ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERIA EN ELECTRICIDAD

Tópico de Redes de Area Local

Informe presentado
Previo a la Obtención del Título de:

INGENIERO EN ELECTRICIDAD
Especialización ELECTRONICA

Perteneciente a:

Freddy Vizuete M. - Zenen Delgado M.
Fernando Mórtola B. - Marco Villalba D.

Tema:

Análisis, Programación e Implementación de una Red de Area Local "SERIAL"

Profesor: JAIME PUENTE PEREZ

1993 - 1994

Guayaquil - Ecuador

TABLA DE CONTENIDO

CAPITULO IV

CAPITULO V

CAPITULO VI

INTRODUCCION:

Nuestro conocimiento se compone de un sin número de experiencias, las cuales han llegado hasta nosotros por diferentes vías. Algunas de estas experiencias nos han sido pasadas por nuestros mayores, otras han sido fruto de la simple observación. Pero ninguna otra experiencia esta tan arraigada en nuestro conocimiento como el de las propias experiencias vividas.

En efecto, no hay mejor manera de entender un problema complejo, que el enfrentarse cara a cara con él, y sortear los obstáculos que aparecen a nuestro paso, sacando al final de la lucha conclusiones, fruto de nuestras propias experiencias.

Esta es justamente la filosofía con la que esta desarrollado este proyecto de **comunicaciones entre PC's**, llamado BP-LAN.

Este trabajo netamente didáctico nos lleva de la mano, mostrándonos los problemas que están involucrados en las comunicaciones dentro del entorno de una Red de Área Local (LAN), a la vez que nos muestra las soluciones y las razones que condujeron a tomarlas. De esta manera, muchos conceptos aprendidos en clase tienen entonces una base mas firme quedando las ideas sólidas y completas.

En este caso en particular, se trata de una red serial, es decir que para su implementación no hace falta ninguna tarjeta de red especial. Basta con que los computadores involucrados en la misma tengan uno o más puertos seriales, los cuales ya son considerados equipo estándar en los computadores. El caso un tanto excepcional es el servidor, el cuál es el único que requiere un puerto serial para cada terminal, de tal manera que al desear conectar mas de dos terminales, hará falta el uso de una tarjeta multipuerto.

El nombre muy sugestivo, (Planificación de una Red) nos indica que el proyecto ha sido desarrollado para entender como trabaja una LAN y cuáles son los factores a considerar en su diseño.
Trata también detalles como el cableado de la red y hace un estudio de las diferentes posibilidades existentes al respecto. Tópicos discutidos incluyen técnicas de programación, BIOS DE BP-LAN, Drivers de comunicación, y compartición de recursos.
La principal motivación para el desarrollo de este trabajo es el de que se nos introduce en conceptos y técnicas básicas de programas residentes de memoria, dispositivos instalables y hasta cierto punto conceptos de sistemas operativos. Ademas de lo expuesto, influenció notablemente la facilidad en la implementación de esta red.
Pero quizás uno de los objetivos primordiales de este proyecto es la de introducir al estudiante a la programación de la computadora personal bajo este novedoso concepto de "Grupo", que significa una ***Red de Computadoras***.

BP-LAN BOSQUEJO Y PROGRAMACIÓN

El proyecto BP-LAN va ha consistir de cuatros diferentes tipos de programas :

Bloques que manejan dispositivos	TSRs
Archivos de sistemas de operación de red	Aplicaciones

Todos los programas de BP-LAN interactúan a través de llamadas al Sistema Básico de Entrada / Salida de BP-LAN (BIOS DE BP-LAN) .

La función y valor de BP-LAN pueden ser mejor entendidos considerando sus objetivos de diseño .

VELOCIDAD: BP-LAN transmite y recibe información en velocidades de hasta 115.200 baudios.

MODULARIDAD: Tanto el código fuente como archivos ejecutables para BP-LAN deben ser altamente modulares, simplificando futuras adecuaciones y modificaciones del código .

CONFIGURACIÓN: La versión actual de BP-LAN fue completamente re diseñada. Toda la información de configuración ha sido externalizada de modo que ninguna reensamblación es necesaria para configuraciones estándar . La información de la configuración está ahora pasada a los diversos programas de BP-LAN ya sea en la línea de comando o a través de tablas en BIOS DE BP-LAN .

ADAPTACIÓN AL PROGRAMADOR: El código fuente de BP-LAN es altamente modular. Cada función tiene una interface bien definida con el resto del código, permitiendo al programador modificar cualquier caracteristica de BP-LAN sin requerir un conocimiento íntimo del programa completo .

EL USO DE MEMORIA: El uso de memoria de BP-LAN para programas residente en memoria ha sido minimizado permitiendo al administrador del sistema cargar solamente las características consideradas necesarias .

COMPARTICIÓN DE ARCHIVOS: Cualquier dispositivo de bloque (lectura / escritura) en el Servidor puede ser accedido de cualquier nodo en la red .

COMPARTICIÓN DE IMPRESORA: Cualquier impresora en el Servidor puede ser accedida como si fuera una impresora local del cliente , ademas se le provee de un mecanismo para evitar acceso simultáneo por otros clientes .

EJECUCIÓN DE COMANDO REMOTOS: Si un cliente inicia la ejecución de un comando de DOS en el Servidor , éste no responde a cualesquier solicitudes de clientes mientras esté ejecutando el comando.

SEMÁFOROS: La versión actual de BP-LAN no permite el BLOQUEO DE FICHERO . Si dos NO-APLICACIONES de BP-LAN (cualquier programa no específicamente diseñado para correr en una red BP-LAN) intenta acceso de escritura simultáneamente a un archivo ,es probable que se corrompa . Las APLICACIONES de BP-LAN pueden utilizar opcionalmente semáforos para evitar el acceso a escritura simultáneo a un archivo. Un Semáforo es una variable en el servidor que corresponde al estado de un dispositivo, archivo, o grupo de archivos compartidos .

TOPOLOGÍA: Es de topología Estrella , en donde una computadora central es tanto el conector (HUB) de red como el administrador de recurso único . En una configuración más compleja , más de un administrador de recursos puede existir.

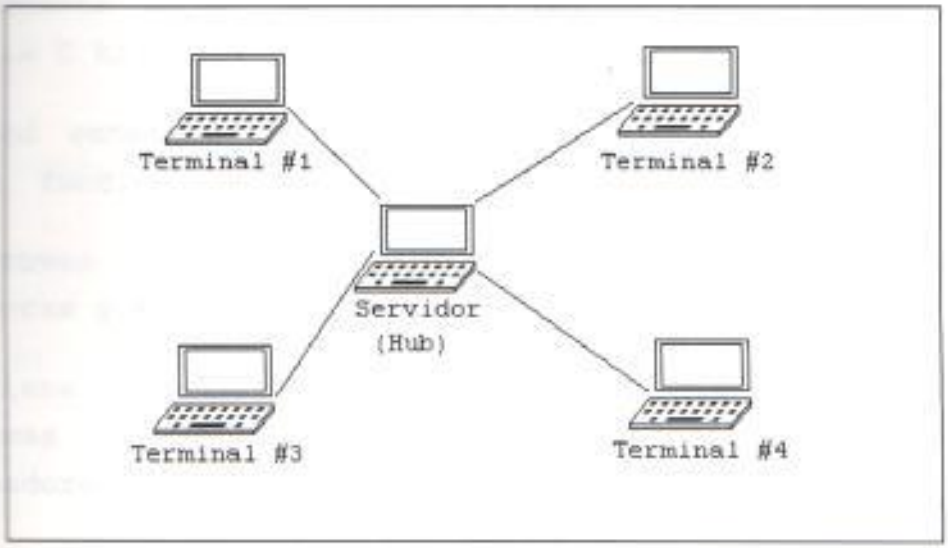


Fig. #1

1.2 CARACTERÍSTICAS DISTINTIVAS DE BP-LAN EN LA COMUNICACIÓN .

1. BP-LAN utiliza el RS-232C puerto de comunicación serie .

2. Un paquete de comunicaciones contiene muy pocos gastos (overhead).

3. Soporta hasta 256 conectores (dispositivos de red, incluyendo nodos de red, impresoras series, plotters, y modems) .

4. El cliente puede accesar a un Servidor a través de un modem (2400 baudios o 9600 baudios)

5. La red puede ser anidada, así un Servidor en una red puede ser cliente en otra red .

EN LA COMPARTICIÓN DE RECURSOS .

1. Soporta acceso de hasta 24 drives para compartición (desde la C hasta la Z) .

2. Red sensible , hasta aplicaciones , como utilitarios Norton, funcionan adecuadamente .

3. Provee a cada cliente acceso transparente a las impresoras y modem en el Servidor .

4. Tiene 256 semáforos disponibles para ser usados por programas de aplicaciones en dispositivos o archivos bloqueadores .

5. Un cliente puede iniciar la ejecución de un mando de DOS en el Servidor .

MISCELÁNEOS .

1. El Servidor puede correr ya sea en modo dedicado (procesando comandos de red en primer plano) o en modo no dedicado, es decir que puede ejecutar los mismos procesos de cualquier terminal.

2. Es provisto de un programa de instalación basado en ventanas.

3. Es provisto de un menú que maneja fácilmente correspondencia electrónica

4. El código fuente , está diseñado especialmente para simplificar la tarea de diseñar o modificar el código .

1.3. DESVENTAJAS DE BP-LAN EN LA COMUNICACIÓN :

1. A 115.200 baudios, la comunicación es casi 100 veces más lenta que los 10 Mhz , tasa de comunicación de las ultimas grandes redes .

EN LA COMPARTICIÓN DE RECURSOS :

1. El bloqueo de ficheros no es soportado .
2. La seguridad de los archivos no es soportado .

MISCELÁNEOS :

1. BP-LAN , como todas las Redes de Área Local, requieren al menos una persona con conocimientos de computación ,para instalar el hardware y configurar el software en una Red BP-LAN.

1.4. REQUERIMIENTOS

Al ser éste proyecto educativo , los requerimientos son pocos y de bajo costo, entre los que podemos nombrar los siguientes :

- La tarjeta adaptadora de red (NIC) es la tarjeta del puerto serial RS-232C.
- El servidor, que es sin duda el centro de la ESTRELLA deberá tener tantos puertos seriales RS-232C como estaciones deseara tener , de manera que al querer instalar más de dos terminales, hará falta el uso de una tarjeta multipuertos.
- En cuanto al cableado , se necesita un cable en cuyos extremos existan conectores DB-9 ó DB-25 según sean las computadoras que vayan a escoger como servidor y como estación. Estos conectores serán del tipo " hembra " .
- La conexión de los puertos RS-232C será del tipo MODEM NULO.

• Al ser una Red de baja performance por su característica educacional exclusivamente, está soportada por un software no muy complejo, que fácilmente se podrá manejar con las poderosas plataformas de hardware instaladas en las PC's en la actualidad, por lo que el hardware no es una limitante . Los requerimientos básicos de éste software son :

256K en RAM

256K espacio disponible en disco

Un Driver y Disco Duro .

1.5. SERVIDORES DEDICADOS VERSUS NO DEDICADOS

BLUE PRINT LAN, puede trabajar en las dos formas, con servidor DEDICADO, y con servidor NO DEDICADO.

Un servidor DEDICADO significa que la computadora que está ejerciendo de servidor, sólo corre aplicaciones de red, es decir se concentra en atender a los usuarios de la red exclusivamente, procesando todos los requerimientos solicitados por ellos, y no puede atender un trabajo, como estación de la red, es decir no se puede accesar por medio del teclado a el, el servidor NO DEDICADO si lo permite, realizando solicitudes de red a la vez en el BACKGROUND , mientras actúa como estación en el FOREGROUND,esperando ser accesado por teclado.

1.5.1. CARACTERÍSTICAS DEL SERVIDOR EN MODO DEDICADO

- Procesa de una manera más rápida y confiable que uno dedicado,ya que no hay ninguna otra aplicación en competencia.
- Realiza en forma segura comandos de ejecución remota y de ingreso de archivo remoto.

- Es más costoso porque requiere de la compra de un computador extra, para tenerlo sólo como servidor.

1.5.2. CARACTERÍSTICAS DEL SERVER EN MODO NO DEDICADO

- Es muy atractivo porque no requiere una computadora extra para usar como servidor, por lo tanto es económico.
- Tiene una calidad pobre y bajo rendimiento debido a que comparte tareas de estación de trabajo y de servidor.
- Sólo realiza en forma segura impresiones remotas, funciones de bajo nivel y semáforos, mientras que no son soportados o los realiza en forma insegura los comandos de ejecución remota e ingreso a archivo remoto.

2. CAPITULO

2.1. BOSQUEJO DEL BIOS DE BP-LAN.

Todas las computadoras modernas vienen equipadas con un BIOS (sistema básico de entrada\salida) éste contiene una biblioteca de funciones de entrada\salida que pueden ser llamadas desde programas de aplicación . El BIOS esta almacenado generalmente, al menos en parte, en memoria ROM, hay EXTENSIONES al BIOS que pueden ser cargadas desde disco flexible o disco duro.

Las funciones de la biblioteca del BIOS son iniciadas a través de llamadas a interrupciones (instrucción INT) , por ejemplo INT 5 llama a una dirección que esta ubicada en la tabla 5 en ésta tabla están almacenadas los vectores de interrupción (el valor de las direcciones de las diferentes funciones), debido a esto podemos cambiar el código que desempeña una función simplemente cambiando la dirección en un vector de interrupción .

Ahora el objetivo es que BP-LAN provea una extensión al BIOS de una IBM PC, mediante un programa residente en un bloque reservado de memoria RAM (TSR) , que puede ser cargado dinámicamente, es decir que puede ser cargado en cualquier momento en una sesión de DOS . El propósito de un programa TSR es cargar funciones de remplazo o extensión al sistema de operación existente , de esta manera exteriorizamos las funciones de manipulación de recursos compartidos , logrando una integración de estas capacidades , al sistema de operación , entonces todas las aplicaciones pueden beneficiarse de estos recursos .

Por ejemplo si reemplazamos INT 17H, la función del BIOS para comunicar la impresora local conectada a la PC, con código que re dirige la salida deseada para el puerto paralelo local hacia un dispositivo en el Servidor, entonces podemos

utilizar esos dispositivos . Sin embargo, no es posible soportar dispositivos de, bloque, como drives duros compartidos, simplemente reemplazando llamadas al BIOS. En lugar de esto , es necesario utilizar dispositivo driver instalable , que es un programa que reside en un bloque reservado de RAM , diferenciándose con un TSR , en que debe ser cargado en tiempo de arranque del sistema (Boot) , Siendo asi , éste dispositivo drivers tiene que ser especificado en el archivo CONFIG.SYS en el directorio principal del driver de arranque , cuyo formato es como sigue :

```
DEVICE = [PATH] FILENAME [/ PARAMETER(S)]
Ejemplo 1: DEVICE = C: \ DOS \ ANSI.SYS
Ejemplo 2: DEVICE = C: \BP-LAN \ BPMOUNT.SYS / C / O
```

Las extensiones provistas por BP-LAN al BIOS de IBM PC , son cargadas con el programa BPBIOS.COM , y las funciones pueden ser accedidas a través de INT 80 H hasta INT 82 H . La versión del BIOS del BP-LAN soporta solamente las dos capas más bajas del modelo de referencia ISO\OSI , *LA CAPA FÍSICA* que proporciona transmisión y recepción de bytes y *LA CAPA DE ENLACE DE DATOS* que es responsable de proporcionar detección y corrección de error . Todas las capas más altas son implementadas directamente en los programas de aplicaciones de BP-LAN .

Las interrupciones de capa física da funcionamiento a todo conector de entrada /salida . Un conector es cualquier dispositivo de hardware que puede desempeñar las siguientes funciones de E/S de la red :

1. Prueba si un byte ha sido 2. Lectura de un byte recibido

3. Prueba si un byte puede ser 4. Transmisión de un byte transmitido

La siguiente tabla documenta todas las funciones que son soportadas por el BIOS de BP-LAN .

2.1.1. LA CARTA DE INTERRUPCIONES DEL BIOS DE BP-LAN

2.1.1.1. Nivel de Interrupción, Interrupciones de Control (INT 80 H)

INT 80H Función 00H- Obtiene número de conectores definidos

Llamar con : AH = 00H

Retorna : AL = Número de conectores definidos

INT 80H Función 04H- Instale conector en la siguiente posición disponible

LLamar con : AH = 04H

Retorna : Nada

INT 80H Función 05H - El punto del siguiente conector disponible

LLamar con : AH = 05H

Retorna : Nada

2.1.1.2. Nivel de Interrupción , Interrupciones Físicas (INT 81 H)

INT 81H Función 00H - Prueba si un byte ha sido recibido

LLamar con : AH = 00H

BL = Número de conector

Retorna : NZ Dato recibido

Z Dato no recibido

INT 81H Función 01H - Recibe un byte

LLamar con : AH = 01H

BL = Número de conector

Retorna : AL = Byte recibido

INT 81H Función 02H - Prueba si puede transmitir desde un conector

LLamar con : AH = 02H

BL = Número del conector

Retorna : NZ Si buffer de transmisión esta vacante

Z Si buffer de transmisión esta lleno

INT 81H Función 03H - Transmite un byte

LLamar con : AH = 03H

BL = Número del conector

AL = Byte a ser transmitido

Retorna : Nada

2.1.1.3. Nivel de Interrupción, Interrupciones de Enlace de Datos (INT 82 H)

INT 82H Función 00H - Limpia el verificador de integridad de datos

LLamar con : AH = 00H

Retorna : Nada

INT 82H Función 01H - Calcula el verificador de integridad de datos

LLamar con : AH = 01H

AL = Byte a añadir al verificador de integridad de datos

Retorna : Nada

INT 82H Función 02H - Obtiene el verificador de integridad de datos

LLamar con : AH = 02H

Retorna : AX = Retorna el verificador de integridad de datos

INT 82H Función 03H - Transmite un paquete

LLamar con : AH = 03H

BL = Número del conector

DS : DX = Dirección de transmisión del paquete

CX = Longitud del paquete a transmitir

Retorna : Nada

INT 82H Función 04H - Paquete Recibido

LLamar con : AH = 04H

BL = Número del conector

DS : DX = Dirección del paquete recibido

Retorna : C X = Longitud del paquete recibido.

2.1.2. COMUNICACIÓN SERIAL

En la red BP-LAN los nodos usaran el adaptador de comunicaciones asincrónicas RS 232-C para compartir información . La tabla siguiente ilustra la interfaz normalizada RS 232-C diseñada por IBM.

Terminal	*Nombre Señal*	*Comentarios*
1	Tierra-chasis	Unidades conectadas a tierra
2	Datos transmitidos(TD)	Salida de datos de la interfaz
3	Datos recibidos (RD)	Entrada de datos a la interfaz
4	Petición para enviar (RTS)	Define condición de dato,listo para envio
5	Limpia para enviar (CTS)	Señal de entrada que permite la transmisión
6	Datos colocados listos (DSR)	Entrada que notifica estado de disponibilidad

7	Señal de tierra	Señal común de tierra única
8	Detector portador de datos (CD)	Indica presencia de datos
20	Terminal de datos lista (DTR)	Señal de salida que indica disponible para transmitir
22	Indicador de alerta	Se lleva a cabo , debido a timbre de teléfono

En la IBM-PC ,el puerto RS 232-C es controlado con el INS8250 Semiconductor Nacional Transmisor Receptor Asincrónico Universal (UART) . Este chip convierte los datos paralelos que llegan del sistema del microprocesador a datos en serie . De manera similar convierte los datos en serie que llegan desde el RS 232-C a datos en paralelo que salen hacia el sistema microprocesador.

El INS8250 contiene 11 registros de 8 bits, de los cuales se pueden accesar a 10 de ellos por medio de mapeo de memoria. El BIOS de cada computadora contiene una tabla, en la cual se puede determinar la dirección base da cada puerto. Esto se lo realiza leyendo el contenido que se encuentra en 0040:0000. Aquí la primera palabra contiene la dirección base del puerto COM1, la segunda la del puerto COM2 y así sucesivamente, dependiendo da la cantidad de puertos instalados.

La velocidad de transmisión y recepción de datos (en bits por segundo) se la determina dividiendo la frecuencia del reloj de entrada al UART (1.8432 Mhz) para un valor de 16 bits leídos de las posiciones MSB y LSB del registro "Divisor Latch" y luego dividiendo para 16.

Para programar el UART a una velocidad determinada, se efectúa lo contrario a lo explicado anteriormente, es decir, se almacena un valor determinado en ese registro.
Estos son los pasos que efectúa el BIOS del computador cuando se lo invoca a inicializar al UART (por lo general esto ocurre al encender el computador) a través de la rutina de servicio del BIOS para comunicaciones asincrónicas, INT 14H, que contiene cuatro funciones :

1. Inicializa el puerto de comandos (AH=0)
2. Envia un carácter (en AL) fuera del puerto (AH=1)
3. Envia un carácter (establecido en AL) desde el puerto (AH=2)
4. Retorna el estado del puerto en AL (AH=3)

En el INS8250 la información del formato de datos se lo obtiene de los últimos 6 bits del registro "Line Control". Las designaciones del formato de paridad se hallan en los bits 3 al 5, el número de bits de parada se hallan en el bit #2, y un código correspondiente al número de bits de datos se encuentra en los bits 0 y 1. El bit mas significativo de este registro se lo conoce como "Divisor Latch Access Bit" (DLAB). Este es usado como un registro adicional para compartir dos puertos. Si el DLAB es 0 , entonces los puertos y la dirección base+1 corresponden a los registros de recepción / transmisión y al interrupt eneable . Si DLAB es 1 , entonces los puertos están conectados lógicamente a los MSB y LSB del divisor latch.

El INS8250 UART solamente soporta velocidades de hasta 115.200 baudios , mientras que las tarjetas adaptadoras de LAN soportan tasa de comunicación de 10 Mbps(10.000.000 baudios).

Más que una baja velocidad, es la carga añadida al procesador central CPU. Las tarjetas adaptadoras de LANs transmiten y receptan paquetes completos de información independientemente del CPU , mientras el INS8250 UART puede solamente manipular un simple byte a la vez .
La información tiene que ser formateada en paquetes de comunicación antes de que sea pasado entre dos computadoras. Tales paquetes consisten generalmente de un encabezado descriptivo, un campo de información , y un campo de chequeo de error .
Antes de transmitir un paquete en BP-LAN , la computadora transmisora tiene que tener la entera atención de la computadora receptora. La computadora transmisora esta transmitiendo repetidamente el byte 01H , conocido como encabezado de inicio , o SOH , Los caracteres van a la computadora receptora hasta que la computadora hasta que la computadora receptora reconoce el encabezado de inicio con un byte 06H, conocido como un reconocimiento de caracter o ACK . La computadora transmisora escucha el eco del byte de reconocimiento , y transmite el paquete de comunicación.

La conexión entre los puertos de comunicación asincrónico RS-232C de las distintas computadoras de la realiza mediante el uso de un cable de "modem nulo" (null modem) que se lo debe construir de la siguiente manera:

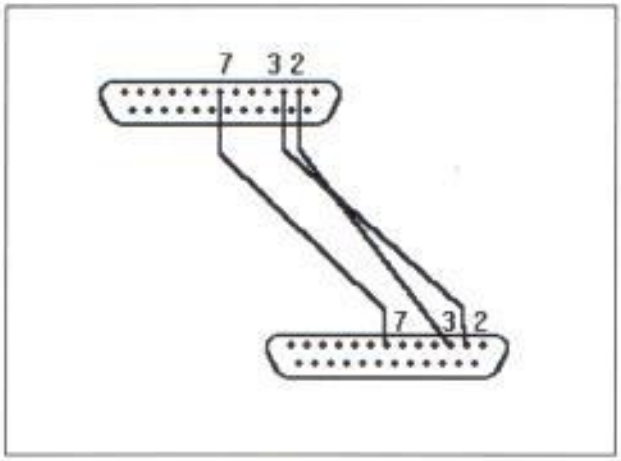


Fig. #2

Como se puede apreciar en el gráfico, se interconectan los pines de transmisión de un lado con los de recepción del otro lado y viceversa. Ademas de estos, se deberá conectar también entre ellos los pines que corresponden a la tierra de la señal.

2.1.3. TRANSFERENCIA DE DATOS

Toda la transmisión de datos se la realiza a través de un modelo determinado de datos llamados paquetes. Estos paquetes son totalmente transparentes al usuario, ya que los arma cada computador antes de transmitirlos, y los interpreta cada computador luego de haberlos recibido. Los paquetes no son mas que una cadena de datos, la cual lleva un formato preestablecido y bien definido, para facilitar de esta manera los procesos de recepción y transmisión. Es asi como, con solo interpretar la cabecera del paquete (identificador) se podrá identificar el proceso que se desea realizar, y se sabrá que tipo de datos es el que le sigue.

3 CAPITULO

3.1. TIPOS DE PAQUETES (COMANDOS):

Fig. #3

3.1.1. COMANDOS DE MENOR ESTADO.-

Cuando el usuario de la estación inicializa este tipo de comando, lo hace en el server, no espera retorno de ninguna señal de reconocimiento, es decir que hace uso de 1 sólo paquete, todo el proceso es realizado por el server, mientras que la estación puede encargarse de otras labores, un ejemplo de este tipo de comando, son los comandos de ejecución remota. ejm: llamada a impresora, llamada a las señales de DOS, llamada a comandos del DOS en el server.

3.1.1.1. COMANDOS DE MÚLTIPLE ESTADO.-

Un ejemplo de este comando son los semáforos, ya que cuando ellos actúan, concentran la actividad de la red en el usuario que le solicitó primero, mientras que los demás esperan en la cola, como ejemplo si estamos realizando un RÉCORD LOCKING, para extraer un dato de un archivo, y en ese mismo instante otro usuario está cambiando los datos, actualizándolos, vamos a tener una lectura errónea, asi que es conveniente que se bloquee a nosotros hasta que se complete la modificación.

3.1.1.2. SOLICITUD.-

A diferencia de los anteriores, retorna un estatus de información u otro dato luego de realizar su tarea, algunos siempre requieren el cambio de 2 paquetes por comando, por lo tanto son menos eficientes que los comandos de múltiple y menor estado que requieren un sólo paquete.

