23

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

CAMPUS ARAGÓN

SISTEMA AUTOMATICO DE CONSULTA DE HORARIOS VIA INTERNET PARA LA UNAM CAMPUS ARAGON

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACION

P R E S E N T A :

ERNESTO MANZO SALAZAR

ASESOR :

ING. VICTOR RAUL VELASCO VEGA

MÉXICO 2000

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ARAGÓN
DIRECCION

ERNESTO MANZO SALAZAR
P R E S E N T E .

En contestación a la solicitud de fecha 4 de julio del año en curso, relativa a la autorización que se le debe conceder para que el señor profesor, Ing. VICTOR RAUL VELASCO VEGA, pueda dirigirle el trabajo de Tesis denominado, "SISTEMA AUTOMATICO DE CONSULTA DE HORARIOS VIA INTERNET PARA LA U.N.A.M. CAMPUS ARAGON", con fundamento en el punto 6 y siguientes, del Reglamento para Exámenes Profesionales en esta Escuela, y toda vez que la documentación presentada por usted reúne los requisitos que establece el precitado Reglamento; me permito comunicarle que ha sido aprobada su solicitud.

Aprovecho la ocasión para reiterarle mi distinguida consideración.

ATENTAMENTE
"POR MI RAZA HABLARA EL ESPÍRITU"
San Juan de Aragón, México., 8 de julio de 1997
EL DIRECTOR

M en I CLAUDIO C. MERRIFIELD CASTRO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGÓN
DIRECCION

c c p Jefe de la Unidad Académica.
c c p Jefatura de Carrera de Ingeniería en Computación .
c c p Asesor de Tesis.

CCMC/AIR/unac.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MEXICO

San Juan de Aragón Edo. de México, 4 de Agosto de 1998

Lic. Alberto Ibarra Rosas.
Jefe de la Unidad Académica

Presente.

Informo a Usted que con esta fecha el alumno **ERNESTO MANZO SALAZAR** de la carrera de Ingeniería en Computación, con número de cuenta **8920572-7**, ha concluido con su trabajo de tesis titulado "**Sistema Automático de consulta de horarios vía Internet para la UNAM Campus Aragón**" por lo que solicito se le permita continuar con los trámites para su titulación.

Sin más por el momento me despido enviando a usted un cordial saludo.

Atentamente

Ing. Victor Raúl Velasco Vega.
Asesor

Vo.Bo.

Ing. Juan Gastaldi Pérez.
Jefe de carrera de
Ingeniería en Computación.

DEDICATORIAS Y AGRADECIMIENTOS

Albert Einstein dijo un día: "Es más importante la imaginación que el conocimiento". Imaginar grandes cosas es lo que propicia el deseo, la investigación, los inventos, las soluciones... lo que ahora me hace feliz es saber que algo que un día imaginé, hoy es realidad.

Agradezco a Dios por permitirme vivir en la época más interesante que puedo imaginar y por estar conmigo cuando más lo he necesitado.

A mis padres, María Elena Salazar Martínez y Reveriano Manzo Beltrán por la vida, el amor, el apoyo y la oportunidad de estudiar, pese a los sacrificios que tuvieron que hacer para ello. Gracias por todo.

A mis hermanos, Anabel e Itsmael por compartir conmigo buenos y malos momentos, por enseñarme que las cosas no son fáciles, sin embargo pueden hacerse, pero sobre todo por el ejemplo de perseverancia que me han dado.

A la Universidad Nacional Autónoma de México y sus profesores, por formarme en un hombre capaz de transformar la imaginación en realidades útiles para la sociedad. "Por mi raza hablará el espíritu"

Al amor de mi vida, Jeannette Domínguez Juárez, por impulsarme en todo momento a ser mejor, por la fuerza de tu carácter, pero sobre todo por el amor que me has demostrado.

A mi asesor y amigo, Ing. Víctor Raúl Velasco Vega por todo el apoyo que me ha brindado en la realización de este proyecto, por sus consejos y regaños, que siempre han sido para mejorar.

A Esther Leandro y Marisa Gallegos por brindarme su amistad incondicional y permitirme ser un apoyo en sus vidas.

A Cesar, Manuel, Arturo, Claudia, Edgardo. Edna, Julio, Roberto, José Luis, Liliana y Alejando, por su amistad y apoyo a lo largo de esta ingeniería.

A todos los que han sido mis compañeros de trabajo: Victor, Estéfana, Gustavo, Itsmael, Marcelo, Francisco, Adolfo, Gladis, Luis, Alberto, Marco, Juan y Eric, por formar parte de mi vida profesional y por haberme permitido aprender con ustedes lo que es el trabajo en equipo.

A todos aquellos que directa o indirectamente me han apoyado, GRACIAS.

Índice.

I

Introducción

Introducción.

La UNAM Campus Aragón cuenta con una de las poblaciones estudiantiles más grandes en lo que a número de alumnos se refiere. Esto obliga a que se piense en una forma eficiente de dar atención e información a los mismos.

La tecnología ha crecido ampliamente en los últimos años, por tal motivo, ha modificado la forma en que actualmente se accede a la información. El Campus Aragón de la Universidad cuenta desde 1995 con un servidor de páginas Web, el cual es administrado por el Departamento de Informática de la misma. El servidor permite tener un vínculo con otras universidades y empresas, pero sobre todo con los alumnos de la misma. Esto permite tener un intercambio de información que enriquece la comunicación con los estudiantes, ayudando a conocer sus necesidades e inquietudes.

Una de las inquietudes presentadas por ellos, es que debido al trabajo, no siempre era posible estar en contacto con la escuela para informarse, de trámites, servicios que se prestan, de los horarios de clases, etc., sin embargo tienen acceso a Internet.

Al parecer existe la necesidad de proporcionar al exterior la información necesaria vía Internet, ya que esta tecnología se ha vuelto popular y cotidiana.

El problema aquí es que no se cuenta con un sistema que proporcione dinámicamente la información que los estudiantes necesitan, para este caso, los horarios de clases, y es esto lo que este trabajo pretende resolver, considerando con ello el siguiente objetivo :

> "*Utilizar las técnicas de programación y tecnologías actualmente disponibles en el Campus Aragón para desarrollar un sistema de consulta de horarios que satisfaga las necesidades de información que se tienen a un nivel tanto institucional, como externo, de tal forma que exista un mayor aprovechamiento de los recursos informáticos*".

Particularmente con este trabajo se desean cumplir los siguientes puntos :

1. Proporcionar un sistema de consulta de horarios que sea útil para la comunidad universitaria.
2. Determinar la forma de interactuar con el usuario que consulte el sistema.
3. Conocer la forma en la que se puede exportar una base de datos con los horarios a un ambiente de hipertexto como lo es una página Web.
4. Conocer el funcionamiento de una Interfaz Común de Puerta de Enlace (Common Gateway Interface) o CGI.
5. Proporcionar los conocimientos básicos de funcionamiento del lenguaje de hipertexto HTML.

A lo largo de este trabajo se obtendrán todos los conocimientos necesarios para que finalmente pueda realizarse un sistema completo.

En el capítulo I se analizará de forma general la manera en la que se hará el sistema.

El capítulo II es básicamente un manual de funcionamiento de Cronos, esto con el fin de conocer un poco la forma en la que se capturan los datos que posteriormente serán utilizados por el sistema. Al final de éste se presentan todos los archivos que intervienen en los módulos requeridos, así como la estructura de los mismos.

El capítulo III explica un poco la forma en la que funciona un sitio Web, dando a conocer también algunas de las cosas con las que cuenta el servidor del Campus Aragón.

La selección de un lenguaje de programación adecuado para realizar el sistema es analizado en el capítulo IV.

Con el capítulo V se analizan las formas en las que se puede recibir información de un usuario de Internet a través de la Interfaz Común de Puerta de Entrada (Common Gateway Interfase).

El capítulo VI muestra la forma en la que se construye una página Web utilizando HTML (HiperText Marckup Languaje - Lenguaje de Marcado de HiperTexto) y con él crear hojas de respuesta dinámica que puedan ser consultadas por un navegador.

El sistema completo es presentado en el capítulo VII y el capítulo VIII es el encargado de prevenir un poco la entrada de intrusos en el sistema, es decir, muestra algunas de las cosas que se deben tomar en cuenta para que un sistema (no únicamente éste) pueda ser seguro. Hay que aclarar que no se pretende dar una receta de seguridad ya que cada día aparecen nuevas formas de invadir un equipo, solamente se muestran algunas cosas de manera muy general ya que no es objetivo del presente trabajo.

El proyecto en general, describe la forma en la que se ha resuelto este problema en particular, sin embrago, cuenta con información que puede ser muy útil en el desarrollo de otros sistemas que requieran de Internet para funcionar, aclarando que en ningún caso es la única solución.

1

Capítulo I

Revisión general del problema.

Capítulo I: Revisión general del problema.

1.1. ¿Cuál es el problema?

La UNAM Campus Aragón cuenta con una de las poblaciones estudiantiles más grandes de la Universidad. Con sus doce carreras atiende las necesidades de educación de una de las zonas metropolitanas más alejadas de Ciudad Universitaria.

En distintas áreas se presentan diversas necesidades de distribución de la información, como es, planes de estudios, horarios de clases, horarios de exámenes finales y exámenes extraordinarios, organización de cursos y seminarios, etc., que los estudiantes consultan en áreas de la escuela correspondientes a la carrera específica.

Las instituciones de educación media superior (Bachillerato), así como las de nivel licenciatura, pretenden disponer de la información de forma dinámica, es decir, que no sea necesario asistir a la Universidad para poder actualizar los datos que se tienen acerca de:

1. Carreras.
2. Planes de Estudios.
3. Materias impartidas.
4. Periodos de exámenes finales y/o extraordinarios.

Existen algunos casos en los que los estudiantes que se encuentran fuera de la ciudad necesitan conocer los horarios. Lo que hacen es mandar un correo electrónico, que es recibido por el Departamento de Informática, posteriormente es canalizado a la Jefatura de Carrera correspondiente. Una vez ahí es leído y contestado por el Jefe o Secretario Técnico. Todo este proceso por cada persona que necesita conocer este tipo de datos.

Como puede verse todo este mecanismo ocupa tiempo del personal, que por concepto tiene un costo para la Institución

Esto hace notar que si fuera posible que las personas que requieran esta información y tengan los medios para hacerlo, consulten por su propia cuenta estos datos sin afectar las actividades del personal de la Institución, lo cual se traduce en productividad.

En la actualidad se cuenta con medios de información extremadamente modernos, la tecnología de las telecomunicaciones va cada vez más lejos y de una forma más veloz, Pueden ser desde las tradicionales, como teléfono y fax, así como las más modernas, que incluyen a las computadoras como medios de comunicación.

Internet es uno de los medios de comunicación más poderosos con los que cuenta actualmente el planeta. Permite mantener la información actualizada y su distribución desde cualquier lugar del mundo.

El problema para este caso, es utilizar los medios informáticos con los que cuenta el Campus para proporcionar a las personas que lo requieran, una forma de consultar los horarios de las materias de los viejos y nuevos planes de estudio, sin necesidad de acudir al plantel, de forma que no sea necesario atenderlos personalmente.

Esto involucra múltiples consideraciones, las cuales se tratarán a continuación:

1.2. Consideración de los recursos.

Primero que nada, se debe hacer un recuento de los recursos con los que se cuenta para resolver este problema.

- Los horarios se pegan cada inicio de semestre en áreas específicas de la escuela. Éstos son capturados en el Sistema Administrativo de Banco de Horas llamado Cronos, el cual es utilizado para ello por las doce carreras del plantel. Esto implica que cada semestre ya se cuenta con una base de datos con los horarios ya actualizados, por lo cual éstas pueden ser utilizadas.

- Se cuenta con un servidor de paginas Web por el cual se puede distribuir información a las personas que lo requieran ya que utiliza a Internet como medio de comunicación. Este servidor está montado sobre un equipo SUN SPARC STATION 4.

- En el equipo SUN se cuenta con lenguajes de programación de alto nivel como son el Lenguaje Pascal y el Lenguaje C.

- Se cuenta con los conocimientos de programación necesarios para crear módulos que analicen las bases de datos de Cronos y crear documentos HTML para consulta de horarios.

1.3. Análisis de Propuestas.

Se cuenta con dos formas de resolver el problema:

1. Analizar las bases de datos de Cronos y por medio de un programa generar todas las páginas necesarias para la consulta de los horarios. Estas páginas serían 100% estáticas, sin embargo ahorrarían el proceso de generación de las páginas una vez publicadas, es decir, solamente se haría una vez un horario, y su consulta sería directa.

2. Analizar las bases de datos de Cronos y crear un programa de tal forma que con una petición del usuario se generara en tiempo real la pagina Web con el horario especificado. Esta página únicamente existiría en el momento en que se consultara,

después dejaría de existir, y si fuera necesario volver a presentar el mismo horario, se tendría que generar nuevamente el mismo horario, lo que implica tiempo de máquina.

La propuesta número uno involucra un ahorro en tiempo de consulta, ya que en el momento que el usuario de Internet hiciera la petición, se mandaría una página prediseñada, la cual requeriría de una regeneración constante para mantenerla actualizada de los cambios que invariablemente se realizan a los horarios, sobre todo en las fechas próximas a las inscripciones. Las desventajas de utilizar este método en primera instancia sería el espacio en disco duro que las páginas ocuparían, considerando el volumen de información, además de reflejarse en la administración de esas páginas, las cuales requerirían de regeneración constante, involucrando así un derroche de recursos y quitando su autonomía. Por último es posible que existan páginas que nunca se consulten y que se encontrarían de cualquier forma en el sistema.

La propuesta número dos, involucra una regeneración constante de cada uno de los horarios a consultar, esto provoca que el sistema utilice recursos extras para generar las páginas en forma dinámica. Sin embargo propone un ahorro de espacio en el disco duro, ya que no se contará con un desglosado de todos los horarios, sino con las bases de dados con los horarios, las cuales se encuentran optimizadas. Otra ventaja que se puede observar con respecto a la propuesta número uno, es que solamente se generarán las páginas que realmente se quieran consultar y no todas, lo que involucra una optimización del sistema.

Considerando los puntos anteriores puede expresarse que el método número dos tiene un mayor aprovechamiento de los recursos, considerando que lo que se pretende es que sea lo más automático posible, es decir que no requiera demasiada administración.

La seguridad es un punto muy importante a tratar en el desarrollo del sistema, considerando que se montará en un servidor Web y por lo tanto en Internet, es necesario asegurar la confidencialidad de la información. Si consideramos que cada vez que se colocan datos en Internet se abre una puerta de entrada a millones de personas en todo el mundo, de entre las cuales se encuentran las que realmente harán uso de ella, para las cuales no se tendrá ningún temor. Pero entre todos los usuarios se encuentran los que únicamente intentarán tirar el sistema por medio de esa puerta de entrada.

En los capítulos siguientes se revisarán conceptos y tecnologías necesarias para la realización de este proyecto y se tomarán en cuenta algunas consideraciones de seguridad con que debe de contar el sistema.

2

Capítulo II

Cronos, un sistema administrativo del Campus Aragón.

Capítulo II. Cronos, un sistema administrativo del Campus Aragón

2.1. Características del sistema Cronos.

Cronos es un sistema administrativo utilizado por el Campus Aragón: fue programado por el Departamento de Informática de la Institución en el año de 1994 para facilitar la administración de los bancos de horas de profesores. En su etapa de desarrollo se consideró la necesidad de que el mismo sistema administrara también los horarios de los profesores y no sólo las horas contratadas, así que se agregaron también dos módulos más a dicho sistema. Los cuatro módulos con los que cuenta son:

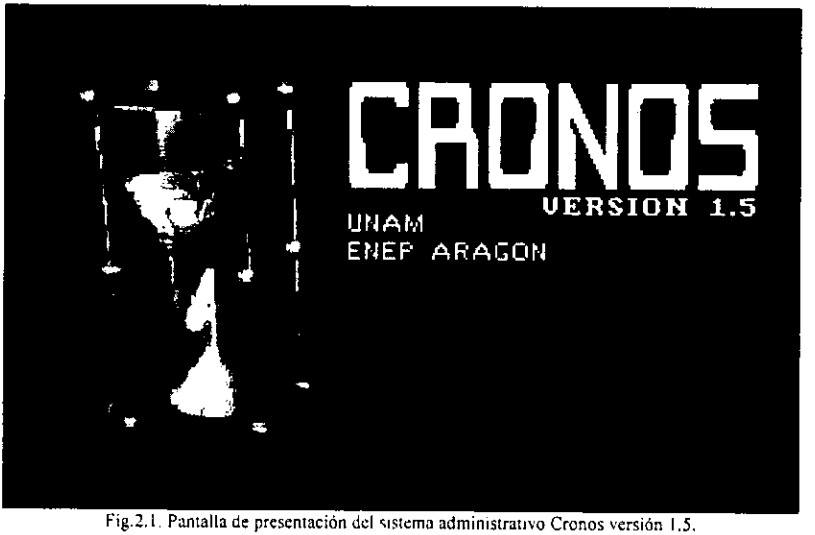

Fig.2.1. Pantalla de presentación del sistema administrativo Cronos versión 1.5.

1. Banco de horas.
2. Horarios de profesores.
3. Exámenes finales.
4. Exámenes extraordinarios.

Dado que todos los módulos con los que cuenta este sistema están relacionados con el tiempo, es que se decidió ponerle el nombre de CRONOS.

El sistema está diseñado para funcionar en red y atender -principalmente- las necesidades de las doce carreras, aunque no son a las únicas que apoya, ya que es útil para casi todas las áreas que cuentan con personal docente y de investigación.

Está programado en lenguaje Turbo Pascal 7.0 de Borland International, el cual cuenta con múltiples extensiones sobre el lenguaje Pascal original. Éstas permiten un manejo de los datos y/o procesos más poderosa y amigablemente que las del lenguaje estándar.

2.1.1. ¿Cómo iniciar Cronos?

Lo primero que se necesita, es entrar a una de las redes escolares que cuentan con el sistema[1], en particular para nuestro caso a la red del Edificio de Gobierno[2].

Por ejemplo, si se tratara de la carrera de Ingeniería en Computación, desde su directorio de trabajo teclear:

```
F:\COMPUTAC>CRONOS <ENTER>
```

Esto desplegará la pantalla mostrada por la figura 2.1, la cual presenta al sistema. Posteriormente en la figura 2.2. se solicita el semestre con el que se va a trabajar.

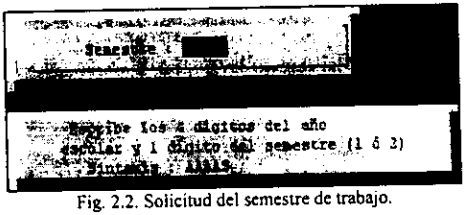

Fig. 2.2. Solicitud del semestre de trabajo.

Esta pantalla realiza el proceso de determinar si es que existe una base de datos dedicada a ese semestre en particular, siendo los cuatro primeros dígitos los correspondientes al ciclo escolar y el quinto al semestre impar o par (1 y 2 respectivamente).

Una vez elegido el semestre de trabajo, el sistema presentará el menú principal (figura 2.3), en él se apreciarán los diferentes módulos con los que cuenta, los cuales se mencionaron con anterioridad.

[1] Existen nueve redes en la escuela, cinco de las cuales son administrativas y cuatro académicas. Cronos se encuentra instalado en tres redes administrativas, que son: La del Departamento de Informática, Edificio de Gobierno y Edificio de Posgrado.

[2] En este caso se considera que la persona que entró a la red conoce el nombre de usuario (LOGIN) y su palabra clave (PASSWORD), de otro modo no podrá accesar a la red y por lo tanto utilizar CRONOS.

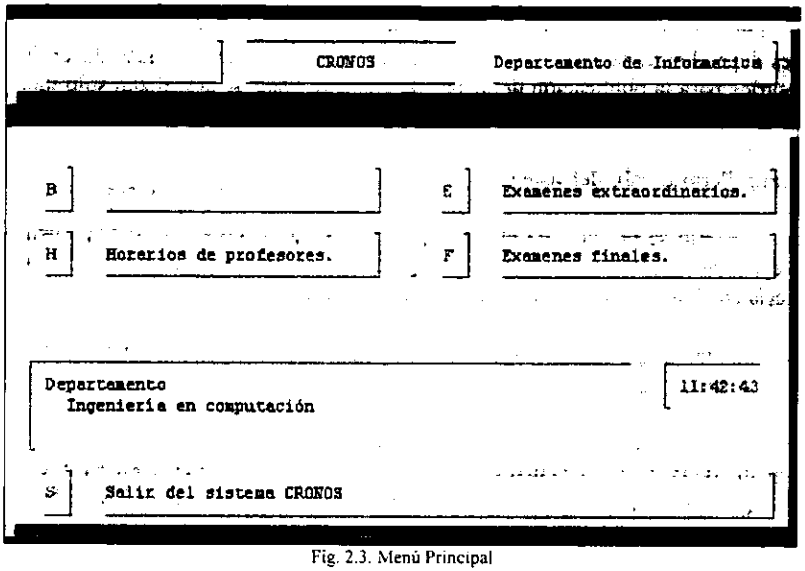

Fig. 2.3. Menú Principal

2.1.2. Descripción del menú principal.

La pantalla de la figura 2.3 nos proporciona una forma de acceder a los diferentes módulos con que cuenta en sistema[3], los cuales son:

- [B] Banco de Horas. El banco de horas se encarga de administrar las horas en las que se encuentra contratado un profesor. Es en él donde se dan de alta los profesores, se les asignan materias, y el número de horas semanales en las que impartirá la clase o realizará alguna otra actividad, así como la categoría del profesor.

- [H] Horarios de Profesores. Una vez dado de alta el profesor, se le asigna la materia respectiva, un horario de clases, un grupo y un salón. Este módulo se analizará un poco más adelante en este mismo capítulo.

- [E] Exámenes Extraordinarios. Este módulo es utilizado para programar las fechas de exámenes extraordinarios. Depende del banco de horas para designar profesores a las materias, es decir, permite proponer sínodos a las asignaturas de las que se presentará examen extraordinario. Para esta opción se cuenta con dos vueltas en cada semestre, por lo cual debe ser actualizado en cada una de ellas.

[3] Las letras encerradas entre corchetes representan teclas que se pueden pulsar para entrar al módulo que se desee.

- [F] Exámenes Finales. Cada semestre cuenta con un periodo de exámenes finales, los cuales dependen de la carrera, materia y grupo. En este módulo se introducen los datos necesarios para la publicación de fechas de examen. Éstas se actualizan sólo una vez en el semestre. En esta opción se pueden capturar dos vueltas de exámenes finales.

- [S] Salir. Permite salir del sistema.

En todo momento se presenta en el menú principal la hora actual y el departamento en el que se está trabajando. En el caso de la figura 2.3, se presenta el nombre de la carrera "Ingeniería en Computación" como departamento de trabajo.

Se cuenta con tres formas de accesar a cada una de las opciones de este menú y a todos los temas que se presentan en el sistema, éstas son:

1. Presionar la letra que se encuentra a la izquierda de cada una de las opciones.
2. Por medio de las teclas del cursor, moverse por el menú hasta que se muestre resaltada la opción que queremos seleccionar y presionar posteriormente la tecla <ENTER>.
3. Utilizar el puntero del ratón para seleccionar la opción deseada y presionar después el botón izquierdo del mismo.

2.1.3. Módulo de horario de profesores.

Aquí se puede introducir el horario de clases de los profesores capturados en el banco de horas. Este módulo es uno de los más importantes en el desarrollo de este trabajo de tesis ya que en él se captura la base de datos de los horarios de profesores.

Al entrar al módulo de horarios de profesores se presenta al usuario la pantalla mostrada en la figura 2.4. Este módulo tiene un listado de todos los registros de la base de datos de horarios, mostrados con un formato de hoja de cálculo. Esto es con la finalidad de que se puedan visualizar mejor los registros.

La presentación de la información se da por medio de dos renglones (Nombre y Materia) y la utilización de 11 columnas en las cuales se presenta:

- Salón.
- RFC del profesor (donde sólo se presentan las primeras 4 letras del mismo).
- Clave de la materia.
- Grupo.
- Los días del lunes al sábado.
- Y finalmente el número del registro en el que nos encontramos.

En este módulo se pueden realizar las operaciones comunes de manejo de cualquier base de datos como son: inserción, supresión, ordenamiento, saltos a algún registro, y por supuesto,

impresión de reportes. Estas opciones pueden seleccionarse por medio del ratón y también utilizando las teclas marcadas para dicha acción.

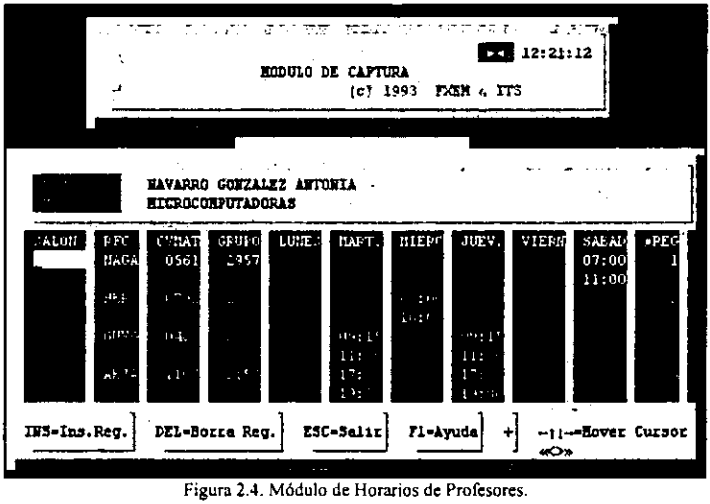

Figura 2.4. Módulo de Horarios de Profesores.

2.1.3.1. Desplazamiento por los registros.

En la pantalla 2.4 puede verse que únicamente se pueden visualizar simultáneamente cuatro registros, por lo que para ver los demás es necesario utilizar las flechas del cursor y las de control de página[4] como son:

- [←], [↑], [→], [↓]: se utilizan para desplazarse por los renglones y las columnas de las tablas de uno a uno. También puede utilizarse el puntero del ratón para estas funciones, haciendo click sobre los mismos símbolos mostrados en pantalla.
- [INICIO], [FIN]: señala al primero y al último registro de la base de datos respectivamente. Utilizando el ratón, puede hacerse click en "⟨⟨" y en "⟩⟩" respectivamente.
- [AvPág], [RePág]: avanza o retrocede, respectivamente, cuatro registros en la base de datos. Puede utilizarse el ratón en los símbolos ">", "<".
- [F6]: salta a un registro dado especificado por el usuario.

[4] De ahora en adelante, siempre que se haga referencia a las teclas de control de página, se considerarán las mismas características de funcionamiento de las teclas [INICIO], [FIN], [AvPág] y [RePág].

2.1.3.2. Insertar y borrar registros.

Para insertar un registro, es necesario colocarse sobre el que quedará antes del que se desea insertar, y presionar la tecla [INS], con esta opción se abre un espacio en blanco justo debajo del registro en el que se encontraba el cursor. Todos los que quedan debajo de éste se recorren una posición hacia abajo.

Para eliminar un registro, únicamente es necesario colocarse sobre el que se quiere borrar y presionar la tecla [DEL]. Esta acción elimina el que se encuentra seleccionado y recorre un lugar hacia arriba todos los que se están debajo de el, ocupándose el lugar del registro eliminado.

Una vez borrado un registro de la base de datos, éste no puede ser recuperado.

2.1.3.3. Edición de horarios.

Una vez que se ha agregado un espacio en blanco, que es en sí mismo un registro vacío, éste puede ser llenado, insertando los elementos descritos en cada una de las columnas. En la captura existen dos tipos de entradas de datos: las que despliegan una lista de opciones posibles (SALON, RFC y CVMAT) y los que requieren de un número entero (GRUPO, LUNES...SABADO). En caso que la captura sea por medio de lista desplegable, aparecerá una lista automáticamente en dichos campos y permitirá seleccionar la opción descada por medio de las teclas del cursor y control de página.

La captura en los campos que no tienen listas desplegables se realiza por medio del teclado. En el caso del campo GRUPO, se puede utilizar cualquier número de 4 dígitos, ya que no existe restricción en el número que un grupo pueda tener[5] , mientras que para el caso de los campos LUNES...SABADO, el programa verifica que sea un horario válido y que no se encuentre ocupado el salón correspondiente por otro registro. Se puede capturar cualquier horario entre 07:00 y 22:00 horas, sin embargo, todos estos deben ser capturados con un redondeo referente a los cinco minutos más cercanos, ya que de otra forma, el programa truncará el horario, a los cinco minutos más cercanos hacia atrás[6].

[5] Esto no significa que los grupos no tengan reglas de captura. Un grupo debe de tener como primer dígito el tipo de semestre al que pertenece (1 = impar, 2 = par), el segundo dígito representa el semestre al que pertenece (del primero al noveno semestre: 1..9, para el décimo semestre: 0); y por último los dos dígitos finales representan al número de grupo y el turno.

[6] Por ejemplo capturar el horario 07:00 - 08:30 será almacenado de esa forma, pero una captura como 07:02 - 08:28 será almacenada como 07:00 - 08:25.

2.1.3.4. Salida del módulo de horarios.

La salida del módulo de horarios se realiza mediante la tecla [ESC], o por medio del ratón haciendo *click* sobre el botón "ESC=Salir".

2.1.4. Módulo de reportes para horarios.

En el módulo de reportes, se obtiene una impresión de los horarios de clases. Ésta se puede realizar con diferentes características y en dos modalidades: dirigido a la impresora o a la pantalla del monitor en modo de previsualización.

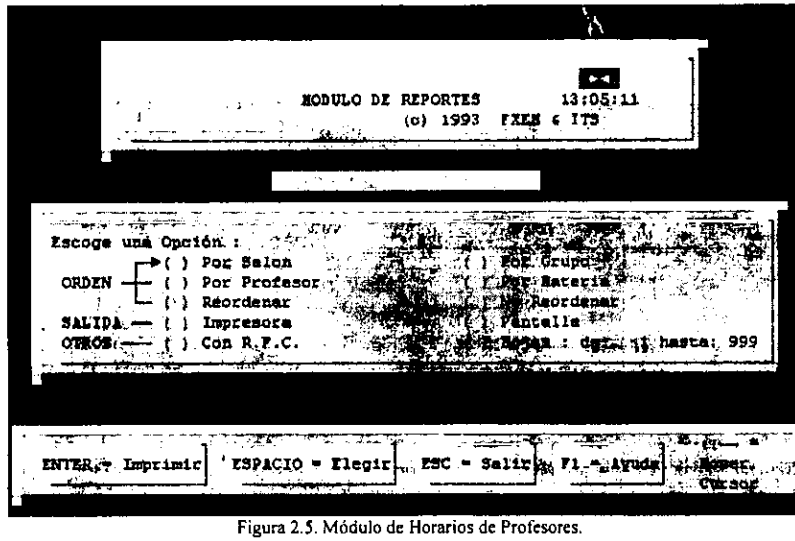

Figura 2.5. Módulo de Horarios de Profesores.

La figura 2.5 muestra todas las opciones con las que cuenta el módulo de reportes las cuales son las siguientes:

- Orden: existen cuatro formas de colocar los registros en el reporte, y esto se hace sin alterar la formación con la que se encuentran los datos en el módulo de captura. Estas formas son: por salón, por grupo, por profesor y por materia. Es importante destacar que si se selecciona la opción "No Reordenar", los horarios aparecerán de la misma forma en la que fueron capturados.
- Salida: esta opción permite seleccionar entre dos valores posibles, los cuales son: "Impresora" y "Pantalla". Si se selecciona la primera, el reporte se mandará directamente a la impresora, y en la segunda opción se mandará el reporte a la pantalla en modo de previsualización.

- Con RFC: si está activada, mandará junto con el reporte el RFC de cada uno de los profesores. Esto es con el fin de generar horarios para control administrativo o para ser consultado por los alumnos de los diferentes grupos.
- Hojas: esta opción es muy útil cuando lo que queremos es simplemente imprimir algunas hojas del reporte. Permite seleccionar la hoja inicial y la hoja final para el mismo.
- Ayuda: con este botón podemos obtener una breve explicación sobre las opciones aquí descritas, de tal forma que siempre se tiene información disponible sobre su funcionamiento.

El formato de salida de los reportes incluyen la clave y el nombre de la materia, el nombre del profesor, el RFC, el grupo, los horarios para cada día, el salón y el cupo del salón. En la figura 2.6 se presenta el formato del reporte en pantalla, El cual tiene la misma forma al imprimirse.

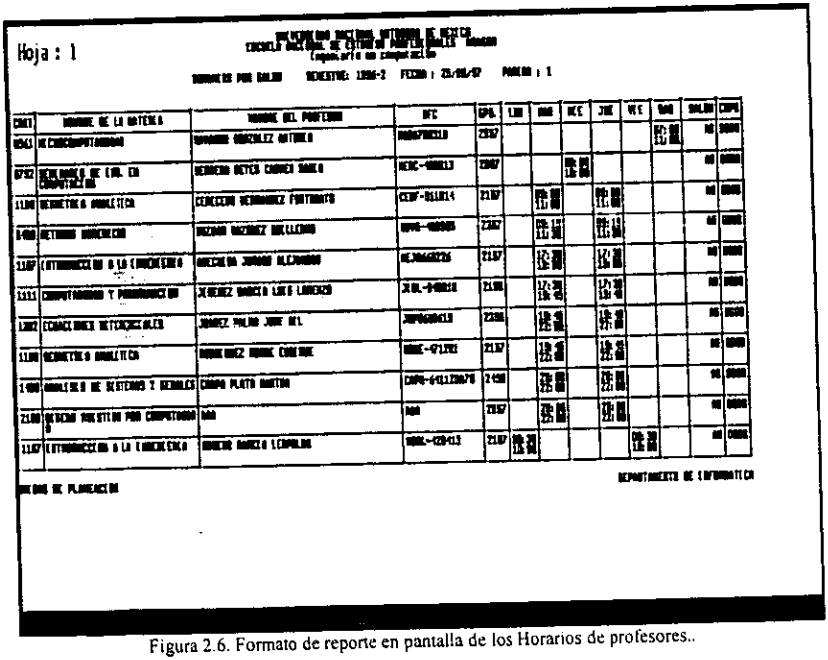

Figura 2.6. Formato de reporte en pantalla de los Horarios de profesores..

2.1.5. Módulo de exámenes extraordinarios.

El módulo de exámenes extraordinarios se utiliza para generar las listas de horarios de los exámenes, que se presentarán a los alumnos del plantel. Este módulo se actualiza dos veces en un semestre, esto es a medio semestre (Primera vuelta) y al final del semestre (Segunda vuelta).

Para entrar a este módulo es necesario seleccionar la opción correspondiente en el menú principal, ya sea con las flechas del cursor o utilizando el ratón.

Lo primero que se debe de hacer es elegir entre la primera y la segunda vuelta. Para eso utilizamos las teclas de desplazamiento del cursor hacia arriba y hacia abajo y presionando la tecla [ENTER] para elegir.

Las funciones con las que cuenta este módulo pueden ser seleccionadas utilizando el puntero del ratón o usando el teclado.

2.1.5.1. Menú principal de exámenes extraordinarios.

El menú principal de este módulo cuenta únicamente con dos opciones, las cuales son: el módulo de captura y el módulo de reportes.

Para salir de este módulo basta con presionar la tecla de escape [ESC] y contestar afirmativamente a la pregunta de si queremos salir. Esto nos devolverá al menú principal de CRONOS.

2.1.5.2. Módulo de captura de exámenes extraordinarios.

Dentro de éste encontramos la pantalla mostrada en la figura 2.7. en ella se puede apreciar un resumen de los datos capturados, ordenados por nombre de materia. Si no se encuentra ningún dato, la pantalla aparecerá vacía.

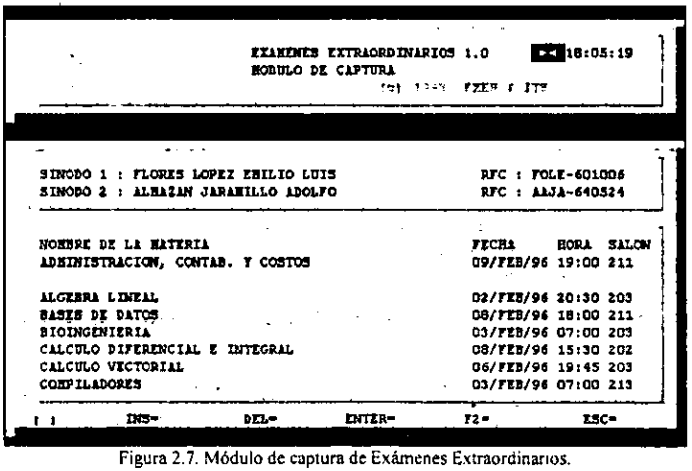

Figura 2.7. Módulo de captura de Exámenes Extraordinarios.

En esta ventana aparecen el nombre de la materia, la fecha, hora y el salón donde se realizará el examen, así como el nombre de los dos sínodos que aplicarán el mismo. Esto es con el fin de visualizar rápidamente los datos capturados. Con las flechas del cursor se puede desplazar el usuario por los diferentes registros y al hacerlo se puede ver la actualización de los nombres y RFC´s de los profesores.

En la parte inferior de la ventana se pueden apreciar distintas funciones del módulo de captura y las teclas con las cuales pueden activarse. Estas teclas son las siguientes:

- [INS]: la tecla de insertar se utiliza para dar de alta una nueva materia a la lista de extraordinarios.
- [DEL]: es utilizada para borrar una materia de la lista. Al presionarla presenta una ventana de confirmación de la acción a seguir. Esto es como medida de protección.
- [ENTER]: se utiliza para editar los datos del extraordinario seleccionado en ese momento.
- [F2]: se utiliza para ordenar alfabéticamente los datos capturados.
- [ESC]: sale del módulo de captura.

2.1.5.3. Inserción y edición de una nueva materia.

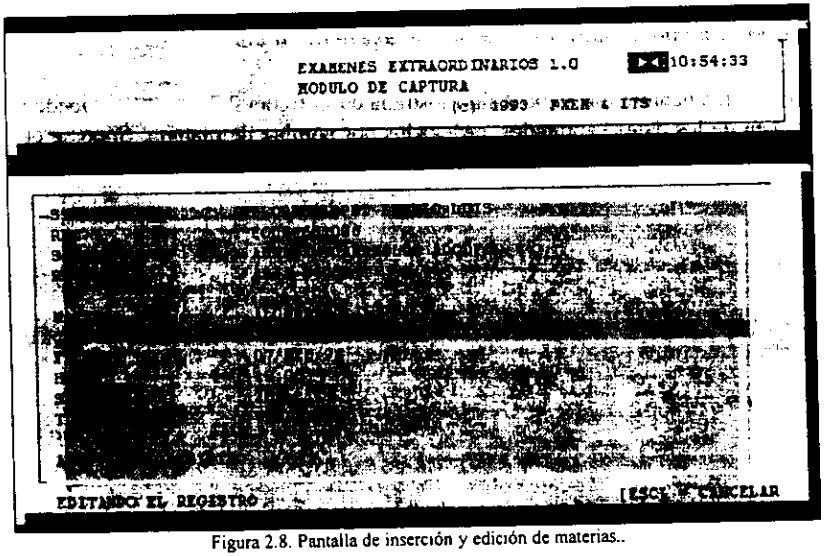

Figura 2.8. Pantalla de inserción y edición de materias..

Las pantallas de inserción y edición de materias (figura 2.8), son exactamente iguales. La única diferencia entre una y otra es que al insertar se presenta la pantalla sin datos capturados, y al editar se presentan los datos y uno puede moverse por todos ellos para modificar el que sea necesario. Para la primera se presiona la tecla [INS] y para la segunda,

se resalta el extraordinario a editar y se presiona la tecla [ENTER], ambas desde el módulo de captura mostrado en la figura 2.7.

En la parte izquierda de la pantalla mostrada en la figura 2.8. se puede observar una flecha que apunta hacia el campo que será editado. Para desplazarnos por los diferentes campos utilizamos las teclas del cursor y una vez colocados en el que nos interesa modificar, se presiona la tecla [ENTER] para hacerlo.

Al final de la pantalla se presenta el letrero "ACEPTAR EL REGISTRO", el cual al ser seleccionado salvará los cambios hechos al examen extraordinario. Esto solamente será posible si se han capturado todos los datos, de lo contrario no podrá realizarse. Si se desea salir sin salvar, es suficiente con presionar la tecla [ESC].

Para la captura de los nombres de los profesores, su RFC, materia, clave de la misma, y el salón, se utilizan menús que comparten un funcionamiento en común. Se pueden hacer desplazamientos con las teclas del cursor por los diferentes elementos que los conforman hasta encontrar el elemento deseado, ya sea una materia, un salón, o el nombre de un profesor. En la figura 2.9 se muestra el menú de selección de "NOMBRE DE PROFESOR", el cual es similar al de "NOMBRE DE MATERIA" y "SALON".

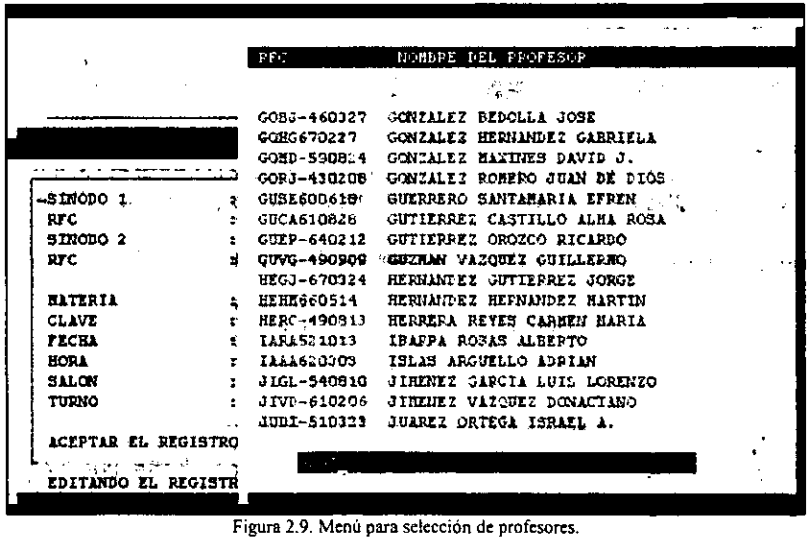

Figura 2.9. Menú para selección de profesores.

