SISTEMA B-20

SISTEMA B-20

Octubre de 1985
Unidad de Computo

D-58

SISTEMA B-20

MANEJO DEL SISTEMA

REALIZO' :

GABRIEL MAURICIO ALVAREZ MEDINA

INDICE

LA DIV SION DE ESTUDIOS DE POSGRADO DE LA FACULTAD DE INGENIERIA, POR MEI IO DE LA UNIDAC DE COMPUTO, HA REALIZADO SEMINARIOS SOBRE EL SIS EMA B20 DIRIGIDC S AL PERSONAL ACADEMICO, ADMINISTRATIVO Y A LOS ESTUDIANTES DE POSGRADO. DE ESTOS SEMINARIOS, SURGIO LA IDEA D CREAR UN MANUAL DE OPERACION Y MANEJO DE DICHO SISTEMA CON EL FIN DE CONTAR CON UN ELEMENTO BIBLIOGRAFICO IMPORTANTE PARA F TUROS SEMINARIOS O CURSOS BASADOS EN EL MISMO.

PRACTICA I

INTRODUCCION

En 1983 la compañía Burroughs introdujo la familia de minicomputadoras de escritorio B20. Esta familia está formada por la series B21 a B25. La diferencia entre cada serie está basada en elementos de hardware y software. Por ejemplo, las B21 no soportan manejo de display para realizar graficación, lo cual es soportado por las demás series.

Dentro de la familia B21 existen las versiones desde B21-1 a B21-5 y dentro de cada versión existen versiones más específicas como la B21-55T que cae dentro de la versión B21-5.

La Unidad de Cómputo de la División de Estudios de Posgrado cuenta con la familia B21.

La configuración con la que se cuenta está formada por 3 estaciones B21-1 y una estación maestra B21-55T.

Además se cuenta con una impresora paralela (B9252).

Las estaciones B21-1 están conectadas como estaciones asociadas a una maestra que es la estación B21-55T.

Las características de los elementos del sistema son las siguientes :

Estaciones B21-1 :

Cuentan con un procesador 8086 de Intel de 16 bits. 8 Mhz.

384Kbytes de memoria RAM.

ROM para pruebas de la estación

Puerto RS-422 de alta velocidad

Pantalla de despliegue de 28 x 80

Teclado móvil con 98 teclas

Estas estaciones sólo pueden trabajar como estaciones asociadas.

Estacion 821-55T

Procesador 8086 de 16 bits

Unidad de floppy de 5 1/4 de pulgada

Unidad de disco duro de 5 1/4 de pulgada con tecnología Winchester

Dos puertos RS-422 (307 Kbauds a 410Kbauds)

Dos puertos RS-232C (Canal A y B) de 110 bauds a 9600 bauds

512Kwords

Pantalla de despliegue 28 x 80

Teclado móvil

Puede funcionar como estación asociada a otra o como estación maestra o como estación independiente.

La conexión de estaciones asociadas y maestra forma una red local de tipo anillo.

Las estaciones de trabajo comparten la unidad de disco duro.

Una de las posibilidades de estas estaciones es que pueden trabajar como una red local o como un conjunto de remote job de la computadora central B7800 a través de una interfase RJE.

CARACTERISTICAS DEL TECLADO DEL SISTEMA B-20

El teclado del sistema es del tipo móvil. Esta formado por 98 teclas.

Todo el teclado esta traducido al español e inclusive cuenta con la letra "ñ", lo cual lo coloca como un modelo único en el mercado de la computación.

Esta dividido en diferentes áreas :

Alfanumérica

Numérica

De manejo de cursor

De función

De despliegue

De control

Dentro de las alfanuméricas las mas importantes son :

Mayúscula utilizada para fijar los caracteres localizados en la parte superior de las teclas.

Fijar mayúscula. fija sólo el alfabeto en mayúsculas.

Retroceso borra el caracter recién entrado

Return coloca en la primera columna del siguiente renglón.
Una tecla particular es la de CODIGO la cual es utilizada por el editor para insertar caracteres dado un código hexadecimal. Esto es particularmente útil para la creación de archivos de delegación , los cuales veremos más adelante.

Teclado de aplicación.
Está formado por 10 teclas (F1 a F10) que se utilizan cuando se hace uso de alguna aplicación como por ejemplo MULTIPLAN o DATA MANAGER.
En el editor son utilizadas para realizar procesos de borrado o repetición de algún comando.

Teclas de control.

Es un conjunto de 8 teclas :
CNCL utilizada para cancelar el proceso que se está realizando.
MOVER utilizada por el editor para mover información dentro del texto.
AYUDA cuando se oprime da información que nos auxilia a conocer comandos o funciones de la aplicación que estemos corriendo. Proporciona información general de comandos y opciones de las aplicaciones.
BORRE utilizada en cualquier momento para borrar el carácter

sobre el cual está posicionado el cursor.

ACCION funciona en conjunto con otra tecla. Genera códigos de operación que permiten, por ejemplo, detener una corrida, o la ejecución de algun comando.

SOBREESCRIBIR (tiene un led). utilizado en la edición, permite cambiar el contenido de la posición en la cual está el cursor sin tener que borrar primero.

FIN utilizada normalmente para dar por teminado un proceso o para salir de alguna aplicación.

COPIE utilizada al momento de editar. Permite copiar el contenido de texto de un lugar a otro sin borrarlo.

Teclas de despliegue.

Están formadas por 4 teclas :

PROX PAG utilizada para desplegar la siguiente página de texto. Se utiliza cuando el sistema lo pide o en el editor para avanzar a la siguiente página de texto.

PAG ANT utilizada exclusivamente en el editor. Muestra la página anterior de texto.

SUBIR utilizada en el editor para desplegar el texto en aumento de porcentaje.

BAJAR igual que subir pero bajando el porcentaje de texto desplegado.

Teclas de cursor.

Formadas por 6 teclas :

MARCA utilizada por el editor para marcar texto.

BORDE una vez marcado el texto, su función es indicar el borde de la marca

Cuatro flechas (arriba,abajo,derecha e izquierda). Utilizadas en la mayoría de las aplicaciones para moverse según el sentido que indican.

La tecla de marca puede utilizarse en conjunto con las teclas de dirección para generar varios tipos de marcado de texto. Esto se verá cuando discutamos el manejo del editor.

Teclas numéricas.

Estan formadas por un juego de 14 teclas :

Ls digitos , además del punto,el signo "-" y las teclas SIGA y PROX.

SIGA es de especial interés. Es utilizada en todas las aplicaciones del sistema y podría interpretarse como "ejecuta las órdenes que he dado". Le indica al procesador que el buffer de caracteres que se ha cargado está listo y puede tomarlo para decodificarlo y ejecutar los comandos.

PROX equivale a la tecla de RETURN pero es la más apropiada para cuando se introducen dígitos.

El software disponible en el sistema es el siguiente :

Compiladores PASCAL, FORTRAN, COBOL.

Intérprete BASIC

Ensamblador de 8086

Servicio ISAM (Método de acceso secuencial indexado)

Servicio DATA MANAGER

Servicio MULTIPLAN.

A continuación se proporcionan una serie de definiciones que se utilizarán constantemente durante los siguientes documentos :

VOLUMEN es un medio de almacenamiento masivo. Típicamente es un disco duro o un disco floppy.

