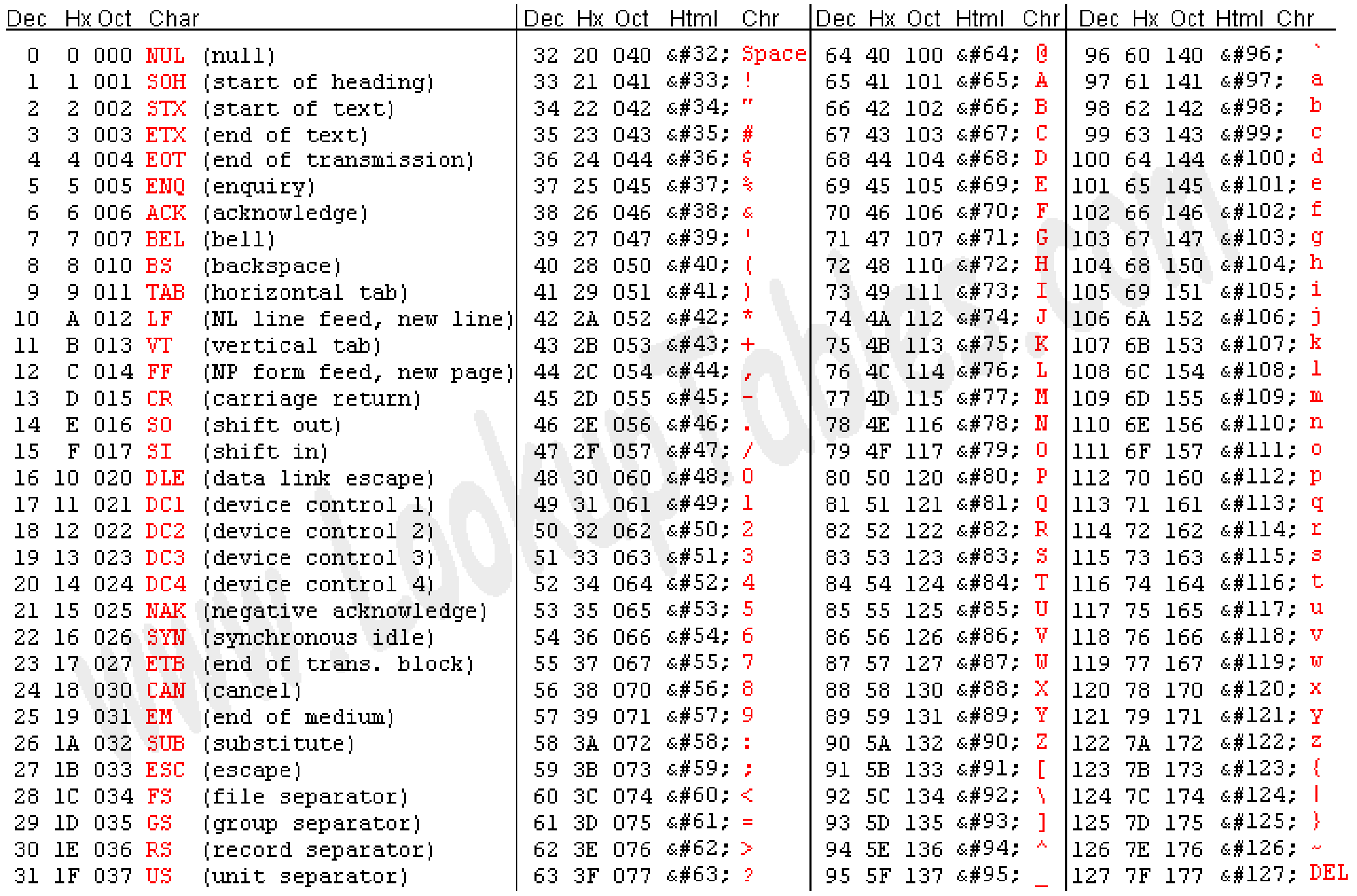
Files

Lecture 1

Basis of Information

- bit – 0 or 1
- byte: 8 bits
  - It can store 2^8 = 256 different symbols (number)
- Kilo, Mega, Giga, Tera…
  - 2^10 = 1024
  - 2^20 bit more than 1 million
  - 2^30 bit more than 1 billion
  - 2^40 bit more than 1 trillion