Para acelerar la localización del elemento del menú que nos interesa, se puede utilizar la barra de texto localizada en la parte inferior del menú y capturar un fragmento del nombre de la materia o del profesor. Al ir escribiendo, se seleccionará el primer elemento que inicie con ese texto. Una vez colocado sobre el elemento que se busca, se presiona la tecla [ENTER] para aceptarlo.

Para la captura de los campos de fecha y hora así como de turno, se siguen utilizando las flechas del cursor, esto es con el fin de facilitar la labor.

La fecha en la que se realiza el examen extraordinario tiene el formato DIA / MES / AÑO. Cuando la flecha del cursor señala al campo de fecha y presionamos la tecla [ENTER] se selecciona el DIA y puede conmutar entre MES y AÑO utilizando las teclas del cursor de derecha e izquierda y para modificar los valores de estos se utilizan las teclas de arriba y abajo hasta encontrar el deseado.

Una vez que se ha seleccionado la fecha correcta se presiona la tecla [ENTER] para aceptar los cambios.

Para capturar la hora se sigue el mismo procedimiento que en el caso anterior. El formato de la hora es HORA: MINUTOS.

El único campo que aún hace falta es el de turno. Este campo sirve para diferenciar las claves de los exámenes. Esta clave empieza con una letra "E" seguida de una letra que indica lo siguiente:

A Primera vuelta en semestre non.
B Segunda vuelta en semestre non.
C Primera vuelta en semestre par.
D Segunda vuelta en semestre par.

Después de esto se encuentran dos números que indican el turno o número de examen. Este número es determinado por el contenido de este campo. Para seleccionar el valor del campo "TURNO", se utilizan las teclas del cursor de arriba y abajo, y se acepa con [ENTER]

Los números que se generan con este campo son los siguientes:

Matutino	01	Matutino 3	03
Matutino 2	02	Matutino 4	04
Vespertino	51	Vespertino 3	53
Vespertino 2	52	Vespertino 4	54

Por ejemplo, la clave completa EC03 significaría: Examen extraordinario primera vuelta, de semestre par, turno vespertino 2.

2.1.5.4. Borrar una materia.

Para eliminar una materia, regresamos al módulo de captura de exámenes extraordinarios (figura 2.7), nos colocamos sobre la materia que queremos borrar y presionamos la tecla [DEL]. Esto mostrará en pantalla una confirmación de borrado de examen. Si deseamos continuar presionamos la tecla [ENTER], de lo contrario presionamos [ESC] para cancelar la acción.

2.1.5.5. Ordenar alfabéticamente.

Al capturar los exámenes, puede ser que sean introducidos sin un orden muy claro, esto dificulta la consulta y posterior búsqueda de alguno de los mismos. La tecla [F2] proporciona la función de ordenación automática por nombre de materia de todos los exámenes.

Esta función se utilizaría únicamente si se desea ordenar los registros. Si la captura se realizó en un colocación especial, lo mejor es no utilizar esta opción, ya que una vez modificada la lista, no hay forma de devolverla a su arreglo anterior.

2.1.5.6. Módulo de reportes.

A partir del módulo principal de exámenes extraordinarios podemos elegir la opción de "Módulo de reportes", la cual mostrará la pantalla de la figura 2.10.

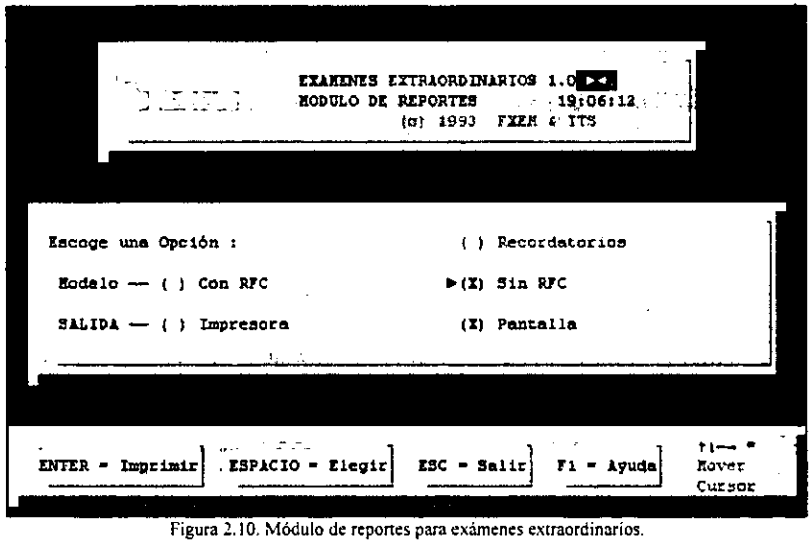

Figura 2.10. Módulo de reportes para exámenes extraordinarios.

Con éste se puede generar un reporte con RFC o sin él. Además puede servir para imprimir recordatorios de los exámenes para los profesores.

Además permite mandar el reporte en pantalla o directamente a la impresora, y en ambos casos se verán exactamente iguales. En la figura 2.11. se muestra un reporte en pantalla generado por este módulo.

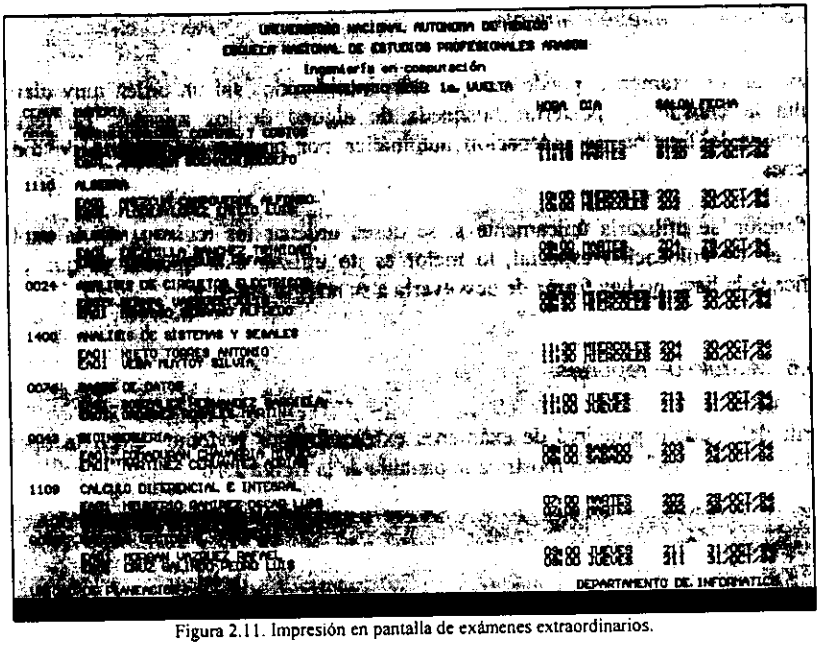

Figura 2.11. Impresión en pantalla de exámenes extraordinarios.

2.1.6. Módulo de exámenes finales.

El módulo de exámenes finales es el que se utiliza para generar las listas que se publican para ser consultadas por los alumnos al final de cada semestre.

En él se capturan las fechas y horas en las que se realizarán los exámenes finales, así como los salones respectivos.

Para entrar al módulo de exámenes finales se selecciona la opción correspondiente en el menú mostrado en la figura 2.3. (Menú principal). Al entrar en él, se mostrarán dos opciones más: El Módulo de captura y el Modulo de reportes.

2.1.6.1. Módulo de captura de exámenes finales.

Al entrar al módulo de captura de exámenes finales se presenta la pantalla de la figura 2.12. En ella se puede apreciar un listado con las diferentes materias, y la información capturada para cada una de ellas. En dado caso que no se haya insertado ningún examen final, entonces la pantalla aparecerá vacía. El usuario puede desplazarse por las diferentes materias con las teclas del cursor.

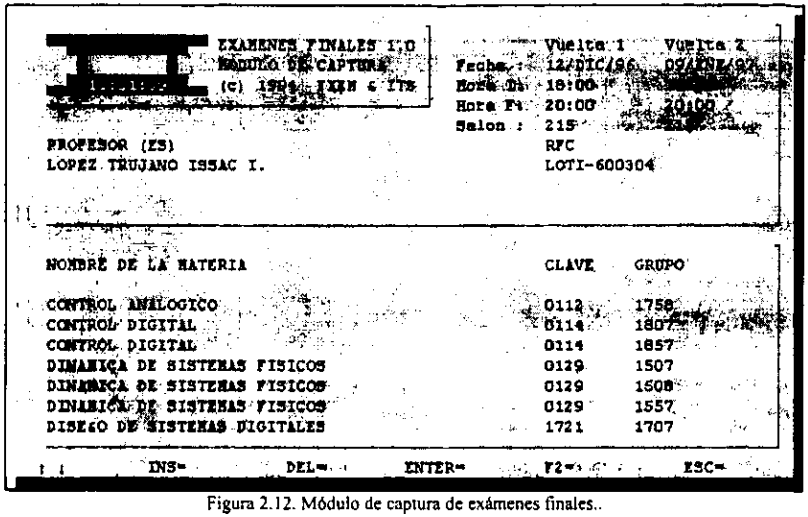

Figura 2.12. Módulo de captura de exámenes finales..

Su funcionamiento es muy parecido al de captura de exámenes finales. La diferencia consta únicamente en la información que se introduce en él. Se muestra el nombre de la materia, la clave de la misma, el grupo, la fecha, hora inicial, hora final, y salón de la primera y segunda vuelta donde se aplicará el examen, el nombre y RFC de los profesores que imparten la materia (de uno a tres). Esto permite revisar rápidamente los datos que hasta el momento existan.

En la parte inferior de la pantalla se muestran algunas opciones disponibles para este módulo, las cuales se presentan a continuación:

- [INS]: la tecla de insertar se utiliza para dar de alta una nueva materia a la lista de extraordinarios.
- [DEL]: es utilizada para borrar una materia de la lista. Al presionarla presenta una ventana de confirmación de la acción a seguir. Esto es como medida de protección.
- [ENTER]: se utiliza para editar los datos del extraordinario seleccionado en ese momento.
- [F2]: se utiliza para ordenar alfabéticamente los datos capturados.
- [ESC]: sale del módulo de captura.

2.1.6.2. Inserción y edición de una nueva materia.

Al igual que en el módulo de exámenes extraordinarios, las pantallas de inserción y edición (figura 2.13) son exactamente iguales, la diferencia consiste en que al insertar no se cuenta con información previamente capturada, en cambio para la edición de una materia, es necesario que ya se encuentre información registrada previamente, la cual únicamente será modificada si el usuario así lo desea.

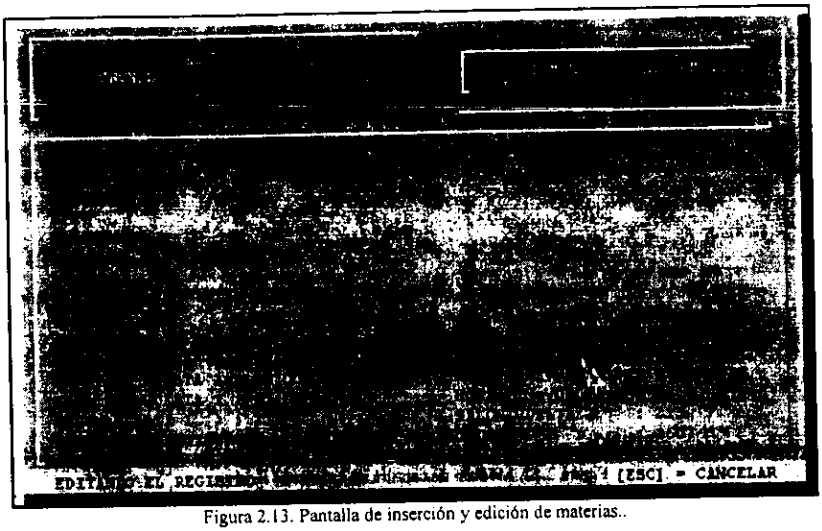

Figura 2.13. Pantalla de inserción y edición de materias..

En la parte izquierda de la pantalla mostrada en la figura 2.13., se puede observar una flecha que apunta hacia el campo que será editado. Para desplazarnos por los diferentes campos utilizamos las teclas del cursor y una vez colocados en el que nos interesa, se presionar la tecla [ENTER] para modificarlo.

Al final de la pantalla se presenta el letrero "ACEPTAR EL REGISTRO" el cual al ser seleccionado, salvará los cambios hechos al examen final y sólo será posible si se han capturado todos los datos, de lo contrario no lo hará. Si se desea salir sin salvar, es suficiente con presionar la tecla [ESC].

Para la captura de los nombres de los profesores y su RFC, Materia y clave de la misma, y los salones, se utilizan menús que comparten un funcionamiento en común. Se pueden hacer desplazamientos con las teclas del cursor por los diferentes elementos que los conforman hasta encontrar el elemento deseado, ya sea una materia, un salón, o el nombre de un profesor. En la figura 2.14 se muestra el menú de selección de "NOMBRE DE MATERIA", el cual es similar al de "NOMBRE DE PROFESOR" (mostrado en la figura 2.9) y "SALON".

Para acelerar la búsqueda del elemento del menú que nos interesa se puede utilizar la barra de texto localizada en la parte inferior del menú y capturar un fragmento del nombre de la materia o del profesor. Al ir escribiendo, se seleccionará el primero que inicie con el texto capturado. Una vez colocado sobre él, se presiona la tecla [ENTER] para aceptarlo.

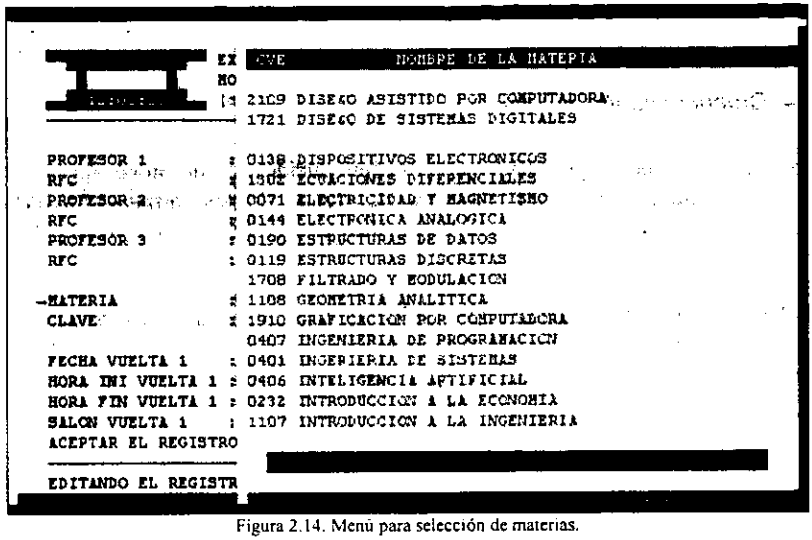

Figura 2.14. Menú para selección de materias.

Para la captura de los campos de fecha y hora se siguen utilizando las flechas del cursor, esto es con el fin de facilitar el trabajo.

La fecha en la que se realiza el examen final tiene el formato DIA / MES / AÑO. Cuando la flecha del cursor señala al campo de fecha y presionamos la tecla [ENTER] se selecciona el DIA y puede conmutar entre MES y AÑO utilizando las teclas del cursor de derecha e izquierda y para modificar los valores de estos se utilizan las teclas de arriba y abajo hasta encontrar el valor deseado.

Una vez que se ha ubicado la fecha correcta se presiona la tecla [ENTER] para aceptar los cambios.

Para capturar la hora se sigue el mismo procedimiento que en el caso anterior. El formato de la hora es HORA: MINUTOS.

El único campo que nos hace falta es el de grupo. Este campo es libre, es decir que se puede capturar directamente por el usuario sin la utilización de menús o teclas auxiliadoras. Es muy importante que se inserte correctamente, ya que el sistema no revisará si el grupo existe o no. Este campo está compuesto por cuatro dígitos.

2.1.6.3. Borrar una materia.

Para borrar una materia, regresamos al módulo de captura de exámenes finales (figura 2.12), nos colocamos sobre la materia que queremos y presionamos la tecla [DEL].

Esto mostrará en pantalla una confirmación de borrado de examen. Si deseamos continuar presionamos la tecla [ENTER], de lo contrario presionamos [ESC] para cancelar la acción.

2.1.6.4. Ordenar alfabéticamente.

Al capturar los exámenes puede ser que sean introducidos sin un orden muy claro, esto dificulta la consulta y posterior búsqueda de alguno de los mismos. La tecla [F2] proporciona la función de ordenación automática por nombre de materia de todos los exámenes.

Sin embargo, se utilizaría únicamente si se desea ordenar los registros. Si la captura se realizó en un orden especial, lo mejor es no emplear esta opción ya que una vez ordenada la lista, no hay forma de devolverla a su arreglo anterior.

2.1.6.5. Módulo de reportes.

A partir del módulo principal de exámenes finales podemos elegir la opción de "Módulo de reportes", la cual mostrará la pantalla de la figura 2.15.

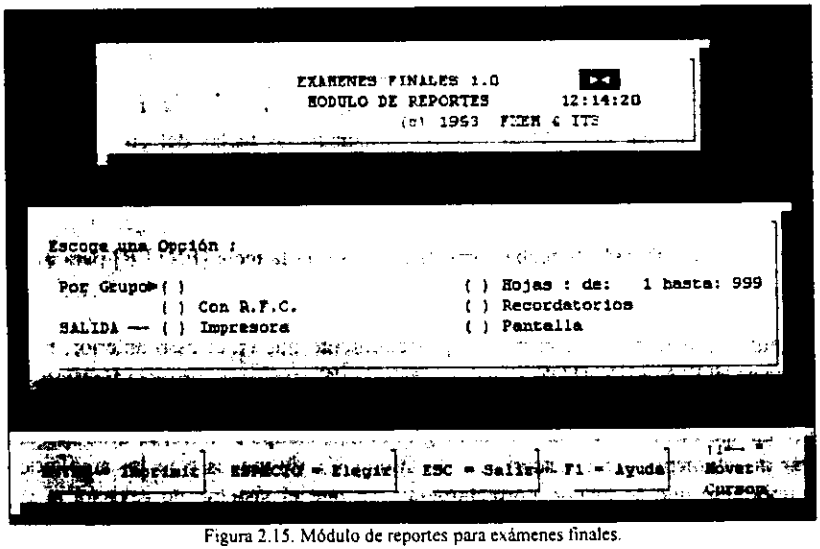

Figura 2.15. Módulo de reportes para exámenes finales.

Con éste se puede generar un reporte con RFC o sin él. Por omisión los reportes se generan ordenados por nombre de la materia, sin embargo se pueden listar por grupo, dependiendo de las necesidades de la Jefatura de Carrera correspondiente. Para esto sólo es necesario activar la casilla "Por Grupo (X)". También puede imprimir los recordatorios de examen para los profesores.

También permite mandar el reporte en pantalla o directamente a la impresora, y en ambos casos se verán exactamente iguales. En la figura 2.16. se muestra un reporte en pantalla, ordenado por grupo, generado por este módulo.

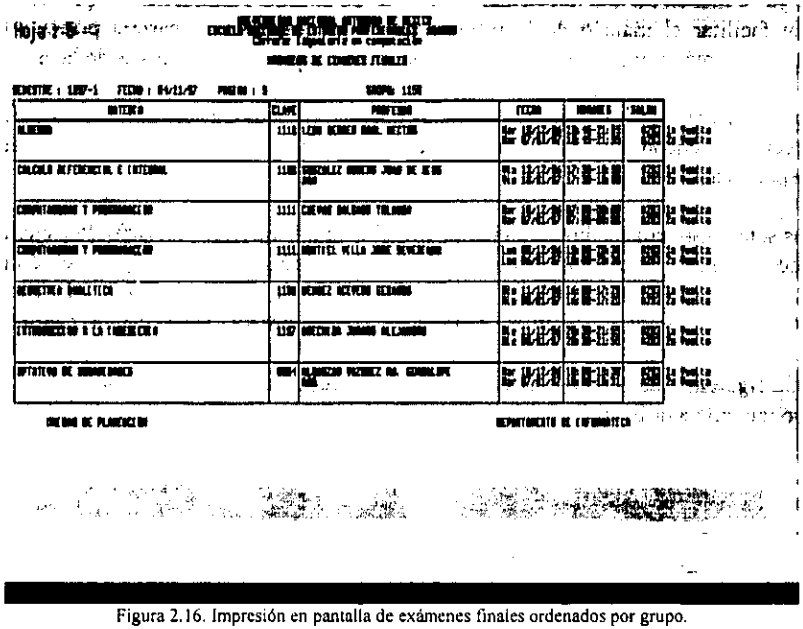

Figura 2.16. Impresión en pantalla de exámenes finales ordenados por grupo.

En cada una de las pantallas de reportes mostrados en este capítulo, puede ser que no sea muy legible el tamaño de la letra. Para ello el programa cuenta, en algunos casos con dos teclas zoom para ayudar al usuario.

[-]: para realizar un zoom del lado izquierdo de la pantalla.
[+]: para realizar un zoom del lado derecho de la pantalla.
[ESPACIO]: para regresar a la pantalla normal.

Para pasar a la página siguiente únicamente es necesario presionar la tecla [ENTER] y para salir del reporte se presiona la tecla [ESC].

2.2. Cronos bajo la cubierta.

Como ya se mencionó anteriormente, Cronos es un sistema administrativo que tiene como función facilitar el manejo de la información a las jefaturas de carrera principalmente, aunque es también utilizado por otras áreas académicas y/o administrativas de la escuela.

El sistema maneja la información de cada una de las áreas que lo utilizan en diferentes subdirectorios, encontrándose en ellos todos los archivos correspondientes únicamente al área específica.

Cronos selecciona por medio de un parámetro el área a la que dará servicio. Por ejemplo, si se quiere utilizar para administrar la Carrera de Pedagogía se utilizará el comando: `F:>CRONOS PED`, esto hará que el programa se sitúe en el directorio "`F:\CRONOS\CRONOS.PED>`" y ahí realice todos los procesos que sean necesarios.

La tabla siguiente muestra las carreras que tienen acceso al sistema, así como el parámetro correspondiente a utilizar:

Carrera	Parámetro
Jefatura de Arquitectura.	ARQ
Jefatura de Comunicación y Periodismo.	PER
Jefatura de Derecho.	DER
Jefatura de Diseño Industrial.	DIS
Jefatura de Economía.	ECO
Jefatura de Ingeniería Civil.	CIV
Jefatura de Ingeniería en Computación	COM
Jefatura de Ingeniería Mecánica Eléctrica.	MEC
Jefatura de Pedagogía.	PED
Jefatura de Planificación para el Desarrollo Agropecuario	PDA
Jefatura de Relaciones Internacionales.	REL
Jefatura de Sociología.	SOC

2.2.1. Directorios de trabajo de Cronos.

El sistema Cronos hace referencia a directorios de trabajo fijos que se encuentran en la misma unidad de discos que el sistema. A partir del directorio raíz se crea el directorio \CRONOS y dentro de él, los archivos de trabajo y los programas de apoyo.

El directorio principal del sistema es \CRONOS\CRONOS.INF, ya que es aquí donde se localiza el sistema, y cada uno de los programas auxiliares. En el directorio \CRONOS encontramos además, todos los directorios correspondientes a las áreas académicas y/o administrativas que hacen uso de él. El formato de nombre de los directorios de trabajo de las diferentes áreas cuenta con el nombre del sistema "CRONOS" y como extensión la

clave del área: CRONOS.[área]. A continuación se muestra el árbol de directorios para las doce carreras de la UNAM Campus Aragón:

Directorio de trabajo	Descripción
F:\CRONOS	Directorio base
F:\CRONOS\CRONOS.ARQ	Arquitectura
F:\CRONOS\CRONOS.PER	Comunicación y Periodismo
F:\CRONOS\CRONOS.DER	Derecho
F:\CRONOS\CRONOS.DIS	Diseño Industrial
F:\CRONOS\CRONOS.ECO	Economía
F:\CRONOS\CRONOS.CIV	Ingeniería Civil
F:\CRONOS\CRONOS.COM	Ingeniería en Computación
F:\CRONOS\CRONOS.MEC	Ingeniería Mecánica Eléctrica
F:\CRONOS\CRONOS.PED	Pedagogía
F:\CRONOS\CRONOS.PDA	Planificación para el Desarrollo Agropecuario
F:\CRONOS\CRONOS.INF	Programas y utilerías
F:\CRONOS\CRONOS.REL	Relaciones Internacionales
F:\CRONOS\CRONOS.SOC	Sociología

En cada uno se puede encontrar la información referente a la carrera en particular, independientemente del semestre del que se trate.

2.2.1.1. Subdirectorio \CRONOS\CRONOS.INF

En este subdirectorio podemos encontrar todos los programas necesarios para el correcto funcionamiento de Cronos. En él se encuentra el programa principal y un programa de utilerías que únicamente pueden ser usadas por el supervisor para la generación de archivos de trabajo y reportes en general.

2.2.1.2. Subdirectorios \CRONOS\CRONOS.[ÁREA]

En estos subdirectorios se encuentra toda la información referente al área académica a la que se refiere, los datos se clasifican por semestre, siendo éste la extensión de los archivos.

La extensión de los archivos se basa en los dos últimos dígitos del año, más un 1 si es semestre impar y un 2 si el semestre es par.

El archivo que permanece para todos los semestres es " MATERIAS.CRO" ya que las materias no varían de semestre a semestre.

Como ejemplo se colocan a continuación los archivos referentes al semestre 97-2.

Archivo	Descripción
PROFESOR.972	Lista de todos los profesores.
HOROCP.972	Lista de salones a los que tiene acceso el área.
HORAS.972	Archivo de datos del banco de horas.
HORDAT.972	Archivo de datos de horarios de profesores.
FINALES.972	Archivo de datos de exámenes finales.
EXTRA_C.972	Archivo de datos de exámenes extraordinarios vuelta uno.
EXTRA_D.972	Archivo de datos de exámenes extraordinarios vuelta dos.

Como ya se vio anteriormente, los módulos con los que cuenta Cronos son:

- Banco de horas.
- Horarios de profesores.
- Exámenes extraordinarios.
- Exámenes finales.

Dado que no todos son necesarios para la elaboración de este trabajo, solamente se describirán los archivos referentes a los últimos tres.

2.2.2. Archivos utilizados por módulo.

Los archivos que se utilizan en cada unos de los módulos son:

Módulo	Archivos utilizados
Horarios de profesores	MATERIAS.CRO PROFESOR.[SEMESTRE] HORDAT.[SEMESTRE]
Exámenes extraordinarios	MATERIAS.CRO PROFESOR.[SEMESTRE] EXTRA_A.[SEMESTRE NON] EXTRA_B.[SEMESTRE NON] EXTRA_C.[SEMESTRE PAR] EXTRA_D.[SEMESTRE PAR]
Exámenes finales	MATERIAS.CRO PROFESOR.[SEMESTRE] FINALES.[SEMESTRE]

Para poder utilizar estos archivos en el presente trabajo, es necesario conocer el formato de los mismos.

2.2.3. Formato de los archivos de trabajo.

Antes de explicar el formato de los archivos de trabajo, se debe recordar que el sistema fue programado en lenguaje Borland Pascal 7.0. Aunque más adelante se verá otro poco acerca

de este lenguaje, por el momento es importante mostrar el tamaño en bytes que tiene cada uno de los tipos de datos asociados a los registros de los archivos.

En la tabla siguiente se muestran los tipos de datos y tamaños que se utilizan en los registros del sistema:

Tipo de dato	Tamaño en bytes
Byte	1
Char	1
String[tamaño]	Tamaño + 1
Word	2
Integer	2
Longint	4
Boolean	1

Como se vio en el apartado anterior, los archivos "MATERIAS.CRO", "PROFESOR.[SEMESTRE]", y "HOROCP.[SEMESTRE]" son comunes a todos los módulos, mientras que los demás sólo son utilizados en el módulo correspondiente. Las características de los archivos de trabajo son los que se muestran a continuación.

2.2.3.1. Archivo "PROFESOR.[SEMESTRE]".

- **Estructura del registro**:

```
Profesor = Record
  ClaveProf : Word;          ( 2 bytes) Clave control del profesor.
  RFC       : String[15];    (16 bytes)
  Nombre    : String[40];    (41 bytes) Nombre del profesor.
  Funciona  : Boolean;       ( 1 byte)  Funcionario Si o No.
End;
```

- **Estructura del archivo.**

Byte 0-39 usados como cabecera del archivo.			
Clave del profesor 1 2 bytes	RFC 1 16 bytes	Nombre 1 41 bytes	Funcionario 1 1 byte
Clave del profesor 2 2 bytes	RFC 2 16 bytes	Nombre 2 41 bytes	Funcionario 2 1 byte
. . .	. . .	. . .	. . .
Clave del profesor N 2 bytes	RFC N 16 bytes	Nombre N 41 bytes	Funcionario N 1 byte

- **Tamaño del registro: 60 bytes.**

2.2.3.2. Archivo "MATERIAS.CRO".

- **Estructura del registro**:

```
Profesor = Record
  Clave     : Word;          ( 2 bytes)  Clave de materia.
  NomMate   : String[50];    (51 bytes)  Nombre de la materia.
  Sem       : Byte;          ( 1 byte)   Semestre de la materia.
End;
```

- **Estructura del archivo**.

Clave materia 1 2 bytes	Nombre materia 1 51 bytes	Semestre materia 1 1 byte
Clave materia 2 2 bytes	Nombre materia 2 51 bytes	Semestre materia 2 1 byte
. . .	. . .	. . .
Clave materia 1 2 bytes	Nombre materia 1 51 bytes	Semestre materia 1 1 byte

- **Tamaño del registro: 54 bytes.**

2.2.3.3. Archivo "HORDAT.[SEMESTRE]".

- **Estructura del registro**:

```
HoraComp = Array[1..6] of Word; (Alto = Inicio; Bajo = Fin)
Datos = Record
  RegOrd    : Word;       ( 2 bytes)  No se usa.
  Salon     : Word;       ( 2 bytes)  Salón de clases.
  Grupo     : Word;       ( 2 bytes)  Grupo.
  CveMat    : Word;       ( 2 bytes)  Clave de la materia.
  CveProf   : Word;       ( 2 bytes)  Clave del Profesor.
  Hora      : HoraComp;   (12 bytes)  Horarios.
End;
```

- **Estructura del archivo**.

Sin uso 2 bytes	Salón 1 2 bytes	Grupo 1 2 bytes	Clave mat 1 2 bytes	Clave prof 1 2 bytes	Horas 1 12 bytes
Sin uso 2 bytes	Salón 2 2 bytes	Grupo 2 2 bytes	Clave mat 2 2 bytes	Clave prof 2 2 bytes	Horas 2 12 bytes
. . .	. . .	. . .	. . .	. . .	. . .
Sin uso 2 bytes	Salón N 2 bytes	Grupo N 2 bytes	Clave mat N 2 bytes	Clave prof N 2 bytes	Horas N 12 bytes

- **Tamaño del registro: 22 bytes.**

2.2.3.4. Archivo "EXTRA_[VUELTA].[SEMESTRE]".

- **Estructura del registro:**

```
Examen = Record
  Prof1     : Word;     ( 2 bytes)  Clave del profesor 1.
  Prof2     : Word;     ( 2 bytes)  Clave del profesor 2.
  ClaveMat  : Word;     ( 2 bytes)  Clave de la materia.
  Salon     : Word;     ( 2 bytes)  Salón 1.
  Fecha     : Longint;  ( 4 bytes)  Fecha y hora 1.
  Salon2    : Word;     ( 2 bytes)  Salón 2.
  Fecha2    : Longint;  ( 4 bytes)  Fecha y hora 2.
  Turno     : Byte;     ( 1 byte)   Turno del examen.
End;
```

- **Estructura del archivo.**

Prof1 1 2 bytes	Prof2 1 2 bytes	Mat 1 2 bytes	Salón1 1 2 bytes	Fecha1 1 4 bytes	Salón2 1 2 bytes	Fecha2 1 4 bytes	Turno 1 1 byte
Prof1 2 2 bytes	Prof2 2 2 bytes	Mat 2 2 bytes	Salón1 2 2 bytes	Fecha1 2 4 bytes	Salón2 2 2 bytes	Fecha2 2 4 bytes	Turno 2 1 byte
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
Prof1 N 2 bytes	Prof2 N 2 bytes	Mat N 2 bytes	Salón1 N 2 bytes	Fecha1 N 4 bytes	Salón2 N 2 bytes	Fecha2 N 4 bytes	Turno N 1 byte

- **Tamaño del registro: 19 bytes.**

En este registro se empaqueta la fecha y la hora en un tipo especial de Borland Pascal 7.0. Este es convertido a un tipo Longint para ahorrar espacio. La forma en la que se puede sacar la fecha correcta de este tipo de datos se vera en otro capítulo más adelante.

2.2.3.5. Archivo "FINALES.[SEMESTRE]".

- **Estructura del registro:**

```
Examen = Record
  ClaveMat  : Word;     ( 2 byte)   Clave de la materia.
  Prof1     : Word;     ( 2 bytes)  Clave del profesor 1.
  Prof2     : Word;     ( 2 bytes)  Clave del profesor 2.
  Prof3     : Word;     ( 2 bytes)  Clave del profesor 3.
  Grupo     : Word;     ( 2 bytes)  Grupo de examen.
  FechaI1   : Longint;  ( 4 bytes)  Fecha y hora inicial vuelta 1.
  FechaF1   : Longint;  ( 4 bytes)  Fecha y hora final vuelta 1.
  Salon1    : Word;     ( 2 bytes)  Salón vuelta 1.
  FechaI2   : Longint;  ( 4 bytes)  Fecha y hora inicial vuelta 2.
  FechaF2   : Longint;  ( 4 bytes)  Fecha y hora final vuelta 2.
  Salon2    : Word;     ( 2 bytes)  Salón vuelta 2.
End;
```

- **Estructura del archivo.**

Mat 1 2 b	Prof1 1 2 b	Prof2 1 2 b	Prof3 1 2 b	Grup 1 2 b	Fecha Ini1 1 4 b	Fecha Fin1 1 4 b	Salón 1 2 b	Fecha Ini2 1 4 b	Fecha Fin2 1 4 b	Salón 1 2 b
Mat 2 2 b	Prof1 2 2 b	Prof2 2 2 b	Prof3 2 2 b	Grup 2 2 b	Fecha Ini1 2 4 b	Fecha Fin1 2 4 b	Salón 2 2 b	Fecha Ini2 2 4 b	Fecha Fin2 2 4 b	Salón 2 2 b
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
Mat N 2 b	Prof1 N 2 b	Prof2 N 2 b	Prof3 N 2 b	Grup N 2 b	Fecha Ini1 N 4 b	Fecha Fin1 N 4 b	Salón N 2 b	Fecha Ini2 N 4 b	Fecha Fin2 N 4 b	Salón N 2 b

- Tamaño del registro: 30 bytes.

3

Capítulo III

Servidor Web del Campus Aragón.

Capítulo III. Servidor Web del Campus Aragón.

Antes de describir las características con las que cuenta el Servidor Web del Campus Aragón, es necesario conocer lo que es el Web en si mismo.

3.1. El World Wide Web

El World Wide Web, WWW o simplemente Web puede ser definido como un sistema de información formado por hipertexto, en el cual se puede obtener información ordenada y fuera de orden, es decir, que puede ser consultada tal como aparece en pantalla o bien se puede saltar entre los diferentes vínculos que en este sistema se presentan, tiene la característica de ser múltiplataforma, fácil de utilizar, además de ser gráfico, dinámico y distribuido.

Para entender mejor los elementos de los que consta el WWW se desglosará de la siguiente manera;

3.1.1. Sistema de información de Hipertexto.

El hipertexto basa su funcionamiento a una estructura de enlaces entre diferentes conceptos o estructuras. Como se mencionó anteriormente, permite hacer consultas ordenadas o fuera de orden.

Para entender mejor esto, se pondrá como analogía la lectura de un libro. Al leer un libro podemos empezar desde la primera página, leer secuencialmente y terminar en la última. El hipertexto sería equivalente a las notas a pie de página, en donde se hace referencia a algún libro, autor, tema, etc. El lector puede tomar la decisión de "saltar" a la referencia citada y obtener información un poco más detallada de lo que se está buscando.

En la computadora se maneja con texto resaltado, que representan un ligamiento con otro documento, de tal forma que al seleccionarlo automáticamente presentará la información a la que dicha liga se este refiriendo (si es que esta información existe), que a su vez puede contener ligas a otros documentos relacionados con el texto presentado, continuando con esta cadena de forma prácticamente ilimitada.

El siguiente diagrama ilustra lo que se ha mencionado;

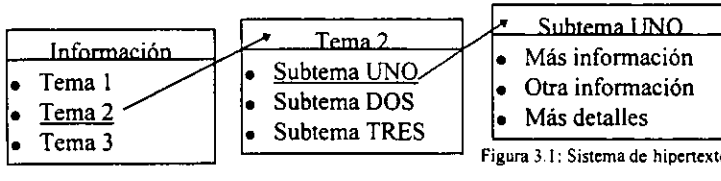

Figura 3.1: Sistema de hipertexto

El Web está basado precisamente en esta forma de estructurar la información, permite ir avanzando entre los diferentes temas y subtemas hasta encontrar lo deseado.

Esta forma de presentar la información, a través de páginas de hipertexto, es la que se utiliza en el Web. Al crear una página Web se especifican enlaces hacia otras partes del mismo documento o hacia otros colocados en el mismo servidor. La ventaja de un sitio Web es que está montado en Internet, lo que permite que las páginas puedan tener vínculos a otras que no estén en el servidor, sino en otros servidores.

3.1.2. El Web es gráfico.

Otra de las características que hacen que el Web sea tan popular es que además del hipertexto, el Web es gráfico, funciona con los mismos elementos con los que cuenta el manejador de ventanas, es decir, textos, imágenes y hasta controles como botones y cajas de texto. Esta característica lo hace extremadamente amigable al usuario, además permite crear documentos muy parecidos a las páginas de un libro o de una revista. Por todo esto el Web se ha convertido en la forma más amigable para usar Internet.

3.1.2. El Web soporta cualquier plataforma.

Considerando que las páginas Web se construyen a partir de etiquetas o *TAGS*, los cuales son simples marcas de texto que describen, un texto, un formato, una imagen o cualquier otro tipo de elemento soportado por una página Web, es que éstas son soportadas por cualquier plataforma, sólo es necesario contar con un interprete, comúnmente denominados *visualizadores*, *navegadores* o *browser's* , que se van a encargar de mostrar la información independientemente de la plataforma en la que esté funcionando.

3.1.3. Toda la información en el Web está distribuida.

Como se dijo anteriormente, estas páginas cuentan con enlaces a otras partes de un documento, o a otros en el mismo servidor. Pero al ser el Web un servicio más de Internet, es posible que los enlaces apunten a páginas localizadas en otros servidores en cualquier parte del mundo. Las posibilidades son ilimitadas ya que cualquier persona que cuente con una conexión a Internet puede poner una página en el Web en su propia computadora, y cualquier otra persona en el mundo con una conexión a Internet y un navegador puede consultar su página de forma totalmente transparente.

3.1.3. El Web es dinámico.

Cuando se lee un libro o se consulta una enciclopedia, muchas veces se encuentra información que en el momento de la impresión del libro, ésta era la más reciente. Sin embargo con el paso del tiempo la información se vuelve obsoleta, lo que provoca que poco

a poco pierda toda su utilidad. En un sitio Web no pasa lo mismo. La información que ahí se presenta puede variar en cualquier momento de acuerdo con la evolución de la información. Por poner un ejemplo, existen páginas dedicadas a presentar eventos deportivos, de tal forma que al terminar algún encuentro entre dos equipos inmediatamente se puede publicar quien fue el ganador, y esté hecho puede ser conocido en todo el mundo en cuestión de segundos.

3.1.4. El Web soporta muchos tipos de información.

Aunque el protocolo principal en el Web es el HTTP (HiperText Transfer Protocol - Protocolo de Transferencia de HiperTexto), en el cual se transportan páginas de hipertexto, en el Web también se pueden utilizar otros protocolos ya que se cuenta con una forma de accesar a la información llamada *Localizador Uniforme de Recursos* o URL. Es este el que permite accesar a otros servicios como FTP, gopher, noticias de usenet, correo electrónico, bases de datos WAIS, y telnet. Todos, excepto el último pueden utilizar la misma aplicación para usar esos servicios.

Con esta característica el Web se convierte actualmente en un estándar en Internet para la transferencia de información, ya que en las páginas de hipertexto pueden ser incluidos enlaces a los demás servicios y funcionar de forma totalmente transparente al usuario.

3.1.5. El Web es interactivo.

Por la misma naturaleza del hipertexto, la consulta de las páginas Web es totalmente interactiva, es decir, permite consultar en forma ordenada y fuera de orden. Además, el usuario decide cual será la siguiente a consultar dependiendo de los enlaces que se encuentren en el documento actualmente presentado en pantalla. Sin embargo otra característica que permite al usuario interactuar con las páginas es la inclusión de formas o controles, con los cuales se pueden especificar opciones de consulta, haciendo dependiente de esta decisión la forma o información que contendrá la siguiente página a desplegar. Otra forma de interactividad con una página Web es utilizando mapas de bits. Éstos son imágenes que cuentan con puntos de activación (hot spots) que permiten hacer vínculos con otras, dependiendo del lugar en la imagen en la que se haga un clic del ratón.

