情報通信システム論　レポート課題

学籍番号: 1240293
氏名: 植田蓮

2023 年 10 月 16 日

概要

この文書は 2023 年度の情報通信システム論レポート課題である．“現在利用している環境での byte order を判別するプログラムを C 言語で作成せよ．” という課題で作成したプログラムの説明，実行例，考察，感想を記述する．

1　課題の要件

作成するプログラムの要件は以下を満たすことである．

- ライブラリーを用いて判別しない
- 判別には union を利用する
- キャスト (型変換) は使用しない

2　プログラムの説明

作成したプログラムを示す．なおヘッダファイルの読み込みは省略している

```
----------------------main.c-----------------
#define IP_ADDRESS 0xC0A80A01 // 192.168.10.1

union u {
    unsigned int ipaddr;
    unsigned char doted_ipaddr[4];
};

int main(void) {
union u u1;
    u1.ipaddr = IP_ADDRESS;
    if(u1.doted_ipaddr[0] == 0xC0) {
        printf("Byte order of your environment is Big Endian\n");
    } else if(u1.doted_ipaddr[0] == 0x01) {
        printf("Byte order of your environment is Little Endian\n");
    }
}
-----------------------------------------------
```

u はメンバ変数として，unsigned int 型の ipaddr と，unsigned char 型の doted_ipaddr を持つ．u1

の ipaddr には IP アドレス 192.168.10.1 に対応する 0xC0A80A01 を代入している．u は共用体であるから，ipaddr と doted_ipaddr のメモリ領域は重複している．バイトオーダーがビッグエンディアンであれば，MSB からメモリに格納されるから doted_ipaddr+0 番地には MSB が格納される．すなわち，0xC0A80A01 の 0xC0 が格納される．一方で，リトルエンディアンであれば LSB からメモリに格納されるから doted_ipaddr+0 番地には 0xC0A80A01 の 0x01 が格納される．プログラム中ではこれを用いて，u1.doted_ipaddr[0] が 0xc0 であれば “Byte order of your environment is Big Endian” を，u1.doted_ipaddr[0] が 0x01 であれば “Byte order of your environment is Little Endian” を標準出力で出力している．

3 プログラムの実行例

今回作成したプログラムを 2.3 GHz クアッドコア Intel Core i5 で実行したところ，“Byte order of your environment is Little Endian” が出力された．また，AWS EC2 インスンスの t2.micro にて実行したところ，同じく “Byte order of your environment is Little Endian” が出力された．

4 考察

BigEndian とリトルエンディアンについて．今回試した 2 つの環境ではリトルエンディアンが採用されていた．これは，どちらも Intel の (影響を受けた) プロセッサーを採用しているからだと思われる．一方で，IP におけるバイトオーダーはビッグエンディアンである．すなわち，IP を用いた通信を行う場合には，リトルエンディアンをビッグエンディアンにして送信した後に，受信側で再度リトルエンディアンに復元している．これは，非常に無駄な処理であると考えられる．1 度の処理はそこまで大きなものではないかもしれないが，サーバが送受信する，単位時間あたりの IP パケットは膨大であり，それらのパケット全てに対してそのような処理をすると，計算資源の無駄が起きている．

CPU にはビッグエンディアンとリトルエンディアンが存在するが，これらの違いについて調べたところ，位取り記数法で考えた時に，リトルエンディアンの方がシンプルに考えられるという意見があった．そこで以下のようなプログラムを作成した．ただし，ヘッダファイルの読み込みは省略している．

```
-----------------------------------------------
#define IP_ADDRESS 0xC0A80A01 // 192.168.10.1

union u {
    unsigned int ipaddr;
    unsigned char doted_ipaddr[4];
};

int main(void) {
union u u1;
    u1.ipaddr = IP_ADDRESS; // 192.168.10.1

// リトルエンデアンの int を 1byte ずつアクセスして取得する
    unsigned int sum1 = 0;
int i;
    for(i = 0; i < sizeof(u1.doted_ipaddr); i++) {
        sum1 += (unsigned int)(u1.doted_ipaddr[i] * pow(256, i));
    }
    printf("sum1 = %x\n", sum1);

    // ビッグエンディアンにする
```

```
    union u u2;
    u2.ipaddr = htonl(0xC0A80a01); // 192.168.10.1

    // ビッグエンディアンの int を 1byte ずつアクセスして出力する
    unsigned int sum2 = 0;
    for(i = 0; i < sizeof(u2.doted_ipaddr); i++) {
        sum2 += (unsigned int)(u2.doted_ipaddr[i] * pow(256, sizeof(u2.doted_ipaddr) - i));
    }
    printf("sum2 = %x\n", sum2);
}
-----------------------------------------------
```

リトルエンディアンは，LSB から格納する方法で，位取り記数法を用いて考える人間にとってはあまり直感的ではないが，ソースコード上では，“pow(256, i)” となっており，直感的に感じる．

ビッグエンディアンは，MSB から格納し，人間にとって直感的に感じる一方で，“pow(256, sizeof(u2.doted_ipaddr) - i)” となっており，ソースコード上ではあまり直感的ではない．

このように，たしかに差はあるものの計算機上では MSB と LSB どちらから格納するかは本室ではないし，どちらであっても性能は変わらないだろうと考える．ビッグエンディアンとリトルエンディアンの優劣ではなく，OS や通信プロトコル，組み込みソフトの設計時にには，2 種類のエンディアンがあるということを意識して設計することが重要である．

5 感想

CPU のビッグエンディアン，リトルエンディアンの優劣について考えてみたが，自分としてはどちらも同じだろうという考えになった．一方で，2 種類の方式が存在することは，低いレイヤーのプログラミングをしている人いにとっては悩ましい問題ではないかとも感じた．ビッグエンディアンとリトルエンディアンそれぞれにソフトウェアを対応させなければいけないのは大きなコストがかかるのではないかと思う．どちらかに統一できないのかとも感じるが，この辺りは CPU メーカーのビジネスマーケテイングに大きく関わっていると思うと，こんなんだろうと感じる．プロトコルも同様に，ビジネスが絡むと，お粗末なプロトコルがあたりまえに使われていたりして愚かだと感じる．

授業で，ネットワークバイトオーダーについての話があったので少しインターネットで調べてみたところ，ビッグエンディアンを明記しているわけではないらしいということがわかった．しかし，IP について規定している，RFC791[1] の p.39 において，実質的に IP がビッグエンディアンを採用していることを記しており，ネットワークバイトオーダーはビッグエンディアンとなっているようである．ビッグエンディアンを用いなければならないと明記されておらず，いい加減であると感じもしたが，RFC らしいとも感じた．

参考文献

[1] IETF, “RFC791”, https://datatracker.ietf.org/doc/html/rfc791, 2023/10/15 閲覧