The ASCII Table



Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH	(start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX	(start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX	(end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT	(end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ	(enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK	(acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL	(bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS	(backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB	(horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT	(vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR	(carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO	(shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI	(shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE	(data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1	(device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2	(device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4	(device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK	(negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN	(synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB	(end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN	(cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM	(end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB	(substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC	(escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS	(file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS	(group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS	(record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US	(unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

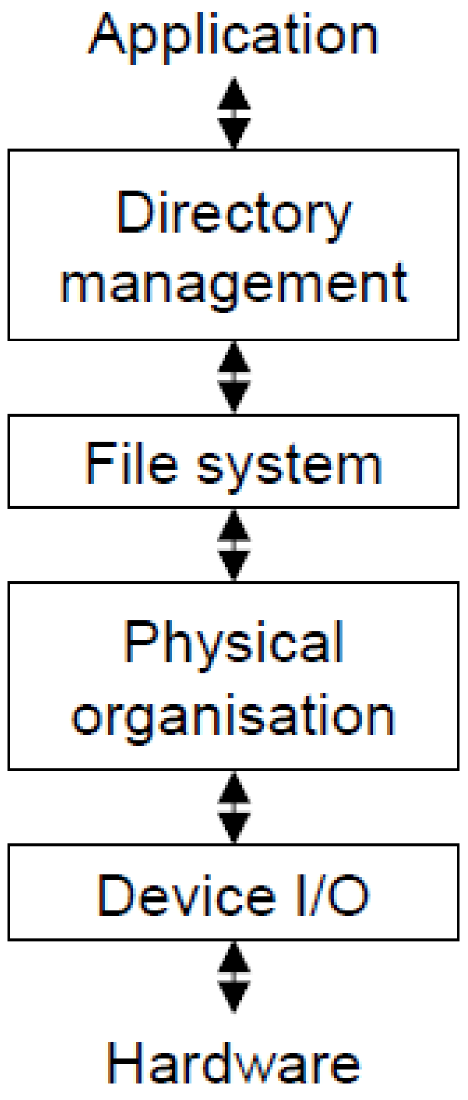
What is a file

Definition

- A collection of records involving a set of entities with certain aspects in common and organized for some particular purpose *Tremblay and Sorenson [1984]*
- A collection of similar records kept on secondary computer storage devices *Wiederhold [1983]*

- *In C when a file is open it's a sequence of data (stream of bytes).*
- There are two types of files in C*: text and binary*

Where files are



- Abstract layer approach
- Application works with names
- In order to translate names into hardware location and instructions, many steps are needed

File organisation

Organised?