3.1.6. El Web no tiene dueño.

Considerando que el Web es un servicio montado en Internet, la cantidad de páginas de información es realmente ilimitada. Internet es una red mundial, cuyo funcionamiento ha querido ser regulado por algunos países, entre los que destacan los Estados Unidos y Alemania. Sin embargo, como se ha dicho antes, la red es mundial, esto hace imposible que las leyes de un país afecten a todos los demás. Internet está diseñada para no depender de un

solo país para mantener las conexiones. Es una red distribuida preparada para tener más de un camino para conectar a dos equipos que se encuentren conectados a ella.

Sin embargo, aunque no existe ninguna entidad "dueña" del Web u organización que pueda regular la información contenida en ella, existe una World Wide Web (W3) Consortium, cuya base se encuentra en MIT en Estados Unidos, y un INRIA en Europa.

"El World Wide Web Consortium es una organización de individuos y organizaciones interesados en dar soporte y definición a los lenguajes y protocolos que conforman a Web (HTTP, HTML, etcétera). También proporciona productos (visualizadores, servidores, entre otros) que están disponibles sin costo a cualquiera que desee utilizarlos. El W3 Consortium es lo más cercano que se puede llegar en el establecimiento de los estándares y hacer cumplir las reglas acerca de World Wide Web"[1].

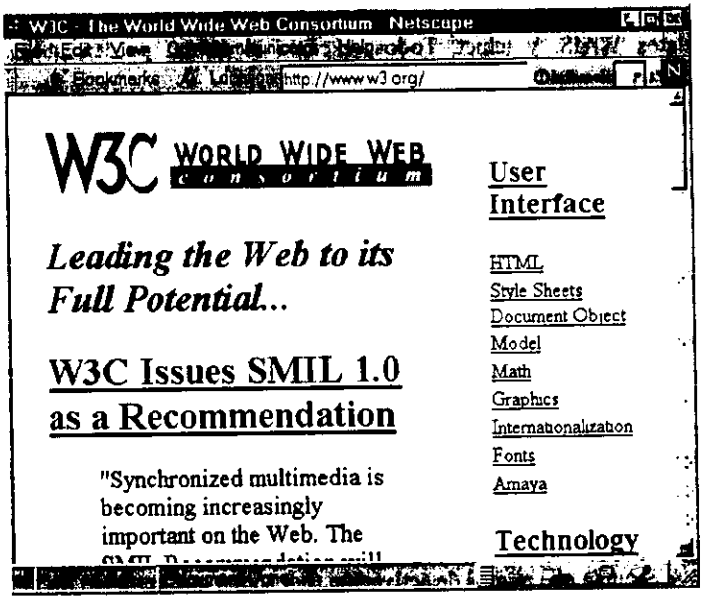

Figura 3.2: Página base del World Wide Web Consortium.

3.2. Visualizadores para Web.

El acceso al Web se logra por medio de un programa llamado *visualizador (browser) para Web*. Estos programas hacen funciones de clientes e intérpretes, ya que obtienen la

[1] Laura Lemay, "Aprendiendo HTML 3.0 para Web en una semana", segunda edición, Editorial Prentice-Hall, Página 12.

información en forma de código HTML del servidor, y lo interpretan para dar como resultado las páginas Web.

En la actualidad hay una amplia variedad de navegadores, sin embargo los dos más importantes para este momento son el Navigator o Communicator de Netscape Communications Corporation y el Internet Explorer de Microsoft.

Sin embargo, se encuentra disponible en el mercado un gran rango de visualizadores Web para casi todas las plataformas, incluyendo sistemas basados en interfaces gráficas como puede ser Windows, Mac o X11, y también los hay para sistemas de solo texto como UNIX.

3.2.1. Función de un visualizador.

"El trabajo del visualizador es doble; dado un apuntador hacia una porción de información en Internet (un URL), debe ser capaz de accesar la información u operar de alguna forma basándose en los contenidos de dicho apuntador. En el caso de los documentos Web con hipertexto, esto significa que el visualizador debe ser capaz de comunicarse con el servidor mediante el protocolo HTTP. Debido a que Web también puede manejar la información que está contenida en los servidores FTP y gopher, en grupos de noticias, en correo electrónico, etcétera, el visualizador también debe comprender el lenguaje de dichas herramientas"[2].

Sin embargo, un visualizador generalmente es utilizado sólo para el manejo de páginas en lenguaje HTML. Cada una es un documento único que incluye el texto del mismo, su estructura y cualquier enlace con otros documentos, imágenes u otros medios de información en Internet.

3.2.2. Los visualizadores populares.

Existe una gran variedad de visualizadores sin embargo aquí se mostrarán los dos más populares hasta el momento de imprimir esta tesis.

3.2.2.1. Netscape.

Netscape es uno de los dos más importantes visualizadores del momento, tiene soporte para Applets de Java y JavaScript[3], además de contar con un conjunto enriquecido de etiquetas que permiten visualizar páginas mucho mejor que otros visualizadores. Desde su surgimiento ha avanzado a pasos agigantados para convertirse en el visualizador más utilizado del mundo, al grado de que la palabra Netscape y Web, llegaron a ser prácticamente un sinónimo. Este visualizador existe para prácticamente todas las

[2] Laura Lemay, "Aprendiendo HTML 3.0 para Web en una semana", segunda edición, Editorial Prentice-Hall, Página 13.
[3] En el Capítulo V se presentan brevemente los conceptos de Applet, Java, JavaScript y ActiveX.

plataformas, desde UNIX hasta Windows 95, OS/2 y otras. Actualmente Netscape Communicator se distribuye de manera gratuita en su versión estándar. Puede encontrarse en http://home.netscape.com.

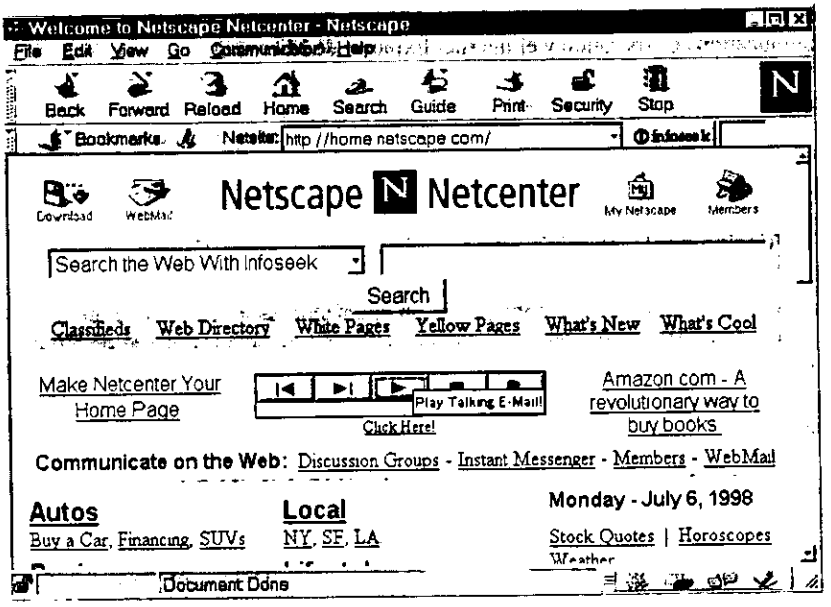

Figura 3.3: Netscape Communicator para Windows 95.

3.2.2.2. *Internet Explorer.*

Internet Explorer es el visualizador de Microsoft. Dado el crecimiento tan acelerado que ha experimentado el Web, Microsoft decidió entrar al mercado de visualizadores. En sus primeras versiones se notaba una gran desventaja sobre su principal competidor; Netscape.

Sin embargo, desde su versión 3.0 dio una buena competencia a su rival, sin llegar a hacer mucho daño. Pero con la llegada de la versión 4.0, Microsoft consolida su navegador como uno de los más importantes, incluso más que el de Netscape. Con su política de regalar el visualizador y además integrarlo al sistema operativo Windows 98, la competencia está prácticamente derrotada.

Aunque este navegador es ahora muy poderoso, sigue teniendo ciertas diferencias que lo hacen diferente. Menos poderoso en unas cosas y mucho más poderoso en otras. Soporta Applets de Java, JavaScript, Active X, y HTML dinámico.

Este navegador puede obtenerse en la dirección http://www.microsoft.com.

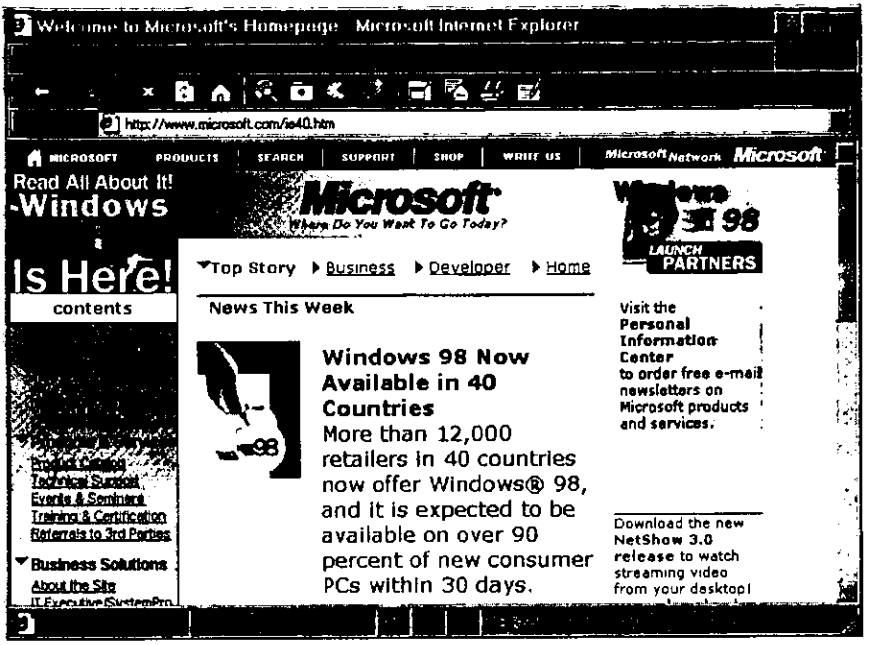

Figura 3.4: Microsoft Internet Explorer 4.0.

Otra de las cosas que soporta este navegador, es una tecnología denominada "*Canales Activos*". Estos tienen la característica de presentar el contenido de alguna página Web como si fueran canales de televisión, utilizando una barra con todos los disponibles

3.3. Funcionamiento de un Servidor Web.

"Un servidor no es otra cosa que un programa que se encuentra escuchando constantemente de la red. Cuando recibe una petición realiza la acción correspondiente y vuelve de nuevo a la escucha"[4].

Un servidor WWW, es un programa que dialoga con sus clientes utilizando el protocolo http. Su funcionamiento es realmente simple, ya que solamente tiene que mandar al cliente los archivos que éste solicita.

Cuando el cliente solicita al servidor una página, el servidor analiza la petición y determina de donde tomará el archivo que contiene la página y lo transmitirá directamente, cerrando al termino la conexión. El cliente posteriormente analiza la estructura del documento HTML,

[4] PROGRAMACION ACTUAL, Editor: Eduardo Toribio, Revista Mensual, España, Año 1, número 5, Agosto de 1997. Página 12.

si determina que necesita otros archivos como imágenes, sonidos, videos, applets de Java o algún otro, vuelve a solicitar una conexión con el servidor, y éste manda el archivo requerido, cerrando nuevamente la conexión. Este proceso se repite hasta que se terminan de transmitir todos los archivos que el cliente necesita.

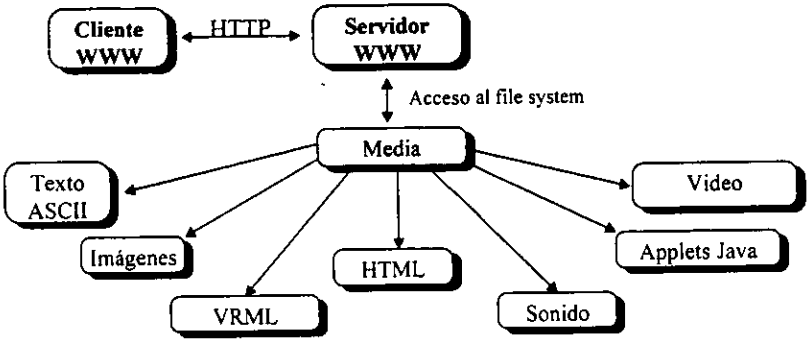

Figura 3.5: Esquema funcional mínimo de un servicio WWW.

Con este esquema básico, la complejidad del sistema reside en los clientes, ya que es el mismo cliente el encargado de procesar toda la información a fin de que la página Web pueda desplegarse completamente, debe interpretar todo el código HTML y determinar si es correcto o incorrecto.

Sin embargo, el esquema funcional de un sitio WWW puede ser mucho más complejo que lo que se acaba de ver, puesto que para hacer interactivo el Web, es necesario que el servidor realice también algunos procesos para presentar páginas que sean diferentes de acuerdo a lo que el usuario requiera, para lo cual es necesaria la utilización de funciones especiales, acceso a bases de datos, o la ejecución de un programa CGI.

3.4. El Campus Aragón en Internet.

Internet debe su éxito a la gran facilidad de uso del sistema de hipertexto. La UNAM Aragón, contando con la infraestructura necesaria para la instalación de un servidor, abre al mundo la posibilidad de conocer esta institución a cualquiera que cuente con Internet y un cliente para Web. En 1995 se lanza el proyecto "Servidor WWW de la UNAM Campus Aragón" teniendo como director de proyecto al Ing. Víctor Raúl Velasco Vega, Jefe del Departamento de Informática de la misma institución y como líder de proyecto al Ing. Itsmael Manzo Salazar. Con esto el Campus Aragón se convirtió en la primera Unidad Múltidisciplinaria en contar con este servicio[5].

[5] Para ese momento la Ciudad Universitaria ya contaba con este servicio, incluyendo a varias facultades, sin embargo de las escuelas externas, Aragón fue la primera en agregarse al mundo del Web.

Figura 3.6: Página principal del sitio Web del Campus Aragón.

El servidor fue instalado en un equipo Sun Sparc Clasic que contaba con 16 MB de memoria RAM y 500 MB de disco duro. En éste se montaron las primeras páginas para la difusión de la institución, en donde se incluyó inicialmente;

- Una carta de presentación.
- Un paseo por el Campus.
- Información sobre las doce carreras con las que cuenta el plantel.
- Información referente a los posgrados.
- El calendario escolar.
- El directorio telefónico.
- Otros lugares de interés.

El continuo incremento de las peticiones de información diaria fue determinante para cambiar el servidor por un equipo Sun Sparc Station 4, que contaba con 32 MB de memoria RAM y 2 discos duros de 500 MB, el cual se sigue utilizando actualmente. Además de disponer de más espació de almacenamiento, este servidor permite realizar consultas y procesos con mayor velocidad que el equipo anterior.

En este nuevo servidor se incluyeron las siguientes páginas;

- Revista Ilhuícatl; Especializada en temas astronómicos.
- Los satélites mexicanos.
- Estadísticas del Campus Aragón; Dedicada a mostrar los resultados obtenidos en diferentes aspectos académico-administrativos durante el periodo especificado, que generalmente es de un año.

La información presentada hasta este momento ha sido de utilidad para la difusión de las actividades desarrolladas en la institución; la constante actualización de las páginas ha permitido mantener a la vanguardia los contenidos de las mismas.

Con el paso del tiempo se han agregado páginas de acuerdo a las necesidades de la dependencia, entre ellas las que informan a los estudiantes los tramites requeridos para la titulación, el perfil de los egresados de cada carrera, así como las aptitudes y actitudes de los mismos.

El crecimiento de la tecnología y la llegada de nuevos equipos a la escuela permitieron que en el Centro de Computo Aragón se ampliara este proyecto, con la instalación de un servidor dedicado a la difusión de las actividades que en éste se realizan. También se ha utilizado para dar espacio a otras páginas, encontrándose entre ellas las de las doce carreras.

Aunque se cuenta actualmente con equipo necesario para lograr grandes proyectos de difusión, las necesidades del estudiante son mayores. Los elementos con los que cuentan hoy en día los sitios Web más modernos no son utilizados en la institución de forma constante, únicamente se utilizan los esquemas tradicionales para texto e imágenes.

En 1997, en el departamento de Informática se ve la necesidad de mejorar los contenidos dinámicos de las páginas Web, lanzándose el proyecto enmarcado con esta tesis. El objetivo de esto es utilizar las características del servidor para hacer un programa que permitiera mostrar los horarios del Campus Aragón; lo siguiente sería investigar las técnicas necesarias para lograrlo y poder utilizar los recursos con los que ya se cuenta.

En los siguientes capítulos se analizan todos los elementos necesarios para producir páginas Web dinámicas, mediante programación en un lenguaje de alto nivel que permita desarrollar aplicaciones especiales para Web y lo necesario para interactuar con el usuario de Internet que utilice un navegador y la forma en la que se pueden presentar los resultados al mismo.

4

Capítulo IV

Selección del lenguaje a utilizar.

Capítulo IV. Selección del lenguaje a utilizar.

El lenguaje de programación es uno de los elementos más importantes en el desarrollo de aplicaciones. La correcta selección del mismo ayudará a codificar de forma eficiente el programa.

Como se vio en el capítulo II, el sistema administrativo CRONOS fue codificado en el lenguaje de programación Borland Pascal 7.0 el cual cuenta con muchas extensiones al lenguaje Pascal original, esto es, cuenta con una enriquecida biblioteca de procedimientos y funciones, tipos de datos no estándar, adiciones al manejo de archivos, etc.

En este capítulo se seleccionará de entre dos lenguajes de programación: el C y el Pascal, ya que ambos se encuentran instalados en el equipo SUN Sparc Station 4 y a que son de nivel medio y alto respectivamente, y porque ambos cuentan con semejanzas importantes con el lenguaje de Borland.

Considerando que de CRONOS se tomarán las bases de datos y que éstas tienen el formato proporcionado por el manejo de archivos de Borland Pascal, se tomarán en cuenta para la selección del lenguaje la facilidad de uso de los archivos, la riqueza en tipos de datos y la sencillez para reproducir tipos de datos no estándar.

4.1. Lenguaje de programación Pascal estándar.

"El lenguaje de programación Pascal fue el primero en incorporarse, en forma coherente a los conceptos de programación estructurada definidos por Edsger Dijkstra y C.A.R. Hoare. Como tal, es una marca registrada dentro del desarrollo de los lenguajes de programación. PASCAL fue desarrollado en Zurich por Niklaus Wirth" en Eidgenossische Technische Hochschule; se deriva del lenguaje ALGOL 60, pero es más completo y fácil de usar. En la actualidad, el lenguaje PASCAL está ampliamente aceptado como un lenguaje útil que puede ser implementado con eficiencia y como una excelente herramienta de enseñanza"[1].

Al ser diseñado el lenguaje Pascal para utilizarse el la programación estructurada, puede ser usado en una gran cantidad de proyectos ya que cuenta con estructuras como:

Datos.
Asignaciones.
Procedimientos y funciones.
Tipos estructurados.
Expresiones.
Estructuras de decisión y repetición.
Tipos de variables.
Archivos.

[1] GROGONO Peter, "Programación en Pascal" Editorial; Addisin-Wesley Iberoamericana. Página vii.

La correcta utilización de cada uno de estos elementos permite realizar programas muy complejos con una mayor facilidad que con la programación tradicional.

En esta sección se espera que el lector conozca los conceptos de programación, de forma que no se profundizará demasiado en ello. Sin embargo, se analizarán elementos del lenguaje que son claves para el desarrollo del sistema tratado en esta tesis.

4.1.1. Tipos de datos simples en Pascal.

Los datos son un elemento muy importante en el desarrollo de todo sistema, sin ellos no tendría ningún objetivo, es por ello que es necesario conocer cuales son los tipos compatibles entre el lenguaje de un sistema ya desarrollado y el de un sistema que apenas se va a desarrollar.

Los tipos de datos en Pascal estándar son más limitados que los de Borland Pascal, de modo que se requeriría de una adaptación. A continuación se explican con detalle los diferentes tipos de datos que se pueden manejar en Pascal.

*4.1.1.1. El tipo **integer**.*

El termino integer es utilizado en el sentido común de la palabra. Representa a los números enteros que pueden ser positivos, negativos ó 0. De esta forma pertenecerán a este tipo los números 0, 386, -34, 7476 etc, mientras que no pertenecerán al el los números 2.465, -3.6, PI, etc. Las computadoras pueden representar sólo un número finito de números enteros, esto es por el tamaño de los registros de la computadora. Como este tamaño puede variar de una arquitectura a otra, es que se utiliza una variable estándar llamada *maxint* que almacenará el valor entero más grande que puede ser representado. Si suponemos que *I* es un número entero, entonces podemos decir que en una computadora éste se puede representar si: $-maxint \leq I \leq maxint$.

Los operadores que pueden ser utilizados con este tipo de datos son: +, -, *, div y mod, los cuales representan las operaciones de suma, resta, multiplicación, división y residuo. Una expresión formada por estos operadores será evaluada de acuerdo con el álgebra tradicional, es decir, primero se evaluarán las multiplicaciones, divisiones y residuos y después las sumas y restas.

El tipo integer, así como los tipos char y boolean, son considerados tipos enumerados, ya que mantienen un número finito de elementos y siempre se puede conocer el siguiente valor y el anterior dependiente de un valor inicial.

4.1.1.2. El tipo ***real.***

Se pueden utilizar las variables de tipo real de la misma forma en que se usar en las matemáticas aplicadas. Al igual que con los números enteros, en una computadora se pueden representar solo una cifra finita de números reales. Aunque puede representarse casi cualquier número real, puede perderse la precisión de los mismos en cálculos que requieran demasiados decimales para ser representados. Para la representación de los números reales puede utilizarse la notación normal como es: 1, 2 , -4.64, etc. o la notación científica para casos más grandes o pequeños, como puede ser: 5.125E-12, que sería equivalente a escribir 5.125×10^{-12}.

Las operaciones matemáticas que se pueden realizar con el tipo real son: suma, resta, multiplicación y división (+, -, *, /).

4.1.1.3. El tipo ***boolean.***

El tipo boolean sólo puede tener uno de dos valores: *true* o *false* (verdadero o falso). Este tipo es utilizado para almacenar estados de verdad por lo que son muy útiles para las estructuras de repetición y para las de decisión.

Existen solamente tres operadores booleanos en Pascal los cuales son: **and**, **or** y **not**. Estos operadores se pueden combinar para formar estados de verdad. El orden de evaluación que mantienen estos operadores es el siguiente: siempre se evalúa primero el not, después and y al final or en cualquier expresión, sin embargo este orden puede alterarse utilizando paréntesis.

Los valores de estas expresiones de Boole se muestran en la tabla siguiente.

A	*B*	*not A*	*A and B*	*A or B*
true	true	false	true	true
true	false	false	false	true
false	true	true	false	true
false	false	true	false	false

Tabla 4.1. Estados de verdad de las expresiones de Boole.

Otra forma de obtener estados de verdad es por medio de la comparación entre dos expresiones. Esto puede darse con los operadores de relación, que son:

< menor que...	<= menor o igual que...
= igual a...	<> diferente de...
>= mayor o igual que...	> mayor que...

Con ellos se pueden evaluar expresiones que resulten en uno de dos estados: *true* o *false*. Por ejemplo, la expresión 3>2 tendrá como resultado *true*, mientras que 3<2 será *false*.

4.1.1.4. El tipo ***char.***

El valor de una variable de tipo char es un caracter. Pascal utiliza el juego de caracteres definido en el sistema operativo en el que esté funcionando y aunque este juego de caracteres puede variar mucho de plataforma en plataforma, Pascal estándar requiere de ciertas características en la implementasión. Estas son:

1. Cada carácter debe tener un valor ordinal distinto.
2. Los valores ordinales de los dígitos 0, 1, 2, ..., 9, deben estar ordenados y ser consecutivos.
3. Los valores ordinales de las mayúsculas A, B, ..., Z, deben estar ordenados aunque no necesariamente consecutivos.
4. Los valores ordinales de las letras minúsculas a, b, ..., z (si éstas existen), deben estar ordenadas pero no necesariamente consecutivas.

Los conjuntos de caracteres más difundidos son el ASCII (American Standard Code for Information Interchange) y EBCDIC (Extended Binary Coded Decimal Information Code) y ambos tienen las propiedades requeridas por Pascal.

Los valores de una variable char deben ser encerrados entre apóstrofes simples, por ejemplo:

'A': representa la letra A.
'a': representa la letra a.
' ': representa un espacio en blanco.

Los únicos operadores que se pueden utilizar con este tipo de variables son los de relación: <, <=, =, <>, >=, >. Esto es por que los diferentes valores que lo forman pueden tener un orden específico en la tabla de caracteres y que además cuentan con un número finito de elementos.

4.1.1.5. Funciones estándar.

Para cada uno de los tipos de datos que se han presentado hasta el momento, se cuenta con una tabla de funciones que actúan directamente con ese tipo de datos, cada una de estas funciones son estándar y se presentan en todas las implementaciones de Pascal. La tabla 4.2[2] muestra las funciones estándar de Pascal con los tipos de argumentos y el tipo de datos que devuelve la función.

[2] Op Cit. Pág. 42.

Valor\Argumento	*entero*	*real*	*booleano*	*char*
entero	pred succ abs sqr	trunc round	ord	ord
real	sin cos arctan ln exp sqrt	abs aqr sin cos arctan ln exp sqrt		
booleano	odd		pred succ	
char	chr			pred succ

Tabla 4.2. Funciones estándar.

Las funciones pred y succ reciben una variable enumerada (por ejemplo integer) y devuelven el predecesor y el sucesor del argumento respectivamente.

La función abs devuelve el valor absoluto y sqr el cuadrado del argumento.

Las funciones sin, cos, arctan, ln, exp y sqrt representan al seno, coseno, arco tangente, logaritmo natural, exponencial y raíz cuadrada del argumento, siendo resultados reales los que salen de estas funciones.

Para el caso de odd el resultado es true cuando el argumento es impar y false en caso contrario.

Chr devuelve el carácter correspondiente a la posición ordinal del argumento. La función inversa de ésta es ord, que devuelve la posición ordinal del argumento; solamente puede ser utilizada en tipos ordinales como boolean, char, etc.

Cuando se tiene un número real y quiere asignarse a una variable entera, se utilizan las funciones trunc y round, que truncan y redondean el argumento respectivamente al entero más cercano.

4.1.2. Tipos de datos estructurados en Pascal.

Los tipos de datos que se han presentado hasta el momento son simples, es decir, cada variable de alguno de los tipos anteriores, puede tener solamente un dato a la vez, éste

puede ser entero, booleano, etc. Sin embargo, cuando tenemos algún tipo de datos que se relacionan con la misma cosa o se requiere que se maneje con el mismo nombre para facilitar los procesos, es cuando se utiliza lo que se denomina *tipos de datos estructurados*.

En Pascal se cuenta con algunas formas de crear tipos de datos estructurados, como los siguientes:

4.1.2.1. Los arreglos.

El arreglo es el tipo de datos estructurado más utilizado. Está formado por una colección ordenada de variables que comparten el mismo tipo. Por ejemplo se puede decir que una cadena de texto está formada por un arreglo de caracteres, ya que se cuenta con el primer elemento que seria la primera letra, el segundo elemento seria la segunda letra y así sucesivamente hasta formar el mensaje.

En Pascal se utiliza la palabra reservada *array* para formar un arreglo. Por ejemplo, si se quisiera declarar un tipo arreglo que se llamara **DiasSemana** con 7 elementos se podría escribir:

```
type
   DiasSemana = array[1..7] of integer;
```

y podría declararse una variable del tipo *DiasSemana* que contendrá los siete elementos:

```
var
   Dia : DiasSemana;
```

Para accesar a cada uno de los elementos de la variable *Dia* seria necesario especificar exactamente cual se desea. Por ejemplo si quisiéramos asignar un 5, al día número 3 se haría de la siguiente manera.

```
Dia[3]: = 5;
```

El objetivo de utilizar arreglos de datos es que se puedan tener colecciones de datos con el mismo nombre, de tal forma que puedan controlarse por medio de índices, variables o constantes. Esto acelera notablemente el manejo de la información.

Otra forma de manejar variables de tipo arreglo es declararlas directamente en la variable como se muestra a continuación:

```
var
   Dia: array[1..7] of integer;
```

Los elementos que conforman la declaración son:

Dia	Corresponde al nombre de la variable con la que se manejarán los datos.
:	Los dos puntos se utilizan siempre para definir de que tipo será la variable.
array	Palabra reservada de Pascal para definir un arreglo.
[1..7]	Índice inicial y final del arreglo separados por dos puntos (..) y encerrados entre corchetes. El primer índice debe ser menor o igual que el segundo y puede empezar con cualquier valor, aún con números negativos. Éstos no están restringidos a números enteros, los índices pueden ser de cualquier tipo enumerado.
of	Se utiliza para definir de que tipo será el arreglo.
integer	Es el tipo de datos que se maneja para este caso, sin embargo puede substituirse por cualquiera que sea válido. Éste puede ser simple o estructurado (incluyendo obviamente otro arreglo).

En ocasiones se requiere de arreglos de más de una dimensión para realizar operaciones con matrices, aplicaciones científicas o alguna otra aplicación. Es en esos casos cuando se pueden utilizar arreglos multidimensionales. La declaración de un arreglo de más de una dimensión es prácticamente la misma que en el caso anterior. La diferencia es que al especificar el tamaño, se incluye la siguiente dimensión, de tal forma que podría tenerse una declaración como la siguiente:

```
type
   Ejemplo = array[1..7,1..30,'a'..'z'] of integer;
```

en donde se tendrán tres dimensiones. La primera de 1 a 7, la segunda 1..30 y la tercera del carácter 'a' al 'z'.

Otra forma de hacer lo mismo seria de la siguiente manera:

```
type
   Ejemplo2 = array[1..7] of array[1..30] of array['a'..'z'] of integer;
```

4.1.2.2. Los registros.

Aunque los arreglos resuelven una gran cantidad de necesidades de manejo de datos, tiene la desventaja de que todos los elementos deben ser del mismo tipo. Si se deseara programar una base de datos, seria necesario utilizar más de un tipo de datos por elemento. Pascal cuenta con un tipo de datos estructurado llamado *record* que cubre esta necesidad.

Un registro es similar a un arreglo, ya que cuenta con más de un componente. Sin embargo cada uno de estos componentes puede ser diferente y por tanto se accesa a ellos por nombre y no por índice.

Una aplicación de los registros se puede ver si se requiriera, por ejemplo, del manejo de personal. Una declaración podría hacerse de la siguiente forma:

```
type
   Ejemplo3 = record
                 Nombre      : array [1..30] of char;
                 Puesto      : array [1..20] of char;
                 Sueldo      : real;
              end;
```

En este caso el tipo de datos Ejemplo3 cuenta con tres elementos: *Nombre*, *Puesto* y *Sueldo*, para los cuales se asignó un tipo de datos específico. El tipo de datos de los componentes puede ser cualquiera que Pascal acepte, incluso puede contener variables de tipo registro.

La forma en la que se accede a los datos es la siguiente:

```
var
     Variable: Ejemplo3;

begin
     . . .
     Variable.Nombre:= 'JUAN GONZALEZ PEREZ';
     Variable.Puesto:= 'JEFE DE PISO';
     Variable.Sueldo:= 6234.32;
     . . .
end.
```

Como puede observarse, se utiliza el nombre de la variable, un punto y el nombre del componente para accesar a los elementos individuales. Cada uno de ellos puede utilizarse para realizar cualquier operación válida para su tipo de datos.

Para evitar escribir todo el tiempo el nombre de la variable y solamente utilizar el componente, es utilizando la palabra reservada *with*. La forma como esto funciona es la siguiente:

```
begin
     . . .
     with Variable do
          begin
           Nombre:= 'JUAN GONZALEZ PEREZ';
           Puesto:= 'JEFE DE PISO';
           Sueldo := 6234.32;
      end;
     . . .
end.
```

Al hacer este tipo de notación se puede ahorrar un poco de código, además de facilitar al compilador la localización de los componentes ya que solamente es necesario calcular la dirección de memoria una vez.

Estos son los tipos de datos estándar más utilizados en Pascal y aunque existen otras variantes en el manejo de la información y en la formación de otros tipos de datos, no es necesario mencionarlos por no ser relevantes para el desarrollo del proyecto.

4.1.3. Manejo de Archivos.

Los programas de computadora generalmente son realizados para proporcionar resultados obtenidos por la salida de algún proceso. En Pascal tanto las entradas como las salidas de información son consideradas como en el manejo de archivos. Éstos son utilizados en la cabecera del programa. Para el caso de las entradas del teclado se cuenta con un archivo llamado *input* y para el caso de las salidas a monitor se utiliza el archivo *output*. Estos archivos siempre deben de aparecer en la cabecera del programa de la siguiente manera:

```
program UnPrograma(input,output);
```

Existen tres razones principales por lo que los archivos son importantes. "Primera, un proceso sólo se puede comunicar con su medio ambiente por medio de archivos. Segunda, un proceso es usualmente de vida corta: un programa se carga en la memoria y se ejecuta, y tan pronto como termina la memoria es usada por otro programa. Si el programa no modifica un archivo durante su ejecución, no habrá evidencia de que éste corrió del todo. La tercera razón para la importancia de los archivos es que pueden almacenarse cantidades más grandes de datos en un archivo que en la memoria.[3]"

En Pascal los archivos se manejan por medio de nombres de variables, los cuales deben de seguir las reglas de construcción de un identificador[4]. Debe existir una relación análoga entre el nombre asignado por el sistema operativo y el que se utiliza en el programa.

Existen dos tipos de archivos en Pascal: secuenciales (binarios) y de texto. A continuación se analizarán estos tipos de archivo.

4.1.3.1. Archivos secuenciales.

Para declarar un tipo ***file*** es necesario especificar las características del archivo. Por ejemplo si se desea manejar en disco una base de datos utilizando registros con el formato del ejemplo visto anteriormente, se tendría lo siguiente:

[3] Op Cit. Pág. 201.

[4] La sintaxis de construcción de un identificador que servirá para manejar variables, constantes, tipos, etc, dice que puede ser cualquier secesión de caracteres donde el primero de ellos es una letra, y los demás pueden ser letras o números. En algunos sistemas también es aceptado el carácter de subrayado ('_').

```
program UnPrograma(input,output,datos);
type
     Ejemplo3 = record
                   Nombre: array [1..30] of char;
                   Puesto: array [1..20] of char;
                   Sueldo: real;
                end;

var
      datos: file of Ejemplo3;
```

Como puede observarse la variable "datos" también se encuentra declarada en el encabezado del programa. Este archivo debe de existir en el directorio de trabajo y llamarse precisamente "datos" para poder ser utilizado.

El tipo base del archivo *datos* es *Ejemplo3*. Siempre que un archivo está abierto se tiene disponible uno de los elementos del archivo en una variable para ser utilizado por el programa, este elemento puede ser accesado mediante una variable de tipo *Ejemplo3*, que se escribe como:

```
datos↑
```

Cada uno de los componentes de este tipo de archivos suele denominarse registro, ya que siempre se tiene un registro completo en cada posición accesible por Pascal.

El archivo se estructura como una secuencia, donde los elementos se almacenan uno después del otro.

Cuando se escribe en un archivo éste se crea o se extiende, añadiendo al final cada registro escrito. El compilador de Pascal controla con un apuntador el número de registro actual del que se lee o al que se escribe. Cuando se añade un registro, éste es agregado al final del archivo de datos.

Cundo se desee crear una archivo nuevo se utiliza la instrucción ***rewrite***, que en este caso se escribiría como:

```
rewrite(datos);
```

Con esta instrucción se inicializa la variable de archivo de modo que si éste no existe se crea, y si ya existe, se borra todo su contenido y se espera a recibir el primer dato.

La instrucción necesaria para escribir un elemento en un archivo es put. Para utilizarlo se coloca la información que se va a capturar en la variable del archivo e inmediatamente después se utiliza dicha instrucción. Para hacer esto se declara una variable que funcionará como medio de comunicación entre el programa y el archivo, como se ve en el siguiente ejemplo:

```
program UnPrograma(input,output,datos);

type
     Ejemplo3 = record
                  Nombre: array [1..30] of char;
                  Puesto: array [1..20] of char;
                  Sueldo: real;
                end;

var
      datos: file of Ejemplo3;
      buffer: Ejemplo3;

begin
     buffer.Nombre:= 'JUAN GONZALEZ PEREZ           ';
     buffer.Puesto:= 'JEFE DE PISO       ';
     buffer.Sueldo:= 6234.32;
     rewrite(datos);
     datos↑:= buffer;
     put(datos);
     close(datos);
end.
```

Para abreviar la escritura a un archivo, puede manejarse el procedimiento estándar write. Para esto sólo es necesario especificar el nombre del archivo en el que se va a escribir, para este caso: *datos*. Entonces

```
datos↑:= buffer;
put(datos);
```

es equivalente a:

```
write(datos,buffer);
```

Cuando ya existe el archivo y se ha almacenado información en él, puede ser leída esta información. La instrucción para abrir un archivo y posteriormente leer de él es:

```
reset(datos);
```

Esta instrucción mueve la marca de posición de lectura del archivo al inicio del mismo, también transfiere la información del primer registro a la variable *datos*↑. Para continuar con la lectura del siguiente elemento del archivo se utiliza la instrucción

```
get(datos);
```

esto hará que la marca del archivo avance una posición adelante y almacene la información de la nueva posición en la variable *datos*↑. En caso de que no exista la nueva posición, la función ***eof***(*datos*) devolverá *true* y *datos*↑ estará indefinido. Al igual que en el caso de la escritura, también puede ser utilizada alternativamente la instrucción *read*:

```
buffer:= datos↑;
get(datos);
```

es equivalente a:

```
write(datos,buffer);
```

Las funciones para los archivos de Pascal estándar son extremadamente limitadas y por tanto se tienen algunas restricciones en su uso, ya que el tipo *file* es una abstracción de una cinta magnética. Dadas las características de estos dispositivos, para poder usar para lectura y escritura a la vez un archivo, es necesario recorrerlo todo y escribir, volverlo a recorrer y leer, con las implicaciones de tiempo que esto representa.

4.1.3.2. Archivos de texto.

Los archivos de texto tienen características similares que las del tipo file, la diferencia es que las del mismo ya están predefinidas. Éstas son las siguientes:

```
type
    text = packed file of char;
```

Los archivos estándar *input* y *output* son utilizados para la comunicación directa con el usuario, es decir, son los archivos que Pascal relaciona con el teclado y con el monitor. Podemos asignar este tipo de datos a otro archivo ya que el tipo *text* es estándar.

Los procedimientos de entrada y salida estándar *read* y *write*, utilizan los archivos antes mencionados por default, es por ello que si no se especifica un nombre de archivo, se asume que se leerá del teclado o se desplegará en el monitor el parámetro asociado.

La instrucción *read(datos,variable)*; leerá del archivo *datos*, un elemento del mismo tipo que el de *variable* y lo almacenará ahí mismo. Esta misma orden puede especificar más de una lectura a la vez, declarando más variables en la expresión, siendo éstas de tipo *integer*, *char* o *real* , utilizando la sintaxis: *read(datos, var1, var2, var3, ...)*;. Algo que es importante señalar, es que siempre debe haber información que leer antes de hacerlo, ya que puede intentarse leer información inexistente y provocar un error en tiempo de ejecución. Para hacer esto existen las funciones estándar *eof* y *eoln* (fin de archivo y fin de línea respectivamente). Estas instrucciones devuelven *true* si se encuentran en el fin del archivo o de la línea y *false* en todos los demás casos. Es importante usarlas antes de realizar cualquier lectura de información en cualquier archivo.

La instrucción encargada de escribir a un archivo es *write*. Ésta, al igual que *read* puede almacenar cualquiera de los tipos de datos anteriormente mencionados, además del *booleano*. La sintaxis *write(datos, var1, var2, var3, var4, ...)*; se encargará de mandar en una sola línea sobre el archivo *datos* el contenido de var1, var2, var3, var4, etc; si se desea capturar en una sola línea lo anterior, y después saltar a la siguiente se utilizaría *writeln* en

lugar de *write*. Lo mismo sucede con *read*, si se desea leer de una línea, y después saltar a la siguiente a continuar la lectura, se utilizaría *readln*.

4.1.4. Ventajas y desventajas de utilizar el lenguaje Pascal estándar.

Los lenguajes de programación están diseñados para sacar el provecho necesario de la plataforma en la que están implementados. Sin embargo la forma en la que estos interactuan con el mismo puede diferir mucho entre una plataforma y otra, así como de lenguaje a lenguaje.

Generalmente se utiliza en una plataforma el lenguaje en el cual estamos más familiarizados, para mi caso ese lenguaje es el Pascal. Sin embargo es muy importante verificar si un lenguaje en particular nos facilitará el trabajo por conocerlo o nos complicará el mismo por no contar con los elementos necesarios para resolver un problema especifico de manera directa.

4.1.4.1. Ventajas.

Una de las características importantes de Pascal es su compacticidad, ya que existe un número relativamente pequeño de instrucciones básicas, y éstas pueden ser combinadas de muchas formas, lo cual lo hace muy poderoso. Los algoritmos y las estructuras de datos pueden ser construidos jerárquicamente.

En particular, la naturaleza dispersa de Pascal es su ventaja más importante y benéfica en por lo menos cuatro aspectos:

1. Pascal es relativamente simple de compilar y los programas en Pascal corren eficientemente en la mayoría de las implementaciones.
2. Los programas en Pascal bien escritos son fáciles de leer y entender.
3. La naturaleza recursiva de las estructuras de Pascal permite que las técnicas de desarrollo de arriba hacia abajo sean aplicadas en una manera natural y simple.
4. Con pocas excepciones, las estructuras de control de Pascal son fáciles de verificar porque su semántica es simple.

En Pascal se ha logrado un arreglo: hay una variedad suficiente para expresar de una manera concisa la mayoría de los algoritmos básicos, pero no tanta que el lenguaje resulte cargado.