A continuación presentamos los paquetes de comandos utilizados en BP-LAN que utilizan solicitudes:

3.1.2. SOLICITUDES AL FILE SERVER:

Packet ID	Drive #	Sector #
2	1 Byte	2 Bytes

(FILESERV.MOD)
Lee un sector determinado en el drive especificado.

Packet ID	Drive #	Sector #	Sector de datos
4	1 Byte	2 Bytes	sobre 1024 Bytes

(FILESERV.MOD)
Escribe un sector de datos para el drive especificado y sector en el server.

Packet ID
5

(FILESERV.MOD)
Indica que el comando escribir en el sector terminó.

Packet ID	Drive #
6	1 Byte

(FILESERV.MOD)
Chequea si media en drive especificado del server tuvo que ser modificada.

3.1.2.1. SOLICITUD COMANDO DE EJECUCIÓN REMOTA:

Packet ID	Comando de DOS terminado con retorno
0	Sobre los 512 Bytes

(REMID.MOD)
Ejecuta comando especificado en el DOS en el server.

Packet ID	Device #	Caracter
P	1 Byte	1 Byte

(PRINSERV.MOD)
Ejecuta int 17H, impresora de línea llama al BIOS en el server.

Packet ID	Función	Device #	Caracter
S	1 Byte	1 Byte	0 a 1 Bytes

(PRINSERV.MOD)
Ejecuta int 14H, I/O serial llama al BIOS en el server.

3.1.2.2. SOLICITUD DE FUNCIONES DE BAJO NIVEL:

Packet ID	Dirección Offset	Dirección segmento	Longitud
c	2 Bytes	2 Bytes	2 Bytes

(LOWLEVEL.MOD)
LEE EL CONTENIDO DE MEMORIA DE UN RANGO DE DIRECCIONES ESPECIFICADAS EN EL SERVER.

Packet ID	Offset	Segment	Length	Data
m	2 Bytes	2 Bytes	2 Bytes	> 2 Bytes

(LOWLEVEL.MOD)
ESCRIBE DATOS PARA UN RANGO DE DIRECCIÓN ESPECIFICADA EN EL SERVER.

Packet ID	I/O Direcciones
i	2 Bytes

(LOWLEVEL.MOD)
LEE (ENTRADA) BYTE DESDE DIRECCIONES ESPECIFICADAS DE I/O EN EL SERVER.

Packet ID	I/O Direcciones	Datos
o	2 Bytes	1 Byte

(LOWLWVEL.MOD)
ESCRIBE
(SALIDA) BYTE LA DIRECCIÓN I/O ESPECIFICADA EN EL SERVER.

Packet ID	Código de maquina a ser ejecutado
e	Sobre los 512 Bytes

(LOWLEVEL.MOD)

EJECUTA INSTRUCCIONES DE CÓDIGO PASADAS EN EL PAQUETE PREVIO EN EL SERVER.

Packet ID	Offset	Segmento	Longitud	Llenar Caracter
	2 Bytes	2 Bytes	2 Bytes	1 Byte

(LOWLEVEL.MOD)
LLENA UN RANGO DE DIRECCIONES ESPECIFICADAS EN EL SERVER CON UN BYTE ESPECIFICADO

Packet ID	Offset1	Seg. 1	Offset2	Seg. 2	Longitud
	2 Bytes	2 Bytes	2 Bytes	2 Bytes	2 Bytes

(LOWLEVEL.MOD)
MUEVE UN RANGO DE DIRECCIONES ESPECIFICADAS EN MEMORIA EN EL SERVER.

3.1.2.3. SOLICITUD DE ACCESO AL SEMÁFORO:

Packet ID	Numero del Semáforo
	1 Byte

(SEMAPHOR.MOD)
CIERRA EL SEMÁFORO ESPECIFICADO EN EL SERVER, SOLAMENTE SI ESTE ES CORRIENTEMENTE NO APROPIADO.

Packet ID	Numero del Semáforo
	1 Byte

(SEMAPHOR.MOD)
ABRE EL SEMÁFORO ESPECIFICADO EN SERVER, SOLAMENTE SI ESTE ES APROPIADO PARA EL CLIENTE.

Packet ID	Numero del Semáforo
	1 Byte

(SEMAPHOR.MOD)
LIMPIA EL SEMÁFORO ESPECIFICADO EN EL SERVER SIN HACER CASO DE SU ESTADO CORRIENTE.

En muchos de los programas que componen el software de BP-LAN, es común el uso de las letras de los paquetes de identificación,he aquí su gran importancia en conocerlos.

4 CAPITULO

4.1. COMPARTICIÓN DE DISCOS EN RED

Los discos de red compartidos pueden ser accesados en dos modos:

- MODO SECTOR
- MODO ARCHIVO

En modo sector la computadora del usuario realiza todo el manejo de funciones de disco que implican el drive ingresado. En modo archivo la computadora del server es responsable de manejo de las funciones de disco.
En el software de BP-LAN se utiliza el modo sector para accesar a discos de red compartidos.

4.1.1. MODO SECTOR

Los datos están almacenados en sectores, en los dispositivos de acceso aleatorio tales como, floppy drives, hard drives y drives RAM.
El sector es el bloque de datos direccionable más pequeño en un disk drive. A fin de accesar datos en modo sector, el sistema operativo debe conocer, la organización física del medio de almacenaje (por ejemplo, ¿ cuántas cabezas ?, cilindros y sectores y ¿ cuántos bytes por sector ?).
El dato es escrito y leído un sector a la vez. Si un programa de aplicación requiere un sólo byte desde un archivo, el sistema operativo, debe determinar, ¿ cuál sector contiene aquel byte y luego leer el sector entero.

Compartir un disco de red en modo sector requiere una cantidad muy pequeña de código para implementar. El modo

sector se accesa a menudo de una manera más eficiente que el modo archivo, en terminos del número de paquetes que deben ser transmitidos para igual cantidad de datos.

En **modo archivo** , los datos son a menudo transmitidos, en pequeños paquetes, a menudo conteniendo un sólo caracter, encima de eso requieren algunos bytes para el header del paquete e información del chequeo del error.

En **modo sector**, el dato es transmitido en paquetes más grandes, asi genera menos transmisión de gasto u OVERHEAD. Los discos en modo sector son compatibles con casi todo tipo de software, incluyendo utilitarios, defragmentación del disco duro, ordenamiento de directorios, ordenamiento de directorios, DOS, programa CHKDSK (chequeo de disco) y utilitarios NORTON.

4.1.2. PROGRAMAS "TERMINATE AND STAY RESIDENT" (TSR)

Son programas capaces de extender el sistema operativo, es decir sus funciones ya existentes; como su nombre lo indica una vez que se corren quedan residentes en un bloque reservado de memoria de la RAM del computador.
Estos programas fueron creados con la finalidad de suplir deficiencias tales como las que se presentan al usar un EMULADOR DE TERMINAL ,el que suministra un usuario con acceso completo a los recursos de un computador remoto, pero una vez finalizada la aplicación,todo el acceso a estos recursos acaba y no pueden beneficiarse las demás aplicaciones de tales recursos.
Los TSR reemplazan o complementan las llamadas al DOS o BIOS con nuestras propias llamadas a función acostumbradas,

entonces se está añadiendo soporte de recursos compartidos donde es necesario.

Como ejemplo si reemplazamos int 17H (función de imprimir caracter del BIOS, con un código que redireccione la salida destinada para los puertos locales paralelos a un dispositivo en el server, entonces podemos utilizar aquel dispositivo.
Se debe tener mucho cuidado de añadir algún programa TSR nuevo a BP-LAN debido a que esta fue creada para un determinado número de programas TSR, tales como aquellos con los que BP-LAN se conecta por si sola a algunos vectores interrupción tales como int 14H (RS232C I/O), int 17H (salida a impresora de línea) int 21H (funciones del DOS) e int 28H (DOS desocupado), algún nuevo TSR podría reemplazar una o más de estas interrupciones.

BLOCK DEVICE .- Es un dispositivo que corres a una letra drive, en el server o en el computador hub, puede ser accesado desde algún nodo en la red, un dispositivo puede ser añadido como un volumen de lectura/escritura o lectura solamente.

4.1.2.1. LOS DEVICE DRIVERS

Los **"installable device drivers"** son programas que tienen la misión de reemplazar a los TSR's, ya que ellos además de extender rápido la funcionalidad del DOS soportan los **block devices** , cosa que no lo realizan los TSR's.
No se puede compartir un hard drive simplemente con una llamada al BIOS como lo harian los TSR's.
Reside en un bloque reservado de memoria de la RAM, al igual que los TSR's. Es cargado en el momento del booteo, mientras que los TSR's lo hacen en una sesión de DOS.

El device driver debe ser especificado en el drive de booteo, por una entrada especifica en el CONFIG.SYS, el siguiente es el formato.

DEVICE = [path] filename [/parameter (s)]

Ejemplo 1 : DEVICE = C : \ DOS \ ANSI.SYS

Ejemplo 2 : DEVICE = C : \ BP-LAN \ BPMOUNT.SYS

4.1.2.2. BPMOUNT.SYS

Es un **installable** MSDOS **device driver** que permite accesar a nivel sector desde una estación a un drive compartido específico en el server. Este **block device driver** es instalado en la cabeza del DOS device driver enlazando listas para una entrada en el archivo CONFIG.SYS file, colocando así el block device como el próximo drive disponible.

El código fuente para BPMOUNT.SYS reside en 4 archivos:

BPMOUNT.ASM (es el principal archivo fuente).

BPDISKED.MOD (es un archivo encabezador conteniendo varias definiciones red relacionada).

MISC.MOD (es un módulo conteniendo varios macros en lenguaje ensamblador y subrutinas de propósito general).

CONSOLE.MOD (contiene rutinas, las cuales manejan la salida por pantalla y entradas por teclado en el cliente).

El BPMOUNT.SYS se ejecuta de la siguiente forma:

MASM BPMOUNT;

LINK BPMOUNT;

EXE2BIN BPMOUNT BPMOUNT.SYS

El BPMOUNT.SYS se encarga de configurar el primer paso de una estación de BP-LAN como es el de añadir todos los drives compartidos. Este debe ser cargado usando la palabra clave DEVICE. La sintaxis para lograr esto es :

DEVICE = BPMOUNT.SYS / shared_drive / socket_number / write_protect_flag / max_sector_length

shared_drive es la letra que el server usa para hacer referencia a el drive, por ejemplo si la estación añade drive C desde el server, entonces en el formato va **C**.
socket_number como su nombre lo indica en este espacio va el número del socket que puede ser un número de **0** a **255** y que se coloca de acuerdo al orden en que fue definido, por ejemplo socket 0 el primero, socket 1, el segundo, etc.
write_protect_flag, en este espacio va **0** para indicar lectura/escritura y **1** para lectura solamente.
max_sector_length se refiere al número máximo de bytes por sector para este drive, usualmente **512**, otras versiones de MSDOS,versión 3.30 en particular usan longitudes de 1024 a 2048 bytes, este valor no debe ser excedido en los local hard drives.

4.1.3. COMPARTIENDO DISPOSITIVOS DE RED PARALELO Y SERIE

BP-LAN tiene incorporado en su sistema, las opciones de compartir los dispositivos ya sean paralelos o seriales. La idea de compartir uno de estos dispositivos, es la de que teniendo solo uno conectado en la red, cualquier usuario desde cualquier estación lo pueda utilizar. Generalmente se conecta a dispositivos paralelos las impresoras, y a dispositivos seriales los modems o las impresoras seriales. Si se refiere a compartir dispositivos seriales, estos pueden estar configurados como COM1, COM2, COM3 o COM4, y la comparición se la realiza mediante el uso del BPCOMDEF, que es un programa residente de memoria que intercepta todas las interrupciones 14H de DOS, que son las que corresponden a los dispositivos seriales.

En cuanto a dispositivos paralelos, nos referimos a impresoras. Básicamente el proceso que realiza esta parte es idéntico al de los dispositivos seriales, con la diferencia que aquí lo que se intercepta son la interrupciones 17H de DOS, que son las que están relacionadas con la impresión.

Al igual que en todos los procesos de BP-LAN, aquí también se hace uso de los paquetes, y para el caso de por ejemplo una impresión remota, se enviara un paquete con el siguiente formato:

Packet ID	Device #	Caracter
	1 Byte	1 Byte

Se puede usar este paquete para imprimir un caracter, pero se presentan ciertos problemas, como cuando el dispositivo está

apagado o esta fuera de papel, entonces entra a un estado que se llama **bit bucket** que significa irremediablemente perdido.
Para solucionar esto, vamos a tener un paquete llamado RETURN SERVER, cada vez que la estación desea imprimir un caracter en el server, este construye y transmite el paquete especificado anteriormente, y luego espera por un paquete estatus, el cual indica el éxito o falla de la tarea de impresión.
Luego sobreviene otro problema, que sucede si dos usuarios intentan imprimir simultáneamente,los caracteres se imprimen en el servidor en el orden en que llegan si son enviados simultáneamente, la salida será un revoltijo de los dos.
Una manera para prevenir este revoltijo es permitir el acceso exclusivo de impresión para el primero de los usuarios, y que luego libere la impresora para que el segundo usuario imprima, esto se logra sensando el marcador de fin de archivo, pero por lo general la información que se imprime es información captada al vuelo, por lo tanto carente de marcador de fin de archivo, y entonces o no se imprime por completo o nunca se libera la impresora, algún mecanismo debe liberar la impresora cuando el usuario está obviamente inactivo.
Una solución es controlar el tiempo de trabajo de la impresora, si se la detecta inactiva, el server puede liberarla sin problema, si se realizan impresiones demasiado largas, el server puede interrumpir la impresión con cierto retardo, para que no monopolice la red y luego volverla a activar.

4.1.4. SOLUCIÓN DE BP-LAN PARA PROBLEMAS DE IMPRESIÓN

El TSR BPLPTDEF.COM en el usuario ,recibe las int 17H, el IBM PCBIOS llama a imprimir un caracter para una impresora paralela, construye el paquete de impresión, y espera por un paquete estatus enviado por el server , este paquete es creado por un módulo de BPSERVER.ASM llamado PRINSERV.MOD. Si la impresora es apropiada para otro usuario, el usuario corriente espera 2 seg y luego reinicia, para asignar a otro usuario, el server detecta impresora inactiva por 30 seg, y luego asigna á otro usuario.

En caso de impresiones remotas seriales, que usan los puertos de comunicación serial predefinidos (COM 1, COM 2, COM 3, O COM 4),se ejecutara el programa BPCOMDEF.COM que se encarga de elaborar el paquete necesario conteniendo toda la información de int 14H.

Esta versión de BP-LAN no soporta dispositivo de cierre (Bloqueo) para dispositivos seriales accesados, el uso de semáforos para prevenir acceso simultáneo para usuarios multiples es recomendado.

En el lado del server PRINSERV.MOD ,se encarga de recibir las solicitudes de impresión remota y procesarlas para seleccionar la más indicada y luego mandar el paquete de estatus.

5. CAPITULO

5.1. UTILITARIOS DE BP-LAN

5.1.1. Comandos de ejecución remota:

Determinadas funciones son altamente ineficientes al ser ejecutadas en una red, lo cual se puede ilustrar en el siguiente ejemplo:

Un cliente trabajando en un terminal desea copiar un archivo largo de un lado del disco de red a otro lado en el mismo disco. Esto implica que el archivo debe pasar inicialmente del servidor al terminal, para luego regresar del terminal al servidor y almacenarlo. Como se aprecia claramente, este proceso involucra un gran trafico innecesario en la red.

Para eliminar problemas de este tipo, se han implementado los comandos de ejecución remota. Este proceso es mucho mas sencillo, ya que el cliente envia solo la información del nombre del archivo y el lugar donde deberá ser copiado. Luego de esto, la estación ya no se involucra en el proceso, ya que ahora el SERVIDOR hará todo el proceso de copia directamente, eliminando de esta manera el trafico innecesario.

Este tipo de ejecuciones se la realiza con el programa BPREMOTE que lleva los siguientes parámetros:

```
BPREMOTE /numero_socket/comando_DOS
```

Por ejemplo: BPREMOTE /0 /COPY C:\WP51\LISTADO.TXT C:\

En este ejemplo se indicara el servidor que deberá copiar el archivo LISTADO.TXT que se encuentra en el subdirectorio WP51 al directorio raíz.

El parámetro numero_socket se refiere al numero de socket que conecta al terminal con el servidor.

5.1.2. Comandos de Impresión remota:

En los capítulos anteriores se explico el porque de la necesidad de poder compartir impresoras remotas seriales o paralelas.

Para compartir una impresora remota se usara el siguiente comando:

```
HELPTDEF /numero_socket /dispositivo_cliente /dispositivo_
servidor
```

Nótese que el DISPOSITIVO para LPTn = n - 1

De tal manera que si deseamos imprimir los datos que deberían salir en el puerto LPT1 en el terminal, en el puerto LPT2 del servidor, y la comunicación se realiza a través del numero_socket = 3, deberíamos ejecutar el comando:

```
HELPTDEF /3 /0 /1
```

Para impresoras seriales se deberá ejecutar el comando HELCMDEF con parámetros idénticos al caso anterior.
Nótese que el DISPOSITIVO para COMn = n - 1

Ej. el mismo ejemplo anterior, pero en lugar de puertos paralelos usaremos puertos seriales, la notación será:

```
HELCMDEF /3 /0 /1
```

5.1.3. Mostrando y modificando definiciones de discos:

Los discos de BP-LAN son especificados en el CONFIG.SYS a la hora de arrancar el computador. Pero estos dispositivos pueden ser modificados para una sesión en particular mediante el uso de BPDRIVES. Este comando permite cambiar el acceso a los discos, el acceso a escritura y el numero de socket con el cual están conectados cliente y servidor.

BPDRIVES /disco_cliente /disco_server /derechos_acceso /numero_socket

Disco_cliente se refiere a la letra del disco virtual en el terminal.
Disco_server se refiere a la letra del disco del server con la cual se hace referencia a dicho disco
Derechos_acceso indica si en el disco del servidor solo se puede leer o hacer el proceso completo de lectura-escritura.

5.1.4. Hora del servidor:

La hora del servidor se puede leer a cualquier momento desde cualquier terminal. Esto se lo usa para sincronizar todos los relojes a la misma hora del servidor.

BPTIME /numero_socket

Correo electrónico:

El correo electrónico es un mecanismo para intercambiar datos en forma de texto entre dos usuarios de una red.
Estos datos son convertidos a mensajes de correo, mediante la superposición de la información tanto del que envía, cuanto la del que recibe.

5.1.5. Ejecución del correo electrónico:

BP-LAN utiliza las siguientes variables de ambiente de DOS: BPDIR, BPEDITOR, BPMAILDIR, PBUSER y COMSPEC. BPDIR es el directorio donde se encuentran los archivos ejecutables del correo electrónico. BPEDITOR es la ruta de acceso completa del editor de texto definido por el usuario. BPMAILDIR es el directorio en el server, en el cual se almacenaran los distintos mensajes. BPUSER es el nombre del usuario, tanto para el que envía, como para el que recibe mensajes. COMSPEC es la ruta completa de acceso al COMAND.COM. En caso de haber una de estas variables no definidas, el programa pedirá al usuario que las defina.
Ejemplo:

SET BPMAIL=D:\BPMAIL (Especifica como directorio el BPMAIL en el disco D)
SET BPEDITOR=C:\DOS\EDLIN.COM (Especifica como editor al EDLIN que esta en el disco C y en el directorio DOS)

El correo electrónico se ejecuta mediante el comando BPMAIL desde el DOS. En caso de no tener correo el usuario, se mostrara un mensaje indicando ese hecho; Caso contrario se mostraran todos los correos que el usuario haya recibido.
Luego de haber leído algún mensaje, se mostrara un menú de cuatro opciones: S - enviar un mensaje; V - leer un mensaje; E - salir y grabar mensaje; X - salir sin grabar mensaje.

5.1.5.1. S - Enviar mensaje:

Al oprimir la "S" el programa permitirá crear un archivo de mensaje y enviarlo a algún usuario. Inicialmente se pedirá el nombre del usuario y una breve descripción del mensaje.

Seguido se pedirá especificar el origen del ingreso del mensaje, esto puede ser: K - para escribir el mensaje; E - para ejecutar el editor de linea especificado y F - para un archivo ya existente.

5.1.5.2. V - Ver un mensaje:

Esta opción permitirá leer el contenido de un mensaje. Los mensajes son mostrados por partes de 16 lineas a la vez. Oprimiendo la barra espaciadora se mostraran las siguientes 16 lineas. Con la tecla "T" se podrá regresar al inicio del mensaje.

Con la flecha hacia abajo seleccionaremos el siguiente mensaje y con la flecha hacia arriba seleccionaremos el mensaje anterior.

Hay la opción de almacenar algún mensaje especifico como archivo de tipo texto. Esto se logra oprimiendo la tecla "F" y luego habrá que ingresar el nombre que deseamos dar al archivo.

Si se desease obtener una copia impresa del mensaje, deberemos oprimir la "P", y se imprimirá en el puerto LPT1.

La opción de borrado de mensajes se la realiza con la tecla "D". Pero esto solamente le pone una marca al mensaje, y este será borrado al salir del programa. En caso de algún arrepentimiento, se podrá desmarcar el mensaje mediante la tecla "U".

5.1.5.3. Q - Salir y grabar mensaje:

Esta opción sirve para terminar la ejecución del programa y borrar todos los mensajes leídos.

5.1.5.4. X - Salir sin grabar mensaje:

Oprimiendo "X" terminaremos con la ejecución del programa sin almacenar o actualizar los cambios realizados. Tampoco serán borrados los archivos marcados para tal efecto.

6. CAPITULO

6.1. UTILITARIO AUTOMÁTICO DE INSTALACIÓN BP-LAN

Para una fácil instalación de la LAN SERIAL, es necesario correr el programa automático de instalación BPINST.EXE, el cual nos provee una interface muy amigable con el usuario, que en este caso es el administrador de la Red.

Este programa, cuyo código fuente está escrito en TurboPascal 5.0, de una manera interactiva y a través de preguntas sencillas modifica el CONFIG.SYS y AUTOEXEC.BAT de la computadora apropiadamente.

6.1.1. REQUERIMIENTOS GENERALES:

- Un IBM PC o compatible
- Como mínimo 256K en RAM
- 256K disponible de espacio en disco
- Dos Drives o un drive y un Disco Duro.
- El disco de instalación conteniendo los archivos:
  - BPINST.EXE
  - BPBIOS.COM
  - BPSERIAL.COM
  - BPMOUNT.SYS
  - BPMAIL.EXE
- Impresora Opcional.

6.1.2. REQUERIMIENTOS DEL CLIENTE:

-Tener disponible un puerto serial RS-232C para cada servidor.

-Por lo menos 1 drive
-Disco duro opcional
-256k en RAM

6.1.3. REQUERIMIENTOS EN EL SERVIDOR:

-Tener disponible un puerto serial RS-232C para cada cliente.
-Por lo menos 1 drive
-Disco duro opcional, pero recomendado.
-256K en RAM
-Impresora opcional pero recomendada.

6.1.4. INSTALACIÓN:

El programa de instalación automática BPINST.EXE puede correrse desde cualquier disco de instalación previamente preparado con todos los archivos ejecutables o por lo menos los que se indican inicialmente. Esto es especialmente recomendado si no se va a usar frecuentemente la RED.

Si el uso de la red es muy frecuente, se recomienda bootear desde el disco duro.

Todo el procedimiento de instalación se lo realiza a través de ventanas:

- Ventana de TRANSCRIPCIÓN
- Ventana de AYUDA
- Ventana de PREGUNTA

6.1.4.1. PREGUNTAS GENERALES

1. La primera pregunta trata de que drive contiene el Disco de instalación.

A: o B:

2. La segunda pregunta trata de cuál es el boot drive

A: o C:

3. La tercera pregunta es respecto al sendero y nombre completo del directorio en donde se desea almacenar los archivos de BP-LAN, por ejemplo:

C:\BP-LAN

4. Por ultimo, el programa automático de instalación pregunta al administrador de la red si todas las respuestas a este bloque de preguntas están correctas

Y/N/E (Si, No, Exit)

6.1.4.2. PREGUNTAS DEL PUERTO SERIAL

1. La primera pregunta al respecto de los puertos seriales es cuantos sockets de red (RS-232C) desea instalar.

Aqui se pueden definir un máximo de 256 sockets que es el máximo numero posible en un byte (socket 0 a socket 255),

pero sabemos que esto es teórico y se aleja de toda realidad práctica.

Cabe anotar que si el caso es de más de dos puertos seriales,es decir más de dos clientes, habrá que instalar una tarjeta multipuerto en el servidor.

2.El programa preguntará cuál es la dirección base hexadecimal de cada socket, generalmente 3F8, 2F8, 3E8, etc.

3.La siguiente pregunta es respecto a la rata de baudios (baudrate) de cada socket, donde velocidades de 57600 baudios para lasmás lentas y 115200 baudios para las más rápidas.

6.1.4.3. PREGUNTAS DE DRIVE COMPARTIDOS

1.El programa pregunta cuántos drives compartidos desea montar.
Si es en el servidor la respuesta debe ser "0" y las siguientes preguntas no se responderán ya que el concepto es que el servidor no tiene drives compartidos.

Si el caso es en los clientes, se deberá responder el numerode drives del servidor que desearia compartir el cliente, puede ser disco duro, floppy drives, etc.

2.La siguiente pregunta es con respecto al drive de mas alta prioridad para la instalación de BP-LAN que no está en red.

Si el sistema tiene dos drives la respuesta puede ser b:
Si el sistema es de disco duro la respuesta puede ser c:

3.El utilitario también pregunta por cada socket conectado cada cliente para cada drive compartido.

Así por ejemplo: socket # 0 para el drive compartido D:

4.La cuarta pregunta se refiere a que drive en el servidor corresponde a tal drive compartido, por ejemplo:

El drive C: en el server corresponde al drive compartido dem el mismo servidor.

