Example 1

We wish to five 0 bytes (alias null characters, alias '0' characters) to a file named data. That is, we wish to write these five bytes to the file:

\[00000000 \ 00000000 \ 00000000 \ 00000000 \ 00000000\]

Open the File
FILE *psFile;
psFile = fopen("data", "w");

Attempt 1 (Incorrect)
fprintf(psFile, "00000"); /* Writes 00110000 00110000 00110000 00110000 00110000 */

Attempt 2 (Incorrect)
for (i = 0; i < 5; i++)
    fprintf(psFile, "%c", '0'); /* Writes 00110000*/

Attempt 3 (Incorrect)
for (i = 0; i < 5; i++)
    putc('0', psFile); /* Writes 00110000 */

Attempt 4 (Correct)
for (i = 0; i < 5; i++)
    fprintf(psFile, "%c", '\0'); /* Writes 00000000*/

Attempt 5 (Correct)
for (i = 0; i < 5; i++)
    fprintf(psFile, "%c", 0); /* Writes 00000000*/

Attempt 6 (Correct)
for (i = 0; i < 5; i++)
    fprintf(psFile, "%c", 0x00); /* Writes 00000000 */

Attempt 7 (Correct)
for (i = 0; i < 5; i++)
    putc('\0', psFile); /* Writes 00000000 */

Attempt 8 (Correct)
for (i = 0; i < 5; i++)
    putc(0, psFile); /* Writes 00000000 */

Attempt 9 (Correct)
for (i = 0; i < 5; i++)
    putc(0x00, psFile); /* Writes 00000000 */

Close the File
fclose(psFile);
Example 2

We wish to write the unsigned long 0x0123456789abcdef to a file named data as it would appear in memory as an eight-byte entity. As humans, we would express the unsigned long 0x0123456789abcdef in binary like this:

```
00000001 00100011 01100011 10001001 10101011 11001101 11101111
most sig byte
least sig byte
```

But ARMv8 is a little-endian architecture. In the memory of a little-endian computer, the least significant byte of an integer is in the lowest memory location. So the unsigned long 0x0123456789abcdef appears in memory like this:

```
11101111 11001101 10101011 10001001 01100111 01000101 00100011 00000001
least sig byte
most sig byte
```

Or, more precisely, like this:

```
pretend address
1000 11101111 least sig byte
1001 11001101
1002 10101011
1003 10001001
1004 01100111
1005 01000101
1006 00100011
1007 00000001 most sig byte
```

Open the File
FILE *psFile;
psFile = fopen("data", "w");

Attempt 1 (Incorrect)
fprintf(psFile, "%x", 0x0123456789abcdef);
/* Writes 00110000 00110001 00110010 00110011 00110100 00110101 00110110 00110111 00111000 00111001 01100001 01100010 01100011 01100100 01100101 01100110 */

Attempt 2 (Incorrect)
fprintf(psFile, "%x", 0x0123456789abcdef);
/* Writes 00110000 00110001 00110010 00110011 00110100 00110101 00110110 00110111 00111000 00111001 01100001 01100010 01100011 01100100 01100101 01100110 */

Attempt 3 (Correct)
fprintf(psFile, "%c", 0xef); /* Writes 11101111 */
fprintf(psFile, "%c", 0xcd); /* Writes 11001101 */
fprintf(psFile, "%c", 0xab); /* Writes 10101011 */
fprintf(psFile, "%c", 0x89); /* Writes 10001001 */
fprintf(psFile, "%c", 0x67); /* Writes 01100111 */
fprintf(psFile, "%c", 0x45); /* Writes 01000101 */
fprintf(psFile, "%c", 0x23); /* Writes 00100011 */
fprintf(psFile, "%c", 0x01); /* Writes 00000001 */

Attempt 4 (Correct)
putc(0xef, psFile); /* Writes 11101111 */
putc(0xcd, psFile); /* Writes 11001101 */
putc(0xab, psFile); /* Writes 10101011 */
putc(0x89, psFile); /* Writes 10001001 */
putc(0x67, psFile); /* Writes 01100111 */
putc(0x45, psFile); /* Writes 01000101 */
putc(0x23, psFile); /* Writes 00100011 */
putc(0x01, psFile); /* Writes 00000001 */

Attempt 5 (Correct) <-- the easiest approach
unsigned long uiData;
...
Example 3

We wish to write the unsigned int 0x01234567 to a file named data as it would appear in memory as a four-byte entity. As humans, we would express the unsigned int 0x01234567 in binary like this:

```
00000001 00100011 01000101 01100111
most sig  least sig
byte      byte
```

But ARMv8 is a little-endian architecture. In the memory of a little-endian computer, the least significant byte of an integer is in the lowest memory location. So the unsigned int 0x01234567 appears in memory like this:

```
01100111 01000101 00100011 00000001
least sig                         most sig
byte                              byte
```

Or, more precisely, like this:

```
pretend
address
  1000 01100111 least sig byte
  1001 01000101
  1002 00100011
  1003 00000001 most sig byte
```

Open the File
```
FILE *psFile;
psFile = fopen("data", "w");
```

Attempt 1 (Incorrect)
```
fprintf(psFile, "%d", 0x01234567);
/* Writes 01100111 01000101 00100011 00000001 */
```

Attempt 2 (Incorrect)
```
fprintf(psFile, "%x", 0x01234567);
/* Writes 01100111 01000101 00100011 00000001 */
```

Attempt 3 (Correct)
```
fprintf(psFile, "%c", 0x67);  /* Writes 01100111 */
fprintf(psFile, "%c", 0x45);  /* Writes 01000101 */
fprintf(psFile, "%c", 0x23);  /* Writes 00100011 */
fprintf(psFile, "%c", 0x01);  /* Writes 00000001 */
```

Attempt 4 (Correct)
```
putc(0x67, psFile);  /* Writes 01100111 */
putc(0x45, psFile);  /* Writes 01000101 */
putc(0x23, psFile);  /* Writes 00100011 */
putc(0x01, psFile);  /* Writes 00000001 */
```

Attempt 5 (Correct)  <-- the easiest approach
```
unsigned int uiData;
...
uiData = 0x01234567;
fwrite(&uiData, sizeof(unsigned int), 1, psFile);
/* Writes 01100111 01000101 00100011 00000001 */
```

Close the File
```
fclose(psFile);
```