112294			
10	C6H14	0.0220	0.14		-0.018920	0.235152078	-0.0804585704		-0.0162712	0.055296499722	-0.294253707			
11	C7H16	0.0434	0.09		-0.039494	0.190684175	-0.2071173442		-0.03593954	0.036360454752	-0.988423831			
12						**total**	-0.0000000003			**total**	-1.683639475			
13														
14					**xi**	**yi**								
15					0.002914930	0.004809634								
16					0.487587854	0.877658137								
17					0.040247676	0.037832815								
18					0.027176229	0.014946926								
19					0.010720753	0.004288301								
20					0.016495838	0.006268418								
21					0.093698767	0.020613729								
22					0.093556477	0.013097907								
23					0.227601477	0.020484133								

Para hacer que esta hoja de cálculo funcione se tendrá que escribir en las celdas correspondientes lo siguiente:

Celda	Significado	Operaciones
E3	Z*(K-1)	=B3*(C3-1)
F3	1+ Ψ*(K-1)	=1+(M4*(C3-1))
G3	F(Ψ)	=E3/F3
I3	-Z*(K-1)²	=(-B3)*POTENCIA((C3-1),2)
J3	(1+Ψ*(K-1))²	=POTENCIA((1+(M4*(C3-1))),2)
K3	F'(Ψ)	=I3/J3
N4	nueva Ψ	=M4-(G12/K12)
E15	xi	=(M8*B3)/(M8+N4*(C3-1))
F15	yi	=C3*E15
G12	suma para la Ψ	=SUMA(G3:G11)
K12	suma la nueva Ψ	=SUMA(K3:K11)

Las siguientes celdas por el momento pueden tener un valor cualquiera entre 0 y 1 debido a la normalización hecha anteriormente (aunque se sugiere se les ponga un cero por el momento), pero al encontrar los valores de resolución del problema se deberían de obtener los valores que se encuentran en la anterior figura.

La celda M4 servirá para introducir el valor inicial con el que se comience a iterar el nuevo valor de Ψ.

La celda M8 servirá para introducir el valor del flujo en moles/hr que se quiera estudiar, o bien con el fin de experimentar en forma de prueba y error nuevas posibilidades para este problema en particular.

Cabe aclarar que los títulos de cada una de las columnas correspondientes el usuario debe de determinarlos de forma que le recuerden su significado, ahora bien si se ha seguido fielmente cada uno de los pasos anteriores la hoja habrá quedado lista para trabajar en ella.

Al ir escribiendo en cada una de las celdas correspondientes las fórmulas y los valores necesarios se habrá dado cuenta que toda la hoja de cálculo dependerá de los valores que se ingresen dentro de las celdas M4 (Ψ) y N8 (Flujo).

Éste es justo el momento momento de proponer valores para iniciar con la primera iteración; un buen valor inicial para la celda M4 sería el de 0 (también se puede utilizar el de 0.5, quedando el hacerlo como un ejercicio posterior). En el caso de la celda M8 se propone el de 1mol/hr; justo cuando se observe que la hoja arroja valores coherentes y se halla comprobado que ésta correctamente escrita se podrán cambiar según convenga el caso.

Una vez que se de Enter a la hoja de cálculo con los valores correspondientes se obtendrá un nuevo valor de Ψ (celda N4) ; simplemente se toma este nuevo valor y se introduce en la celda M4 y se vuelve a obtener un nuevo valor de Ψ (celda N4), haciendo justamente lo que en el paso anterior; este proceso se repite hasta que los valores tengan la precisión requerida para nuestros fines.

Lo hermoso de este procedimiento es que converge en unos cuantos pasos, lo cual lo hace muy viable en el caso de que no se disponga de otra herramienta más que la de OpenOffice.

¿Se puede utilizar esta hoja de cálculo para hacer predicciones?

Ejemplos de funciones avanzadas de OpenOffice

Una de las aplicaciones más utilizadas por los ingenieros químicos son las hojas de cálculo, a continuación se muestran dos ejemplos acerca de sus más avanzadas aplicaciones.

Solver en OpenOffice

Debido a que por defecto OpenOffice sólo puede resolver ecuaciones de tipo lineal; se reducen mucho las posibilidades de utilizar esta valiosa herramienta. Para salvar este problema se tiene que obtener la extensión solver, muy valiosa en cuanto a sus aplicaciones a la ingeniería. Lo mejor de ésta es que se dispone de su código y por ende se puede estar seguro de que se trata de una opción seria y segura en cuanto a las necesidades técnicas y científicas.

A continuación se muestra como utilizar el entorno solver dentro de la hoja de cálculo de Openoffice primero se va a añadir esta aplicación a través del siguiente hipervínculo Solver for Nonlinear Programming [BETA] el cual tiene una licencia de tipo opensource (GNU Lesser General Public License) by Sun Microsystems, Inc. Una vez descargado este se va a añadir al conjunto de extensiones de OpenOffice en la barra de menú en la opción de herramientas se escoge al ***Administrador de extensiones***.

