

## ASCII Coding

### A.1 International alphabet No. 5

ANSI defined a standard alphabet known as ASCII. This has since been adopted by the CCITT as a standard, known as IA5 (International Alphabet No. 5). The following tables define this alphabet in binary, as a decimal value, as a hexadecimal value and as a character.

Binary	Decimal	Hex	Character	Binary	Decimal	Hex	Character
00000000	0	00	NUL	00010000	16	10	DLE
00000001	1	01	SOH	00010001	17	11	DC1
00000010	2	02	STX	00010010	18	12	DC2
00000011	3	03	ETX	00010011	19	13	DC3
00000100	4	04	EOT	00010100	20	14	DC4
00000101	5	05	ENQ	00010101	21	15	NAK
00000110	6	06	ACK	00010110	22	16	SYN
00000111	7	07	BEL	00010111	23	17	ETB
00001000	8	08	BS	00011000	24	18	CAN
00001001	9	09	HT	00011001	25	19	EM
00001010	10	0A	LF	00011010	26	1A	SUB
00001011	11	0B	VT	00011011	27	1B	ESC
00001100	12	0C	FF	00011100	28	1C	FS
00001101	13	0D	CR	00011101	29	1D	GS
00001110	14	0E	SO	00011110	30	1E	RS
00001111	15	0F	SI	00011111	31	1F	US

Binary	Decimal	Hex	Character	Binary	Decimal	Hex	Character
00100000	32	20	SPACE	00110000	48	30	0
00100001	33	21	!	00110001	49	31	1
00100010	34	22	"	00110010	50	32	2
00100011	35	23	#	00110011	51	33	3
00100100	36	24	\$	00110100	52	34	4
00100101	37	25	%	00110101	53	35	5
00100110	38	26	&	00110110	54	36	6
00100111	39	27	/	00110111	55	37	7
00101000	40	28	(	00111000	56	38	8
00101001	41	29	)	00111001	57	39	9
00101010	42	2A	*	00111010	58	3A	:
00101011	43	2B	+	00111011	59	3B	;
00101100	44	2C	,	00111100	60	3C	<
00101101	45	2D	-	00111101	61	3D	=
00101110	46	2E	.	00111110	62	3E	>
00101111	47	2F	/	00111111	63	3F	?

<i>Binary</i>	<i>Decimal</i>	<i>Hex</i>	<i>Character</i>	<i>Binary</i>	<i>Decimal</i>	<i>Hex</i>	<i>Character</i>
01000000	64	40	@	01010000	80	50	P
01000001	65	41	A	01010001	81	51	Q
01000010	66	42	B	01010010	82	52	R
01000011	67	43	C	01010011	83	53	S
01000100	68	44	D	01010100	84	54	T
01000101	69	45	E	01010101	85	55	U
01000110	70	46	F	01010110	86	56	V
01000111	71	47	G	01010111	87	57	W
01001000	72	48	H	01011000	88	58	X
01001001	73	49	I	01011001	89	59	Y
01001010	74	4A	J	01011010	90	5A	Z
01001011	75	4B	K	01011011	91	5B	[
01001100	76	4C	L	01011100	92	5C	\
01001101	77	4D	M	01011101	93	5D	]
01001110	78	4E	N	01011110	94	5E	'
01001111	79	4F	O	01011111	95	5F	

<i>Binary</i>	<i>Decimal</i>	<i>Hex</i>	<i>Character</i>	<i>Binary</i>	<i>Decimal</i>	<i>Hex</i>	<i>Character</i>
01100000	96	60		01110000	112	70	p
01100001	97	61	a	01110001	113	71	q
01100010	98	62	b	01110010	114	72	r
01100011	99	63	c	01110011	115	73	s
01100100	100	64	d	01110100	116	74	t
01100101	101	65	e	01110101	117	75	u
01100110	102	66	f	01110110	118	76	v
01100111	103	67	g	01110111	119	77	w
01101000	104	68	h	01111000	120	78	x
01101001	105	69	i	01111001	121	79	y
01101010	106	6A	j	01111010	122	7A	z
01101011	107	6B	k	01111011	123	7B	{
01101100	108	6C	l	01111100	124	7C	:
01101101	109	6D	m	01111101	125	7D	}
01101110	110	6E	n	01111110	126	7E	~
01101111	111	6F	o	01111111	127	7F	DEL

## A.2 Extended ASCII code

---

The standard ASCII character has 7 bits and the basic set ranges from 0 to 127. This code is rather limited as it does not contain symbols such as Greek letters, lines, and so on. For this purpose the extended ASCII code has been defined. This fits into character numbers 128 to 255. The following four tables define a typical extended ASCII character set.

<i>Binary</i>	<i>Decimal</i>	<i>Hex</i>	<i>Character</i>	<i>Binary</i>	<i>Decimal</i>	<i>Hex</i>	<i>Character</i>
10000000	128	80	ç	10010000	144	90	é
10000001	129	81	ü	10010001	145	91	æ
10000010	130	82	é	10010010	146	92	æ
10000011	131	83	â	10010011	147	93	ô
10000100	132	84	ä	10010100	148	94	ö
10000101	133	85	à	10010101	149	95	ò
10000110	134	86	å	10010110	150	96	û
10000111	135	87	ç	10010111	151	97	ù
10001000	136	88	ê	10011000	152	98	ÿ
10001001	137	89	ë	10011001	153	99	ö
10001010	138	8A	è	10011010	154	9A	Ü
10001011	139	8B	ï	10011011	155	9B	¢
10001100	140	8C	î	10011100	156	9C	£
10001101	141	8D	ì	10011101	157	9D	¥
10001110	142	8E	Ä	10011110	158	9E	•
10001111	143	8F	Å	10011111	159	9F	f

<i>Binary</i>	<i>Decimal</i>	<i>Hex</i>	<i>Character</i>	<i>Binary</i>	<i>Decimal</i>	<i>Hex</i>	<i>Character</i>
10100000	160	A0	á	10110000	176	B0	
10100001	161	A1	í	10110001	177	B1	
10100010	162	A2	ó	10110010	178	B2	
10100011	163	A3	ú	10110011	179	B3	
10100100	164	A4	ñ	10110100	180	B4	
10100101	165	A5	Ñ	10110101	181	B5	
10100110	166	A6	¤	10110110	182	B6	
10100111	167	A7	¤	10110111	183	B7	
10101000	168	A8	¿	10111000	184	B8	
10101001	169	A9	•	10111001	185	B9	
10101010	170	AA	¬	10111010	186	BA	
10101011	171	AB	½	10111011	187	BB	
10101100	172	AC	¼	10111100	188	BC	
10101101	173	AD	¡	10111101	189	BD	
10101110	174	AE	«	10111110	190	BE	
10101111	175	AF	»	10111111	191	BF	

<i>Binary</i>	<i>Decimal</i>	<i>Hex</i>	<i>Character</i>	<i>Binary</i>	<i>Decimal</i>	<i>Hex</i>	<i>Character</i>
11000000	192	C0		11010000	208	D0	
11000001	193	C1		11010001	209	D1	
11000010	194	C2		11010010	210	D2	
11000011	195	C3		11010011	211	D3	
11000100	196	C4		11010100	212	D4	
11000101	197	C5		11010101	213	D5	
11000110	198	C6		11010110	214	D6	
11000111	199	C7		11010111	215	D7	
11001000	200	C8		11011000	216	D8	
11001001	201	C9		11011001	217	D9	
11001010	202	CA		11011010	218	DA	
11001011	203	CB		11011011	219	DB	
11001100	204	CC		11011100	220	DC	
11001101	205	CD		11011101	221	DD	
11001110	206	CE		11011110	222	DE	
11001111	207	CF		11011111	223	DF	

<i>Binary</i>	<i>Decimal</i>	<i>Hex</i>	<i>Character</i>	<i>Binary</i>	<i>Decimal</i>	<i>Hex</i>	<i>Character</i>
11100000	224	E0		11110000	240	F0	
11100001	225	E1		11110001	241	F1	
11100010	226	E2		11110010	242	F2	
11100011	227	E3		11110011	243	F3	
11100100	228	E4		11110100	244	F4	
11100101	229	E5		11110101	245	F5	
11100110	230	E6		11110110	246	F6	
11100111	231	E7		11110111	247	F7	
11101000	232	E8		11111000	248	F8	
11101001	233	E9		11111001	249	F9	
11101010	234	EA		11111010	250	FA	
11101011	235	EB		11111011	251	FB	
11101100	236	EC		11111100	252	FC	
11101101	237	ED		11111101	253	FD	
11101110	238	EE		11111110	254	FE	
11101111	239	EF		11111111	255	FF	



---

## RLE Program

---

### B.1 RLE program

---

Program B.1 is a very simple program which scans a file IN.DAT and, using RLE, stores to a file OUT.DAT. The special character sequence is:

ZZc<sub>x</sub>x

where ZZ is the flag sequence, c is the repetitive character and xx the number of times the character occurs. The ZZ flag sequence is chosen because, in a text file, it is unlikely to occur within the file. File listing B.1 shows a sample IN.DAT and File listing B.2 shows the RLE encoded file (OUT.DAT).

```
FILE Program B.1
/*      ENCODE.C      */
#include <stdio.h>
int main(void)
{
FILE *in,*out;
char previous,current;
int count;

if ((in=fopen("in.dat","r"))==NULL)
{
    printf("Cannot open <in.dat>");
    return(1);
}
if ((out=fopen("out.dat","w"))==NULL)
{
    printf("Cannot open <out.dat>");
    return(1);
}
do {
    count=1;
    previous=current;
    current=fgetc(in);
    do {
        previous=current;
        current=fgetc(in);
        if (previous!=current) ungetc(current,in);
        else count++;
    } while (previous==current);
    if (count>1) printf(out,"ZZ%c%02d",previous,count);
    else fprintf(out,"%c",previous);
} while (!feof(in));
fclose(in); fclose(out);
return(0);
}
```



### File list B.1

```
The      bbbbbbbboy stood onnnnn the burning
deck      and still did.
1.000000000
3.000000010
5.000000000
```



### File list B.2

```
TheZZ 05zzb07oy stZZo02d ozZn05 the burning
deckZZ 09and stizzl02 did.
1.ZZ009
3.ZZ00710
5.ZZ009
```

Program B.2 gives a simple C program which unencodes the RLE file produced by the previous program.



### Program B.2

```
/*      UNENCODE.C      */
#include <stdio.h>

int  main(void)
{
FILE *in,*out;
char ch;
int count,i;

if ((in=fopen("out.dat","r"))==NULL)
{
    printf("Cannot open <out.dat>");
    return(1);
}
if ((out=fopen("in1.dat","w"))==NULL)
{
    printf("Cannot open <in1.dat>");
    return(1);
}

do
{
    ch=fgetc(in);

    if (ch=='Z')
    {
        ch=fgetc(in);
        if (ch=='Z')
        {
            fscanf(in,"%c%02d",&ch,&count);
            for (i=0;i<count;i++)
                fprintf(out,"%c",ch);
        }
        else ungetc(ch,in);
    }
    else fprintf(out,"%c",ch);
}  while (!feof(in));
fclose(in);
fclose(out);
return(0);
}
```

The ZZ flag sequence is inefficient as it uses two characters to store the flag; a better flag could be an 8-bit character that cannot occur, such as 1111111b or ffh. Program B.3 shows an example of this and Program B.4 shows the decoder.

#### **Program B.3**

```
#include <stdio.h>
#define FLAG 0xff /* 1111 1111b */

int main(void)
{
FILE *in,*out;
char previous,current;
int count;

;;;;;;
    if (count>1) fprintf(out,"%c%c%02d",FLAG,previous,count);
    else fprintf(out,"%c",previous);
} while (!feof(in));
fclose(in); fclose(out); return(0);
}
```

#### **Program B.4**

```
/* UNENCODE.C */
#include <stdio.h>
#define FLAG 0xff /* 1111 1111b */

int main(void)
{
FILE *in,*out;
char ch;
int count,i;

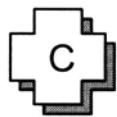
;;;;;;
do
{
    ch=fgetc(in);
    if (ch==FLAG) {
        ch=fgetc(in);
        fscanf(in,"%c%02d",&ch,&count);
        for (i=0;i<count;i++) fprintf(out,"%c",ch);
    }
    else fprintf(out,"%c",ch);
} while (!feof(in));
fclose(in);
fclose(out);
return(0);
}
```

In a binary file any bit sequence can occur. To overcome this, a flag sequence, such as 10101010 can be used to identify the flag. If this sequence occurs within the data, it will be coded with two flags two consecutive flags in the data are coded with three flags and so on.

For example: 00000000 10101010 10101010 00011100 01001100

would be encoded as: 00000000 10101010 10101010 10101010 00011100 01001100

thus when the three flags are detected, one of them is deleted.



## SNR for PCM

### C.1 SNR

If a waveform has a maximum signal amplitude of  $V$ , then the relative signal power will be:

$$\text{Signal power} = v_{rms}^2 = \left( \frac{V}{\sqrt{2}} \right)^2 = \frac{V^2}{2}$$

If  $n$ -bit PCM coding is used then there will be  $2^n$  different levels, as illustrated in Figure C.1.

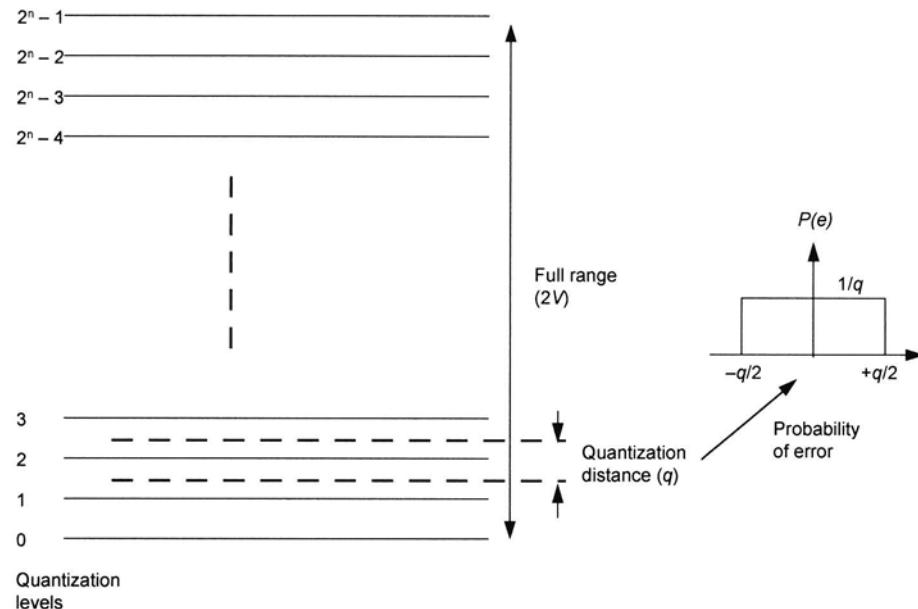


Figure C.1 Quantization.

Thus, if the input signal ranges between  $+V$  and  $-V$ , the error in the quantization signal will range from:

$$+\frac{q}{2} \text{ to } -\frac{q}{2}$$

where  $q$  is the quantization distance and is given by:

$$q = \frac{2V}{2^n}$$

Figure C.1 shows that the probability of error will be constant from between  $-q/2$  to  $+q/2$ . The area over the interval the  $P(e)$  should equal unity, hence the  $y$ -axis value for  $P(e)$  will be  $1/q$ . Thus the noise power will be:

$$\begin{aligned} \text{Noise power} &= \int_{-\infty}^{+\infty} v^2 P(v) dv = 2 \int_0^{+q/2} \frac{v^2}{q} dv \\ &= \frac{2}{q} \int_0^{+q/2} v^2 dv = \frac{2}{q} \left[ \frac{v^3}{3} \right]_0^{+q/2} = \frac{2}{3q} \left( \frac{q^3}{2^3} \right) \\ &= \frac{q^2}{12} \end{aligned}$$

The signal-to-noise ratio will thus be:

$$\begin{aligned} \text{SNR(dB)} &= 10 \log_{10} \left( \frac{\text{Signal power}}{\text{Noise power}} \right) = 10 \log_{10} \left( \frac{V^2/2}{q^2/12} \right) = 10 \log_{10} \left( \frac{6V^2}{q^2} \right) \\ &= 10 \log_{10} \left( \frac{6V^2}{(2^n)^2 / (2^n)^2} \right) = 10 \log_{10} \left( \frac{3}{2} (2^n)^2 \right) \\ &= 10 \log_{10}(1.5) + 20 \log_{10}(2^n) = 10 \log_{10}(1.5) + 20 \log_{10}(2^n) \\ &= 1.76 + 20 n \log_{10}(2) \\ &= 6.02 n + 1.76 \end{aligned}$$



---

## RFC Standards

---

The IAB (Internet Advisor Board) has published many documents on the TCP/IP protocol family. They are known as RFC (request for comment) and can be obtained using FTP from the following:

- Internet Network Information Center (NIC) at `nic.ddn.mil`, or one of several other FTP sites, such as from the InterNIC Directory and Database Services server at `ds.internic.net`.
- Through electronic mail from the automated InterNIC Directory and Database Services mail server at `mailserv@ds.internic.net`. The main body of the message should contain the command:

`document-by-name rfcNNNN`

where NNNN is the number of the RFC. Multiple requests can be made by sending a single message with each specified document separated by comma-separated list.

The main RFC documents are:

- RFC768 User Datagram Protocol.  
RFC775 Directory-Oriented FTP Commands.  
RFC781 Specification of the Internet Protocol Timestamp Option.  
RFC783 TFTP Protocol.  
RFC786 User Datagram Protocol (UDP).  
RFC791 Internet Protocol (IP).  
RFC792 Internet Control Message Protocol (ICMP).  
RFC793 Transmission Control Protocol (TCP).  
RFC799 Internet Name Domains.  
RFC813 Window and Acknowledgment in TCP.  
RFC815 IP Datagram Reassembly Algorithms.  
RFC821 Simple Mail-Transfer Protocol (SMTP).  
RFC822 Standard for the Format of ARPA Internet Text Messages.  
RFC823 DARPA Internet Gateway.  
RFC827 Exterior Gateway Protocol (EGP).  
RFC877 Standard for the Transmission of IP Datagrams over Public Data Networks.  
RFC879 TCP Maximum Segment Size and Related Topics.  
RFC886 Proposed Standard for Message Header Munging.  
RFC893 Trailer Encapsulations.  
RFC894 Standard for the Transmission of IP Datagrams over Ethernet Networks.  
RFC895 Standard for the Transmission of IP Datagrams over Experimental Ethernet Networks.  
RFC896 Congestion Control in TCP/IP Internetworks.  
RFC903 Reverse Address Resolution Protocol.  
RFC904 Exterior Gateway Protocol Formal Specifications.  
RFC906 Bootstrap Loading Using TFTP.  
RFC919 Broadcast Internet Datagram.

- RFC920 Domain Requirements.  
RFC932 Subnetwork Addressing Schema.  
RFC949 FTP Unique-Named Store Command.  
RFC950 Internet Standard Subnetting Procedure.  
RFC951 Bootstrap Protocol.  
RFC959 File Transfer Protocol.  
RFC974 Mail Routing and the Domain System.  
RFC980 Protocol Document Order Information.  
RFC1009 Requirements for Internet Gateways.  
RFC1011 Official Internet Protocol.  
RFC1013 X Windows System Protocol.  
RFC1014 XDR: External Data Representation Standard.  
RFC1027 Using ARP to Implement Transparent Subnet Gateways.  
RFC1032 Domain Administrators Guide.  
RFC1033 Domain Administrators Operation Guide.  
RFC1034 Domain Names - Concepts and Facilities.  
RFC1035 Domain Names - Implementation and Specifications.  
RFC1041 Telnet 3270 Regime Option.  
RFC1042 Standard for the Transmission of IP Datagrams over IEEE 802 Networks.  
RFC1043 Telnet Data Entry Terminal Option.  
RFC1044 Internet Protocol on Network System's HYPERchannel.  
RFC1053 Telnet X.3 PAD Option.  
RFC1055 Nonstandard for Transmission of IP Datagrams over Serial Lines.  
RFC1056 PCMAIL: A Distributed Mail System for Personal Computers.  
RFC1058 Routing Information Protocol.  
RFC1068 Background File Transfer Program (BFTP).  
RFC1072 TCP Extensions of Long-Delay Paths.  
RFC1073 Telnet Window Size Option.  
RFC1074 NSFNET Backbone SPF-based Interior Gateway Protocol.  
RFC1079 Telnet Terminal Speed Option.  
RFC1080 Telnet Remote Flow Control Option.  
RFC1084 BOOTP Vendor Information Extensions.  
RFC1088 Standard for the Transmission of IP Datagrams over NetBIOS Network.  
RFC1089 SNMP over Ethernet.  
RFC1091 Telnet Terminal-Type Option.  
RFC1094 NFS: Network File System Protocol Specification.  
RFC1101 DNS Encoding of Network Names and Other Types.  
RFC1102 Policy Routing in Internet Protocols.  
RFC1104 Models of Policy-Based Routing.  
RFC1112 Host Extension for IP Multicasting.  
RFC1122 Requirement for Internet Hosts - Communication Layers.  
RFC1123 Requirement for Internet Hosts - Application and Support.  
RFC1124 Policy Issues in Interconnecting Networks.  
RFC1125 Policy Requirements for Inter-Administrative Domain Routing.  
RFC1127 Perspective on the Host Requirements RFC.  
RFC1129 Internet Time Protocol.  
RFC1143 Q Method of Implementing Telnet Option Negotiation.  
RFC1147 FYI on a Network Management Tool Catalog.  
RFC1149 Standard for the Transmission of IP Datagrams over Avian Carriers.  
RFC1155 Structure and Identification of Management Information for TCP/IP-Based Internets.  
RFC1156 Management Information Base for Network Management of TCP/IP-Based Internets.  
RFC1157 Simple Network Management Protocol (SNMP).  
RFC1163 Border Gateway Protocol (BGP).

- RFC1164 Application of the Border Gateway Protocol in the Internet.
- RFC1166 Internet Numbers.
- RFC1171 Point-to-Point Protocol for the Transmission of Multi-Protocol Datagrams.
- RFC1172 Point-to-Point Protocol Initial Configuration Options.
- RFC1173 Responsibilities of Host and Network Managers.
- RFC1175 FYI on Where to Start: A Bibliography of Internetworking Information.
- RFC1178 Choosing a Name For Your Computer.
- RFC1179 Line Printer Daemon Protocol.
- RFC1184 Telnet Linemode Option.
- RFC1187 Bulk Table Retrieval with the SNMP.
- RFC1188 Proposed Standard for the Transmission of TP Datagrams over FDDI Networks.
- RFC1195 Use of OSI IS-IS for Routing in TCP/IP and Dual Environments.
- RFC1196 Finger User Information Protocol.
- RFC1198 FYI on the X Windows System.
- RFC1201 Transmitting IP Traffic over ARCNET Networks.
- RFC1205 520 Telnet Interface.
- RFC1208 Glossary of Networking Terms.
- RFC1209 Transmission of IP Datagrams over the SMDS Service.
- RFC1212 Concise MIB Definitions.
- RFC1213 MIB for Network Management of TCP/IP-Based Internets.
- RFC1214 OSI Internet Management: Management Information Base.
- RFC1215 Convention for Defining Traps for Use with the SNMP.
- RFC1219 On the Assignment of Subnet Numbers.
- RFC1220 Point-to-Point Protocol Extensions for Bridges.
- RFC1224 Techniques for Managing Asynchronous Generated Alerts.
- RFC1227 SNMP MUX Protocol and MIB.
- RFC1228 SNMP-DPI: Simple Network Management Protocol Distributed Program Interface.
- RFC1229 Extensions to the Generic-interface MIB.
- RFC1230 IEEE 802.4 Token Bus MIB.
- RFC1231 IEEE 802.5 Token Ring MIB.
- RFC1232 Definitions of Managed Objects for the DS1 Interface Type.
- RFC1233 Definitions of Managed Objects for the DS3 Interface Type.
- RFC1236 IP to X.121 Address Mapping for DDN IP.
- RFC1238 CLNS MIB for Use with Connectionless Network Protocol.
- RFC1239 Reassignment of Experiment MIBs to Standard MIBs.
- RFC1243 Appletalk Management Information Base.
- RFC1245 OSPF Protocol Analysis.
- RFC1246 Experience with the OSPF Protocol.
- RFC1247 OSPF Version 2.
- RFC1253 OSPF Version 2: Management Information Base.
- RFC1254 Gateway Congestion Control Survey.
- RFC1267 A Border Gateway Protocol (BGP-3).
- RFC1271 Remote Network Monitoring Management Information Base.
- RFC1321 The MD5 Message-Digest Algorithm.
- RFC1340 Assigned Numbers.
- RFC1341 MIME Mechanism for Specifying and Describing the Format of Internet Message Bodies.
- RFC1360 IAB Official Protocol Standards.
- RFC1522 MIME (Multipurpose Internet Mail Extensions)Part Two : Message Header Extensions for Non-ASCII Text.
- RFC1521 MIME (Multipurpose Internet Mail Extensions) Part One : Mechanisms for Specifying and Describing the Format of Internet Mail Message Bodies.
- RFC1583 OSPF Version 2.
- RFC1630 Universal Resource Identifiers in WWW.

- RFC1738 Uniform Resource Identifiers (URL).
- RFC1752 The Recommendation for the IP Next-Generation Protocol.
- RFC1771 A Border Gateway Protocol 4 (BGP-4).
- RFC1808 Relative Uniform Resource Identifiers.
- RFC1809 Using the Flow Label in IPv6.
- RFC1825 Security Architecture for the Internet Protocol.
- RFC1826 IP Authentication Header.
- RFC1827 IP Encapsulating Security Payload (ESP).
- RFC1828 IP Authentication Using Keyed MD5.
- RFC1829 The ESP DES-CBC Transform.
- RFC1883 Internet Protocol, Version 6 Specification.
- RFC1884 IP Version 6 Addressing Architecture.
- RFC1885 Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version-6 (IPv6) Specification.
- RFC1886 DNS Extensions to Support IP Version 6.
- RFC1887 An Architecture for IPv6 Unicast Address Allocation.
- RFC1901 Introduction to Community-Based SNMPv2.
- RFC1902 Structure of Management Information for SNMPv2.
- RFC1903 Textual Conventions for SNMPv2.
- RFC1904 Conformance Statements for SNMPv2.
- RFC1905 Protocol Operations for SNMPv2.
- RFC1906 Transport Mappings for SNMPv2.
- RFC1907 Management Information Base for SNMPv2.
- RFC1908 Coexistence Between Version 1 and Version 2 of the Internet-Standard Network Management Framework.
- RFC1909 An Administrative Infrastructure for SNMPv2.
- RFC1910 User-based Security Model for SNMPv2.
- RFC1911 Voice Profile for Internet Mail.
- RFC1912 Common DNS Operational and Configuration Errors.
- RFC1913 Architecture of the Whois++ Index Service.
- RFC1914 How to Interact with a Whois++ Mesh.
- RFC1915 Variance for The PPP Connection Control Protocol and The PPP Encryption Control Protocol
- RFC1916 Enterprise Renumbering: Experience and Information Solicitation.
- RFC1917 An Appeal to the Internet Community to Return Unused IP Networks (Prefixes) to the IANA
- RFC1918 Address Allocation for Private Internets.
- RFC1919 Classical versus Transparent IP Proxies.
- RFC1920 INTERNET OFFICIAL PROTOCOL STANDARDS.
- RFC1922 Chinese Character Encoding for Internet Messages.
- RFC1923 RIPv1 Applicability Statement for Historic Status.
- RFC1924 A Compact Representation of IPv6 Addresses.
- RFC1925 The Twelve Networking Truths.
- RFC1926 An Experimental Encapsulation of IP Datagrams on Top of ATM.
- RFC1927 Suggested Additional MIME Types for Associating Documents.
- RFC1928 SOCKS Protocol Version 5.
- RFC1929 Username/Password Authentication for SOCKS V5.
- RFC1930 Guidelines for creation, selection, and registration of an Autonomous System (AS).
- RFC1931 Dynamic RARP Extensions for Automatic Network Address Acquisition.
- RFC1932 IP over ATM: A Framework Document.
- RFC1933 Transition Mechanisms for IPv6 Hosts and Routers.
- RFC1934 Ascend's Multilink Protocol Plus (MP+).
- RFC1935 What is the Internet, Anyway?

- RFC1936 Implementing the Internet Checksum in Hardware.
- RFC1937 “Local/Remote” Forwarding Decision in Switched Data Link Subnetworks.
- RFC 1938 A One-Time Password System.
- RFC 1939 Post Office Protocol - Version 3.
- RFC 1940 Source Demand Routing: Packet Format and Forwarding Specification (Version 1).
- RFC 1941 Frequently Asked Questions for Schools.
- RFC1942 HTML Tables.
- RFC1943 Building an X.500 Directory Service in the US.
- RFC1944 Benchmarking Methodology for Network Interconnect Devices.
- RFC1945 Hypertext Transfer Protocol -- HTTP/1.0.
- RFC1946 Native ATM Support for ST2+.
- RFC1947 Greek Character Encoding for Electronic Mail Messages.
- RFC1948 Defending Against Sequence Number Attacks.
- RFC1949 Scalable Multicast Key Distribution.
- RFC1950 ZLIB Compressed Data Format Specification version 3.3.
- RFC1951 DEFLATE Compressed Data Format Specification version 1.3.
- RFC1952 GZIP file format specification version 4.3.
- RFC1953 Ipsilon Flow Management Protocol Specification for IPv4 Version 1.0.
- RFC1954 Transmission of Flow Labelled IPv4 on ATM Data Links Ipsilon Version 1.0.
- RFC1955 New Scheme for Internet Routing and Addressing (ENCAPS) for IPNG.
- RFC1956 Registration in the MIL Domain.
- RFC1957 Some Observations on Implementations of the Post Office Protocol (POP3).
- RFC1958 Architectural Principles of the Internet.
- RFC1959 An LDAP URL Format.
- RFC1960 A String Representation of LDAP Search Filters.
- RFC1961 GSS-API Authentication Method for SOCKS Version 5.
- RFC1962 The PPP Compression Control Protocol (CCP).
- RFC1963 PPP Serial Data Transport Protocol (SDTP).
- RFC1964 The Kerberos Version 5 GSS-API Mechanism.
- RFC1965 Autonomous System Confederations for BGP.
- RFC1966 BGP Route Reflection An alternative to full mesh IBGP.
- RFC1967 PPP Lzs-Dcp Compression Protocol (Lzs-DCP).
- RFC1968 The PPP Encryption Control Protocol (ECP).
- RFC1969 The PPP DES Encryption Protocol (DESE).
- RFC1970 Neighbor Discovery for IP Version 6 (IPv6).
- RFC1971 IPv6 Stateless Address Autoconfiguration.
- RFC1972 A Method for the Transmission of IPv6 Packets over Ethernet Networks.
- RFC1973 PPP in Frame Relay.
- RFC1974 PPP Stac Lzs Compression Protocol.
- RFC1975 PPP Magnalink Variable Resource Compression.
- RFC1976 PPP for Data Compression in Data Circuit-Terminating Equipment (DCE).
- RFC1977 PPP BSD Compression Protocol.
- RFC1978 PPP Predictor Compression Protocol.
- RFC1979 PPP Deflate Protocol.
- RFC1980 A Proposed Extension to HTML : Client-Side Image Maps.
- RFC1981 Path MTU Discovery for IP version 6.
- RFC1982 Serial Number Arithmetic.
- RFC1983 Internet Users’ Glossary.
- RFC1984 IAB and IESG Statement on Cryptographic Technology and the Internet.
- RFC1985 SMTP Service Extension for Remote Message Queue Starting.
- RFC1986 Experiments with a Simple File Transfer Protocol for Radio Links using Enhanced Trivial File Transfer Protocol (ETFTP).
- RFC1987 Ipsilon’s General Switch Management Protocol Specification Version 1.1.

- RFC1988 Conditional Grant of Rights to Specific Hewlett-Packard Patents In Conjunction With the Internet Engineering Task Force's Internet-Standard Network Management Framework.
- RFC1989 PPP Link Quality Monitoring.
- RFC1990 The PPP Multilink Protocol.
- RFC1991 PGP Message Exchange Formats.
- RFC1992 The Nimrod Routing Architecture.
- RFC1993 PPP Gandalf FZA Compression Protocol.
- RFC1994 PPP Challenge Handshake Authentication Protocol (CHAP).
- RFC1995 Incremental Zone Transfer in DNS.
- RFC1996 A Mechanism for Prompt Notification of Zone Changes.
- RFC1997 BGP Communities Attribute.
- RFC1998 An Application of the BGP Community Attribute in Multi-home Routing.
- RFC1999 Request for Comments Summary RFC Numbers 1900-1999.
- RFC2000 INTERNET OFFICIAL PROTOCOL STANDARDS.
- RFC2001 TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast
- RFC2002 IP Mobility Support.
- RFC2003 IP Encapsulation within IP.
- RFC2004 Minimal Encapsulation within IP.
- RFC2005 Applicability Statement for IP Mobility Support.
- RFC2006 The Definitions of Managed Objects for IP Mobility Support using SMIv2.
- RFC2007 Catalogue of Network Training Materials.
- RFC2008 Implications of Various Address Allocation Policies for Internet Routing.
- RFC2009 GPS-Based Addressing and Routing.
- RFC2010 Operational Criteria for Root Name Servers.
- RFC2011 SNMPv2 Management Information Base for the Internet Protocol using SMIv2.
- RFC2012 SNMPv2 Management Information Base for the Transmission Control Protocol using SMIv2.
- RFC2013 SNMPv2 Management Information Base for the User Datagram Protocol using SMIv2.
- RFC2014 IRTF Research Group Guidelines and Procedures.
- RFC2015 MIME Security with Pretty Good Privacy (PGP).
- RFC2016 Uniform Resource Agents (URAs).
- RFC2017 Definition of the URL MIME External-Body Access-Type.
- RFC2018 TCP Selective Acknowledgement Options.
- RFC2019 Transmission of IPv6 Packets Over FDDI.
- RFC2020 IEEE 802.12 Interface MIB.
- RFC2021 Remote Network Monitoring Management Information Base Version 2 using SMIv2.
- RFC2022 Support for Multicast over UNI 3.0/3.1 based ATM Networks.
- RFC2023 IP Version 6 over PPP.
- RFC2024 Definitions of Managed Objects for Data Link Switching using SMIv2.
- RFC2025 The Simple Public-Key GSS-API Mechanism (SPKM).
- RFC2026 The Internet Standards Process -- Revision 3.
- RFC2027 IAB and IESG Selection, Confirmation, and Recall Process: Operation of the Nominating and Recall Committees.
- RFC2028 The Organizations Involved in the IETF Standards Process.
- RFC2029 RTP Payload Format of Sun's CellB Video Encoding.
- RFC2030 Simple Network Time Protocol (SNTP).
- RFC2031 IETF-ISOC relationship.
- RFC2032 RTP Payload Format for H.261 Video Streams.
- RFC2033 Local Mail Transfer Protocol.
- RFC2034 SMTP Service Extension for Returning Enhanced Error Codes.
- RFC2035 RTP Payload Format for JPEG-compressed Video.
- RFC2036 Observations on the use of Components of the Class A Address Space within the Internet.
- RFC2037 Entity MIB using SMIv2.

- RFC2038 RTP Payload Format for MPEG1/MPEG2 Video.
- RFC2039 Applicability of Standards Track MIBs to Management of World Wide Web Servers.
- RFC2040 The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms.
- RFC2041 Mobile Network Tracing.
- RFC2042 Registering New BGP Attribute Types.
- RFC2043 The PPP SNA Control Protocol (SNACP).
- RFC2044 UTF-8, a transformation format of Unicode and ISO 10646.
- RFC2045 Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies.
- RFC2046 Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types.
- RFC2047 MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text.
- RFC2048 Multipurpose Internet Mail Extension (MIME) Part Four: Registration Procedures.
- RFC2049 Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples.
- RFC2050 INTERNET REGISTRY IP ALLOCATION GUIDELINES.
- RFC2051 Definitions of Managed Objects for APPC using SMIv2.
- RFC2052 A DNS RR for specifying the location of services (DNS SRV).
- RFC2053 The AM (Armenia) Domain.
- RFC2054 WebNFS Client Specification.
- RFC2055 WebNFS Server Specification.
- RFC2056 Uniform Resource Locators for Z39.50.
- RFC2057 Source Directed Access Control on the Internet.
- RFC2058 Remote Authentication Dial In User Service (RADIUS).
- RFC2059 RADIUS Accounting.
- RFC2060 INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1.
- RFC2061 IMAP4 COMPATIBILITY WITH IMAP2BIS.
- RFC2062 Internet Message Access Protocol - Obsolete Syntax.
- RFC2063 Traffic Flow Measurement: Architecture.
- RFC2064 Traffic Flow Measurement: Meter MIB.
- RFC2065 Domain Name System Security Extensions.
- RFC2066 TELNET CHARSET Option.
- RFC2067 IP over HIPPI.
- RFC2068 Hypertext Transfer Protocol -- HTTP/1.1.
- RFC2069 An Extension to HTTP: Digest Access Authentication.
- RFC2070 Internationalization of the Hypertext Markup Language.
- RFC2071 Network Renumbering Overview: Why would I want it and what is it anyway?.
- RFC2072 Router Renumbering Guide.
- RFC2073 An IPv6 Provider-Based Unicast Address Format.
- RFC2074 Remote Network Monitoring MIB Protocol Identifiers.
- RFC2075 IP Echo Host Service.
- RFC2076 Common Internet Message Headers.
- RFC2077 The Model Primary Content Type for Multipurpose Internet Mail Extensions.
- RFC2078 Generic Security Service Application Program Interface, Version 2.
- RFC2079 Definition of an X.500 Attribute Type and an Object Class to Hold Uniform Resource Identifiers (URIs).
- RFC2080 RIPng for IPv6.
- RFC2081 RIPng Protocol Applicability Statement.
- RFC2082 RIP-2 MD5 Authentication.
- RFC2083 PNG (Portable Network Graphics) Specification.
- RFC2084 Considerations for Web Transaction Security.
- RFC2085 HMAC-MD5 IP Authentication with Replay Prevention.
- RFC2086 IMAP4 ACL extension.

- RFC2087 IMAP4 QUOTA extension.
- RFC2088 IMAP4 non-synchronizing literals.
- RFC2089 V2ToV1 Mapping SNMPv2 onto SNMPv1 within a bi-lingual SNMP agent.
- RFC2090 TFTP Multicast Option.
- RFC2091 Triggered Extensions to RIP to Support Demand Circuits.
- RFC2092 Protocol Analysis for Triggered RIP.
- RFC2093 Group Key Management Protocol (GKMP) Specification.
- RFC2094 Group Key Management Protocol (GKMP) Architecture.
- RFC2095 IMAP/POP AUTHorize Extension for Simple Challenge/Response.
- RFC2096 IP Forwarding Table MIB.
- RFC2097 The PPP NetBIOS Frames Control Protocol (NBFCP).
- RFC2098 Toshiba's Router Architecture Extensions for ATM : Overview.
- RFC2099 Request for Comments Summary RFC Numbers 2000-2099.
- RFC2100 The Naming of Hosts.
- RFC2101 IPv4 Address Behavior Today.
- RFC2102 Multicast Support for Nimrod : Requirements and Solution Approaches.
- RFC2103 Mobility Support for Nimrod : Challenges and Solution Approaches.
- RFC2104 HMAC: Keyed-Hashing for Message Authentication.
- RFC2105 Cisco Systems' Tag Switching Architecture Overview.
- RFC2106 Data Link Switching Remote Access Protocol.
- RFC2107 Ascend Tunnel Management Protocol - ATMP.
- RFC2108 Definitions of Managed Objects for IEEE 802.3 Repeater Devices using SMIv2.
- RFC2109 HTTP State Management Mechanism.
- RFC2110 MIME E-mail Encapsulation of Aggregate Documents, such as HTML (MHTML).
- RFC2111 Content-ID and Message-ID Uniform Resource Locators.
- RFC2112 The MIME Multipart/Related Content-type.
- RFC2113 IP Router Alert Option.
- RFC2114 Data Link Switching Client Access Protocol.
- RFC2115 Management Information Base for Frame Relay DTEs Using SMIv2.
- RFC2116 X.500 Implementations Catalog-96.
- RFC2117 Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol
- RFC2118 Microsoft Point-To-Point Compression (MPPC) Protocol.
- RFC2119 Key words for use in RFCs to Indicate Requirement Level.
- RFC2120 Managing the X.500 Root Naming Context.
- RFC2121 Issues affecting MARS Cluster Size.
- RFC2122 VEMMI URL Specification.
- RFC2123 Traffic Flow Measurement: Experiences with NeTraMet.
- RFC2124 Cabletron's Light-weight Flow Admission Protocol Specification.
- RFC2125 The PPP Bandwidth Allocation Protocol (BAP) / The PPP Bandwidth Allocation Control Protocol (BACP).
- RFC2126 ISO Transport Service on top of TCP (ITOT).
- RFC2127 ISDN Management Information Base using SMIv2.
- RFC2128 Dial Control Management Information Base using SMIv2.
- RFC2129 Toshiba's Flow Attribute Notification Protocol (FANP).
- RFC2130 The Report of the IAB Character Set Workshop held 29 February - 1 March, 1996.
- RFC2131 Dynamic Host Configuration Protocol.
- RFC2132 DHCP Options and BOOTP Vendor Extensions.
- RFC2133 Basic Socket Interface Extensions for Ipv6.
- RFC2134 Articles of Incorporation of Internet Society.
- RFC2135 Internet Society By-Laws. ISOC Board of Trustees.
- RFC2136 Dynamic Updates in the Domain Name System (DNS UPDATE).
- RFC2137 Secure Domain Name System Dynamic Update.
- RFC2138 Remote Authentication Dial In User Service (RADIUS).

- RFC2139 RADIUS Accounting.
- RFC2140 TCP Control Block Interdependence.
- RFC2141 URN Syntax.
- RFC2142 Mailbox Names for Common Services, Roles and Functions.
- RFC2143 Encapsulating IP with the Small Computer System Interface.
- RFC2145 Use and Interpretation of HTTP Version Numbers.
- RFC2146 U.S. Government Internet Domain Names. Federal Networking
- RFC2147 TCP and UDP over IPv6 Jumbograms.
- RFC2148 Deployment of the Internet White Pages Service.
- RFC2149 Multicast Server Architectures for MARS-based ATM multicasting.
- RFC2150 Humanities and Arts: Sharing Center Stage on the Internet.
- RFC2151 A Primer On Internet and TCP/IP Tools and Utilities.
- RFC2152 UTF-7 A Mail-Safe Transformation Format of Unicode.
- RFC2153 PPP Vendor Extensions.
- RFC2154 OSPF with Digital Signatures.
- RFC2155 Definitions of Managed Objects for APPN using SMIv2.
- RFC2165 Service Location Protocol.
- RFC2166 APPN Implementer's Workshop Closed Pages Document DLSw v2.0 Enhancements.
- RFC2167 Referral Whois (RWhois) Protocol V1.5.
- RFC2168 Resolution of Uniform Resource Identifiers using the Domain Name System.
- RFC2169 A Trivial Convention for using HTTP in URN Resolution.
- RFC2170 Application REQuested IP over ATM (AREQUIPA).
- RFC2171 MAPOS - Multiple Access Protocol over SONET/SDH Version 1.
- RFC2172 MAPOS Version 1 Assigned Numbers.
- RFC2173 A MAPOS version 1 Extension - Node Switch Protocol.
- RFC2174 A MAPOS version 1 Extension - Switch-Switch Protocol.
- RFC2175 MAPOS 16 - Multiple Access Protocol over SONET/SDH with 16 Bit Addressing.
- RFC2176 IPv4 over MAPOS Version 1
- RFC2177 IMAP4 IDLE command.
- RFC2178 OSPF Version 2.
- RFC2179 Network Security For Trade Shows.
- RFC2180 IMAP4 Multi-Accessed Mailbox Practice.
- RFC2181 Clarifications to the DNS Specification.
- RFC2182 Selection and Operation of Secondary DNS Servers.
- RFC2183 Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field.
- RFC2184 MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations.
- RFC2185 Routing Aspects of IPv6 Transition.
- RFC2186 Internet Cache Protocol (ICP), version 2.
- RFC2187 Application of Internet Cache Protocol (ICP), version 2.
- RFC2188 AT&T/Neda's Efficient Short Remote Operations (ESRO) Protocol Specification Version 1.2.
- RFC2189 Core Based Trees (CBT version 2) Multicast Routing.
- RFC2190 RTP Payload Format for H.263 Video Streams.
- RFC2191 VENUS - Very Extensive Non-Unicast Service.
- RFC2192 IMAP URL Scheme.
- RFC2193 IMAP4 Mailbox Referrals.
- RFC2194 Review of Roaming Implementations.
- RFC2195 IMAP/POP AUTHorize Extension for Simple Challenge/Response.
- RFC2196 Site Security Handbook.
- RFC2197 SMTP Service Extension for Command Pipelining.
- RFC2198 RTP Payload for Redundant Audio Data.

- RFC2200 INTERNET OFFICIAL PROTOCOL STANDARDS.
- RFC2201 Core Based Trees (CBT) Multicast Routing Architecture.
- RFC2202 Test Cases for HMAC-MD5 and HMAC-SHA-1.
- RFC2203 RPCSEC\_GSS Protocol Specification.
- RFC2204 ODETTE File Transfer Protocol.
- RFC2205 Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification.
- RFC2206 RSVP Management Information Base using SMIv2.
- RFC2207 RSVP Extensions for IPSEC Data Flows.
- RFC2208 Resource ReSerVation Protocol (RSVP) -- Version 1 Applicability Statement Some Guidelines on Deployment.
- RFC2209 Resource ReSerVation Protocol (RSVP) -- Version 1 Message Processing Rules.
- RFC2210 The Use of RSVP with IETF Integrated Services.
- RFC2211 Specification of the Controlled-Load Network Element Service.
- RFC2212 Specification of Guaranteed Quality of Service.
- RFC2213 Integrated Services Management Information Base using SMIv2.
- RFC2214 Integrated Services Management Information Base Guaranteed Service Extensions using SMIv2.
- RFC2215 General Characterization Parameters for Integrated Service Network Elements.
- RFC2216 Network Element Service Specification Template.
- RFC2217 Telnet Com Port Control Option.
- RFC2218 A Common Schema for the Internet White Pages Service.
- RFC2219 Use of DNS Aliases for Network Services.
- RFC2220 The Application/MARC Content-type.
- RFC2221 IMAP4 Login Referrals.
- RFC2222 Simple Authentication and Security Layer (SASL).
- RFC2223 Instructions to RFC Authors.
- RFC2224 NFS URL Scheme.
- RFC2226 IP Broadcast over ATM Networks.
- RFC2227 Simple Hit-Metering and Usage-Limiting for HTTP.
- RFC2228 FTP Security Extensions.
- RFC2229 A Dictionary Server Protocol.
- RFC2230 Key Exchange Delegation Record for the DNS.
- RFC2231 MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations.
- RFC2232 Definitions of Managed Objects for DLUR using SMIv2.
- RFC2233 The Interfaces Group MIB using SMIv2.
- RFC2234 Augmented BNF for Syntax Specifications: ABNF.
- RFC2235 Hobbes' Internet Timeline.
- RFC2236 Internet Group Management Protocol, Version 2.
- RFC2237 Japanese Character Encoding for Internet Messages.
- RFC2238 Definitions of Managed Objects for HPR using SMIv2.
- RFC2239 Definitions of Managed Objects for IEEE 802.3 Medium Attachment Units (MAUs) using SMIv2.
- RFC2240 A Legal Basis for Domain Name Allocation.
- RFC2241 DHCP Options for Novell Directory Services.
- RFC2242 NetWare/IP Domain Name and Information.



---

# UNIX Network Startup Files

---

## E.1 netnfsrc file

---

This appendix documents a typical netnfsrc (NFS startup file) file. In the script portion given below the `NFS_CLIENT` is set to a 1 if the host is set to a client (else it will be 0) and the `NFS_SERVER` parameter is set to a 1 if the host is set to a server (else it will be 0). Initially the NFS clients and servers are started. Note that a host can be a client, a server, both or neither.

Next the `mountd` daemon is started, after which the NFS daemons (`nfsd`) are started (only on servers). After this the `biod` daemon is run.

---

```
NFS_CLIENT=1
NFS_SERVER=1
START_MOUNTD=0
#      Read in /etc/exports
##
if [ $LFS -eq 0 -a $NFS_SERVER -ne 0 -a -f /etc/exports ] ; then
    > /etc/xtab
    /usr/etc/exportfs -a  && echo "      Reading in /etc/exports"
    set_return
fi

if [ $NFS_SERVER -ne 0 -a $START_MOUNTD -ne 0 -a -f /usr/etc/rpc.mountd ] ;
then
    /usr/etc/rpc.mountd && echo "starting up the mountd" && echo
                                "\t/usr/etc/rpc.mountd"
    set_return
fi
##
if [ $LFS -eq 0 -a $NFS_SERVER -ne 0 -a -f /etc/nfsd ] ; then
    /etc/nfsd 4 && echo "starting up the NFS daemons" && echo "\t/etc/nfsd 4"
    set_return
fi
##
if [ $NFS_CLIENT -ne 0 ] ; then
    if [ -f /etc/biod ] ; then
        /etc/biod 4 && echo
            "starting up the BIO daemons" && echo "\t/etc/biod 4"
        set_return
    fi
    /bin/cat /dev/null > /etc/nfs.up
fi
```

---

The next part of the `netnfsrc` file deals with the NIS services. There are three states: `NIS_MASTER_SERVER`, `NIS_SLAVE_SERVER` and `NIS_CLIENT`. A host can either be a master server or a slave server, but cannot be both. All NIS servers must also be NIS clients, so the `NIS_MASTER_SERVER` or `NIS_SLAVE_SERVER` parameters shoud be set to 1. Initally the domain name is set using the command `domainname` (in this case it is `eece`).

---

```
NIS_MASTER_SERVER=1
NIS_SLAVE_SERVER=0
NIS_CLIENT=1
NISDOMAIN=eece
NISDOMAIN_ERR=""

if [ "$NISDOMAIN" -a -f /bin/domainname ] ; then
    echo "\t/bin/domainname $NISDOMAIN"
    /bin/domainname $NISDOMAIN
    if [ $? -ne 0 ] ; then
        echo "Error: NIS domain name not set" >&2
        NISDOMAIN_ERR=TRUE
    fi
else
    echo "\tNIS domain name not set"
    NISDOMAIN_ERR=TRUE
fi
```

---

Next portmap is started for ARPA clients.

---

```
if [ -f /etc/portmap ] ; then
    echo "\t/etc/portmap"
    /etc/portmap
    if [ $? -ne 0 ] ; then
        echo "Error: NFS portmapper NOT powered up" >&2
        exit 1
    fi
fi
```

---

Next the NIS is started.

---

```
if [ "$NISDOMAIN_ERR" -o \( $NIS_MASTER_SERVER -eq 0 -a $NIS_SLAVE_SERVER -eq
0 \
-a $NIS_CLIENT -eq 0 \) ] ; then
    echo "      Network Information Service not started."
else
    echo "      starting up the Network Information Service"

    HOSTNAME=`hostname`

    if [ $NIS_MASTER_SERVER -ne 0 -o $NIS_SLAVE_SERVER -ne 0 ]; then
        NIS_SERVER=TRUE
    fi

    if [ $NIS_MASTER_SERVER -ne 0 -a $NIS_SLAVE_SERVER -ne 0 ]; then
        echo "NOTICE:both NIS_MASTER_SERVER and NIS_SLAVE_SERVER variables set;"
        echo "\t$HOSTNAME will be only a NIS slave server."
        NIS_MASTER_SERVER=0
    fi

    if [ $NIS_CLIENT -eq 0 ]; then
        echo "NOTICE:$HOSTNAME will be a NIS server, but the NIS_CLIENT variable is"
        echo "\tnot set; $HOSTNAME will also be a NIS client."
        NIS_CLIENT=1
    fi
```

---

Next the yp services are started.

---

```

# The verify_ypserv function determines if it is OK to start ypserv(1M)
# (and yppasswdd(1M) for the master NIS server). It returns its result
# in the variable NISSERV_OK - if non-null, it is OK to start ypserv(1M);
# if it is null, ypserv(1M) will not be started.
#
# First, the filesystem containing /usr/etc/yp is examined to see if it
# supports long or short filenames. Once this is known, the proper list
# of standard NIS map filenames is examined to verify that each map exists
# in the NIS domain subdirectory. If any map is missing, verify_ypserv
# sets NISSERV_OK to null and returns.
##

verify_ypserv() {
    ##
    # LONGNAMES are the names of the NIS maps on a filesystem that
    # supports long filenames.
    ##

    LONGNAMES="group.bygid.dir group.bygid.pag group.bynam.dir \
              group.bynam.pag hosts.byaddr.dir hosts.byaddr.pag \
              hosts.bynam.dir hosts.bynam.pag networks.byaddr.dir \
              networks.byaddr.pag networks.bynam.dir networks.bynam.pag \
              passwd.bynam.dir passwd.bynam.pag passwd.byuid.dir \
              passwd.byuid.pag protocols.bynam.dir protocols.bynam.pag \
              protocols.bynumber.dir protocols.bynumber.pag \
              rpc.bynumber.dir rpc.bynumber.pag services.bynam.dir \
              services.bynam.pag ypservers.dir ypservers.pag"

    ##
    # SHORTNAMES are the names of the NIS maps on a filesystem that
    # supports only short filenames (14 characters or less).
    ##

    SHORTNAMES="group.bygi.dir group.bygi.pag group.byna.dir \
               group.byna.pag hosts.byad.dir hosts.byad.pag \
               hosts.byna.dir hosts.byna.pag netwk.byad.dir \
               netwk.byad.pag netwk.byna.dir netwk.byna.pag \
               passw.byna.dir passw.byna.pag passw.byui.dir \
               passw.byui.pag proto.byna.dir proto.byna.pag \
               proto.bynu.dir proto.bynu.pag rpc.bynu.dir \
               rpc.bynu.pag servi.byna.dir servi.byna.pag \
               ypservers.dir ypservers.pag"

    NISSERV_OK=TRUE

    if `/usr/etc/yp/longfiles`; then
        NAMES=$LONGNAMES
    else
        NAMES=$SHORTNAMES
    fi

    for NAME in $NAMES ; do
        if [ ! -f /usr/etc/yp/$NISDOMAIN/$NAME ] ; then
            NISSERV_OK=
            return
        fi
    done
}

```

---

Next ypserv and ypbind are started.

---

```
if [ "$NIS_SERVER" -a -f /usr/etc/ypserv ] ; then
    verify_ypserv
    if [ "$NISSERV_OK" ] ; then
        /usr/etc/ypserv && echo "\t/usr/etc/ypserv"
            set_return
    else
        echo "\tWARNING: /usr/etc/ypserv not started: either"
        echo "\t      - the directory /usr/etc/yp/$NISDOMAIN does not ex-
ist or"
        echo "\t      - some or all of the $NISDOMAIN NIS domain's"
        echo "\t            maps are missing."
        echo "\tTo initialize $HOSTNAME as a NIS server, see ypinit(1M)."
        returnstatus=1
    fi
    fi
    if [ $NIS_CLIENT -ne 0 -a -f /etc/ypbind ] ; then
        /etc/ypbind && echo "\t/etc/ypbind"
            set_return

        ##
        #   check if the NIS domain is bound. If not disable NIS
        ##
        CNT=0;
        MAX_NISCHECKS=2
        NIS_CHECK=YES
        echo " Checking NIS binding."
        while [ ${CNT} -le ${MAX_NISCHECKS} -a "${NIS_CHECK}" = "YES" ]; do
            /usr/bin/ypwhich 2>&1 | /bin/fgrep 'not bound ypwhich' > /dev/null

        if [ $? -eq 0 ] ; then
            CNT=`expr $CNT + 1`
            if [ ${CNT} -le 2 ] ; then
                sleep 5
            else
                echo " Unable to bind to NIS server using domain ${NISDOMAIN}."
                echo " Disabling NIS"
                /bin/domainname ""
                /bin/ps -e | /bin/grep ypbnd | \
                    kill -15 `/usr/bin/awk '{ print $1 }'`'
            NIS_CHECK=NO
            returnstatus=1
            break;
        fi
        else
            echo " Bound to NIS server using domain ${NISDOMAIN}."
            NIS_CHECK=NO
        fi
        done
    fi

    ##
    if [ $NIS_MASTER_SERVER -ne 0 -a -f /usr/etc/rpc.yppasswdd ] ; then
        if [ "$NISSERV_OK" ] ; then
            echo "\t/usr/etc/rpc.yppasswdd"
            /usr/etc/rpc.yppasswdd /etc/passwd -m passwd PWFFILE=/etc/passwd
            set_return
        else
            echo "\tWARNING: /usr/etc/rpc.yppasswdd not started: refer to the"
            echo "\t      reasons listed in the WARNING above."
            returnstatus=1
        fi
    fi
fi
```

---

Finally the PC-NFS daemons (`pcnfsd`) and the lock manager daemon (`rpc.lockd`) status monitor daemon (`rpc.statd`) are started.

---

```
PCNFS_SERVER=1
if [ $LFS -eq 0 -a $PCNFS_SERVER -ne 0 -a -f /etc/pcnfsd ] ; then
    /etc/pcnfsd && echo "starting up the PC-NFS daemon" && echo
"\t/etc/pcnfsd"
    set_return
fi

if [ $NFS_CLIENT -ne 0 -o $NFS_SERVER -ne 0 ] ; then
    if [ -f /usr/etc/rpc.statd ] ; then
        /usr/etc/rpc.statd && echo "starting up the Status Monitor daemon" && echo
"\t/usr/etc/rpc.statd"
        set_return
    fi
    if [ -f /usr/etc/rpc.lockd ] ; then
        /usr/etc/rpc.lockd && echo "starting up the Lock Manager daemon" && echo
"\t/usr/etc/rpc.lockd"
        set_return
    fi
fi
exit $returnstatus
```

---

## **E.2 rc file**

The `rc` file is executed when the UNIX node starts. It contains a number of functions (such as `localrc()`, `hfsmount()`, and so on) which are called from a main section. The example script given next contains some of the functions defined in Table E.1.

**Table E.1** Sample rc functions.

<i>Function</i>	<i>Description</i>
<code>localrc()</code>	Add local configuration to the node. In the example script the Bones-Licensing 2.4 is started locally on the node. This part of the script will probably be the only function which is different on different nodes.
<code>hfsmount()</code>	Mounts local disk drives
<code>map_keyboard()</code>	Loads appropriate keymap
<code>syncer_start()</code>	The syncer helps to minimize file damage when this is a power failure or a system crash
<code>lp_start()</code>	Starts the lp (line printer) scheduler
<code>net_start()</code>	Starts networking through netlinkrc
<code>swap_start()</code>	Starts swapping on alternate swap devices

---

```

initialize()
{
    if [ "$SYSTEM_NAME" = "" ]
    then
        SYSTEM_NAME=pollux
        export SYSTEM_NAME
    fi
}

localrc()
{

#%&CSIBeginFeature: Bones-Licensing 2.4
DESIGNERHOME=/win/designer-2.0
export DESIGNERHOME

echo -n "Starting Bones-Licensing 2.4 ..."
if [ -f ${DESIGNERHOME}/bin/start-lmgrd ]; then
    ${DESIGNERHOME}/bin/start-lmgrd
    echo " lmgrd."
else
    echo " failed."
fi
}

set_date()
{
    if [ $SET_PARMS_RUN -eq 0 ] ; then
        if [ $TIMEOUT -ne 0 ] ; then
            echo "\007Is the date `date` correct? (y or n, default: y) \c"
            reply=`line -t $TIMEOUT`
            echo ""
        fi
        if [ "$reply" = y -o "$reply" = "" -o "$reply" = Y ]
        then
            return
        else
            if [ -x /etc/set_parms ]; then
                /etc/set_parms time_only
            fi
        fi
    fi
    fi # if SET_PARMS_RUN
}

hfsmount()
{
    # create /etc/mnttab with valid root entry
    /etc/mount -u >/dev/null

    # enable quotas on the root file system
    # (others are enabled by mount)
    [ -f /quotas -a -x /etc/quotaon ] && /etc/quotaon -v /

    # Mount the HFS volumes listed in /etc/checklist:
    /etc/mount -a -t hfs -v
    # (NFS volumes are mounted via net_start() function)

    # Uncomment the following mount command to mount CDFS's
    /etc/mount -a -t cdfs -v

    # Preen quota statistics
    [ -x /etc/quotacheck ] && echo checking quotas && /etc/quotacheck -ap
}

```

---

---

```

map_keyboard()
{
#
itemap_option=""
if [ -f /etc/kbdlang ]
then
    read MAP_NAME filler < /etc/kbdlang
    if [ $MAP_NAME ]
    then
        itemap_option="-l $MAP_NAME"
    fi
fi

if [ -x /etc/itemap ]
then
    itemap -i -L $itemap_option -w /etc/kbdlang
fi
}

syncer_start()
{
    if /usr/bin/rtprio 127 /etc/syncer
    then
        echo syncer started
    fi
}

lp_start()
{
    if [ -s /usr/spool/lp/pstatus ]
    then
        lpshut > /dev/null 2>&1
        rm -f /usr/spool/lp/SCHEDLOCK
        lpsched
        echo line printer scheduler started
    fi
}

clean_ex()
{
    if [ -x /usr/bin/ex ]
    then
        echo "preserving editor files (if any)"
        ( cd /tmp; expreserve -a )
    fi
}

clean_uucp()
{
    if [ -x /usr/lib/uucp/uuclean ]
    then
        echo "cleaning up uucp"
        /usr/lib/uucp/uuclean -pSTST -pLCK -n0
    fi
}

net_start()
{
    if [ -x /etc/netlinkrc ] && /etc/netlinkrc
    then
        echo NETWORKING started.
    fi
}

swap_start()

```

---

```

{
    if /etc/swapon -a
    then
        echo 'swap device(s) active'
    fi
}

cron_start()
{
    if [ -x /etc/cron ]
    then
        if [ -f /usr/lib/cron/log ]
        then
            mv /usr/lib/cron/log /usr/lib/cron/OLDlog
        fi
        /etc/cron && echo cron started
    fi
}

audio_start ()
{
    # Start up the audio server
    if [ -x /etc/audiorc ] && /etc/audiorc
    then
        echo "Audio server started"
    fi
}

#
# The main section of the rc script
#
# Where to find commands:
PATH=/bin:/usr/bin:/usr/lib:/etc

# Set termio configuration for output device.
stty clocal icanon echo opost onlcr ixon icrnl ignpar

if [ ! -f /etc/rcflag ]    # Boot time invocation only
then
    # /etc/rcflag is removed by /etc/brc at boot and by shutdown
    touch /etc/rcflag

    hfsmount
    map_keyboard
    setparms
    initialize
    switch_over
    uname -s $SYSTEM_NAME
    hostname $SYSTEM_NAME

    swap_start
    syncer_start
    lp_start
    clean_ex
    clean_uucp
    net_start
    audio_start
    localrc
fi

```



---

## Ethernet Monitoring System

---

### F.1 Ethernet receiver

---

The next page gives a schematic for an Ethernet monitoring system. Its component values are:

$$R1 = 1 \text{ k}\Omega$$

$$R2 = 500 \Omega$$

$$R3 = 10 \text{ M}\Omega$$

$$R4 = 39 \Omega$$

$$R5 = 1.5 \text{ k}\Omega$$

$$R6 = 10 \Omega$$

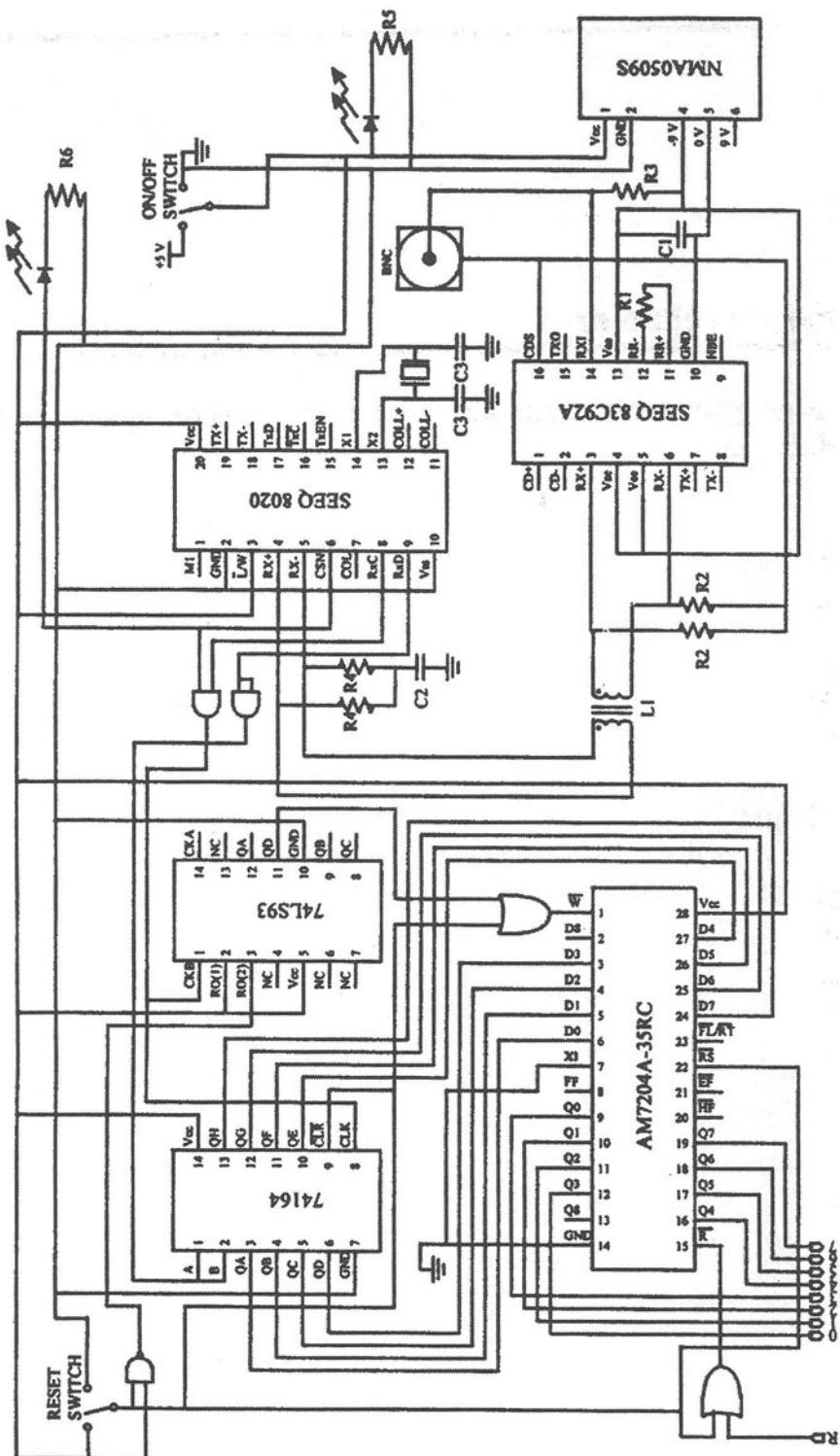
$$L1 = 1:1, 200 \mu\text{H}$$

$$C1 = 100 \text{ mF}$$

$$C2 = 0.1 \mu\text{F}$$

$$C3 = 1.5 \text{ pF}$$

$$\text{XTAL} = 20 \text{ MHz}$$





# RS-232 and Parallel Port

## G.1 RS-232

### G.1.1 The UART

The main device used to construct an RS-232 interface is the UART (universal asynchronous receiver and transmitter) which is a fully programmable device. It can be set-up to have a defined Baud rate, number of data bits, number of start bits, number of stop bits, type of parity, manipulation of the output and input handshaking lines. Figure G.1 shows a block diagram. The UART is fully compatible with TTL devices and has a +5 V and 0 V power supply. Typical UARTs are Intel's 8250, Motorola's MC8650, Motorola's MC68681 and National's INS 8250. The 8250/1 device is compatible with Intel's 80×86 microprocessors and is thus used in many PC applications. A 16-bit UART device is available and is named the 16450.

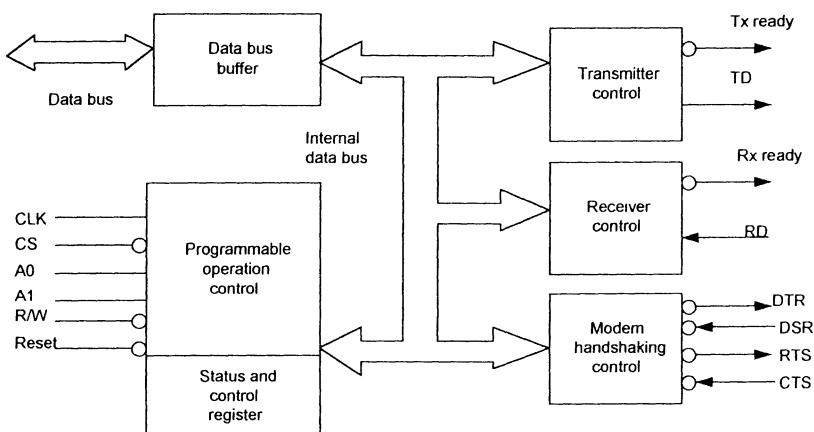
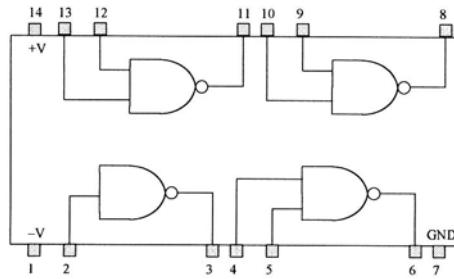


Figure G.1 UART.

### G.1.2 Drivers and receivers

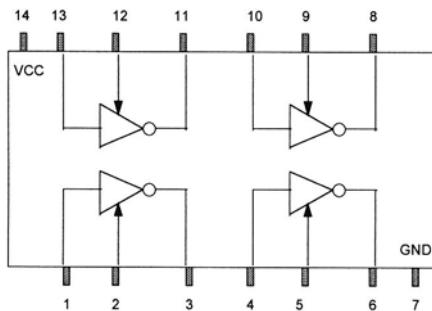
Driver ICs convert the TTL output/input of the UART to RS-232 voltage levels. The MC1488 quad line driver contains four NAND logic buffer gates and has supply lines of +/− 12V and GND, as illustrated in Figure G.2. If the two inputs to the drivers are tied together to create an inverter function, a table of the conversion is shown next:

- 0 V input gives +12 V output.
- 5 V input gives -12 V output.

**Figure G.2** MC1488 pin out.

The second input of the NAND gate can also be used to generate a Break signal on the transmit line, where a low input will cause a high output (+12 V).

The MC1489 is a line receiver that converts RS-232 line voltage back to TTL levels. This device is specially designed to receive pulses over a long and highly capacitive transmission line. Figure G.3 shows its pin connections.

**Figure G.3** MC1489 pin out.

### **G.1.3 Transmitter/receiver interface**

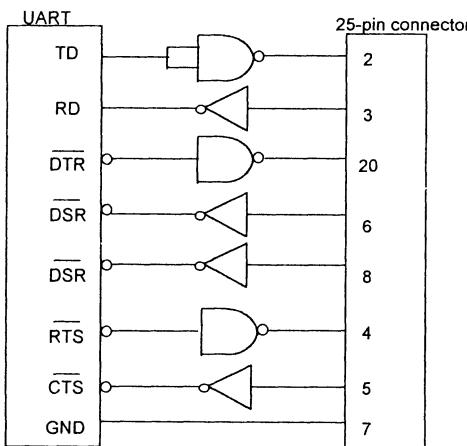
The output lines from the UART are TD, DTR, and RTS and the input lines are RD, DSR, DCD and CTS. Figure G.4 shows how these lines are buffered via the driver and receiver gates.

---

## **G.2 Parallel port**

### **G.2.1 IEEE 1284 cables assemblies**

The IEEE 1284 standard defined standards for compatibility between different platforms and peripherals. The most typical connector is a DB25 male on one end and a 36-pin Champ plug connector on the other. The cable has from 18 to 25 conductors and from one to eight ground wires. This cable is acceptable for low speed (10kB/sec) at a maximum distance of 2 m but will not operate properly for high bit rates (such as 2MB/sec) over relatively long runs (such as 10 m).



**Figure G.4** Transmitter/receiver interface.

A 1284 connection complies with the following:

1. Signals connect to a twisted pair with a signal line and ground return.
2. Each signal has a characteristic unbalanced impedance of approx.  $62\ \Omega$ .
3. Crosstalk is less than 10%.
4. 85% minimum coverage of braid over foil.
5. Cable shield connects to the connector backshell using a  $360^\circ$  concentric method (and no pigtail connections).
6. Cables are marked with the label: IEEE Std 1284-1994.

#### G.2.2 IEEE 1284 connectors

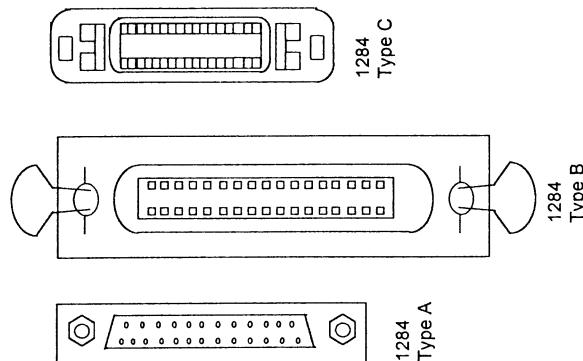
The 1284 standard defines the standard connectors to be used with the parallel port, for its both its mechanical and electrical specification. This ensures compatibility with existing and future applications.

The three connectors defined are:

- 1284 Type A. 25-pin DB25 connector.
- 1284 Type B. 36-pin, 0.085 centerline Champ connector with bale locks
- 1284 Type C. 36-pin, 0.050 centerline mini-connector with clip latches.

Figure G.5 illustrates these connectors. The best specification is the type C connector as:

- It has a smaller footprint than the others.
- It has a simple-to-use clip latch for cable retention.
- It has the easiest cable assembly with the optimal electrical properties.
- Its cable assembly has two extra signals, which are Peripheral Logic High and Host Logic High. These can be used to determine if the device at the other end of the cable is powered on. This allows for some degree of intelligent power management for 1284 interfaces.



**Figure G.5** 1284 interface I/O connectors.

Typical assembly types are:

AMAM	Type A Male to Type A Male	AMAF	Type A Male to Type A Female
AB	Type A Male to Type B Plug	AC	Type A Male to Type C Plug
BC	Type B Plug to Type C Plug	CC	Type C Plug to Type C Plug

### G.2.3 IEEE 1284 Electrical Interface

The 1284 standard defines two levels of interface compatibility:

- Level I. Designed for low-speed applications, but which need the reverse channel capabilities.
- Level II. Design for advanced modes, high bit rates and long cables.

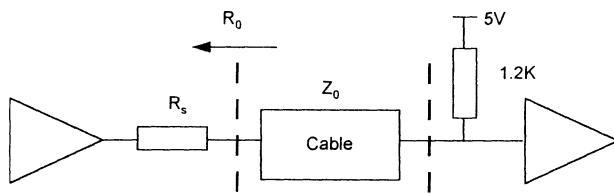
The electrical specifications for Level II drivers and receivers are defined at the connector interface. For drivers these are:

1. Open-circuit high-level output voltage: less than +5.5 V.
2. Open-circuit low-level output voltage: not less than -0.5 V.
3. DC steady-state, high-level output voltage: greater than +2.4 V with a source current of 14 mA.
4. DC steady-state, low-level output voltage: less than +0.4 V with a sink current of 14 mA.
5. Output impedance:  $50 \pm 5 \Omega$ .
6. Output slew rate: between 0.05 and  $0.40 \text{ V.ns}^{-1}$ .

For drivers these are:

1. Peak input voltage transients without damage: between -2.0 V and +7.0 V.
2. High-level input threshold: less than 2.0 V
3. Low-level input threshold: greater than 0.8 V.
4. Input hysteresis: at least 0.2 V, but not more than 1.2 V.
5. High-level sink current: less than  $20 \mu\text{A}$  at +2.0 V.
6. Low-level input source current: less than  $20 \mu\text{A}$  at +0.8 V.
7. Circuit and stray capacitance: less than 50 pF.

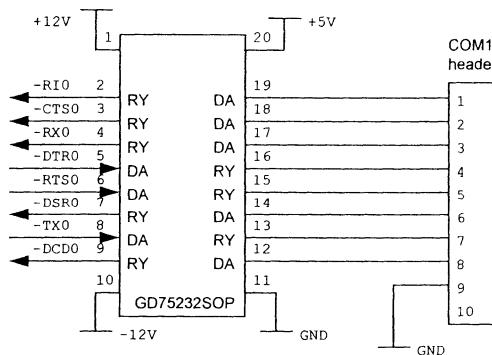
Figure A.6 shows the recommended termination for a driver/receiver pair, where  $R_0$  is the output impedance at the connector. This should be matched to the impedance of the cable ( $Z_0$ ) so that noise and reflections can be minimized. A series resistance ( $R_s$ ) can thus be added to obtain a match.



**Figure G.6** Level II driver/receiver pair termination example.

### G.3 PC connections

The 430HX motherboard uses the 82091AA (API) device. Thus incorporates two 8250 UARTs, a parallel port connection and a floppy disk interface. Refer to Figure 11.3 to pin out. It shows the connection between the AIP and the serial port interface. It shows the signal lines for COM1; these are fed into the GD75232SOP device which converts between 0/+5 V and +/-12 V. It then connects to a 10-pin header which is wired to the serial port.



**Figure G.7** COM1 connection.

### G.4 Parallel port connection

Figure G.9 shows the connections from the AIP to the 26-pin parallel port header. The  $33\Omega$  resistors on the output lines are the to protect against a short circuit on the output pins. This limits the short circuit current to less than 150 mA ( $5\text{V}/33\Omega$ ). The  $1\text{K}\Omega$  pull-up resistor

causes the input line to be a high input, if there is nothing connected to the port. Thus, when nothing is connected, the BUSY, SLCT and PE lines will be active, while ERR and ACK will be inactive. Note that the physical layout of the 26-pin header is:

1 (-STB)	2 (-AFD)
3 (PDO)	4 (-ERR)
5 (PD1)	6 (-INIT)
7 (PD2)	8 (-SLIN)
9 (PD3)	10 (GND)
11 (PD4)	12 (GND)
13 (PD5)	14 (GND)
15 (PD6)	16 (GND)
17 (PD7)	18 (GND)
19 (-ACK)	20 (GND)
21 (BUSY)	22 (GND)
23 (PE)	24 (GND)
25 (SLCT)	26 (N/C)

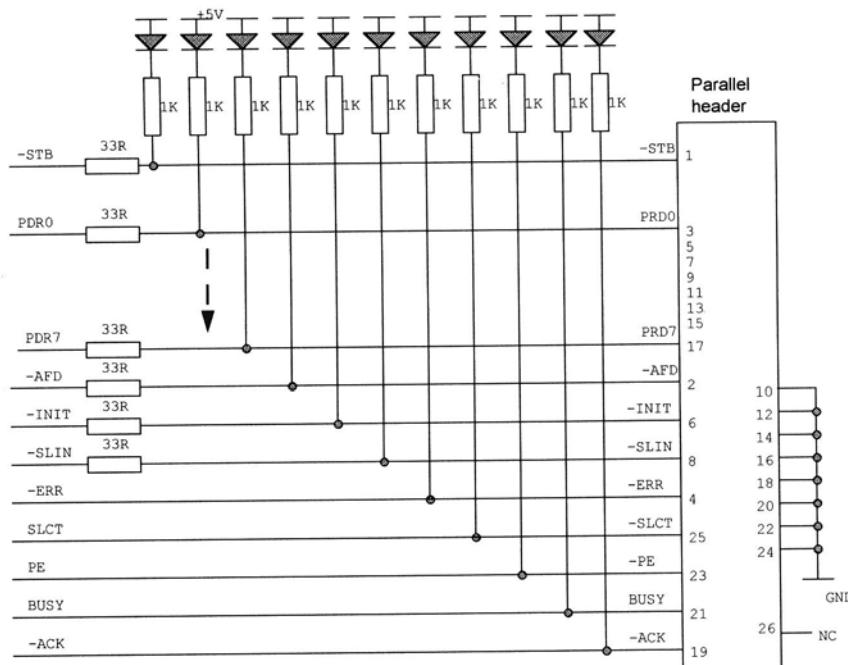


Figure G.8 Parallel port connection.

## G.5 RS-232C interface

---

**Table G.1** RS-232C connections.

<i>9-pin D-type</i>	<i>25-pin D-type</i>	<i>Name</i>	<i>RS-232 name</i>	<i>Description</i>	<i>Signal direction on DCE</i>
	1		AA	Protective GND	
3	2	TXD	BA	Transmit Data	IN
2	3	RXD	BB	Receive Data	OUT
7	4	RTS	CA	Request to Send	IN
8	5	CTS	CB	Clear to Send	OUT
6	6	DSR	CC	Data Set Ready	OUT
5	7	GND	AB	Signal GND	
1	8	DCD	CF	Received Line Signal detect	OUT
	9		-	RESERVED	-
	10		-	RESERVED	-
	11			UNASSIGNED	-
	12		SCF	Secondary Received Line Signal Detector	OUT
	13		SCB	Secondary Clear to Send	OUT
	14		SBA	Secondary Transmitted Data	IN
	15		DB	Transmission Signal Element Detector	OUT
	16		SBB	Secondary Received Data	OUT
	17		DD	Receiver Signal Element Time	OUT
	18			UNASSIGNED	-
	19		SCA	Secondary Request to Send	IN
4	20	DTR	CD	Data Terminal Ready	IN
	21		CG	Signal Quality Detector	OUT
9	22	RI	CE	Ring Indicator	OUT
	23		CH/CI	Data Signal Rate Selector	IN/OUT
	24		DA	Transmit Signal Element Timing	IN
	25			UNASSIGNED	-

## G.6 RS-449 interface

---

RS-449 defines a standard for the functional/mechanical interface of DTEs/DCEs for serial communications and is usually used with synchronous transmissions. Table G.2 lists the main connections.

**Table G.2** RS-449 connections.

<i>Pin number</i>	<i>Mnemonic</i>	<i>Description</i>
1		Shield
2	SI	Signaling Rate Indicator
3, 21		Spare
4, 22	SD	Sending Time
5, 23	ST	Receive Data
6, 24	RD	Receive Data
7, 25	RS	Request to Send
8, 26	RT	Receive Timing
9, 27	CS	Clear to Send
10	LL	Local Loopback
11, 29	DM	Data Mode
12, 30	TR	Terminal Ready
13, 31	RR	Receiver Ready
14	RL	Remote Loopback
15	IC	Incoming Call
16	SF/SR	Select Frequency/ Signaling Rate Select
17, 37	TT	Terminal Timing
18	TM	Test Mode
19	SG	Signal Ground
20	RC	Receive Common
28	IS	Terminal in Service
32	SS	Select Standby
33	SQ	Signal Quality
34	NS	New Signal
36	SB	Standby Indicator
37	SC	Send Common



---

# Modem Codes

---

## H.1 AT commands

---

The AT commands are preceded by the attention code AT. They are:

- A      Go on-line in answer mode**  
Instructs the modem to go off-hook immediately and then make a connection with a remote modem
- Bn     Select protocol to 300 bps to 1200 bps**  
B0        Selects CCITT operation at 300 bps or 1200 bps  
B1        Selects BELL operation at 300 bps or 1200 bps
- D      Go on-line in originate mode**  
Instructs the modem to go off-hook and automatically dials the number contained in the dial string which follows the D command
- En     Command echo**  
E0        Disable command echo  
E1        Enables command echo (default)
- Fn     Select line modulation**  
F0        Select auto-detect mode  
F1        Select V.21 or Bell 103  
F4        Select V.22 or Bell 212A 1200 bps  
F5        Select V.22bis line modulation.  
F6        Select V.32bis or V.32 4800 bps line modulation  
F7        Select V.32bis or V.32 7200 bps line modulation  
F8        Select V.32bis or V.32 9600 bps line modulation  
F9        Select V.32bis 12000 line modulation  
F10      Select V.32bis 14400 line modulation
- Hn     Hang-up**  
H0        Go on-hook (hang-up connection)  
H1        Goes off-hook
- In      Request product code or ROM checksum**  
I0        Reports the product code  
I1/I2     Reports the hardware ROM checksum  
I3        Reports the product revision code  
I4        Reports response programmed by an OEM  
I5        Reports the country code number
- Ln      Control speaker volume**  
L0        Low volume  
L1        Low volume  
L2        Medium volume (default)  
L3        High volume
- Mn      Monitor speaker on/off**  
M0/M     Speaker is always off  
M1        Speaker is off while receiving carrier (default)  
M2        Speaker is always on  
M3        Speaker is on when dialing but is off at any other time
- Nn      Automode enable**  
N0        Automode detection is disabled

	N1	Automode detection is enabled
<b>O</b>	<b>n</b>	<b>Return to the on-line state</b>
	O0	Enters on-line data mode with a retrain
	O1	Enters on-line data mode without a retrain
<b>P</b>		<b>Set pulse dial as default</b>
<b>Q</b>		<b>Result code display</b>
	Q0	Send result codes to the computer
	Q1	No return codes
<b>S</b>	<b>n</b>	<b>Reading and writing to S registers</b>
	Sn?	Reads the Sn register
	Sn=val	Writes the value of val to the Sn register
<b>T</b>		<b>Set tone dial as default</b>
<b>V</b>	<b>n</b>	<b>Select word or digit result code</b>
	V0	Display result codes in a numeric form
	V1	Display result code in a long form (default)
<b>W</b>	<b>n</b>	<b>Error correction message control</b>
	W0	When connected report computer connection speed
	W1	When connected report computer connection speed, error correcting protocol and line speed
	W2	When connected report modem connection speed
<b>X</b>	<b>n</b>	<b>Select result code</b>
	X0	Partial connect message, dial-tone monitor off, busy tone monitor off
	X1	Full connect message, dial-tone monitor off, busy tone monitor off
	X2	Full connect message, dial-tone monitor on, busy tone monitor off
	X3	Full connect message, dial-tone monitor off, busy tone monitor on
	X4	Full connect message, dial-tone monitor on, busy tone monitor on
<b>Y</b>	<b>n</b>	<b>Enables or disables long space disconnection</b>
	Y0	Disables long space disconnect (default)
	Y1	Enables long space disconnect
<b>Z</b>	<b>n</b>	<b>Reset</b>
	Z0	Resets modem and load stored profile 0
	Z1	Resets modem and load stored profile 1
<b>&amp;C</b>	<b>n</b>	<b>Select DCD options</b>
	&C0	Sets DCD permanently on
	&C1	Use state of carrier to set DCD (default)
<b>&amp;D</b>	<b>n</b>	<b>DTR option</b>
		This is used with the &Qn setting to determine the operation of the DTR signal
		&D0      &D1      &D2      &D3
	&Q0	a            c            d            e
	&Q1	b            c            d            e
	&Q2	d            d            d            d
	&Q3	d            d            d            d
	&Q4	b            c            d            e
	&Q5	a            c            d            e
	&Q6	a            c            d            e
		where
	a	- modem ignore DTR signal
	b	- modem disconnects and sends OK result code
	c	- modem goes into command mode and sends OK result code
	d	- modem disconnects and sends OK result code.
<b>&amp;F</b>		<b>Restore factory configuration</b>
<b>&amp;G</b>	<b>n</b>	<b>Set guard tone</b>
	&G0	Disables guard tone (default)
	&G1	Disables guard tone
	&G2	Selects 1800 Hz guard tone
<b>&amp;K</b>	<b>n</b>	<b>DTE/modem flow control</b>
	&K0	Disables DTE/DCE flow control
	&K3	Enables RTS/CTS handshaking flow control (Default)

	&K4	Enables XON/XOFF flow control		
	&K5	Enables transparent XON/XOFF flow control		
	&K6	Enables RTS/CTS and XON/XOFF flow control		
<b>&amp;L Line selection</b>				
	&L0	Selects dial-up line operation (Default)		
	&L1	Selects leased line operation		
<b>&amp;Mn Communications mode</b>				
<b>&amp;Pn Select pulse dialing make/break ratio</b>				
	&P0	Sets a 39/61 make-break ratio at 10 pps (Default)		
	&P1	Sets a 33/67 make-break ratio at 10 pps (Default)		
	&P2	Sets a 39/61 make-break ratio at 20 pps (Default)		
	&P3	Sets a 33/67 make-break ratio at 20 pps (Default)		
<b>&amp;Qn Asynchronous/synchronous mode selection</b>				
	&Q0	Set direct asynchronous operation		
	&Q1	Set synchronous operation with asynchronous off-line		
	&Q2	Set synchronous connect mode with asynchronous off-line		
	&Q3	Set synchronous connect mode		
	&Q5	Modem negotiation for error-corrected link		
	&Q6	Set asynchronous operation in normal mode		
<b>&amp;Rn RTS/CTS option</b>				
	&R0	In synchronous mode, CTS changes with RTS (the delay is defined by the S26 register)		
	&R1	In synchronous mode, CTS is always ON		
<b>&amp;Sn DSR option</b>				
	&S0	DSR is always ON (Default)		
	&S1	DSR is active after the answer tone has been detected		
<b>&amp;Tn Testing and diagnostics</b>				
	&T0	Terminates any current test	&T1	Local analogue loopback test
	&T2	Local digital loopback test		
<b>&amp;V View configuration profiles</b>				
<b>&amp;Wn Store the current configuration in non-volatile RAM</b>				
	&W0	Writes current settings to profile 0 in nonvolatile RAM		
	&W1	Writes current settings to profile 1 in nonvolatile RAM		
<b>&amp;Xn Clock source selection</b>				
	&X0	Selects internal timing, where the modem uses its own clock for transmitted data		
	&X1	Selects external timing, where the modem gets its timing from the DTE (computer)		
	&X2	Selects slave receive timing, where the modem gets its timing from the received signal		
<b>&amp;Yn Select default profile</b>				
	&Y0	Use profile 0 on power-up (Default)		
	&Y1	Use profile 1 on power-up		
<b>&amp;Zn Store telephone numbers</b>				
	&Z0	Store telephone number 1	&Z1	Store telephone number 2
	&Z2	Store telephone number 3	&Z3	Store telephone number 4
<b>\An Maximum MNP block size</b>				
	\A0	64 characters		
	\A1	128 characters		
	\A2	192 characters		
	\A3	256 characters		
<b>\Bn Transmit break</b>				
	\B1	Break length 100 ms	\B2	Break length 200 ms
	\B3	Break length 300 ms (Default)	<i>and so on.</i>	
<b">\Gn Modem/modem flow control</b">				
	\G0	Disable (Default)		
	\G1	Enable		
<b>\Jn Enable/disable DTE auto rate adjustment</b>				
	\J0	Disable	\J1	Enable
<b>\Kn Break control</b>				
	\K0	Enter on-line command mode with no break signal		

	\K1	Clear data buffers and send a break to the remote modem
	\K3	Send a break to the remote modem immediately
	\K5	Send a break to the remote modem with transmitted data
\Ln		<b>MNP block transfer control</b>
	\L0	Use stream mode for MNP connection (Default)
	\L1	Use interactive MNP block mode.

