An introduction to Programming

Guido Klingbeil
Centre for Mathematical Biology
University of Oxford

Worksheet 13

Binary file input / output
Tuesday the 23rd of October 2012

In this exercise you will work with binary data and files containing such data. In principal, any file which is not a text file may be called a binary file. The smallest entity in a text file is a character. Historically, a single character has the length of a byte or 8-bits. Several characters are then structured into lines which are separated by line ends.

A binary file lacks this structure. While in a binary the bits are still grouped into bytes, each bit or arbitrarily long sequences of bits may represents a data entry. This means that you have to know the internal data layout of a binary file.

Many different binary file layouts exist such as JPEG images or the well known Microsoft Office data files. Often, as it is the case for the Microsoft Office data files, the internal structure is secret to a company.

This exercise sheet shall make you familiar with handling binary data and how to use it in your own programs.

Essential:

1. Download the file Lesson13.tar.gz from weblearn and decompress it into your home directory. You will find three directories. One for the essential tasks, one for the optional tasks, and one for the advanced tasks.
2. Run the provided Perl code “read_ppm_p6.pl” to read the colour binary encoded PPM file (P6) “ppm_bin_example_p6.ppm” and understand the code.

3. Extend the provided code to store the data of the three colour channels into 3 distinct suitably sized 2-dimensional arrays.

4. Print only the green colour channel either to screen or write it into a file. If your code works, run it using the larger P6 image “microscopy.ppm”.

5. Repeat the same task in C.

   Keep in mind that the data portion of a PPM file is organised in triplets. For each pixels its red , green, and blue value is stored in a byte of 8 bits. Just as a reminder, the header portion of a PPM file looks like this:

   ```
   P6
   ```

```
# The colour values are in binary, then 3 columns and 2 rows of pixels,
# then 255 for max color, then the RGB (red , green, blue) triplets
3 2
255
```

6. Parse binary data into characters:
   The file “binary_data.txt” contains a sequence of 0s and 1s. Each of them represents a single “bit” and you should consider it as binary data. This sequence of “bits” encodes some ASCII text and 8 “bits” encode for a single character. Your task is to write a program (in your language of choice) converting the sequence of “bits” into a sequence of characters.

   E.g. given the bit string in binary 01001111 (which represents the character 'O'). This can be expressed as:
   0*2^7+1*2^6 +0*2^5+0*2^4+1*2^3+1*2^2+1*2^1+1*2^0=64+8+4+2+1=79
   where 79 is the ASCII encoding for the character 'O'.

   Please keep in mind that the ASCII table only requires 7 bits per character and that the decimal number 0 does not encode the character '0'.

   This is a copy of the ASCII encoding table (a table giving you the decimal representation can be found at http://en.wikipedia.org/wiki/ASCII):

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Optional:

7. Convert a binary encoded PBM (P4) file into an ASCII encoded PBM (P1) file:
   Write a program that reads the the binary encoded file “feep2b.pbm” and converts it into an ASCII encoded file.
   While this exercise is similar to the previous one now 8 bits are now bundled into a byte and you have to extract the single bits form it in order to write them into an ASCII encoded PBM (P1) file. One possibility is to use the algorithm given in the lecture to convert the a decimal number into a binary one.

   You find stubs giving you the code to read and write the file line by line in the files “binary2ascii.c” and “binary2ascii.pl”.

To make sure your code works as expected try convert the file “jellyfish.pbm”. What can you say about the different file

Advanced:

8. Search a non-aligned bit pattern in a binary file:
   Open the file “Feep_binary.bpm” in a hex editor and change a byte sequence at a position of your choice to the values e7 9e 1B 9E 78 and save the file.

   Write a programme that reads your modified file and searches for the bit pattern 1110 0111 1001 1110 . Please be aware that in a binary file the bit pattern does not necessarily start at the beginning of a byte. Your programme should not consider overlaps.

   Since this is a quite advanced task, you get the solution in the file “search_binary.c”.

End of worksheet 13.