DIRECTORIO nombre lógico bajo el cual podemos almacenar una serie de archivos.

ARCHIVO es un conjunto de datos, guardados bajo un directorio.

Antes de continuar, indicaré la manera de entrar en una sesión normal de trabajo.
Primeramente debe revisarse que la estación maestra de trabajo esté encendida. Esto es responsabilidad del o los administradores del sistema.
El usuario debe encender su estación de trabajo.
Observará una serie de letras, secuencias de puntos y asteriscos, que significan una serie de pruebas del sistema.
Terminadas estas secuencias aparecera un área en la cual estarán tres campos.
El campo de selección, el de código de seguridad y el de fecha.
El campo que está activo y que es en el cual el usuario debe

introducir información es el que se presenta con video inverso.

El campo de selección requiere que el usuario especifique el nombre de su directorio. Una vez dado éste,se da (return) y se posiciona en el campo de código de seguridad.

Si su directorio tiene código de seguridad, aquí es donde deberá epecificarse.

Se posiciona después en el campo de fecha.

Si no está dada la fecha (lo cual se observa en la parte superior derecha de la pantalla),el usuario deberá especificar una. Debe tenerse especial interés en los días miércoles y sábado que requieren acento para ser aceptados.

Una vez dada esta información, se oprime la tecla de SIGA y el sistema se posiciona directamente en el volúmen del sistema y el directorio especificado por el usuario.

SISTEMA B20

PRACTICA II COMANDOS

El sistema B20 cuenta con una serie de comandos, en español, que permiten al usuario manipular su información.
Los 60 comandos que forman el conjunto disponible están divididos en las siguientes categorías :

Volumen
Directorios
Archivos
Aplicaciones (llamadas a ISAM,MULTIPLAN,DATA MANAGER,WRITEONE)
Compiladores
Otros

La tecla AYUDA permite al usuario ver los comandos que tiene disponibles y si se oprime dos veces se observa una breve descripción de la función de cada comando.
La forma de llamar a cada comando es dar el nombre de éste en el campo de aplicación (se observa este campo con video inverso) y después oprimir la tecla de return. Seguido a esto,aparecerán los subcampos de cada comando y un área (en video inverso) sobre la cual el usuario deberá dar información que se solicite.
Para identificar un comando sólo es necesario dar los caracteres

necesarios que lo identifican. por ejemplo :

el comando EDITAR puede se llamado como
ED o EDI o EDIT o EDITA o EDITAR.

Sin embargo existen algunos comandos que requieren que el nombre del comando se dé completo. como por ejemplo el comando Dar Protección . Este comando además, necesita que se dé acento, lo cual se hace oprimiendo la tecla de acento y después la tecla que tiene la letra.

Existen dos tipos de subcampos que se observan al llamar un comando: los obligatorios y los opcionales.
Los primeros se identifican por no estar encerrados en paréntesis y el usuario debe dar información en éstos; en caso contrario el sistema indicará la falta de éstos con algún mensaje de error.
Los segundos son opcionales y se identifican porque están encerrados entre paréntesis cuadrados. Estos campos pueden dejarse sin llenar o ser llenados con información que el usuario desee y esté correcta.

Por ejemplo, el comando copia presenta la siguiente sintaxis :

Copiar
  Desde archivo
  Hacia archivo
    [Sobregrabar aprobado?]

[Verificar cada uno?]

Este comando presenta cuatro subcampos; dos obligatorios y dos opcionales.
Los subcampos Desde archivo y Hacia archivo deben ser llenados con información válida y los otros dos campos son opcionales.

Comandos de volumen.

Los comandos que actúan directamente sobre un volumen son los siguientes :

CDFlexible nos permite copiar toda la información de un volumen que debe ser un disco flexible hacia otro en forma íntegra. Se utiliza para generar respaldos de información.
Todos sus campos son opcionales.

Estado Volumen permite que el usuario observe el estado de un volumen.
Da información de fecha de creación, última modificación, número de páginas libres, número de encabezamientos de archivo libres, además de los directorios.

IVOLUMEN permite inicializar un volumen para ser usado como elemento de almacenamiento masivo. Es posible indicar los

sectores que no estén correctos para que de esta manera el sistema no grabe sobre regiones malas de disco.
Permite dar código de seguridad al volumen.
Requiere dos campos obligatorios : nombre de la unidad (f0 o d0 en nuestro caso) y nombre que llevará el volumen (por ej win,sys,datos).

IVRESPALDO permite generar copias de un volumen para mantenerlo como respaldo. A diferencia del CDFlexible, realiza optimización del espacio en disco.

Comandos de directorios

CREAR DIRECTORIO permite crear directorios para almacenar los archivos en una manera ordenada. Requiere como campo obligatorio el nombre del directorio. Tiene opción de dar niveles de protección al directorio. El máximo número de archivos que se puede tener en un directorio es de 45, aunque varía dependiendo del tamaño de éstos. Este valor es configurable mediante un campo opcional.

DAR PROTECCION DIRECTORIO permite dar protección a un directorio que ha sido creado. El campo obligatorio es el nombre del directorio.

ELIMINAR DIRECTORIO permite eliminar un directorio. Es necesario dar el nombre de un directorio. Un directorio no puede ser removido si aún tiene archivos que dependan de él, lo cual implica que si queremos eliminar el directorio, antes debemos eliminar los archivos que están bajo su nombre.

Comandos de archivo

ARCHIVOS permite ver la lista de archivos de cierto directorio. Es posible pedir detalles de los archivos (fecha de creación, tipo de protección, tamaño).

ANEXAR se utiliza para agregar archivos a un archivo. Se proporciona una lista de archivos origen y un archivo destino. Los archivos origen son agregados al archivo destino a partir de la marca de fin de archivo de éste y en el orden especificado en la lista.

EDITAR ARCHIVO permite editar un archivo. Es necesario un nombre de archivo.

COPIAR permite copiar un archivo hacia otro. Es necesario especificar el nombre del archivo fuente y el nombre del archivo destino. Presenta los campos de sobregrabar aprobado y de verificación.
Es posible utilizar "*" para indicar una familia completa de

archivos. Este comando se utiliza para copiar archivos que se quieran imprimir hacia el área de impresión llamada SPL.

CAMBIAR NOMBRE nos permite cambiar el nombre de un archivo. Es necesario especificar el nombre de un archivo y el nuevo nombre del archivo.

BORRAR permite borrar un archivo o una serie de archivos indicando el nombre o la lista de archivos. Es posible utilizar el "*" para trabajar sobre un conjunto de archivos. Permite trabajar en modo "pregunta" sobre el borrado de archivos en cuyo caso, antes de borrar un archivo, nos preguntará el sistema si queremos borrarlo o no.

DAR PROTECCION coloca un nivel de protección en los archivos o a una lista de archivos. Los niveles más útiles son

	desplegar	copiar	borrar	editar
15 sin protección	sí	sí	sí	sí
5 protegido de modificación	sí	sí	no	no
0 protegido de acceso	no	no	no	no

Es necesario dar la lista de archivos y el nivel de protección que tendrán.

DESPLEGAR permite desplegar en la pantalla un archivo sin tener

que editarlo.