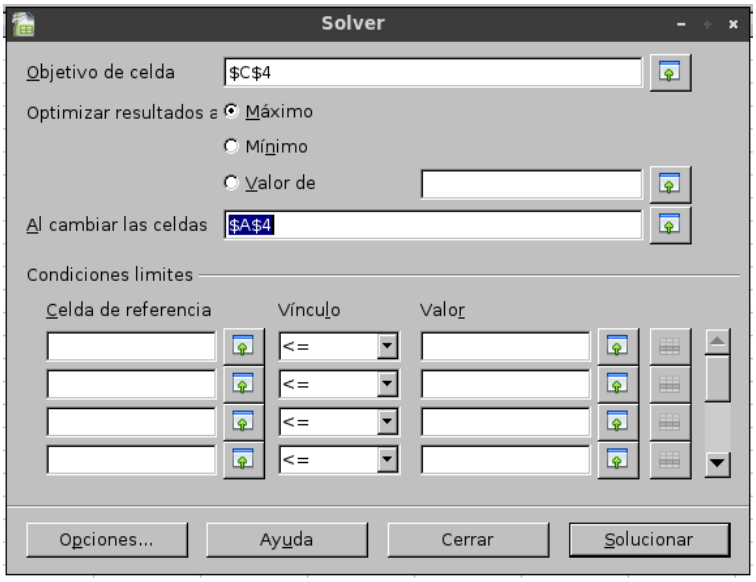
Ya que se tiene la ventana correspondiente se escoge la opción añadir y se busca el archivo NPLSolver.oxt dentro de la carpeta en que lo hemos descargado, lo seleccionamos y entonces se da clic en la palabra abrir, se espera a que cargue la extensión correspondiente dentro de OpenOffice lo que da nuevas posibilidades a ésta ya poderosa Hoja de cálculo, con esto ya tiene la posibilidad de hacer cálculos como el del siguiente ejemplo:

Primero se escogemos la celda A4 a la que se le asigna el valor de 7; luego a la B4 el de 100 y posteriormente en la celda C4 se introduce la siguiente formula =A4*POTENCIA(B4-2*A4,2); la cual arrojara un valor de 51 772 (si se ha hecho de forma correcta).

Posteriormente se elige la opción solver que viene integrada dentro del menú herramientas; se escoge como celda objetivo a C4 y dentro de la celda ***Al cambiar las celdas*** se elige a la A4.

Otra vez se vuelve a repetir la misma indicación de siempre, repita en su propia computadora o en la de la escuela todos y cada uno de los ejemplos que le interesen hasta que obtenga su correcta comprensión. Sólo con la práctica se logrará tener la certeza de que sus programas tendrán la suficiente coherencia y que además de ser muy útiles para usted, estos tendrán cada vez menos errores en la sintaxis de los mismos.

Como se puede observar en la siguiente figura el entorno solver ofrece la posibilidad de utilizar las opciones de optimización de resultados para encontrar ya sea el valor máximo o el mínimo.



Una vez hecho todo lo anterior se da clic en el botón solucionar con lo cual se obtienen los siguientes datos y resultados:

Los valores de estas celdas habrán cambiado como sigue:

para la celda A4 se obtiene: 16.6666681208736

para la celda C4 se tiene: 74074.07407

Programación en Calc con OpenOffice

OpenOffice permite hacer uso de un entorno de programación basado en el lenguaje Basic lo cual lo vuelve un entorno aún más poderoso al momento de diseñar nuevas aplicaciones para su uso en cálculos dentro de la ingeniería química. En el ejemplo siguiente se dará una pequeña muestra a través de un ejemplo muy sencillo.

Con el programa siguiente se puede encontrar a través del método de Newton-Rapshon una raíz del polinomio $x^3+2x^2+10x-20$

La instrucción REM en BASIC define un comentario y por tanto todo lo que se escriba a la derecha de la misma no será tomado en cuenta para la ejecución del mismo.

```
REM  *****  BASIC  *****

Sub iteracion                              REM define el nombre del programa

dim x as single                            REM define el nombre y el tipo de variable

x = inputbox(" desde donde comienzo ")     REM muestra una caja de dialogo en pantalla

for cuenta = 1 to 5                        REM esto sirve para 5 pasos

y= x^3+2*x^2+10*x-20
y1= 3*x^2+ 4*x+ 10

x=x - (y/ y1)                              REM método de Newton-Rapshon

MsgBox Str (x), 32, "respuesta"            REM muestra una caja con la respuesta

Next                                       REM repite el proceso de iteración

End Sub                                    REM termina con el programa
```

Este programa es semimanual, pero puede ser mejorado a través del uso de más comandos de los cuales dispone OpenOffice, pero no serán tratados en ésta tesis.

Todos los programas escritos en esta tesis se apegan a la licencia de tipo GPL3, con la esperanza de que se sigan distribuyendo en forma libre, y no dejen de serlo en un futuro. Misma que se anexa al final de este documento.

Alejandro Juárez Contreras 2010-2011 aleph.juarez@gmail.com COPYLEFT

Conclusiones

El uso de herramientas de software libre supone muchas más ventajas sobre el uso de las que son de tipo privativo como:

1) La certeza de que al estudiar el código se conoce con lujo de detalles la posibilidad de inserción de código malicioso, espionaje, robo de datos, etc. Por parte de quien vende el software. Con respecto a este punto siempre es indispensable el obtener los programas de fuente fidedignas y con prestigio dentro de los movimientos de software libre, como DEBIAN, Fedora, Ubuntu, etc. Debido a que se encuentra toda una comunidad atrás de ellos que cuidan de la integridad del código que se distribuye a través de la red mundial.

2) La posibilidad de crear nuevas fuentes de investigación, empleo, estudio, etc. Como ramas de la ingeniería química al involucrarse con el desarrollo de nuevo software y hardware.

3) Ofrece la posibilidad de que el país se encuentre a la vanguardia de los avances tecnológicos como en otros países del mundo.

4) Es muy importante no tener una dependencia total de un sólo proveedor de software y por ende la libertad de ser capaces de crear nuestras propias herramientas.

5) Puede ser la fuente de generación de entradas de capital monetario a la Universidad.

6) El costo de muchas licencias es muy elevado y se tienen que renovar en forma constante, situación contraria en el software libre. Con esto no se pretende decir que el software privativo sea malo, aún es necesario el uso de algunas tecnologías de este tipo.

7) Se espera que este trabajo sea de verdadera utilidad para los futuros ingenieros químicos, con la esperanza de que vean a un México cada vez más independiente y fortalecido por ellos.

