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

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
  int freq[MAXPAROLA] ; /* vettore di contatori
  delle frequenze delle lunghezze delle parole */
  char riga[MAXRIGA] ;
  int i, inizio, lunghezza ;
  FILE * f ;

  for(i=0; i<MAXPAROLA; i++)
    freq[i]=0 ;

  if(argc != 2)
  {
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
  }
  f = fopen(argv[1], "rt") ;
  if(f==NULL)
  {
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
  }

  while( fgets( riga, MAXRIGA, f ) != NULL )
```

The File System

Files in Linux

Stefano Quer
Dipartimento di Automatica e Informatica
Politecnico di Torino

Text and binary files

- A file is basically a sequence of bytes written one after the other
  - Each byte includes 8 bits, with possible values 0 or 1
  - As a consequence all files are binary
- However, we normally make a distinction between
  - Text files (ASCII or UNICODE)
  - Binary files

C sources, C++, Java, Perl, etc.

Executables, Word, Excel, etc.

Remark:
The UNIX/Linux kernel does not distinguish between binary and text files

Text files

Or UNICODE

- Files consisting of data encoded in ASCII
  - Sequence of 0 and 1, which (in groups of 8) encode ASCII symbols
- Textual files are usually “line-oriented”
  - Newline: go to the next line
    - UNIX/Linux and Mac OSX
      - Newline = 1 character
      - Line Feed (go to next line, LF, 10_{10})
    - Windows
      - Newline = 2 characters
      - Line Feed (go to next line, LF, 10_{10})
        + Carriage Return (go to beginning of the line, CR, 13_{10})

Binary files

- A sequence of 0 and 1, not “byte-oriented”
- The smallest unit that we can read/write is the bit
  - It is not easy to manage single bits
  - Sequence of 8 bits do not necessarily correspond to printable characters, new-line, etc.
- Why are binary files used?
  - Compactness
    - Examples
      - Number 100000_{10}
      - Text/ASCII format: 6 characters, i.e., 6 bytes
      - Binary format: coded as integer (short) on 4 bytes

Example

A string in a text or binary file

"ciao"

'c' 'i' 'a' 'o'

99_{10} 105_{10} 97_{10} 111_{10}

01100011_{2} 01101001_{2} 01100100_{2} 01101111_{2}

An integer number in a text file

"231"

'2' '3' '1'

50_{10} 51_{10} 49_{10}

00110010_{2} 00110011_{2} 00110001_{2}

An integer number (on one byte) in a binary file

"231"

"231_{10}"

11100111_{2}

Example

```
FILE *fp;
int fd;
char sv[] = "This is a string";
int iv = 10;
float fv = 15.55;

fp = fopen ("my_file_1.txt", "w");
fprintf (fp, ...);
fclose (fp);

fd = open ("my_file_1.bin", O_WRONLY|O_CREAT|O_TRUNC,
  S_IRUSR|S_IWUSR);
write (fd, ...);
close (fd);
```

ASCII file

Binary file

Example

ASCII file

```
fprintf (fp, "%s", sv);
fprintf (fp, "%d", iv);
fprintf (fp, "%f", fv);
```

T = 54 hex