4.1.4.2. Desventajas.

Las desventajas de un lenguaje siempre van acompañadas de las características que los desarrolladores quisieran localizar y no encuentran en Pascal. Algunas de las que se han dado son las siguientes:

1. Las capacidades de manejo de sus archivos son inadecuadas, ya que Pascal proporciona facilidades automáticas de conversión para los archivos de texto, pero no pueden ser usadas en programas reales porque no hay previsión para condiciones de error. Además de que en Pascal estándar no existe el acceso aleatorio a los datos.

2. El tamaño de un arreglo es determinado en el tiempo de compilación, lo que dificulta notablemente la programación de aplicaciones de análisis numérico o manipulación de cadenas porque no hay arreglos dinámicos y aunque este inconveniente puede ser superado con algunas técnicas de programación, esto no lo hace la mejor opción para este tipo de aplicaciones.

3. No existe la construcción de un ciclo con una salida. Hay muchas ocasiones en las que es necesario salir a la mitad de un ciclo y la única manera en la que se puede hacer esto en Pascal es introduciendo variables booleanas adulteradas o usando proposiciones goto, lo cual afecta notablemente a la programación estructurada.

Otro de los inconvenientes de Pascal es que no todas las implantaciones comparten todas las instrucciones o tipos de datos, lo que evita enormemente la portabilidad de código entre una plataforma y otra, además de la imposibilidad de estar seguro de que el juego de caracteres será el mismo, lo cual incrementa los inconvenientes.

Pascal ha tenido una enorme evolución con respecto a su versión estándar, sin embargo, las implementaciones más modernas no corren en todas las plataformas, entre ellas la de SUN, que se apega más al estándar.

4.2. Lenguaje de programación ANSI C.

C es un lenguaje de programación de propósito general, esto significa que puede ser utilizado para desarrollar casi cualquier tipo de aplicación, desde sistemas operativos, hasta sistemas de bases de datos. Aunque C se ha asociado mucho con el sistema operativo UNIX porque el sistema y los programas escritos en él están en este lenguaje, C no está ligado a ningún sistema operativo o plataforma, lo que hace que sea un excelente lenguaje de desarrollo.

C es un lenguaje de "nivel medio", es decir, trabaja en muchos casos con elementos del sistema operativo y de la plataforma de forma natural, lo que hace que sea muy rápido, comparado con otros lenguajes. Sin embargo, por la misma causa no trabaja de manera

natural con elementos compuestos, lo que obliga a utilizar funciones para realizar tareas como manejo de cadenas, entradas y salidas de información a archivos o a la salida estándar, etc.

Este lenguaje es relativamente pequeño, pues cuenta con pocas instrucciones nativas, pero tiene todo lo necesario para hacer programas muy complejos.

La especificación del estándar ANSI C se inicia en 1983, cuando el *American National Standards Institute* (ANSI) establece un comité que se encargó de proporcionar una definición moderna y comprensible del lenguaje. Esta definición fue aprobada en 1988 con relativamente pocos cambios con respecto a su primera especificación.

En esta sección se tratarán los elementos que anteriormente se especificaron sobre el lenguaje Pascal, no se abundará mucho al respecto, sólo en lo referente al interés de este trabajo.

4.2.1. Tipos de datos en ANSI C.

Aunque C no cuenta con una gran cantidad de tipos de datos, los existentes son realmente flexibles, ya que disponen de modificadores especiales que pueden aumentar o disminuir la precisión de un dato. Al igual que en Pascal, una variable, tipo, nombre de función constante, o algún otro identificador tiene reglas de construcción. El estándar de C permite que los identificadores se construyan con letras y dígitos, además del carácter de subrayado '_' el cual es considerado como una letra. Un identificador siempre debe iniciar con una letra y su longitud máxima debe de ser de 31 caracteres para que se garantice que el compilador reconocerá la diferencia entre dos identificadores, aunque éstos pueden ser aun más grandes. Otro punto importante es que las letras mayúsculas y minúsculas son diferentes, por lo que *variable* y *Variable*, serán identificadores distintos.

4.2.1.1. Los tipos básicos y sus tamaños.

Los tipos básicos de C son los siguientes:

char	mide un sólo byte, es capaz de contener un carácter del conjunto de caracteres local.
int	entero, es normalmente del tamaño natural de los enteros de la máquina en la que se ejecuta.
float	punto flotante de precisión normal.
double	punto flotante de doble precisión.
void	carencia de tipo.

Además, existen algunos modificadores de tipo, que pueden proporcionar otras características, estos son: *short, long, signed, y unsigned.*

Los modificadores *short* y *long* se aplican para aumentar la precisión de los tipos numéricos principalmente. Cuando se utilizan en el tipo *int*, éste disminuye o aumenta el número de bits que utiliza para expresar una cantidad. El tipo *long* debe ser de cuando menos 32 bits, el tipo *short* debe ser cuando menos de 16, y el tipo *int* puede ser de 16 o de 32, pero nunca menor que *short* o mayor que *long*.

Cuando se utiliza *signed* o *unsigned*, se le está especificando a los tipos enteros o *char* que van a usar números enteros de cualquier signo, o sólo positivos respectivamente. La forma en la que se utilizan los tipos al declarar una variable es:

```
[modificador] tipo identificador;
```

Por ejemplo, para declarar un entero corto se utiliza:

```
short int EnteroCorto;
```

Cuando se utiliza el tipo *long double*, se especifica un punto flotante de precisión extendida. El tamaño que tendrá depende de la implantación y es por ello que el los tipos *float*, *double* y *long double* pueden representar uno, dos o tres tamaños diferentes.

4.2.1.2. Operadores.

C cuenta con una serie de operadores de diferentes tipos que son: aritméticos, de relación, lógicos, de incremento/decremento, manejo de bits y de asignación/expresión.

Los operadores aritméticos son los comúnmente conocidos: +, -, *, /, y el módulo o residuo de la división % que sólo puede ser aplicado a enteros.

Los operadores de relación comparan un operando con otro y dependiendo de ésta, regresan un valor verdadero o falso. Estos son: >, >=, ==, <=, < y !=, los cuales representan mayor que, mayor o igual, igual a, menor o igual, menor que y diferente que.

Los operadores lógicos son aquellos que permiten evaluar las expresiones de Boole. En C, se considera como positivo un valor 1, y como falso el valor 0. Por lo tanto al negar una variable con valor 1, ésta valdrá 0 y viceversa. Los operadores son: && , || y !, que representan en "*y*" lógico, el "*o*" lógico y la *negación* lógica respectivamente. Los valores que se pueden obtener con estos operadores son en funcionalidad iguales a los operadores lógicos de Pascal.

Los operadores de incremento y decremento son operadores especiales de C. El operador ++ agrega 1 a su operando, mientras que – – le resta 1 a su operando. Cualquiera puede ser utilizado antes o después de su operando. El resultado es aumentar o disminuir en uno al mismo. La diferencia es que si se utiliza ++n, el valor de n se incrementa antes de ser utilizado, mientras que si se utiliza n++, el valor de n es incrementado después de que su valor es utilizado.

Los operadores de manejo de bits son muy útiles cuando se quiere manejar información a nivel de bits en una palabra de computadora completa. C proporciona 6 operadores de manejo de bits:

Operador	Acción
&	AND de bits
|	OR inclusivo de bits
^	OR exclusivo de bits
<<	corrimiento a la izquierda
>>	corrimiento a la derecha
~	complemento a uno (unario)

Tabla 4.3. Operadores de manejo de bits.

Estos operadores son muy útiles en el enmascaramiento de bits o en el encendido y apagado de los mismos. Estos operadores son extremadamente rápidos. Por ejemplo es más rápido realizar una multiplicación por 2 de la forma x = x << 1; que x = x * 2; puesto que las operaciones con bit son directas al procesador, mientras que las operaciones aritméticas son interpretadas por el compilador y luego mandadas al procesador.

Por último, los operadores de asignación y expresión son una contracción de operaciones comunes en los lenguajes. Por ejemplo la operación x = x +5; puede ser expresada como x += 5; el operador += es denominado operador de asignación.

Operadores	Asociatividad
() [] ->	Izquierda a Derecha
! ~ ++ -- + - * & (tipo) sizeof	Derecha a Izquierda
* / %	Izquierda a Derecha
+ -	Izquierda a Derecha
>> <<	Izquierda a Derecha
< <= > >=	Izquierda a Derecha
== !=	Izquierda a Derecha
&	Izquierda a Derecha
^	Izquierda a Derecha
|	Izquierda a Derecha
&&	Izquierda a Derecha
||	Izquierda a Derecha
?:	Derecha a Izquierda
= += -= *= /= %= &= ^= |= <<= >>=	Derecha a Izquierda
,	Izquierda a Derecha

Tabla 4.4. Precedencia y asociatividad de operadores.

La mayoría de los operadores binarios (como por ejemplo +, que tiene un operador derecho y uno izquierdo) tienen un correspondiente operador de asignación del tipo *op*=, en donde op es uno de los siguientes: +, -, *, /, %, <<, >>, &, | y ^.

Si expr1 y expr2 son expresiones, entonces: *expr1 op= expr2* es equivalente a *expr1 = (expr1) op (expr2).* Por ejemplo *x *= y + 4;* es equivalente a *x = x * (y+4);*

La tabla 4.4., contiene las reglas de precedencia de todos los operadores con los que cuenta C, incluyendo aquellos que no se tratan en este texto. La correcta utilización de esta tabla evitará resultados erróneos en la evaluación de expresiones, así como en la asignación de las mismas.

4.2.1.3. Los arreglos.

Los arreglos son una forma muy común de trabajar en C. Los arreglos y los apuntadores están completamente relacionados, de forma que funcionan prácticamente igual.

En esta sección no se hablará mucho de los apuntadores, sino de la forma en la que se manejan y declaran los arreglos.

Como se recordará en Pascal se manejó el concepto de arreglo, estos son colecciones de variables que comparten el mismo nombre y tipo de datos, y la forma de acceder a cada uno de los elementos es por medio de índices.

En C la forma de declarar un arreglo es:

```
tipo_de_datos identificador [Tamaño];
```

Por ejemplo podemos tener un arreglo como:

```
int Nuestro_Arreglo [20];
```

Para este caso se cuenta con una variable llamada *Nuestro_Arreglo* que cuenta con 20 elementos. Es importante señalar que los índices inician en 0 (cero) y que el último de ellos es el 19 de modo que se tengan 20 elementos en total.

Cuando se desea declarar un arreglo de más de una dimensión se puede hacer con múltiples corchetes, uno por cada dimensión y si se requiere un arreglo de dos dimensiones, la declaración es:

```
int Nuestro_Arreglo[20][20];
```

Con esta declaración se cuenta con una matriz de 20 × 20 de enteros, en donde el primer elemento se encuentra en *Nuestro_Arreglo[0][0]* y el último en *Nuestro_Arreglo[19][19]*.

4.2.1.4. Las Estructuras.

Como se ha visto, una estructura es una colección de una o más variables, de tipos iguales o posiblemente diferentes, las cuales se encuentran agrupadas en el mismo nombre para facilitar su utilización. Esto resulta muy útil en un sinnúmero de aplicaciones, entre las que destacan las bases de datos.

La forma básica de declarar una estructura es con la palabra reservada *struct* de la siguiente manera;

```
struct Una_Estructura
   {  int elemento1;
      float elemento2;
      char cadena[25];
   };
```

Aquí se ha definido una con tres elementos que tienen más de un tipo de datos. Dicha estructura puede ser utilizada bajo el nombre de *Una_Estructura*. Por ejemplo si se desea utilizar la estructura en una variable llamada *datos*, se utilizaría la siguiente instrucción:

```
struct Una_Estructura datos;
```

Para acceder a los elementos de la estructura se utiliza el nombre de la estructura, seguida de un punto y el nombre del elemento, como se muestra a continuación:

```
datos.elemento1 = 245;
datos.elemento2 = 42.45;
datos.cadena = "Cadena de texto";
```

Todas estas variables se manejan de la misma forma que en los casos en los que no se encuentran dentro de una estructura, debe recordarse que ella sólo agrupa las variables, sin embargo, éstas mantienen todas las características del tipo de datos.

La palabra reservada *typedef* asigna a un tipo de datos cualquiera otro nombre. Por ejemplo, si se quisiera utilizar uno nuevo para manejar un tipo de datos que fuera estructura, pero también es necesario evitar usar la palabra *struct* en la declaración de variables, sería muy útil crearlo como se muestra a continuación:

```
typedef struct Una_Estructura
   {  int elemento1;
      float elemento2;
      char cadena[25];
   } Nuevo_Tipo;
```

Con esto la declaración de variables sería:

```
Nuevo_Tipo datos;
```

Gracias a esta forma, se puede tener un mejor control de los datos que se manejen.

En ocasiones se tiene que trabajar con bits independientes en donde el primer elemento de un byte representa un estado, los siguientes dos, representan otra cosa y los cinco restantes tienen una función más. La forma en la que puede trabajarse con estos bits es por medio de máscaras con funciones booleanas. Sin embargo C cuenta con una forma que resulta más fácil de utilizar, al mismo tiempo que produce código más legible. Esta forma de separar los bits es con los campos de bits, que es una estructura especial que permite hacer esto. El formato de trabajo es el siguiente:

```
struct {
     unsigned int PrimerDato: 1;
     unsigned int SegundoDato: 2;
     unsigned int TercerDato: 5;
} Campo_de_bits;
```

Como puede verse, la estructura es muy parecida a la que se ha visto anteriormente, la diferencia es que los tipos de datos son *unsigned int* y que al final debe especificarse cuantos bits medirá el campo. De esta forma puede verse que la estructura solamente medirá, para este caso 8 bits únicamente. La forma de accesar a los campos es la misma que en otras estructuras, lo mismo sucede con las asignaciones y operaciones que con ellos se realicen. El tipo *unsigned int* se utiliza para asegurar que se manejarán como enteros sin signo.

4.2.2. Archivos.

C en sí mismo, no cuenta con instrucciones especificas para el manejo de archivos ni para manejo de dispositivos de entrada/salida como se da en otros lenguajes de programación, sin embargo cuenta con una biblioteca de funciones especificas para manejo de entradas y salidas de cualquier tipo. En esta sección se explicarán solamente algunas de las instrucciones necesarias para el manejo de archivos sin profundizar demasiado en ellos.

4.2.2.1. Entrada y salida estándar.

La entrada y salida estándar corresponden principalmente al teclado y al monitor respectivamente, aunque por medio del sistema operativo pueden dirigirse a otras fuentes o destinos, como puede ser un archivo, una impresora, un scanner, etc. La característica principal de la entrada y salida estándar es que se manejan como documentos de texto, cada uno de los datos que en ellos se encuentran se leen o escriben como cadenas de texto. Las variables de tipos diferentes al de texto, pueden utilizar caracteres de conversión para que el compilador detecte de que tipo son y pueda interpretarlas como de texto.

Las instrucciones que se utilizan para la entrada y salida estándar se encuentran en la librería *stdio.h* (standard input/output). Las instrucciones que son comúnmente utilizadas para realizar estas funciones son: *printf* y *scanf*.

La instrucción *printf* manda a la salida estándar la información contenida en sus parámetros. Su uso es el siguiente:

```
int printf (char *formato, arg1, arg2, ...);
```

printf convierte, da formato e imprime sus argumentos en la salida estándar bajo en control de *formato*. Regresa el número de caracteres impresos.

La cadena de formato contiene dos tipos de objetos: caracteres ordinarios, que son copiados directamente al flujo de salida y especificaciones de conversión, cada uno de los cuales causa la conversión e impresión de los siguientes argumentos sucesivos de *printf*. Cada especificación de conversión comienza con un % y termina con un caracter de conversión. Entre el % y el caracter de conversión pueden estar, en orden:

- Un signo menos, que especifica el ajuste a la izquierda del argumento convertido.
- Un número que especifica el ancho mínimo de campo. El argumento convertido será impreso dentro de un campo de al menos este ancho. Si es necesario será llenado de blancos a la izquierda (o a la derecha, si se requiere ajuste a la izquierda) para completar la amplitud del campo.
- Un punto, que separa el ancho del campo de precisión.
- Un número, la precisión, que especifica el número máximo de caracteres de una cadena que serán impresos, o el número de dígitos después del punto decimal de un valor de punto flotante, o el número mínimo de dígitos para un entero.
- Una *h* si el entero será impreso como un *short*, o una l (letra ele) si será como *long*.

Los caracteres de conversión se muestran en la tabla 4.5. Si el caracter después del % no es una especificación de conversión, el comportamiento no está definido.

scanf es la entrada análoga de *printf*, y proporciona muchas de las facilidades de conversión pero en la dirección opuesta.

```
int scanf (char *formato, ...);
```

La instrucción *scanf* lee caracteres de la entrada estándar de acuerdo con las especificaciones dadas en *formato*, y almacena los resultados en los argumentos restantes. Los argumentos que se utilizan después del formato deben ser todos apuntadores y representan las direcciones de las variables en las que se almacenarán las correspondientes entradas.

Esta instrucción se detiene cuando se termina de analizar la cadena de formato o cuando alguna de las variables no corresponde a la cadena de conversión. Regresa el número de ítems de entrada que coinciden con éxito.

CARACTER	TIPO DE ARGUMENTO: IMPRESO COMO
d, i	int: número decimal.
o	int: número octal sin signo (sin cero inicial).
x, X	int: número hexadecimal sin signo (con 0x o 0X inicial, usando abcdef o ABCDEF para 10, ..., 15.
u	int: entero decimal sin signo.
c	int: caracter sencillo.
s	char*: imprime caracteres de una cadena hasta un '\0' o le número de caracteres dado por la presición.
f	double: [-]m.dddddd, en donde el número de d's está dado por la precisión (predeterminada a 6).
e, E	double: [-]m.ddddde±xx o [-]m.ddddddE±xx, en donde el número de d's está dado por la precisión (predeterminada a 6).
g, G	double: usa %e o %E si el exponente es menor que -4 o mayor o igual a la precisión: de otra forma usa %f. Los ceros o el punto al final no se imprimen.
p	void *: apuntador (representación dependiente de la instalación).
%	no es convertido en ningún argumento: imprime un %.

Tabla 4.5. Conversiones básicas de *printf*.

La cadena de formato puede contener caracteres de conversión de la siguiente forma:

- Blancos o tabuladores, los cuales son ignorados.
- Caracteres ordinarios (no %), que se espera coincidan con el siguiente caracter que no sea espacio en blanco del flujo de entrada.
- Especificaciones de conversión, consistentes en el símbolo %, un caracter optativo de supresión de asignación *, un número optativo que especifica el ancho máximo de campo, una h, l, o L optativa que indica la amplitud del objetivo, y un caracter de conversión.

4.2.2.2. Acceso a archivos.

El lenguaje de programación C cuenta con una librería muy completa de instrucciones relacionadas con el manejo de los archivos. Aquí se analizarán las más generales del manejo de archivos, únicamente para conocer la potencialidad de los mismos.

En C, existe una independencia total entre el nombre de un archivo en el sistema operativo y el que será manejado dentro de un programa. Las reglas para el manejo de archivos indican que antes de que un archivo pueda ser leído o escrito, éste debe ser abierto, posteriormente leído, escrito o ambas cosas y al final cerrado.

C maneja sus archivos por medio de apuntadores, que son variables especiales que apuntan a objetos particulares del lenguaje como pueden ser cadenas de texto, variables, funciones, archivos, etc.

Lo primero que debe hacerse es abrir el archivo. Esto se hace con la instrucción *fopen* que se encuentra en la librería <stdio.h>. Lo que hace esta instrucción es asignar el nombre externo del archivo con el nombre con el que se manejará en el programa de forma que exista correspondencia entre ambos. También detectará si es posible que sea abierto el archivo. De no poderse realizar, la instrucción devolverá un apuntador nulo, que involucrará que no se realizó la conexión entre el programa y el archivo almacenado en disco. La forma de declarar un apuntador para archivo es con el tipo de datos *FILE* * el cual contiene información pertinente al uso del archivo. La forma en la que está declarada la instrucción *fopen* y su uso es:

```
FILE *fp;
FILE *fopen(char *nombre, char *modo);
```

Estas instrucciones dicen que *fp* es un apuntador a un *FILE* y que *fopen* regresa un apuntador a *FILE*. Esto hace compatibles la variable con la función. La llamada a *fopen* en un programa es:

```
fp = fopen(nombre, modo);
```

El primer argumento de *fopen* es una cadena de caracteres que contienen el nombre del archivo. El segundo, - también una cadena de caracteres - especifica el modo en el cual será abierto, si éste será para lectura ("*r*"), escritura ("*w*") o añadido ("*a*"). Algunos sistemas reconocen entre archivos de texto y binarios, para estos casos se debe agregar una "*b*" después de la cadena de modo.

Si un archivo que no existe se abre para escribir o añadir, se crea si es posible y si ya existe, toda la información que éste contiene es borrada, si se desea conservar la información es preferible abrirlo para añadir únicamente. Como se mencionó anteriormente el intentar abrir un archivo no existente o que no permite ser abierto provoca un error. Afortunadamente es relativamente sencillo capturar esos errores e impedir que el programa se caiga por ello.

Una vez abierto el archivo se requiere de una forma de leer o escribir información en él. La forma más simple de hacer esto es con las instrucciones *getc* y *putc* que se encargarán de leer y escribir respectivamente, un caracter en un archivo.

```
int getc(FILE *fp);
int putc(int c, FILE *fp);
```

getc regresa el siguiente caracter de la entrada a la que se refiera *fp*; regresa *EOF* (fin de archivo) si ocurre algún error, mientras que la instrucción *putc* escribe el caracter c en el archivo apuntado por *fp* y regresa el caracter escrito o *EOF* si ocurre algún error.

Es probable que se requiera de entradas o salidas con formato como las que se vieron anteriormente. Para realizar esta tarea C cuenta con las instrucciones especificas *fscanf* y *fprintf*, las cuales son idénticas en funcionamiento a *scanf* y *printf*, con la diferencia de que el primer argumento es el apuntador al archivo de entrada o salida.

```
int printf (FILE *fp, char *formato, arg1, arg2, ...);
int scanf (FILE *fp, char *formato, ...);
```

Por último, en el manejo básico de archivos, después de realizar todas las operaciones de entrada y salida necesarias, debe cerrarse el archivo, de otro modo, después de varias corridas del programa marcará errores por la existencia de muchos archivos abiertos. La instrucción que se utiliza para cerrar un archivo es *fclose*:

```
int fclose(FILE *fp);
```

Esta instrucción interrumpe la conexión establecida por *fopen* entre el apuntador de archivo y el nombre externo, liberando el apuntador de archivo para usarse otro.

Básicamente estos son los pasos necesarios para manejar archivos. Existen instrucciones realmente útiles para el manejo de archivos binarios, comúnmente utilizados en las bases de datos. Ellas realizan operaciones de lectura o escritura y movimiento del apuntador de lectura/escritura dentro de los archivos. La correcta utilización de éstas facilita en gran medida en manejo de archivos. Tres de estas instrucciones son:

```
int fseek (FILE *flujo, long desp, int origen);
int fread (void *buf, int tam, int cuenta, FILE *flujo);
int fwrite (const void *buf, int tam, int cuenta, FILE *flujo);
```

La función *fseek*, permite colocar en cualquier lugar del archivo el apuntador de posición del mismo. Esto es especialmente útil cuando se utilizan archivos de acceso aleatorio. El parámetro *desp* controla el número de bytes de desplazamiento del apuntador a partir del *origen* especificado, que puede ser 0 para el inicio del archivo, 1 para la posición actual y 2 para hacer desplazamientos a partir del final del archivo.

La función *fread* lee *cuenta* números de objetos, cada uno de *tam* número de caracteres y los sitúa en el arreglo apuntado por *buf*. Devuelve el número de caracteres que realmente se leyeron.

Por último la función fwrite escribe en el archivo de salida una cantidad *cuenta* de objetos de tamaño *tam* de caracteres. Estos objetos son tomados del arreglo apuntado por *buf*.

4.2.3. ¿Por qué es mejor usar ANSI C?.

Hasta el momento se han presentado una serie de instrucciones útiles para el programador y para la toma de decisiones a la hora de seleccionar el lenguaje de programación para la realización de este sistema.

C cuenta con una gran cantidad de instrucciones que lo hacen ideal para el desarrollo de casi cualquier tipo de aplicaciones. Para el caso particular de este proyecto, las instrucciones para el manejo de archivos son lo más importante en este análisis. Sin embargo no sólo ese aspecto es importante para seleccionar el lenguaje C sobre Pascal, siendo aún que el sistema original está en Pascal, sino que la robustez del lenguaje C lo hacen muy poderoso ya que:

- Cuenta con los tipos de datos suficientes para implementar cualquier tipo de datos de Turbo Pascal (aunque en algunos casos es necesario de algunos trucos de programación para lograr la compatibilidad).

- Maneja campos de bits como variables de tipo entero (esto facilita mucho la decodificación de los tipos de datos de fecha que se manejan en Turbo Pascal y que se encuentran en las bases de datos de Cronos).

- Cuenta con instrucciones para el manejo de archivos mucho más poderosas (permite que los archivos se encuentren en cualquier lugar del disco duro y que estén almacenados bajo cualquier nombre en el sistema de archivos del sistema operativo[5], además permite el acceso aleatorio a los archivos).

- Cuenta con estructuras de control versátiles y poderosas.

- Es un lenguaje de nivel medio, lo que hace de C un lenguaje muy rápido.

En general las características con las que cuenta el lenguaje C permiten tomar la decisión de utilizarlo en el desarrollo del sistema que en este proyecto se presenta.

[5] Se debe recordar que en Pascal estándar los nombres de los archivos deben tener el mismo nombre que las variables a los que estaban relacionados, obligando a que dichos nombres cumplieran con las reglas de construcción de los identificadores.

5

Capítulo V

Procedimientos de recepción de información.

Capítulo V. Procedimientos de recepción de información.

En la mayoría de los desarrollos para el Web se manejan una serie de parámetros que permitirán al sistema saber que tipo de información se debe presentar al usuario. Esto es con el fin de proporcionar sólo la información que el usuario requiere. Para esto se cuenta con algunos métodos comúnmente utilizados en el Web. Estos son:

- El CGI (Common Gateway Interface- Interfaz Común de Puerta de Enlace) que es una forma de programar para el Web aplicaciones del lado del servidor, esto quiere decir, que el CGI es la interfaz entre un servidor con protocolo HTTP (HiperText Transfer Protocol - Protocolo de Transferencia de Hipertexto), y los demás recursos de la computadora host del servidor Web.

- El lenguaje Java™ y Javascript™ que permiten realizar aplicaciones del lado del servidor, pero que en su mayoría se ejecutarán en el lado del cliente. Con Java pueden insertarse aplicaciones llamadas Applets en una página Web. Es un lenguaje multiplataforma a nivel de código fuente y de código binario (bytecodes). Definitivamente un lenguaje muy poderoso, aunque también es muy limitado.

- Los ActiveX™, que son aplicaciones extremadamente potentes, las cuales pueden ser desarrolladas en los nuevos lenguajes visuales compatibles con esta tecnología, entre los que se encuentran Visual Basic™ 5.0 y Delphi™ 3.0. Estas aplicaciones solamente pueden montarse en servidores de Microsoft™ y por el momento, sólo pueden ser visualizadas en el Internet Explorer, de la misma compañía.

No se profundizará demasiado en la explicación del segundo y tercer método, ya que para este proyecto se utilizará el CGI. Unicamente se mencionarán algunos de los motivos por los que no se eligieron dichos métodos:

- El lenguaje Java™ genera código interpretado, lo cual hace que se vuelva un poco lento, además que tarda mucho tiempo en cargarse una página que contenga una de estas aplicaciones (Applets). Requiere de muchos recursos del lado del cliente y no es compatible con todos los navegadores (aunque sí con los más utilizados).

- Los ActiveX™ sólo pueden montarse en servidores de Microsoft™ y únicamente funcionan en el Internet Explorer, lo cual reduce la aplicación a computadoras personales (PC) utilizando Windows™. Se consideró también que el servidor Web está montado en un equipo SUN Sparc Station 4 y no en una PC.

Las características con las que cuenta un CGI lo hace idóneo para la realización de este proyecto, ya que permite de una forma, relativamente sencilla la transferencia de parámetros que permitan seleccionar que información se mandará al usuario.

Para conocer un poco más acerca de como trabaja el CGI se explicarán algunos conceptos y métodos que en esta Interfaz se manejan.

5.1. El CGI.

Como ya se mencionó anteriormente, el CGI es la interfaz entre un servidor con protocolo HTTP y los demás recursos de la computadora en la que se encuentra instalado el mismo, como puede verse en la figura 5.1. El CGI no es un lenguaje de programación, tampoco es un protocolo, sino que son una serie de variables de ambiente con las que se establece comunicación entre el cliente y el servidor. El correcto manejo de éstas variables permite conocer las necesidades del usuario, así como algunos otros parámetros que el cliente manda aún sin saberlo.

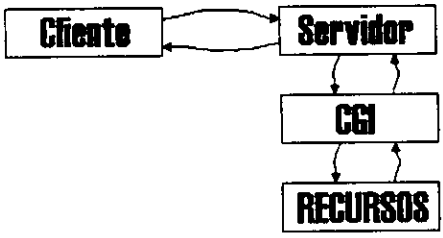

Figura 5.1: Interacción entre el cliente, el servidor y el CGI.

Algo muy importante para manejar el CGI, es tener un servidor Web en donde montar los programas, ya que de otra forma no se podrán ejecutar desde un navegador.

Otra cosa que se necesita para realizar un CGI es un leguaje de programación que sea soportado por el sistema operativo en el que se encuentre instalado el servidor, debe de permitir hacer fácilmente las tareas que de él se requieren, además de ser un lenguaje con el que se esté familiarizado, para no tener problemas a la hora de codificar en él. También es muy importante que el lenguaje seleccionado tenga la capacidad de manejar variables de ambiente, de otro modo será imposible la comunicación entre el CGI y la aplicación.

Normalmente un programa CGI toma la entrada de las variables de ambiente relacionadas y envía resultados al flujo de salida estándar, la cual debe estar en un formato que el navegador pueda interpretar, que por lo general será el HTML estándar, aunque pueden ser una amplia variedad de formatos.

5.1.1. El tipo MIME.

Antes de mandar la información, el programa debe informar al navegador de que tipo de datos se trata. Esto se realiza con la siguiente directiva:

```
Content-type: <Tipo-MIME>
```

Content-type especifica el tipo MIME del flujo que se va a mandar al usuario. Este tipo normalmente será "text/html" para los documentos HTML, y "text/plain" para documentos de texto plano. Aunque los anteriores son los más utilizados, existe una gran lista de tipos MIME que mantienen el formato "tipo/subtipo" de forma que el visualizador pueda tomar decisiones sobre que hacer con un tipo de datos específico.

Una vez mandado el encabezado MIME, debe dejarse un espacio en blanco antes de mandar los datos que componen el tipo, que para el caso de este proyecto será una página Web con la información requerida por el usuario.

Para ver como funciona este proceso, se mostrará a continuación un ejemplo.

5.1.2. "Hola Mundo".

Como es común entre los programadores, generalmente se explica como mandar un mensaje a la pantalla en la mayoría de los lenguajes. Como se mencionó anteriormente, el programa CGI manda la información por la Salida Estándar, por lo que no requiere de un tratamiento especial. Se presenta a continuación el código para mostrar el "Hola Mundo" en un navegador común.

```
/*
Ejemplo: "Hola Mundo"
*/
#include <stdio.h>

main (int argc, char **argv)   {
      /*Se envía el tipo MIME y el espacio en blanco */
      printf("Content-type: text/html\n\n");
      /*Después se manda la información en formato HTML */
      printf("<HTML><HEAT><TITLE>Hola !</TITLE></HEAD>\n");
      printf("<BODY><H 1>Hola, Mundo!</H1></BODY></HTML>\n");
   }
```

Después de compilar el programa se coloca en el servidor, y se llama desde el navegador. Generalmente el lugar adecuado para colocar un programa CGI es en un directorio especial llamado "cgi-bin" que se encuentra en el mismo lugar en que está instalado el servidor, por lo que si el programa se llama "*hola.cgi*" y el nombre del servidor fuera "*informatica.aragon.unam.mx*", entonces la forma de llamarlo desde el navegador sería: "*http://informatica.aragon.unam.mx/cgi-bin/hola.cgi*". Esta dirección electrónica pediría al servidor que se ejecute el programa "*hola.cgi*", dando como salida una página Web conteniendo el mensaje "*Hola, Mundo!*".

Este es un sencillo ejemplo de como crear una página con un CGI. Como puede verse, la salida del presente programa pudo ser cualquier cantidad de líneas de texto HTML, pudiendo con esto hacer páginas muy complejas. Sin embargo, para poder hacer páginas interactivas es necesario que exista una comunicación entre el cliente y el servidor. La forma de hacerlo se explica a continuación.

5.2. Las variables de ambiente.

Como se mencionó anteriormente, el CGI es una serie de variables de ambiente con las que se puede establecer comunicación entre el cliente y el servidor. Dada la situación, es importante saber cuales son estas variables de ambiente, así como la forma de utilizarlas. Algunas de las variables forman parte del estándar, mientras que otras de éstas son específicas del Navegador o de servidores Web particulares.

Existen variables que proporcionan información sobre el usuario, sobre el servidor, y las que son especificas de petición de información, en otras palabras, las de paso de parámetros. A continuación serán analizadas.

5.2.1. Información del usuario.

Se puede obtener información relacionada con el usuario. Las variables que se encargan de ello son: HTTP_USER_AGENT, HTTP_ACCEPT, REMOTE_HOST y REMOTE_ADDR. A continuación se detallan sus características.

- **HTTP_USER_AGENT.**

Esta variable proporciona información sobre el nombre y la versión del navegador con el que se está realizando la petición. En los casos en que la petición pase por algún gateway proxy, la variable incluirá información acerca de éste.

- **HTTP_ACCEPT.**

Cuando la salida de un CGI no va a ser texto plano o un archivo HTML, sino otro tipo de datos, es importante revisar esta variable de ambiente, ya que proporciona los formatos de los tipos MIME que el navegador es capaz de aceptar. Si por ejemplo se quisiera mandar como salida un video, es bueno saber si se cuenta con la capacidad necesaria para desplegarlo. El formato de esta variable es tipo/subtipo, tipo/subtipo, ...

- **REMOTE_HOST y REMOTE_ADDR.**

Estas variables proporcionan la dirección IP, desde la que el usuario realiza la petición. La primera de ellas proporciona el nombre lógico con el que está declarada en el servidor de nombres, mientras que la segunda proporciona el número IP de dicha dirección. En algunos casos en los que no se tenga dada de alta alguna dirección IP en el servidor de nombres, la variable REMOTE_HOST estará en blanco o contendrá el número IP.

5.2.2. Información sobre el servidor.

Las variables de este grupo proporcionan información referente al servidor y al software que ejecuta. Estas variables son: SERVER_SOFTWARE, SERVER_NAME, y GATEWAY_INTERFACE.

- **SERVER_SOFTWARE.**

La variable SERVER_SOFTWARE proporciona el nombre y la versión del software de servidor que está ejecutando la máquina HOST. El formato que utiliza es *nombre/versión.* En ocasiones es importante saber esto, aunque la mayoría de las aplicaciones no utilizan esta información.

- **SERVER_NAME.**

Proporciona el nombre del HOST donde está instalado el servidor, los alias DNS o la dirección IP. Las variaciones en el contenido de esta variable puede ser que el HOST tenga un nombre y un alias distinto en el DNS, o que no se encuentre dado de alta en el mismo. Esta variable es especialmente útil cuando se hacen CGI que funcionarán en más de un servidor. Se puede controlar que solamente funcione para algunos servidores especificados.

- **GATEWAY_INTERFACE.**

Esta variable contiene la versión o revisión de la especificación CGI que está utilizando el servidor Web. El formato que utiliza es *CGI/revisión.*

5.2.3. Información sobre la petición.

Las variables de este tipo cambian de acuerdo a la solicitud específica que se les mande. Son las que permiten conocer lo que el usuario está pidiendo, así como el tamaño en bytes de dichas peticiones. Debido a la cantidad de variables de las que consta, se explicarán conforme se van presentando.

- **QUERY_STRING.**

Esta es la variable de ambiente más importante de los programas CGI, ya que es ésta la que contiene la petición del usuario. En ella se mandan todos los parámetros de forma codificada. Su utilización se explicará con más detalle más adelante.

Existen dos métodos para mandar parámetros a esta variable: el método GET y el POST. Estos se analizarán con detalle en otro apartado.

- **SCRIPT_NAME.**

Esta variable contiene el nombre del programa CGI. Es útil cuando el nombre del programa varía con las ejecuciones.

- **SERVER_PROTOCOL.**

En esta variable se encuentra el nombre y número de revisión del protocolo desde donde viene la solicitud. El formato es *protocolo/revisión.*

- **SERVER_PORT.**

La variable SERVER_PORT contiene información sobre el puerto de entrada de la petición. Cuando se tiene más de un servidor Web en un mismo HOST, es preferible conocer hacia cual de ellos se dirige una llamada. Esta es una forma de administrar mejor los servicios en CGI.

- **PATH_INFO y PATH_TRANSLATED.**

Estas variables contienen otro parámetro para el programa CGI. Con éstas se especifica una ruta adicional que el programa puede utilizar. La forma en la que se le puede mandar una ruta, es agregándola al final de la dirección URL. Por ejemplo, si se agregara a la dirección "*http://informatica.aragon.unam.mx/cgi-bin/hola.cgi/a/b/c*"; en la primera variable se encontrará la cadena "/a/b/c", mientras que en la segunda, se ubicará al final de la ruta raíz del documento en el servidor. Pensando que esta ruta fuera "/WWW" entonces se tendría:

```
PATH_INFO=/a/b/c
PATH_TRANSLATED=/WWW/a/b/c
```

- **CONTENT_TYPE y CONTENT_LENGTH.**

La variable CONTENT_TYPE proporciona el tipo MIME de la consulta. Esta variable se llena cuando la petición va acompañada de información adjunta como en el caso de las solicitudes POST. La variable CONTENT_LENGTH especifica el tamaño en bytes de los datos que acompañan la petición.

- **AUTH_TYPE.**

Cuando se utilizan claves de acceso para usuarios registrados al sitio Web la variable AUTH_TYPE especifica que método de autentificación utilizado para validar al usuario.

- **REQUEST_METHOD.**

Por último, esta variable especifica que método se utilizó para hacer la solicitud. Por el momento los métodos utilizados son GET y POST, los cuales de analizarán más adelante.

Las variables de ambiente que se han presentado hasta el momento son las variables estándar. Algunos navegadores, servidores o páginas Web pueden agregar más variables sin embargo, al ser éstas las estándar, se puede asegurar que existen en todos los servidores Web.

5.2.4. Lectura de variables de ambiente en C.

C es un lenguaje de programación muy poderoso para el manejo de programas CGI. Pero considerando que es un lenguaje de propósito general, y relativamente de bajo nivel, se debe tener cuidado con su uso, ya que un error de codificación puede acarrear serios problemas en el sistema. C cuenta con una instrucción para la lectura de variables de ambiente. Esta instrucción es getenv("VARIABLE _DE_AMBIENTE"), la cual devuelve un apuntador al contenido de la variable de ambiente. Sin embargo, puede devolver un apuntador NULL si no encuentra la variable. Es por ello que antes de usarse debe verificarse que el apuntador no sea NULL. Una forma de remediar el problema es utilizar una macro que se encargue de atrapar ese resultado.

El siguiente código fuente, muestra una forma de leer las variables de ambiente. Este código puede funcionar en cualquier servidor Web[1].

```
#include <stdio.h>
#include <stdlib.h>

/* almacenamiento temporal de la variable de ambiente */
char *cp;
char *empty = "<empty>";

main (int argc, char ** argv) {
/* enviar el encabezado de tipo mime */
printf ("Content-type: text/plain\n\n");

/* macro para desplegar variables de ambiente */
#define safenv(a) ((cp = getenv(a)) ? cp : empty)
```

[1] Código tomado del libro William E. Weinman, "El libro del CGI", editorial : Prentice-Hall, México 1996. Página : 36.

```
/* desplegar todas las variables CGI estándar */
printf("GATEWAY_INTERFACE = %s \n", safenv("GATEWAY_INTERFACE"));
printf("REQUEST_METHOD  = %s \n", safenv("REQUEST_METHOD"));
printf("SCRIPT_NAME = %s \n", safenv("SCRIPT_NAME"));
printf("QUERY_STRING = %s \n", safenv("QUERY_STRING"));
printf("SERVER_SOFTWARE = %s \n", safenv("SERVER_SOFTWARE"));
printf("SERVER_NAME = %s \n", safenv("SERVER_NAME"));
printf("SERVER_PROTOCOL = %s \n", safenv("SERVER_PROTOCOL"));
printf("SERVER_PORT = %s \n", safenv("SERVER_PORT"));
printf("HTTP_USER_AGENT = %s \n", safenv("HTTP_USER_AGENT"));
printf("HTTP_ACCEPT = %s \n", safenv("HTTP_ACCEPT"));
printf("PATH_INFO = %s \n", safenv("PATH_INFO"));
printf("PATH_TRANSLATED = %s \n", safenv("PATH_TRANSLATED"));
printf("REMOTE_HOST = %s \n", safenv("REMOTE_HOST"));
printf("REMOTE_ADDR = %s \n", safenv("REMOTE_ADDR"));
printf("REMOTE_USER = %s \n", safenv("REMOTE_USER"));
printf("REMOTE_IDENT = %s \n", safenv("REMOTE_IDENT"));
printf("AUTH_TYPE = %s \n", safenv("AUTH_TYPE"));
printf("CONTENT_TYPE = %s \n", safenv("CONTENT_TYPE"));
printf("CONTENT_LENGTH = %s \n", safenv("CONTENT_LENGTH"));
}
```

Como puede verse es relativamente sencilla la forma en la que C captura las variables de ambiente. Como se ve, únicamente se obtiene y se verifica que no esté vacía. Esto se hace con la macro safenv(a), que captura la variable y la checa, devolviendo la cadena "<empty>" si no se encuentra la variable.