Ya que es imposible abarcar muchos aspectos importantes acerca del uso de estas herramientas, queda como trabajo de quien quiera adentrarse aún más en este terreno.

Se espera que después de haber estudiado este material que para los nuevos principiantes en la filosofía del software libre debería de ser una meta la idea de cooperar y seguir aprendiendo, más allá de cualquier tipo de distribución que se utilice, y para los ingenieros químicos la de seguir aportando con nuevas ideas y proyectos a la sociedad y sobretodo a la UNAM.

Por otro lado existe mucho material de consulta, como para seguir escribiendo y sacando muchas más conclusiones, pero se espera que las anteriores líneas lograrán saltar la curiosidad de muchos logrando que quieran adentrarse un poco más a estas interesantes historias y porque no la de crear las suyas.

Como se pudo observar el uso de comentarios en los programas además de clarificar el contenido de los mismos ayudan a otros a aprender acerca de este tema y a que puedan ser corregidos o bien puedan ser mejorados.

Todo el código desarrollado a lo largo de esta tesis fue comprobado de tal manera que se puede asegurar que funcionara correctamente dentro de una distribución GNU/Linux cualquiera y por lo tanto es reutilizable en otros desarrollos. Cumpliendo con el objetivo doble de enseñar y dar herramientas útiles a los nuevos y antiguos ingenieros químicos. Logrando de esta manera que no se trate de una mera exposición de resultados si no más bien de una invitación a seguir desarrollando nuevos proyectos.

Es importante el notar como países como Francia se han beneficiado tremendamente al incluir en su patrimonio educativo y científico el uso de herramientas de software libre; pues empresas como Renault, Peugeot-Citroen, CEA Commissariat à l'Energie Atomique, CNES Centre National d'Etudes Spatiales, Dassault Aviation, EDF Electricit́ de France, Thales,...

Tal es el caso de Scilab, al cual además de brindarle apoyo económico para continuar con su desarrollo, hace que estas instituciones generen grandes cantidades de recursos que a su vez se traducen en verdaderas derramas de capital y desarrollo para su país. Brasil se ha visto muy beneficiado en el resurgimiento de su industria petrolera y esto también a contribuido en cierta medida a la que llegará a ser una nación poderosa dentro del mercosur.

Muchos países de América Latina están apostando por el empleo de estas tecnología y seguramente en algunos años más se podrán ver los resultados de estas medidas. No se debe de olvidar los casos de Japón y China que a través de mucho tiempo han resurgido como potencias mundiales debido a que han apostado por la educación y el desarrollo de tecnología.

En este momento que ha comenzado una carrera por la implementación de tecnologías libres y México no puede quedarse atrás de todos los demás países; o tristemente se tendrá que preparar para ver como otros países van generando y progresando en estos rubros y simplemente como en otros casos se tendrán que ver pasar las oportunidades de desarrollar tecnologías propias y perder la oportunidad de competir con otros.

Un país sin desarrollo tecnológico, es un país sin futuro para las nuevas generaciones de ciudadanos que habitan en él. Y estos mismos ciudadanos simplemente tendrán que comprar tecnología a quienes si se prepararon para ese momento bajo las condiciones les impongan estos.

El campo de aplicación, producción, diseño e implementación de este tipo de herramientas es tan amplio que sería necesario el extenderse durante mucho tiempo para abarcar aun más cosas, pero se pretende que con este primer acercamiento el lector quede enamorado de todas estas herramientas, su filosofía y su historia, queriendo ahondar mucho más en cada una de ellas.

Propuesta de plan de estudio para la materia optativa software libre en ingeniería química

Como una medida más y con la meta de contribuir a lograr todo lo anteriormente escrito se propone la creación de una asignatura optativa llamada “Implementación de software libre en ingeniería química”, con el fin de mejorar tanto el perfil académico, como el del campo laboral de los egresados de la carrera de ingeniería química de la Facultad de Química de la UNAM. Además de ir obteniendo por parte de los alumnos ayuda para la creación de una biblioteca de programas de software libre propios de la UNAM.

El estudio de esta materia quedaría repartido de la siguiente manera:

3 horas de teoría por 2 de práctica durante las 16 semanas que dura el semestre haciendo un total de 80 horas efectivas de estudio.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE QUÍMICA

PROPUESTA DE PROGRAMAS DE ESTUDIO
MATERIA OPTATIVA

ASIGNATURA	CICLO	ÁREA
software libre EN ingeniería química	OPTATIVAS	MATEMÁTICAS

HORAS/SEMANA

OPTATIVA	Clave SLIQ	Teoría 3hrs	PROBLEMAS 2hrs	CREDITOS 8

Tipo de asignatura	TEÓRICA
Modalidad de la asignatura	CURSO

ASIGNATURA PRECENDENTE cálculo de una variable, cálculo de varias variables, ecuaciones diferenciales,etc.

ASIGNATURA SUBSECUENTE

OBJETIVOS:

- Dotar de más herramientas al alumno para el desarrollo académico y laboral del mismo.
- Enseñar la importancia de crear herramientas de código abierto.
- Enseñar el uso de la programación en el área de ingeniería química.
- Resolver problemas con aplicaciones a la industria nacional.

