

Forum:[Hack Forums](#)

Thread: [Endian read/write](#)

zNinja - April 5, 2020, 5:34 am

Hi, I am trying to figure out if I am doing something wrong by swapping bits of floats to big endian and back to little endian on little endian machine.

I am able to convert float and double to bit endian, write to file, read from file and convert back to little endian on my project.

I tried to use nodejs to check correctness of my implementation.

I get right conversion of double from file, but float precision is incorrect.

C++ test

PHP Code

```
<?php
#include <iostream>
#include <fstream>
#include <stdint.h>
#include <cassert>
#define NW_LITTLE_ENDIAN
uint32_t nw_swap_uint32( uint32_t a_val);
uint64_t nw_swap_uint64( uint64_t a_val);
```

```
float nw_swap_float(float a_val);
double nw_swap_double(double a_val);
```

```
float nw_readFloatLE( float a_val );
float nw_writeFloatLE( float a_val );
float nw_readFloatBE( float a_val );
float nw_writeFloatBE( float a_val );
```

```
double nw_readDoubleLE( double a_val );
double nw_writeDoubleLE( double a_val );
double nw_readDoubleBE( double a_val );
double nw_writeDoubleBE( double a_val );
```

```
int main()
{
    float l_test_f = 5.1;
    double l_test_d = 5.1;
    std::ofstream l_ofs;
    std::ifstream l_ifs;
    float l_a;
    double l_b;
    float l_f = nw_writeFloatBE(5.1);
    double l_d = nw_writeDoubleBE(5.1);

    l_ofs.open ("test.txt", std::ofstream::out | std::ofstream::binary);
    l_ofs.write(reinterpret_cast<const char*>(&l_f),sizeof(l_f));
```

```

l_ofs.write(reinterpret_cast<const char*>(&l_d),sizeof(l_d));
l_ofs.close();

l_ifs.open ("test.txt", std::ifstream::in | std::ifstream::binary);
l_ifs.read(reinterpret_cast<char*>(&l_a), sizeof(l_a));
l_ifs.read(reinterpret_cast<char*>(&l_b), sizeof(l_b));
l_f = nw_readFloatBE(l_a);
l_d = nw_readDoubleBE(l_b);
l_ifs.close();

std::cout << l_f << " : " << l_d << std::endl;

assert(l_test_f == l_f);
assert(l_test_d == l_d);
return 0;
}
uint32_t nw_swap_uint32( uint32_t a_val )
{
    a_val = ((a_val << 8) & 0xFF00FF00 ) | ((a_val >> 8) & 0xFF00FF );
    return (a_val << 16) | (a_val >> 16);
}

uint64_t nw_swap_uint64( uint64_t a_val )
{
    a_val = ((a_val << 8) & 0xFF00FF00FF00FF00ULL ) | ((a_val >> 8) &
0x00FF00FF00FF00FFULL );
    a_val = ((a_val << 16) & 0xFFFF0000FFFF0000ULL ) | ((a_val >> 16) &
0x0000FFFF0000FFFFULL );
    return (a_val << 32) | (a_val >> 32);
}

float nw_swap_float(float x)
{
    union
    {
        float f;
        int32_t ui32;
    } swapper;
    swapper.f = x;
    swapper.ui32 = nw_swap_uint32(swapper.ui32);
    return swapper.f;
}

float nw_readFloatLE( float a_val )
{
#ifdef NW_LITTLE_ENDIAN
    return a_val;
#else
    return nw_swap_float(a_val);
#endif
}

```

```
#endif
}  
float nw_writeFloatLE( float a_val )  
{  
#ifdef NW_LITTLE_ENDIAN  
    return a_val;  
#else  
    return nw_swap_float(a_val);  
#endif  
}
```

```
float nw_readFloatBE( float a_val )  
{  
#ifdef NW_LITTLE_ENDIAN  
    return nw_swap_float(a_val);  
#else  
    return a_val;  
#endif  
}
```

```
float nw_writeFloatBE( float a_val )  
{  
#ifdef NW_LITTLE_ENDIAN  
    return nw_swap_float(a_val);  
#else  
    return a_val;  
#endif  
}
```

```
double nw_swap_double(double x)  
{  
    union  
    {  
        double f;  
        uint64_t ui64;  
    } swapper;  
    swapper.f = x;  
    swapper.ui64 = nw_swap_uint64(swapper.ui64);  
    return swapper.f;  
}
```

```
double nw_readDoubleLE( double a_val )  
{  
#ifdef NW_LITTLE_ENDIAN  
    return a_val;  
#else  
    return nw_swap_double(a_val);  
}
```

```

#endif
}
double nw_writeDoubleLE( double a_val )
{
#ifdef NW_LITTLE_ENDIAN
    return a_val;
#else
    return nw_swap_double(a_val);
#endif
}