Es necesario dar el nombre de un archivo o una lista de archivos. Es posible utilizar el "*" para desplegar un conjunto completo de archivos.

EJECUTAR ARCHIVO permite ejecutar un archivo ejecutable, resultado de ligar un archivo compilado con las rutinas del sistema. Es necesario un nombre de archivo ejecutable.

RESPALDO SELECTIVO permite respaldar selectivamente archivos de un volumen hacia otro.

RESTAURAR permite restaurar archivos a un volumen desde un archivo de respaldo.

BIBLIOTECARIO es un comando especial para manejo de archivos. Permite crear bibliotecas de archivos compilados para poder después referirse al nombre de la biblioteca y generar módulos objetos. Por ejemplo, para utilizar la utilería de ISAM es necesario realizar un ligado del programa en alto nivel con llamadas de ISAM al modulo ISAMMULTI.lib que contiene las expansiones de las rutinas que utilizan los archivos de aplicación.

Existen comandos de propósito especial que tienen que ver directamente con la edición de archivos, como lo son :

GRABAR permite generar un archivo de comandos. Este archivo se crea para después ejecutarlo con sólo llamar el comando DELEGAR.

DETENER GRABACION detiene la grabación de la secuencia de comandos que se inició con GRABAR y salva el archivo de comandos generado.

DELEGAR permite ejecutar archivos de comandos. Se crea el archivo de comandos a través del editor (utilizando códigos hexadecimales que son proporcionados mediante la secuencia CODIGO I y el número hexadecimal correspondiente al comando que se desea ejecutar), o se crea a través del comando GRABAR.

Comandos de compilación y link.

Para compilar un programa generado a partir del editor y con esto generar un archivo que pueda ligarse con las rutinas del sistema y de esa manera obtener un archivo ejecutable, se utilizan los siguientes comandos.

COBOL permite compilar un archivo fuente escrito en Cobol. Es necesario especificar el nombre del archivo fuente.
Los archivos ejecutables generados mediante Cobol deben ejecutarse con el comando CRun, al cual se le especifica el nombre del archivo que resultó del link.

FORTRAN permite compilar un programa fuente escrito en Fortran. Es necesario especificar el nombre del archivo fuente.

PASCAL compila un programa fuente escrito en Pascal. Es necesario el nombre del archivo fuente.

ENSAMBLAR ensambla un programa escrito en lenguaje ensamblador. En este caso es posible ensamblar una serie de archivos al mismo tiempo.

Para ligar los archivos objeto resultado de compilar o ligar se utiliza el comando ENLAZAR, el cual requiere de los campos de módulos objeto y nombre del archivo resultante.
Es posible enlazar archivos objeto resultantes de diversas compilaciones de archivos fuente escritos en diferentes lenguajes de programación.

Es sistema B20 incluye además de compiladores,un intérprete de BASIC.
Los programas realizados en BASIC se ejecutan llamando a éste comando. Es posible entrar sin dar un nombre de programa (se cargará desde modo comando en BASIC) o entrar dando un nombre de programa, el cual se cargará automáticamente al entrar a modo comando. Los detalles se explicarán más adelante.

Comandos de aplicaciones.

Esta categoría incluye comandos que llaman aplicaciones del sistema.
Una aplicación se entiende como software proporcionado por el fabricante para realizar aplicaciones específicas.
Se cuenta con un manejador de base de datos llamado DATA MANAGER, un mecanismo de acceso secuencial indexado llamado ISAM, y un sistema de hoja electrónica llamado MULTIPLAN.

Los comandos para manipular datos mediante DATA MANAGER son los siguientes:

DMCreate permite crear archivos para manipular mediante del manejador de datos.
DMPlist muestra el contenido de un archivo creado mediante el manejador de datos.
DMRun ejecuta una aplicación generada mediante el manejador.
DMUpdate permite actualizar la biblioteca de archivos generados mediante el manejador de datos.

Los comandos para manipular información mediante ISAM son los siguientes :

ISAM Cambiar nombre permite cambiar el nombre a un conjunto de

datos generados mediante el sistema ISAM. Esta información pudo crearse a través de un programa de aplicación o a través de comandos que manipulan y crean un archivo.

ISAM Copiar permite copiar información generada a través de ISAM hacia otros archivos.

ISAM Estado permite ver el estado de los archivos generados por ISAM.

ISAM Instalar permite instalar permanentemente el servicio ISAM para su utilización por múltiples usuarios.

ISAM Reorganizar permite reorganizar la información que se tiene en archivos. Es posible generar nuevos archivos utilizando la información de los archivos existentes.

Mantener Archivo permite combinar la información contenida en diferentes archivos generados mediante ISAM.

Otra aplicación del sistema es MULTIPLAN el cual es un conjunto de rutinas que permiten manipular una hoja de trabajo de 63 columnas (de un máximo de 32 subcolumnas) por 255 renglones.
La forma de llamar este sistema es invocando el comando MULTIPLAN.

Comandos varios

RUTA permite cambiarnos hacia otro volumen o directorio. Si sólo se especifica el volumen, se tomará por default el directorio en el que nos encontremos. Si especificamos el directorio, se tomará por default el volumen en el que nos encontramos. Si especificamos ambos, nos cambiaremos de volumen y directorio.

INICIAR tiene la misma función que RUTA.

COMANDO NUEVO permite crear un comando para la aplicación propia específica de un usuario.

CONFORMAR PANTALLA permite modificar los atributos del video. Es posible tener video inverso, obscurecer el cursor, modificar el tamaño de la ventana.

PONER HORA permite modificar la hora que tiene el sistema.

IMPRIMIR ACTIVIDADES permite observar un listado de las actividades realizadas durante una sesión.

CREAR ARCHIVO DE CONFIGURACION permite reconfigurar los puertos del sistema, así como los archivos que controlan los dispositivos periféricos.

INSTALAR ADMINISTRADOR DE COLAS es utilizado por el administrador del sistema para generar colas de impresión.

TERMINAR es el comando final de una sesión. Se utiliza para terminar de una forma normal la sesión de trabajo.

SISTEMA B20

PRACTICA III. USO DE LOS COMPILADORES

El sistema B20 cuenta con los siguientes compiladores :

PASCAL, FORTRAN, y COBOL. Además se cuenta con un ensamblador (microprocesador 8086) y con un intérprete de BASIC.
En este documento mostraremos la manera de utilizar los compiladores y el intérprete así como mencionaremos las diferentes directivas útiles para el usuario y para la propia protección del sistema.
También se mostrará la forma de ligar módulos objeto generados por diferentes compiladores (BINDERS).

La forma de llamar a los compiladores es mediante su nombre,así,para llamar al compilador de pascal, invocamos el comando PASCAL y proporcionamos el nombre de un programa fuente escrito en pascal.

Uso del compilador PASCAL

El PASCAL que maneja el sistema B20 está orientado para cumplir con las normas establecidas por la Organización Internacional de

Standars (ISO).

Entre sus facilidades permite ligar sus módulos objetos con módulos objetos generados mediante FORTRAN y ENSAMBLADOR. Además es posible realizar metallamadas a los servicios proporcionados con el sistema tales como ISAM.

