26 Files

In C, using files is almost as easy as using printf and scanf.

To introduce named files:

```
FILE * in, * out, * append;

in = fopen ( argv[1], "r" ); // for example
out = fopen ( argv[2], "w" );
append = fopen ( "append", "a" );
```

You can read from input files (”r”, of course) using

```
char buffer[200];

while ( fgets ( buffer, 200, input ) != NULL )
{
}

line-by line, or ...
```

The function `fgets()` reads the input file up to 199 characters (from the 200 given above), or up to a newline character or end-of-data, whichever comes first, and appends `′`
`0′` so the buffer contents are a valid character string.

It (weirdly) returns the starting address of the buffer, except returning a special NULL value to indicate end-of-data. This is weird but very useful.

```
or character-by-character ...

int x;
x = ' ';

while ( x != EOF )
{
  ... do something with x ...
  x = fgetc ( input );
}

This is artificial, as it introduces an extra blank
at the start.

Better:

int finished;

finished = 0;
```

```
  while ( ! finished )
  {
    x = fgetc ( input );
    if ( x == EOF )
      finished = 1;
    else
    {
       ...
    }
  }

  Or, using fscanf.  Now, fscanf() is a FUNCTION,
  returning the number of elements scanned.

  while ( fscanf ( "%d", &x, input ) == 1 )
  {
    ...
  }

  On end of input, scanf may return 0 or -1 --- this
  differs  on different machines.  For that reason,
  use
  while ( fscanf .... == 1 )
  not
  while ( fscanf .... != 0 )

  fscanf (and scanf) should be used only for inputting integers
  and doubles; anything else will probably get messy.

So, in descending order

fgets  reads a line
fscanf reads an item
fgetc reads a character
```

What is end of data? For keyboard input, CTRL-D. If using it this way

```
a.out < mydata
```

end-of-data will be encountered in the natural way at the end of the file mydata.

Closing files. Properly, a file opened for reading should be closed when you are finished with it, but little harm is done if you don't.

A file opened "w" or "a" must be closed, or all the updates will be lost.

```
  fclose ( out );
  fclose ( append );
```