Una vez compilado en el servidor y puesto en el directorio "cgi-bin" es posible accesar al CGI desde cualquier navegador conectado a Internet.

Así como se muestra el contenido de las variables, es como se puede hacer la manipulación de las mismas para obtener la información que proporciona el usuario. En las siguientes secciones se verá como puede mandar la información.

5.3. Métodos para el envío de la información.

Para que un CGI pueda interactuar con el usuario, es necesario que éste mande los parámetros para que su petición sea procesada. El instrumento utilizado por el usuario es el Navegador o "Browser". En él se pueden hacer formularios que sirven de interface para el usuario. Cada elemento del formulario es relacionado con un nombre de variable y es transmitido al servidor en modo texto y codificado a través de la variable de ambiente *QUERY_STRING*.

Para explicar los métodos que en esta sección se tratarán, se utilizarán algunas de las variables de ambiente explicadas en la sección anterior.

```
#include <stdio.h>
#include <stdlib.h>

/* almacenamiento temporal de la variable de ambiente */
char *cp;
```

```
char *empty = "<empty>";

main (int argc, char ** argv){
/* enviar el encabezado de tipo mime */
printf ("Content-type: text/plain\n\n");

/* macro para desplegar variables de ambiente */
#define safenv(a) ((cp = getenv(a)) ? cp : empty)

/* desplegar variables CGI estándar para el ejemplo */
printf("HTTP_USER_AGENT = %s \n", safenv("HTTP_USER_AGENT"));
printf("REQUEST_METHOD  = %s \n", safenv("REQUEST_METHOD"));
printf("SCRIPT_NAME = %s \n", safenv("SCRIPT_NAME"));
printf("QUERY_STRING = %s \n", safenv("QUERY_STRING"));
printf("REMOTE_HOST = %s \n", safenv("REMOTE_HOST"));
printf("REMOTE_ADDR = %s \n", safenv("REMOTE_ADDR"));
printf("CONTENT_TYPE = %s \n", safenv("CONTENT_TYPE"));
printf("CONTENT_LENGTH = %s \n", safenv("CONTENT_LENGTH"));
}
```

Una vez capturado este código, se compila en el servidor con el nombre de *form-pru.cgi* y está listo para recibir información. Para poder probar el CGI se utilizará una página Web de prueba que aloje los parámetros que se van a utilizar.

```
<HTML>

<HEAD>
<TITLE>Forma de prueba 1</TITLE>
</HEAD>

<BODY>
<H1>Forma de prueba 1</H1>
<HR><BR>

<FORM    METHOD="GET"    ACTION="http://nombre.del.servidor/cgi-bin/form-
pru.cgi">

Por favor introduzca su nombre:<BR>
<INPUT TYPE="TEXT" SIZE=50 NAME="Nombre">
<P>Por favor introduzca su dirección:<BR>
<INPUT TYPE="TEXT" SIZE=50 NAME="Direccion">
<P>Por favor introduzca su E-Mail:<BR>
<INPUT TYPE="TEXT" SIZE=50 NAME="Email">
<P>
<INPUT TYPE="SUBMIT"  VALUE="ENVIAR">
<INPUT TYPE="RESET"  VALUE="LIMPIAR">
</FORM>

</BODY>
</HTML>
```

Después de capturar el documento HTML, se debe cargar en el navegador para poder usarlo. Debe tenerse cuidado de que el atributo ACTION del formulario, apunte a la dirección correcta del CGI.

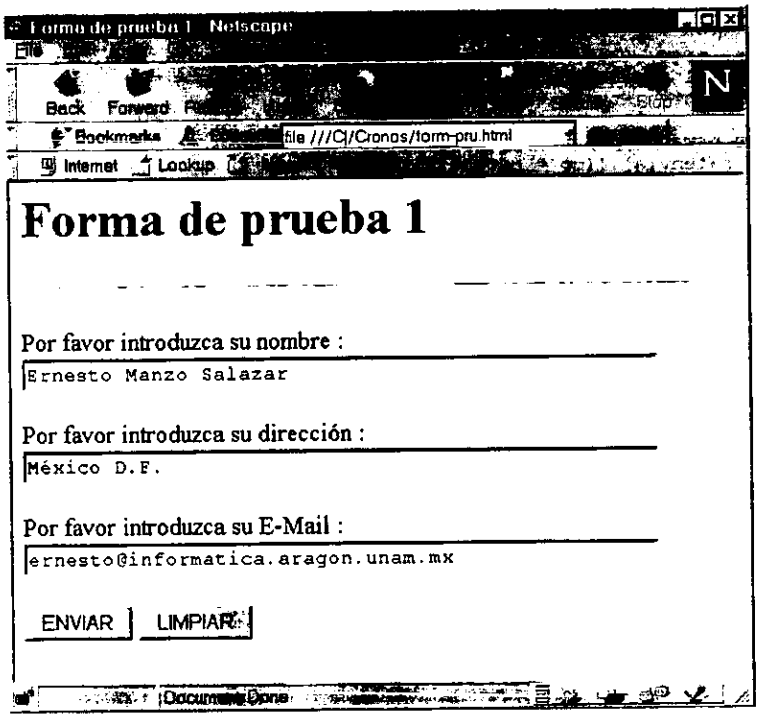

Figura 5.2: Formulario visto desde Netscape

Al oprimir el botón "*ENVIAR*" y si está correctamente capturado, deberá aparecer un resultado similar al siguiente:

```
HTTP_USER_AGENT = Mozilla/4.01 [en] (Win95; I)
REQUEST_METHOD = GET
SCRIPT_NAME = /cgi-bin/form-pru.cgi
QUERY_STRING = Nombre=Ernesto+Manzo+Salazar&Direccion=M%E9xico+D.F.&
Email=ernesto@informatica.aragon.unam.mx
REMOTE_HOST = infor6.aragon.unam.mx
REMOTE_ADDR = 132.248.44.70
CONTENT_TYPE = <empty>
CONTENT_LENGTH = <empty>
```

De ser así se tendrá lo necesario para verificar el funcionamiento de los dos diferentes tipos de transmisión de información: el método GET y el POST.

5.3.1. El método GET.

En el último ejemplo que se realizó, se puede observar que el método de transmisión utilizado es el GET, esto se especifica en la etiqueta <FORM> con el atributo *METHOD="GET"*. Cuando se manda un formulario con el método GET, todo el contenido de los diferentes elementos del formulario se pasan al CGI como parte del URL. Una vez transmitidos estos datos, la información quedará asociada a la variable de ambiente QUERY_STRING, y en la variable REQUEST_METHOD se especificará el método utilizado, que en este caso será GET.

5.3.1.1. La variable QUERY_STRING.

Como ya se ha mencionado la variable QUERY_STRING es en la que se almacenan los datos de consulta. Sin embargo, esta información viene con un formato especial. Éste es un flujo de pares name=value, separados por un signo *ampersand* (&). La parte name es la que se especifica en la etiqueta INPUT del documento HTML.

Codificación de consulta.

"Las cadenas de consulta se codifican de acuerdo con las especificaciones estándar de URL. Esto significa de los caracteres de espacio se codifican como signo mas, y que la mayoría de los caracteres no alfanuméricos se codifican como números hexadecimales, precedidos por el carácter de porcentaje (%)."[2] Ahora bien, como ya conocemos el formato de la cadena QUERY_STRING, se pueden hacer algunas funciones que se encarguen de obtener los diferentes elementos de la cadena en un formato que se pueda utilizar, asignando los valores a variables auxiliares que permitan manipular esa información.

Lo que debe tomarse en cuenta de la cita anterior es que muchos de los caracteres contenidos en el URL pasarán codificados aunque no sea necesario, por lo que en todos los casos se debe considerar la posibilidad de que así sean transmitidos. Ahora bien, como ya conocemos el formato de la cadena QUERY_STRING, se pueden hacer algunas funciones que se encarguen de realizar este trabajo.

5.3.1.2. Decodificación de la consulta utilizando ANSI C.

Como se ha mencionado anteriormente, se puede utilizar casi cualquier lenguaje de programación para desarrollar aplicaciones CGI, lo único que se espera a la hora de seleccionar un lenguaje es que sea conocido por el programador y que además permita desarrollar fácilmente este tipo de aplicaciones. En el capitulo IV se analizaron las características del lenguaje C y del Pascal con el fin de determinar cual era el más adecuado

[2] William E. Weinman, "El libro del CGI", editorial : Prentice-Hall, México 1996. Página : 45

para realizar este proyecto, es por eso que en esta sección solamente se tratará la decodificación de la consulta en lenguaje C.

La decodificación en lenguaje C es ciertamente más rápida que la de otros lenguajes de programación sin embargo, dadas las características de éste se debe tener cuidado con la forma de programar, porque cualquier error puede provocar que se caiga el sistema sobre el que está corriendo. A continuación se presenta un código fuente[3] que permite decodificar la consulta y mostrar el resultado de la misma a través de un navegador.

```
/* Archivo: formtest-2.
(c) 1995 William E. Weinman
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*Esta es la estructura que usamos para las variables CGI */
struct {
  char name[128];
  char val[128];
} elements[16];

/*Almacenamiento temporal de la variable de ambiente*/
char *cp;
char *empty ="<vacio>";

/*Macro para desplegar variables de ambiente*/
#define safenv(a) ((cp = getenv(a)) ? cp : empty)

void splitword(char *out,char *in,char stop) {
   int i,j;
   for (i=0;in[i] && (in[i]!=stop);i++) out[i] = in[i];
   out[i]='\0';
   if (in[i]) ++i;
   for (j=0;in[j]; ) /*Desplazar el resto de in*/
     in[j++]=in[i++];
}

char x2c(char *x) {
   char c;

   /* nota: (x & 0xdf) pone en mayusculas a x */
   c = (x[0]>='A' ? ((x[0] & 0xdf) - 'A') + 10 : (x[0]-'0'));
   c *= 16;
   c += (x[1]>='A' ? ((x[1] & 0xdf) - 'A') + 10 : (x[1]-'0'));
   return (c);
}

/*Esta funcion repasa el URL caracter por caracter y convierte todas
  las secuencias de "escape" (hexadecimales codificados) en caracteres
  Esta version tambien convierte en espacios los signos de mas. */
```

[3] Código construido a partir del presentado en el libro William E. Weinman, "El libro del CGI", editorial : Prentice-Hall, México 1996. Página : 51

```
void unescape_url(char *url) {
   int i,j;

   for (i=0,j=0; url[j];++i, ++j)   {
      if ((url[i] = url[j]) == '%')   {
        url[i] = x2c(&url[j+1]);
        j+=2;
      }
      else if (url[i]=='+') url[i] = ' ';
   }
   url[i] = '\0'; /*terminar en la nueva longitud*/
}

void main(int argc, char **argv) {
   char qs[128]; /*qs es para el QUERY_STRING*/
   int    i;
   char *temporal;

   /* Enviar el encabezado de tipo MIME */
   printf("Content-type: text/plain\n\n");

   printf("formtest-2.c\n");
   printf("Reporte del formtest\n\n");

   /*Desplegar todas las variables CGI estandar*/
   printf("HTTP_USER_AGENT = %s\n",safenv("HTTP_USER_AGENT "));
   printf("REQUEST_METHOD = %s\n",safenv("REQUEST_METHOD"));
   printf("SCRIPT_NAME = %s\n",safenv("SCRIPT_NAME"));
   printf("QUERY_STRING = %s\n",safenv("QUERY_STRING"));
   printf("REMOTE_HOST = %s\n",safenv("REMOTE_HOST"));
   printf("REMOTE_ADDR = %s\n",safenv("REMOTE_ADDR"));
   printf("CONTENT_TYPE = %s\n",safenv("CONTENT_TYPE"));
   printf("CONTENT_LENGTH = %s\n",safenv("CONTENT_LENGTH"));

   /*Asignar la cadena de consulta a qs, o abortar si esta vacia*/
   temporal=safenv("QUERY_STRING");
   if (*temporal != NULL) {strcpy(qs,temporal);}
      else exit(1);

   /*Dividircada uno de los parametros de consulta */
   for (i=0;qs[0] != '\0';i++)
   {  /*primero dividir por '&' cada parametro*/
      splitword(elements[i].val,qs, '&');
      /*Convertir la cadena para caracteres hexadecimales y signos mas */
      unescape_url(elements[i].val);
      /*Separar ahora name y value */
      splitword(elements[i].name,elements[i].val,'=');
   }

   printf("Variables: \n\n");

   /*Imprimir todo*/

   for (i=0;elements[i].name[0];i++)
     printf("%s=%s\n",elements[i].name,elements[i].val);
}
```

En este caso, se manejan tres funciones que se encargan de separar y decodificar los elementos de la consulta, estos son *splitword*(), que separa los elementos; *x2c*(), que traduce

de hexadecimal a caracter y *unescape_url*(), que analiza la consulta para verificar donde se harán las traducciones hexadecimales y la inserción de espacios.

Una vez compilado este código, es posible presentar los elementos de la variable QUERY_STRING al usuario por medio de un navegador, utilizando el formulario anteriormente construido, el resultado del CGI será:

```
formtest-2.c
Reporte del formtest

HTTP_USER_AGENT = <vacio>
REQUEST_METHOD = GET
SCRIPT_NAME = /cgi-bin/formtest-2.cgi
QUERY_STRING = Nombre=Ernesto+Manzo+Salzar&Direccion=M%E9xico+D.F.&
Email=ernesto@informaica.aragon.unam.mx
REMOTE_HOST = infor6.aragon.unam.mx
REMOTE_ADDR = 132.248.44.70
CONTENT_TYPE = <vacio>
CONTENT_LENGTH = <vacio>
Variables:

Nombre=Ernesto Manzo Salzar
Direccion=México D.F.
Email=ernesto@informaica.aragon.unam.mx
```

Este es un ejemplo muy simple de como manejar una consulta GET, sin embargo, proporciona los elementos necesarios para manejar variables como por ejemplo, *Nombre*, *Dirección* y *Email*. Aunque para el caso del presente trabajo, lo que al sistema le puede servir, es el nombre de la carrera, el semestre, o el grupo del que quieran conocer los horarios.

5.3.2. El método POST.

Como se vio en el punto anterior, el método GET utiliza la variable QUERY_STRING para realizar el paso de parámetros, ya que la información es transmitida como parte del URL. Esta información viene en un formato especial que es necesario decodificar para su correcta utilización.

El método POST utiliza el mismo formato para la consulta, con la diferencia de que ésta no es recibida como parte del URL. La consulta para este caso es transmitida al servidor por la *entrada estándar* (STDIN). Cuando se transmite la consulta con el método POST, parte de las variables de ambiente que no se utilizan en el método GET, ahora si son usadas. La variable CONTENT_TYPE recibe el tipo MIME de la consulta transmitida para verificar que sea valida. La variable CONTENT_LENGTH especifica el tamaño de la consulta, es decir, el tamaño de la cadena que debe leerse de la entrada estándar, como no es seguro que se

transmita al final un carácter de terminación de la cadena, es importante solamente leer el número de caracteres ahí especificados.

Para que un formulario pueda transmitir con el método POST, es necesario modificar la etiqueta <FORM>, cambiando el argumento METHOD="GET" por METHOD="POST".

5.3.2.1. Decodificación de la consulta utilizando ANSI C.

La decodificación de una consulta POST en C es prácticamente igual que con el método GET, la diferencia es que ahora, ésta será cargada desde el flujo de entrada estándar. El código[4] siguiente, presenta la forma de leer la consulta y procesar la información.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*Esta es la estructura que usamos para las variables CGI */
struct {
  char name[128];
  char val[128];
} elements[16];

/*Almacenamiento temporal de la variable de ambiente*/
char *cp;
char *empty ="<vacio>";

/*Macro para desplegar variables de ambiente*/
#define safenv(a) ((cp = getenv(a)) ? cp : empty)

void splitword(char *out, char *in, char stop){
   int i,j;
   for (i=0;in[i] && (in[i]!=stop);i++) out[i] = in[i];
   out[i]='\0';
   if (in[i]) ++i;
   for (j=0;in[j]; ) /*Desplazar el resto de in*/
     in[j++]=in[i++];
}

char x2c(char *x) {
   char c;
   /* nota: (x & 0xdf) pone en mayusculas a x */
   c = (x[0]>='A' ? ((x[0] & 0xdf) - 'A') + 10 : (x[0]-'0'));
   c *= 16;
   c += (x[1]>='A' ? ((x[1] & 0xdf) - 'A') + 10 : (x[1]-'0'));
   return (c);
}

/*Esta funcion repasa el URL caracter por caracter y convierte todas
  las secuencias de "escape" (hexadecimales codificados) en caracteres

  Esta version tambien convierte en espacios los signos de mas. */
```

[4] Código construido a partir del presentado en el libro William E. Weinman, "El libro del CGI", editorial : Prentice-Hall, México 1996. Página : 58

```
void unescape_url(char *url){
    int i,j;

   for (i=0,j=0; url[j];++i, ++j)    {
      if ((url[i] = url[j]) == '%')       {
        url[i] = x2c(&url[j+1]);
        j+=2;
      }
      else if (url[i]=='+') url[i] = ' ';
   }
   url[i] = '\0'; /*terminar en la nueva longitud*/
}

void main(int argc, char **argv) {
   char *ct;  /*Para el Content-type */
   char *cl;  /*Para el Content-length */
   int  icl;  /*Content-length */
   char *qs;  /*Query String*/
   int rc;
   int i;

   /* Enviar el encabezado de tipo MIME */
   printf("Content-type: text/plain\n\n");

   printf("formtest-3.c\n");
   printf("Reporte del formtest-3\n\n");

   /*Tomar Content-type y content-length
     y revisar su validez*/

   ct = safenv("CONTENT_TYPE");
   cl = getenv("CONTENT_LENGTH");

   if (cl==NULL)    {
     printf("La longitud de la peticion no esta definida\n");
     exit(1);
   }
   icl = atoi(cl);

   /* Se cuenta con una consulta valida? */

   if (strcmp(ct,"application/x-www-form-urlencoded"))  {
     printf("El Content-type: %s no es soportado\n",ct);
     exit(1);
   }
   else if (icl == 0)    {
     printf("La informacion no esta completa. Favor de reintentar\n");
     exit(1); }

   /*Se asigna la memoria para el flujo de entrada */
   if ((qs = malloc(icl + 1)) == NULL)    {
     printf("No se encontro memoria para realizar la operacion\n");
     exit(1);
   }

   if ((rc = fread(qs,icl,1,stdin)) != 1)    {
     printf("No fue posible leer los datos (%d)! Contacte al WebMaster\n"
            ,rc);
     exit(1);
   }
   qs[icl]='\0';

   /*Dividir cada uno de los parametros del flujo de consulta */
```

```
    for(i=0;qs[0]!='\0';i++)
    {  /*primero dividir por '&' cada parametro*/
       printf("i=%d, qs[0]=%c\n",i,qs[0]);
       splitword(elements[i].val,qs, '&');
       /*Convertir la cadena para caracteres hexadecimales y signos mas */
       unescape_url(elements[i].val);
       /*Separar ahora name y value */
       splitword(elements[i].name,elements[i].val,'=');
    }

    printf("Variables: \n\n");
    /*Imprimir todo*/
    for (i=0;elements[i].name[0];i++)
      printf("%s=%s\n",elements[i].name,elements[i].val);
}
```

La salida del código anterior, después de ser compilada y consultada desde un navegador es:

formtest-3.c
Reporte del formtest

Variables:

Nombre=Ernesto Manzo Salzar
Direccion=México D.F.
Email=ernesto@informaica.aragon.unam.mx

Como se puede ver es necesario verificar más cosas con el método POST, ya que las condiciones en las que se realiza la consulta puede variar mucho, además de que involucra la entrada estándar, que de alguna forma pudiera no estar disponible en algún momento.

5.3.3. El método GET vs POST.

Los dos métodos presentados en este capítulo son igualmente útiles para el paso de parámetros a un programa CGI, sin embargo cuentan con algunas diferencias:

GET

- La consulta pasan al CGI como parte del URL.
- Tiene un tamaño máximo, es decir, si la consulta es muy grande, puede pasar incompleta.
- Permite que el usuario vea el contenido de la consulta, dificultando la inclusión de campos invisibles.
- Es más fácil de manejar, ya que todo se obtiene de la variable de ambiente QUERY_STRING.

POST

- La consulta pasa a través de la entrada estándar.
- El tamaño es fijado por la cantidad de información que el servidor pueda leer en una variable.
- Oculta por naturaleza el contenido de las consultas, de modo que es posible utilizar campos ocultos.
- Es más complicado su manejo, al ser necesario apartar memoria para leer la consulta, además de verificar más variables de ambiente para la validación de la consulta.

- Permite mandar consultas sin necesidad de pasar por el formulario correspondiente, poniendo en riesgo el orden en que deben leerse las variables, o la inclusión de variables no existentes para el CGI.
- Aunque es posible manejarlos desde otros formularios, hace necesaria la existencia de uno para la transmisión de la consulta, permitiendo con esto tener un mayor control de la forma en que se hará el paso de parámetros.

Otra forma de ver las diferencias es por medio de diagramas de flujo que muestran la forma en la se debe desarrollar un programa CGI con cualquiera de los dos métodos:

5.3.3.1. Diagrama del método GET.

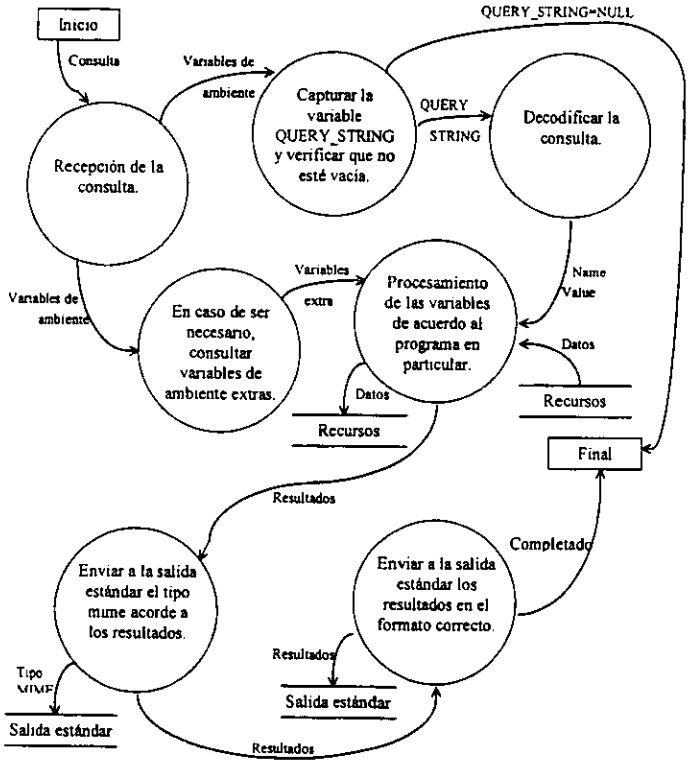

Figura 5.3: Diagrama de flujo del método GET.

En la figura 5.3 se muestra la metodología básica para el desarrollo de un CGI con el método GET. Cada uno de los elementos tiene un significado particular:

○ Representa un proceso, que puede ser simple o compuesto.

→ Representa un flujo, que puede ser simple o compuesto.

▭ Representa el inicio y el fin de todo el diagrama.

═══ Representa un archivo. En este caso en particular representa un recurso del sistema.

Como se puede observar se representan todos los pasos tomados en el código del mismo método.

5.3.3.2. *Diagrama del método POST.*

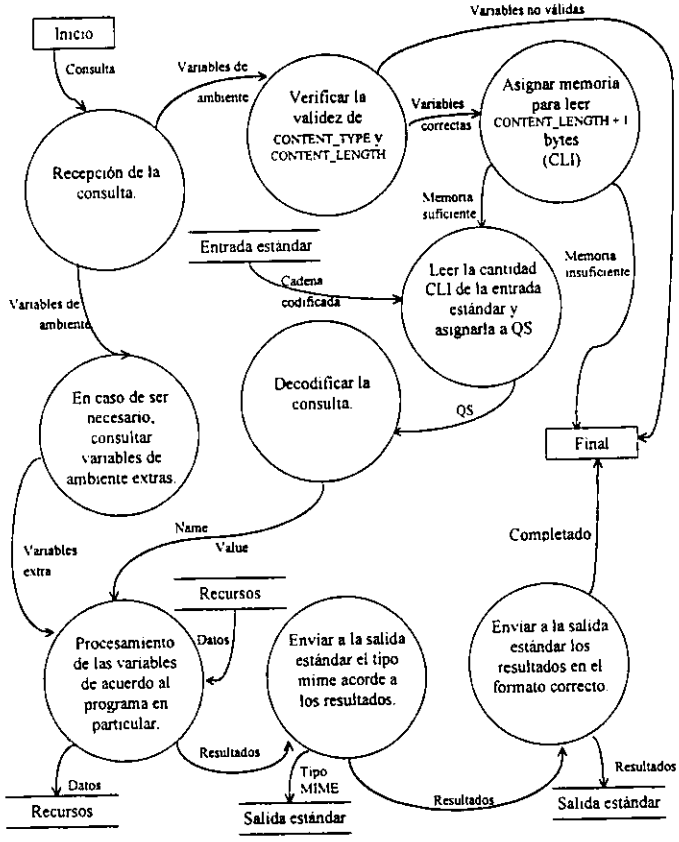

Figura 5.4: Diagrama de flujo del método POST.

La figura 5.4 muestra la metodología básica para el desarrollo de un CGI utilizando el método POST. Como puede verse la forma de manejar los dos tipos de consultas no tiene mucha variación, sin embargo las características con las que cuentan las hacen más adecuadas para alguna aplicación en particular. La elección del método debe ser considerado por el programador a fin de que se cumplan los objetivos del programa que se esté realizando.

5.4. Elementos de la etiqueta FORM del HTML.

Hasta el momento se han analizado los dos métodos de recepción de la información, sin embargo no se ha dicho mucho sobre los formularios que utilizan los usuarios para transmitirla. En el siguiente capítulo se verán otras etiquetas con las que cuenta el HTML, pero a continuación se presentará la sintaxis de los elementos con los que se construye un formulario en forma básica.

```
<FORM ACTION={URL} METHOD={"GET" | "POST"}>
```

Cualquier código HTML valido puede ser incluido entre <FORM> y </FORM> excepto otra etiqueta <FORM>.

```
<INPUT [TYPE="text"] NAME={nombre} [SIZE = {número}] [VALUE={default}]
 [MAXLENGTH={numero}]>

<INPUT [TYPE="password"] NAME={nombre} [SIZE = {número}]
[VALUE={default}] [MAXLENGTH={numero}]>

<INPUT [TYPE="checkbox"] NAME={nombre} [VALUE={valor}] [CHECKED]>

<INPUT [TYPE="radio"] NAME={nombre} [VALUE={valor}] [CHECKED]>

<INPUT [TYPE="hidden"] NAME={nombre} [VALUE={valor}]>

<INPUT [TYPE="submit"] NAME={nombre} [VALUE={etiqueta}] >

<INPUT [TYPE="reset"] [VALUE={etiqueta}>

<INPUT [TYPE="image"] NAME={nombre} SRC= {URL} [ALIGN={alineación}]>

<SELECT  NAME={nombre} [SIZE = {número}] [MULTIPLE]>
  <OPTION [SELECTED] [VALUE={valor}]>{Texto}
  [<OPTION [SELECTED] [VALUE={valor}]>{Texto} ...]
</SELECT>

<TEXTAREA  NAME={nombre}  ROWS = {número}  COLS = {número}>
     Texto por default
</TEXTAREA>
```

En donde los diversos parámetros que se presentan se muestran a continuación:

Parámetro	Descripción
{default}	Valor predeterminado.
{etiqueta}	Rótulo con el que aparecerá el botón.
{nombre}	La nombre que tendrá el elemento, es decir, la parte *name* del par *name/value.*
{número}	Valor numérico para definir tamaños normalmente.
{texto}	Texto en el elemento SELECT.
{url}	Un URL válido.
{valor}	La parte *value* de un elemento.

La utilidad de cada uno de los controladores utilizados en un formulario se muestra en la tabla siguiente:

Control	Descripción
TEXT	Es el tipo predeterminado de la etiqueta INPUT. Crea un cuadro de texto en el que el usuario puede escribir.
PASSWORD	Igual que TEXT, pero con la diferencia de que al escribir, los caracteres originales se ocultan con asteriscos.
CHECKBOX	Despliega una casilla de verificación, el cual transmite su par name/value, sólo si está seleccionado.
RADIO	Proporciona botones de radio que permiten elegir, una de entre varias opciones. Únicamente se transmite la opción seleccionada.
HIDDEN	Permite ocultar la transmisión de un parámetro, sólo se transmite el nombre y el valor. No despliega nada en pantalla.
SUBMIT	Proporciona un botón, que al ser oprimido transmite la consulta generada por el formulario al que pertenezca.
RESET	Pone en condiciones iniciales todos los controles del formulario.
IMAGE	Coloca una imagen que funcionará como un botón SUBMIT, con la diferencia de que se transmitirán las coordenadas de la imagen en la que el usuario presionó el botón del ratón.
SELECT	Crea una lista de selecciones alternativas, la cual puede ser múltiple o simple.
TEXTAREA	Crea un cuadro de texto de varias líneas.

Estos son los controles que se pueden manejar en un formulario, la correcta combinación de estos elementos y un buen diseño HTML permitirá hacer una buena interface de usuario para la transmisión de las consultas al programa CGI.

6

Capítulo VI

Presentación de la información al usuario.

Capítulo VI. Presentación de la información al usuario.

La parte más importante en el desarrollo de todo sistema son los resultados, es el objetivo final por el esfuerzo de programación. De la calidad y claridad de ellos dependerá el éxito o fracaso del sistema en general.

Es por ello que en este capítulo se pretende mostrar la forma de presentar la información al usuario del sistema que aquí se desarrolla. En el capítulo anterior se revisaron los procedimientos de recepción de la información a través de una página Web y dirigido a un CGI, mediante la utilización de formularios. Sin embargo no se revisó la forma en la que las páginas son generadas ni el formato de un documento HTML (HiperText Markup Languaje - Lenguaje para Marcado de HiperTexto). Es importante hacerlo, ya que los resultados serán generados en HTML.

6.1. ¿Qué es el HTML ?

El HTML como su nombre lo dice es un lenguaje de marcado de hipertexto. Es un lenguaje específico para el diseño de páginas, ya que cuenta con instrucciones para el formato de texto, imágenes, tablas y otros elementos multimedia, indispensables para presentar información en el Web.

El HTML está basado en el SGML (Lenguaje Estándar de Marcación Generalizada - Standard Generalized Markup Language) que es un sistema mucho más grande de procesamiento de documentos. El SGML se utiliza para describir la estructura general de varios tipos de documentos. A diferencia de otros lenguajes de manejo de documentos como el PostScript, el SGML y por tanto el HTML tienen como objetivo principal el contenido del documento y no su apariencia.

6.1.1. El HTML describe la apariencia de un documento.

El HTML al igual que el SGML, es un lenguaje para describir *documentos estructurados*. Esto quiere decir que se centra en los elementos comunes que todos los documentos tienen, como pueden ser títulos, párrafos, listas, etc. Si se enumeran todos los elementos con los que cuenta un documento, entonces se puede dar un identificador a cada uno. A estos identificadores se les conoce como *tags* o etiquetas.

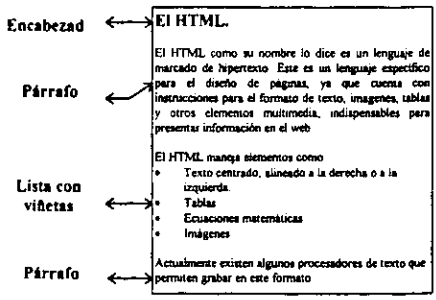

Figura 6.1: Elementos de un documento.

6.1.2. El HTML no describe el diseño de la página.

Una de las características de las hojas de estilo y de las plantillas de procesadores de texto que el HTML no tiene, es que proporciona la apariencia de cada parte del documento sobre la página o la pantalla. Al momento de capturar el documento se especifica exactamente la forma que va a tener. Los procesadores de texto actuales permiten ver previamente como saldrá impreso un documento y en cualquier computadora podrá ser visto de la misma manera.

Salvo algunas excepciones, HTML no cuenta con etiquetas invariables. Lo importante de un documento HTML es el contenido, es decir la información y no la forma en la que se estructure en la pantalla. Esto es debido a que el visualizador Web es el que interpreta la página y la presenta al usuario de acuerdo a la resolución de su monitor, el tamaño de la ventana de presentación, tipos de letra, etc.

Como existen diferentes visualizadores, cada uno de ellos con capacidades diversas, resulta muy complicado elaborar páginas que se vean iguales en todos ellos. Es por eso que el diseñador de páginas debe concentrarse más en el contenido de un documento que en su forma, únicamente respetando las estructuras.

6.1.3. Ventajas del HTML.

Actualmente la mayoría de los editores de texto manejan el concepto de WYSIWYG (lo que se ve, es lo que se tiene). HTML no maneja este concepto en el momento de diseñar sus páginas. Sin embargo, aunque este sistema parezca tan limitado, tiene ciertas ventajas sobre otras formas de edición:

- Los documentos HTML son muy pequeños, motivo por el cual la transferencia en Internet es rápida. Esto es gracias a que no es necesario transmitir fuentes de letra o información para formatear que retrase la transmisión.
- Los documentos son compatibles entre plataformas. Esto quiere decir que puede leerse en cualquier computadora con cualquier sistema operativo, siempre y cuando se cuente con un visualizador compatible con la plataforma, el cual puede estar en modo texto o en modo gráfico, tener o no algún tipo de letra, un monitor en color o monocromático. Por todo esto el lenguaje HTML es altamente portable.

Una ventaja más del HTML es que, aunque sea un lenguaje de marcación, es especialmente pequeño y fácil de aprender. Actualmente existe una gran variedad de editores que permiten manejar con mayor facilidad este formato. Entre ellos se encuentran los que acompañan al los visualizadores: Netscape Comunicator o Internet Explorer.

6.2. ¿Cómo se crean los archivos HTML?

Los documentos escritos en HTML se encuentran en texto simple (ASCII) y contienen dos cosas:

- El texto del documento mismo.
- Etiquetas HTML que indican los elementos, estructura y formato del documento, así como vínculos de hipertexto a otros documentos o a los medios que se incluyen.

La mayoría de las etiquetas HTML son más o menos así:

<Nombre_de_la_etiqueta>Texto afectado por la etiqueta</ Nombre_de_la_etiqueta>

El nombre de la etiqueta se encierra entre paréntesis angulares (<>).

Las etiquetas HTML tiene por lo general una etiqueta inicial y una final, las cuales rodean al texto que van a afectar. La etiqueta inicial "activa" alguna característica (encabezados, letras negritas, etc.) y la etiqueta final la "desactiva". Las etiquetas de cierre tienen el mismo nombre que las de apertura con la diferencia de que las precede una diagonal (/).

Sin embargo, no todas las etiquetas tienen principio y final. Algunas son de un sólo lado y otras son "contenedoras" puesto que tiene información adicional y texto dentro de los paréntesis angulares.

Todas las etiquetas HTML son insensibles al tipo de letra. Esto quiere decir que por ejemplo *<HTML>* es igual a *<html>* o incluso a *<hTmL>*. La forma de escribir las etiquetas es decisión del diseñador, aunque es recomendable utilizar siempre un mismo estilo para evitar confusiones o para dar claridad al código.

6.3. ¿Cómo se estructura el HTML?

HTML define tres etiquetas que se usan para describir la estructura general de un documento. Sirven para que el visualizador o alguna otra herramienta para Web identifique el documento. No modifican la apariencia del documento, únicamente se usan como apoyo para el visualizador. Aunque en algunos de ellos no es necesario que los documentos incluyan estas etiquetas, poco a poco se van volviendo más necesarias para mantener compatibilidad entre diferentes versiones, aunque en la definición estricta del HTML son opcionales.

<HTML> es la primera etiqueta de estructura. Esta etiqueta especifica el inicio y el fin de todo el documento. Todo el texto y etiquetas con que cuente el código deben estar entre <HTML> y </HTML>.

<HEAD>..</HEAD> Especifica que las líneas dentro de los puntos de inicio y final de la etiqueta son el prólogo del resto del documento. No hay muchas que se utilicen en esta sección, sin embargo en ésta se especifica el título de la página, el cual aparecerá en la barra de título de la ventana del navegador. Para ponerlo se utiliza la etiqueta <TITLE>.

<BODY>.. </BODY> Entre el inicio y el fin de esta etiqueta se coloca el resto de la página Web, entre ellos: tablas, párrafos, imágenes, ligas, listas, formas, etc.

Por lo tanto la estructura general de un documento en HTML debe ser como se muestra:

```
<HTML>
<HEAD>
<TITLE>Este es el t&iacute;tulo de la p&aacute;gina</TITLE>
</HEAD>
<BODY>
   . . . . .
</BODY>
</HTML>
```

En este ejemplo se puede observar que para colocar acentos en HTML se debe utilizar la vocal a acentuar de la siguiente manera: á é í ó &uacute, esto se hace para mantener la compatibilidad con otros sistemas operativos.

6.3.1. El título.

Cada documento HTML debe de contener un título, aunque no sea estrictamente necesario, es recomendable ponerlo en todas las páginas que se realicen.

Como se mencionó anteriormente con la etiqueta <TITLE></TITLE> se puede poner título a una página. Es importante que al hacerlo se utilice un nombre coherente con el contenido del documento.

Un documento solamente puede contener un título y dentro de él, sólo se puede incluir texto simple (o códigos de caracteres especiales), es decir, que no puede contener otras etiquetas dentro de él.

6.3.2. Encabezados.

Los encabezados se utilizan para dividir secciones de texto, como en un libro. Estos encabezados están definidos con las etiquetas <H1> ... <H6> introduciendo sus respectivos inicios y finales. Al usarlos se proporciona un cierto nivel al texto afectado. Por ejemplo, los títulos de cada sección de este capítulo pueden ser representados con encabezados: el título principal utilizaría el <H1>, el segundo <H2>, y así sucesivamente. Cada encabezado cuenta con un tamaño o un tipo de letra diferente, en algunos navegadores pueden predefinirse los tipos de letra, tamaños y atributos que utilizará al encontrar estas etiquetas.

En el caso de navegadores en modo texto, cada encabezado se distinguirá por una sangría de diferente profundidad.

6.3.3. Párrafos y saltos de línea.

Los párrafos son lo que más se encuentra en un documento. Estos son declarados con la etiqueta <P>. Existen tres formas de ser interpretados, dependiendo del visualizador que se esté utilizando. Los primeros navegadores utilizaban <P> al final de un párrafo para agregar un retorno de carro y un salto de línea. Esta etiqueta por tanto era de un solo lado sin contener inicio y fin. En otros casos esperaban que la etiqueta estuviera al principio y no se cerraba al final. Sin embargo, con la llegada del HTML 2.0 se acordó utilizar ahora el inicio y el fin para dar claridad, por lo que actualmente se usa el <P>...</P>. El problema de esto es que los viejos navegadores al encontrar <P> al principio del párrafo insertarán un retorno de carro y salto de línea. Esto puede representar un inconveniente en algunos casos, pero es recomendable utilizar la definición 2.0 o superior, ya que los viejos navegadores ya casi no son utilizados, pero es bueno tenerlos en cuenta.

Cuando se desea insertar un salto de línea la etiqueta que permitirá hacer esto es:
. Esta etiqueta es de un solo lado. Cuando el visualizador la encuentra inmediatamente provoca un retorno de carro y salto de línea.

6.3.4. Comentarios.

Los comentarios son muy útiles en todos los lenguajes de programación para evitar perderse entre el código. Permiten documentar el código fuente. Esto es especialmente útil cuando se tienen documentos muy grandes. HTML también cuenta con este recurso por las mismas razones.

En HTML un comentario se estructura como:

```
<!-- Este es el comentario -->
```

Cada línea a ser comentada debe de tener su propio par de inicio y fin.

Con todo esto ya es posible generar documentos desplegables, pero muy sencillos. No es función de este capitulo mostrar todas las etiquetas con las que cuenta el HTML, sobre todo considerando su enorme evolución con las constantes revisiones a los estándares. Sin embargo se presentarán las que son necesarias para la elaboración de este sistema.

6.4. Etiquetas de utilidad para el sistema.

Obviamente en cada sistema de páginas Web que se desarrolla en los múltiples sitios existentes, se utilizan las etiquetas necesarias para cada caso. No hay una regla de que etiquetas usar y cuales no, simplemente se deben manejar las necesarias.

6.4.1. Vínculos.

Una de las etiquetas más usadas y de las más importantes es el "ancla" <A>...</A>. El ancla es utilizada para realizar vínculos entre una página y otra, o con algún otro recurso o servicio (gopher, telnet, ftp, etc.). Para hacerlo es necesario introducir el URL (Localizador Uniforme de Recursos) correspondiente. Esta etiqueta requiere de otros parámetros para realizar su función. A este tipo de etiquetas se les denomina compuestas. Los parámetros de mayor utilidad en esta etiqueta son los siguientes: HREF, NAME, TITLE.

De todos estos parámetros, el más importante es HREF (Hipertext REFerence) ya que en él se especifica el URL al que se debe hacer el enlace.

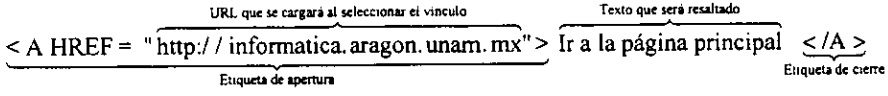

Figura 6.2: Vínculo a un documento externo.

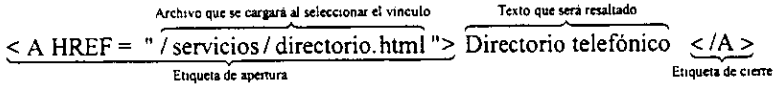

Figura 6.3: Vínculo a un documento dentro del servidor actual.

En el ejemplo anterior se muestra la utilización típica de la etiqueta <A>. El URL al que apunta HREF puede ser cualquiera que sea valido. Este puede ser una página contenida dentro del mismo servidor, una archivo de otro servidor, o una sección dentro del mismo documento, así como otros servicios.

Cuando se desea hacer referencia a otra parte del documento, o a una parte específica de algún documento, es necesario crear un "ancla" a esa parte del código. Esto se hace con el parámetro NAME. Primero es necesario dar un nombre al lugar en el que se desea el ancla y después hacer referencia a ella de la misma forma que se mostró en las figuras 6.2 y 6.3. Suponiendo que se tiene una página llamada "tesis.html" y dentro de ella se encuentra una sección llamada "Capítulo VI" y se quiere hacer referencia a esa sección, será necesario colocar un ancla en esa parte del documento como se muestra en la figura 6.4.

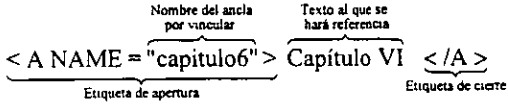

Figura 6.4: Ancla en el archivo "tesis.html".

Después se podría hacer referencia a ella mediante la instrucción mostrada en la figura 6.5. desde cualquier página Web, incluyendo la misma página en la que se encuentra el ancla,

para lo cual no será necesario especificar el nombre del archivo, sino simplemente el nombre del ancla precedida del signo de número (#).

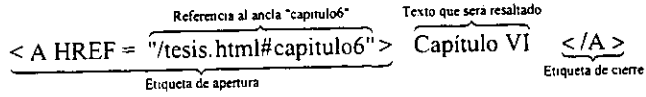

Figura 6.5: Referencia al ancla "capitulo6" en "tesis.html".

El parámetro TITLE no es muy utilizado, sin embargo permite en algunos visualizadores mostrar un recuadro con el texto asignado a dicho parámetro en el momento de pasar el puntero del mouse sobre el texto resaltado, de forma que puede utilizarse para dar una breve explicación de lo que se encontrará al seleccionar ese vinculo.

6.4.2. Listas.

Las etiquetas de lista son especialmente útiles para ordenar partes del texto de forma que aparezcan alineadas como lo que son: "listas". Existen de varios tipos: las ordenadas, desordenadas o con viñetas, las de menú, directorio y las de glosario. Las listas tienen una etiqueta de especificación con su correspondiente inicio y fin, y cada elemento que incluye tiene únicamente la etiqueta de inicio. El navegador asume que se termina el texto del elemento cuando encuentra otro elemento de lista o ésta se cierra. Para que esto quede más claro se pondrán algunos ejemplos.

6.4.2.1. Listas ordenadas.

La lista ordenada tiene la característica de mostrar numerada de forma ascendente todos sus elementos. Para hacerla se utiliza la etiqueta <OL>...</OL> y para cada elemento la etiqueta <LI>.