## H.2 Result codes

---

After the modem has received an AT command it responds with a return code. A complete set of return codes are given in Table H.1.

**Table H.1 Modem return codes.**

Message	Digit	Description
OK	0	Command executed without errors
CONNECT	1	A connection has been made
RING	2	An incoming call has been detected
NO CARRIER	3	No carrier detected
ERROR	4	Invalid command
CONNECT 1200	5	Connected to a 1200 bps modem
NO DIAL-TONE	6	Dial-tone not detected
BUSY	7	Remote line is busy
NO ANSWER	8	No answer from remote line
CONNECT 600	9	Connected to a 600 bps modem
CONNECT 2400	10	Connected to a 2400 bps modem
CONNECT 4800	11	Connected to a 4800 bps modem
CONNECT 9600	13	Connected to a 9600 bps modem
CONNECT 14400	15	Connected to a 14 400 bps modem
CONNECT 19200	16	Connected to a 19200 bps modem
CONNECT 28400	17	Connected to a 28400 bps modem
CONNECT 38400	18	Connected to a 38400 bps modem
CONNECT 115200	19	Connected to a 115200 bps modem
FAX	33	Connected to a FAX modem in FAX mode
DATA	35	Connected to a data modem in FAX mode
CARRIER 300	40	Connected to V.21 or Bell 103 modem
CARRIER 1200/75	44	Connected to V.23 backward channel carrier modem
CARRIER 75/1200	45	Connected to V.23 forwards channel carrier modem
CARRIER 1200	46	Connected to V.22 or Bell 212 modem
CARRIER 2400	47	Connected to V.22 modem
CARRIER 4800	48	Connected to V.32bis 4800 bps modem
CONNECT 7200	49	Connected to V.32bis 7200 bps modem
CONNECT 9600	50	Connected to V.32bis 9600 bps modem
CONNECT 12000	51	Connected to V.32bis 12000 bps modem
CONNECT 14400	52	Connected to V.32bis 14400 bps modem
CONNECT 19200	61	Connected to a 19 200 bps modem
CONNECT 28800	65	Connected to a 28 800 bps modem
COMPRESSION: CLASS 5	66	Connected to modem with MNP Class 5 compression
COMPRESSION: V.42bis	67	Connected to a V.42bis modem with compression
COMPRESSION: NONE	69	Connection to a modem with no data compression
PROTOCOL: NONE	70	
PROTOCOL: LAPM	77	
PROTOCOL: ALT	80	

### H.3 S-registers

---

The modem contains various status registers called the S-registers which store modem settings. Table H.2 lists these registers.

**Table H.2 Modem registers.**

Register	Function	Range [typical default]
S0	Rings to Auto-answer	0–255 rings [0 rings]
S1	Ring counter	0–255 rings [0 rings]
S2	Escape character	[43]
S3	Carriage return character	[13]
S6	Wait time for dial-tone	2–255 s [2 s]
S7	Wait time for carrier	1–255 s [50 s]
S8	Pause time for automatic dialing	0–255 s [2 s]
S9	Carrier detect response time	1–255 in 0.1 s units [6]
S10	Carrier loss disconnection time	1–255 in 0.1 s units [14]
S11	DTMF tone duration	50–255 in 0.001 s units [95]
S12	Escape code guard time	0–255 in 0.02 s units [50]
S13	Reserved	
S14	General bitmapped options	[8Ah (1000 1010b)]
S15	Reserved	
S16	Test mode bitmapped options (&T)	[0]
S17	Reserved	
S18	Test timer	0–255 s [0]
S19–S20	Reserved	
S21	V.24/General bitmapped options	[04h (0000 0100b)]
S22	Speak/results bitmapped options	[75h (0111 0101b)]
S23	General bitmapped options	[37h (0011 0111b)]
S24	Sleep activity timer	0–255 s [0]
S25	Delay to DSR off	0–255 s [5]
S26	RTS–CTS delay	0–255 in 0.01 s [1]
S27	General bitmapped options	[49h (0100 1001b)]
S28	General bitmapped options	[00h]
S29	Flash dial modifier time	0–255 in 10 ms [0]
S30	Disconnect inactivity timer	0–255 in 10 s [0]
S31	General bitmapped options	[02h (0000 0010b)]
S32	XON character	[Cntrl-Q, 11h (0001 0001b)]
S33	XOFF character	[Cntrl-S, 13h (0001 0011b)]
S34–S35	Reserved	
S36	LAMP failure control	[7]
S37	Line connection speed	[0]
S38	Delay before forced hang-up	0–255 s [20]
S39	Flow control	[3]
S40	General bitmapped options	[69h (0110 1001b)]
S41	General bitmapped options	[3]
S42–S45	Reserved	
S46	Data compression control	[8Ah (1000 1010b)]
S48	V.42 negotiation control	[07h (0000 0111b)]
S80	Soft-switch functions	[0]
S82	LAPM break control	[40h (0100 0000b)]
S86	Call failure reason code	0–255
S91	PSTN transmit attenuation level	0–15 dBm [10]
S92	Fax transmit attenuation level	0–15 dBm [10]
S95	Result code message control	[0]
S99	Leased line transmit level	0–15 dBm [10]

S14	<b>Bitmapped options</b>	0	1
	Bit 0		
	Bit 1	E0	<b>E1</b>
	Bit 2	<b>Q0</b>	Q1
	Bit 3	V0	<b>V1</b>
	Bit 4	Reserved	
	Bit 5	T (tone dial)	P (pulse dial)
	Bit 6	Reserved	
	Bit 7	Answer mode	<b>Originate mode</b>
S16	<b>Modem test mode register</b>	0	1
	Bit 0	Local analogue loopback terminated	Local analogue loopback test in progress
	Bit 1	Unused	
	Bit 2	Local digital loopback terminated	Local digital loopback test in progress
	Bit 3	Remote modem analogue loopback test terminated	Remote modem analogue loopback test in progress
	Bit 4	Remote modem digital loopback test terminated	Remote modem digital loopback test in progress
	Bit 5	Remote modem digital self-test terminated	Remote modem digital self-test in progress
	Bit 6	Remote modem analogue self-test terminated	Remote modem analogue self-test in progress
	Bit 7	Unused	
S21	<b>Bitmapped options</b>	0	1
	Bit 0	<b>&amp;J0</b>	&J1
	Bit 1		
	Bit 2	<b>&amp;R0</b>	<b>&amp;R1</b>
	Bit 5	<b>&amp;C0</b>	<b>&amp;C1</b>
	Bit 6	<b>&amp;S0</b>	<b>&amp;S1</b>
	Bit 7	<b>Y0</b>	Y1
	Bit 4, 3 = 00	<b>&amp;D0</b>	
	Bit 4, 3 = 01	<b>&amp;D1</b>	
	Bit 4, 3 = 10	<b>&amp;D2</b>	
	Bit 4, 3 = 11	<b>&amp;D3</b>	
S22	<b>Speaker/results bitmapped options</b>		
	Bit 1, 0 = 00	L0	
	Bit 1, 0 = 01	<b>L1</b>	
	Bit 1, 0 = 10	L2	
	Bit 1, 0 = 11	L3	
	Bit 3, 2 = 00	M0	
	Bit 3, 2 = 01	<b>M1</b>	
	Bit 3, 2 = 10	M2	
	Bit 3, 2 = 11	M3	
	Bit 6, 5, 4 = 000	X0	
	Bit 6, 5, 4 = 001	Reserved	
	Bit 6, 5, 4 = 010	Reserved	
	Bit 6, 5, 4 = 011	Reserved	
	Bit 6, 5, 4 = 100	X1	
	Bit 6, 5, 4 = 101	X2	
	Bit 6, 5, 4 = 110	X3	
	Bit 6, 5, 4 = 111	<b>X4</b>	
	Bit 7 Reserved		
S23	<b>Bitmapped options</b>	0	1
	Bit 0	<b>&amp;T5</b>	<b>&amp;T4</b>

	Bit 3, 2, 1 = 000	300 bps communications rate
	Bit 3, 2, 1 = 001	600 bps communications rate
	Bit 3, 2, 1 = 010	1200 bps communications rate
	Bit 3, 2, 1 = 011	<b>2400 bps communications rate</b>
	Bit 3, 2, 1 = 100	4800 bps communications rate
	Bit 3, 2, 1 = 101	960 bps communications rate
	Bit 3, 2, 1 = 110	19200 bps communications rate
	Bit 3, 2, 1 = 111	Reserved
	Bit 5, 4 = 00	Even parity
	Bit 5, 4 = 01	<b>Not used</b>
	Bit 5, 4 = 10	Odd parity
	Bit 5, 4 = 11	No parity
	Bit 7, 6 = 00	<b>G0</b>
	Bit 7, 6 = 01	G1
	Bit 7, 6 = 10	G2
	Bit 7, 6 = 11	G3
S23	<b>Bitmapped options</b>	
	Bit 3, 1, 0 = 000 &M0 or &Q0	
	Bit 3, 1, 0 = 001 &M1 or &Q1	
	Bit 3, 1, 0 = 010 &M2 or &Q2	
	Bit 3, 1, 0 = 011 &M3 or &Q3	
	Bit 3, 1, 0 = 100 &Q3	
	Bit 3, 1, 0 = 101 &Q4	
	Bit 3, 1, 0 = 110 <b>&amp;Q5</b>	
	Bit 3, 1, 0 = 111 &Q6	
	0	1
	Bit 2 <b>&amp;L0</b>	<b>&amp;L1</b>
	Bit 6      B0	<b>B1</b>
	Bit 5, 4 = 00	X0
	Bit 5, 4 = 01	X1
	Bit 5, 4 = 10	X2
S28	<b>Bitmapped options</b>	
	Bits 0, 1, 2 Reserved	
	Bit 4, 3 = 00 <b>&amp;P0</b>	
	Bit 4, 3 = 01    &P1	
	Bit 4, 3 = 10    &P2	
	Bit 4, 3 = 11    &P3	
S31	<b>Bitmapped options</b>	
	0	1
	Bit 1      N0	N1
	Bit 3, 2 = 00	<b>W0</b>
	Bit 3, 2 = 01	W1
	Bit 3, 2 = 10	W2
S36	<b>LAPM failure control</b>	
	Bit 2, 1, 0 = 000 Modem disconnect	
	Bit 2, 1, 0 = 001 Modem stays on-line and a direct mode connection	
	Bit 2, 1, 0 = 010 Reserved	
	Bit 2, 1, 0 = 011 Modem stays on-line and normal mode connection is established	
	Bit 2, 1, 0 = 100 An MNP connection is made, if it fails then the modem disconnects	
	Bit 2, 1, 0 = 101 An MNP connection is made, if it fails then the modem makes a direct connection	
	Bit 2, 1, 0 = 110 Reserved	
	Bit 2, 1, 0 = 111 An MNP connection is made, if it fails then the modem makes a normal mode connection	
S37	<b>Desired line connection speed</b>	
	Bit 3, 2, 1, 0 = 0000 <b>Auto mode connection (F0)</b>	
	Bit 3, 2, 1, 0 = 0001 Modem connects at 300 bps (F1)	
	Bit 3, 2, 1, 0 = 0010 Modem connects at 300 bps (F1)	
	Bit 3, 2, 1, 0 = 0011 Modem connects at 300 bps (F1)	

	Bit 3, 2, 1, 0 = 0100 Reserved
	Bit 3, 2, 1, 0 = 0101 Modem connects at 1200 bps (F4)
	Bit 3, 2, 1, 0 = 0110 Modem connects at 2400 bps (F5)
	Bit 3, 2, 1, 0 = 0111 Modem connects at V.23 (F3)
	Bit 3, 2, 1, 0 = 1000 Modem connects at 4800 bps (F6)
	Bit 3, 2, 1, 0 = 1001 Modem connects at 9600 bps (F8)
	Bit 3, 2, 1, 0 = 1010 Modem connects at 12000 bps (F9)
	Bit 3, 2, 1, 0 = 1011 Modem connects at 144000 bps (F10)
	Bit 3, 2, 1, 0 = 1100 Modem connects at 7200 bps (F7)
S39	<b>Flow control</b>
	Bit 2, 1, 0 = 000 No flow control
	Bit 2, 1, 0 = 011 <b>RTS/CTS (&amp;K3)</b>
	Bit 2, 1, 0 = 100 XON/XOFF (&K4)
	Bit 2, 1, 0 = 101 Transparent XON (&K5)
	Bit 2, 1, 0 = 110 RTS/CTS and XON/XOFF (&K6)
S39	<b>General bitmapped options</b>
	Bit 5, 4, 3 = 000 \K0
	Bit 5, 4, 3 = 001 \K1
	Bit 5, 4, 3 = 010 \K2
	Bit 5, 4, 3 = 011 \K3
	Bit 5, 4, 3 = 100 \K4
	Bit 5, 4, 3 = 101 \K5
	Bit 7, 6 = 00 MNP 64 character block size (\A0)
	Bit 7, 6 = 01 <b>MNP 128 character block size (\A1)</b>
	Bit 7, 6 = 10 MNP 192 character block size (\A2)
	Bit 7, 6 = 11 MNP 256 character block size (\A3)



---

# PC Interfacing and Interrupts

---

## I.1 Software Interrupts

---

### I.1.1 Introduction

An interrupt allows a program or an external device to interrupt the execution of a program. The generation of an interrupt can occur by hardware (hardware interrupt) or software (software interrupt). When an interrupt occurs an interrupt service routine (ISR) is called. For a hardware interrupt the ISR then communicates with the device and processes any data. When it has finished the program execution returns to the original program. A software interrupt causes the program to interrupt its execution and goes to an interrupt service routine. Typical software interrupts include reading a key from the keyboard, outputting text to the screen and reading the current date and time.

### I.1.2 BIOS and the operating system

The Basic Input/Output System (BIOS) communicates directly with the hardware of the computer. It consists of a set of programs which interface with devices such as keyboards, displays, printers, serial ports and disk drives. These programs allow the user to write application programs that contain calls to these functions, without having to worry about controlling them or which type of equipment is being used. Without BIOS the computer system would simply consist of a bundle of wires and electronic devices.

There are two main parts to BIOS. The first is the part permanently stored in a ROM (the ROM BIOS). It is this part that starts the computer (or boots it) and contains programs which communicate with resident devices. The second stage is loaded when the operating system is started. This part is non-permanent.

An operating system allows the user to access the hardware in an easy-to-use manner. It accepts commands from the keyboard and displays them to the monitor. The Disk Operating System, or DOS, gained its name from its original purpose of providing a controller for the computer to access its disk drives. The language of DOS consists of a set of commands which are entered directly by the user and are interpreted to perform file management tasks, program execution and system configuration. It makes calls to BIOS to execute these. The main functions of DOS are to run programs, copy and remove files, create directories, move within a directory structure and to list files. Microsoft Windows 95/98NT call BIOS programs directly.

### I.1.3 Interrupt vectors

Interrupt vectors are addresses which inform the interrupt handler as to where to find the ISR. All interrupts are assigned a number from 0 to 255. The interrupt vectors associated with each interrupt number are stored in the lower 1024 bytes of PC memory. For example, interrupt 0 is stored from 0000:0000 to 0000:0003, interrupt 1 from 0000:0004 to

0000:0007, and so on. The first two bytes store the offset and the next two store the segment address. Each interrupt number is assigned a predetermined task, as outlined in Table I.1. An interrupt can be generated either by external hardware, software, or by the processor. Interrupts 0, 1, 3, 4, 6 and 7 are generated by the processor. Interrupts from 8 to 15 and interrupt 2 are generated by external hardware. These get the attention of the processor by activating a interrupt request (IRQ) line. The IRQ0 line connects to the system timer, the keyboard to IRQ1, and so on. Most other interrupts are generated by software.

**Table I.1** Interrupt handling.

<i>Interrupt</i>	<i>Name</i>	<i>Generated by</i>
00 (00h)	Divide error	processor
01 (00h)	Single step	processor
02 (02h)	Non-maskable interrupt	external equipment
03 (03h)	Breakpoint	processor
04 (04h)	Overflow	processor
05 (05h)	Print screen	Shift-Print screen key stroke
06 (06h)	Reserved	processor
07 (07h)	Reserved	processor
08 (08h)	System timer	hardware via IRQ0
09 (09h)	Keyboard	hardware via IRQ1
10 (0Ah)	Reserved	hardware via IRQ2
11 (0Bh)	Serial communications (COM2)	hardware via IRQ3
12 (0Ch)	Serial communications (COM1)	hardware via IRQ4
13 (0Dh)	Reserved	hardware via IRQ5
14 (0Eh)	Floppy disk controller	hardware via IRQ6
15 (0Fh)	Parallel printer	hardware via IRQ7
16 (10h)	BIOS – Video access	software
17 (11h)	BIOS – Equipment check	software
18 (12h)	BIOS – Memory size	software
19 (13h)	BIOS – Disk operations	software
20 (14h)	BIOS – Serial communications	software
22 (16h)	BIOS – Keyboard	software
23 (17h)	BIOS – Printer	software
25 (19h)	BIOS – Reboot	software
26 (1Ah)	BIOS – Time of day	software
28 (1Ch)	BIOS – Ticker timer	software
33 (21h)	DOS – DOS services	software
39 (27h)	DOS – Terminate and stay resident	software

#### I.1.4 Processor interrupts

The processor-generated interrupts normally occur either when a program causes a certain type of error or if it is being used in a debug mode. In the debug mode the program can be made to break from its execution when a break-point occurs. This allows the user to test the current status of the computer. It can also be forced to step through a program one operation at a time (single step mode).

## I.2 Hardware Interrupts

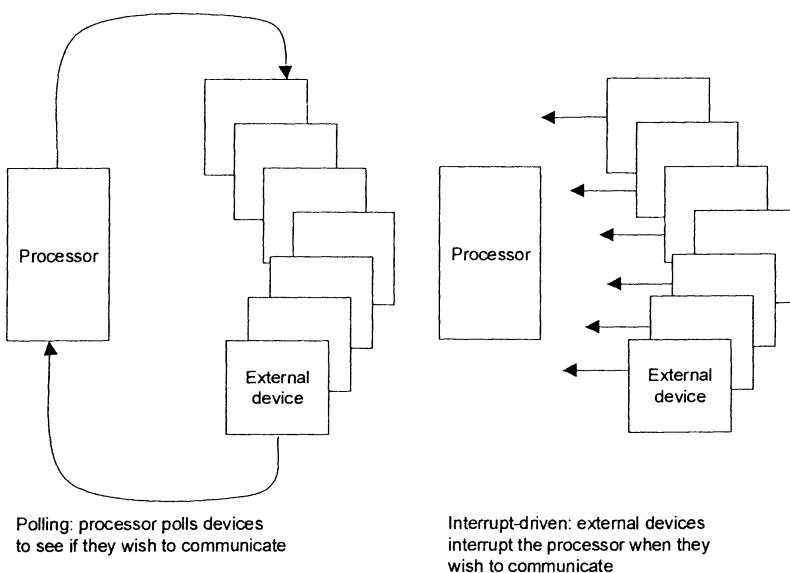
---

### I.2.1 Introduction

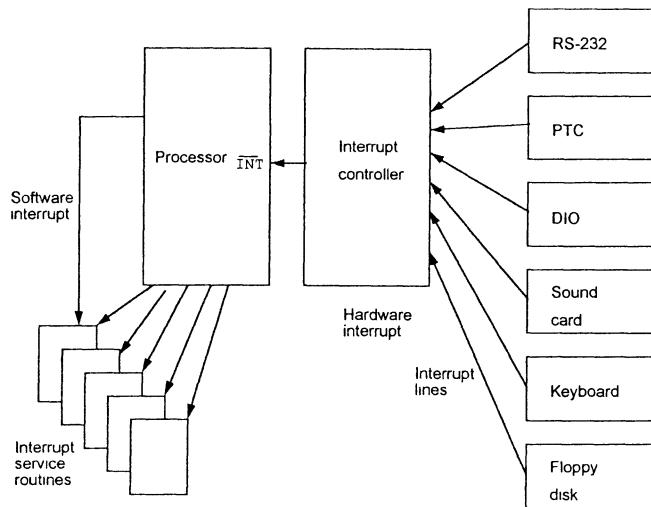
Computer systems either use polling or interrupt-driven software to service external equipment. With polling the computer continually monitors a status line and waits for it to become active, while an interrupt-driven device sends an interrupt request to the computer, which is then serviced by an interrupt service routine (ISR). Interrupt-driven devices are normally better in that the computer is thus free to do other things, while polling slows the system down as it must continually monitor the external device. Polling can also cause problems because a device may be ready to send data and the computer is not watching the status line at that point. Figure I.1 illustrates polling and interrupt-driven devices.

The generation of an interrupt can occur by hardware or software, as illustrated in Figure I.2. If a device wishes to interrupt the processor, it informs the programmable interrupt controller (PIC). The PIC then decides whether it should interrupt the processor. If there is a processor interrupt then the processor reads the PIC to determine which device caused the interrupt. Then, depending on the device that caused the interrupt, a call to an ISR is made. The ISR then communicates with the device and processes any data. When it has finished the program execution returns to the original program.

A software interrupt causes the program to interrupt its execution and go to an interrupt service routine. Typical software interrupts include reading a key from the keyboard, outputting text to the screen and reading the current date and time.



**Figure I.1** Polling and interrupt-driven communications.



**Figure I.2** Interrupt handling.

### I.2.2 Hardware interrupts

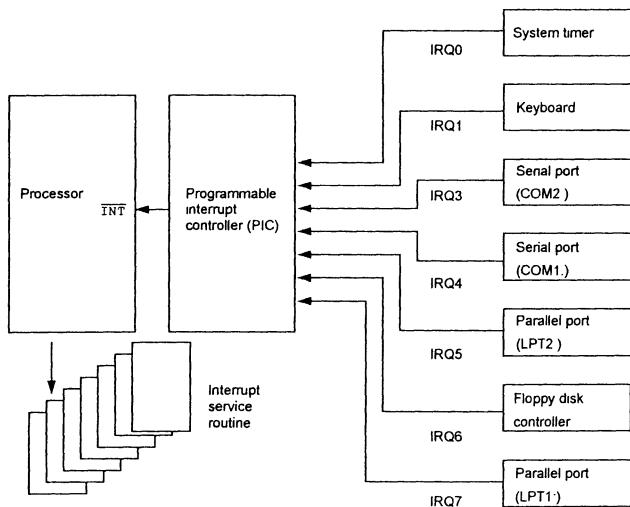
Hardware interrupts allow external devices to gain the attention of the processor. Depending on the type of interrupt the processor leaves the current program and goes to a special program called an interrupt service routine (ISR). This program communicates with the device and processes any data. After it has completed its task then program execution returns to the program that was running before the interrupt occurred. Examples of interrupts include the processing of keys from a keyboard and data from a sound card.

As previously mentioned, a device informs the processor that it wants to interrupt it by setting an interrupt line on the PC. Then, depending on the device that caused the interrupt, a call to an ISR is made. Each PIC allows access to eight interrupt request lines. Most PCs use two PICs which gives access to 16 interrupt lines.

### I.2.3 Interrupt vectors

Each device that requires to be ‘interrupt-driven’ is assigned an IRQ (interrupt request) line. Each IRQ is active high. The first eight ( $\text{IRQ}0\text{--IRQ}7$ ) map into interrupts 8 to 15 (08h–0Fh) and the next eight ( $\text{IRQ}8\text{--IRQ}15$ ) into interrupts 112 to 119 (70h–77h). Table I.1 outlines the usage of each of these interrupts. When  $\text{IRQ}0$  is made active the ISR corresponds to interrupt vector 8.  $\text{IRQ}0$  normally connects to the system timer, the keyboard to  $\text{IRQ}1$ , and so on. The standard set up of these interrupts is illustrated in Figure I.3. The system timer interrupts the processor 18.2 times per second and is used to update the system time. When the keyboard has data it interrupts the processor with the  $\text{IRQ}1$  line.

Data received from serial ports interrupts the processor with  $\text{IRQ}3$  and  $\text{IRQ}4$  and the parallel ports use  $\text{IRQ}5$  and  $\text{IRQ}7$ . If one of the parallel, or serial, ports does not exist then the IRQ line normally assigned to it can be used by another device. It is typical for interrupt-driven I/O cards, such as a sound card, to have a programmable IRQ line which is mapped to an IRQ line that is not being used.



**Figure I.3** Standard usage of IRQ lines.

Note that several devices can use the same interrupt line. A typical example is COM1: and COM3: sharing IRQ4 and COM2: and COM4: sharing IRQ3. If they do share then the ISR must be able to poll the shared devices to determine which of them caused the interrupt. If two different types of device (such as a sound card and a serial port) use the same IRQ line then there may be a contention problem as the ISR may not be able to communicate with different types of interfaces.

**Table I.2** Interrupt handling.

Interrupt	Name	Generated by
08 (08h)	System timer	IRQ0
09 (09h)	Keyboard	IRQ1
10 (0Ah)	Reserved	IRQ2
11 (0Bh)	Serial communications (COM2:)	IRQ3
12 (0Ch)	Serial communications (COM1:)	IRQ4
13 (0Dh)	Parallel port (LPT2:)	IRQ5
14 (0Eh)	Floppy disk controller	IRQ6
15 (0Fh)	Parallel printer (LPT1:)	IRQ7
112 (70h)	Real-time clock	IRQ8
113 (71h)	Redirection of IRQ2	IRQ9
114 (72h)	Reserved	IRQ10
115 (73h)	Reserved	IRQ11
116 (74h)	Reserved	IRQ12
117 (75h)	Math co-processor	IRQ13
118 (76h)	Hard disk controller	IRQ14
119 (77h)	Reserved	IRQ15

Microsoft Windows 95/98 contains a useful program which determines the usage of the system interrupts. It is selected from Control Panel by selecting System → Device Manager → Properties. Figure I.4 shows a sample window. In this case, it can be seen that the system timer uses IRQ0, the keyboard uses IRQ1, the PIC uses IRQ2, and so on. Notice that a Sound

Blaster is using IRQ5. This interrupt is normally reserved for the secondary printer port. If there is no printer connected then IRQ5 can be used by another device. Some devices can have their I/O address and interrupt line changed. An example is given in Figure I.5. In this case the IRQ line is set to IRQ7 and the base address is 378h.

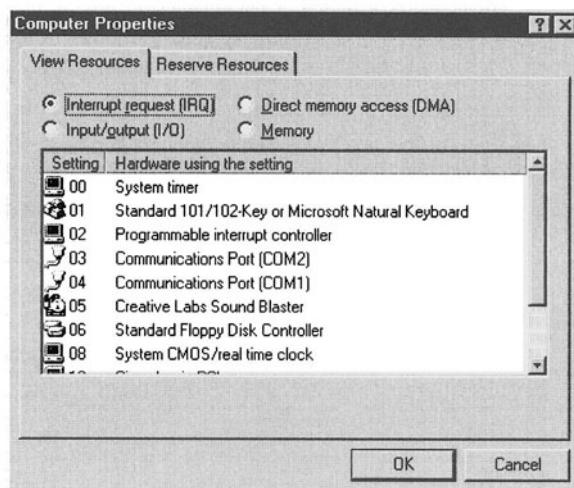


Figure I.4 Standard usage of IRQ lines.

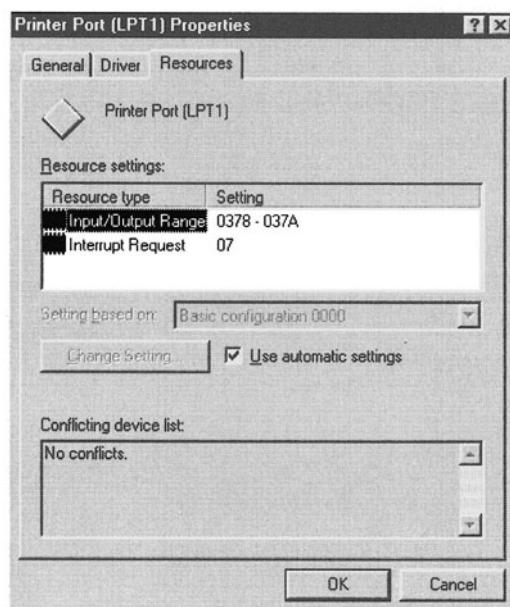


Figure I.5 Standard set up of IRQ lines.

### **IRQ0: System timer**

The system timer uses IRQ0 to interrupt the processor 18.2 times per second and is used to keep the time-of-day clock updated.

### **IRQ1: Keyboard data ready**

The keyboard uses IRQ1 to signal to the processor that data is ready to be received from the keyboard. This data is normally a scan code, but the interrupt handler performs differently for the following special keystrokes:

- *Ctrl-Break* invokes interrupt 1Bh.
- *SysRq* invokes interrupt 15h/AH=85h.
- *Ctrl-Alt-Del* performs hard or soft reboot.
- *Shift-PrtSc* invokes interrupt 05h.

### **IRQ2: Redirection of IRQ9**

The BIOS redirects the interrupt for IRQ9 back here.

### **IRQ3: Secondary serial port (COM2:)**

The secondary serial port (COM2:) uses IRQ3 to interrupt the processor. Typically, COM3: to COM8: also use it, although COM3: may use IRQ4.

### **IRQ4: Primary serial port (COM1:)**

The primary serial port (COM1:) uses IRQ4 to interrupt the processor. Typically, COM3: also uses it.

### **IRQ5: Secondary parallel port (LPT2:)**

On older PCs the IRQ5 line was used by the fixed disk. On new systems the secondary parallel port uses it. Typically, it is used by a sound card on PCs which have no secondary parallel port connected.

### **IRQ6: Floppy disk controller**

The floppy disk controller activates the IRQ6 line on completion of a disk operation.

### **IRQ7: Primary parallel port (LPT1:)**

Printers (or other parallel devices) activate the IRQ7 line when they become active. As with IRQ5 it may be used by another device, if there are no other devices connected to this line.

### **IRQ9**

Redirected to IRQ2 service routine.

#### ***I.2.4 Programmable interrupt controller (PIC)***

The PC uses the 8259 IC to control hardware-generated interrupts. It is known as a programmable interrupt controller and has eight input interrupt request lines and an output line to interrupt the processor. Originally, PCs only had one PIC and eight IRQ lines (IRQ0-IRQ7). Modern PCs can use up to 15 IRQ lines which are set up by connecting a secondary PIC interrupt request output line to the IRQ2 line of the primary PIC. The interrupt lines on the secondary PIC are then assigned IRQ lines of IRQ8 to IRQ15. This set up is shown in Figure I.6.

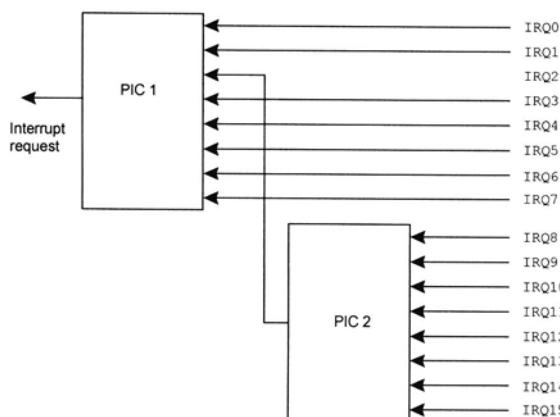
When an interrupt occurs on any of these lines it is sensed by the processor on the IRQ2 line. The processor then interrogates the primary and secondary PIC for the interrupt line which caused the interrupt.

The primary and secondary PICs are programmed via port addresses 20h and 21h, as given in Table I.3.

The operation of the PIC is programmed using registers. The IRQ input lines are either configured as level-sensitive or edge-triggered interrupt. With edge-triggered interrupts, a change from a low to a high on the IRQ line causes the interrupt. A level-sensitive interrupt occurs when the IRQ line is high. Most devices generate edge-triggered interrupts.

**Table I.3** Interrupt port addresses.

Port address	Name	Description
20h	Interrupt control port (ICR)	Controls interrupts and signifies the end of an interrupt
21h	Interrupt mask register (IMR)	Used to enable and disable interrupt lines



**Figure I.6** PC PIC connections.

In the IMR an interrupt line is enabled by setting the assigned bit to a 0 (zero). This allows the interrupt line to interrupt the processor. Figure I.7 shows the bit definitions of the IMR. For example, if bit 0 is set to a 0 then the system timer on IRQ0 is enabled.

In the example code given next the lines IRQ0, IRQ1 and IRQ6 are allowed to interrupt the processor, whereas, IRQ2, IRQ3, IRQ4 and IRQ7 are disabled.

```
outportb(0x21)=0xBC; /* 1011 1100 enable disk (bit 6), keyboard (1)
and timer (0) interrupts */
```

When an interrupt occurs all other interrupts are disabled and no other device can interrupt the processor. Interrupts are enabled again by setting the EOI bit on the interrupt control port, as shown in Figure I.8.

The following code enables interrupts:

```
outportb(0x20,0x20); /* EOI command */
```

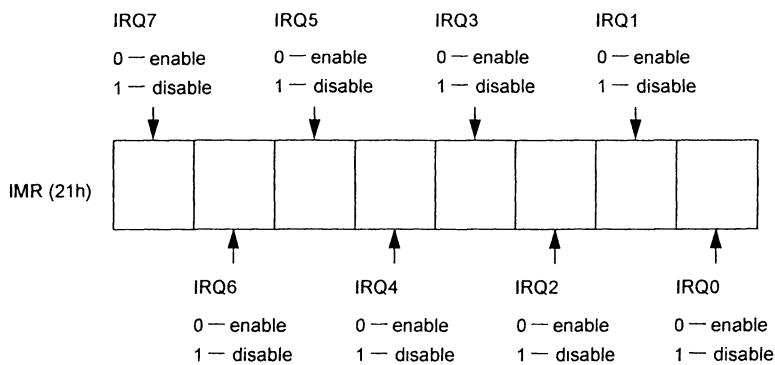


Figure I.7 Interrupt mask register bit definitions.

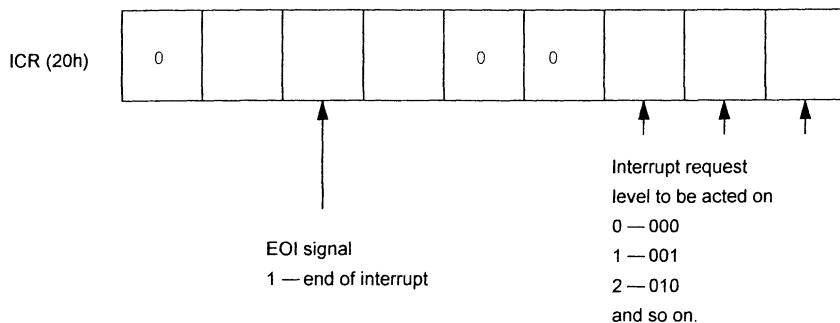


Figure I.8 Interrupt control register bit definitions.

## I.3 Interfacing

---

### I.3.1 Introduction

There are two main methods of communicating external equipment, either they are mapped into the physical memory and given a real address on the address bus (memory mapped I/O) or they are mapped into a special area of input/output memory (isolated I/O). Figure I.9 shows the two methods. Devices mapped into memory are accessed by reading or writing to the physical address. Isolated I/O provides ports which are gateways between the interface device and the processor. They are isolated from the system using a buffering system and are accessed by four machine code instructions. The `IN` instruction inputs a byte, or a word, and the `OUT` instruction outputs a byte, or a word. A high-level compiler interprets the equivalent high-level functions and produces machine code which uses these instructions.

### I.3.2 Interfacing with memory

The 80x86 processor interfaces with memory through a bus controller, as shown in Figure I.10. This device interprets the microprocessor signals and generates the required memory signals. Two main output lines differentiate between a read or a write operation ( $R/\bar{W}$ ) and

between direct and isolated memory access ( $M/\bar{IO}$ ). The  $R/W$  line is low when data is being written to memory and high when data is being read. When  $M/\bar{IO}$  is high, direct memory access is selected and when low, the isolated memory is selected.

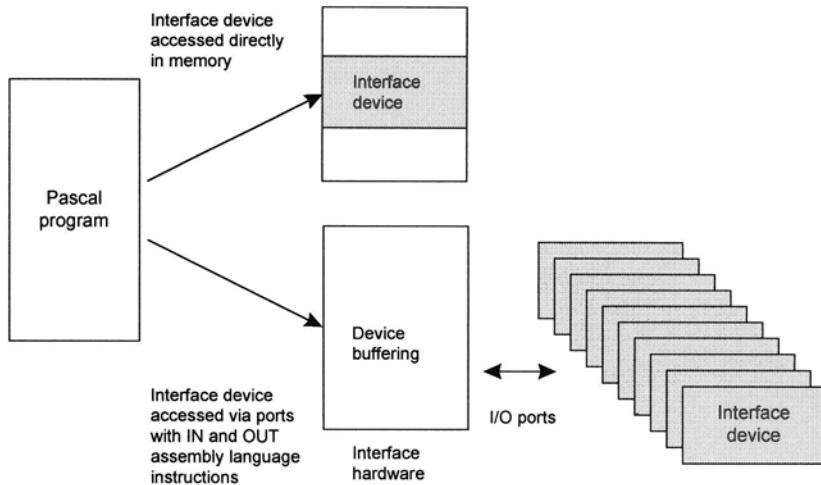


Figure I.9 Memory mapping or isolated interfacing.

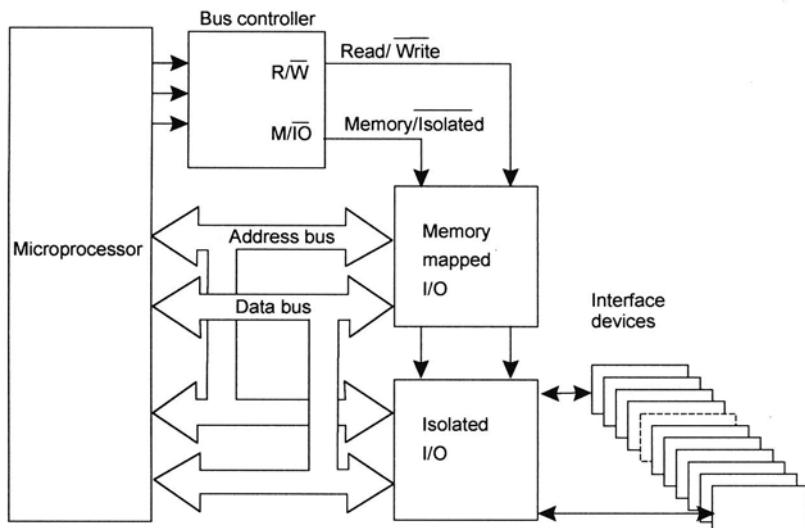


Figure I.10 Access memory mapped and isolated I/O.

### I.3.3 Memory mapped I/O

Interface devices can map directly onto the system address and data bus. In a PC-compatible system the address bus is 20 bits wide, from address 00000h to FFFFFh (1 MB). If the PC is being used in an enhanced mode (such as with Microsoft Windows) it can access the area of

memory above the 1 MB. If it uses 16-bit software (such as Microsoft Windows 3.1) then it can address up to 16 MB of physical memory, from 000000h to FFFFFFFh. If it uses 32-bit software (such as Microsoft Windows 95/98) then the software can address up to 4 GB of physical memory, from 00000000h to FFFFFFFFh. Table I.4 and Figure I.11 gives a typical memory allocation.

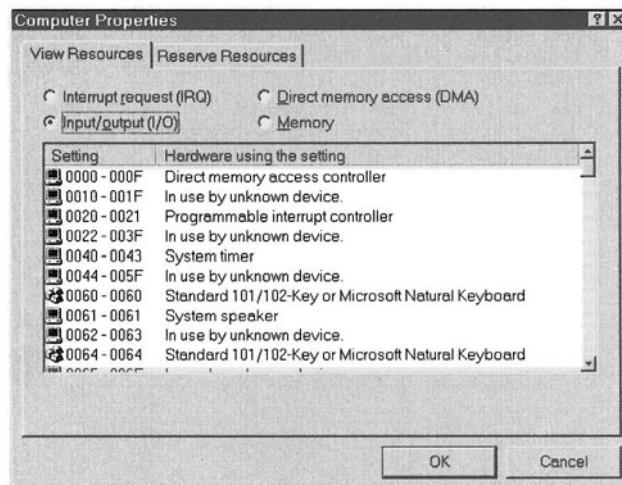
**Table I.4** Memory allocation for a PC

Address	Device
00000h-00FFFh	Interrupt vectors
00400h-0047Fh	ROM BIOS RAM
00600h-9FFFFh	Program memory
A0000h-AFFFFh	EGA/VGA graphics
B0000h-BFFFFh	EGA/VGA graphics
C0000h-C7FFFh	EGA/VGA graphics

#### I.3.4 Isolated I/O

Devices are not normally connected directly onto the address and data bus of the computer because they may use part of the memory that a program uses or they could cause a hardware fault. On modern PCs only the graphics adaptor is mapped directly into memory, the rest communicate through a specially reserved area of memory, known as isolated I/O memory.

Isolated I/O uses 16-bit addressing from 0000h to FFFFh, thus up to 64 KB of memory can be mapped. Microsoft Windows 95/98 can display the isolated I/O memory map by selecting Control Panel → System → Device Manager, then selecting Properties. From the computer properties window the Input/output (I/O) option is selected. Figure I.11 shows an example for a computer in the range from 0000h to 0064h and Figure I.13 shows from 0378h to 03FFh.



**Figure I.11** Example I/O memory map from 0000h to 0064h.

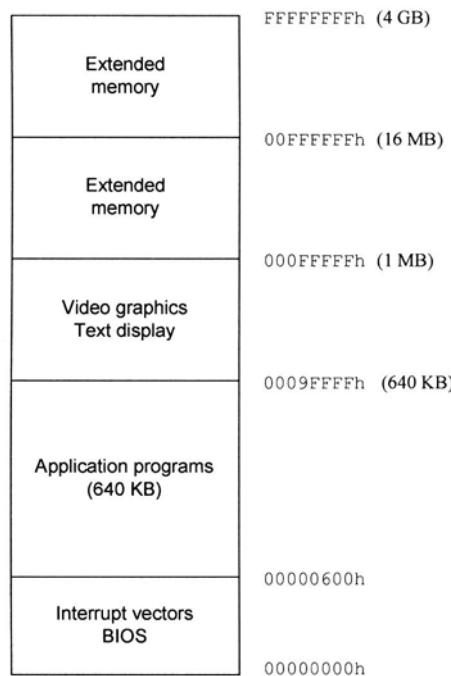


Figure I.12 Typical PC memory map.

It can be seen from Figure I.5 that the keyboard maps into address 0060h and 0064h, the speaker maps to address 0061h and the system timer between 0040h and 0043h. Table I.5 shows the typical uses of the isolated memory area.

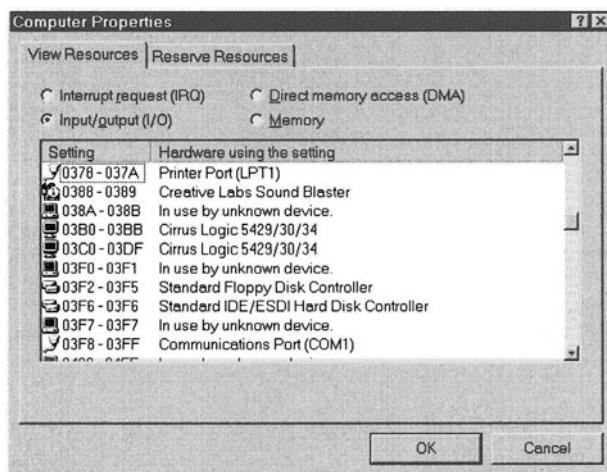


Figure I.13 Example I/O memory map from 0378h to 03FFh.

**Table I.5** Typical isolated I/O memory map.

<i>Address</i>	<i>Device</i>
000h-01Fh	DMA controller
020h-021h	Programmable interrupt controller
040h-05Fh	Counter/Timer
060h-07Fh	Digital I/O
080h-09Fh	DMA controller
0A0h-0BFh	NMI reset
0C0h-0DFh	DMA controller
0E0h-0FFh	Math coprocessor
170h-178h	Hard disk (Secondary IDE drive or CD-ROM drive)
1F0h-1F8h	Hard disk (Primary IDE drive)
200h-20Fh	Game I/O adapter
210h-217h	Expansion unit
278h-27Fh	Second parallel port (LPT2:)
2F8h-2FFh	Second serial port (COM2:)
300h-31Fh	Prototype card
378h-37Fh	Primary parallel port (LPT1:)
380h-38Ch	SDLC interface
3A0h-3AFh	Primary binary synchronous port
3B0h-3BFh	Graphics adapter
3C0h-3DFh	Graphics adapter
3F0h-3F7h	Floppy disk controller
3F8h-3FFh	Primary serial port (COM1:)

### **Inputting a byte from an I/O port**

The assembly language command to input a byte is:

```
IN AL,DX
```

where DX is the Data Register which contains the address of the input port. The 8-bit value loaded from this address is put into the register AL.

For Turbo/Borland C the equivalent function is `inportb()`. Its general syntax is as follows:

```
value=inportb(PORTADDRESS);
```

where PORTADDRESS is the address of the input port and value is loaded with the 8-bit value from this address. This function is prototyped in the header file dos.h.

For Turbo Pascal the equivalent is accessed via the `port[]` array. Its general syntax is as follows:

```
value:=port[PORTADDRESS];
```

where PORTADDRESS is the address of the input port and value the 8-bit value at this address. To gain access to this function the statement uses `dos` requires to be placed near the top of the program.

Microsoft C++ uses the equivalent `_inp()` function (which is prototyped in `conio.h`).

### **Inputting a word from a port**

The assembly language command to input a word is:

```
IN AX, DX
```

where `DX` is the Data Register which contains the address of the input port. The 16-bit value loaded from this address is put into the register `AX`.

For Turbo/Borland C the equivalent function is `inport()`. Its general syntax is as follows:

```
value=inport(PORTADDRESS);
```

where `PORTADDRESS` is the address of the input port and `value` is loaded with the 16-bit value at this address. This function is prototyped in the header file `dos.h`.

For Turbo Pascal the equivalent is accessed via the `portw[]` array. Its general syntax is as follows:

```
value:=portw[PORTADDRESS];
```

where `PORTADDRESS` is the address of the input port and `value` is the 16-bit value at this address. To gain access to this function the statement `uses dos` requires to be placed near the top of the program.

Microsoft C++ uses the equivalent `_inpw()` function (which is prototyped in `conio.h`).

### **Outputting a byte to an I/O port**

The assembly language command to output a byte is:

```
OUT DX, AL
```

where `DX` is the Data Register which contains the address of the output port. The 8-bit value sent to this address is stored in register `AL`.

For Turbo/Borland C the equivalent function is `outportb()`. Its general syntax is as follows:

```
outportb(PORTADDRESS,value);
```

where `PORTADDRESS` is the address of the output port and `value` is the 8-bit value to be sent to this address. This function is prototyped in the header file `dos.h`.

For Turbo Pascal the equivalent is accessed via the `port[]` array. Its general syntax is as follows:

```
port[PORTADDRESS]:=value;
```

where PORTADDRESS is the address of the output port and value is the 8-bit value to be sent to that address. To gain access to this function the statement uses dos requires to be placed near the top of the program.

Microsoft C++ uses the equivalent `_outp()` function (which is prototyped in `conio.h`).

### **Outputting a word**

The assembly language command to input a byte is:

```
OUT DX,AX
```

where DX is the Data Register which contains the address of the output port. The 16-bit value sent to this address is stored in register AX.

For Turbo/Borland C the equivalent function is `outport()`. Its general syntax is as follows:

```
outport(PORTADDRESS,value);
```

where PORTADDRESS is the address of the output port and value is the 16-bit value to be sent to that address. This function is prototyped in the header file `dos.h`.

For Turbo Pascal the equivalent is accessed via the `port[]` array. Its general syntax is as follows:

```
portw[PORTADDRESS]:=value;
```

where PORTADDRESS is the address of the output port and value is the 16-bit value to be sent to that address. To gain access to this function the statement uses dos requires to be placed near the top of the program.

Microsoft C++ uses the equivalent `_outp()` function (which is prototyped in `conio.h`).



---

## PC Processors

---

### J.1 Introduction

---

Intel marketed the first microprocessor, named the 4004. This device caused a revolution in the electronics industry because previous electronic systems had a fixed functionality. With this processor the functionality could be programmed by software. Amazingly, by today's standards, it could only handle four bits of data at a time (a nibble), contained 2000 transistors, had 46 instructions and allowed 4 KB of program code and 1 KB of data. From this humble start the PC has since evolved using Intel microprocessors (Intel is a contraction of *Integrated Electronics*).

The second generation of Intel microprocessors began in 1974. These could handle 8 bits (a byte) of data at a time and were named the 8008, 8080 and the 8085. They were much more powerful than the previous 4-bit devices and were used in many early microcomputers and in applications such as electronic instruments and printers. The 8008 has a 14-bit address bus and can thus address up to 16 KB of memory (the 8080 has a 16-bit address bus giving it a 64 KB limit).

The third generation of microprocessors began with the launch of the 16-bit processors. Intel released the 8086 microprocessor which was mainly an extension to the original 8080 processor and thus retained a degree of software compatibility. IBM's designers realized the power of the 8086 and used it in the original IBM PC and IBM XT (eXtended Technology). It has a 16-bit data bus and a 20-bit address bus, and thus has a maximum addressable capacity of 1 MB. The 8086 could handle either 8 or 16 bits of data at a time (although in a messy way).

A stripped-down, 8-bit external data bus, version called the 8088 is also available. This stripped-down processor allowed designers to produce less complex (and cheaper) computer systems. An improved architecture version, called the 80286, was launched in 1982, and was used in the IBM AT (Advanced Technology).

In 1985, Intel introduced its first 32-bit microprocessor, the 80386DX. This device was compatible with the previous 8088/8086/80286 (80×86) processors and gave excellent performance, handling 8, 16 or 32 bits at a time. It has a full 32-bit data and address buses and can thus address up to 4 GB of physical memory. A stripped-down 16-bit external data bus and 24-bit address bus version called the 80386SX was released in 1988. This stripped-down processor can thus only access up to 16 MB of physical memory.

In 1989, Intel introduced the 80486DX which is basically an improved 80386DX with a memory cache and math co-processor integrated onto the chip. It had an improved internal structure making it around 50% faster with a comparable 80386. The 80486SX was also introduced, which is merely an 80486DX with the link to the math co-processor broken. Clock doubler/treble 80486 processors were also released. In these the processor runs at a higher speed than the system clock. Typically, systems with clock doubler processors are around

75% faster than the comparable non-doubled processors. Typical clock doubler processors are DX2-66 and DX2-50 which run from 33 MHz and 25 MHz clocks, respectively. Intel have also produced a range of 80486 microprocessors which run at three or four times the system clock speed and are referred to as DX4 processors. These include the Intel DX4-100 (25 MHz clock) and Intel DX4-75 (25 MHz clock).

The Pentium (or P-5) is a 64-bit ‘superscalar’ processor. It can execute more than one instruction at a time and has a full 64-bit (8-byte) data bus and a 32-bit address bus. In terms of performance, it operates almost twice as fast as the equivalent 80486. It also has improved floating-point operations (roughly three times faster) and is fully compatible with previous 80×86 processors.

The Pentium II (or P-6) is an enhancement of the P-5 and has a bus that supports up to four processors on the same bus without extra supporting logic, with clock multiplying speeds of over 300 MHz. It also has major savings of electrical power and the minimization of electromagnetic interference (EMI). A great enhancement of the P-6 bus is that it detects and corrects all single-bit data bus errors and also detects multiple-bit errors on the data bus.

## **J.2 8088 microprocessor**

---

The great revolution in processing power arrived with the 16-bit 8086 processor. This has a 20-bit address bus and a 16-bit data bus, while the 8088 has an 8-bit external data bus. Figure J.1 shows the pin connections of the 8088 and also the main connections to the processor. Many of the 40 pins of the 8086 have dual functions. For example, the lines AD0–AD7 act either as the lower 8 bits of the address bus (A0–A7) or as the lower 8 bits of the data bus (D0–D7). The lines A16/S3–A19/S6 also have a dual function, S3–S6 are normally not used by the PC and thus they are used as the 4 upper bits of the address bus. The latching of the address is achieved when the ALE (address latch enable) goes from a high to a low.

The bus controller (8288) generates the required control signals from the 8088 status lines  $\overline{S_0}$ – $\overline{S_1}$ . For example, if  $\overline{S_0}$  is high,  $\overline{S_1}$  is low and  $\overline{S_2}$  is low then the  $\overline{\text{MEMR}}$  line goes low. The main control signals are:

- $\overline{\text{IOR}}$  (I/O read) which means that the processor is reading from the contents of the address which is on the I/O bus.
- $\overline{\text{IOW}}$  (I/O write) which means that the processor is writing the contents of the data bus to the address which is on the I/O bus.
- $\overline{\text{MEMR}}$  (memory read) which means that the processor is reading from the contents of the address which is on the address bus.
- $\overline{\text{MEMW}}$  (memory write) which means that the processor is writing the contents of the data bus to the address which is on the address bus.
- $\overline{\text{INTA}}$  (interrupt acknowledgement) which is used by the processor to acknowledge an interrupt ( $\overline{S_0}$ ,  $\overline{S_1}$  and  $\overline{S_2}$  all go low). When a peripheral wants the attention of the processor it sends an interrupt request to the 8259 which, if it is allowed, sets INTR high.

The processor either communicates directly with memory (with  $\overline{\text{MEMW}}$  and  $\overline{\text{MEMR}}$ ) or communicates with peripherals through isolated I/O ports (with  $\overline{\text{IOR}}$  and  $\overline{\text{IOW}}$ ).

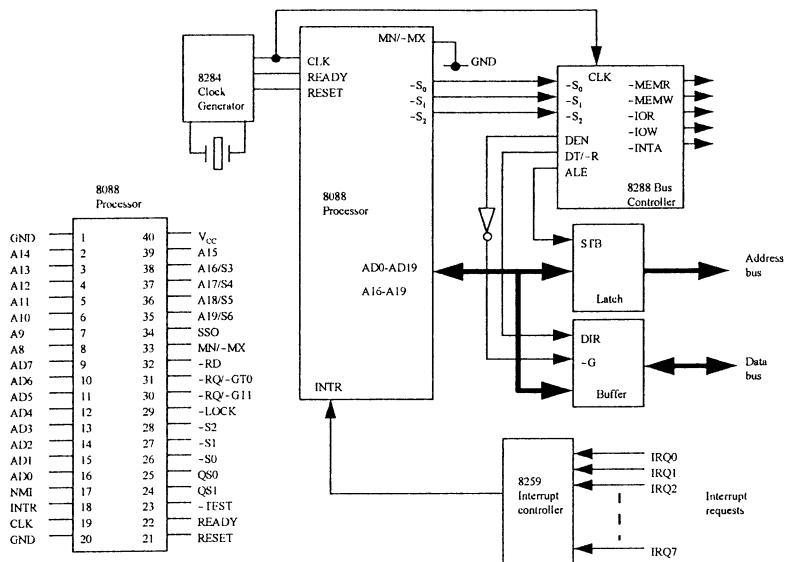


Figure J.1 8088 connections.

### J.2.1 Registers

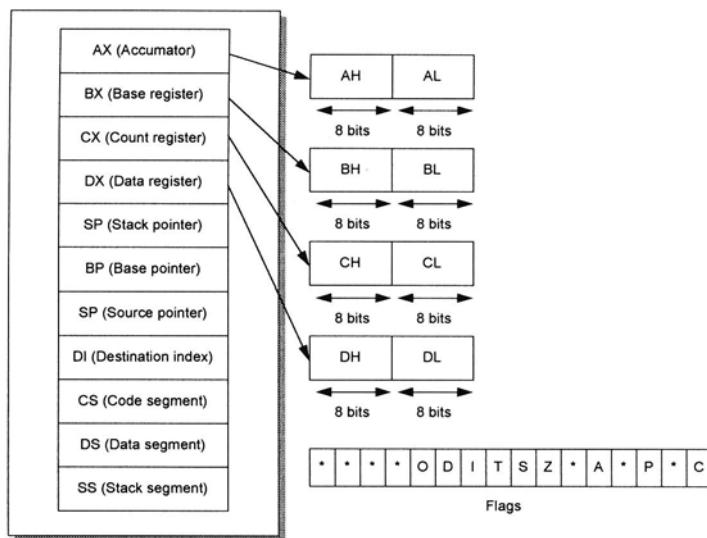
Each of the PC-based Intel microprocessors is compatible with the original 8086 processor and is normally backwardly compatible. Thus, for example, a Pentium can run 8086, 80386 and 80486 code. Microprocessors use registers to perform their operations. These registers are basically special memory locations within the processor that have special names. The 8086/88 has 14 registers which are grouped into four categories, as illustrated in Figure J.2.

#### General-purpose registers

There are four general-purpose registers that are AX, BX, CX and DX. Each can be used to manipulate a whole 16-bit word or with two separate 8-bit bytes. These bytes are called the lower and upper order bytes. Each of these registers can be used as two 8-bit registers, for example, AL represents an 8-bit register that is the lower half of AX and AH represents the upper half of AX.

The AX register is the most general purpose of the four registers and is normally used for all types of operations. Each of the other registers has one or more implied extra functions. These are:

- AX is the accumulator. It is used for all input/output operations and some arithmetic operations. For example, multiply, divide and translate instructions assume the use of AX.
- BX is the base register. It can be used as an address register.
- CX is the count register. It is used by instructions which require to count. Typically it is used for controlling the number of times a loop is repeated and in bit-shift operations.
- DX is the data register. It is used for some input/output and also when multiplying and dividing.



**Figure J.2** 8086/88 registers.

### Addressing registers

The addressing registers are used in memory addressing operations, such as holding the source address of the memory and the destination address. These address registers are named BP, SP, SI and DI, which are:

- SI is the source index and is used with extended addressing commands.
- DI is the destination index and is used in some addressing modes.
- BP is the base pointer.
- SP is the stack pointer.

### Status registers

Status registers are used to test for various conditions in an operation, such as ‘is the result negative’, ‘is the result zero’, and so on. The two status registers have 16 bits and are called the instruction pointer (IP) and the flag register (F):

- IP is the instruction pointer and contains the address of the next instruction of the program.
- Flag register holds a collection of 16 different conditions. Table J.1 outlines the most used flags.

### Segments registers

There are four areas of memory called segments, each of which are 16 bits and can thus address up to 64 KB (from 0000h to FFFFh). These segments are:

- Code segment (cs register). This defines the memory location where the program code (or instructions) is stored.
- Data segment (ds register). This defines where data from the program will be stored (ds

stands for data segment register).

- Stack segment (ss register). This defined where the stack is stored.
- Extra segment (es).

All addresses are with reference to the segment registers.

The 8086 has a segmented memory, the segment registers are used to manipulate memory within these segments. Each segment provides 64 KB of memory, this area of memory is known as the current segment. Segmented memory will be discussed in more detail in Section J.3.

**Table J.1** Processor flags.

Bit	Flag position	Name	Description
C	0	Set on carry	Contains the carry from the most significant bit (left-hand bit) following a shift, rotate or arithmetic operation.
A	4	Set on 1/2 carry	
S	7	Set on negative result	Contains the sign of an arithmetic operation (0 for positive, 1 for negative).
Z	6	Set on zero result	Contains results of last arithmetic or compare result (0 for nonzero, 1 for zero).
O	11	Set on overflow	Indicates that an overflow has occurred in the most significant bit from an arithmetic operation.
P	2	Set on even parity	
D	10	Direction	
I	9	Interrupt enable	Indicates whether the interrupt has been disabled.
T	8	Trap	

### Memory addressing

There are several methods of accessing memory locations, these are:

- Implied addressing which uses an instruction in which it is known which registers are used.
- Immediate (or literal) addressing uses a simple constant number to define the address location.
- Register addressing which uses the address registers for the addressing (such as AX, BX, and so on).
- Memory addressing which is used to read or write to a specified memory location.

---

### J.3 Memory segmentation

The 80386, 80486 and Pentium processors run in one of two modes, either virtual or real. In

virtual mode they act as a pseudo-8086 16-bit processor, known as the protected mode. In real-mode they can use the full capabilities of their address and data bus. This mode normally depends on the addressing capabilities of the operating system. All DOS-based programs use the virtual mode.

The 8086 has a 20-bit address bus so that when the PC is running 8086-compatible code it can only address up to 1 MB of physical memory. It also has a segmented memory architecture and can only directly address 64 KB of data at a time. A chunk of memory is known as a segment and hence the phrase ‘segmented memory architecture’.

Memory addresses are normally defined by their hexadecimal address. A 4-bit address bus can address 16 locations from 0000<sub>b</sub> to 1111<sub>b</sub>. This can be represented in hexadecimal as 0<sub>h</sub> to F<sub>h</sub>. An 8-bit bus can address up to 256 locations from 00<sub>h</sub> to FF<sub>h</sub>. Section J.10 outlines the addressing capabilities for a given address bus size.

Two important addressing capabilities for the PC relate to a 16- and a 20-bit address bus. A 16-bit address bus addresses up to 64 KB of memory from 0000<sub>h</sub> to FFFF<sub>h</sub> and a 20-bit address bus addresses up to 1 MB from 000000<sub>h</sub> to FFFFFFFF<sub>h</sub>. The 80386/80486/ Pentium processors have a 32-bit address bus and can address from 00000000<sub>h</sub> to FFFFFFFF<sub>h</sub>.

A segmented memory address location is identified with a segment and an offset address. The standard notation is *segment:offset*. A segment address is a 4-digit hexadecimal address which points to the start of a 64 KB chunk of data. The offset is also a 4-digit hexadecimal address which defines the address offset from the segment base pointer. This is illustrated in Figure J.3.

The *segment:offset* address is defined as the logical address, the actual physical address is calculated by shifting the segment address 4 bits to the left and adding the offset. The example given next shows that the actual address of 2F84:0532 is 2FD72h.

Segment (2F84):	0010	1111	1000	0100	0000
Offset (0532):		0000	0101	0011	0010
Actual address:	0010	1111	1101	0111	0010

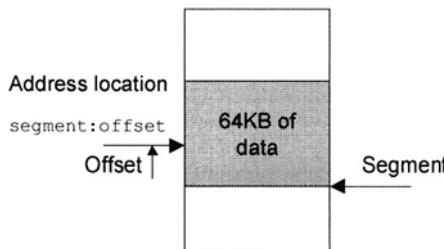


Figure J.3 Memory addressing.

## J.4 80386/80486

### J.4.1 Introduction

The PC had grown from the 8086 processor, which could run 8-bit or 16-bit software. This processor was fine with text-based applications, but struggled with graphical programs, espe-

cially with GUIs (Graphical User Interfaces). The original version of Microsoft Windows (Windows Version 1.0 and Version 2.0) ran on these limited processes. The great leap in computing power came with the development of the Intel 80386 processor and with Microsoft Windows 3.0. A key to the success of the 80386 was that it was fully compatible with the previous 8088/8086/80286 processors. This allowed it run all existing DOS-based program and new 32-bit applications. The DX version has full 32-bit data and address bus and can thus address up to 4 GB of physical memory. An SX version with a stripped-down 16-bit external data bus and 24-bit address bus version can access only up to 16 MB of physical memory (at its time of release this was a large amount of memory). Most of the time, with Microsoft Windows 3.0, the processor was using only 16 bits, and thus not using the full power of the processor.

The 80486DX basically consists of an improved 80386 with a memory cache and a math co-processor integrated onto the chip. An SX version had the link to the math co-processor broken. At the time, a limiting factor was the speed of the system clock (which was limited to around 25 MHz or 33 MHz). Thus clock doubler, treblers or quadrupers allow the processor to multiply the system clock frequency to a high speed. Thus, internal operations of the processor are carried out at much higher speeds, but accesses outside the processor must slow down the system clock. As most of the operations within the computer involve operations within the processor then the overall speed of the computer is improved (roughly by about 75% for a clock doubler).

#### J.4.2 80486 pin out

To allow for easy upgrades and to save space the 80486 and Pentium processors are available in a pin-grid array (PGA) form. A 168-pin PGA 80486 processor is illustrated in Figure J.4.

It can be seen that the 486 processor has a 32-bit address bus (A0–A31) and a 32-bit data bus (D0–D31). The pin definitions are given in Table J.2.

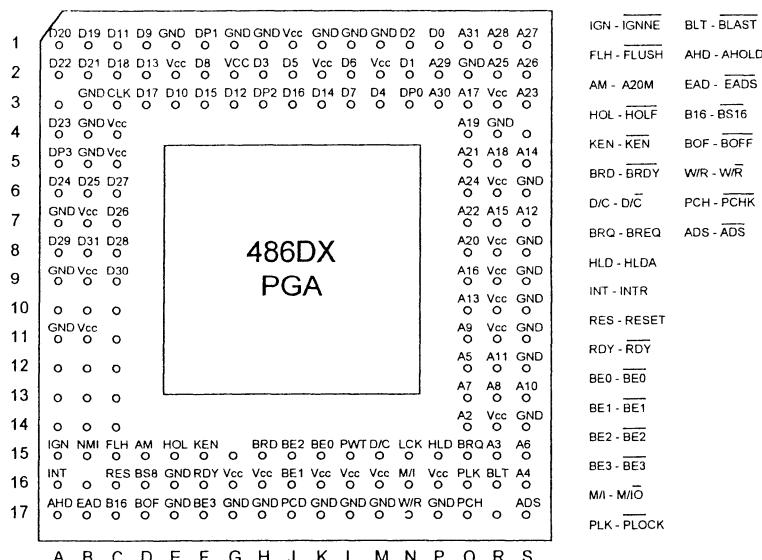


Figure J.4 i486DX processor.

Table J.3 defines the how the control signals are interpreted. For the STOP/special bus cycle, the byte enable signals ( $\overline{\text{BE0}} - \overline{\text{BE3}}$ ) further define the cycle. These are:

- Write back cycle  $\overline{\text{BE0}} = 1, \overline{\text{BE1}} = 1, \overline{\text{BE2}} = 1, \overline{\text{BE3}} = 0$ .
- Halt cycle  $\overline{\text{BE0}} = 1, \overline{\text{BE1}} = 1, \overline{\text{BE2}} = 0, \overline{\text{BE3}} = 1$ .
- Flush cycle  $\overline{\text{BE0}} = 1, \overline{\text{BE1}} = 0, \overline{\text{BE2}} = 1, \overline{\text{BE3}} = 1$ .
- Shutdown cycle  $\overline{\text{BE0}} = 0, \overline{\text{BE1}} = 1, \overline{\text{BE2}} = 1, \overline{\text{BE3}} = 1$ .

**Table J.2** 80486 signal lines.

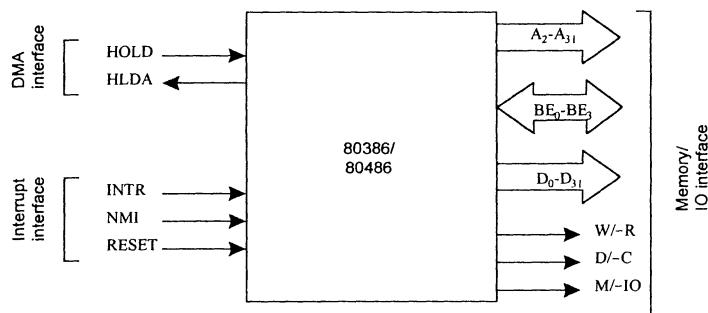
Signals	I/O	Description
A2-A31	I/O	The 30 most significant bits of the address bus.
$\overline{\text{A20M}}$	I	When active low, the processor internally masks the address bit A20 before every memory access.
$\overline{\text{ADS}}$	O	Indicates that the processor has valid control signals and valid address signals.
AHOLD	I	When active a different bus controller can have access to the address bus. This is typically used in a multi-processor system.
$\overline{\text{BE0}} - \overline{\text{BE3}}$	O	The byte enable lines indicate which of the bytes of the 32-bit data bus are active.
$\overline{\text{BLAST}}$	O	Indicates that the current burst cycle will end after the next $\overline{\text{BRDY}}$ signal.
$\overline{\text{BOFF}}$	I	The backoff signal informs the processor to deactivate the bus on the next clock cycle.
$\overline{\text{BRDY}}$	I	The burst ready signal is used by an addressed system that has sent data on the data bus or read data from the bus.
BREQ	O	Indicates that the processor has internally requested the bus.
$\overline{\text{BS16}}, \overline{\text{BS8}}$	I	The $\overline{\text{BS16}}$ signal indicates that a 16-bit data bus is used, the $\overline{\text{BS8}}$ signal indicates that an 8-bit data bus is used. If both are high then a 32-bit data bus is used.
DP0-DP3	I/O	The data parity bits gives a parity check for each byte of the 32-bit data bus. The parity bits are always even parity.
$\overline{\text{EADS}}$	I	Indicates that an external bus controller has put a valid address on the address bus.
$\overline{\text{FERR}}$	O	Indicates that the processor has detected an error in the internal floating-point unit.

<u>FLUSH</u>	I	When active the processor writes the complete contents of the cache to memory.
HOLD, HLDA	I/O	The bus hold (HOLD) and acknowledge (HLDA) are used for bus arbitration and allow other bus controllers to take control of the buses.
<u>IGNNE</u>	I	When active the processor ignores any numeric errors.
INTR	I	External devices to interrupt the processor use the interrupt request line.
<u>KEN</u>	I	This signal stops caching of a specific address.
<u>LOCK</u>	O	If active the processor will not pass control to an external bus controller, when it receives a HOLD signal.
M/ $\overline{IO}$ , D/ $\overline{C}$ , W/ $\overline{R}$	O	See Table J.3.
NMI	I	The non-maskable interrupt signal causes an interrupt 2.
<u>PCHK</u>	O	If it is set active then a data parity error has occurred.
<u>PLOCK</u>	O	The active pseudo lock signal identifies that the current data transfer requires more than one bus cycle.
PWT, PCD	O	The page write-through (PWT) and page cache disable (PCD) are used with cache control.
<u>RDY</u>	I	When active the addressed system has sent data on the data bus or read data from the bus.
RESET	I	If the reset signal is high for more than 15 clock cycles then the processor will reset itself.

The 486 integrates a processor, cache and a math co-processor onto a single IC. Figure J.5 shows the main 80386/80486 processor connections. The Pentium processor connections are similar but it has a 64-bit data bus. There are three main interface connections: the memory/IO interface, interrupt interface and DMA interface.

**Table J.3** Control signals.

M/ $\overline{IO}$	D/ $\overline{C}$	W/ $\overline{R}$	Description
0	0	0	Interrupt acknowledge sequence
0	0	1	STOP/special bus cycle
0	1	0	Reading from an I/O port
0	1	1	Writing to an I/O port
1	0	0	Reading an instruction from memory
1	0	1	Reserved
1	1	0	Reading data from memory
1	1	1	Writing data to memory



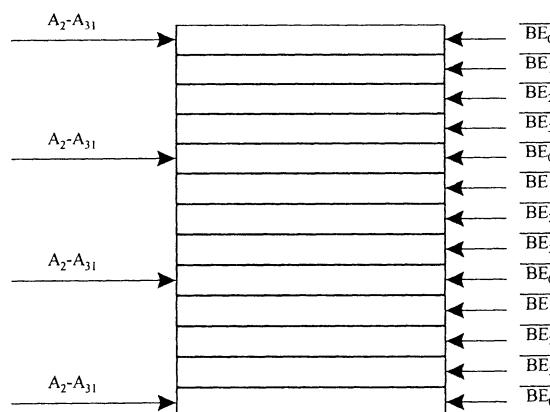
**Figure J.5** Some of the 80386/80486 signal connections.

The write/read ( $W/\bar{R}$ ) line determines whether data is written to ( $W$ ) or read from ( $\bar{R}$ ) memory. PCs can interface directly with memory or can interface to isolated memory. The signal line  $M/\bar{IO}$  differentiates between the two types. If it is high then the processor accesses direct memory; else if it is low then it accesses isolated memory.

The 80386DX and 80486 have an external 32-bit data bus ( $D_0-D_{31}$ ) and a 32-bit address bus ranging from  $A_2$  to  $A_{31}$ . The two lower address lines,  $A_0$  and  $A_1$ , are decoded to produce the byte enable signals  $\overline{BE}_0$ ,  $\overline{BE}_1$ ,  $\overline{BE}_2$  and  $\overline{BE}_3$ . The  $\overline{BE}_0$  line activates when  $A_1A_0$  is 00,  $\overline{BE}_1$  activates when  $A_1A_0$  is 01,  $\overline{BE}_2$  activates when  $A_1A_0$  and  $\overline{BE}_3$  activates when  $A_1A_0$  is 11. Figure J.6 illustrates this addressing.

The byte enable lines are also used to access either 8, 16, 24 or 32 bits of data at a time. When addressing a single byte, only the  $\overline{BE}_0$  line is active ( $D_0-D_7$ ), if 16 bits of data are to be accessed then  $\overline{BE}_0$  and  $\overline{BE}_1$  are active ( $D_0-D_{15}$ ), if 32 bits are to be accessed then  $\overline{BE}_0$ ,  $\overline{BE}_1$ ,  $\overline{BE}_2$  and  $\overline{BE}_3$  are active ( $D_0-D_{31}$ ).

The  $D/\bar{C}$  line differentiates between data and control signals. When it is high then data is read from or written to memory, else if it is low then a control operation is indicated, such as a shutdown command.



**Figure J.6** Memory addressing.

The interrupt lines are interrupt request (INTR), non-maskable interrupt request (NMI) and system reset (RESET), all of which are active high signals. The INTR line is activated when an external device, such as a hard disk or a serial port, wishes to communicate with the processor. This interrupt is maskable and the processor can ignore the interrupt if it wants. NMI is a non-maskable interrupt and is always acted on. When it becomes active the processor calls the non-maskable interrupt service routine. The RESET signal causes a hardware reset and is normally made active when the processor is powered-up.

#### J.4.3 80386/80486 registers

The 80386 and 80486 are 32-bit processors and can thus operate on 32-bits at a time. They have expanded 32-bit registers, which can also be used as either 16-bit or 8-bit registers (mainly to keep compatibility with other processors and software). The general purpose registers, such as AX, BX, CX, DX, SI, DI and BP are expanded from the 8086 processor and are named EAX, EBX, ECX, EDX, ESI, EDI and EBP, respectively, as illustrated in Figure J.7. The CS, SS and DS registers are still 16 bits, but the flag register has been expanded to 32 bits and is named EFLAG.

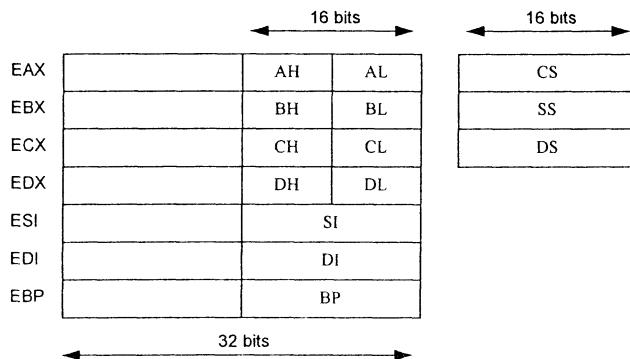


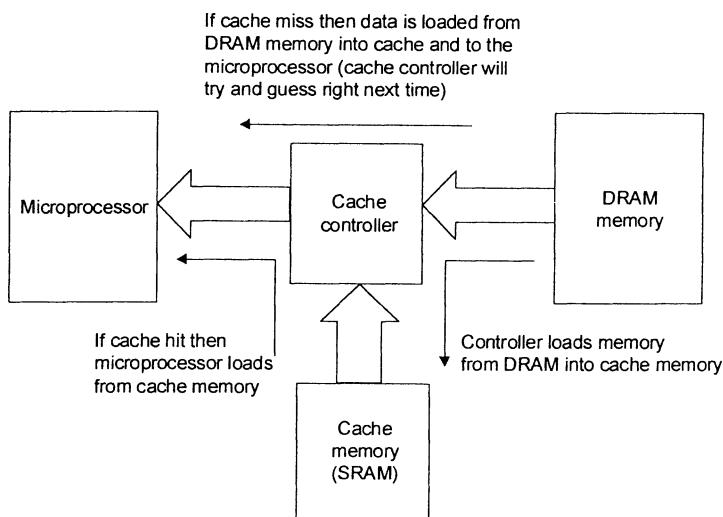
Figure J.7 80386/80486 registers.

#### J.4.4 Memory cache

DRAM is based on the charging and discharge of tiny capacitors. It is thus a relatively slow type of memory compared with SRAM. For example, typical access time for DRAM is 80 ns and a typical motherboard clock speed is 50 MHz. This gives a clock period of 20 ns. Thus, the processor would require five wait states before the data becomes available. A cache memory can be used to overcome this problem. This is a bank of fast memory (SRAM) that uses a cache controller to load data from main memory (typically DRAM) into it. The cache controller guesses the data the processor requires and loads this into the cache memory. Figure J.8 shows that if the controller guesses correctly then it is a cache hit, else if it is wrong it is a cache miss. A miss causes the processor to access the memory in the normal way (that is, there may be wait states as the DRAM memory need time to get the data). Typical cache memory sizes are 16 KB, 32 KB and 64 KB for 80486 processors and 256 KB and 512 KB for Pentium processors. This should be compared with the size of the RAM on a typical PC which is typically at least 32 MB.

The 80486 and Pentium have built-in cache controllers and, at least, 64 KB (or 256 KB

for the Pentium) of local SRAM cache memory. This is a first-level cache and the total cache size can be increased with an off-chip (or near-chip) memory (second-level cache).



**Figure J.8** Cache operation.

### Cache architecture

The main cache architectures are:

- **Look-through cache.** In a look-through cache the system memory is isolated from the processor address and control busses. In this case the processor directly sends a memory request to the cache controller which then determines whether it should forward it to its own memory or the system memory. Figure J.9 illustrates this type of cache. It can be seen that the cache controls whether the processor address contents are latched through to the DRAM memory and it also controls whether the contents of the DRAM's memory is loaded onto the processor data bus (through the data transceiver). The operation is described as bus cycle forwarding.
- **Look-aside cache.** A look-aside cache is where the cache and system memory connect to a common bus. System memory and the cache controller see the beginning of the processor bus cycle at the same time. If the cache controller detects a cache hit then it must inform the system memory before it tries to find the data. If a cache miss is found then the memory access is allowed to continue.
- **Write-through cache.** With a write-through cache all memory address accesses are seen by the system memory when the processor performs a bus cycle.
- **Write-back cache.** With a write-back cache the cache controller controls all system writes. It thus does not write the system memory unless it has to.

### Second-level caches

An L1-cache (first-level cache) provides a relatively small on-chip cache, where an L2-cache (second-level cache) provides an external, on-board, cache, which provides a cache memory

of between 128 and 512 KB. The processor looks in its own L1-cache for a cache hit; if none is found then it searches in the on-board L2-cache. A cache hit in the L1-cache will obviously be faster than the off-chip cache.

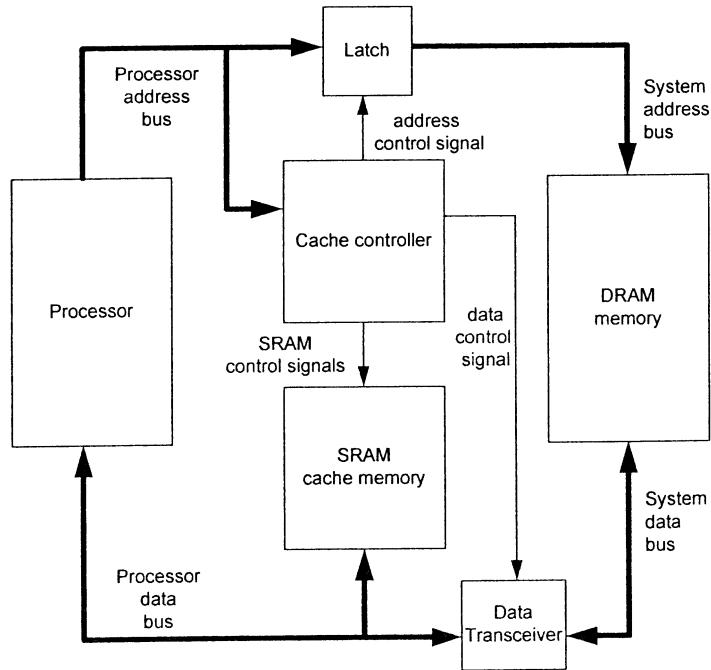


Figure J.9 Look-through cache.

## J.5 Pentium/Pentium Pro

---

### J.5.1 Introduction

Intel have gradually developed their range of processors from the original 16-bit 8086 processor to the 32-bit processors, such as the Pentium II. Table J.1 contrasts the Intel processor range. It can also be seen from the table that the Pentium II processor is nearly more than a thousand times more powerful than an 8086 processor. The original 8086 had just 29 000 transistors and operated at a clock speed of 8 MHz. It had an external 20-bit bus and could thus only access up to 1 MB of physical memory. Compare this with the Pentium II which can operate at over 300 MHz, contains over 6 000 000 transistors and can access up to 64 GB of physical memory.

**Table J.4** Processor comparison.

<i>Processor</i>	<i>Clock (when released)</i>	<i>Register size</i>	<i>External data bus</i>	<i>Max. external memory</i>	<i>Cache</i>	<i>Power (MIPs)</i>
8086	8 MHz	16	16	1 MB		0.8
286	12.5 MHz	16	16	16 MB		2.7
386DX	20 MHz	32	32	4 GB		6.0
486DX	25 MHz	32	32	4 GB	8 KB L-1	20
Pentium	60 MHz	32	64	4 GB	16 KB L-1	100
Pentium Pro	200 MHz	32	64	64 GB	16 KB L-1 256 K L-2	440
Pentium II	200 MHz	32	64	64 GB	16 KB L-1 512 KB L-2	700

## J.6 Intel processor development

The 80386 processor was a great leap in processing power over the 8086 and 80286, but it required an on-board math co-processor to enhance its mathematical operations and it could also only execute one instruction at a time. The 80486 brought many enhancements, such as:

- The addition of parallel execution with the expansion of the Instruction decode and execution units into five pipelined stages. Each of these stages operate in parallel, with the others, on up to five instructions in different stages of execution. This allows up to five instructions to be completed at a time.
- The addition of an 8 KB on-chip cache to greatly reduce the data and code access times.
- The addition of an integrated floating-point unit.
- Support for more complex and powerful systems, such as off-board L-2 cache support and multiprocessor operation.

With the increase in notebook and palmtop computers, the 80486 was also enhanced to support many energy and system management capabilities. These processors were named the 80486SL processors. The new enhancements included:

- System Management Mode. This mode is triggered by the processor's own interrupt pin and allows complex system management features to be added to a system transparently to the operating system and application programs.
- Stop Clock and Auto Halt Powerdown. These allow the processor to either shut itself down (and preserve its current state) or run at a reduced clock rate.

The Intel Pentium processor added many enhancements to the previous processors, including:

- The addition of a second execution pipeline. These two pipelines, named u and v, can execute two instructions per clock cycle. This is known as superscalar operation.
- Increased on-chip L-1 cache, 8 KB for code and another 8 KB for data. It uses the MESI

protocol to support write-back mode, as well as the write-through mode (which is used by the 80486 processor).

- Branch prediction with an on-chip branch table that improves looping characteristics.
- Enhancement to the virtual-8086 mode to allow for 4 MB as well as 4 KB pages.
- 128-bit and 256-bit data paths are possible (although the main registers are still 32 bits).
- Burstable 64-bit external data bus.
- Addition of Advanced Programmable Interrupt Controller (APIC) to support multiple Pentium processors.
- New dual processing mode to support dual processor systems.

The Pentium processor has been extremely successful and has helped support enhanced multitasking operating systems such as Windows NT and Windows 95/98. The Intel Pentium Pro enhanced the Pentium processor with the following:

- Addition of a 3-way superscalar architecture, as opposed to a 2-way for the Pentium. This allows three instructions to be executed for every clock cycle.
- Uses enhanced prediction of parallel code (called dynamic execution micro-architecture) for the superscalar operation. This includes methods such as micro-data flow analysis, out-of-order execution, enhanced branch prediction and speculative execution. The three instruction decode units work in parallel to decode object code into smaller operations called micro-ops. These micro-ops then go into an instruction pool, and, when there are no interdependencies they can be executed out-of-order by the five parallel execution units (two integer units, two for floating-point operations and one for memory). A retirement unit retires completed micro-ops in their original program order and takes account of any branches. This recovers the original program flow.
- Addition of register renaming. Multiple instructions not dependent on each other, using the same registers, allow the source and destination registers to be temporarily renamed. The original register names are used when instructions are retired and program flow is maintained.
- Addition of a closely coupled, on-package, 256 KB L-2 cache that has a dedicated 64-bit full clock speed bus. The L-2 cache also supports up to four concurrent accesses through a 64-bit external data bus. Each of these accesses is transaction-oriented where each access is handled as a separate request and response. This allows for numerous requests while awaiting a response.
- Expanded 36-bit address bus to give a physical address size of 64 GB.

The Pentium II processor is a further enhancement to the processor range. Apart from increasing the clock speed it has several enhancements over the Pentium Pro, including:

- Integration of MMX technology. MMX instructions support high-speed multimedia operations and include the addition of eight new registers (MM0 to MM7), four MMX data types and an MMX instruction set.
- Single edge contact (SEC) cartridge packaging. This gives improved handling performance and socketability. It uses surface mount component and has a thermal plate (which accepts a standard heat sink), a cover and a substrate with an edge finger connection.
- Integrated on-chip L-1 cache 16 KB for code and another 16 KB for data.
- Increased size, on-package, 512 KB L-2 cache.

- Enhanced low-power states, such as AutoHALT, Stop-Grant, Sleep and Deep Sleep.

## **J.7 Terms**

---

Before introducing the Pentium Pro (P-6) various terms have to be defined. These are:

Transaction	Used to define a bus cycle. It consists of a set of phases, which relate to a single bus request.
Bus agent	Devices that reside on the processor bus, that is, the processor, PCI bridge and memory controller.
Priority agent	The device handling reset, configuration, initialization, error detection and handling; generally the processor-to-PCI bridge.
Requesting agent	The device driving the transaction, that is, busmaster.
Addressed agent	The slave device addressed by the transaction, that is, target agent.
Responding agent	The device that provides the transaction response on $\overline{\text{RS2}} - \overline{\text{RS0}}$ signals.
Snooping agent	A caching device that snoops on the transactions to maintain cache coherency.
Implicit write-back	When a hit to a modified line is detected during the snoop phase, an implicit write-back occurs. This is the mechanism used to write-back the cache line.

## **J.8 Pentium II and Pentium Pro**

---

A major objective of electronic systems design is the saving of electrical power and the minimization of electromagnetic interference (EMI). Thus gunning transceiver logic (GTL) has been used to reduce both power consumption and EMI as it has a low voltage swing. GTL requires a 1 V reference signal and signals which use GTL logic are terminated to 1.5 V. If a signal is 0.2 V above the reference voltage, that is, 1.2 V, then it is considered high. If a signal is 0.2 V below the reference voltage, that is, 0.8 V, then it is considered low.

The Pentium Pro and II support up to four processors on the same bus without extra supporting logic. Integrated into the bus structure are cache coherency signals, advanced programmable interrupt control signals and bus arbitration.

A great enhancement of the Pentium Pro bus is data error detection and correction. The Pentium Pro bus detects and corrects all single-bit data bus errors and also detects multiple-

bit errors on the data bus. Address and control bus signals also have basic parity protection.

The Pentium Pro bus has a modified line write-back performed without backing off the current bus owner, where the processor must perform a write-back to memory when it detects a hit to a modified line. The following mechanism eliminates the need to back-off the current busmaster. If a memory write is being performed by the current bus owner then two writes will be seen on the bus, that is, the original one followed by the write-back. The memory controller latches, and merges the data from the two cycles, and performs one write to DRAM. If the current bus owner is performing a memory read then it accepts the data when it is being written to memory.

Other enhanced features are:

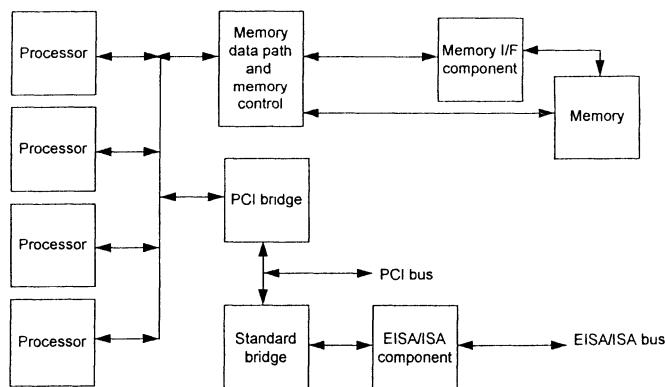
- Deferred reply transactions stop the processor from having to wait for slow devices; transactions that require a long time can be completed later, that is, deferred.
- Deeply pipelined bus transactions where the bus supports up to eight outstanding pipelined transactions.