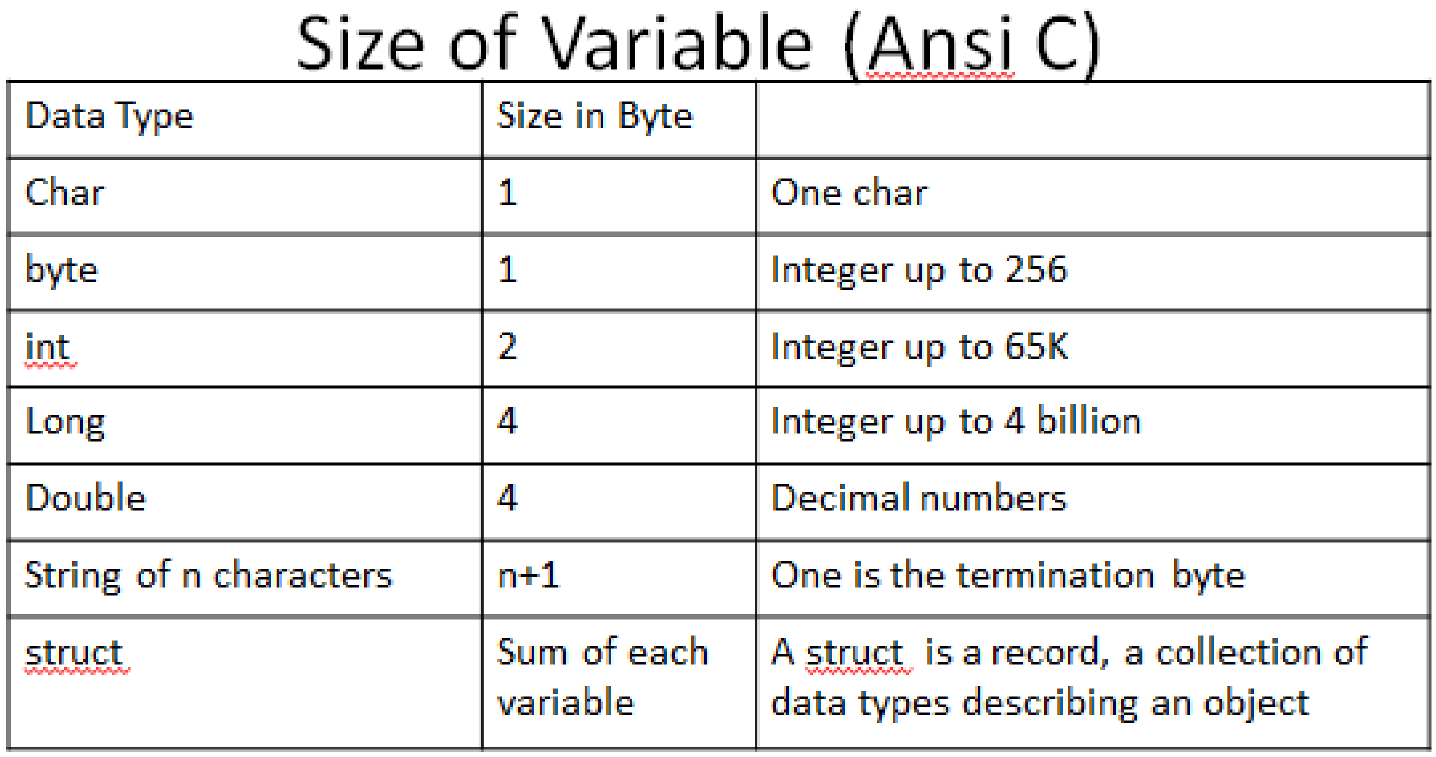
- On disk they are all the same
- Data in a file does not somehow organize itself.
- The decisions related to structuring a file are among the most critical decisions made by the designer of a file system.

<stdio.h>

Standard Input/Output library

What do we know about this library?

What’s in a C file?



Size of Variable (Ansi C)

Data Type	Size in Byte	
Char	1	One char
byte	1	Integer up to 256
int	2	Integer up to 65K
Long	4	Integer up to 4 billion
Double	4	Decimal numbers
String of n characters	n+1	One is the termination byte
struct	Sum of each variable	A struct is a record, a collection of data types describing an object

- To determine the size in the code before assigning resources = `sizeof(variableName)`

Text files

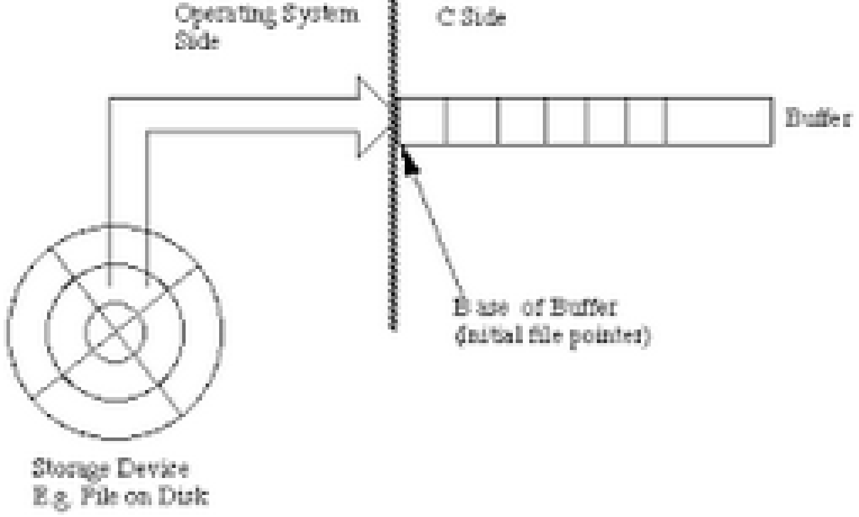
- A *text* file stores a sequence of printable characters.
- Each character is a byte
- In C a character is represented by the variable *char*

```
char ch;

ch='a';

printf("%c", ch);  [output is 'a']
//Remember that internally char are number (ASCII)
printf("%i", ch);  [output is 97]
```