UNIDADES TEMÁTICAS

NÚMERO DE HORAS POR UNIDAD	UNIDAD
15	1. Introducción al software libre 1.1 Historia y filosofía del software libre 1.2 Impacto social, económico académico y cultural en la sociedad mundial 1.3 Importancia de su aprovechamiento en México y la UNAM
15	2. Entornos gráficos y terminal 2.1 Introducción al uso del sistema operativo GNU/Linux 2.2 Uso de terminal 2.3 Manejo de ficheros, documentos y bases de seguridad para el usuario en GNU/Linux
15	3. Principios de programación y planificación de un programa 3.1 Diagramas de flujo 3.2 Algoritmos 3.3 Iteración 3.4 Variables 3.5 Tipos de errores 3.6 Introducción a Gfortran 3.7 Compilación y ejecución de un programa dentro del ambiente GNU/Linux

15	**4 Introducción a los lenguajes de programación** **4.1 bc** **4.1 Gpascal** **4.2 Python** **4.3 C** **4.5 Uso de hoja de cálculo libre**
20	**5 Aplicaciones** **5.1 Desarrollo de una aplicación con filosofía libre** **5.2 Proyecto final**

BIBLIOGRAFÍA BÁSICA

- ✔ **Tesis "Implementación de software libre en la ingeniería química" Juárez Contreras Alejandro. Fac. de química UNAM**
- ✔ **Linux Guía Práctica Sebastián Sánchez Prieto y Óscar García Población Editorial Alfaomega Ra-Ma 2008.**
- ✔ **Lenguajes de Diagramas de flujo Forsythe et al, Editorial Limusa séptima reimpresión 1983.**

BIBLIOGRAFÍA COMPLEMENTARIA

- McCracken D. D., Métodos numéricos y programación Fortran con aplicaciones en ciencias e ingeniería, editorial Limusa, primera reimpresión 1967.
- Cairó Battistutti Hall Osvaldo, Fundamentos de programación piensa en C, Pearson Prentice, primera edición, México 2006.
- Jan-Jaap van der Heijden, Peter Gerwinski, Frank Heckenbach, Berend de Boer, Dominik Freche, Eike Lange, Peter N Lewis, and others, The GNU Pascal Manual, Published by the Free Software Foundation 1988-2005.
- Los demás manuales y libros serán repartidos a lo largo del desarrollo del curso.

SUGERENCIAS DIDÁCTICAS

Emplear técnicas que promuevan la participación de los alumnos y propicien la formación de grupos de trabajo; exposición con preguntas, interrogatorio, discusión por grupos y resolución de problemas y ejercicios dirigidos por el profesor. Seminarios en grupos pequeños. Series y ejercicios para resolver en casa. Pequeños trabajos de investigación.

FORMA DE EVALUAR

Trabajos individuales y por equipos (se considera la presentación de los trabajos en tiempo, forma y contenido)
Apuntes, notas, series, trabajos de investigación.
Exámenes parciales
Exámenes ordinarios
Proyecto final

PERFIL PROFESIOGRÁFICO DE QUIENES PUEDEN IMPARTIR LA ASIGNATURA
matemáticos, actuarios, físicos, ingenieros, ingenieros químicos, químicos, ingenieros químicos metalurgistas.

Anexos

Método de aproximaciones sucesivas

(adaptado del libro Métodos Numéricos de Rodolfo Luthe et al, editorial Limusa, 1978)

Si se tiene una ecuación ya sea de tipo algebraica o trascendente; tal que ésta sea definida como:

$$F(x)=0$$

y además se dice que dado un valor ***x=a*** este la verifica idénticamente, entonces este valor ***x*** se tratara de una raíz de la misma.

$$F(a)=0$$

Si se suma un valor ***x*** en ambos extremos de la función se obtiene:

$$F(x)+x=x+f(x)$$

y dado que ***F(x)=0*** se tiene que:

$$x=f(x)$$

y dado que a es una raíz de la misma se puede reescribir de la siguiente manera:

$$a=f(a)$$

Entonces se puede comenzar a definir el método de aproximaciones sucesivas; debido a que se substituye ***a*** por ***x0***, en donde ***x0*** es un valor aproximado de la raíz, con lo que se obtiene

$$x_1=f(x_0)$$

en donde ***x1*** es un valor aproximado de la raíz buscada y si se procede con este nuevo valor se tiene

$$x_2=f(x_1)$$

Si se continua de esta forma un número reiterado de veces se obtiene lo siguiente:

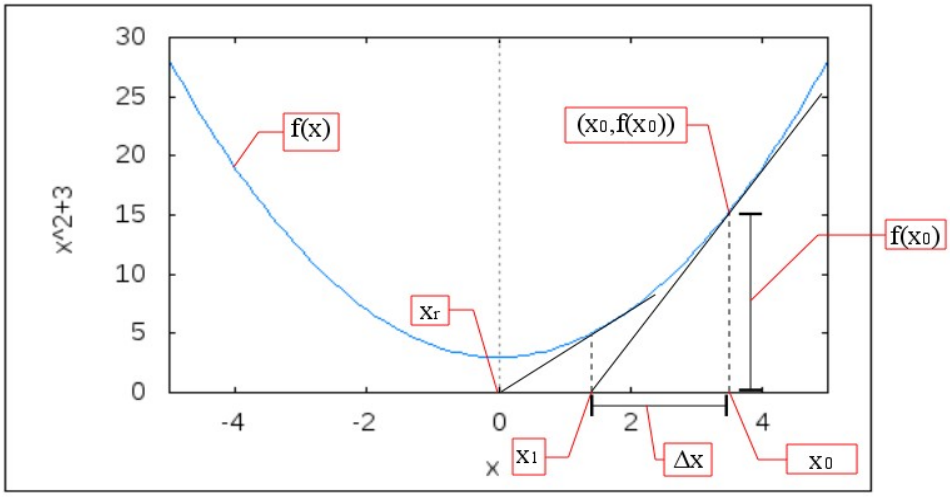
$$x_n=f(x_{n-1})$$

En la medida de que el número de iteraciones es mayor si el valor de x_n tiende a ser el de nuestra raíz ***a***, se dice que entonces que el método converge, en el caso contrario se puede decir que este se vuelve divergente y entonces se tiene que volver a proponer un nuevo valor como punto de partida.