## J.9 System overview

---

Figure J.01 outlines the main components of a Pentium. A major upgrade is the support for up to four processors. The memory control and data path control logic provides the memory control signals, that is, memory address,  $\overline{\text{RAS}}$  and  $\overline{\text{CAS}}$  signals. The data path logic moves the data between the processor bus and the memory data bus. The memory interface component interfaces the memory data bus with the DRAM memory. Both interleaved and non-interleaved methods are generally supported. The memory consists of dual-in-line memory modules, that is, DIMMs. A DIMM module supports 64 data bits, and 8 parity or ECC bits.

The PCI bridge provides the interface between the processor bus and the PCI bus. The standard bridge provides an interface between the PCI bus and the EISA / ISA bus. EISA / ISA Support Component provides the EISA / ISA bus support functions, for example, timers, interrupt control, flash ROM, keyboard interface, LA/SA translation and XD bus control.



**Figure J.10** P-6 architecture.

## J.10 Memory address reference

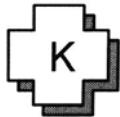
---

<i>Address bus size</i>	<i>Addressable memory (bytes)</i>	<i>Address bus size</i>	<i>Addressable memory (bytes)</i>
1	2	15	32K
2	4	16	64K
3	8	17	128K
4	16	18	256K
5	32	19	512K
6	64	20	1M†
7	128	21	2M
8	256	22	4M
9	512	23	8M
10	1K*	24	16M
11	2K	25	32M
12	4K	26	64M
13	8K	32	4G‡
14	16K	64	16GG

\* 1K represents 1024

† 1M represents 1 048 576 (1024 K)

‡ 1G represents 1 073 741 824 (1024 M)



---

# HTML Reference

---

## K.1 Introduction

---

### K.1.1 Data Characters

Characters which are not markup text are mapped directly to strings of data characters. An ampersand followed by a character reference or a number value can be used to define a character. Table K.1 defines these characters (the equivalent ampersand character reference is given in brackets). For example:

```
Fred&#174&ampBert&iquest;
```

will be displayed as:

```
Fred®&Bert;`
```

An ampersand is only recognized as markup when it is followed by a letter or a `#' and a digit:

```
Fred & Bert
```

will be displayed as:

```
Fred & Bert
```

In the HTML document character set only three control characters are allowed: Horizontal Tab, Carriage Return, and Line Feed (code positions 9, 13, and 10).

**Table K.1** Character mappings.

&#00-&#08	Unused	&#09	Horizontal tab
&#10	Line feed	&#11,&#12	Unused
&#13	Carriage Return	&#14-&#31	Unused
&#32	Space	&#33	Exclamation mark
&#34	Quotation mark (&quot)	&#35	Number sign
&#36	Dollar sign	&#37	Percent sign
&#38	Ampersand (&amp)	&#39	Apostrophe
&#40	Left parenthesis	&#41	Right parenthesis
&#42	Asterisk	&#43	Plus sign
&#44	Comma	&#45	Hyphen
&#46	Period (fullstop)	&#47	Solidus

&#48-&#57	Digits 0-9	&#58	Colon
&#59	Semi-colon	&#60	Less than (&lt)
&#61	Equals sign	&#62	Greater than (&gt)
&#63	Question mark	&#64	Commercial at
&#65-&#90	Letters A-Z	&#91	Left square bracket
&#92	Reverse solidus ()	&#93	Right square bracket
&#94	Caret	&#95	Underscore
&#96	Acute accent	&#97-&#122	Letters a-z
&#123	Left curly brace	&#124	Vertical bar
&#125	Right curly brace	&#126	Tilde
&#127-&#159	Unused		
&#160	Non-breaking Space (&nbsnbsp)		
&#161	Inverted exclamation, ¡ (&iexcl)		
&#162	Cent sign ¢ (&cent)		
&#163	Pound sterling £ (&pound)		
&#164	General currency sign,¤ (&curren)		
&#165	Yen sign, ¥ (&yen)		
&#166	Broken vertical bar,   (&brvbar)		
&#167	Section sign, § (&sect)		
&#168	Umlaut, „ (&uml)		
&#169	Copyright, © (&copy)		
&#170	Feminine ordinal, ª (&ordf)		
&#171	Left angle quote, « (&laquo)		
&#172	Not sign, ¬ (&not)		
&#173	Soft hyphen, – (&shy)		
&#174	Registered trademark, ® (&reg)		
&#175	Macron accent, — (&macr)		
&#176	Degree sign, ° (&deg)		
&#177	Plus or minus, ± (&plusmn)		
&#178	Superscript two, ¨ (&sup2)		
&#179	Superscript three, ¸ (&sup3)		
&#180	Acute accent, ´ (&acute)		
&#181	Micro sign, µ (&micro)		
&#182	Paragraph sign, ¶ (&para)		
&#183	Middle dot, · (&middot)		
&#184	Cedilla, . (&cedil)		
&#185	Superscript one, ¸ (&sup1)		
&#186	Masculine ordinal, º (&ordm)		
&#187	Right angle quote, » (&raquo)		
&#188	Fraction one-fourth, ¼ (&frac14)		
&#189	Fraction one-half, ½ (&frac12)		
&#190	Fraction three-fourths, ¾ (&frac34)		
&#191	Inverted question mark, ¿ (&iquest)		
&#192	Capital A, grave accent, Á (&Agrave)		
&#193	Capital A, acute accent, Á (&Aacute)		
&#194	Capital A, circumflex accent, Â (&Acirc)		
&#195	Capital A, tilde, Ã (&Atilde)		

&#196	Capital A, dieresis, Ä (&Auml)
&#197	Capital A, ring, Å (&Aring)
&#198	Capital AE dipthong, Æ (&AElig)
&#199	Capital C, cedilla, Ç (&Ccedil)
&#200	Capital E, grave accent, È (&Egrave)
&#201	Capital E, acute accent, É (&Eacute)
&#202	Capital E, circumflex accent, Ê (&Ecirc)
&#203	Capital E, dieresis, Ë (&Euml)
&#204	Capital I, grave accent, Ì (&Igrave)
&#205	Capital I, acute accent, Í (&Iacute)
&#206	Capital I, circumflex accent, Î (&Icirc)
&#207	Capital I, dieresis, Ï (&Iuml)
&#208	Capital Eth, Icelandic, Ð (&ETH)
&#209	Capital N, tilde, Ñ (&Ntilde)
&#210	Capital O, grave accent, Ò (&Ograve)
&#211	Capital O, acute accent, Ó (&Oacute)
&#212	Capital O, circumflex accent, Ô (&Ocirc)
&#213	Capital O, tilde, Õ (&Otilde)
&#214	Capital O, dieresis, Ö (&Ouml)
&#215	Multiply sign, × (&times)
&#216	Capital O, slash, Ø (&Oslash)
&#217	Capital U, grave accent, Ù (&Ugrave)
&#218	Capital U, acute accent, Ú (&Uacute)
&#219	Capital U, circumflex accent, Û (&Ucirc)
&#220	Capital U, dieresis or umlaut mark, Ü (&Uuml)
&#221	Capital Y, acute accent, Ý (&Yacute)
&#222	Capital THORN, Icelandic, Þ (&THORN)
&#223	Small sharp s, German, ß (&szlig)
&#224	Small a, grave accent, à (&agrave)
&#225	Small a, acute accent, á (&aacute)
&#226	Small a, circumflex accent, â (&acirc)
&#227	Small a, tilde, ã (&atilde)
&#228	Small a, dieresis or umlaut mark, ä (&auml)
&#229	Small a, ring, å (&aring)
&#230	Small ae dipthong, æ (&aelig)
&#231	Small c, cedilla, ç (&ccedil)
&#232	Small e, grave accent, è (&egrave)
&#233	Small e, acute accent, é (&eacute)
&#234	Small e, circumflex accent, ê (&ecirc)
&#235	Small e, dieresis or umlaut mark, ë (&euml)
&#236	Small i, grave accent, ï (&igrave)
&#237	Small i, acute accent, í (&iacute)
&#238	Small i, circumflex accent, î (&icirc)
&#239	Small i, dieresis or umlaut mark, ï (&iuml)
&#240	Small eth, Icelandic, ð (&eth)
&#241	Small n, tilde, ñ (&ntilde)
&#242	Small o, grave accent, ò (&ograve)

&#243	Small o, acute accent, ó (&oacute)
&#244	Small o, circumflex accent, ô (&ocirc)
&#245	Small o, tilde, õ (&otilde)
&#246	Small o, dieresis or umlaut mark, ö (&ouml)
&#247	Division sign, ÷ (&divide)
&#248	Small o, slash, ø (&oslash)
&#249	Small u, grave accent, ù (&ugrave)
&#250	Small u, acute accent, ú (&uacute)
&#251	Small u, circumflex accent, û (&ucirc)
&#252	Small u, dieresis or umlaut mark, ü (&uuml)
&#253	Small y, acute accent, ý (&yacute)
&#254	Small thorn, Icelandic, þ (&thorn),
&#255	Small y, dieresis or umlaut mark, ÿ (&yuml)

### K.1.2 Tags

Tags are used to delimit elements such as headings, paragraphs, lists, character highlighting and links. Normally an HTML element consists of a start-tag, which gives the element name and attributes, followed by the content and then the end tag. A start-tags is defined between a ‘<’ and ‘>’, and end tags between a ‘</’ and ‘>’. For example to display text as bold:

```
<B>Header Level 1</B>
```

Some of the HTML only require a single start tag, these include:

</BR>. Line break.	</P>. Paragraph.
</LI>. List Item.	</DT>. Definition term.
</DD>. Definition Description.	

Element content is a sequence of data character strings and nested elements. Some elements, such as anchors, cannot be nested.

### K.1.3 Names

Names consist of a letter followed by letters, digits, periods or hyphens (normally limited to 72 characters). Entity names are case sensitive, but element and attribute names are not. For example:

‘<FONT>’, ‘<Font>’, and ‘<font>’

are equivalent, but

‘&lt’ is different from ‘&LT’.

Start-tags always begin directly after the opening delimiter (‘<’).

### K.1.4 Attributes

In a start-tag, white space and attributes are allowed between the element name and the closing delimiter. Attributes typically consist of an attribute name, an equal sign, and a value, which can be:

- A string literal, delimited by single quotes or double quotes and not containing any occurrences of the delimiting character.
- A name token (a sequence of letters, digits, periods, or hyphens). Name tokens are not case sensitive.

### K.1.5 Comments

Comments are defined with a '<!' and ends with a '>'. Each comment starts with '--' and includes all text up to and including the next occurrence of '--'. When defining a comment, white space is allowed after each comment, but not before the first comment. The entire comment declaration is ignored.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HEAD>
<TITLE>Comment Document</TITLE>
<!-- Comment field 1 -->
<!-- Comment field 2 -->
<!-->
</HEAD> <BODY>
```

### K.1.6 HTML Public Text Identifiers

Documents that conform to the HTML 2.0 specification can include the following line at the start of the document:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

---

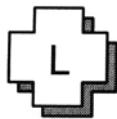
## K.2 Document structure and block structuring

An HTML document is a tree of elements, including a head and body, headings, paragraphs, lists, and so on. These include:

<HTML>	Document element. Consists of a head and a body. The head contains the title and optional elements, and the body is the main text consisting of paragraphs, lists and other elements.
<HEAD>	Head element. An unordered collection of information about the document.
<TITLE>	Title. Identifies the contents of the document in a global context.
<BASE>	Base address. Provides a base address for interpreting relative URLs when the document is read out of context.
<ISINDEX>	Keyword index. Indicates that the user agent should allow the user to search an index by giving keywords.
<LINK>	Link. Represents a hyperlink ( <i>see</i> Hyperlinks) and has the same attributes as the <A> element.
<META>	Associated meta-information. A container for identifying specialized document meta-information.
<BODY>	Body element. Contains the text flow of the document, including headings, paragraphs, lists, etc.
<H1>...<H6>	Headings. The six heading elements, <H1> to <H6> identify section headings. Typical renderings are:

H1	Bold, very-large centered font. One or two blank lines above and below.
H2	Bold, large flush-left font. One or two blank lines above and below.
H3	Italic, large font, slightly indented from the left margin. One or two blank lines above and below.
H4	Bold, normal font, indented more than H3. One blank line above and below.
H5	Italic, normal font, indented as H4. One blank line above.
H6	Bold, indented same as normal text, more than H5. One blank line above.
<P>	Paragraph. Indicates a paragraph. Typically, paragraphs are surrounded by a vertical space of one line or half a line. The first line in a paragraph is indented in some cases.
<PRE>	Preformatted text. Represents a character cell block of text and can be used to define monospaced font. It may be used with the optional WIDTH attribute, which specifies the maximum number of characters for a line.
<ADDRESS>	Address. Contains information such as address, signature and authorship. It is often used at the beginning or end of the body of a document.
<BLOCKQUOTE>	Block quote. Contains text quoted from another source. A typical rendering is a slight extra left and right indent, and/or italic font, and typically provides space above and below the quote.
<UL>, <LI>	Unordered List. <UL> represents a list of items and is typically rendered as a bulleted list. The content of a <UL> element is a sequence of <LI> elements.
<OL>	Ordered List. <OL> represents an ordered list of items which are sorted by sequence or order of importance and is typically rendered as a numbered list. The content of a <OL> element is a sequence of <LI> elements.
<DIR>	Directory List. <DIR> is similar to the <UL> element and represents a list of short items. The content of a <DIR> element is a sequence of <LI> elements.
<MENU>	Menu List. <MENU> is a list of items with typically one line per item. It is typically a more compact than an unordered list. The content of a <MENU> element is a sequence of <LI> elements.
<DL>, <DT>, <DD>	Definition list. Lists terms and corresponding definitions. Definition lists are typically formatted with the term flush-left and the definition, formatted paragraph style, indented after the term. The content of a <DL> element is a sequence of <DT> elements and/or <DD> elements, usually in pairs.
<CITE>	Citation. <CITE> is used to indicate the title of a book or other citation. It is typically rendered as italics.
<CODE>	Code. <CODE> indicates an example of code and is typically rendered in a mono-spaced font. It is intended for short words or phrases of code.
<EM>	Emphasis. <EM> indicates an emphasized phrase and is typically rendered as italics.
<KBD>	Typed text. <KBD> indicates text typed by a user and is typically rendered in a mono-spaced font.

<SAMP>	Literal characters. <SAMP> indicates a sequence of literal characters and is typically rendered in a mono-spaced font.
<STRONG>	Strong emphasis. <STRONG> indicates strong emphasis and is typically rendered in bold.
<VAR>	Placeholder variable. <VAR> indicates a placeholder variable and is typically rendered as italic.
<B>	Bold. <B> indicates bold text.
<I>	Italic. <I> indicates italic text.
<TT>	Teletype. <TT> indicates teletype (monospaced) text.
<A>	Anchor. The <A> element indicates a hyperlink . Attributes of the <A> element are: HREF   URI of the head anchor of a hyperlink. NAME   Name of the anchor. TITLE   Advisory title of the destination resource. REL     The REL attribute gives the relationship(s) described by the hyperlink. REV     Same as the REL attribute, but the semantics of the relationship are in the reverse direction. URN     Specifies a preferred, more persistent identifier for the head anchor of the hyperlink.
	METHODS Specifies methods to be used in accessing the destination, as a whitespace-separated list of names.
 	Line Break.   specifies a line break between words.
<HR>	Horizontal Rule. <HR> is a divider between sections of text and is typically a full width horizontal rule.
<IMG>	Image. <IMG> refers to an image or icon. Attributes are: ALIGN   alignment of the image with respect to the text baseline: • ‘TOP’ specifies that the top of the image aligns with the tallest item on the line containing the image. • ‘MIDDLE’ specifies that the center of the image aligns with the baseline of the line containing the image. • ‘BOTTOM’ specifies that the bottom of the image aligns with the baseline of the line containing the image. ALT     Text to use in place of the referenced image resource. ISMAP   Indicates an image map. SRC     Specifies the URI of the image resource.



---

# Java Reference

---

## L.1 Package *java.applet*

---

### L.1.1 Class *java.applet.Applet*

The Applet class is a superclass of any applet. It provides a standard interface between applets and their environment. The following are defined:

```
// Constructors
public Applet();

// Methods
public void destroy();
public AppletContext getAppletContext();
public String getAppletInfo();
public AudioClip getAudioClip(URL url);
public AudioClip getAudioClip(URL url, String name);
public URL getCodeBase();
public URL getDocumentBase();
public Image getImage(URL url);
public Image getImage(URL url, String name);
public String getLocale();
public String getParameter(String name); // Java 1.1
public String[][] getParameterInfo();
public void init();
public boolean isActive();
public void play(URL url);
public void play(URL url, String name);
public void resize(Dimension d);
public void resize(int width, int height);
public final void setStub(AppletStub stub);
public void showStatus(String msg);
public void start();
public void stop();
```

### L.1.2 Interface *java.applet.AppletContext*

The AppletContext interface corresponds to the applet's environment. The following are defined:

```
// Methods
public abstract Applet getApplet(String name);
public abstract Enumeration getApplets();
public abstract AudioClip getAudioClip(URL url);
public abstract Image getImage(URL url);
public abstract void showDocument(URL url);
public abstract void showDocument(URL url, String target);
public abstract void showStatus(String status);
```

### L.1.3 Interface *java.applet.AppletStub*

The AppletStub interface acts as the interface between the applet and the browser environment or applet viewer environment. The following are defined:

```
// Methods
public abstract void appletResize(int width, int height);
public abstract AppletContext getAppletContext();
public abstract URL getCodeBase();
public abstract URL getDocumentBase();
public abstract String getParameter(String name);
public abstract boolean isActive();
```

### L.1.4 Interface *java.applet.AudioClip*

The AudioClip interface is a simple abstraction for playing a sound clip. Multiple AudioClip items can be playing at the same time, and the resulting sound is mixed together to produce a composite. The following are defined:

```
// Methods
public abstract void loop();
public abstract void play();
public abstract void stop();
```

---

## L.2 Package *java.awt*

---

### L.2.1 Class *java.awt.BorderLayout*

The BorderLayout class contains members named “North”, “South”, “East”, “West”, and “Center”. These are laid out with a given size and constraints. The “North” and “South” components can be stretched horizontally and the “East” and “West” components can be stretched vertically. The “Center” component can be stretched horizontally and vertically. The following are defined:

```
// Constructors
public BorderLayout();
public BorderLayout(int hgap, int vgap);
// Constants
public static final String CENTER;                                // Java 1.1
public static final String EAST;                                     // Java 1.1
public static final String NORTH;                                    // Java 1.1
public static final String SOUTH;                                    // Java 1.1
public static final String WEST;                                     // Java 1.1
// Methods
public void addLayoutComponent(Component comp, Object obj);        // Java 1.1
public void addLayoutComponent(String name, Component comp);        // Java 1.0
public int getHgap();                                              // Java 1.1
public float getLayoutAlignmentX(Container parent);                // Java 1.1
public float getLayoutAlignmentY(Container parent);                // Java 1.1
public int getVgap();                                              // Java 1.1
public void invalidateLayout(Container target);                   // Java 1.1
public void layoutContainer(Container target);                     // Java 1.1
public Dimension maximumLayoutSize(Container target);            // Java 1.1
public Dimension minimumLayoutSize(Container target);
```

```

public Dimension preferredLayoutSize(Container target);
public void removeLayoutComponent(Component comp);
public int setHgap();                                         // Java 1.1
public int setVgap();                                         // Java 1.1
public String toString();

```

### L.2.2 Class *java.awt.Button*

The Button class creates labelled buttons, which can have an associated action when pushed. Three typical actions are: normal, when it has the input focus (the darkening of the outline lets the user know that this is an active object) and when the user clicks the mouse over the button. The following are defined:

```

// Constructors
public Button();
public Button(String label);

// Methods
public synchronized void addActionListener(ActionListern l);           // Java 1.1
public void addNotify();
public String getActionCommand();                                         // Java 1.1
public String getLabel();
protected String paramString();
public synchronized void removeActionListener(ActionListener l);          // Java 1.1
public setActionCommand(String command);                                     // Java 1.1
public void setLabel(String label);

```

### L.2.3 Class *java.awt.Checkbox*

The Checkbox class contains a checkbox which has an on/off state. The following are defined:

```

// Constructors
public Checkbox();
public Checkbox(String label);
public Checkbox(String label, boolean state);                           // Java 1.1
public Checkbox(String label, boolean state, Checkbox group);         // Java 1.1
public Checkbox(String label, CheckboxGroup group, boolean state);

// Methods
public synchronized void addItemListener(ItemListener l);             // Java 1.1
public void addNotify();
public CheckboxGroup getCheckboxGroup();
public String getLabel();
public boolean getState();
public Object[] getSelectedObject();                                    // Java 1.1
protected String paramString();
public synchronized void removeItemListener(ItemListener l);           // Java 1.1
public void setCheckboxGroup(CheckboxGroup g);
public void setLabel(String label);
public void setState(boolean state);

```

### L.2.4 Class *java.awt.CheckboxGroup*

The CheckGroup class groups a number of checkbox buttons. Only one of the checkboxes can be true (on) at a time. When one button is made true (on) the others will become false (off). The following are defined:

```
// Constructors
```

```

public CheckboxGroup();

// Methods
public Checkbox getCurrent();                                // Java 1.0
public Checkbox getSelectedCurrent();                         // Java 1.1
public void setCurrent(Checkbox box);                      // Java 1.0
public void setSelectedCheckbox(Checkbox box);             // Java 1.1
public String toString();

```

### **L.2.5 Class *java.awt.CheckboxMenuItem***

The CheckboxMenuItem class allows for a checkbox that can be included in a menu. The following are defined:

```

// Constructors
public CheckboxMenuItem();                                     // Java 1.1
public CheckboxMenuItem(String label);                        // Java 1.1
public CheckboxMenuItem(String label, boolean state);        // Java 1.1

// Methods
public synchronized void addItemClickListener(ItemListener l); // Java 1.1
public void addNotify();
public boolean getState();
public synchronized Object[] getSelectObjects();              // Java 1.1
public String paramString();
public synchronized void removeItemClickListener(ItemListener l); // Java 1.1
public void setState(boolean t);

```

### **L.2.6 Class *java.awt.Choice***

The Choice class allows for a pop-up menu. The following are defined:

```

// Constructors
public Choice();

// Methods
public synchronized add(String item);                           // Java 1.1
public void addItem(String item);
public synchronized void addItemClickListener(ItemListener l); // Java 1.1
public void addNotify();
public int countItems();                                       // Java 1.0
public String getItem(int index);
public String getItemCount();                                  // Java 1.1
public int getSelectedIndex();
public String getSelectedItem();
protected String paramString();
public synchronized Object[] getSelectedObjects();            // Java 1.1
public synchronized void insert(String item, int index);     // Java 1.1
public synchronized void remove(String item);                 // Java 1.1
public synchronized void remove(int position);               // Java 1.1
public synchronized void removeAll();                          // Java 1.1
public synchronized void removeItemClickListener(ItemListener l); // Java 1.1
public void select(int pos);
public void select(String str);

```

### **L.2.7 Class *java.awt.Color***

This Color class supports the RGB colour format. A colour is represented by a 24-bit value of which the red, green and blue components are represented by an 8-bit value (0 to 255). The minimum intensity is 0, and the maximum is 255. The following are defined:

```

// Constants
public final static Color black, blue, cyan, darkGray, gray, green;
public final static Color lightGray, magenta, orange, pink, red;
public final static Color white, yellow;

// Constructors
public Color(float r, float g, float b);
public Color(int rgb);
public Color(int r, int g, int b);

// Methods
public Color brighter();
public Color darker();
public static Color decode (String nm);                                // Java 1.1
public boolean equals(Object obj);
public int getBlue();
public static Color getColor(String nm);
public static Color getColor(String nm, Color v);
public static Color getColor(String nm, int v);
public int getGreen();
public static Color getHSBColor(float h, float s, float b);
public int getRed();
public int getRGB();
public int hashCode();
public static int HSBtoRGB(float hue, float saturation, float brightness);
public static float[] RGBtoHSB(int r, int g, int b, float hsbvals[]);
public String toString();

```

### *L.2.8 Class java.awt.Component*

The Component class is the abstract superclass for many of the Abstract Window Toolkit classes. The following are defined:

```

// Constants
public static final float BOTTOM_ALIGNMENT, CENTER_ALIGNMENT;
public static final float LEFT_ALIGNMENT, RIGHT_ALIGNMENT;
public static final float TOP_ALIGNMENT;

// Methods
public boolean action(Event evt, Object what);                           // Java 1.0
public synchronized void add(PopupMenu popup);                            // Java 1.1
public synchronized void addComponentListener(ComponentListener l);        // Java 1.1
public synchronized void addFocusListener(FocusListener l);                // Java 1.1
public synchronized void addKeyListener(KeyListener l);                  // Java 1.1
public synchronized void addMouseListener(MouseListener l);              // Java 1.1
public synchronized void addMouseMotionListener(MouseMotionListener l); // Java 1.1
public void addNotify();                                                 // Java 1.0
public Rectangle bounds();                                                // Java 1.0
public int checkImage(Image image, ImageObserver observer);           // Java 1.0
public int checkImage(Image image, int width, int height,
                      ImageObserver observer);
public boolean contains(int x, int y);                                     // Java 1.1
public boolean contains(Point p);                                         // Java 1.1
public Image createImage(ImageProducer producer);
public Image createImage(int width, int height);
public void deliverEvent(Event evt);                                       // Java 1.0
public void disable();                                                    // Java 1.0
public final void displayEvent(AWTEvent e);                             // Java 1.1

```

```
public void doLayout();                                     // Java 1.1
public void enable();                                      // Java 1.0
public void enable(boolean cond);                         // Java 1.0
public float getAlignmentX();                            // Java 1.1
public float getAlignmentY();                            // Java 1.1
public Color getBackground();                           // Java 1.1
public Rectangle getBounds();                           // Java 1.1
public ColorModel getColorModel();                     // Java 1.1
public Component getComponentAt(int x, int y);        // Java 1.1
public Component getComponentAt(Point p);              // Java 1.1
public Cursor getCursor();                             // Java 1.1
public Font getFont();                                // Java 1.1
public FontMetrics getFontMetrics(Font font);         // Java 1.1
public Color getForeground();                          // Java 1.1
public Graphics getGraphics();                         // Java 1.1
public Locale getLocale();                            // Java 1.1
public Point getLocation();                           // Java 1.1
public Point getLocationOnScreen();                   // Java 1.1
public Dimension getMaximumSize();                   // Java 1.1
public Dimension getMinimumSize();                   // Java 1.1
public Container getParent();                         // Java 1.1
public ComponentPeer getPeer();                        // Java 1.0
public Dimension getPreferredSize();                  // Java 1.1
public Dimension getSize();                           // Java 1.1
public Toolkit getToolkit();                          // Java 1.1
public final Object getTreeLock();                   // Java 1.1
public boolean gotFocus(Event evt, Object what);     // Java 1.0
public boolean handleEvent(Event evt);                // Java 1.0
public void hide();                                  // Java 1.0
public boolean imageUpdate(Image img, int flags, int x, int y, int w, int h); // Java 1.0
public boolean inside(int x, int y);                 // Java 1.0
public void invalidate();                           // Java 1.0
public boolean isEnabled();                          // Java 1.1
public boolean isFocusTransversable();               // Java 1.1
public boolean isShowing();                           // Java 1.0
public boolean isValid();                            // Java 1.0
public boolean isVisible();                           // Java 1.0
public boolean keyDown(Event evt, int key);          // Java 1.0
public boolean keyUp(Event evt, int key);            // Java 1.0
public void layout();                               // Java 1.0
public void list();                                 // Java 1.0
public void list(PrintStream out);                  // Java 1.1
public void list(PrintStream out, int indent);       // Java 1.1
public void list(PrintStream out);                  // Java 1.1
public Component locate(int x, int y);              // Java 1.0
public Point location();                           // Java 1.0
public boolean lostFocus(Event evt, Object what);    // Java 1.0
public Dimension minimumSize();                    // Java 1.0
public boolean mouseDown(Event evt, int x, int y);   // Java 1.0
public boolean mouseDrag(Event evt, int x, int y);   // Java 1.0
public boolean mouseEnter(Event evt, int x, int y);  // Java 1.0
public boolean mouseExit(Event evt, int x, int y);   // Java 1.0
public boolean mouseMove(Event evt, int x, int y);   // Java 1.0
public boolean mouseUp(Event evt, int x, int y);     // Java 1.0
public void move(int x, int y);                    // Java 1.0
public void nextFocus();                           // Java 1.0
public void paint(Graphics g);                      // Java 1.0
public void paintAll(Graphics g);                  // Java 1.0
protected String paramString();                     // Java 1.0
public boolean postEvent(Event evt);                // Java 1.0
public Dimension preferredSize();                  // Java 1.0
```

```

public boolean prepareImage(Image image, ImageObserver observer);
public prepareImage(Image image, int width, int height, ImageObserver observer);
public void print(Graphics g);
public void printAll(Graphics g);
public synchronized void remove(MenuComponent popup); // Java 1.1
public synchronized void removeComponentListener(ComponentListener l); // Java 1.1
public synchronized void removeFocusListener(FocusListener l); // Java 1.1
public synchronized void removeKeyListener(KeyListener l); // Java 1.1
public synchronized void removeMouseListener(MouseListener l); // Java 1.1
public synchronized void removeMouseMotionListener(MouseMotionListener l); // Java 1.1
public void removeNotify();
public void repaint();
public void repaint(int x, int y, int width, int height);
public void repaint(long tm);
public void repaint(long tm, int x, int y, int width, int height);
public void requestFocus();
public void reshape(int x, int y, int width, int height); // Java 1.0
public void resize(Dimension d); // Java 1.0
public void resize(int width, int height); // Java 1.0
public void setBackground(Color c);
public void setBounds(int x, int y, int width, int height); // Java 1.1
public void setBounds(Rectangle r); // Java 1.1
public synchronized void setCursor(Cursor cursor); // Java 1.1
public void setEnabled(boolean b); // Java 1.1
public void setFont(Font f);
public void setForeground(Color c);
public void setLocale(Locale l); // Java 1.1
public void  setLocation(int x, int y); // Java 1.1
public void setLocation(Point p); // Java 1.1
public void setName(String name); // Java 1.1
public void setSize(int width, int height); // Java 1.1
public void setSize(Dimension d); // Java 1.1
public void setVisible(boolean b); // Java 1.1
public void show(); // Java 1.0
public void show(boolean cond); // Java 1.0
public Dimension size(); // Java 1.0
public String toString();
public void transferFocus(); // Java 1.1
public void update(Graphics g);
public void validate();

```

### **L.2.9 Class java.awt.Container**

The Container class is the abstract superclass representing all components that can hold other components. The following are defined:

```

// Methods
public Component add(Component comp);
public Component add(Component comp, int pos);
public Component add(String name, Component comp);
public void add(Component comp, Object constraints); // Java 1.1
public void add(Component comp, Object constraints, int index); // Java 1.1
public void addContainerListener(ContainerListener l); // Java 1.1
public void addNotify();
public int countComponents(); // Java 1.0
public void deliverEvent(Event evt); // Java 1.0
public void doLayout(); // Java 1.1
public void getAlignmentX(); // Java 1.1
public void getAlignmentY(); // Java 1.1

```

```

public Component getComponent(int n);                                // Java 1.1
public Component getComponentAt(int x, int y);                      // Java 1.1
public Component getComponentAt(Point p);                          // Java 1.1
public int getComponentCount();                                     // Java 1.1
public Component[] getComponents();                                 // Java 1.1
public getInsets();                                              // Java 1.1
public LayoutManager getLayout();                                    // Java 1.1
public Dimension getMaximumSize();                                 // Java 1.1
public Dimension getMinimumSize();                                 // Java 1.1
public Dimension getPreferredSize();                                // Java 1.1
public Insets insets();                                            // Java 1.0
public void invalidate();                                         // Java 1.1
public boolean isAncestorOf(Component c);                         // Java 1.1
public void layout();                                             // Java 1.0
public void list(PrintStream out, int indent);                     // Java 1.1
public void list(PrintWriter out, int indent);                     // Java 1.0
public Component locate(int x, int y);                           // Java 1.0
public Dimension minimumSize();                                   // Java 1.0
public void paintComponents(Graphics g);                         // Java 1.0
protected String  paramString();                                  // Java 1.1
public Dimension preferredSize();                                 // Java 1.1
public void print(Graphics g);                                    // Java 1.1
public void printComponents(Graphics g);                         // Java 1.1
public void remove(int index);                                    // Java 1.1
public void remove(Component comp);                               // Java 1.1
public void removeAll();                                         // Java 1.1
public void removeContainerListener(ContainerListener l);        // Java 1.1
public void removeNotify();                                       // Java 1.1
public void setLayout(LayoutManager mgr);                         // Java 1.1
public void validate();

```

### L.2.10 Class *java.awt.Cursor*

The Cursor class represents a mouse cursor. The following are defined:

```

// Constructors
public Cursor(int type);                                         // Java 1.1

// Constants
public static final int DEFAULT_CURSOR;                          // Java 1.1
public static final int CROSSHAIR_CURSOR, HAND_CURSOR;           // Java 1.1
public static final int MOVE_CURSOR;                             // Java 1.1
public static final int TEXT_CURSOR, WAIT_CURSOR;                // Java 1.1
public static final int N_RESIZE_CURSOR, S_RESIZE_CURSOR;        // Java 1.1
public static final int E_RESIZE_CURSOR, W_RESIZE_CURSOR;        // Java 1.1
public static final int NE_RESIZE_CURSOR, NW_RESIZE_CURSOR;      // Java 1.1
public static final int SE_RESIZE_CURSOR, SW_RESIZE_CURSOR;      // Java 1.1

// Methods
public static Cursor getDefaultCursor();                      // Java 1.1
public static Cursor getPredefinedCursor();                    // Java 1.1

```

### L.2.11 Class *java.awt.Dialog*

The Dialog class supports a dialog window, in which a user can enter data. Dialog windows are invisible until the show method is used. The following are defined:

```

// Constructors
public Dialog(Frame parent);                                     // Java 1.1
public Dialog(Frame parent, boolean modal);

```

```

public Dialog(Frame parent, String title);                                // Java 1.1
public Dialog(Frame parent, String title, boolean modal);

// Methods
public void addNotify();
public String getTitle();
public boolean isModal();
public boolean isResizable();
public void setModal(boolean b);                                         // Java 1.1
protected String paramString();
public void setResizable(boolean resizable);
public void setTitle(String title);
public void show();                                                       // Java 1.1

```

### **L.2.12 Class *java.awt.Dimension***

The Dimension class contains the width and height of a component in an object. The following are defined:

```

// Fields
public int height;
public int width;

// Constructors
public Dimension();
public Dimension(Dimension d);
public Dimension(int width, int height);

// Methods
public boolean equals(Object obj);                                         // Java 1.1
public Dimension getSize();                                                 // Java 1.1
public void setSize(Dimension d);                                         // Java 1.1
public void setSize(int width, int height);                                 // Java 1.1
public String toString();

```

### **L.2.13 Class *java.awt.Event***

The Event class encapsulates user events from the GUI. The following are defined:

```

// Fields
public Object arg;
public int clickCount;
public Event evt;
public int id;
public int key;
public int modifiers;
public Object target;
public long when;
public int x;
public int y;

// possible values for the id field
public final static int ACTION_EVENT, GOT_FOCUS;
public final static int KEY_ACTION, KEY_ACTION_RELEASE;
public final static int KEY_PRESS, KEY_RELEASE;
public final static int LIST_DESELECT, LIST_SELECT;
public final static int LOAD_FILE, LOST_FOCUS;
public final static int MOUSE_DOWN, MOUSE_DRAG;
public final static int MOUSE_ENTER, MOUSE_EXIT;
public final static int MOUSE_MOVE, MOUSE_UP;
public final static int SAVE_FILE, SCROLL_ABSOLUTE;

```

```

public final static int SCROLL_BEGIN, SCROLL_END; // Java 1.1

public final static int SCROLL_LINE_DOWN, SCROLL_LINE_UP;
public final static int SCROLL_PAGE_DOWN, SCROLL_PAGE_UP;
public final static int WINDOW_DEICONIFY, WINDOW_DESTROY;
public final static int WINDOW_EXPOSE, WINDOW_ICONIFY;
public final static int WINDOW_MOVED;

// possible values for the key field when the
// action is KEY_ACTION or KEY_ACTION_RELEASE
public final static int DOWN, END;
public final static int F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12
public final static int HOME, LEFT, PGDN, PGUP, RIGHT, UP;
public final static int INSERT, DELETE; // Java 1.1
public final static int BACK_SPACE, ENTER; // Java 1.1
public final static int TAB, ESCAPE; // Java 1.1
public final static int CAPS_LOCK, NUM_LOCK; // Java 1.1
public final static int SCROLL_LOCK, PAUSE; // Java 1.1
public final static int PRINT_SCREEN; // Java 1.1

// possible masks for the modifiers field
public final static int ALT_MASK;
public final static int CTRL_MASK;
public final static int META_MASK;
public final static int SHIFT_MASK;

// Constructors
public Event(Object target, int id, Object arg);
public Event(Object target, long when, int id,
            int x, int y, int key, int modifiers);
public Event(Object target, long when, int id,
            int x, int y, int key, int modifiers, Object arg);

// Methods
public boolean controlDown();
public boolean metaDown();
protected String paramString();
public boolean shiftDown();
public String toString();
public void translate(int dX, int dY);

```

### L.2.14 Class *java.awt.FileDialog*

The *FileDialog* class displays a dialog window. The following are defined:

```

// Fields
public final static int LOAD, SAVE;

// Constructors
public FileDialog(Frame parent); // Java 1.1
public FileDialog(Frame parent, String title);
public FileDialog(Frame parent, String title, int mode);
// Methods
public void addNotify();
public String getDirectory();
public String getFile();
public FilenameFilter getFilenameFilter();
public int getMode();
protected String paramString();
public void setDirectory(String dir);
public void setFile(String file);
public void setFilenameFilter(FilenameFilter filter);

```

### L.2.15 Class `java.awt.FlowLayout`

The `FlowLayout` class arranges components from left to right. The following are defined:

```
// Fields
public final static int CENTER, LEFT, RIGHT;

// Constructors
public FlowLayout();
public FlowLayout(int align);
public FlowLayout(int align, int hgap, int vgap);

// Methods
public void addLayoutComponent(String name, Component comp);
public int getAlignment(); // Java 1.1
public int getHgap(); // Java 1.1
public int getVgap(); // Java 1.1
public void layoutContainer(Container target);
public Dimension minimumLayoutSize(Container target);
public Dimension preferredLayoutSize(Container target);
public void removeLayoutComponent(Component comp);
public void setAlignment(int align); // Java 1.1
public void setHgap(int hgap); // Java 1.1
public void setVgap(int vgap); // Java 1.1
public String toString();
```

### L.2.16 Class `java.awt.Font`

The `Font` class represents fonts. The following are defined:

```
// Fields
protected String name;
protected int size;
protected int style;

// style has the following bit masks
public final static int BOLD, ITALIC, PLAIN;

// Constructors
public Font(String name, int style, int size);

// Methods
public static Font decode(String str); // Java 1.1
public boolean equals(Object obj);
public String getFamily();
public static Font getFont(String nm);
public static Font getFont(String nm, Font font);
public String getName();
public int getSize();
public int getStyle();
public FontPeer getPeer(); // Java 1.1
public int hashCode();
public boolean isBold();
public boolean isItalic();
public boolean isPlain();
public String toString();
```

### L.2.17 Class `java.awt.FontMetrics`

The `FontMetrics` class provides information about the rendering of a particular font. The following are defined:

```

// Fields
protected Font font;

// Constructors
protected FontMetrics(Font font);

// Methods
public int bytesWidth(byte data[], int off, int len);
public int charsWidth(char data[], int off, int len);
public int charWidth(char ch);
public int charWidth(int ch);
public int getAscent();
public int getDescent();
public Font getFont();
public int getHeight();
public int getLeading();
public int getMaxAdvance();
public int getMaxAscent();
public int getMaxDescent();
public int[] getWidths();
public int stringWidth(String str);
public String toString();

// Java 1.0

```

### L.2.18 Class *java.awt.Frame*

The Frame class contains information on the top-level window. The following are defined:

```

// possible cursor types for the setCursor method
public final static int CROSSHAIR_CURSOR, DEFAULT_CURSOR;
public final static int E_RESIZE_CURSOR, HAND_CURSOR;
public final static int MOVE_CURSOR, N_RESIZE_CURSOR;
public final static int NE_RESIZE_CURSOR, NW_RESIZE_CURSOR;
public final static int S_RESIZE_CURSOR, SE_RESIZE_CURSOR;
public final static int SW_RESIZE_CURSOR, TEXT_CURSOR;
public final static int W_RESIZE_CURSOR, WAIT_CURSOR;
// Constructors
public Frame();
public Frame(String title);
// Methods
public void addNotify();
public void dispose();
public int getCursorType();
public Image getIconImage();
public MenuBar getMenuBar();
public String getTitle();
public boolean isResizable();
protected String  paramString();
public void remove(MenuComponent m);
public void setCursor(int cursorType);
public void setIconImage(Image image);
public void setMenuBar(MenuBar mb);
public void setResizable(boolean resizable);
public void setTitle(String title);

// Java 1.0
// Java 1.0

```

### L.2.19 Class *java.awt.Graphics*

The Graphics class is an abstract class for all graphics contexts. This allows an application to draw onto components or onto off-screen images. The following are defined:

```
// Constructors
```

```

protected Graphics();

// Methods
public abstract void clearRect(int x, int y, int width, int height);
public abstract void clipRect(int x, int y, int width, int height);
public abstract void copyArea(int x, int y, int width, int height,
    int dx, int dy);
public abstract Graphics create();
public Graphics create(int x, int y, int width, int height);
public abstract void dispose();
public void draw3DRect(int x, int y, int width, int height, boolean raised);
public abstract void drawArc(int x, int y, int width, int height,
    int startAngle, int arcAngle);
public void drawBytes(byte data[], int offset, int length, int x, int y);
public void drawChars(char data[], int offset, int length, int x, int y);
public abstract boolean drawImage(Image img, int x, int y, Color bgcolor,
    ImageObserver observer);
public abstract boolean drawImage(Image img, int x, int y,
    ImageObserver observer);
public abstract boolean drawImage(Image img, int x, int y, int width,
    int height, Color bgcolor, ImageObserver observer);
public abstract boolean drawImage(Image img, int x, int y, int width,
    int height, ImageObserver observer);
public abstract boolean drawImage(Image img, int x, int y, int width,
    int height, Color bgcolor, ImageObserver observer); // Java 1.1
public abstract void drawLine(int x1, int y1, int x2, int y2);
public abstract void drawOval(int x, int y, int width, int height);
public abstract void drawPolygon(int xPoints[], int yPoints[], int nPoints);
public void drawPolygon(Polygon p);
public abstract void drawPolyline(int xPoints[], int yPoints[], int nPoints);
                                         // Java 1.1
public void drawRect(int x, int y, int width, int height);
public abstract void drawRoundRect(int x, int y, int width,
    int height, int arcWidth, int arcHeight);
public abstract void drawString(String str, int x, int y);
public void fill3DRect(int x, int y, int width, int height, boolean raised);
public abstract void fillArc(int x, int y, int width, int height,
    int startAngle, int arcAngle);
public abstract void fillOval(int x, int y, int width, int height);
public abstract void fillPolygon(int xPoints[], int yPoints[], int nPoints);
public void fillPolygon(Polygon p);
public abstract void fillRect(int x, int y, int width, int height);
public abstract void fillRoundRect(int x, int y, int width, int height,
    int arcWidth, int arcHeight);
public void finalize();
public abstract Shape getClip(); // Java 1.1
public abstract Rectangle getClipBounds(); // Java 1.1
public abstract Rectangle getClipRect(); // Java 1.0
public abstract Color getColor();
public abstract Font getFont();
public FontMetrics getFontMetrics();
public abstract FontMetrics getFontMetrics(Font f);
public abstract void setClip(int x, int y, int width, int height); // Java 1.1
public abstract void setClip(Shape clip); // Java 1.1
public abstract void setColor(Color c);
public abstract void setFont(Font font);
public abstract void setPaintMode();
public abstract void setXORMode(Color c1);
public String toString();
public abstract void translate(int x, int y);

```

### L.2.20 Class *java.awt.Image*

The Image abstract class is the superclass of all classes that represents graphical images.

```
// Constants
public static final int SCALE_AREA_AVERAGING, SCALE_DEFAULT;
public static final int SCALE_FAST, SCALE_REPLICATE;
public static final int SCALE_SMOOTH;

// Fields
public final static Object UndefinedProperty;

// Constructors
public Image();

// Methods
public abstract void flush();
public abstract Graphics getGraphics();
public abstract int getHeight(ImageObserver observer);
public abstract Object getProperty(String name, ImageObserver observer);
public Image getScaledInstance(int width, int height, int hints); // Java 1.1
public abstract ImageProducer getSource();
public abstract int getWidth(ImageObserver observer)
```

### L.2.21 Class *java.awt.Insets*

The Insets object represents borders of a container and specifies the space that should be left around the edges of a container. The following are defined:

```
// Fields
public int bottom, left;
public int right, top;
// Constructors
public Insets(int top, int left, int bottom, int right);

// Methods
public Object clone();
public boolean equals(Object obj); // Java 1.1
public String toString();
```

### L.2.22 Class *java.awt.Label*

The label class is a component for placing text in a container. The following are defined:

```
// Fields
public final static int CENTER, LEFT, RIGHT;

// Constructors
public Label();
public Label(String label);
public Label(String label, int alignment);

// Methods
public void addNotify();
public int getAlignment();
public String getText();
protected String paramString();
public void setAlignment(int alignment);
public void setText(String label);
```

### L.2.23 Class *java.awt.List*

The List object can be used to produce a scrolling list of text items. It can be set up so that

the user can either pick one or many items. The following are defined:

```
// Constructors
public List();
public List(int rows);
public List(int rows, boolean multipleSelections); // Java 1.1

// Methods
public void add(String item); // Java 1.1
public void addActionListener(ActionListener l); // Java 1.1
public void addItem(String item);
public void addItem(String item, int index);
public synchronized void addItemClickListener(ItemListener l); // Java 1.1
public void addNotify();
public boolean allowsMultipleSelections(); // Java 1.0
public void clear(); // Java 1.0
public int countItems(); // Java 1.0
public void delItem(int position);
public void delItems(int start, int end); // Java 1.0
public void deselect(int index);
public String getItem(int index);
public int getItemCount(); // Java 1.1
public synchronized String[] getItems(); // Java 1.1
public Dimension getMinimumSize(int rows); // Java 1.1
public Dimension getPreferredSize(); // Java 1.1
public Dimension getPreferredSize(int rows); // Java 1.1
public Dimension getPreferredSize(); // Java 1.1
public int getRows(); // Java 1.1
public int getSelectedIndex(); // Java 1.1
public int[] getSelectedIndexes(); // Java 1.1
public String getSelectedItem(); // Java 1.1
public String[] getSelectedItems(); // Java 1.1
public Object[] getSelectedObjects(); // Java 1.1
public int getVisibleIndex(); // Java 1.1
public boolean isIndexSelected(int index); // Java 1.1
public MultipleMode(); // Java 1.1
public boolean isSelected(int index); // Java 1.0
public void makeVisible(int index);
public Dimension minimumSize(); // Java 1.0
public Dimension minimumSize(int rows); // Java 1.0
protected String paramString();
public Dimension preferredSize(); // Java 1.0
public Dimension preferredSize(int rows); // Java 1.0
public synchronized void remove(String item); // Java 1.1
public synchronized void remove(int position); // Java 1.1
public synchronized void removeActionListener(ActionListener l); // Java 1.1
public synchronized void removeAll(); // Java 1.1
public synchronized void removeItemClickListener(ItemListener l); // Java 1.1
public void removeNotify();
public void replaceItem(String newValue, int index);
public void select(int index);
public synchronized void setMultipleMode(boolean b); // Java 1.1
public void setMultipleSelections(boolean v);
```

#### **L.2.24 Class *java.awt.MediaTracker***

The *MediaTracker* class contains a number of media objects, such as images and audio. The following are defined:

```
// Fields
public final static int ABORTED, COMPLETE;
```

```

public final static int ERRORED, LOADING;

// Constructors
public MediaTracker(Component comp);

// Methods
public void addImage(Image image, int id);
public void addImage(Image image, int id, int w, int h);
public boolean checkAll();
public boolean checkAll(boolean load);
public boolean checkID(int id);
public boolean checkID(int id, boolean load);
public Object[] getErrorsAny();
public Object[] getErrorsID(int id);
public boolean isErrorAny();
public boolean isErrorID(int id);
public synchronized removeImage(Image image); // Java 1.1
public synchronized removeImage(Image image, int id); // Java 1.1
public synchronized removeImage(Image image, int id, int width, int height); // Java 1.1
public int statusAll(boolean load);
public int statusID(int id, boolean load);
public void waitForAll();
public boolean waitForAll(long ms);
public void waitForID(int id);
public boolean waitForID(int id, long ms);

```

### L.2.25 Class *java.awt.Menu*

The Menu object contains a pull-down component for a menu bar. The following are defined:

```

// Constructors
public Menu(); // Java 1.1
public Menu(String label);
public Menu(String label, boolean tearOff);

// Methods
public MenuItem add(MenuItem mi);
public void add(String label);
public void addNotify();
public void addSeparator();
public int countItems();
public MenuItem getItem(int index);
public int getItemCount(); // Java 1.1
public synchronized void insert(MenuItem menuitem, int index); // Java 1.1
public void insertSeparator(int index); // Java 1.1
public boolean isTearOff();
public void remove(int index);
public void remove(MenuComponent item);
public synchronized void removeAll(); // Java 1.1
public void removeNotify();

```

### L.2.26 Class *java.awtMenuBar*

TheMenuBar object contains a menu bar which is bound to a frame. The following are defined:

```

// Constructors
public MenuBar();

// Methods

```

```
public Menu add(Menu m);
public void addNotify();
public int countMenus();
public void deleteShortCut(MenuShortCut s); // Java 1.1
public Menu getHelpMenu();
public Menu getMenu(int i);
public int getMenuCount(); // Java 1.1
public MenuItem getShortcutMenuItem(MenuShortcut s);
public void remove(int index); // Java 1.1
public void remove(MenuComponent m);
public void removeNotify();
public void setHelpMenu(Menu m);
public synchronized Enumeration shortcuts(); // Java 1.1
```

### L.2.27 Class *java.awt.MenuComponent*

The `MenuComponent` abstract class is the superclass of all menu-related components. The following are defined:

```
// Constructors
public MenuComponent();

// Methods
public final void dispatchEvent(AWTEvent e); // Java 1.1
public Font getFont();
public String getName(); // Java 1.1
public MenuContainer getParent();
public MenuComponentPeer getPeer();
protected String paramString();
public boolean postEvent(Event evt);
public void removeNotify();
public void setFont(Font f);
public void setName(String name); // Java 1.1
public String toString();
```

### L.2.28 Class *java.awt.MenuItem*

The `MenuItem` class contains all menu items. The following are defined:

```
// Constructors
public MenuItem(); // Java 1.1
public MenuItem(String label);
public MenuItem(String label, MenuShortcut s); // Java 1.1

// Methods
public void addActionListener(ActionListener l); // Java 1.1
public void addNotify();
public void deleteShortcut(); // Java 1.1
public void disable(); // Java 1.0
public void enable(); // Java 1.0
public void enable(boolean cond); // Java 1.0
public String getLabel(); // Java 1.1
public MenuShortcut getShortcut(); // Java 1.1
public boolean isEnabled(); // Java 1.1
public String paramString();
public synchronized void removeActionListener(ActionListener l); // Java 1.1

public void setActionCommand(String command); // Java 1.1
public synchronized void setEnabled(boolean b); // Java 1.1
public void setLabel(String label); // Java 1.1
```

```
public void setShortcut(MenuShortcut s); // Java 1.1
```

### L.2.29 Class *java.awt.MenuShortcut*

The MenuShortcut class has been added with Java 1.1. It represents a keystroke used to select a MenuItem. The following are defined:

```
// Constructors
public MenuShortcut(int key); // Java 1.1
public MenuShortcut(int key, boolean useShiftModifier); // Java 1.1

// Methods
public boolean equals(MenuShortcut s); // Java 1.1
public int getKey(); // Java 1.1
public String toString(); // Java 1.1
public boolean usesShiftModifier(); // Java 1.1
```

### L.2.30 Class *java.awt.Panel*

The Panel class provides space into which an application can attach a component. The following are defined:

```
// Constructors
public Panel();
public Panel(LayoutManger layout); // Java 1.1

// Methods
public void addNotify();
```

### L.2.31 Class *java.awt.Point*

The Point class represents an  $(x, y)$  co-ordinate. The following are defined:

```
// Fields
public int x;
public int y;

// Constructors
public Point(); // Java 1.1
public Point(Point p); // Java 1.1
public Point(int x, int y);

// Methods
public boolean equals(Object obj); // Java 1.1
public Point getLocation();
public int hashCode();
public void move(int x, int y);
public void setLocation(Point p);
public void setLocation(int x, int y); // Java 1.1
public String toString(); // Java 1.1
public void translate(int dx, int dy);
```

### L.2.32 Class *java.awt.Polygon*

The Polygon class consists of an array of  $(x, y)$ , which define the sides of a polygon. The following are defined:

```
// Fields
public int npoints, xpoints[], ypoints[];

// Constructors
```

```

public Polygon();
public Polygon(int xpoints[], int ypoints[], int npoints);

// Methods
public void addPoint(int x, int y);                                // Java 1.1
public boolean contains(Point p);                                     // Java 1.1
public boolean contains(int x, int y);                                     // Java 1.0
public Rectangle getBoundingBox();                                     // Java 1.0
public Rectangle getBounds();                                         // Java 1.1
public boolean inside(int x, int y);                                     // Java 1.0

```

### L.2.33 Class *java.awt.PopupMenu*

The PopupMenu class has been added with Java 1.1. It represents a pop-up menu rather than a pull-down menu. The following are defined:

```

// Constructors
public PopupMenu();                                              // Java 1.1
public PopupMenu(String label);                                    // Java 1.1

// Methods
public synchronized void addNotify();                                // Java 1.1
public void show(Component origin, int x, int y);                  // Java 1.1

```

### L.2.34 Class *java.awt.Rectangle*

The Rectangle class defines an area defined by its top-left (*x*, *y*) co-ordinate, its width and its height. The following are defined:

```

// Fields
public int height, width, x, y;

// Constructors
public Rectangle();
public Rectangle(Rectangle r);                                     // Java 1.1
public Rectangle(Dimension d);
public Rectangle(int width, int height);
public Rectangle(int x, int y, int width, int height);
public Rectangle(Point p);
public Rectangle(Point p, Dimension d);
// Methods
public void add(int newx, int newy);
public void add(Point pt);
public void add(Rectangle r);
public boolean contains(Point p);                                     // Java 1.1
public boolean contains(int x, int y);                                     // Java 1.1
public boolean equals(Object obj);
public Rectangle getBounds();                                         // Java 1.1
public Point getLocation();                                         // Java 1.1
public Dimension getSize();                                         // Java 1.1
public void grow(int h, int v);
public int hashCode();                                            // Java 1.0
public boolean inside(int x, int y);                                     // Java 1.0
public Rectangle intersection(Rectangle r);
public boolean intersects(Rectangle r);
public boolean isEmpty();
public void move(int x, int y);                                         // Java 1.0
public void reshape(int x, int y, int width, int height);           // Java 1.0
public void resize(int width, int height);                           // Java 1.0
public void setBounds(Rectangle r);                                     // Java 1.1
public void setBounds(int x, int y, int width, int height);          // Java 1.1

```

```

public void setLocation(Point p); // Java 1.1
public void setLocation(int x, int y); // Java 1.1
public void setSize(Dimension d); // Java 1.1
public void setSize(int x, int y); // Java 1.1
public String toString(); // Java 1.1
public void translate(int dx, int dy); // Java 1.1
public Rectangle union(Rectangle r);

```

### L.2.35 Class *java.awt.Scrollbar*

The Scrollbar class is a convenient means of allowing a user to select from a range of values. The following are defined:

```

// Fields
public final static int HORIZONTAL, VERTICAL;

// Constructors
public Scrollbar();
public Scrollbar(int orientation);
public Scrollbar(int orientation, int value, int visible, int minimum,
    int maximum);

// Methods
public synchronized void addAdjustmenuListener(AdjustmentListener l); // Java 1.1
public void addNotify(); // Java 1.1
public int getBlockIncrement(); // Java 1.1
public int getLineIncrement(); // Java 1.0
public int getMaximum(); // Java 1.0
public int getMinimum(); // Java 1.0
public int getOrientation(); // Java 1.0
public int getPageIncrement(); // Java 1.0
public int getUnitIncrement(); // Java 1.1
public int getValue(); // Java 1.0
public int getVisible(); // Java 1.0
protected String paramString();
public void setLineIncrement(int l); // Java 1.0
public synchronized void setMaximum(int max); // Java 1.1
public synchronized void setMinimum(int min); // Java 1.1
public synchronized void setOrientation(int orien); // Java 1.1
public void setPageIncrement(int l); // Java 1.0
public void setValue(int value);
public void setValues(int value, int visible, int minimum, int maximum);
public void setVisibleAmount(int am); // Java 1.1

```

### L.2.36 Class *java.awt.TextArea*

The TextArea class allows for a multi-line area for displaying text. The following are defined:

```

// Constructors
public TextArea();
public TextArea(int rows, int cols);
public TextArea(String text);
public TextArea(String text, int rows, int cols);
public TextArea(String text, int rows, int cols, int scrollbars); // Java 1.1

// Constants
public static final int SCROLLBARS_BOTH; // Java 1.1
public static final int SCROLLBARS_HORIZONTAL_ONLY; // Java 1.1

```

```

public static final int SCROLLBARS_NONE;                                // Java 1.1
public static final int SCROLLBARS_VERTICAL_ONLY;                         // Java 1.1

// Methods
public void addNotify();                                                 
public synchronized void append(String str);                             // Java 1.1
public void appendText(String str);                                       // Java 1.0
public int getColumns();                                                 
public Dimension getMinimumSize(int rows, int cols);                     // Java 1.1
public Dimension getPreferredSize();                                      // Java 1.1
public Dimension getPreferredSize();                                     // Java 1.1
public Dimension getPreferredSize();                                     // Java 1.1
public int getRows();                                                 
public int getScrollbarVisibility();                                     // Java 1.1
public void insertText(String str, int pos);                            // Java 1.1
public Dimension minimumSize();                                         // Java 1.0
public Dimension minimumSize(int rows, int cols);                      // Java 1.0
protected String paramString();                                          
public Dimension preferredSize();                                         // Java 1.0
public Dimension preferredSize(int rows, int cols);                     // Java 1.0
public void replaceText(String str, int start, int end);                // Java 1.0
public void setColumns(int cols);                                         // Java 1.1
public void setRows(int rows);                                         // Java 1.1

```

### L.2.37 Class *java.awt.TextComponent*

The *TextComponent* class is the superclass of any component that allows the editing of some text. The following are defined:

```

// Methods
public void addTextListener(TextListener l);                           // Java 1.1
public int getCaretPosition();                                         // Java 1.1
public String getSelectedText();                                         
public int getSelectionEnd();                                          
public int getSelectionStart();                                         
public String getText();                                              
public boolean isEditable();                                           
protected String paramString();                                          
public void removeNotify();                                           
public void removeTextListener(TextListener l);                         // Java 1.1
public void select(int selStart, int selEnd);                                 
public void selectAll();                                                 
public void setCaretPosition(int position);                            // Java 1.1
public void setEditable(boolean t);                                   
public synchronized setSelectionEnd(int selectionEnd);                 // Java 1.1
public synchronized setSelectionStart(int selectionStart);            // Java 1.1
public void setText(String t);                                          

```

### L.2.38 Class *java.awt.TextField*

The *TextField* class is a component that presents the user with a single editable line of text. The following are defined:

```

// Constructors
public TextField();                                                 
public TextField(int cols);                                          
public TextField(String text);                                         
public TextField(String text, int cols);                              

// Methods
public synchronized void addActionListener(ActionListener l);        // Java 1.1

```

```

public void addNotify();
public boolean echoCharIsSet();
public int getColumns();
public char getEchoChar();
public Dimension getMinimumSize(int cols);                                // Java 1.1
public Dimension getPreferredSize();                                         // Java 1.1
public Dimension getPreferredSize(int cols);                                // Java 1.1
public Dimension getPreferredSize();                                         // Java 1.1
public Dimension minimumSize();                                             // Java 1.0

public Dimension minimumSize(int cols);                                     // Java 1.0
protected String paramString();
public Dimension preferredSize();                                           // Java 1.0
public Dimension preferredSize(int cols);                                    // Java 1.0
public void setColumns(int cols);                                          // Java 1.1
public void setEchoChar(char c);                                           // Java 1.1
public void setEchoCharacter(char c);                                       // Java 1.0

```

### L.2.39 Class *java.awt.Toolkit*

The Toolkit class is the abstract superclass of all actual implementations of the Abstract Window Toolkit. The following are defined:

```

// Constructors
public Toolkit();

// Methods
public abstract int beep();                                                 // Java 1.1
public abstract int checkImage(Image image, int width,
                               int height, ImageObserver observer);
public abstract Image createImage(ImageProducer producer);
public Image createImage(byte[] imagedata);                                 // Java 1.1
public Image createImage(byte[] imagedata, int imageoffset,
                        int imagelength);                                         // Java 1.1
public abstract ColorModel getColorModel();
public static Toolkit getDefaultToolkit();
public abstract String[] getFontList();
public abstract FontMetrics getFontMetrics(Font font);
public abstract Image getImage(String filename);
public abstract Image getImage(URL url);
public int getMenuShortcutKeyMask();                                         // Java 1.1
public abstract PrintJob getPrintJob(Frame frame, String jobtitle,
                                      Properties props);                                // Java 1.1
public abstract int getScreenResolution();
public abstract Dimension getScreenSize();
public abstract Clipboard getSystemClipboard();                            // Java 1.1
public abstract EventQueue getSystemEventQueue();                         // Java 1.1
public abstract boolean prepareImage(Image image, int width,
                                      int height, ImageObserver observer);
public abstract void sync();

```

### L.2.40 Class *java.awt.Window*

The Window class is the top-level window; it has no borders and no menu bar. The following are defined:

```

// Constructors
public Window(Frame parent);
// Methods
public void addNotify();
public synchronized void addWindowListener(WindowListener l);           // Java 1.1

```

```

public void dispose();
public Component getFocusOwner();                                // Java 1.1
public Locale getLocale();                                     // Java 1.1
public Toolkit getToolkit();
public final String getWarningString();
public boolean isShowing();                                    // Java 1.1
public void pack();
public postEvent(Event e);                                    // Java 1.1
public synchronized void removeWindowListener(WindowListener l);
public void show();
public void toBack();
public void toFront();

```

## **L.3 Package java.awt.datatransfer**

---

### **L.3.1 Class *java.awt.datatransfer.Clipboard***

The Clipboard class has been added with Java 1.1. It represents a clipboard onto which data can be transferred using cut-and-paste techniques. The following are defined:

```

// Constructors
public Clipboard(String name);                                // Java 1.1

// Methods
public synchronized Transferable getContents(Object requestor); // Java 1.1
public String getName();                                       // Java 1.1
public synchronized void setContents(Transferable contents,    // Java 1.1
                                    Clipboard owner);           // Java 1.1

```

## **L.4 Package java.awt.event**

---

### **L.4.1 Class *java.awt.event.ActionEvent***

The ActionEvent class has been added with Java 1.1. It occurs when a event happens for a Button, List, MenuItem or TextField. The following are defined:

```

// Constructors
public ActionEvent(Object src, String cmd);                  // Java 1.1

// Methods
public String getActionCommand();                            // Java 1.1
public int getModifiers();                                  // Java 1.1
public int paramString();                                   // Java 1.1

```

### **L.4.2 Interface *java.awt.event.ActionListener***

The ActionListener interface has been added with Java 1.1. It defines the method which is called by an ActionEvent. The following is defined:

```
public void actionPerformed(ActionEvent e);                  // Java 1.1
```

#### **L.4.3 Class *java.awt.event.AdjustmentEvent***

The AdjustmentEvent class has been added with Java 1.1. It occurs when a event happens for a Scrollbar. The following are defined:

```
// Constructors
public AdjustmentEvent(Object src, int id, int type, int value); // Java 1.1

// Methods
public Adjustable getAdjustable(); // Java 1.1
public int getAdjustmentType(); // Java 1.1
public int getValue(); // Java 1.1
public String  paramString(); // Java 1.1
```

#### **L.4.4 Class *java.awt.event.AdjustmentListener***

The AdjustmentListener interface has been added with Java 1.1. It defines the method which is called by an AdjustmentEvent. The following is defined:

```
public void adjustmentValueChanged(AdjustmentEvent e); // Java 1.1
```

#### **L.4.5 Class *java.awt.event.ComponentEvent***

The ComponentEvent class has been added with Java 1.1. It occurs when a event happens for a Component. The following are defined:

```
// Constructors
public ComponentEvent(Object src, int id, int type, int value);

// Methods
public Component getComponent(); // Java 1.1
public String  paramString(); // Java 1.1
```

#### **L.4.6 Class *java.awt.event.ComponentListener***

The ComponentListener interface has been added with Java 1.1. It defines the method which is called by a ComponentEvent. The following are defined:

```
public void componentHidden(ComponentEvent e); // Java 1.1
public void componentMoved(ComponentEvent e); // Java 1.1
public void componentResized(ComponentEvent e); // Java 1.1
public void componentShown(ComponentEvent e); // Java 1.1
```

#### **L.4.7 Class *java.awt.event.ContainerEvent***

The ComponentEvent class has been added with Java 1.1. It occurs when a event happens for a Container. The following are defined:

```
// Constructors
public ContainerEvent(Component src, int id, Compoent child);

// Methods
public Component  getChild(); // Java 1.1
public Component  getContainer(); // Java 1.1
public String  paramString(); // Java 1.1
```

#### **L.4.8 Class *java.awt.event.ContainerListener***

The ContainerListener interface has been added with Java 1.1. It defines the method

which is called by a ContainerEvent. The following are defined:

```
public void componentAdded(ComponentEvent e);                                // Java 1.1
public void componentRemoved(ComponentEvent e);                            // Java 1.1
```

#### **L.4.9 Class *java.awt.event.ItemEvent***

The ItemEvent class has been added with Java 1.1. It occurs when a event happens for a Container. The following are defined:

```
// Constructors
public ItemEvent(ItemSelectable src, int id, Object item, int stateChanged);      // Java 1.1

// Methods
public Object getItem();                                              // Java 1.1
public ItemSelectable getItemSelectable();    // Java 1.1
public int getStateChange();                                         // Java 1.1
public String paramString();                                           // Java 1.1
```

#### **L.4.10 Class *java.awt.event.ItemListener***

The ItemListener interface has been added with Java 1.1. It defines the method which is called by an ItemEvent. The following is defined:

```
public void itemStateChanged(ItemEvent e);                                // Java 1.1
```

#### **L.4.11 Class *java.awt.event.KeyEvent***

The KeyEvent class has been added with Java 1.1. It occurs when a event happens for a keypress. The following are defined:

```
// Constructors
public KeyEvent(Component src, int id, long when, int modifiers,
                 int keyCode, char keyChar);                                // Java 1.1

// Constants
public static final int KEY_LAST, KEY_PRESSED, KEY_RELEASED, KEY_TYPED;
                           // Undefined Key and Character (Java 1.1)
public static final int VK_UNDEFINED, CHAR_UNDEFINED;                    // Alphanumeric keys (Java 1.1)
public static final int VK_A, VK_B, VK_C, VK_D, VK_E, VK_F, VK_G, VK_H;
public static final int VK_I, VK_J, VK_K, VK_L, VK_M, VK_N, VK_O, VK_P;
public static final int VK_Q, VK_R, VK_S, VK_T, VK_U, VK_V, VK_W, VK_X;
public static final int VK_Y, VK_Z;
public static final int VK_SPACE;
public static final int VK_0, VK_1, VK_2, VK_3, VK_4, VK_5, VK_6, VK_7;
public static final int VK_8, VK_9;
public static final int VK_NUMPAD0, VK_NUMPAD1, VK_NUMPAD2, VK_NUMPAD3;
public static final int VK_NUMPAD4, VK_NUMPAD5, VK_NUMPAD6, VK_NUMPAD7;
public static final int VK_NUMPAD8, VK_NUMPAD9;                          // Control keys (Java 1.1)
public static final int VK_BACK_SPACE, VK_ENTER, VK_ESCAPE, VK_TAB;
                           // Modifier keys (Java 1.1)
public static final int VK_ALT, VK_CAPS_LOCK, VK_CONTROL, VK_META, VK_SHIFT;
                           // Function keys (Java 1.1)
public static final int VK_F0, VK_F1, VK_F2, VK_F3, VK_F4, VK_F5, VK_F6;
public static final int VK_F7, VK_F8, VK_F9;
public static final int VK_PRINTSCREEN, VK_SCROLL_LOCK, VK_PAUSE;
public static final int VK_PAGE_DOWN, VK_PAGE_UP;
```

```

public static final int VK_DOWN, VK_UP, VK_RIGHT, VK_LEFT;
public static final int VK_END, VK_HOME, VK_ACCEPT, VK_NUM_LOCK, VK_CANCEL;
public static final int VK_CLEAR, VK_CONVERT, VK_FINAL, VK_HELP;
public static final int VK_KANA, VK_KANJI, VK_MODECHANGE, VK_NONCONVERT;
                                         // Punctuation keys (Java 1.1)
public static final int VK_ADD, VK_BACK_QUOTE, VK_BACK_SLASH;
public static final int VK_CLOSE_BRACKET, VK_COMMA, VK_DECIMAL;
public static final int VK_DIVIDE, VK_EQUALS, VK_MULTIPLY;
public static final int VK_OPEN_BRACKET, VK_PERIOD, VK_QUOTE;
public static final int VK_SEMICOLON, VK_SEPARATOR, VK_SLASH;
public static final int VK_SUBTRACT;

// Methods
public void getKeyChar();                                     // Java 1.1

public int getKeyCode();                                     // Java 1.1
public boolean isActionKey();                                // Java 1.1
public String paramString();                                 // Java 1.1
public void setKeyChar(char keyChar);                         // Java 1.1
public void setKeyCode(int keyCode);                         // Java 1.1
public void setModifiers(int modifiers);                     // Java 1.1

```

#### L.4.12 Class *java.awt.event.KeyListener*

The KeyListener interface has been added with Java 1.1. It defines the method which is called by a KeyEvent. The following is defined:

```

public void keyPressed(KeyEvent e);                           // Java 1.1
public void keyReleased(KeyEvent e);                         // Java 1.1
public void keyTyped(KeyEvent e);                           // Java 1.1

```

#### L.4.13 Class *java.awt.event.MouseEvent*

The MouseEvent class has been added with Java 1.1. It occurs when a event happens for a MouseEvent. The following are defined:

```

// Constructors
public MouseEvent(Component src, int id, long when, int modifiers, int x,
                   int y, int clickCount, boolean popupTrigger);           // Java 1.1

// Constants
public static final int MOUSE_CLICKED, MOUSE_DRAGGED;
public static final int MOUSE_ENTERED, MOUSE_EXITED;
public static final int MOUSE_FIRST, MOUSE_LAST;
public static final int MOUSE_MOVED, MOUSE_PRESSED;
public static final int MOUSE_RELEASED;

// Methods
public int getClickCount();                                  // Java 1.1

public Point getPoint();                                    // Java 1.1
public int getX();                                         // Java 1.1
public int getY();                                         // Java 1.1
public boolean isPopupTrigger();                            // Java 1.1
public String paramString();                               // Java 1.1
public synchronized void translatePoint(int x, int y);    // Java 1.1

```

#### L.4.14 Class *java.awt.event.MouseListener*

The MouseListener interface has been added with Java 1.1. It defines the method which is called by a mouse click event. The following are defined:

```

public void mouseClicked(MouseEvent e);                                // Java 1.1
public void mouseEntered(MouseEvent e);                                // Java 1.1
public void mouseExited(MouseEvent e);                                // Java 1.1
public void mousePressed(MouseEvent e);                                // Java 1.1
public void mouseReleased(MouseEvent e);                                // Java 1.1

```

#### **L.4.15 Class *java.awt.event.MouseMouseListener***

The MouseMouseListener interface has been added with Java 1.1. It defines the method which is called by a mouse drag or move event. The following are defined:

```

public void mouseDragged(MouseEvent e);                                // Java 1.1
public void mouseMoved(MouseEvent e);                                // Java 1.1

```

#### **L.4.16 Class *java.awt.event.TextEvent***

The TextEvent class has been added with Java 1.1. It occurs when a event happens for an event within TextField, TextArea or other TextComponent. The following are defined:

```

// Constructors
public TextEvent(Object src, int id);                                // Java 1.1

// Constants
public static final int TEXT_FIRST, TEXT_LAST;
public static final int TEXT_VALUE_CHANGED;

// Methods
public String paramString();                                         // Java 1.1

```

#### **L.4.17 Class *java.awt.event.TextListener***

The TextListener interface has been added with Java 1.1. It defines the method which is called by a TextEvent. The following is defined:

```
public void textValueChanged(TextEvent e);                                // Java 1.1
```

#### **L.4.18 Class *java.awt.event.WindowEvent***

The WindowEvent class has been added with Java 1.1. It occurs when an event happens within a Window object. The following are defined:

```

// Constructors
public WindowEvent(Window src, int id);                                // Java 1.1

// Constants
public static final int WINDOW_ACTIVATED, WINDOW_CLOSED;
public static final int WINDOW_CLOSING, WINDOW_DEACTIVATED;
public static final int WINDOW_DEICONIFIED, WINDOW_FIRST;
public static final int WINDOW_ICONIFIED, WINDOW_LAST;
public static final int WINDOW_OPENED;

// Methods
public Window getWindow();                                              // Java 1.1
public String paramString();                                             // Java 1.1

```

#### **L.4.19 Class *java.awt.event.WindowListener***

The WindowListener interface has been added with Java 1.1. It defines the method which is called by an WindowEvent. The following are defined:

```

public void windowActivated(WindowEvent e);                                // Java 1.1
public void windowClosed(WindowEvent e);                                 // Java 1.1
public void windowDeactivated(WindowEvent e);                            // Java 1.1
public void windowDeiconified(WindowEvent e);                           // Java 1.1
public void windowIconified(WindowEvent e);                             // Java 1.1
public void windowOpened(WindowEvent e);                                // Java 1.1

```

## L.5 Package `java.awt.image`

---

This package has been added with Java 1.1 and supports image processing classes.

## L.6 Package `java.io`

---

### L.6.1 Class `java.io.BufferedOutputStream`

The `BufferedOutputStream` implements a buffered output stream. These streams allow the program to write to an input device without having to worry about the interfacing method. The following are defined:

```

// Fields
protected byte buf[];
protected int count;

// Constructors
public BufferedOutputStream(OutputStream out);
public BufferedOutputStream(OutputStream out, int size);

// Methods
public void flush();
public void write(byte b[], int off, int len);
public void write(int b);

```

### L.6.2 Class `java.io.BufferedReader`

The `BufferedReader` class has been added with Java 1.1. It represents a buffered character input stream. The following are defined:

```

// Constructors
public BufferedReader(Reader in, int sz);                                // Java 1.1
public BufferedReader(Reader in);                                         // Java 1.1

// Methods
public void close() throws IOException;                                    // Java 1.1
public void mark(int readAheadLimit) throws IOException;                  // Java 1.1
public boolean markSupported() throws IOException;                        // Java 1.1
public int read() throws IOException;                                     // Java 1.1
public int read(char [] cbuf, int off, int len) throws IOException; // Java 1.1
public String readLine() throws IOException;                            // Java 1.1
public boolean ready() throws IOException;                               // Java 1.1
public void reset() throws IOException;                                 // Java 1.1

```

```
public long skip(long n) throws IOException; // Java 1.1
```

### **L.6.3 Class *java.io.BufferedWriter***

The BufferedWriter class has been added with Java 1.1. It represents a buffered character output stream. The following are defined:

```
// Constructors
public BufferedWriter(Writer out, int sz); // Java 1.1
public BufferedWriter(Writer in); // Java 1.1

// Methods
public void close() throws IOException; // Java 1.1
public void flush() throws IOException; // Java 1.1
public void newLine() throws IOException; // Java 1.1
public void write(int c) throws IOException; // Java 1.1
public void write(char [] cbuf, int off, int len) throws IOException; // Java 1.1
```

### **L.6.4 Class *java.io.ByteArrayInputStream***

The ByteArrayInputStream class supports input from a byte array. The following are defined:

```
// Fields
protected byte buf[];
protected int count;
protected int mark; // Java 1.1
protected int pos;

// Constructors
public ByteArrayInputStream(byte buf[]);
public ByteArrayInputStream(byte buf[], int offset, int length);

// Methods
public int available();
public void mark(int markpos); // Java 1.1
public boolean markSupported(); // Java 1.1
public int read();
public int read(byte b[], int off, int len);
public void reset();
public long skip(long n);
```

### **L.6.5 Class *java.io.ByteArrayOutputStream***

The ByteArrayOutputStream class allows supports output to a byte array. The following are defined:

```
// Fields
protected byte buf[];
protected int count;

// Constructors
public ByteArrayOutputStream();
public ByteArrayOutputStream(int size);

// Methods
public void reset();
public int size();
public byte[] toByteArray();
```

```

public String toString();
public String toString(int hibyte);
public String toString(String enc); // Java 1.0
public void write(byte b[], int off, int len); // Java 1.1
public void write(int b);
public void writeTo(OutputStream out);

```

### L.6.6 Interface *java.io.DataInput*

The DataInput interface gives support for streams to read in a machine-independent way. The following are defined:

```

// Methods
public abstract boolean readBoolean();
public abstract byte readByte();
public abstract char readChar();
public abstract double readDouble();
public abstract float readFloat();
public abstract void readFully(byte b[]);
public abstract void readFully(byte b[], int off, int len);
public abstract int readInt();
public abstract String readLine();
public abstract long readLong();
public abstract short readShort();
public abstract int readUnsignedByte();
public abstract int readUnsignedShort();
public abstract String readUTF();
public abstract int skipBytes(int n);

```

### L.6.7 Class *java.io.DataInputStream*

The DataInputStream class allows an application to read data in a machine-independent way. It uses standard Unicode strings which conforms to the UTF-81 specification. The following are defined:

```

// Constructors
public DataInputStream(InputStream in);

// Methods
public final int read(byte b[]);
public final int read(byte b[], int off, int len);
public final boolean readBoolean();
public final byte readByte();
public final char readChar();
public final double readDouble();
public final float readFloat();
public final void readFully(byte b[]);
public final void readFully(byte b[], int off, int len);
public final int readInt();
public final String readLine(); // Java 1.0
public final long readLong();
public final short readShort();
public final int readUnsignedByte();
public final int readUnsignedShort();
public final String readUTF();
public final static String readUTF(DataInput in);
public final int skipBytes(int n);

```

### L.6.8 Interface *java.io.DataOutput*

The DataOutput interface gives support for streams to write in a machine-independent way.

The following are defined:

```
// Methods
public abstract void write(byte b[]);
public abstract void write(byte b[], int off, int len);
public abstract void write(int b);
public abstract void writeBoolean(boolean v);
public abstract void writeByte(int v);
public abstract void writeBytes(String s);
public abstract void writeChar(int v);
public abstract void writeChars(String s);
public abstract void writeDouble(double v);
public abstract void writeFloat(float v);
public abstract void writeInt(int v);
public abstract void writeLong(long v);
public abstract void writeShort(int v);
public abstract void writeUTF(String str);
```

#### **L.6.9 Class *java.io.DataOutputStream***

The *DataOutputStream* class allows an application to write data in a machine-independent way. It uses standard Unicode strings which conforms to the UTF-81 specification. The following are defined:

```
// Fields
protected int written;
// Constructors
public DataOutputStream(OutputStream out);

// Methods
public void flush();
public final int size();
public void write(byte b[], int off, int len);
public void write(int b);
public final void writeBoolean(boolean v);
public final void writeByte(int v);
public final void writeBytes(String s);
public final void writeChar(int v);
public final void writeChars(String s);
public final void writeDouble(double v);
public final void writeFloat(float v);
public final void writeInt(int v);
public final void writeLong(long v);
public final void writeShort(int v);
public final void writeUTF(String str);
```

#### **L.6.10 Class *java.io.EOFException***

Exception that identifies that the end-of-file has been reached unexpectedly during input. The following are defined:

```
// Constructors
public EOFException();
public EOFException(String s);
```

#### **L.6.11 Class *java.io.File***

The *File* class implements the file manipulation operations in an operating system independent way. The following are defined:

```

// Fields
public final static String pathSeparator;
public final static char pathSeparatorChar;
public final static String separator;
public final static char separatorChar;

// Constructors
public File(File dir, String name);
public File(String path);
public File(String path, String name);

// Methods
public boolean canRead();
public boolean canWrite();
public boolean delete();
public boolean equals(Object obj);
public boolean exists();
public String getAbsolutePath();
public String getCanonicalPath(); // Java 1.1
public String getName();
public String getParent();
public String getPath();
public int hashCode();
public boolean isAbsolute();
public boolean isDirectory();
public boolean isFile();
public long lastModified();
public long length();
public String[] list();
public String[] list(FilenameFilter filter);
public boolean mkdir();
public boolean mkdirs();
public boolean renameTo(File dest);
public String toString();

```

### **L.6.12 Class *java.io.FileDescriptor***

The *FileDescriptor* class provides a way to cope with opening files or sockets. The following are defined:

```

// Fields
public final static FileDescriptor err, in, out;

// Constructors
public FileDescriptor();

// Methods
public void sync(); // Java 1.1
public boolean valid();

```

### **L.6.13 Class *java.io.FileInputStream***

The *FileInputStream* class provides supports for an input file. The following are defined:

```

// Constructors
public FileInputStream(File file);
public FileInputStream(FileDescriptor fdObj);
public FileInputStream(String name);

// Methods

```

```

public int available();
public void close();
protected void finalize();
public final FileDescriptor getFD();
public int read();
public int read(byte b[]);
public int read(byte b[], int off, int len);
public long skip(long n);

```

#### **L.6.14 Interface *java.io.FilenameFilter***

The *FilenameFilter* interface is used to filter filenames. The following is defined:

```

// Methods
public abstract boolean accept(File dir, String name);

```

#### **L.6.15 Class *java.io.FileNotFoundException***

Exception that identifies that a file could not be found. The following are defined:

```

// Constructors
public FileNotFoundException();
public FileNotFoundException(String s);

```

#### **L.6.16 Class *java.io.FileOutputStream***

The *FileOutputStream* class provides supports for an output file. The following are defined:

```

// Constructors
public FileOutputStream(File file);
public FileOutputStream(String name, boolean append); // Java 1.1
public FileOutputStream(FileDescriptor fdObj);
public FileOutputStream(String name);

// Methods
public void close();
protected void finalize();
public final FileDescriptor getFD();
public void write(byte b[]);
public void write(byte b[], int off, int len);
public void write(int b);

```

#### **L.6.17 Class *java.io.FilterInputStream***

The *FilterInputStream* class is the superclass of all classes that filter input streams. The following are defined:

```

// Fields
protected InputStream in;

// Constructors
protected FilterInputStream(InputStream in);

// Methods
public int available();
public void close();
public void mark(int readlimit);
public boolean markSupported();
public int read();
public int read(byte b[]);

```

```
public int read(byte b[], int off, int len);
public void reset();
public long skip(long n);
```

#### **L.6.18 Class *java.io.FilterOutputStream***

The `FilterOutputStream` class is the superclass of all classes that filter output streams. The following are defined:

```
// Fields
protected OutputStream out;

// Constructors
public FilterOutputStream(OutputStream out);

// Methods
public void close();
public void flush();
public void write(byte b[]);
public void write(byte b[], int off, int len);
public void write(int b);
```

#### **L.6.19 Class *java.io.InputStream***

The `InputStream` class is the superclass of all classes representing an input stream of bytes. The following are defined:

```
// Constructors
public InputStream();

// Methods
public int available();
public void close();
public void mark(int readlimit);
public boolean markSupported();
public abstract int read();
public int read(byte b[]);
public int read(byte b[], int off, int len);
public void reset();
public long skip(long n);
```

#### **L.6.20 Class *java.io.InterruptedIOException***

Exception that identifies that an I/O operation has been interrupted. The following are defined:

```
// Fields
public int bytesTransferred;

// Constructors
public InterruptedIOException();
public InterruptedIOException(String s);
```

#### **L.6.21 Class *java.io.IOException***

Exception that identifies that an I/O exception has occurred. The following are defined:

```
// Constructors
public IOException();
public IOException(String s);
```

### **L.6.22 Class *java.io.LineNumberInputStream***

The LineNumberInputStream class provides support for the current line number in an input stream. Each line is delimited by either a carriage return character ('\r'), new-line character ('\n') or both together. The following are defined:

```
// Constructors
public LineNumberInputStream(InputStream in);

// Methods
public int available();
public int getLineNumber();
public void mark(int readlimit);
public int read();
public int read(byte b[], int off, int len);
public void reset();
public void setLineNumber(int lineNumber);
public long skip(long n);
```

### **L.6.23 Class *java.io.OutputStream***

The InputStream class is the superclass of all classes representing an output stream of bytes. The following are defined:

```
// Constructors
public OutputStream();

// Methods
public void close();
public void flush();
public void write(byte b[]);
public void write(byte b[], int off, int len);
public abstract void write(int b);
```

### **L.6.24 Class *java.io.PipedInputStream***

The PipedInputStream class provides support for pipelined input communications. The following are defined:

```
// Constructors
public PipedInputStream();
public PipedInputStream(PipedOutputStream src);

// Methods
public void close();
public void connect(PipedOutputStream src);
public int read();
public int read(byte b[], int off, int len);
```

### **L.6.25 Class *java.io.PipedOutputStream***

The PipedOutputStream class provides support for pipelined output communications. The following are defined:

```
// Constructors
public PipedOutputStream();
public PipedOutputStream(PipedInputStream snk);

// Methods
public void close();
```

```
public void connect(PipedInputStream snk);
public void write(byte b[], int off, int len);
public void write(int b);
```

### L.6.26 Class *java.io.PrintStream*

The PrintStream class provides support for output print streams. The following are defined:

```
// Constructors
public PrintStream(OutputStream out); // Java 1.0
public PrintStream(OutputStream out, boolean autoflush); // Java 1.0

// Methods
public boolean checkError();
public void close();
public void flush();
public void print(boolean b);
public void print(char c);
public void print(char s[]);
public void print(double d);
public void print(float f);
public void print(int i);
public void print(long l);
public void print(Object obj);
public void print(String s);
public void println();
public void println(boolean b);
public void println(char c);
public void println(char s[]);
public void println(double d);
public void println(float f);
public void println(int i);
public void println(long l);
public void println(Object obj);
public void println(String s);
public void write(byte b[], int off, int len);
public void write(int b);
```

### L.6.27 Class *java.io.PushbackInputStream*

The PushbackInputStream class provides support to put bytes back into an input stream. The following are defined:

```
// Fields
protected int pushBack;

// Constructors
public PushbackInputStream(InputStream in);

// Methods
public int available();
public boolean markSupported();
public int read();
public int read(byte bytes[], int offset, int length);
public void unread(int ch);
```

### L.6.28 Class *java.io.RandomAccessFile*

The RandomAccessFile class support reading and writing from a random access file. The following are defined:

```

// Constructors
public RandomAccessFile(File file, String mode);
public RandomAccessFile(String name, String mode);

// Methods
public void close();
public final FileDescriptor getFD();
public long getFilePointer();
public long length();
public int read();
public int read(byte b[]);
public int read(byte b[], int off, int len);
public final boolean readBoolean();
public final byte readByte();
public final char readChar();
public final double readDouble();
public final float readFloat();
public final void readFully(byte b[]);
public final void readFully(byte b[], int off, int len);
public final int readInt();
public final String readLine();
public final long readLong();
public final short readShort();
public final int readUnsignedByte();
public final int readUnsignedShort();
public final String readUTF();
public void seek(long pos);
public int skipBytes(int n);
public void write(byte b[]);
public void write(byte b[], int off, int len);
public void write(int b);
public final void writeBoolean(boolean v);
public final void writeByte(int v);
public final void writeBytes(String s);
public final void writeChar(int v);
public final void writeChars(String s);
public final void writeDouble(double v);
public final void writeFloat(float v);
public final void writeInt(int v);
public final void writeLong(long v);
public final void writeShort(int v);
public final void writeUTF(String str);

```

#### **L.6.29 Class *java.io.SequenceInputStream***

The SequenceInputStream supports the combination of several input streams into a single input stream. The following are defined:

```

// Constructors
public SequenceInputStream(Enumeration e);
public SequenceInputStream(InputStream s1, InputStream s2);

// Methods
public void avialable(); // Java 1.1
public void close();
public int read();
public int read(byte buf[], int pos, int len);

```

#### **L.6.30 Class *java.io.StreamTokenizer***

The StreamTokenizer class splits an input stream into tokens. These tokens can be defined

by number, quotes strings or comment styles. The following are defined:

```
// Fields
public double nval;
public String sval;
public int ttype;

// possible values for the ttype field
public final static int TT_EOF, TT_EOL, TT_NUMBER, TT_WORD;

// Constructors
public StreamTokenizer(InputStream I);

// Methods
public void commentChar(int ch);
public void eolIsSignificant(boolean flag);
public int lineno();
public void lowerCaseMode(boolean fl);
public int nextToken();
public void ordinaryChar(int ch);
public void ordinaryChars(int low, int hi);
public void parseNumbers();
public void pushBack();
public void quoteChar(int ch);
public void resetSyntax();
public void whitespaceChars(int low, int hi);
public void slashStarComments(boolean flag);
public String toString();
public void whitespaceChars(int low, int hi);
public void wordChars(int low, int hi);
```

### **L.6.31 Class *java.io.StringBufferInputStream***

The StringBufferInputStream class supports stream input buffers. The following are defined:

```
// Fields
protected String buffer;
protected int count, pos;

// Constructors
public StringBufferInputStream(String s);

// Methods
public int available();
public int read();
public int read(byte b[], int off, int len);
public void reset();
public long skip(long n);
```

### **L.6.32 Class *java.io.UTFDataFormatException***

Exception that identifies that a malformed UTF-8 string has been read in a data input stream. The following are defined:

```
// Constructors
public UTFDataFormatException();
public UTFDataFormatException(String s);
```

## **L.7 Package *java.lang***

---

### **L.7.1 Class *java.lang.ArithmetiException***

Exception that is thrown when an exceptional arithmetic condition has occurred, such as a division-by-zero or a square root of a negative number. The following are defined:

```
// Constructors
public ArithmetiException();
public ArithmetiException(String s);
```

### **L.7.2 Class *java.lang.ArrayIndexOutOfBoundsException***

Exception that is thrown when an illegal index term in an array has been accessed. The following are defined:

```
// Constructors
public ArrayIndexOutOfBoundsException();
public ArrayIndexOutOfBoundsException(int index);
public ArrayIndexOutOfBoundsException(String s);
```

### **L.7.3 Class *java.lang.ArrayStoreException***

Exception that is thrown when the wrong type of object is stored in an array of objects. The following are defined:

```
// Constructors
public ArrayStoreException();
public ArrayStoreException(String s);
```

### **L.7.4 Class *java.lang.Boolean***

The Boolean class implements the primitive type boolean of an object. Other methods are included for a converting a boolean to a String and vice versa. The following are defined:

```
public final static Boolean FALSE, TRUE;
public final static Boolean TYPE; // Java 1.1

// Constructors
public Boolean(boolean value);
public Boolean(String s);

// Methods
public boolean booleanValue();
public boolean equals(Object obj);
public static boolean getBoolean(String name);
public int hashCode();
public String toString();
public static Boolean valueOf(String s);
```

### **L.7.5 Class *java.lang.Character***

The Character class implements the primitive type character of an object. Other methods are defined for determining the type of a character, and converting characters from uppercase to lowercase and vice versa. The following are defined:

```

// Constants
public final static int MAX_RADIX, MAX_VALUE;
public final static int MIN_RADIX, MIN_VALUE;
public final static int TYPE; // Java 1.1
// Character type constants
public final static byte COMBINING_SPACE_MARK; // Java 1.1
public final static byte CONNECTOR_PUNCTUATION, CONTROL; // Java 1.1
public final static byte CURRENCY_SYMBOL, DASH_PUNCTUATION; // Java 1.1
public final static byte DIGIT_NUMBER, ENCLOSING_MARK; // Java 1.1
public final static byte END_PUNCTUATION, FORMAT; // Java 1.1
public final static byte LETTER_NUMBER, LINE_SEPERATOR; // Java 1.1
public final static byte LOWERCASE_LETTER, MATH_SYMBOL; // Java 1.1
public final static byte MODIFIER_LETTER, MODIFIER_SYMBOL; // Java 1.1

public final static byte NON_SPACING_MARK, OTHER_LETTER; // Java 1.1
public final static byte OTHER_NUMBER, OTHER_PUNCTUATION; // Java 1.1
public final static byte OTHER_SYMBOL, PARAGRAPH_SEPARATOR; // Java 1.1
public final static byte PRIVATE_USE, SPACE_SEPARATOR; // Java 1.1
public final static byte START_PUNCTUATION, SURROGATE; // Java 1.1
public final static byte TITLECASE_LETTER, UNASSIGNED; // Java 1.1
public final static byte UPPERCASE_LETTER; // Java 1.1

// Constructors
public Character(char value);

// Methods
public char charValue();
public static int digit(char ch, int radix);
public boolean equals(Object obj);
public static char forDigit(int digit, int radix);
public static char getNumericValue(char ch); // Java 1.1
public static char getType(char ch); // Java 1.1
public static boolean isDefined(char ch);
public static boolean isDigit(char ch);
public static boolean isISOControl(char ch); // Java 1.1
public static boolean isIdentifierIgnoreable(char ch); // Java 1.1
public static boolean isJavaIdentifierPart(char ch); // Java 1.1
public static boolean isJavaIdentifierStart(char ch); // Java 1.1
public static boolean isJavaLetter(char ch); // Java 1.0
public static boolean isJavaLetterOrDigit(char ch); // Java 1.0
public static boolean isLetter(char ch);
public static boolean isLetterOrDigit(char ch);
public static boolean isLowerCase(char ch); // Java 1.0
public static boolean isSpace(char ch); // Java 1.0
public static boolean isSpaceChar(char ch);
public static boolean isTitleCase(char ch);
public static boolean isUnicodeIdentifierPart(char ch); // Java 1.1
public static boolean isUnicodeIdentifierStart(char ch); // Java 1.1
public static boolean isUpperCase(char ch);
public static boolean isWhitespace(char ch); // Java 1.1
public static char toLowerCase(char ch);
public String toString();
public static char toTitleCase(char ch);
public static char toUpperCase(char ch);

```

### L.7.6 Class *java.lang.Class*

The *Class* class implements the class *Class* and interfaces in a running Java application. The following are defined:

```

// Methods
public static Class forName(String className);

```

```

public ClassLoader getClassLoader();
public Class[] getInterfaces();
public String getName();
public Class getSuperclass();
public boolean isInterface();
public Object newInstance();
public String toString();

```

### **L.7.7 Class *java.lang.ClassCastException***

Exception that is thrown when an object is casted to a subclass which it is not an instance. The following are defined:

```

// Constructors
public ClassCastException();
public ClassCastException(String s);

```

### **L.7.8 Class *java.lang.Compiler***

The Compiler class supports Java-to-native-code compilers and related services. The following are defined:

```

// Methods
public static Object command(Object any);
public static boolean compileClass(Class clazz);
public static boolean compileClasses(String string);
public static void disable();
public static void enable();

```

### **L.7.9 Class *java.lang.Double***

The Double class implements the primitive type double of an object. Other methods are included for a converting a double to a String and vice versa. The following are defined:

```

// Fields
public final static double MAX_VALUE, MIN_VALUE;
public final static double NaN, NEGATIVE_INFINITY, POSITIVE_INFINITY;
public final static double TYPE; // Java 1.1

// Constructors
public Double(double value);
public Double(String s);

// Methods
public static long doubleToLongBits(double value);
public double doubleValue();
public boolean equals(Object obj);
public float floatValue();
public int hashCode();
public int intValue();
public boolean isInfinite();
public static boolean isInfinite(double v);
public boolean isNaN();
public static boolean isNaN(double v);
public static double longBitsToDouble(long bits);
public long longValue();
public String toString();
public static String toString(double d);
public static Double valueOf(String s);

```

### L.7.10 Class *java.lang.Error*

Exception that is thrown when there are serious problems that a reasonable application should not try to catch. The following are defined:

```
// Constructors
public Error();
public Error(String s);
```

### L.7.11 Class *java.lang.Exception*

Exception that is thrown that indicates conditions that a reasonable application might want to catch.

