CSC209H Worksheet: Binary I/O and Binary Files

Consider the following C statement: `char ch = 'A';` One byte of memory is allocated for the variable `ch` and in the first row of table below we see that the bits are set to `01000001`. This is the 65, the ASCII code for the letter 'A'. (Aside: We don't expect you to memorize any ASCII codes. You can look them up in this table http://www.asciicode.com.)**Hint for the table:** the ASCII code for '5' is 53, in decimal.
Notice in the second row that when we use `fprintf` to print a string containing the uppercase A to a file, one byte is written to the file. Again it is the ASCII code for `A`.

Complete the rest of the table showing for each statement what is saved in memory and what is written to the file. (When you get to the point of storing integers don't stress about the actual 2's complement details that you might be learning in 258 – just make sure you know the difference between storing the sizeof(int) bytes in binary vs. the ASCII representation for each digit.)

What's written to the file? (show the bits)	**Code**	**What's in the Memory?** (show the bits)
N/A	`char ch = 'A';`	0 1 0 0 0 0 0 1
0 1 0 0 0 0 0 1	`fprintf(fp, "A");`	N/A
	`char ch = 'A';` `fprintf(fp, "%c", ch);`	
	`fprintf(fp, "5");`	
	`char ch = '5';` `fprintf(fp, "%c", ch);`	
	`int i = 5;`	
	`int i = 5;` `fprintf(fp, "%d", i);`	
	`int i = 5;` `fwrite(&i, sizeof(int), 1, fp);`	
	`char ch = '5';` `fwrite(&ch, sizeof(char), 1, fp);`	

Flip over the page for another question.

CSC209H Worksheet: Binary I/O and Binary Files

You should have noticed that more bytes are written for the int i than just a single character. Is this always the case? Identify (with justification) the range of integers i such that:

(a) more bytes are required to store i in binary – using `fwrite(&i, sizeof(int), 1, fp)` – than in text – using `fprintf(fp, "%d", i)`.

(b) the same number of bytes are required to store i in binary and in text.

(c) fewer bytes are required to store i in binary than in text.

Solutions

In binary, each integer always takes `sizeof(int)` bytes. So if we assume that `sizeof(int)` is 4, then values that take exactly 4 characters will use the same number of bytes. Positive number X can be represented in exactly 4 characters if $1000 < X < 9999$. To represent a negative number in text, we use one char for the negative sign. So negative X takes 4 characters if $-999 < X < -100$
The range of X where X takes 3 or refer characters is $-99 < X < 999$. These values take more bytes to store in binary than in text.
All other numbers ($X < -999$ or $X > 9999$) take more than 4 characters so take fewer bytes to stop in binary than n text.
Solutions for first question are on the next page.

CSC209H Worksheet: Binary I/O and Binary Files

SOLUTIONS

What's written to the file? (show the bits)	**Code**	**What's in the Memory?** (show the bits)
	`char ch = 'A';`	0 1 0 0 0 0 0 1
0 1 0 0 0 0 0 1	`fprintf(fp,"A");`	
0 1 0 0 0 0 0 1	`char ch = 'A';` `fprintf(fp,"%c", ch);`	0 1 0 0 0 0 0 1
0 0 1 1 0 1 0 1	`fprintf(fp,"5");`	
0 0 1 1 0 1 0 1	`char ch = '5';` `fprintf(fp,"%c", ch);`	0 0 1 1 0 1 0 1
	`int i = 5;`	00000000 00000000 00000000 00000101
0 0 1 1 0 1 0 1	`int i = 5;` `fprintf(fp,"%d", i);`	00000000 00000000 00000000 00000101
00000000 00000000 00000000 00000101	`int i = 5;` `fwrite(&i, sizeof(int), 1,fp);`	00000000 00000000 00000000 00000101
0 0 1 1 0 1 0 1	`char ch = '5';` `fwrite(&ch, sizeof(char), 1, fp);`	0 0 1 1 0 1 0 1