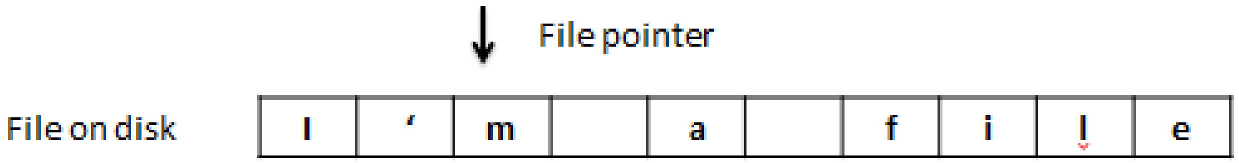
Streams in C


Operating System Side
C Side
Buffer
Base of Buffer (initial file pointer)
Storage Device E.g. File on Disk

- What is a stream?
- What is a modern hardware alternate?
- Why are streams needed?
- Who is Cs sister system?
- C programs have 3 streams
  - `stdin, stdout & stderr`
- For each file to be worked on a stream must be made.
- All streams are equal.

The file pointer

- A file pointer is a C variable, used so far with arrays: `int *p`
- Now the pointer is used with the data type FILE *stream*.
- **`FILE *fp`**;
- If you access the file twice you may use another pointer – to location not another stream.
- A file is always accessed at the position of the pointer!
- The pointer can be moved if needed
- If a pointer is destroyed, this not affect the file!



Files & the *

A file is a pointer to the data type FILE *stream*.

One must be declared & it goes live on the open:

```
FILE *fp;
fp = fopen("write.txt","w");
```

//**fopen** allows you to open a text file

```
 if (fp == NULL)
      {printf("Can't open file.\n");
//check for error, if can't open or you don't
have permission to access then error returns
//very important to always check before working
on the file
```

fopen() & fclose()

- In order to open a file:
  - Specify the name (“write.txt” in the example)
  - Specify the opening mode (“w” in the example)
- If the file cannot be open for any reason, fopen() will set the value of fp to `NULL`

```
//error msg returns if you have one
or the program terminates
```

- In order to close a file: fclose(fp);

```
//very important to manage your
buffer, close when finished.
```

Text File, opening modes

- r : open for reading
- w : Truncate to zero length or create file for writing
- a : open or create file for writing at end-of-file
- r+ : open for reading and writing, start at beginning
- w+ : open for reading and writing (overwrite file)
- a+ : open for reading and writing (append if file exists)

```
// be very careful of w with existing
files
```

Review

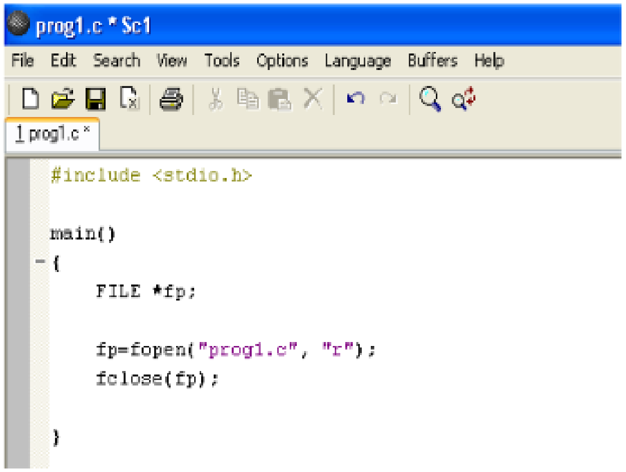
- In C there are buffers required to work with files.
- Streams are declared using **`FILE *fp`**;
- These streams are required for each file that you work on.
- To open and use the stream, error check that the file exists then close when finished:

```
fp = fopen("write.txt","w");
if (fp == NULL)
     {printf("Can't open file.\n");
fclose(fp);
```

Initial lab0

- Opening and closing streams and error checking; rewrite in borlands or gedit etc.
- In webcourses/lab material
- 2 weeks to complete
- Get to know your Teammate; teams start next week; get their # to let them know when your absent etc.
- Only time you

will get screenshots



```
#include <stdio.h>

main()
{
    FILE *fp;

    fp=fopen("prog1.c", "r");
    fclose(fp);

}
```

You should open the bcc5.5 SCI e
manually key in the following sho
program, "prog1.c". *(NB: comr*
each line of the code saying what
for the program).

Compile it and then execute it, id
from the DOS prompt.