```
// Constructors
public Exception();
public Exception(String s);
```

### L.7.12 Class *java.lang.Float*

The *Float* class implements the primitive type float of an object. Other methods are included for a converting a float to a String and vice versa. The following are defined:

```
// Fields
public final static float MAX_VALUE MIN_VALUE;
public final static float NaN, NEGATIVE_INFINITY, POSITIVE_INFINITY;
public final static float TYPE; // Java 1.1

// Constructors
public Float(double value);
public Float(float value);
public Float(String s);

// Methods
public double doubleValue();
public boolean equals(Object obj);
public static int floatToIntBits(float value);
public float floatValue();
public int hashCode();
public static float intBitsToFloat(int bits);
public int intValue();
public boolean isInfinite();
public static boolean isInfinite(float v);
public boolean isNaN();
public static boolean isNaN(float v);
public long longValue();
public String toString();
public static String toString(float f);
public static Float valueOf(String s);
```

### L.7.13 Class *java.lang.IllegalAccessException*

Exception that is thrown when an application attempts to access or modify a field, or to call a method that it does not have access to. The following are defined:

```
// Constructors
public IllegalAccessException();
public IllegalAccessException(String s);
```

#### **L.7.14 Class *java.lang.IllegalArgumentException***

Exception that is thrown when a method has been passed an illegal or inappropriate argument. The following are defined:

```
// Constructors
public IllegalArgumentException();
public IllegalArgumentException(String s);
```

#### **L.7.15 Class *java.lang.IllegalThreadStateException***

Exception that is thrown to indicate that a thread is not in an appropriate state for the requested operation. The following are defined:

```
// Constructors
public IllegalThreadStateException();
public IllegalThreadStateException(String s);
```

#### **L.7.16 Class *java.lang.IndexOutOfBoundsException***

Exception that is thrown to indicate that an index term is out of range. The following are defined:

```
// Constructors
public IndexOutOfBoundsException();
public IndexOutOfBoundsException(String s);
```

#### **L.7.17 Class *java.lang.Integer***

The Integer class implements the primitive type integer of an object. Other methods are included for a converting a integer to a String and vice versa. The following are defined:

```
// Fields
public final static int MAX_VALUE, MIN_VALUE;
public final static int TYPE; // Java 1.1

// Constructors
public Integer(int value);
public Integer(String s);

// Methods
public Integer decode(String nm); // Java 1.1
public double doubleValue();
public boolean equals(Object obj);
public float floatValue();
public static Integer getInteger(String nm);
public static Integer getInteger(String nm, int val);
public static Integer getInteger(String nm, Integer val);
public int hashCode();
public int intValue();
public long longValue();
public static int parseInt(String s);
public static int parseInt(String s, int radix);
public static String toBinaryString(int i);
public static String toHexString(int i);
public static String toOctalString(int i);
public String toString();
public static String toString(int i);
public static String toString(int i, int radix);
public static Integer valueOf(String s);
```

```
public static Integer valueOf(String s, int radix);
```

### L.7.18 Class *java.lang.InternalError*

Exception that is thrown when an unexpected internal error has occurs. The following are defined:

```
// Constructors
public InternalError();
public InternalError(String s);
```

### L.7.19 Class *java.lang.InterruptedException*

Exception that is thrown when a thread is waiting, sleeping, or otherwise paused for a long time and another thread interrupts it using the interrupt method in class Thread. The following are defined:

```
// Constructors
public InterruptedException();
public InterruptedException(String s);
```

### L.7.20 Class *java.lang.Long*

The Long class implements the primitive type long of an object. Other methods are included for a converting a long to a String and vice versa. The following are defined:

```
// Fields
public final static long MAX_VALUE, MIN_VALUE;
public final static long TYPE; // Java 1.1

// Constructors
public Long(long value);
public Long(String s);

// Methods
public double doubleValue();
public boolean equals(Object obj);
public float floatValue();
public static Long getLong(String nm);
public static Long getLong(String nm, long val);
public static Long getLong(String nm, Long val);
public int hashCode();
public int intValue();
public long longValue();
public static long parseLong(String s);
public static long parseLong(String s, int radix);
public static String toBinaryString(long i);
public static String toHexString(long i);
public static String toOctalString(long i);
public String toString();
public static String toString(long i);
public static String toString(long i, int radix);
public static Long valueOf(String s);
public static Long valueOf(String s, int radix);
```

### L.7.21 Class *java.lang.Math*

The Math class contains methods to perform basic mathematical operations. The following are defined:

```

// Fields
public final static double E;
public final static double PI;

// Methods
public static double abs(double a);
public static float abs(float a);
public static int abs(int a);
public static long abs(long a);
public static double acos(double a);
public static double asin(double a);
public static double atan(double a);
public static double atan2(double a, double b);
public static double ceil(double a);
public static double cos(double a);
public static double exp(double a);
public static double floor(double a);
public static double IEEEremainder(double f1, double f2);
public static double log(double a);
public static double max(double a, double b);
public static float max(float a, float b);
public static int max(int a, int b);
public static long max(long a, long b);
public static double min(double a, double b);
public static float min(float a, float b);
public static int min(int a, int b);
public static long min(long a, long b);
public static double pow(double a, double b);
public static double random();
public static double rint(double a);
public static long round(double a);
public static int round(float a);
public static double sin(double a);
public static double sqrt(double a);
public static double tan(double a);

```

#### **L.7.22 Class *java.lang.NegativeArraySizeException***

Exception that is thrown when an array is created with a negative size.

```

// Constructors
public NegativeArraySizeException();
public NegativeArraySizeException(String s);

```

#### **L.7.23 Class *java.lang.NullPointerException***

Exception that is thrown when an application attempts to use a null pointer. The following are defined:

```

// Constructors
public NullPointerException();
public NullPointerException(String s);

```

#### **L.7.24 Class *java.lang.Number***

The Number class contains the superclass of classes for float, double, integer and long. It can be used to convert values into int, long, float or double. The following are defined:

```

// Methods
public abstract double doubleValue();
public abstract float floatValue();

```

```
public abstract int intValue();
public abstract long longValue();
```

### **L.7.25 Class *java.lang.NumberFormatException***

Exception that is thrown when an application attempts to convert a string to one of the numeric types, but that the string does not have the appropriate format.

```
// Constructors
public NumberFormatException();
public NumberFormatException(String s);
```

### **L.7.26 Class *java.lang.Object***

The Object class contains the root of the class hierarchy. The following are defined:

```
// Constructors
public Object();

// Methods
protected Object clone();
public boolean equals(Object obj);
protected void finalize();
public final Class getClass();
public int hashCode();
public final void notify();
public final void notifyAll();
public String toString();
public final void wait();
public final void wait(long timeout);
public final void wait(long timeout, int nanos);
```

### **L.7.27 Class *java.lang.OutOfMemoryError***

Exception that is thrown when an application runs out of memory. The following are defined:

```
// Constructors
public OutOfMemoryError();
public OutOfMemoryError(String s);
```

### **L.7.28 Class *java.lang.Process***

The Process class contains methods which are used to control the process. The following are defined:

```
// Constructors
public Process();
// Methods
public abstract void destroy();
public abstract int exitValue();
public abstract InputStream getErrorStream();
public abstract InputStream getInputStream();
public abstract OutputStream getOutputStream();
public abstract int waitFor();
```

### **L.7.29 Class *java.lang.Runtime***

The Runtime class allows the application to interface with the environment in which it is running. The following are defined:

```

// Methods
public Process exec(String command);
public Process exec(String command, String envp[]);
public Process exec(String cmdarray[]);
public Process exec(String cmdarray[], String envp[]);
public void exit(int status);
public long freeMemory();
public void gc();
public InputStream getLocalizedInputStream(InputStream in);           // Java 1.0
public OutputStream getLocalizedOutputStream(OutputStream out);      // Java 1.0
public static Runtime getRuntime();
public void load(String filename);
public void loadLibrary(String libname);
public void runFinalization();
public long totalMemory();
public void traceInstructions(boolean on);
public void traceMethodCalls(boolean on);

```

### **L.7.30 Class *java.lang.SecurityManager***

The SecurityManager class is an abstract class that allows applications to determine if it is safe to execute a given operation. The following are defined:

```

// Fields
protected boolean inCheck;

// Constructors
protected SecurityManager();

// Methods
public void checkAccept(String host, int port);
public void checkAccess(Thread g);
public void checkAccess(ThreadGroup g);
public void checkConnect(String host, int port);
public void checkConnect(String host, int port, Object context);
public void checkCreateClassLoader();
public void checkDelete(String file);
public void checkExec(String cmd);
public void checkExit(int status);
public void checkLink(String lib);
public void checkListen(int port);
public void checkPackageAccess(String pkg);
public void checkPackageDefinition(String pkg);
public void checkPropertiesAccess();
public void checkPropertyAccess(String key);
public void checkRead(FileDescriptor fd);
public void checkRead(String file);
public void checkRead(String file, Object context);
public void checkSetFactory();
public boolean checkTopLevelWindow(Object window);
public void checkWrite(FileDescriptor fd);
public void checkWrite(String file);
protected int classDepth(String name);
protected int classLoaderDepth();
protected ClassLoader currentClassLoader();
protected Class[] getClassContext();
public boolean getInCheck();
public Object getSecurityContext();
protected boolean inClass(String name);
protected boolean inClassLoader();

```

### L.7.31 Class *java.lang.StackOverflowError*

Exception that is thrown when a stack overflow occurs. The following are defined:

```
// Constructors
public StackOverflowError();
public StackOverflowError(String s);
```

### L.7.32 Class *java.lang.String*

The String class represents character strings. As in C, a string is delimited by inverted commas. It contains string manipulation methods, such as concat (string concatenation), equals (if string is equal to), toLowerCase (to convert a string to lowercase), and so on. The following are defined:

```
// Constructors
public String();
public String(byte ascii[], int hibyte);                                // Java 1.0
public String(byte ascii[], int hibyte, int offset, int count);        // Java 1.0
public String(char value[]);
public String(char value[], int offset, int count);
public String(String value);
public String(StringBuffer buffer);
public String(byte ascii[], int offset, int length, String enc);      // Java 1.1

// Methods
public char charAt(int index);
public int compareTo(String anotherString);
public String concat(String str);
public static String copyValueOf(char data[]);
public static String copyValueOf(char data[], int offset, int count);
public boolean endsWith(String suffix);
public boolean equals(Object anObject);
public boolean equalsIgnoreCase(String anotherString);
public void getBytes(int srcBegin, int srcEnd, byte dst[], int dstBegin);
public void getChars(int srcBegin, int srcEnd,           char dst[], int dstBegin);
public int hashCode();
public int indexOf(int ch);
public int indexOf(int ch, int fromIndex);
public int indexOf(String str);
public int indexOf(String str, int fromIndex);
public String intern();
public int lastIndexOf(int ch);
public int lastIndexOf(int ch, int fromIndex);
public int lastIndexOf(String str);
public int lastIndexOf(String str, int fromIndex);
public int length();
public boolean regionMatches(boolean ignoreCase, int toffset,
                           String other, int ooffset, int len);
public boolean regionMatches(int toffset, String other,   int offset, int len);
public String replace(char oldChar, char newChar);
public boolean startsWith(String prefix);
public boolean startsWith(String prefix, int toffset);
public String substring(int beginIndex);
public String substring(int beginIndex, int endIndex);
public char[] toCharArray();
public String toLowerCase();
public String toLowerCase(Locale locale);                                // Java 1.1
```

```

public String toString();
public String toUpperCase();
public String toUpperCase(Locale locale); // Java 1.1

public String trim();
public static String valueOf(boolean b);
public static String valueOf(char c);
public static String valueOf(char data[]);
public static String valueOf(char data[], int offset, int count);
public static String valueOf(double d);
public static String valueOf(float f);
public static String valueOf(int i);
public static String valueOf(long l);
public static String valueOf(Object obj);

```

### **L.7.33 Class *java.lang.StringBuffer***

The StringBuffer class implements a string buffer. The following are defined:

```

// Constructors
public StringBuffer();
public StringBuffer(int length);
public StringBuffer(String str);
// Methods
public StringBuffer append(boolean b);
public StringBuffer append(char c);
public StringBuffer append(char str[]);
public StringBuffer append(char str[], int offset, int len);
public StringBuffer append(double d);
public StringBuffer append(float f);
public StringBuffer append(int i);
public StringBuffer append(long l);
public StringBuffer append(Object obj);
public StringBuffer append(String str);
public int capacity();
public char charAt(int index);
public void ensureCapacity(int minimumCapacity);
public void getChars(int srcBegin, int srcEnd, char dst[], int dstBegin);
public StringBuffer insert(int offset, boolean b);
public StringBuffer insert(int offset, char c);
public StringBuffer insert(int offset, char str[]);
public StringBuffer insert(int offset, double d);
public StringBuffer insert(int offset, float f);
public StringBuffer insert(int offset, int i);
public StringBuffer insert(int offset, long l);
public StringBuffer insert(int offset, Object obj);
public StringBuffer insert(int offset, String str);
public int length();
public StringBuffer reverse();
public void setCharAt(int index, char ch);
public void setLength(int newLength);
public String toString();

```

### **L.7.34 Class *java.lang.StringIndexOutOfBoundsException***

Exception that is thrown when a string is indexed with a negative value or a value which is greater than or equal to the size of the string. The following are defined:

```

// Constructors
public StringIndexOutOfBoundsException();
public StringIndexOutOfBoundsException(int index);

```

```
public StringIndexOutOfBoundsException(String s)
```

### L.7.35 Class *java.lang.System*

The System class implements a number of system methods. The following are defined:

```
// Fields
public static PrintStream err, in, out;
// Methods
public static void arraycopy(Object src, int src_position,
    Object dst, int dst_position, int length);
public static long currentTimeMillis();
public static void exit(int status);
public static void gc();
public static Properties getProperties();
public static String getProperty(String key);
public static String getProperty(String key, String def);
public static SecurityManager getSecurityManager();
public static void load(String filename);
public static void loadLibrary(String libname);
public static void runFinalization();
public static void setProperties(Properties props);
public static void setSecurityManager(SecurityManager s);
```

### L.7.36 Class *java.lang.Thread*

The Thread class implements one or more threads. The following are defined:

```
// Fields
public final static int MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY;

// Constructors
public Thread();
public Thread(Runnable target);
public Thread(Runnable target, String name);
public Thread(String name);
public Thread(ThreadGroup group, Runnable target);
public Thread(ThreadGroup group, Runnable target, String name);
public Thread(ThreadGroup group, String name);

// Methods
public static int activeCount();
public void checkAccess();
public int countStackFrames();
public static Thread currentThread();
public void destroy();
public static void dumpStack();
public static int enumerate(Thread tarray[]);
public final String getName();
public final int getPriority();
public final ThreadGroup getThreadGroup();
public void interrupt();
public static boolean interrupted();
public final boolean isAlive();
public final boolean isDaemon();
public boolean isInterrupted();
public final void join();
public final void join(long millis);
public final void join(long millis, int nanos)
public final void resume();
public void run();
public final void setDaemon(boolean on);
```

```

public final void setName(String name);
public final void setPriority(int newPriority);
public static void sleep(long millis);
public static void sleep(long millis, int nanos)
public void start();
public final void stop();
public final void stop(Throwable obj);
public final void suspend();
public String toString();
public static void yield();

```

### **L.7.37 Class *java.lang.ThreadGroup***

The ThreadGroup class implements a set of threads. The following are defined:

```

// Constructors
public ThreadGroup(String name);
public ThreadGroup(ThreadGroup parent, String name);

// Methods
public int activeCount();
public int activeGroupCount();
public final void checkAccess();
public final void destroy();
public int enumerate(Thread list[]);
public int enumerate(Thread list[], boolean recurse);
public int enumerate(ThreadGroup list[]);
public int enumerate(ThreadGroup list[], boolean recurse);
public final int getMaxPriority();
public final String getName();
public final ThreadGroup getParent();
public final boolean isDaemon();
public void list();
public final boolean parentOf(ThreadGroup g);
public final void resume();
public final void setDaemon(boolean daemon);
public final void setMaxPriority(int pri);
public final void stop();
public final void suspend();
public String toString();
public void uncaughtException(Thread t, Throwable e);

```

### **L.7.38 Class *java.lang.Throwable***

The Throwable class is the superclass of all errors and exceptions in the Java language. The following are defined:

```

// Constructors
public Throwable();
public Throwable(String message);

// Methods
public Throwable fillInStackTrace();
public String getMessage();
public void printStackTrace();
public void printStackTrace(PrintStream s);
public String toString();

```

### **L.7.39 Class *java.lang.UnknownError***

Exception that is thrown when an unknown error occurs. The following are defined:

```
// Constructors
public UnknownError();
public UnknownError(String s);
```

## L.8 Package java.net

---

### L.8.1 Class *java.net.DatagramPacket*

The DatagramPacket class implements datagram packets. The following are defined:

```
// Constructors
public DatagramPacket(byte[] ibuf, int ilength);
public DatagramPacket(byte[] ibuf, int ilength, InetAddress iaddr, int iport);

// Methods
public synchronized InetAddress getAddress();
public synchronized byte[] getData();
public synchronized int getLength();
public synchronized int getPort();
public synchronized void setAddress(InetAddress iaddr);                                // Java 1.1
public synchronized void setDate(byte[] ibuf);                                         // Java 1.1
public synchronized void setLength(int ilength);                                       // Java 1.1
public synchronized void setPort(int iport);                                         // Java 1.1
```

### L.8.2 Class *java.net.InetAddress*

The InetAddress class represents Internet addresses. The following are defined:

```
// Methods
public InetAddress[] getAllByName(String host);
public InetAddress getByName(String host);
public InetAddress getLocalHost(String host);
public boolean equals(Object obj);
public byte[] getAddress();
public String getHostAddress();
public String getHostName();
public int hashCode();
public boolean isMulticastAddress();                                              // Java 1.1
public String toString();
```

### L.8.3 Class *java.net.ServerSocket*

The ServerSocket class represents servers which listen for a connection from clients. The following are defined:

```
// Constructors
public ServerSocket(int port);
public ServerSocket(int port, int backlog);
public ServerSocket(int port, int backlog, InetAddress bindAddr);                      // Java 1.1

// Methods
public Socket accept();
public void close();
public InetAddress getInetAddress();
```

```
public synchronized int getSoTimeout();                                // Java 1.1
public String toString();
```

#### L.8.4 Class *java.net.Socket*

The Socket class represents socket connections over a network. The following are defined:

```
// Constructors
public Socket(String host, int port);
public Socket(InetAddress addr, int port);
public Socket(InetAddress addr, int port, boolean stream);           // Java 1.0
public Socket(String host, int port, InetAddress addr, int localport); // Java 1.1
public Socket(InetAddress addr, int port, InetAddress localAddress,
              int localport);                                         // Java 1.1

// Methods
public synchronized void close();
public InetAddress getInetAddress();
public InputStream getInputStream();
public InetAddress getLocalAddress();                                     // Java 1.1
public int getLocalPort();
public OutputStream getOutputStream();
public int getPort();
public int getSoLinger();                                              // Java 1.1
public synchronized int getSoTimed();                                     // Java 1.1
public boolean getTcpNoDelay();                                         // Java 1.1
public void setSoLinger(boolean on, int val);                           // Java 1.1
public synchronized void setSoTimed(int timeout);                      // Java 1.1
public void setTcpNoDelay(boolean on);                                    // Java 1.1
public String toString();
```

#### L.8.5 Class *java.net.SocketImpl*

The SocketImpl class represents socket connections over a network. The following are defined:

```
// Methods
public abstract void accept(SocketImpl s);
public abstract int available();
public abstract void bind(InetAddress host, int port);
public abstract void close();
public abstract void connect(String host, int port);
public abstract void connect(InetAddress addr, int port);
public abstract void create(boolean stream);
public FileDescriptor getFileDescriptor();
public InetAddress getInetAddress();
public abstract InetAddress getInputStream();
```

#### L.8.6 Class *java.net.URL*

The URL class represents Uniform Resource Locators. The following are defined:

```
// Constructors
public URL(String protocol, String host, int port, String file);
public URL(String protocol, String host, String file);
public URL(String spec);
public URL(URL context, String spec);

// Methods
public boolean equals(Object obj);
```

```

public final Object getContent();
public String getFile();
public String getHost();
public int getPort();
public String getProtocol();
public String getRef();
public int hashcode();
public URLConnection openConnection();
public final InputStream openStream();
public boolean sameFile(URL other);
public String toExternalForm();
public String toString();

```

## **L.9 Package java.utils**

---

### **L.9.1 Class java.utils.BitSet**

The BitSet class implements boolean operations. The following are defined:

```

// Constructors
public BitSet();
public BitSet(int nbits);

// Methods
public void and(BitSet set);
public void clear(int bit);
public Object clone();
public boolean equals(Object obj);
public boolean get(int bit);
public int hashCode();
public void or(BitSet set);
public void set(int bit);
public int size();
public String toString();
public void xor(BitSet set);

```

### **L.9.2 Class java.utils.Calender**

The Calender class has been added with Java 1.1. It supports dates and times.

### **L.9.3 Class java.utils.Date**

The Date class supports dates and times. The following are defined:

```

// Constructors
public Date();
public Date(int year, int month, int date); // Java 1.0
public Date(int year, int month, int date, int hrs, int min); // Java 1.0
public Date(int year, int month, int date, int hrs, int min, int sec); // Java 1.0
public Date(long date); // Java 1.0
public Date(String s); // Java 1.0

// Methods
public boolean after(Date when);
public boolean before(Date when);
public boolean equals(Object obj);

```

```

public int getDate();                                // Java 1.0
public int getDay();                                 // Java 1.0
public int getHours();                               // Java 1.0
public int getMinutes();                             // Java 1.0
public int getMonth();                              // Java 1.0
public int getSeconds();                            // Java 1.0
public long getTime();                             // Java 1.0
public int getTimezoneOffset();                     // Java 1.0
public int getYear();                               // Java 1.0
public int hashCode();                            // Java 1.0
public static long parse(String s);                // Java 1.0
public void  setDate(int date);                   // Java 1.0
public void setHours(int hours);                  // Java 1.0
public void setMinutes(int minutes);               // Java 1.0
public void setMonth(int month);                  // Java 1.0
public void setSeconds(int seconds);               // Java 1.0
public void  setTime(long time);                  // Java 1.0
public void setYear(int year);                    // Java 1.0
public String toGMTString();                      // Java 1.0
public String toLocaleString();                    // Java 1.0
public String toString();                           // Java 1.0
public static long UTC(int year, int month, int date, int hrs, int min,
                      int sec);                         // Java 1.0

```

#### L.9.4 Class *java.util.Dictionary*

The Dictionary class is the abstract parent of any class which maps keys to values. The following are defined:

```

// Constructors
public Dictionary();

// Methods
public abstract Enumeration elements();
public abstract Object get(Object key);
public abstract boolean isEmpty();
public abstract Enumeration keys();
public abstract Object put(Object key, Object value);
public abstract Object remove(Object key);
public abstract int size();

```

#### L.9.5 Class *java.util.EmptyStackException*

The EmptyStackException is thrown when the stack is empty. The following is defined:

```

// Constructors
public EmptyStackException();

```

#### L.9.6 Class *java.util.Hashtable*

This Hashtable class supports a hashtable which maps keys to values. The following are defined:

```

// Constructors
public Hashtable();
public Hashtable(int initialCapacity);
public Hashtable(int initialCapacity, float loadFactor);
// Methods
public void clear();
public Object clone();

```

```

public boolean contains(Object value);
public boolean containsKey(Object key);
public Enumeration elements();
public Object get(Object key);
public boolean isEmpty();
public Enumeration keys();
public Object put(Object key, Object value);
protected void rehash();
public Object remove(Object key);
public int size();
public String toString();

```

### **L.9.7 Class *java.util.NoSuchElementException***

The `NoSuchElementException` is thrown when there are no more elements in the enumeration. The following are defined:

```

// Constructors
public NoSuchElementException();
public NoSuchElementException(String s);

```

### **L.9.8 Class *java.util.Observable***

The `Observable` class represents an observable object. The following are defined:

```

// Constructors
public Observable();
// Methods
public void addObserver(Observer o);
protected void clearChanged();
public int countObservers();
public void deleteObserver(Observer o);
public void deleteObservers();
public boolean hasChanged();
public void notifyObservers();
public void notifyObservers(Object arg);
protected void setChanged();

```

### **L.9.9 Class *java.util.Properties***

The `Properties` class represents a persistent set of properties. The following are defined:

```

// Fields
protected Properties defaults;

// Constructors
public Properties();
public Properties(Properties defaults);

// Methods
public String getProperty(String key);
public String getProperty(String key, String defaultValue);
public void list(PrintStream out);
public void load(InputStream in);
public Enumeration propertyNames();
public void save(OutputStream out, String header);

```

### **L.9.10 Class *java.util.Random***

The `Random` class implements pseudo-random generator functions. The following are defined:

```
// Constructors
public Random();
public Random(long seed);

// Methods
public double nextDouble();
public float nextFloat();
public double nextGaussian();
public int nextInt();
public long nextLong();
public void setSeed(long seed);
```

#### **L.9.11 Class *java.util.Stack***

The Stack class implements a last-in-first-out (LIFO) stack.

```
// Constructors
public Stack();

// Methods
public boolean empty();
public Object peek();
public Object pop();
public Object push(Object item);
public int search(Object o);
```

#### **L.9.12 Class *java.util.StringTokenizer***

The StringTokenizer class allows strings to be split into tokens. The following are defined:

```
// Constructors
public StringTokenizer(String str);
public StringTokenizer(String str, String delim);
public StringTokenizer(String str, String delim, boolean returnTokens);

// Methods
public int countTokens();
public boolean hasMoreElements();
public boolean hasMoreTokens();
public Object nextElement();
public String nextToken();
public String nextToken(String delim);
```

#### **L.9.13 Class *java.util.Vector***

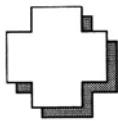
The Vector class implements a growable array of objects. The following are defined:

```
// Fields
protected int capacityIncrement;
protected int elementCount;
protected Object elementData[];

// Constructors
public Vector();
public Vector(int initialCapacity);
public Vector(int initialCapacity, int capacityIncrement);

// Methods
public final void addElement(Object obj);
public final int capacity();
```

```
public Object clone();
public final boolean contains(Object elem);
public final void copyInto(Object anArray[]);
public final Object elementAt(int index);
public final Enumeration elements();
public final void ensureCapacity(int minCapacity)
public final Object firstElement();
public final int indexOf(Object elem);
public final int indexOf(Object elem, int index);
public final void insertElementAt(Object obj, int index);
public final boolean isEmpty();
public final Object lastElement();
public final int lastIndexOf(Object elem);
public final int lastIndexOf(Object elem, int index);
public final void removeAllElements();
public final boolean removeElement(Object obj);
public final void removeElementAt(int index);
public final void setElementAt(Object obj, int index);
public final void setSize(int newSize);
public final int size();
public final String toString();
public final void trimToSize();
```



---

## Glossary

---

**100Base-FX** IEEE-defined standard for 100 Mbps Ethernet using multimode fiber-optic cable.

**100Base-TX (802.3u)** IEEE-defined standard for 100Mbps Ethernet using two pairs of Cat-5 twisted-pair cable.

**100VG-AnyLAN** HP-derived network architecture based on the IEEE 802.12 standard that uses 100Mbps transmission rates. It uses a centrally controlled access method referred to as the Demand Priority Protocol (DPP), where the end node requests permission to transmit and the hub determines which node may do so, depending on the priority of the traffic.

**10Base-T** IEEE-defined standard for 10Mbps Ethernet using twisted-pair cables.

**802.10** IEEE-defined standard for LAN security. It is sometimes used by network switches as a VLAN protocol and uses a technique where frames on any LAN carry a virtual LAN identification. For large networks this can be modified to provide security over the Internet.

**802.12 Demand Priority Protocol**

IEEE-defined standard of transmitting 100Mbps over voice grade (telephone) twisted-pair cabling. See 100VG-AnyLAN.

**802.1d** IEEE-defined bridging standard for Spanning Tree protocol that is used to determine factors on how bridges (or switches) forward packets and avoid networking loops. Networks, which use redundant loops (for alternative routes), need to implement the IEEE 802.1d standard to stop packets from looping forever.

**802.2** A set of IEEE-defined specifications for Logical Link Control (LLC) layer. It provides some network functions and interfaces the IEEE 802.5, or IEEE 802.3, standards to the transport layer.

**802.3** IEEE-defined standard for CSMA/CD networks. IEEE 802.3 is the most popular implement of Ethernet.

**802.3u** IEEE-defined standard for 100Mbps Fast Ethernet. It also covers a technique called autosensing which allows 100 Mbps devices to connect to 10 Mbps devices.

<b>802.4</b>	IEEE-defined token bus specifications.
<b>802.5</b>	IEEE-defined standard for token ring networks.
<b>Adapter</b>	Device which usually connects a node onto a network, normally called a network interface adapter (NIC).
<b>Adaptive cut-through switching</b>	A forwarding technique on a switch which determines when the error count on frames received has exceeded the pre-configured limits. When this count is exceeded, it modifies its own operating state so that it no longer performs cut-through switching and goes into a store-and-forward mode. The cut-through method is extremely fast but suffers from the inability to check the CRC field. Thus if incorrect frames are transmitted they could have severe effects on the network segment. This is overcome with an adaptive cut-through switch by checking the CRC as the frame moves through the switch. When errors become too great the switch implements a store-and-forward method.
<b>Adaptive delta modulation PCM</b>	Similar to delta modulation PCM, but uses a number of bits to code the slope of the signal.
<b>Adaptive Huffman coding</b>	Uses a variable Huffman coding technique which responds to local changes in probabilities.
<b>Address</b>	A unique label for the location of data or the identity of a communications device. This address can either be numeric or alphanumeric.
<b>Address aging</b>	The time that a dynamic address stays in the address routing table of a bridge or switch.
<b>Address Resolution Protocol (ARP)</b>	A TCP/IP process which dynamically binds an IP address to a hardware address (such as an Ethernet MAC address). It can only operate across a single network segment.
<b>Address tables</b>	These are used routers, switches and hubs to store either physical (such as MAC addresses) or higher-level addresses (such as IP addresses). The tables map node addresses to network addresses or physical domains. These address tables are dynamic and change due to nodes moving around the network.
<b>Agent</b>	A program which allows users to configure or fault-find nodes on a network.

**Aging** The removing of address in the address table of a router or switch that no longer are referenced to forward a packet.

**American National Standards Institute (ANSI)**

ANSI is a non-profit organization which is made up of expert committees that publish standards for national industries.

**American Standard Code for Information Interchange (ASCII)**

An ANSI-defined character alphabet which has since been adopted as a standard international alphabet for the interchange of characters.

**Amplitude modulation (AM)**

Information is contained in the amplitude of a carrier.

**Amplitude-Shift Keying (ASK)**

Uses two, or more, amplitudes to represent binary digits. Typically used to transmit binary over speech-limited channels.

**Application layer** The highest layer of the OSI model.

**Asynchronous** Communication which does not depend on a clock.

**Asynchronous transmission**

Transmission where individual characters are sent one-by-one. Normally each character is delimited by a start and a stop bit. With asynchronous communications the transmitter and receiver only have to be roughly synchronized.

**ATM (Asynchronous Transfer Mode)**

Networking technology which involves sending 53-byte fast packets (ATM cell), as specified by the ANSI T1S1 subcommittee. The first 5 bytes are the header and the remaining bytes are the information field which can hold 48 bytes of data. Optionally the data can contain a 4-byte ATM adaptation layer and 44 bytes of actual data. The ATM adaptation layer field allows for fragmentation and reassembly of cells into larger packets at the source and destination respectively. The control field also contains bits which specify whether this is a flow control cell or an ordinary data cell, a bit to indicate whether this packet can be deleted in a congested network, and so on.

**AUI** Connection between the network adapter and an external transceiver.

**Automatic broadcast control**

Technique which minimizes broadcast and multicast traffic flooding through a switch. A switch acts as a proxy server and screens previously resolved ARP. This eliminates broadcasts associated with them.

<b>Autonegotiation</b>	Technique used by a IEEE 802.3u node which determines whether a device that it is receiving or transmitting data in one of a number of Ethernet modes (100Base-TX, 100Base-TX Full Duplex, 10Base-T, 10Base-T Full Duplex or 100Base-T4). When the mode is learned, the device then adjusts to either transmission mode.
<b>Autosensing</b>	Used by a 100Base-TX device to determine if the incoming data is transmitted at 10Mbps or 100Mbps.
<b>Back pressure</b>	Technique which slows the incoming data rate into the buffer of a 802.3 port preventing it from receiving too much data. Switches which implement back pressure will transmit a jam signal to stop data input.
<b>Backbone network</b>	The portion of a communications facility that connects primary nodes. A primary shared communications path that serves multiple users at designated jumping-off points.
<b>Bandwidth</b>	In an analogue system it is defined as the range of frequencies contained in a signal. As an approximation it is the difference between the highest and lowest frequency in the signal. In a digital transmission system it is normally quoted at the bit per second.
<b>Bandwidth allocation control protocol (BACP)</b>	Protocol which monitors network traffic and allows or disallows access to users, depending on their needs. It is awaiting approval by the IETF.
<b>Baseband</b>	Data transmission using unmodulated signals.
<b>Basic rate interface (BRI)</b>	Connection between ISDN and the user. It has three separate channels, one D-channel (which carries control information) and two B channels (which carry data).
<b>Baud rate</b>	The number of signaling elements sent per second with a RS-232, or modem, communications. In RS-232 the baud rate is equal to the bit-rate. With modems, two or more bits can be encoded as a single signaling element, such as two bits being represented by four different phase shifts (or one signaling element). The signaling element could change its amplitude, frequency or phase-shift to increase the bit-rate. Thus the bit-rate is a better measure of information transfer.
<b>Bit stuffing</b>	The insertion of extra bits to stop the appearance of a defined sequence. In HDLC the bit sequence 01111110 delimits the start and end of a frame. Bit stuffing stops this bit sequence from occurring anywhere in the frame by the receiver inserting a 0 whenever there are five consecutive 1's transmitted. At the receive if five consecutive 1's are followed by a 0 then the 0 is deleted.

<b>BNC</b>	A commonly used connector for coaxial cable.
<b>BOOTP</b>	A standard TCP/IP protocol which allows nodes to be dynamically allocated an IP address.
<b>Bridge</b>	A device which physically links two or more networks using the same communications protocols, such as Ethernet/ Ethernet or token ring/ token ring. It allows for the filtering of data between network segments.
<b>Broadband</b>	Data transmission using multiplexed data using an analogue signal or high-frequency electromagnetic waves.
<b>Broadcast</b>	Message sent to all users on the network.
<b>Broadcast domain</b>	Network where broadcasts can be reported to all nodes on the network bounded by routers. A broadcast packet cannot traverse a router.
<b>Broadcast storm</b>	Flood of broadcast packets generated by a broadcast transmission where high numbers of receivers are targeted for a long period of time.
<b>Buffer</b>	A temporary-storage space in memory.
<b>Bus</b>	A network topology where all nodes share a common transmission medium.
<b>Byte</b>	A group of eight bits, see octet.
<b>Capacity</b>	The maximum data rate in Mbps.
<b>Carrier Sense Multiple Access/ Carrier Detect (CSMA/CD)</b>	
	A network where all node share a common bus. Nodes must contend for the bus and if a collision occurs then all colliding nodes back-off for a random time period.
<b>Cat-3 cable</b>	An EIA/TIA-568 wiring standard for unshield or shielded twisted pair cables.
<b>Cat-5 cable</b>	An EIA/TIA-568 wiring standard for unshield or shielded twisted pair cables for the transmission of over 100 Mbps.
<b>CHAP (challenge-handshake authentication protocol)</b>	
	Identification method used by PPP to determine the originator of a connection.
<b>Checksum</b>	An error-detection scheme in which bits are grouped to form integer values and then each of the value is summated. Normally, the negative

of this value is then added as a checksum. At the receiver, all the grouped values and the checksum are summated and, in the absence of errors, the result should be zero.

<b>Client</b>	Node or program that connects to a server node or program.
<b>Coaxial cable</b>	A transmission medium consisting of one or more central wire conductors, surrounded by a insulating layer and encased in either a wire mesh or extruded metal sheathing. It supports RF frequencies from 50 to about 500 MHz. It comes in either a 10-mm diameter (thick coax) or a 5-mm diameter (thin coax).
<b>Collision</b>	Occurs when one or more devices try to transmit over an Ethernet network.
<b>Copper distributed data interface (CDDI)</b>	FDDI over copper.
<b>Cost</b>	An arbitrary value used by routers to compare different routes. Typically it is measured by hop counts, typical time delays or bandwidth.
<b>CRC</b>	Cyclic Redundancy Check. An error-detection scheme. Used in most HDLC-related data link applications.
<b>cross-talk</b>	Interference noise caused by conductors radiating electromagnetic radiation to couple into other conductors.
<b>Cut-through switching</b>	Technique where a switching device directs a packet to the destination port(s) as soon as it receives the destination and source address scanned from the packet header.
<b>Data Communications Equipment (DCE)</b>	Devices which establish, maintain and terminate a data communications conversation.
<b>Data link layer</b>	Second layer of the OSI model which is responsible for link, error and flow control. It normally covers the framing of data packets, error control and physical addressing. Typical data link layers include Ethernet and FDDI.
<b>Data Terminal Equipment (DTE)</b>	Device at the end of the data communications connection.
<b>Delta modulation PCM</b>	Uses a single-bit code to represent the analogue signal. A 1 is transmission when the current sample increases its level, else a 0 is transmitted.

Delta modulation PCM requires a higher sampling rate than the Nyquist rate, but the actual bit rate is normally lower.

#### **Destination MAC address**

A 6-byte data unique of the destination MAC address. It is normally quoted as a 12-digit hexadecimal number (such as A5:B2:10:64:01:44).

#### **Destination network address**

A unique Internet Protocol (IP) or Internet Packet Exchange (IPX) address of the destination node.

#### **Differential encoding**

Source coding method which is used to code the difference between two samples. Typically used in real-time signals where there is limited change between one sample and the next, such as in audio and speech.

#### **Dynamic host control protocol (DHCP)**

It manages a pool of IP addresses for computers without a known IP address. This allows a finite number of IP addresses to be reused quickly and efficiently by many clients.

#### **Electronic Industries Association (EIA)**

Organization that have defined many of the serial communications standards.

#### **Entropy coding**

Coding scheme which does not take into account the characteristics of the data and treats all the bits in the same way. It produces lossless coding. Typical methods used are statistical encoding and suppressing repetitive sequences.

#### **Ethernet**

A local area network which uses coaxial, twisted-pair or fiber optic cable as a communication medium. It transmits at a rate of 10 Mbps and was developed by DEC, Intel and Xerox Corporation. The IEEE 802.3 network standard is based upon Ethernet.

#### **Ethernet address**

A 48-bit number that identifies a node on an Ethernet network. Ethernet addresses are assigned by the Xerox Corporation.

#### **Even parity**

An error-detection scheme where defined bit-grouping have an even number of 1's.

#### **Extended Binary Coded Decimal Interchange Code (EBCDIC)**

An 8-bit code alphabet developed by IBM allowing 256 different bit patterns for character definitions.

#### **Fast Ethernet**

See IEEE 802.3u standard.

#### **Fat pipe**

Term used to indicate a high level of bandwidth for the defined port.

**Fiber Distributed Data Interface (FDDI)**

A standard network technology that uses a dual counter-rotating token-passing fiber ring. It operates at 100 Mbps and provides for reliable backbone connections.

**File server** Computer that allows the sharing of file over a network.

**File transfer protocol (FTP)**

A protocol for transmitting files between host computers using the TCP/IP protocol.

**Firewall** Device which filters incoming and outgoing traffic.

**Flow control** Procedure to regulate the flow of data between two nodes.

**Forward adaptive bit allocation**

This technique, used in audio compression, makes bit allocation decisions adaptively, depending on signal content.

**Fragment free cut-through switching**

A modified cut-through switching technique where a switch or switch module waits until it has received a large enough packet to determine if it is error free.

**Frame** Normally associated a packet which has layer 2 information added to it. Packets are thus contained within frames. Frames and packets have variable lengths as opposed to cells which have fixed length.

**Frame check sequence (FCS)**

Standard error detection scheme.

**Frequency-shift Keying (FSK)**

Uses two, or more, frequencies to represent binary digits. Typically used to transmit binary data over speech-limited channels.

**Full duplex** Simultaneous, two-way communications.

**Gateway** A device that connects networks using different communications protocols, such as between Ethernet and FDDI. It provides protocol translation, in contrast to a bridge which connects two networks that are of the same protocol.

**GIF** Standard image compression technique which is copyrighted by CompuServe Incorporated. It uses LZW compression and supports a palette of 256 24-bit colors (16.7M colors). GIF support local and global color tables and animated images.

**Half-duplex (HDX)**

	Two-way communications, one at a time.
<b>Handshaking</b>	A reliable method for two devices to pass data.
<b>HDLC</b>	ISO standard for the data link layer.
<b>Hello packet</b>	Message transmitted from a root bridge to all other bridges in the network to constantly verify the Spanning Tree setup.
<b>Hop</b>	The number of gateways and routers in a transmission path.
<b>Hop count</b>	Used by the RIP routing protocol to measure the distance between a source and a destination.
<b>Host</b>	A computer that communicates over a network. A host can both initiate communications and respond to communications that are addressed to it.
<b>Huffman coding</b>	Uses a variable length code for each of the elements within the data. It normally analyzes the probability of element in the data and codes the most probable with fewer bits than the least probable.
<b>Hub</b>	A hub is a concentration point for data and repeats data from one node to all other connected nodes.
<b>Hypertext markup language (HTML)</b>	
Standard language that allows the integration of text and images over a distributed network.	
<b>Integrated systems digital network (ISDN)</b>	
Communication technology that contains two data channels (2B) and a control channel (H). It supports two 64 kbps data channels and sets up a circuit-switched connection.	
<b>International Telegraph Union Telecommunications Standards Sector (ITU-TSS)</b>	
Organization which has replaced the CCITT.	
<b>Internet</b>	Connection of nodes on a global network which use a DARPA-defined Internet address.
<b>internet</b>	Two or more connected networks that may, or may not, use the same communication protocol.
<b>Internet address</b>	An address that conforms to the DARPA-defined Internet protocol. A unique, four-byte number identifies a host or gateway on the Internet. This consists of a network number followed by a host number. The host number can be further divide into a subnet number.

**Internet Engineering Task Force (IETF)**

A committee that reviews and supports Internet protocol proposals.

**IP (Internet Protocol)**

Part of the TCP/IP which provides for node addressing.

**IP address** An address which is used to identify a node on the Internet.

**IP multicast** Addressing technique that allows IP traffic to be propagated from one source to a group of destinations.

**IPX (Internet Packet Exchange)**

Novell NetWare communications protocol which is similar to the IP protocol. The packets include network addresses and can be routed from one network to another.

**IPX address** Station address on a Novell NetWare network. It consists of two fields: a network number field and a node number field. The node number is the station address of the device and the network number is assigned to the network when the network is started-up. It is written in the form: NNNNNNNN:XXXXXX-XXXXXX, where N's represent the network number and X's represent the station address. An example of an IPX address is: DC105333:542C10-FF1432.

**ISO** International Standards Organization

**ITU-T** The Consultative Committee for International Telephone and Telegraph (now known as the ITU-TSS) is an advisory committee established by the United Nations. They attempt to establish standards for inter-country data transmission on a worldwide basis.

**Jabber** Occurs when the transmission of network signals exceeds the maximum allowable transmission time (20 ms to 150 ms). The medium becomes overrun with data packets caused by a faulty node or wiring connection.

**Jitter** Movement of the edges of pulse over time, that may introduce error and loss of synchronization.

**JPEG** Image compression technique defined by the Joint Photographic Expert Group (JPEG), a subcommittee of the ISO/IEC. It uses a DCT, quantization, run-length and Huffman coding.

**Latency** Defines the amount of time between a device receiving data and it being forwarded on. Hubs have the lowest latency (less than 10 $\mu$ s), switches the next lowest (between 40 $\mu$ s and 60 $\mu$ s), then bridges (200 $\mu$ s to 300 $\mu$ s) and routers have the highest latency (around 1000  $\mu$ s).

**Learning bridge** Bridge which learns the connected nodes to it. It uses this information to forward or drop frames.

**Leased line** A permanent telephone line connection reserved exclusively by the leased customer. There is no need for any connection and disconnection procedures.

**Lempel-Ziv coding** Coding method which takes into account repetition in phases, words or parts of words. It uses pointers to refer to previously defined sequences.

#### **Lempel-Ziv Welsh (LZW) coding**

Coding method which takes into account repetition in phases, words or parts of words. It builds up a dictionary of previously send (or stored) sequences.

**Line driver** A device which converts an electrical signal to a form that is transmittable over a transmission line. Typically, it provides the required power, current and timing characteristics.

**Link layer** Layer 2 of the OSI model.

**Link segment** A point-to-point link terminated on either side by a repeater. Nodes can not be attached to a link segment.

#### **Lossless compression**

Where information, once uncompressed is identical to the original uncompressed data.

**Lossy compression** Where information, once uncompressed, cannot be fully recovered.

**MAC address** A 6-byte data unique data-link layer address. It is normally quoted as a 12-digit hexadecimal number (such as A5:B2:10:64:01:44).

**Masking effect** Where noise is only heard by a person when there are no other sounds to mask it.

#### **MDI (Medium Dependent Interface)**

The IEEE standard for the twisted-pair interface to 10Base-T (or 100Base-TX).

#### **Media Access Control (MAC)**

Media-specific access-control for Token Ring and Ethernet.

#### **Media Interface Controller (MIC)**

Media-specific access-control for token ring and Ethernet.

**Medium Attachment Unit (MAU)**

Method of converting digital data into a form which can be transmitted over a band-limited channel. Methods use either ASK, FSK, PSK or a mixture of ASK, FSK and PSK.

**Modem (Modulator- Demodulator)**

A device which converts binary digits into a form which can be transmitted over a speech-limited transmission channel.

**MTU (Maximum Transmission Unit)**

The largest packet that the IP protocol will send through the selected interface or segment.

**Multicast** Packets which are sent to all nodes on a subnet of a group within a network. This differs from a broadcast which forwards packet to all users on the network.

**Multimode fiber** Fiber-optic cable that has the ability to carry more than one frequency (mode) of light at a time.

**N-series connectors** Connector used with thick coaxial cable.

**Network driver interface specification (NDIS)**

Software specification for network adapter drivers. It support multiple protocols and multiple adapters, and is used in many operating systems, such as Windows 95/88/NT.

**Network layer** Third layer of the OSI model, which is responsible for ensuring that data passed to it from the transport layer is routed and delivered through the network. It provides end-to-end addressing and routing, and has support for a number of protocols, including IP, IPX, CLNP, X.25, or DDP.

**Network termination (NT1)**

Network termination for ISDN.

**Node** Any point in a network which provides communications services or where devices interconnect.

**Intranet** A company specific network which has additional security against external users.

**Octet** Same as a byte, a group of eight bits (typically used in communications terminology).

**Odd parity** An error-detection scheme where a defined bit-grouping has an odd number of 1's.

**Open Data-Link Interface (ODLI)**

Software specification for network adapter drivers using in NetWare and Apple networks. It supports multiple protocols and multiple adapters.

**Optical Repeater** A device that receives, restores, and re-times signals from one optical fiber segment to another.

**Packet** A sequence of binary digits that is transmitted as a unit in a computer network. A packet usually contains control information and data. They normally are contain with data link frames.

**Packet switching** Network switching in which data is processed in units of whole packets rather than attempting process data by dividing packets into fixed-length cells.

**Password authentication protocol (PAP)**

Protocol which checks a users password.

**Phase-Locked Loop (PLL)**

Tunes into a small range of frequencies in a signal and follows any variations in them.

**Phase-Shift Keying (PSK)**

Uses two, or more, phase-shifts to represent binary digits. Typically used to transmit binary data over speech-limited channels.

**Physical layer** Lowest layer of the OSI model which is responsible for the electrical, mechanical, and handshaking procedures over the interface that connects a device to a transmission medium

**Ping** Standard protocol used to determine if TCP/IP node are alive. Initially a node sends an ICMP (Internet Control Message Protocol) echo request packet to the remote node with the specified IP address and waits for echo response packets to return.

**Point of presence (POP)**

Physical access point to a long distance carrier interchange.

**Point-to-point protocol (PPP)**

Standard protocol to transfer data over the Internet asynchronously or synchronously.

**Port** Physical connection on a bridge or hub that connects to a network, node or other device.

<b>Protocol</b>	A specification for coding of messages exchanged between two communications processes.
<b>Quadrature modulation</b>	Technique used in PAL and NSTC where the U and V information are added to the carrier with a 90° phase difference between them.
<b>Quantization</b>	Involves converting an analogue level into a discrete quantized level. The number of bits used in the quantization process determines the number of quantization levels.
<b>Quartet signaling</b>	Signaling technique used in 100VG-AnyLAN networks that allows data transmission at 100 Mbps over frame pairs of UTP cabling.
<b>Repeater</b>	A device that receives, restores, and re-times signals from one segment of a network and passes them on to another. Both segments must have the same type of transmission medium and share the same set of protocols. A repeater cannot translate protocols.
<b>Reverse address resolution protocol (RARP)</b>	The opposite of ARP which maps an IP address to a MAC address.
<b>RJ-45</b>	Connector used with US telephones and with twisted-pair cables. It is also used in ISDN networks, hubs and switches
<b>RMON</b>	An SNMP MIB that specifies the types of information listed in a number of special MIB groups that are commonly used for traffic management. Some of the popular groups used are Statistics, History, Alarms, Hosts, Hosts Top N, Matrix, Filters, Events, and Packet Capture.
<b>Routing node</b>	A node that transmits packets between similar networks. A node that transmits packets between dissimilar networks is called a gateway.
<b>RS-232C</b>	EIA-defined standard for serial communications.
<b>RS-422, 423</b>	EIA-defined standard which uses a higher transmission rates and cable lengths than RS-232.
<b>RS-449</b>	EIA-defined standard for the interface between a DTE and DCE for 9- and 37-way D-type connectors.
<b>RS-485</b>	EIA-defined standard which is similar to RS-422 but uses a balanced connection.
<b>Run-length encoding (RLE)</b>	Coding technique which represents long runs of a certain bit sequence with a special character.

**SAP** Service Access Point. Field defined by the IEEE 802.2 specification that is part of the address specification.

**SAP** Service Advertisement Protocol, Used by the IPX protocol to provide a means of informing network clients, via routers and servers of available network resources and services.

**Segment** A segment is any length of LAN cable terminated at both ends. In a bus network, segments are electrically continuous pieces of the bus, connected at by repeaters. It can also be bounded by bridges and routers.

#### **Serial line internet protocol (SLIP)**

A standard used for the point-to-point serial connections running TCP/IP.

**Simplex** One-way communication.

#### **SNMP (Simple Network Management Protocol)**

Standard protocol for managing network devices, such as hubs, bridges, and switches.

**Source encoding** Coding method which takes into account the characteristics of the information. Typically used in motion video and still image compression.

**Statistical encoding** Coding method analyses the statistical pattern of the data. Commonly occurring data is coded with a few bits and uncommon data by a large number of bits.

**Subsampling** Reduces digitized bit rate by sampling the luminance and chrominance at different rates.

#### **Suppressing repetitive sequences**

Compression technique where long sequences of the same data is compressed with a short code.

**Switch** A very fast, low-latency, multiport bridge that is used to segment local area networks.

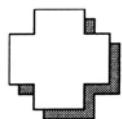
**Synchronous** Data which is synchronized by a clock.

**TCP** Part of the TCP/IP protocol and provides an error-free connection between two cooperating programs.

**TCP/IP Internet** An Internet is made up of networks of nodes that can communicate with each other using TCP/IP protocols.

<b>Telnet</b>	Standard program which allows remote users to log into a station using the TCP/IP protocol.
<b>TIFF</b>	Graphics format that supports many different types of images in a number of modes. It is supported by most packages and, in one mode provides for enhanced high-resolution images with 48-bit color.
<b>Time to live</b>	A field in the IP header which defines the number of routers that a packet is allowed to traverse before being discarded.
<b>Token</b>	A token transmits data around a token ring network.
<b>Topology</b>	The physical and logical geometry governing placement of nodes on a network.
<b>Transceiver</b>	A device that transmits and receives signals.
<b>Transform encoding</b>	Source-encoding scheme where the data is transformed by a mathematical transform in order to reduce the transmitted (or stored) data. A typical technique is the discrete cosine transform (DCT) and the fast fourier transform (FFT).
<b>Transport layer</b>	Fourth layer of the OSI model. It allows end-to-end control of transmitted data and the optimized use of network resources.
<b>Universal asynchronous receiver transmitter (UART)</b>	
	Device which converts parallel data into a serial form, which can be transmitted over a serial line, and vice-versa.
<b>V.24</b>	ITU-T-defined specification, similar to RS-232C.
<b>V.25 bis</b>	ITU-T specification describing procedures for call setup and disconnection over the DTE-DCE interface in a PSDN.
<b>V.32</b>	ITU-T standard serial communication for bi-directional data transmissions at speeds of 4.8 or 9.6 Kbps.
<b>V.34</b>	Improved v.32 specification with higher transmission rates (28.8 Kbps) and enhanced data compression.
<b>Variable-length-code LZW (VLC-LZW) code</b>	
	Uses a variation of LZW coding where variable-length codes are used to replace patterns detected in the original data.
<b>Virtual circuit</b>	Logical circuit which connects two networked devices together.

- X-ON/ X-OFF** The Transmitter On/ Transmitter Off characters are used to control the flow of information between two nodes.
- X.21** ITU-T-defined specification for the interconnection of DTEs and DCEs for synchronous communications.
- X.25** ITU-T-defined specification for packet-switched network connections.



---

## Abbreviations

---

AA	auto-answer
AAN	autonomously attached network
ABM	asynchronous balanced mode
AC	access control
ACK	acknowledge
ACL	access control list
ADC	analogue-to-digital converter
ADPCM	adaptive delta pulse code modulation
AES	audio engineering society
AFI	authority and format identifier
AM	amplitude modulation
AMI	alternative mark inversion
ANSI	American National Standard Institute
APCM	adaptive pulse code modulation
API	application program interface
ARM	asynchronous response mode
ARP	address resolution protocol
ASCII	American standard code for information exchange
ASK	amplitude-shift keying
AT	attention
ATM	asynchronous transfer mode
AUI	attachment unit interface
BCC	blind carbon copy
BCD	binary coded decimal
BIOS	basic input/output system
B-ISDN	broadband ISDN
BMP	bitmapped
BNC	British Naval Connector
BOOTP	boot protocol
BPDU	bridge protocol data units
bps	bits per second
CAD	computer-aided design
CAN	concentrated area network
CASE	common applications service elements
CATNIP	common architecture for the Internet
CCITT	International Telegraph and Telephone Consultative
CD	carrier detect
CDE	common desktop environment
CD-R	CD-recordable
CD-ROM	compact disk - read-only memory

CGI	common gateway interface
CGM	computer graphics metafile
CIF	common interface format
CMC	common mail call
CMOS	complementary MOS
CPU	central processing unit
CRC	cyclic redundancy
CRLF	carriage return, line feed
CRT	cathode ray tube
CSDN	circuit-switched data network
CSMA	carrier sense multiple access
CSMA/CA	CSMA with collision avoidance
CSMA/CD	CSMA with collision detection
CS-MUX	circuit-switched multiplexer
CSPDN	circuit-switched public data network
CTS	clear to send
DA	destination address
DAC	digital-to-analogue converter
DAC	dual attachment concentrator
DARPA	Defense Advanced Research Projects Agency
DAS	dual attachment station
DAT	digital audio tape
dB	decibel
DBF	NetBEUI frame
DC	direct current
DCC	digital compact cassette
DCD	data carrier detect
DCE	data circuit-terminating equipment
DCT	discrete cosine transform
DD	double density
DDE	dynamic data exchange
DES	data encryption standard
DHCP	dynamic host configuration program
DIB	directory information base
DISC	disconnect
DLC	data link control
DLL	dynamic link library
DM	disconnect mode
DNS	domain name server
DOS	disk operating system
DPCM	differential PCM
DPSK	differential phase-shift keying
DQDB	distributed queue dual bus
DR	dynamic range
DRAM	dynamic RAM
DSP	domain specific part
DSS	digital signature standard

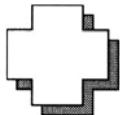
DTE	data terminal equipment
DTR	data terminal ready
EaStMAN	Edinburgh/Stirling MAN
EBCDIC	extended binary coded decimal interchange code
EBU	European broadcast union
EEPROM	electrically erasable PROM
EF	empty flag
EFM	eight-to-fourteen modulation
EGP	exterior gateway protocol
EIA	Electrical Industries Association
EISA	extended international standard interface
ENQ	inquiry
EOT	end of transmission
EPROM	erasable PROM
EPS	encapsulated postscript
ETB	end of transmitted block
ETX	end of text
FAT	file allocation table
FAX	facsimile
FC	frame control
FCS	frame check sequence
FDDI	fiber distributed data interface
FDM	frequency division multiplexing
FDX	full duplex
FEC	forward error correction
FM	frequency modulation
FRMR	frame reject
FSK	frequency-shift keying
FTP	file transfer protocol
GFI	group format identifier
GGP	gateway-gateway protocol
GIF	graphics interface format
GUI	graphical user interface
HD	high density
HDB3	high-density bipolar code no. 3
HDLC	high-level data link control
HDTV	high-definition television
HDX	half duplex
HF	high frequency
HMUX	hybrid multiplexer
HPFS	high performance file system
HTML	hypertext mark-up language
HTTP	hypertext transfer protocol
Hz	Hertz
I/O	input/output
IA5	international alphabet no. 5
IAB	internet advisory board

IAP	internet access provider
ICMP	internet control message protocol
ICP	internet connectivity provider
IDEA	international data encryption algorithm
IDI	initial domain identifier
IDP	initial domain part
IEEE	Institute of Electrical and Electronic Engineers
IEFF	internet engineering task force
IGP	interior gateway protocol
ILD	injector laser diode
IMAC	isochronous MAC
IP	internet protocol
IPP	internet presence provider
IPX	internet packet exchange
ISA	international standard interface
ISDN	integrated services digital network
IS-IS	immediate system to intermediate system
ISO	International Standards Organization
ISP	internet service provider
ITU	International Telecommunications Union
JANET	joint academic network
JFIF	jpeg file interchange format
JISC	Joint Information Systems Committee
JPEG	Joint Photographic Expert Group
LAN	local area network
LAPB	link access procedure balanced
LAPD	link access procedure
LCN	logical channel number
LD-CELP	low-delay code excited linear prediction
LED	light emitting diode
LGN	logical group number
LIP	large IPX packets
LLC	logical link control
LRC	longitudinal redundancy check
LSL	link support level
LSP	link state protocol
LZ	Lempel-Ziv
LZW	LZ-Welsh
MAC	media access control
MAN	metropolitan area network
MAU	multi-station access unit
MDCT	modified discrete cosine transform
MDI	media dependent interface
MHS	message handling service
MIC	media interface connector
MIME	multi-purpose internet mail extension
MODEM	modulation/demodulator

MOS	metal oxide semiconductor
MPEG	motion picture experts group
NAK	negative acknowledge
NCP	netware control protocols
NCSA	National Center for Supercomputer Applications
NDIS	network device interface standard
NETBEUI	NetBIOS extended user interface
NIC	network interface card
NIS	network information system
NLSP	netware link-state routing protocol
NRZI	non-return to zero with inversion
NSAP	network service access point
NSCA	National Center for Supercomputer Applications
NTE	network terminal equipment
NTFS	NT file system
NTP	network time protocol
NTSC	National Television Standards Committee
ODI	open data-link interface
OH	off-hook
OSI	open systems interconnection
OSPF	open shortest path first
OUI	originators unique identifier
PA	point of attachment
PAL	phase alternation line
PC	personal computer
PCM	pulse code modulation
PDN	public data network
PHY	physical layer protocol
PING	packet internet gopher
PISO	parallel-in-serial-out
PKP	public key partners
PLL	phase-locked loop
PLS	physical signaling
PMA	physical medium attachment
PMD	physical medium dependent
PPP	point-to-point protocol
PPSDN	public packet-switched data network
PS	postscript
PSDN	packet-switched data network
PSE	packet switched exchange
PSK	phase-shift keying
PSTN	public-switched telephone network
QAM	quadrature amplitude modulation
QCIF	quarter common interface format
QIC	quarter inch cartridge
QT	quicktime
RAID	redundant array of inexpensive disks

RAM	random-access memory
RD	receive data
REJ	reject
RFC	request for comment
RGB	red, green and blue
RI	ring in
RIP	routing information protocol
RLE	run-length encoding
RNR	receiver not ready
RO	ring out
ROM	read-only memory
RPC	remote procedure call
RR	receiver ready
RSA	Rivest, Shamir and Adleman
RTF	rich text format
RTMP	routing table maintenance protocol
S/PDIF	Sony/Philips digital interface format
SABME	set asynchronous balanced mode extended
SAC	single attachment concentrator
SAP	service advertising protocol
SAPI	service access point identifier
SAS	single attachment station
SB-ADCMP	sub-band ADPCM
SCMS	serial copy management system
SCSI	small computer systems interface
SD	sending data
SDH	synchronous digital hierarchy
SDIF	Sony digital interface
SDLC	synchronous data link control
SECAM	séquentiel couleur à mémoire
SEL	selector/extension local address
SHEFC	Scottish Higher Education Funding Council
SIPO	serial-in parallel-out
SIPP	simple internet protocol plus
SMDS	switched multi-bit data stream
SMP	symmetrical multiprocessing
SMT	station management
SMTP	simple message transport protocol
SNA	systems network architecture (IBM)
SNMP	simple network management protocol
SNR	signal-to-noise ratio
SONET	synchronous optical network
SPX	sequenced packet exchange
QTV	studio-quality television
SRAM	static RAM
STA	spanning-tree architecture
STM	synchronous transfer mode

STP	shielded twisted-pair
SVGA	super VGA
TCP	transmission control protocol
TDAC	time-division aliasing cancellation
TDM	time-division multiplexing
TEI	terminal equipment identifier
TIFF	tagged input file format
TR	transmit data
TUBA	TCP and UDP with bigger addresses
UDP	user datagram protocol
UI	unnumbered information
UNI	universal network interface
UPS	uninterruptable power supplies
URI	universal resource identifier
URL	uniform resource locator
UTP	unshielded twisted pair
UV	ultra violet
VCI	virtual circuit identifier
VCR	video cassette recorder
VGA	variable graphics adapter
VIM	vendor-independent messaging
VLC-LZW	variable-length-code LZW
VLM	virtual loadable modules
VRC	vertical redundancy check
WAIS	wide area information servers
WAN	wide area network
WIMPs	windows, icons, menus and pointers
WINS	windows internet name service
WINSOCK	windows sockets
WORM	write-once read many
WWW	World Wide Web
XDR	external data representation
XOR	exclusive-OR



---

## Miscellaneous and Quick Reference

---

### NetBIOS name types

Microsoft networks identify computers by their NetBIOS name. Each is 16 characters long, and the 16th character represents the purpose of the name. An example list of a WINS database is:

Name	Type	Status
FRED	<00>	UNIQUE Registered
BERT	<00>	UNIQUE Registered
STAFF	<1C>	GROUP Registered
STAFF	<1E>	GROUP Registered

The values for the 16th byte are:

00	Workstation	03	Message service
06	RAS server service	1B	Domain master browser
1C	Domain group name	1D	Master browsr name
1E	Normal group name (workgroup)	1F	NetDDE service
20	Server service	21	RAS client
BE	Network Monitor Agent	BF	Network Monitor Utility

On Windows NT, the names in the WINS database can be shown with the `nbstat` command.

### Windows NT TCP/IP setup

Windows NT uses the files LMHOSTS, HOSTS and NETWORKS to map TCP/IP names and network addresses. These are stored in the `<winNT root>\SYSTEM32\DRIVERS\ETC`. LMHOSTS maps IP addresses to a computer name. An example format is:

```
#IP-address      host-name
146.176.1.3    bills_pc
146.176.144.10  fred_pc  #DOM:STAFF
```

where comments have a preceding '#' symbol. To preserve compatibility with previous version of Microsoft LAN Manager, special commands have been included after the comment symbol. These include:

```
#PRE
#DOM:domain
#include fname
#BEGIN_ALTERNATE
#END_ALTERNATE
```

where

#PRE specifies the name is preloaded into the memory of the computer and no further references to the LMHOSTS file will be made.  
#DOM:*domain* specifies the name of the domain that the node belongs to.  
#BEGIN\_ALTERNATE and #END\_ALTERNATE are used to group multiple #include's  
#include *fname* specifies other LMHOST files to include.

The HOSTS file format is IP address followed by the fully qualified name (FQDN) and then any aliases. Comments have a preceding '#' symbol. For example:

#IP Address	FQDN	Aliases
146.176.1.3	superjanet	janet
146.176.144.10	hp	
146.176.145.21	mimas	
146.176.144.11	mwave	
146.176.144.13	vax	
146.176.146.23	oberon	
146.176.145.23	oberon	

### Windows NT TCP/IP commands (quick reference)

Command	Description	Examples
arp	Modifies Address Resolution Protocol tables.	arp -s 146.176.151.10 FF-AA-10-3F-A1-3F  -s <i>IP-address</i> [ <i>MAC-address</i> ] ; manually modify -a [ <i>IP-address</i> ] ; display ARP entry -d <i>IP-address</i> ; delete entry
finger	Queries users on a remote computer.	finger -l fred@miranda finger @moon  @ <i>hostname</i> ; name of remote computer -l ; extend list
ftp	Remote file transfer. After connected the following commands can be used:	ftp intel.com  ascii binary bye cd dir get hash help lcd ls mget mput open prompt pwd quit remote help user
hostname	Displays the TCP/IP hostname of the local node.	hostname
ipconfig	Displays the TCP/IP settings on the local computer.	ipconfig /all  /all ; show all settings
lpq	Sends a query to a TCP/IP host or printer.	lpq -p lp_laser lpq -s mirands -p dot_matrix  -S <i>print_server</i> -P <i>printer</i>
lpr	Prints to a TCP/IP-based printer.	lpr -p lp_laser file.ps  -S <i>print_server</i> -P <i>printer</i>

---

nbstat	Displays mapping of NetBIOS names to IP addresses.	nbstat -A freds
	<pre> -a <i>NetBIOS-name</i> ; display name table for                    ; computer -A <i>IP-address</i>   ; display name table for                    ; computer -n                ; display NetBIOS table of                    ; local computer </pre>	
netstat	Displays status of TCP/IP connections.	<i>See Section 19.4.5</i>
	<pre> -p <i>protocol</i>    ; display for given protocol -r              ; show routing tables -s              ; display statistics -R              ; reload HHOSTS -S              ; display NetBIOS sessions by                    ; NetBIOS names -s              ; display NetBIOS sessions by IP                    ; addresses </pre>	
nslookup	Queries DNS servers. After connected the following commands can be used:	<i>See Section 19.4.4</i>
	<pre> help finger [<i>username</i>] port=<i>port</i> querytype=<i>type</i> ; <i>type</i> can be A (address), ; CNAME (canonical name which is an alias for ; another host), MX (mail exchanger which ; handles mail for a given host), NS (name server ; for the domain), PTR (pointer record which ; maps an IP address to a hostname), SOA (start ; of authority record) or ANY. </pre>	
ping	Test TCP/IP connectivity.	<i>See Section 19.4.1</i>
	<pre> -a          ; resolve IP addresses to hostnames -n <i>count</i>   ; set number of echo packets -l <i>size</i>    ; specify packet size -t          ; continuously ping -i <i>ttl</i>     ; set time-to-live field -w <i>timeout</i> ; specify timeout in ms </pre>	
rcp	Remote copy.	<pre> rcp -r *.txt miranda.bill/home </pre>
	<pre> [<i>hostname</i>[.<i>username</i>]] -a          ; ASCII copy -b          ; binary copy -h          ; also hidden files -r          ; recursively copy </pre>	
rexec	Execute remote command.	<pre> rexec miranda -l bill "ls -l" </pre>

---

---

route	Manipulates TCP/IP routing table.	route gateway 146.151.176.12
	-f ; delete all routes -p ; make a permanent route add ; add a route change ; modify an existing route delete ; delete a route gateway ; specifies gateway mask <i>netmask</i> ; define subnet mask print ; print current table	
rsh	Executes remote shell.	rsh -l bill "ls -l"
	-l <i>username</i> ; user name command ; command to execute	
telnet	Remote login.	telnet www.intel.com
tftp	Trivial FTP (uses UDP).	
tracert	Trace route.	<i>See Section 19.4.6</i>
	-d ; do not resolve IP addresses -h <i>max_hops</i> ; maximum number of hops -w <i>timeout</i> ; specify timeout	

---

### Windows NT system administration commands (quick reference)

---

Command	Description	Examples
at	Runs commands at a specified time. Options include:  \computer-name time /every:date ; such as day of the week such ; as M/T/W/Th/F/Su or day ; of the month	at 14:00 \\freds "cmd ping miranda > log"  at 00:00 /every:M/W/F "cmd lpr log.txt"
attrib	Displays or changes file attributes. Attributes include:  +r, -r, (read) +a, -a, (archive) +s, -s, (system) +h, -h, (hidden) /s (include sub-directories)	attrib +h test.txt
backup	Backup program.	
cacls	Command-line Access Control Lists (ACLs).  /g <i>username:right</i> ; grant user the following ; rights: r (read), c (change), ; f (full control). /p <i>username</i> ; replace rights, these are as ; above, but n (none) is added /r <i>username</i> ; delete all rights /t ; recursive change	calcs list.txt /g fred:cf calcs *.* /r bill /t

---

chkdsk	Checks disk. Options include:  /f ; automatically fix errors	chkdsk c: /f
cmd	Run command-line shell.	
convert	Converts drive partition from FAT to NTFS,	convert d: /fs:ntfs
convlog	Converts files from Microsoft Information Server, FTP server and Gopher servers, and produces log files in NSCA or EMWAC format.  -t [emwave   ncsa] ; specify EMWAC or NCSA -s [f  w   g] ; specify FTP (f), WWW (w) or ; Gopher (g) -o <i>outdir</i> ; specify output directory	convlog -sg -ncsa -o c:\temp *.log
diskperf	Toggles the disk performance counter.	
ipxroute	IPX routing.  servers ; list NetWare servers	ipxsroute ervers
jetpack	Compacts WINS databases.	net stop wins jetpack win.mdb tmp.mdb net start wins
netmon	Network monitoring tool.	
ntbackup	Backup file system.	
rasadmin	Remote Access Server (RAS) administration.	
rasautou	Remote Access Server (RAS) debugging.	
rasdial	Remote Access Server (RAS) dial-up.  /phone <i>tel</i> ; telephone number	rasdial miranda /phone:1112222
rasphone	Edit RAS phonebook.	
rdisk	Create emergency repair disk.	
regedit	Edit registry.	
restore	Restores files after a backup.	
start	Starts applications from the command line.	
winnt	16-bit Window NT installation program.  /r <i>dir</i> ; specify install directory /s <i>dir</i> ; installation source files	
winnt32	32-bit Window NT installation program.	

**NT control services commands (quick reference)**

<i>Command</i>	<i>Description</i>	<i>Example</i>
net accounts	Controls account settings  /domain <i>dom</i> ; specify default domain	
net computer	Adds or deletes computers from current domain.  \computer-name /add; add computer /del; delete computer	net computer \\freds /add net computer \\bills /del
net config server	Configure server.	
net config workstation	Configure workstation.	
net continue	Unpauses a command that was paused with net pause.	
net file	Closes an opened file. When used on its own without arguments it gives the ID of all opened files. The /close option is used with the ID number to close a given file.  /close ; close file	
net group	Creates, edits or deletes groups.  /add ; add new group or users to the named group ; specified group /delete ; delete group or users to the named group ; specified group	net group "Staff" /add net group "Staff" /add fred
net help	Help messages for net.	
net helpmsg	Detailed help for a given error message.	
net localgroup	Create or deletes local groups or local users.	
net name	Administers list of names for the Messenger service.	
net pause	Pauses a service.	net pause lpdsvc ; pause print service
net print	Administers print queues.  \computer-name /delete ; delete job	
net send	Sends a text message to users or computers.	net send bill "Hello"
net session	Displays information of a current session.	
net share	Administers networks shares.	

---

net start	Starts a service.	net start snmp ; start SNMP
net statistics	Displays service statistics.	
net stop	Stops a service.	net stop lpdsvc ; stop print server
net time	Sets or queries time on a remote computer.	
net use	Administers networked resources.	
net user	Administers user accounts.  password ; prompts for password /active:[y/n] ; active status /add ; add user /delete ; delete user /expires:[date NEVER] ; expire time /fullname: "name" ; full name /homedir: homedirpath ; home directory /passwdchg: [y  n ] ; password change /times: [times   ALL] ; login times	net user bill_c /add net user bill_c /active:y
net view	Displays networked resources.  \computer-name /domain [domain] ; list of domain or ; computers within the ; specified domain	net view \\freds net view /domain

---

### Example Internet domain name server files

In the following example setup, there are two name servers (ees99 and eepc02) within the eece.napier.ac.uk domain. The Internet domain naming process is run with the named program and reads from the named.boot file (to use a file other than /etc/named.boot the -b option is used). Its contents are given next and it lists six main subnet (146.176.144.x to 146.176.151.x). The files net/net144 to net/net151 contain the definition of the hosts that connect to these subnets.

```
; #(named.boot 1.13      (Berkeley)  87/07/21
; boot file for secondary name server
; Note that there should be one primary entry for each SOA record.
;
directory      /usr/local/adm/named

; type      domain          source host/file      backup file
primary    eece.napier.ac.uk   eece.napier.ac.uk
primary    144.176.146.in-addr.arpa  net/net144
primary    145.176.146.in-addr.arpa  net/net145
primary    146.176.146.in-addr.arpa  net/net146
primary    147.176.146.in-addr.arpa  net/net147
primary    150.176.146.in-addr.arpa  net/net150
primary    151.176.146.in-addr.arpa  net/net151
primary    0.0.127.IN-ADDR.ARPA    named.local
primary    0.0.127.IN-ADDR.ARPA    named.local
cache     .                   root.cache
```

The first line (after the comments, which begin with a semi-colon) defines that the master file (`eece.napier.ac.uk`) contains authoritative data for the `eece.napier.ac.uk` domain. All domain names are then relative to this domain. For example, a computer within this domain which has name `pc444` will have the full domain name of:

```
pc444.eece.napier.ac.uk
```

The second line of the file defines that the file `net/net144` contains the authoritative data on the `144.176.146.in-addr.arpa` domain, and so on. The `cache` line specifies that data in the `root.cache` file is to be placed in the backup cache.

The following shows the contents of the `eece.napier.ac.uk` file on the `ees99` computer. Each master zone should begin with an SOA record for the zone. The A entry defines an address, NS defines a name server, CNAME defines an alias and MX a mail server. It can be seen that `ees99` has the `mw` alias, thus `ees99.eece.napier.ac.uk` is the same as `mw.eece.napier.ac.uk`.

```
@      IN  SOA  ees99.eece.napier.ac.uk.  mike.ees99.eece.napier.ac.uk.
          199806171      ; Serial
          10800        ; Refresh every 3 hrs
          1800         ; Retry every 1/2 hr
          604800       ; Expire (seconds)
          259200 )    ; Minimum time-to-live
      IN  NS  ees99.eece.napier.ac.uk.
      IN  NS  eepc02.eece.napier.ac.uk.
localhost IN  A   127.0.0.1
@      IN  MX  11   146.176.151.139.
hp350   IN  A   146.176.144.10
mwave   IN  A   146.176.144.11
hplb69  IN  CNAME mwave
vax     IN  A   146.176.144.13
miranda IN  A   146.176.144.14
triton   IN  A   146.176.144.20
mimas   IN  A   146.176.146.21
ees99   IN  A   146.176.151.99
mw      IN  CNAME ees99
```

The SOA lists a serial number, which has to be increased each time the master file is changed. This is because secondary servers check the serial number at an interval specified by the refresh time (Refresh). If the serial number increases then a zone transfer is done to load the new data. If the master server cannot be contacted when a refresh is due then the retry time (Retry) specifies the interval at which refreshes occur. If the master server cannot be contacted within the expire time interval (Expire) then all data from the zone is discarded by secondary servers.

The following file lists some of the contents of the `net/net151` file. The NS entry defines the two name servers (`ees99` and `eepc02`) and the PTR entry maps an IP address to a domain name.