6.1.4.4. PREGUNTAS DE CORREO ELECTRÓNICO

Como la red tiene un correo electrónico es necesario en el proceso de instalación la inclusión de un bloque de preguntas destinado al efecto.

Si el proceso de instalación se lo está haciendo en el servidor, no se deberá responder a las preguntas de este bloque,sino que se las parará presionando RETURN.

Como paso previo debemos crear manualmente un directorio donde almacenar los mensajes del correo electrónico en el server y en un drive compartido, ejemplo: D:\mail

1.Se preguntará por el sendero del directorio donde se ha decidido previamente almacenar los mensajes: D:\mail

2.Luego preguntará por el sendero del cliente donde está el procesador EDLIN.COM (editor de texto ASCII), ejemplo:

c:\dos\edlin.com

3.La ultima pregunta se refiere al único nombre o identificación del usuario. El nombre de usuario corresponde

archivo en el directorio de correo por lo que se deberá escoger un nombre compatible, por ejemplo:

PEDRO, PEDRO_1, etc.

El utilitario copia todos los archivos que comienzan con "\" del drive fuente al directorio de la red o destino.

También el AUTOEXEC.BAT y CONFIG.SYS en el drive de booteo son copiados como AUTOEXEC.OLD y CONFIG.OLD respectivamente en el directorio de destino de la red. Un archivo tipo BAT llamado BPRUN.BAT, el cual setea la variables del DOS, incluyendo el sendero, y configura todos los sockets de red, es creado.

Una linea que llama a BPRUN.BAT es añadida al final del AUTOEXEC.BAT en el drive de booteo. Los comandos de los dispositivos drivers a ser montados en los drives compartidos son añadidos al final del CONFIG.SYS en el drive de booteo.

4.1.5. REMOCIÓN DE LA RED

Para remover la red del sistema es necesario seguir los siguientes pasos:

1.Copiar el CONFIG.OLD del directorio destino de la red a CONFIG.SYS en el drive de booteo.

2.Copiar el AUTOEXEC.OLD del directorio destino de la red a AUTOEXEC.BAT en el drive del booteo.

3.Borrar todos los archivos de la red del directorio de destino de la misma.

4.Remover el directorio destino de la Red.

7. CAPITULO

7.1. Análisis de los programas

Para poder implementar la red por medio del BP-LAN, hay que cargar inicialmente unos programas residentes de memoria, de los cuales se hablo hace unos cuantos párrafos.
Para la implementaron de este tipo de programas hay algunas alternativas. Los que están implementados en este sistema son los mas sencillos, ya que solo hacen uso de la INT 27h.
Decimos que es la manera mas sencilla, ya que lo único que se hace es apuntar con bx al segmento que deseamos mantener residente. Hay otro tipo de programas residentes en memoria, que utilizan otras técnicas. Por ejemplo se suele reservar un bloque de memoria, que es en donde se cargara el programa. Este método tiene la ventaja de que se puede hacer una verificación, para determinar si ya se cargo el programa como residente o no. Ademas estos por lo general también se los puede desinstalar.
En nuestro caso no se los puede desinstalar ni tampoco se puede determinar si es que ya ha sido instalado.
En BP-LAN tenemos dos principales programas residentes. Son BPBIOS.COM y BPSERIAL.COM y son los básicos para su operación.
BPBIOS es el que redefine los bios y le incluye las nuevas interrupciones que serán usadas por esta red.
BPSERIAL es el que especifica el puerto serial, la velocidad de transmisión, el tipo de paridad, la longitud de datos y el numero de bits de parada.

7.1.1. BPBIOS.ASM:

En este modulo se especifican las nuevas interrupciones y se establece que parte de programa se debe ejecutar al ser invocada una de ellas. Esto se lo realiza mediante la INT 21H con AH=25H, que es SET INTERRUPT VECTOR. En AL se carga el número de la interrupción y en DX se carga el offset al puntero del manejador de interrupciones. Asi tenemos por ejemplo:

```
mov  ah,25h
mov  al,phys_int
mov  dx,offset physical_interrupt
int  21h
```

En este ejemplo se ve claramente como se cargan los distintos registros (Nótese que phys_int esta definido como 81H. El offset physical_interrupt apunta a ese procedimiento que se halla en PHYSICAL.MOD.

El mismo procedimiento se hace para las otras interrupciones definidas.

7.1.2. BPBIOSHD.MOD

Es un programa en el que se definen muchas constantes de operación, por lo que debe estar incluido en la mayoría de los programas.

Aquí se definen constantes de uso general como son las de cancelación, inicio de cabecera, reconocimiento y negación del reconocimiento.

Aparte se definen las interrupciones para cada capa. Se nota que hay capas que no están implementadas en este tipo de red, pero que se han dejado definidas sus interrupciones para implementaciones futuras.

Sin duda que las partes de mas importancia la constituyen los macros de BPBIOS, GET_PACKET y PUT_PACKET.

BPBIOS: Al invocar este macro, debemos incluirle una serie de parámetros, para que se pueda realizar su ejecución. Estos parámetros son:

intnum: Se especifica el numero de interrupción que se desea ejecutar.

funcnum: Se indica la función que se desea realizar. Esto es, ya que una misma interrupción puede ejecutar varias funciones distintas. Estas son cargadas en AH.

portnum: Se establece a través de que puerto de esta realizando la comunicación.

Aparte de estas hay otras de menor importancia, y que para el análisis que se va a realizar no intervienen.

GET_PACKET: Este macro y el de PUT_PACKET son idénticos. Se reciben una serie de parámetros, los cuales son asociados con sus respectivos registros.

Los distintos parámetros que se establecen son los siguientes:

portnum: Se especifica el puerto de comunicación.

packet_length: longitud del paquete.

packet_address: Dirección del paquete.

Una vez realizado esto se ejecuta INT LINK_INT. Esto es una interrupción concerniente a la capa de enlace. Se intercepta esa ejecución y ejecutan procesos definidos en DATALINK.MOD. Es en ese modulo que se realizan las operaciones de transmitir o recibir paquetes.

7.1.3. MISC.MOD:

Este modulo esta incluido en la gran mayoría de los programas, ya que incluye funciones sencillas pero de uso

general. Aqui hallamos macros y procedimientos como los siguientes:

pushall: Hace un push de todos los registros

popall: Hace un pop de todos los registros, pero manteniendo consistencia con el pushall

get_opt: Aqui se determinan los parámetros que han sido ingresados en la linea de comando, y que son necesarios para la ejecución de un programa.

get_dec: Es llamado en caso de que el parámetro este ingresado en forma hexadecimal. Esto puede ocurrir al ingresar posiciones de memoria correspondientes a los distintos puertos que se usaran.

get_hex: Idéntico al caso anterior, pero es llamado en caso de que los datos estén en forma hexadecimal.

get_ticks: Esta es la manera básica de obtener la hora de un computador. Especifica mente en la posición 0000:046h se almacena un valor que corresponde al numero de "TICKS" ocurridos desde media noche.

wait_a_sec: Es un pequeño retardo de aproximadamente un segundo.

slash: En la linea de comando, al ingresar los parámetros, estos son ingresados precedidos de un "/". Entonces este programa trocea la linea carácter por carácter y cuando encuentra un "/" sabe que lo que sigue es un parámetro u opción del programa.

uppercase: Este convierte los caracteres minúsculas de AL en mayúsculas.

Aparte de estos, hay algunos macros para el movimiento de datos entre registros y segmentos.

7.1.4. LOWLEVEL.MOD:

Aqui se encuentran todas la funciones y procedimientos de bajo nivel, necesarios para el proceso de los paquetes.

Muchas de estas funciones son llamadas principalmente mediante el **PROCESS_PACKET** del **BPSERVER.ASM**

Todos los paquetes utilizados por BPLAN llevan en la cabecera un carácter que determina el tipo de paquete.

Estos caracteres pueden ser los siguientes:

r: Indica que el paquete lleva un requerimiento de lectura de un rango especifico de memoria del servidos. Es ejecutado por medio de peekmem.

w: Es lo contrario del caso anterior, es decir que escribe en un rango de memoria dado. Lo ejecuta pokemem.

i: Lee un byte de una dirección especifica de I/O del server. Es procesado por portin.

o: Escribe un byte en una dirección especifica de I/O del server. (portout).

e: Ejecuta un código de instrucciones, pasados en el paquete anterior, en el servidor.

f: Llena una dirección especifica del servidor con un byte especifico (fillmem).

m: Mueve un rango determinado de memoria del servidor a otro rango de memoria del mismo servidor.

De todas estas funciones, ahora nos dedicaremos a analizar una de ellas que por el momento las consideramos las mas importantes. Estas son:

1.- **portin:** El formato del paquete que llega es el siguiente:

Packet ID.	I/O Address
" i "	2 Bytes

Mediante la interpretación de la cabecera del paquete, determinamos que corresponde a un ingreso de un byte por un

puerto (Ya que es "i"). Así mismo podemos determinar la dirección del puerto que se encuentra en los 2 bytes que siguen en el paquete. Estos se hallan en PACKET_BUFFER, y son cargados al registro DX. Ahora se hace un IN.
Los siguientes pasos son un tanto confusos, ya que no se ajustan a los diagramas de flujo indicados, y asi mismo crean un nuevo tipo de paquete (Con cabecera "D" y PACKET_LENGTH = 2), del cual no se hace ninguna mención en el libro.
Una vez que ha leido el byte del puerto, envia el paquete por medio de put_packet y los siguientes parámetros: socket_num, packet_length, offset packet_buffer. A su vez carga BL con portnum, CX con packet_legth, DX con packet_address y AH con transmit_packet. Este ultimo corresponde a una interrupción del Data Link Layer, para luego hacer una int link_int. Esta interrupción es interceptada por el bpbios, que es una especie de "interpretador". Le podemos decir asi, ya que interpreta la interrupción y determina si es una interrupción normal del DOS o es una de las ampliadas por BPLAN. En vista de que ya esta instalado un nuevo vector de interrupciones, lo que se ejecuta con esa orden es un procesa llamado out_packet que se encuentra en DATALINK.MOD. Aqui es donde se utiliza el transmit_byte, de manera que el DATALINK sabe que se debe ejecutar el out_packet.
2.-**PORTOUT**: Este al contrario del anterior, pone un dato en el puerto para ser enviado. Al recibir el paquete, este contiene en la cabecera una "o", seguido de 2 bytes de la dirección de I/O y seguido de un bytete de DATA. Con esto carga la dirección del puerto en DX, y en AL carga el dato a ser enviado. Para enviar el dato se ejecuta la función OUT.

7.1.5. FILESERV.MOD

Este modulo es el que se encarga de los requerimientos de lectura, escritura y chequeo de MEDIA. Los paquetes van encabezados con cualquiera de las siguientes opciones:

R: Para ejecutar una acción de lectura

W: Para realizar una escritura

M: Para hacer una revisión al MEDIA

ESCRITURA: En caso de que la cabecera del paquete lleve una "W", se ejecutara el proceso llamado "wrisec". Lo que se hace inicialmente es cambiar una bandera, indicando de esta manera que el media ha cambiado. El media se refiera propiamente dicho al disco duro, su estructura, tipo y contenido. El objetivo de cambiar esta bandera es el indicar que el contenido que tenia el disco ha cambiado. Esto se hace principalmente debido a que los servidores están provistos de una memoria cache para el disco duro; de tal manera que, antes de buscar la información en el disco, lo hará primero en el cache. La idea de hacer esto es la de incrementar la velocidad, ya que en la memoria cache no se involucran movimientos mecánicos, que son los que en definitiva retrasan los tiempos de proceso.

Entonces, al recibir un requerimiento de lectura de disco, se verificara primero la bandera de MEDIA. Si la bandera indica que hubo un cambio, se accesará directamente al disco, caso contrario se accesará al cache, evitando así datos incorrectos o desactualizados.

En el proceso normal de recibir un paquete con cabecera "W", luego de cambiar la bandera, se creara un nuevo paquete con cabecera "D". Esto al igual que en el caso de portin es un paquete desconocido, del cual no se hace mención en el libro. Seguido se puede leer del paquete la información del disco, de los bits MSB y LSB del sector inicial y la longitud.

Packet ID	Drive #	Sector #	Length of Data
" W "	2 Bytes	2 Bytes	2 hasta 1024 Bytes

Dentro del programa se inicializa otra variable llamada "funcion" que especifica a otros segmentos del programa si se trata de lectura (si funcion=0) o escritura (si funcion=1). En BX serán almacenados los datos que deberán ser escritos y luego se ejecutara el modulo "sector_io".
En este caso se inicia CX=1 para trabajar con 1 sector y en DX se especifica el sector inicial. En BX serán cargados los datos y luego se ejecutara la INT 26h que es la de escritura absoluta de sector.
En este punto se termina con la ejecución de sector_io y se retorna a wrisec, donde leemos un nuevo paquete y repetimos el proceso hasta encontrar un paquete con cabecera "T", que indica fin de los paquetes.
Aquí encontramos otra inconsistencia, ya que el servidor de archivos debería ser el que genere un paquete con cabecera "T", para que de tal manera la estación sepa que el proceso de escritura ha sido realizado exitosamente, pero, siguiendo la secuencia del programa, se interpreta que uno de los paquetes que envía el terminal lleva cabecera "T". Según la tabla de la pagina 81 del libro guia, entre los paquetes del server se halla uno que es indicador de que la escritura de un sector se ha ejecutado, lo que se contradice con la secuencia del programa.
LECTURA: La lectura se produce casi idénticamente a la de la escritura. El paquete lleva el siguiente formato:

Packet ID.	Drive #	Sector #
" R "	1 Byte	2 Bytes

Se reciben los parámetros necesarios para la lectura de un sector especifico del disco. Nótese que en estas circunstancias no será necesario modificar la bandera de MEDIA, ya que la lectura no altera en nada el contenido del disco. Es por esto que directamente se ejecuta la lectura por intermedio de la INT 25h. Los registros que se cargan son:

AL= Numero del disco
BX= Offset de dirección de transferencia
CX= Numero de sectores a leer
DX= Sector inicial

Una vez realizada la lectura, se creara un nuevo paquete con cabecera "E". Aquí encontramos un nuevo problema, ya que un paquete de este tipo es de ejecución remota en el servidor. En realidad no se entiende el porque de este tipo de paquete, el cual será ensamblado y enviado por put_packet. De esta manera se termina el proceso de lectura de disco.

Vale anotar que la interrupciones 25H y 26H corresponden a lectura y escritura absoluta de disco. Es decir que en este tipo de acceso a disco se pierden las ventaja de usar la INT 21H que provee manejo de directorios y bloqueo y desbloqueo de archivos.

Debido a que INT 26H y 25H tratan todos los registros como una longitud de sector, se debe implementar un propio sistema de bloqueo y desbloqueo de archivos. Esta operación accesa directamente un sector o bloque de sectores.

Dentro de este modulo también se determina la cantidad de espacio libre en disco y se lo hace mediante:

```
mov  ah,36h
mov  dl,drive
int  21h
```

Retorna:
AX= sectores por cluster
BX= # de clusters disponibles
CX= # de bytes por sector
DX= # de clusters por drive

7.1.6. CONSOLE.MOD

Contiene una serie de procedimientos de muestreo y manejo de pantallas así como de lectura de teclado. Forma parte de REMOUNT.SYS.
Inicialmente se define BACKGROUND=0, indicando que la operación se realiza en FOREGROUND.
Contiene procedimientos como por ejemplo:
ESCTST: Este sensa si se ha oprimido la tecla ESC. Para esto usa la INT 16h con AH=01 para leer el status. Si no hay carácter disponible se enciende la bandera de ZERO. Caso contrario el valor de carácter se carga en AL. En caso de contener AL el numero 27, que es el asociado con la tecla ESC, se sabe si ha sido o no oprimido esa tecla.
ESCAPE: Este proceso hace uso de ESCTST para verificar si la tecla fue o no presionada. En caso afirmativo de ejecuta INT 20h que es para finalizar el programa (Program Terminate).
CLS: Proceso para limpiar la pantalla.
CHAROUT: Aqui se hace escritura en pantalla en modo teletipo, ya que se usa AH=0eH e INT 10H. Al contiene el carácter que deseamos mostrar.

7.1.7. BPTIME.ASM

Este modulo sirve como ejemplo para explicar un poco mejor algunas de las operaciones realizadas por LOWLEVEL.MOD. Este modulo se ejecuta en las estaciones de trabajo, y su función

es la de obtener información de la hora del servidor. Esto se hace, debido a que por lo general las estaciones de trabajo suelen ser computadoras muy limitadas, es decir, muchas veces hasta carecen de una tarjeta que retenga la información de la hora y fecha. Esto implica que cada vez que se enciende el computador, haya que actualizar manualmente la fecha y hora. Entonces, al ejecutar este programa, se la fecha del servidor, que es una computadora mas compleja que si esta provista de ese tipo de tarjeta. Ademas de esta manera se unifica la hora te todas las computadoras conectadas.
Este programa arma un paquete de la siguiente manera:

PACKET ID.	ADDRESS OFF.	ADDRESS SEG.	LENGTH OF DATA
" r "	046CH	0000	0004

El paquete lleva esta forma debido a que es un proceso de lectura ("r") de un valor de memoria del servidor.
Se leerá la posición 0000:0046CH que es donde todas las computadoras almacenan la información de la hora. Y se especifica una longitud de 4 debido a que esa es la longitud de la hora.
La notación para la ejecución es la siguiente:
RTIME / numero_de_socket
Donde numero_de_socket determina a través de que socket se realizara la comunicación.
Se empieza por hacer las siguientes definiciones:
time_request: Esto corresponde al requerimiento que deseamos hacer, que en este caso es "r" para hacer la lectura en el server.
time_addoff: Aquí se especifica el offset de la dirección de al memoria a leer que es 046CH.

time_addseg: Se especifica el segmento (0000)
time_length: Longitud del dato a leer. En este caso es 4.
Inicialmente se hace una verificación de que la linea de comando incluya el numero_de_socket. Luego se ejecutan dos funciones que se definen en BPBIOSHD.MOD que son las de put_packet que pone el paquete en el medio de transmisión para que el servidor lo interprete. Luego el servidor devuelve el paquete y se lo recibe con get_packet. Una vez recibido el paquete, pone los valores en sus registros correspondientes y de esta manera se actualiza la hora. El programa finaliza con INT 20H de terminación de programa.

Este es un programa que viene incluido en la red, aunque no tenga nada que ver con la red propiamente dicho. Este solo sirve de ayuda para poder determinar si los puertos de comunicación están bien definidos.
Este programa simplemente muestra lo digitado en un computador en la pantalla del otro y viceversa. Si se establece correctamente la comunicación, no deberian haber problemas posteriormente.
Lo interesante de este programa es que utiliza el BPBIOS y el SERIAL como programas de soporte.
El modo de operación de este programa es muy sencillo y opera de la siguiente manera:
Inicialmente se establecen los parámetros que han sido ingresados en la linea de comando, específicamente del numero de puerto.
Luego se lee el estatus del teclado mediante INT 16H AH=1.
Esto se lo hace para determinar si ha sido oprimida alguna tecla. Si se ha detectado la presencia de alguna tecla, se determina cual fue, mediante la INT 16H AH=0. El valor de la tecla que fue oprimida se cargara en AL.

Se verifica que la tecla presionada no haya sido ESC, ya que en este caso se terminara con la ejecución del programa.
Una vez que se tiene cargado el registro AL, se invoca al BPBIOS con los sgtes parámetros: PHYS_INT,TRANSMIT_BYTE, PORTNUM. Con PHYS_INT se especifica que queremos realizar una interrupción a la capa fisica. TRANSMIT_BYTE indica que deseamos transmitir el byte que tenemos en AL, y PORTNUM especifica el numero del puerto.
En caso de que no se detecte la presencia de ninguna tecla, se hace un BPBIOS PHYS_INT,RECIEVE_STATUS,PORTNUM. Con esto se hace una verificación al puerto de comunicación, para determinar si hay algún caracter presente. En caso negativo se repite todo el proceso, pero en caso positivo se lee el caracter mediante BPBIOS PHYS_INT,RECIEVE_BYTE,PORTNUM. Luego de leer el caracter del puerto se lo muestra con INT 10H y AH=0EH.

8. CAPITULO

8.1. PROBLEMAS Y SOLUCIONES DE HARDWARE

8.1.1. CABLEADO

Cerca del 95% de los problemas en una Red de Computadores son ocacionados por los cables. Estos problemas se pueden identificar y solu-cionar siguiendo los items de chequeo que a continuación se relata:

1.- Corresponde el cable al tipo de conexión recomendada?
El tipo de conexión recomendada es la de Modem Nulo donde se contemplan líneas de recepción, transmisión y de tierra.

2.- Es el cable de la longitud adecuada?
La longitud del cable debe minimizarce por razones económicas y de operación, éstas últimas ocacionadas por la degradación de la señal en este medio de transmisión.
Puede ser necesario reducir el baud rate para comunicaciones en cables de longitud mayor a 50 metros.

3.- Pasa el cable con por lo menos 6 pies de cualquier equipo generador de fuertes campos electromagnéticos?
Tales como: lámparas fluorescentes, motores de neveras.
Estos campos electromagnéticos inducen corrientes en los cables, degradándo la señal a transmitir.

4.- Tiene el cable torceduras?
Una torcedura puede indicar rotura interna de alambres, por lo que será un factor a chequear.

5.- Está algún alambre del cable roto?
Use continuamente el multimetro u otro tipo de probador para comprobar que ninguna conexión haya discontinuidad eléctrica.

6.- Está algún cable cortocircuitado?
Verificar ésto, ya que se produce frecuentemente en el mismo conector.

7.- Están los cables atornillados en el lugar correcto?
Muchos cables podrían atornillarse antes de verificar que sea la conexión correcta.

8.- Ha probado reemplazando un cable bajo sospecha de falla con uno que se tenga la seguridad de una buena operación?
La única manera de asegurarnos de que tal o cual cable es el que falla es reempalzándolo con uno ya probado como bueno.

PUERTO RS-232C

Una vez asegurados de que los cables no presentan falla, los siguientes puntos a revisar son los relativos a los puertos RS- 232C, a continuación enumerados:

1.- Está la tarjeta ajustada firmemente en el slot?
Hay que asegurarse que este ajustada y atornillada.
2.- Si la tarjeta del puerto está en el slot 8 de una IBM XT-compatible?
Estas tarjetas deben configurarse de manera diferente en el slot 8 de una IBM XT- compatible, si la instalación manual no

provee la información de configurar la tarjeta para le slot o no lo instale en ese slot

3.- Está alguno de los slot de la tarjeta dañado?
Slots sucios, con polvo, o torcidos son fallas potenciales en las tarjetas.

4.- Ha probado mover la tarjeta hacia otro slot?
Algunas veces las tarjetas madre estan combadas y no ofrecen un buen contacto en algún slot o sino algún punto de soldadura está roto o partido.

5.- Están dos puertos RS-323C mapeados en el mismo lugar?
En ése es el caso ninguno de los dos puertos funcionará adecuadamente.

6.- Está el puerto RS-232C siendo manejado por interrupciones?
Muchas tarjetas de puerto RS-232C soportan ser manejados por interrupciones de entrada/salida. Las comunicaciones de alta velocidad BP-LAN pueden fallar cuando a la tarjeta se le permite generar interrupciones. Usualmente es posible desabilitar esta facultad removiendo el jumper el cual selecciona una linea de solicitud de interrupción.

7.- Ha probado el puerto RS-232C con el programa BPTERM.COM de BP-LAN?

Conecte el cable entre el cliente y el servidor. Instale el BP-LAN BIOS tanto en el cliente como en el servidor ejecutando BPBIOS.COM. Configure el socket 0 tanto en el cliente como en el servidor con BPSERIAL.COM, por ejemplo:

BPSERIAL /$3F8 /3 /1

esto configura COM1 para 115.200 baudios.
Escriba BPTERM /0 para inicializar el programa de terminal tanto tanto en el cliente como en el servidor. Escriba un mensaje para cada computadora.

a) El mensaje no tiene eco en ninguna pantalla?
Esto puede ser un indicativo de que no funcionan los puertos seriales o una impropia configuración en el puerto. Ambos puertos deben estar seteados a la misma velocidad (baudios) para que la comunicación ocurra.

b) Los mensajes tienen eco pero aparecen montados.
Puede ser un indicativo de que se ha configurado incorrectamente la longitud de la palabra, la paridad o el bit de parada.

c) Están los mensajes teniendo eco en la misma pantalla mientras se son digitados?
Esto es un indicativo de que existe un corto entre las lineas de transmisión y recepción del cable. Algunas veces la señal que transmite le UART ilnduce una corriente en la linea de recepción .

d) La computadora transmite mensejes apropiadamente pero no los recibe de la misma manera.

La línea de recepción puede estar rota, o algún programa TSR puede estar interceptando los caracteres que llegan. Tanto el MOUSE.COM como MOUSE.SYS pueden ser la causa de éste problema.

e) La computadora recibe los mensajes apropiedamente pero no los transmite de la misma manera.
Se puede asumir que puede haber una rotura en la línea de transmisión. Chequear la línea con un puerto que se tenga la seguridad que reciba correctamente.

f) Falla el acceso a la red a pesar de que ambos puertos RS-232C funcionan adecuadamente con BPTERM.COM?

El BPTERM.COM transmite por caracteres, asi, la transmisión en red consiste en transmitir un bloque de caracteres en secuencia rápida, dando un muy pequeño espacio de tiempo para procesar cada caracter y recibir el siguiente. Reducir el baud rate en ambos puertos RS-232C incrementa el tiempo de procesamiento de cada caracter, pudiendo ésto ser la solución a nuestro problema.

CONCLUSIONES Y RECOMENDACIONES

El sistema BP-LAN tiene un gran valor didactico, pero en la practica es un sistema muy limitado comparado con los sistemas que se venden en el mercado. Es limitado principalmente en cuanto a niveles de seguridad se refiere. No cuenta con ningún sistema de asignasión de claves para los usuarios, lo que lo hace muy vulnerable en ese sentido.