```
This is a string
```

```
> hexdump -C my_file_1.txt
00000000  54 68 69 73 20 69 73 20  61 20 73 74 72 69 6e 67
00000010
```

Memory addresses

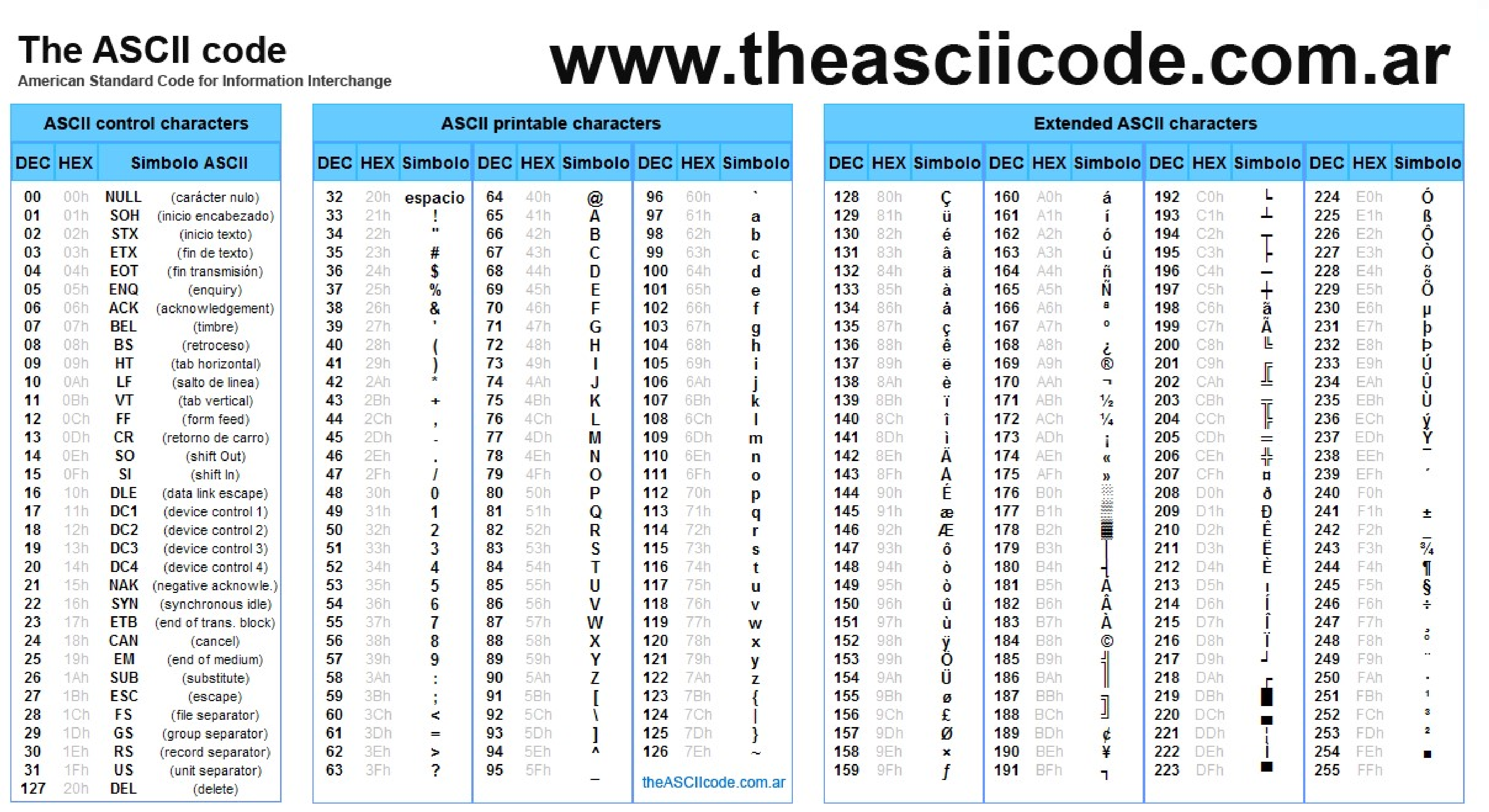
Binary file

```
write (fd, sv, strlen (sv));
write (fd, &iv, sizeof (int));
write (fd, &fv, sizeof (float));
```

Same content

```
> hexdump -C my_file_1.bin
00000000  54 68 69 73 20 69 73 20  61 20 73 74 72 69 6e 67
00000010
```

Extended ASCI



The ASCII code

American Standard Code for Information Interchange

www.theasciicode.com.ar

ASCII control characters

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

ASCII printable characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_			

theASCIIcode.com.ar

Extended ASCII characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	å	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	ë	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	240	F0h	
145	91h	æ	177	B1h	▒	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	▓	210	D2h	Ê	242	F2h	‗
147	93h	ô	179	B3h	│	211	D3h	Ë	243	F3h	¾
148	94h	ö	180	B4h	┤	212	D4h	È	244	F4h	¶
149	95h	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°
153	99h	Ö	185	B9h	╣	217	D9h	┘	249	F9h	¨
154	9Ah	Ü	186	BAh	║	218	DAh	┌	250	FAh	·
155	9Bh	ø	187	BBh	╗	219	DBh	█	251	FBh	¹
156	9Ch	£	188	BCh	╝	220	DCh	▄	252	FCh	³
157	9Dh	Ø	189	BDh	¢	221	DDh	¦	253	FDh	²
158	9Eh	×	190	BEh	¥	222	DEh	Ì	254	FEh	■
159	9Fh	ƒ	191	BFh	┐	223	DFh	▀	255	FFh	

Example

ASCII file

```
fprintf (fp, "%s", sv);
fprintf (fp, "%d", iv);
fprintf (fp, "%f", fv);
```

T = 31 hex

```
10
```

```
> hexdump -C my_file_2.txt
00000000  31 30
00000002
```

Memory addresses

Binary file

```
write (fd, sv, strlen (sv));
write (fd, &iv, sizeof (int));
write (fd, &fv, sizeof (float));
```

0000-1010 0000-0000 0000-etc.
Litte endian = Least significant value is stored first

```
> hexdump -C my_file_2.bin
00000000  0a 00 00 00
00000004
```

0a = 0000-1010
= one byte

Example

ASCII file

```
fprintf (fp, "%s", sv);
fprintf (fp, "%d", iv);
fprintf (fp, "%f", fv);
```

T = 31 hex

```
15.550000
```

```
> hexdump -C my_file_3.txt
00000000  31 35 2e 35 35 30 30 30 30
00000009
```

Memory addresses

Binary file

```
write (fd, sv, strlen (sv));
write (fd, &iv, sizeof (int));
write (fd, &fv, sizeof (float));
```

```
> hexdump -C my_file_3.bin
00000000  cd cc 78 41
00000004
```

The IEEE 754 notation for floating point numbers plus litte endian