```
@      IN  SOA  ees99.eece.napier.ac.uk.  mike.ees99.eece.napier.ac.uk. (
          199706091      ; Serial
          10800        ; Refresh
          1800         ; Retry
          604800       ; Expire
          259200 )    ; Minimum
      IN  NS  ees99.eece.napier.ac.uk.
      IN  NS  eepc02.eece.napier.ac.uk.
```

```

50      IN    PTR    ee50.eece.napier.ac.uk.
61      IN    PTR    eepc01.eece.napier.ac.uk.
62      IN    PTR    eepc02.eece.napier.ac.uk.
222     IN    PTR    pctest.eece.napier.ac.uk.
2       IN    PTR    pc345.eece.napier.ac.uk.
3       IN    PTR    pc307.eece.napier.ac.uk.
4       IN    PTR    pc320.eece.napier.ac.uk.
5       IN    PTR    pc331.eece.napier.ac.uk.
6       IN    PTR    pc401.eece.napier.ac.uk.
7       IN    PTR    pc404.eece.napier.ac.uk.

```

and for the net/net151 file:

```

@      IN    SOA    ees99.eece.napier.ac.uk. mike.ees99.eece.napier.ac.uk. (
                    199806171      ; Serial
                    10800        ; Refresh
                    1800         ; Retry
                    604800       ; Expire
                    259200 )     ; Minimum
      IN    NS    ees99.eece.napier.ac.uk.
      IN    NS    eepc02.eece.napier.ac.uk.
10     IN    PTR    ees10.eece.napier.ac.uk.
11     IN    PTR    ees11.eece.napier.ac.uk.
12     IN    PTR    ees12.eece.napier.ac.uk.
13     IN    PTR    ees13.eece.napier.ac.uk.
14     IN    PTR    ees14.eece.napier.ac.uk.
15     IN    PTR    ees15.eece.napier.ac.uk.

```

The entries take the form:

*<domain>*    *<op\_ttl>*    *<opt\_class>*    *<type>*    *<resource\_record\_data>*

where

*domain* either a . which defines the root domain, a @ which defines the current origin, or a standard domain name. If it is a standard domain and it does not end with a ., then current origin is appended to the domain, else the domain names unmodified.

*op\_ttl* is an optional number for the time to live.

*opt\_class* is the object address type. This can be IN (for DARPA Internet) or HS (for Hesiod class).

*resource\_record* This contains one of the following definitions:

- A (address).
- CNAME (canonical name which is an alias for another host).
- GID (group ID).
- HINFO (host information)
- MB (mailbox domain name).
- MG (mailbox domain name).
- MIFO (mailbox or mail list information).
- MR (mail rename domain name).
- MX (mail exchanger which handles mail for a given host).
- NS (name server for the domain).
- PTR (pointer record which maps an IP address to a hostname).
- SOA (start of authority record). The domain of originating host,

domain address of maintainer, a serial and the other parameters (refresh, retry time, expire time and minimum TTL) are also defined.

- TXT (text string).
- UID (user information).
- WKS (well known service). This defines an IP address followed by a list of services.

The contents of the `sobasefile` is given next. This defines the two domain name servers.

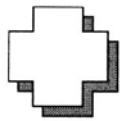
```
@      IN      SOA    ees99.eece.napier.ac.uk. mike.ees99.eece.napier.ac.uk. (
                    19970307      ; Serial
                    10800        ; Refresh
                    1800         ; Retry
                    604800       ; Expire
                    259200 )     ; Minimum
IN      NS      ees99.eece.napier.ac.uk.
IN      NS      eepc02.eece.napier.ac.uk.
```

## TCP/IP services

<i>Port</i>	<i>Protocol</i>	<i>Service</i>	<i>Comment</i>
1	TCP	TCPmux	
7	TCP/UDP	echo	
9	TCP/UDP	discard	Null
11	TCP	systat	Users
13	TCP/UDP	daytime	
15	TCP	netstat	
17	TCP	qotd	Quote
18	TCP/UDP	msp	Message send protocol
19	TCP/UDP	chargen	ttytst source
21	TCP	ftp	
23	TCP	telnet	
25	TCP	smtp	Mail
37	TCP/UDP	time	Timserver
39	UDP	rlp	Resource location
42	TCP	nameserver	IEN 116
43	TCP	whois	Nicname
53	TCP/UDP	domain	Domain name server
57	TCP	mtp	Deprecated
67	TCP	bootps	BOOTP server
67	UDP	bootps	
68	TCP/UDP	bootpc	BOOTP client
69	UDP	tftp	
70	TCP/UDP	gopher	Internet Gopher
77	TCP	rje	Netrjs
79	TCP	finger	
80	TCP/UDP	www	WWW HTTP
87	TCP	link	Ttylink
88	TCP/UDP	kerberos	Kerberos v5
95	TCP	supdup	
101	TCP	hostnames	
102	TCP	iso-tsap	ISODE.
105	TCP/UDP	csnet-ns	CSO name server

107	TCP/UDP	rtelnet	Remote Telnet
109	TCP/UDP	pop2	POP version 2
110	TCP/UDP	pop3	POP version 3
111	TCP/UDP	sunrpc	
113	TCP	auth	Rap identity authentication
115	TCP	sftp	
117	TCP	uucp-path	
119	TCP	nntp	USENET News Transfer Protocol
123	TCP/UDP	ntp	Network Time Protocol
137	TCP/UDP	netbios-ns	NETBIOS Name Service
138	TCP/UDP	netbios-dgm	NETBIOS Datagram Service
139	TCP/UDP	netbios-ssn	NETBIOS session service
143	TCP/UDP	imap2	Interim Mail Access Protocol Ver2
161	UDP	snmp	Simple Net Management Protocol
162	UDP	snmp-trap	SNMP trap
163	TCP/UDP	cmip-man	ISO management over IP (CMOT)
164	TCP/UDP	cmip-agent	
177	TCP/UDP	xdmcp	X Display Manager
178	TCP/UDP	nextstep	NeXTStep NextStep
179	TCP/UDP	bgp	BGP
191	TCP/UDP	prospero	
194	TCP/UDP	irc	Internet Relay Chat
199	TCP/UDP	smux	SNMP Unix Multiplexer
201	TCP/UDP	at-rtmp	AppleTalk routing
202	TCP/UDP	at-nbp	AppleTalk name binding
204	TCP/UDP	at-echo	AppleTalk echo
206	TCP/UDP	at-zis	AppleTalk zone information
210	TCP/UDP	z3950	NISO Z39.50 database
213	TCP/UDP	ipx	IPX
220	TCP/UDP	imap3	Interactive Mail Access
372	TCP/UDP	ulistserv	UNIX Listserv
512	TCP/UDP	exec	Comsat
513	TCP	login	
513	UDP	who	Whod
514	TCP	shell	No passwords used
514	UDP	syslog	
515	TCP	printer	Line printer spooler
517	UDP	talk	
518	UDP	ntalk	
520	UDP	route	RIP
525	UDP	timed	Timeserver
526	TCP	tempo	Newdate
530	TCP	courier	Rpc
531	TCP	conference	Chat
532	TCP	netnews	Readnews
533	UDP	netwall	Emergency broadcasts
540	TCP	uucp	Uucp daemon
543	TCP	klogin	Kerberized ‘rlogin’ (v5)
544	TCP	kshell	Kerberized ‘rsh’ (v5)
556	TCP	remotefs	Brunhoff remote filesystem
749	TCP	kerberos-admin	Kerberos ‘kadmin’ (v5)
750	UDP	#kerberos	Kerberos (server) UDP
750	TCP	#kerberos	Kerberos (server) TCP

760	TCP	krbupdate	Kerberos registration
761	TCP	kpasswd	Kerberos "passwd"
765	TCP	webster	Network dictionary
871	TCP	supfilesrv	SUP server
1127	TCP	supfiledbg	SUP debugging
1524	TCP	ingreslock	
1524	UDP	ingreslock	
1525	TCP	prospero-np	Prospero non-privileged
1525	UDP	prospero-np	
2105	TCP	eklogin	Kerberos encrypted rlogin
5002	TCP	rfe	Radio Free Ethernet



## Additional Material

### P.1 PGP

PGP (Pretty Good Privacy) uses the RSA algorithm with a 128-bit key. It was developed by Phil Zimmermann and gives encryption, authentication, digital signatures and compression. Its source code is freely available over the Internet and its usage is also free of charge, but it has encountered two main problems:

- The source code is freely available on the Internet causing the US government to claim that it violates laws which relate to the export of munitions. Current versions have since been produced outside of the US to overcome this problem.
- It uses algorithms which have patents, such as RSA, IDEA and MD5.

Figure P.1 shows the basic encryption process.

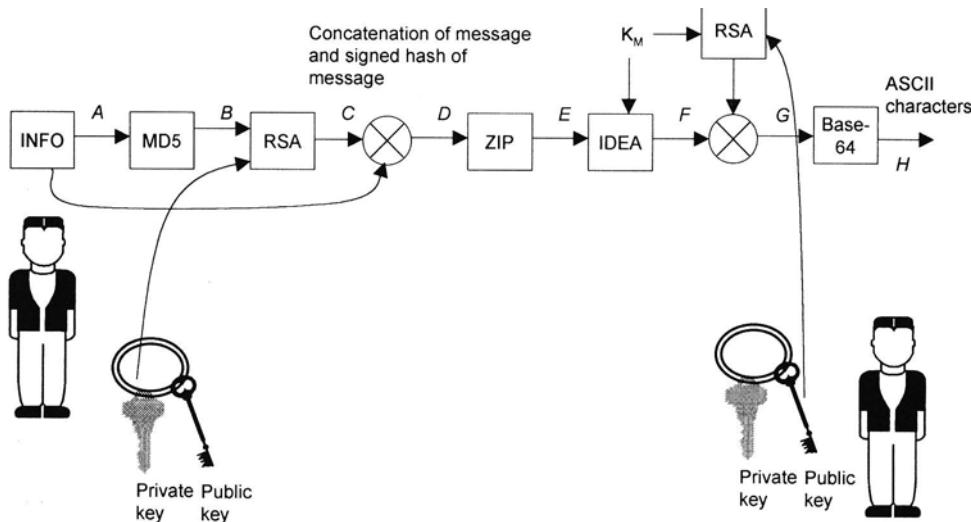


Figure P.1 PGP encryption.

The steps taken are:

- A. Sender hashes the information using the MD5 algorithm.
- B. Hashed message is encrypted using RSA with the sender's private key (this is used to authenticate the sender as the senders public key will be used to decrypt the message).

- C. Encrypted message is then concatenated with the original message.
- D. Message is compressed using LZ compression.
- E. A 128-bit IDEA key ( $K_M$ ) is generated by some random input, such as the content of the message and the typing speed.
- F.  $K_M$  is then used with the IDEA encryption.  $K_M$  is also encrypted with the receiver's public key.
- G. Output from IDEA encryption and the encrypted  $K_M$  key are concatenated together.
- H. Output is encoded as ASCII characters using Base-64 (See Section 21.7.4 on Electronic Mail).

To decrypt the message the receiver goes through the following steps:

- A. Receiver reverses the Base-64 conversion.
- B. The receiver decrypts the  $K_M$  key using their own private RSA key.
- C. The  $K_M$  key is then used with the IDEA algorithm to decode the message.
- D. The message is then decompressed using an UNZIP program.
- E. The two fragments produced after the uncompression will be the plaintext message and an MD5/RSA encrypted message. The plaintext message is the original message, where the MD5/RSA encrypted message can be used to authenticate the sender. This is done by applying the sender's public key to the uncompressed encrypted part of the message. This should produce the original plaintext message.

PGP allows for three RSA key sizes. There are:

- 384 bits. This is intended for the casual user and can be cracked by serious crackers.
- 512 bits. This is intended for the commercial user and can only be cracked by organizations with a large budget and extensive computing facilities.
- 1024 bits. This is intended for military uses and, at present, cannot be cracked by anyone. This is the recommended key size for most users of reasonably powerful computers (386/486/Pentium/etc). In the future, a 2048-bit code may be used.

## P.2 NetWare 4 and NDS

---

### P.2.1 Introduction

The main disadvantages of NetWare 3.x are:

- It uses SPX/IPX which is incompatible with TCP/IP traffic.
- It is difficult to synchronize servers with user information.
- The file structure is local to individual servers.
- Server architecture is flat and cannot be organized into a hierarchical structure.

These have been addressed with NetWare 4.1, in which the bindery has been replaced by Novell Directory Services (NDS). NDS is a combination of features from OSI X.500 and Banyan StreetTalk. Its main characteristics are:

- Hierarchical server structure.
- Network-wide users and groups.
- Global objects. NDS integrates users, groups, printers, servers, volumes and other physical resources into a hierarchical tree structure.
- System-wide login with a single password. This allows users to access resources which are connected to remote servers.
- NDS processes logins between NetWare 3.1 and NetWare 4.1 servers, if the login names and passwords are the same.
- Supports distributed file system.
- Synchronization services. NDS allows for directory synchronization, which allows directories to be mirrored on different partitions or different servers. This provides increased reliability in that if a server develops a fault then the files on that server can be replicated by another server.
- Standardized organizational structure for applications, printers, servers and services. This provides a common structure across different organizations.
- It integrates most of the administrative task in Windows-based NWADMIN.EXE program.
- It is a truly distributed system where the directory information can be distributed around the tree.
- Unlimited number of licenses per server. NetWare 3.1 limits the number of licenses to 250 per server.
- Support for NFS server for UNIX resources.
- Multiple login scripts, as opposed to system and user login scripts in NetWare 3.1.
- Windows NT support.

NDS is basically a common, distributed Directory database of logical and physical resources into a single information system. Many other applications have used Directory databases, such as electronic mail and network management. NDS servers within a network access the Directory database for the connected resources and how they are accessed. Thus application programs do not need to know the physical location and on which server it is connected, only its logical name.

The main reason to upgrade to NDS is that it better reflects the organizational structure of networked equipment within the organization. NetWare 3.1 is a server-based approach where resources are grouped around servers. This leads to increased maintenance around these servers, thus updates to one server may have to be updated on other servers. NDS allows for a central administration with a structure that reflects organizational structures.

#### *P.2.2 NetWare directory services (NDS)*

One of the major changes between NetWare 3.x and NetWare 4.1 is NDS. A major drawback of the NetWare 3.x bindery files is that they were independently maintained on each server. NDS addresses this by setting up a single logical database, which contains information on all network-attached resources. It is logically a single database, but may be physically located on different servers over the network. As the database is global to the network, a user can log in to all authorized network-attached resources, rather than requiring to login into each separate server. Thus, administration is focused on the single database.

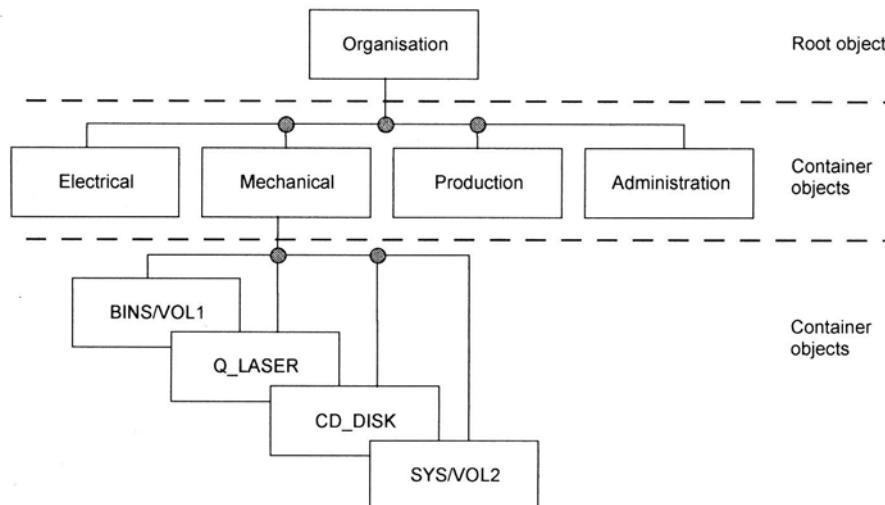
As with NetWare 3.x bindery services, NDS organizes network resources by objects,

properties, and values. NDS differs from the bindery services in that it defines two types of object:

- Leaf objects – which are network resources such as disk volumes, printers, printer queues, and so on.
- Container objects – which are cascadable organization units that contain leaf objects. A typical organizational unit might be company, department or group.

NDS organizes networked resources in a hierarchical or tree structure (as most organizations are structured in this way). The top of the tree is the root object, to which there is only a single root for an entire global NDS database. Servers then use container objects to connect to branches coming off the root object. This structure is similar to the organization of a directory file structure and can be used to represent the hierarchical structure of an organization. Figure P.2 illustrates a sample NDS database with root, container and leaf objects. In this case, the organization splits into four main containers: Electrical, Mechanical, Production and Administration. Each of these containers has associated leaf objects, such as disk volumes, printer queues, and so on. This is a similar approach to Workgroups in Microsoft Windows.

To improve fault tolerance, NDS allows branches of the tree (or partitions) to be stored on multiple file servers. These mirrors are then synchronized to keep them up-to-date. Another advantage of replicating partitions is that local copies of files can be stored so that network traffic is reduced.



**Figure P.2** NDS structure.

The container objects are:



[ROOT]. This is the top level of the inverted tree and contains all the objects within the organizational structure.



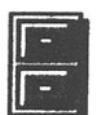
Organization. This object class defines the organizational name (such as FRED\_ AND\_ CO). It is normally the next level after [ROOT] (or below the C=Country object).



User. This object defines an individual user. The first user created in a NetWare 4 system is the ADMIN user, which is typically the only user with rights to add and delete objects on the whole of the NDS structure.



NCP (NetWare Control Protocol) Server. This appears for all NetWare 4 servers.



Volume. This identifies the mounted volume for file services. A network file system data links to the Directory tree through Volume objects.

The most commonly used objects are:



Bindery. These allow compatibility with existing Bindery-based NetWare 3, NetWare 3 clients and NetWare 4 servers which do not completely implement NDS. They display objects that isn't a user, group, queue, profile or print server, which was created using the bindery services.



Organizational unit. This object represents the OU part of the NDS tree. These divide the NDS tree into subdivisions, which can represent different geographical site, different divisions or workgroups. Different division might be PRODUCTION, ACCOUNT, RESEARCH, and so on. Each Organizational Unit has its own login script.



Organization role. This object represents a defined role within an organization object. It is thus easy to identify users who have an administrative role with the organization.



Group. This object represents a grouping of users. All users within a group inherit the same access rights.



Directory map. This object points to a file system directory on a mounted volume. It is typically used to create a global file system which has physically separate parts.



Alias. This identifies an object with another name. For example, a print queue which is called NET\_PRINT1 might an alias name of HP\_LASER\_JET\_6.



Printer. This can either be connected to the printer port of a PC, or connected to a NetWare server.



Print queue. This object represents the queue of print jobs.



Profile. This object defines a special scripting file. This can be a global login script, a location login script or a special login script.



Print server. This object allows print jobs to be queued, waiting to be serviced by the associated printer.

### P.2.3 NDS tree

Figure P.3 shows the top levels of the NDS tree. These are:

- [ROOT]. This is the top level of the tree. The top of the NDS tree is the [ROOT] object.
- C=Country. This object can be used, or not, to represent different countries, typically where an organization is distributed over two or more countries. If it is used then it must be placed below the [ROOT] object. NDS normally does not use the Country object and uses the Organization Unit to define the geographically located sites, such as SALES\_UK.[ROOT], SALES\_USA.[ROOT], and so on.
- L=Locality. This object defines locations within other objects, and identifies network portions. The Country and Locality objects are included in the X.500 specification, but they are not normally used, because many NetWare 4 utilities do not recognize it. When used, it must be placed between the [Root] object, Country object, Organization object, or Organizational Unit object.
- LP=Licensed Product. This object is automatically created when a license certificate is installed. When used, it must be placed between the [Root] object, Country object, Organization object, or Organizational Unit object.
- O=Organization. This object represents the name of the organization, a company division or a department. Each NDS Directory tree has at least one Organization object, and it must be placed below the [Root] object (unless the tree uses the Country or Locality object).
- OU=Organization Unit. This object normally represents the name of the organizational unit within the organization, such as Production, Accounts, and so on. At this level, User objects can be added and a system level login script is created. It is normally placed below the Organization object.

The structure of the NDS should reflect the organization of the company, for its organizational structure, its locations and the organization of its networks. Normally there is only one Organization object as this makes it easier to merge the NDS tree with other organizations. With every Organization object, there are normally several Organization Units.

Apart from the container objects (C, O, OU, and so on) there are leaf objects. These are assigned a CN (for Common Name). They include:

CN=AFP Server	CN=Bindery	CN=Bindery Queue
CN=Computer	CN=Directory Map	CN=Group
CN=Organizational Role	CN=Print Queue	CN=Print Server
CN=Printer	CN=Profile	CN=Server
CN=User	CN=Volume	

If possible, the NDS tree depth should have between four and eight levels. This makes management easier and allows resources to be easily accessed.

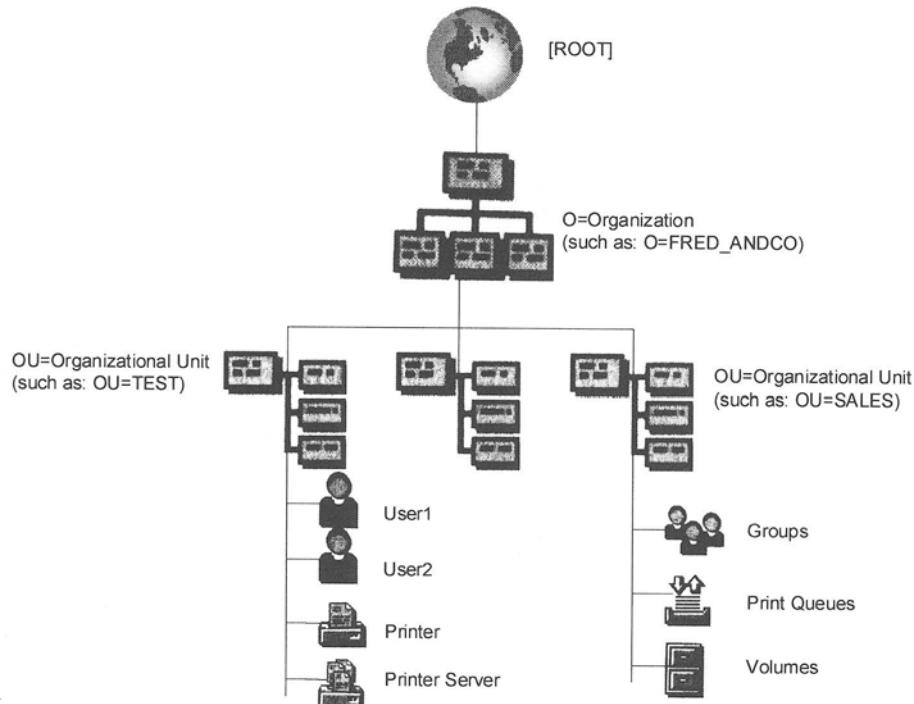


Figure P.3 NDS structure.

#### P.2.4 Typical naming syntax

The NDS tree can use many different naming formats, but a standardized naming structure has been developed. These are:

	Syntax	Example
[ROOT]	<i>company_TREE</i>	FRED_TREE
Organization	<i>company_name</i>	O=FRED
Organization Units	<i>location (or department)</i>	OU=SALES
Servers	<i>location-department-SRV#</i>	SALES-SRV1
Printer Servers	<i>location-department-PS#</i>	SALES-LZ5-PS3
Printers	<i>printer-P#</i>	HPLJ5-P2
Print Queues	<i>type-P#</i>	HPLJ5-P2
Volumes	<i>server_volume</i>	SALES-SRV1_DATA

### P.2.5 Object names

The place at which an object is placed is called its context. Two objects which are placed in the same container have the same context. For example, if the user FRED\_B works for the Fred & Co. (O=FRED\_AND\_CO), within the Test Department (OU=TEST), which is within the Engineering Unit (OU=ENGINEERING) then his context will be:

```
OU=TEST.OU=ENGINEERING.O=FRED_AND_CO
```

An object is either identified by its distinguishing name (such as LP\_LASER5) or by its complete name (CN). In the name, periods separate the objects (these periods are similar to back slashes or forward slashes, which is common in many operating systems). For a complete name, which is referred to from the [ROOT] object, a leading period is used. Whereas, a relative name does not have a leading period. For example, a complete name for a User object FRED\_B could be:

```
.CN=FRED_B.OU=TEST.OU=ENGINEERING.O=FRED_AND_CO
```

This defines a User, which has an Organization of FRED\_AND\_CO, which has an Organization Unit called ENGINEERING, there is then a subdivision below this called TEST. It is also possible to define a relative distinguishing name (RDN) which defines the relative path with respect to the current context.

Periods can be added to the start or the end of the context. They have the following definitions:

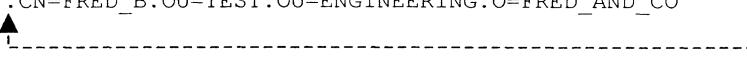
- Leading period. NDS ignores the current context of the object and resolves the name at the [ROOT] object.
- Trailing period. NDS selects a new context when resolving an object's complete name at the [ROOT] object.

For example, the partial name for the User object FRED\_B relative to other objects in OU=TEST would be:

```
.CN=FRED_B.
```

The partial name of the User object FRED\_B that has a complete name of:

```
.CN=FRED_B.OU=TEST.OU=ENGINEERING.O=FRED_AND_CO
```

  
relative to a server object with a complete name of:

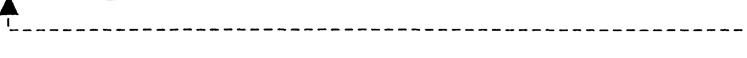
*Distinguishing  
name. Resolve from  
[ROOT]*

```
.CN=OU=SALES-SRV1.OU=SALES.O=FRED_AND_CO
```

is:

  
*Move up one  
level*

```
CN=FRED_B.OU=TEST.OU=ENGINEERING.
```

  
*Relative name.*

The HPLJ5-P2 printer object which has the complete name of:

.CN=HPLJ5-P2.OU=TEST.OU=ENGINEERING.O=FRED\_AND\_CO

would be referred, within the OU=TEST.OU=ENGINEERING.O=FRED\_AND\_CO container, as:

CN=HPLJ5-P2

### Typeless name

Notice that a relative name has a trailing period to identify that it is a partial name. It is also possible not to include the object types (such as CN for common name, OU for Organizational Unit and O for Organization). This is called a typeless name, and NDS makes a guess as to the object types. For example:

FRED\_B.TEST.ENGINEERING.FRED\_AND\_CO

is the same as one of the previous examples. When guessing NDS uses the following rules:

- The object which is furthest to the left is assumed to be a common name (leaf object).
- The object which is furthest to the right is assumed to be the organization (container object).
- All other objects are assumed to be Organizational Units (container objects).

### P.2.6 CX

The CX (Change conteXt) command is used to display or modify the context, or to view containers and leaf objects in the Directory tree. In a Command Prompt window, the following can be used:

<i>Command</i>	<i>Description</i>
CX	displays current context
CX /?	display help manual
CX /CONT	display all containers in the current context
CX /T	display all containers at and below the current context
CX .	move up one level
CX ..	move up two levels
CX /CONT	display containers in the [ROOT]
CX <i>content</i>	display context for <i>content</i>
CX /R	change current context to [ROOT]
CX /A	display all containers and objects in the current context.
CX /R /A /T	display all containers and objects, from the [ROOT] down.

For example to set the current context to the TEST.ENGINEERING.FRED\_AND\_CO container:

CX TEST.ENGINEERING.FRED\_AND\_CO

Then to change the context to ENGINEERING:

CX ENGINEERING.FRED\_AND\_CO

or

CX .

### P.2.7 Startup files and scripts

Much of the initialization of a client is done with start-up files and scripts. The main startup files are:

- CONFIG.SYS and AUTOEXEC.BAT. These are standard start-up files for the PC and normally setup the environment of the computer. The AUTOEXEC.BAT file should include the STARTNET.BAT file.
- STARTNET.BAT. Provides a network connection.
- NET.CFG. Customizes the NetWare set up, such as setting ODI and VLM settings.

The login scripts are:

- Container Login Scripts. These set up the Organization and Organizational Unit properties, and generally replace System login scripts.
- Profile Login Scripts. These set up the environment of User groups.
- User Login Scripts. These customize the User environment. If no User login script exists then a Default Login Script is executed.

The NET.CFG file is similar to NetWare 3 but has extra lines to define the NetWare 4 options. An example file is:

```
Link Driver NE2000
    Int #1 11
    Port #1 320
    Frame Ethernet_II
    Frame Ethernet_802.3
    Protocol IPX 0 Ethernet_802.3

NetWare DOS Requester
    NAME CONTEXT= "OU=electrical.OU=engineering.O=napier"
    PREFERRED SERVER = EEE-SRV1
    FIRST NETWORK DRIVE = G
    NETWARE PROTOCOL = NDS, BIND
```

This defines that the name context for the user (with NAME CONTEXT) and that the preferred server is EEE-SRV1. The first network drive will be G: and the NetWare protocol is NDS and Bindery.

Drive disks can be mounted by adding lines to the Login Script (such as NETSTART.BAT, which is started from the AUTOEXEC.BAT file). For example, to mount the F:, G: and M: drives then the following could be added:

```
MAP ROOT F:= .EEE-SRV1.ENGINEERING.NAPIER\SYS:APPS
MAP ROOT G:= .CRAIGLOCKHART_1.MAJOR.NAPIER.AC.UK\SYS:APPS
MAP ROOT M:= .CRAIGLOCKHART_3.MAJOR.NAPIER.AC.UK\SYS:MAIL
```

### P.2.8 Volume mapping

Volumes can be mounted as drives using the syntax:

**MAP *drive\_letter*:=CN=*servername\_volumename.context*:**

For example, to map the DATA volume of the TEST server to drive letter F: then the following is used:

```
MAP F:=CN=TEST_DATA.OU=TEST.:
```

### **P.2.9 Country Object**

The country object is commonly not used as it fixes the geographical location of objects. It has the advantage, though, is that it fits into a common Internet naming structure (such as, www.eece.napier.ac.uk) or X.500 names. Most network though have the Organizational Unit following the [ROOT] level. For example and an educational organization in the UK will have a country object of UK and the organization object of AC (as defined in the Internet name). The Organization Unit would then be the name of the academic organization (in this case, Napier). Next, the facilities and departments are defined, as follows:

```
[Root]
c=uk
o=ac
ou=napier
ou=Arts
ou=Business
ou=Engineering
ou=electrical
ou=mechanical
ou=computing
```

Thus, the context name for a printer (LJET5) in the Electrical department would be:

```
CN=LJET5.OU=ELECTRICAL.OU=ENGINEERING.O=NAPIER.C=UK
```

This is obviously similar to the Internet name for the device, which would be:

```
ljet5.electrical.engineering.napier.ac.uk
```

### **P.2.10 User class**

NDS has an object-oriented database, where each object (such as Users, Printers, and so on) has associated properties. The User object has the following properties:

- Login Name. Normally the first character of the first name followed by the last name, such as BBuchanan (for Bill Buchanan). [Required]
- Given Name. Users first name. [Required]
- Last Name. Users last name. [Required]
- Full Name. Users full name. [Required]
- Generational Qualification. [Optional]
- Middle Initial. [Required]
- Other Name. [Optional]
- Title. Job title. [Required]

- Description. [Optional]
- Location. City or location. [Required]
- Department. [Optional]
- Telephone Number. Full telephone number [Required]
- Fax Number. Full Fax number [Required]
- Language. Spoken language [Optional]
- Network address. [System adds this]
- Default server. Server that the user initially logs into [Optional]
- Home Directory. Volume:subdirectory\user, such as DATA:HOME\BILL\_B [Required]
- Required Password. Force password, or not. [Required]
- Account Balance. [Optional]
- Login Script. [Optional]
- Print Job Configuration. [Optional]
- Post Office Box. [Optional]
- Street. [Optional]
- City. [Optional]
- Start or Province. [Optional]
- Zip Code. [Optional]
- See also. [Optional]

#### **P.2.11 Bindery services**

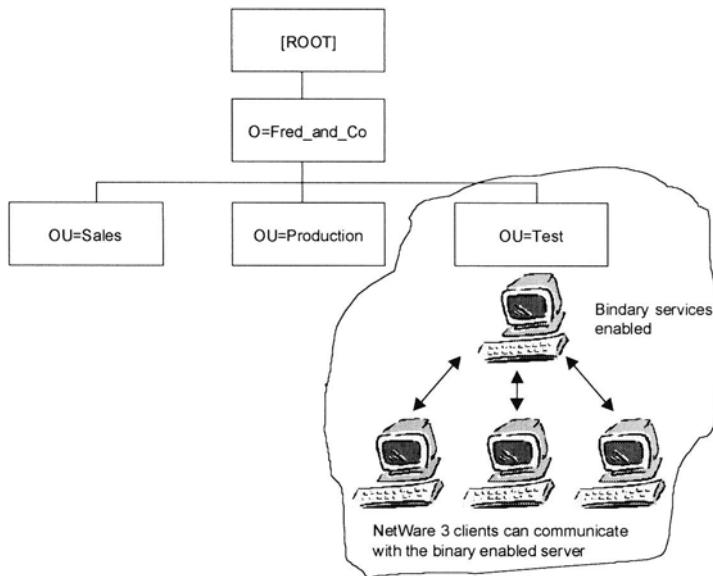
In NetWare 4.1 the bindery services have been replaced by NetWare Directory Services (NDS). Many networks take some time to be fully upgrade from bindery to NDS, thus NetWare 4.1 supports bindery. This allows a NetWare 3 server to be upgraded to NetWare 4.1, but still run a bindery. It also allows NetWare 4 to integrate with NetWare 3 servers (typically, which run print services).

NetWare 4 servers support the following bindery-based resources:

- Bindery objects class.
- Bindery programs.
- Bindery-based NetWare client software.
- Groups.
- Print servers.
- Profiles.
- Queues.
- Users.

On a bindery service, NDS supports a flat structure for leaf objects in an Organization or Organizational Unit object. All objects within the specified container can then be accessed by NDS objects and by bindery-based servers and client workstations. On a bindery enabled server, a client gets its login script from that server. The login and script are not automatically transmitted to other servers (as they would with NDS).

Figure P.4 shows an example of a bindery-enabled server which is an Organizational Unit object (OU).



**Figure P.4 Bindery services.**

### P.2.12 Time synchronization

Time synchronization between servers is important as it allows NDS events and modifications to be accurately time stamped. There are two main options:

- Single reference configuration. This provides a single source of time reference and is typically used with networks with less than thirty servers.
- Time provider group configuration. This provides a single reference primary server and, at least, two other primary time providers. It is typically used when there are more than thirty servers connected to the network. The other primary time providers allow for a system failure of the reference primary server.

The time synchronization is provided with:

- SAP. With SAP the SAP type is 0000 0010 0110 1011 (026Bh). The main disadvantage with this method is that SAP generally adds to network traffic and, as SAP is self configuring, an incorrectly set up can cause incorrect time to be transmitted.
- Configured List Communication. This method uses allows each server to keep a list of server which it can communicate with. This will generally lead to less network traffic as it does not use an SAP broadcast. It also stops incorrectly set up servers from transmitting incorrect time information (as the server will only communicate with preferred time servers).

NetWare uses the TIMESYNC.NLM (NetWare Loadable Module) module to synchronize their local time. The server then calculates Universal Coordinated Time (UTC) which provides a world standard time. UTC (Co-ordinated Universal Time) is a machine-independent

time standard. It assumes that there are 86 400 seconds each day ( $24 \times 60 \times 60$ ) and once every year or two an extra second is added (a “leap” second). This is normally added on 31<sup>st</sup> December or 30<sup>th</sup> June.

Most computer systems define time with GMT (Greenwich Mean Time), which is UT (Universal Time). UTC is based on an atomic clock, whereas GMT is based on astronomical observations. Unfortunately, because the earth rotation GMT is not uniform, and is not as accurate as UTC. UTC is calculated by:

$$\text{UTC} = \text{LOCAL TIME} + \text{timezone\_offset} + \text{current\_daylight\_adjustment}$$

With TIMESYNC.NLM the system time is not actually changed, the local clock is either speeded up (if the time is behind) or slowed down (if it is ahead). This makes for gradual changes in the system time. This is especially important for server synchronization where directories and files are kept up-to-date between servers. An incorrectly set time on one server could cause an older file to replace a newer file. Every NDS object and their associated properties have a timestamp associated with them.

### NDS timestamp

Particular problems caused with time on computer systems are the Year 2000 bug (where dates are referenced to just the last two digits of the year) and where there is a roll-over in the counter value which stores the system time. The Year 2000 bug can be easily eradicated by making sure that all references to time take into account the full year format.

The PC contains a 32-bit counter which is updated every second and is reference to the 1<sup>st</sup> January 1970 (the starting date for the PC). This provides for 4,294,967,296 seconds (715,827,882 minutes, 11,930,465 hours, 497,103 days and 1361 years). The format of the NDS timestamp uses this format and adds other fields to define the place the event occurred and an indication of events that occur within a single second. It uses 64 bits and its format is:

- Seconds (32 bits). This stores the number of seconds since 1/1/1970. This allows for 4 billion seconds, which is approximately 1371 years, before a roll-over occurs.
- Replica Number (16 bits). This is a unique number which defines where the event occurred and the timestamp issued.
- Event ID (16 bits). Defines each event that occurs within a second a different Event ID. This is requires as many events can occur within a single second. This value is reset on every second, and thus allows up to 65,536 events each second.

NDS always uses the most recently time stamped object or properties for any updates. When an object is deleted its property is marked as “not present”. It will only be deleted once the replica synchronization process propagates the change to all other replicas.

### Time server types

NetWare 4.1 servers are set up as time servers when they are installed. They can either be:

- Primary time servers. A primary time server provides time information to others, but must contact at least primary (or reference) server for their own time.
- Secondary time servers. These are time consumers, which receive their time from other

servers (such as from a primary, reference or single reference time server).

- Reference time servers. These servers do not need to contact any other servers, and provide a time source for other primary time servers. This is a good option where there is a large network, as the primary time servers can provide local time information (this is called a time provider group).
- Single reference time servers. These servers do not need to contact other time servers to get their own time and are used as a single source of time. This is normally used in a small network, where there is a single reference time server with one or more secondary time servers. The single reference time server and reference time server normally get their local time information from another source, such as Internet time, radio or satellite time. This is the default condition for installation.

## **P.3 Windows NT**

---

### **P.3.1 Administrative tools**

Windows NT has a whole host of administrative tools. Figure P.5 shows the available tools. These include:

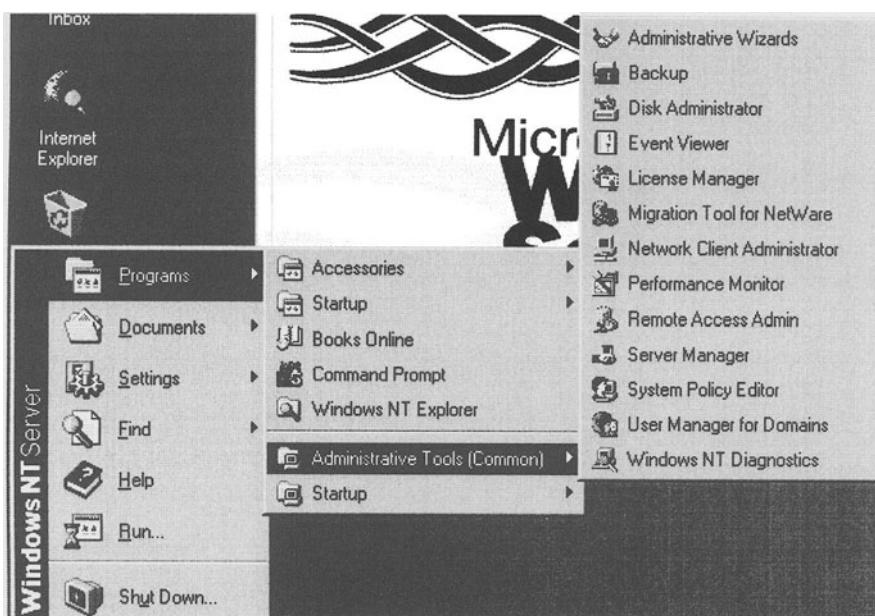
- User Manager for Domains. This tool supports the management of security for a domain.
- System Policy Manager. This tool manages the rights granted to groups and user accounts.
- Remote Access Admin. This is a tool which allows users to remotely login into a network server.
- Event Viewer. This is a tool which monitors system, security and application events.
- Disk Administrator. This is a tool which gives a graphical interface for managing disks.
- License Manager. This tool helps to manage and track licenses within an organization. Licensing can be done per seat or per sever.

### **P.3.2 User and group accounts**

Each user within a domain has a user account and is assigned to one or more groups. Each group is granted permissions for the file system, accessing printers, and so on. Group accounts are useful because they simplify an organization into a single administrative unit. They also provide a convenient method of controlling access for several users who will be using Windows NT to perform similar tasks. By placing multiple users in a group, the administrator can assign rights and/or permissions to the group.

Each user on a Windows NT system has the following:

- A user name (such as fred\_bloggs).
- A password (assigned by the administrator then changed by the user).
- The groups in which the user account is a member (for example, staff).
- Any user rights for using the assigned computer.



**Figure P.5** Windows NT administrative tools.

Each time a user attempts to perform a particular action on a computer, Windows NT checks the user account to determine whether the user has the authority to perform that action (such as read the file, write to the file, delete the file, and so on).

Normally there are three main default user accounts: Administrator, Guest and an 'Initial User' account. The system manager uses the Administrator account to perform such tasks as installing software, adding/deleting user accounts, setting up network peripherals, installing hardware, and so on.

Guest accounts allow occasional users to log on and be granted limited rights on the local computer. The system manager must be sure that the access rights are limited so that hackers or inexperienced users cannot do damage to the local system.

The 'Initial User' account is created during installation of the Windows NT workstation. This account, assigned a name during installation, is a member of the Administrator's group and therefore has all the Administrator's rights and privileges.

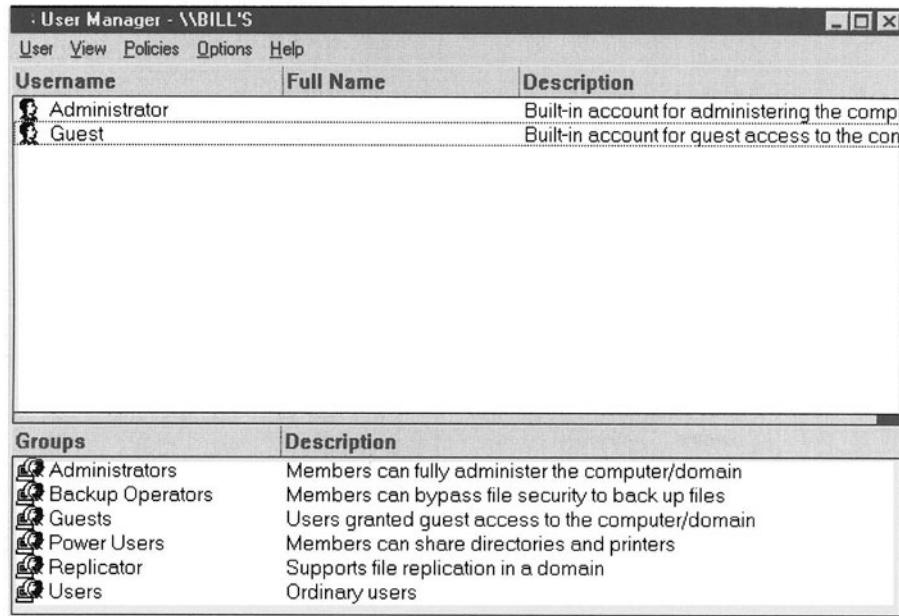
After the system has been installed, the Administrator can allocate new user accounts, either by creating new user accounts, or by copying existing accounts.

### P.3.3 User Manager

The User Manager allows the creation and management user accounts, creation and management of groups, and the management the security policies. Figure P.6 shows the user addition window (which is selected with the User → New User option). The title bar defines the domain or computer that is displayed for administration. Figure P.7 shows a window with several added users. It can be seen that initially that there are two users: Administrator (the System Manager) and the Guest. It can be seen that there are also some default groups:

- Administrators. Members can fully administer the computer/domain.
- Backup Operators. Members can bypass file security to backup files.

- Guests. Users granted guest logins.
- Power Users. Members can share files and directories.
- Replicators. Supports file replication in a domain.
- Users. Ordinary users.



**Figure P.6** User Manager.

When administering a computer running Windows NT Workstation or Windows NT Server that is not a domain controller, the list contains only local groups. These computers do not maintain global groups.

Users either have a global account icon or a local account icon. These are defined as:

- Global account. Typically, users have a global account which is a normal user account in a user's home domain. To define trust relationships for another domain, the user must have a global account on one domain to be granted access to another domain.
- Local account. A local account accommodates a user whose global account is in a domain which is not trusted by the current domain.

Typically, the system manager creates new accounts by copying existing users accounts. The items copied directly from an existing user account to a new user account are as follows:

- The description of the user (such as Fred Bloggs, Ext 4444).
- Group account membership (such as Production).
- Profile settings (such as home directory).
- If set, the attribute to stop the user from changing their password (sometimes the manager does not want the user to change the default password).
- If set, the attribute that causes the password to remain unexpired (sometimes the system manager forces users to change their passwords from time to time).

The items which are cleared and completed by the system manager are:

- The username and full name.
- The attribute that prompts users to change their passwords when they next login (normally a default password is initially set up and is changed when a user initially logs in).
- The attribute which disables the account (the manager must reset this before a user can log in).

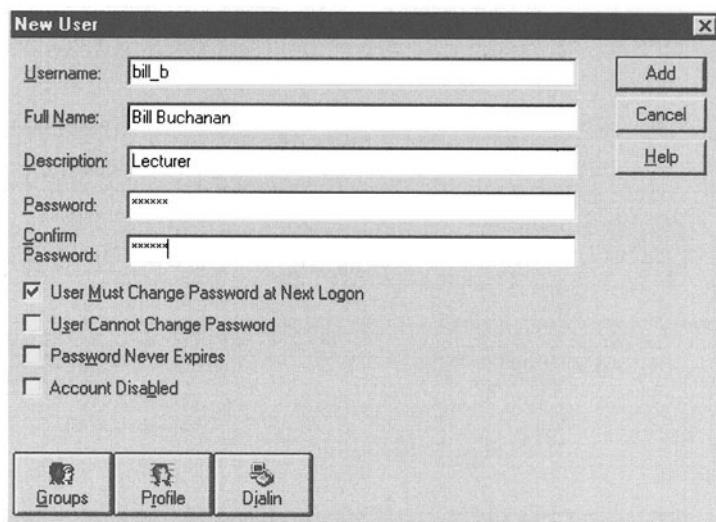


Figure P.7 Adding a user.

Figure P.8 shows the addition of several users and Figure P.9 shows the members of the Users group (showing the full name of the users). The groups that a user belongs to can be defined by selecting User → Properties then User Properties → Groups. Figure P.10 shows an example for the user bill\_b. It can be seen that this user is a member of Users, but not of the other groups (such as Administrators). A user can be added to a group by selecting the group in the right-hand side window and selecting the Add button. A user can be removed from a group by selecting the group on the left-hand window and selecting the Remove button.

User Manager - \\BILL'S

The screenshot shows the Windows User Manager interface. At the top, there's a menu bar with User, View, Policies, Options, and Help. Below the menu is a table titled 'User Manager' showing user accounts. The columns are 'Username', 'Full Name', and 'Description'. There are five entries:

Username	Full Name	Description
Administrator		Built-in account for administering the computer
bill_b	Bill Buchanan	Lecturer
david	David Buchanan	Games User
Guest		Built-in account for guest access to the computer
jamie	Jamie Buchanan	Games user

Below this is another table titled 'Groups' with columns 'Groups' and 'Description'.

Groups	Description
Administrators	Members can fully administer the computer/domain
Backup Operators	Members can bypass file security to back up files
Guests	Users granted guest access to the computer/domain
Power Users	Members can share directories and printers
Replicator	Supports file replication in a domain
Users	Ordinary users

Figure P.8 Multiple users.

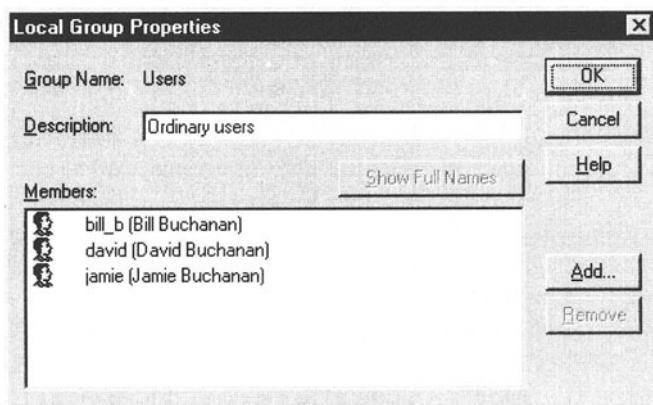


Figure P.9 Local group properties.

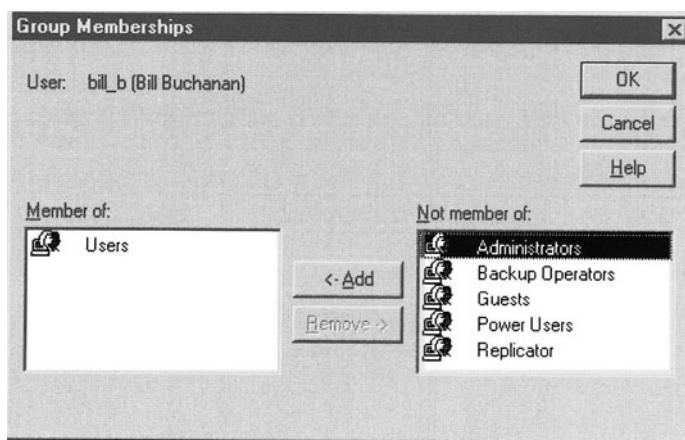


Figure P.10 Group membership.

#### P.3.4 System Policies Editor

The Systems Policy Editor provides policies for the following:

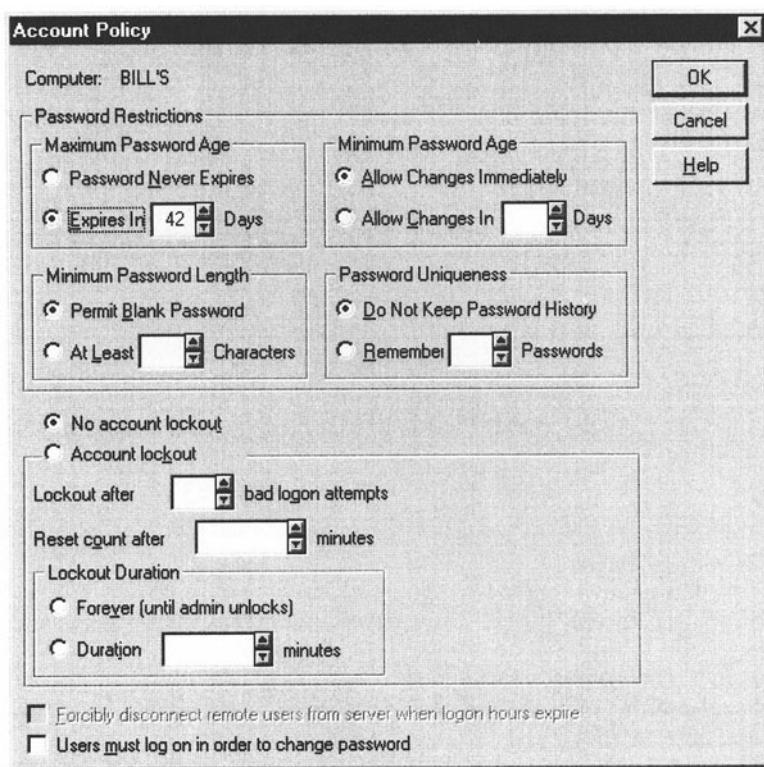
- Account policy. This policy defines the operation of the user password, such as minimum password size, number of acceptable bad passwords, whether there is a lock-out on the login, and so on.
- User Rights policy. This policy manages the rights of groups and user accounts. These rights allow users to access certain resources within the domain.
- Audit policy. This policy tracks auditing security events. These might be successful and/or unsuccessful events. The Event Viewer program can be used to view these events.
- Trust Relationships. Trust relationships create a link between two Windows NT Server domains and they allow users to be granted rights on other servers.

#### Account policy

This policy defines the operation of all user passwords, such as minimum password size, number of acceptable bad passwords, whether there is a lock-out on the login, and so on. Figure P.11 shows a sample window. It can be seen that, in this case, all passwords expire in 42 days. There is also no limit in the minimum time for a password change. Figure P.11 also shows that blank passwords are accepted and that there is no account lockout on a bad password entry. It can be seen that the number of acceptable bad passwords can also be set. This lockout can be reset after a number of minutes, if required.

#### User rights

The user rights policy manages the rights of groups and user accounts. These rights allow users to access certain resources within the domain. Normally these rights are granted to groups rather than to individual user accounts, as it is easier to administer. Figure P.12 shows an example.

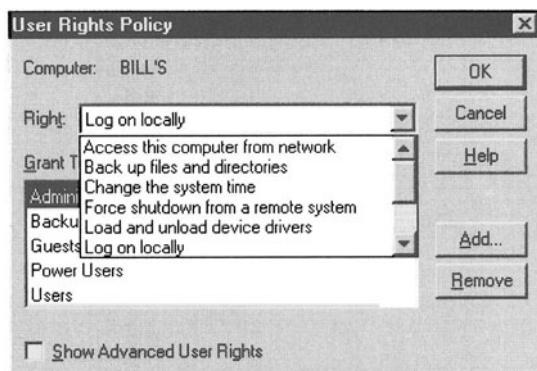


**Figure P.11** Account policy window.

The rights include (with typical user groups are):

- Access this computer from network (Administrator, Everyone, Power Users).
- Backup files and directories (Administrator, Backup Operators)
- Change system time (Administrator, Power Users).
- Force shutdown from a remote computer (Administrator, Power Users).
- Load and unload device drivers (Administrator).
- Log on locally (Administrators, Backup Operators, Guests, Power Users, Users).
- Managing audits and security log (Administrator).
- Restore files and directories (Administrator, Backup Operators).
- Shut down the system (Administrators, Backup Operators, Power Users, Users).
- Take ownership of files or other objects (Administrator).

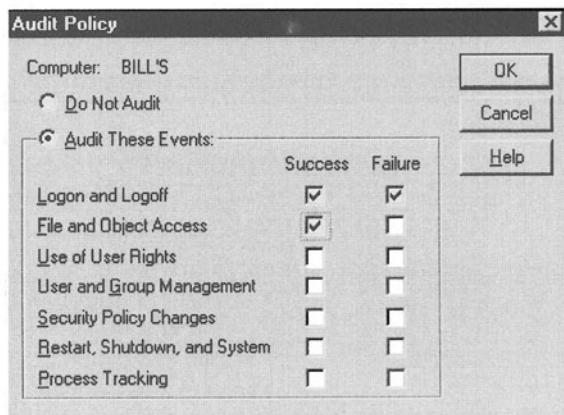
Initially the administrator selects the right and then chooses to Add... or Remove the group or user from the right.



**Figure P.12** User rights policy.

## Audit

This policy tracks auditing security events. These might be successful and/or unsuccessful events. The Event Viewer program can be used to view these events. Figure P.13 shows an example of an audit policy window. In this case, all successful and unsuccessful login and logoffs will be logged. In addition, successful file and object accesses will be logged.



**Figure P.13** Audit Policy.

## Trust Relationships

Trust relationships create a link between two Windows NT Server domains and they allow users to be granted rights on other servers.

### P.3.5 Control Panel and Windows NT Diagnostics

As with Windows 95 and Windows 98, Windows NT has a Control Panel from which users can modify or view system settings. Figure P.14 shows the Control Panel. The Windows NT Diagnostics program allows the user to view system settings. Figure P.15 shows the initial window.



Figure P.14 Control Panel.

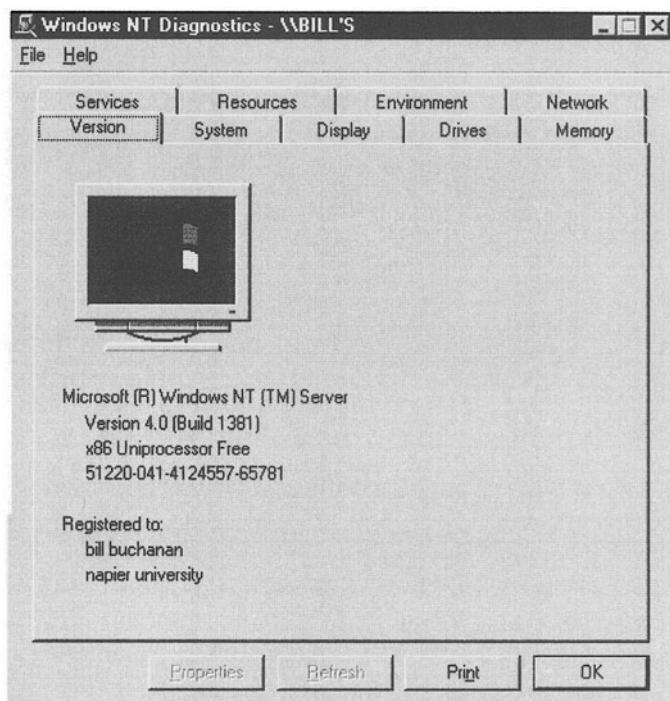


Figure P.15 Windows NT Diagnostics.

From the system selection (called the Windows NT Diagnostics window). The user can select to view the Resources, Services, Environment, Network, Memory, Drives, Display, System and Version. Figure P.16 shows an example Resources window. It can be seen that the currently assigned IRQs are displayed (for example, the serial port is using IRQ3 and IRQ4). The user can select to view the I/O memory map (I/O Port), DMA channels, memory and devices.

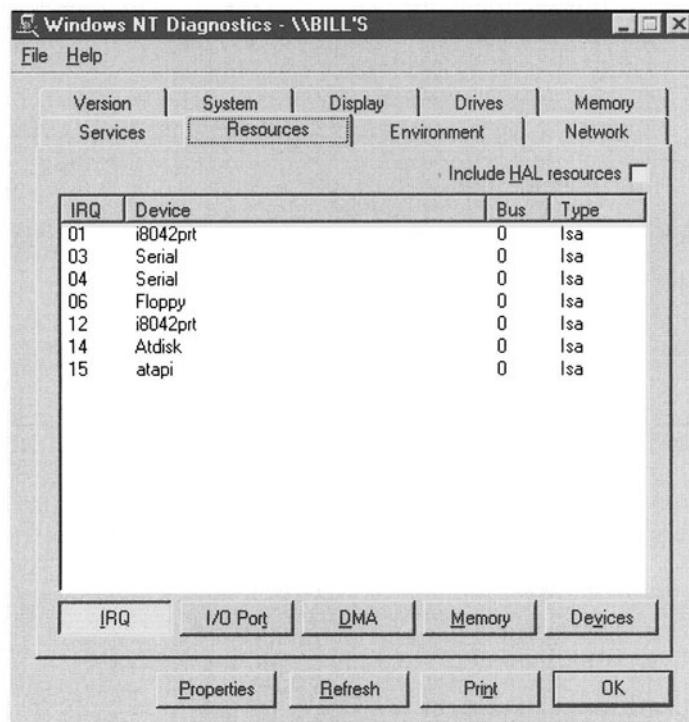


Figure P.16 Resources window.

Figure P.17 shows the system devices. In this case the devices connected are:

- PC Compatiable EISA/ISA (Device for handling I/O ports, system timer and interrupts).
- i8042ptr. PS/2 style mouse/ keyboard (both use the Intel 8042 IC).
- Parport. Parallel port (LPT1: 378h-37Ah).
- Serial. Serial port (COM1: IRQ4, 3F8h-3FEh; COM2: IRQ3, 2F8h-2FEh). See Figure P.19 for an example setting.
- Atdisk (Hard disk drive: IRQ14, 1F0h-1F7h, 3F6h).
- Floppy (floppy disk drive: IRQ6, 3F0h-3F5h, 3F7h, DMA2).
- Atapi (CDROM: IRQ15, 170h-177h).
- VgaSave (VGA: 3B0h-3BBh, 3C0h-3DFh, 1CEh-1CFh).

Figure P.19 shows the usage of the I/O ports.

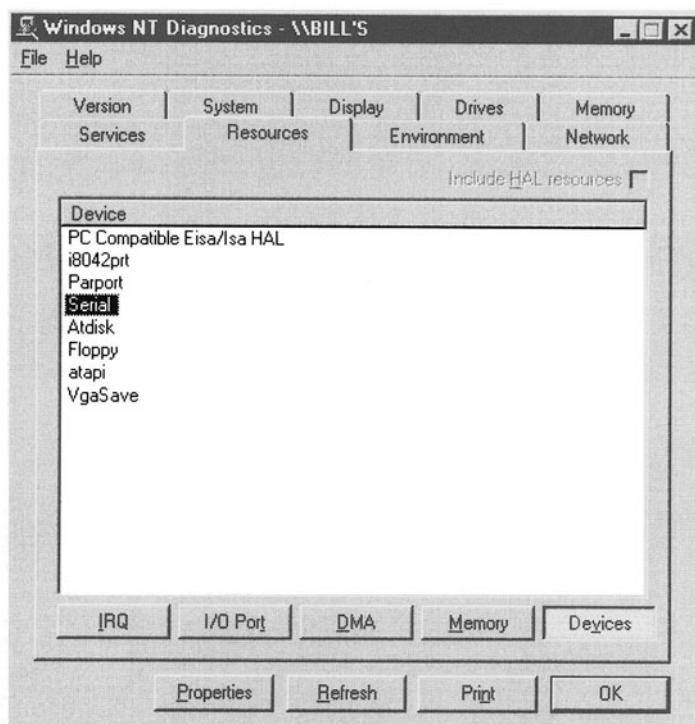


Figure P.17 Devices used.

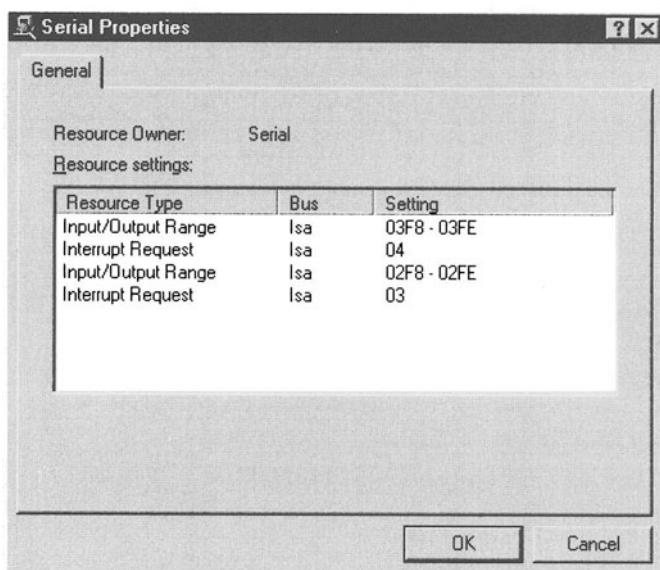


Figure P.18 Serial port properties.

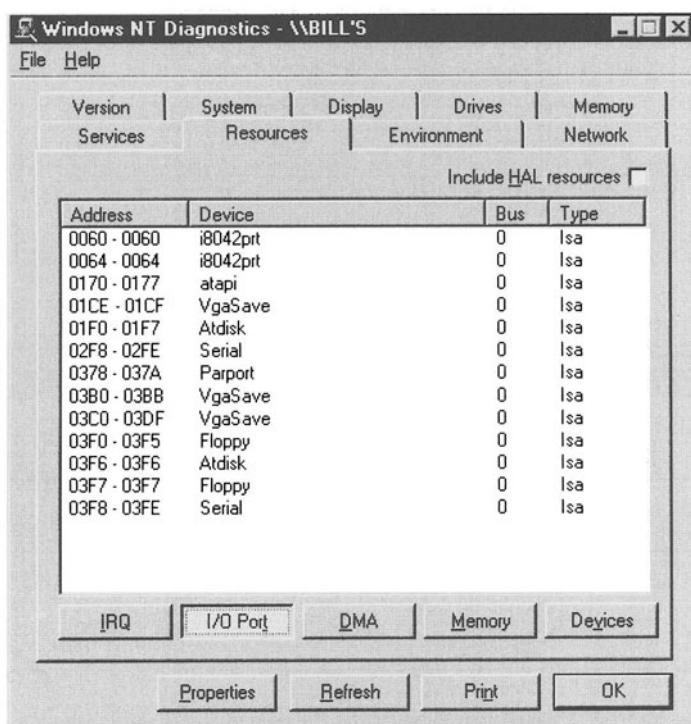


Figure P.19 I/O memory usage.

### P.3.6 Server Manager

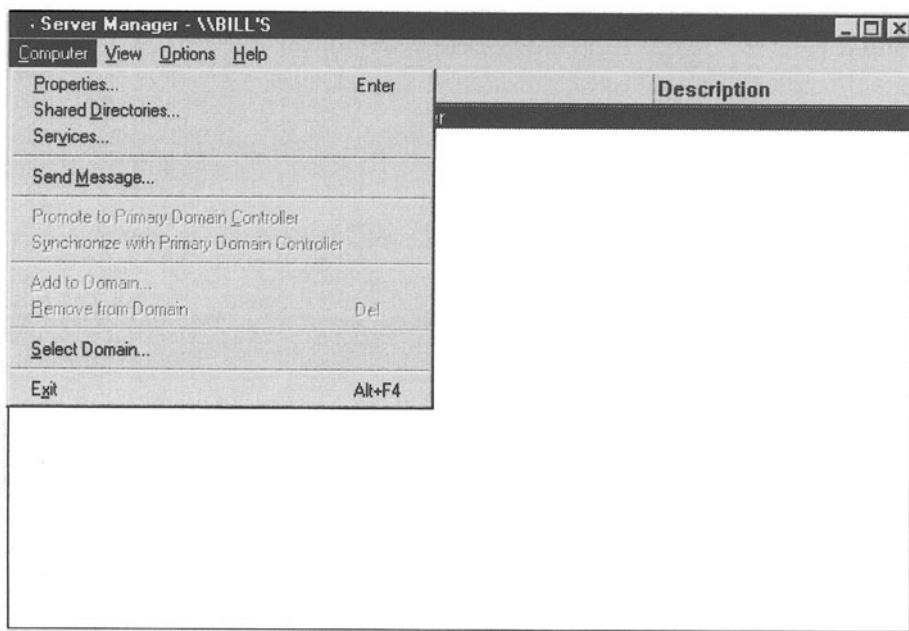
The Server Manager allows the Administrator to manage domains and computers. Figure P.20 shows the initial window. The computers within the domain are displayed within the window and Figure P.21 shows the window which appears after the Computer is selected. With computers in the domain the user can view information on the selected computer, such as:

- View connected users (see Figure P.22) and send messages to them.
- Open and share resources. Figure P.23 shows an example of a system with shared resources (C\$, ADMIN\$ and IPC\$). An icon of a folder with a hand holding it represents a shared directory, the icon of a connector represents a named pipe, a printer icon represents a shared printer and a question mark represents an unrecognized resource. The IPC\$ resources allows programs to intercommunicate (inter processes control) and must be running for shared resources. The ADMIN\$ name is used during the remote administration of the computer. This name always points to the directory in which NT is installed (in Figure P.23 this is C:\WINNT). The C\$ name represents the C: drive (other drives can be mounted using D\$, E\$, and so on). Other names are: NETLOGON\$ (Domain Net Login), PRINT\$ (print queues) and REPL\$ (replication export server).
- View and manage shared files and directories. Figure P.24 shows an example window. From this window the user can add or delete shared directories (New Share and Stop Share).

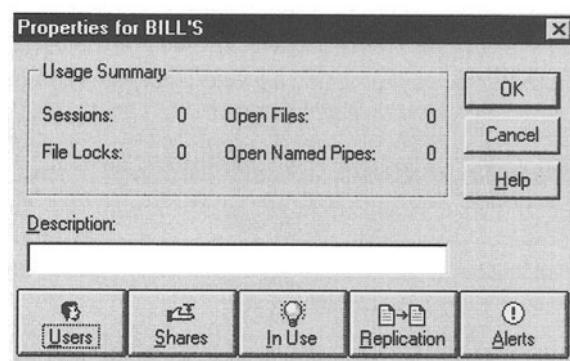
- Manage directory replication.
- Manage the list of administrative alert recipients.
- Manage services of any computer in the domain (Server Manager manages only the local computer).

For domains the following can be administered:

- Promote backup domain controllers to the primary domain controller.
- Synchronize servers with the primary domain controller.
- Add and/or remove computers to/from a domain.



**Figure P.20** Server Manager.



**Figure P.21** Properties.

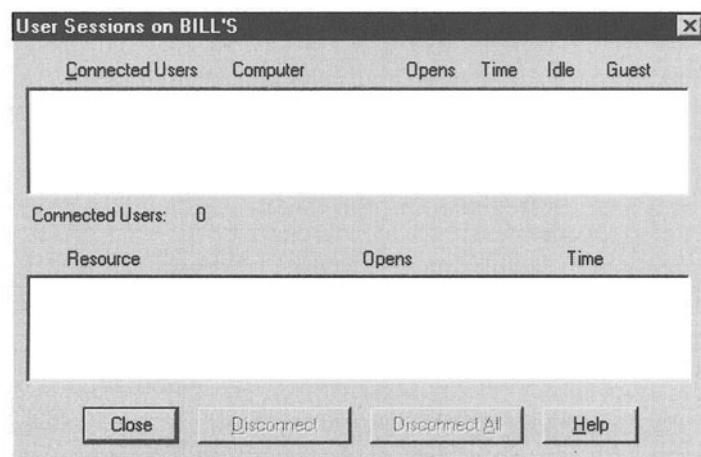


Figure P.22 User Session.

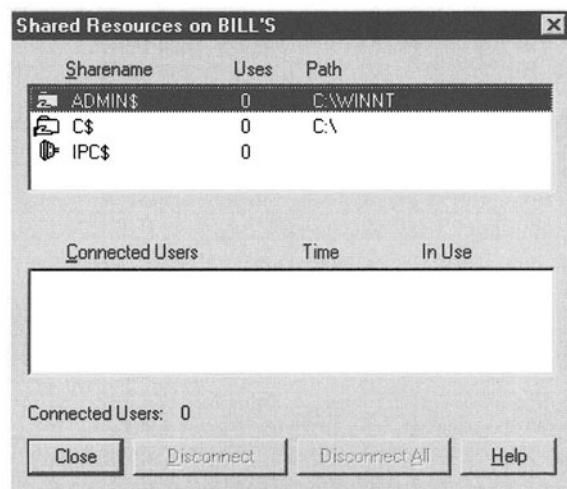


Figure P.23 Shared resources.

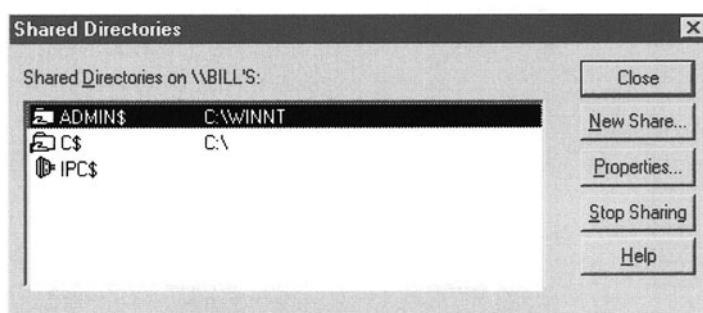
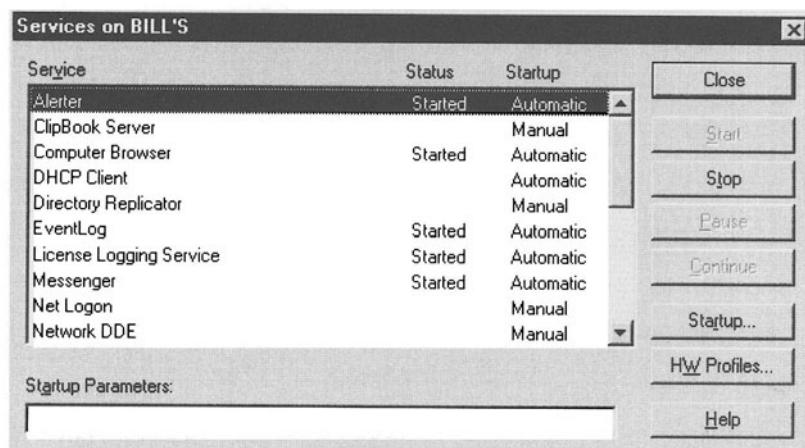


Figure P.24 Shared directories.

### P.3.7 Services

Services are processes which perform certain functions within the computer, but will generally slow the computer down. They can be run either from the Server Manager (for local and remote computers) or from Control Panel, then Services. Figure P.25 shows a sample window. A service can either be started, stopped, paused or continued, this is shown in the Status column. The Startup column identifies how the service is started. If it is Manual then the user or a dependent service must start it, if it is Disabled then the service is not started, else if it is automatic then the service started every time the computer starts. Table P.1 gives a list of typical services, their status and their startup condition.



**Figure P.25** Services.

**Table P.1** Typical services.

Service	Status	Startup	Description
Alerter	Started	Automatic	Notifies selected users and computers of administrative alerts that occur on the selected computer
ClipBook Server		Manual	Allow the Clipbook to be viewed on other remote computers
Computer Browser	Started	Automatic	Builds-up an up-to-date list of computers. For example, Server Manager uses this to display connected computers with a domain
DHCP Client	Started	Automatic	
Directory Replicator		Manual	Replicates directories between computers
EventLog	Started	Automatic	
License Logging Service	Started	Automatic	

Messenger	Started	Automatic	Sends and receives messages sent by administrators or by the Alerter service.
Net Logon		Manual	Network login for the primary and backup domain controllers.
Network DDE		Manual	Network Dynamic Data Exchange (DDE) between applications
Network DDE DSDM		Manual	Network Dynamic Data Exchange Share Database Manager
NT LM Security Support Provider		Manual	
Plug and Play	Started	Automatic	
Remote Procedure Call (RPC) Locator		Manual	Interprocess communication over a network
RPC Service	Started	Automatic	
Schedule		Manual	Runs scheduler
Server	Started	Automatic	File, print sharing and RCP
Spooler	Started	Automatic	Provides print spooler services
TCP/IP NetBIOS Helper	Started	Manual	
Telephony Service		Manual	
UPS		Manual	Manages an interruptible power support (UPS)
Workstation	Started	Automatic	Provides network connections and communications

### P.3.8 Event Viewer

The Event Viewer allows the user (normally the Administrator) to view events on the system. It is normally started by default when NT is started, but can be stopped by selecting Services from Control Panel, and then stopping event logging. Figure P.26 shows a security log and Figure P.27 shows an application log. Each window has certain columns, such as: Source (software that logged the event), User (user name), Category (classification of the event, as defined by the source, such as Logon and Logoff, Policy Change, Privilege Use, System Event, Object Access, Detailed Tracking, and Account Management), Computer (name of computer), Event (event number) and Type (such as Error, Warning, Information, Success Audit, or Failure Audit).

Event Viewer - Security Log on \\BILL'S							
Date	Time	Source	Category	Event	User	Co	Ca
8/23/98	10:16:49 PM	Security	Object Access	562	SYSTEM	BIL	
8/23/98	10:16:49 PM	Security	Object Access	560	Administrator	BIL	
8/23/98	10:16:49 PM	Security	Object Access	562	SYSTEM	BIL	
8/23/98	10:16:49 PM	Security	Object Access	560	Administrator	BIL	
8/23/98	10:16:35 PM	Security	Object Access	562	SYSTEM	BIL	
8/23/98	10:16:35 PM	Security	Object Access	560	Administrator	BIL	
8/23/98	10:16:35 PM	Security	Object Access	562	SYSTEM	BIL	
8/23/98	10:16:35 PM	Security	Object Access	560	Administrator	BIL	
8/23/98	10:04:33 PM	Security	Object Access	562	SYSTEM	BIL	
8/23/98	10:04:33 PM	Security	Object Access	560	Administrator	BIL	
8/23/98	10:04:33 PM	Security	Object Access	562	SYSTEM	BIL	
8/23/98	10:04:33 PM	Security	Object Access	560	Administrator	BIL	

Figure P.26 Security log.

Event Viewer - Application Log on \\BILL'S							
Date	Time	Source	Category	Event	User	Co	Ca
i 8/23/98	7:18:28 PM	Autochk	None	1001	N/A	BIL	
i 8/23/98	7:03:39 PM	Autochk	None	1001	N/A	BIL	

Figure P.27 Application log.

### P.3.9 Master server

A master server holds all the licensing data for an organization. Licenses can be:

- **Per Seat.** In this mode, each computer must have a Client Access License to access a Windows NT Server for basic network services, such as file and print services. This type of licensing is typically used when clients connect to more than one server.
- **Per Server.** In this mode, the Client Access Licenses are assigned to a particular server, which allows one connection to that server for basic network services (such as file and print services). There must thus be enough licenses for the number of clients which connect to the server. This type of licensing is typical when computers only connect to a single server.

### P.3.10 Boot process

Normally a computer runs a POST (Power On Self Test), then the MBR (Master Boot Record) is loaded. It then loads the PBR (Primary Boot Record) from the primary partition (normally the C: drive, A: drive, or, even a bootable CD-ROM drive). Next, the boot processes continues with the following files (and their location):

Boot.ini.	Root directory.
Bootsect.dos.	Root directory.
Device drivers.	<Ntroot>\System32\Drivers
Hal.dll.	<Ntroot>\System32
Ntboodd.sys.	Root directory.
Ntdetect.com.	Root directory.
Ntldr.	Root directory.
Ntoskrnl.exe.	<Ntroot>\System32
System.	<Ntroot>\System32\Config

The boot sequence is then as follows:

1. Ntldr is loaded and switches the processor to operate as a 32-bit processor with a flat memory structure.
2. Ntldr starts the appropriate file system mini-drivers (such as NTFS or FAT).
3. Ntldr reads the Boot.ini file and displays the bootable operating systems, such as Windows NT, Windows NT (VGA mode) and Windows 98. The user can then select the operating system to boot. Normally, there is a default operating system which is chosen within a given time.
4. Ntldr loads the required operating system. For NT, it runs Ntdetect.com which automatically scans the hardware and reports the hardware list to Ntldr. For non-NT, Bootsect.dos is loaded.
5. Ntldr loads Ntoskrnl.exe and Hal.dll.
6. Ntldr load Ntoskrnl.exe. This starts the system loader.

An example boot.ini file is given next:

```
[boot loader]
timeout=30
default=C:\="Microsoft Windows"
```

```
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Server
Version 4.00"
C:\="Microsoft Windows"
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Server
Version 4.00 [VGA mode]" /basevideo /sos
```

The timeout parameter is defined in the [boot loader] section. It contains the number of seconds that boot options are displayed, before the default operating system is loaded. The [operating systems] option defines each of the bootable systems. The basevideo operation uses a standard VGA driver. It can be seen, in this case, that there are three bootable file systems, these are:

```
Windows NT Server Version 4.00
Microsoft Windows
Windows NT Server Version 4.00 [VGA mode]
```

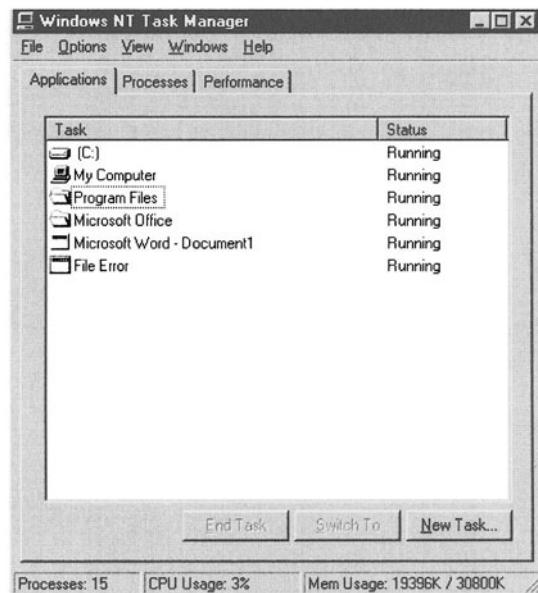
If the system has MS-DOS (such as Windows 3.x) then the line:

```
C:\= "MS-DOS"
```

is included.

### **P.3.11 Task Manager**

The Task Manager shows the currently running processes, and, if required, they can be stopped. It can be called by pressing Ctrl-Alt-Del and then selecting Task Manager. Figure P.28 shows a example of some processes running. The window icon with gray indicates a program, and a window icon with white indicates a status window. The open file icon indicates an open folder.



**Figure P.28 Applications.**

The processes window (as shown in Figure P.29) gives an indication of:

- Image name. Name of the process.
- PID. Process Identification.
- CPU Time. Total time that the process has used the processor.
- Memory usage. Total memory usage.

Figure P.30 shows the performance window. It shows memory and processor usage.

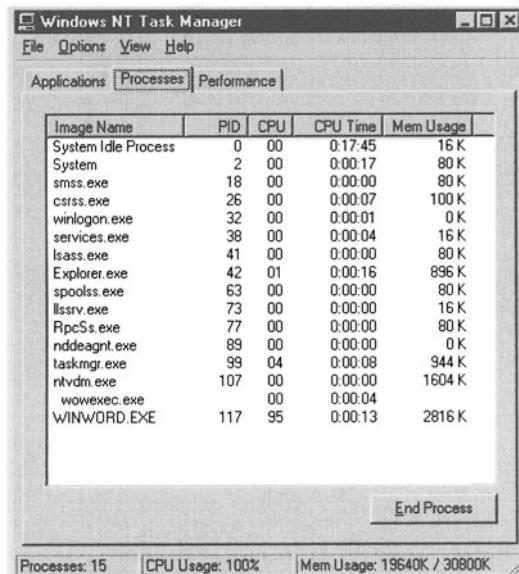


Figure P.29 Processes

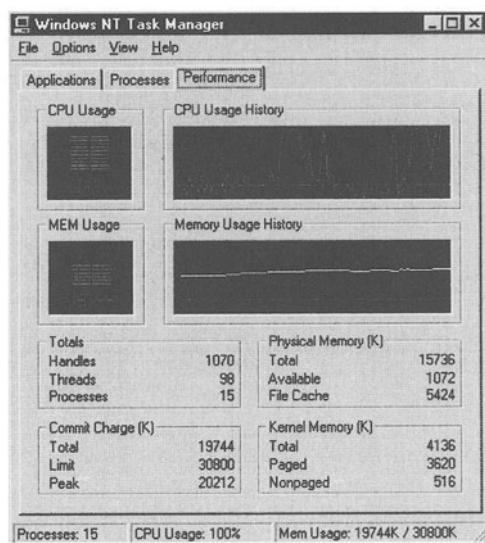
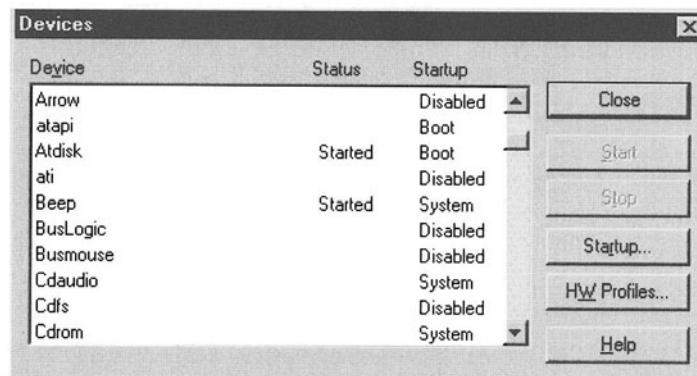


Figure P.30 Performance.

### **P.3.12 Enabling/disabling drivers**

Device drivers can be enabled and disabled from Control Panel→Devices. Figure P.31 shows a sample window. The devices (with their method of starting) include:

Atapi (Boot)	Atdisk (Boot)	Beep (Boot)
BusMouse (System)	Cdaudio (System)	Cdfs (System)
Cdrom (System)	Disk (Boot)	Floppy (System)
i8042 (System)	Keyboard (System)	NDIS (System)
Modem (Manual)	Mouse (System)	NetBEUI Protocol (Automatic)
NetBEUI Interface (Manual)		NetDetect (Manual)
NTFS (System)	Null (System)	NWLInk IPX/SPX (Automatic)
NWLink	NetBIOS (Automatic)	Parallel (Automatic)
Parport (Automatic)	Pcmcia (Boot)	PnP ISA (System)
Scsiprt (System)	TCP/IP Service (Automatic)	
VgaSave (System)	VgaStart (System)	WINS Client (Automatic)



**Figure P.31** Devices.

---

## **P.4 Viruses**

Computer systems, especially PCs, are susceptible to computer viruses. A computer virus is a program which infects other programs by modifying them and making a direct, or a modified, copy of itself. Most viruses are annoying and do little damage to the computer system, apart from replicating themselves.

The PC has always been plagued with viruses for many reasons, including:

- Ease of access to system files. These include boot files (IO.SYS and MSDOS. SYS) and start-up files.
- Ease of access to hardware drivers and interrupts. This danger has been reduced with Windows 95/98 and NT, which do not allow user programs to access the hardware directly and the operating system.

- The lack of file attributes. Most PC systems have simple file attributes for read, write and execute. These attributes can also be easily changed, as the files are not owned by anyone. UNIX makes this difficult as users have limited access to files in other user directories and also to system files.
- The challenge. There are more PCs on the planet than all the other types of computer systems put together. Thus, for some people, there is a thrill in the thought of infecting computers around the world.
- Ease to transfer. Most users use floppy disk drives to transfer files. These are a perfect mechanism for transporting virus. Many companies have even banned the use of floppy disk drives to try and reduce the spread of viruses.
- Boot disks. Until recently, many floppy disks were formatted so that they could be used as a boot disk. Unfortunately, a boot disk allows viruses to be transferred through the boot sector and remain in memory after the computer is booted.
- Accidental release. Many viruses started life as a prototype program or an experimental program, which were accidentally released in the ‘wild’. These are typically developed by undergraduate students who had just learned machine code and also PC system architecture. Most virus are not meant to damage to the computer system, but they typically have bugs in them which causes them to do some damage, such as to slow the computer down, or halt the computer.
- Thrill of event-driven programming. Many programmers love to write programs which are triggered by an event. Typical virus events are date (such as Friday the 13<sup>th</sup> and April 1<sup>st</sup>), time, and hardware or software interrupts.
- Criminal activity. This is actually not very common, as there are very few criminals who are also computer programmers. Virus programmers typically write virus programs for the thrill of it.
- Jokes. These are harmless programs that do ‘amusing’ things (although many users think they have been infected by a serious virus).

The damage caused by viruses is normally graded into five main definitions:

- Trivial. No physical damage and all the user has to do is to run an anti-virus program to get rid of the virus. Typically, the virus will occasionally display a message.
- Minor. Some files require to be re-installed from a backup or from a disk.
- Moderate. Virus trashes the hard disk, scrambles the FAT, or low-level formats the drive. This is typically recovered from a recent backup.
- Major. Virus gradually corrupts data files, of which the user is unaware of. This is more severe than moderate damage as earlier backups are likely to contain the virus. Many weeks or months worth of data may be lost.
- Severe. Virus gradually corrupts data files, but the user continues to use the computer. There is thus no simple way of knowing whether the data files, or the backup files, are good or bad).
- Unlimited. Virus gives a hacker access to your network, by stealing the supervisor password. The damage is then done by the hacker, who controls the network.

#### **P.4.1 Virus types**

##### **Partition Viruses**

When starting PCs, the system automatically reads the partition sector and executes the code it finds there. The partition sector (or Master Boot Record) is the first sector on a hard disk and it contains system start-up information, such as:

- Number of sectors in each partition.
- The start of the DOS partition starts.
- Small programs.

Viruses, which attach themselves to the partition sector, modify the code associated with the partition section. The system must be booted from a clean boot virus to eradicate this type of virus. Partition viruses only attack hard disks, as floppy disks do not have partition sectors. A partition is created with the FDISK program. Hard disks cannot be accessed unless they have a valid partition. Typical partitions are FAT-16, FAT-32 and NTFS.

##### **Boot sector viruses**

The boot sector resides on the first sector of a partition on a hard disk, or the first sector on a floppy disk. On starting, the PC reads from the active partition on the hard disk (identified by C:) or tries to read from the boot sector of the floppy disk. Boot sector viruses replace the boot sector with new code and moves, or deletes the original code.

Non-bootable floppy disks have executable code in their boot sector, which displays the message “Not bootable disk” when the computer is booted from it. Thus any floppy disk, whether it is bootable or non-bootable, can contain a virus, and can thus infect the PC.

##### **File viruses**

File viruses append or insert themselves into/onto executable files, such as .COM and .EXE programs. An indirect-action file virus installs itself into memory when the infected file is run, and infects other files when they are subsequently accessed.

##### **Overwriting viruses**

Overwriting viruses overwrite all, or part, of the original program. They are easy to detect as missing files are easily detected.

#### **P.4.2 Anti-virus programs**

Viruses use two main methods to introduce themselves to a computer system. These are through:

- Boot records on floppy and hard disks.
- Files. These are contained in either binary executable files or are macro viruses, which require another program to run them. A typical virus is the Word Macro virus, which effects the macros in Word 6.0. It cannot run itself and needs to hide within a Word document.

Virus scanning programs use a number of techniques to detect virus, these include:

- Scanning. Most viruses have a digital signature within a file or boot sector which identifies the virus. For example, the Murphy virus has the ASCII characters of Murphy added to the end of a file. The virus-scanning program thus has a table of known digital signatures which it compares with a byte-by-byte read of the disk drive. Care must be taken, though, that the file does not normally contain a genuine sequence of bytes which are mistaken for the virus. The scanner will thus report a virus when there is none.
- Change detection. This allows the virus checker to keep a list of the date and time that files were last changed. Any changes to the files can then be checked for viruses. Typically, the change detector keeps a track of changes to the main boot files. Unfortunately, virus programmers can change the date of a file so that it has the date of the original 'uninfected' file.
- Heuristic analysis. This technique attempts to detect viruses by watching for appearance or behavior that is characteristic of some class of known viruses. This allows for anti-virus program to detect new viruses which are not defined in the scanning technique.
- Verification. Scanning, change detection and heuristics can only identify a possible virus, only a verification program can prove that it really exists. Verifying program operates on the identified virus file and proves if it has been infected.

### Disinfection

Disinfection involves removing a virus from the system, if possible it should be used with the minimum of damage. The two main methods are:

- Specific knowledge disinfection. With this method the disinfection program reverses the actions of the virus infection. This requires some knowledge of how the virus infects the system in the first place.
- Generic disinfection. This method has information about what the original file or boot record. This typically involves methods, such as storing system files, re-installing system files, and rebuilding the boot record.

It is sometimes difficult to disinfect system which has a memory-resident virus, as the disinfection program could be tampered with as it is being run. Thus, it is often advisable to boot the anti-virus program from a clean boot disk.

An disinfecter is intelligent in that it will not damage an infected file, and will generally prompt the user to delete the infected file rather than damage it (which could cause more problems).

#### P.4.3 Trojan horses

A Trojan horse virus hides itself as a tempting-looking executable file or in a compressed file. Until recently virus checkers did not test compressed files for viruses, but most currently available virus checkers checker test the uncompressed files within a compressed file. Many also test files as they are executed, and the virus would thus be detected as the compressed file was uncompressed. A Trojan horse virus attached to an executable file will typically have an interesting name, such as:

hello.exe  
nice.exe

mail.exe

One Trojan horse virus, available from a WWW page, contains a single executable which is said to contain over 100 virus. This, the owner of the WWW page, states that it can be used against someone's worst enemy. The WWW page quotes that it can be spread by sending the user an email with the attached executable file or to leave it on a targeted users computer. Writing a Trojan horse virus is extremely simple and can simply involve a batch file which has:

