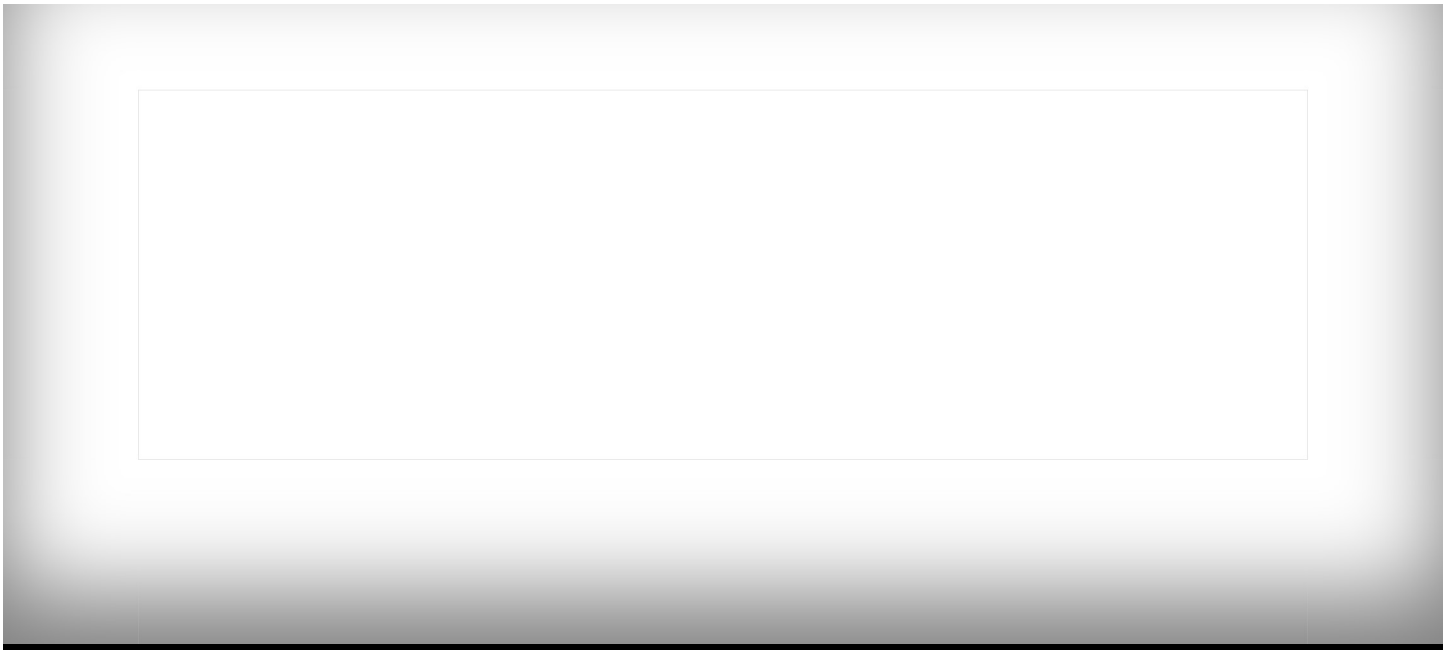


File information

PBM, PGM, and PPM files are all image file formats that hold data in regards to pixels of an image. Compared to formats like PNG, JPG, etc, these formats are very simplistic, offering no compression. These are simple formats that store the colours of pixels as bytes which can be read into your program.

There are 3 types for a reason:

- PBM (Portable BitMap) - 2 colours only. Black and White (0-1)
- PGM (Portable GrayMap) - 255 colours only. Black-Gray-White (0-255)
- PPM (Portable PixMap) - 16,777,216 colours. Coloured RGB (0-255, 0-255, 0-255)

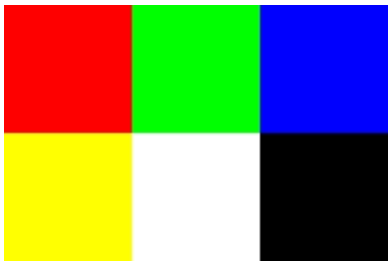
Each of these files also contain a magic number (Wikipedia) which tells if the information is stored as text or as bytes. We will cover this later in the documentation.