```
<OL>
<LI>Primer elemento.
<LI>Segundo elemento.
<LI>Tercer elemento.
</OL>
```

Cuando un navegador analiza el código presentará:

1. Primer elemento.
2. Segundo elemento.
3. Tercer elemento.

Como puede observarse, a cada elemento de la lista se le agregó un número de forma ascendente. Cabe mencionar que el navegador no separa los elementos por el hecho de que

estén en diferente línea, sino que lo hace porque encuentra una siguiente etiqueta <LI> o el final de la lista. A cada elemento de la lista se le pueden agregar otras etiquetas para dar formato o algún vínculo como los mostrados anteriormente.

6.4.2.2. Listas desordenadas.

Este tipo de lista es básicamente igual a la anterior, con la diferencia de que en lugar de aparecer números aparecerán viñetas. Los elementos aparecerán en el mismo orden en el que fueron capturados. Para este tipo de listas se utiliza la etiqueta <UL>...</UL> y para los elementos <LI>.

```
<UL>
<LI>Primer elemento.
<LI>Segundo elemento.
<LI>Tercer elemento.
</UL>
```

Cuando un navegador analiza este código presentará:

- Primer elemento.
- Segundo elemento.
- Tercer elemento.

6.4.2.3. Listas de menú y de directorio.

Los menús son listas de elementos o párrafos cortos, en los que no se presentan viñetas ni números ni cualquier otro tipo de marca. Este tipo de lista desaparece en la especificación HTML 3.0 debido a que no son muy utilizadas, puesto que en algunos navegadores no se hace diferencia entre lista de menú y lista desordenada. La etiqueta que utiliza es <MENU>...</MENU> y de cada elemento es <LI>.

La lista de directorio es muy parecida a la anterior, pero ésta fue pensada para mostrar información sobre los directorios de archivos y ser desplegados de forma horizontal. Sin embargo, la mayoría de los visualizadores la tratan como una lista sin orden. También desaparece en la especificación 3.0 del HTML. La etiqueta asociada es <DIR>...</DIR> y cada elemento con <LI>.

6.4.2.4. Listas de glosario.

Una lista de glosario o de definición tiene entre sus elementos uno que representará el concepto y otro su definición, aunque puede ser utilizado para cualquier otro propósito que use el mismo formato. Para este tipo de lista se tiene la etiqueta <DL>...</DL>. Para cada elemento de concepto se tiene <DT> y para la definición <DD>.

```
<DL>
<DT>HTML: <DD>HiperText Markup Language.
<DT>URL: <DD>Uniform Resource Locator.
<DT>WWW: <DD>World Wide Web.
</DL>
```

Al ser cargado este código en un navegador se presentará algo como lo siguiente:

HTML:
    HiperText Markup Language
URL:
    Uniform Resource Locator
WWW:
    World Wide Web

En caso de ser necesario anidar elementos en una lista, solamente se necesita que al definir los elementos, se coloque otro par de inicio y fin de lista con sus respectivos elementos. De esta forma, los elementos de más adentro aparecerán con una sangría de mayor tamaño.

6.4.3. Estilos para caracteres.

Cuando se utilizan etiquetas de HTML para párrafos, encabezados y listas, esas etiquetas afectan al texto en su totalidad, cambiando el tamaño del texto, el tipo de letra, los espacios por encima y por debajo de la línea, etc. Cuando se utilizan los estilos para caracteres, se puede cambiar la apariencia de otras partes del texto, de forma que resalte del resto del texto al rededor.

Existen dos tipos de estilos: los lógicos y los físicos, los primeros indican la forma en que se utiliza el texto; mientras que los segundos indican la forma exacta en que debe verse.

6.4.3.1. Estilos lógicos.

Los estilos lógicos indican cómo se va a emplear determinado texto realzado, no cómo se va a desplegar. Estos funcionan de forma similar a las etiquetas de elementos comunes para párrafos o encabezados. Además, estos no indican como será formateado el texto, sino en qué forma se van a utilizar: citas, códigos, variables, definiciones, etc. Cada visualizador determina la forma en que realizará el formateo de la información, por lo que si en un caso el texto aparece en cursivas, no se puede estar seguro que en otro navegador sucederá lo mismo. Estas etiquetas son recomendables únicamente en casos en que no se requiera que el texto aparezca de cierta manera.

Actualmente existen ocho etiquetas lógicas:

1. <EM>...</EM>: indica que los caracteres aparecerán enfatizados de alguna manera, es decir, que aparecerán con un formato distinto al resto del texto. En los visualizadores gráficos generalmente aparecen en cursivas.
2. <STRONG>...</STRONG>: con esta etiqueta los caracteres tendrán mayor énfasis que con <EM>. en muchos casos el texto termina en negritas.
3. <CODE>...</CODE>: permite mostrar código fuente. En los visualizadores gráficos se usa generalmente alguna fuente monoespaciada, como Courier.
4. <SAMP>...</SAMP>: ejemplo de texto. Similar a <CODE>.
5. <KBD>...</KBD>: para texto que el usuario deba escribir.
6. <VAR>...</VAR>: nombre de alguna variable o entidad que deba ser remplazada por su valor real.
7. <DFN>...</DFN>: para definiciones.
8. <CITE>...</CITE>: para citas cortas.

6.4.3.1. Estilos físicos.

Los estilos físicos especifican la forma exacta en la que el texto debe formatearse. Obviamente la forma real de como aparecerá la información depende del navegador, pero al menos en los visualizadores gráficos se respetan los estilos definidos. En la especificación HTML 3.0 existen cuatro tipos definidos de estilos físicos:

- <B>...</B>: para una fuente en negritas.
- <I>...</I>: para fuentes en itálica o cursiva.
- <TT>...</TT>: para texto monoespaciado, como el de las maquinas de escribir.
- <U>...</U>: para texto subrayado.

Al combinar estas etiquetas se pueden obtener en algunos casos la combinación de los atributos, por ejemplo si se utiliza:

```
<B><I>Texto en negritas y cursivas</I></B>
```

el texto será presentado como:

Texto en negritas y cursivas

6.4.4. Tablas.

Las tablas son los elementos más importantes del sistema que esta tesis presenta, ya que con ellas se presentará la información referente a los horarios. Aquí se mostrará la forma de utilizarlas y los parámetros con los que cuenta, así como las etiquetas que interactúan en ellas.

Una tabla <TABLE>...</TABLE> se compone de renglones <TR>...</TR> y cada renglón está separado en celdas. Las celdas pueden ser de dos tipos: celdas normales <TD>...</TD> o celdas de encabezado <TH>...</TH>, que son iguales que las anteriores pero con el texto resaltado de alguna manera. Es importante que el número de celdas en cada renglón sea el mismo, ya que con ellas se forman y alinean las columnas. La tabla puede contener también un título <CAPTION>...</CAPTION>.

Ahora bien, una tabla puede ser personalizada utilizando una serie de parámetros que permiten, configurar los bordes, la alineación de los textos de las celdas, el color del fondo, etc.

La etiqueta <TABLE> permite definir una tabla. Los parámetros que maneja son[1]:

- ALIGN — Establece la alineación de la tabla mediante ALIGN=LEFT o ALIGN=RIGHT.
- BGCOLOR — Establece el color de fondo de las celdas de una tabla.
- BORDER — Determina el ancho del borde, en pixeles. Si no se utiliza la tabla no tendrá borde.
- BORDERCOLOR — Asigna el color predeterminado de un borde.
- BORDERDARK — Determina el color de la parte obscura de un borde de tres dimensiones.
- BORDERLIGHT — Asigna el color de la parte clara de un borde de tres dimensiones.
- CAPTION — Especifica el título de la tabla.
- CALLPADDING — Establece la cantidad de espacio libre junto al contenido de una celda.
- CELLSPACING — Asigna la cantidad de espacio entre las celdas de una tabla.
- WIDTH — Determina el ancho de la tabla en pixeles o en porcentaje de la ventana del navegador.

Entre la etiqueta de inicio y fin de tabla se deben definir los renglones. Cuando se define un renglón se pueden dar una serie de parámetros, que son:

- ALIGN — Establece la alineación del texto de las celdas del renglón mediante ALIGN=LEFT o ALIGN=RIGHT.
- BGCOLOR — Establece el color de fondo de las celdas de todo el renglón.

Los renglones deben de tener celdas, que pueden ser de encabezado o normales. Los parámetros que se pueden usar en ambos casos son los mismos, los cuales son:

[1] No todos los navegadores reconocen la totalidad de estos parámetros, sin embargo se espera que los más importantes los acepten totalmente.

- ALIGN — Establece la alineación del texto contenido en la celda mediante ALIGN=LEFT o ALIGN=RIGHT.
- BGCOLOR — Establece el color de fondo de la celda
- COLSPAN — Establece cuantas celdas en las columnas cubre la celda a la que se le aplica.
- NOWARP — Evita que el texto se acomode conforme cambia en ancho de una celda, por lo que si el texto colocado abarca más que el ancho de la columna, ésta aumentará su tamaño horizontal.
- ROWSPAN — Permite especificar cuantos renglones ocupará la celda a la que se le aplica. Esto es especialmente útil cuando se quieren hacer tablas con características de cuadros sinópticos.
- VALIGN — Establece la alineación vertical de la celda utilizando VALIGN=TOP, VALIGN=MIDDLE, VALIGN=BOTTOM.
- WIDTH — Determina el ancho de la celda en pixeles o en porcentaje del tamaño total de la tabla.

Como ejemplo del funcionamiento de una tabla, se presenta a continuación el código y la salida del mismo para una tabla de colores.

```
<HTML>
<HEAD>
<TITLE>Tabla de ejemplo</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF>
<TABLE ALIGN=CENTER BORDER=5 CALLPADDING=5 CELLSPACING=5 WIDTH=70%>
<CENTER><CAPTURE>Tabla de colores</CAPTURE></CENTER>
<TR BGCOLOR=#CCCCCC>
<TD><BR></TD>
<TH COLSPAN=4>Color</TH></TR>
<TR ALIGN=CENTER BGCOLOR=#CCCCCC>
<TH ROWSPAN=4 ALIGN=CENTER>Color</TH>
<TH><BR></TH>
<TH BGCOLOR=RED>Rojo</TH>
<TH BGCOLOR =GREN>Verde</TH>
<TH BGCOLOR =BLUE>Azul</TH></TR>
<TR ALIGN=CENTER>
<TH BGCOLOR =RED>Rojo</TH>
<TD BGCOLOR =#FF0000>Rojo</TD>
<TD BGCOLOR =#FFFF00>Amarillo</TD>
<TD BGCOLOR =#FF00FF>Violeta</TD></TR>
<TR ALIGN=CENTER>
<TH BGCOLOR =GREN>VERDE</TH>
<TD BGCOLOR =#FFFF00>Amarillo</TD>
<TD BGCOLOR =#00FF00>Verde</TD>
<TD BGCOLOR =#00FFFF>Cian</TD></TR>
<TR ALIGN=CENTER>
<TH BGCOLOR =BLUE>Azul</TH>
<TD BGCOLOR =#FF00FF>Violeta</TD>
<TD BGCOLOR =#00FFFF>Cian</TD>
<TD BGCOLOR =#0000FF>Azul</TD></TR>
</TABLE>
</BODY>
</HTML>
```

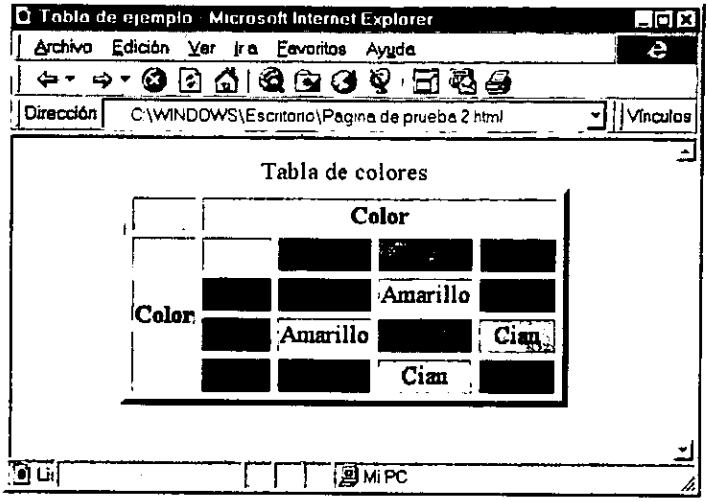

Figura 6.6: Salida del código HTML para la tabla de colores.

Como puede observarse, es muy sencillo hacer tablas con las características que se deseen, lo único que se requiere es tener cuidado con la forma en la que se colocan las celdas para que queden alineadas de acuerdo a las necesidades que se tengan.

6.4.5. Imágenes.

"Mucha gente sostiene que la única razón por la que el World Wide Web se ha vuelto tan popular es que en la página pueden verse texto formateado y gráficos. El texto está bien, pero no hay nada como una llamativa foto a colores para llamar realmente la atención de los usuarios a nuestras páginas Web"[2].

Esto en realidad es muy cierto. Si consideramos que hay más de un servicio montado en Internet y que de un tiempo a la fecha las palabras Web e Internet son prácticamente sinónimos para mucha gente. Puede suponerse que esto se debe a que es el servicio que, después del correo electrónico, más se utiliza en la actualidad. La razón de esto radica en la facilidad de uso, pero sobre todo a que es gráfico. Al ser gráfico permite la utilización de medios como las imágenes. Siempre será más agradable ver una página llena de color, dibujos y fotografías, que una con un montón de texto en la pantalla.

Las imágenes que se utilizan comúnmente en el Web son las de formato JPEG y GIF. Estas pueden ser generadas actualmente con casi cualquier editor gráfico profesional o

[2] Laura Lemay, "Aprendiendo HTML 3.0 para Web en una semana", segunda edición, Editorial Prentice-Hall, Página 234.

semiprofesional. De no contar con una herramienta capaz de grabar en estos formatos, hay una amplia variedad de editores y manejadores de imágenes en Internet capaces de hacerlo.

Cuando ya se tiene una imagen en alguno de estos formatos se puede colocar en una página mediante la etiqueta <IMG>. Esta etiqueta es de un solo lado, pero cuenta con una serie de parámetros que permiten dar condiciones especificas a la imagen a desplegar. Estos se presentan a continuación.

- ALT — Cuando se asigna algún texto a ALT, se sustituye la imagen por ese texto cuando el navegador no soporta imágenes o en el caso de que por alguna razón no pudiera leerse el archivo. Cuando se utiliza un navegador gráfico, este texto aparece en un pequeño recuadro cuando se pasa sobre la imagen con el puntero de mouse.
- ALIGN — Cuando se utiliza ALIGN, es posible alinear la imagen con respecto a la línea de texto adyacente o a otras imágenes de la misma línea. Los posibles valores son TOP, MIDDLE, BOTTOM, LEFT y RIGHT[3].
- BORDER — Asigna un borde a la imagen en pixeles. Si se asigna a BORDER el valor de 0 (cero), el borde desaparecerá aun si la imagen fuera un vínculo.
- HEIGHT — Permite especificar la altura en pixeles de la imagen. Cuando se utiliza un tamaño mayor o uno menor que el real, la imagen es escalada a la altura especificada.
- SRC — El más importante de los parámetros. Especifica el archivo que contiene la imagen. Se puede colocar la ruta en la que se encuentra el archivo en el servidor, o se puede especificar todo el URL, se es que se encuentra en otro servidor.
- WIDTH — Igual que HEIGHT pero en sentido horizontal.

Al utilizar imágenes se debe tener cuidado con el tamaño de la misma, no en el área de la pantalla que abarque, sino en el tamaño en bytes que tenga. Considerando que la información que viaja en Internet generalmente llegará a su destino con clientes que la consultan utilizando proveedores de servicio, y por tanto utilizando un modem. Cuando se utiliza un modem, el tiempo de espera es crítico, pues el servicio se cobra por hora. Si para bajar una imagen se desperdicia mucho tiempo, los clientes no querrán volver a entrar a esa página.

La recomendación es utilizar únicamente las imágenes que sean necesarias sin caer en el exceso, y mantener éstas del menor tamaño posible.

[3] LEFT y RIGHT no son aceptados en todos los visualizadores por el momento, por lo que no se puede confiar mucho en estos valores cuando se espera que no varíe el resultado de navegador a navegador.

6.4.6. Los marcos (FRAMES).

Desde la aparición de la versión 2.0 del navegador Netscape, apareció una nueva forma de presentar la información, ya que por medio de algo llamado "marcos", era posible colocar más de un documento HTML en la ventana del visualizador cada uno en una sección predefinida de la misma. Para hacer esto se debía de disponer de todos los documentos Web que fueran a presentarse al mismo tiempo. Estos documentos se ligaban utilizando un documento maestro con las instrucciones para formar los marcos y asignar a cada uno el documento correspondiente.

Esta forma de presentar la información fue exclusiva de Netscape, hasta que Microsoft decidió incluir soporte para marcos en su versión 4.0 de Internet Explorer. Al tener soporte en los dos principales navegadores del mundo, el uso de marcos se ha extendido de forma considerable. Debido a que son útiles para mejorar la navegación, se ha decidido incluir marcos en el sistema, por lo que se explican a continuación las etiquetas con las que cuenta el documento maestro.

Cuando se crea un documento maestro, la etiqueta <BODY> no aparece en el código HTML, ya que éste no tiene cuerpo. Las etiquetas que se utilizan para crear marcos son: <FRAMESET>...</FRAMESET>, <FRAME>...</FRAME> y <NOFRAMES>...<NOFRAMES>.

La etiqueta <FRAMESET>...</FRAMESET>, define la región de la pantalla en donde el navegador cliente puede crear un marco. Esta etiqueta cuenta con dos parámetros que son:

- ROWS — Cuando se usa este parámetro, se determina la altura de cada marco en pixeles o en un porcentaje del espacio disponible, así como el número de los mismos. Cada una de las alturas se divide por comas. Si se utiliza como altura del marco un asterisco (*), el marco correspondiente ocupará el resto del espacio disponible.
- BORDER — Asigna un borde entre marco y marco en pixeles. Si se asigna un valor de 0 (cero) al borde, éste es eliminado, haciendo parecer que los marcos son continuos.
- COLS — Igual que ROWS pero en sentido horizontal.

Por ejemplo la línea de apertura

```
<FRAMESET ROWS="25,50%,*">
```

especifica que el se formarán tres marcos. Uno de 25 pixeles de alto, otro del 50% del espacio aún sin utilizar y el tercero de lo que aun está disponible.

Con esta etiqueta se especificó cuantos marcos hay, ahora es necesario especificar el contenido de cada uno de ellos. La etiqueta <FRAME>...</FRAME> da atributos a cada marco. Los parámetros de esta etiqueta son:

- MARGINHEIGHT — Determina que tanto espacio vertical en pixeles existe entre el objeto ubicado en un marco, y los extremos superiores o inferiores de éste. Si no se especifica, el navegador cliente asigna el margen apropiado.
- MARGINWIDTH — Igual a MARGINHEIGHT pero en sentido horizontal.
- NAME — Ofrece al autor HTML la capacidad de dar un nombre a las ventanas FRAME, de tal forma que se puedan utilizar como el destino en los hipervínculos de otros documentos.
- NORESIZE — Cuando se usa, deshabilita la posibilidad de cambiar el tamaño del marco por el usuario.
- SCROLLING — Permite habilitar o deshabilitar una barra de desplazamiento para el marco afectado. Los valores posibles son: "YES", "NO" y "AUTO".
- SRC — El más importante. Indica al navegador cliente cuál es la dirección del documento HTML que se exhibirá en ese marco. Puede ser cualquier URL valido.

Cuando se requiere que uno de los marcos tenga a su vez otros, en lugar de utilizar <FRAME> se vuelve a definir con <FRAMESET> el conjunto de marcos. Así se pueden anidar las definiciones conforme se requieran.

Ejemplo:

```
<HTML>
<HEAD>
   <TITLE>Horarios de la UNAM Campus Arag&oacute;n</TITLE>
</HEAD>
<FRAMESET ROWS="49,*" Border=0>
  <FRAME SRC="..." NORESIZE scrolling="no">
  <FRAME SRC="..." NAME="Pantalla">
</FRAMESET>
</HTML>
```

Aquí se crean dos marcos. Uno de los cuales tendrá 49 pixeles de alto y el segundo ocupará el resto de la ventana. El primero de ellos no permitirá que se cambie su tamaño ni tendrá barras de desplazamiento. El segundo tomará los parámetros por omisión y tendrá como nombre "Pantalla". Así cuando en hipervínculo se quiera hacer referencia a este marco lo podrá hacer con:

```
<A HREF="..." TARGET="Pantalla">...</A>
```

[4] La propiedad TARGET del hipervínculo mostrado no fue mencionada cuando se explicó la utilización del "ancla" <A> ya que su utilización realmente se da en los marcos y porque no todos los navegadores lo

La última etiqueta para el manejo de marcos es destinada para los visualizadores que no los manejan. Esta etiqueta es <NOFRAMES>...</NOFRAMES>. Sabiendo que si un navegador que no tenga soporte para marcos no podrá ver nada de lo que se espera que vea, entre las etiquetas de inicio u cierre de <NOFRAMES> se coloca el código correcto para ellos. Los navegadores que si tengan soporte para marcos ignorarán todo lo que se encuentre entre el inicio y el fin de ésta.

El HTML es un lenguaje muy limitado que poco a poco va siendo más robusto y útil. La guerra de los visualizadores permite que los diseñadores tengan un poco más recursos para embellecer las páginas.

Explicar todo lo que se puede hacer con HTML llevaría un libro entero para ello. Sin embargo, con lo que se ha expuesto es posible hacer lo que se propone en este trabajo.

En el siguiente capítulo se utilizará todo el conocimiento desglosado a lo largo de estos primeros capítulos, para desarrollar por fin el sistema de consulta de horarios que será implantado en el Campus Aragón de la UNAM.

soportan. Quizá en futuras versiones de los navegadores menos populares se cuente con que los marcos estén totalmente soportados.

7

Capítulo VII

Desarrollo del sistema e implementación

Capítulo VII. Desarrollo del sistema e implementación.

En los capítulos anteriores se han analizado todos los elementos con los que contará el sistema:

- *Capítulo I*: se analizaron las necesidades del sistema, así como las posibilidades de solución.
- *Capítulo II*: se estudió la forma en la que trabaja Cronos y la estructura de los archivos que servirán de bases de datos que serán utilizadas para extraer la información.
- *Capítulo III*: se vieron las características de un sitio Web y se revisó la información con la que cuenta el servidor del Campus Aragón.
- *Capítulo IV*: se analizaron dos lenguajes de programación, el Pascal y el ANSI C, para determinar con cual de ellos se realizaría el sistema. Dadas las características de manejo de archivos y en generar de las librerías con las que cuenta se decidió por utilizar el segundo de ellos.
- *Capítulo V*: se presentó la forma en la que el usuario puede mandar una petición al sistema y la forma de procesar la misma, así como los métodos GET y POST para transmisión de información, seleccionando al segundo para la realización del sistema.
- *Capítulo VI*: se revisaron los elementos estructurales del lenguaje HTML y algunas de las etiquetas que serán de utilidad para hacer páginas Web conteniendo los resultados de la petición.

Todos los temas tratados han servido para reforzar los conocimientos necesarios para hacer el sistema y casi cualquier otro con características similares. Ya que se tiene la información se procede a realizar el sistema.

El sistema cuenta con más de un módulo, de hecho será necesario hacer uno por cada tipo de reporte, es decir, uno para los horarios de clases, otro para los exámenes finales y uno más para los exámenes extraordinarios; aunque obviamente contaran con elementos en común que serán analizados.

7.1. Análisis para "Horarios de Clases".

Como se recordará, en el capítulo II se revisaron las tablas con las que cuentan las bases de datos de Cronos. El módulo de horarios de profesores utiliza los archivos:

- HORDAT.[SEMESTRE]
- PROFESOR.[SEMESTRE]
- MATERIAS.CRO

7.1.1. Transformación de registros.

Los formatos de estos archivos también fueron presentados en el capítulo II. Considerando que los archivos ya están creados y que el sistema se debe adaptar al servidor Sun Sparc Station 4, es necesario crear una estructura que guarde los mismos tamaños de variables para cada elemento del registro declarado en Pascal.

```
typedef unsigned char HoraComp[12];        /*{ Alto = Ini;  Bajo = Fin }*/
typedef struct {
          unsigned char RegOrd2,RegOrd1;          /*{ 2 bytes}*/
          unsigned char Salon2,Salon1;            /*{ 2 bytes}*/
          unsigned char Grupo2,Grupo1;            /*{ 2 bytes}*/
          unsigned char CveMat2,CveMat1;          /*{ 2 bytes}*/
          unsigned char CveProf2,CveProf1;        /*{ 2 bytes}*/
          HoraComp Hora;                          /*{12 bytes}*/
        } Datos;                                  /*{22 bytes}*/
      /*Datos contiene la estructura del archivo HORDAT.[Semestre]*/

typedef struct {
         unsigned char  Aux2,Aux1;                /*{ 2 bytes}*/
         char           RFC[16];                  /*{16 bytes}*/
         char           Nombre[41];               /*{41 bytes}*/
         char           Funciona;                 /*{ 1 byte }*/
         } Profesor;                               /*{60 total}*/
      /* Profesor contiene la estructura del archivo PROFESOR.[Semestre]
*/

typedef struct {
         unsigned char  Clave2,Clave1;            /*{ 2 bytes}*/
         char           NomMate[51];              /*{51 bytes}*/
         unsigned char  Sem;                      /*{ 1 byte }*/
         } Materias;                               /*{54 total}*/
      /* Materias contiene la estructura del archivo MATERIAS.CRO */
```

Como puede observarse, la mayoría de los elementos fueron definidos con *unsigned char*; esto se debe a que al utilizar otros tipos de variables el registro cambiaba su tamaño, siendo contraproducente para la estructura física de los archivos.

7. 1.2. Necesidades para procesar la información.

Lo importante de la información con la que se cuenta es que ya está almacenada en los archivos y estos tienen ciertas características especificadas en el capítulo II. Como se recordará estos están divididos por carrera y por semestre, por lo que es necesario conocer estos dos datos para procesar la información.

La información requiere de la intervención del usuario para poder obtenerse, además éste debe especificar el número de grupo para el cual se solicita la misma.

En base a esto se pueden especificar los siguientes pasos para el procesamiento y presentación de la información:

1. *Interfaz de usuario para solicitar carrera, semestre y grupo.* Para realizar esto se aplicará lo mostrado en los capítulos V y VI, ya que en ellos se analizaron los conceptos para el manejo del HTML y del CGI.

2. *Procesamiento de formularios.* Para obtener la información del usuario como se mostró en el capítulo V, además de utilizar el lenguaje ANSI C para procesarla.

3. *Consulta de archivos.* Ya que se tienen los parámetros necesarios, abrir el archivo HORDAT.[Semestre] y leer el primer registro. En caso de corresponder al grupo solicitado, obtener de forma secuencial[1] el nombre del profesor y la materia a la que se refiere el registro de los archivos PROFESOR.[Semestre] y MATERIAS.CRO respectivamente.

4. Teniendo toda la información referente al registro que coincida con la petición del usuario, procesarla para mostrarla en formato HTML y enviarla al navegador. Repetir hasta que acabe el archivo y no se tengan más coincidencias.

7.2. Implementación de los módulos para Horarios.

7.2.1. La interfaz de usuario.

El primer paso será obtener la información del usuario, por lo que inicialmente se hará la interfaz de usuario. Ésta debe ser agradable a la vista, sencilla de utilizar y debe de lograr el objetivo para lo que fue hecha. Utilizando FRAMES se pretende facilitar la navegación, por lo que se utilizan tres páginas:

- index.html: Contendrá la estructura de los marcos.
- menu.html: Contendrá un menú para navegar entre los tres elementos del sistema (horarios, finales y extraordinarios), así como un botón para correo electrónico[2].
- horarios.html: Página de interface para los horarios de clases.

A continuación se presentan los códigos que conforman las páginas que ahora se comentan.

[1] Se propone el método secuencial por dos razones: 1) La cantidad de registros en cada archivo es muy pequeña. 2) los archivos se encuentran desordenados al momento de tomarlos de Cronos. Debido a la cantidad de la información, no es necesario realizar un proceso de ordenación y búsqueda binaria.

[2] Las páginas index.html y menu.html serán las mismas para las tres partes del sistema, por lo que solo se presentarán en este momento.

7.2.1.1. El menú de opciones.

```
<!-- Menu de opciones para el Proyecto de Horarios para el Campus Aragón.
     Unidad de Planeación, Departamento de Informática.
     Desarrollado por: Ernesto Manzo Salazar.  -->

<HTML>
<HEAD>
<SCRIPT language="JavaScript">

<!-- // Ocultar a navegadores viejos
function display(num) {
  num % 2 == 1 ? indice=eval((num - 1) / 2 ): indice = eval((num - 2)/2);
  document.images[indice].src="boton"+(indice+1)+"_"+
                              (num % 2 == 1 ? 1:2)+".gif";
   }
// -- Fin del codigo JavaScript ---------- -->

</SCRIPT>
</HEAD>
<BODY BGCOLOR="#000000" BACKGROUND="fondo1.jpg">
<CENTER><TABLE BORDER=0 WIDTH="530" >
<TR>
<TD ALIGN=CENTER WIDTH="128">
      <A HREF="extra.html" TARGET="Pantalla"
            onMouseOver="display(2)"; onMouseOut="display(1)">
      <IMG SRC="boton1_1.gif" ALT="Ex&aacute;menes extraordinarios"
            HEIGHT=21 WIDTH=128 BORDER=0>
</TD>
<TD ALIGN=CENTER WIDTH="128">
      <A HREF="finales.html" TARGET="Pantalla"
            onMouseOver="display(4)"; onMouseOut="display(3)">
      <IMG SRC="boton2_1.gif" ALT="Ex&aacute;menes finales"
            HEIGHT=21 WIDTH=128 BORDER=0>
</TD>
<TD ALIGN=CENTER WIDTH="128">
      <A HREF="horarios.html" TARGET="Pantalla"
            onMouseOver="display(6)"; onMouseOut="display(5)">
      <IMG SRC="boton3_1.gif" ALT="Horarios de clases"
            HEIGHT=21 WIDTH=128 BORDER=0>
</TD>
<TD ALIGN=CENTER WIDTH="128">
      <A HREF="mailto:www@informatica.aragon.unam.mx" TARGET="Pantalla"
            onMouseOver="display(8)"; onMouseOut="display(7)">
      <IMG SRC="boton4_1.gif" ALT="Comentarios o sugerencias"
            HEIGHT=21 WIDTH=128 BORDER=0>
</TD>
</TR>
</TABLE></CENTER>
</BODY>
</HTML>
```

menu html Código fuente del menú de opciones HTML para horarios, exámenes finales y extraordinarios

En este menú se utilizan técnicas de animación a base de funciones de JavaScript. Al colocar el puntero del mouse sobre una de las imágenes que conforman al menú, la cual es reemplazada por otra con características similares pero de diferente color. Al hacer esto se produce la sensación de que la imagen se ha seleccionado. Al quitar el puntero de mouse de

dicha imagen, se vuelve a colocar la imagen original. En Internet se ha utilizado muchas veces este efecto sin embargo, la función aquí utilizada es totalmente original.

El menú se coloca en la parte superior de las tres páginas que se manejan.

7.2.1.2. Página para horarios.

A continuación se presenta el código fuente de la página para los HORARIOS DE CLASES, y posteriormente se presentará el código que unirá el menú con los horarios.

```
<!--  Página de Horarios de Clases para el Proyecto de Horarios para el
      Campus Aragón.
      Unidad de Planeación, Departamento de Informática.
      Desarrollado por: Ernesto Manzo Salazar.  -->

<HTML>
<HEAD>
<TITLE>Horarios de Clases de la UNAM CAMPUS ARAGON</TITLE>
</HEAD>
<BODY BACKGROUND="/servicios/horarios/fondo.jpg" Width=640>
<CENTER>
<Table BORDER=0>
<TR>
   <TD Width="134"> <img src="/servicios/horarios/escudo.gif" alt="UNAM"
      width=134 height=100 Border=0>
   </TD>
   <TD     Width="376">     <img     src="/servicios/horarios/HorClas.GIF"
alt="Horarios de Clases"
      width=376 height=37 Border=0>
   </TD>
   <TD Width="134"> <img src="/servicios/horarios/escudo.gif" alt="UNAM"
      width=134 height=100 Border=0>
   </TD>
</TR>
</Table>
</CENTER>
<BR>
<CENTER>
<Table BORDER=0 CELLSPACING=10 CELPADDING=5 WIDTH="630">
<TR>
   <TD      align=Center      Background="/servicios/horarios/fondo1.jpg"
BGCOLOR="NAVY">
      <Font Size=+1 Color="White"><B> Instrucciones</B></Font>
   </TD>
</TR>
<TR>
<TD>
<P  ALIGN=JUSTIFY>Introduce en el formulario el grupo para el cual
requieres información, tomando en cuenta que los grupos se organizan de
la siguiente manera:</P>
<OL>
<LI><P ALIGN=JUSTIFY>El primer d&iacute;gito representa si es semestre
par o impar por lo que solo puede ser 1 y 2.
<LI><P ALIGN=JUSTIFY>El segundo d&iacute;gito generalmente representa el
semestre a consultar,
<LI><P ALIGN=JUSTIFY>Los dos &uacute;ltimos d&iacute;gitos representan al
n&uacute;mero de grupo en si mismo, empezando de 01 en adelante para el
turno matutino y de 51 en adelante para el turno vespertino.
```

```
</OL>
<P ALIGN=JUSTIFY>Nota: Los dos &uacute;ltimos d&iacute;gitos pueden no
iniciar en 01 o en 51, sino en un n&uacute;mero posterior. Esto depende
de la carrera a la que se refieran, sin embargo conservan el mismo
formato.</P>
</TD>
</TR>
</Table>
</CENTER>
<CENTER><TABLE BORDER=0 CELLSPACING=5 CELPADDING=5 WIDTH="630">
<TR>
<TD        align=Center        Background="/servicios/horarios/fondo1.jpg"
BGCOLOR="NAVY">
      <Font Size=+1 Color="White"><B>Datos de Consulta</B></Font>
</TD>
</TR>
<TR>
<TD Align=Center BGColor="Silver">
<FORM METHOD="POST"
ACTION="http://informatica.aragon.unam.mx/cgi-bin/horarios/horarios.cgi">
Semestre:
<SELECT NAME="Semestre" SIZE="1">
   <OPTION>1999-1
   <OPTION>1998-2
</SELECT>
Grupo:
<INPUT TYPE="text" SIZE=4 NAME="Grupo" MAXLENGTH=4><BR>
Carrera:
<SELECT NAME="Carrera" SIZE="1">
   <OPTION>Arquitectura
   <OPTION>Dise&ntilde;o Industrial
   <OPTION>Relaciones Internacionales
   <OPTION>Sociolog&iacute;a
   <OPTION>Comunicaci&oacute;n y Periodismo
   <OPTION>Derecho
   <OPTION>Econom&iacute;a
   <OPTION>Pedagog&iacute;a
   <OPTION>Ing. Civil
   <OPTION>Ing. Mec&aacute;nica El&eacute;ctrica
   <OPTION>Ing. en Computaci&oacute;n
   <OPTION>Planificaci&oacute;n para el Desarrollo Agropecuario
</SELECT><BR>
<INPUT TYPE="submit" VALUE="Consultar">
<INPUT TYPE="reset" VALUE="  Borrar  ">
</FORM>
</TD>
</TR>
</TABLE>
</CENTER>
<HR WIDTH=100% SIZE=5 ALIGN=CENTER NOSHADE>
<Table BORDER=0 WIDTH="100%">
<TR>
   <TD Width="306">
      <img src="/servicios/horarios/PLANEACION.GIF"
           alt="Unidad de Planeaci&oacute;n"
      width=220 height=11 Border=0 ALIGN=LEFT>
   </TD>
   <TD Width="220">
      <img src="/servicios/horarios/INFORMATICA.GIF"
      alt="Departamento de Inform&aacute;tica" width=306
           height=11 Border=0
```

```
      ALIGN=Right>
   </TD>
</TR>
</Table>
</BODY>
</HTML>
```

horarios.html: Código fuente de la página interface para la consulta de horarios.

Esta página tiene una pequeña sección de instrucciones que antecede a la entrada de información de consulta. Con ella se pretende que el usuario tenga las bases para formar un grupo y a partir de él navegar por todos los demás. Cabe notar que muchas personas que utilizan cualquier sistema se saltan las instrucciones, así que se debe tomar en cuenta que si se manda un formulario vacío, el sistema será capaz de permitir que el usuario llegue hasta el grupo deseado.

7.2.1.3. Fusión de las páginas.

Esta página puede funcionar independiente de el menú, sin embargo para mejorar la interacción con el usuario se utiliza una página maestra que se encargará de juntar el menú con los horarios. A continuación se presenta el código fuente de la misma.