```
cd \ ; or 'cd /' on a UNIX system
rm -r *.*
```

which, on a UNIX system or a PC system with the `rm` utility installed, will change the directory to the top-level and then try to delete all subdirectories below the top-level directory. It is also extremely easy for a user to write a program which scans all the subdirectories below the current directory, and delete their contents.

#### **P.4.4 Polymorphic viruses**

A polymorphic virus encrypts itself so that its signature is invisible to a virus scanner. It propagates itself by first decrypting itself using a decryption routine. This decrypted program can then do damage, such as spreading itself or deleting files on the computer. A non-polymorphic encrypted virus uses a decryption routine that does not change. Thus is it easy to detect as the signature is unchanging. A polymorphic virus uses a changeable decryption routine (known as a mutation engine), in which the signature is different each time. The mutation engine normally uses a random number and a simple mathematical algorithm to change the virus's signature.

Typical COM/EXE virus infectors are:

- Yankee Doodle. It infects EXE and COM files (and adds 2772 bytes). When an infected file is run, the virus automatically loads into memory and infects any program which is run. At 5:00 pm the virus causes the system to sometimes play "Yankee Doodle".
- SatanBug. It infects EXE and COM files (and adds between 3500 to 5000 bytes). When an infected file is executed, the virus installs itself in memory and infects any program which is executed. A SatanBug infected system generally runs slow and some programs may fail, but does no long term damage to the computer.
- Frodo. It infects EXE and COM files (and adds between 4096 bytes). As above it infects all programs which are executed. When the date is between September 22 and December 31, the virus typically hangs the computer. This is because there are bugs in the virus code) which intends to overwrite the boot record with a program to display the message "Frodo Lives" when the machine boots.

#### **P.4.5 Stealth viruses**

Stealth viruses hide themselves in system files, or in boot records. They are then called when a certain action occurs on the computer. Typically, the system files which are used to start the computer are infected, such as IO.SYS and MSDOS.SYS (which are called when the computer starts). Once the virus is in memory, it can hook itself onto a given action, such as interrupt routine. For example, by hooking on interrupt 21h the virus can interrupt all accesses to DOS services, such as reading and writing to the hard disk.

A stealth size virus attaches itself to a file, when the file is accessed the virus copies itself, which increases the size of the file. A read stealth virus intercepts a request to an infected file and presents uninfected contents. Thus to the user, files seem to be operating as normal. Unfortunately, this stealth rapidly spreads through the system.

A stealth virus is relatively easy to detect and erase, as they do not change their digital signature. It is important, though, when eradicating a stealth virus that the system is booted with a clean boot disk. Typical stealth viruses include:

- Stoned. The Stoned virus infects floppy and hard disk boot sectors. When the computer starts from an infected diskette, the virus infects the master boot record of the first physical hard disk. It installs itself in memory and occasionally displays the message “Your PC is now Stoned!”. It then infects the boot sectors of any inserted disk. Luckily, it does not cause damage to computer files and there is no loss of data.
- Ping Pong. This virus infects diskettes and the hard disk partition (non-master) boot record. It sometimes produces a bouncing dot on the screen after booting and adds approximately 975 bytes to the size of files.

#### **P.4.6 Slow viruses**

A slow virus operates by targeting the copying or modify of a file, and leave the original file untouched. Thus, a virus scanner may not pick-up that the file has been modified by the virus as it looks as if it was modified by some other normal operation. An example slow virus is:

- Dark Avenger. This is a resident COM and EXE file virus and adds 1800 bytes onto infected COM files. When an infected program is run, the virus installs itself in memory. It then only infects EXE or COM files which are run, opened, renamed, or operated on. Roughly every 16 times an infected program is run, it overwrites a random sector of the disk and displays the message: “Eddie lives...somewhere in time!”.

These viruses can be defeated by memory-resident virus checkers who test the creation of new files (an integrity shell).

#### **P.4.7 Retro viruses**

Retro viruses (or anti-anti viruses) try to disrupt the operation of an anti-virus program. They are designed by analyzing publicly available anti-virus programs (such as Dr Solomon’s or McAfee’s) and identifying potential weaknesses. Typical methods include:

- Modifying operation. In this method the virus program changes the operation of the virus scanner, such as stopping it from scanning certain files, or blocking any system messages, or even, mimicking the output of the anti-virus program.
- Modify database. In this method the virus program modifies the virus signature database file so that the retro virus has the wrong digital signature (or even modify it when it is loaded into the computer’s memory).
- Detect and run. This method involves the virus program detecting when the anti-virus program is run, and then hiding from it, either by deleting itself from memory or hiding in a file for a short time.

#### P.4.8 Worms

Before the advent of LANs and the Internet, the most common mechanism for spreading viruses was through floppy disks and CD-ROM disks. Anti-virus programs can easily keep up-to-date with the latest virus, and modify their databases. This is a relatively slow method of spreading a virus and will take many months, if not years, to spread a virus over a large geographical area. Figure P.32 illustrates the spread of viruses.

LANs and the Internet have changed all this. A virus can now be transmitted over a LAN in a fraction of a second, and around the world in less than a second. Thus a virus can be created and transmitted around the world before an anti-virus program can even detect that it is available.

A worm is a program which runs on a computer and creates two threads. A thread in a program is a unit of code that can get a time slice from the operating system to run concurrently with other code units. Each process consists of one or more execution threads that identify the code path flow as it is run on the operating system. This enhances the running of an application by improving throughput and responsiveness. With the worm, the first thread searches for a network connection and when it finds a connection it copies itself to that computer. Next, the worm makes a copy of itself, and runs it on the system. Thus, a single copy will become two, then four, eight, and so on. This continues until the system, and the other connected systems, will be shutdown. The only way to stop the worm is to shutdown all the effected computers at the same time and then restart them. Figure P.33 illustrates a worm virus.

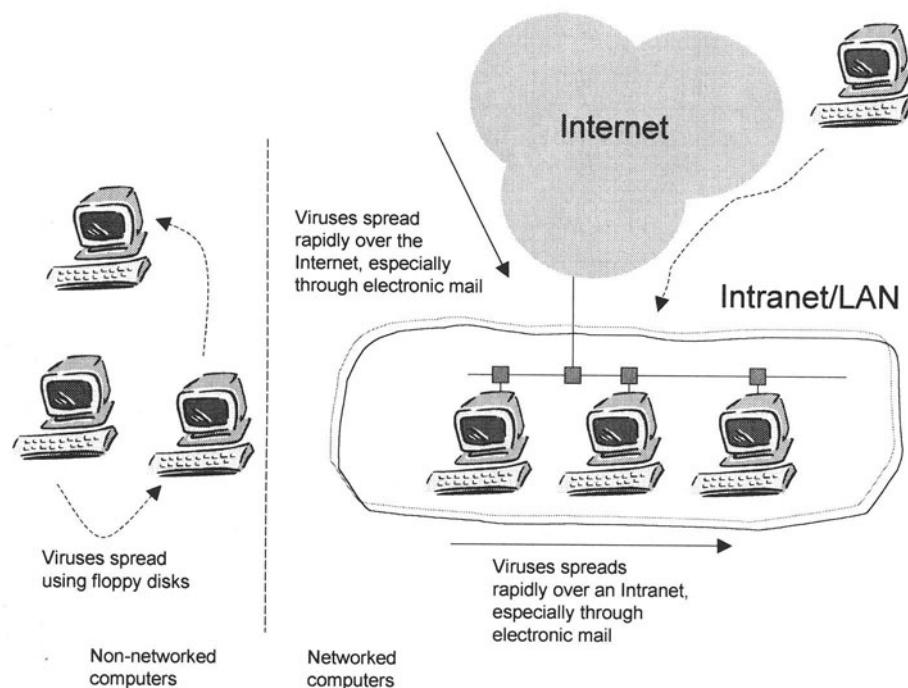
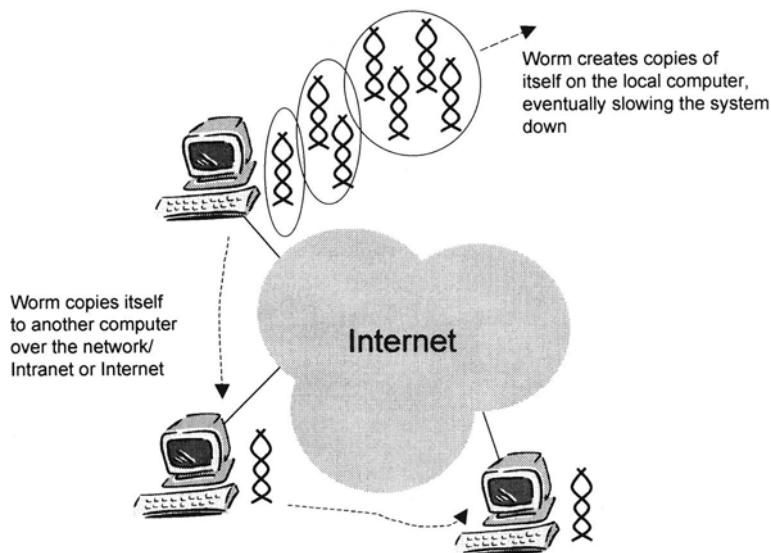


Figure P.32 Spread of viruses.



**Figure P.33 Worm viruses.**

#### P.4.9 Macro viruses

One of the most common viruses is the macro virus, which attacks macro or scripting facilities which are available in word processors (such as Microsoft Word), spread-sheets (such as Microsoft Excel), and remote transfer programs. A typical virus is the WM/CAP virus which modifies macros within Microsoft Word Version 6.0. When a macro is executed it can cause considerable damage, such as: deleting files, corrupting files, and so on.

The greatest increase in macro viruses is the number of viruses which use Microsoft Visual Basic for Applications (VBA) as this integrates with Microsoft Office (although recent releases have guarded against macro viruses). A macro virus in Word 6.0 is spread by:

- The file is either transmitted over email, over a LAN or from floppy disk.
- The infected file is opened and the `normal.dot` main template file is modified so that it contains the modified macros. Any files that are opened or created will now have the modified macros. The WM/CAP macro virus does not do much damage and simply overwrites existing macros.
- VBA is made to be event-driven, so operations, such as File Open or File Close, can have an attached macro. This makes it easy for virus programmers to write new macros.

Figure P.34 shows an example of a macro created by Visual Basic programming. The developed macro (`Macro1()`) simply loads a file called `AUTHOR.doc`, selects all the text, converts the text to bold, and then saves the file as `AUTHOR.rtf`. It can be seen that this macro is associated with `normal.dot`.

#### P.4.10 Other viruses

Other viruses include:

- Armored. This type of virus uses special code which makes them difficult to detect, or even understand.
- Companion. This type of virus creates an executable program which contains the virus, which then calls the mimicked program. For example, a virus could create a program called EXCEL.EXE which would contain the virus, and this could call the standard Microsoft Excel program.
- Phage. This type of virus modifies the code of a program, and unlike the previously discussed viruses they do not attach themselves onto a file. This type of virus is extremely dangerous as it is very difficult to recover the original program code, without having some knowledge of the changes that the phage virus is likely to do.

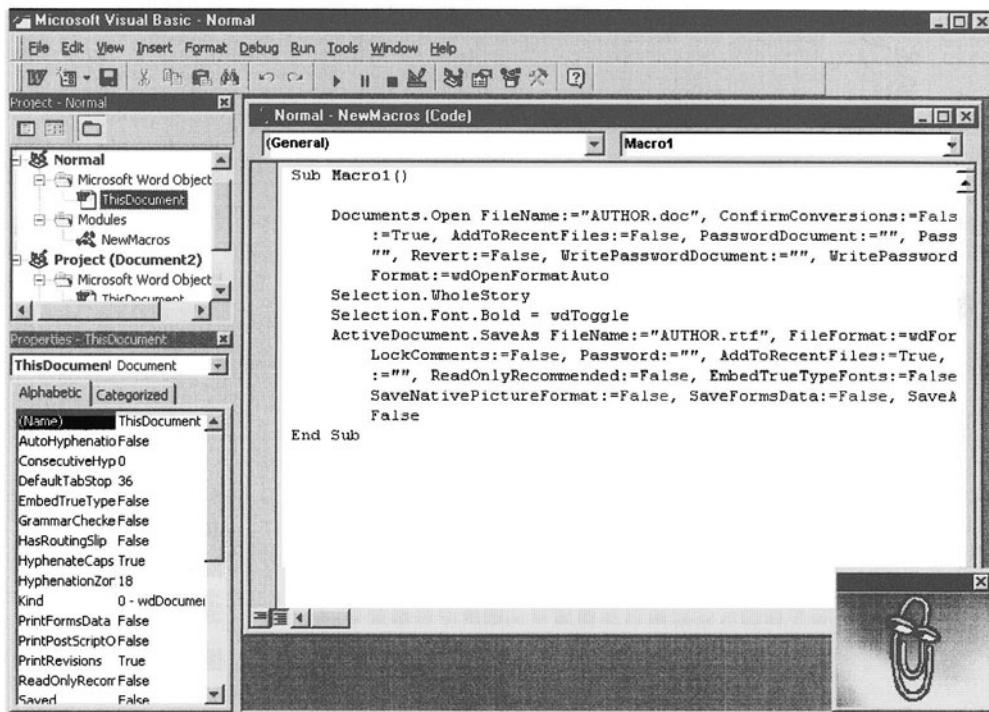


Figure P.34 Sample macro using VB programming.

## P.5 Security

---

### P.5.1 Introduction

Security involves protecting the hardware and software on a system from both internal attack and from external attack (hackers). An internal attack normally involves uneducated users causing damage, such as deleting important files or crashing systems. This effect can be

minimized if the system manager properly protects the system. Typical actions are to limit the files that certain users can access and also the actions they can perform on the system.

Most system managers have seen the following:

- Sending a file of the wrong format to the system printer (such as sending a binary file). Another typical one is where there is a problem on a networked printer (such as lack of paper), but the user keeps re-sending the same print job.
- Deleting the contents of sub-directories, or moving files from one place to another (typically, these days, with the dragging of a mouse cursor). This problem can be reduced by regular backups.
- Deleting important system files (in a PC, these are normally AUTOEXEC.BAT and CONFIG.SYS). This can be overcome by the system administrator protecting important system files, such as making them read-only or hidden.
- Telling other people their user passwords or not changing a password from the initial default one. This can be overcome by the system administrator forcing the user to change their password at given time periods.

Security takes many forms, such as:

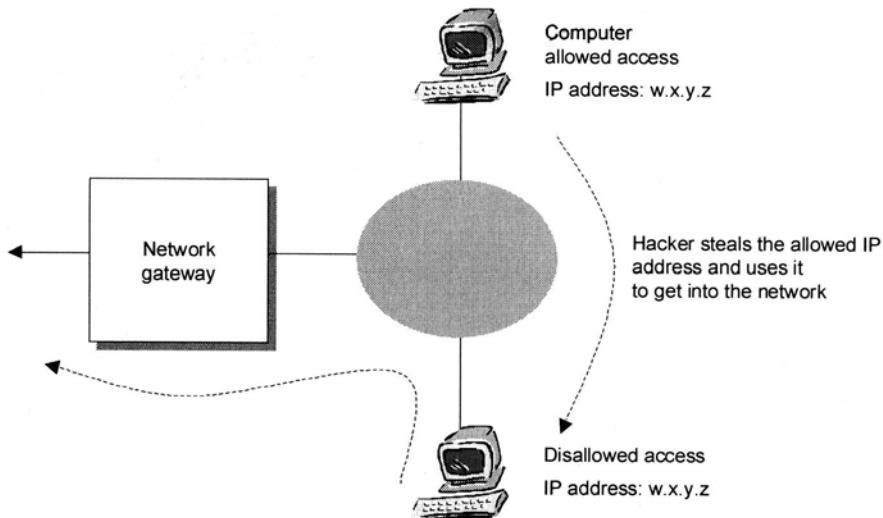
- Data protection. This is typically where sensitive or commercially important information is kept. It might include information databases, design files or software code files. One method of reducing this risk is to encrypt important files with a password and/or some form of data encryption.
- Software protection. This involves protecting all the software packages from damage or from being misconfigured. A misconfigured software package can cause as much damage as a physical attack on a system, because it can take a long time to find the problem.
- Physical system protection. This involves protecting systems from intruders who might physically attack the systems. Normally, important systems are locked in rooms and then within locked rack-mounted cabinets.
- Transmission protection. This involves a hacker tampering with a transmission connection. It might involve tapping into a network connection or total disconnection. Tapping can be avoided by many methods, including using optical fibers which are almost impossible to tap into (as it would typically involve sawing through a cable with hundreds of fiber cables, which would each have to be connected back as they were connected initially). Underground cables can avoid total disconnection, or its damage can be reduced by having redundant paths (such as different connections to the Internet).

### **P.5.2 Hacking methods**

The best form of protection is to disallow hackers into the network in the first place. Organizational networks are hacked for a number of reasons and in a number of ways. The most common methods are:

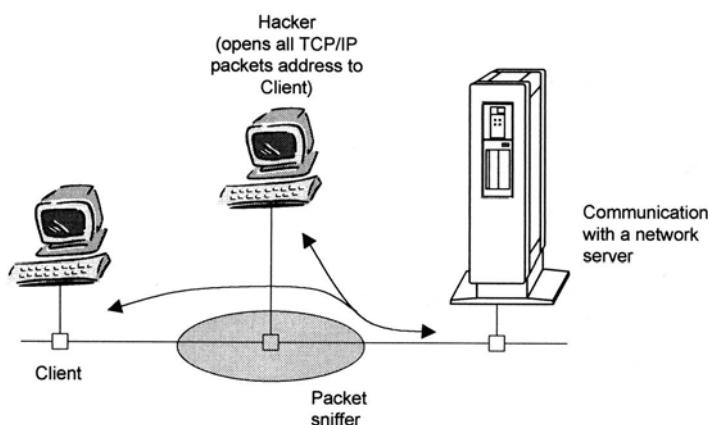
- IP spoofing attacks. This is where the hacker steals an authorized IP address, as illustrated in Figure P.35. Typically, it is done by determining the IP address of a computer and waiting until there is no-one using that computer, then using the unused IP address. Several users have been accused of accessing unauthorized material because other users have used their IP address. A login system which monitors IP addresses and the files that

they are accessing over the Internet cannot be used as evidence against the user, as it is easy to steal IP addresses.

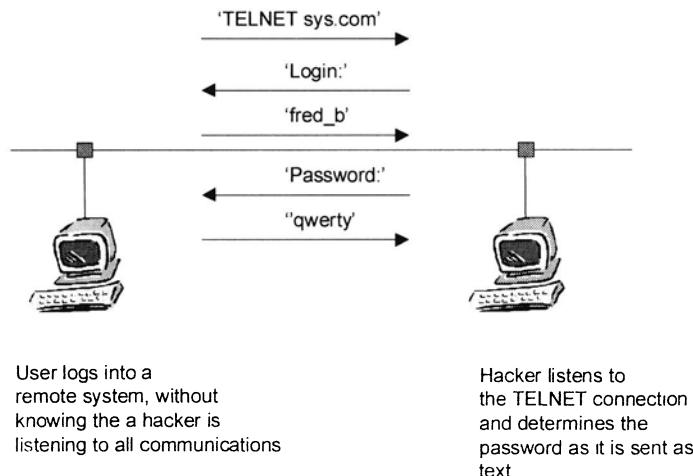


**Figure P.35** IP spoofing.

- Packet-sniffing. This is where the hacker listens to TCP/IP packets which come out of the network and steals the information in them. Typical information includes user logins, e-mail messages, credit card number, and so on. This method is typically used to steal an IP address, before an IP spoofing attack. Figure P.36 shows an example where a hacker listens to a conversation between a server and a client. Most TELNET and FTP program actually transmit the user name and password as text values, these can be easily viewed by a hacker, as illustrated in Figure P.37.



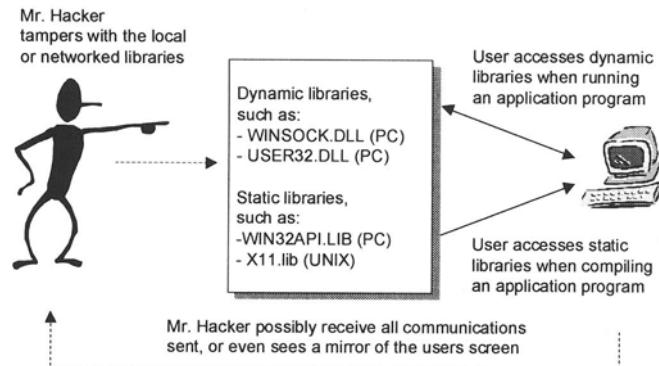
**Figure P.36** Packet sniffing.



**Figure P.37** Packet sniffing on a TELNET connection.

- Passwords attacks. This is a common weak-point in any system, and hackers will generally either find a user with a easy password (especially users which have the same password as their login name) or will use a special program which cycles through a range of passwords. This type of attack is normally easy to detect. The worst nightmare of this type of attack is when a hacker determines the system administrator password (or a user who has system privileges). This allows the hacker to change system set-ups, delete files, and even change user passwords.
- Sequence number prediction attacks. Initially, in a TCP/IP connection, the two computers exchange a start-up packet which contains sequence numbers. These sequence numbers are based on the computer's system clock and then run in a predictable manner, which can be determined by the hacker.
- Session hi-jacking attacks. In this method, the hacker taps into a connection between two computers, typically between a client and a server. The hacker then simulates the connection by using its IP address.
- Shared library attacks. Many systems have an area of shared library files. These are called by applications when they are required (for input/output, networking, graphics, and so on). A hacker may replace standard libraries for ones that have been tampered with, which allows the hacker to access system files and to change file privileges. Figure P.38 illustrates how a hacker might tamper with dynamic libraries (which are called as a program runs), or with static libraries (which are used when compiling a program). This would allow the hacker to possibly do damage to the local computer, send all communications to a remote computer, or even view everything that is viewed on the user screen. The hacker could also introduce viruses and cause unpredictable damage to the computer (such as remotely rebooting it, or crashing it at given times).
- Social engineering attacks. This type of attack is aimed at users who have little understanding of their computer system. A typical attack is where the hacker sends an email message to a user, asking for their password. Many unknowing users are tricked by this attack. A few examples are illustrated in Figure P.39. From the initial user login, the hacker can then access the system and further invade the system.

- Technological vulnerability attack. This normally involves attacking some part of the system (typically the operating system) which allows a hacker to be access to the system. A typical one is for the user to gain access to a system and then run a program which re-boots the system or slows it down by running a processor intensive program. This can be overcome in operating systems such as NT and UNIX by granting re-boot rights only to the system administrator.



**Figure P.38** Shared library attack.



**Figure P.39** Social Engineering attack.

- Trust-access attacks. This allows a hacker to add their system to the list of systems which are allowed to log into the system without a user password. In UNIX this file is the *.rhosts* (trusted hosts) which is contained in the user's home directory. A major problem

is when the trusted hosts file is contained in the root directory, as this allows a user to log in as the system administrator.

### P.5.3 Security policies

A well protected system depends mainly on the system manager. It is up to the manager to define security policies which define how users can operate the system. A good set of policies would be:

- Restrictions on users who can use a given account. The system administrator needs to define the users who can login on a certain account.
- Password requirements and prohibitions. This defines the parameters of the password, such as minimum password size, time between password changes, and so on.
- Internet access/restrictions. This limits whether or not a user is allowed access to the Internet.
- User account deletion. The system administrator automatically deletes user accounts which are either not in use or moved to another system.
- Application program rules. This defines the programs which a user is allowed to run (typically games can be barred from some users).
- Monitoring consent. Users should be informed about how the system monitors their activities. It is important, for example, to tell users that their Internet accesses are being monitored. This gives the user no excuse when they are found to be accessing restricted sites.

### P.5.4 Passwords

Passwords are normally an important part of any secure network. They can be easily hacked with the use of a program which continually tries different passwords within a given range (normally called directory-based attacks). These can be easily overcome by only allowing a user three bad logins before the system locks the user out for a defined time. NetWare and Windows NT both use this method, but UNIX does not. The system manager, though, can determine if an attack has occurred with the BADLOG file. This file stores a list of all the bad logins for user and the location of the user.

Passwords are a basic system for provide security on a network, and they are only as secure as the user makes them. Good rules for passwords are:

- Use slightly usual names, such as *vinegarwine*, *dancertop* or *helpcuddle*. Do not use names of a wife, husband, child or pet. Many users, especially ones who know the user can easily guess the user's password.
- Use numbers after the name, such as *vinedrink55* and *applefox32*. This makes the password difficult to crack as users are normally only allowed a few chances to login correctly before they are logged out (and a bad login event written to a bad login file).
- Have a few passwords which are changed at regular intervals. This is especially important for system managers. Every so often, these passwords should be changed to new ones.
- Make the password at least six characters long. This stops 'hackers' from watching the movement of the users fingers when they login, or from running a program which tries every permutation of characters. Every character added, multiplies the number of combinations by a great factor (for example, if just the characters from 'a' to 'z' and '0' to '9'

are taken then every character added increases the number of combinations by a factor of 36).

- Change some letters for numbers, or special characters. Typically, ‘o’ becomes a 0 (zero), ‘i’ becomes 1 (one), ‘s’ becomes 5 (five), spaces become ‘\$’, ‘b’ becomes ‘6’, and so on. So a password of ‘silly password’ might become ‘Silly\$pa55w0rd’ (the user makes a rule for ‘s’ and ‘o’). The user must obviously remember the rule that has been used for changing the letters to other characters. This method overcomes the technique of hackers and hacker programs, where combinations of words from a dictionary are hashed to try and make the hashed password.

The two main protocols used are:

- Password Authentication Protocol (PAP). This provides for a list of encrypted passwords.
- Challenge Handshake Authentication Protocol (CHAP). This is a challenge-response system which requires a list of unencrypted passwords. When a user logs into the system a random key is generated and sent to the user for encrypting the password. The user then uses this key to encrypt the password, and the encrypted password is sent back to the system. If it matches its copy of the encrypted password then it lets the user log into the system. The CHAP system then continues to challenge the user for encrypted data. If the user gets these wrong then the system disconnects the login.

### **Administrator user name**

The administrator account has the highest privileges and will thus be the most highly prized hackable account. If it is hacked into then the whole system is vulnerable to attack until it can be proved that there are no hidden traps. One method of reducing the risk of being hacked is to change the name of the administrator from time to time. The administrator password should be totally secure with a relatively large password size.

### **Password security**

The US Government defines certain security levels: D, C1, C2, B1, B2, B3 and A1, which are published in the *Trusted Computer Security Evaluation Criteria* books (each which have different colored cover to define their function), these include:

- Orange book. Describes system security.
- Red book. Interpretation of the Orange book in a network context.
- Blue book. Application of Orange book to systems not covered in the original book.

Windows NT uses the C2 security level It has the following features:

- Object control. Users own certain objects and they have control over how they are accessed.
- User names and passwords.
- No object reuse. Once a user or a group has been deleted, the user and group numerical IDs are not used again. New users or groups are granted a new ID number.
- Security auditing system. This allows the system administrator to trace security aspects,

- such as user login, bad logins, program access, file access, and so on.
- Defined keystroke for system access. In Windows NT, the CNTRL-ALT-DEL keystroke is used by a user to log into the system.

### Password hacking programs

There are many password cracking hacking programs, especially for UNIX and Windows NT, these include:

- Crack. This is a freely available program which can be used on Windows NT or UNIX systems.
- PwDump. This utility allows the user to view the encrypted password file for a local or a remote Windows NT system. These encoded passwords can then be used to hack into the system.
- ScanNT. This is a commercial package for Windows NT.

With PwDump the encoded (hashed) passwords are sent back to user, such as:

```
Administrator:500:EF10EDD3421010325A3B2178::Sys Admin:::  
Fred:501:32FAB36412032188::::Fred Bloggs::
```

These encoded passwords can be used to hack into the system because the login process is as follows:

- Server contacts the clients and sends a random nonsense (nonce) message.
- Client then encrypts the nonce and user account name using the encoded password.
- Client sends the result and the text username to the server.
- Server validates this response using the encoded user password.

Many systems are prone to the man in the middle attack, where a hacker sends a forged challenge to the server. Next, the hacker waits for a real request by same user that is to be used to hack into the system. The hacker then sends a challenge to the user, as if it was the server, and incepts the response. This response can then be sent onto the server, from which the hacker can log in. If the hacker already knows the encoded password then there is no need to wait on a response from the hacked user as it can be impersonated immediately with the encoded password.

#### **P.5.5 Hardware security**

Passwords are a simple method of securing a system. A better method is to use a hardware-restricted system which either bar users from a specific area or even restrict users from login into a system. Typical methods are:

- Smart cards. With this method a user can only gain access to the system after they have inserted their personal smart card into the computer and then enter their PIN code.
- Biometrics. This is a better method than a smart card where a physical feature of the user is scanned. The scanned parameter requires to be unchanging, such as fingerprints or a retina images.

### **P.5.6 Hacker problems**

Once a hacker has entered into a system, there are many methods which can be used to further penetrate into the system, such as:

- Modifying search paths. All systems set up a search path in which the system looks into to find the required executable. For example, in a UNIX system, a typical search path is /bin, /usr/bin, and so on. A hacker can change the search paths for a user and then replace standard programs with ones that have been modified. For example, the hacker could replace the email program for one that sends emails directly to the hacker or any directory listings could be sent to the hacker's screen.
- Modifying shared libraries. As discussed previously.
- Running processor intensive task which slows the system down, this task will be run in the background and will generally not be seen by the user. The hacker can further attack the system by adding the processor intensive task to the system start-up file (such as the rc file on a UNIX system).
- Running network intensive tasks which will slow the network down, and typically slow down all the connected computers. As with the processor intensive task, the networking intensive task can be added to the system start-up file.
- Infecting the system with a virus or worm.

Most PCs have now virus scanners which test the memory and files for viruses and thus virus are easy to detect. A more sinister virus is spread over the Internet, such as the Internet worm which was released on November 1988. This is a program which runs on a computer and creates two threads. A thread in a program is a unit of code that can get a time slice from the operating system to run concurrently with other code units. Each process consists of one or more execution threads that identify the code path flow as it is run on the operating system. This enhances the running of an application by improving throughput and responsiveness. With the worm, the first thread searches for a network connection and when it finds a connection it copies itself to that computer. Next, the worm makes a copy of itself and runs it on the system. Thus a single copy will become two, then four, eight, and so on. This will then continue until the system, and the other connected systems, will be shutdown. The only way to stop the worm is to shutdown all the effected computers at the same time and then restart them.

---

## **P.6 Firewalls**

---

### **P.6.1 Firewalls**

A firewall (or security gateway) protects a network against intrusion from outside sources. They tend to differ in their approach but can be characterized as follows:

- Firewalls which block traffic.
- Firewalls which permit traffic.

They can be split into three main types:

- Network-level firewalls (packet filters). This type of firewalls examines the parameters of the TCP/IP packet to determine if it should dropped or not. This can be done by examining the destination address, the source address, the destination port, the source port, and so on. The firewall must thus contain a list of barred IP addresses or allowable IP addresses. Typically a system manager will determine IP addresses of sites which are barred and add them to the table. Certain port numbers will also be barred, typically TELNET and FTP ports are barred and SMTP is allowed, as this allows mail to be routed into and out of the network, but no remote connections.
- Application-level firewalls. This type of firewall uses an intermediate system (a proxy server) to isolate the local computer from the external network. The local computer communicates with the proxy server, which in turns communicates with the external network, the external computer then communicates with the proxy which in turn communicates with the local computer. The external network never actually communicates directly with the local computer. The proxy server can then be set-up to be limited to certain types of data transfer, such as allowing HTTP (for WWW access), SMTP (for electronic mail), outgoing FTP, but blocking incoming FTP.
- Circuit-level firewalls. A circuit-level firewall is similar to an application-level firewall but it does not bother about the transferred protocol.

### Network-level firewalls

The network-level firewall (or packet filter) is the simplest form of firewall and are also known as screen routers. It basically keeps a record of allowable source and destination IP addresses, and deletes all packets which do not have them. This technique is known as address filtering. The packet filter keeps a separate source and destination table for both directions, that is, into and out of the intranet. This type of method is useful for companies which have geographically spread sites, as the packet filter allows incoming traffic from other friendly sites, but blocks other non-friendly traffic. This is illustrated by Figure P.40.

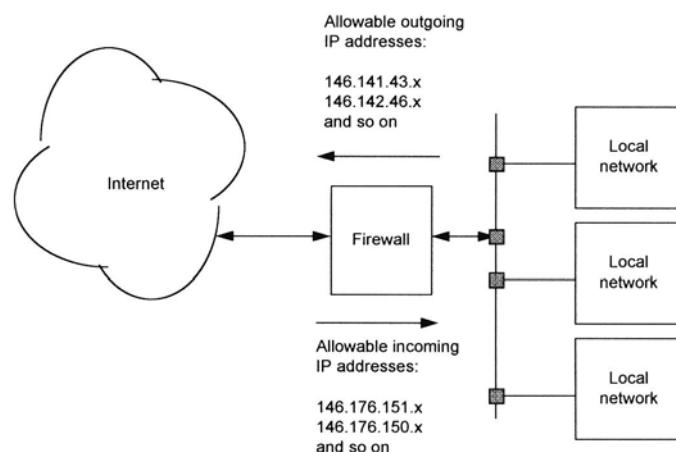
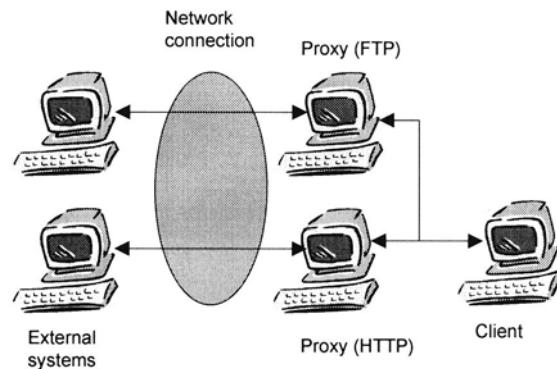


Figure P.40 Packet filter firewalls.

Unfortunately, this method suffers from the fact that IP addresses can be easily forged. For example, a hacker might determine the list of good source addresses and then add one of them to any packets which are addressed to the intranet. This type of attack is known as address spoofing and is the most common method of attacking a network.

### **Application-level firewall**

The application-level firewall uses a proxy server to act as an intermediate system between the external network and the local computer. Normally the proxy only supports a given number of protocols, such as HTTP (for WWW access) or FTP. It is thus possible to block certain types of protocols, typically outgoing FTP (Figure P.41).



**Figure P.41** Application-level firewall.

The proxy server thus isolates the local computer from the external network. The local computer communicates with the proxy server, which in turn communicates with the external network, the external computer then communicates with the proxy, which in turn, communicates with the local computer. The external network never actually communicates directly with the local computer. Figure P.42 shows a WWW browser is set up to communicate with a proxy server to get its access. In the advanced options (Figure P.43) different proxy servers can be specified. In this case for HTTP (WWW access), FTP, Gopher, Secure and Socks (Windows Sockets). It can also be seen that a proxy server can be bypassed by specifying a number of IP addresses (or DNS).

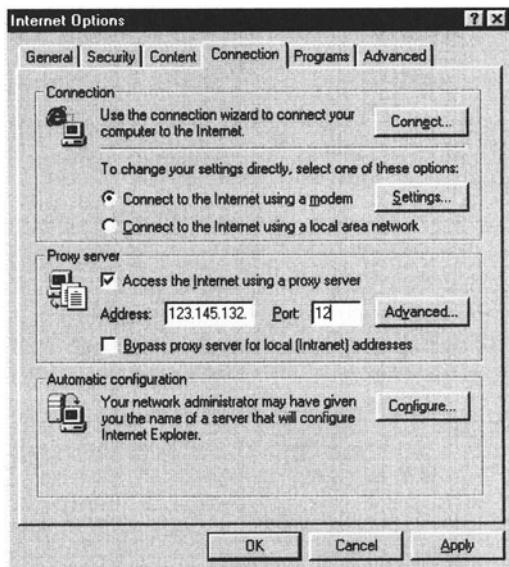


Figure P.42 Internet options showing proxy server selection.

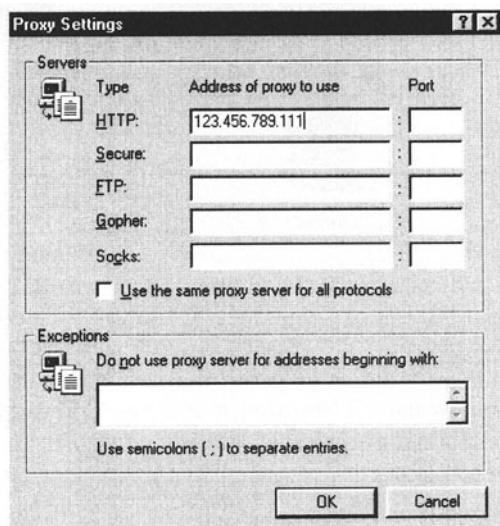


Figure P.43 Proxy settings.

### P.6.2 Firewall architectures

The three main types of firewall are shown in Figure P.44, these are:

- Dual-homed host firewall. With this type of firewall a dedicated computer isolates the local network from the external network (typically, the Internet). It contains two network cards to connect to the two networks and should not run any routing software, and must rely on application-layer routing. This type of firewall is fairly secure and easy to maintain.

- Screen-host firewall. With this method, a router is placed between the external network and the firewall. Users on the local network connect to the firewall and the firewall only communicates with the router. There is thus no direct connection between the external network and the firewall (as all communications must go through the router). The security of this method is better than that of the dual-homed type, as there is no direct electrical connection to the external network.
- Screened-subnet firewall. With this method, a router is placed on either side of the firewall. The firewall is thus isolated from the two connected networks. One router filters traffic between the local network and the firewall, and the other filters traffic between the external network and firewall. This provides the greatest level of security for both incoming and outgoing traffic.

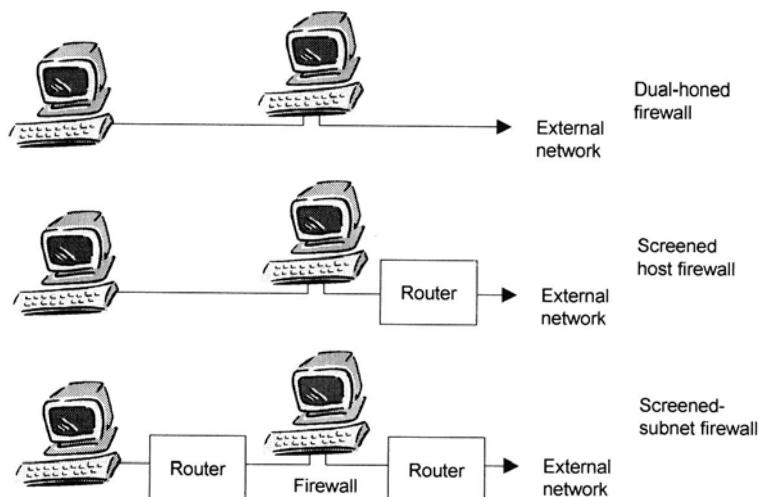


Figure P.44 Firewall types.

### P.6.3 Security ratings

The Orange Book produced by the US Department of Defense (DOD) defines levels of security for systems. It is an important measure of a firewall's security. There are four main divisions, which split into seven main security ratings. Division D is the lowest security level and Division A is the highest. The ratings are:

- Division D. This rating provides no protection on files or for users. For example, a DOS-based computer has no real security on files and users, thus it has a Division D rating.
- Division C. This rating splits into two groups: C1 rating and C2 rating. C1 contains a trust computing base (TCB) which separates users and data. It suffers from the fact that all the data on the system has the same security level. Thus, users cannot make distinctions between highly secure data and not-so-secure data. A C1 system has user names and passwords, as well as some form of control of users and objects. C2 has a higher level of security and provides for some form of accountability and audit. This allows events to be logged and traced, for example, it might contain a list of user logins, network address logins, resource accesses, bad logins, and so on.

- Division B. This rating splits into three groups: B1, B2 and B3. Division B rated systems have all the security of a C2 rating, but have more security because they have a different level of security for all system accesses. For example, each computer can have a different security level, each printer can also have different security levels, and so on. Each object (such as a computer, printer, and so on) has a label associated with it. It is with this label that the security is set by. Non-labeled resources cannot be connected to the system. In a B2 rated system, users are notified of any changes of an object that they are using. The TCB also includes separate operator and administrator functions. In a B3 rated system the TCB excludes information which is not related to security. The system should also be designed to be simple to trace, but also well tested to prevent external hackers. It should also have a full-time administrator, audit trails and system recovery methods.
- Division A. This is the highest level of security. It is similar to B3, but has formal methods for the systems security policy. The system should also have a security manager, who should document the installation of the system, and any changes to the security model.

#### **P.6.4 Application level gateways**

Application-level gateways provide an extra layer of security when connecting an intranet to the Internet. They have three main components:

- A gateway node.
- Two firewalls which connect on either side of the gateway and only transmit packets which are destined for or to the gateway.

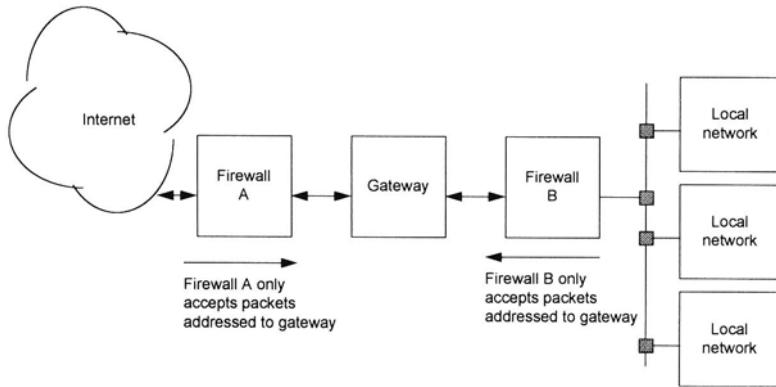
Figure P.45 shows the operation of an application level gateway. In this case, Firewall A discards anything that is not addressed to the gateway node, and discards anything that is not sent by the gateway node. Firewall B, similarly discards anything from the local network that is not addressed to the gateway node, and discards anything that is not sent by the gateway node. Thus, to transfer files from the local network into the global network, the user must do the following:

- Log onto the gateway node.
- Transfer the file onto the gateway.
- Transfer the file from the gateway onto the global network.

To copy a file from the network, an external user must:

- Log onto the gateway node.
- Transfer from the global network onto the gateway.
- Transfer the file from the gateway onto the local network.

A common strategy in organizations is to allow only electronic mail to pass from the Internet to the local network. This specifically disallows file transfer and remote login. Unfortunately, electronic mail can be used to transfer files. To overcome this problem the firewall can be designed specifically to disallow very large electronic mail messages, so it will limit the ability to transfer files. This tends not to be a good method as large files can be split up into small parts then sent individually.



**Figure P.45** Application level gateway.

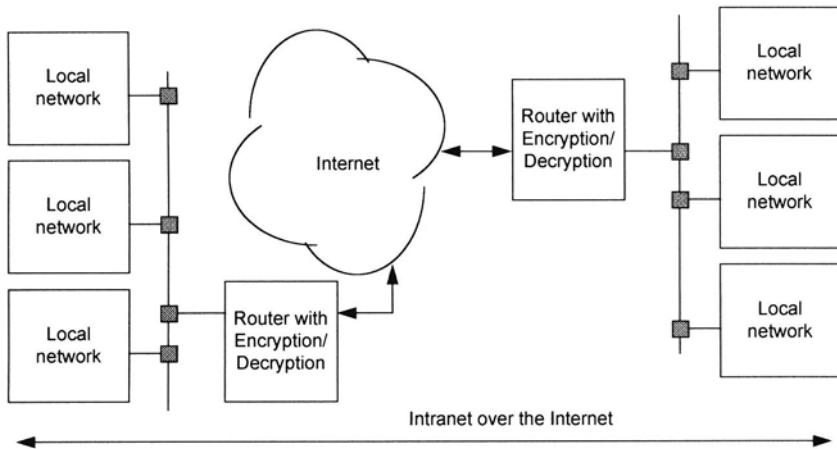
### P.6.5 Encrypted tunnels

Packet filters and application level gateways suffer from insecurity, which can allow non-friendly users into the local network. Packet filters can be tricked with fake IP addresses and application level gateways can be hacked into by determining the password of certain users of the gateway then transferring the files from the network to the firewall, on to the gateway, on to the next firewall and out. The best form of protection for this type of attack is to allow only a limited number of people to transfer files onto the gateway.

The best method of protection is to encrypt the data leaving the network then to decrypt it on the remote site. Only friendly sites will have the required encryption key to receive and send data. This has the extra advantage that the information cannot be easily tapped-into.

Only the routers which connect to the Internet require to encrypt and decrypt, as illustrated in Figure P.46. Typically, remote users connect to a corporation intranet by connecting over a modem which is connected to the corporation intranet, and using a standard Internet connection protocol, such as Point-to-Point Protocol (PPP). This can be expensive in both phone calls or in providing enough modem for all connected users. These costs can be drastically reduced if the user connects to an ISP, as they provide local rate charges. For this a new protocol, called Point-to-Point Tunneling Protocol (PPTP) has been developed to allow remote users connections to intranets from a remote connection (such as from a modem or ISDN). It operates as follows:

- Users connect to an ISP, using a protocol such as Point-to-Point Protocol (PPP) and requests that the information is sent to an intranet. The ISP has special software and hardware to handle PPTP.
- The data sent to the ISP, using PPTP, is encrypted before it is sent into the Internet.
- The ISP sends the encrypted data (wrapped in an IP packet) to the Intranet.
- Data is passed through the firewall, which has the software and hardware to process PPTP packets.
- Next, the user logs in using Password Authentication Protocol (PAP) and Challenge Handshake Authentication (CHAP).
- Finally, the intranet server reads the IP packet and decrypts the data.



**Figure P.46** Encryption tunnels.

#### P.6.6 Filtering routers

Filtering routers run software which allows many different parameters of the incoming and outgoing packets to be examined, such as:

- Source IP address. The router will have a table of acceptable source IP addresses. This will limit the access to the external network as only authorized users will be granted IP addresses. This unfortunately is prone to IP spoofing, where a local user can steal an authorized IP address. Typically, it is done by determining the IP address of a computer and waiting until there is no-one using the computer, then using the unused IP address. Several users have been accused of accessing unauthorized material because other users have used their IP address. A login system which monitors IP addresses and the files that they are accessing over the Internet cannot be used as evidence against the user, as it is easy to steal IP addresses.
- Destination IP address. The router will have a table of acceptable outgoing destination IP addresses, addresses which are not in the table are blocked. Typically, this will be used to limit the range of destination addresses to the connected organizational intranet, or to block certain addresses (such as pornography sites).
- Protocol. The router holds a table of acceptable protocols, such as TCP and/or UDP.
- Source port. The router will have a table of acceptable TCP ports. For example, electronic mail (SMTP) on port 25 could be acceptable, but remote login on port 543 will be blocked.
- Destination port. The router will have a table of acceptable TCP ports. For example, ftp on port 20 could be acceptable, but telnet connections on port 23 will be blocked.
- Rules. Other rules can be added to the system which define a mixture of the above. For example, a range of IP addresses can be allowed to transfer on a certain port, but another range can be blocked for this transfer.

Filter routers are either tightly bound when they are installed and then relaxed, or are relaxed and then bound. The type depends on the type of organization. For example, a financial in-

stitution will have a very strict router which will allow very little traffic, apart from the authorized traffic. The router can be opened-up when the systems have been proved to be secure (they can also be closed quickly when problems occur).

An open organization, such as an education institution will typically have an open system, where users are allowed to access any location on any port, and external users are allowed any access to the internal network. This can then be closed slowly when internal or external users breach the security or access unauthorized information. For example, if a student is access a pornographic site consistently then the IP address for that site could be blocked (this method is basically closing the door after the horse has bolted).

To most users the filtering router is an excellent method of limited traffic access, but to the determined hacker it can be easily breached, as the hacker can fake both IP addresses and also port addresses. It is extremely easy for a hacker to write their own TCP/IP driver software to address whichever IP address, and port numbers that they want.

## **P.7 Authentication**

---

### **P.7.1 Introduction**

It is obviously important to encrypt a transmitted message, but how can it be proved that the message was sent by the user who encrypted the message. This is achieved with message authentication. The two users who are communicating are sometimes known as the principals. It should be assumed that an intruder (hacker) can intercept and listen to messages at any part of the communications, whether it be the initial communication between the two parties and their encryption keys or when the encrypted messages are sent. The intruder could thus play-back any communications between the parties and pretend to be the other.

### **P.7.2 Shared secret-key authentication**

With this approach a secret key,  $K_{12}$  (between Fred and Bert) is used by both users. This would be transmitted through a secure channel, such as a telephone call, personal contact, mail message, and so on. The conversation will then be:

- The initiator (Fred) sends a challenge to the responder (Bert) which is a random number.
- The responder transmits it back using a special algorithm and the secret key. If the initiator receives back the correctly encrypted value then it knows that the responder is allowed to communicate with the user.

The random number should be large enough so that it is not possible for an intruder to listen to the communication and repeat it. There is little chance of the same 128-bit random number occurring within days, months or even years.

This method has validated Bert to Fred, but not Fred to Bert. Thus, Bert needs to know that the person receiving his communications is Fred. Thus Bert initiates the same procedure as before, sending a random number to Fred, who then encrypts it and sends it back. After this has been successfully received by Bert, encrypted communications can begin.

### **P.7.3 Diffie-Hellman key exchange**

In the previous section, a private key was passed over a secure line. The Diffie-Hellman

method allows for keys to be passed electronically. For this, Fred and Bert pick two large prime numbers:

$a$ =Prime Number 1

$b$ =Prime Number 2

where:

$(a-1)/2$  is also prime. The values of  $a$  and  $b$  are public keys. Next, Fred picks a private key ( $c$ ) and Bert picks a private key ( $d$ ). Fred sends the values of:

$$(a, b, b^c \bmod a)$$

Bert then responds by sending:

$$(b^d \bmod a)$$

For example:

$a=43$  (first prime number),  $b=7$  (second prime number),  $c=9$  (Fred's private key),  $d=8$  (Bert's private key). Note, that the value of  $a$  (43) would not be used as  $(a-1)/2$  is not prime (21).

Thus, the values sent by Fred will be:

$$(43, 7, 42)$$

The last value is 42 as  $7^8$  is 40,353,607, and  $40,353,607 \bmod 43$ , is 42. Bert will respond back with:

$$(6)$$

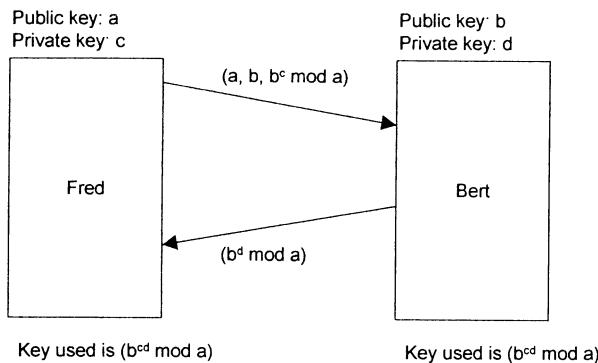
as  $7^8$  is 5,764,801, and  $5,764,801 \bmod 43$  is 6.

Next Fred and Bert will calculate:

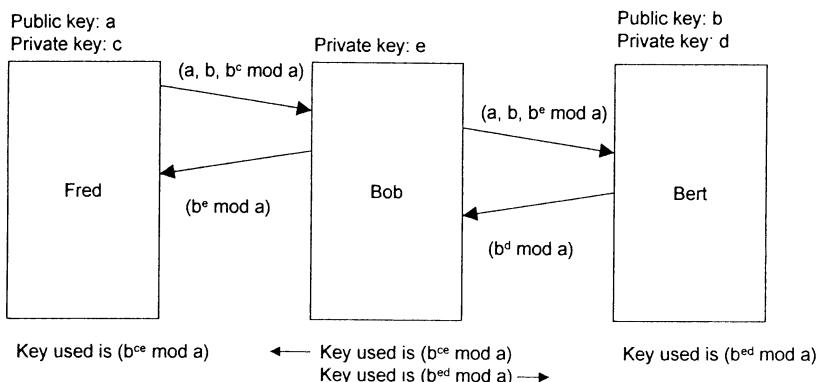
$$b^{cd} \bmod a$$

and both use this as their secret key. Figure P.47 shows an example of the interchange. It is difficult for an intruder to determine the values of  $c$  and  $d$ , when the values of  $a$ ,  $b$ ,  $c$  and  $d$  are large.

Unfortunately, this method suffers from the man-in-the-middle attack, where the intruder intercepts the communications between Fred and Bert. Figure P.48 shows an interceptor (Bob) who has chosen a private key of  $e$ . Thus, Fred thinks he is talking to Bert, and vice-versa, but Bob is attacking as the man-in-the-middle. Bob then uses two different keys when talking with Fred and Bert.



**Figure P.47** Diffie-Hellman key exchange.



**Figure P.48** Man-in-the-middle attack.

#### P.7.4 Key distribution center

The Diffie-Hellman method suffers from the man-in-the-middle attack, it also requires a separate key for each communication channel. A KDC (key distribution center) overcomes these problems with a single key and a secure channel for authentication. In a KDC, the authentication and session keys are managed through the KDC. One method is the wide-mouth protocol, which does the following:

- Fred selects a session key ( $K_{SESS}$ ).
- Fred sends an encrypted message which contains the session key. The message is encrypted with  $K_{KDC1}$ , which is the key that Fred uses to pass messages to and from the KDC.
- The KDC decrypts this encrypted message using the  $K_{KDC1}$  key. It also extracts the session key ( $K_{SESS}$ ). This session key is added to the encrypted message and then encrypted with  $K_{KDC2}$ , which is the key that Bert uses to pass messages to and from the KDC.

This method is relatively secure as there is a separate key used between the transmission between Fred and the KDC, and between Bert and the KDC. These keys are secret to Fred and

the KDC, and between Bert and the KDC. The drawback with the method is that if the intruder determines secret key used for Fred to communicate with the KDC then it is possible to trick the KDC that it is communicating with Fred. As the key is unchanging, the theft of a key may take some time to discover and the possible damage widespread. The intruder can simply choose a new session key each time there is a new session.

### P.7.5 Digital signatures

Digital signatures provide a way of validating an electronic document, in the same way as a hand-written signature does on a document. It must provide:

- Authentication of the sender. This is important as the recipient can verify the sender of the message.
- Authentication of the contents of the message. This is important, as the recipient knows that a third party has not modified the original contents of the message. Normally, this is also time-stamped.
- Authentication that the contents have not been changed by the recipient. This is important in legal cases where the recipient can prove that the message was as the original.

### Secret-key signatures

The secret-key signature involves a user selecting a secret-key which is passed to a central authority, who keeps the key private. When Fred wants to communicate with Bert, he passes the plaintext to the central authority and encrypts it with a secret-key and the time-stamp. The central authority then passes an encrypted message to Bert using the required secret-key. A time-stamp is added to the message that is sent to Bert. This provides for a legal verification of the time the message was sent, and also stops intruders from replaying a transmitted message. The main problem with this method is that the central authority (typically, banks, government departmental or legal professionals) must be trust-worthy and reliable. They can also read all of the transmitted messages.

### Message digests

Public and private-key signatures provide for both authentication and secrecy, but in many cases all that is required is that a text message is sent with the required authentication. A method of producing authentication is message digest, which generates a unique message digest for every message. The most common form of message digest is MD5 (RFC1321, R.Rivest). It is designed to be relatively fast to compute and does not require any large substitution tables. In summary, its operation is:

- It takes as input a message of arbitrary length.
- Produces an 128-bit “fingerprint” (or message digest) of the input.
- It is not possible to produce two message which have the same message digest, or to produce any message from a prespecified target message digest.

### MD5 algorithm

Initially, the message with  $b$  bits is arranged as follows:

$m_0 m_1 m_2 m_3 m_4 m_5 m_6 \dots m_{b-1}$

Next five steps are performed:

- Adding padding bits. The message is padded so that its length is 64 bits less than being a multiple of 512 bits. For example, if the message is 900 bits long, then an extra 60 bits will be added so that it is 64 bits short of 1024 bits. The padded bits are a single ‘1’ bit followed by ‘0’ bits. At least one bit must be added and, at the most, 512 bits are added.
- Append Length. A 64-bit representation of  $b$  (the length of the message before the padding bits were added) is appended to the result of the previous step. The resulting message will thus be a multiple of 512 bits, or:

$m_0 m_1 m_2 m_3 m_4 m_5 m_6 \dots m_{n-1}$

where  $n$  is a multiple of 512.

- MD Buffer initialized. A four-word buffer ( $A, B, C, D$ ) is used to compute the message digest. These are initialized to the following hexadecimal values (low-order bytes first):

A: 01 23 45 67h (0000 0001 0010 ... 0111)    B: 89 ab cd efh  
 C: fe dc ba 98h (1111 1110 1101 ... 1000)    D: 76 54 32 10h

- Message processed in 16-word blocks. Next four auxiliary functions are defined which operate on three 32-bit words and produce a single 32-bit word. These are:

$$F(x, y, z) = X.Y + \overline{X}.Z$$

$$G(x, y, z) = X.Z + Y.\overline{Z}$$

$$H(x, y, z) = X \oplus Y \oplus Z$$

$$I(x, y, z) = Y \oplus (X + \overline{Z})$$

This step also involves a 64-element table  $T[1 \dots 64]$  which is made up of the function.  $T[i]$  is equal to the integer part of  $4,294,967,296$  times  $\text{abs}(\sin(i))$ , where  $i$  is in radians.

The algorithm is as follows:

```

/* Process each 16-word block. */
For i = 0 to N/16-1 do
  /* Copy block i into X. */
  For j = 0 to 15 do
    Set X[j] to M[i*16+j].
  end /* of loop on j */

  /* Save A as AA, B as BB, C as CC, and D as DD. */
  AA = A  BB = B
  CC = C  DD = D

  /* Round 1. */
  /* Let [abcd k s i] denote the operation a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
  /* Do the following 16 operations. */
  [ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4]

```

```

[ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]
[ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]
[ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]

/* Round 2.*/
/* Let [abcd k s i] denote the operation a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */

/* Do the following 16 operations.*/
[ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]
[ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA 4 20 24]
[ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 28]
[ABCD 13 5 29] [DABC 2 9 30] [CDAB 7 14 31] [BCDA 12 20 32]

/* Round 3.*/
/* Let [abcd k s t] denote the operation a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */

/* Do the following 16 operations.*/
[ABCD 5 4 33] [DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]
[ABCD 1 4 37] [DABC 4 11 38] [CDAB 7 16 39] [BCDA 10 23 40]
[ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA 6 23 44]
[ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]

/* Round 4.*/
/* Let [abcd k s t] denote the operation a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations.*/
[ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52]
[ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]
[ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]
[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]

/* Then perform the following additions */

A = A + AA   B = B + BB   C = C + CC   D = D + DD
{
f loop on i */

```

Note that the <<< symbol represent the rotate left operation, where the bits are rotated to the left.

- Output. The message digest is produced from A, B, C and D, where A is the low-order byte and D the high-order byte.

Standard test results give the following message digests:

<i>Message</i>	<i>Message digest</i>
“”	d41d8cd98f00b204e9800998ecf8427e
“a”	0cc175b9c0f1b6a831c399e269772661
“abc”	900150983cd24fb0d6963f7d28e17f72
“abcdefghijklmnopqrstuvwxyz”	f96b697d7cb7938d525a2f31aaaf161d0
“ABCDEFGHIJKLMNOPQRSTUVWXYZ”	c3fcfd3d76192e4007dfb496cca67e13b
YZabcdefghijklmnopqrstuvwxyz01234567890”	
“12345678901234567890123456789012345678901234567890”	57edf4a22be3c955ac49da2e2107b67a

## P.8 ATM

---

### P.8.1 ATM signaling and call set-up

ATM, as with most telecommunications systems, uses a single-pass approach to setting up a connection. Initially, the source connection (the source end-system) communicates a connection request to the destination connection (the destination end-point). The routing protocol manages the routing of the connection request and all subsequent data flow. The call is established with:

- A set-up message. This is initially sent, across the UNI, to the first ATM switch. It contains:
  - Destination end-system address.
  - Desired traffic.
  - Quality of service.
  - Information Elements (IE) defining particular desired higher layer protocol bindings and so on.
- The initial ATM switch sends back a local call proceeding acknowledgement to the source end-system.
- The initial ATM switch invokes an ATM routing protocol, and propagates a signaling request across the network, it finally reaches the ATM switch connected to the destination end-system.
- The destination ATM switch connected to the destination end-system forwards the set-up message to the end-system, across its UNI.
- The destination end-system either accepts or rejects the connection. If necessary, the destination can negotiate the connection parameters. If the destination end-system rejects the connection request, it returns a release message. This is also sent back to the source end-system and clears the connection, such as clearing any allocated VCI labels. A release message can also be used by any of the end-systems, or by the network, to clear an established connection.
- If the destination end-system accepts the call then the ATM switch, which connects to it, returns a connect message through the network, along the same path.
- When the source end-system receives and acknowledges the connect message, either node can then start transmitting data on the connection.

### P.8.2 ATM addressing scheme

ATM, like any other network, requires a network address scheme which identifies the source and destination addresses. The ITU-T have developed a standardized, telephone-like, numbering system called E.164 for addressing public ATM networks. Unfortunately, E.164 addresses are public addresses and cannot typically be used within private networks. The ATM Forum have since extended ATM addressing to include private networks. For a private networking scheme in UNI 3.0/3.1 they evaluated two different models. These basically differ in the way that the ATM protocol layer is viewed in relation to existing protocol layers, such as IP and IPX layers, and are:

- Peer model. This model treats the ATM layer as a peer of existing network layers and uses the same addressing schemes within the ATM networks. Thus, ATM endpoints would be identified by their existing network layer address (such as an IP or an IPX address). The ATM signaling requests would then carry these addresses for the source and destination ATM switch. Also network layer routing protocols, such as RIP, and so on, can be used to route ATM signaling requests using existing network layer addresses. The peer model allows for simplified addressing.
- Overlay model. This model decouples the ATM layer from any existing network protocol and defines a new addressing structure and a new routing protocol. This, as with Ethernet, FDDI and Token Ring which use MAC addresses, allow all existing protocols to operate over an ATM network. For this reason, the model is known as the subnetwork or overlay model. Thus, all ATM switches need an ATM address, and possibly, also a network layer address (such as an IP or IPX address).

The disadvantage with the overlay model is that there needs to be an ATM address resolution protocol which maps network addresses (IP or IPX). The peer model does not need address resolution protocols, and, because it uses existing routing protocols.

The ATM Forum decided to implement the overlay model for UNI 3.0/3.1 signaling. This is mainly the peer model would be difficult to implement as they must essentially act as multiprotocol routers and support address tables for all current protocols, as well as all of their existing routing protocols. In addition, currently available routing protocols for LANs and WANs do not map well into the QoS parameter.

The ATM Forum chose a private network addressing scheme based on the OSI Network Service Access Point (NSAP) address. They are not true NSAP addresses, and are either ATM private network addresses or ATM end-point identifiers. They basically are subnetwork points of attachment.

An NSAP ATM address for a private network has 20 bytes, while a public network uses a E.164 address, as defined by the ITU-T. NSAP-based addresses have three main fields:

- Authority and Format Identifier (AFI). This defines type and format of the Initial Domain Identifier (IDI).
- Initial Domain Identifier (IDI). This defines the address allocation and administration authority.
- Domain Specific Part (DSP). This defines the actual routing information.

There are three formats that private ATM addressing use for different definitions for the AFI and IDI parts. These are:

- NSAP Encoded E.164 format. The E.164 number is contained in the IDI.
- DCC Format. The IDI is a Data Country Code (DCC) which identifies the country, as specified in ISO 3166. These addresses are administered, in each country, by the ISO National Member Body.
- ICD Format. The IDI contains the International Code Designator (ICD). The ICD is allocated by the ISO 6523 registration authority. They identify particular international organizations.

## NSAP

NSAP is defined in ISO/IEC 8348. It divides the address into two main parts: Initial Domain Part (IDP), which splits into the Authority and Format Identifier (AFI), and Initial Domain Identifier (IDI). The format is thus:

IDP	DSP
AFI   IDI	

In the ISO/IEC 10589 specification, the DSP address includes an ID and SEL (1 byte selector) field which are used by level 1 routing. Typically, the ID part is taken from the ISO/IEC 8802 48-bit MAC address. The format is thus:

IDP	DSP	
AFI   IDI		ID   SEL

In the UNI-3.1 specification, the ID field is six bytes. It is also defines that the NSAP address format uses a maximum length of 20 bytes.

## ICD Format

The format of an ICD scheme is:

AFI	47	1 byte
ICI	xxxx	2 bytes
Version	xx	1 byte
Network	xxxxxx	3 bytes
Tele traffic area	xx	1 byte
Member identifier	xxxx	2 bytes
Member access point	xx	1 byte
Area	xx	1 byte
Switch	xx	1 byte
MAC-address	xxxxxxxxxxxx	6 bytes
Nselector	xx	1 byte

An example format is:

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|47|00 23|00|00 00 03|xx|xx xx|xx|xx|xx|xx| ESI MAC address |xx|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

## DCC Format

The DCC format is defined by the National Standards Organization 39528+1100. Its fields include:

AFI (39 or 38)	ISO DCC format	(1 byte)
IDI		(2 bytes)
CFI	Country Format Identifier	(4 bits)
CDI	Country Domain Identifier	(12 bits)
SFI	SURFNet Format Identifier	(4 bits)
	0=CLN S in case of an organizational SDI	
	1=in case of a network SDI	
	2=ATM in case of an organization SDI	
SDI	SURFNet Domain Identifier	(2 bytes)
	decimal encoded administrative numbers	
ASDI	Additional Domain Identifier	(4 bits)

NYU	Not yet used	(5 bytes)
ESI	End System Identifier	(6 bytes)
SEL	Selector	(1 byte)

The EaStMAN network uses a 13 byte prefix, in the form:

39.826f.1107.16.7000.00.nnmm.ee.ff

where 39 is the AFI – ISO DCC, 826f is the IDI (indicating the UK), 1 is the CFI, 107 is the CDI, 1 for the SFI, 107 the SDI (country and domain), 16 for ASNI (for region), 00 (not yet used), nnmm for site code, ee for campus number and ff for switch number.

The nn part represents the institution, these are:

01	University of Edinburgh	02	Moray House
03	Queen Margaret College	04	Napier University
05	Heriot-Watt University	06	Edinburgh College of Art
07	University of Stirling		

and mm is the ring access point. These are assigned in a clockwise direction on the ring, starting from Kings Building (University of Edinburgh). In summary, the nnmm codes are assigned as follows:

University of Edinburgh:

Kings Buildings	0101	Pollock Halls	0102
Old College	0103	New College	0106

Moray House:

MH-H	0204	MH-Cramond	0209
------	------	------------	------

Queen Margaret College:

QMC-Leith	0305	QMC-Corstorphine	030a
-----------	------	------------------	------

Napier University:

Merchiston	0407	Sighthill	040b
------------	------	-----------	------

Heriot-Watt:

Riccarton campus	0508
------------------	------

Edinburgh College of Art:

ECA-L	060c	ECA-G	060d
-------	------	-------	------

University of Stirling:

Stirling 070e

The last two byte values (ee and ff) are allocated by the local institution. Typically, they can be used to identify the campus number (ee) and the switch number within the campus (ff).

#### **E.164 Format (ATM Forum/95-0427R1)**

The E.164 format provides a geographical scheme, but, as it is derived from the telephone system, the addresses are in short supply. Therefore, the E.164 NSAP format is recommended, which is extensible to ISDN. Its format is:

```
+-----+-----+-----+-----+-----+-----+-----+-----+
|45| Internat E.164 number | HO-DSP | xx xx xx xx xx xx |xx|
+-----+-----+-----+-----+-----+-----+-----+-----+
```

Where:

45	AFI for E.164 binary syntax
IDI	International E.164 number
HO-DSP	Extends the E.164 address to logically identify many devices in a single geographical location.
ESI	Similar to HO-DSP, extends the E.164 address.
SEL	Selector

---

## **P.9 Gigabit Ethernet**

The IEEE 802.3 working group initiated the 802.3z Gigabit Ethernet task force to create the Gigabit Ethernet standard (which was finally defined in 1998). The Gigabit Ethernet Alliance (GEA) was founded in May 1996 and promotes Gigabit Ethernet collaboration between organizations. Companies, which initially were involved in the GEA, include: 3Com, Bay Networks, Cisco Systems, Compaq, Intel, LSI Logic, Sun and VLSI.

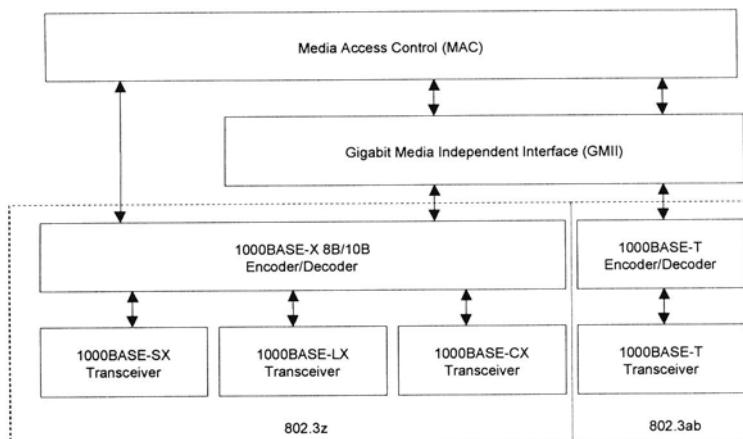
The amount of available bandwidth for a single segment is massive. For example, almost 125 million characters (125MB) can be sent in a single second. For example, a large reference book with over 1000 pages could be sent over a network segment, ten times in a single second. Compare it also with a  $\times 24$ , CD-ROM drive which transmits at a maximum rate of 3.6MB/s ( $24 \times 150\text{ kB/sec}$ ). Gigabit Ethernet operates almost 35 times faster than this drive. With network switches, this bandwidth can be multiplied a given factor, as they allow multiple simultaneous connections.

Gigabit Ethernet is excellent challenger for network backbones as it interconnects 10/100BASE-T switches, and also provides a high-bandwidth to high-performance servers. Initial aims were:

- Half/full-duplex operation at 1000Mbps.
- Standard 802.3 Ethernet frame format. Gigabit Ethernet uses the same variable-length frame (64- to 1514-byte packets), and thus allows for easy upgrades.
- Standard CSMA/CD access method.
- Compatiblity with existing 10BASE-T and 100BASE-T technologies.
- Development of an optional Gigabit Media Independent Interface (GMII).

The compatibility with existing 10/100BASE standards make the upgrading to Gigabit Ethernet much easier, and considerably less risky than changing to other networking types, such as FDDI and ATM. It will happily interconnect with, and autosense, existing slower rated Ethernet devices. Figure P.49 illustrates the functional elements of Gigabit Ethernet, its main characteristics are:

- Full-duplex communication. As defined by the IEEE 802.3x specification, two nodes connected via a full-duplex, switched path can simultaneously send and receive frames. Gigabit Ethernet supports new full-duplex operating modes for switch-to-switch and switch-to-end-station connections, and half-duplex operating modes for shared connections using repeaters and the CSMA/CD access method.
- Standard flow control. Gigabit Ethernet uses standard Ethernet flow control to avoid congestion and overloading. When operating in half-duplex mode, Gigabit Ethernet adopts the same fundamental CSMA/CD access method to resolve contention for the shared media.
- Enhanced CSMA/CD method. This maintains a 200m collision diameter at gigabit speeds. Without this, small Ethernet packets could complete their transmission before the transmitting node could sense a collision, thereby violating the CSMA/CD method. To resolve this issue, both the minimum CSMA/CD carrier time and the Ethernet slot time (the time, measured in bits, required for a node to detect a collision) have been extended from 64 bytes (which is  $51.2\mu s$  for 10BASE and  $5.12\mu s$  for 100BASE) to 512 bytes (which is  $4.1\mu s$  for 1000BASE). The minimum frame length is still 64 bytes. Thus, frames smaller than 512 bytes have a new carrier extension field following the CRC field. Packets larger than 512 bytes are not extended.
- Packet bursting. The slot time changes effect the small-packet performance, but this has been offset by a new enhancement to the CSMA/CD algorithm, called packet bursting. This allows servers, switches and other devices to send bursts of small packets in order to fully utilize the bandwidth.



**Figure P.49** Gigabit Ethernet functional elements.

Devices operating in full-duplex mode (such as switches and buffered distributors) are not subject to the carrier extension, slot time extension or packet bursting changes. Full-duplex devices use the regular Ethernet 96-bit interframe gap (IFG) and 64-byte minimum frame size.

### **P.9.1 Ethernet transceiver**

The IEEE 802.3z task force spent much of their time defining the Gigabit Ethernet standard for the transceiver (physical layer), which is responsible for the mechanical, electrical and procedural characteristics for establishing, maintaining and deactivating the physical link between network devices. The physical layers are:

- 1000BASE-SX (Low cost, multi-mode fiber cables). These can be used for short interconnections and short backbone networks. The IEEE 802.3z task force have tried to integrate the new standard with existing cabling, whether it be twisted-pair cable, coaxial cable or fiber optic cable. These tests involved firing lasers in long lengths of multi-mode fiber cables. Through these test it was found that a jitter component results which is caused by a phenomenon known as differential mode delay (DMD). The 1000BASE-SX standard has resolved this by defining the launch of the laser signal, and enhanced conformance tests. Typical lengths are: 62.5 µm, multi-mode fiber (up to 220m). 50µm, multi-mode fiber (550m).
- 1000BASE-LX (Multi-mode/single mode-mode fiber cables). These can be used for longer runs, such as on backbones and campus networks. Single-mode fibers are covered by the long-wavelength standard, and provide for greater distances. External patch cords are used to reduce DMD. Typical lengths are: 62.5 µm, multi-mode fiber (up to 550m), 50µm, multi-mode fiber (up to 550m). 50µm, single-mode fiber (up to 5km).
- 1000BASE-CX (Shielded Balanced Copper). This standard supports interconnection of equipment using a copper-based cable, typically up to 25m. As with the upper two standard, it uses the Fiber Channel-based 8B/10B coding at the serial line rate of 1.25Gbps. The 1000BASE-T is likely to supersede this standard, but it has been relatively easy to define, and to implement.
- 1000BASE-T (UTP). This is a useful standard for connecting directly to workstations. The 802.3ab Task Force has been assigned the task of defining the 1000BASE-T physical layer standard for Gigabit Ethernet over four pairs of Cat-5 UTP cable, for cable distances of up to 100m, or networks with a diameter of 200m. As it can be used with existing cabling, it allows easy upgrades. Unfortunately, it requires new technology and new coding schemes in order to meet the potentially difficult and demanding parameters set by the previous Ethernet and Fast Ethernet standards.

### **P.9.2 Fiber Channel Components**

The IEEE 802.3 committee based much of the physical layer technology on the ANSI-backed X3.230 Fiber Channel project. This allowed many manufacturers to re-use physical-layer Fiber Channel components for new Gigabit Ethernet designs, and has allowed a faster development time than is normal, and increased the volume production of the components. These include optical components and high-speed 8B/10B encoders.

The 1000BASE-T standard uses enhanced DSP (Digital Signal Processing) and enhanced silicon technology to enable Gigabit Ethernet over UTP cabling. As Figure P.49 shows, it does not use the 8B/10B encoding.

### P.9.3 *Buffered distributors*

Along with repeaters, bridges and switches, a new device, called a buffered distributor (or full-duplex repeater), has been developed for Gigabit Ethernet. It is a full-duplex, multiport, hub-like device that connects two or more Gigabit Ethernet segments. Unlike a bridge, and like a repeater, it forwards all the Ethernet frames from one segment to the others, but unlike a standard repeater, a buffered distributor buffers one, or more, incoming frames on each link before forwarding them. This reduces collisions on connected segments. The maximum bandwidth for a buffered distributor will still only be 1Gbps, as opposed to Gigabit switches which allow multi-gigabit bandwidths.

### P.9.4 *Quality of Service*

Many, real-time, networked applications require a given Quality of Server (QoS), which might relate to bandwidth requirements, latency (network delays) and jitter. Unfortunately, there is nothing built-into Ethernet that allows for a QoS, thus new techniques have been developed to overcome this. These include:

- RSVP. Allows nodes to request and guarantee a QoS, and works at a higher-level to Ethernet. For this, each network component in the chain must support RSVP and communicate appropriately. Unfortunately, this may require an extensive investment to totally support RSVP, thus many vendors have responded in implementing proprietary schemes, which may make parts of the network vendor-specific.
- IEEE 802.1p and IEEE 802.1Q. Allows a QoS over Ethernet by “tagging” packets with an indication of the priority or class of service desired for the frames. These tags allow applications to communicate the priority of frames to internetworking devices. RSVP support can be achieved by mapping RSVP sessions into 802.1p service classes.
- Routing. Implemented at a higher layer.

### P.9.5 *Gigabit Ethernet migration*

The greatest advantage of Gigabit Ethernet is that it is easy to upgrade existing Ethernet-based networks to higher bit rates. Typical migration might be:

- Switch-to-switch links. Involves upgrading the connections between switches to 1Gbps. As 1000BASE switches support both 100BASE and 1000BASE then not all the switches require to be upgraded at the same time, this allows for gradual migration.
- Switch-to-Server Links. Involves upgrading the connection between a switch and the server to 1Gbps. The server requires an upgraded Gigabit Ethernet interface card.
- Switched Fast Ethernet Backbone. Involves upgrading a Fast Ethernet backbone switch to a 100/1000BASE switch. If this supports both 100BASE and 1000BASE switching, using existing cabling.
- Shared FDDI Backbone. Involves replacing FDDI attachments on the ring with Gigabit Ethernet switches or repeaters. The Gigabit uses the existing fiber-optic cable, and provides a greatly increased segment bandwidth.
- Upgrade NICs on nodes to 1Gbps. It is unlikely that users will require 1Gbps connections, but this facility is possible.

### P.9.6 *1000BASE-T*

One of the biggest challenges of Gigabit Ethernet is to use existing Cat-5 cables, as this will

allow fast upgrades. Two critical parameters, which are negligible at 10BASE speeds, are:

- Return loss. Defines the amount of signal energy that is reflected back towards the transmitter due to impedance mismatches in the link (typically from connector and cable bends).
- Far-End Crosstalk. Noise that is leaked from another cable pair.

The 1000BASE-T Task Force estimates that less than 10% of the existing Cat-5 cable was improperly installed (as defined in ANSI/TIA/EIA568-A in 1995) and might not support 1000BASE-T (or even, 100BASE-TX). 100BASE-T uses two pairs, one for transmit and one for receive, and transmits at a symbol rate of 125Mbaud with a 3-level code. 1000BASE-T uses:

- All four pairs with a symbol rate of 125 Mbaud (symbols/sec). One symbol contains two bits of information.
- Each transmitted pulse uses a 5-level PAM (Pulse Amplitude Modulation) line code, which allows two bits to be transmitted at a time.
- Simultaneous sends and receives on each pair. Each connection uses a hybrid circuit to split the send and receive signals.
- Pulse shaping. Matches the characteristics of the transmitted signal to the channel so that the signal-to-noise ratio is minimized. It effectively reduces low frequency terms (which contain little data information, can cause distortion and cannot be passed over the transformer-coupled hybrid circuit), reduces high-frequency terms (which increases crosstalk) and rejects any external high-frequency noise. It is thought that the transmitted signal spectrum for 1000BASE will be similar to 100BASE.
- Forward Error Correction (FEC). This provides a second level of coding that helps to recover the transmitted symbols in the presence of high noise and crosstalk. The FEC bit uses the fifth level of the 5-level PAM.

A 5-level code (-2, -1, 0, +1, +2) allows two bits to be sent at a time, if all four pairs are used then 8 bits are sent at a time. If each pair transmits at a rate of 125Mbaud (symbols/sec), the resulting bit rate will be 1Gbps.

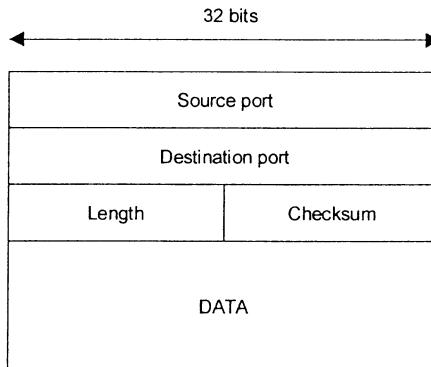
## P.10 UDP

---

TCP allows for a reliable connection-based transfer of data. The User Datagram Protocol (UDP) is an unreliable connection-less approach, where datagrams are sent into the network without an acknowledgements or connections. It is defined in RFC 768 and uses IP as its underlying protocol. It has the advantage of TCP in that it has a minimal protocol mechanism, but does not guarantee delivery of any of the data. Figure P.50 shows its format. The fields are:

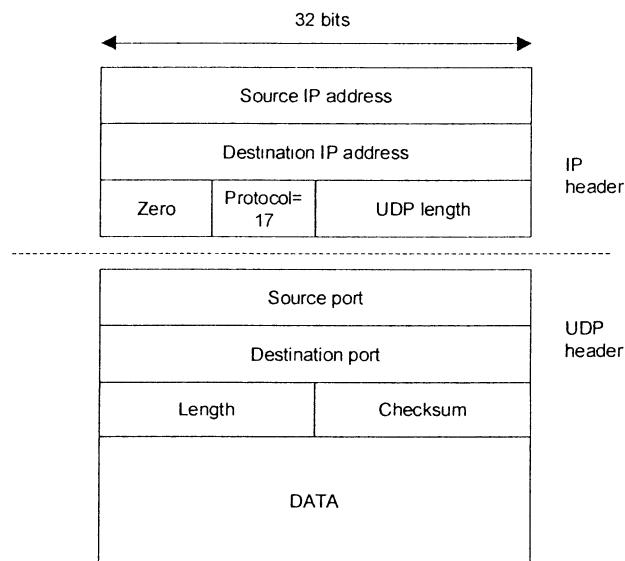
- Source port. This is an optional field is set to a zero if not used. It identifies the local port number which should be used when the destination host requires to contact the originator
- Destination Port to connect to on the destination.

- Length. Number of bytes in the datagram, including the UDP header and the data.
- Checksum. The 16-bit 1's complement of the 1's complement sum of the IP header, the UDP header, the data (which, if necessary, is padded with zero bytes at the end, to make an even number of bytes).



**Figure P.50** UDP header format.

When used with IP the UDP/IP header is shown in Figure P.51. The Protocol field is set to 17 to identify UDP.



**Figure P.51** UDP/IP header format.

## P.11 TCP specification

---

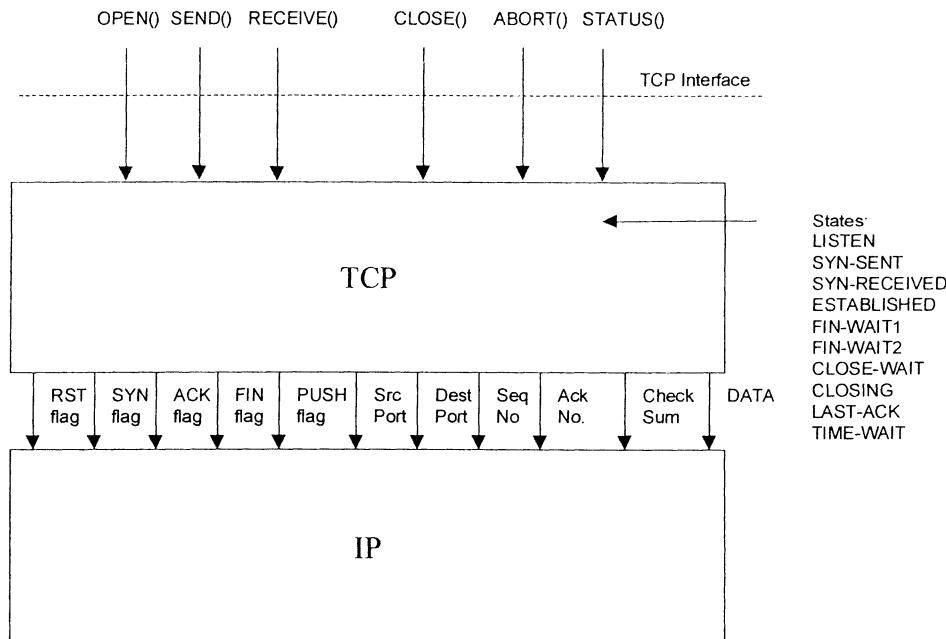
TCP is made reliable with the following:

- Sequence numbers. Each TCP packet is sent with a sequence number. Theoretically, each data byte is assigned a sequence number. This sequence number of the first data byte in the segment is transmitted with that segment and is called the segment sequence number (SSN).
- Acknowledgements. Packets contain an acknowledgement number, which is the sequence number of the next expected transmitted data byte in the reverse direction. On sending, a host puts stores the transmitted data in a storage buffer, and starts a time. If the packet is acknowledged then this data is deleted, else, if no acknowledgement is received before the timer runs out, the packet is retransmitted.
- Window. With this, a host sends a window value which specifies the number of bytes, starting with the acknowledgement number, that the host can receive.

### P.11.1 Connection establishment, clearing and data transmission

The main interfaces in TCP are shown in Figure P.52. The calls from the application program to TCP include:

- OPEN and CLOSE. To open and close a connection.
- SEND and RECEIVE. To send and receive.
- STATUS. To receive status information.



**Figure P.52** TCP interface.

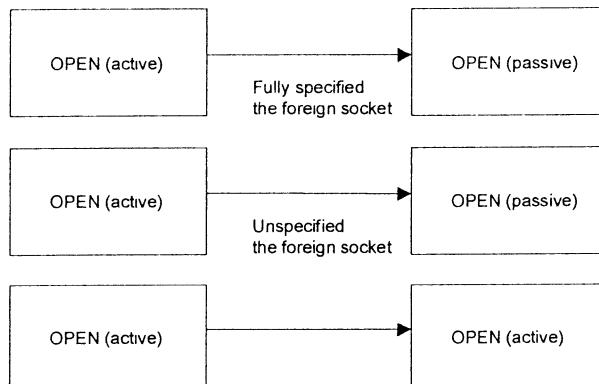
The OPEN call initiates a connection with a local port and foreign socket arguments. A Transmission Control Block (TCB) stores the information on the connection. After a successful connection, TCP adds a local connection name by which the application program refers to the connection in subsequent calls.

The OPEN call supports two different types of call, as illustrated in Figure P.53, these are:

- Passive OPEN. TCP waits for a connection from a foreign host, such as from an active OPEN. In this case, the foreign socket is defined by a zero. This is typically used by servers, such as TELNET and FTP servers. The connection can either be from a fully specified or an unspecified socket.
- Active OPEN. TCP actively connects to a foreign host, typically a server (which is opened with a passive OPEN). Two processes which issue active OPENs to each other, at the same time, will also be connected.

A connect is established with the transmission of TCP packets with the SYN control flag set and uses a three-way handshake. A connection is cleared by the exchange of packets with the FIN control flag set. Data flows in a stream using the SEND call to send data and RECEIVE to receive data.

The PUSH flag is used to send data in the SEND immediately to the recipient. This is required as a sending TCP is allowed to collect data from the sending application program and sends the data in segments when convenient. Thus, the push flag forces it to be sent. When the receiving TCP sees the PUSH flag, it does not wait for any more data from the sending TCP before passing the data to the receiving process.



**Figure P.53** TCP connections.

#### P.11.2 TCB parameters

Table P.2 outlines the send and receive packet parameters, as well as the current segment parameter, which are stored in the TCB. Along with this, the local and remote port number require to be stored.

**Table P.1** TCB parameters.

Send Sequence Variables	Receive Sequence Variables	Current Packet Variable
SND.UNA Send unacknowledged	RCV.NXT Receive next	SEG.SEQ segment sequence number
SND.NXT Send next	RCV.WND Receive window	SEG.ACK segment acknowledgement number
SND.WND Send window	RCV.UP Receive urgent pointer	IRS Initial receive sequence number
SND.UP Send urgent pointer		SEG.LEN segment length
SND.WL1 Segment sequence number used for last window update		SEG.WND segment window
SND.WL2 Segment acknowledgement number used for last window update		SEG.UP segment urgent pointer
ISS Initial send sequence number		SEG.PRC segment precedence value

### **P.11.3 Connection states**

Figure P.54 outlines the states in which the connection goes into, and the events which cause them. The events from applications programs are: OPEN, SEND, RECEIVE, CLOSE, ABORT, and STATUS, and the events from the incoming TCP packets include the SYN, ACK, RST and FIN flags. The definition of each of the connection states are:

- LISTEN. This is the state in which TCP is waiting for a remote connection on a given port.
- SYN-SENT. This is the state where TCP is waiting for a matching connection request after it has sent a connection request.
- SYN-RECEIVED. This is the state where TCP is waiting for a confirming connection request acknowledgement after having both received and sent a connection request.
- ESTABLISHED. This is the state that represents an open connection. Any data received can be delivered to the application program. This is the normal state after for data to be transmitted.
- FIN-WAIT-1. This is the state in which TCP is waiting for a connection termination request, or an acknowledgement of a connection termination, from the remote TCP.
- FIN-WAIT-2. This is the state in which TCP is waiting for a connection termination request from the remote TCP.
- CLOSE-WAIT. This is the state where TCP is waiting for a connection termination request from the local application.
- CLOSING. This is the state where TCP is waiting for a connection termination request acknowledgement from the remote TCP.
- LAST-ACK. This is the state where TCP is waiting for an acknowledgement of the connection termination request previously sent to the remote TCP.
- TIME-WAIT. This is the state in which TCP is waiting for enough time to pass to be sure the remote TCP received the acknowledgement of its connection termination request.
- CLOSED. This is the fictional state, which occurs after the connection has been closed.

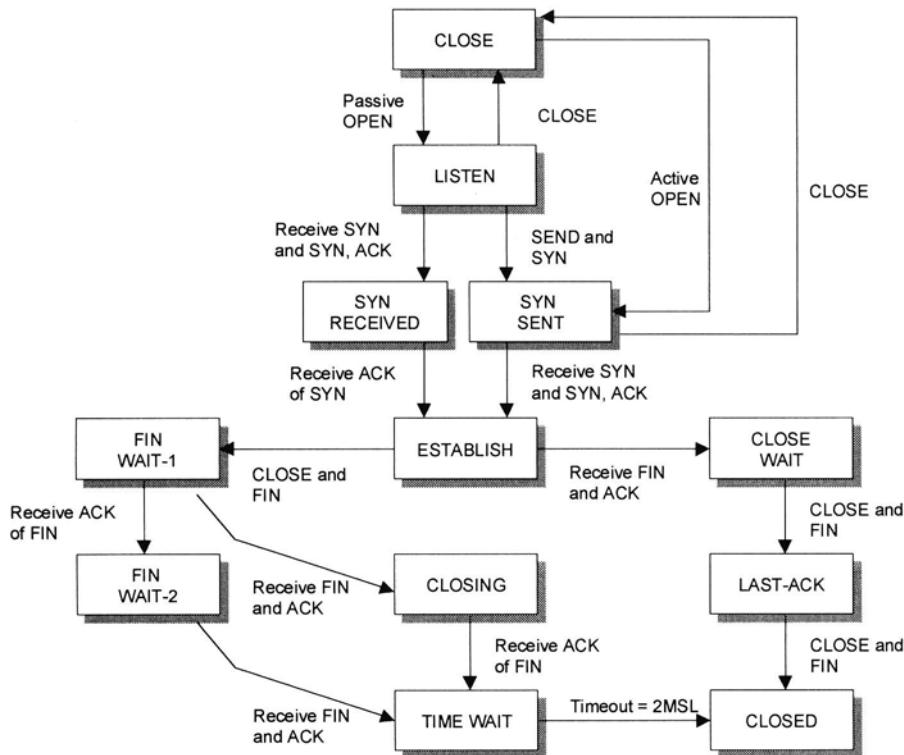


Figure P.54 TCP connection states.

### Sequence Numbers

TCP packets contain a 32-bit sequence number (0 to 4,294,967,295), which relate to every byte sent. It uses a cumulative acknowledgement scheme, where an acknowledgement with a value of VAL, validates all bytes up to, but not including, byte VAL. The number of bytes which the packet starts at the first data byte and are then numbered consecutively.

When sending data, TCP should receive acknowledgements for the transmitted data. The required TCB parameters will be:

SND.UNA	Oldest unacknowledged sequence number.
SND.NXT	Next sequence number to send.
SEG.ACK	Acknowledgement from the receiving TCP (next sequence number expected by the receiving TCP).
SEG.SEQ	First sequence number of a segment.
SEG.LEN	Number of bytes in the TCP packet.
SEG.SEQ+SEG.LEN-1	Last sequence number of a segment.

One receiving data, the following TCB parameters are required:

RCV.NXT	Next sequence number expected on an incoming segments, and is the left or lower edge of the receive window.
---------	---

RCV.NXT+RCV.WND-1	Last sequence number expected on an incoming segment, and is the right or upper edge of the receive window.
SEG.SEQ	First sequence number occupied by the incoming segment.
SEG.SEQ+SEG.LEN-1	Last sequence number occupied by the incoming segment.

### ISN selection

The Initial Sequence Number (ISN) is selected so that previous sockets are not confused with new sockets. Typically this can happen when a host application crashes and then quickly re-establishes the connection before the other side can time-out the connection. To avoid this a 32-bit initial sequence number (ISN) generator is created when the connection is made. It is generated by a 32-bit clock, which is incremented approximately every  $4\mu s$  (giving a ISN cycle of 4.55 hours). Thus within 4.55 hours, each ISN will be unique.

As each connection has a send and receive sequence number, these is an initial send sequence number (ISS) and an initial receive sequence number (IRS). When establishing a connection, the two TCPs synchronize their initial sequence numbers. This is done by exchanging connection establishing packets, with the SYN bit set and with the initial sequence numbers (these packets are typically called SYNs). Thus four packets must be initially exchanged, these are:

- A sends to B. SYN with  $A_{SEQ}$ .
- B sends to A. ACK of the sequence number ( $A_{SEQ}$ ).
- B sends to A. SYN with  $B_{SEQ}$ .
- A sends to B. ACK of the sequence number ( $B_{SEQ}$ ).

Note that the two intermediate steps can be combined into a single message. This is sometimes known as a three-way handshake. This handshake is necessary as the sequence numbers are not tied to a global clock, only to local clocks, and has many advantages, including the fact that old packets will be discarded as they occurred in a previous time.

To make sure that a sequence number is not duplicated, a host must wait for a maximum segment lifetime (MSL) before starting to retransmit packets (segments) after start-up or when recovering from a crash. An example MSL is 2 minutes. Although, if it is recovering, and it has a memory of the previous sequence numbers, it may not need to wait for the MSL, as it can use sequence numbers which are much greater than the previously used sequence numbers.

#### **P.11.4 Opening and closing a connection**

Figure P.55 shows a basic three-way handshake. The steps are:

1. The initial state on the initiator is CLOSED and, on the recipient, it is LISTEN (the recipient is waiting for a connection).
2. The initiator goes into the SYN-SENT state and sends a packet with the SYN bit set and the indicates that the starting sequence number will be 999 (the current sequence number, thus the next number sent will be 1000). When this is received the recipient goes into the SYN-RECEIVED state.
3. The recipient sends back a TCP packet with the SYN and ACK bits set (which identifies that it is a SYN packet and also that it is acknowledging the previous SYN packet). In this case, the recipient tells the originator that it will start transmitting at a sequence

number of 100. The acknowledgement number is 1000, which is the sequence number that the recipient expects to receive next. When this is received, the originator goes into the ESTABLISHED state.

4. The originator sends back a TCP packet with the SYN and ACK bits set and the acknowledgement number is 101, which is the sequence number it expects to see next.
5. The originator transmits data with the sequence number of 1000.

<b>Originator</b>		<b>Recipient</b>
1. CLOSED		LISTEN
2. SYN-SENT → <SEQ=999><CTL=SYN>		SYN-RECEIVED
3. ESTABLISHED <SEQ=100><ACK=1000> <CTL=SYN,ACK> ←		SYN-RECEIVED
4. ESTABLISHED → <SEQ=1000><ACK=101> <CTL=ACK>		ESTABLISHED
5. ESTABLISHED → <SEQ=1000><ACK=101> <CTL=ACK><DATA>		ESTABLISHED

**Figure P.55** TCP connection.

Note that the acknowledgement number acknowledges every sequence number up to, but not including the acknowledgement number.

Figure P.56 shows how the three-way handshake prevents old duplicate connection initiations from causing confusion. In state 3, a duplicate SYN has received, which is from a previous connection. The Recipient sends back an acknowledgement for this (4), but when this is received by the Originator, the Originator sends back a RST (reset) packet. This causes the Recipient to go back into a LISTEN state. It will then receive the SYN packet send in 2, and after acknowledging it, a connection is made.

TCP connections are half-open if one of the TCPs has closed or aborted, and the other end is still connected. It can also occur if the two connections have become desynchronized because of a system crash. This connection is automatically reset if data is sent in either direction. This is because the sequence numbers will be incorrect, otherwise the connection will time-out.

A connection is normally closed with the CLOSE call. A host who has closed cannot continue to send, but can continue to RECEIVE until it is told to close by the other side. Figure P.57 shows a typical sequence for closing a connection. Normally the application program sends a CLOSE call for the given connection. Next, a TCP packet is sent with the FIN bit set, the originator enters into the FIN-WAIT-1 state. When the other TCP has acknowledged the FIN and sent a FIN of its own, the first TCP can ACK this FIN.

<b>Originator</b>		<b>Recipient</b>
1. CLOSED		LISTEN
2. SYN-SENT → <SEQ=999><CTL=SYN>		LISTEN
3. (duplicate) → <SEQ=900><CTL=SYN>		SYN-RECEIVED
4. SYN-SENT <SEQ=100><ACK=901> <CTL=SYN,ACK> ←		SYN-RECEIVED
5. SYN-SENT → <SEQ=901><CTL=RST>		LISTEN
6. (packet 2 received) →		
7. SYN-SENT <SEQ=100><ACK=1000><CTL=SYN,ACK> ←		SYN-RECEIVED
8. ESTABLISHED → <SEQ=1000><ACK=101><CTL=ACK><DATA>		ESTABLISHED

**Figure P.56** TCP connection with duplicate connections.

<b>Originator</b>		<b>Recipient</b>
1. ESTABLISHED <i>(CLOSE call)</i>		ESTABLISHED
2. FIN-WAIT-1      → <SEQ=1000><ACK=99> <CTL=SFN,ACK>		CLOSE-WAIT
3. FIN-WAIT-2      <SEQ=99><ACK=1001><CTL=ACK>      ←		CLOSE-WAIT
4. TIME-WAIT      <SEQ=99><ACK=101><CTL=FIN,ACK>      ←		LAST-ACK
5. TIME-WAIT      → <SEQ=1001><ACK=102><CTL=ACK>		CLOSED

**Figure P.57** TCP close connection.

## P.12 Example PGP encryption

---

The Pretty Good Privacy (PGP) program developed by Philip Zimmermann is widely available over the Internet. It runs as a stand-alone application, and uses various options to use the package. Table P.2 outlines some of the options.

**Table P.2** PGP options

<i>Option</i>	<i>Description</i>
pgp -e textfile her_userid [other userids]	Encrypts a plaintext file with the recipient's public key. In this case, it produces a file named textfile.pgp.
pgp -s textfile [-u your_userid]	Sign a plaintext file with a secret key. In this case, it produces a file named textfile.pgp.
pgp -es textfile her_userid [other userids] [-u your_userid]	Signs a plaintext file with the senders secret key, and then encrypt it with recipient's public key. In this case, it produces a file named textfile.pgp.
pgp -c textfile	Encrypt with conventional encryption only.
pgp ciphertextfile [-o plaintextfile]	Decrypt or check a signature for a ciphertext (.pgp) file.

To produce output in ASCII for email or to publish over the Internet, the -a option is used with other options. Table P.3 shows the key management functions.

**Table P.3** PGP key management options

<i>Option</i>	<i>Description</i>
pgp -kg	Generate a unique public and private key.
pgp -ka keyfile [keyring]	Adds key file's contents to the user's public or secret key ring.

---

pgp -kr userid [keyring]	Removes a key or a user ID from the user's public or secret key ring.
pgp -ke your_userid [keyring]	Edit user ID or pass phrase.
pgp -kx userid keyfile [keyring]	Extract a key from the public or secret key ring.
pgp -kv[v] [userid] [keyring]	View the contents of the public key ring.
pgp -kc [userid] [keyring]	Check signatures on the public key ring.
pgp -ks her_userid [-u your_userid] [keyring]	Sign someone else's public key on your public key ring.
pgp -krs userid [keyring]	Remove selected signatures from a userid on a keyring.

---

### P.12.1 RSA Key Generation

Both the public and the private keys are generated with:

```
pgp -kg
```

Initially, the user is asked about the key sizes. The larger the key the more secure it is. A 1024 bit key is very secure.

---

```
C:\pgp> pgp -kg
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-96 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-18
International version - not for use in the USA. Does not use RSAREF.
Current time: 1998/12/29 23:13 GMT

Pick your RSA key size:
 1) 512 bits- Low commercial grade, fast but less secure
 2) 768 bits- High commercial grade, medium speed, good security
 3) 1024 bits- "Military" grade, slow, highest security
Choose 1, 2, or 3, or enter desired number of bits: 3

Generating an RSA key with a 1024-bit modulus.
```

---

Next, the program asks for a user ID, which is normally the users name and his/her password. This ID helps other users to find the required public key.

---

```
You need a user ID for your public key. The desired form for this
user ID is your name, followed by your E-mail address enclosed in
<angle brackets>, if you have an E-mail address.
For example: John Q. Smith <12345.6789@compuserve.com>
Enter a user ID for your public key:
Fred Bloggs <fred_b@myserver.com>
```

---

Next PGP also asks for a pass phrase, which is used to protect the private key if another person gets hold of it. No person can use the secret key file, unless they know the pass phrase. Thus the pass phase is like a password but is typically much longer. The phase is also required when the user is encrypting a message with his/her private key.

---

You need a pass phrase to protect your RSA secret key.  
 Your pass phrase can be any sentence or phrase and may have many  
 words, spaces, punctuation, or any other printable characters.  
 Enter pass phrase: **fred bloggs**  
 Enter same pass phrase again: **fred bloggs**  
 Note that key generation is a lengthy process.

---

The public and private keys are randomly derived from measuring the intervals between key-strokes. For this the software asks for the user to type a number of keys.

---

We need to generate 384 random bits. This is done by measuring the time intervals between your keystrokes. Please enter some random text on your keyboard until you hear the beep:

We need to generate 384 random bits. This is done by measuring the time intervals between your keystrokes. Please enter some random text on your keyboard until you hear the beep:

<keyboard typing>

0 \* -Enough, thank you.  
 .....\*\*\*\*\*  
 .....\*\*\*\*\*

Pass phrase is good. Just a moment....  
 Key signature certificate added.  
 Key generation completed.

---

This has successfully generated the public and private keys. The public key is placed on the public key ring (PUBLISHING.PGP) and the private key is placed on the users secret key ring (SECRING.PGP).

---

C:\pgp> <b>dir *.pgp</b>	
SECRING PGP	518 12-29-98 11:20p secring.pgp
PUBLISHING PGP	340 12-29-98 11:20p pubring.pgp

---

The -kx option can be used to extract the new public key from the public key ring and place it in a separate public key file, which can be sent to people who want to send an encrypted message to the user.

---

C:\pgp> **pgp -kx fred\_b**  
 Extracting from key ring: 'pubring.pgp', userid "fred\_b".  
 Key for user ID: Fred Bloggs <fred\_b@myserver.com>  
 1024-bit key, key ID CD5AE745, created 1998/12/29  
 Extract the above key into which file? mykey  
 Key extracted to file 'mykey.pgp'.

---

The public key file (mykey.pgp) can be sent to other users, and can be added to their public key rings. Care must be taken never to send anyone a private key, but even if it is sent then it is still protected by the pass phase.

Often a user wants to publish their public key on their WWW page or transmit it by email. Thus, it requires to be converted into an ASCII format. For this the -kxa options can be used,

such as:

---

```
C:\pgp> pgp -kxa fred_b

Extracting from key ring: 'pubring.pgp', userid "fred_b".
Key for user ID: Fred Bloggs <fred_b@myserver.com>
1024-bit key, key ID CD5AE745, created 1998/12/29

Extract the above key into which file? mykey
Transport armor file: mykey.asc
Key extracted to file 'mykey.asc'.