Método de Newton-Rapshon

(adaptado del libro métodos numéricos aplicados a la ingeniería, 2da edición, Antonio Nieves y Francisco C. Domínguez)

Este método puede ser fácilmente analizado con ayuda de una gráfica como la que sigue:



Se puede observar por medio de la gráfica que: $x_1 = x_0 - \Delta x$; y debido a que pendiente de la tangente a la curva en un punto (x_0, $f(x_0)$) está dada por:

$$f'(x_0) = \frac{f(x_0)}{\Delta x}$$

así se tiene que:

$$\Delta x = \frac{f(x_0)}{f'(x_0)}$$

que al ser substituida en la primera ecuación da como resultado:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

y al convertir esta formula a una forma general se obtiene:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = g(x_i)$$

la cual es la formula recursiva del método Newton-Rapshon. Además por medio de la gráfica se puede apreciar fácilmente que la raíz cumple la condición de $g(x_r) = 0$

Palabras reservadas

Son las palabras utilizadas por el lenguaje de programación y que son exclusivas de este sin poderse usar como nombre de variable. En esta sección se encontrara una serie de listas con las palabras reservadas para cada uno de los programas tratados en esta tesis. Se debe de tener mucho cuidado antes de tomarlas en cuenta para evitar problemas posteriores al momento de programar.

Se debe de recordar siempre que éstas palabras no deberán ser tomadas nunca como el nombre de alguna función creada por el usuario y mucho menos como el nombre de algún programa.

Además de esto se presentan algunas indicaciones importantes que deberán de tomar muy en cuenta si no quiere perderse horas o días por causa de una mala sintaxis. Por otra parte teniendo en cuenta que si bien cada uno de los distintos entornos de programación tienen sus propias palabras estas podrán ser fácilmente ubicadas en cualquiera de los manuales que pueden ser descargados de la red y que no es el fin de esta tesis profundizar en cada uno de ellos. Sin embargo los siguientes apartados hacen referencia a los lenguajes de mayor uso en entornos educativos y por ende de necesidad básica para los estudiantes. Queda hacer una búsqueda mayor a partir de la bibliografía recomendada por parte del estudiante.

Por último haga uso de los siguientes espacios en blanco para hacer sus propias anotaciones.

Gfortran

Como regla general los compiladores Fortran permiten a los nombres de las variables ser de 6 caracteres como máximo.

Las variables de tipo integer no pueden comenzar con la letra I, J, K, L, M ó N.

Las variables de tipo real podrán comenzar con cualquier otra letra.

Las siguiente lista corresponde a las palabras reservadas para el programa:

allocate, allocatable , call, case, character, complex, contains, cycle, deallocate, default, dimension, do, end, else, elsewhere, exit, external , function, if, implicit, in, inout, integer, intent, interface, intrinsic , kind, logical, module , none, only, open, out, parameter, pointer, print, program, read, real, recursive, result, return , save, select, size, stat, stop, subroutine , target, then, type , unit, use , where, write

Gpc

Las palabras reservadas para Gpc son las siguientes:

absolute , and , array , asm , begin , case , const , constructor , destructor , div , do , downto , else , end , file , for, function , goto , if , implementation , in , inherited , inline , interface , label , mod , nil , not , object , of , on , operator , or , packed , procedure , program , record , reintroduce , repeat , self , set , shl shr , string , then , to , type , unit , until , uses , var , while , with , xor

Gcc

El lenguaje C estandar ANSI tiene 32 palabras reservadas que son las siguientes:

auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while.

Todas estas palabras deberán de ser escritas siempre en minúsculas debido a que Gcc no es de tipo sensitivo es decir no se reconocerá como palabra clave al escribirla en mayúsculas. Esto es muy importante debido a que nunca será lo mismo una variable con los siguientes nombres:

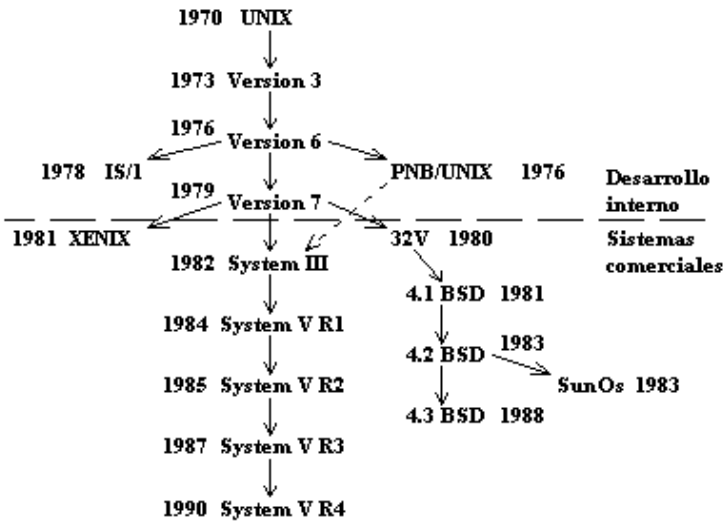
numero, Numero, NUMERO ó cualesquier otro tipo de combinación de letras mayúsculas y minúsculas para designar a ésta.

Tabla 1. Hechos importantes en la historia de UNIX.