El sistema tal como esta implementado, (con acceso de modo sector) nunca sera un sistema eficiente, ya que como se ha mencionado antes, no hay posibilidad de bloqueo de registros o archivos. Esto lo hace poco seguro para usarlo.

La velocidad de comunicacion de un puerto serial es mucho menor que la de los actuales sistemas de comunicacion y es mucho mas limitada, ya que no fue diseñada para ese proposito especifico.

El programa solo funcionará eficientemente en discos de capacidad máxima de 30Mb. Esto se debe principalmente a que fue diseñado para trabajar solo con FAT de 12 bits y no en FAT extendida de 16 bits. Esto hace que el programa presente grandes problemas al tratar de trabajar con versiones de D.O.S. mayores a la 3.30, ya que esas versiones ya incluyen FAT de 16 bits para poder soportar discos de gran capacidad.

Por el hecho de realizarse la comunicación a traves de cables multipares y al sencillo código de transferencia de datos, este es un sistema que no se presta para abarcar grandes distancias.

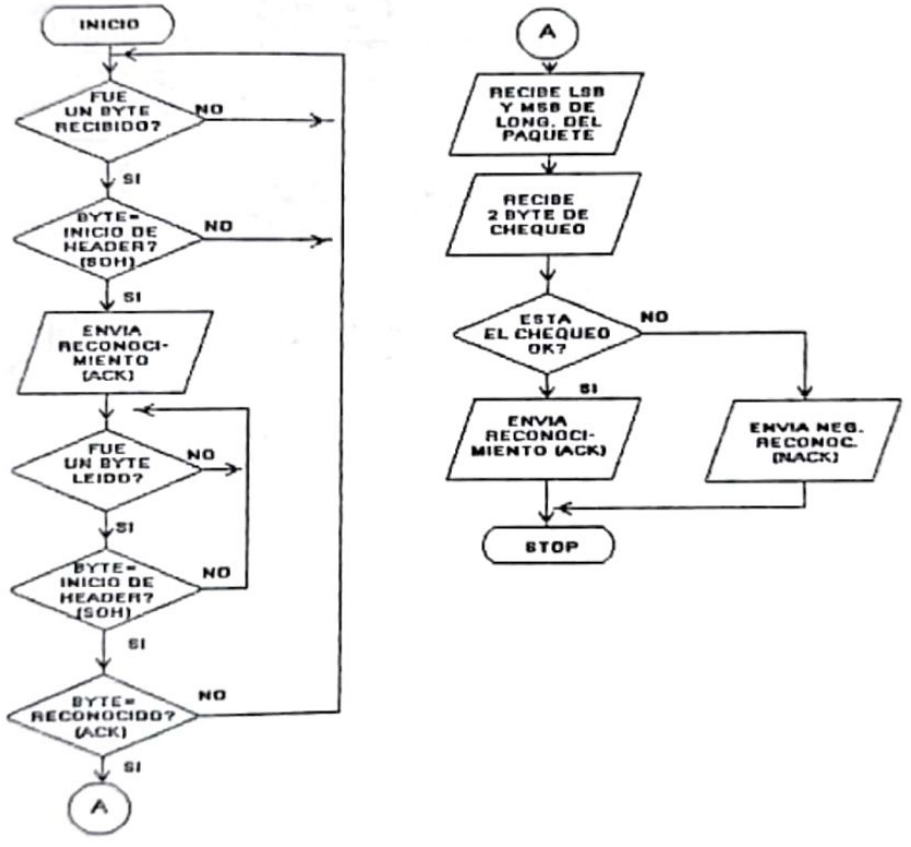
Se ha podido interpretar y comprender la idea de los programas residentes de memoria, sobre todo para interceptar los vectores de interrupción. El sistema de enrutamiento de trabajos de impresión se basa en este esquema y es muy similar al de otro tipo de redes comerciales. Lamentablemente

se tiene previsto la comparticion de equipos que no esten conectados al servidor.
Incluye un programa de correo electrónico que es muy completo, y el cual se lo podria adaptar para que funcione con otras marcas de redes, ya que es sencillo de interpretar y manejar.
La forma modular en que está diseñado este programa permite hacer futuras ampliaciones y modificaciones al mismo, para obtener un mejor rendimiento y mayor confiabilidad. Mas aún tomando en cuenta que se basa en el modelo OSI e implementa las dos capas inferiores para su funcionamiento. Dentro del mismo programa se ha tenido el cuidado de dejar listas las especificaciones e interrupciones para la implementacion futura de las siguientes capas superiores.
Incluye un programa de ejecucion remota para disminuir el trafico en la red. Este es muy útil, ya que para determinados procesos se recargaría de un trafico intenso e innecesario a la red.
Debido a la manera en que se implementan los programas residentes de memoria en este programa, es casi imposible poder cargar algún otro que no forme parte de la red. Esto es, ya que los programas no reservan partes de memoria ni la bloquean. Entonces al cargar otro programa residente, se podria cargar encima de alguno de los de la red, e interferir de esta manera con las funciones básicas de la red.
En definitiva podriamos decir que el programa nos ha ilustrado y enseñado los pasos necesarios para la implementación de sistemas operativos, programas residentes en memoria y los dispositivos instalables.
Se ha podido apreciar los distintos niveles de seguridad que se deben implementar para la correcte transmisión y recepción de datos. Definitivamente, en esto nos ayuda en gran mayoria

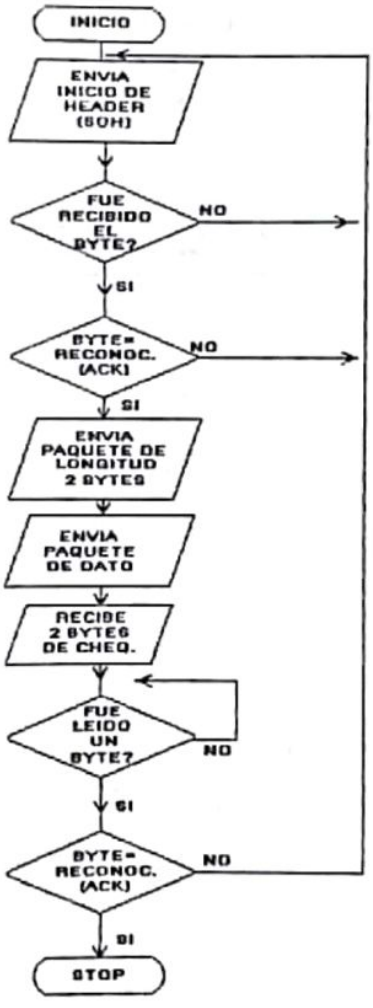
el uso de los paquetes para la transmisión de datos, ya que de antemano se puede establecer que tipo de dato se espera. Con los conocimientos que nos ha proporionado este trabajo, podriamos pensar en futuras modificaciones, una de las principales que deberiamos tener en cuenta es la de cambiar el modo de acceso al disco. Apesar de que esto complicaria notablemente el programa, no habria la limitación de trabajar con FAT de DOS, obteniendo de esta manera un programa que pueda trabajar en cualquier versión de sistema operativo y disco duro. Una vez logrado esto, podriamos reemplazar el sistema de semaforos por sistemas de bloqueo de registros y archivos.

ANEXO A

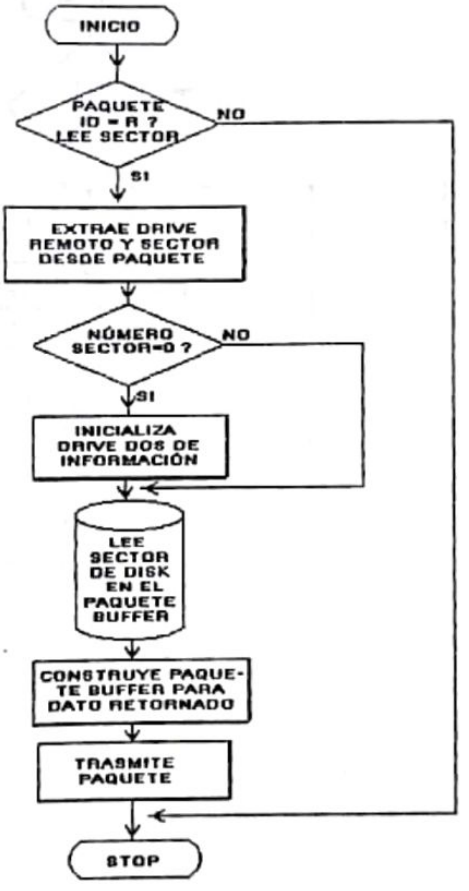
RECIBIR UN PAQUETE



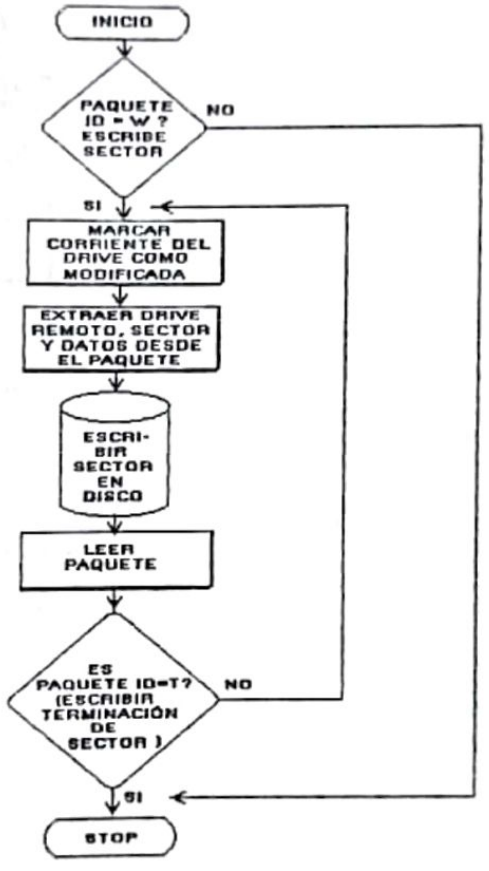
TRANSMITIR UN PAQUETE



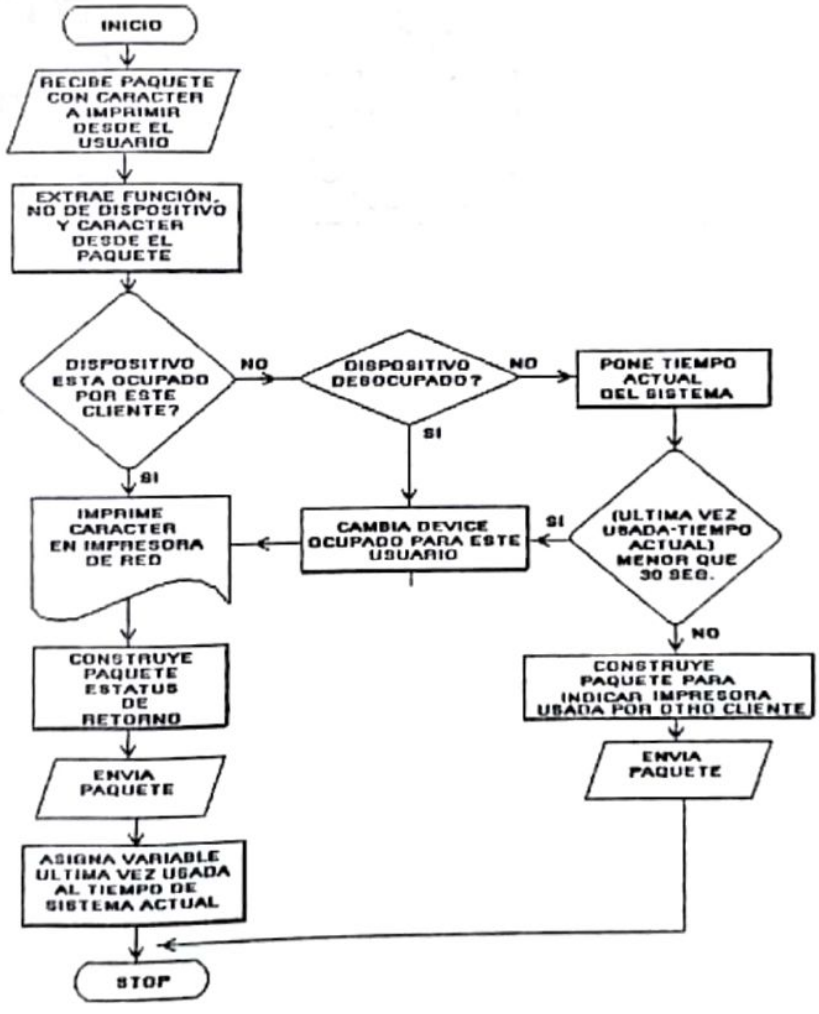
ENTRADA / SALIDA DE ARCHIVOS LEER SECTOR (LADO DEL SERVIDOR)



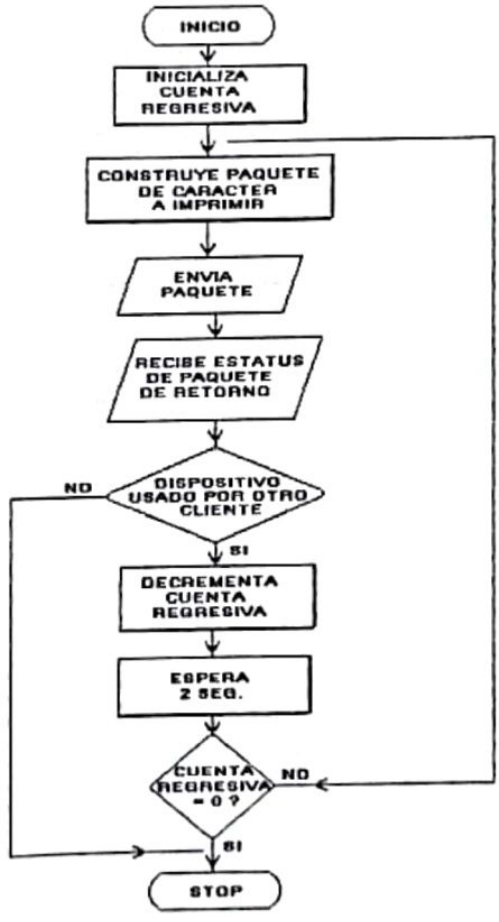
ENTRADA / SALIDA DE ARCHIVOS ESCRIBE SECTOR (LADO DEL SERVIDOR)



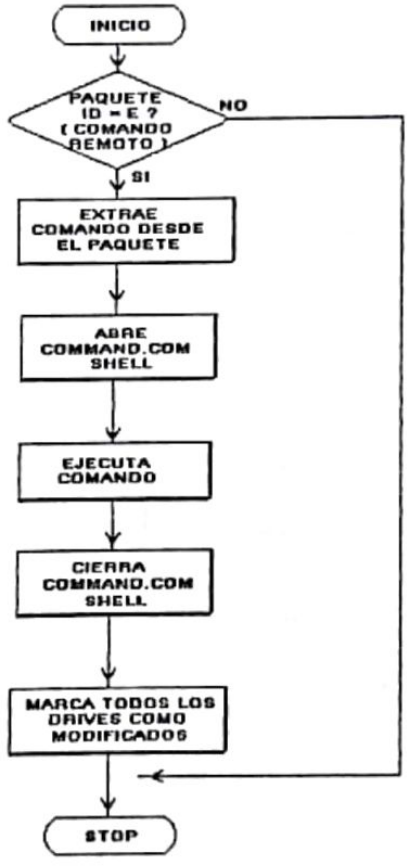
IMPRESION EN RED (LADO DEL SERVIDOR)



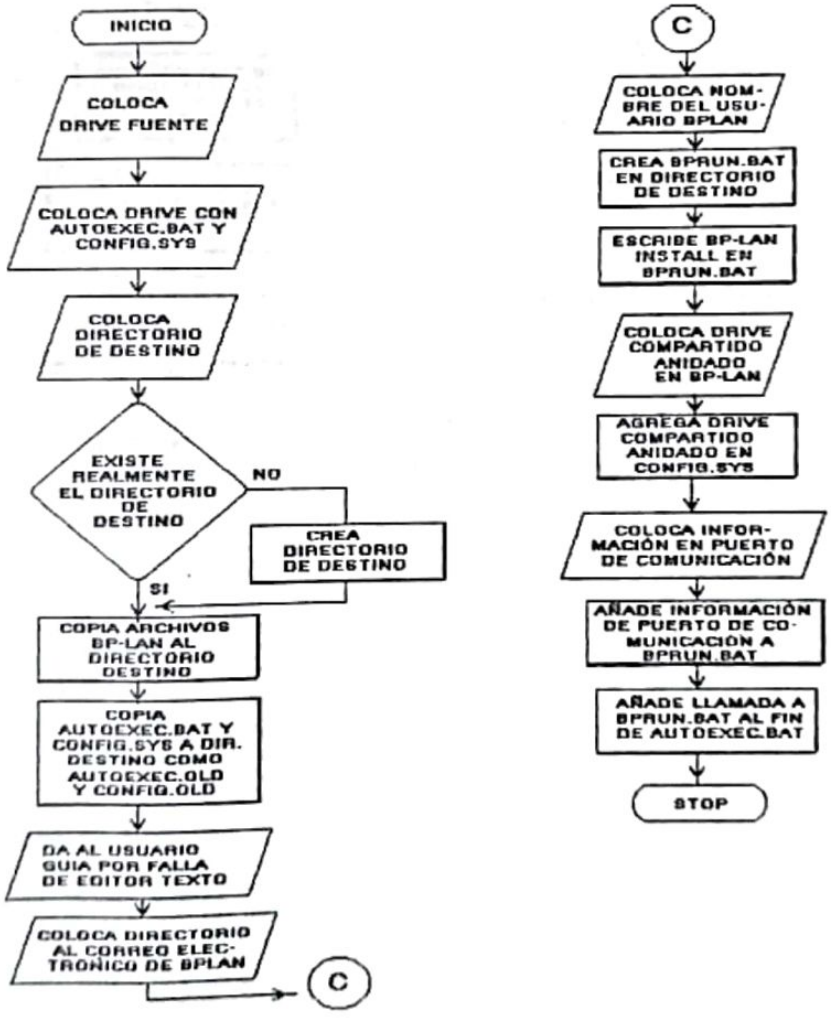
IMPRESION EN RED (LADO DEL USUARIO)



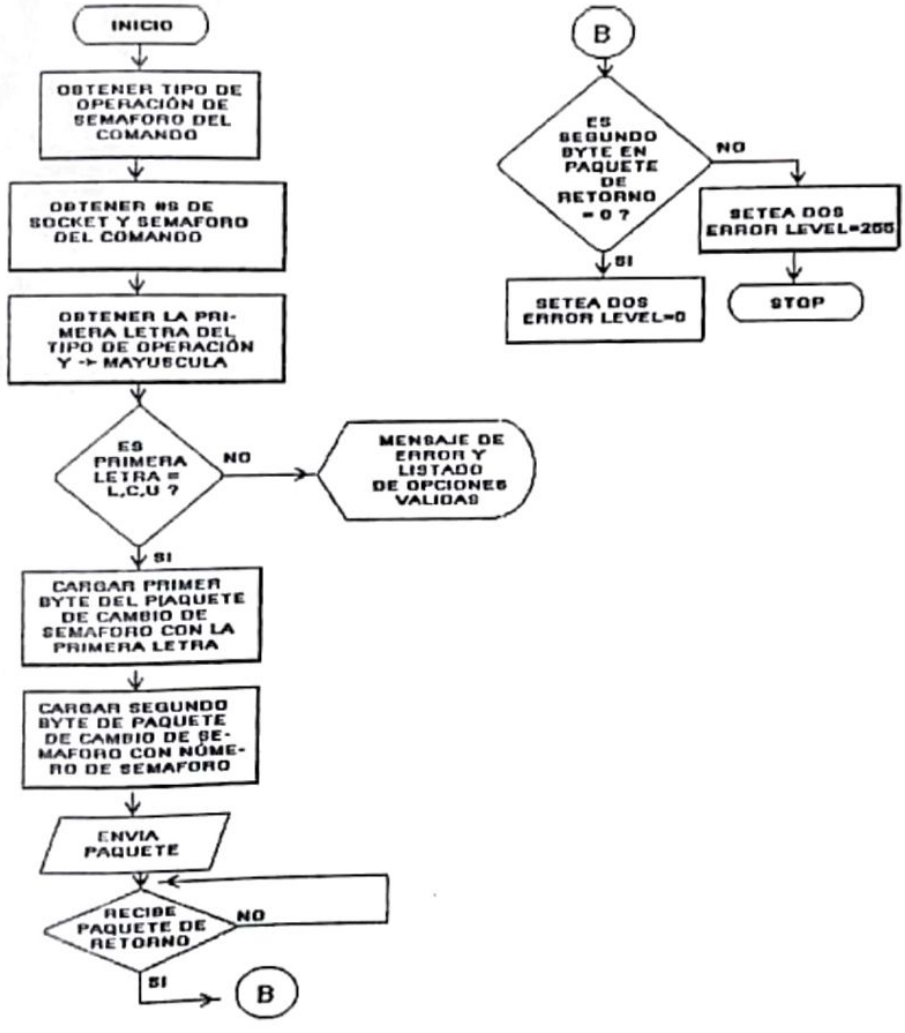
COMANDO DE EJECUCION REMOTA (LADO DEL SERVIDOR)



BPINST.EXE


INICIO
COLOCA DRIVE FUENTE
COLOCA DRIVE CON AUTOEXEC.BAT Y CONFIG.SYS
COLOCA DIRECTORIO DE DESTINO
EXISTE REALMENTE EL DIRECTORIO DE DESTINO
NO
CREA DIRECTORIO DE DESTINO
SI
COPIA ARCHIVOS BP-LAN AL DIRECTORIO DESTINO
COPIA AUTOEXEC.BAT Y CONFIG.SYS A DIR. DESTINO COMO AUTOEXEC.OLD Y CONFIG.OLD
DA AL USUARIO GUIA POR FALLA DE EDITOR TEXTO
COLOCA DIRECTORIO AL CORREO ELECTRONICO DE BPLAN
C
C
COLOCA NOMBRE DEL USUARIO BPLAN
CREA BPRUN.BAT EN DIRECTORIO DE DESTINO
ESCRIBE BP-LAN INSTALL EN BPRUN.BAT
COLOCA DRIVE COMPARTIDO ANIDADO EN BP-LAN
AGREGA DRIVE COMPARTIDO ANIDADO EN CONFIG.SYS
COLOCA INFORMACIÓN EN PUERTO DE COMUNICACIÓN
AÑADE INFORMACIÓN DE PUERTO DE COMUNICACIÓN A BPRUN.BAT
AÑADE LLAMADA A BPRUN.BAT AL FIN DE AUTOEXEC.BAT
STOP