Extract the above key into which file? mykey
Transport armor file: mykey.asc
Key extracted to file 'mykey.asc'.
```

---

The file mykey.asc now contains an ASCII form of the key, such as:

---

```
Type Bits/KeyID      Date        User ID
pub 1024/CD5AE745  1998/12/29  Fred Bloggs <fred_b@myserver.com>

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3i

mQCNAAzJY84AAAAEAK0nvnuYcwGEaNdeqcDGXD6IrMFwX3iKtdGkZgyPyiENLb+C
bGX7P2zSG0z1d8c4f5OKYR/RgxzN4ILsAKthGaweGD0FJRgeIvn6FHJxEzmdBWIh
ME/8h2HZfegSxta8hFAMc8o9ASamolk5KBL0YWFsQ1DNbR+dMJpPqQ7NWudFAAUT
tCFGcmVkJEjsb2dnycA8ZnJ1ZF9iQG15c2VydmVyLmNvbT6JAJUDBRA2iWPOmk+p
Dsla50UBAfkoA/4gO5D11Yk04DfjPng4itDtN55SgoE3upPWL52R5RQZf1BoJEF6
eLT/kejD5b7gl1/yPls456bh/k8ifi9RwSPUFN/zFUsVVYrSjZKD3kzClV1/QgTy
YmlDHHHgou6rYFXk7mGETWc4g4Dlrzds+ppc/UjN8uNp5KQUg1FsVatvPA==
=X5XX
-----END PGP PUBLIC KEY BLOCK-----
```

---

Now, someone's public key can be added to the Fred's public key ring. In this case, Fred Bloggs wants to send a message to Bert Smith. Bert's public key, in an ASCII form, is:

---

```
Type Bits/KeyID      Date        User ID
pub 1024/770CA60D  1998/12/30  Bert Smith <Bert_s.otherserver.com>

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3i

mQCNAAzKE5AAAAEAN+5td9acG1PcTKp5J42UpwbDqz6mHOaxcO1lp6CoPE3+AXT
jfREQ+TC0ZxMP6cCcwtEMnjVqu2M7F6li3v/AVqQIRZZkFSE0Z+8hlseHB0FR8Y
f8FDpmgld6wNpp8oc0yVul/sBQl549uOC/KnVQ6LtXo7UlsBtbnua9J3DKYNAUR
tCNCZXJ01FNtaXR0IDxCZXJ0X3MuB3RoZXJzZXJ2ZXIUY29tPokAlQMFEDaKE5B2
7mvSdwymDQE82xxEANLMEDncVrFjR71abUIWHqquEFK+sqnOHPBHyIBni18x03UM
jeQJM1WA9/uIPqzeABJd6anX4oK3yiByQjI5CT5+OdmU0y4e2+klab5mxxUWs7S
Tib3K5LLvPGxsOInOdunjFKaBLkrFU/L+ziid3iw9FV6Zy8P07yDL2SmobRbh
=6rTj
-----END PGP PUBLIC KEY BLOCK-----
```

---

Fred can add Bert's key onto his public key ring with the -ka option:

---

```
C:\pgp> pgp -ka bert.pgp

Looking for new keys...
pub 1024/770CA60D 1998/12/30 Bert Smith <Bert_s.otherserver.com>

Checking signatures...
pub 1024/770CA60D 1998/12/30 Bert Smith <Bert_s.otherserver.com>
sig! 770CA60D 1998/12/30 Bert Smith <Bert_s.otherserver.com>

Keyfile contains:
 1 new key(s)

One or more of the new keys are not fully certified.
Do you want to certify any of these keys yourself (y/N)?
```

---

Bert's key has been added to Fred's public key ring. This ring can be listed with the `-kv`, as given next:

---

```
C:\pgp> pgp -kv

Key ring: 'pubring.pgp'
Type Bits/KeyID      Date          User ID
pub 1024/770CA60D 1998/12/30 Bert Smith <Bert_s.otherserver.com>
pub 1024/CD5AE745 1998/12/29 Fred Bloggs <fred_b@myserver.com>
2 matching keys found.
```

---

Next, a message can be send to Bert, using his public key.

---

```
C:\pgp>edit message.txt
Bert,

This is a secret message. Please
delete it after you have read it!

Fred.

C:\pgp>pgp -e message.txt

Recipients' public key(s) will be used to encrypt.
A user ID is required to select the recipient's public key.
Enter the recipient's user ID: bert smith

Key for user ID: Bert Smith <Bert_s.otherserver.com>
1024-bit key, key ID 770CA60D, created 1998/12/30

WARNING: Because this public key is not certified with a trusted
signature, it is not known with high confidence that this public key
actually belongs to: "Bert Smith <Bert_s.otherserver.com>".

Are you sure you want to use this public key (y/N)? y

Ciphertext file: message.pgp
```

---

If the message needs to be transmitted by electronic mail or via a WWW page, it can be converted into text format with the `-ea` option, as given next:

---

```
C:\pgp>pgp -ea message.txt
Recipients' public key(s) will be used to encrypt.
A user ID is required to select the recipient's public key.
Enter the recipient's user ID: bert smith

Key for user ID: Bert Smith <Bert_s.otherserver.com>
1024-bit key, key ID 770CA60D, created 1998/12/30

WARNING: Because this public key is not certified with a trusted
signature, it is not known with high confidence that this public key
actually belongs to: "Bert Smith <Bert_s.otherserver.com>".
But you previously approved using this public key anyway.

.
Transport armor file: message.asc
```

---

The message.asc file is now in a form which can be transmitted in ASCII characters. In this case, it is:

---

```
-----BEGIN PGP MESSAGE-----
Version: 2.6.3i

hIWdDu5r0ncMpg0BBAC7jOUx74vLb7011OC00/5Fkc6pDJinqpA7isJH+JYbFkDj
wSv6vF/jAEonEPL8RVTqWhcNDwj{jwwV9OVPEzeaZ0qgZTWbdSUlfqxZsaBo8Uz
dmmbzxd7CDTpnsYEyFWosPyzdxJqlsICig79Loh7l1BdJXehKnMy+1VMieNYtKYA
AABrB8LTMj21kk9t6JfS2y0c1t9EfPvMLX+rxtPZ+TqlaCowfid4E77Fy1KN260N
APzF8J6elXhBgNM3zesA8fR8KdEnrI2BYC2XsBzTxOiKnpqoLMwWl0A7TTyhv24L
1PhwFi/YQ2SPhemdpqY=
=ooNT
-----END PGP MESSAGE-----
```

---

Bert can now simply decrypt the received message.

---

```
C:\pgp\bert> pgp message.pgp

File is encrypted. Secret key is required to read it.
Key for user ID: Bert Smith <Bert_s.otherserver.com>
1024-bit key, key ID 770CA60D, created 1998/12/30

You need a pass phrase to unlock your RSA secret key.
Enter pass phrase: Bert Smith
Pass phrase is good. Just a moment.....
Plaintext filename: message
```

---

Or Bert can convert the ASCII form into a binary format with the -da option, and the decrypt the message as before.

---

```
C:\pgp\bert> pgp -da message.asc
Stripped transport armor from 'message.asc', producing 'message.pgp'.
```

---

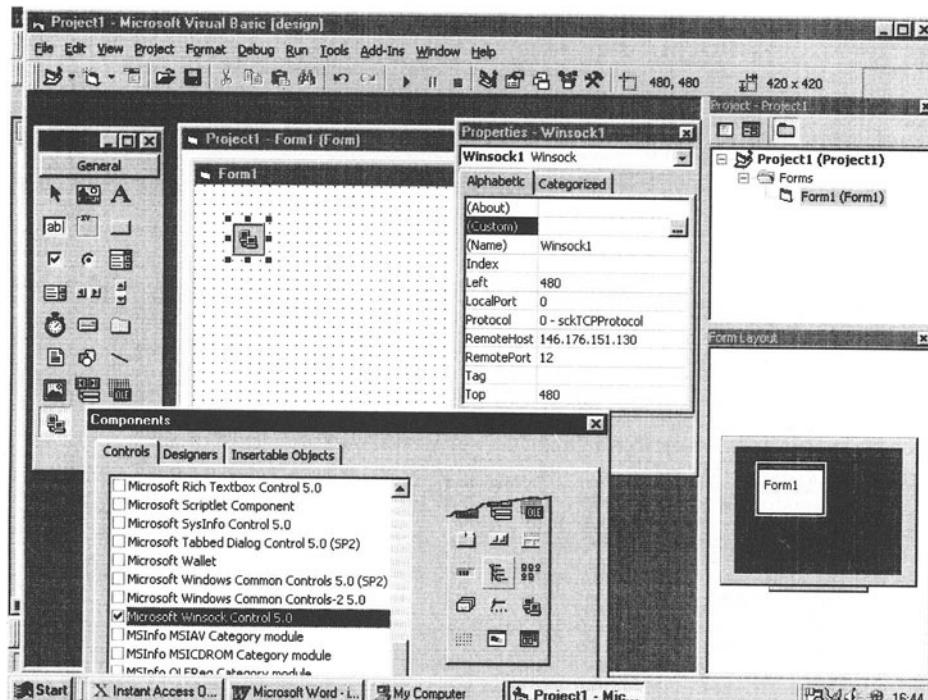
## P.13 Visual Basic implementation of sockets

---

Visual Basic support a WinSock control which allows the connection of hosts over a network. It supports both UDP and TCP. Figure P.58 shows a sample Visual Basic screen with a

WinSock object (in this case, it is named Winsock1). To set the Protocol used then either select the Properties window on the WinSock object, click Protocol and select either sckTCPProtocol, or sckUDPProtocol. Otherwise, within the code it can be set to TCP with:

```
Winsock1.Protocol = sckTCPProtocol
```



**Figure P.58** WinSock object.

The WinSock object has various properties, such as:

- |                        |   |
|------------------------|---|
| <i>obj</i> .RemoteHost | Defines the IP address or DNS of the remote host. |
| <i>obj</i> .LocalPort  | Defines the local port number                     |

The methods that are used with the WinSock object are:

- |                      |   |
|----------------------|---|
| <i>obj</i> .Connect  | Connects to a remote host (client invoked). |
| <i>obj</i> .Listen   | Lists for a connection (server invoked).    |
| <i>obj</i> .GetData  | Reads data from the input stream.           |
| <i>obj</i> .SendData | Sends data to an output stream.             |

The main events are:

- |                   |   |
|-------------------|---|
| ConnectionRequest | Occurs when a remote host wants to make a connection with a server. |
|-------------------|---|

DataArrival Occurs when data has arrived from a connection (data is then read with GetData).

### P.13.1 Creating a server

A server must listen for connection. To do this, do the following:

1. Create a new Standard EXE project.
2. Change the name of the default form to myServer.
3. Change the caption of the form to “Server Application” (see Figure P.59).
4. Put a Winsock control on the main format and change its name to myTCPServer.
5. Add two TextBox controls to the form. Name the first SendTextData, and the second ShowText.
6. Add the code given below to the form.

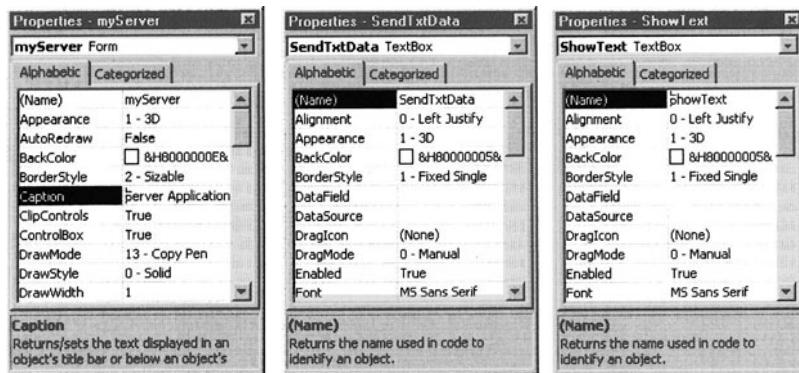


Figure P.59 Server setups.

```

Private Sub Form_Load()
    ' Set the local port to 1001 and listen for a connection
    myTCPServer.LocalPort = 1001
    myTCPServer.Listen
    myClient.Show
End Sub

Private Sub myTCPServer_ConnectionRequest (ByVal requestID As Long)
    ' Check state of socket, if it is not closed then close it.
    If myTCPServer.State <> sckClosed Then myTCPServer.Close
    ' Accept the request with the requestID parameter.
    myTCPServer.Accept requestID
End Sub

Private Sub SendTextData_Change()
    ' SendTextData contains the data to be sent.
    ' This data is setn using the SendData method
    myTCPServer.SendData = SendTextData.Text
End Sub

Private Sub myTCPServer_DataArrival (ByVal bytesTotal As Long)
    ' Read incoming data into the str variable,
    ' then display it to ShowText
    Dim str As String
    myTCPServer.GetData = str

```

```

ShowText.Text = str
End Sub

```

Figure P.60 shows the server setup.

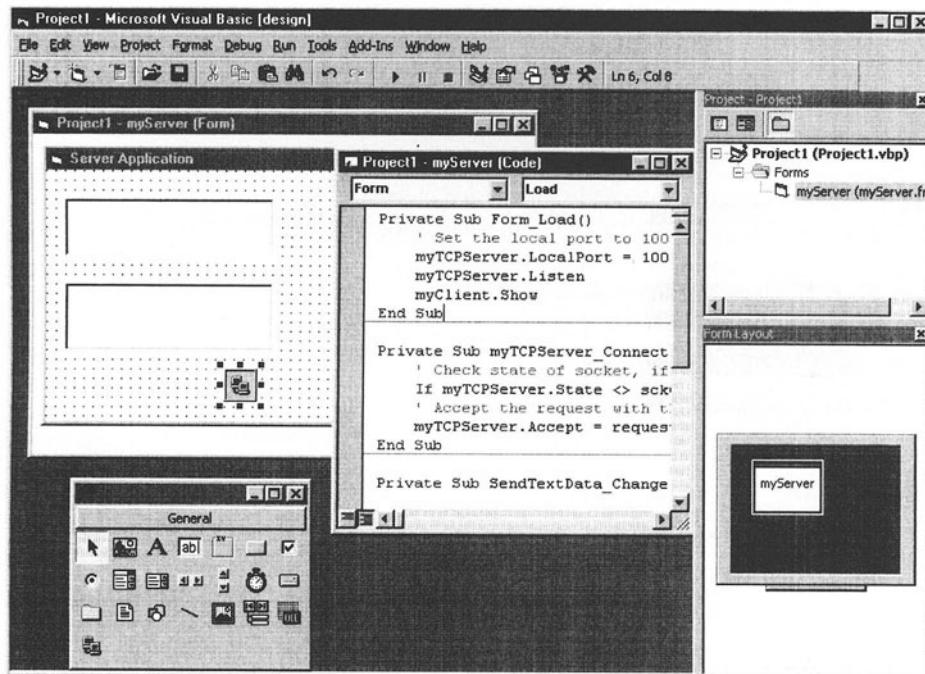


Figure P.60 Server form.

### P.13.2 Creating a client

The client must actively seek a connection. To create a client, do the following:

1. A new form to the project, and name it myClient.
2. Change the caption of the form to “Client Application”.
3. Add a Winsock control to the form and name it myTCPClient.
4. Add two TextBox controls to the form. Name the first SendTextData, and the second ShowText.
5. Draw a CommandButton control on the form and name it cmdConnect.
6. Change the caption of the CommandButton control to Connect (see Figure P.61).
7. Add the code given below to the form.

```

Private Sub Form_Load()
    ' In this case it will connect to 146.176.151.130
    ' change this to the local IP address or DNS of the local computer
    myTCPClient.RemoteHost = "146.176.151.130"
    myTCPClient.RemotePort = 1001
End Sub

Private Sub cmdConnect_Click()

```

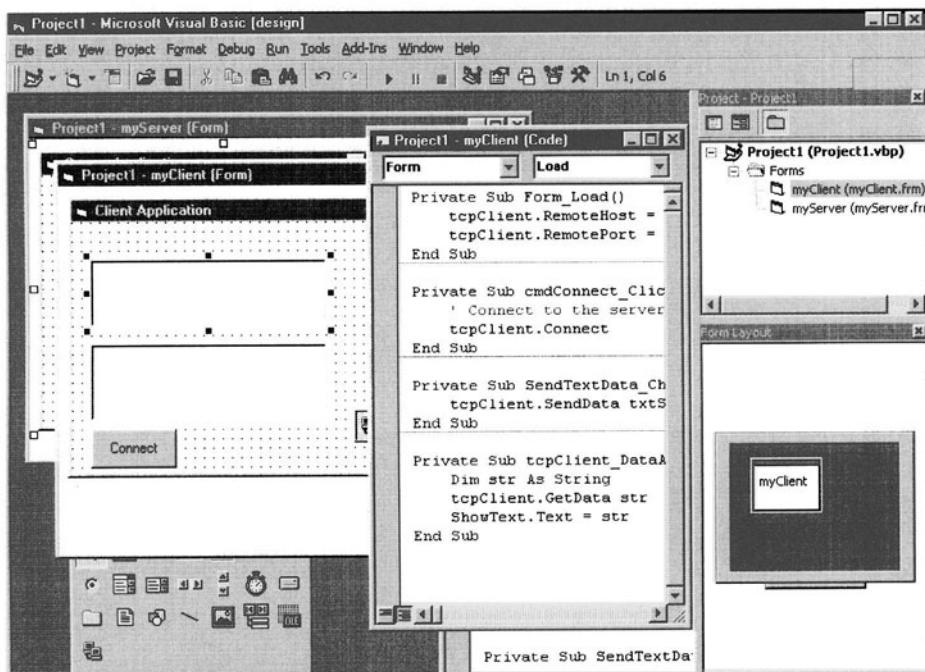
```

' Connect to the server
myTCPClient.Connect
End Sub

Private Sub SendTextData_Change()
    tcpClient.SendData txtSend.Text
End Sub

Private Sub tcpClient_DataArrival (ByVal bytesTotal As Long)
    Dim str As String
    myTCPClient.GetData str
    ShowText.Text = str
End Sub

```



**Figure P.61** Client form.

The program, when it is run, will act as a client and a server. Any text typed in the SendTxtData TextBox will be sent to the ShowText TextBox on the other form.

### P.13.3 Multiple connections

In Visual Basic, it is also possible to create multiple connections to a server. This is done by creating multiple occurrences of the server object. A new one is created every time there is a new connection (with the Connection\_Request event). Each new server accepts the incoming connection. The following code, which has a Winsock control on a form called multServer, is given below.

```

Private ConnectNo As Long

Private Sub Form_Load()

```

```

    ConnectNo = 0
    multServer(0).LocalPort = 1001
    multServer(0).Listen
End Sub

Private Sub multServer_ConnectionRequest _           (Index As Integer, ByVal requestID As Long)
    If Index = 0 Then
        ConnectNo = ConnectNo + 1
        Load multServer(ConnectNo)
        multServer(ConnectNo).LocalPort = 0
        multServer(ConnectNo).Accept requestID
        Load txtData(ConnectNo)
    End If
End Sub

```

#### **P.13.4 Connect event**

The Connect event connects to a server. If an error occurs then a flag (ErrorOccurred) is set to True, else it is False. Its syntax is:

```
Private Sub object.Connect (ErrorOccurred As Boolean)
```

#### **P.13.5 Close event**

The Close event occurs when the remote computer closes the connection. Applications should use the Close method to correctly close their connection. Its syntax is:

```
object_Close()
```

#### **P.13.6 DataArrival event**

The DataArrival event occurs when new data arrives, and returns the number of bytes read (bytesTotal). Its syntax is:

```
object_DataArrival (bytesTotal As Long)
```

#### **P.13.7 Bind method**

The Bind method specifies the Local port (LocalPort) and the Local IP address (LocalIP) to be used for TCP connections. Its syntax is:

```
object.Bind LocalPort, LocalIP
```

#### **P.13.8 Listen method**

The Listen method creates a socket and goes into listen mode (for server applications). It stays in this mode until a ConnectionRequest event occurs, which indicates an incoming connection. After this, the Accept method should be used to accept the connection. Its syntax is:

```
object.Listen
```

#### **P.13.9 Accept method**

The Accept method accepts incoming connections after a ConnectionRequest event. Its syntax is:

*object.Accept requestID*

The requestID parameter is passed into the ConnectionRequest event and is used with the Accept method.

#### P.13.10 Close method

The Close method closes a TCP connection. Its syntax is:

*object.Close*

#### P.13.11 SendData method

The SendData methods sends data (Data) to a remote computer. Its syntax is:

*object.SendData Data*

#### P.13.12 GetData method

The GetData method gets data (Data) from an object. Its syntax is:

*object.GetData data, [type,] [maxLen]*

---

## P.14 ARP

---

ARP (Address Resolution Protocol) translates IP addresses to Ethernet addresses. This is used when IP packets are send from a computer, and the Ethernet address is added to the Ethernet frame. A table look-up, called the ARP table, is used to translate the addresses. One column has the IP address and the other has the Ethernet address. The following is an example ARP table:

---

IP address	Ethernet address
146.176.150.2	00-80-C8-22-6BE2
146.176.150.3	00-80-C8-22-CD4E
146.176.150.4	00-80-C8-23-114C

---

A typical conversation is as follows:

1. Application sends an application message to TCP.
2. TCP sends the corresponding TCP message to the IP module. The destination IP address is known by the application, the TCP module, and the IP module.
3. At this point the IP packet has been constructed and is ready to be given to the Ethernet driver, but first the destination Ethernet address must be determined.
4. The ARP table is used to look-up the destination Ethernet address.

The sequence is as follows:

1. An ARP request packet with a broadcast Ethernet address (FF-FF-FF-FF-FF-FF) is sent out on the network to every computer. Other typical Ethernet broadcast addresses are

- given in Section P.14.1.
2. The outgoing IP packet is queued.
  3. All the computers on the network segment read the broadcast Ethernet frame, and examine the Type field to determine if it is an ARP packet. If it is then it is passed to the ARP module.
  4. If the IP address of a receiving station matches the IP address in the IP packet then it sends a response directly to the source Ethernet address.
  5. The originator then receives the Ethernet frame and checks the Type field to determine if it is an ARP packet. If it is then it adds the sender's IP address and Ethernet address to its ARP table.
  6. The IP packet can now be sent with the correct Ethernet address.

Each computer has a separate ARP table for each of its Ethernet interfaces.

#### **P.14.1 Ethernet Multicast/Broadcast Addresses**

The following is a list of typical Ethernet Multicast addresses:

<i>Ethernet address</i>	<i>Type field</i>	<i>Usage</i>
01-00-5E-00-00-00	0800	Internet Multicast (RFC-1112)
01-80-C2-00-00-00	0802	Spanning tree (for bridges)
09-00-09-00-00-01	8005	HP Probe
09-00-09-00-00-04	8005	HP DTC
09-00-1E-00-00-00	8019	Apollo DOMAIN
09-00-2B-00-00-03	8038	DEC Lanbridge Traffic Monitor (LTM)
09-00-4E-00-00-02	8137	Novell IPX
0D-1E-15-BA-DD-06	????	HP
CF-00-00-00-00-00	9000	Ethernet Configuration Test protocol

The following is a list of typical Ethernet Broadcast addresses:

<i>Ethernet address</i>	<i>Type field</i>	<i>Usage</i>
FF-FF-FF-FF-FF-FF	0600	XNS packets, Hello or gateway search.
FF-FF-FF-FF-FF-FF	0800	IP (such as RWHOD with UDP)
FF-FF-FF-FF-FF-FF	0804	CHAOS
FF-FF-FF-FF-FF-FF	0806	ARP (for IP and CHAOS) as needed
FF-FF-FF-FF-FF-FF	0BAD	Banyan
FF-FF-FF-FF-FF-FF	1600	VALID packets. Hello or gateway search.
FF-FF-FF-FF-FF-FF	8035	Reverse ARP
FF-FF-FF-FF-FF-FF	807C	Merit Internodal (INP)
FF-FF-FF-FF-FF-FF	809B	EtherTalk

---

## **P.15 IP multicasting**

Many applications of modern communications require the transmission of IP datagrams to multiple hosts. Typical applications are video conferencing, remote teaching, and so on. This is supported by IP multicasting, where a host group is identified by a single IP address. The main parameters of IP multicasting are:

1. The group membership is dynamic.

2. Hosts may join and leave the group at any time.
3. There is also no limit to the location or number of members in a host group.
4. A host may be a member of more than one group at a time.
5. A host group may be permanent or transient. Permanent groups are well-known and are administratively assigned a permanent IP address. The group is then dynamically associated with this IP address. IP multicast addresses that are not reserved to permanent groups are available for dynamic assignment to transient groups.
6. Multicast routers forward IP multicast datagrams into the Internet.

#### **P.15.1 Group addresses**

A special group of addresses are assigned to multicasting. These are known as Class D address, an the begin with 1110 as their starting 4 bits (Class E addresses with the upper bits of 1111 are reserved for future uses). The Class D addresses thus range from:

224.0.0.0                   (11100000 00000000 00000000 00000000)

239.255.255.255   (11101111 11111111 11111111 11111111)

The address 224.0.0.0 is reserved. 224.0.0.1 is also assigned to the permanent group of all IP hosts (including gateways), and is used to address all multicast hosts on the directly connected network. Reserved and allocated addresses are:

224.0.0.0	Reserved
224.0.0.1	All Systems on current subnet
224.0.0.2	All Routers on current subnet
224.0.0.3	Unassigned
224.0.0.4	DVMRP Routers
224.0.0.5	OSPFIGP All Routers
224.0.0.6	OSPFIGP Designated Routers
224.0.0.7	ST Routers
224.0.0.8	ST Hosts
224.0.0.9	RIP2 Routers
224.0.0.10-224.0.0.255	Unassigned
224.0.1.0	VMTCP Managers Group
224.0.1.1	NTP Network Time Protocol
224.0.1.2	SGI-Dogfight
224.0.1.3	Rwhod
224.0.1.4	VNP
224.0.1.5	Artificial Horizons - Aviator
224.0.1.6	NSS - Name Service Server
224.0.1.7	AUDIONEWS - Audio News Multicast
224.0.1.8	SUN NIS+ Information Service
224.0.1.9	MTP Multicast Transport Protocol
224.0.1.10-224.0.1.255	Unassigned
224.0.2.1	rwho Group (BSD) (unofficial)
224.0.2.2	SUN RPC PMAPPROC_CALLIT
224.0.3.0-224.0.3.255	RFE Generic Service

224.0.4.0-224.0.4.255	RFE Individual Conferences
224.1.0.0-224.1.255.255	ST Multicast Groups
224.2.0.0-224.2.255.255	Multimedia Conference Calls
232.x.x.x	VMTP transient groups

All the above addresses are listing in the Domain Name Service under MCAST.NET and 224.IN-ADDR.ARPA. On an Ethernet or IEEE 802 network, the 23 low-order bits of the IP Multicast address are placed in the low-order 23 bits of the Ethernet or IEEE 802 net multi-cast address.

### **P.15.2 Conformance**

There are three levels of conformance:

1. Level 0. No IP multicasting support. In this, a Level 0 host ignores, or deletes, all Class D addressed datagrams.
2. Level 1. Sending support, but no receiving. In this, a Level 1 host can send multicast datagrams, but cannot receive them.
3. Level 2: Full multicasting support. In this, a Level 2 host can send and receive IP multicasting. It also requires the implementation of the Internet Group Management Protocol (IGMP).

---

## **P.16 Assigned Internet Protocol numbers**

---

This section contains information extracted from RFC1700 [Reynolds and Postel] on assigned number values.

### **IP Special addresses**

The main forms of IP addresses are:

IP-address ::= { <Network-number>, <Host-number> }

and

IP-address ::= { <Network-number>, <Subnet-number>, <Host-number> }

Special addresses are:

{0, 0}.	Host on this network. This address can only be used as a source address
{0, <Host-number>}	Host on this network
{-1, -1}.	Limited broadcast. This address can only be used as a destination address, and should not be forwarded outside the current subnet.
{<Network-number>, -1}	Directed broadcast to specified network. This address can only be used as a destination address.
{<Network-number>, <Subnet-number>, -1}	

		Directed broadcast to specified subnet. This address can only be used as a destination address.
{<Network-number>, -1, -1}		Directed broadcast to all subnets of specified subnetted network. This address can only be used as a destination address.
{127, <any>}		Internal host loopback address. This address should never appear outside a host.

where -1 represents an address of all 1's.

## IP Versions

Decimal	Keyword	Version
0		Reserved
1-3		Unassigned
4	IP	Internet Protocol
5	ST	ST Datagram Mode
6	SIP	Simple Internet Protocol
7	TP/IX	TP/IX: The Next Internet
8	PIP	The P Internet Protocol
9	TUBA	TUBA
10-14		Unassigned
15		Reserved

## IP protocol numbers

Decimal	Keyword	Protocol
0		Reserved
1	ICMP	Internet Control Message
2	IGMP	Internet Group Management
3	GGP	Gateway-to-Gateway
4	IP	IP in IP (encapsulation)
5	ST	Stream
6	TCP	Transmission Control
7	UCL	UCL
8	EGP	Exterior Gateway Protocol
9	IGP	any private interior gateway
10	BBN-RCC-MON	BBN RCC Monitoring
11	NVP-II	Network Voice Protocol
12	PUP	PUP
13	ARGUS	ARGUS
14	EMCON	EMCON
15	XNET	Cross Net Debugger
16	CHAOS	Chaos
17	UDP	User Datagram
18	MUX	Multiplexing
19	DCN-MEAS	DCN Measurement Subsystems
20	HMP	Host Monitoring
21	PRM	Packet Radio Measurement
22	XNS-IDP	XEROX NS IDP
23	TRUNK-1	Trunk-1
24	TRUNK-2	Trunk-2
25	LEAF-1	Leaf-1
26	LEAF-2	Leaf-2
27	RDP	Reliable Data Protocol
28	IRTP	Internet Reliable Transaction
29	ISO-TP4	ISO Transport Protocol Class 4
30	NETBLT	Bulk Data Transfer Protocol
31	MFE-NSP	MFE Network Services Protocol
32	MERIT-INP	MERIT Internodal Protocol
33	SEP	Sequential Exchange Protocol
34	3PC	Third Party Connect Protocol
35	IDPR	Inter-Domain Policy Routing Protocol
36	XTP	XTP
37	DDP	Datagram Delivery Protocol
38	IDPR-CMTP	IDPR Control Message Transport Proto
39	TP++	TP++ Transport Protocol
40	IL	IL Transport Protocol
41	SIP	Simple Internet Protocol
42	SDRP	Source Demand Routing Protocol

43	SIP-SR	SIP Source Route
44	SIP-FRAG	SIP Fragment
45	IDRP	Inter-Domain Routing Protocol
46	RSVP	Reservation Protocol
47	GRE	General Routing Encapsulation
48	MHRP	Mobile Host Routing Protocol
49	BNA	BNA
50	SIPP-ESP	SIPP Encap Security Payload
51	SIPP-AH	SIPP Authentication Header
52	I-NLSP	Integrated Net Layer Security TUBA
53	SWIPE	IP with Encryption
54	NHRP	NBMA Next Hop Resolution Protocol
61		any host internal protocol
62	CFTP	CFTP
63		any local network
64	SAT-EXPAK	SATNET and Backroom EXPAK
65	KRYPTOLAN	Kryptolan
66	RVD	MIT Remote Virtual Disk Protocol
67	IPPC	Internet Pluribus Packet Core
68		any distributed file system
69	SAT-MON	SATNET Monitoring
70	VISA	VISA Protocol
71	IPCV	Internet Packet Core Utility
72	CPNX	Computer Protocol Network Executive
73	CPHB	Computer Protocol Heart Beat
74	WSN	Wang Span Network
75	PVP	Packet Video Protocol
76	BR-SAT-MON	Backroom SATNET Monitoring
77	SUN-ND	SUN ND PROTOCOL-Temporary
78	WB-MON	WIDEBAND Monitoring
79	WB-EXPAK	WIDEBAND EXPAK
80	ISO-IP	ISO Internet Protocol
81	VMTCP	VMTCP
82	SECURE-VMTCP	SECURE-VMTCP
83	VINES	VINES
84	TTP	TTP
85	NSFNET-IGP	NSFNET-IGP
86	DGP	Dissimilar Gateway Protocol
87	TCF	TCF
88	IGRP	IGRP
89	OSPF-IGP	OSPF-IGP
90	Sprite-RPC	Sprite RPC Protocol
91	LARP	Locus Address Resolution Protocol
92	MTP	Multicast Transport Protocol
93	AX.25	AX.25 Frames
94	IP-IP	IP-within-IP Encapsulation Protocol
95	MICP	Mobile Internetworking Control Pro.
96	SCC-SP	Semaphore Communications Sec. Pro.
97	ETHERIP	Ethernet-within-IP Encapsulation
98	ENCAP	Encapsulation Header
99		any private encryption scheme
100	GMTP	GMTP

## Ports

0		Reserved
1	tcpmux	TCP Port Service Multiplexer
2	compressnet	Management Utility
3	compressnet	Compression Process
5	rje	Remote Job Entry
7	echo	Echo
11	discard	Discard
13	systat	Active Users
15	daytime	Daytime
17	qotd	Quote of the Day
18	msp	Message Send Protocol
19	chargen	Character Generator
20	ftp-data	File Transfer [Default Data]
21	ftp	File Transfer [Control]
23	telnet	Telnet
25	smtp	Simple Mail Transfer
27	nsw-fe	NSW User System FE
29	msg-icp	MSG ICP
31	msg-auth	MSG Authentication

33	dsp	Display Support Protocol
37	time	Time
38	rap	Route Access Protocol
39	rlp	Resource Location Protocol
41	graphics	Graphics
42	nameserver	Host Name Server
43	nickname	Who Is
44	mpm-flags	MPM FLAGS Protocol
45	mpm	Message Processing Module
46	mpm-snd	MPM {default send}
47	ni-ftp	NI FTP
48	auditd	Digital Audit Daemon
49	login	Login Host Protocol
50	re-mail-ck	Remote Mail Checking Protocol
51	la-maint	IMP Logical Address Maintenance
52	xns-time	XNS Time Protocol
53	domain	Domain Name Server
54	xns-ch	XNS Clearinghouse
55	isi-gl	ISI Graphics Language
56	xns-auth	XNS Authentication
58	xns-mail	XNS Mail
61	ni-mail	NI MAIL
62	acas	ACA Services
64	covia	Communications Integrator (CI)
65	tacacs-ds	TACACS-Database Service
66	sql*net	Oracle SQL*NET
67	bootps	Bootstrap Protocol Server
68	bootpc	Bootstrap Protocol Client
69	tftp	Trivial File Transfer
70	gopher	Gopher
71	netrjs-1	Remote Job Service
72	netrjs-2	Remote Job Service
73	netrjs-3	Remote Job Service
74	netrjs-4	Remote Job Service
76	deos	Distributed External Object St
78	vettcp	vettcp
79	finger	Finger
80	www-http	World Wide Web HTTP
81	hosts2-ns	HOSTS2 Name Server
82	xfer	XFER Utility
83	mit-ml-dev	MIT ML Device
84	ctf	Common Trace Facility
85	mit-ml-dev	MIT ML Device
86	mfcobol	Micro Focus Cobol
88	kerberos	Kerberos
89	su-mit-tg	SU/MIT Telnet Gateway
90	dnsix	DNSIX Securit Attribute Token
91	mit-dov	MIT Dover Spooler
92	npp	Network Printing Protocol
93	dcp	Device Control Protocol
94	objcall	Tivoli Object Dispatcher
95	supdup	SUPDUP
96	ixie	DIXIE Protocol Specification
97	swift-rvf	Swift Remote Vitural File Prot
98	tacnews	TAC News
99	metagram	Metagram Relay
101	hostname	NIC Host Name Server
102	iso-tsap	ISO-TSAP
103	gppitnp	Genesis Point-to-Point Trans N
104	acr-nema	ACR-NEMA Digital Imag. & Comm.
105	csnet-ns	Mailbox Name Nameserver
106	3com-tsmux	3COM-TSMUX
107	rtelnet	Remote Telnet Service
108	snagas	SNA Gateway Access Server
109	pop2	Post Office Protocol - Version 2
110	pop3	Post Office Protocol - Version 3
111	sunrpc	SUN Remote Procedure Call
112	mcidas	McIDAS Data Transmission Proto
113	auth	Authentication Service
114	audionews	Audio News Multicast
115	sftp	Simple File Transfer Protocol
116	ansanotify	ANSA REX Notify
117	uucp-path	UUCP Path Service
118	sqlserv	SQL Services
119	nntp	Network News Transfer Protocol

120	cfdpkt	CFDPKT
121	erpc	Encore Expedited Remote Pro.Ca
122	smaiknet	SMAKYNET
123	ntp	Network Time Protocol
124	ansatrader	ANSA REX Trader
125	locus-map	Locus PC-Interface Net Map Ser
126	unitary	Unisys Unitary Login
127	locus-con	Locus PC-Interface Conn Server
128	gss-xlicen	GSS X License Verification
129	pwdgen	Password Generator Protocol
130	cisco-fna	cisco FNATIVE
131	cisco-tna	cisco TNATIVE
132	cisco-sys	cisco SYSMINT
133	statsrv	Statistics Service
134	ingres-net	INGRES-NET Service
135	loc-srv	Location Service
136	profile	PROFILE Naming System
137	netbios-ns	NETBIOS Name Service
138	netbios-dgm	NETBIOS Datagram Service
139	netbios-ssn	NETBIOS Session Service
140	emfis-data	EMFIS Data Service
141	emfis-cntl	EMFIS Control Service
142	bl-idm	Britton-Lee IDM
143	imap2	Interim Mail Access Protocol
144	news	NewS
145	uaac	UAAC Protocol
146	iso-tp0	ISO-IP0
147	iso-ip	ISO-IP
148	cronus	CRONUS-SUPPORT
149	aed-512	AED 512 Emulation Service
150	sql-net	SQL-NET
151	hems	HEMS
152	bftp	Background File Transfer Progr
153	sgmp SGMP	
154	netsc-prod	NETSC
155	netsc-dev	NETSC
156	sqlsrv	SQL Service
157	knet-cmp	KNET/VM Command/Message Protoc
158	pcmail-srv	PCMail Server
159	nss-routing	NSS-Routing
160	sgmp-traps	SGMP-TRAPS
161	snmp	SNMP
162	snmptrap	SNMPTRAP
163	cmip-man	CMIP Manager
164	cmip-agent	CMIP Agent
165	xns-courier	Xerox
166	s-net	Sirius Systems
167	namp	NAMP
168	rsvd	RSVD
169	send	SEND
170	print-srv	Network PostScript
171	multiplex	Network Innovations Multiplex
172	cl/1	Network Innovations CL/1
173	xyplex-mux	Xyplex
174	mailq	MAILQ
175	vmnet	VMNET
176	genrad-mux	GENRAD-MUX
177	xdmcpx	Display Manager Control Prot
178	nextstep	NextStep Window Server
179	bgp	Border Gateway Protocol
180	ris	Intergraph
181	unify	Unify
182	audit	Unisys Audit SITP
183	ocbinder	OCBINDER
184	ocserver	OCSERVER
185	remote-kis	Remote-KIS
186	kis	KIS Protocol
187	aci	Application Communication Inte
188	mumps	Plus Five's MUMPS
189	qft	Queued File Transport
190	gacp	Gateway Access Control Protoco
191	prospero	Prospero Directory Service
192	osu-nms	OSU Network Monitoring System
193	srmp	Spider Remote Monitoring Proto
194	irc	Internet Relay Chat Protocol

195 dn6-nlm-aud DNSIX Network Level Module Aud  
196 dn6-smm-red DNSIX Session Mgt Module Audit  
197 dls Directory Location Service  
198 dls-mon Directory Location Service Mon  
199 smux SMUX  
200 src IBM System Resource Controller  
201 at-rtmp AppleTalk Routing Maintenance  
202 at-nbp AppleTalk Name Binding  
203 at-3 AppleTalk Unused  
204 at-echo AppleTalk Echo  
205 at-5 AppleTalk Unused  
206 at-zis AppleTalk Zone Information  
207 at-7 AppleTalk Unused  
208 at-8 AppleTalk Unused  
209 tam Trivial Authenticated Mail Pro  
210 z39.50 ANSI Z39.50  
211 914c/g Texas Instruments 914C/G Termi  
212 anet ATEXSSTR  
213 ipx IPX  
214 vmpwscs VM PWSCS  
215 softpc Insignia Solutions  
216 atls Access Technology License Serv  
217 dbase dBASE Unix  
218 mpp Netix Message Posting Protocol  
219 uarp Unisys ARPs  
220 imap3 Interactive Mail Access Protoc  
221 fln-spx Berkeley rlogind with SPX auth  
222 rsh-spx Berkeley rshd with SPX auth  
223 cdc Certificate Distribution Cente  
243 sur-meas Survey Measurement  
245 link LINK  
246 dsp3270 Display Systems Protocol  
344 pdap Prospero Data Access Protocol  
345 wserv Perf Analysis Workbench  
346 zserv Zebra server  
347 fatserv Fatmen Server  
348 csi-sqwp Cabletron Management Protocol  
371 clearcase Clearcase  
372 ulistserv Unix Listserv  
373 legent-1 Legent Corporation  
374 legent-2 Legent Corporation  
375 hassle Hassle  
376 nip Amiga Envoy Network Inquiry Pr  
377 tnETOS NEC Corporation  
378 dsETOS NEC Corporation  
379 is99c TIA/EIA/IS-99 modem client  
380 is99s TIA/EIA/IS-99 modem server  
381 hp-collector hp performance data collector  
382 hp-managed-node hp performance data managed no  
383 hp-alarm-mgr hp performance data alarm mana  
384 arns A Remote Network Server System  
385 ibm-app IBM Application  
386 asa ASA Message Router Object Def.  
387 . aurp Appletalk Update-Based Routing  
388 unidata-ldm Unidata LDM Version 4  
389 ldap Lightweight Directory Access P  
390 uis UIS  
391 synoptics-relay SynOptics SNMP Relay Port  
392 synoptics-broker SynOptics Port Broker Port  
393 dis Data Interpretation System  
394 embl-ndt EMBL Nucleic Data Transfer  
395 netcp NETscout Control Protocol  
396 netware-ip Novell Netware over IP  
397 mptn Multi Protocol Trans. Net.  
398 kryptolan Kryptolan  
400 work-sol Workstation Solutions  
401 ups Uninterruptible Power Supply  
402 genie Genie Protocol  
403 decapdecap  
404 nced nced  
405 ncld ncld  
406 imsp Interactive Mail Support Proto  
407 timbuktu Timbuktu  
408 prm-sm Prospero Resource Manager Sys.  
409 prm-nm Prospero Resource Manager Node

```

410 declarebug      DECLAdedbug Remote Debug Protoc
411 rmt            Remote MT Protocol
412 synoptics-trap Trap Convention Port
413 smsp SMSP
414 infoseek       InfoSeek
415 bnet BNNet
416 silverplatter Silverplatter
417 onmuxOnmux
418 hyper-g        Hyper-G
419 ariell         Ariel
420 smpteSMPTe
421 ariel2         Ariel
422 ariel3         Ariel
423 opc-job-start IBM Operations Planning and Co
424 opc-job-track IBM Operations Planning and Co
425 icad-el        ICAD
426 smartsdp       smartsdp
427 svrlloc        Server Location
428 ocs_cmu        OCS_CMU
429 ocs_amu        OCS_AMU
430 utmpsd         UTMPSD
431 utmpcd         UTMPCD
432 iasd IASD
433 nnsp NNTP
434 mobileip-agent MobileIP-Agent
435 mobilip-mn     MobilIP-MN
436 dna-cml        DNA-CML
437 comscm         comscm
438 dsfgwdsgfw
439 dasp           dasp
440 sgcp           sgcp
441 decvms-sysmgt decvms-sysmgt
442 cvc_hostd      cvc_hostd
443 https          https MCom
444 snpp           Simple Network Paging Protocol
445 microsoft-ds   Microsoft-DS
446 ddm-rdb        DDM-RDB
447 ddm-dfm        DDM-RFM
448 ddm-byte       DDM-BYTE
449 as-servermap   AS Server Mapper
450 tserver         TServer
451 exec           remote process execution;
452 login          remote login
453 cmd            like exec, but automatic
454 printer         spooler
455 talk           like tenex link, but across
456 ntalk          like tenex link, but across
457 utimeunixtime
458 efs            extended file name server
459 timedtimeserver
460 tempo newdate
461 courier         rpc
462 conference      chat
463 netnews         readnews
464 netwall         for emergency broadcasts
465 apertus-ldp    Apertus Technologies Load Dete
466 uucp uucpd
467 uucp-rlogin    uucp-rlogin Stuart Lynne
468 klogin          krcmd
469 kshell          new-who
470 new-rwho
471 dsf             rfs server
472 rmonitor        rmonitord
473 monitor         chcmd
474 9pf9            plan 9 file service
475 whoami          whoami
476 meterdemon
477 meterudemon
478 ipcserver       Sun IPC server
479 nqs nqs
480 urm             Cray Unified Resource Manager
481 sift-uft       Sender-Initiated/Unsolicited F
482 npmp-trap

```

```

610 npmp-local      npmp-local
611 npmp-gui        npmp-gui
634 ginadginad
666 mdqs
666 doom           doom Id Software
704 elcsd          errlog copy/server daemon
709 entrustmanager EntrustManager
729 netviewdm1     IBM NetView DM/6000 Server/Cli
730 netviewdm2     IBM NetView DM/6000 send
731 netviewdm3     IBM NetView DM/6000 receive/tc
741 netgwnetGW
742 netrcs          Network based Rev. Cont. Sys.
744 flexlm          Flexible License Manager
747 fujitsu-dev    Fujitsu Device Control
748 ris-cm          Russell Info Sci Calendar Mana
749 kerberos-adm   kerberos administration
750 rfile
751 pump
752 qrh
753 rrh
754 tell send
758 nlogin
759 con
760 ns
761 rxe
762 quotad
763 cycleserv
764 omserv
765 webster
767 phonebook      phone
769 vid
770 cadlock
771 rtip
772 cycleserv2
773 submit
774 rpasswd
775 entomb
776 wpages
780 wpgs
786 concert        Concert
800 mdbus_daemon
801 device
996 xtreelic       Central Point Software
997 maitrd
998 busboy
999 garcon
1000 cadlock

```

## Multicast

224.0.0.0	Base Address (Reserved)
224.0.0.1	All Systems on this Subnet
224.0.0.2	All Routers on this Subnet
224.0.0.3	Unassigned
224.0.0.4	DVMRP    Routers
224.0.0.5	OSPFIGP   OSPFIGP All Routers
224.0.0.6	OSPFIGP   OSPFIGP Designated Routers
224.0.0.7	ST Routers
224.0.0.8	ST Hosts
224.0.0.9	RIP2 Routers
224.0.0.10	IGRP Routers
224.0.0.11	Mobile-Agents
224.0.0.12-224.0.0.255	Unassigned
224.0.1.0	VMTCP Managers Group
224.0.1.1	NTP      Network Time Protocol
224.0.1.2	SGI-Dogfight
224.0.1.3	Rwhod
224.0.1.4	VNP
224.0.1.5	Artificial Horizons - Aviator
224.0.1.6	NSS - Name Service Server
224.0.1.7	AUDIONEWS - Audio News Multicast
224.0.1.8	SUN NIS+ Information Service
224.0.1.9	MTP Multicast Transport Protocol
224.0.1.10	IETF-1-LOW-AUDIO

224.0.1.11	IETF-1-AUDIO
224.0.1.12	IETF-1-VIDEO
224.0.1.13	IETF-2-LOW-AUDIO
224.0.1.14	IETF-2-AUDIO
224.0.1.15	IETF-2-VIDEO
224.0.1.16	MUSIC-SERVICE
224.0.1.17	SEANET-TELEMETRY
224.0.1.18	SEANET-IMAGE
224.0.1.19	MLLOADD
224.0.1.20	any private experiment
224.0.1.21	DVMRP on MOSPF
224.0.1.22	SVRLOC
224.0.1.23	XINGTV
224.0.1.24	microsoft-ds
224.0.1.25	nbc-pro
224.0.1.26	nbc-pfn
224.0.1.27-224.0.1.255	Unassigned
224.0.2.1	"rwho" Group (BSD) (unofficial)
224.0.2.2	SUN RPC PMAPPROC_CALLIT
224.0.3.000-224.0.3.255	RFE Generic Service
224.0.4.000-224.0.4.255	RFE Individual Conferences
224.0.5.000-224.0.5.127	CDPD Groups
224.0.5.128-224.0.5.255	Unassigned
224.0.6.000-224.0.6.127	Cornell ISIS Project
224.0.6.128-224.0.6.255	Unassigned
224.1.0.0-224.1.255.255	ST Multicast Groups
224.2.0.0-224.2.255.255	Multimedia Conference Calls
224.252.0.0-224.255.255.255	DIS transient groups
232.0.0.0-232.255.255.255	VMTP transient groups

## IP type of service

TOS Value	Description
0000	Default
0001	Minimize Monetary Cost
0010	Maximize Reliability
0100	Maximize Throughput
1000	Minimize Delay
1111	Maximize Security

Type of Service recommended values:

Protocol	TOS Value	
TELNET (1)	1000	(minimize delay)
FTP		
Control	1000	(minimize delay)
Data (2)	0100	(maximize throughput)
TFTP	1000	(minimize delay)
SMTP (3)		
Command phase	1000	(minimize delay)
DATA phase	0100	(maximize throughput)
Domain Name Service		
UDP Query	1000	(minimize delay)
TCP Query	0000	
Zone Transfer	0100	(maximize throughput)
NNTP	0001	(minimize monetary cost)
ICMP		
Errors	0000	
Requests	0000 (4)	
Responses	<same as request> (4)	
Any IGP	0010	(maximize reliability)
EGP	0000	
SNMP	0010	(maximize reliability)
BOOTP	0000	

## ICMP type numbers

Type	Name
0	Echo Reply
1	Unassigned
2	Unassigned
3	Destination Unreachable
4	Source Quench

```

5   Redirect
6   Alternate Host Address
7   Unassigned
8   Echo
9   Router Advertisement
10  Router Selection
11  Time Exceeded
12  Parameter Problem
13  Timestamp
14  Timestamp Reply
15  Information Request
16  Information Reply
17  Address Mask Request
18  Address Mask Reply
19  Reserved (for Security)
20-29 Reserved (for Robustness Experiment)
30  Traceroute
31  Datagram Conversion Error
32  Mobile Host Redirect
33  IPv6 Where-Are-You
34  IPv6 I-Am-Here
35  Mobile Registration Request
36  Mobile Registration Reply
37-255 Reserved



| Type | Name                                                                    |
|------|-------------------------------------------------------------------------|
| 0    | Echo Reply                                                              |
|      | Codes                                                                   |
|      | 0 No Code                                                               |
| 3    | Destination Unreachable                                                 |
|      | Codes                                                                   |
|      | 0 Net Unreachable                                                       |
|      | 1 Host Unreachable                                                      |
|      | 2 Protocol Unreachable                                                  |
|      | 3 Port Unreachable                                                      |
|      | 4 Fragmentation Needed and Don't Fragment was Set                       |
|      | 5 Source Route Failed                                                   |
|      | 6 Destination Network Unknown                                           |
|      | 7 Destination Host Unknown                                              |
|      | 8 Source Host Isolated                                                  |
|      | 9 Communication with Destination Network is Administratively Prohibited |
|      | 10 Communication with Destination Host is Administratively Prohibited   |
|      | 11 Destination Network Unreachable for Type of Service                  |
|      | 12 Destination Host Unreachable for Type of Service                     |
| 4    | Source Quench                                                           |
|      | Codes                                                                   |
|      | 0 No Code                                                               |
| 5    | Redirect                                                                |
|      | Codes                                                                   |
|      | 0 Redirect Datagram for the Network (or subnet)                         |
|      | 1 Redirect Datagram for the Host                                        |
|      | 2 Redirect Datagram for the Type of Service and Network                 |
|      | 3 Redirect Datagram for the Type of Service and Host                    |
| 6    | Alternate Host Address                                                  |
|      | Codes                                                                   |
|      | . 0 Alternate Address for Host                                          |
| 7    | Unassigned                                                              |
| 8    | Echo                                                                    |
|      | Codes                                                                   |
|      | 0 No Code                                                               |
| 9    | Router Advertisement                                                    |
|      | Codes                                                                   |
|      | 0 No Code                                                               |
| 10   | Router Selection                                                        |
|      | Codes                                                                   |
|      | 0 No Code                                                               |
| 11   | Time Exceeded                                                           |
|      | Codes                                                                   |
|      | 0 Time to Live exceeded in Transit                                      |
|      | 1 Fragment Reassembly Time Exceeded                                     |
| 12   | Parameter Problem                                                       |
|      | Codes                                                                   |
|      | 0 Pointer indicates the error                                           |


```

```

          1 Missing a Required Option
          2 Bad Length
13   Timestamp
Codes
          0 No Code
14   Timestamp Reply
Codes
          0 No Code
15   Information Request
Codes
          0 No Code
16   Information Reply
Codes
          0 No Code
17   Address Mask Request
Codes
          0 No Code
18   Address Mask Reply
Codes
          0 No Code
30   Traceroute
31   Datagram Conversion Error
32   Mobile Host Redirect
33   IPv6 Where-Are-You
34   IPv6 I-Am-Here
35   Mobile Registration Request
36   Mobile Registration Reply

```

## TCP options

Type	Length	Description
0	-	End of Option List
1	-	No-Operation
2	4	Maximum Segment Lifetime
3	3	WSOPT - Window Scale
4	2	SACK Permitted
5	N	SACK
6	6	Echo (obsoleted by option 8)
7	6	Echo Reply (obsoleted by option by 8)
8	10	TSOPT - Time Stamp Option
9	2	Partial Order Connection Permitted
10	5	Partial Order Service Profile
11		CC
12		CC.NEW
13		CC.ECHO
14	3	TCP Alternate Checksum Request
15	N	TCP Alternate Checksum Data
16		Skeeter
17		Bubba
18	3	Trailer Checksum Option

## Domain Names

Decimal	Name
0	Reserved
1	Internet (IN)
2	Unassigned
3	Chaos (CH)
4	Hessoid (HS)
5-65534	Unassigned
65535	Reserved

For the Internet (IN) class the following are defined:

TYPE	Value	Description
A	1	Host address
NS	2	Authoritative name server
MD	3	Mail destination (Obsolete - use MX)
MF	4	Mail forwarder (Obsolete - use MX)
CNAME	5	Canonical name for an alias
SOA	6	Start of a zone of authority
MB	7	Mailbox domain name
MG	8	Mail group member

MR	9	Mail rename domain name
NULL	10	Null RR
WKS	11	Well-known service description
PTR	12	Domain name pointer
HINFO	13	Host information
MINFO	14	Mailbox or mail list information
MX	15	Mail exchange
TXT	16	Text strings
RP	17	For Responsible Person
AFSDB	18	For AFS Data Base location

### Mail encoder header types

Keyword	Description
EDIFACT	EDIFACT format
EDI-X12	EDI X12 format
EVFU	FORTRAN format
FS	File System format
Hex	Hex binary format
LZJU90	LZJU90 format
LZW	LZW format
Message	Encapsulated Message
PEM	Privacy Enhanced Mail
PGP	Pretty Good Privacy
Postscript	Postscript format
Shar	Shell Archive format
Signature	Signature
Tar	Tar format
Text	Text
uuencode	uuencode format
URL	external URL-reference

### BOOTP and DHCP parameters

Tag	Name	Data Length	Meaning
0	Pad	0	None
1	Subnet Mask	4	Subnet Mask Value
2	Time Offset	4	Time Offset in Seconds from UTC
3	Gateways	N	N/4 Gateway addresses
4	Time Server	N	N/4 Timeserver addresses
5	Name Server	N	N/4 IEN-116 Server addresses
6	Domain Server	N	N/4 DNS Server addresses
7	Log Server	N	N/4 Logging Server addresses
8	Quotes Server	N	N/4 Quotes Server addresses
9	LPR Server	N	N/4 Printer Server addresses
10	Impress Server	N	N/4 Impress Server addresses
11	PLP Server	N	N/4 PLP Server addresses
12	Hostname	N	Hostname string
13	Boot File Size	2	Size of boot file in 512 byte chunks
14	Merit Dump File		Client to dump and name the file to dump it to
15	Domain Name	N	The DNS domain name of the client
16	Swap Server	N	Swap Server address
17	Root Path	N	Path name for root disk
18	Extension File	N	Path name for more BOOTP info
19	Forward On/Off	1	Enable/Disable IP Forwarding
20	SrcRte On/Off	1	Enable/Disable Source Routing
21	Policy Filter	N	Routing Policy Filters
22	Max DG Assembly	2	Max Datagram Reassembly Size
23	Default IP TTL	1	Default IP Time to Live
24	MTU Timeout	4	Path MTU Aging Timeout
25	MTU Plateau	N	Path MTU Plateau Table
26	MTU Interface	2	Interface MTU Size
27	MTU Subnet	1	All Subnets are Local
28	Broadcast Address	4	Broadcast Address
29	Mask Discovery	1	Perform Mask Discovery
30	Mask Supplier	1	Provide Mask to Others
31	Router Discovery	1	Perform Router Discovery
32	Router Request	4	Router Solicitation Address
33	Static Route	N	Static Routing Table
34	Trailers	1	Trailer Encapsulation
35	ARP Timeout	4	ARP Cache Timeout
36	Ethernet	1	Ethernet Encapsulation
37	Default TCP TTL	1	Default TCP Time to Live

38	Keepalive Time	4	TCP Keepalive Interval
39	Keepalive Data	1	TCP Keepalive Garbage
40	NIS Domain	N	NIS Domain Name
41	NIS Servers	N	NIS Server Addresses
42	NTP Servers	N	NTP Server Addresses
43	Vendor Specific	N	Vendor Specific Information
44	NETBIOS Name Srv	N	NETBIOS Name Servers
45	NETBIOS Dist Srv	N	NETBIOS Datagram Distribution
46	NETBIOS Note Type	1	NETBIOS Note Type
47	NETBIOS Scope	N	NETBIOS Scope
48	X Window Font	N	X Window Font Server
49	X Window Manager	N	X Window Display Manager
50	Address Request	4	Requested IP Address
51	Address Time	4	IP Address Lease Time
52	Overload	1	Overload "sname" or "file"
53	DHCP Msg Type	1	DHCP Message Type
54	DHCP Server Id	4	DHCP Server Identification
55	Parameter List	N	Parameter Request List
56	DHCP Message	N	DHCP Error Message
57	DHCP Max Msg Size	2	DHCP Maximum Message Size
58	Renewal Time	4	DHCP Renewal (T1) Time
59	Rebinding Time	4	DHCP Rebinding (T2) Time
60	Class Id	N	Class Identifier
61	Client Id	N	Client Identifier
62	Netware/IP Domain	N	Netware/IP Domain Name
63	Netware/IP Option	N	Netware/IP sub Options
128-154	Reserved		
255	End	0	None

## Directory system names

Keyword	Attribute (X.500 keys)
CN	CommonName
L	LocalityName
ST	StateOrProvinceName
O	OrganizationName
OU	OrganizationalUnitName
C	CountryName

## Content types and subtypes

Type	Subtype
text	plain richtext tab-separated-val
multipart	mixed alternative digest parallel appledouble header-set
message	rfc822 partial external-body news
application	octet-stream postscript oda atomicmail andrew-inset slate wita dec-dx dca-rft activemessage rtf applefile mac-binhex40 news-message-id news-transmission wordperfect5.1 pdf zip

	macwriteii
	msword
	remote-printing
image	jpeg
	gif
	ief
	tiff
audio	basic
video	mpeg
	quicktime

### Character Sets

US-ASCII	ISO-8859-1	ISO-8859-2	ISO-8859-3
ISO-8859-4	ISO-8859-5	ISO-8859-6	ISO-8859-7
ISO-8859-8	ISO-8859-9		

### Access Types

FTP	ANON-FTP	TFTP	AFS
LOCAL-FILE	MAIL-SERVER		

### Conversion Values

7BIT	8BIT	BASE64	BINARY
QUOTED-PRINTABLE			

### MIME / X.400 mapping table

MIME content-type	X.400 Body Part
text/plain	
charset=us-ascii	ia5-text
charset=iso-8859-x	Extended Body Part - GeneralText
text/richtext	no mapping defined
application/oda	Extended Body Part - ODA
application/octet-stream	bilaterally-defined
application/postscript	Extended Body Part - mime-postscript-body
image/g3fax	q3-facsimile
image/jpeg	Extended Body Part - mime-jpeg-body
image/gif	Extended Body Part - mime-gif-body
audio/basic	no mapping defined
video/mpeg	no mapping defined

### X.400 to MIME Table

X.400 Basic Body Part	MIME content-type
ia5-text	text/plain; charset=us-ascii
voice	No Mapping Defined
g3-facsimile	image/g3fax
g4-class1	no mapping defined
teletex	no mapping defined
videotex	no mapping defined
encrypted	no mapping defined
bilaterally-defined	application/octet-stream
nationally-defined	no mapping defined
externally-defined	See Extended Body Parts

### X.400 Extended body part conversion

X.400 Extended Body Part	MIME content-type
GeneralText	text/plain; charset=iso-8859-x
ODA	application/oda
mime-postscript-body	application/postscript
mime-jpeg-body	image/jpeg
mime-gif-body	image/gif

### Inverse ARP

Number	Operation Code (op)
1	REQUEST
2	REPLY
3	request Reverse
4	reply Reverse
5	DRARP-Request
6	DRARP-Reply
7	DRARP-Error
8	InARP-Request
9	InARP-Reply
10	ARP-NAK

Number	Hardware Type (hrd)
1	Ethernet (10Mb)
2	Experimental Ethernet (3Mb)
3	Amateur Radio AX.25
4	Proteon ProNET Token Ring
5	Chaos
6	IEEE 802 Networks
7	ARCNET
8	Hyperchannel
9	Lanstar
10	Autonet Short Address
11	LocalTalk
12	LocalNET
13	Ultra link
14	SMDS
15	Frame Relay
16	Asynchronous Transmission Mode
17	HDLC
18	Fibre Channel
19	Asynchronous Transmission Mode
20	Serial Line
21	Asynchronous Transmission Mode

## IEEE 802 numbers of interest

Link Service Access Point				
IEEE	Internet			
binary	binary	decimal	Description	
00000000	00000000	0	Null LSAP	
01000000	00000010	2	Indiv LLC Sublayer Mgt	
11000000	00000011	3	Group LLC Sublayer Mgt	
00100000	00000100	4	SNA Path Control	
01100000	00000110	6	Reserved (DOD IP)	
01110000	00001110	14	PROWAY-LAN	
01110010	01001110	78	EIA-RS 511	
01111010	01011110	94	ISI IP	
01110001	10001110	142	PROWAY-LAN	
01010101	10101010	170	SNAP	
01111111	11111110	254	ISO CLNS IS 8473	
11111111	11111111	255	Global DSAP	

## IANA Ethernet address block

The Internet Assigning Numbers Authority (IANA) owns the starting Ethernet address of:

0000 0000 0000 0000 0111 1010 (which is 00-00-5E)

This address can be used with the multicast bit (which is the first bit to the address) to create an Internet Multicast. It has the form:

```
1000 0000 0000 0000 0111 1010 xxxx xxxx0 xxxx xxxx xxxx xxxx
| | | | | |
Multicast Bit 0 = Internet Multicast
               1 = Assigned by IANA for
                  other uses
```

This gives an address range from 01-00-5E-00-00-00 to 01-00-5E-7F-FF-FF .

### P.16.2 Ethernet vendor address

An Ethernet address is 48 bits. The first 24 bits identifies the manufacturer and the next 24 bits identifies the serial number. The manufacturer codes include:

00000C	Cisco
00000E	Fujitsu
00000F	NeXT
000010	Sytek
00001D	Cabletron
000020	DTAB
000022	Visual Technology
00002A	TRW
000032	GEC Computers Ltd
00005A	S & Koch
00005E	IANA
000065	Network General
00006B	MIPS
000077	MIPS
00007A	Ardent
000089	Cayman Systems
000093	Proteon
00009F	Ameristar Technology
0000A2	Wellfleet
0000A3	NAT
0000A6	Network General
0000A7	NCD
0000A9	Network Systems
0000AA	Xerox
0000B3	CIMLinc
0000B7	Dove
0000BC	Allen-Bradley
0000C0	Western Digital
0000C5	Farallon phone net card
0000C6	HP INO
0000C8	Altos
0000C9	Emulex
0000D7	Dartmouth College
0000DD	Gould
0000DE	Unigraph
0000E2	Acer Counterpoint
0000EF	Alantec
0000FD	High Level Hardvare
000102	BBN
001700	Kabel
008064	Wyse Technology
00802D	Xylogics, Inc.
00808C	Frontier Software Development
0080C2	IEEE 802.1 Committee
0080D3	Shiva
00AA00	Intel
00DD00	Ungermann-Bass
00DD01	Ungermann-Bass
020701	Racal InterLan
020406	BBN
026086	Satelcom MegaPac
02608C	3Com
02CF1F	CMC
080002	3Com
080003	ACC
080005	Symbolics
080008	BBN
080009	Hewlett-Packard
08000A	Nestor Systems
08000B	Unisys
080011	Tektronix, Inc.
080014	Excelan
080017	NSC
08001A	Data General
08001B	Data General