Año	***Evento***	***Descripción***
1965	Origen	Bell Telephone Laboratories y General Electric Company intervienen en el proyecto MAC (del MIT) para desarrollar MULTICS.
1969-71	Infancia del UNIX	El primer UNIX llamado Versión 1 o Primera edición, nace de las cenizas de MULTICS.
1972-73	Nace el C	En la Versión 2 el soporte del lenguaje C y los pipes son añadidos. En la Versión 4 el ciclo se completa con la reescritura de UNIX en C.
1974-75	El momento	Las Versiones 5 y 6 de UNIX se distribuyen a las universidades. La Versión 6 circula en algunos ambientes comerciales y gubernamentales. AT&T impone ahora pagar una licencia, a pesar de que no puede promocionar UNIX por las duras regulaciones de EEUU del monopolio telefónico de AT&T.
1977	UNIX como producto	Interactive Systems es la primera compañía comercial que ofrece UNIX.
1977	Nace BSD	1BSD incluye un Shell Pascal, dispositivos y el editor ex.
1979	Versión 7	La Versión 7 de UNIX incluye el compilador completo K&R con uniones y definiciones de tipos. Versión 7 también añade el Bourne Shell.
1979	Trabajo en Red	BSD acrecentado por BBN incluye soporte para trabajar en red.
1979	Nace XENIX	Implementación para microcomputadoras ampliamente distribuido en hardware de bajo coste.
1980	Memoria Virtual	La capacidad de memoria virtual se añade en 4BSD.
1980	Nace ULTRIX	DEC realiza una versión de UNIX basado en BCD.
1980	Licencias en AT&T	La distribución de licencias abre el mercado
1982	Atracción de los negocios	Soporte importante para procesos de transacciones desde UNIX System Development Lab.
1983	Nace System V	La versión más común de AT&T obtiene sus bases.
1984	Salida de SVR3	AT&T desata la versión más popular de System V hasta ahora.
1988	Motif vs Open Look	Sistemas por ventanas rivales son anunciados por OSF y UI.
1988	Siguiente paso	Un UNIX gráfico usa el Kernel Mach.

1988	Motif vs Open Look	Sistemas por ventanas rivales son anunciados por OSF y UI.
1988	Siguiente paso	Un UNIX gráfico usa el Kernel Mach.
1990	OSF/1 vs SVR4	Versiones rivales de UNIX son anunciadas por OSF y UI.
1992-95	Socialización	OSF/1 abandona la escena; SVR4 se convierte en el estándar; Sun vende más estaciones de trabajo para usuarios de Motif que para usuarios de Open Windows; y crece Windows/NT de Microsoft

Historia gráfica del desarrollo UNIX



información obtenida del libro Manual de UNIX Comandos y Administración del Sistema José Manuel Flomesta Abenza Juan José Gallardo Gila 1o ASI

Notas finales

El software que nace libre debería de ser siempre libre y la única forma de lograrlo es a través de tomar la responsabilidad que cada individuo tiene para contribuir con lo necesario para que esta filosofía perdure.

La verdadera libertad conlleva siempre responsabilidades si no ésta sería libertinaje.

Esta tesis ha sido creada haciendo uso exclusivo de herramientas de software libre.

Como una reflexión final, no se debería de perder la oportunidad de seguir creando y desarrollando software libre, haciendo uso del derecho que todo ser humano tiene a ser informado, instruido y apoyado, no tan solo en esta área, si no en cualquier otra a través de su vida.

Todos los programas presentados en esta tesis fueron probados, para asegurarse de que corrieran en forma aceptable, queda de antemano con libertad de estudiarlos, mejorarlos y redistribuirlos.

Aunque se trata de un trabajo serio, el principal objetivo siempre fue el de hacer más amena la tarea de adentrarse a la programación, al desarrollarse con el mayor cuidado cada uno de los temas que se han tratado. Es cierto que se pudo haber cometido algún error u omisión pero se pretende que el sentimiento de compartir con alguien más el conocimiento sea mas que suficiente para que se pueda mejorar este trabajo a través de la colaboración de la comunidad.

Para poder entender el significado de 3,1415926535 en la dedicatoria inicial, la siguiente regla mnemotécnica satisface a: *"Sol y luna y cielo proclaman al Divino Autor del Cosmo"*, en la que el número de letras de cada palabra representa la secuencia ordenada de los primeros décimales del número pi; ahora bien por otro lado la fórmula matemática $\rho=a(1+\cos\theta)$ es la representación de una cardiode.

Ya por ultimo me gustaría que usted sienta la suficiente confianza de compartir conmigo y con los demás sus experiencias acerca de este material y por que no de involucrarse en el movimiento del software libre.

El ser hacker es todo un estilo de vida como lo describe en su libro Erick Raymond en su libro la Catedral y el bazar

Bibliografía

GNU/Linux y UNIX

Strobel Stefan y Uhl Thomas , Linux Universe, editorial Springer-Verlag. año 1995.

Sánchez Prieto Sebastián y García Población Óscar, Linux Guía Practica, editorial Alfaomega Ra-Ma, año 2008.

Stan Kelly-Bootle, Como usar UNIX sistema V, versión 4.0, Megabyte, grupo Noriega editores, SYBEX.

Pérez César, Domine Corel Linux, Alfaomega Ra-Ma, año 2001

Diagramas de flujo

Forsythe et al, Lenguajes de Diagramas de flujo, editorial Limusa, séptima reimpresión 1983.

gfortran

Luthe Rodolfo et al, Métodos Numéricos, editorial Limusa, 1978

J. San Fabián, Introducción a la programación Fortran, Informática Aplicada a la química, Departamento de química Física Aplicada, Universidad Autónoma de Madrid, Primera versión en Latex: 3 Nov 2004 revisión del 3 de noviembre de 2005 .

García de Jalón Javier et al, Aprenda Fortran 8.0 como si estuviera en primero, Escuela Técnica Superior de Ingenieros Industriales Universidad Politécnica de Madrid, Francisco de Asís de Ribera, Madrid julio 2004.

Miguel Alcubierre, Introducción a FORTRAN, Instituto de Ciencias Nucleares, UNAM, Abril 2005.

Forsythe, Keenan et al, Programación Fortran, editorial Limusa, tercera reimpresión 1981.