Una vez editado un programa con sintaxis de Pascal, el mecanismo de compilación es el siguiente:

a) Invocar al compilador mediante el comando PASCAL
b) Indicar como información necesaria en el subcampo del comando el nombre del archivo fuente a compilar.
c) Indicar información, si se desea, en los subcampos opcionales del comando para la generación o no de ciertos tipos de listado.

Después de llamar al comando PASCAL observamos los siguientes campos

Nombre del archivo fuente
[Archivo objeto]
[Archivo de listado]
[Archivo de lista de objeto]

El primer campo es necesario e indica el programa a compilar (normalmente tiene la extensión ".pas").
En el campo de Archivo objeto el usuario puede indicar un nombre que desee para su archivo compilado. Si no especifica ninguno, el

sistema creará un archivo con el mismo nombre que el archivo fuente pero con la extensión ".obj".

El campo de Archivo de listado admite un nombre que quiera dar el usuario al archivo donde se indicará el resultado de la compilación. Si no es especificado por el usuario, el sistema generará uno con la extensión ".lst". Este archivo muestra el programa fuente con una serie de anotaciones resultado de la compilación. Si existieron errores se podrán observar en este achivo. Además, en este archivo se puede observar el listado en una secuencia numérica que es exactamente la secuencia que utiliza el compilador para indicar la línea en la que existió error.

El Archivo de listado objeto da una serie de indicaciones acerca del código objeto generado. Es útil para personas con conocimiento de bibliotecas que utiliza el sistema. Normalmente se recomienda que en este campo se especifique [NUL] para no perder tiempo en su creación y la compilacion sea más rápida.

En un programa escrito en Pascal es posible utilizar una serie de directivas que guían o habilitan una serie de procedimientos que resultan útiles para el usuario y para el sistema.

Una directiva o metacomando se especifica entre paréntesis y con el símbolo de "$" al inicio, por ejemplo

{$PAGE:12}

Las directivas pueden ir en cualquier línea del programa y siempre al inicio de una línea.
Para deshabilitarlas, se coloca cualquier símbolo diferente de un "tab" o un espacio antes del símbolo "$", por ejemplo

```
 {x$PAGE:12}
```

Los metacomandos pueden ser de cuatro tipos diferentes

a) De uso directo. Con invocar el nombre del metacomando se ejecuta la acción y no es necesario especificar ningún parámetro, por ejemplo

```
   $PUSH
```

b) De tipo entero, las cuales requiren proporcionar un parámetro de tipo entero , por ejemplo $PAGE:10.
c) De conmutación, en las cuales un valor mayor que cero enciende una opción y un valor menor o igual a 0 apaga la opción.
Por ejemplo $MATHCK:1
d) De tipo string, las cuales nos permiten manejar mensajes durante la compilación, por ejemplo $TITLE:'comentario'

Entre los metacomandos más útiles podemos encontrar los siguientes :

prendido

$SIZE minimiza el código generado. n

prendido

$BRAVE envía los mensajes de error y warning a la terminal s

$DEBUG+ enciende los mecanismos de chequeo y nos permite obtener información de la localizacion de errores al momento de ejecución. Se recomienda prender esta opción cuando se tiene un programa libre de errores de compilación debido a que su existencia incrementa la cantidad de código generado.

$ERRORS:(número) indica el máximo de errores que se permitirán n al momento de compilar.

$INDEXCK revisa los rangos de los arreglos.

$MATHCK+ revisa errores numéricos tales como integer overflow n

$INITCK revisa que las variables estén inicializadas antes de n utilizarse.

$RANGECK revisa la validez de los rangos tales como los índices s válidos en un CASE.

$STACKCK+ revisa el estado del stack. Básicamente pregunta por s condición de stackoverflow.

$INCLUDE:'(nombre de archivo) permite incluir código en pascal n especificando nombre de un archivo creado por el editor

$INCONST:(texto) permite actuar interactivamente al momento de n la compilación y asignarle un valor a una variable que se utilizará para controlar el flujo de la compilación

Existen metacomandos que nos permiten controlar el flujo de la

compilación, y tal vez la manera más fácil de entender su funcionamiento sea mediante un pequeño ejemplo :

Supongamos que tenemos la siguiente estructura :

```
{$IF constante}
   {$THEN} (texto1)
   {$ELSE} (texto2)
{$END}
```

En este ejemplo, cuando "constante" tenga un valor positivo mayor que cero, se procesará (texto1), en caso contrario se procesará (texto2).
La manera de dar valor a "constante" puede ser de dos formas: mediante asignación directa o interactivamente con el metacomando $INCONST.

El compilador Pascal puede generar un máximo de 64Kbytes de código. Si se desea compilar programas que generan un código mayor es posible utilizar una serie de opciones que provee el lenguaje, por ejemplo podemos escribir módulos separados en Pascal e indicar que serán módulos que podrán accesarse por otros módulos compilados. Esto se puede lograr indicando la opcion PUBLIC en una función, por ejemplo

```
program público(input,output);
        type str = lstring(28);
```

```
        procedure exter(var st:str) [PUBLIC];
        begin
         st:='cadena de caracteres'
        end;
        begin
  (* ejemplo de una rutina pública *)
 end.
```

Este programa puede ser compilado y la rutina "exter" puede ser llamada por otro módulo, como por ejemplo

```
 program llamada(input,output);
        type s = lstring(28);
        var  st : s;
        procedure exter(var cadena:s); EXTERN;
 begin
  exter(st);
 end.
```

Una vez compilados ambos módulos, para ejecutarlos debemos ligarlos y generar un archivo ejecutable, lo cual lo logramos colocando el nombre de cada módulo resultante de la compilación (llamada.obj y público.obj) en el subcampo del ligador donde dice "nombre de archivos objeto".

Esta técnica nos permite ser más eficientes al momento de crear nuestros archivos porque podemos probar rutinas en forma separada sin tener que compilar todo un programa.

Una rutina que tiene la especificación PUBLIC puede ser ligada con otros módulos generados por otros compiladores (FORTRAN y ENSAMBLADOR) simplemente haciendo la llamada a estas rutinas con la especificación EXTERN.

Compilador Fortran

El sistema B20 soporta FORTRAN 77 y algunas versiones más antiguas como el FORTRAN 66 y FORTRAN 4. Mediante éste lenguaje es posible hacer uso de paquetería del sistema tal como ISAM a través de metallamadas. Además se permite ligar módulos objetos generados por este compilador con módulos objetos generados con PASCAL y ENSAMBLADOR.
La manera de llamar el compilador Fortran es a través del comando FORTRAN.
Una vez invocado, se presentan al usuario los siguientes subcampos :

Archivo fuente
[Archivo objeto]
[Archivo de listado]
[Archivo de listado objeto]

El primer subcampo es obligatorio y especifica el archivo fuente escrito en FORTRAN que será compilado.
El Archivo objeto es el nombre del archivo resultado de la

compilación. Si no se especifica ningún nombre,se generará un archivo con el mismo nombre que el fuente pero con la extensión ".obj".
En el Archivo listado se escribe un listado de la compilación. Si no se especifica ningún nombre, se genera un archivo con el mismo nombre que el fuente pero con la extensión ".lst".
En el Archivo de listado objeto se escribe un listado del código objeto generado.