BPSEMAPH.EXE



ANEXO B

LISTADO DEL PROGRAMA

```
;***********************************************
;* Mensaje de eco a socket de BP-LAN           *
;* FUNCION: mensaje de eco a un socket definido*
:* Formato del Comando: BPECHO /socket /message*
;*         BPECHO.ASM, BPECHO.COM              *
***********************************************

codeseg segment byte
        assume  cs:codeseg,ds:codeseg,es:codeseg
        org     100h
start:  jmp     echo
;
socket_num      db      0
        include bpbioshd.mod
        include misc.mod
echo    proc    near                    ;Salir si no hay parametros
mov     al,cs:[80h]
        or      al,al
        jnz     echo1
        jmp     echo3
echo1:  mov     si,81h
        call    get_opt
        jb      echo3
        mov     socket_num,cl
        call    wslash
echo2   mov     al,[si]
        bpbios  phys_int,trasmit_byte,socket_num
        inc     si
        cmp     al,13
        jnz     echo2
echo3   int     20h
echo    endp
codeseg ends
        end     star


;*******************************************
;* BP LAN BIOS Header  (BPBIOSHD.MOD)      *
;*                                         *
;* Define todas las constantes BP-LAN BIOS *
;* Debe ser incluido en todos los programas *
;* de los sistemas BP-LAN para mantener    *
;*             customizacion               *
;*******************************************
;
;*** Constantes ***
;
soh     =       01h     ;inicio del encabezonamiento del byte
ack     =       06h     ;reconocimiento de  byte
nak     =       15h     ;no reconocimiento de byte
can     =       18h     ;cancelar byte
;
maxsockets  equ 10h     ;espacio de localizacion para 16 sockets de red
;
ctrl_int    equ 80h     ;BIOS del Control de Interrupciones
```

```
        phys_int    equ 81h     ;interrupcion de I/O fisicas
        link_int    equ 82h     ;interrupcion de enlace de datos
        netw_int    equ 83h     ;interrupcion de Red
        tran_int    equ 84h     ;interrupcion de transporte de datos punto a
punto interrupt
        sess_int    equ 85h     ;control de interrupcion de sesion
        pres_int    equ 86h     ;interrupcion de presentacion
        appl_int    equ 87h     ;interrupcion de aplicacion

        ;*** Subfunciones para control de interrupciones del BIOS***
        get_nodes       equ 0
        set_nodes       equ 1
        attach_printer  equ 2
        detach_printer  equ 3
        install_port    equ 4
        next_port       equ 5
        install_appl    equ 6
        get_info        equ 7

        ;*** Subfunciones para interrupciones Fisicas ***
        ;recieve_status  equ 0
        ;recieve_byte    equ 1
        ;transmit_status equ 2
        ;transmit_byte   equ 3

        ;*** Subfunciones para interrupcion de capa de enlace ***
        clear_checksum  equ 0
        calc_checksum   equ 1
        get_checksum    equ 2
        transmit_packet equ 3
        recieve_packet  equ 4
        get_retries     equ 5

        ;*** subfunciones para interrupcion de capa de Red ***
        ; reservado para customizacion

        ;*** subfunciones para interrupcion de capa de transporte **
        ; reservado para customizacion

        ;*** subfunciones para interrupcion de capa de sesion ***
        lock_printer    equ     0
        unlock_printer  equ     1
        set_semaphore   equ     2
        get_semaphore   equ     3
        clear_semaphore equ     4

        ;*** subfunciones para interrupciones de capa de Presentacion y
aplicacion ***
        ; reservadas para customizacion

        ;*** Macros parar accesar a funciones de BPBIOS ***
        bpbios macro intnum,funcnum,portnum,count,buffer_addr,overflow
                mov     ah,funcnum
                IFNB    <portnum>
                mov     bl,portnum
                ENDIF
                IFNB    <count>
                mov     cx,count
                ENDIF
                IFNB    <buffer_addr>
                mov     dx,buffer_addr overflow ;
```

```
        ENDIF
        int     intnum
        endm

get_packet macro portnum,packet_length,packet_address,more
        mov     ah,recieve_packet
        IFNB    <portnum>
        mov     bl,portnum
        ENDIF
        IFNB    <packet_addr>
        mov     dx,packet_address more

put_packet macro portnum, packet_length, packet_address, more
        mov     ah, transmit_packet
        IFNB    <portnum>
        mov     bl,portnum
        ENDIF
        IFNB    <packet_length>
        mov     cx,packet_length
        ENDIF
        IFNB    <packet_address>
        mov     dx,packet_address more
        ENDIF
        int     link_int
        endm


;************************************************
:* Utiliitario de instalacion BP-lan modem remoto *
;* funcion: INstala impresora de red remota      *
;* Formato del Comando :                         *
;* BPCOMDEF/socket_num/local_modem/remote_modem  *
;************************************************

;
codeseg segment
        assume  cs:codeseg,ds:codeseg,es:codeseg
        org     100h

start:  jmp     intinst
        include bpbioshd.mod

;*** variables ***
;
old_int14h_vector       label dword
old_int14h_offs         dw      ?
old_int14h_seg  dw      ?
socket_num              db      0
local_printer   db      0
remote_printer  db      0
packet_buffer   db      6  dup (0)

;*****************************************
;*** Manejador de Nuevas interrupciones ***
;*****************************************
new_int14h      proc    far
        cmp     dl,cs:local_printer
        jz      new_il
        jmp     cs:old_int14h_vector
new_il: pushf
        push    bx
```

```
                push    cx
                push    dx
                push    si
                push    di
                push    ds
                push    es

                push    cs              ;Copy CS into DS
                pop     ds
                mov     packet_buffer,'S'
                mov     packet_buffer+1,ah
                mov     ah,remote_printer
                mov     packet_buffer+2,ah
                mov     packet_buffer+3,al
                put_packet      socket_num,4,offset packet_buffer
                get_packet      socket_num,cx,offset packet_buffer
                mov     ah,packet_buffer
                mov     al,packet_buffer+1
        new_pl:         pop     es
                        pop     ds
                        pop     di
                        pop     si
                        pop     dx
                        pop     cx
                        pop     bx
                        popf
                        iret
        new_int14h      endp

        ;*************************************************
        :*** Instala manejador de nuevas interrupciones ***
        ;*************************************************
        intinst proc            near
                mov             al,cs:[80h]
                or              al,al
                jz              default
                mov             si,81h
                call            get_opt ;obtiene socket_num
                jb              default
                mov             socket,num,cl
                call            get_opt ;obtiene numero de impresora local
                jb              default
                mov             remote_printer,cl
        default: mov            ah,35h  ; obtiene funcion de vector de
interrupcion
                mov             al,14h
                int             21h
                mov             old_int14h_offs,bx ; salva interrupcion
anterior
                mov             old_int14h_seg,es       ;   direccion
                mov             ah,25h  ;setea funcion de ventor de
interrupcion
                mov             al,14h
                mov             dx,offset new_int14h ; apunta a nueva rutina
                int             21h
                lea             dx,intinst
                int             27h
        intinst endp
                include misc.mod

        codeseg ends
```

```
        end     start

;*****************************************************************
;* Fin de modulo de instalacion de manejador de interrupcion*
;*****************************************************************


program bp_drive_info:
(*********************************
***   BPDRIVES.PAS,BPDRIVES.EXE   ***
*********************************
uses dos;
{$I BPPASCAL.INC}
var
        ofs,seg:word;
        attributes,s1,01:word;
        client_drive,new_server_drive,new_access,new_socket:byte;
        s:string;

begin
        if paramcount>0 then
        begin
                s:=paramstr(1);
                client_drive:=ord(upcase(s[2]))-65;
        end;
        if paramcount>1 then
        begin
                s:=paramstr(2);
                new_server_drive:=ord(upcase(s[2]))-65;
        end;
        if paramcount>2 then
        begin
                s:=paramstr(3);
                if (s='/RW') or (s='/rw') then
                        new_access:=0
                else
                        new_access:=1
        end;
        if paramcount>3 then
                new_socket:=get_opt(4);
        writeln('BP-LAN Network Drive Information:');
        writeln('');
      writeln('Client Drive',chr(9),'Server Drive',chr(9),'Socket',
               chr(9),'Access');
      writeln('------------',chr(9),'------------',chr(9),'------',
               chr(9),'------');
      regs.ah:=$52;
      intr($21,regs);
         ofs:=regs.bx+$22;
         seg:=regs.es;
         while memw[seg:ofs]<>$ffff do
      begin
         attributes:=memw[seg:ofs+4];
                if ((attributes and $8000)<>$8000) then
                        if memw[seg:ofs+11]=$5042 then
                        begin
                                (*** ofs+$d = BP-LAN Version ***)
                                (*** ofs+$e = Socket Number  ***)
                                (*** ofs+$f = Client Drive#  ***)
                                (*** ofs+$10= Server Drive#  ***)
                                (*** ofs+$11= Access         ***)
```

```
                        if paramcount>0 then
                            if mem[seg:ofs+$f]=client_drive then
                                                    if paramcount>1 then
                                                    begin
                    mem[seg:ofs+$10]:=new_server_drive;
                                                    if paramcount>2 then
mem[seg:ofs+$11]:=new_access;
                                                if paramcount>3 then
                    mem[seg:ofs+$e]:=new_socket
                                                        end;
                                                regs.ah:=$36;
                                                regs.dl:=client_drive;
                                                msdos(regs);
                                    write(char(65+mem[seg:ofs+$f]),chr(9),chr(9),
                                    chr(65+mem[seg:ofs+$10]),chr(9),chr(9),
                                            mem[seg:ofs+$e],chr(9);
                                            if mem[seg:ofs+$11]=0 then
                                                    writeln(`Read/ Write`)
                                            else
                                                    writeln(`Read Only`);
                    end;
                    01:=memw[seg:ofs];s1:=memw[seg:ofs+2];ofs:=01;seg:=s1;
            end;
    end.



    ;*****************************************************
    ;*      M O D U L O   D E   A P L I C A C I O N      *
    ;*                    APPLICAT.MOD                   *
    ;*****************************************************

    ;*** VARIABLES***

    application_interrupt_table     dd 256 dup (0)

    application_interrupt   proc    far
            pushall
            xor     bx,bx
            mov     bl,al
            add     bx,bx
            add     bx,bx
    ;ignora requerimiento de interrupcion si no esta definida en la tabla
            cmp word ptr cs:application_interrupt_table[bx],0
            jnz     appl_1
            cmp word ptr cs:application_interrupt_table[bx+2],0
            jnz     appl_1
            jmp     appl_2
    appl_1: call    cs:application_interrupt_table[bx]
    appl_2: popall
            retf
    application_interrupt    endp


    ;*****************************************************
    ;*      BPBIOS.ASM                                   *
    ;*****************************************************
            include bpbioshd.mod
```

```
code    segment
        assume  cs:code,ds:code,es:code;
        org     100h

start:  jmp     install

maxnode db      0

        include misc.mod
        include physical.mod
        include datalink.mod
        include applicat.mod
        include console.mod

control_interrupt       proc    far
        push    bx
        cmp     ah,get_nodes
        jnz     cont_1
        mov     al,cs:maxnode
        jmp     cont_5
cont_1: cmp     ah,set_nodes
        jnz     cont_2
        mov     cs:maxnode,al
        jmp     cont_5
cont_2: cmp     ah,install_port
        jnz     cont_3
        mov     bl,cs:maxnode
        xor     bh,bh   ;forma el offset para vector de interrupciones
fisicas
        push    ax
        add     bx,bx
        add     bx,bx
        add     bx,bx
        add     bx,bx
        add     al,al
        add     al,al
        xor     ah,ah
        add     bx,bx
        pop     ax
        mov  word ptr  cs:physical_interrupt_table[bx],dx  ;instala
vector
        mov  word ptr  cs:physical_interrupt_table[bx+2],ds
        jmp     cont_5
cont_3: cmp     ahl,next_port
        jnz     cont_4
        inc     cs:maxnode
        jmp     cont_5
cont_4: cmp     ah,get_info
        jnz     cont_5
        mov     ax,word ptr  cs:physical_interrupt_table[bx]+2
cont_5: pop     bx
        iret
control_interrupt       endp


install proc    near
        mov     ah,25h
        mov     al,ctrl_int
        mov     dx,offset control_interrupt
        int     21h     ;instala control interrupciones bpbios
        mov     ah,25h
```

```
                mov     al,phys_int
                mov     dx,offset physical_interrupt
                int     21h     ;instala interrupcion de capa fisica
                mov     ah,25h
                mov     al,link_int
                mov     dx,offset_datalink_interrupt
                int     21h     ;instala interrupcion de capa de enlace
                mov     ah,25h
                mov     al,appl_int
                mov     dx,offset application_interrupt
                int     21h     ;instala interrupcion de capa de aplicacion


                mov     dx,offset install+1


                int     27h
        install endp

        code    ends
                end     start

        ;****************************************
        ;*   FIN   DE  ARCHIVO   BPBIOS.ASM     *
        ;****************************************


;*************************************************************
;* Funcion: Instala impresora remota                         *
;*                                                           *
;* Formato de comando:                                       *
;* BPLPTDEF /socket_num/local_printer/remote_printer         *
;*          /time_out                                        *
;*************************************************************
;
codeseg segment
                        assume  cs:codeseg,ds:codeseg,es:codeseg
                        org 100h

start:  jmp                 intinst
                        include bpbioshd.mod
                        include misc.mod

;*****  V A R I A B L E S *******
;
old_int17h_vector     label   dword
old_int17h_offs     dw      ?
old_int17h_seg      dw      ?
old_int21h_vector     label   dword
old_int21h_offs     dw      ?
old_int21h_seg      dw      ?
socket_num          db      0
local_printer       db      0
remote_printer      db      0
packet_buffer       db      6 dup(0)
time_out            db      40    ; 40 seg para reintentar si esta ocupada
printer
retry_count         db      ?

;****************************************
```

```
;* Manejador de nuevas interrupciones *
;*************************************
new_int17h      proc    far
        cmp  dl,cs:local_printer
        jz   new_1
        jmp  cs:old_int17h_vector
new_1:   push bx
        push cx
        push dx
        push sx
        push di
        push ds
        push es
        push ax
        push cs    ; COPIA CS A DS
        pop  ds
        mov  cl,time_out
        shr  cl,1
        mov  retry_cout,cl  ; reintenta count=time_out /2
        mov  cx,4
        cmp  ah,0
        jz   new_2
        dec  cx
new_2:  mov  packet_buffer,'P'
                        mov  packet_buffer+1,ah
                        mov  ah,remote_printer
                        mov  packet_buffer+2,ah
                        mov  packet_buffer+3,al
                        put_packet   socket_num,cx,offset packet_buffer
                        get_packet   socket_num,cx,offset packet_buffer
                        cmp  packet_buffer,255
                        jnz  new_3
                        call wait_a_sec
                        call wait_a_sec
                        dec  retry_count
                        jnz  new_2
                new_3:   pop  ax
                        mov  ah,packet_buffer
                        pop  es
                        pop  ds
                        pop  di
                        pop  si
                        pop  dx
                        pop  cx
                        pop  bx
                        iret
                new_int17h    endp

                new_int21h    proc   far
                        pushf
                        cmp  ah,40h   ;escribiendo al dispositivo
                        jz   new_i2
                        popf
                        jmp  cs:old_int21h_vector
                new_i2:  cmp  bx,4    ; manejo de impresora??
                        jz   new_i3
                        popf
                        jmp  cs:old_int21h_vector
                new_i3:  cmp  cs:remote_printer,0  ;lpt1?
                        jz   new_i4
                        popf
```

```
                jmp  cs:old_int21h_vector
new_i4:  cld
                push cx
                push dx
                push si
                xor  dx,dx
                jcxz new_i6
new_i5:  lodsb
                mov  ah,0
                int  17h
                loop new_i5
new_i6:  pop  si
                pop  dx
                pop  cx
                mov  ax,cx
                popf
                clc
                sti
                ret  2
new_int21h    endp

;***********************************************
;* Instala manejador de nuevas interrupciones *
;***********************************************
intinst  proc  near
                        mov   al,cs:[80h]
                        or    al,al
                        jz    default
                        mov   si,81h
                        call  gat_opt ; obtiene el socket_num
                        jb    default
                        mov   socket_num,cl
                        call  get_opt : obtiene el numero de impresora local
                        jb    default
                        mov   local_printer, cl
                        call  get_opt ;obtiene el numero de get remote
printer number
                        jb    default
                        mov   remote_printer,cl
                        call  get_opt ;obtiene el time out en segundos
                        jb    default
                        mov   time_out,cl
                default: mov   ah,35h  ;obtiene funcion de vector de
interrupcion
                        mov   al,17h
                        int   21h
                        mov   old_int17h_offs,bx  ;salva interrupt anterior
                        mov   old_int17h_seg,es   ;direccion
                        mov   ah,25h              ;setea funcion de vector
de interrupcion
                        mov   al,21h
                        mov   dx,offset new_int21h ;apunta a nueva rutina
                        int   21h
                        lea   dx,intinst
                        int   27h
                intinst  endp

                codeseg  ends
                        end   start
```

```
;*****************************************************************************
;* Blueprint LAN Capa de Enlace de Datos                          (DATALINK.MOD)*
;*****************************************************************************
;
;*** VARIABLES ***
;
temp            dw      ?
checksum        dw      0
packet_length           dw      0
retries db      0
socket_error db         0
inta01 db       0
intb01 db       0

datalink_table          label  word
                org             datalink_table+2*clear_checksum
                dw              datalink1
                org             datalink_table+2*calc_checksum
                dw              datalink2
                org             datalink_table+2*get_checksum
                dw              datalink3
                org             datalink_table+2*transmit_packet
                dw              datalink4
                org             datalink_table+2*receive_packet
                dw              datalink5
                org             datalink_table+2*get_retries
                dw              datalink6
                org             datalink_table+16

datalink_interrupt              proc            far
                push            bx
                xor             bx,bx
                mov             bl,ah
                add             bx,bx
                jmp             cs:[datalink_table+bx]

datalink1:              pop             bx              ;limpia el checksum
                mov             cs:checksum,0
                iret

datalink2:              pop             bx              ;a¤ade AL al checksum
                xor             ah,ah
                add             cs:checksum,ax
                iret

datalink3:              pop             bx              ;coge el checksum dentro de AX
                mov             ax,cs:checksum
                iret

datalink4:              pop             bx              ;transmite paquete hacia DS:SI
                call            out_packet
                iret

datalink5:              pop             bx              ;recibe paquete dentro de DS:SI
                call            in_packet
                mov             cx,cs:packet_length
                iret

datalink6:              pop             bx              ;AL= paquete retransmitido
```

```
                mov     al,cs:retries
                iret
datalink_interrupt      endp

;*** Bloque de Drivers de Entrada/Salida ***
;
out_packet      proc    near            ;*** transmite paquete via Socket actual ***
                push    ax
                in      al,21h

                mov     cs:inta01,al
                in      al,0a1h
                mov     cs:intb01,al
                mov     al,0ffh
                out     21h,al
                out     0a1h,al
                pop     ax
                mov     cs:retries,-1
trans1:         inc     cs:retries
                call    put_header
                push    di
                push    cx
                bpbios  link_int,clear_checksum
                mov     al,cl
                bpbios  phys_int,transmit_byte
                mov     al,ch
                bpbios  phys_int,transmit_byte
                mov     di,dx
trans2:         mov     al,[di]
                inc     di
                bpbios  phys_int,transmit_byte
                loop    trans2
                pop     cx
                pop     di
                call    send_checksum
                cmp     cs:socket_error,0
                jz      trans3
                jmp     trans4
trans3:         cmp     al,ack
                jnz     trans1
trans4:         push    ax
                mov     al,cs:inta01
                out     21h,al
                mov     al,cs:intb01
                out     0a1h,al
                pop     ax
                xor     ah,ah
                ret
out_packet      endp


in_packet       proc    near            ;*** Recibe Paquete ***
                in      al,21h
                mov     cs:inta01,al
                in      al,0a1h
                mov     cs:int01,al
                mov     al,0ffh
                out     21h,al
                out     0a1h,al
                call    get_header
```

```
                push        di
                push        cx
                bpbios link_int,clear_checksum
                bpbios phys_int,receive_byte
                mov         cl,al
                bpbios ch,al
                mov         cs:packet_length,cx
                mov         di,dx
receil:         bpbios phys_intreceive_byte
                mov         [di],al
                inc         di
                loop        receil
                pop         cx
                pop         di
                call        verify_checksum
                cmp         cs:socket_error,0
                jz          recei2
                jmp         recei3
recei2:         cmp         al,ack
                jnz         in_packet
recei3:         push        ax
                mov         al,cs:int101
                out         21h,al
                mov         al,cs:intb01
                out         0a1h,al
                pop         ax
                xor         ah,ah
                ret
in_packet       endp

put_header      proc        near        ;*** Envia el Encabezado del
Paquete a Transmitir ***
      push      ax
put_h1:         mov         al,soh
                bpbios phys_int,transmit_status
                jz          put_h2
                bpbios phys_int,transmit_byte
put_h2:         bpbios phys_int,receive_status
                jz          put_h1
                bpbios phys_int,receive_byte
                cmp         al,ack
                jnz         put_h1
                call        acknowledge
                pop         ax
                cli
                ret
put_header      endp

get_header      proc        near   ;*** Aguarda por el Encabezado del
paquete a transmitir ***
                push        ax
get_h1:         bpbios phys_int,receive_byte
                cmp         al,soh      ;Verifica si el caracter es SOH
                jnz         get_h1
                call        acknowledge  ;si es asi, reconoce SOH
get_h2:         bpbios phys_int,receive_byte     ;consume todos los duplicados SOHs
                cmp         al,soh
                jz          get_h2
                cmp         al,ack      ;es SOH seguido por ACK?
                jnz         get_h1      ;no, entonces aguardar por retransmisi¢n
                pop         ax
```

```
                cli
                ret
get_header      endp

verify_checksum     proc    near   ;*** Verifica el Checksum del Paquete ***
                push        cx
                bpbios link_int,get_checksum
                mov         cs:temp,ax
                bpbios phys_int,receive_byte
                mov         cl,al
                bpbios phys_int,transmit_byte
                bpbios phys,receive_byte
                mov         ch,al
                cmp         cx,cs:temp
                pop         cx
                sti
                jz          verif1
                jmp         neg_ack
verif1:         jmp         acknowledge
verify_checksum     endp

send_checksum               proc    near ;*** transmite el Checksum del Paquete
***
                push        cx
                bpbios link_int,get_checksum
                mov         cx,ax
                mov         al,cl
                bpbios phys_int,transmit_byte
                bpbios phys_int,receive_byte
                mov         al,ch
                bpbios phys_int,transmit_byte
                bpbios phys_int,receive_byte
                pop         cx
                sti
                ret
send_checksum               endp

acknowledge  proc           near   ;*** Reconocimiento de la Recepci¢n del Paquete
***
                mov         al,ack
                bpbios phys_int,transmit_byte
                ret
acknowledge         endp

neg_ack      proc     near       ;*** Reconocimiento Negativo, Paquete da¤ado
***
                mov         al,nak
                bpbios phys_int,transmit_byte
                ret
neg_ack      endp

;*****************************************************************************
;* Fin de DATALINK.MOD                                                       *
;*****************************************************************************


;*****************************************************************************
;* Modulo del Servidor de Archivos como centro de actividad de la Red        *
;*                                                          (FILESERV.MOD)*
;*                                                                           *
;* Funci¢n : Procesos de Solicitudes de Conexi¢n, lee/escribe en Sector.     *
;* Requerimientos : Ninguno                                                  *
```

```
;*****************************************************************************
;
;*** Variables ***
;
startsect               dw              ?
function                db              ?
drive           db              ?
drive_table     db              maxsockets*32 dup (-1)  ;Todos los drives
necesitan actualizaci¢n
seclen_table    dw              26 dup (512)        ;Longitud del sector de
drives A-Z
fileser proc    near            ;*** Lee desde o escribe a un Sector conectado
***
            mov             al,packet_buffer
            cmp             al,'R'
            jnz             files1
            jmp             reasec
files1: cmp             al,'w'
            jnz             files2
            jmp             wrisec
files2: cmp             al,'M'
            jnz             files3
            jmp             ckmedia
files3: ret
fileser endp

reasec  proc            near                    ;*** Lee desde Sector sobre el
Drive conectado
            call            charout
            mov             packet_buffer,'D'       ;descriptor del paquete de
datos
            mov             al,packet_buffer[1]
            mov             drive,al                        ;Obtiene drive
            mov             cl,packet_buffer[2]     ;Obtiene el LSB absoluto
del sector
            mov             ch,packet_buffer[3]     ;Obtiene el MSB absoluto
del sector
            mov             startsect,cx
            or              cx,0
            jnz             rloop1
            mov             ah,36h
            mov             dl,drive
            inc             dl
            int             21h
            push            bx
            xor             bh,bh
            mov             bl,drive
            add             bx,bx
            mov             seclen_table[bx],cx
            pop             bx
rloop1: mov             function,0              ;funci¢n=leer sector
            call            sector_io
            push            bx
            push    cx
            xor             bh,bh
            mov             bl,drive
            add             bx,bx
            mov             cx,seclen_table[bx]
            add             cx,4
            mov             word ptr packet_length,cx
            pop             cx
```

```
            pop             bx
           put_packet              socket_num,packet_length,offset
packet_buffer

            ret
reasec  endp

wrisec  proc            near            ;*** Escribe en el Sector sobre el
Drive conectado ***
            call            change_flag
wrise0: call            charout
            mov             packet_buffer,'D'
            mov             al,packet_buffer[1]
            mov             drive,al                                ;Obtiene
el drive
            mov             cl,packet_buffer[2]             ;Obtiene el LSB
absoluto del sector
            mov             ch,packet_buffer[3]             ;OBtiene el MSB
absoluto del sector
            mov             startsect,cx
wrise1: mov             function,1
            lea             bx,packet_buffer[4]
            call            sector_io
                  get_packet
socket_num,packet_length,offset packet_buffer
            mov             al,packet_buffer
            cmp             al,'T'
            jnz             wrise0
            ret
            wrisec  endp

 sector_io              proc            near            ;*** Proceso de un
Sector Sobre el Drive conectado ***
            pushall
            mov             ax,cs
            mov     es,ax
            mov             ds,ax
            mov             al,' '
            call            charout
            mov             al,drive
            add             al,'A'
            call            charout
            mov             ax,startsect
            call            decout
            mov             al,13
            call            charout
            mov             al,drive
            mov             cx,1                            ;manipula 1 sector
            mov             dx,startsect    ;Obtiene el sector de inicio