McCracken D. D., Métodos numéricos y programación Fortran con aplicaciones en ciencias e ingeniería, editorial Limusa, primera reimpresión 1967.

www.gfortran.org/

gpascal

Nell Dale, Chip Weems, Pascal segunda edición, editorial McGraw-Hill 1991

Jan-Jaap van der Heijden, Peter Gerwinski, Frank Heckenbach, Berend de Boer, Dominik Freche, Eike Lange, Peter N Lewis, and others, The GNU Pascal Manual, Published by the Free Software Foundation 1988-2005.

Van Canneyt Michaël, Reference guide for Free Pascal, version 2.4 Document version 2.4 , March 2010.

http://www.freepascal.org/

C

Cairó Battistutti Hall Osvaldo, Fundamentos de programación piensa en C, Pearson Prentice, primera edición, México 2006.

Bustamente Paul et al, Aprenda C++ básico como si estuviera en primero, colección Aprenda..., como si estuviera en primero TECNUM, Navarra, año 2004.

Rogan C. José, Muñoz G. Víctor, Apuntes de un curso de programación y métodos numéricos, cuarta edición, Departamento de Física, Facultad de Ciencias, Universidad de Chile.

Salas Ángel, Curso de lenguaje "C", Centro de cálculo de la Universidad de Zaragoza, 1991

http://gcc.gnu.org/

bc

http://linux.about.com/od/commands/l/blcmdl1_bc.htm

http://bulma.net/body.phtml?nIdNoticia=204

http://www.computerhope.com/unix/ubc.htm

http://www.go2linux.org/bc-linux-on-line-calculator-scientific-interactive-programmable

python

Gonzalez Duque Raúl, Python para todos, España, 2008.

Van Rossum Guido, Fred L. Drake, Jr. editor, Guía de aprendizaje de Python, Release 2.0, BeOpen PythonLabs, 16 de octubre de 2000

Maruch Stef y Maruch Aahz, Python for dummies, Wiley Publishing, Inc, QA76.73 p98 M38

www.python.org/

OpenOffice

Ayuda de OpenOffice.org F1

Barreras Alconchel Miguel, Matemáticas con Microsoft Excel, 2da edición, editorial Alfaomega Ra-Ma, año 2007

http://www.openoffice.org

libreOffice

www.documentfoundation.org.

Octave

Introducción a GNU Octave Versión 1.0, Unidad de Diseminación de software libre (http://nux.ula.ve) Corporación Parque Tecnológico de Mérida ,Universidad de Los Andes , Agosto 2007 .

W. Eaton , John, Bateman David, Hauberg Søren, GNU Octave A high-level interactive language for numerical computations Edition 3 for Octave version 2.9.17, July 2007 .

Guillem Borrell i Nogueras, Introducción informal a Matlab y Octave, http://iimyo.forja.rediris.es/ , 31 de octubre de 2008

García Rojo Juan José, Herramientas en GNU/Linux para estudiantes universitarios Gnu/Octave: cálculo numérico por ordenador, año 2003

www.gnu.org/software/octave

Maxima

Arsuaga Franco Miguel, Ramos Palanco Rosa Lic. CC. Matemáticas, Introducción a Maxima, Profesores del Departamento de Matemáticas del I.E.S., Almunia de Jerez de la Frontera. Año 2004.

Rodríguez Riotorto Mario,Primeros pasos en Maxima, 11 de marzo de 2008 .

Juan Pablo Romero Bernal y colaboradores del proyecto GLUD-CLog del Grupo Linux Universidad Distrital (Colombia), Manual de Maxima primera traducción, (http://glud.udistrital.edu.co/clog).

Manual de Maxima version 5.21, http://maxima.sourceforge.net/

http://maxima.sourceforge.net/

R

Dr. José A. García, Apuntes de Bioestadística, Laboratorio de Biología Teórica Posgrado e Investigación, Universidad La Salle, México, 31 de mayo de 2005

Fox John, Iniciación a R Commander*, 21 de abril de 2008

www.r-project.org/

Scilab

Scilab manual

Mora Escobar Héctor Manuel, Introducción a SCILAB, Departamento de Matemáticas, Universidad Nacional de Colombia, Bogotá, mayo del 2005

http://www.scilab.org/

Geany

http://www.geany.org/manual/current/geany.txt

http://exprimepc.blogspot.com/2010/11/geany-0191.html

Vídeos acerca de GNU/Linux

http://www.eduvlog.org/2007/09/linux-en-la-ciencia-y-la-investigacion.html

Videos Richard Stallman

http://vimeo.com/4152803

Página con muchísimos programas científicos para GNU/Linux

http://www2.uca.es/serv/softwarelibre-cientifico/

Noticias europeas relacionadas con el software y el hardware

http://www.internautas.tv

Noticias sobre linux ruso

http://www.neoteo.com/el-super-linux-ruso-14626.neo

Noticias sobre linux cubano

http://www.legox.com/software/linux-cubano/

Noticias sobre GNU/Linux

http://www.taringa.net/posts/linux/7752697/Importancia-de-Linux-en-el-mundo-de-hoy.html

Cuantos linuxeros somos

http://brujohacker.blogspot.com/2011/01/cuantos-linuxeros-somos-en-el-orbe.html

http://linuxadictos.com/2009/04/20/%C2%BFlos-linuxeros-somos-tantos-como-pensamos/

Distribuciones GNU/Linux

Aunque existen miles de las mismas las siguientes son muy recomendables para iniciarse en este mundo del software libre.