Después de compilado el programa es necesario ligarlo para generar código ejecutable. Esto se hace a través del comando LIGAR que trataremos más adelante.

Existen una serie de metacomandos que permiten manipular el control de la compilación así como permitir el chequeo de errores, así como otras tareas generales tales como indicar títulos y subtítulos a las hojas de compilación; indicar el máximo de errores permitidos en la compilación; enviar mensajes al momento de compilar.
La manera de indicar un metacomando en FORTRAN es colocando el carácter "$" en la primera columna seguido del nombre del metacomando.

Las directivas hacia el compilador son las siguientes :

$DEBUG indica que todas las operaciones siguientes a esta

llamada serán revisadas por si se encuentran errores como overflow y división por cero. También revisa que los índices de los arreglos estén dentro de rangos válidos. Esta directiva debe ser indicada por el usuario.
$NODEBUG apaga $DEBUG. Está encendida normalmente por el sistema.

$D066 permite escribir programas en código FORTRAN con DOs de semántica utilizada en FORTRAN 66.

$DYNAMIC permite utilizar recursión en el programa.

$INCLUDE indica al compilador que compile el segmento de código en FORTRAN especificado como si éste estuviese insertado en el punto donde está el INCLUDE. Su sintaxis es la siguiente :
$INCLUDE:'nomb.file'

$LINESIZE permite alterar el ancho de las líneas que resultan en un listado de compilación. Su sintaxis es : $LINESIZE:valor donde valor es un número entero. Por default "valor es de 132.

$PAGE permite imprimir el archivo de listado de compilación con una nueva página. Cada vez que el compilador encuentre esta directiva, generará una nueva página de listado de compilación.

$PAGESIZE indica que las subsecuentes páginas serán formateadas según un valor.
Su sintaxis es : $PAGESIZE: valor donde valor es un entero

positivo.

$STORAGE permite indicar cuántos bytes deseamos que ocupe una variable de tipo INTEGER o LOGICAL. Su sintaxis es : $STORAGE:n donde "n" especifica el número de bytes. Si existen variables definidas dentro del programa con una longitud definida explícitamente (INTEGER*n) entonces esta directiva respeta su longitud.

$TITLE permite poner títulos a las hojas del archivo de compilación. Su sintaxis es : $TITLE:'título'.

Es posible escribir rutinas de Fortran en forma separada y ligarlas en una forma semejante a la forma de hacerlo en pascal. También es posible hacer llamadas desde FORTRAN a rutinas escritas en otro lenguaje, como por ejemplo

```
program prueba
subrutine exter(e)
e:=10
return
end
```

El programa en Pascal para llamar esta rutina sería

```
program llamada(input,output);
var parámetro:real;
```

```
begin
  exter(parametro); EXTERN:
  write(parametro)
end.
```

Compilador COBOL

B20 permite escribir código fuente en una versión de Cobol llamada LII/COBOL, el cual cumple con las normas establecidas por ANSI.

La forma de llamar el compilador Cobol para compilar un programa escrito con el editor es a través del comando COBOL.

El único subcampo de este comando que requiere información es el de "Archivo fuente" en el cual se especifica el nombre del archivo fuente en cobol que tiene extensión ".cob"

Este comando cuenta además con una serie de opciones tales que permiten encender banderas para controlar el proceso de compilación. Normalmente se requiere que el usuario sea un experto para colocar información en estos campos, lo cual normalmente no sucede.

Existen opciones útiles que no requieren que el usuario sea un experto en el sistema, y éstas son normalmente las que todo usuario de Cobol utilizará y son las que presento en el siguiente

párrafo.

Las opciones generales que se presentan son las siguientes : Archivo de listado (archivo con extension ".lst"); Listar sólo errores (no se lista el archivo compilado); Supresión de código intermedio (no se guarda el archivo de código intermedio que se genera en la compilación); Suprimir eco de errores (no se envía a la pantalla los errores encontrados en la compilación).

Después de compilado el programa escrito en cobol, la manera de ejecutarlo es mediante el comando CRun en el cual es necesario especificar un nombre de archivo intermedio. Se presentan las opciones para encender un analizador ya sea de Cobol o de ANSI.

Un inconveniente de utilizar Cobol en el sistema B20 es la imposibilidad de realizar binders con programas escritos en otros lenguajes, lo cual representa no contar con una herramienta poderosa de software.

BASIC

El intérprete que proporciona el sistema es llamado mediante el comando BASIC. Este comando muestra los siguientes subcampos

[Programa inicial]

[Máximo número de archivos]

[Máximo tamaño de registro RANDOM]

Como se puede observar, todos los campos son opcionales, esto es porque nosotros podemos escribir un programa en Basic directamente a través de la llamada del comando o podemos, estando en el modo comando, llamar un archivo generado mediante el editor.
El primer campo se refiere al nombre del programa que deseamos cargar para trabajar con él. Debe tener extensión ".bas". y cuando oprimamos la tecla SIGA este archivo se cargará y se desplegará en la pantalla.

El campo "Máximo número de archivos" se refiere al número de archivos que podemos tener abiertos al mismo tiempo. El máximo es de 15 y el de default es 6.
Este parámetro puede afectar la cantidad de memoria que se asigna a un programa en Basic. Es bueno indicar el número exacto de achivos que se utilizarán para, de esta forma, asignar el máximo espacio para el programa de aplicación.
El espacio de memoria asignado a una aplicación en Basic es de 64Kbytes.
El campo "Máximo tamaño de registro RANDOM" especifica el tamaño de los registros de los archivos que se accesarán en forma random.
Este campo puede afectar la cantidad de memoria disponible para una aplicación en Basic y estará en función del parámetro especificado en el subcampo del número de archivos.

Una vez entrado al intérprete, estaremos en modo comando.
Si no especificamos ningún nombre en el campo de "programa inicial" podemos cargar alguno mediante el comando LOAD "nombre de archivo.bas".
Es posible escribir un programa directamente en modo comando y depués salvarlo a través del comando SAVE "nombre del archivo", A donde la extensión A es útil para salvarlo como un archivo visible para el usuario (se graba en forma ASCII).
Los comandos de BASIC son estándar en prácticamente todas las máquinas y B20 no es una particularidad, así, por ejemplo, para ejecutar un programa utilizamos el comando RUN, para dar secuencia utilizamos el comando AUTO, para editar damos el comando EDIT (número de línea). Cuando se entra en secuencia mediante AUTO, la manera de salirse de ésta es oprimiendo la tecla CNCL.
Algunos comandos útiles proporcionados por el usuario durante la operación en Basic son los siguientes :

ACCION-CANCEL detiene la ejecución de un programa sin salirse de Basic. Para activarlo nuevamente se proporciona el comando CONT.
ACCION-O detiene la salida de resultados. Oprimiéndolo nuevamente se restablece la salida de resultados.
CANCEL permite salir de secuencia cuando hemos entrado a esta mediante el comando AUTO.
FIN permite salir de Basic.

Ensamblador

El sistema B20 está basado en un procesador de 16 bits, el 8086 de INTEL con tecnología HMOS. Es posible escribir programas en lenguaje ensamblador para después ensamblarlas. La ventaja de escribir programas en ensamblador es que podemos generar procesos más rápidos debido a la propia optimización que realiza el programador. El código que se genera es mucho menor que el de un programa escrito en un lenguaje de alto nivel y compilado. Los usuarios que utilizan el ensamblador directamente, son personas más especializadas y con conocimiento del sistema y el procesador.