            mov             bx,Packet_buffer[4]
            lea             ah,function
            mov             ah,ah
            or              rsect
            jz              26h
            int             secto2
            jnc             secto1
            jmp             25h
rsect:  int             secto2
            jnc             packet_buffer,'E'
secto1: mov             word ptr packet_buffer[2],ax
            mov
secto2: popf
```

```
            popall
            ret
sector_io               endp

get_drive               proc            near            ;SI=Balance del drive
modoficando Bandera
            push            ax
            push            bx
            push            cx
            mov             cl,5
            xor             bh,bh
            shl             bx,cl
            xor             ah,ah
            add             bx,ax
            lea             si,drive_table
            add             si,bx
            pop             cx
            pop             bx
            pop             ax
            ret
get_drive               endp

change_flag             proc            near            ;marca el drive como
modificado para  todos los sockets
            push            ax
            push            cx
            push            si
            mov             cx,maxsockets
            lea             si,drive_table
            xor             ah,ah
            mov             al,drive
            add             si,ax
chang1: mov             byte ptr [si],-1                ;marca el drive como
modificado
            add             si,32                           ;pr¢ximo socket
            loop            chang1
            pop             si
            pop             cx
            pop             ax
            ret
            change_flag             endp

ckmedia proc            near
            call            charout
            push            si
            call            get_drive
            mov             al,[si]
            mov             byte ptr [si],1
            pop             si
            mov             packet_buffer[2],al
            mov             word ptr packet_length,4
                  put_packet                socket_num,packet_length,offset
packet_buffer
            ret
ckmedia         endp

;*****************************************************************************
;* Fin del M¢dulo del Servidor de Archivos de la Red                         *
;*****************************************************************************
```

```
;*****************************************************************************
;* M¢dulo de Ejecuci¢n  de Funciones de de BAjo Nivel          (LOWLEVEL.MOD)*
;* Funci¢n: Ejecuta Funciones de Bajo Nivel en PC Remotas                     *
;*****************************************************************************
;

lowlev  proc            near
            cmp             packet_buffer,'r'
            jnz             lowle1
            jmp             peekmem
lowle1: cmp             packet_buffer,'w'
            jnz             lowle2
            jmp             pokemem
lowle2: cmp             packet_buffer,'i'
            jnz             lowle3
            jmp             portin
lowle3: cmp             packet_buffer,'o'
            jnz             lowle4
            jmp             portout
lowle4: cmp             packet_buffer,'e'
            jnz             lowle5
            jmp             execute
lowle5: cmp             packet_buffer,'f'
            jnz             lowle6
            jmp             fillmem
lowle6: cmp             packet_buffer,'m'
            jnz             lowlev7
            jmp             movemem
lowlev7: ret
lowlev          endp

peekmem         proc            near
            pushall
            mov             si,word ptr cs:packet_buffer+1
            mov             es,word ptr cs:packet_buffer+3
            mov             cx,word ptr cs:packet_buffer+5
            mov             di,offset  cs:packet_buffer+1
            mov             bx,cx
            inc             bx
            mov             cs:packet_length,bx
            mov             byte ptr cs:packet_buffer,'D'
            call            essitodi
                  put_packet      cs:socket_num,cs:packet_length,offset packet
buffer

            popall
            ret
peekmem         endp

pokemem         proc            near
            pushall
            mov             di,word ptr cs:packet_buffer+1
            mov             es,word ptr cs:packet_buffer+3
            mov             cx,word ptr cs:packet_buffer+5
            mov             si,offset cs:packet_buffer+7
            mov             bx,cx
            inc             bx
            mov             cs:packet_length,bx
            call            sitoesdi
            popall
```

```
                ret
pokemem         endp

portin          proc            near
                pushall
                mov             dx,word ptr cs:packet_buffer+1
                in              al,dx
                mov             cs:packet_buffer+1,al
                mov             cs:packet_buffer,'D'
                mov             word ptr cs:packet_length,2
                put_packet              cs:socket_num,cs:packet_length,offset
packet_buffer

                popall
                ret
portin          endp

portout proc            near
                pushall
                mov             dx,word ptr cs:packet_buffer+1
                mov             al,cs:packet_buffer+3
                out             dx,al
                popall
                ret
portout         endp

execute         proc            near
                pushall
                call            word ptr cs:packet_buffer+1
                popall
                ret
execute         endp

fillmem         proc            near
                pushall
                mov             di,word ptr cs:packet_buffer+1
                mov             es,word ptr cs:packet_buffer+3
                mov             cx,word ptr cs:packet_buffer+9
                mov             al,cs:packet_buffer+10
                cld
                rep             stosb
                popall
                ret
                fillmem         endp

movemem         proc            near
                pushall
                mov             si,word ptr cs:packet_buffer+1
                mov             ds,word ptr cs:packet_buffer+3
                mov             di,word ptr cs:packet_buffer+5
                mov             es,word ptr cs:packet_buffer+7
                mov             cx,word  ptr cs:packet_buffer+9
                call            sitoesdi
                popall
                ret
movemem         endp

;*****************************************************************************
;Fin del M¢dulo que Ejecuta Funciones de Bajo Nivel                           *
;*****************************************************************************
```

```
;*****************************************************************************
;* M¢dulo de Subrutinas Miscelaneas                                (MISC.MOD)*
;* Funci¢n: Subrutinas y Macros de Propositos Generales                      *
;*****************************************************************************
;
ticks_low           dw          ?
ticks_high    dw                ?
cmz           macro             ;***Complemento de la bandera Z  ***
              lahf
              xor               ah,64
              sahf
              endm

stz           macro             ;*** Establece el acarreo de la bandera ***
              cmp               al,al
              endm

clz           macro             ;*** Limpia el acarreo de la bandera ***
              lahf
              and               ah,255-64
              sahf
              endm

pushall       macro             ;*** Graba todos los Registros ***
              pushf
              push              es
              push              ds
              push              bp
              push              di
              push              si
              push              dx
              push              cx
              push              bx
              push              ax
              endm

popall macro              ;*** Restaura todos los Registros ***
              pop               ax
              pop               bx
              pop               cx
              pop               dx
              pop               si
              pop               di
              pop               bp
              pop               ds
              pop               es
              popf
              endm

get_opt  proc                   near        ;*** CX=Pr¢ximo Entero despues de DS:SI de
forma/nnnnn *
              xor               cx,cx
              call              wslash
              cmp               al,1fh
              jb                get_o1
              call              get_dec
get_o1:       ret
get_opt       endp
```

```
get_dec     proc        near        ;*** CX=ASCII N&mero D,cimal en DS:SI
***
            push        ax          ;*** o ASCII HEX Precedido Por '$'
***
            pushf
            xor         cx,cx
            xor         ah,ah
            cmp         byte ptr [si],'$'
            jz          get_hex
get_d1:     mov         al,[si]
            cmp         al,'0'
            jb          get_d2
            cmp         al,'9'
            ja          get_d2
            sub         al,30h
            call        cx10
            add         cx,ax
            inc         si
            jmp         get_d1
get_d2:     popf
            pop         ax
            ret

get_hex:
            inc         si
get_d3:     mov         al,[si]
            cmp         al,'0'
            jc          get_d2
            cmp         al,'9'+1
            jc          get_d4
            and         al,255-32           ;convierte a May&scula
            cmp         al,'A'
            jc          get_d2
            cmp         al,'F'
            ja          get_d2
            sub         al,7
get_d4:     sub         al,'0'
            add         cx,cx
            add         cx,cx
            add         cx,cx
            add         cx,cx
            add         cx,ax
            inc         si
            jmp         get_d3
get_dec     endp

get_ticks               proc        near        ;DX=MSW marca desde el
momento de arranque
            push        es                      ;CX=LSW marca desde el
momento de arranque
            push        di
            push        ax
            mov         ah,1
            int         16h
            mov         ax,40h
            mov         es,ax
            mov         di,6ch
            mov         cx,es:[di]
            mov         dx,es:[di+2]
            pop         ax
            pop         di
```

```
                pop             es
                ret
get_ticks                       endp

wait_a_sec      proc            near            ;pausa aprˆximadamente de un
segundo
                push            cx
                push            dx
                call            get_ticks
                mov             cs:ticks_low,cx
                mov             cs:ticks_high,dx
wait_1:         call            get_ticks
                sub             cx,cs:ticks_low
                sbb             dx,cs:ticks_high
                cmp             cx,19
                jc              wait_1
                pop             dx
                pop             cx
                ret
wait_a_sec      endp

cx10            proc            near            ;*** Multiplica CX por 10 ***
                push            bx
                add             cx,cx
                mov             bx,cx
                add             cx,cx
                add             cx,cx
                add             cx,bx
                pop             bx
                ret
cx10            endp
wslash proc             near            ;al=Primer caracter despues del slash
                mov             al,[si]         ;o controla el caracter si no hay el
slash
                inc             si
                cmp             al,'/'
                jz              wslas1
                cmp             al,1fh
                ja              wslash
                ret
wslas1:         mov             al,[si]
                ret
wslash endp

sitodi proc             near            ;*** Mueve CX byte desde DS:SI hacia DS:DI ***
                pushf
                push            es
                push            ds
                pop             es
                cld
                shr             cx,1
                rep             movsw
                rcl             cx,1
                rep             movsb
                pop             es
                popf
                ret
sitodi endp

essitodi                        proc            near    ;*** Mueve CX byte  desde ES:SI
hacia DS:DI *
```

```
                pushf
                push            es
                push            ds
                push            es              ;cambia ES y DS
                push    ds
                pop             es
                pop             ds
                cld
                shr             cx,1
                rep             movsw
                rcl             cx,1
                rep             movsb
                pop             ds
                pop             es
                popf
                ret
essitodi                        endp

sitoesdi                        proc            near    ;*** Mueve CX byte desde DS:SI
hacia ES:DI *
                pushf
                cld
                shr             cx,1
                rep             movsw
                rcl             cx,1
                rep             movsb
                popf
                ret
sitoesdi                        endp

uppercase                       proc            near            ;*** Convierte los Byte en
AL a May£sculas *
                cmp             al,'a'
                jb              upper1
                cmp             al,'z'
                ja              upper1
                and             al,255-32
upper1:         ret
uppercase                       endp

;*****************************************************************************
;* Fin del Modulo de Subrutinas Miscelaneas                                  *
;*****************************************************************************
;


;*****************************************************************************
;* Blueprint LAN BIOS Interrupciones de la Capa Fisica        (PHYSICAL.MOD)*
;*****************************************************************************
;

physical_interrupt_table        dd              4*maxsockets dup (?)

physical_interrupt              proc    far
                push            bx
                push            cx
                xor             bh,bh ;construye el offset para la entrada del vector
de interrupci¢n F¡sico *
                mov             cl,2
                shl             bx,cl
```

```
                add             bl,ah
                shl             bx,cl
                call            cs:physical_interrupt_table[bx]
                pop             cx
                pop             bx
                retf            2
physical-interrupt              endp

;*****************************************************************************
;* Fin de PHYSICAL.MOD                                                       *
;*****************************************************************************


;*****************************************************************************
;* BP-LAN Modulo del Servidor de Impresoras                    (PRINSERV.MOD)*
;* Funci¢n: Imprime Caracteres en la Impresora Especificada                  *
;*****************************************************************************
;
;*** Variables ***
;
parown  dw              3 dup ('UU',546,0,0)  ;546=# del reloj que marca en 30
segundos

prinser proc            near            ;*** Comprueba el Bloque de Comandos
para imprimir ***
                mov             al,packet_buffer
                cmp             al,'P'
                jnz             prins1
                jmp             parallel
prins1: cmp             al,'S'
                jnz             prins2
                jmp             serial
prins2: ret
prinser endp

parallel                proc            near    ;*** Imprime Caracteres en
Impresora Parallela ***
                                        ;*** Obtiene o libera la Impresora
adjudicada ***
                push            bx
                push            cx
                push            dx
                push            si
                lea             si,parown
                mov             bl,packet_buffer+2
                xor             bh,bh
                mov             cl,3
                shl             bx,cl
                add             si,bx
                mov             bl,socket_num
                cmp             [si+1],bl               ;Esta la Impresora
adjudicada a este cliente?
                jz              parall
                cmp             byte ptr [si],'U'               ;Esta la Impresora
sin Uso?
                jnz             parall2
para10: mov             byte ptr [si],'L'               ;Cambiar la
adjuducaci¢n a este cliente
                mov             bl,socket_num
                mov             [si+1],bl
```

```
para11: mov             ah,packet_buffer+1              ;Imprimir Caracteres
        mov             dl,packet_buffer+2
        mov             al,packet_buffer+3
        xor             dh,dh
        int             17h
        call            get_ticks
        mov             [si+4],cx
        mov             [si+6],dx
para13: mov             packet_buffer,ah                ;Retornar el estatus de
la Impresora
        put_packet              socket_num,1,offset packet_buffer
        pop             si
        pop             dx
        pop             cx
        pop             bx
        ret
para12:    call            get_ticks
        sub             cx,[si+4]
        sbb             dx,[si+6]
        cmp             dx,0
        jnz             para10
        cmp             cx,[si+2]
        ja              para10
        mov             ah,0ffh         ;Indica que Impresora esta con
seguro
        jmp             para13
parallel                endp

serial                  proc            near    ;*** Imprime Caracteres en
Impresora Serial **
                                                ;*** Obtiene o Libera la Impresora
del poseedor **
        mov             ah,packet_buffer+1
        mov             dl,packet_buffer+2
        mov             al,packet_buffer+3
        xor             dh,dh
        int             14h
        mov             packet_buffer,ah
        mov             packet_buffer+1,al
        put_packet              socket_num,2,offset packet_buffer
        ret
serial                  endp
;*****************************************************************************
;* Fin del Modulo Servidor de Impresora de la Red                             *
;*****************************************************************************



;*****************************************************************************
;* Modulo para  manipulacion de los Semaforos                  (SEMAPHOR.MOD)*
;*Funcion: Bloque , Desbloquea o Limpia el Semaforo especificado             *
;*****************************************************************************
;
semaphore_table         dw              256 dup ('UU')  ;Todos los semaforos
sin uso

semaph  proc            near
        cmp             packet_buffer,'L'               ;semaforo de
Bloqueo
        jnz             semap1
        jnz             lock_sem
```

```
semap1: cmp             packet_buffer,'U'               ;semaforo de Desbloqueo
        jnz                 semap2
        jmp                 unlock_sem
semap2: cmp             packet_buffer,'C'               ;Limpia  semaforo
        jnz                 semap3
        jmp                 clear_sem
semap3: ret
semaph  endp

pre_sem proc            near            ;retorna los estos del semaforo
        mov                 packet_buffer,'D'               ;paquete ID para
Responder Solicitud
        mov                 packet_buffer+1,255     ;Adopta Cambiar un
Semaforo sin exito
        lea                 si,semaphore_table
        xor                 bh,bh
        mov                 bl,packet_buffer[1]             ;Obtiene el N&mero
del semaforon
        add                 bx,bx                           ;Construye la
tabla offset del Semaforo
        ret
pre_sem     endp

lock_sem                        proc            near
;Semaforo de Bloqueo sin uso
        pushall
        call                pre_sem
        cmp                 byte ptr [si+bx],'U'    ;Semaphore no es usado?
        jnz                 lock_1
        mov                 packet_buffer[1],0              ;Si es asi, indica
el suceso
        mov                 al,'L'                          ;y semaforo de
bloqueo
        mov                 ah,packet_buffer[1]
        mov                 [si+bx],ax
lock_1: put_packet      cs:socket_num,2,offset packet_buffer
        popall
        ret
lock_sem                        endp

unlock_sem      proc            near            ;semaforo de desbloqueo si es
poseido por este cliente
        pushall
        call            pre_sem
        mov             ax,[si+bx]
        mov             cl,'L'
        mov             ch,socket_num
        cmp             ax,cx                   ;Esta el semaforo
Bloqueado por este cliente?
        jnz             unloc1
        mov             packet_buffer[1],0      ;si es asi, indica un
suceso
        mov             al,'U'                  ;y desbloquea el semaforo
        mov             ah,packet_buffer[1]
        mov             [si+bx],ax
unloc1: put_packet          cs:socket_num,2,offset packet_buffer
        popall
        ret
unlock_sem              endp
```

```
clear_sem               proc            near    ;El semaforo Desbloqueado no
tiene en cuenta el estado actual
            pushall
            call            pre_sem
            mov             packet_buffer[1],0              ;indica suceso
            mov             al,'U'                          ;y Bloquea el
Semaforo
            mov             ah,packet_buffer[1]
            mov             [si+bx],ax
            put_packet              cs:socket_num,2,offset packet_buffer
            popall
            ret
clear_sem               endp

;*****************************************************************************
;* Fin del Modulo para manipulaci¢n de Semaforos      (SEMAPHOR.MOD)         *
;*****************************************************************************



program set_time_from_server:
(*********************************************************************
* Establece el tiempo usando el tiempo del Servidor                  *
* (BPSCREEN.PAS, BPSCREEN.EXE)                                       *
* Funcion:                                                           *
*      Este programa copia la pantalla del                           *
*      servidor al cliente.                                          *
*                                                                    *
* Formato del Comando:                                                  *
*            BPSCREEN /numero del socket                            *
*********************************************************************)
uses dos;
($I bppascal.inc)
var
            i,temp:integer;
procedure screen_block(block_number:integer);
var         i:integer;
begin
            socket_number:=get_opt(1);
            packet_buffer[0]:=ord('r');
            packet_buffer[1]:=0;
            packet_buffer[2]:=2*block_number;
            packet_buffer[3]:=0;
            oacket_buffer[4]:=$b8;
            packet_buffer[5]:=0;
            packet_buffer[6]:=2;
            packet_length:=7;
            put_packet(socket_number):
            temp:=get_packet(socket_number);
            for i=1 to 512 do
                  mem[$b800:512*block_number+i-1]:=packet_buffer[i];
end;

begin
            fori:=0 to 7 do
                  screen_block(i);
end.
```

```
program mail_utility;
uses crt,dos;
(*************************************
* BPMAIL.EXE BPMAIL.PAS              *
*************************************)
const
        title='Blueprint LAN MAIL FACILITY';
        minx=1;maxx=80;miny=1;maxy=25;
        max_messages=100;
        name_length=9;
        line_length=90;
        savefile='mbox';
        mailfile='mailfile.tmp';
        editfile='editfile.tmp';
        savemail='savemail.tmp';
        printer='LPT1';

type
        mailrec=
                record
                                        startpos:longint;
                                        endpos:longint;
                                        from:string[line_length];
                                        cc:string[line_length];
                                        to_:string[line_length];
                                        subject:string[line_length];
                                        date:string[line_length];
                                        return_path:string[line_length];
                                        save,delete:boolean;
                end;

var
            mail:array[1..max_messages+1] of mailrec;
            start_of_line,end_of_line:longint;
            menu_level,keytemp,savekey,message_num,num_messages:integer;
            line:string[line_length];
            username:string[name_length];
            in_message,mail_processed,mail_modified:boolean;
            infilevar,outfilevar:file;
            ch:char;
            editor,maildir,comspec,homedir,netdir:string;

function whoami:string;
var s:string;
begin
             maildir:=getenv('BPMAILDIR');
             if maildir=' ' then
             begin
                         write('BP-LAN Mail DIrectory:');
                         readln(maildir);
             end;

             maildir:=maildir+'\';
             editor:=getenv('BPEDITOR');
             if editor=' ' then
             begin
                         write('BP-LAN Text Editor:');
                         readln(editor);
             end;

             comspec:=getenv('COMSPEC');
```

```
                if comspec=' ' then
                begin
                        write('COMMAND.COM Path:');
                        readln(comspec);
                end;

                comspec:=comspec;

                netdir:=getenv('BPDIR');
                if netdir=' ' then
                begin
                        write('BP-LAN Directory:');
                        readln(netdir);
                end;
                netdir:=netdir+'\';
                homedir:=netdir;

                username:=getenv('BPUSER');
                if username='' then
                begin
                        write('Username:');
                        readln(username);
                end;
                whoami:=username;
end;

function starts_with(line:string;s:string):boolean;
begin
                starts_with:=(copy(line,1,length(s))=s);
end;

function keyval:integer;
var
        temp:integer;
begin
        if savekey<>0 then
        begin
                        savekey:=0;
                        temp:=keytemp
        end
        else
        begin
                        temp:=ord(readkey);
                        if temp=0 then
                                temp:=ord(readkey) shl 8
        end;
        keyval:=temp
end;

function charloc(line:string;start:integer;findchar:char):integer;
var
                s:string[line_length];
begin
                s:=copy(line,start,length(line));
                charloc:=pos(findchar,s)+start-1
end;

procedure delete_file(filename:string);
begin
                exec(comspec,'/c del '+filename+' > nul');
end;
```

```
function date_string:string;
var
              s1,s2:string;
              year,month,day,dayofweek,
              hour,minute,second,sec100:word;
begin
              getdate(year,month,day,dayofweek);
              gettime(hour,minute,second,sec100);
              s1:='';
              str(month,s2);if length(s2)=1 then s2:='0'+s2;s1:=s1+s2+'/';
              str(day,s2);if length(s2)=1 then s2:='0'+s2;s1:=s1+s2+'/';
              str(year,s2);if length(s2)=1 then s2:='0'+s2;s1:=s1+s2+' ';
              str(hour,s2);if length(s2)=1 then s2:='0'+s2;s1:=s1+s2+':';
              str(minute,s2);if length(s2)=1 then s2:='0'+s2;s1:=s1+s2;
              date_string:=s1
end;

procedure reverse_write(message:string);
var
              i:integer;
begin
              i:=textattr;
              textattr:=(i and $80) or ((i and $70) shr 4) or ((i and $7) shl
4);
              write(message);
              textattr:=i;
end;

function get_choice(message:string;choices:string):char;
var
      c:char;
begin
      c:=chr(255);
      while ((c<>choices[1]) and (c<>choices[2]) and
                        (c<>choices[3]) and (c<>choices[4]) and
                        (c<>choices[5]) and (c<>choices[6])) do
                        begin
                                    write(message);
                                    c:=readkey;
                                    writeln(' ',c)
                        end;
                        get_choice:=c
      end;

procedure box(x1,y1,x2,y2:integer;top,bottom:string);
var
      i,x1a,y1a,x2a,y2a,topoffset,bottomoffset:integer;
      clipx,clipy:boolean;
begin
              clipx:=false;
              clipy:=false;
              x1a:=x1; x2a:=x2;
              y1a:=y1; y2a:=y2;
              if (x1a<minx) then
                    begin
                              clipx:=true;
                              x1a:=minx
                    end;
              if (x2a>maxx) then
                    begin
```

```
                        clipx:=true;
                        x2a:=maxx;
                end;
        if (y1a<miny) then
                begin
                        clipy:=true;
                        y1a:=miny
                end;
        if (y2a>maxy) then
                begin
                        clipy:=true;
                        y2a:=maxy;
                end;
window(minx,miny,maxx,maxy);
assigncrt(input); reset(input);
assigncrt(output); rewrite(output);
if not clipy then
for i:=x1a to x2a do
        begin
                gotoxy(i,y1a); write(chr(219));
                gotoxy(i,y2a); write(chr(219));
        end;
if not clipx then
for i:=y1a to y2a do
        begin
                gotoxy(x1a,i); write(chr(219));
                gotoxy(x2a,i); write(chr(219));
        end;
topoffset:=((x2a-x1a-length(top)) shr 1);
bottomoffset:=((x2a-x1a-length(bottom)) shr 1);
for i:=1 to length(top) do
        begin
                gotoxy(x1a+i-1+topoffset,y1a); reverse_write(top[i]);
        end;
for i:=1 to length(bottom) do
        begin
                gotoxy(x1a+i-1+bottomoffset,y2a); reverse_write(bottom[i]);
        end;
window(x1+1,y1+1,x2-1,y2-1);
clrscr;
end;

procedure unix_readln;
var
        i,num_read:integer;
        c:char;
begin
        i:=0;
        num_read:=1;
        c:=' ';
        while (num_read=1) and (c<>chr(10)) and (i<line_length-3) do
                begin
                        blockread(infilevar,c,1,num_read);
                        if num_read<>0 then
                        case ord(c) of
                                9:
                                        begin
                                                if(i and 7)=0 then
                                                begin
                                                        inc(i);
                                                        line[i]:=' '
```

```
                                        end;
                                        while (i and 7)<>0 do
                                        begin
                                                inc(i);
                                                line[i]:=' '
                                        end;
                                end;
                        13:
                                begin
                                        inc(i);
                                        line[i]:=c;
                                        inc(i);
                                        line[i]:=chr(10)
                                end;
                        32..255:
                                begin
                                        inc(i);
                                        line[i]:=c
                                end
                end
        end;
line[0]:=chr(255 and i);
end;

procedure get_header(i:integer);
begin
        if starts_with(line,'From') then
                begin
                        mail[i+1].return_path:=copy(line,6,charloc(line,6,chr(13))-
6);
                        in_message:=true;
                        inc(message_num);
                        if i>0 then
                                mail[i].endpos:=start_of_line-2;