DEBIAN

www.debian.org

Ubuntu

www.ubuntu.com

Fedora

www.fedora.redhat.com

Linuxmint

http://linuxmint.com/

Distribuciones GNU/IllumOS

Illumos

http://www.illumos.org

Belenix

http://www.belenix.org

nexenta

http://www.nexenta.org

Schillix

http://www.schillix.org

Nota: En un principio estas hicieron uso de un kernel OpenSolaris pero con la aparición de GNU/IllumOS,

Revista Tuxinfo

Increíble revista de software libre de gratuita y de libre distribución, aunque hoy se encuentra peligrando su edición por falta de fondos. (¡Animo! Amigo Ariel Cogartelli y a todo el equipo)

www.tuxinfo.com.ar

Tipos de licencias

http://es.wikipedia.org/wiki/Software_libre

QCAD

Manual de referencia del usuario de qcad, http://www.qcad.org/qcad/manual_reference/es/

Clasificación de virus y otros peligros.

Informática para cursos de bachillerato Gonzalo Ferreyra Cortés editorial Alfaomega impreso en Colombia año 2000 pag. 140-149

http://es.wikipedia.org/wiki/Virus_inform%C3%A1tico

Misceláneos

La Santa Biblia, sociedades bíblicas unidas antigua versión de Casiodoro de Reina revisada por Cipriano de Valera, revisión de 1960

Huevo de chocolate una excelente página web sobre matemáticas para niños y no tan niños. http://www.elhuevodechocolate.com/mates.htm

Edward de Bono, Seis sombreros para pensar, Granica Ediciones

Glosario

Glosario de informática de Guido J. Pace año 1970–2003

exa.unne.edu.ar/depar/areas/informatica/.../GlosInform_N3_1.PDF

Imágenes

Free Software Sticker Book volumen I y II

La imagen de GNU Pascal fue tomada de la portada del libro oficial de este programa.

la mayor parte de las imágenes para esta tesis fueron tomadas con el programa de captura de pantalla scrot

Para el apéndice sobre el método de Newton-Rapshon se utilizó el programa Maxima y Scrot (Screen Shot), con retoques hechos en OpenOffice.

Las gráficas para los ejemplos de cálculos con OpenOffice y el problema de los tres tanques de mezclado perfecto fueron creados con Qcad un excelente programa tipo GPL para CAD de 2 dimensiones, aunque este programa esta traducido al español, éste por el momento no permite introducir acentos ni subíndices dentro de los letreros. Se espera que en un futuro no muy lejano se arreglen estos pequeños problemas.

Imagen del satélite de la página 102 http://es.wikipedia.org/wiki/Mars_Climate_Orbiter

Glosario

Bug: Término genérico que se utiliza para designar una falla en el hardware o error en algún programa de aplicación.

Encriptación: Sistema de codificación de archivos que valiéndose de algún programa especial y mediante una palabra clave personal, modifica totalmente el contenido del mismo volviéndolo indescifrable a la vista de todos. Para volver al archivo original será necesario realizar la operación inversa; es decir, desencriptarlo. Ésta última operación, solo puede ser realizada mediante el conocimiento de la palabra clave personal con que fue originalmente codificado.

Hardware: Su traducción literal al castellano es quincalla o chatarra informática. Cuando se habla de él, se hace referencia estrictamente a la parte física de la computadora y los periféricos.

Lenguaje: En informática, cuando se habla de lenguaje, se hace referencia generalmente a cualquier código de programación, que consiste en el conjunto de instrucciones permitidas y definidas por sus reglas sintácticas y su valor semántico, para la expresión de soluciones a problemas que las aplicaciones necesitan para que la computadora ejecute determinadas operaciones. Existen lenguajes de alto y bajo nivel, de tercera y cuarta generación, lenguajes naturales, lenguajes de máquina, etc.

RAM (Random Acces Memory): Memoria de acceso aleatorio. Nombre que recibe el medio de almacenamiento principal de la computadora. Posee un sistema de direccionamiento total, lo cual permite, tanto al microprocesador como a otros dispositivos, en virtud de la tecnología DMA, acceder a cualquier dirección de la misma. Es de lectura y escritura, pero necesita de energía para mantener la información almacenada en ella; es decir, cuando se apaga la computadora se borra todo lo que hay en ella. Existen de muchos tipos como SIMM, DIMM, RIMM, etc.

ROM (Read Only Memory): Memoria de sólo lectura. Controla las rutinas fundamentales para el funcionamiento de la computadora y que no pueden ser borradas por el usuario. Aunque se desconecte la máquina, estos datos nunca se pueden perder pues contienen las órdenes grabadas en los circuitos internos de los microprocesadores.

Sistema Operativo: Es el programa básico que constituye la plataforma lógica de la computadora e interactúa entre el microprocesador y los demás programas de aplicación, controladores, dispositivos, etc. Además permite gestionar el sistema, copiar y transferir datos, estructurar el disco, utilizar programas, optimizar la memoria, etc. Existen muchos Sistemas Operativos, como el DOS, Windows, Linux, UNIX, XENIX, OS/2, Solaris, McOS, AteOS, BeOs, VsTa, BSD, AIX y muchos otros.

Software: También conocido como “soporte lógico”, compendia todo tipo de programas, utilidades, aplicaciones, sistemas operativos, drivers o controladores que hacen posible que el usuario pueda trabajar con la computadora. El término ha sido adoptado y está totalmente integrado en el idioma español ya que, al igual que sucede con hardware, no ha habido nadie capaz de encontrar una traducción apta para de englobar el concepto en una sola palabra.

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-

purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or

warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and

only if you received the object code with such an offer, in accord with subsection 6b.

- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any

attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot

convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING

WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify
> it under the terms of the GNU General Public License as published by
> the Free Software Foundation, either version 3 of the License, or
> (at your option) any later version.
>
> This program is distributed in the hope that it will be useful,
> but WITHOUT ANY WARRANTY; without even the implied warranty of
> MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
> GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License
> along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

> <program> Copyright (C) <year> <name of author>

```
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.