What these files look like (PPM Text Format)

Let's get directly into the details. For this example, we will use PPM files since they are the most used. Try opening up a PPM file in a text editor of your choice (as opposed to an image viewer). If the format of the PPM file is stored as numbers, you will likely see this garbage:

```
P3
3 2
255
255   0   0     0 255   0     0   0 255
255 255   0   255 255 255     0   0   0
```

This is a 3x2 image that I stole from Wikipedia (It was public domain, hah). As it is 3x2, it is a very small image. Here is what it looks like when scaled up:



Breaking it down (PPM Text Format)

So you either found out how this format works by now (Because it is simple), or you just want me to explain it. Let's look at that last file again. I will colour code the first three lines.

```
P3
3 2
255
255   0   0     0 255   0     0   0 255
255 255   0   255 255 255     0   0   0
```

P3 - Magic Number (Tells the program this is a PPM file)
3 - Width
2 - Height
255 - Colour Range (0 = Black, This Number = White)

Everything beyond these numbers is pixel information about the image. Let's get to that.

```
255   0   0     0 255   0     0   0 255
255 255   0   255 255 255     0   0   0
```

We all have used Microsoft Paint before, at least I hope we have. We know the image has a size of 3x2. Since this is also a PPM file (Since the "P3" Magic Number was there), we can read the file accordingly. A PPM file stores 3 numbers per pixel. The first is the RED value, followed by the GREEN value, and then finally the BLUE value.

In relation to the data, let's colour code it.

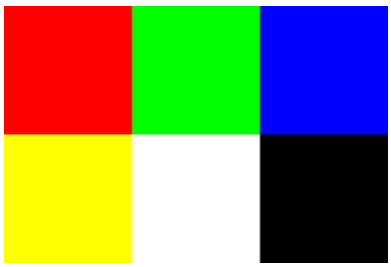
```
255   0   0     0 255   0     0   0 255
255 255   0   255 255 255     0   0   0
```

Therefore, the first pixel has an RGB value of (255, 0, 0), which is literally red. Second pixel has an RGB value of (0, 255, 0), which is literally green. And so forth. Looking back at our image above, that is pretty accurate.

Here are some common colours you should know on the top of your head:

Name	Red	Green	Blue	Preview
Black	0	0	0	
Gray	127	127	127	
Red	255	0	0	
Green	0	255	0	
Blue	0	0	255	
Yellow	255	255	0	
Magenta	255	0	255	
Cyan	0	255	255	
White	255	255	255	

In relation to the image:



Name	Red	Green	Blue	Preview
Red	255	0	0	
Green	0	255	0	

Blue	0	0	255	
Yellow	255	255	0	
White	255	255	255	
Black	0	0	0	

These are very simple colours and I am barely breaking the ice with this one. However, you should be getting the concept by now.

Text Format vs Binary Format?

Up to this point, I have been generous to you, showing you examples with a very simple image file. However, realistically, computers read things differently than humans do. They like binary compared to numbers/text. Let's look at the file above one more time:

```
P3
3 2
255
255   0   0     0 255   0     0   0 255
255 255   0   255 255 255     0   0   0
```

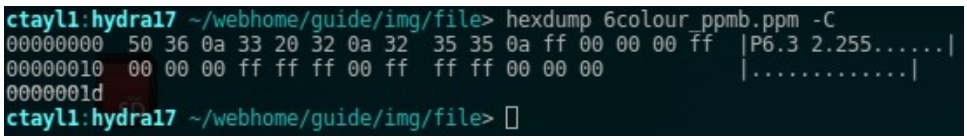
This is the text format. Notice how the numbers after the "255" colour range identifier are human readable numbers. What determines whether or not a program should read a file like this as binary or text is its magic number at the very top of the file. Here is a table of how the magic number works for PBM, PGM, and PPM files (Highlighted Yellow the type of the file we talked about up to this point):

Magic Number	File Type	Extension	Type
P1	Portable BitMap	PBM	ASCII
P2	Portable GrayMap	PGM	ASCII
P3	Portable PixMap	PPM	ASCII
P4	Portable BitMap	PBM	Binary

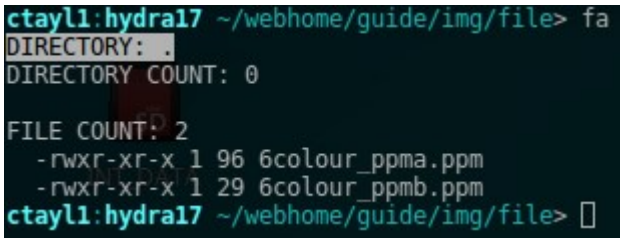
P5	Portable GrayMap	PGM	Binary
P6	Portable PixMap	PPM	Binary
P7	Portable ArbitraryMap	PAM	Unknown

"So Clara, what does a binary PPM file look like?"

Glad you asked. Chances are that your web browser can't even support it being shown here... so I'll just show you a hex dump screenshot.



As you can see, the first 3 lines are readable text (line break = 0x0A). Notice the magic number is "P6". This isn't that bad either to be honest. Since the file is binary, we can store information on a colour in 3 bytes as opposed to the text format which stored a byte for a "single number". Horrible. As a result, binary files are much smaller than their ASCII counterparts. Don't believe me?



96 bytes for ASCII vs 29 bytes for Binary. And they both store the exact same information.

Links	Resources
Home	Professor Marz's page
UTK Home	cplusplus.com
Web_FS	My Github