                        mail[i+1].from:=copy(line,6,charloc(line,6,chr(13))-6);

                        mail[i+1].cc:=chr(0);
                        mail[i+1].to_:=chr(0);
                        mail[i+1].subject:=chr(0);
                        mail[i+1].date:=chr(0);
                        mail[i+1].delete:=false;
                        mail[i+1].save:=false
                        end
        else
        if starts_with(line,'Date:') then
                mail[i].date:=copy(line,7,charloc(line,7,chr(13))-7)
        else
        if starts_with(line,'To:') then
                mail[i].to_:=copy(line,5,charloc(line,5,chr(13))-5)
        else
        if starts_with(line,'Cc:') then
                mail[i].cc:=copy(line,5,charloc(line,5,chr(13))-5)
        else
        if starts_with(line,'Subject:') then
                mail[i].subject:=copy(line,10,charloc(line,10,chr(13))-10)
        else
        if starts_with(line,'Return-Path:') then
                mail[i].return_path:=copy(line,15,charloc(line,15,'>')-15)
end;
```

```
procedure process;
var
      s:string[line_length];
      linepos:integer;
begin
      mail_processed:=true;
      num_messages:=0;
      if paramstr(1)='' then
            assign(infilevar,paramstr(1));
      {$I-}
      reset(infilevar,1);
      {$I+}
      if (ioresult<>0) or (filesize(infilevar)=0) then
      begin
            rewrite(infilevar,1);
            highvideo;
            writeln('No mail for ',username);
            writeln('Press <Return> to continue');
            readln;
            lowvideo
      end
      else
      begin
            highvideo;
            write('Processing Mail File');
            in_message:=false;
            message_num:=0;
            mail[1].subject[0]:=chr(0);
            mail[1].date[0]:=chr(0);
            mail[1].return_path[0]:=chr(0);
            start_of_line:=filepos(infilevar);
            unix_readln;
            end_of_line:=filepos(infilevar);
            while line[0]<>chr(0) do
            begin
                  write('.');
                  get_header(message_num);
                  if (in_message and (line[0]=chr(2))) then
                  begin
                        in_message:=false;
                        if message_num>0 then
                              mail[message_num].startpos:=end_of_line;
                        end;
                        start_of_line:=filepos(infilevar);
                        unix_readln;
                        end_of_line:=filepos(infilevar);
                  end;
                  num_messages:=message_num;
                  mail[num_messages].endpos:=start_of_line-1;
                  lowvideo
            end;
      end;


procedure headers(i:integer);
begin
      box(1,1,80,7,title+'  View Message','');
      if i<=num_messages then
            writeln('Message    : ',i,' of ',num_messages)
      else
            writeln('Send Message');
```

```
        if i>num_messages then
                writeln('To          : ',mail[i].return_path);
        if i<=num_messages then
                writeln('From        : ',mail[i].from);
        writeln('Subject       : ',mail[i].subject);
        if i<=num_messages then
                writeln('Date        : ',mail[i].date);
        if i<=num_messages then
                write('Distribution : ',mail[i].to_,'',mail[i].cc)
end;


procedure body(i:integer);
const
        lines_per_screen=16;
var
        startpos,endpos:longint;
        s:string[line_length];
        linenum:integer;
begin
        box(0,7,81,24,'Message Body',chr(24)+'-Prev '+chr(25)+'-Next  D-Del  F-
File'
        +'  P-Print  R-Replay  T-Top  U-Undel  Q-Quit');
        startpos:=mail[i].startpos;
        endpos:=mail[i].endpos;
        linenum:=0;
        seek(infilevar,startpos);
        if not mail[i].delete then
        begin
 mail[i].save:=true;
 while (filepos(infilevar)<endpos) do
         begin
                inc(linenum);
                if linenum>=lines_per_screen then
                        begin
                                highvideo;
                                write
                                ('*** Press SPACE for more ***');
                                lowvideo;
                                keytemp:=keyval;
                                writeln;
                                if (keytemp=ord(' ')) then
                                        begin
                                                clrscr;
                                                linenum:=1;
                                        end
                                else
                                begin
                                        seek(infilevar,endpos);
                                        savekey:=keytemp
                                end;
                        end;
                unix_readln;
                write(line);
                if ord(line[0])>79 then
                        inc(linenum);
         end
         end
 else
         begin
                clrscr;
```

```
                highvideo;
                writeln('This message has been tagged for deletion');
                writeln;
                lowvideo
        end;
 highvideo;
 write('** End of message **');
 lowvideo;
end;


procedure delete (message_num:integer);
begin
        mail_modified:=true;
        mail[message_num].delete:=true;
        clrscr;
        highvideo;
        writeln('Message # ',message_num, ' has been tagged for deletion');
        lowvideo
end;


procedure undelete (message_num:integer);
begin
        mail[message_num].delete:=false;
        clrscr;
        headers(message_num);
        body(message_num);
end;

procedure write_message(message_num:integer);
var
        num_read:integer;
        i,headerpos:longint;
        buffer:array[1..20000] of char;
begin
        headerpos:=0;
        if message_num>1 then
                headerpos:=mail[message_num-1].endpos+1;
        seek(infilevar,headerpos);
        i:=mail[message_num].endpos-(headerpos-1);
        blockread(infilevar,buffer,i,num_read);
        blockwrite(outfilevar,buffer,num_read);
        write('.')
end;


procedure file1 (message_num:integer);
var
        tempfilevar:text;
        filename:string[line_length];
        startpos,endpos:longint;
begin
        clrscr;
        highvideo;
        if (keytemp=ord('p')) or (keytemp=ord('P')) then
                begin
                write('Printing message # ',message_num, ' to ',printer);
                filename:=printer
                end
        else
```

```
            begin
            write('Save message # ',message_num,' to which filename? ');
            readln(filename)
      end;
      assign(tempfilevar,filename);
      rewrite(tempfilevar);
      startpos:=mail[message_num].startpos;
      endpos:=mail[message_num].endpos;
      seek(infilevar,startpos);
      if (keytemp=ord('p')) or (keytemp=ord('P')) then
            write('')
      else
            write('Saving message # ',message_num,' to ',filename);
            writeln(tempfilevar,'Message       : ',message_num,' to
',filename);
            writeln(tempfilevar,'From          : ',mail[message_num].from);
            writeln(tempfilevar,'Subject       : ',mail[message_num].subject);
            writeln(tempfilevar,'Date          : ',mail[message_num].date);
            writeln(tempfilevar,'Return - Path : ',mail[message_num].to_,'  ',
            mail[message_num].cc);
            writeln(tempfilevar);
            while (filepos(infilevar)<endpos) do
            begin
                  unix_readln;
                  write('.');
                  write(tempfilevar,line)
            end;
      if (keytemp=ord('p')) or (keytemp=ord('P')) then
            write(tempfilevar,chr(12));
      close(tempfilevar);
      writeln;
      if (keytemp=ord('p')) or (keytemp=ord('P')) then
            writeln('Print completed')
      else
            writeln('Save completed');
      lowvideo
end;

procedure save(message_num:integer);
begin
      {$I-}
      assign(outfilevar,homedir+savefile);
      reset(outfilevar,1);
      {$I+}
      if ioresult<>0 then
            rewrite(outfilevar,1);
      seek(outfilevar,filesize(outfilevar));
      write_message(message_num);
      close(outfilevar);
end;


procedure update;
var
      i:integer;
begin
      clrscr;
      highvideo;
      begin
      {$I-}
            assign(outfilevar,homedir+savefile);
```

```
                reset(outfilevar,1);
                {$I+}
                if ioresult<>0 then
                        rewrite(outfilevar,1);
                write('Appending undeleted message to ',homedir+savefile);
                for i:=1 to num_messages do
                        if (not(mail[i].delete)) and (mail[i].save) then
                                save(i);
                writeln
        end;
                if mail_processed then
                begin
                        assign(outfilevar,homedir+mailfile);
                        rewrite(outfilevar);
                        write('Updating mail file');
                        for i:=1 to num_messages do
                                if (not mail[i].delete) and (not mail[i].save) then
                                write_message(i);
                        writeln;
                        close(outfilevar);
                        if paramstr(1)='' then
                                exec(comspec,'/c copy /b'+homedir+mailfile+'
'+maildir+username
                                +'> nul')
                        else
                        exec(comspec,'/c copy /b'+homedir+mailfile+' '+paramstr(1)
                        +'> nul');
                        if doserror=0 then
                                delete_file(homedir+mailfile);
                end;
                lowvideo
end;


procedure edit;
var
        i:integer;
        filename:string[line_length];
        c:char;
begin
        box(0,7,81,24,title+' Send Message ','');
        c:=get_choice('Originate message from keyboard, Editor or File (K/E/F)?',
        'KkEeFf');
        case c of
        'K','k':
        begin
                c:=chr(10);
                assign(outfilevar,homedir+mailfile);
                rewrite(outfilevar,1);
                box(0,7,81,24,title+' Edit Message',
                   'Type a ''.'' and < Return > alone on a line to terminate
message');
                clrscr;
                readln(line);
                while (line[0]<>chr(1)) or (line[1]<>'.') do
                begin
                        blockwrite(outfilevar,line[1],length(line));
                        c:=chr(13);
                        blockwrite(outfilevar,c,1);
                        c:=chr(10);
                        blockwrite(outfilevar,c,1);
```

```
                readln(line);
            end;
            close(outfilevar);
        end;
'E','e':
begin
        exec(editor,homedir+mailfile);
        clrscr;
        if menu_level=1 then
            headers(max_messages+1)
        else
            headers(message_num);
        box(0,7,81,24,title+'Send Message','');
end;
'F','f':
begin
        write('Enter Filename of Message: ');
        readln(filename);
        exec(comspec,'/c copy '+filename+' '+ homedir+mailfile+'> nul');
end
end
end;

procedure reply (i:integer);
var
        c:char;
        outfilevar:text;
begin
        edit;
        highvideo;
        c:=get_choice('Do you really want to send this message (Y/N)?','YyNn');
        if (c='Y') or (c='y') then
        begin
            writeln('Sending message');
        {$I-}
        assign(outfilevar,maildir+mail[i].return_path);
        append(outfilevar);
        {$I+}
        if ioresult<>0 then
            rewrite(outfilevar);
            writeln(outfilevar,'From ',username);
            writeln(outfilevar,'To: ',mail[i].return_path);
            writeln(outfilevar,'Subject: ',mail[i].subject);
            writeln(outfilevar,'Date: ',date_string);
            writeln(outfilevar);
            close(outfilevar);
            exec(comspec,'/c type '+homedir+mailfile+'>> '
            +maildir+mail[i].return_path);
            if doserror<>0 then
                writeln('Dos Error = ',doserror)
        end
         else
             writeln('Message aborted by user');
         assign(outfilevar,homedir+mailfile);
         erase(outfilevar);
         writeln('Press <return> to continue');
        lowvideo;
        readln;
        if menu_level=1 then
        begin
            headers(max_messages+1);
```

```
                body(max_messages+1)
        end
end;

procedure send;
begin
        box(12,8,68,17,title+'Send Message','');
        highvideo;
        mail[max_messages+1].from:=username;
        mail[max_messages+1].date:='Today';
        write('To:');
        readln(mail[max_messages+1].return_path);
        write('Subject: ');
        readln(mail[max_messages+1].subject);
        lowvideo;
        mail[max_messages+1].save:=false;
        mail[max_messages+1].delete:=false;
        headers(max_messages+1);
        reply(max_messages+1);
end;

procedure view;
begin
        clrscr;
        message_num:=1;
        headers(message_num);
        body(message_num);
        keytemp:=keyval;
        while (keytemp<>ord('q')) and (keytemp<>ord('Q')) do
        begin
                case keytemp of
                $4800:{Up-Arrow}
                begin
                        if message_num>1 then
                        begin
                                dec(message_num);
                                headers(message_num);
                                body(message_num);
                        end;
                end;
                $5000:{Down_Arrow}
                begin
                        if message_num<num_messages then
                        begin
                                inc(message_num);
                                headers(message_num);
                                body(message_num);
                        end;
                end;
                68,100:{D or d}
                delete(message_num);
                70,102,80,112:{F,f,P or p}
                file1(message_num);
                82,114:{R or r}
                begin
                 reply(message_num);
                 headers(message_num);
                 body(message_num);
                end;
                84,116:{T or t}
                begin
```

```
                    headers(message_num);
                    body(message_num)
                end;
                85,117:{U or u}
                undelete(message_num);
        end;
        keytemp:=keyval;
        end;
        window(minx,miny,maxx,maxy);
        clrscr;
end;

begin
        clrscr;
        menu_level:=0;
        mail_processed:=false;
        mail_modified:=false;
        keytemp:=0;savekey:=0;
        num_messages:=0;
        username:=whoami;
        box(12,8,68,17,title+' Initialization ','Please Wait');
        process;
        while (keytemp<>ord('q')) and (keytemp<>ord('Q'))
         and (keytemp<>ord('x')) and (keytemp<>ord('X')) do
        begin
                if (keytemp=ord('s')) or (keytemp=ord('S')) then
                begin
                        menu_level:=1;
                        send;
                        window(minx,miny,maxx,maxy);
                        clrscr;
                        menu_level:=0
                end
                else
                if (keytemp=ord('y')) or (keytemp=ord('Y')) then
                        if num_messages>0 then
                         view;
                        box(12,8,68,17,title+'Main Menu','Please Select an Option');
                        writeln('Q - Quit mail and update mail file');
                        writeln;
                        writeln('S - Send a mail message or file');
                        writeln;
                        writeln('V - View mail messages (Messages waiting = '
                                ,num_messages,')');
                        writeln;
                        writeln('X - Exit without updating mail file');
                        keytemp:=keyval;
                        end;
                        if (keytemp=ord('q')) or (keytemp=ord('Q')) then
                        update;
                        window(minx,miny,maxx,maxy);
                        clrscr;
end.


;************************************************
;*                                              *
;* Driver para compartir el disco   BPMOUNT.ASM *
```

```
;*                                                    *
*
;****************************************************

code    segment public 'CODE'
                    assume  cs:code,ds:code,es:code

                    org     0

;
; Device driver header
;

header  dd    -1  ;link to the next device
                      dw    2000h
                      dw    start  ;punto de entrada
                      dw    intr   ;punto de entrada de interrupc.
blkdev  db    1       ;one block device
                      db    'BP'   ;indica que el dispositivo es de BPLan
version db    10h     ;BP-LAN version 1.0
socket_num    db      ?
local_drive   db      3    ;dispositivo d
remote_drive  db      2    ;dispositivo c
write_protect db      0    ;por defecto no protejido contra escitura
;
rh_ptr        dd      ?
;
;*** Tabla de vectores del MS DOS ***
;
vector_table:
        dw    init          ;0=inicializa el dispositivo
        dw    media_check   ;1=hacer un media_check
        dw    build_bpb     ;2=arma bloque de parametros del bios
        dw    bad
        dw    read          ;4=lee del dispositivo
        dw    bad,bad,bad
        dw    write         ;8=escribe al dispositivo
        dw    write_verify ;9=escritura con verificacion

        include bpbioshd.mod
        include misc.mod
        include console.mod

packet_length   dw  ?
socket_error    db  ?

;*** Rutina principal ***

start  proc   far
                    mov     word   ptr cs:[rh_ptr],bx
                    mov     word   ptr cs:[rh_ptr+2],es
                    ret
start  endp

;*** Rutina de interrupcion ***

intr  proc  far
                   pushf
                   push  ax
   cli
   mov cs:stackseg,ss
```

```
   mov cs:stackptr,sp
   mov ax,cs
   mov ss,ax
   mov sp,offset local_stack
   sti

            push bx
            push cx
            push dx
            push ds
            push es
            push di
            push si
            push bp

            mov ax,cs
            mov ds,ax
            les di,cs:rh_ptr
            xor bx,bx
            mov bl,es:[di+2] ; salta si es numero valido de comando
            cmp bx,9
            jle intr1
            mov ax,3
            jmp short error

intr1:
            shl bx,1
            mov si,offset vector_table
            jmp word ptr cs:[si+bx]

error:
            or  ax,800Ch
            cmp cs:write_protect,0
            jz  success
            and ax,0ff00h  ; indica error de escritura (protejido)

success:
            or  ax,0100h  ;setea bit de procesado (ok)
            les di,cs:rh_ptr
            mov es:[di+3],ax  ;retorna palabra de status
            pop  bp
            pop  si
            pop  di
            pop  es
            pop  ds
            pop  dx
            pop  cx
            pop  bx

  cli
  mov  sp,cs:stackptr
  mov ss,cs:stackseg
  sti

            pop  ax
            popf
            ret
intr endp

; Setea status para todos los codigos no soportados
bad  proc  near
```

```
                xor   ax,ax
                jmp   success
bad   endp

; Media Check -- Asume que es un dico fijo el de la red
media_check   proc   near
                mov   word ptr packet_length,2
                mov   packet_buffer,'M'
                mov   al,remote_drive
                mov   packet_buffer[1],al
                put_packet socket_num, packet_length, offset packet_buffer
                get_packet
                mov   al,packet_buffer[2]
                mov   byte ptr es:[di+14],al
                xor   ax,ax
                jmp   success
media_check   endp


; Crea bloque de parametros del BIOS
build_bpb   proc   near
                mov   word ptr es:[di+18],offset bpb
                mov   es:[di+20],cs
                call  getboot ; obtiene configuracion del dispositivo
                xor   ax,ax
                jmp   success
build_bpb   endp


; Lee del dispositivo
read   proc   near
                mov   function,0
                jmp   io
read   endp

reasec   proc   near   ; lee un sector absoluto ***
                mov   word ptr packet_length,4
                mov   packet_buffer,'R'
                mov   al,remote_drive
                mov   packet_buffer[1],al
                mov   cx,startsect
                mov   packet_buffer[2],cl
                mov packet_buffer[3],ch
                put_packet socket_num,packet_length,offset packet_buffer
                get_packet
                mov   cx,cs:seclen
                ret
reasec endp

;*** escritura al dispositivo ***
write_verify:
write   proc   near
                cmp   cs:write_protect,0
                jz    write1
                jmp   error
write1: mov   function,1
                jmp   io
write   endp

wrisec   proc   near
                mov   packet_buffer,'W'
```

```
                mov  al,remote_drive
                mov  packet_buffer[1],al
                mov  cx,startsect
                mov  packet_buffer[2],cl
                mov  packet_buffer[3],ch
                mov  cx,cs:seclen
                mov  word ptr packet_length,cx
                add  word ptr packet_length,4
                put_packet socket_num,packet_length,offset packet_buffer
                ret
wrisec endp

; *** Manejador de entradas/salidas del dispositivo ***
io   proc   near
                mov   ax,es:[di+14]
                mov   dtaoff,ax
                mov   ax,es:[di+16]
                mov   dtaseg,ax
                mov   ax,es:[di+18]    ; # sectores
                mov   numsectors,ax
                mov   ax,es:[di+20]    ; sector inicial
                mov   startsect,ax
                push  es
                push  di

io1: mov  ax,numsectors    ; sale si sectores =0
                or    ax,ax
                jz    io5
                mov   es,dtaseg
                mov   al,function
                or    al,al
                jnz   io2
                cmp   startsect,0
                jnz   io4a
                call getboot
                jmp   io3
io4a: call reasec
                      jmp  io3

io2: push cx
                push si
                push di
                mov  si,dtaoff
                lea  di,packet_buffer+4
                mov  cx,cs:seclen
                call essitodi
                pop  di
                pop  si
                pop  cx
                call wrisec
                jmp  io4

io3: pop  di
                push di
                push cx
                push si
                push di
                lea  si,packet_buffer+4
                mov  di,dtaoff
                mov  es,dtaseg
                mov  cx,cs:seclen
```

```
                call sitoesdi
                pop  di
                pop  si
                pop  cx

io4: pop  di
                push di
                dec  numsectors
                inc  startsect
                push di
                mov  di,dtaoff
                add  di,cs:seclen
                mov  dtaoff,di
                jnc  sameseg
                mov  ax,es
                add  ax,1000h
                mov  es,ax
                mov  dtaseg,ax

sameseg: pop  di
                                jmp  io1

io5: pop  di
                pop  es
                cmp  socket_error,0
                jnz  io6
                call ewrite
                jmp  success

io6: mov  ax,word ptr packet_buffer[2]
                call ewrite
                jmp  error

ewrite: mov  al,function
                cmp  al,1
                jnz  ewrit1
                mov  packet_buffer,'T'
                mov  word ptr packet_length,2
                put_packet socket_num,packet_length,offset packet_buffer
ewrit1: ret
io   endp


getboot  proc  near ;** Obtiene configuracion del disco de arranque ***
                push cs
                pop  ds
                mov  startsect,0
                call reasec
                lea  di,packet_buffer+4
                mov  ax,[di+0bh]
                mov  seclen,ax
                mov  al,[di+0dh]
                mov  clulen,al
                mov  ax,[di+0eh]
                mov  numres,ax
                mov  al,[di+10h]
                mov  numfat,al
                mov  ax,[di+11h]
                mov  numdir,ax
                mov  ax,[di+13h]
                mov  numsec,ax
```

```
                mov  al,[di+15h]
                mov  medtyp,al
                mov  ax,[di+16h]
                mov  fatlen,ax
                ret
getboot  endp

;*** BPB ***
bpb_vec  dw  offset bpb

bpb:    ;valores de ejemplo. Son reemplazados en el arranque de la red
seclen  dw  512
clulen   db  1
numres  dw  1
numfat  db  2
numdir  dw  0e0h
numsec  dw  960h
medtyp  db  0f9h
fatlen  dw  7
; * variables *
function   db  ?
startsect  dw  ?
numsectors dw  ?
dtaseg     dw  ?
dtaoff     dw  ?
stackptr   dw  ?
stackseg   dw  ?
; * stack local *
even
              dw  3fh  dup (?)
local_stack    dw  ?
packet_buffer  db  ?
; * messages *
mess1    db  'Mounting Server Drive ',0
mess2    db  ': as client drive ',0
; * inicializa *
init  proc  near
              push di
              push si
              push ds
              push di
              lea  di,mess1
              call messout
              pop  di
              mov  ax,es:[di+20]
              mov  ds,ax
              mov  si,es:[di+18]
              call wslash
              call charout
              sub  al,41h
              mov  cs:remote_drive,al
              push di
              push ds
              push cs
              pop  ds
              lea  di,mess2
              call messout
              pop  ds
              pop  di
              mov  al,es:[di+22]
              mov  cs:local_drive,al
```

```
                add  al,41h
                call charout
                mov  al,':'
                call charout
                mov  al,13
                call charout
                mov al,10
                push cx
                call charout
                call get_opt
                jb   default
                mov  cs:socket_num,cl
                call get_opt
                jb   default
                mov  cs:write_protect,cl
                call get_opt
                jb   default
                mov  cs:seclen,cx
default: pop cx
                pop  ds
                pop  si
                pop  di
                mov  byte ptr es:[di+13],1
                mov  word ptr es:[di+14],offset packet_buffer
                mov  ax,cs:seclen
                add  ax,16
                add  word ptr es:[di+14],ax
                mov  word ptr es:[di+16],cs
                mov  word ptr es:[di+18],offset bpb_vec
                mov  es:[di+20],cs
                jmp  success
init  endp
code  ends
                end



;****************
;* BPREMOTE.ASM *
;****************

codese  segment byte
        assume  cs:codese,ds:codese,es:codese
        org     100h

start:  jmp     remote
;
remote_buffer   db      256 dup (?)
socket_num      db      0
packet_length   dw      0
        include bpbioshd.mod
        include misc.mod

remote  proc    near                    ;Salida si no hay parametros

       mov    al,cs:[80h]
       or     al,al
       jnz    remot1
       ret
```

```
remotl:mov    si,81h
       call   get_opt
       mov    socket_num,cl        ;obtien el # de socket de linea de comando
      call    wslash
      lea     di,remote_buffer[1]
      mov     word ptr packet_length,0

remot2:mov    al,[si]
       inc    si
       cmp    al,'"'
       jz     remot2
       mov    [di],al
       inc    di
       inc    packet_length
       cmp    al,13
       jnz    remot2
       inc    packet_length
       mov    byte ptr remote_buffer,'E'
       put_packet socket_num,packet_length,offset remote_buffer
       int    20h

remote endp
codese ends
        end   start



program change_server__semaphore;

uses dos;
{$I bppascal.inc}
var
    semaphore_number:byte;

procedure change_semaphore;
{*********************************}
{**lock specified semaphore ********}
{*********************************}
var
    temp_string:string;

begin
      {*** Requerimiento de cambio de semaforo ***}
      temp_string:=paramstr(3);
      packet_buffer[0]:=ord(upcase(temp_string[2]));
      packet_buffer[1]:=semaphore_number;
      packet_length:=2;
      put_packet(socket_number);
      {****  status de recepcion correcta ***}
      packet_length:=get_packet(socket_number);
      {*** salida al DOS; seteo variable DOS ERRORLEVEL ***}
      if packet_buffer[1]<>0 then
                  exit_with_error(255);

end;
begin
     socket_number:=get_opt(1);
     semaphore_number:=get_opt(2);
     if ((paramstr(3)='/LOCK') or (paramstr(3)='/lock') or
```

```
        (paramstr(3)='/UNLOCK') or (paramstr(3)='/unlock') or
          (paramstr(3)='/CLEAR') or (paramstr(3)='/clear')) then
          change_semaphore
          else
        begin
             writeln('Syntax: BPSEMAPH /socket_number /semaphore_number',
                     '/semaphore_change_switch');
             writeln(' '' ' ,paramstr(3), ' '' ' ,
             ' Unsupported Semaphore change switch!');
             writeln('Supported switches: /LOCK, /UNLOCK,and /CLEAR')
             end;

end.


;****************
;* BPSERIAL.ASM *
;****************
;
include bpbioshd.mod

codese          segment
                assume  cs:codese,ds:codese
                org     100h

start:          jmp     install
in_stat         dd      0
out_stat        dd      0
;*** byte de manejador de entrada/salida ***
;
in_byte         proc    near            ;*** recibe byte del nodo***
                push    dx
                push    ax
                mov     dx,cs:curport
                add     dx,5

getby1:         in      al,dx
                test    al,1
                jz      getby1
                pop     ax
                mov     dx,cs:curport
                in      al,dx
                pop     dx
                bpbios  link_int,calc_checksum
                retf

in_byte         endp
out_byte        proc    near        ;*** transmite byte al node***
                bpbios  link_int,calc_checksum

putby1:         call    cs:out_stat
                jz      putby1
                call    outport
                retf

out_byte        endp
;*** manejadores de E/S de nivel de hardware ***
;
inport          proc    near        ;***lee puerto de datos del nodo***
                push    dx
```

```
                mov     dx,cs:curport
                in      al,dx
                pop     dx
                ret

inport          endp
outport         proc    near        ;*** escribe al puerto de datos***
                push    dx
                mov     dx,cs:curport
                out     dx,al
                pop     dx
                ret

outport         endp
in_status       proc    near      ;***retorna nz si hay datos en el buffer***
                push    ax
                push    dx
                mov     dx,cs:curport
                add     dx,5
                in      al,dx
                test    al,1
                pop     dx
                pop     ax
                retf

in_status       endp
out_status      proc    near     ;***retorna nz si el buffer de trans- ***
                push    ax       ;*** mision esta vacio                ***
                push    dx
                mov     dx,cs:curport
                add     dx,5
                in      al,dx
                test    al,20h
                pop     dx
                pop     ax
                retf

out_status      endp
curport         dw      03f8h
conf            dw      3
brw             dw      1
;*** fin del codigo residente ***
setbaud         proc    near   ;***setea velocidad en baudios***
                push    cx
                push    dx
                mov     dx,curport   ;Divisor Latch = 1
                add     dx,3
                mov     al,80h
                out     dx,al
                dec     dx            ; Deshabilito interrupciones
                mov     al,0
                out     dx,al
                mov     cx,brw
                mov     dx,curport
                mov     al,cl
                out     dx,al
                mov     al,ch
                inc     dx
                out     dx,al
                inc     dx            ; configura UART
                inc     dx
```

```
                mov     ax,conf
                out     dx,al
                pop     dx
                pop     cx
                push    ax
                mov     ax, offset in_status
                mov     word ptr cs:in_stat,ax
                pop     ax
                mov     word ptr cs:in_stat+2,cs
                push    ax
                mov     ax, offset out_status
                mov     word ptr cs:out_stat,ax
                pop     ax
                mov     word ptr cs:out_stat+2,cs
                ret

setbaud         endp
                include misc.mod