```

```

double nw_readDoubleBE( double a_val )
{
#ifdef NW_LITTLE_ENDIAN
    return nw_swap_double(a_val);
#else
    return a_val;
#endif
}

```

```

double nw_writeDoubleBE( double a_val )
{
#ifdef NW_LITTLE_ENDIAN
    return nw_swap_double(a_val);
#else
    return a_val;
#endif
}

```

c++ test output

PHP Code

```
<?php
```

5.1 : 5.1

Nodejs test

PHP Code

```
<?php
```

```

var fs = require('fs'),
    path = require('path'),
    filePath = path.join(__dirname, 'test.txt');

```

```

fs.readFile(filePath, function(err,data){
    if (!err) {
        var buf = Buffer.from(data);
        var float_valBE = buf.readFloatBE(0);
        var double_valBE = buf.readDoubleBE(4);

```

```

console.log("float_valBE ",float_valBE,(float_valBE === 5.1));
console.log("double_valBE ",double_valBE,(double_valBE === 5.1));

```

```

    } else {
        console.log(err);

```

```
}  
});
```

Nodejs output

PHP Code

```
<?php
```

```
float_valBE 5.099999904632568 false
```

```
double_valBE 5.1 true
```

I need help to figure out is my implementation wrong or nodejs treats floats as other type/size?

EDIT:

Just figured out with this test.

Nodejs does not read floats as expected.

PHP Code

```
<?php
```

```
var fs = require('fs'),  
    path = require('path'),  
    filePath = path.join(__dirname, 'test.txt');
```

```
const buf = Buffer.allocUnsafe(12);  
var f = 5.1;  
buf.writeFloatBE(f,0);  
buf.writeDoubleBE(f,4);  
fs.writeFile(filePath, buf, function(err) {  
    if(err) {  
        return console.log(err);  
    }  
    fs.readFile(filePath, function(err,data){  
        if (!err) {  
            var buf = Buffer.from(data);  
            var float_valBE = buf.readFloatBE(0);  
            var double_valBE = buf.readDoubleBE(4);  
  
            console.log("float_valBE ",float_valBE,(float_valBE == f));  
            console.log("double_valBE ",double_valBE,(double_valBE == f));  
  
        } else {  
            console.log(err);  
        }  
    }  
});  
  
});
```

saltycracker - April 5, 2020, 6:30 am

Curious about these two lines...

```
Code float l_f = nw_writeFloatBE(5.1);  
double l_d = nw_writeDoubleBE(5.1);  
You spend the time creating two variables here:  
Code float l_test_f = 5.1;  
double l_test_d = 5.1;
```

And then use literals in the function calls.. I know. Picky.

Good work BTW.. Its nice to see something that C++ was designed to handle.

zNinja - April 5, 2020, 9:11 am

(Apr 5, 2020 06:30 AM)saltcracker Wrote: [\[CITE\]](#) Curious about these two lines...

```
Code float l_f = nw_writeFloatBE(5.1);  
double l_d = nw_writeDoubleBE(5.1);  
You spend the time creating two variables here:  
Code float l_test_f = 5.1;  
double l_test_d = 5.1;
```

And then use literals in the function calls.. I know. Picky.I have edited the code multiple times in short period to eventually pass some test standards and show results.

(Apr 5, 2020 06:30 AM)saltcracker Wrote: [\[CITE\]](#) Good work BTW.. Its nice to see something that C++ was designed to handle.Thanks.

Mysticism - April 5, 2020, 1:24 pm

Floating point arithmetic can sometimes be tricky and a pain in the ass so this is well-common