```
<!-- Unidad de Planeación, Departamento de Informática.
     Desarrollado por: Ernesto Manzo Salazar.  -->
<HTML>
<HEAD>
   <TITLE>Horarios de la UNAM Campus Arag&oacute;n</TITLE>
</HEAD>
<FRAMESET ROWS="49,*" Border=0>
  <FRAME SRC="menu.html" NORESIZE>
  <FRAMESET COLS="*" Border=0>
     <FRAME SRC="horarios.html" NAME="Pantalla">
  </FRAMESET>
</FRAMESET>
</HTML>
```

indexs.html: Código fuente de la página principal, que junta los horarios con el menú.

En el capítulo VI se presentaron las etiquetas <FRAME> y <FRAMESET> que tienen la finalidad de combinar varias páginas en una sola ventana de navegación, separadas por marcos que hacen independientes a una de la otra, pero con la capacidad de interactuar entre ellas. En el código que se presenta se hace referencia a dos ventanas, la primera que tomara el ancho de la ventana y únicamente 49 pixeles desde la parte superior; el resto será para la segunda página. Teniendo esto, sólo se debe especificar que documentos se cargan en cada uno de los marcos.

Una vez hecho esto se colocan los tres archivos HTML y las imágenes correspondientes en el directorio *página_base/servicios/horarios*[3] ubicado en el servidor, dando como resultado la página mostrada en la figura 7.1. la cual puede ser accesada por la dirección "http://informatica.aragon.unam.mx/servicios/horarios".

[3] Por razones de seguridad no se presenta aquí la dirección exacta de la página base.

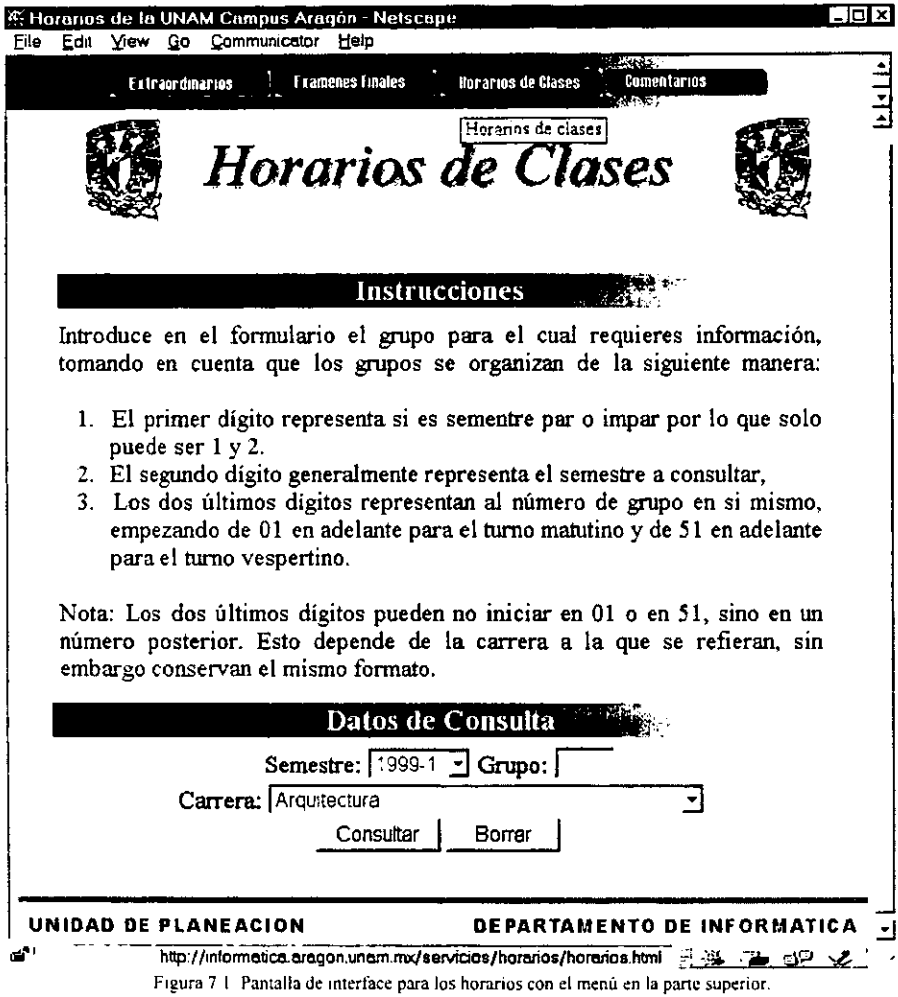

Figura 7.1 Pantalla de interface para los horarios con el menú en la parte superior.

7.2.2. Procesamiento de formularios.

Hasta el momento se ha estudiado la forma en la que el usuario verá el sistema. Desde este instante la información pasará vía navegador, el sistema basado en CGI, hasta el servidor, donde será analizado y en base a la consulta, se generará dinámicamente la página de resultados. El usuario final siempre verá únicamente código HTML.

Los formularios serán analizados de la forma que se hizo en el capítulo V. Como se recordará existen formas de mandar la información: una es con el método GET y la segunda con el POST; en el presente sistema se utiliza el segundo de ellos.

Para este caso es necesario revisar inicialmente las variable de ambiente CONTENT_TYPE y CONTENT_LENGTH con las cuales se determinará si la consulta es del tipo deseado y además si puede ser leído. Con estas variables se debe hacer todo el proceso de validación y lectura del QUERY_STRING, una vez hecho esto, se debe verificar sea correcto.

La separación de las diferentes partes de la consulta se realiza con las funciones antes vistas:

- `void splitword(char *out,char *in,char stop;`
- `void unescape_url(char *url);`

La primera se encargará de separar una cadena en dos partes dependiendo del carácter de paro, en donde encuentre la primera aparición de un caracter en particular detendrá el proceso, eliminando toda la parte inicial de la cadena original. La segunda función convertirá todos los elementos hexadecimales en caracteres reales. Se utilizarán las variables *Grupo*, *Semestre* y *Carrera* para almacenar la petición decodificada del usuario, de tal forma que pueda ser manipulada con mayor facilidad.

7.2.2.1. Pasos a seguir para leer el formulario.

El proceso completo de lectura consta de los siguientes pasos[4]:

- Se toman *CONTENT_TYPE* y *CONTENT_LENGTH* para verificar su validez y se almacenan en las variables **ct* y **cl* respectivamente.
  - Se verifica que *cl* exista y sea mayor que cero.
  - Se verifica que *ct* sea *application/x-www-form-urlencoded*, que es el tipo MIME correcto para aplicaciones CGI.
- Se asigna memoria de forma dinámica para leer la consulta en base al valor de *cl + 1*. En caso de encontrarse algún error en este paso, desplegar un mensaje de error y terminar el programa. Se utilizará la variable **qt* para almacenar la consulta.
- Leer la consulta desde la entrada estándar (stdin) y agregar al final un fin de cadena.
- Dividir cada uno de los parámetros del flujo de consulta.
  - Dividir por '&' cada parámetro.
  - Convertir la cadena para caracteres hexadecimales y signos más.
  - Separar los elementos por nombre y valor usando división por "=".
  - Repetir hasta que se acabe la cadena.
- Verificar que los elementos de nombre correspondan únicamente las variables esperadas que son: *Grupo*, *Semestre* y *Carrera*, de ser así asignarlos a las variables globales mencionadas anteriormente.

[4] Básicamente esta es la forma en la que se hacen este tipo de consultas. Por motivos de seguridad no se agrega el código fuente de este segmento del programa, sin embargo con lo que se muestra se puede codificar en cualquier lenguaje que se desee.

- Ya que se leyeron estos valores se debe verificar que la información contenida en ellas sea la correcta.
    - En el caso de *Grupo* se verifica que sea un número, en caso de no serlo se convertirá en CERO.
    - En caso de ser el *Semestre*, se verifica la que sea semestre válido, de no serlo, no se presentará la información.
    - Para el caso de la *Carrera* se utiliza un arreglo con todos los nombres de carreras validos, así como la ruta de acceso al directorio en el que se encontrarán los archivos de horarios para esa carrera.

Una vez hecho todo lo anterior se cuenta con todos los valores necesarios para realizar la consulta a los archivos de base de datos de horarios y así devolver las información que el usuario pide.

7.2.3. Consulta de los archivos y presentación de resultados.

Cuando el sistema llega a abrir los archivos, quiere decir que se devolverá una página con información o sin ella, es decir, en el caso de encontrarse el grupo solicitado en algún horario, se procederá a enviarlo, de lo contrario se procederá a dar alternativas de grupos, pero en cualquier caso se dará una respuesta.

Sabiendo lo anterior es necesario implementar los módulos que se encarguen de:

- Mostrar una cabecera para resultados (lo que aparecerá en todas las consultas que se realicen).
- Un módulo que analice el archivo HORARIOS.[Semestre].
- Módulo que busque y despliegue al profesor correspondiente a el horario encontrado.
- Módulo que busque y despliegue el nombre de la materia.
- Módulo que convierta horas comprimidas[5] a formato HTML.
- Módulo que despliegue todo un registro de horarios en formato HTML.
- Parte final de la página Web.

7.2.3.1. Algunas variables necesarias.

Llamaremos *Datos* a la estructura de cada registro del archivo HORDAT. Como se recordará uno de los campos se llama *Salon* y hace referencia tanto al salón como al cupo del mismo. Cronos utiliza este valor para tomarlo como índice de un arreglo que contiene ambos valores. Es llamado SalonFisico, y se presenta a continuación:

```
#define  MaxSalon   276;
short int SalonFisico[] =
```

[5] En la sección 7.1.1, se puede apreciar que el campo *Hora* se encuentra comprimido, así que se debe descomprimir y mostrar como *hora inicial - hora final.*

```
{111,60,112,60,113,60,114,60,115,60,116,60,
121,30,122,30,123,30,124,30,125,30,126,30,
127,30,128,30,129,30,130,60,
200,60,201,60,202,60,203,60,204,60,205,60,
211,60,212,60,213,60,214,60,215,60,216,60,
221,60,222,60,223,60,224,60,225,60,226,60,
301,60,302,60,303,60,304,60,305,60,306,60,
311,60,312,60,313,60,314,60,315,60,316,60,
321,60,322,60,323,60,324,60,325,60,
411,60,412,60,413,60,414,60,415,60,416,60,
421,60,422,60,423,60,424,60,425,60,426,60,
501,60,502,60,503,60,504,60,505,60,506,60,
511,31,512,60,513,31,514,31,515,31,516,31,
521,61,522,31,523,31,524,60,525,60,
601,60,602,60,603,60,604,60,605,60,606,60,
611,60,612,60,613,60,614,60,615,60,616,60,
621,60,622,60,623,30,624,30,625,30,626,30,  627,60,629,60,
701,60,702,60,703,60,704,60,705,60,706,60,
711,60,712,60,713,60,714,60,715,60,716,60,
721,60,722,60,723,60,724,60,725,60,726,60,
801,60,802,60,803,60,804,60,805,60,806,60,
8110,30,8111,30,8112,30,8113,30,8114,30,8115,30,
8116,30,8117,30,8118,30,
8119,30,8120,30,8121,30,
8221,40,8222,40,8223,40,8224,40,8225,40,8226,40,8227,40,8228,40,
8229,40,8230,40,
901,60,902,60,903,60,904,60,
911,31,912,31,913,31,914,31,915,31,916,31,
921,60,922,31,923,31,924,31,925,31,926,31,
1001,40,1002,40,1003,40,1004,40,1005,40,1006,40,1007,60,1008,60,
1011,40,1012,40,1013,40,1014,40,1015,40,1016,40,1017,60,1018,60,
1021,40,1022,40,1023,40,1024,40,1025,40,1026,40,1027,60,1028,60,
1101,60,1102,60,1103,60,1104,60,1105,60,1106,  60,
11101,30,11102,30,11103,30,11104,30,11105,30,11106,30,11107,30,
11108,30,11109,30,
11110,30,11111,30,11112,30,
11201,40,11202,40,11203,40,11204,40,11205,40,11206,40,11207,40,
11208,40,11209,40,
1201,60,1202,60,1203,60,1204,60,1205,60,1206,  60,
12101,30,12102,30,12103,30,12104,30,12105,30,12106,30,12107,30,
12108,30,12109,30,
12110,30,12111,30,12112,30,
12201,40,12202,40,12203,40,12204,40,12205,40,12206,40,12207,40,
12208,40,12209,40,
5000,60,6000,60,7000,60,5001,60,5002,60,5003,  60,5004,60,5005,60,
14101,60,14102,60,14103,60,14104,60,14105,60,14107,60,
14201,60,14202,60,14203,60,14204,60,14218,60,
14303,60,14304,60,14313,60,14315,60,14316,60,
14317,60,14323,60,14324,60,  14211,60,
14401,60,14402,60,14403,60,14404,60,14405,60,
14406,60,14407,60,14408,60,14409,60,14410,60,14411,60};
```

Se utilizarán las siguientes variables para el manejo de los archivos:

```
• FILE *ArchProf;       /*Apuntador al archivo de profesores */
• FILE *ArchMat;        /*Apuntador al archivo de materias */
• FILE *ArchDat;        /*Apuntador al archivo de horarios */

• unsigned int MaxProf;/*Máximo numero de profesores */
```

```
• unsigned int MaxMat; /* Máximo numero de materias */
• unsigned int MaxDat; /* Máximo numero de horarios */

• char Acceso[40];      /*Contiene el directorio en el cual se encuentran
  los archivos */
```

7.2.3.2. La función BuscaProf(unsigned int ClaveProf).

Esta función recibe como parámetro la clave del profesor, la cual se obtiene del registro actual de los horarios. A continuación se presenta su funcionamiento.

```
void BuscaProf(unsigned int ClaveProf) {
     Profesor ProfDat;
     int i;
     char Cadena[41];
     char NomArch[50];
     strcpy(NomArch,Acceso);
     strcat(NomArch,"PROFESOR.");
     strcat(NomArch,AuxSemestre);
     if ((ArchProf=fopen(NomArch,"rb"))== NULL) {
        DespliegaArchivo("página_base/servicios/horarios/horerror.html",
                         "No se encontr&oacute; el archivo de profesores\n");
        exit(0);
     }
   fseek(ArchProf,40,0);
     while (!feof(ArchProf)) {
           if (fread(&ProfDat,sizeof(Profesor),1,ArchProf)) {
             ProfDat.Nombre[40]='\0';
             ProfDat.Nombre[0]=' ';
             if (ClaveProf==(ProfDat.Aux1 << 8) + ProfDat.Aux2) {break;}
           }
        }
     if (feof(ArchProf) && (ClaveProf != (ProfDat.Aux1 << 8) +
                                          ProfDat.Aux2)) {
        strcpy(ProfDat.Nombre," ");
        strcpy(ProfDat.RFC," ");
        ProfDat.Funciona = '0';
     }
     fclose(ArchProf);
     strcpy(Cadena,ProfDat.Nombre);
     printf(Cadena);
   }
```

En esta función lo primero que se hace es determinar la localización del archivo de profesores y una vez encontrado se procede a la correspondiente lectura y búsqueda de el profesor especificado. Como se recordará el archivo PROFESORES.[Semestre]. Cuenta con una cabecera de 40 bytes, estos deben ser tomados por alto para llegar por fin a los datos.
A partir de ese punto se entra a un loop que lee y compara cada uno de los registros hasta que encuentra el buscado o se termina el archivo, dependiendo del resultado se manda a la salida estándar el nombre del profesor o una cadena de un solo espacio.

7.2.3.3. La función BuscaMat(unsigned int ClaveMat).

```
void BuscaMat(unsigned int ClaveMat)  {
     Materias MatDat;
     int i;
     char Cadena[41];
     char NomArch[50];
     strcpy(NomArch,Acceso);
     strcat(NomArch,"MATERIAS.CRO");
     if ((ArchMat=fopen(NomArch,"rb"))== NULL)  {
        DespliegaArchivo("página_base/servicios/horarios/horerror.html",
                         "No se encontr&oacute; el archivo de materias\n");
        exit(0);
     }
     while (!feof(ArchMat)) {
           if (fread(&MatDat,sizeof(Materias),1,ArchMat)) {
             MatDat.NomMate[50]='\0';
             MatDat.NomMate[0]=' ';
             if (ClaveMat==(MatDat.Clave1 << 8) + MatDat.Clave2){break;}
           }
        }
     if (feof(ArchMat)&&(ClaveMat!=(MatDat.Clave1 << 8)+MatDat.Clave2)) {
        strcpy(MatDat.NomMate," ");
        MatDat.Sem = 0;
     }
     fclose(ArchMat);
     strcpy(Cadena,MatDat.NomMate);
     printf(Cadena);
   }
```

Básicamente la forma en la que trabaja esta función es la misma que en *BuscaProf*, la diferencia es que no es necesario saltarse un espacio para la cabecera, puesto que no cuenta con ninguna.

7.2.3.4. La función DespliegaArchivo(char *Nombre, char *Texto).

En las dos funciones anteriores y en otras partes del código se ha manejado el control de errores. Cuando el sistema detecta algún error crítico, como puede ser la inexistencia de algún archivo, insuficiente memoria o que no puedan ser leídos los registros, se presenta un archivo HTML notificando al usuario la existencia de un error. La función que se encarga de hacerlo es la siguiente:

```
void DespliegaArchivo(char *Nombre, char *Texto)  {
     FILE *Pagina;
      if ((Pagina=fopen(Nombre,"r"))== NULL) {
         printf(Texto);
         exit(1);
      }
      while (!feof(Pagina)) {
         char ch = fgetc(Pagina);
         if (ch != EOF) putc(ch,stdout);
       }
      fclose(Pagina);
  }
```

7.2.3.5. Sección principal de la función void main(void).

La sección que se encarga de buscar cada uno de los horarios será ejecutada solamente una vez, por lo que no es necesario agregarla a una función aparte. En *main* se presentan las instrucciones necesarias para abrir el archivo, obtener los registros y comparar si corresponden al grupo solicitado. El código fuente es el siguiente:

```
      strcpy(NomArch,Acceso);
      strcat(NomArch,"HORDAT.");
      strcat(NomArch,AuxSemestre);

      if ((ArchDat=fopen(NomArch,"rb"))== NULL) {
         DespliegaArchivo("/página_base/servicios/horarios/horerror.html",
                          "No se encontr&oacute; el archivo de Horarios\n");
         exit(0);
      }

       Encabezado();
       while (!feof(ArchDat))  {
         if (fread(&Dato,sizeof(Datos),1,ArchDat)) {
           unsigned int AuxGrupo = (Dato.Grupo1 << 8) + Dato.Grupo2;
           if ((AuxGrupo<Grupo)&&(AuxGrupo > GrupoAnt)) GrupoAnt=AuxGrupo;
           if ((AuxGrupo>Grupo)&&(AuxGrupo < GrupoSig)) GrupoSig=AuxGrupo;
           if ((Dato.Grupo1 << 8) + Dato.Grupo2 == Grupo) {
              DespliegaDatos(Dato);
              Encontrado = 1;
           }
         }
       }
       fclose(ArchDat);
       if (!Encontrado) {
          printf("<TR>\n<TD COLSPAN=12 ALING=CENTER BGCOLOR=%cNavy%c>No se
han encontrado materias para este grupo, el grupo no existe, o no han
sido actualizados los horarios.<BR></TD>\n</TR>\n",34,34);
       }
       printf("</TABLE>\n</CENTER><BR>\n\n");
```

En esta sección se hace la llamada al encabezado de la página Web de respuesta y a partir de ahí se hace una búsqueda secuencial del horario solicitado. Como se puede apreciar, se utilizan las variables *GrupoAnt* y *GrupoSig*, para poder resolver el problema generado cuando el usuario no sabe el número de grupo que quiere consultar. En caso de no encontrar ninguna coincidencia del grupo solicitado se presentará el mensaje colocado en la penúltima instrucción, la cual utiliza yá el formato HTML. Las funciones que son llamadas aquí son: *Encabezado() y DespliegaDatos(Dato).*

7.2.3.6. La función Encabezado(void).

Las páginas generadas tienen una sección similar en la parte superior que es la que muestra el título de la misma, la carrera a la que se refiere, el semestre y el grupo; además de esto

presenta la primera parte de la tabla de resultados con los encabezados de la misma. La función que se encarga de todo lo anterior es la siguiente[6]:

```
void Encabezado(void)  {
  printf("<HTML>\n");
  printf("<HEAD>\n");
  printf("<TITLE>Horarios de Clases en el Campus Arag&oacute;n",
       "</TITLE>\n");
  printf("</HEAD>\n");

  printf("<BODY BACKGROUND=%c/servicios/horarios/fondo.jpg%c
       TEXT=%cWhite%c>\n",34,34,34,34);
  printf("<CENTER><TABLE BORDER=0>\n<TR>\n");
  printf("<TD Width=%c134%c><img src=%c/servicios/horarios/escudo.gif%c
      alt=%cUNAM%c width=134 height=100 Border=0></TD>\n"
      ,34,34,34,34,34,34);
  printf("<TD Width=%c376%c><IMG SRC=%c/servicios/horarios/HorClas.GIF%c
      alt=%cHorarios de Clases%c WIDTH=376 HEIGHT=37
      BORDER=0></TD>\n",34,34,34,34,34,34);
  printf("<TD Width=%c134%c><img src=%c/servicios/horarios/escudo.gif%c
      alt=%cUNAM%c width=134 height=100 Border=0></TD>\n"
      ,34,34,34,34,34,34);
  printf("</TR>\n</TABLE></CENTER>\n\n");

  printf("<CENTER>\n<TABLE BORDER=0 CELLSPACING=5 CELPADDING=5
       WIDTH=%c630%c>\n",34,34);
  printf("<TR>\n");
  printf("<TD COLSPAN=2 ALIGN=CENTER
       Background=%c/servicios/horarios/fondo1.jpg%c
       BGCOLOR=%cNAVY%c><Font Size=+1 Color=%cWhite%c><B>%s</B></Font>
      </TD>\n",34,34,34,34,34,34,Carrera);
  printf("</TR>\n");
  printf("<TR>\n");
  printf("<TD ALIGN=LEFT><FONT SIZE=+1 COLOR=%cBlack%c><B>Grupo: %d</B>
      </FONT>\n</TD>\n",34,34,Grupo);
  printf("<TD ALIGN=RIGHT><FONT SIZE=+1 COLOR=%cBlack%c><B>
      Semestre: %s</B></FONT>\n</TD>\n",34,34,elements[Semestre].val);
  printf("</TR>\n");
  printf("<TR>\n");
  printf("<TD COLSPAN=2 ALIGN=CENTER><FONT SIZE=+1 COLOR=%cBlack%c>
      <B>%s semestre</B></FONT>\n</TD>\n",34,34,
      Grupo>=1000 ? NumSemestre[((Grupo % 1000)/100)]:"Se desconoce el");
  printf("</TR>\n");
  printf("</TABLE>\n</CENTER><BR>\n\n");

  printf("<CENTER>\n<TABLE BORDER=0 CELLSPACING=5 CELPADDING=5
      WIDTH=%c100%%%c>\n",34,34);
  printf("<TR ALIGN=CENTER BGCOLOR=%cBlack%c>\n",34,34);
  printf("<TH>CMAT<BR></TH>\n",34,34);
  printf("<TH>NOMBRE DE LA MATERIA<BR></TH>\n",34,34);
  printf("<TH>NOMBRE DEL PROFESOR<BR></TH>\n",34,34);
  printf("<TH>GRUPO<BR></TH>\n",34,34);
  printf("<TH>LUN<BR></TH>\n",34,34);
  printf("<TH>MAR<BR></TH>\n",34,34);
  printf("<TH>MIE<BR></TH>\n",34,34);
```

[6] En el código que se presenta se hacen algunos saltos de línea que no deben colocarse, esto es por motivos de espacio y presentación. Al transcribir este código se debe tener cuidado de mantener en una sola línea todo lo que este entre comillas dobles: "".

```
  printf("<TH>JUE<BR></TH>\n",34,34);
  printf("<TH>VIE<BR></TH>\n",34,34);
  printf("<TH>SAB<BR></TH>\n",34,34);
  printf("<TH>SALON<BR></TH>\n",34,34);
  printf("<TH>CUPO<BR></TH>\n",34,34);
  printf("</TR>\n\n");
  }
```

Aquí se hace un cálculo en base a los datos aportados por el usuario para presentar el encabezado correspondiente. Como el cliente es un navegador Web toda la salida debe ser mandada en formato HTML.

7.2.3.7. La función void DespliegaDatos(Datos Dato).

Una vez conseguido el registro que contiene el horario con el grupo correspondiente, se utiliza la siguiente función para traducirlo al formato HTML e integrarlo a la tabla de resultados.

```
void DespliegaDatos(Datos Dato)   {
     int AuxSalon = (Dato.Salon1 << 8) + Dato.Salon2;
     printf("<TR BGCOLOR=%cNavy%c ALIGN=CENTER>\n<TD>",34,34);
     printf("<FONT SIZE=-1>%d</FONT><BR>",(Dato.CveMat1 << 8)
             +Dato.CveMat2);
     printf("</TD>\n");
     printf("<TD ALIGN=LEFT><FONT SIZE=-1>");
     BuscaMat((Dato.CveMat1 << 8) + Dato.CveMat2);
     printf("</FONT><BR></TD>\n");
     printf("<TD ALIGN=LEFT><FONT SIZE=-1>");
     BuscaProf((Dato.CveProf1 << 8) + Dato.CveProf2);
     printf("</FONT><BR></TD>\n");
     printf("<TD>");
     printf("<FONT SIZE=-1>%d</FONT><BR>",(Dato.Grupo1 << 8)
             + Dato.Grupo2);
     printf("</TD>\n");
     printf("<TD><FONT SIZE=-1>"); ConvierteHorario(Dato.Hora[0]);
     printf("<BR>"); ConvierteHorario(Dato.Hora[1]);
     printf("</FONT><BR></TD>\n");
     printf("<TD><FONT SIZE=-1>"); ConvierteHorario(Dato.Hora[2]);
     printf("<BR>"); ConvierteHorario(Dato.Hora[3]);
     printf("</FONT><BR></TD>\n");
     printf("<TD><FONT SIZE=-1>"); ConvierteHorario(Dato.Hora[4]);
     printf("<BR>"); ConvierteHorario(Dato.Hora[5]);
     printf("</FONT><BR></TD>\n");
     printf("<TD><FONT SIZE=-1>"); ConvierteHorario(Dato.Hora[6]);
     printf("<BR>"); ConvierteHorario(Dato.Hora[7]);
     printf("</FONT><BR></TD>\n");
     printf("<TD><FONT SIZE=-1>"); ConvierteHorario(Dato.Hora[8]);
     printf("<BR>"); ConvierteHorario(Dato.Hora[9]);
     printf("</FONT><BR></TD>\n");
     printf("<TD><FONT SIZE=-1>"); ConvierteHorario(Dato.Hora[10]);
     printf("<BR>"); ConvierteHorario(Dato.Hora[11]);
     printf("</FONT><BR></TD>\n");
     printf("<TD>");
     printf("<FONT SIZE=-1>%d</FONT><BR>",
      AuxSalon!=0 ? SalonFisico[(AuxSalon-1)*2] : 0);
     printf("</TD>\n");
```

```
    printf("<TD>");
    printf("<FONT SIZE=-1>%d</FONT><BR>",
     AuxSalon!=0 ? SalonFisico[AuxSalon*2-1] : 0);
    printf("</TD>\n");
    printf("</TR>\n\n");
  }
```

Prácticamente se realiza una traducción directa del registro en modo binario al texto conformado por etiquetas e información. En algunas de las funciones presentadas y en particular en ésta se utiliza una instrucción como la siguiente:

```
int AuxSalon = (Dato.Salon1 << 8) + Dato.Salon2;
```

Esto es equivalente a escribir:

```
int AuxSalon = (Dato.Salon1 *  256) + Dato.Salon2;
```

Esto se realiza porque la forma en la que se tienen que leer los registros para mantener el tamaño de los mismos no permite utilizar variables de 16 bits, así que se realiza la traducción a 16 bits por medio de esta operación, obteniéndose así el verdadero valor almacenado. El motivo por el que se utilizan desplazamientos a la izquierda es por que el procesador usa menos ciclos de reloj en hacer un simple desplazamiento de bits que al aplicar el algoritmo de la multiplicación, lo cual se traduce en menor tiempo de procesamiento.

7.2.3.8. La función void ConvierteHorario(unsigned char Num).

La función ConvierteHorario se encarga de traducir de un formato reducido a uno de 24 horas. Cronos reduce el espacio de almacenamiento de la hora presuponiendo que nunca se darán clases antes de las 07:00 ni después de las 22:00 y que se contará con horarios redondeados a los cinco minutos más cercanos. Por ejemplo los siguientes horarios son validos:10:35 - 11:15, 07:00 - 09:00, 14:55 - 15:35; mientras que no sería interpretado correctamente algo como lo siguiente: 06:30 - 07:23, 14:12 - 15:17, 21:14 - 22:32. Esto es por obvias razones: no resulta práctico programar horarios que no estén redondeados a los cinco minutos más cercanos.

La siguiente función únicamente aplica una fórmula matemática para determinar la hora y el minuto especificados:

$$H = (Num \text{ div } 12) + 7 \qquad M = (Num \bmod 12) * 5$$

```
void ConvierteHorario(unsigned char Num)  {
     unsigned int Aux;
     if (Num > 0)     {
      Num--;
      Aux = Num / 12+7;
            if (Aux < 10) printf("0%d:",Aux);
            else  printf("%d:",Aux);
```

```
            Aux = (Num % 12)*5;
            if (Aux < 10) printf("0%d",Aux);
            else  printf("%d",Aux);
        }
        else printf("-");
    ;
```

7.2.3.9. La sección final de la función void main(void).

Con todas las funciones presentadas anteriormente se puede asegurar que si existe el grupo solicitado, éste será desplegado, sin embargo se encuentra aún el problema de que el usuario no conozca el grupo que está buscando. Podría pasar que escribiera varias combinaciones que según las instrucciones fueran correctas y el grupo no existiera; el programa no sería útil en este caso. Para solucionar el problema se han agregado dos variables:

```
unsigned int GrupoAnt=0;
unsigned int GrupoSig=9999;
```

Estas variables se irán actualizando en cada iteración, es decir que por cada elemento del archivo de horarios se verificarán estos datos. Si se descubre un grupo que se encuentre ubicado entre el grupo solicitado y el valor de la variable, la cual adoptará el nuevo valor, hasta encontrar el grupo inmediato anterior o inmediato siguiente respectivamente. El código fuente que las utiliza se encuentra en la sección principal de la función *main*.

Una vez que se tienen estos dos valores se procede a enviarlos a la página HTML como nuevos formularios de consulta, de forma que al hacer uso de ellos se mande una cadena de consulta referente al grupo anterior o al siguiente. El código fuente a utilizar es el siguiente[7]:

```
printf("<CENTER>\n<TABLE BORDER=0 CELLSPACING=5 CELPADDING=5
       WIDTH=%c630%c>\n",34,34);
printf("<TR>\n");
printf("<TH COLSPAN=3 ALIGN=CENTER
      Background=%c/servicios/horarios/fondo1.jpg%c
      BGCOLOR=%cNAVY%c>
      <Font Size=+1 Color=%cWhite%c><B>Nueva Consulta</B></Font>
      </TH>\n",34,34,34,34,34,34);
printf("</TR>\n");
printf("<TR>\n");
printf("<TD Align=Center BGColor=%cSilver%c>\n",34,34);
printf("<FORM METHOD=%cPOST%c ",34,34);
printf("ACTION=%chttp://informatica.aragon.unam.mx
      /cgi-bin/horarios/horarios.cgi%c>\n",34,34);
printf("<INPUT TYPE=%chidden%c NAME=%cSemestre%c VALUE=%c%s%c>\n"
      ,34,34,34,34,34,elements[Semestre].val,34);
printf("<INPUT TYPE=%chidden%c NAME=%cCarrera%c VALUE=%c%s%c>\n"
```

[7] Por razones de espacio no se coloca el código fuente en forma correcta, sino que en algunas sentencias printf se han agregado algunos saltos de línea. El lector debe tomar todas estas instrucciones como si estuvieran en una sola línea hasta el punto y coma (;).

```
      ,34,34,34,34,34,Carrera,34);
printf("<FONT COLOR=%cblack%c>Grupo: %d</FONT>\n<INPUT TYPE=%chidden%c
      NAME=%cGrupo%c VALUE=%c%d%c><BR>\n",34,34
      ,GrupoAnt >0 ? GrupoAnt:Grupo,34,34,34,34,34
      ,GrupoAnt >0 ? GrupoAnt:Grupo,34);
      printf("<INPUT TYPE=%csubmit%c VALUE=%cGrupo anterior%c>\n"
      ,34,34,34,34);
printf("</FORM>\n");
printf("</TD>\n");
printf("<TD Align=Center BGColor=%cSilver%c>\n",34,34);
printf("<FORM METHOD=%cPOST%c ",34,34);
printf("ACTION=%chttp://informatica.aragon.unam.mx
      /cgi-bin/horarios/horarios.cgi%c>\n",34,34);
printf("<INPUT TYPE=%chidden%c NAME=%cSemestre%c
      VALUE=%c%s%c>\n",34,34,34,34,34,elements[Semestre].val,34);
printf("<INPUT TYPE=%chidden%c NAME=%cCarrera%c
      VALUE=%c%s%c>\n",34,34,34,34,34,Carrera,34);
printf("<FONT COLOR=%cblack%c>Grupo:</FONT>\n<INPUT TYPE=%ctext%c
      NAME=%cGrupo%c VALUE=%c%d%c SIZE=%c4%c><BR>\n",34,34,34,34,34,34,34
      ,Grupo,34,34,34);
printf("<INPUT TYPE=%csubmit%c VALUE=%cNueva consulta%c>\n",34,34,34,34);
printf("</FORM>\n");
printf("</TD>\n");
printf("<TD Align=Center BGColor=%cSilver%c>\n",34,34);
printf("<FORM METHOD=%cPOST%c ",34,34);
printf("ACTION=%chttp://informatica.aragon.unam.mx
      /cgi-bin/horarios/horarios.cgi%c>\n",34,34);
printf("<INPUT TYPE=%chidden%c NAME=%cSemestre%c VALUE=%c%s%c>\n"
      ,34,34,34,34,34,elements[Semestre].val,34);
printf("<INPUT TYPE=%chidden%c NAME=%cCarrera%c VALUE=%c%s%c>\n"
      ,34,34,34,34,34,Carrera,34);
printf("<FONT COLOR=%cblack%c>Grupo: %d</FONT>\n<INPUT TYPE=%chidden%c
      NAME=%cGrupo%c VALUE=%c%d%c><BR>\n",34,34
      ,GrupoSig <9999 ? GrupoSig:Grupo,34,34,34,34,34
      ,GrupoSig < 9999 ? GrupoSig:Grupo,34);
printf("<INPUT TYPE=%csubmit%c VALUE=%cGrupo siguiente%c>\n"
      ,34,34,34,34);
printf("</FORM>\n");
printf("</TD>\n");
printf("</TR>\n");
printf("</TABLE>\n");
printf("</CENTER>\n\n");

printf("<HR WIDTH=100%% SIZE=5 ALIGN=CENTER NOSHADE>\n");

printf("<TABLE BORDER=0 WIDTH=%c100%%%c>\n<TR>\n",34,34);
printf("<TD WIDTH=%c360%c>
      <IMG SRC=%c/servicios/horarios/PLANEACION.GIF%c
      ALT=%Unidad de Planeaci&oacurte;n%c WIDTH=220 HEIGHT=11 BORDER=0
      ALIGN=LEFT></TD>\n",34,34,34,34,34,34,34);
printf("<TD WIDTH=%c220%c>
      <IMG SRC=%c/servicios/horarios/INFORMATICA.GIF%c
      ALT=%cDepartamento de Inform&aacute;tica%c WIDTH=306 HEIGHT=11
      BORDER=0 ALIGN=RIGHT></TD>\n",34,34,34,34,34,34,34);
printf("</TR>\n</TABLE>\n");

printf("</BODY>\n");
printf("</HTML>\n");
```

En esta sección además se ha incluido la parte final del código HTML, que presenta las frases: "UNIDAD DE PLANEACION" y "DEPARTAMENTO DE INFORMATICA", ya que es el lugar en el que se ha desarrollado e implantado todo el sistema.

7.2.4. Los resultados.

Una vez compilado el código fuente, es necesario que se manden al servidor vía FTP los archivos del semestre que se desea presentar. Estos deben mandarse en formato binario, es decir que el formato de los mismos debe pasar tal como se tienen en el servidor Novell Netware.

Teniendo el programa CGI y los archivos en el servidor, el usuario podrá consultar el grupo que requiera desde un navegador. La pantalla mostrada en la figura 7.3. muestra algunos resultados arrojados por el sistema. Suponiendo que se desea saber el horario del grupo 2757 de la carrera de Ingeniería en Computación para el semestre 1999-2, la entrada y salida del sistema sería:

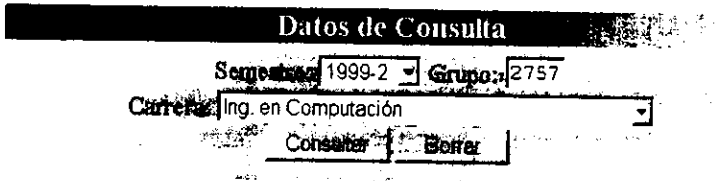

Figura 7.2. Datos de consulta para un grupo de Ingeniería en Computación.

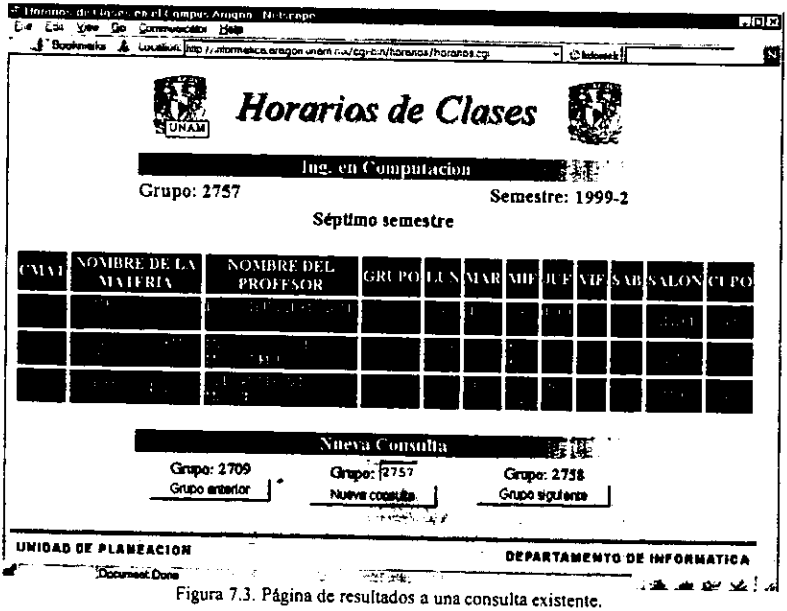

Figura 7.3. Página de resultados a una consulta existente.

Como puede observarse se presentan los siguientes datos:

- Nombre de la carrera.
- Grupo que se consulta.
- Semestre al que pertenece.
- Ciclo escolar.
- El horario en si mismo.

Además de lo anterior se permite cambiar el grupo de consulta sin necesidad de especificar nuevamente la carrera, de forma que se pueda hacer más rápidamente otra consulta. También incluye dos botones que realizan consultas automáticas al grupo anterior y al grupo siguiente.

7.3. Análisis para "Exámenes Finales".

Una vez realizado el sistema para horarios de clases, la forma en la que hará el de exámenes finales es básicamente el mismo, ya que muchos de los módulos son iguales y sólo requieren de pequeñas adaptaciones.

Los archivos que se utilizarán para este caso son:
- FINALES.[SEMESTRE]
- PROFESOR.[SEMESTRE]
- MATERIAS.CRO

7.3.1. Transformación de registros.

Al igual que en el caso anterior es necesario hacer la conversión de los registros a un formato en el que puedan mantener los tamaños originales. Las nuevas estructuras quedan como sigue:

```
typedef struct {
          unsigned Year : 7;
          unsigned Mes  : 4;
          unsigned Dia  : 5;
          unsigned Hora : 5;
          unsigned Min  : 6;
          unsigned Seg  : 5;
       }FechaYHora;
```

```
typedef struct Final{
          unsigned char CveMat2,CveMat1;
          unsigned char Prof1_2,Prof1_1;
          unsigned char Prof2_2,Prof2_1;
          unsigned char Prof3_2,Prof3_1;
          unsigned char Grupo2,Grupo1;
          unsigned char FechaI1[4];
          unsigned char FechaF1[4];
          unsigned char Salon1_2,Salon1_1;
          unsigned char FechaI2[4];
          unsigned char FechaF2[4];
          unsigned char Salon2_2,Salon2_1;
        } Final;
```

Se puede apreciar que se define el tipo *FechaYHora*. Esta estructura permitirá decodificar los campos de fecha: *FechaI1, FechaF1, FechaI2 y FechaF2*. Lo anterior se debe a que en Turbo Pascal se utiliza un tipo propietario que comprime la fecha y la hora en solo 4 bytes y por lo tanto es necesario descompactarla. Con ella la descompresión se hace en un paso como se verá más adelante.

7. 3.2. Necesidades para procesar la información.

Para procesar la información de los exámenes finales, es necesario realizar los mismos pasos que en el caso anterior, como son:

1. *Interfaz de usuario para solicitar carrera, semestre y grupo.*

2. *Procesamiento de formularios.*

3. *Consulta de archivos*. Ya que se tienen los parámetros necesarios, abrir el archivo FINALES.[Semestre] y leer el primer registro. En caso de corresponder al grupo solicitado, obtener de forma secuencial el nombre del profesor y la materia a la que se refiere el registro de los archivos PROFESOR.[Semestre] y MATERIAS.CRO respectivamente.

4. Teniendo toda la información referente al registro que coincida con la petición del usuario, procesarla para mostrarla en formato HTML y enviarla al navegador. Repetir hasta que acabe el archivo y no se tengan más coincidencias.

7.4. Implementación de los módulos para Exámenes Finales.

7.4.1. La interfaz de usuario.

Utilizando la implementación anterior solamente es necesario variar algunas líneas del archivo horarios.html para convertirlo en el de finales.html:

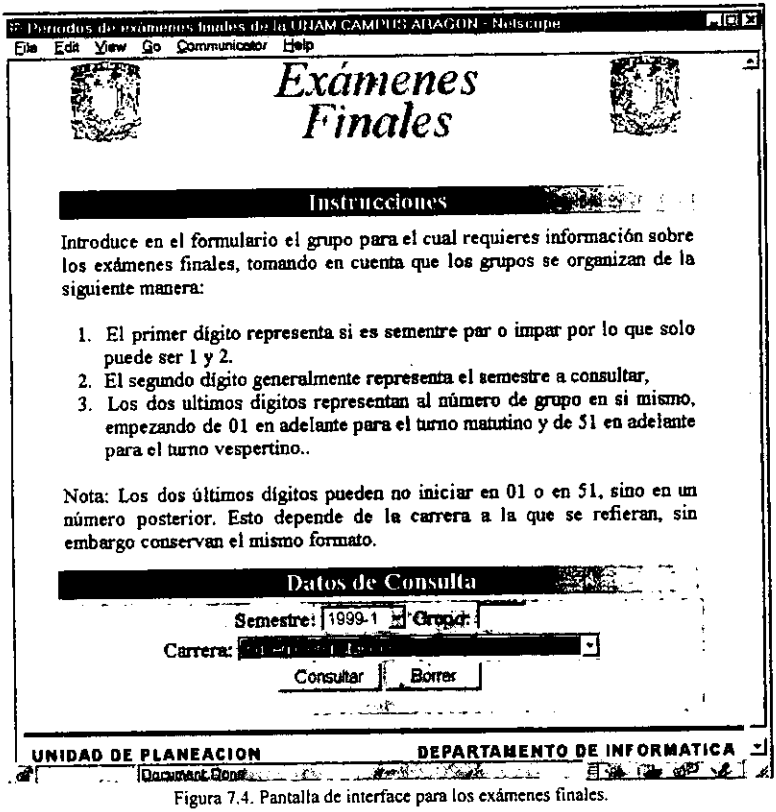

Figura 7.4. Pantalla de interface para los exámenes finales.

- Cambiar el título de la página por el de "*Periodos de exámenes finales de la UNAM CAMPUS ARAGON*", en la etiqueta <TITLE>.
- Cambiar la imagen que muestra el titulo de horarios por la de "Exámenes Finales".
- El resto de la página permanece igual, a excepción de la etiqueta <FORM>...</FORM>, ya que aunque las opciones que se manejan son las mismas, el atributo ACTION= se debe dirigir a "*http://informatica.aragon.unam.mx /cgi-bin/horarios/finales.cgi*".

El resultado de los cambios se muestra en la figura 7.4.

7.4.2. Procesamiento y presentación de los resultados.

Muchos de los módulos desarrollados para Horarios de Clases se repiten aquí. Si bien puede observarse la única diferencia entre un sistema y otro es el formato del archivo que contiene los datos; para este caso el archivo HORDAT.[Semestre] y FINALES.[Semestre].

Dada esta situación únicamente se presentarán los módulos que tengan peso en el sistema, y se dará por presentado lo que no tenga cambios significativos.

7.4.2.1. La función void DespliegaDatos(Final Dato).

Esta función es muy parecida a la anterior, salvo algunas diferencias, considerando que los datos son otros, además de que es aquí donde se convierten los campos de fecha y hora de 4 bytes al formato correcto.