install         proc    near
                mov     al,cs:[80h]   ;uso valores de defecto si la
                or      al,al         ;linea de comando esta vacia
                jz      default
                mov     si,81h
                call    get_opt  ;obtengo direcc. base del puerto serial
                jb      default
                mov     cs:curport,cx
                call    get_opt
                jb      default
                mov     cs:conf,cx
                call    get_opt
                jb      default
                mov     cs:brw,cx

default:
                call    setbaud
                mov     al,receive_status
                bpbios  ctrl_int,install_port,,,offset in_status
                mov     al,transmit_status
                bpbios  ctrl_int,install_port,,,offset out_status
                mov     al,receive_byte
                bpbios  ctrl_int,install_port,,,offset in_byte
                mov     al,transmit_byte
                bpbios  ctrl_int,install_port,,,offset out_byte
                bpbios  ctrl_int,next_port
                mov     dx,offset setbaud
                int     27h

install        endp
codese         ends
               end      start

{*****************************************************************}
{*   FIN DE BPSERIAL.ASM                                         *}
{*****************************************************************}



;******************
;*  BPSERVER.ASM  *
;******************
```

```
codese          segment     byte
                assume      cs:codese,ds:codese,es:codese
                org         100h

init            proc        near     ;***Inicializo todos los nodos ***)
                mov         al,cs:[80h]
                or          al,al
                jz          default
                mov         si,81h
                call        get_opt  ;obtengo buffer_length
                jb          default
                mov         buffer_length,cx
                call        get_opt   ;obtengo background_flag
                mov         background,cl
                cmp         cl,0
                jz          default
                call        intinst

default:        call        cls
                lea         di,banner    ;muestro mensaje
                call        messout
                jmp         poll

init            endp
                include     bpbioshd.mod
                include     misc.mod
                include     console.mod
                include     fileserv.mod       ;comparticion remota de arch.
                include     child.mod          ;ejecucion remota
                include     lowlevel.mod       ;ejecucion de bajo nivel
                include     prinserv.mod       ;impresion remota
                include     semaphor.mod        ;manejo de semaforos

process_packet  proc        near           ;***procesa paquete ***
                call        fileser        ;entra a FILESRV.MOD
                call        comman         ; entra a CHILD.MOD
                call        lowlev         ;entra a LOWLEVEL.MOD
                call        prinser        ;entra a PRINSERV.MOD
                call        semaph         ;entra a SEMAPHOR.MOD
                ret

process_packet  endp
poll            proc        near           ;***reviso todos los nodos por algun
dato
                mov         socket_num,0

poll1:          call        packet
                inc         socket_num
                bpbios      ctrl_int,get_nodes
                cmp         al,socket_num
                jg          poll1
                jmp         poll

poll            endp
packet          proc        near           ;***procesa paquete ***
                mov         ax,ss          ;almaceno inform. de stack
                mov         cs:sslocal,ax
                mov         ax,sp
                mov         cs:splocal,ax
                cli                         ;cambio a stack local
```

```
                mov         ax,cs
                mov         ss,ax
                mov         ax,offset cs:sflocal
                mov         sp,ax
                sti
                call        escape          ;reviso si <ESC> ha sido oprimida
                bpbios      phys_int,receive_status,socket_num
                jnz         packe1
                jmp         packe3

packe1:         bpbios      phys_int,receive_byte,socket_num
                cmp         al,soh          ;leo bloque si recibo cabecera de
datos
                jz          packe2
                jmp         packe3

packe2 :        get_packet  socket_num,packet_length,offset packet_buffer
                 call       process_packet

packe3 :        cli                 ;restauro stack anterior
                mov         ss,cs:sslocal
                mov         sp,cs:splocal
                sti
                push        cs
                pop         ds
                ret

packet          endp
;
;***variables ***
;
sslocal         dw          ?
splocal         dw          ?
                dw          128 dup(?)
sflocal         dw          ?
banner          db          32,201,76 dup(205),187,13,10,32,186
                db          19 dup (32)
                db          "  BP-LAN  version 1.00 server active"
                db          19 dup (32)
                db          186,13,10,32,200,76 dup (205),188,13,10,10,0

packet_length   dw          ?
socket_num      db          0
buffer_length   dw          512
intnum          equ         28h
new_int         proc        far
                pushf
                push        es
                push        bx
                push        ax
                cmp         byte ptr cs:in_28,0
                jnz         new_i4
                mov         byte ptr cs:in_28,1
                pushall
                push        cs
                pop         ds
                mov         cs:socket_num,0

new_i1:         call        packet
                inc         socket_num
                bpbios      ctrl_int,get_nodes
```

```
                cmp             ah,cs:socket_num
                jg              new_i1
                popall
                mov             byte ptr cs:in_28,0
new_i4:
                pop             ax
                pop             bx
                pop             es
                popf
                jmp             cs:old_int_vector
new_int         endp

;*************************************************
;*** instalo nuevo manejador de interrupciones ***
;*************************************************
intinst         proc            near
                mov             ah,34h
                int             21h
                mov             dos_flag_offs,bx
                mov             dos_flag_seg,es
                mov             ah,35h                  ;obtengo vector de interrupcion
                mov             al,intnum               ;obtengo numero de interrupcion
                int             21h
                mov             old_int_offs,bx         ;almaceno interrupcion anterior
                mov             old_int_seg,es
                mov             ah,25h                  ;seteo vector de interrupcion
                mov             al,intnum               ;seteo numero de interrupcion
                mov             dx,offset new_int       ;apunto a nueva rutina
                int             21h
                lea             dx,packet_buffer
                add             dx,16
                add             dx, buffer_length
                int             27h

intinst         endp
;*** variables  ***
;
dos_flag_offs dw                ?
dos_flag_seg  dw                ?
old_int_vector label            dword
old_int_offs  dw                ?
old_int_seg   dw                ?
int_src_vector label            dword
int_src_offs  dw                ?
int_src_seg   dw                ?
        dw    100               dup(?)

newstack        dw              ?
stackseg        dw              ?
stackoff        dw              ?
in_28           db              0
packet_buffer db                ?
codese          ends
                end             init



;***************
;*  BPTERM.ASM *
```

```
;***************
codese          segment     byte
                assume      cs:codese,ds:codese,es:codese
                org         100h

start:          jmp         term
;
portnum         db          0
                include     bpbioshd.mod
                include     misc.mod

term            proc        near            ;salgo si no hay parametros
                mov         al,cs:[80]
                or          al,al
                jnz         term1
                ret

term1:          mov         si,81h
                call        get_opt
                mov         portnum,cl

term2:          mov         ah,1
                int         16h
                jz          term3
                mov         ah,0
                int         16h
                cmp         al,1bh
                jz          term5
                bpbios      phys_int,transmit_byte,portnum

term3:          bpbios      phys_int,receive_status,portnum
                jz          term4
                bpbios      phys_int,receive_byte,portnum
                mov         ah,0eh          ;muestro caracter en AL
                int         10h

term4:          jmp         term2
term5:          int         20h
term            endp
codese          ends
                end         start



;***************
;* BPTIME.ASM *
;***************

codeseg         segment     byte
                assume      cs:codeseg,es:codeseg
                org         100h

start:          jmp         time
;
socket_num      db          0
time_request    db          "r"
time_addoff     dw          046ch
time_addseg     dw          0000h
time_length     dw          0004h
```

```
include         bpbioshd.mod
include         misc.mod

time            proc            near            ;sale si no hay parametros
                mov             al,cs:[80h]
                or              al,al
                jnz             time1
                ret

time1:          mov             si,81h
                call            get_opt
                mov             socket_num,cl
                put_packet      socket_num7,offset time_request
                get_packet      socket_num,cx, offset time_request
                push            es
                push            ds
                mov             cx,4
                mow             ax,0
                mov             es,ax
                mov             di,046ch
                mov             si,offset time_request+1
                call            sitoesdi
                pop             ds
                pop             es
                int             20h

time            endp
codeseg         ends
                end             start



;***************
;* CHILD.MOD   *
;***************
;
;***variables***
;
pblock:
envstr              dw          ?
comlin              dw          ?,?
fcb1                dw          ?,?
fcb2                dw          ?,?
fname               db          '/command.com',0
command             db          3,'/c'
                    db          128 dup (?)

drvmod              db          26 dup (-1)
comman              proc        near
                    cmp         packet_buffer,'E'
                    jz          comma1
                    ret

comma1:             mov         cx,packet_length
                    dec         cx
                    add         byte ptr command,cl
                    lea         si,packet_buffer+1
                    lea         di,command+3
                    call        sitodi
                    jmp         child
```

```
comman          endp
child           proc    near            ;***ejecuta proceso child  ***
                call    drive_unknown_state
                mov     ax,4a00h        ;determina memoria libre
                mov     bx,07ffh
                int     21h
                mov     ax,cs           ;inicializa ambiente child

                mov     [comlin+2],ax
                mov     [fcb1+2],ax       ;segmento 1 del FCB
                mov     [fcb2+2],ax
                mov     ax,[2ch]           ;segmento del ambiente
                mov     word ptr [envstr],0
                lea     ax,command
                mov     [comlin],ax
                mov     ax,5ch              ;offset de fcb1
                mov     [fcb1],ax
                mov     al,6ch              ;offset  de fcb2
                mov     [fcb2],ax
                mov     ax,4b00h            ;carga y ejecuta commando
                lea     bx,pblock
                lea     dx,fname
                int     21h
                mov     ax,4900h            ;libera memoria
                int     21h
                cli
                mov     ss,cs:sslocal
                mov     sp,cs:splocal
                sti
                push    cs
                pop     ds
                ret

child           endp
drive_unknown_state   proc  near            ;marca todos los drives como
                lea     si,drive_table      ;modificados
                mov     bx,maxsockets
                mov     cl,5
                shl     bx,cl
                mov     al,-1
                rep     stosb
                ret

drive_unknown_state     endp
;*******************************************
;**  fin del proceso child                **
;*******************************************


;***************
;* CONSOLE.MOD *
;***************
background      db      0   ;ejecucion en background si byte  > 0
escape          proc    near
                cmp     byte ptr cs:background,1
                jz      escap1
                call    esctst
                jnz     escap1
```

```
                int     20h

escap1:         ret
escape          endp
esctst          proc    near        ;*** returna bandera z si esc ha
                mov     ah,1        ;sido oprimida ***
                int     16h
                jz      escts2

escts1:         xor     ah,ah
                int     16h
                cmp     al,27
                ret
escts2:         xor     al,al
                cmp     al,1
                ret

esctst          endp
cls             proc    near        ;borra la pantalla
                push    ax
                push    bx
                mov     ah,0fh
                int     10h
                mov     ah,0
                int     10h
                pop     bx
                pop     ax
                ret

cls             endp
charout         proc    near        ;muestra character en al a la consola
                cmp     cs:background,1
                jz      charo1
                push    ax
                push    bx
                mov     ah,0eh
                xor     bx,bx
                int     10h
                pop     bx
                pop     ax

charo1:         ret
charout         endp
decout          proc    near
                push    ax
                push    bx
                push    dx
                push    ax
                mov     al,':'
                call    charout
                pop     dx
                mov     bx,10000
                call    decou1
                mov     bx,1000
                call    decou1
                mov     bx,100
                call    decou1
                mov     bx,10
                call    decou1
                mov     bx,1
                call    decou1
```

```
                pop     dx
                pop     bx
                pop     ax
                ret

decou1:         mov     ax,dx
                xor     dx,dx
                div     bx
                add     al,'0'
                jmp     charout

decout          endp
messout         proc    near
                push    ax
                push    di

messo1:         mov     al,[di]
                or      al,al
                jz      messo2
                call    charout
                inc     di
                jmp     messo1

messo2:         pop     di
                pop     ax
                ret

messout         endp
;****************************************
;*  fin del modulo de consola           ***
;****************************************



      {$M $7000,0,0 }
      program mail_utility;
      uses crt,dos;
      (*****************************
      *   (BPINST.EXE,BPINST.PAS   *
      *****************************)
      const
            title='Blueprint  LAN Version  1.00  Installation  Utility
07/06/93/';
            minx=1;maxx=80;miny=1;maxy=25;

      type    serial_type=record
                    base_address:string[8];
                    baud_rate_word:word;
                    config_byte:byte;
              end;
              drive_type=record
                    client_drive:char;
                    server_drive:char;
                    socket:string[8];
                    write_protect_flag:byte
              end;
      var
              menu_level,keytemp,savekey,message_num,num_messages,
              transcript_window_x,transcript_window_y,
              help_window_x, help_window_y,
```

```
        question_window_x,question_window_y:integer;
        infilevar,outfilevar,transcript:text;
        ch,sourcedrive,bootdrive:char;
        bpdir,bpmaildir,bpuser,bpeditor:string[255];
        dirinfo:searchrec;
        num_sockets,num_drives:byte;
        last_drive:char;
        serial:array[0..255] of serial_type;
        drive:array[1..24] of drive_type;

procedure exit_to_dos;
var regs:registers;
begin
        regs.ax:=$4c00;
        msdos(regs)
end;
procedure delete_file(filename:string);
begin
        exec(getenv('COMSPEC'),'/C del '+filename+'>nul');
end;

procedure copy_file(filename1,filename2:string);
begin
        exec(getenv('COMSPEC'),'/c copy '+filename1+' '+filename2+'>nul');
end;

function starts_with(line:string;s:string) :boolean;
begin
        starts_with:=(copy(line,1,length(s))=s);
end;

function keyval:integer;
var
        temp:integer;
begin
        if savekey<>0 then
      begin
        savekey:=0;
          temp:=keytemp
      end
      else
      begin
          temp:=ord(readkey);
             if temp=0 then
             temp:=ord(readkey) shl 8
      end;
      keyval:=temp
end;

function charloc(line:string;start:integer;findchar:char):integer;
var
        s:string;
begin
        s:=copy(line,start,length(line));
      charloc:=pos(findchar,s)+start-1
end;
procedure reverse_write(message:string);
var
        i:integer;
begin
        i:=textattr;
```

```
     textattr:=(i and $80) or((i and $70) shr 4) or ((i and $7) shl
4);
     write(message);
     textattr:=i;
end;
function get_choice(message:string;choices:string):char;
var
      c:char;
begin
      c:=chr(225);
     while ((c<>choices[1]) and (c<>choices[2]) and
      (c<>choices[3]) and (c<>choices[4]) and
        (c<>choices[5]) and (c<>choices[6])) do
      begin
            write(message);
            c:=readkey;
            writeln(' ',c)
            end;
      get_choice:=c
end;
procedure box(x1,y1,x2,y2:integer;top,bottom:string);
var
      i,x1a,y1a,x2a,y2a,topoffset,bottomoffset:integer;
     clipx,clipy:boolean;
begin
      clipx:=false;
     clipy:=false;
      x1a:=x1;x2a:=x2;
      y1a:=y1;y2a:=y2;
      if (x1a<minx)then
      begin
        clipx:=true;
           x1a:=minx
      end;
      if (x2a>maxx) then
      begin
        clipx:=true;
           x2a:=maxx
      end;
      if (y1a<miny) then
      begin
         clipy:=true;
            y1a:=miny
      end;
      if (y2a>maxy) then
      begin
         clipy:=true;
            y2a:=maxy
      end;
      window(minx,miny,maxx,maxy);
      assigncrt(input);reset(input);
      assigncrt(output);rewrite(output);
      if not clipy then
      for i:=x1a to x2a do
      begin
         gotoxy(i,y1a);write(chr(219));
         gotoxy(i,y2a);write(chr(219))
      end;
      if not clipx then
      for i:=y1a to y2a do
      begin
```

```
            gotoxy(x1a,i);write(chr(219));
               gotoxy(x2a,i);write(chr(219));
         end;
         topoffset:=((x2a-x1a-length(top)) shr 1);
         bottomoffset:=((x2a-x1a-length(bottom)) shr 1);
         for i:=1 to length(top) do
         begin
            gotoxy(x1a+i-1+topoffset,y1a);reverse_write(top[i]);
         end;
         for i:=1 to length(bottom) do
         begin
             gotoxy(x1a+i-1+bottomoffset,y2a);reverse_write(bottom[i]);
         end;
         window(x1+1,y1+1,x2-1,y2-1);
         clrscr;
end;

procedure open_transcript;
begin
       box(1,1,80,13,title,'INSTALLATION TRANSCRIPT');
       transcript_window_x:=wherex;
       transcript_window_y:=wherey;
end;
procedure open_help(s:string);
begin
       box(1,15,80,18,s,'HELP WINDOW');
       help_window_x:=wherex;
       help_window_y:=wherey;
end;

procedure open_question;
begin
       box(1,20,80,22,'Please Answer the Following Question:',
             'INSTALLATION QUESTIONS');
       question_window_x:=wherex;
       question_window_y:=wherey;
end;

procedure write_question(s:string);
begin
       highvideo;
       window(2,21,79,21);
       gotoxy(question_window_x,question_window_y);
       write(s);
       lowvideo;
end;

procedure writeln_transcript(s:string);
begin
       window(2,2,79,10);
       gotoxy(transcript_window_x,transcript_window_y);
       writeln(s);
       if paramstr(1)<>'' then
         writeln(transcript,s);
       transcript_window_x:=wherex;transcript_window_y:=wherey;
end;

procedure writeln_help(s:string);
begin
       window(2,16,79,18);
       gotoxy(help_window_x,help_window_y);
```

```
            writeln(s);
            help_window_x:=wherex;help_window_y:=wherey
      end;
      procedure install;
      var
            i:integer;
            outfilevar:text;
      begin
            open_help('Installing BP-LAN...');
            open_transcript;
            writeln_transcript('Creating BP-LAN Directory: '+bpdir);
            {$I-}
            mkdir(bpdir);
            {$I+}
            if ioresult<>0 then
            writeln_transcript('Directory ('+bpdir+') Already Exists');
            writeln_transcript('Copying     BP-LAN   Files   from
'+sourcedrive+':to'+ bpdir);
            copy_file(sourcedrive+':\bp*.*',bpdir);
            writeln_transcript(
      'Saving Unmodified CONFIG.SYS and AUTOEXEC.BAT files to '+bpdir);
            copy_file(bootdrive+':\autoexec.bat ',bpdir+'\*.old');
            copy_file(bootdrive+':\config.sys',bpdir+'\*.old');
            writeln_transcript('Creating BP-LAN Startud Batch File: '+
                  bpdir+'\bprun.bat');
            assign(outfilevar,bpdir+'\bprun.bat');
            rewrite(outfilevar);
            writeln(outfilevar,'set bpdir=',bpdir);
            writeln(outfilevar,'set bpuser=',bpuser);
            writeln(outfilevar,'set bpmaildir=',bpmaildir);
            writeln(outfilevar,'set bpeditor=',bpeditor);
            writeln(outfilevar,'paht=%paht%;%bpdir%');
            writeln(outfilevar,'bpbios');
            for i:=0 to num_sockets-1 do
            begin
                  writeln(outfilevar,'bpserial /$',serial[i].base_address,' /3
/',serial[i].baud_rate_word);
            end;
            close(outfilevar);
            assign(outfilevar,bootdrive+':\autoexec.bat');
            append(outfilevar);
            writeln(outfilevar,bpdir,'\bprun.bat');
            close(outfilevar);
            assign(outfilevar,bootdrive+':\config.sys');
            append(outfilevar);
            writeln(outfilevar,'shell=',getenv('COMSPEC'),' /p /e:999');
            for i:=1 to num_drives do
            begin
                  writeln(outfilevar,'device=',bpdir,'\bpmount.sys
/',drive[i].server_drive,'/',drive[i].socket);
            end;
            close(outfilevar);
            writeln_help('Reboot this machine to activate BP-LAN Software.');
      end;
      procedure install_info;
      begin
            open_help('Source Drive');
            writeln_help('The Source Drive refers to the letter of the drive
(usually A: or B:) which');
            writeln_help('contains the BP_LAN version 1.00 Software,
installation program.');
```

```
            write_question('Enter Letter Source Drive: ');
            readln(sourcedrive);
              writeln_transcript( 'Source       Drive
='+upcase(sourcedrive)+':');
            open_help('Boot Drive');
            writeln_help(
      'The Boot Drive refers to the letter of the drive (usually A: or
C:)which');
            writeln_help(
      'contains the CONFIG.SYS and AUTOEXEC.BAT files to be modified.');
          write_question('Enter Letter Boot Drive:');
          readln(bootdrive);
            writeln_transcript('Boot Drive = '+upcase(bootdrive)+':');
            open_help('Destination Directory');
            writeln_help(
      'The Destination Directory refers to the Directory on the Local PC,
where');
            writeln_help(
      'BP-LAN files are to be copied (example, C:\BP-LAN).');
            write_question('Enter BP-LAN Destination Directory: ');
            readln(bpdir);
            writeln_transcript('Destination Directory = '+bpdir);
            open_help('Verify Answers');
            writeln_help(
      'Press Y if the information in the transcript window is correct.');
            writeln_help(
      'Press N if not correct. Press E to Exit Installation Utility.');
            write_question('');
            highvideo;
            ch:=get_choice('Are the above answers correct?  (Y/N/E)
','YyNnEe');
            lowvideo;
            if (ch='N') or (ch='n') then
              install_info
            else
            if (ch='E') or (ch='e') then
                  exit_to_dos
      end;

      procedure mail_info;
      begin
            open_help('Electronic Mail Directory');
            writeln_help(
      'The Electronic Mail Directory refers to the directory on the server');
            writeln_help(
      'where shared mail messages are to be stored (example, D:\MAIL).');
       write_question('Enter BP-LAN Electronic Mail Directory: ');
      readln(bpmaildir);
      writeln_transcript('BP-LAN Electronic Mail Directory ='+bpmaildir);
            open_help('Default Text Editor');
            writeln_help(
      'The Default Text Editor refers to the path of the editor with which the
user');
            writeln_help(
      'will create outgoing electronic mail messages
(example,C:\BRIEF\B.EXE).');
            write_question('Enter Path of BP-LAN Default Text Editor :');
            readln(bpeditor);
            writeln_transcript('BP-LAN Default Text editor = '+bpeditor);
            open_help('Username');
            writeln_help(
```

```
        'The Username refers to the 8 characters, or less name which will
uniquely');
              writeln_help(
        'identify the user of the machine (example, MSMITH).');
              write_question('Enter Username (8 characters or less) : ');
              readln(bpuser);
              writeln_transcript('Username = '+bpuser);
              open_help('verify answers ');
              writeln_help(
         'Press Y if the information in the transcript window is correct.');
              writeln_help(
         'Press N if not correct.  Press E to exit Installation Utility.');
              write_question('');
              highvideo;
              ch:=get_choice('Are the above answers correct?  (Y/N/E)
','YyNnEe');
              lowvideo;
              if (ch='N') or (ch='n') then
                 mail_info
              else
              if (ch='E') or (ch='e') then
                    exit_to_dos
        end;
        procedure serial_info;
        var     s,s2:string;baud_rate:longint;i:integer;
        begin
              open_help('Network Sockets');
              writeln_help(
        'The number of Network Sockets refers the number of RS232C serial
ports');
              writeln_help(
        'that you wish to connect to other network nodes  (clients and
servers).');
              write_question('Enter the Number of BP-LAN Network Sockets:');
              readln(num_sockets);
              str(num_sockets,s);
              writeln_transcript('Number of BP-LAN Network Sockets = '+s);
              for i:=0 to num_sockets-1 do
              begin
              open_help('Serial Port Base Address');
        writeln_help(
        'Base Addres refers to the hexadecimal number of the lowest');
        writeln_help(
        'I/O addres used by this RS232C Port.  For example, COM1=3F8 and
COM2=2F8');
              str(i,s);
              write_question('Enter the Base Address for Socket '+s+': ');
              readln(s2);serial[i].base_address:=s2;
              writeln_transcript('Base Addres for Socket '+s+' = '+s2);
              open_help('Serial Port Baud Rate');
        writeln_help(
        'Baud Rate refers to the data transmission speed. Slow (4.77 Mhz)
computers');
        writeln_help(
        'can transmit at 57600 baud.  Faster computers may transmit at 115200
baud');
              write_question('Enter Baud Rate: ');
              readln(baud_rate);
              serial[i].baud_rate_word:=115200 div baud_rate;
              str(baud_rate,s);
              writeln_transcript('Baud Rate = '+s);
```

```
            end;
            open_help('Verify Answers');
            writeln_help(
      'Press Y if the information in the transcript window is correct.');
            writeln_help(
      'Press N if not corret.  Press E to Exit Installation Utillity.');
            write_question('');
            highvideo;
            ch:=get_choice('Are the above answer correct?  (Y/N/E)','YyNnEe');
            lowvideo;
            if (ch='N') or (ch='n')  then
               serial_info;
      end;

      procedure drive_info;
      var     s:string;i:integer;
      begin
            open_help('Network Drives');
            writeln_help(
      'The Number of Network Drive refers the number of shared network
drives');
            writeln_help(
      'that you wish to make available to this client.');
            write_question('Enter the Number of BP-LAN Network Drives: ');
            readln(num_drives);
            str(num_drives,s);
            writeln_transcript('Number of BP-LAN Network Drives = '+s);
            if num_drives<>0 then
            begin
            open_help('Last Non-Network Drive');
            writeln_help(
      'The Last Non-Network Drive refers to the letter of the last drive prior
to');
            writeln_help(
      'installing BP-LAN.  Example B for a floppy machine, C for hard drive.');
            write_question('Enter the Letter of the Last Non-Network Drive:
');
            readln(s);
            last_drive:=upcase(s[1]);
            writeln_transcript('The Last Non-Networked Drive on this client =
'+last_drive+':');
          for i:=1 to num_drives do
          begin
          open_help('Shared Drive  '+chr(ord(last_drive)+i)+':Socket');
      writeln_help(
      'Shared Drive Socket refers to the number of the socket connected to
the');
      writeln_help(
      'server for this drive.');
            str(i,s);
            write_question('Enter the socket Number of the Server of client
drive'+
                  chr(ord(last_drive)+i)+':');
            readln(s);drive[i].socket:=s;
            writeln_transcript( ' S e r v e r   with
Driver'+chr(ord(last_drive)+i)+': is '+
                  'connected via socket '+s);
                  open_help('Server Drive Letter');
      writeln_help(
      'Server Drive Letter refers to the letter of the drive on the server
that');
```

```
      writeln_help(
      'you wish to mount as drive '+chr(ord(last_drive)+i)+': on this client');
            write_question('Enter Server Drive Letter:  ');
            readln(s);
            drive[i].server_drive:=upcase(s[1]);
            writeln_transcript('Mounting Server Drive '+upcase(s[1])+':as
Client Drive '+
                  chr(ord(last_drive)+i)+':');
            end;
            end;
            open_help('Verify Answers');
            writeln_help(
      'Press Y if the information in the transcript window is correct.');
            writeln_help(
      'Press N if not correct.  Press E to Exit Installation Utility.');
            write_question('');
            highvideo;
            ch:=get_choice('Are the above answers correct?  (Y/N/E)
','YyNnEe');
            lowvideo;
            if (ch='N') or (ch ='n') then
               drive_info;
      end;

      begin
            clrscr;
            if paramstr(1)<>'' then
            begin
                  assign(transcript,paramstr(1));
                 rewrite(transcript);
            end;
            open_transcript;
            open_question;
            open_help('');
            install_info;
            open_transcript;
            mail_info;
            open_transcript;
            serial_info;
            open_transcript;
            drive_info;
            open_transcript;
            install;
            write_question('Press <Return> to Exit to DOS');
            if paramstr(1)<>'' then
            begin
                  writeln(transcript,chr(12));
                 close(transcript);
            end;
            readln;
      window(minx,miny,maxx,maxy);
      clrscr
   end.
```