La manera de llamar el ensamblador (llamado ASM-86) es a través del comando ENSAMBLAR.

El campo que es obligatorio en este comando es el nombre del archivo fuente escrito en lenguaje ensamblador.

El ensamblador genera, como todos los compiladores, un archivo resultado de la compilación ".obj"; un archivo con los resultados de la compilación ".lst" y además es posible generar listados con el resultado de ambos pases del ensamblador (especificando "si" en el campo "Listado en primer pase"), lo cual es útil cuando se manejan macros, las cuales pueden introducir ciertos errores de programación que impiden llegar a la segunda pasada

del ensamblador.

LINKER

Una vez compilado un programa es necesario ligarlo con rutinas del sistema o con rutinas compiladas en forma separada para de esta forma generar un archivo ejecutable. Esto se logra a través del LINKER del sistema.
La manera de invocar al LINKER es mediante el comando ENLAZAR.

Es necesario especificar dos campos principales para la ejecución del comando : Módulos Objeto y Archivo de ejecución.

"Módulos objeto" requiere que se especifique nombres de archivos resultantes de una compilación, (archivos con extensión ".obj"). Si se tiene más de un archivo pueden separarse por espacios en blanco. Si se cuenta con archivos que se encuentran en una biblioteca de archivos (para generar un biblioteca de archivos se utiliza el comando BIBLIOTECARIO) sólo es necesario especificar el nombre de la biblioteca y los archivos que se utilizarán, por ejemplo :

Enlazar

Módulos objeto biológicos(bacterias celulares virus) metales.obj

en este caso se especifica que ligaremos los módulos objeto bacterias.obj celulares.obj y virus.obj que se encuentran en la biblioteca biológicos y además los ligaremos con el archivo metales.obj que no se encuentra en la biblioteca.

En el campo "Archivo de ejecución" es necesario especificar el nombre de un archivo que será resultado de realizar el link entre los diferentes módulos objeto. Este nombre será el que especificaremos en el comando EJECUTAR cuando ejecutemos nuestro programa.

El campo "Archivo de listado" es opcional y si se especifica un nombre se generará un archivo con el resultado de condiciones generadas durante el enlace. Se recomienda especificar [Nul] para no crear este archivo y lograr un enlace más rápido.

"Públicos" nos permite incluir en el archivo de listado la tabla de símbolos de generada por el linker. En particular, si se desea esta opción es necesario contestar en inglés "yes", lo cual constituye una incongruencia con la política del sistema.

"Producir sistema" nos permite generar una versión especial del sistema operativo de B20 (BTOS), lo cual para algunas aplicaciones muy especializadas es útil. No se recomienda para usuarios novatos.

"Bibliotecas" permite indicar archivos con extensión ".lib" en los cuales se encuentran archivos de tipo ".obj" que serán utilizados por las aplicaciones que estamos ligando. Por ejemplo, para la utilización del sistema ISAM es necesario especificar en este campo el nombre "ISAMMULTI.LIB". El linker siempre busca los módulos objeto que necesita en el archivo [SYS](SYS)Sys.lib y si nosotros especificamos una biblioteca explícita, lograremos eliminar la etapa de búsqueda y por lo tanto será más rápida la etapa de ligar.

"Asignación de DS". Si especificamos "yes" en este campo lograremos que la tarea del linkers sea más rápida porque no se aceptarán las referencias al DGroup.

"[Archivo de símbolos]" permite, dado un nombre de archivo, obtener una tabla de símbolos del archivo de ejecución. Se recomienda responder [NUL] en este campo para lograr un enlace más rápido.

SISTEMA B20

PRACTICA IV : MANEJO DE EDITOR

El sistema B20 cuenta para la edición de archivos con un editor de caracteres. La ventaja de este tipo de editor con respecto a los de línea es la facilidad que presentan al usuario en la manipulación de información. El usuario se olvida de la existencia de secuencias de líneas.

El editor permite observar una ventana del texto del usuario y manipular todo el texto sin tener que tenerlo presente.

El editor permite al usuario generar sus archivos escritos en un lenguaje de programación o editar archivos de datos que utilizará en aplicaciones siguientes, además de editar archivos de texto que serán procesados por el procesador de palabras.

La manera de llamar al editor del sistema es mediante el comando EDITAR.

Los campos que presenta este comando son NOMBRE DE ARCHIVO y NOMBRE DE USUARIO.

El primer campo es necesario y debe ser especificado por el usuario. En este se da el nombre del archivo que deseamos editar. Puede ser el nombre de un archivo ya existente o un nuevo nombre de un archivo que se creará.

Al momento de entrar a modo edición, la pantalla se divide en dos secciones :

El STATUS FRAME y el TEXT FRAME.

En el primero se despliega información del editor y del archivo que estamos editando (versión del editor, posición en porcentaje del archivo, condiciones de error).

En el Text Frame se despliega el texto, y la pantalla sirve como una ventana del texto total. En éste se pueden observar dos elementos que utiliza el editor : un pequeño cuadrado que indica el fin de archivo y un pequeño segmento que parpadea, llamado "cursor".

Es importante conocer estos dos elementos porque diversas funciones del editor se llevan a cabo dependiendo de su posición.

Además de lo anterior definiremos una SELECCION como una secuencia continua de caracteres dentro del archivo. Una selección se realiza a través de las marcas, de las cuales platicaré más adelante. Sólo mencionare que una selección se distingue porque al marcarla se presenta al usuario en video inverso.

Para el manejo adecuado del editor contamos con una serie de elementos en el teclado que permiten manipular el texto.

Las teclas que utiliza el editor son : las alfanuméricas (incluyendo las teclas de función f1,f10 y la tecla de código), las de cursor, las de despliegue, y las de control. El manejo de algunas teclas ya fue practicado en la sesión introductoria del sistema, así que enfocaré los demás elementos del teclado y su función, además de platicar sobre los comandos del editor.

Antes de continuar mencionaremos que una página de texto es la cantidad de texto que se puede observar en la pantalla a un mismo tiempo. La pantalla permite observar 26 renglones por 80 columnas y esto especifica el tamaño de una página.

Elementos importantes del teclado.

AYUDA oprimiendo esta tecla el sistema despliega los comandos disponibles en el editor. Para regresar a modo edición de texto es necesario oprimir la tecla CNCL.

CNCL permite cancelar la ultima operación indicada por el usuario. Esta opción funciona sobre los modos de inserción y los comandos.

PROX PAG. permite observar la siguiente página de texto.

PAG ANT permite observar la página anterior de texto.

SUBIR En la sección de status se observa una leyenda que dice "Posición del archivo" y un porcentaje. SUBIR permite aumentar el porcentaje de texto del archivo que estamos editando y el movimiento se observará de una manera suave.

BAJAR idéntico que SUBIR pero decrementando el porcentaje.

CODIGO utilizada en conjunto con otra tecla permite llamar a los diferentes comandos del editor.

Si mantenemos oprimida esta tecla y la " --> " el cursor se colocara un lugar después del último caracter que tenga el renglón actual.