```

08001E Apollo
080020 Sun
080022 NBI
080025 CDC
080026 Norsk Data (Nord)
080027 PCS Computer Systems
080028 TI
08002B DEC
08002E Metaphor
08002F Prime Computer
080036 Intergraph
080037 Fujitsu-Xerox
080038 Bull
080039 Spider Systems
080041 DCA
080046 Sony
080047 Sequent
080049 Univation
08004C Encore
08004E BICC
080056 Stanford University
08005A IBM
080067 Comdesign
080068 Ridge
080069 Silicon Graphics
08006E Concurrent
080075 DDE
08007C Vitalink
080080 XIOS
080086 Imagen/QMS
080087 Xplex
080089 Kinetics
08008B Pyramid
08008D XyVision
080090 Retix Inc
800010 AT&T

```

### Ethernet multicast addresses

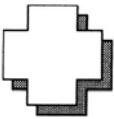
An Ethernet multicast address has a multicast bit, a 23-bit vendor identifier part and a 24-bit vendor assigned part.

Ethernet Address	Type Field	Usage
01-00-5E-00-00-00-	0800	Internet Multicast
01-00-5E-7F-FF-FF		
01-00-5E-80-00-00-	????	Internet reserved by IANA
01-00-5E-FF-FF-FF		
01-80-C2-00-00-00	-802-	Spanning tree (for bridges)
09-00-02-04-00-01?	8080?	Vitalink printer
09-00-02-04-00-02?	8080?	Vitalink management
09-00-09-00-00-01	8005	HP Probe
09-00-09-00-00-01	-802-	HP Probe
09-00-09-00-00-04	8005?	HP DTC
09-00-1E-00-00-00	8019?	Apollo DOMAIN
09-00-2B-00-00-00	6009?	DEC MUMPS?
09-00-2B-00-00-01	8039?	DEC DSM/DTP?
09-00-2B-00-00-02	803B?	DEC VAXELN?
09-00-2B-00-00-03	8038	DEC Lanbridge Traffic Monitor (LTM)
09-00-2B-00-00-04	????	DEC MAP End System Hello
09-00-2B-00-00-05	????	DEC MAP Intermediate System Hello
09-00-2B-00-00-06	803D?	DEC CSMA/CD Encryption?
09-00-2B-00-00-07	8040?	DEC NetBios Emulator?
09-00-2B-00-00-0F	6004	DEC Local Area Transport (LAT)
09-00-2B-00-00-1x	????	DEC Experimental
09-00-2B-01-00-00	8038	DEC LanBridge Copy packets
09-00-2B-01-00-01	8038	DEC LanBridge Hello packets
09-00-4E-00-00-02?	8137?	Novell IPX
09-00-56-00-00-00-	????	Stanford reserved
09-00-56-FE-FF-FF		
09-00-56-FF-00-00-	805C	Stanford V Kernel, version 6.0
09-00-56-FF-FF-FF		
09-00-77-00-00-01	????	Retix spanning tree bridges

09-00-7C-02-00-05	8080?	Vitalink diagnostics
0D-1F-15-BA-DD-06	????	HP
AB-00-00-01-00-00	6001	DEC Maintenance Operation Protocol
AB-00-00-02-00-00	6002	DEC Maintenance Operation Protocol
AB-00-00-03-00-00	6003	DECNET Phase IV end node Hello
AB-00-00-04-00-00	6003	DECNET Phase IV Router Hello packets
AB-00-00-05-00-00	????	Reserved DEC through
AB-00-03-FF-FF-FF		
AB-00-03-00-00-00	6004	DEC Local Area Transport (LAT) - old
AB-00-04-00-xx-xx	????	Reserved DEC customer private use
AB-00-04-01-xx-yy	6007	DEC Local Area VAX Cluster groups
CF-00-00-00-00-00	9000	Sys. Communication Architecture (SCA) Ethernet Configuration Test protocol (Loopback)

### Ethernet broadcast address

FF-FF-FF-FF-FF-FF	0600	XNS packets, Hello or gateway search?
FF-FF-FF-FF-FF-FF	0800	IP (e.g. RWHOD via UDP) as needed
FF-FF-FF-FF-FF-FF	0804	CHAOS
FF-FF-FF-FF-FF-FF	0806	ARP (for IP and CHAOS) as needed
FF-FF-FF-FF-FF-FF	0BAD	Banyan
FF-FF-FF-FF-FF-FF	1600	VALID packets, Hello or gateway search?
FF-FF-FF-FF-FF-FF	8035	Reverse ARP
FF-FF-FF-FF-FF-FF	807C	Merit Internodal (INP)
FF-FF-FF-FF-FF-FF	809B	EtherTalk



---

## Index

---

- 01111110, 29, 120, 565, 566, 568, 573, 576, 762, 780  
1.411200Mbps, 122  
1.536Mbps, 561, 562  
1.544Mbps, 108, 250, 562  
1.920Mbps, 561, 562  
100Ω, 747, 748  
100BASE-FX, 513  
100Mbps, 472, 513–515, 521, 530, 763, 764  
100VG-AnyLAN, 491, 513–521, 762  
10BASE2, 467, 509, 511, 521  
10BASE5, 509, 511  
10BASE-FL, 509, 510  
10BASE-T, 467, 509, 510, 511, 513, 514, 518, 520  
10Mbps, 81, 92, 282, 464, 465, 467, 472, 493, 497, 503, 511–521, 530, 540, 541, 559, 610, 683, 698, 691, 746, 748, 762, 763, 764  
128kbps, 126, 128, 250, 562  
1284 Negotiation, 674  
128-bit code, 189  
128-bit key, 181, 182, 187, 188  
128kbps, 124, 126, 128, 250, 562  
12MHz, 10, 546, 704  
13.5MHz, 78  
14-bit linear coding, 111  
150Ω, 528  
155Mbps, 282, 520, 554, 559  
16.7million, 13, 32, 34, 35, 38, 278  
16450, 710, 808  
16kbps, 107, 108, 110, 560–565  
16Mbps, 282, 468, 471, 520, 522, 527, 529, 748  
192kbps, 121, 127, 128, 134, 562, 563  
2.048 Mbps, 562  
2.4Gbps, 554  
2's complement, 301, 383  
200Mbps, 530  
24-bit color, 13, 32, 35, 38, 39, 51  
256 colors, 32–35, 38, 49, 278, 684  
25Hz, 73  
28.8kbps, 250, 560, 594–596  
2D luminance, 86  
2Mbps, 81, 83, 92  
300bps, 594, 596, 597, 605  
32kbps, 101, 562  
32kHz, 111, 113, 115, 116, 126–128, 132–134  
4:1:1, 52, 79, 84  
4:2:0, 85, 88, 95  
4:2:2, 79, 84, 88  
4kHz, 107, 108, 123, 594  
4004, 5, 839  
400Mbps, 514, 518  
44.1kHz, 10, 111–116, 119, 121, 122, 126–128, 133, 134, 546, 683  
48kHz, 111, 113, 115, 116, 126–128, 132, 134  
4B/5B, 532, 533, 535, 761  
4Mbps, 81, 91, 92, 522, 527, 529, 748  
5 channel, 124  
5.1-channel, 124, 127, 128, 134  
50Ω, 747, 756  
50Hz, 73, 74  
525 lines, 74, 78, 121  
60Hz, 73, 121  
62.5/125, 538, 540  
622Mbps, 554  
625lines, 74, 78  
64kbps, 103, 107, 108, 121, 124, 126, 250, 460, 547, 560–563, 565, 580  
6MHz, 10, 546  
75Ω, 78, 112, 747  
8kHz, 107, 108, 243, 560  
80386, 6, 839, 841, 843, 844, 845, 847–849, 852, 839, 841, 843–845, 847–849, 852  
80486, 6, 685, 839–841, 843–850, 852, 853  
pin out, 845  
signal lines, 846  
8080, 5, 839  
8086, 5, 710, 839, 840–845, 849, 851, 852  
8088, 840  
    accumulator, 841  
    addressing registers, 842  
    connections, 841  
    count register, 717, 841  
    processor flags, 843  
    status registers, 842  
82091AA, 695, 701, 812

- 8237, 698, 705  
82438, 695, 699  
8250, 620, 808, 812  
8B6T, 515, 763, 764  
8mm video tape, 710, 726  
8-to-14 bit modulation, 119, 120, 762  
90kHz reference clock, 85, 89  
96Mbps, 514, 518  
A4 image, 608  
AA, 188, 601, 781, 814, 939  
AAL  
  functionality, 554  
  service levels, 554  
  services, 555  
  type 2, 555  
  type 3/4, 555, 556  
  type 5, 557  
AAL1, 555  
AC, 58, 60, 68, 121, 122, 132–134, 238, 523, 563, 781, 811, 939  
AC components, 58, 60, 68  
AC-1, 121, 132  
AC-2, 121, 132  
AC-3, 121, 122, 132–134  
Accuracy, 7, 8, 55, 116, 131, 132, 547  
ACK, 145, 569, 571, 661, 779, 813, 939  
Acknowledgement, 213, 214, 568, 578, 580, 584, 586, 588, 589, 591, 667, 730, 840  
Action, 341, 344  
Adaptation layer, 550  
Adaptive delta modulation PCM, 22, 100  
Adaptive dictionary data compression, 26  
Adaptive Huffman, 25, 26  
ADC, 8, 96, 100, 110, 116, 547, 668, 756  
Address bus, 676–679, 682, 685, 691, 696, 739, 832, 834, 839, 840, 844–846, 848, 853  
Address field, 198, 468, 476, 499, 502, 536, 551, 565, 566, 573, 574, 593  
Address filtering, 257, 260  
Addressable memory, 678, 694, 739  
Addressing  
  implied, 843  
ADPCM, 101, 102, 108, 555, 939, 944  
Advanced power management, 707  
Advanced television, 122  
AES, 113, 114, 117  
AES/EBU/IEC, 113  
AES10-1991, 117  
AES11-1990, 117  
AES18-1992, 117  
AES3, 113–117  
AES3-1995, 113  
AFI, 593, 939  
AIP, 695, 701–703, 812  
AIX, 439  
A-Law, 102–106  
Alphabet shifting, 174  
ALT, 279, 324, 338–340, 819, 864, 873, 890  
AM, 6, 794, 939  
Amber, 2  
America On-line, 250  
American, 2, 3, 68, 113, 275, 296, 463  
American Civil War, 3  
AMI, 562, 563, 939  
Amphere, 2  
Amplitude modulation, 6, 75, 606, 939, 943  
Analogue, 1, 7, 8, 10, 11, 22, 80, 82, 96, 98, 110, 111, 132, 146, 458–460, 546, 547, 561, 608, 818, 821  
Analogue systems, 7, 8, 146  
Anchors, 281  
AND, 135, 136, 169, 300–302, 304, 383, 384  
Angle, 76, 77, 323, 324, 538, 754, 755, 770, 858  
Animation, 9, 29, 248, 253, 256, 329  
Anonymous, 254  
ANSI, 7, 113, 117, 255, 275, 463, 464, 531, 538, 542, 546, 550, 625, 715, 779, 924  
ANSI S4.40, 113  
API, 220, 231, 232, 240, 329, 404, 439, 702, 812  
APIC, 695, 698, 853  
Apple, 5, 232, 293, 432  
Applet  
  basics, 326  
  creating, 325  
  methods, 326  
  tag, 324  
Applet viewer, 294, 324  
AppleTalk, 412, 413, 415, 419, 468, 469, 470  
AppletViewer.exe, 325  
Application layer, 461  
Application level gateway, 257, 258  
ARC, 29  
Archive, 31, 410  
Argument, 325  
Arithmetic operator, 299, 381, 382  
ARJ, 29  
Arp, 220, 275, 440, 493, 789  
Arrays, 172, 315, 324, 453  
  multi-dimensional, 327  
ASCII, 7, 27, 115, 148, 154, 176, 178, 179, 217, 233, 235, 240, 242–246, 275, 296, 298, 301, 307, 383, 452, 461, 595–597, 612–615, 617, 636, 779, 781

- BEL, 779
- bit stream timings, 614
- CR, 152, 235–237, 599, 620, 623, 779
- DC1, 779
- DC2, 779
- DC3, 779
- DC4, 779
- EOT, 571, 779
- FF, 245, 276, 507, 525, 535, 779, 782
- HT, 779
- LF, 235–237, 779
- NUL, 779
- SUB, 269, 779
- VT, 779
- ASK, 337, 338, 605, 606, 939
- Aspect ratio, 35, 36, 37, 82
- Assembly, 810, 811, 836–838
- Asymmetrical compression, 87
- Asynchronous, 398, 399, 448, 463, 535, 536, 567, 572, 573, 575, 577, 578, 582, 595, 596, 602, 612, 614, 682, 708, 808, 818, 939, 944
- AT, 597, 598, 600–604, 678, 715, 716, 796, 816, 819, 939
- AT commands, 597, 816
- AT task file, 716
  - adaptation layer, 550, 553, 554, 924
  - ATM, 151, 282, 289, 290, 520, 542, 545–559, 791–793, 795, 796, 797, 924, 939
  - backbone, 549
  - cells, 549, 550, 553
  - data link, 554
  - flow control, 558
  - gateway, 553
  - header (GFC), 551
  - header (VPI), 551, 552, 554
  - OSI model, 553, 555
  - physical layer, 554
  - pipe, 558
  - routing cells, 550
  - switches, 550, 558
  - UNI, 549, 550, 553, 558, 793
- Attachment stations, 539
- Attenuation, 466, 540, 543, 544, 746–749, 756, 757, 770, 820
- Attributes, 389, 410, 411, 479, 481–483, 860, 861, 862
- AU, 248, 390, 405
- Audio, 7, 8, 10, 22, 83, 85, 89–93, 96, 98, 108, 110–118, 121–133, 145, 181, 240, 243, 256, 262, 291, 292, 329, 403, 541, 546, 555, 722–726, 740, 761, 806, 879, 940
- digital, 81, 110, 111, 116, 117, 118, 121, 122, 723, 765
- Engineering Society, 113
- flag, 115
- Signals, 96, 110
- subframe, 114
- Authentication, 44, 171, 211, 261, 448, 477, 480, 487
- Auto halt powerdown, 852
- Automatic power control, 723
- Automatically answer, 594, 597
- Automatically dial, 594
- AVI, 248
- Back reference, 29, 30
- Backbone, 459, 465, 477, 510, 519, 537, 538, 549
- Backbone network, 537, 538, 549
- Background images, 277
- Backspace, 297, 298, 380
- Baltimore, 3
- Bank cards, 170
- Baseband video, 70, 73
- Baud, 595, 596, 606, 607, 614–616, 623–644, 650, 651, 808
- Baud rate, 595, 596, 606, 614, 615, 623, 624, 651, 808
- Baudot, 7
- BCC, 233
- BCD, 592, 593, 939
- Bee-sting, 510
- Bell, 2, 3, 4, 6, 439, 596, 816, 819
- BELL, 297, 779, 816
- Bell Lab, 4, 439
- BER, 146
- Binary, 10, 21, 26, 27, 30, 37, 38, 58, 59, 60, 62, 81, 108, 109, 113, 115, 135, 136, 139, 143, 146, 151, 155, 161, 170, 178, 217, 220, 240, 243, 245, 291, 296, 301, 302, 391, 412, 430, 451, 498, 515, 533, 562, 568, 584, 592, 605, 657, 661, 696, 710, 763, 764, 779, 786, 836, 939
  - code tree, 26
  - manipulation, 136
- Binary symmetric channel, 146
- Biod, 448, 449, 798
- BIOS, 393–395, 620, 621, 654, 655, 691, 692, 707, 714, 721, 824, 825, 830, 834, 939
- BIOS printer, 654, 655
- Bi-phase mark coding, 113
- B-ISDN, 561
- BIST, 693
- Bit allocation decoder, 130
- Bit allocations, 132

- Bit error rate, 114, 146  
Bit rate, 22, 78, 79, 81, 83, 88, 91–93, 98, 103, 107, 108, 110, 114, 121, 122, 124, 126–128, 133, 134, 460, 464–468, 471, 472, 497, 502, 513–516, 518, 522, 528, 529, 537, 538, 547–555, 557, 560, 563, 565, 595, 596, 606–608, 610, 614, 642, 643, 708, 746, 753, 762–764, 809, 811  
Bit shift, 162, 170, 179, 302, 384  
Bit stuffing, 566  
Bitmapped, 49, 51, 821, 822  
Bitmapped image, 12, 33, 35  
Bitstream, 85, 89, 90, 93, 95, 129–132, 144, 165, 169, 245, 502, 504, 505, 533, 535, 563, 566, 568, 608, 759, 760, 761, 762  
Bitwise, 169, 298, 301–304, 381, 383–385, 638, 664  
AND, 135, 136, 169, 300–302, 304, 383, 384  
operator, 301–303, 383  
OR, 300–302, 304, 335, 336, 383, 384, 659, 732  
XOR, 301, 304, 613  
Black, 38, 39, 47, 52, 68, 69, 74, 75, 78, 80, 271, 276, 292, 608, 868  
Black run length coding, 47, 609  
Black-box, 292  
Blanking, 74, 75, 81  
Block terminator, 43, 44  
Blue Book, 724  
Blue sky, 39  
Blueness, 52  
BMP, 9, 33, 34, 939  
BNC, 467, 469, 504, 510, 511, 748  
Bold, 238, 269, 270, 737, 860, 862, 863  
Boolean, 331  
BOOTP, 206, 207, 441, 491, 492, 494, 789, 795  
Borland C, 836–838  
Bps, 595–598, 601, 605, 606, 608, 610, 614, 637, 816, 819, 822, 823, 939  
Braces, 304, 307, 308, 326, 386, 387  
Brackets, 324  
Brain, 83, 122  
Braun, 4  
Bridge, 195, 196, 208, 210, 215, 218, 261, 430, 459, 465–469, 485, 490, 518, 520, 521, 530, 549, 602, 604, 685, 688, 689, 691–693, 695, 698, 710, 854, 855  
example, 469  
filtering rate, 467  
forward rate, 467, 468  
source routing, 468  
spanning tree, 468  
Brightness, 51, 52, 68, 74, 75, 608, 868  
Broadband ISDN, 561  
Broadcast communications equipment, 126  
Browser, 247, 248, 250, 251, 253, 256, 262, 269–273, 277, 281, 284, 291, 292, 294, 324–326, 366, 376, 865  
BS, 779  
B-tree, 412  
Buffer, 89, 225–227, 504, 507, 522, 536, 537, 557, 615–618, 621, 622–624, 626, 627, 633–635, 637, 639, 640–652, 681, 688, 710, 715, 716, 719, 738, 808, 903, 914  
Bus  
arbitration, 691  
enumerator, 394, 395  
network, 470, 472, 497  
Button, 287, 289, 331, 334, 340–346, 350, 355, 368, 660, 661  
Byte enable, 681, 688–690, 696, 846, 848  
Byte enable signals, 689, 846, 848  
Bytecodes, 292  
C program, 15, 19, 36, 39, 41, 46, 50, 53, 169, 175, 176, 286, 378, 661, 784  
C/C++, 292, 293, 625, 633, 634  
C++, 292, 293, 313, 377, 380, 386, 625, 633, 634, 720, 837, 838  
Cable, 1, 810  
characteristics, 746  
impedance, 511  
type, 464, 511, 528, 746  
Cabling, 461, 528  
Cache, 264, 266–268, 397, 493, 496, 693, 695, 699, 700, 701, 709, 713, 796, 839, 845, 847, 849–854  
architecture, 850  
L-1, 852–854  
L-2, 695, 852–854  
look-through, 850, 851  
write-back, 850  
write-through, 850  
CAD, 537, 939  
Calling party number, 569  
CAN, 510, 779  
Capacitance, 747  
Carriage return, 235, 245, 297, 380, 461, 597, 600, 655, 900  
Cartridge, 709, 725, 853, 943  
Cat-3, 513–515, 522, 528, 529, 747–750, 763, 764  
Cat-4, 513, 522, 747–750  
Cat-5, 513, 514, 521, 522, 528, 529, 748, 749, 750

- CATNIP, 211  
 cc:Mail, 232, 233, 243  
 CCIR 601, 52  
 CCIR-Rec. 647, 113  
 CCITT, 33, 45, 47, 103, 115, 149–151, 239, 463, 464, 546, 560–562, 568, 576, 581, 593, 596, 601, 779, 816, 939  
 CCITT J.17, 115  
 CD, 7, 13, 83, 111, 116–126, 144, 463, 464, 497, 498, 502, 508, 518, 521, 601, 642, 646, 648, 649, 653, 710, 722, 723, 724, 762, 782, 814, 836, 939, 940  
 audio, 118, 723  
 pit, 8, 119, 710, 721, 722  
 CD/EIAJ/RDAT, 111  
 CD-DA, 723, 724  
 CD-E, 723, 724  
 CD-Enhanced, 724  
 CDFS, 389, 397, 399, 804  
 CD-I, 724  
 CD-quality audio, 123, 124, 126  
 CD-R, 722–724  
 CD-ROM, 7, 13, 83, 93, 122, 144, 389, 394, 397, 628, 633, 653, 666, 670, 672, 710, 713, 714, 717, 722–724, 727, 739, 836  
 disk format, 723  
 format, 723  
 lead in, 723, 724  
 lead out, 723, 724  
 program area, 723  
 CD-ROM-XA, 723, 724  
 CD-RW, 722, 723  
 Centronics, 653, 654, 666, 668, 674  
 CERN, 247  
 CGI, 286  
 script, 286  
 bin, 286, 287  
 CGM, 940  
 Channel identification, 569  
 Character alphabet, 27, 275  
 Characteristic impedance, 503, 508, 528, 746–748, 768, 770, 771, 773–776  
 Characters, 296, 297, 301, 383, 614, 633, 857  
 Cheapernet, 509, 510  
 Checkboxes, 347  
 Checksum, 148, 214, 428, 792  
 Chromatic, 75, 77, 78  
 Chrominance, 52, 72, 76, 79, 80, 83, 84, 86, 88, 89  
 CIF, 84, 90, 941  
 Circuit-switched, 250, 541, 548, 560, 561, 580  
 Cladding, 538, 754–756  
 Classes, 68, 201, 293, 294, 309, 311, 317, 319–321, 359, 554, 868, 877, 892, 898–900, 911  
 code, 692  
 declaration, 333, 339  
 libraries, 320, 321  
 Classes.zip, 293, 294, 320  
 Client, 220, 230, 250, 255–257, 262–268, 292, 359, 370, 371, 376, 377, 407, 412–414, 424, 425, 430, 432, 436, 440, 442, 444–448, 455–458, 491–496, 520, 795, 798, 800  
 Client/server, 215, 230, 250, 262, 292, 376, 412, 413, 440, 442, 457, 491  
 Clipper chip, 171  
 Clock doubler, 839, 845, 839, 845  
 Clock period, 112, 508, 849  
 CMC, 231, 232  
 CMOS, 940  
 Coaxial, 112, 464, 471, 472, 497, 503, 504, 508, 527, 610, 746–748, 767, 771  
 Code  
 block, 143, 161  
 convolutional, 143, 161  
 cyclic, 143, 161  
 digital, 1, 10, 22, 96, 98, 135, 547  
 feedback, 144, 505, 558, 618  
 feedforward, 144  
 gains, 146  
 mappings, 174  
 systematic code, 144  
 Coding  
 error, 135, 147, 154  
 gain, 146  
 methods, 16  
 Morse code, 17  
 simple, 17  
 source, 16, 22  
 Collision, 463, 497, 498, 502–504, 509, 511, 513, 568, 940  
 Collision detection, 497, 940  
 Color, 12, 13, 21, 32, 34–44, 49, 51, 52, 62, 68, 69–80, 83–85, 122, 271, 277, 305, 401, 517, 608, 751, 868  
 burst, 75, 77  
 conversion, 52  
 difference, 69–71, 80, 84  
 modulation, 75  
 pixels, 32  
 saturation, 69  
 COM1, 611, 616, 620, 621, 624, 625, 628, 635–639, 644, 649, 650, 657, 702, 707, 739, 812, 825, 828, 830, 836

- COM2, 611, 620, 621, 636, 651, 652, 657, 701, 707, 739, 744, 825, 828, 830, 836  
Combinations, 120, 140, 174, 175, 606, 607, 762, 765  
Comments, 269, 386–388, 793, 795, 861  
Commodore, 5  
Common words  
  standard English, 170  
Communications  
  digital, 1  
Communications channel, 456  
Communicator, 324  
Compact, 26, 29, 116, 122, 126, 390, 405, 722, 725, 863  
Compact cassette, 116, 122, 126  
Compact disc, 111  
Companding, 103, 104, 132  
Compatibility, 128, 220, 328, 414, 438, 514, 596, 666, 678, 682, 810, 811, 839, 849, 839, 849  
Compilation, 322  
Compiler, 293, 294, 295, 321, 326  
Composite video, 76, 78  
  signals, 78  
Compress, 12, 28, 29, 30, 31, 32, 33, 49, 51, 58, 87, 102, 121, 122, 126, 292, 594, 606, 608  
Compressed images, 83, 248  
Compressed TV, 81  
Compression, 1, 12, 13, 16, 17, 21, 24–34, 42, 45, 47–52, 55, 57, 58, 62, 68, 81, 83–85, 87, 90–93, 102–104, 106–108, 121, 122, 124–126, 132, 266, 268, 278, 410, 438, 596, 606, 608, 672, 819  
  ratio, 29, 33, 34, 47, 51, 81, 92, 608, 672  
  time, 52  
CompuServe, 34, 250, 460, 600, 628  
Computational difficulty, 172  
Computer-generated, 7  
Computer-type data, 725  
Concatenate, 385  
Concatenated, 297  
Concentrator, 470, 539, 944  
Configuration Manager, 391, 393–395  
Connection-oriented, 213, 414, 428, 555  
Connectors, 1, 468, 510, 518, 522, 527, 528, 611, 682, 714, 727, 746, 748, 810, 811  
Consecutive zeros, 60, 532, 533, 535  
Constants, 317, 324, 865, 868, 872, 877, 885, 890–892, 905  
Constructors, 312, 864–868, 871–924  
Consumer flag, 115  
Continuous, 119, 143, 161, 216, 720  
Control field, 500, 501, 523–525, 535, 550, 551, 565–567, 572, 574, 575, 578, 737  
Control field, 268, 535, 566, 567, 573, 574, 737  
Control panel, 828, 834  
Control signals, 847  
Control token, 522, 523, 525  
Convolutional code  
  coding tree, 162  
  k/n code, 161  
  maximum likelihood, 165  
  most favored route, 169  
  rate code, 143  
  Trellis diagram, 163–169, 607  
  Viterbi decoder, 164, 165  
Cooke, 3  
Copy/copyright flag, 115  
Core bit allocations, 132  
Core system components, 399  
Correction, 117, 135, 144, 145, 149, 154, 155, 160, 461, 571, 579, 596, 607, 699, 700, 817, 855  
Cosine, 22, 51, 52, 89, 125, 132, 133, 323  
Coulomb, 2  
Courier, 170  
CP-340-type I, 113  
CPU, 437, 707  
CR, 152, 235–237, 599, 620, 623  
CRC, 115, 117, 128, 143, 149–152, 161, 498, 499, 524, 536, 556, 557, 568, 576  
  CRC-10, 151  
  CRC-12, 151  
  CRC-16, 149, 151, 576  
  CRC-32, 151  
  CRC-8, 151  
  CRC-CCITT, 149–151, 576  
    decoding example, 150  
Crosstalk, 145, 146, 746, 748–750, 757  
CRT, 940  
Cryptographic keys, 171  
Cryptographics, 170  
CSDN, 580, 940  
CSMA, 463, 464, 497, 498, 502, 508, 518, 521, 940  
CSMA/CA, 940  
CSMA/CD, 463, 464, 497, 498, 502, 508, 518, 521, 940  
Ctrl, 337, 338, 873  
Ctrl-Alt-Del, 830  
Ctrl-Break, 830  
CTS, 612, 616, 618, 620, 646–649, 809, 814, 817, 818, 823, 940  
Cube, 327

- Current, 331
- Cyclic redundancy check, 143, 499, 524, 568, 576, 720
- Cyclic redundancy code, 128
- Cylinders, 712, 720, 721, 754
- DA, 523, 536, 705, 782, 814, 940
- DAC, 9, 98, 539
- Daemons, 448
- DARPA, 195, 201, 788
- DAS, 539, 940
- DAT, 18, 116, 117, 710, 725, 726, 765, 783, 940
- Data
  - blocks, 35, 51
  - buffer, 89
  - compression, 1, 12, 13, 24, 26, 596, 792, 819, 820
  - compression, 1, 12, 13, 24, 26, 596
  - computer-type, 13, 145, 545, 547, 548, 559, 560, 725
  - decryption, 172
  - encryption, 170–172, 180–182, 187, 190
  - frame, 196, 461, 501, 522–525, 531, 535, 536, 538, 541, 554, 576
  - integrity, 531
  - link layer, 195, 428, 461–463, 466, 469, 498, 500, 501, 553, 565, 571, 572, 581, 582
  - link, 195, 792, 793, 795
  - rate, 511, 540
  - register, 836–838
  - type conversion, 378
  - type, 295, 322, 324, 326, 327, 331, 378, 449
- Datastream-oriented, 213
- Date, 238, 241, 246, 247, 267, 268, 376, 920
- Date and time, 824
- dB, 97, 111, 746, 747–749, 757
- DB25S, 611
- DB9S, 611
- DC, 53, 58, 68, 75, 113, 503, 504, 507, 515, 563–565, 752, 759, 760, 761, 764–766, 782, 811
- DC balancing, 563, 564
- DCD, 809, 814, 817, 940
- DCE, 581, 582, 611, 612, 618, 619, 792, 814, 817, 940
- DCT, 22, 32, 51–53, 55, 60, 63–65, 69, 84, 87–89, 92, 125, 132, 133, 940
  - baseline, 63–65
  - conversion, 69, 84, 85, 88
  - reverse, 60
- DD, 275, 705, 711, 782, 814, 860, 863, 940
- Debugger, 294
- DEC, 439, 458, 497, 743
- Decimal, 276, 277, 296, 307, 308, 309, 779–782
- Decrement, 300, 381, 382
- Decryption, 172
- Del, 830
- Delta modulation, 22, 98
- Demand Priority Access Method, 516
- Demand-paged, 395, 396
- Demodulator, 594
- Deprecation, 328, 329
- DES, 171, 172, 180–184, 186–191, 792
  - mangler function, 181, 185, 188, 189
  - S-box, 185–187
  - Triple DES, 187
- Destination address, 260, 429, 467, 468, 472, 497, 498, 499, 502, 522, 523, 525, 536, 580, 842, 940
- Detection of movement, 83
- Device driver, 389, 390, 392–395, 398, 400, 404–406, 413
- DHCP, 485, 491–495, 795, 797
- DIB, 205, 401, 488
- Dice, 138, 139
- Dictionary, 26–28, 42, 48
- Dielectric, 748, 754, 771
- Differential encoding, 22, 98
- Differential PCM, 22, 100
- Digital, 1, 7, 9, 78, 80, 96, 110–112, 115, 117, 121, 126, 135, 174, 547, 554, 579, 605, 716, 719, 725, 726, 767, 796, 836
  - audio, 110, 111, 116–118, 121, 122, 723
  - audio tape, 116, 765
  - code, 1, 10, 22, 96, 98, 135, 547
  - communications, 1
  - compact cassette, 116, 126, 593
  - modulation, 605
  - radio, 1
- Digital-to-analogue, 9, 756
- Digitization, 7
- Digitized video, 13, 460, 545
- Digitizing TV signals, 78
- DIMM, 708, 855
- Disassembler, 294
- DISC, 567, 568, 576, 577, 940
- Disconnect mode, 567
- Discrete, 10, 22, 51, 52, 89, 125, 132, 133, 190, 940
- Discriminate, 483, 525
- Disk
  - automatic defect reallocation, 728
  - capacity, 711, 721

- duplexing, 419
- mirroring, 419, 420, 437
- predictive failure analysis, 728
- striping with parity, 419
- striping, 419
- transfers, 472, 684
- Displaying images, 278
- Distributed clock, 531
- Divide error, 825
- Division, 6, 80, 107, 108, 133, 136, 137, 149, 152, 153, 298, 300, 381, 382, 547
- DM, 567, 576, 815, 940
- DMA, 403, 672, 677, 678, 680, 698, 701, 702, 705, 706, 710, 716, 836, 847
- DNS, 203, 205, 206, 208, 217, 415, 417, 440, 495, 789, 791, 793, 794, 795, 796, 797, 940
- Do...while(), 308
- DOC, 411
- Dolby, 122, 132, 133, 134
- Domain names, 204, 445, 448
- Domains, 133, 407, 444
- DOS, 6, 29, 216, 389, 390, 393, 405, 409, 410, 424, 425, 436, 461, 495, 656, 824, 825
- services, 825
- based, 390, 425, 436, 461, 634, 844, 845
- DoubleSpace, 29
- DPCM, 22, 100, 101, 940
- DPSK, 606, 940
- DQDB, 550, 940
- DR, 70, 71, 96, 97
- DRAM, 695, 699, 700, 703, 704, 708, 849, 850, 855, 940
- Drive specifications, 712
- DriveSpace, 29, 30
- DSR, 612, 616, 618, 646, 648, 649, 809, 814, 818, 820
- DSS, 174, 190
- DTE, 581, 582, 602, 611, 612, 617–619, 817, 818, 941
- DTR, 601, 612, 616, 618, 647, 648, 809, 814, 817, 941
- D-type, 510, 595, 611, 653, 814
- Dynamic range, 96, 104, 110, 111, 119, 125
- Dynamic range, 96, 97
- EaStMAN, 218, 542–544, 559
- EBCDIC, 296
- EBU, 113, 114, 124
- ECC, 699, 700, 720, 855
- Echo cancellation, 606, 607
- ECP, 653
- Edinburgh, 3, 204, 216, 218, 542, 559
- Edit, 295, 325
- EDO, 703
- EECE.NAPIER.AC.UK, 204, 206, 215, 217, 218, 240, 375, 377, 415
- EEPROM, 941
- EGA, 834
- EGA/VGA, 834
- EIA, 463, 464, 608, 610, 748, 941
- EIAJ, 111, 113, 115
- E-IDE, 714, 715, 717, 718, 720, 721
- Einstein, 3
- EISA, 393, 676, 682–685, 692, 713, 855, 941
- Electronic mail, 170, 215, 229–243, 250, 252–254, 256, 258, 266, 287, 288, 347–350, 407, 441, 461, 590, 788, 792
- address, 254
- API, 231
- clients, 231
- mail client, 230
- messaging protocol, 231
- overview, 231
- overview, 231
- post office, 230, 231
- shared-file approach, 230
- Electronic token, 471, 522
- EM optoelectronics spectrum, 750
- EMF, 401
- Emphasis flag, 115
- Encrypted tunnels, 259
- Encryption, 1, 170–172, 174–176, 178, 180–183, 187–191, 211, 259, 266, 461
- keys, 191
- program, 176
- tunnels, 259
- End delimiter, 523, 524, 535, 566
- End of interrupt, 635, 637, 639, 661, 662, 665, 831, 832
- End terminator, 43
- End-to-end flow control, 558
- Energy envelope, 127
- England, 2, 4
- English language, 13
- Enhanced parallel port, 653, 666, 670, 671, 674, 675, 710
- byte mode, 666, 669, 670, 674
- nibble mode, 666–668
- protocol, 670
- registers, 670
- ENQ, 779, 941
- Entropy coding, 16
- Entropy-encoding, 51
- EOT, 941
- EPP, 653

- EPROM, 941
- Error
  - coding, 135, 147, 154
  - combinations, 140
  - control, 110, 198, 501, 551, 584, 613
  - detection, 117, 128, 144, 145, 147, 149, 160, 198, 428, 461, 463, 498, 499, 524, 565, 568, 571, 576, 580, 607, 613, 699, 723, 854–855
  - probability of a single error, 140
  - probability of error, 139
  - probability of no errors, 139, 140
  - probability, 138, 139, 475
  - types, 145
- Esc, 640
- Escape sequence, 297
- ESDI, 399, 713, 714
- ETB, 779, 941
- Ether, 207, 208, 463, 497, 503, 504
- Ethernet, 4, 81, 92, 151, 195, 196, 199, 208, 220, 231, 274, 275, 282, 289, 290, 413, 419, 425, 428, 434, 454, 463, 465, 467, 469, 472, 493, 497–524, 530, 537, 545, 549, 579, 683, 684, 691, 692, 698, 725, 762, 788, 789, 792, 806
- 100BASE-4T, 514, 515, 763, 764
- 100BASE-FX, 513
- 100BASE-T, 513, 514, 518, 520
- 100BASE-TX, 509, 513, 514, 517, 518
- 100Mbps, 513, 691
- 100VG-AnyLAN, 491, 509, 513, 514, 516, 517, 518, 520, 521, 762
- 10BASE2, 467, 509, 511, 521
- 10BASE5, 509, 511
- 10BASE-FL, 509, 510
- 10BASE-T connections, 514
- 10BASE-T, 467, 509–511, 513, 514, 518, 520
- AUI, 502
- cheapernet, 509, 510
- distance between transceivers, 509
- DSAP, 499, 500
- fast, 509, 513, 514, 516, 518, 521
- II, 425, 434, 499
- implementation, 507
- limitations, 508
- LLC protocol, 500
- MDI, 502
- migration to Fast Ethernet, 518
- PLS, 502, 503
- repeater lengths, 508
- security, 512
- segment lengths, 508
- SNAP, 425, 499, 500
- SSAP, 499, 500
- thick-wire, 509–511
- thinnet, 509, 511
- thin-wire, 509–511
- transceiver, 503, 507
- types, 509
- ETSI, 546, 550
- ETX, 779, 941
- European, 82, 102, 113, 247, 562
- Event handling, 332, 338
- Event-driven, 328, 642
- Events, 82, 138, 152, 236, 328, 329, 331–334, 336, 338–340, 344, 348, 376, 400, 483, 485, 642, 647, 648, 738, 873
- Exabyte, 726
- Exceptions, 400
- Executable, 286, 291
- Explorer, 251, 292, 376, 468
- Exponent, 301, 380, 383
- Exponential, 129, 323
- Extended capability port, 401, 653, 671–675, 710, 792
  - channel address, 671–673
  - forward data, 673
  - mode signals, 672
  - protocol, 672
  - register settings, 675
  - registers, 674
- Extended parallel port
  - mode signals, 671
  - register definitions, 671
- Faller and Gallager, 25
- Faraday, 2, 3
- Fast packet, 549
- FAT, 397, 409, 410, 412
- Fault tolerance, 419, 436, 522, 534, 536, 537
- FAX, 347, 349–354, 596, 672, 819, 941
  - transmission, 607
- FC, 535, 536, 782, 941
- FCC Advisory Committee, 122
- FCS, 499, 501, 502, 523, 524, 568, 573, 576, 581, 941
- FDDI, 151, 195, 231, 274, 282, 289, 290, 419, 464, 467, 491, 518, 520, 530–542, 545, 549, 559, 692, 761, 790, 793, 941
  - applications, 537
  - attachments, 539
  - backbone network, 538
  - DAC, 539
  - DAS, 539
  - frame format, 535, 536
  - layers, 531

- management, 531
- media, 538
- network, 151, 532, 534, 537–539, 542, 692
- restricted token, 531
- SAC, 539
- SAS, 539, 944
- specification, 540, 541
- timed token, 531
- token format, 535
- token-passing, 530, 537
- unrestricted token, 531
- FDM, 6, 941
- FDX, 941
- FF, 245, 276, 507, 525, 535
- FGK algorithm, 25, 26
- Fiber optic, 6, 282, 464, 465, 471, 472, 510, 513, 527, 554, 750, 756, 757
  - Absorption losses, 755
  - chromatic distortion, 755
  - link, 756
  - material scattering, 755
  - modal dispersion, 755
  - radiation losses, 755
  - single-mode, 538
  - typical characteristics, 756
- FIFO, 507, 674, 710
- File
  - attributes, 411
  - dialog, 361
  - servers, 412, 436
- File systems, 45, 230, 253, 389–390, 397, 398, 407–412, 430, 435, 440, 441, 445, 446, 723, 724, 804
  - FAT, 397, 409, 410, 412
  - HPFS, 410–412
  - NTFS, 410, 412, 421
- File Transfer Protocol, 214, 215, 254, 376, 441
- FINGER, 214
- Firewall, 256–259, 261, 263, 264
- Fixed disks, 711
- Flicker, 80
- Floating-point, 296, 301, 379, 383, 451, 457, 840, 846, 852, 853, 840, 846, 852, 853
- Floppy disks, 711
- Flow, 2, 4, 212, 213, 249, 396, 400, 428, 460, 501, 550, 553, 554, 558, 560, 567, 569, 571, 572, 574–582, 584, 587–591, 616, 817, 818, 823, 853, 862
- Flow control, 428, 501, 550, 553, 554, 558, 567, 569, 571, 574–576, 580–582, 584, 587, 588, 591, 616, 817, 818, 823, 924
- Flying head, 712
- FM, 1, 6, 77, 80, 121, 132, 941
- FM radio, 1
- Font
  - italic, 269, 270, 862, 863
  - Symbol, 533
- For(), 307, 308, 327, 387
- Form feed, 245, 380
- Forms, 281, 286
- Forward Adaptive Bit Allocation, 129
- Forward error correction, 155
- Fourier transform, 22, 126
- Fragment, 196
- Fragmentation, 213, 550, 924
- Frame, 16, 63, 64, 73, 77, 80, 82–95, 108, 109, 114, 125–128, 133, 196, 249, 250, 366, 414, 425, 426, 428, 434, 437, 461, 467, 498–505, 516, 520, 522–525, 531, 535–538, 541, 554, 557, 562–578, 583, 588, 596, 612, 613, 762, 880, 886, 941
- Frame check sequence, 428, 499, 523, 524, 536, 565, 568, 573, 576, 941
- Frame format, 114, 535, 563, 595, 612
- Frame reject, 567
- Framing bits, 463, 500, 562
- France, 2, 4, 204
- Franklin, 2
- Frequency domain, 84, 88, 131, 133
- FRMR, 567, 575, 576, 941
- FSK, 605, 606, 941
- FSR, 152
- FTP, 195, 211–216, 243, 247, 252–254, 263, 321, 376, 406, 415, 421, 441, 788, 789, 792, 797, 941
- Full-duplex, 572, 605
- Full-scale voltage, 96
- Functions, 824, 832
  - G.711, 103, 108
  - G.722, 107, 108
  - G.728, 107, 108
- GaAs, 752
- Gateway, 195–200, 208, 209, 217, 218, 232, 238, 257–259, 263, 264, 266–268, 286, 415, 418, 424, 427, 430, 440, 441, 444, 459, 465, 469, 478, 479, 486, 553, 559, 788–791, 832
- Gauss, 2, 3
- Germany, 2, 3, 124
- GFI, 584, 941
- GIF, 9, 28, 30, 32–44, 48, 51, 243, 248, 277, 278, 594
  - application authentication code, 44
  - application extension, 35, 44
  - comment extension, 35, 44

- data sub-blocks, 35
- disposal method, 43
- file signature, 35
- global color table, 35–41
- graphic control extension, 35, 43
- header, 35
- image conversion, 83
- image description, 35
- image descriptor, 40
- local color table, 35, 41, 42
- logical screen descriptor, 35, 36
- logical screen height, 36
- logical screen width, 36
- plain text extension, 35, 44
- signature, 35
- table based image data, 35, 42
- trailer, 35
- transparent color flag, 43
- transparent color index, 43
- version 87a, 34, 35
- version 89a, 33–36, 38, 40
- GMT, 246, 247
- Gopher, 215, 247, 252, 253, 255, 263
- Government, 170, 171, 796
- Graphical Device Interface, 399, 401
- Graphical User Interface, 256, 454, 683, 845
- Graphics, 12, 27, 32, 33, 34, 45, 49, 63, 277, 278, 292, 295, 321, 325–331, 334–338, 342–355, 361, 363, 390, 401, 406, 454, 456, 458, 594, 608, 633, 672, 683–685, 688, 709, 794, 834, 836, 869–871, 876, 878
- Graphics-based, 281
- Gray scale, 32, 51, 62, 608
- Greek, 2, 170, 781, 792
- Green Book, 724
- Ground, 1, 611, 612, 620, 653, 657, 809, 810
- Group address, 525
- Group III, 608
- Group IV, 608
- Groupware, 256
- GUI, 256, 683, 873, 941
- GZ, 248
- Hackers, 408
- HAL, 404
- Half-duplex, 514, 572
- HALT, 533
- Hamming code, 155, 156, 158, 160, 161
- Hamming distance, 137, 138, 147, 759
- Handshaking, 518, 594, 615–620, 624, 646, 647, 653, 654, 667–670, 680, 681, 688, 690, 698, 733, 808, 817
- Hard disk/CD-ROM interfaces, 713
- Hard-disk, 7, 717
- Hardware, 6, 292, 296, 824, 827, 834
- Hardware handshaking, 615, 618
- Hayes, 597
- HD, 711, 941
- HDLC, 463, 464, 571–574, 576, 579, 581, 582, 584, 941
- HDTV, 68, 81, 82, 91
- HDX, 941
- Header files, 293
- Helix, 725
- Henry, 2
- Heriot-Watt University, 542, 559
- Hertz, 2, 3, 4
- Hexadecimal, 37, 38, 245, 276, 277, 298, 379, 499, 779, 844
- HFS, 723, 804
- Hi-fi, 8, 10, 111, 116, 124, 128, 546, 548, 722, 725
- High-frequency, 55, 58, 68, 84, 88, 89, 122, 749
- High-level language, 376
- Hiss, 8, 122
- Hitchhikers Guide to the Galaxy, 45
- Holland, 4
- Horizontal lines, 280
- Hosts, 196, 198, 200, 201, 203, 204, 209, 211–213, 216, 217, 233, 234, 443, 445, 447, 448, 458, 486, 491, 800
- Hot fixing, 412
- HP, 439, 710
- HPFS, 410, 411, 412
- HP-UX, 439
- HTML, 242, 248, 250, 253–255, 264, 265, 269–292, 324, 325, 327, 330, 331, 335, 337, 376–378, 388, 792, 795, 857, 860, 861, 941
  - anchors, 281
  - background images, 277
  - colors, 276
  - definition lists, 275, 863
  - displaying images, 278
  - horizontal lines, 280
  - input types, 288
  - links, 270, 271
  - lists, 272
  - menus, 290
  - ordered lists, 272
  - tables, 47, 49, 64, 65, 281, 282
  - unordered lists, 272, 274
- HTML script, 269–272, 274, 275, 277, 278, 280–290, 324, 325, 327, 330, 331, 335, 337, 376

- HTTP, 215, 253–255, 256, 262, 263, 264, 265, 266, 271, 277, 293, 321, 359, 366, 368, 792, 794, 795, 796, 797  
entity header fields, 268  
full requests/responses, 264  
general header fields, 267  
message, 264  
request example, 265
- Hub, 260, 470, 485, 510–514, 516, 518–520, 521, 522, 695, 699, 707, 748, 762
- Hues, 69
- Huffman, 17, 24–26, 28–32, 42, 45, 47, 51, 55, 58, 60, 63–65, 68, 84, 87, 89, 92, 608  
coding example, 25  
coding, 24–26, 28, 29, 31, 47, 51, 58, 63, 84, 87, 89, 608  
modified, 608
- Human eye, 13, 52, 68, 83, 84, 88, 122, 750, 751
- Human hearing, 104
- HyperTerminal, 628–630
- Hypertext, 215, 248, 252, 254, 255, 262, 792, 794
- I/O  
digital, 836  
inputting a byte, 836  
isolated, 832, 834  
memory mapped, 832, 833  
outputting a word, 838  
read, 677, 689, 690, 740, 840  
write, 670, 677, 689, 690, 740, 840
- IA5, 779
- IAP, 250
- IBM, 5, 182, 412, 415, 439, 463, 472, 522, 527, 528, 571, 653, 678, 682, 839
- IBM 8228, 527
- IBM AT, 839
- IBM PC, 5, 839
- IBM type A connectors, 527
- ICMP, 198, 199, 212, 215–218, 440, 444, 494, 788
- ICP, 250
- IDE, 398, 680, 692, 695, 698, 701, 705, 706, 710, 713–716, 718, 720, 721, 727, 729, 836
- IDE/ISA interface, 706
- IDEA, 171, 172, 180, 181, 187–189, 191
- IDI, 593, 942
- IDLE, 533, 796
- IDP, 444, 592, 942
- IEC, 51, 113, 115, 116, 748
- IEC-958, 113, 115, 116
- IEEE, 204, 323, 451, 463, 464, 468, 497–502, 513, 514, 516, 522, 523, 524, 535, 537, 550, 653, 666, 789, 790, 793, 795, 797, 809, 810, 811, 942, 1284, 653, 666, 809–811, 802.12, 514, 516, 793, 802.2, 463, 464, 498–500, 802.3, 463, 464, 497–502, 513, 514, 516, 795, 797  
802.3 frame format, 498, 499  
802.3u, 513, 514  
802.4, 464, 790  
802.5, 463, 464, 498, 516, 522–524, 537, 790  
802.6, 535, 550  
standards, 498
- IETF, 262, 793, 797, 861
- If(), 304–306, 385, 386
- ILD, 752, 753, 756
- Image conversion, 83
- Image scanner, 608
- Images, 1, 2, 13, 16, 21, 22, 24, 29, 30, 32, 33, 35, 45, 48, 49, 51, 68, 82, 83, 86, 90, 92, 229, 230, 240, 248, 256, 262, 277–279, 455, 594, 608, 672, 876, 877, 879
- Impedance, 769
- Import statements, 320
- IMR, 636–638, 661–665, 831
- Increment, 11, 96, 269, 300, 381, 382
- Indexing, 324
- Inetd, 206, 448, 449
- Infra-red, 752, 755
- Initialization and exit methods, 328, 329
- Initialization, 312, 328–330, 484, 490, 526, 587, 640, 665, 692, 854
- Injection Laser Diode, 752
- Input/output, 293, 824, 832
- Input/output supervisor, 398
- Installable File System, 397
- Integer division, 298, 381
- Integrity check, 171
- Intel, 5, 6, 45, 46, 204, 390, 497, 676, 685, 695, 708–710, 808, 839, 841, 845, 851–853, 839, 841, 845, 851–853
- Intensity, 21, 52, 127, 722, 752, 753, 868
- Interactive video, 548
- Interconnected networks, 170, 196, 438, 461–463, 468, 552
- Interconnection length, 466, 530
- Interface, 292, 329, 832, 836
- Interfacing, 832  
isolated, 833  
memory, 832
- Interlaced, 32–34, 41, 48, 73, 77, 84, 90, 121

- Interlaced images, 48, 90  
 Interleaving, 117  
 internet, 195–202, 444  
 Internet, 1, 34, 68, 181, 182, 195, 197, 198, 202–206, 208, 211, 215, 217, 221, 222, 229, 230, 232, 233, 240, 247, 249–263, 291–293, 324, 359, 360, 376, 406, 412–415, 424, 427, 440, 441, 443, 445, 448, 463, 469, 474, 476, 477, 481, 492, 495, 594, 630, 788–797, 918  
 access provider, 250  
 addresses, 199, 201  
 connectivity provider, 250  
 datagram, 197, 198  
 email address, 232, 254  
 example domain addresses, 204  
 example, 200  
 Explorer, 251, 292, 324, 376  
 naming structure, 204  
 presence provider, 250  
 primary domain names, 204  
 protocol, 195, 440, 463, 788, 789, 791, 793  
 resources, 251  
 inter-networking, 195  
 Internetworking connections, 466  
 Interrupt, 637, 638, 662, 678, 680, 695, 698, 707, 710, 824, 825, 827, 828, 831, 832, 834, 843, 847  
 control port, 250, 456, 637, 796  
 controller, 707, 710  
 edge-triggered, 831  
 handling, 825, 827, 828  
 hardware, 637, 824  
 mask register, 636–638, 661–665, 831, 832  
 processor, 825  
 request, 678, 680, 825–827, 830, 840, 847, 849  
 service routine, 824, 827  
 software, 824, 826  
 vectors, 824, 827, 834  
 Interval timer, 706  
 Intranets, 247, 256, 259  
 IP, 1, 195–217, 220, 222, 230, 233, 247, 256, 257, 259, 260, 275, 292, 359, 360, 362, 375–377, 390, 412–421, 424, 425, 427, 428, 430, 434–445, 463, 468–470, 475, 483, 486, 491–500, 552, 553, 559, 632, 788–797, 842, 942  
 address, 195–212, 216, 217, 220, 222, 257, 259, 260, 275, 292, 360, 362, 375–377, 415, 417, 418, 440, 443, 445, 475, 483, 491–496, 552, 553  
 address format, 200  
 addressing, 195, 201  
 class A, 200, 201, 203, 555, 793  
 class B, 200, 201, 203, 555  
 class C, 200, 201, 203, 310–314, 555  
 data frames, 196  
 header, 198, 199, 211–213  
 protocol, 196–199, 216, 217, 230, 233, 415, 421, 440, 500, 553  
 time-to-live, 198, 217, 218, 428, 496  
 Ver4, 211, 212  
 Ver6, 211, 212, 791, 792, 793  
 IPX, 393, 406, 412–414, 419, 424–431, 434–437, 468–470, 499  
 IPX.COM, 406  
 IPXODI, 406, 432, 433  
 packet format, 428  
 Ireland, 4  
 IRQ, 390, 616, 642, 659, 661, 680, 685, 694, 697, 698, 727, 825, 827–831  
 IRQ0, 693, 699, 706, 707, 825, 827, 828, 830, 831  
 IRQ1, 399, 697, 699, 708, 825, 827, 828, 830, 831  
 IRQ10, 691, 697, 828  
 IRQ11, 697, 828  
 IRQ12, 697, 698, 699, 707, 708, 828  
 IRQ13, 680, 699, 707, 828  
 IRQ14, 680, 697–699, 707, 714, 715, 719, 828  
 IRQ15, 680, 693, 697–699, 707, 828, 831  
 IRQ2, 699, 707, 825, 828, 830, 831  
 IRQ3, 697, 701, 707, 825, 827, 828, 830, 831  
 IRQ4, 636, 637, 638, 697, 701, 702, 707, 825, 827, 828, 830, 831  
 IRQ5, 697, 701, 825, 827–830  
 IRQ6, 697, 701, 702, 825, 828, 830, 831  
 IRQ7, 661–665, 697, 701, 702, 825, 827–831  
 IRQ8, 828, 831  
 IRQ9, 686, 697, 828, 830  
 ISA, 393, 670, 676–685, 692, 695, 697–706, 710, 713, 714, 855, 942  
 ISDN, 103, 128, 249, 250, 460, 463, 464, 547–549, 560–570, 594, 608, 795, 939, 942  
 basic rate, 560–563  
 B-channels, 561, 562, 565, 569  
 B-ISDN, 561  
 broadband, 561  
 call clearing, 570  
 call establish, 570, 571  
 channels, 561

- data link layer, 565
- D-channel contention, 568
- D-channels, 561, 563, 564, 565, 566, 568, 569, 579
- dial-up, 250
- frame format, 563, 564
- H0, 561, 816
- H11, 561, 562
- H12, 561, 562
- information messages, 570
- network layer, 561, 568
- network messages, 570
- physical layer, 561, 562
- supervisory frame, 566
- system connections, 563, 625
- TEI, 565, 566, 568, 569
- ISO, 51, 195, 397, 460, 462–464, 498, 542, 563, 572, 593, 722–724, 748, 794, 795, 942
- ISO 9660, 397, 723, 724
- ISO/IEC, 51, 748
- ISO-IP, 195
- Isolated I/O, 832, 834
- Isolator, 504
- ISR, 637–639, 664, 665, 824, 826–828
- Italic, 269, 270
- ITU, 78, 108, 463, 596, 606, 608, 942
- ITU-R, 78
- Jamming signal, 497
- JANET, 218, 559, 942
- Japan, 103, 108, 113, 116, 725
- Java, 248, 250, 283, 285, 292–381, 388, 645, 650, 864–921
  - applet, 292–294, 320, 324–327, 330, 331, 333–355, 361–363, 366, 367, 369, 373, 376, 621
  - compiler, 292–295, 321, 326, 328, 376
  - interpreter, 293–295, 357, 374
  - java.exe, 294, 295, 357
  - javac.exe, 294, 295, 325
  - script, 248, 250
- Java/JavaScript, 292
- JavaScript, 248, 250, 376, 377–381, 385, 388, 389, 393
- JDK, 293, 294, 320, 325
- JFIF, 33, 51, 62, 63, 65, 66, 243
- JFIF header information, 63, 65
- JISC, 542, 942
- Jittery, 111
- Joliet, 723
- JPEG, 9, 32, 33, 34, 51, 52, 55, 58, 60, 62–68, 83, 84, 88, 89, 125, 243, 248, 277, 278, 594, 942
  - data, 52
  - decoding, 60
  - file format, 62
  - final compression, 58
  - header, 52
  - hierarchical mode, 68
  - lossless mode, 69
  - modes, 68
  - normalization matrix, 56, 61
  - progressive modes, 68
- JPG, 9, 32, 33, 34, 63–67, 248, 594
- K/n rate code, 143
- Key methods, 338
- Keyboard, 328, 331, 336–338, 704, 825, 828, 830, 835
  - data ready, 830
  - events, 328, 338
  - input, 336
- KeyPress, 337, 338
- Keywords, 304, 386
- LAN, 1, 282, 406, 412, 414, 424, 428, 438, 459, 463–465, 468–470, 491, 497, 498, 509, 518, 520, 528, 549, 571, 684, 746, 748, 942
- LAPD, 565, 569, 579, 942
- Laser, 117, 710, 721–723, 753, 756
- Last packet indicator, 524
- LCN, 584, 592, 942
- LCR, 620, 623–628, 637
- LD-CELP, 108
- Least significant, 45, 46, 113, 149, 169, 179, 308, 309, 449, 450, 614, 638, 659
- LED, 660, 752, 753, 755, 756, 942
- Lee De Forest, 4
- Lempel-Ziv, 24, 26, 28–31, 607
- Lempel-Ziv-Welsh, 26, 30
- Lens, 721
- LF, 235–237
- LFE, 124, 128
- LGN, 584, 592, 942
- Library, 220, 255, 319, 322, 327, 404, 456
- Light, 1, 3, 276, 464, 473, 531, 532, 538, 548, 721, 722, 746, 750–756, 771
- Light emitting diode, 752
- Line break, 269, 270, 285, 530, 610, 863
- Linear quantization, 102, 104
- Link Access Procedure, 565, 582
- Link Support Layer, 406
- Links, 270, 271
- Lip synching, 85
- List box, 359
- Listeners, 332, 333, 338, 339
  - action, 344

- item, 348
- Lists, 63, 272
- LLC, 498–502, 535, 537, 571, 579, 942
- Local bus, 684
- Logarithmically, 103
- Logical block address, 718, 720, 736
- Logical link control, 464, 498, 524, 579, 942
- Logical operator, 300, 301, 383
- Logical unit number, 736
- Login, 195, 214, 215, 253–255, 258, 409, 413, 426, 432, 433, 435, 444
- Loops, 307, 387
- Losses in fiber optic cables, 755
- Lossless compression, 13, 16
- Lossy compression, 13, 32, 51
- Lotus Notes, 256
- Low frequency effects, 124, 128
- Low signal levels, 102
- Lowercase, 905, 913
- LPT1, 611, 656, 657, 662, 664, 702, 828, 830, 836
- LPT2, 656, 828, 830, 836
- LRC, 154, 155, 161
- LRC/VRC, 154
- LSR, 620–628, 635, 637, 640
- Luminance, 52, 68, 69, 72, 73, 75, 78, 79, 80, 83, 84, 86, 88, 89
- Lynx, 205, 206, 251, 256, 439
- LZ coding, 26, 29
- LZ-77, 26
- LZ-78, 26
- LZH, 29, 33
- LZS, 29, 792
- LZW, 26, 28, 30, 32–35, 42, 43, 45, 48, 51, 57
- Mac, 256, 293
- MAC, 80, 81, 82, 195, 196, 199, 206, 207, 208, 220, 260, 261, 274, 406, 428, 431, 433, 467, 494, 497, 498, 499, 500, 501, 502, 504, 522, 523, 525, 531, 532, 535, 537, 541, 542, 550, 692, 942
- address, 195, 196, 199, 206–208, 220, 260, 261, 274, 431, 467, 494, 499, 500
- layer, 196, 406, 433, 498, 501, 502, 504, 522, 525, 532, 541, 550
- protocol, 525, 537
- Machine code, 832
- Machine-specific code, 376
- Macroblocks, 89
- Magnetic
  - disk, 710
  - fields, 3, 8, 710
  - tape, 710, 725
- Magneto-optical, 724
- Magneto-optical (MO) disks, 724
- Mail fragments, 244
- Mailroom, 229
- MAN, 542, 550, 942
- Manchester
  - coding, 503, 505–507, 524
  - decoder, 504, 505
- MANs, 459
- MAP, 232, 804
- Marconi, 2, 3, 4
- M-ary, 135, 605, 606
  - ASK, 606
  - FSK, 606
  - modulation, 605
  - PSK, 606
- Math co-processor, 828
- Matrix representation, 156
- MAU, 503, 526–528, 942
- Maxwell, 2, 3, 4
- MDCT, 125, 132, 133
- Mean, 759
- Media access control, 497, 498, 502, 523, 942
- Media Interface Connector, 540
- Megablocks, 86
- Memory, 38, 80, 87, 88, 116, 133, 264, 277, 296, 326, 329, 389, 390, 393, 395–400, 403–405, 420, 437, 439, 458, 507, 520–522, 525, 615, 620, 621, 654, 657, 670, 676–710, 713, 720, 722, 734, 738–745, 824, 832–836, 839–856, 912, 940, 944
- addressing, 843, 844, 849
- map, 832–836
- mapped I/O, 833
- models, 390
- paging, 395, 396
- segmentation, 843
- segmented, 397, 843, 844
- Menu, 290, 351
  - bar, 357
  - multiple, 355
  - pop-up, 351, 353, 354, 867, 883
- Metafile, 12, 401, 940
- Method overloading, 314
- Methods, 24, 323, 326, 864–924
- Metropolitan Area Network, 542, 550
- MHS, 231, 232
- MIC, 540, 942
- Micro-ops, 853
- Microprocessor, 5, 683, 833, 839
- Microprocessor, 684
- Microsoft, 5, 6, 29, 218, 232, 251, 324, 357, 366, 367, 376, 389, 391, 404, 406, 414,

- 415, 435, 454, 491, 495, 599, 625, 641, 671, 720, 795, 824, 828, 834, 837, 838, 845  
LAN, 414  
Windows, 6, 357, 389, 406, 435, 454, 495, 599, 824, 828, 834, 845  
Migration, 82, 438, 514, 549  
Military, 171, 174, 204, 757  
MIME, 181, 215, 230, 232, 240–245, 253, 264, 265, 267, 268, 291, 359, 790, 791, 793–797  
base64, 245  
boundary name, 242  
content-description, 240  
content-id, 240  
content-transfer-encoding, 240, 244, 245  
content-type, 240, 242, 244, 268, 795, 797  
encoded, 242, 243, 264  
example, 242, 243, 244  
version, 240, 264  
Miniport, 398  
MIPs, 852  
MIT, 161, 174, 454, 458  
Mixer, 116  
MLID, 433, 434  
MMX, 853  
MNP level 5, 606, 607  
Modems, 28, 413, 560, 561, 579, 594–598, 601, 602, 605–607, 614, 684, 739  
asynchronous, 595  
AT commands, 597, 606  
AT prefix, 597  
ATDT, 597–600  
ATH, 597–600  
auto-answer, 594, 599, 601, 820  
auto-dial, 594  
commands, 597, 599, 600  
connection, 195, 250, 603, 604, 619, 624, 817  
dialing, 600, 601  
dial-up, 250  
example return codes, 598, 819  
indicators, 601  
profile viewing, 601  
registers, 599, 820  
setups, 599  
S-registers, 599, 820  
standards, 596  
test modes, 602  
typical, 606  
V.22bis, 596, 606, 607, 816  
V.32, 596, 606, 607, 816, 819  
V.32bis, 596, 606, 607, 816, 819  
V.42bis, 28, 596, 606, 607, 819  
Modified discrete cosine transform, 125, 132, 133  
Modulator, 98, 99, 594  
Modulo-2, 135, 136, 137, 143, 147–149, 152, 153  
arithmetic, 135, 149  
Modulus, 298, 299, 300, 309, 381, 382  
Monophonic, 121  
Moray House Institute of Education, 542  
Morse, 3, 17  
Morse code, 17  
MOS, 5, 940, 943  
MOS Technologies, 5  
Motherboard, 684, 685, 695, 698, 707–709, 713, 714, 812, 849  
Motion estimation, 83, 86, 87, 90, 93, 94  
Motion JPEG, 83  
Motion video, 68, 81, 83, 93, 94, 230  
Motorola, 5, 45, 710, 808  
Mouse, 328, 331, 332, 334, 393, 707–709  
events, 328, 331  
function, 707  
selection, 334  
MOV, 248  
MPEG, 81, 83–94, 121–129, 133, 243, 244, 248, 291, 722  
B-frame, 87, 88  
B-frames, 87, 88, 90, 94, 95  
frame sequence, 88  
group of pictures, 92  
high 1440, 91  
high level, 91  
I-frame, 87, 88  
main level, 91, 95  
P-frame, 87, 88  
presentation time stamps, 91  
slices and macroblocks, 85  
MPEG-1, 81, 83–90, 92, 93, 95, 122, 125–128  
audio frame, 128  
MPEG-2, 81, 90–93, 95, 124, 125, 128  
profiles, 90, 91  
system layer, 91  
MPEG-3, 90, 121  
MPEG-4, 90  
MR, 601  
MS Mail, 232  
MS-DOS, 389, 392, 395, 397, 399, 404, 405, 409, 410  
mode support, 395  
Multi-dimensional arrays, 327  
Multimedia, 249, 289, 291, 521

- Multiple file systems, 397
- Multiplexing, 6, 80, 81, 89, 107, 213, 547, 549, 561, 562, 653, 687
- Multiplexing/demultiplexing, 213
- Multiplication, 136, 137, 300, 382
- Multiprocessing, 439
- Multi-station access unit, 503, 526, 527, 528, 942
- Multitasking, 389, 390, 392, 395–397, 401, 402, 406, 439, 853
  - co-operative, 402
  - pre-emptive, 390, 401, 439
- MUSICAM, 121–124
- MUX, 541, 705, 790
- NACK, 145, 571
- NAK, 779, 943
- Napier, 218, 241, 542
  - EECE department, 204, 206, 215, 217, 218, 240, 375, 377, 415, 447, 799
  - electrical department, 208
- NCP packet format, 431
- NCSA, 247, 251
- NDIS, 393, 406, 413, 415, 432
- NDS, 435, 436
  - structure, 436
- NDS structure, 436
- Negative acknowledgment, 145
- NetBEUI, 412–415, 419
- Netscape, 182, 250–252, 292, 324, 376
- Netstat, 217
- NetWare, 393, 406, 412, 413, 422, 424–438, 468–500, 727, 797
  - 4.1, 427, 430, 435, 437, 438, 469
  - architecture, 424, 425
  - bindery services, 426
  - container objects, 435
  - directory services, 427, 435
  - leaf objects, 435
  - loadable modules, 425
  - protocols, 427
  - SAP, 427, 430
  - setup, 432
  - SMP, 438
  - VLM, 436
- Network, 1, 195, 202, 210, 231, 232, 250, 253, 259, 282, 289, 290, 292, 406, 412, 413, 415, 419, 422–424, 430, 434, 435, 439–442, 448, 457, 462, 464, 470, 472, 475, 478, 479, 481, 482, 485, 486, 497, 504, 514, 518, 542, 547, 550, 552, 582, 630, 788–799
  - addresses, 204, 360, 440, 491, 495, 565
  - cable types, 464
- information center, 202, 788
- layer, 197, 213, 406, 419, 424, 427, 461, 462, 466, 468, 469, 479, 500, 561, 569, 582
- loading, 472
- malfunctions, 534
- management, 424, 485, 486, 490, 518, 522, 524
- NETx, 424, 431, 432, 433, 436
- operating systems, 1
  - statistics, 217, 472
  - topologies, 470
  - traffic, 264, 429, 430, 436, 437, 457, 472, 473, 474, 476, 485, 503, 522
  - transport protocol, 231
- New York Times, 4
- Newfoundland, 6
- Newline, 297
- Newton, 3
- NFS, 440–442, 445, 446, 448, 789, 797–799, 802, 804
  - remote file access, 442, 445
  - RPC, 216, 441, 442, 445, 448
  - services protocol stack, 442
  - XDR, 442, 449, 453, 789
- Nfsd, 448, 449, 798
- NIC, 202, 406, 413, 424, 432, 434, 504, 788, 943
- NIS, 441, 442, 444–448, 798, 799, 800, 801, 802
  - commands, 446
  - database components, 445
  - domain, 444–448, 799, 800, 801
  - master server, 444
- NLSP, 438, 469
- Noise, 1, 7, 8, 22, 70, 88, 90, 96–98, 100, 102, 110–113, 122, 123, 126, 129, 145, 146, 181, 504, 605, 757, 787, 812
  - impulse, 146
  - thermal, 146
- Non real-time, 7
- Non-linear compression, 102, 103
- Non-zero, 57, 133, 160, 304, 386
- North America, 103
- NRZ, 112
- NRZI, 532, 533, 759, 760, 943
- NSAP, 592–594, 943
- NSCA, 251, 943
- Nslookup, 206, 217
- NTE, 460, 943
- NTFS, 410, 412, 421
- NTSC, 68, 73, 75, 78, 81, 82, 84, 92, 943
- N-type, 510, 511

- Numbers, 790, 793, 795, 796  
Nyquist, 9, 22, 98, 127, 546  
criterion, 9, 546  
frequency, 22, 98, 127  
Oban, 6  
Object-oriented, 292, 309, 389  
Objects, 340, 376, 389, 426, 427, 790, 793–797  
Octal, 298, 308, 309, 379  
ODI, 406, 424, 425, 432, 434  
OH, 601, 943  
One's complement, 147, 383  
Operating system, 1, 6, 251, 256, 281, 389, 390, 395–397, 399–407, 410, 413, 414, 424, 425, 438–442, 454, 468, 469, 491, 492, 499, 633, 683, 724, 824, 844, 852, 853  
Operators, 6, 298–304, 380–386  
Optical disk, 710, 721, 724  
Optical fiber, 542, 610, 754, 755  
Optical storage, 721  
Optoelectronics, 750  
OR, 335, 659  
Orange Book, 723, 724  
Ordered, 35, 45, 88, 92, 93, 272, 273, 862  
Organization, 172, 195, 202, 204, 229, 249, 250, 256, 275, 396, 408, 412, 435, 439, 465, 477, 498, 500  
OR-tied, 730, 731  
OS/2, 409, 410, 412, 414, 424, 727  
OSF/1, 439  
OSI, 195–197, 213, 231, 274, 413, 432, 433, 442, 454, 460–463, 468, 469, 498, 501, 532, 553, 555, 562, 571, 573, 581, 582, 790, 943  
model, 195, 196, 213, 413, 432, 433, 442, 454, 460–462, 498, 501, 532, 553, 555, 581, 582  
OSI model, 195, 196, 213, 274, 413, 432, 433, 442, 460, 461, 462, 498, 501, 532, 553, 555  
OUI, 536, 943  
PA, 535, 873, 890, 943  
Packed bit field, 36, 37, 41  
Packet, 81, 114, 145, 197, 199, 212, 215–218, 220, 257, 260, 359, 406, 414, 424, 427–431, 437, 438, 440, 444, 461, 463, 467, 468, 475–479, 494, 496, 524, 540, 541, 542, 549, 550, 553, 571, 579, 581–588, 591–593, 724, 924, 943  
filters, 257, 259  
Packet Internet Gopher, 215  
Packet-switched, 541, 542, 580, 581, 586, 587, 591, 592  
Paint() object, 327  
PAL, 68–93, 943  
Palette, 32, 35, 38, 49, 51  
Paragraph, 269, 270, 862, 863  
Parallel, 110, 111, 172, 173, 242, 243, 405, 438, 518, 610, 611, 633, 653, 654, 656, 657, 659, 661–667, 672, 679, 695, 699, 701, 710, 774–776, 810, 812, 827, 830, 836, 852, 853, 943  
Parallel port, 653, 656, 657, 659, 827, 830, 836  
Parameter list, 324  
Parameter passing, 292  
Parameters, 281, 284, 324, 326, 337  
Parentheses, 384, 493, 737  
Parity, 144, 147, 420, 613, 648, 649, 700, 728  
even, 114, 147, 154, 159, 613, 626, 627, 644, 692, 843, 846  
odd, 147, 154, 182, 612, 613  
Pascal, 281, 292, 309, 327, 621, 622, 657, 836, 837, 838  
Passive, 442, 527  
Password, 253, 254, 259, 289, 408, 409, 435, 443, 447, 476, 477, 791, 792  
Patent laws, 172  
PATH, 293, 483, 806  
PC, 5, 6, 29, 50, 132, 205, 206, 218, 230, 232, 255, 256, 292, 293, 325, 389, 390, 392, 393, 406, 424, 439, 448, 463, 468, 611, 619–621, 653, 654, 661, 666, 667, 669, 676, 677, 678, 682–685, 694, 704, 707, 710, 713, 714, 744, 808, 812, 824, 827, 830, 831, 834, 835, 839, 840, 844, 845, 849  
bus, 676–678, 685  
connectors, 611  
soundcards, 132  
PCI  
interface, 701  
ISA bridge, 695  
man. ID, 692, 710  
unit ID, 692  
burst mode, 687  
multiplexed mode, 687  
PCI bus cycles  
address phase, 689  
configuration address space, 692  
configuration read access, 689, 691  
configuration write access, 689, 691  
dual addressing, 689, 691  
multiple read, 689, 691  
PCM, 7, 22, 96, 98, 100, 101, 103, 107–119,

- 125, 534, 547, 548, 555, 562, 563, 786, 940, 943
- adaptive delta modulation, 22, 100
- adaptive differential, 101
- delta modulation, 22, 98, 108
- differential, 22, 100
- granular noise, 22, 98, 100
- low-delay CELP, 108
- parameters, 96
- slope overload, 22, 98, 99, 100
- PCM-TDM, 107–109, 547, 548, 562
- PCMCIA, 676, 739–741, 744
  - interface controller, 741
  - registers, 741
  - type II, 739
  - type III, 739
  - type IV, 739
- Pcnfsd, 448, 449, 802
- PCX, 32–34, 49, 50, 51
- PCX coding, 49
- PDN, 580, 943
- Peer-to-peer, 390, 407
- Pentium, 6, 397, 680, 685, 691, 695, 697, 708, 709, 840, 841, 843–845, 847, 849–855,
- Pentium II, 695, 697, 708, 709, 840, 851–854
- Pentium Pro, 851–855
- Permanent connection, 561, 580, 595
- Phase quadrature, 75
- Phase shift keying, 605
- Phases, 26, 607, 731, 733, 734, 854
- Phase-shift keying, 605
- Philips, 115, 126, 723, 724
- Phone, 1, 204, 594, 597
- Photographic, 8
- Photons, 753
- Phototransistors, 753
- PHY, 531, 532, 542, 554, 943
- Physical, 195, 406, 464, 502–504, 531, 534, 542
- Physical layer, 406, 461, 463, 466, 501, 513, 541, 554, 561, 562, 581, 583, 943
- Physical media dependent, 531
- PIC, 637, 695, 698, 828, 830, 831, 853
- PIIX3, 695, 697–708
- PIIX4, 695, 697, 710
- PIN numbers, 170
- Ping, 195, 198, 215, 216, 220, 406
- Pipeline, 699, 852, 853, 855
- PISO, 943
- Pixel, 12, 13, 16, 21, 32, 36, 37, 41, 42, 45, 47, 50–53, 63, 68, 73, 80, 83–89, 93, 279–281, 283, 284, 324, 608, 684
- aspect ratio, 36
- ratio, 37
- PKARC, 29
- PKP, 174
- Plain text, 44, 171, 180–182, 187, 190, 262
- Plastic cable, 756
- Platter, 712
- Playing cards, 327
- PLL, 111, 505, 506
- Plug-and-play, 391, 393, 401, 403, 692, 698, 710
- PMD, 531, 542
- Pointer, 30, 45, 46, 214, 243, 252, 297, 292, 454, 507, 621, 720, 842, 844
- far, 621
- Point-to-point protocol, 197, 393, 524, 630, 631, 791, 792, 793, 794, 795, 796
- Polarization, 249, 721
- Polarized, 721
- Poll/final bit, 567
- Polynomial, 136, 137, 149, 150–153, 568, 576
- Port driver, 398
- Port number, 213, 214, 222, 253, 359, 364, 448, 643, 644, 651
- Portmap, 448, 799
- Ports and sockets, 213
- POSIX, 412
- Postal service, 170, 229
- Postscript, 248
- Power management, 695, 707, 710
- Power supplies, 564
- PPI, 623
- PPSDN, 943
- Preamble, 113, 498, 499, 500, 502, 503, 505, 535
- Precedence, 384, 385
- Pre-emptive, 389, 390, 395, 396, 439
- Preemptive multitasking, 390, 439
- Presentation, 91, 213, 249, 281, 449, 461
- Presentation layer, 449, 461
- Primary rate access, 561
- Print servers, 390, 407
- Printable, 50, 179, 245
- Priority, 197, 212, 384, 385, 402–405, 516, 523–525, 551, 573, 577, 727, 730, 916
- Prism, 721
- Private, 311, 646, 649, 650, 652, 791
- Private key, 171, 172, 180, 181, 187, 191, 192, 311
- Probability, 13, 15, 22, 24, 93, 138–142, 146, 175, 475, 787

- Process scheduling, 395  
Processor interrupts, 825  
Professional interface format, 113  
Professional sound equipment, 126  
Programmable Interrupt Controller, 637, 638, 661, 664, 695, 698, 826–828, 830, 831, 853  
PROM, 941  
Prototyped, 836–838  
Proxy, 263, 264, 266, 268, 496  
PS, 248, 432, 682, 698, 699, 707, 708  
PSK, 605, 606  
PSTN, 460, 547, 580, 820  
Psycho-acoustic model, 122  
PTC, 623  
Public, 171, 174, 267, 311, 359, 360, 464, 746, 788, 861  
Public switched data network, 460  
Public switched telecommunications network, 460  
Public telephone, 196, 204, 250, 460, 561, 580, 595  
Public telephone network, 250, 561, 580, 595  
Public-key, 171, 172, 174, 180, 190, 191  
Pulse coded modulation, 547  
QAM, 606, 943  
QCIF, 943  
QIC, 725, 726, 943  
Quadrature modulation, 70  
Quantization, 8, 10, 11, 32, 51, 55, 56, 63–67, 80, 89, 96, 98, 101–105, 125, 129, 786, 787  
error, 8, 11, 96  
level, 11, 96, 102, 105  
noise, 96, 98, 102, 129  
process, 11, 89  
scale factor, 125  
Queen Elizabeth, 2  
Queen Margaret College, 542  
Queen Victoria, 3  
QUIET, 533  
RADAR, 4  
Radiation, 753  
Radio, 289, 350  
Radio button, 289, 350  
RAID, 419, 420, 436  
    level 0, 420  
    level 5, 419, 420  
RAM, 507, 684, 692, 695, 818, 834, 849, 940, 944, 945  
Random frame, 89  
Random numbers, 326  
Raster line, 74  
Raytheon, 5  
RD, 493, 601, 612, 617, 620, 624–627, 809, 815, 944  
Real-time, 7, 29, 87, 230, 405, 516, 541, 545–548, 555, 560, 561, 672, 710  
Real-time data, 516, 545, 547, 548, 560, 561  
Real-time sampling, 546  
Receiver Not Ready, 501, 567  
Receiver Ready, 501, 567, 815  
Recording, 111, 683, 722  
Rectangular, 323  
Red Book, 723, 724  
Redness, 52  
Redundancy, 13, 22, 26, 83, 100, 127, 128, 143, 149, 154, 499, 524, 531, 568, 720  
Redundant data, 13, 34  
Reed-Solomon, 117, 160, 161  
Reel-to-reel, 725  
Reflection coefficient, 772–774, 777  
Reflections, 767, 771, 775, 812  
Refracted, 751  
Refraction, 751, 754  
Refractive index, 751, 752, 754  
Refresh rate, 73, 74, 121  
Register, 836–838  
Registers  
    general-purpose, 841  
REJ, 501, 567, 575, 578, 586–588, 944  
Reject, 501, 567, 575, 585–588  
Relationship, 299, 300, 304, 382, 386  
Relationship operator, 299, 382  
Remote Procedure Call, 216, 441, 442  
Repeater, 466, 467, 508, 518, 521, 528  
Repetitive sequence, 17, 21  
Repetitive sequence suppression, 17, 21  
Repetitive sequences, 16, 33  
Representations, 8  
Reservation bits, 525  
Resistance, 2, 4, 146, 296, 465, 504, 509, 746–748, 767, 771, 775, 812  
Resolution, 13, 32, 36, 45, 68, 82, 94, 129, 131, 415, 440, 495, 496, 548  
Resource arbitrators, 395  
Restricted token, 531  
Return type, 310, 313, 314, 331  
Return value, 326  
Revision, 692, 793  
RFC, 174, 181, 231, 233, 237–243, 246, 264, 487, 493, 788, 789, 792, 793, 795, 797  
    821, 231, 233, 239  
    822, 231, 237–240, 242, 243, 246, 264  
RG-50, 510, 511  
RG-6, 511

- RGB, 33, 38, 52, 69, 83, 85, 276, 868, 944
- RI, 527, 649, 814, 944
- Rich text format, 944
- Ring
  - three, 390
  - zero, 390
- Ring fails, 530
- Ring in, 527, 534, 944
- Ring network, 195, 199, 464, 470, 471, 472, 498, 516, 522, 526–529, 537, 692
- Ring out, 527, 944
- Ring topology, 464, 531
- RIP, 427, 429, 430, 438, 440, 469, 470, 475, 476
  - packet format, 430, 475, 476
- RJ-45, 467, 504, 510, 511, 517, 527, 528, 563, 748
- RLE, 18, 19, 32, 33, 34, 45, 47, 49, 57, 608, 672, 673, 675, 783, 784, 944
  - count, 673
- RNR, 501, 567, 575, 578, 586–588, 944
- RO, 527, 944
- Robust networking, 419
- ROM, 7, 13, 83, 122, 144, 653, 692, 710, 722, 724, 816, 824, 834, 836, 855, 939, 944
- ROM BIOS, 824, 834
- Root, 324
- Routers, 260, 261, 429, 465, 468, 469, 474, 791
- Routing protocol
  - BGP, 469
  - EGP, 441, 444, 469
  - NLSP, 438, 469
  - OSPF, 469, 476, 477, 790, 796
  - RIP, 427, 429, 430, 438, 440, 469, 470, 475, 476, 481, 795
- RPC, 216, 441, 442, 445, 448
- RR, 501, 567, 575, 578, 586–588, 794, 815, 944
- RS-232, 154, 403, 413, 463, 464, 470, 581, 595, 596, 610–620, 624, 628, 633, 634, 637, 638, 641, 644, 651, 808, 809, 814
  - bit stream timings, 614
  - communications, 595, 611, 614, 619, 620, 628
  - DTE-DCE connections, 619
  - frame format, 114, 535, 563, 612
  - programming, 620
  - setup, 616
- RS-232-C, 464
- RS-328, 608
- RS-422, 464, 610
- RS-449, 610, 814, 815
- RSA, 172, 174, 180, 182, 190–192
- RTF, 944
- RTS, 612, 616, 618, 620, 642, 647, 809, 814, 817, 818, 823
- Run length coding
  - black, 47, 609
  - white, 47, 608
- Run length count, 671, 673
- Run-length encoding, 18, 19, 32–34, 45, 47, 49, 51, 57, 608, 672, 673, 675, 783, 784
- S/PDIF, 115, 116
- SABME, 567, 568, 578, 944
- SAC, 944
- Sampled data, 560
- Sampling, 9, 10, 22, 52, 78, 79, 80, 83, 98, 100, 107, 108, 111–115, 119, 121, 126–128, 132, 133, 243, 545, 546, 548, 683
  - frequency, 115
  - rate, 9, 22, 52, 80, 100, 108, 111, 113, 114, 119, 121, 126, 127, 128, 243, 546, 548, 683
  - theory, 9, 546
- SAP packet format, 431
- SAPI, 565, 566, 569, 944
- Satellite TV, 122, 750
- Scalability, 514
- Scanning frequency, 78
- SCMS, 116, 117
- Scot, 3, 4
- Scotland, 4, 6, 542
- Scrambled, 170
- SCSI, 398, 676, 692, 710, 713, 714, 727–736, 738, 944
  - A-cable, 728
  - B-cable, 728
  - commands, 736
  - host, 729
  - interface, 729
  - layer, 398
  - logical unit number, 736
  - message codes, 734
  - message format, 734
  - message system description, 734
  - P-cable, 728
  - pointers, 734
  - SCSI-I, 727–730
  - SCSI-I connections, 729
  - SCSI-II, 714, 727–730
  - SCSI-III, 727, 728
  - tagged command queue, 727, 728
- SCSI states
  - arbitration, 731
  - command, 732

- data, 732
- free/bus, 731
- message, 732
- selection, 731
- status, 732
- SCSI-ID, 727, 730–732
- SCSI-III, 727–730
- SD, 421, 523, 535, 601, 715, 815, 944
- SDH, 554, 559, 796, 944
- SDIF-2, 112, 113
- Seagate Technologies, 713
- SECAM, 68–71, 75, 77, 78, 80, 81, 84, 85, 92, 93
- Secret key, 171, 180
- Section header, 269
- Sectors, 410, 710–712, 715, 717, 720, 721, 723
- Security permissions, 412
- Segment, 63, 65, 67, 68, 102, 104, 105, 144, 195, 199, 213–215, 397, 408, 466, 492, 497, 508–511, 513, 516, 518, 520, 521, 556, 557, 825, 842–844
- Selection statements, 304
- Sensor, 146, 721
- Sequence number, 145, 199, 213, 214, 428, 429, 432, 501, 555, 557, 558, 566, 567, 584, 585, 586, 588
- Sequencing of data, 501
- Serial, 116, 393, 464, 620, 624, 684, 695, 699, 789, 792, 825, 828
- Serial communications, 112, 464, 610, 611, 620, 628, 633, 641, 642, 650, 657, 676, 814
- Serial copy management, 116
- Server, 180, 204–207, 213, 214, 216, 217, 220, 224, 230–234, 236, 246–248, 250–256, 262–268, 286, 289, 290, 292, 320, 359, 370, 371, 374, 375–377, 392, 389, 390, 406, 407, 412–415, 418, 419, 422–427, 430–448, 455–471, 485, 486, 491–496, 518, 520, 631, 632, 661, 704, 788, 794, 798–802, 806, 918
- Server name, 252, 254
- Service quality, 548
- Session layer, 442, 461
- Set asynchronous balance mode extended, 567
- Seven-layer OSI model, 461
- SHEFC, 542, 944
- Shielded twisted pair, 522, 527
- Shift registers, 137, 143
- Sibling Property, 26
- Signal-to-noise ratio, 88, 90, 96, 110, 126, 129, 787
- SIMM, 703
- Sinclair, 5
- Sine, 323, 605
- SIPO, 111
- SIPP, 211
- Slave In, 539
- Slave Out, 539
- SMA, 510, 757
- SMARTDRV.EXE, 397
- SMDS, 218, 790
- SMT, 531, 533, 534, 542, 944
- SMTP, 214, 215, 230–240, 246, 441, 788, 792, 793, 796
- example transmission, 237, 238
- MIME, 232
- responses, 236
- transfer, 235
- SNA DLC, 412
- SNMP, 214, 441, 485, 486, 488–490, 789, 790, 795
- SNR, 90, 91, 96, 97, 104, 119, 786
- Socket, 221–321, 359, 370–376, 431, 708, 709, 795, 918, 919
- connection, 371
- creating, 373
- number, 213
- Software, 292, 653, 824, 825, 834
- Software handshaking, 616–618
- Software interrupts, 824
- Solaris, 439
- SONET, 554, 796, 944
- Sony, 5, 112, 115, 204, 723
- Sony Digital Interface, 112
- Sound, 1, 13, 80, 116, 121–129, 181, 229, 248, 251, 253, 265, 399, 679, 683, 684, 699, 827, 828, 830, 865
- Source
  - coding, 22, 98
  - address, 257, 499, 523–525, 536, 842
  - encoding, 16, 51
- Source and destination address, 472, 497, 498, 522, 525
- SPACE, 15, 148, 149, 177, 779, 873, 890, 905
- Spanning-tree bridge, 466
- Speaker volume, 597, 816
- Specification, 281
- Speech, 2, 3, 7, 8, 10, 22, 101–104, 107, 109, 128, 229, 230, 249, 460, 516, 541, 545, 546, 547, 548, 560, 594
- Speech, 22, 96, 100, 102, 107, 108
- Speech compression, 102, 108
- Spies, 170

- SPX, 406, 412, 413, 414, 419, 424–429, 434–436, 463, 469, 470, 499  
 packet format, 429  
 SPX/IPX, 406, 413, 414, 419, 424, 434–436, 463  
 SPX/IPX, 406, 413, 414, 419, 424, 435, 436, 463  
 SQDV, 68  
 Square root, 324, 904  
 SRAM, 695, 699, 849, 850  
 SREJ, 145, 575  
 ST connector, 510  
 ST-506, 713, 714  
 Standalone, 293, 294, 295, 324, 407, 608  
 Standards, 68, 82, 113, 182, 274, 275, 460, 463, 498, 542, 676, 684, 724, 788, 790, 793, 794  
 Star network, 457, 470, 471, 513  
 Start and end delimiter, 523, 524, 535, 566  
 Start bit, 536, 595, 612–614, 808  
 Start delimiter, 498, 524, 525, 535, 636  
 Stateless protocol, 255, 262  
 Statement block, 307, 308, 387  
 Statements, 301, 381, 382  
 Static interference, 757  
 Static methods, 316  
 Station management, 531  
 Statistical  
   encoding, 16, 17  
   multiplexing, 549  
 Stereo audio, 122, 127  
 Stereophonic, 121  
 Still images, 83  
 Stirling, 542, 559  
 STM, 548–550, 554, 559  
 STM-1, 554, 559  
 Stop bit, 595, 596, 612–614, 616, 623, 626, 627, 637, 644, 650, 652, 808  
 STP, 513, 527–529, 945  
 String manipulation, 913  
 Strings, 28, 295–297, 327, 360, 359, 365, 378, 380, 385, 857, 860, 895, 896, 903, 913, 923  
 Striping with parity, 419  
 Strowger, 6  
 STS-1, 554  
 SUB, 269  
 Sub-band, 125–129, 133  
 Subcarrier, 69, 70, 75, 76, 77  
 Subnet, 195, 202–204, 206–208, 261, 415, 476, 493, 495, 592  
   masks, 203, 476  
 subsampling, 52, 69, 84  
 Subsampling, 79  
 Sun Microsystems, 246, 293, 439, 441, 793  
 SuperJANET, 218, 542, 559  
 Supervisory frames, 567, 574–576  
 Suppressed-carrier, 76  
 Suppressing repetitive sequences, 16  
 SVGA, 82, 945  
 Switch, 4, 218, 260, 261, 306, 518–520, 526, 550, 552, 661, 730, 740, 806  
 Switch(), 306  
 Sync pulse, 70, 74, 75, 77  
 Synchronization, 89  
 Synchronization bits, 561, 563  
 Synchronous, 535, 536, 541, 548, 554, 571, 595, 682, 699, 708, 710, 727, 814, 818, 836, 944  
 Synchronous modems, 595  
 Syndrome, 157–160  
 Syntax, 836–838  
 System Management Mode, 707, 852  
 System policies, 392  
 System reset, 849  
 System timer, 825, 828, 830  
 Tab, 297, 298, 380, 703, 857, 873, 890  
 Tables, 27, 32, 35, 42, 47, 49, 55, 64, 65, 67, 68, 89, 281–283, 425, 429, 436, 479, 591, 779, 781, 792  
 Tape backup, 393, 420  
 TCP, 1, 195–200, 211–221, 230, 233–237, 247, 256, 262, 263, 268, 275, 292, 359, 372, 390, 393, 412–416, 419, 421, 424, 425, 428, 430, 434–440, 444, 454, 456, 463, 468, 469, 486, 491, 493, 495, 497, 499, 500, 553, 632, 788, 789, 790, 793, 795, 796, 945  
   header format, 213, 214  
   protocol data unit, 213  
 TCP/IP, 1, 195–200, 211, 215, 216, 217, 220, 230, 233, 247, 256, 275, 292, 359, 390, 412–419, 421, 424, 425, 430, 434–440, 454, 456, 463, 468, 469, 486, 491, 493, 495, 497, 499, 500, 553, 632, 788, 789, 790, 796  
   class A, 200, 203  
   class B, 200, 201, 203, 555  
   class C, 200, 203, 555  
   gateway, 196, 200, 430  
   implementation, 199  
   internets, 199  
   ports and sockets, 213  
   version number, 197  
 TD, 283, 284, 285, 612, 617, 620, 624–627, 809

- TDAC, 133  
TDI, 413, 419, 687, 692  
TDM, 107, 108, 109, 547, 548, 562, 945  
multiframe, 108, 109  
TE, 563–565, 568, 569  
TEI, 565, 566, 568, 569, 945  
Telephone, 1, 3, 6, 10, 107, 195, 196, 204, 229, 250, 347–349, 460, 464, 522, 546–548, 552, 561, 565, 566, 580, 590, 594, 595, 597, 598, 600, 603, 604, 605, 608, 628, 630, 631, 746, 748, 818  
Telephone number, 195, 196, 204, 552, 598, 628, 630, 631, 818  
Teletex, 561  
Television, 1, 68, 84, 90, 121, 122, 132  
Telnet, 195, 211–215, 217, 247, 253, 255, 321, 376, 406, 415, 421, 422, 441, 789, 790, 794, 797  
Terminal, 599, 612, 616, 628, 634, 640, 678, 789, 814, 815  
Text, 248, 594, 650, 652, 788, 790, 794, 861, 864  
background color, 44  
based, 7, 170, 230, 255, 454, 845  
blink, 269  
editor, 179, 254, 269, 281, 295  
foreground color, 44  
underlined, 270  
Thales, 2  
Throughput, 197, 396, 469, 514, 518, 545, 586, 591, 678, 683  
Ticker timer, 825  
TIF, 33, 34, 945  
TIFF, 32, 45, 46, 47, 48, 51  
coding, 45  
compression, 47  
IFD, 45–47  
Tilde, 313, 859, 860  
Time, 825  
Time domain, 123  
Timestamp, 199  
Token, 289  
Token Ring, 151, 195, 196, 199, 208, 274, 282, 289, 290, 413, 428, 463–465, 467, 468, 472, 491, 498, 516, 520–531, 537, 540, 545, 549, 579, 790  
adding to the ring, 526  
cable, 522  
data exchange, 523  
deletion from ring, 526  
deletion from the ring, 526  
fault management, 526  
jitter suppression, 529  
maintenance, 526  
MAUs, 503, 522, 526–529, 797  
ring initialization, 526  
Top-down, 328  
Topology, 208, 210, 248, 270, 464, 465, 470–472, 477, 478, 497, 513, 531  
Total internal reflection, 754  
TR, 282–285, 601, 815, 945  
Traceroute, 217–219  
Tracks, 710, 711  
Traffic, 1, 81, 92, 212, 229, 249, 250, 256, 257, 260, 261, 264, 424, 429, 430, 435–438, 465–477, 485, 490, 491, 497, 503, 504, 510, 513, 518, 520, 522, 530, 531, 536–538, 541, 545, 548–550, 554, 555, 558–560, 581  
congestion, 558  
profile, 545  
profiles, 545  
statistics, 533  
Transceiver, 503, 504, 507, 509–511, 850, 854  
Transfer length, 736, 737  
Transfer rates, 724  
Transform encoding, 22  
Transistor, 4, 5  
Transmission channel, 139, 236, 605  
Transmission line, 767–776, 809  
Transport, 195, 253, 406, 428, 440, 463, 483, 540, 554, 791, 792, 795  
Transport layer, 195, 197, 213, 406, 427, 440, 461, 462  
Tree topology, 471  
Trellis, 163, 164, 166, 167, 168, 169, 607  
TUBA, 211  
Tunnel, 263, 264, 424  
TV  
compressed, 81  
uncompressed, 81, 93  
system, 52, 73, 82, 122  
Twisted-pair, 464, 466, 470–472, 497, 503, 508–516, 522, 527, 528, 563, 610, 746–748, 762–764, 767, 945  
hubs, 510  
TXC, 695–703  
TXT, 248  
UI, 502, 567, 945  
UK, 68, 80, 81, 107, 108, 216, 218, 238, 239, 248, 265, 559  
Unbalanced, 572  
Uncompress, 30, 31  
Uncompressed, 13, 33, 42, 594, 722  
Uncompressed Audio, 121

- Uncompressed TV, 81, 93
- Underlined text, 270
- UNI, 549, 550, 553, 558, 945
- Unicast, 212, 493
- Unicode, 296, 298, 794, 796, 895, 896
- University College, London, 218, 559
- University of Bath, 204, 218, 219
- University of Edinburgh, 204, 218, 542, 559
- University of Stirling, 542, 559
- UNIX, 26, 29–31, 217, 220, 234, 250, 251, 256, 286, 293, 389, 406, 410, 424, 439, 440, 443, 445, 446, 448, 461, 469, 454, 458, 798, 802
- Unnumbered information, 567
- Unordered lists, 272, 273
- Unrestricted token, 531
- Uppercase, 265, 905
- UPS, 420
- URI, 255, 264–268, 863, 864
- URL, 247, 252–255, 262, 271, 286, 292, 321, 359, 364–369, 791–797, 862, 864–886, 919, 920, 945
- USA, 2, 3, 68, 102, 107, 108, 116, 190, 216, 218, 546, 562
- USB, 695, 698, 699, 704, 707, 710
- UseNet, 247, 252, 255
- User network interfaces, 549
- UTP, 503, 510, 511, 513, 527–529, 747, 748
- UV, 945
- V.21, 596, 605, 606, 816, 819
- V.22bis, 596, 606, 607
- V.32, 596, 606, 607
- V.32bis, 596, 606, 607
- V.42bis, 28, 596, 606
- Vampire, 510
- Variable, 290, 299, 305, 310, 311, 324, 330, 331, 337, 637, 639, 640, 664, 724
- Variable frequency, 548
- Variable length code, 24, 58, 84
- VCI, 549–554, 584, 586, 591, 592, 945
  - header, 551
  - label, 550, 551, 586, 591, 592
- VCO, 505, 506
- VCR, 68, 81, 84, 91, 92, 93, 116, 722, 945
  - quality, 68, 81, 93, 722
  - type, 81, 92
- VDDs, 405
- Vector information, 87
- Velocity, 750, 751, 771
- Vertical redundancy checking, 154
- VESA, 684–686
- VFAT, 389, 397
- VGA, 50, 684, 692, 834, 945
- Video, 2, 7–10, 13, 16, 22, 68, 70, 73–94, 98, 121, 122, 125, 230, 243, 244, 248, 251, 256, 277, 292, 460, 516, 541, 545, 546, 548, 549, 555, 559, 594, 672, 676, 684, 685, 692, 710, 722, 724, 725, 726, 746, 771, 793, 794, 796, 825, 945
  - compression, 77, 81, 84, 87, 93, 122, 125
  - conferencing, 68, 79, 90
  - sequence, 88, 89, 92
- Videotex, 561
- VIM, 231, 232
- Violation, 181, 524, 533
- Virtual
  - channels, 552
  - circuit, 549
  - circuit identifier, 549
  - data flow, 460
  - device driver, 392, 393, 398, 405
  - loadable modules, 436
  - machine manager, 395
  - path, 546, 550–553
- Visible, 177, 359, 360, 541, 755, 884
- Visual Basic, 286, 641, 642
- VLC, 28, 42, 58, 59
- VLC-LZW, 28
- VL-local bus, 683–686, 713
- Volatile, 758
- VRC, 154, 155, 161
- VxD, 392, 393, 398
- WAIS, 252, 255
- WAN, 1, 459, 460, 463, 464, 469, 470, 580
- Washington, 3
- Watt, 2, 3, 4, 542, 559
- WAV, 9, 248
- Waveform, 10, 71, 74, 75, 104, 504, 506, 516, 759, 760, 762, 786
- Web, 204, 215, 247, 248, 250–252, 263, 292, 364, 794
- Western Electric, 5
- Westinghouse, 5
- Wheatstone, 3
- While(), 308, 387
- White, 16, 38, 47, 52, 68, 69, 74, 75, 78, 80, 271, 276, 277, 517, 608, 861, 868
- White Book, 724
- White run length coding, 47, 608
- White-space, 16
- WIMPs, 454
- Win32, 399, 404, 633
- Winchester, 710
- Windows 95/98, 6, 29, 250, 251, 389, 390–401, 410, 412, 599, 616, 628, 630, 631, 633, 824, 828, 834, 853

- registry, 390
- sockets, 419
- Windows 3.x, 389–393, 395, 397, 398, 401, 402, 404, 405, 410, 599, 628
- Windows NT, 6, 250, 251, 293, 325, 389, 390, 392, 398, 401, 404–425, 432, 436, 491, 630, 631, 633, 727, 853
  - network drives, 422
  - network interfaces, 413, 414
  - setting up TCP/IP, 415
- WINS, 415, 418, 485, 495, 496
- WinSock, 220, 221, 393, 419
- WINSOCK.DLL, 419
- WINSOCK32.DLL, 419
- Words, 25, 26, 28, 80, 137, 143, 147, 170, 229, 247–249, 255, 507, 513, 607, 691, 795, 863
- Workgroups, 407, 435
- Workstation, 87, 292, 407, 408, 432, 439, 448, 468
- World War I, 6
- World Wide Web, 204, 215, 247, 252, 364, 794
- WORM, 710, 722
- WP, 743
- WWW, 203, 204, 215, 216, 247, 249, 250, 252, 255, 256, 262, 264, 265–270, 273–281, 292, 293, 324, 359, 363, 366–369, 376, 415, 507, 594, 790
- X-Windows, 789, 790
  - clients, 457
  - fundamentals, 455
  - history, 458
  - networking aspects, 457
  - programs, 455
  - terminals, 208
- xcalc, 458
- xclock, 458
- xhost, 458
- xinit, 458
- xkill, 458
- Xlib, 456
- xload, 458
- xmag, 458
- xpr, 458
- xterm, 458
- xwd, 458
- X.21, 463, 464, 581–583
- X.25, 463, 464, 571, 579, 580, 581, 583, 587, 588, 590, 592
- X.400, 231, 232, 239
- X/Open, 439
- X3T9.5, 464, 538, 542
- XD bus, 855
- Xerox Corporation, 497
- X-OFF, 616–618
- X-ON, 616–618
- XOR, 135–137, 143, 147–149, 155, 156, 161, 169, 176–178, 181, 182, 185, 186, 188, 301, 304, 383, 384, 419, 613, 659
- X-terminals, 208
- X-Windows, 440, 441, 454, 456, 457, 458
- Y:Cb:Cr, 52, 62, 79
- Yellow Book, 723, 724
- Yellow Pages, 442
- Ypcat, 446, 447
- Ypinit, 446, 801
- Ypmake, 447
- Ypwhich, 447, 801
- YUV, 52, 78, 80, 83, 85, 95
- Zero bit-stuffing, 565, 566
- Zigzag, 58, 67, 89
- Zilog, 5
- ZIP, 29, 248, 268, 292, 320, 594
- Zoo, 31