```
void DespliegaDatos(Final Dato)   {

     int AuxSalon1 = Dato.Salon1_1*256+Dato.Salon1_2;
     int AuxSalon2 = Dato.Salon2_1*256+Dato.Salon2_2;

     char CadenaI1[4],CadenaI2[4];
     char CadenaF1[4],CadenaF2[4];

     FechaYHora *FechaI1 = CadenaI1;
     FechaYHora *FechaI2 = CadenaI2;
     FechaYHora *FechaF1 = CadenaF1;
     FechaYHora *FechaF2 = CadenaF2;

     CadenaI1[0] = Dato.FechaI1[3];CadenaI2[0] = Dato.FechaI2[3];
     CadenaI1[1] = Dato.FechaI1[2];CadenaI2[1] = Dato.FechaI2[2];
     CadenaI1[2] = Dato.FechaI1[1];CadenaI2[2] = Dato.FechaI2[1];
     CadenaI1[3] = Dato.FechaI1[0];CadenaI2[3] = Dato.FechaI2[0];
     CadenaF1[0] = Dato.FechaF1[3];CadenaF2[0] = Dato.FechaF2[3];
     CadenaF1[1] = Dato.FechaF1[2];CadenaF2[1] = Dato.FechaF2[2];
     CadenaF1[2] = Dato.FechaF1[1];CadenaF2[2] = Dato.FechaF2[1];
     CadenaF1[3] = Dato.FechaF1[0];CadenaF2[3] = Dato.FechaF2[0];

     printf("<TR ALIGN=CENTER BGCOLOR=%cNavy%c>\n",34,34);
     printf("<TD ALIGN=LEFT><FONT SIZE=-1>");
     BuscaMat(Dato.CveMat1*256+Dato.CveMat2);
     printf("</FONT><BR></TD>\n");

     printf("<TD><FONT SIZE=-1>");
     printf("%d<BR>",Dato.CveMat1*256+Dato.CveMat2);
     printf("</FONT></TD>\n");

     printf("<TD ALIGN=LEFT><FONT SIZE=-1>\n<LI> ");
     BuscaProf(Dato.Prof1_1*256+Dato.Prof1_2);
     printf("<BR>\n<LI>");
     BuscaProf(Dato.Prof2_1*256+Dato.Prof2_2);
     printf("<BR>\n<LI>");
     BuscaProf(Dato.Prof3_1*256+Dato.Prof3_2);
     printf("<BR>\n</FONT></TD>\n");
```

```
    printf("<TD><FONT SIZE=-1>");
    DespliegaFecha(FechaI1);
    printf("<BR>");
    DespliegaFecha(FechaI2);
    printf("</FONT><BR></TD>\n");

    printf("<TD><FONT SIZE=-1>");
    DespliegaHora(FechaI1);
    printf("-");
    DespliegaHora(FechaF1);
    printf("<BR>");
    DespliegaHora(FechaI2);
    printf("-");
    DespliegaHora(FechaF2);
    printf("</FONT><BR></TD>\n");

    printf("<TD><FONT SIZE=-1>");
    printf("A%d<BR>",AuxSalon1);
    printf("A%d</FONT><BR></TD>\n",AuxSalon2);

    printf("<TD><FONT SIZE=-1>");
    printf("1a Vuelta<BR>");
    printf("2a Vuelta</FONT><BR></TD>\n");
    printf("</TR>\n");
  }
```

Este código presenta la información de la consulta en formato HTML, considerando una nueva forma de desplegar el nuevo tipo de horarios. Hace uso de las funciones *DespliegaFecha* y *DespliegaHora* que son las que se encargan de presentar la fecha y la hora respectivamente. Sus códigos se presentan a continuación.

```
void DespliegaHora(FechaYHora *Hora)   {
     if (Hora->Hora < 10) printf("0%d:",Hora->Hora);
     else printf("%d:",Hora->Hora);
     if (Hora->Min < 10) printf("0%d",Hora->Min);
     else printf("%d",Hora->Min);
  }
  void DespliegaFecha(FechaYHora *Fecha)   {
     if (Fecha->Mes > 0) Fecha->Mes--;
     if (Fecha->Dia < 10) printf("0%d%c",Fecha->Dia,47);
     else printf("%d%c",Fecha->Dia,47);
     printf("%s%c%d",Meses[Fecha->Mes],47,Fecha->Year+1980);
  }
```

Estas dos funciones únicamente imprimen la fecha y la hora en el formato de un reloj, es decir que cuando es necesario agregan el cero a la izquierda de la unidad y en el caso de los meses, escribe el nombre abreviado del mismo auxiliándose de el arreglo Meses que se muestra a continuación:

```
char *Meses[] = {"ENE","FEB","MAR","ABR","MAY","JUN",
                 "JUL","AGO","SEP","OCT","NOV","DIC"};
```

El resto del código es realmente parecido al del proyecto anterior, por lo que no será presentado en esta sección. Básicamente la diferencia es que el nombre del archivo es FINALES y no HORDAT.

7.4.3. Los resultados.

Una vez compilado el código fuente, es necesario que se manden al servidor vía FTP los archivos del semestre que se desea presentar. Estos, al igual que todos los que se manejan, deben contener los datos en el formato original.

Teniendo el programa CGI y los archivos en el servidor el usuario podrá consultar el grupo que requiera desde un navegador. La pantalla mostrada en la figura 7.6. muestra algunos resultados arrojados por el sistema. Suponiendo que se desea conocer la fecha y el horario para los exámenes finales del grupo 1701 de la carrera de Economía en el semestre 1999-1, la entrada del sistema sería:

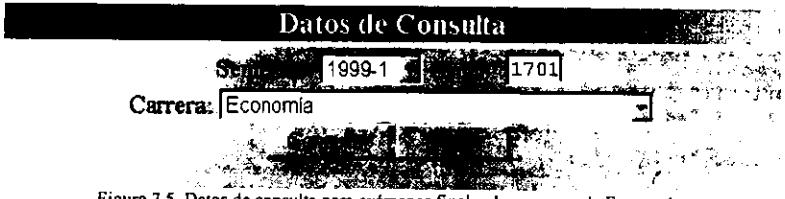

Figura 7.5. Datos de consulta para exámenes finales de un grupo de Economía.

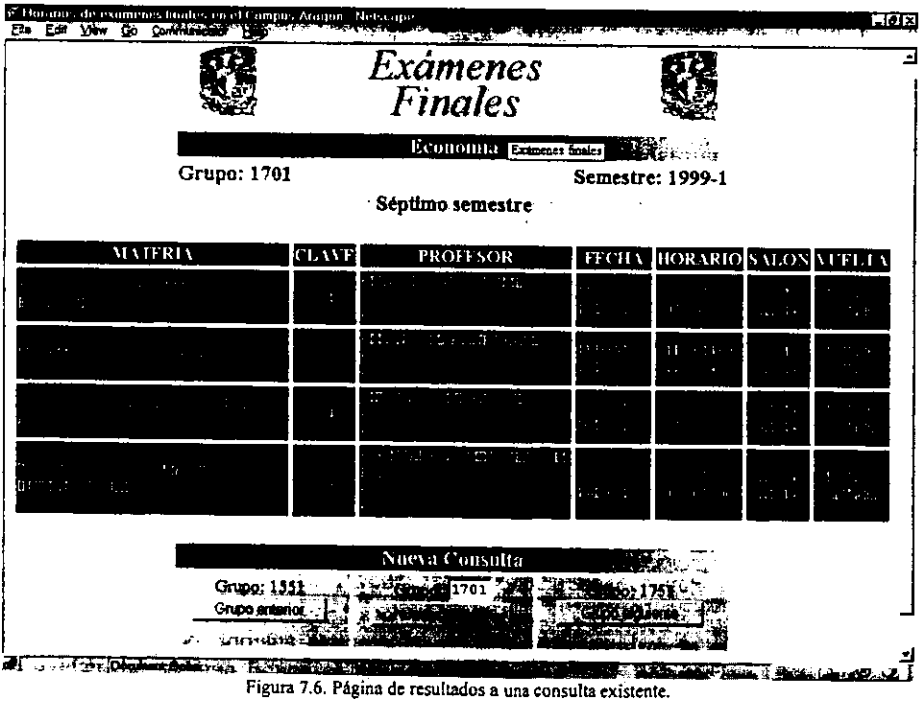

Figura 7.6. Página de resultados a una consulta existente.

Esta página presenta los siguientes datos:

- Nombre de la carrera.
- Grupo que se consulta.
- Semestre al que pertenece.
- Ciclo escolar.

El horario de exámenes que cuenta con:

- El nombre de la materia.
- La clave de la misma.
- El nombre de tres sínodos (en caso de haberse capturado los tres, ya que pueden ser dos o uno).
- La fecha.
- La hora.
- El salón.
- Y la vuelta a la que pertenecen dichas fecha y hora (primera o segunda vuelta).

7.5. Análisis para "Exámenes Extraordinarios".

El sistema para la publicación de las fechas de exámenes extraordinarios es el más sencillo de los tres en el sentido de que todas las instrucciones que pudieran necesitarse ya fueron desarrolladas en los otros sistemas. Además, en este caso no será necesario crear un control para grupo anterior o grupo siguiente considerando que en estos exámenes no se necesita especificar el grupo para determinar el horario, puesto que estos son los mismos para todos.

Los archivos a manejar aquí son:

- PROFESOR.[SEMESTRE]: archivo de profesores.
- MATERIAS.CRO: archivo de Materias.
- EXTRA_A.[SEMESTRE_IMPAR]: extraordinarios de primera vuelta semestre impar.
- EXTRA_B.[SEMESTRE_IMPAR]: extraordinarios de segunda vuelta semestre impar.
- EXTRA_C.[SEMESTRE_PAR]: extraordinarios de primera vuelta semestre par.
- EXTRA_D.[SEMESTRE_PAR]: extraordinarios de segunda vuelta semestre impar.

7.5.1. Transformación de registros.

Al igual que en el caso anterior es necesario hacer la conversión de los registros a un formato en el que puedan mantener los tamaños originales. Las nuevas estructuras quedan de la siguiente manera:

```
char *Vueltas[] = {"_A","_B","_C","_D"};

typedef struct Examen{
          unsigned char Prof1_2,Prof1_1;
          unsigned char Prof2_2,Prof2_1;
          unsigned char CveMat2,CveMat1;
          unsigned char Salon1_2,Salon1_1;
          unsigned char Fecha1[4];
          unsigned char Salon2_2,Salon2_1;
          unsigned char Fecha2[4];
          unsigned char Turno;
        } Examen;
```

La constante de vueltas es usada para determinar que archivo se va a utilizar de acuerdo a la petición del usuario. Considerando el semestre y la vuelta, el archivo puede ser EXTRA_A, EXTRA_B, EXTRA_C y EXTRA_D. Se debe tomar en cuenta que no siempre estarán disponibles las fechas en una vuelta determinada, ya que estas son publicada conforme son capturadas por las jefaturas de carrera.

7. 5.2. Necesidades para procesar la información.

Para procesar la información de los exámenes extraordinarios es necesario realizar los mismos pasos que en el caso anterior, como son:

1. *Interfaz de usuario para solicitar carrera, semestre y vuelta.* La diferencia que existe entre ésta y las que se han manejado hasta el momento es que solamente es necesario seleccionar las opciones de los menús. Esto es por que no existe diferencia de horario entre un grupo y otro. Por ello se realizará una consulta a la totalidad de los exámenes extraordinarios.

2. *Procesamiento de formularios.*

3. *Consulta de archivos.* Ya que se tienen los parámetros necesarios, abrir el archivo EXTRA_[Vuelta].[Semestre] y leer desde el primer registro. Para cada caso obtener de forma secuencial el nombre del profesor y la materia a la que se refiere el registro de los archivos PROFESOR.[Semestre] y MATERIAS.CRO respectivamente.

4. Teniendo los datos completos para el primer registro, repetir el proceso hasta llegar al final del archivo, es decir, hasta que no se tengan más registros por presentar.

7.6. Implementación de los módulos para Exámenes Extraordinarios.

7.6.1. La interfaz de usuario.

La interfaz de usuario es un tanto más fácil de realizar y cuenta con elementos más específicos. Tomando en cuenta que ahora no es necesario capturar el grupo, se puede prescindir de las instrucciones para la formación de un grupo.

- Cambiar el título de la página por el de "*Periodos de exámenes extraordinarios de la UNAM CAMPUS ARAGON*", en la etiqueta <TITLE>.
- Cambiar la imagen que muestra el título de horarios por la de "Exámenes extraordinarios".
- La etiqueta <FORM>...</FORM> se modifica para que apunte hacia la dirección "*http://informatica.aragon.unam.mx/cgi-bin/horarios/extra.cgi*", además de cambiar la variable Grupo por la de Vuelta, quedando como sigue:

```
<SELECT NAME="Vuelta" SIZE="1">
<OPTION>Primera Vuelta
<OPTION>Segunda Vuelta
</SELECT><BR>
```

El resultado de los cambios se muestra en la figura 7.7.

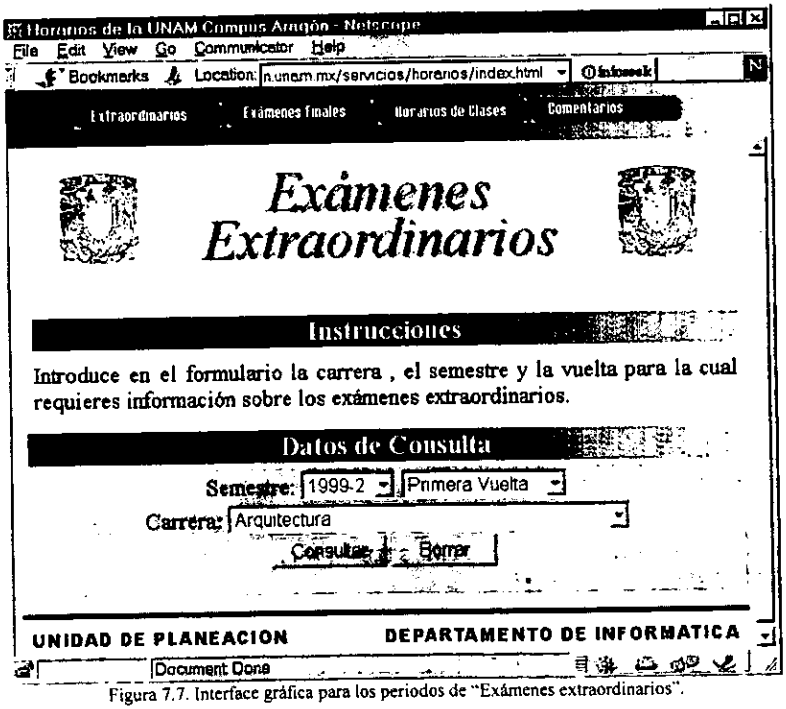

Figura 7.7. Interface gráfica para los periodos de "Exámenes extraordinarios".

7.6.2. Procesamiento y presentación de los resultados.

El código necesario para realizar esta operación es realmente parecido a los anteriores, sin embargo es más corto. Prácticamente sólo es necesario abrir el archivo solicitado y desplegar todos los registros.

7.6.2.1. La función void DespliegaDatos(Examen Dato).

Esta función hace uso de las funciones que se presentaron anteriormente , por lo que sólo se explicará brevemente su funcionamiento.

```
void DespliegaDatos(Examen Dato) {

     int AuxSalon1 = (Dato.Salon1_1 << 8)+Dato.Salon1_2;
     int AuxSalon2 = (Dato.Salon2_1 << 8)+Dato.Salon2_2;
     char Cadena1[4],Cadena2[4];

     FechaYHora *Fecha1 = Cadena1;
     FechaYHora *Fecha2 = Cadena2;

     Cadena1[0] = Dato.Fecha1[3];Cadena2[0] = Dato.Fecha2[3];
     Cadena1[1] = Dato.Fecha1[2];Cadena2[1] = Dato.Fecha2[2];
     Cadena1[2] = Dato.Fecha1[1];Cadena2[2] = Dato.Fecha2[1];
     Cadena1[3] = Dato.Fecha1[0];Cadena2[3] = Dato.Fecha2[0];

     printf("<TR>\n");
     printf("<TD ROWSPAN=3><FONT SIZE=-1>");
     printf("%d</FONT><BR>",Dato.CveMat1*256+Dato.CveMat2);
     printf("</TD>\n");

     printf("<TH COLSPAN=4 ALIGN=LEFT><FONT SIZE=-1>\n");
     BuscaMat(Dato.CveMat1*256+Dato.CveMat2);
     printf("</FONT></TH>\n</TR>\n");

     printf("<TR>\n<TD ALIGN=LEFT><LI><FONT SIZE=-1>");
     BuscaProf(Dato.Prof1_1*256+Dato.Prof1_2);
     printf("</FONT></TD>\n");

     printf("<TD ALIGN=CENTER><FONT SIZE=-1>");
     DespliegaHora(Fecha1);
     printf("</FONT><BR>");
     printf("</TD>\n");

     printf("<TD ALIGN=CENTER><FONT SIZE=-1>");
     printf("A%d</FONT><BR>",AuxSalon1);
     printf("</TD>\n");

     printf("<TD ALIGN=CENTER><FONT SIZE=-1>");
     DespliegaFecha(Fecha1);
     printf("</FONT><BR></TD>\n</TR>\n");

     printf("<TR>\n<TD ALIGN=LEFT><LI><FONT SIZE=-1>");
     BuscaProf(Dato.Prof2_1*256+Dato.Prof2_2);
     printf("</FONT></TD>\n");
```

```
    printf("<TD ALIGN=CENTER><FONT SIZE=-1>");
    DespliegaHora(Fecha2);
    printf("</FONT><BR></TD>\n");

    printf("<TD ALIGN=CENTER><FONT SIZE=-1>");
    printf("A%d</FONT><BR>",AuxSalon2);
    printf("</TD>\n");

    printf("<TD ALIGN=CENTER><FONT SIZE=-1>");
    DespliegaFecha(Fecha2);
    printf("</FONT><BR></TD>\n");

    printf("</TR>\n");
}
```

La primera parte de la función se encarga de inicializar las variables que requieren de una transformación para funcionar. En estos casos se hace uso de apuntadores para voltear variables ya que Pascal en la PC maneja el almacenamiento de las variables en orden contrario a como lo hace el lenguaje C. El resto de la función se encarga únicamente de desplegar todos los elementos de la tabla.

7.6.3. Los resultados.

Una vez compilado el código fuente y transferidos los archivos necesarios por FTP a su lugar correspondiente sólo se requiere de un navegador para poder recibir el resultado de la consulta a alguna vuelta de exámenes extraordinarios. La figura 7.9. muestra la salida de una consulta hecha al semestre 1999-2, primera vuelta de la carrera de Pedagogía (fig. 7.8).

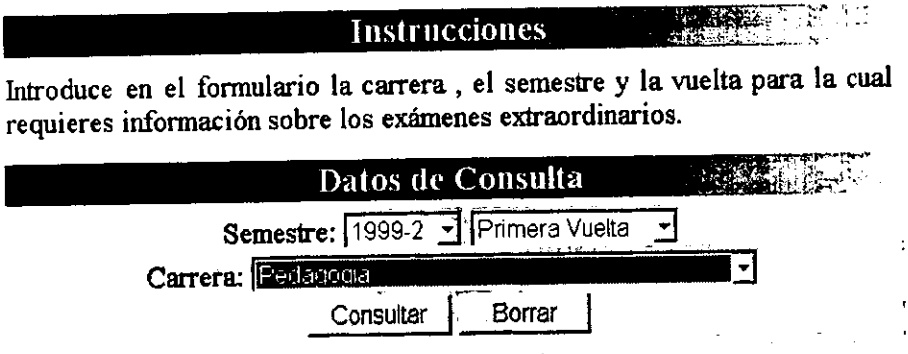

Figura 7.8. Datos de consulta para exámenes extraordinarios de Pedagogía.

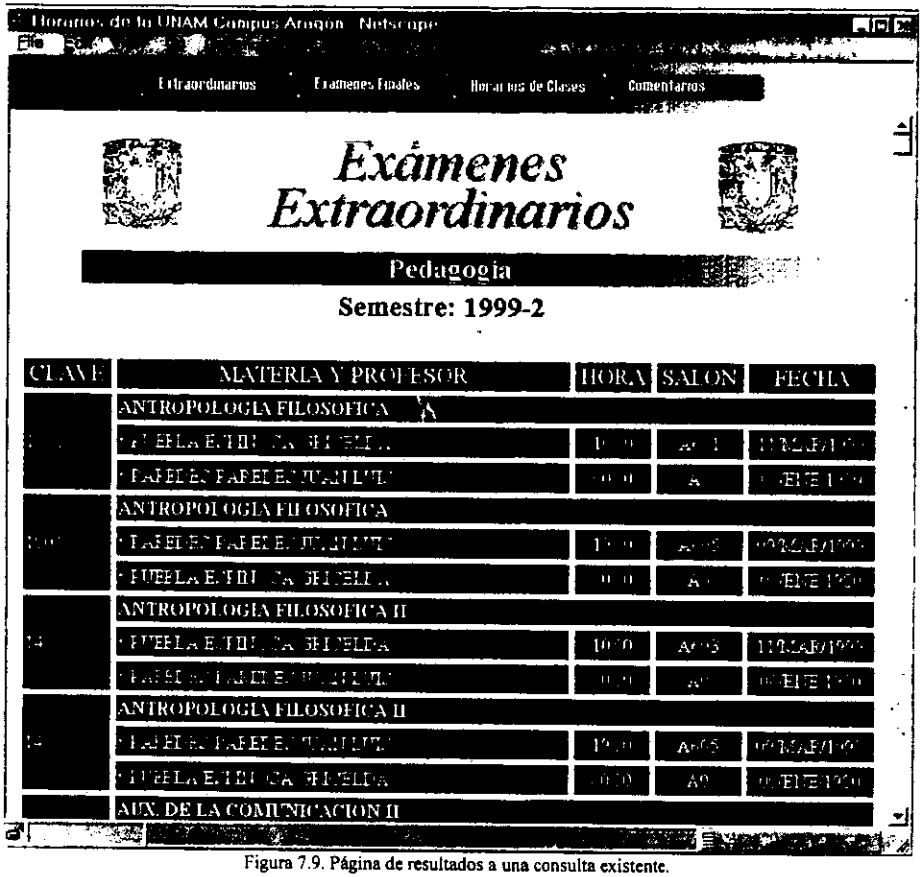

Figura 7.9. Página de resultados a una consulta existente.

Esta página presenta los siguientes datos:

- Nombre de la carrera.
- Ciclo escolar.

El horario de exámenes que cuenta con:

- La clave de la carrera.
- El nombre de la misma.
- El nombre de dos sínodos (puede ser uno).
- La hora.
- El salón.
- La fecha.

7.7. Administración del sistema.

La administración del sistema se hace de forma calendarizada. El H. Consejo Técnico de la Escuela define el calendario escolar para cada uno de los semestres. En el se especifican las fechas en las que se realizarán las inscripciones a las materias que se desean tomar, además de que se define cuándo se aplicarán los exámenes finales y extraordinarios, entre otras fechas.

En base al calendario las jefaturas de carrera capturan en Cronos todos los datos necesarios para la publicación de las fechas y horas de exámenes o inscripciones. Los datos se guardan en los archivos que se han manejado a lo largo de la tesis.

El administrador del sistema debe mantenerse pendiente del calendario para transferir los archivos correspondientes al semestre y las doce carreras al directorio adecuado en el servidor para que el CGI pueda accesar a ellos. En caso de que un usuario pretenda consultar un semestre no activado, el sistema presenta una pantalla de error en donde se explica los motivos por los que no se ha dado de alta esa información.

Otra cosa que se debe hacer cada semestre, es modificar los archivos "horarios.html", "extra.html" y "finales.html" para activar el la lista de opciones el nuevo semestre, dejando únicamente éste y el semestre actual.

Dos semanas antes de la realización de los exámenes o de la fecha de inscripciones, se deben enviar los archivos de datos. Una semana antes de la fecha se deben transferir diariamente los archivos de datos, ya que las carreras realizan cambios de ultimo momento durante la misma y aún durante las inscripciones.

Básicamente es la forma en la que se administra el sistema, como puede verse, sólo es necesario mantenerse al tanto de las fechas del calendario.

8

Capítulo VIII

Consideraciones finales de seguridad.

Capítulo VIII. Consideraciones finales de seguridad.

Siempre que un sistema de computadoras es conectado al mundo exterior se abren potenciales fugas de seguridad, que pueden ser aprovechadas por alguien con intenciones maliciosas. Al instalar un servidor Web, un servicio de correo electrónico, un sitio FTP, etc. o simplemente conectar una computadora a la red, se está colocando explícitamente un tipo de entrada que, al mismo tiempo, debe mantener alejados a los intrusos.

En Internet existen buenos programadores que conocen perfectamente el funcionamiento de la red en todos los niveles. Ellos muchas veces tratan de entrar al sistema únicamente como un reto para demostrar sus conocimientos, pero en otros casos solo lo hacen para dañar el sistema.

Ningún equipo de computo esta excento de ser atacado, por lo que es necesario tomar precauciones. Se debe saber que sin importar todos los métodos existentes para seguridad, ni lo rígida que esta sea, un individuo decidido a infringir el sistema con los conocimientos necesarios, lo hará.

En este capítulo se hacen recomendaciones sobre seguridad que pueden ser aplicados de forma muy general. Las precauciones que deben tomarse en diferentes sistemas dependen mucho de las características del mismo.

8.1. Clases de peligros para la seguridad.

En los sistemas que serán utilizados en una red, como por ejemplo los desarrollados utilizando CGI's, existen tres o cuatro clases principales de brechas de seguridad de las que debe tenerse cuidado :

- Fugas de información acerca del servidor, que a un intruso darían acceso privilegiado al sistema.
- Software con errores que permiten la ejecución de comandos peligrosos del Host, lo que puede dañar el sistema.
- Publicación accidental de información confidencial almacenada en el servidor.
- En ciertos casos, la intercepción de información privada, personal o confidencial que se transmite hacia el servidor desde un usuario confiado (por ejemplo, contraseñas, direcciones personales, información de tarjetas de crédito, etcétera).

Estas categorías de los riesgos de seguridad se traslapan tanto que quizá no sirvan muy bien para delimitar las áreas de estudio. Las técnicas que se presentarán a continuación abordan estas áreas. Es importante que se tomen todas en cuenta sin descuidar las demás.

8.2. La seguridad en CGI.

Los riesgos en la seguridad suelen ser diferentes entre los distintos sistemas operativos. Los sistemas como UNIX que son más complejos, muy flexibles y con muchas características suelen ser más vulnerables a los ataques, sobre todo cuando se trata de un sistema múltiusuario. El ataque a este tipo de sistemas puede utilizarse con más facilidad para atacar a todo un grupo de ellos, multiplicando efectivamente los riesgos existentes en los mismos.

8.3. Axiomas de seguridad.

Los siguientes puntos muestran algunas fallas y violaciones de seguridad comunes en el mundo del WWW y el CGI. La información que aquí se presenta obviamente no es única ni es regla general, por lo que se deben de tomar siempre las precauciones necesarias para cada sistema en particular.

- **Nunca se debe suponer que algo está a salvo o que es perfecto.**

Un error muy común cuando se hace un CGI es creer que ya todo es perfecto. Por muchas precauciones que se hayan tomado, siempre se debe de vigilar el sistema realizado. En cualquier momento alguien pueden intentar violar la seguridad del mismo. Siempre que sea posible se debe mejorar el esquema de seguridad.

- **Se debe leer el código.**

Muchas veces cuando se tiene un problema de programación se opta por recurrir a programas externos, es decir, a programas que se encuentran regados en Internet. Si bien es cierto que la mayoría de ellos son serios, muchas veces son solo puertas de entrada para los autores. Por lo tanto, cuando se utilice un programa externo debe tenerse la precaución de revisar el código fuente del mismo y entenderlo totalmente para evitar que sean trampas. Si no se comprenden en su totalidad los procesos que realiza, se debe evitar usarlo hasta poder entenderlo.

- **Deshabilitar lo que no se use.**

En sistemas múltiusuario es común que varios de ellos quieran tener páginas personales y por supuesto CGI's propios. Es bueno deshabilitar la característica del servidor que permite colocar CGI's en cualquier lugar del disco duro. Es mejor que todos los programas para Internet radiquen en un área especial en la que no se corran riesgos.

- **Si no se usa, debe borrarse.**

Unix se caracteriza por contar con muchos servicios, entre ellos ftp, nfs, correo electrónico, etc. Cuando en el sistema no sea necesario tener estos servicios, deben borrarse. Entre menos puertas de entrada se tengan, más difícil será penetrar ilegalmente al sistema.

- **Mantener CGI´s fuera del árbol de documentos.**

A menos que sea absolutamente necesario que estén en el árbol de documentos, deben ponerse todos los archivos de datos, listas de contraseñas y cualquier archivo secundario fuera de éste. Los archivos que no son susceptibles a ser empleados por GET o POST no necesitan estar dentro de dicho árbol.

- **Verificar los permisos de los archivos.**

Los permisos de archivo son muy importantes en la seguridad. Deben establecerse intencionalmente identificadores de usuario y de grupo. Los permisos predeterminados no son necesariamente la mejor opción para la instalación de un servidor. Esto es generalmente cuando se tienen páginas personales o de grupo que se desea que no se modifiquen por nadie más.

- **No permitir el acceso a los archivos de configuración local.**

Cuando se tienen archivos de acceso local al alcance de los clientes, se abre un agujero de seguridad. Una petición determinada puede provocar que el servidor transfiera información confidencial. Cuando se tengan archivos de configuración de claves o permisos para acceso restringido, se deben usar nombres que no sean fácilmente descubiertos.

- **No permitir que los usuarios suban archivos al árbol de directorios.**

Nunca deben de ponerse los servicios de HTTP y de FTP dirigidos al mismo árbol de directorios, ya que un usuario cualquiera podría mandar por el segundo un programa CGI destructivo y después correrlo desde el primero. Los resultados de esto pueden ser catastróficos considerando que el CGI puede hacer casi cualquier cosa, incluso borrar información.

- **Revisar con frecuencia los registros.**

Es una buena costumbre revisar los archivos LOG de todos los servicios de un sistema. Se deben buscar líneas sospechosas. Cualquiera de ellas que se coloque intentando obtener

información sobre un archivo restringido, ejecución de algún programa o intento de accesar a directorios del sistema, debe ser tomada en cuenta y cuidarse de las agresiones provenientes de la dirección electrónica de donde se haga la petición.

- **No se debe permitir el cambio de contraseña desde el exterior.**

Cuando se tienen sistemas que requieren uso de claves de acceso, no debe permitirse que desde el exterior puedan ser cambiadas; las claves pueden ser interceptadas por Internet, por lo que es preferible utilizar otros medios para hacer este proceso.

- **No se acepte números de tarjeta de crédito por Internet.**

El motivo es similar al anterior. Cuando se intercepta una clave de acceso, el agresor puede entrar libremente al servidor. El daño es directamente a éste. El problema de los números de tarjeta de crédito es mayor para el usuario, puesto que otra persona podría usarlo para comprar a nombre de otra persona. Aunque en este sistema no se utilizan las claves de acceso ni las tarjetas, es bueno tomarlo en cuenta en los que así lo requieran.

- **No pasar una entrada de usuario directamente a un shell de comandos.**

En algunos sistemas de CGI, se utilizan formularios para mandar correo electrónico, obtener opiniones de los usuarios o para poner anuncios en línea. Un problema frecuente es que el programador pasa la información directamente a un shell de comandos o a otra página Web. Considerando que en un shell se pueden ejecutar comandos del sistema, pasarlo directamente puede ocasionar que el servidor se caiga o que transmita información confidencial. Siempre que se necesite una entrada del usuario para mandarlo a un shell, es necesario revisar que no contenga instrucciones peligrosas.

- **No usar Shell's de comandos para programas largos.**

El CGI puede utilizar cualquier lenguaje de programación para ser generado; desafortunadamente eso incluye a los shell's. Estos tienen la desventaja de que, si son muy grandes, pueden quedarse colgados en algún momento o mantener recursos del sistema ocupados aun después de haber terminado su trabajo. Por esta razón, siempre que sea posible debe usarse un lenguaje más confiable, como el C o el Perl, claro que con cuidado.

- **Cuidado con los búffers en C.**

Siempre que sea posible debe usarse *malloc*() para asignar el espacio del buffer explícitamente necesario para la petición. Es un error muy frecuente proponer un tamaño

fijo y suponer que no será mayor. Una forma que tienen los hackers para dañar los sistemas es mandar grandes cantidades de información a un formulario en un intento por sobrepasar las especificaciones; si esto les funciona pueden entrar al directorio raíz.

- **Verificar todos los valores.**

En un formulario pueden ser transferidos datos visibles y ocultos para el usuario. Cuando se utilizan los segundos puede pensarse erróneamente que éstos son inalterables siendo lo anterior totalmente falso. Un usuario cualquiera podría modificar el código fuente para alterar la información provocando que las peticiones sean diferentes a lo esperado.

- **No suponer que un formulario es único.**

Al igual que con los campos ocultos cualquier información puede ser modificada desde otro servidor. Es de todos sabido que el contenido de una página puede ser visualizado, grabado y alterado en cualquier navegador. Por este motivo se debe de verificar que los valores, e incluso los nombres de las variables siempre sean los mismos y en la misma cantidad.

- **Ejecutar el servidor HTTP como nobody.**

Los programas CGI se ejecutan con la misma identificación de usuario y de grupo que el servidor, por lo que no debe ejecutarse este como *root*. La cuenta *nobody* se usa por lo general para el servidor HTTP debido a que este usuario tiene una autoridad en extremo baja.

8.4. Escalas de seguridad.

Los axiomas de seguridad mostrados anteriormente sólo son algunas de las precauciones que deben tomarse al desarrollar sistemas. Sin embargo, también se debe tener en cuenta que es lo que se está protegiendo y la forma en la que se hace la transmisión de la información.

En el sistema realizado conjuntamente con esta tesis se requiere de la entrada de información por parte del usuario para conocer sus requerimientos. La petición de éste no necesita ser protegida con demasiada tenacidad, ya que aún en el supuesto de que pudiera ser interceptada, no es de relevancia para nadie que se consulte un grupo u otro.

Al desarrollarse sistemas como apartado de cursos, inscripciones a materias, pago para tramites escolares, o en otro tipo de aplicaciones como las realizadas por bancos, centros comerciales, mercadeo por Internet, registro de software, etc., la seguridad debe ser

extremadamente rígida. Para ello existen técnicas de programación especiales que hacen más difícil de penetrar o interceptar la información.

Con esto se pretende llegar a la siguiente reflexión : "La protección de la información es costosa y complicada; entre más confiable sea esta, mayores serán los recursos necesarios para lograrla. Cabe resaltar que toda la información es importante y que todos los sistemas deben ser protegidos, pero en función a la importancia de estos recursos es que se deben tomar las decisiones pertinentes para cada caso".

C

Conclusiones

Conclusiones.

A lo largo de la realización de este proyecto, así como de la implantación y mantenimiento del mismo, se obtienen las siguientes apreciaciones:

Internet ha crecido en forma desmesurada en los últimos cuatro años, conforme se va haciendo común el uso del correo electrónico y las páginas Web, estas dejan de ser solo para unos cuantos elegidos para convertirse en una herramienta básica de comunicación.

En el mismo periodo de tiempo han aparecido cientos de aplicaciones para el Web, entre ellos, lenguajes de programación como Java, JavaScript, Vbscript, Visual Basic, Delphi, etc. Otro tanto de estas, se dedican a poner bases de datos en páginas Web. Sin embargo puede decirse que una parte muy importante de la forma en la que estos productos operan se basa en el funcionamiento del llamado CGI. Esta interfaz común de puerta de entrada se sigue utilizando a la fecha por su enorme flexibilidad.

La forma en la que funciona el CGI es con la transmisión de la información de cliente de Web hacia el servidor a través de variables de ambiente, las cuales pueden ser manipuladas con casi cualquier lenguaje de programación.

Con la existencia del CGI y del HTML las posibilidades solamente tienen como límite la capacidad del programador, puesto que puede hacerse casi cualquier tipo de aplicación basada en navegador, siendo que además puede adoptar las nuevas tecnologías de forma transparente.

El sistema de consulta de horarios se ha podido desarrollar con los recursos que cuenta el Departamento de Informática; esto involucra que no es necesario un gasto mayor para obtener soluciones viables a problemas informáticos.

El sistema ha sido implementado desde el semestre 1999-1 obteniéndose buenos resultados, sin embargo en el periodo de inscripciones para el semestre 1999-2 se ha visto el verdadero éxito del sistema ya que el número de consultas se incrementó notablemente, como se muestra en la tabla 9.1.

Como se puede observar, en los meses en los que ha habido inscripciones, el sistema ha mostrado su efectividad. En el primer semestre de su funcionamiento la mayoría de los estudiantes no conocían el sistema, mientras que en el siguiente semestre el número de alumnos en consultarlo aumentó considerablemente.

Con estos resultados se puede afirmar sin temor a equivocaciones que el sistema es un proyecto que realmente beneficia a la comunidad estudiantil del plantel.

Otra apreciación importante de este proyecto es la cantidad de conocimientos que fueron necesarios para su realización. En algún momento puede pensarse que es trivial, sin

embargo el número de detalles que deben tomarse en cuenta y las diferentes opciones para realizar un sistema enriquecen al diseñador, programador y operador del sistema con datos y tecnologías muy variadas entre ellas :

- Ingeniería de Sistemas.
- Administración y mantenimiento de proyectos.
- Lenguajes d programación.
- Manejo de servidores Web.
- La interfaz común de puerta de entrada (CGI) como interface de comunicación entre el usuario y el servidor.
- El HTML como medio de comunicación entre el servidor y el usuario.
- La seguridad de los sistemas, sobretodo los que funcionan sobre Internet.
- La integración de todo lo anterior

Mes	Horarios de clases	Extraordinarios	Exámenes Finales
JUN-1998	849	25	22
JUL-1998	2545	30	16
AGO-1998	**6657**	**125**	**102**
SEP-1998	0	0	0
OCT-1998	833	40	40
NOV-1998	1285	55	64
DIC-1998	1708	118	118
ENE-1999	**32522**	**455**	**144**
FEB-1999	3390	123	60
MAR-1999	1850	106	24
9 de ABR-1999	228	28	137

Tabla 9.1 : Número de consultas por mes.

Por último, se debe resaltar que lo principal que se buscaba con este proyecto fue solucionar un problema de comunicación con los alumnos. Darles la posibilidad de consultar los horarios desde su casa, su trabajo, o la misma escuela, y en algunos casos, pudieran consultarlos desde el extranjero; objetivo alcanzado desde el primer momento en que se puso en funcionamiento.

B

Bibliografía

Bibliografía.

- Brian W. Kernighan y Dennis M. Ritchie.
  "El lenguaje de programación C".
  Segunda Edición.
  Editorial : Prentice-Hall.
  México 1991.
  Pp 294.

- Diego Bonilla y José de Jesús Del Toro
  "Mercadotécnia e imagen en Internet"
  Editorial : Grupo Editorial Iberoamérica.
  México 1996.
  Pp 228.

- Herbert Schildt.
  "Turbo C/C++ 3.1 Manual de Referencia".
  Editorial McGraw-Hill
  España 1994.
  pp. 1029.

- Laura Lemay
  "Aprendiendo HTML 3.0 en una semana"
  Segunda Edición.
  Editorial Prentice-Hall
  México 1996.
  Pp 519.

- Michael Afergan, Rick Darnell, Brian Farrar, Russ Jacobs, David Medinets, Robert Mullen, Mícheál Ó Foghlú.
  "Programación en Web 6 en 1"
  Editorial Prentice-Hall
  México 1997.
  Pp 1084

- Nell Dale y Chip Weems
  "Pascal"
  Segunda Edicion.
  Editorial McGraw-Hill
  México 1991.
  pp 885.

- Peter Grogono.
  "Programación en Pascal".
  Editorial : Addison-Wesley Iberoamericana.
  U.S.A. 1986.
  pp. 371.

- William E. Weinman
  "El libro de CGI".
  Editorial Prentice-Hall
  México 1996.
  pp. 305.

Hemerografía

- PROGRAMACION ACTUAL,
  Editor : Eduardo Toribio,
  Revista Mensual,
  España,
  Año 1, número 5,
  Agosto de 1997.
  Página 12.

Manuales

- Departamento de Informática.
  "Manual Técnico de Cronos".

- Departamento de Informática
  "Manual de Usuario de Cronos"