java.nio.* クラスを使うと、かゆいところに少しだけ手が届くようになります。
... 実は昔書いた実験コードを整理したくて wiki にのせただけだったり

ByteBuffer

こんがらがったとき用
ByteBuffer の wrap と merge の戻り値と元の配列に関するテスト
というか、ふつうにポインタなんだな〜と思えば大丈夫だったり

■実験コード

```
import java.nio.*;
import net.wasamon.mjlib.print.*;

public class Test{

  /**
   * 二つのバイト列をマージし、返す
   * @param dest マージしたデータを格納する配列
   * @param src1 ソースバイト列
   * @param src2 ソースバイト列
   * @return マージ後の配列への参照
   */
  public static byte[] merge(byte dest[], int offset, byte src1[], byte src2[]){
    ByteBuffer buf = ByteBuffer.wrap(dest);
    buf.position(offset);
    buf.put(ByteBuffer.wrap(src1));
    buf.put(ByteBuffer.wrap(src2));
    return dest;
  }

  public static byte[] merge(byte dest[], byte src1[], byte src2[]){
    return merge(dest, 0, src1, src2);
  }

  /**
   * ネットワークオーダの値を long 型の値に変換する
   * @param data もとのバイト列
   * @param offset オフセット
   * @return long の値
   */
  public static long ntoh_long(byte data[], int offset){
    return  ((((long)data[offset+0]) << 56 ) & 0x00ff000000000000L)
          +((((long)data[offset+1]) << 48 ) & 0x0000ff0000000000L)
          +((((long)data[offset+2]) << 40 ) & 0x000000ff00000000L)
          +((((long)data[offset+3]) << 32 ) & 0x00000000ff000000L)
          +((((long)data[offset+4]) << 24 ) & 0x0000000000ff0000L)
          +((((long)data[offset+5]) << 16 ) & 0x000000000000ff00L)
          +((((long)data[offset+6]) << 8  ) & 0x0000000000000ff0L)
          +((((long)data[offset+7])       ) & 0x000000000000000ffL);
  }

  public static void dump(byte[] data){
    int i = 0;
    for(i = 0 ; i < data.length; ++i){
      System.out.print(PrintFormat.print("%02x", (byte)(data[i]&0xff)));
      if(i%16 == 15){
        System.out.println();
      }else{
        System.out.print(" ");
      }
    }
    if((i % 16) != 0){
      System.out.println();
    }
  }

  public static void main(String args[]){
```

```
    byte[] b = new byte[16];

    for(int i = 0; i < b.length; i++){
      b[i] = (byte)(i & 0xff);
    }


    System.out.println("ByteBuffer.wrap(b).getLong( ) = " + ByteBuffer.wrap(b).getLong());
    System.out.println("ntoh_long(b, 0)               = " + ntoh_long(b, 0));

    System.out.println("ByteBuffer.wrap(b).getLong(2) = " + ByteBuffer.wrap(b).getLong(2));
    System.out.println("ntoh_long(b, 2)               = " + ntoh_long(b, 2));

    byte[] x = new byte[8];
    byte[] y = new byte[4];
    byte[] z = new byte[2];

    for(int i = 0; i < x.length; i++){
      x[i] = (byte)(i&0xff);
    }
    for(int i = 0; i < y.length; i++){
      y[i] = (byte)((i+10)&0xff);
    }
    for(int i = 0; i < z.length; i++){
      z[i] = (byte)(-1*(i+1));
    }

   byte[] tmp = null;

   System.out.print("x  : "); dump(x);
   tmp = merge(x, 0, y, z);
   System.out.print("x  : "); dump(x);
   System.out.print("rtn: "); dump(tmp);
   tmp = merge(x, 1, y, z);
   System.out.print("x  : "); dump(x);
   System.out.print("rtn: "); dump(tmp);

  }

}
```

■実行結果

```
ByteBuffer.wrap(b).getLong( ) = 283686952306183
ntoh_long(b, 0)               = 283686952306183
ByteBuffer.wrap(b).getLong(2) = 144964032628459529
ntoh_long(b, 2)               = 144964032628459529
x  : 00 01 02 03 04 05 06 07
x  : 0a 0b 0c 0d ff fe 06 07
rtn: 0a 0b 0c 0d ff fe 06 07
x  : 0a 0a 0b 0c 0d ff fe 07
rtn: 0a 0a 0b 0c 0d ff fe 07
```