Si se utiliza con la tecla " <-- " el cursor se posicionara en la primera columna del renglón actual.

SOBREESCRIBIR cuando se oprime esta tecla se enciende el led que está incluido en ella. Permite modificar los caracteres sin tener que borrarlos antes. Cambia el carácter donde está posicionado el cursor por el carácter que oprimamos. Para dejar este modo, oprimimos nuevamente la tecla y el led se apagará.

COPIE permite copiar una selección al lugar donde se encuentra el cursor.

El texto que se copia permanece en su posición original y obtenemos una copia de él en la posición indicada por el cursor.

MOVER permite mover una porción de texto hacia el lugar especificado por el cursor. A diferencia de COPIE, el texto ya no permanecerá en su posición original.

FIN cuando se le oprime, el sistema envía el mensaje de salvar o no el archivo. Si se especifica SI, se salvará el archivo editado y si se especifica NO, no se salvará. Si se especifica afirmativamente y ya existía el archivo, se creará una versión nueva del archivo y la anterior se renombrara con la extensión "-Old".

BORRE es utilizada para borrar el carácter sobre el cual está el cursor.

RETROCESO es la tecla superior derecha con una flecha apuntando hacia la izquierda. Esta permite borrar el carácter anterior a la posición del cursor. Si la tecla de SOBREESCRIBIR está seleccionada, RETROCESO funcionará como una tecla de avance hacia

atrás y no borrará ningún carácter.

RETORNO es la tecla con una flecha con punta blanca apuntando hacia la izquierda. Permite posicionarnos en la primera columna del siguiente renglón.

F1 utilizada para repetir el último comando dado por el usuario. Un ejemplo útil se encuentra en la búsqueda de una cuerda de caracteres. Se llama al comando de búsqueda y se ejecuta. Después, sólo es necesario oprimir F1 para localizar la siguiente ocurrencia de la cadena que estamos buscando.

F10 una vez realizada una selección a través de los comandos de marca, esta tecla nos permite borrar el texto que esté marcado. Debe tenerse cuidado con su manejo porque podríamos perder todo el archivo si todo está marcado.

FLECHAS se utilizan para mover el cursor a través de la ventana en la dirección a la que apuntan.

MARCA se utiliza para realizar una selección de texto. El texto marcado se coloca en video inverso.

BORDE permite indicar, una vez realizada una selección, el límite de la selección.

TAB si se oprime cuando el cursor está en la primera columna de un renglón, genera 9 espacios. En cualquier otra columna, al oprimirlo generará 7 espacios. Es útil para realizar movimiento rápido a través de un renglón o para editar en ciertos programas. Por ejemplo, FORTRAN requiere iniciar las instrucciones a partir de la columna 7, lo cual se realiza rapidamente a través de esta tecla.

MAYUS permite seleccionar los caracteres localizados en la parte

superior de cada tecla. Si se oprime ésta y una tecla con letra, se generará su carácter en mayúscula.

Marcas

Diversos comandos del editor funcionan sobre selecciones. Una selección de texto se realiza a través de diversos mecanismos:
a) Oprimiendo la tecla MARCA, realizamos la selección del texto localizado desde la posición del cursor hasta el fin de la línea.
b) Oprimiendo marca junto con la tecla " --) " realizamos una selección de todo el renglón en donde esté posicionado el cursor.
c) Oprimiendo marca al mismo tiempo que la flecha que apunta hacia arriba, generamos una marca de la palabra donde está posicionado el cursor.
d) Oprimiendo la tecla CODIGO y M al mismo tiempo, marcamos todo el texto.

Como podemos observar en lo escrito hasta este momento, he mezclado llamadas de comandos a través de teclas y llamadas a través de la acción de CODIGO y otra tecla.

Comandos de edición.

La gran mayoría de estos comandos se invocan mediante la acción conjunta de la tecla de CODIGO y alguna otra.

Este tipo de comandos requieren que el usuario en ciertos casos indique algunos parámetros. Por ejemplo, para buscar una cadena de caracteres, es necesario, después de llamar al comando búsqueda a través de CODIGO B, especificar la cadena que deseamos buscar. El campo destinado para esta información está limitado por los símbolos "<>". El cursor se posicionará sobre el ">" y todo lo que escribamos quedará encerrado en paréntesis. Después se dará SIGA y el sistema buscará la cadena especificada.

CODIGO M marca todo el texto. Coloca el texto en video inverso.
CODIGO B permite realizar la búsqueda de una cuerda de caracteres.
CODIGO I permite realizar la inserción de caracteres hexadecimales.
Este comando es particularmente útil para la edición de archivos de comandos. Si deseamos utilizar el comando de delegación, debemos editar programas utilizando este comando. Entre los códigos hexadecimales más útiles tenemos

0A return
1B SIGA
07 CNCL
04 FIN

CODIGO S permite realizar saltos a secciones del archivo. El porcentaje relativo de donde nos localizamos se puede observar en la sección de status (STATUS FRAME). Si indicamos un salto al

0%, nos posicionaremos al principio del archivo. Si especificamos 50%, a la mitad, y si especificamos 100% nos posicionaremos al final del archivo.

CODIGO L nos permite leer un archivo almacenado en disco e insertarlo en el archivo que estamos editando. La inserción se realiza desde la posición actual del cursor. Es posible indicar el número de caracteres que deseamos insertar y la posición (en porciento) de donde inicia la sección de código que deseamos insertar.

CODIGO R permite reemplazar una cadena de caracteres por otra cadena nueva.

Para utilizar este comando debemos seleccionar (marcando) el área de texto sobre el cual deseamos realizar el reemplazo.

CODIGO V permite hacer visibles los caracteres que normalmente no son visibles tales como los returns, los blancos, los códigos hexadecimales insertados. Repitiendo la secuencia de CODIGO V hacemos invisibles los códigos que se hicieron visibles.

CODIGO N nos posiciona al principio de la selección.

CODIGO Z funciona desde las series 822 hacia arriba. Permite cambiar de una pantalla de 80 columnas a 132 columnas.

CODIGO G permite grabar el archivo editado en disco duro sin tener que salir de modo edición.

La manera de salir de editor es a través de la tecla FIN.

Una vez oprimida ésta, se pedirá indicar si se desea salvar el archivo. Esto se realiza indicando SI en el campo que se muestra al usuario.

CURSO B20

PRACTICA V : MECANISMOS DE IMPRESION

Cada lenguaje que soporta el sistema B20 presenta características propias en sus mecanismos de impresión.
Se mostrará la manera de generar archivos de impresión en los diferentes lenguajes que soporta B20.

Impresión en BASIC

La impresión de archivos en este lenguaje se realiza a través de definiciones ERC%.
Mediante ésta, podemos enviar a impresora los resultados del programa de aplicación, utilizando para ello la cola de impresión a la cual esté asignado el usuario. También se utiliza para enviar un listado del programa a impresora.
La cola de impresión que se utilizará más comúnmente será SPL.
Existen tres tipos de definición de ERC%.

a) ERC%=DEFLPRINT("[SPL]")
b) ERC%=DEFLPRINT("[SPLB]")
c) ERC%=DEFLPRINT("[NUL]")

La primera opción se utiliza para indicar que las instrucciones posteriores se colocarán en el spooler para una impresora paralela.

La segunda opción es idéntica a la primera, pero para impresoras seriales.
La última se utiliza para indicar al sistema que la información almacenada en el spooler puede finalmente enviarse a la impresora.

Veamos algunos ejemplos

```
10 ERC% = DEFLPRINT("[SPL]")
20 FOR I = 1 TO 50
30 LPRINT CHR$(I),
40 NEXT I
50 ERC% = DEFLPRINT("[NUL]")
```

Este programa envia a impresión los primeros 50 caracteres ASCII.
La instrucción LPRINT es necesaria para indicar que los resultados se manden a impresora.
Durante el momento del loop, no se verá ningún resultado en la impresora, lo cual sucederá cuando se ejecuta la línea 50.

Para enviar un listado a impresora utilizaríamos lo siguiente :

```
ERC% = DEFLPRINT("[SPL]")
LLIST
ERC% = DEFLPRINT("[NUL]")
```

Impresión en PASCAL

Las intrucciones que emplea este lenguaje para el despliegue de información son WRITE o WRITELN.

Su sintaxis es la siguiente : write((nom.arch),(variables)).

El campo de nombre de archivo es el que nos permite definir la salida de nuestra información. Si no especificamos ningún nombre, la salida será por video. El archivo de salida puede ser modificado a través del comando ASSIGN.

Podemos indicar un archivo de salida en disco o directamente a la impresora. Para definir un archivo de salida, utilizamos el comando de ASSIGN.

Veamos un ejemplo :

```
PROGRAM IMPRESS(INPUT,OUTPUT)
VAR CARACT : CHAR;
    IMP    : TEXT;
    I      : INTEGER;
BEGIN
   ASSIGN(IMP,'[SPL]');   { asignación del archivo de salida }
   REWRITE(IMP);
   FOR I:=10 TO 29 DO WRITE(IMP,CHR(I))
END.
```

El programa anterior imprimiría los veinte caracteres ASCII, iniciando en el 10 y terminando en el 29, en la impresora parallel.

Debe observarse que el archivo de impresión se denomina IMP y esta asignado a [SPL]. Además, es un archivo de tipo TEXT y no está declarado en el encabezado del programa. Otro detalle es que, después de hacer el ASSIGN, debe realizarse un REWRITE del archivo, con el objeto de abrirlo para impresión.

Existe la posibilidad de alternar la salida de impresión. Supongamos que en cada corrida deseamos cambiar la salida de nuestra información, esto es, en algunos casos deseo que la impresión sea directamente a impresora, en otras, que sea a video, y en otras deseamos generar un archivo de resultados en disco. Una forma sería editar en cada ocasión nuestro programa y cambiar el nombre del archivo de despliegue, lo cual sería muy ineficaz. Otra forma, más racional, sería haciendo una asignación selectiva, desde software, del archivo de despliegue.
Veamos el siguiente ejemplo :

```
PROGRAM IMPRESS(INPUT,OUTPUT);
{ Selección por software del canal de impresión }
VAR SEL : CHAR;
    IMP : TEXT;
    I   : INTEGER;
BEGIN
 WRITE('CANAL DE SALIDA D/I/V )');
 REPEAT READ(SEL)
 UNTIL SEL IN ['D','I','V'];
 CASE SEL OF
```

```
  'D': ASSIGN(IMP,'[RESULTADOS]');
  'I': ASSIGN(IMP,'[SPL]');
  'V': ASSIGN(IMP,'[VID]')
 END;
 REWRITE(IMP);
 FOR I:=10 TO 29 DO WRITE(IMP,CHR(I))
END.
```

En este ejemplo, mostramos la manera más indicada para lograr salida de resultados por canales diversos sin tener que modificar (editando), el nombre del canal de salida.
Este programa podría utilizarse como una subrutina en todas nuestras aplicaciones.
Los detalles que deben observarse son los siguientes :

a) Tenemos la posibilidad de selección de canal de salida múltiple.
b) IMP puede ser asignado a cualesquiera de los nombres de archivo RESULTADOS,SPL,VID. El primer nombre se refiere a un archivo de salida en disco cuyo nombre será RESULTADOS, el segundo es el nombre del spooler, con el cual se tendrá salida inmediata a impresora, y el tercero ([VID]), permite la salida por video.
c) La selección del canal de salida se realiza a través de la variable SEL, a la cual asigna el usuario un valor mediante una lectura.

Una recomendación sería siempre enviar primero los resultados a disco, para de esa forma analizar su formato y decidir si es el que se busca. Si lo es, entonces se manda imprimir utilizando el comando COPIA, copiando a [SPL] nuestro archivo en disco; en caso contrario podemos modificar nuetros formatos de salida con la ventaja de que no desperdiciamos papel.

Impresión en FORTRAN

Es posible realizar operaciones semejantes a las realizadas en PASCAL para seleccionar el canal de salida.
La instrucción más completa que proporciona el lenguaje para el despliegue de los resultados es WRITE.
Su sintaxis es WRITE(⟨unid⟩, ⟨formato⟩) ⟨ variables ⟩, donde ⟨nom.arch⟩ especifica la unidad de salida de nuestros datos. Si en este campo especificamos *, nuestra entrada será a través del teclado y la salida será por video.
La manera de especificar un archivo de salida en disco es a través de la instrucción OPEN.
Su sintaxis es la siguiente : OPEN(⟨ent⟩,FILE=⟨nom.arch⟩,⟨opc⟩) donde ⟨ent⟩ es un número entero que se utilizará en la instrucción WRITE como ⟨unid⟩; ⟨nom.arch⟩ es el nombre lógico asignado a la unidad ⟨ent⟩ y ⟨opc⟩ son características propias de cada archivo.

Sea el siguiente ejemplo :

```
C
C Ejemplo de mecanismos de salida en Fortran
C
      PROGRAM COLSWP
      CHARACTER*64 FNAME,MSG1
      DATA MSGI/'REALIZADO'/
C
C Salida por video utilizando *
C
      WRITE(*,900)
900   FORMAT('DAME EL NOMBRE DE UN ARCHIVO DE SALIDA')
C
C Lectura del nombre de un archivo de salida dado por el usuario
C
      READ(*,910) FNAME
910   FORMAT(A)
C
C Abriendo el archivo especificado por el usuario
C
      OPEN(3,FILE=FNAME)
C
C Abriendo un archivo de salida para los resultados
C La unidad que asignaremos será la 4.
C La indicación de STATUS=NEW se utiliza para especificar un
C nuevo archivo de escritura. Si el archivo de salida existiese
C tendríamos que especificar STATUS=OLD.
```

```
C
      OPEN(4,FILE='OUT.TXT',STATUS='NEW')
C
C Lectura y escritura de los archivos especificados hasta que se
C localice el fin del archivo de lectura
C
100   READ(3,920,END=200)I,J,K
      WRITE(4,920)J,I,K
920   FORMAT(3I7)
      GOTO 100
200   WRITE(*,910)MSG1
      END
```

Este libro se terminó de imprimir en IMPROSA, siendo Jefe de la DEPFI, el Dr. Gabriel Echávez Aldape, y de la Sección Editorial, la Lic. Eugenia M. de Lizalde, a los 25 días del mes de octubre del año de 1985.

Tiraje: 500 ejs.