in include/linux/nls.h

```
struct nls_table {
    const char *charset;
    const char *alias;
    int (*uni2char) (wchar_t uni, unsigned char *out, int boundlen);
    int (*char2uni) (const unsigned char *rawstring, int boundlen,
             wchar_t *uni);
    const unsigned char *charset2lower;
    const unsigned char *charset2upper;
    struct module *owner;
    struct nls_table *next;
};
```

field	description	example
charset	该charset的name	"cp936"
alias	该charset的别名	"gb2312", 这里cp936与gb2312是同一回事
uni2char	uni表示Unicode(可能是unicode_t，可能是wchar_t)	
char2uni	char是某种编码的charset	
charset2lower	charset ==> 小写	不是凡是字符都有大小写之分的
charset2upper	charset ==> 大写	不是凡是字符都有大小写之分的
next	所有nls module都可以是ko,都链接在list中	

```
/* Plane-0 Unicode character */
typedef u16 wchar_t;
#define MAX_WCHAR_T 0xffff

/* Arbitrary Unicode character */
typedef u32 unicode_t;
```

2-byte的wchar_t只能表示plane 0的unicode character.
unicode用4-byte表示。
而整个unicode的code points (2 ^ 32)被分为2 ^ 16 (65536)个planes.
所以wchar_t只能表示plane 0的unicode code points.

每个plane又可以分为page

一个page = 2 ^ 8 (256)个code points,2 ^ 8 (256)个page组成一个plane.

ASCII码位于unicode character中的plane 0, page 0.

```
1.  static int uni2char(wchar_t uni, unsigned char *out, int boundlen)
2.  {
3.      const unsigned char *uni2charset;
4.      unsigned char cl = uni & 0x00ff;              ①
5.  unsigned char ch = (uni & 0xff00) >> 8;       ②
6.
7.      if (boundlen <= 0)
8.          return -ENAMETOOLONG;
9.
10.     uni2charset = page_uni2charset[ch];           ③
11.     if (uni2charset && uni2charset[cl])           ④
12.         out[0] = uni2charset[cl];
13.     else
14.         return -EINVAL;
15.     return 1;
16. }
```

把unicode space中的plane 0的character转化成ascii character!
只有位于page 0的character才可以转换成ascii。

①
cl = page X中的code point
②
ch = page number
③

```
static const unsigned char page00[256] = {
    0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, /* 0x00-0x07 */
    0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, /* 0x08-0x0f */
    0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, /* 0x10-0x17 */
    0x18, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x1e, 0x1f, /* 0x18-0x1f */
    0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, /* 0x20-0x27 */
    0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x2d, 0x2e, 0x2f, /* 0x28-0x2f */
    0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, /* 0x30-0x37 */
    0x38, 0x39, 0x3a, 0x3b, 0x3c, 0x3d, 0x3e, 0x3f, /* 0x38-0x3f */
    0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, /* 0x40-0x47 */
    0x48, 0x49, 0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f, /* 0x48-0x4f */
    0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, /* 0x50-0x57 */
    0x58, 0x59, 0x5a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f, /* 0x58-0x5f */
    0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, /* 0x60-0x67 */
    0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f, /* 0x68-0x6f */
    0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, /* 0x70-0x77 */
    0x78, 0x79, 0x7a, 0x7b, 0x7c, 0x7d, 0x7e, 0x7f, /* 0x78-0x7f */
};

static const unsigned char *const page_uni2charset[256] = {
    page00,
};
```

由于ascii只有page 0才有意义，所以只有(page_uni2charset[0] = page00) != 0

④

uni2charset就是page,如果是ASCII表内的char则不能为0.若为0，表示该plane 0内的unicode code point

不能转换成ASCII，返回0

```
static int char2uni(const unsigned char *rawstring, int boundlen, wchar_t *u
ni)
{
    *uni = charset2uni[*rawstring];
    if (*uni == 0x0000)
        return -EINVAL;
    return 1;
}
```

```
static const wchar_t charset2uni[256] = {
    /* 0x00*/
    0x0000, 0x0001, 0x0002, 0x0003,
    0x0004, 0x0005, 0x0006, 0x0007,
    0x0008, 0x0009, 0x000a, 0x000b,
    0x000c, 0x000d, 0x000e, 0x000f,
    /* 0x10*/
    0x0010, 0x0011, 0x0012, 0x0013,
    0x0014, 0x0015, 0x0016, 0x0017,
    0x0018, 0x0019, 0x001a, 0x001b,
    0x001c, 0x001d, 0x001e, 0x001f,
    /* 0x20*/
    0x0020, 0x0021, 0x0022, 0x0023,
    0x0024, 0x0025, 0x0026, 0x0027,
    0x0028, 0x0029, 0x002a, 0x002b,
    0x002c, 0x002d, 0x002e, 0x002f,
    /* 0x30*/
    0x0030, 0x0031, 0x0032, 0x0033,
    0x0034, 0x0035, 0x0036, 0x0037,
    0x0038, 0x0039, 0x003a, 0x003b,
    0x003c, 0x003d, 0x003e, 0x003f,
    /* 0x40*/
    0x0040, 0x0041, 0x0042, 0x0043,
    0x0044, 0x0045, 0x0046, 0x0047,
    0x0048, 0x0049, 0x004a, 0x004b,
    0x004c, 0x004d, 0x004e, 0x004f,
    /* 0x50*/
    0x0050, 0x0051, 0x0052, 0x0053,
    0x0054, 0x0055, 0x0056, 0x0057,
    0x0058, 0x0059, 0x005a, 0x005b,
    0x005c, 0x005d, 0x005e, 0x005f,
    /* 0x60*/
    0x0060, 0x0061, 0x0062, 0x0063,
    0x0064, 0x0065, 0x0066, 0x0067,
    0x0068, 0x0069, 0x006a, 0x006b,
    0x006c, 0x006d, 0x006e, 0x006f,
    /* 0x70*/
    0x0070, 0x0071, 0x0072, 0x0073,
    0x0074, 0x0075, 0x0076, 0x0077,
    0x0078, 0x0079, 0x007a, 0x007b,
    0x007c, 0x007d, 0x007e, 0x007f,
};
```

ascii char转换成unicode，很简单，高8位为0即可。

```
static const unsigned char charset2lower[256] = {
    0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, /* 0x00-0x07 */
    0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, /* 0x08-0x0f */
    0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, /* 0x10-0x17 */
    0x18, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x1e, 0x1f, /* 0x18-0x1f */
    0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, /* 0x20-0x27 */
    0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x2d, 0x2e, 0x2f, /* 0x28-0x2f */
    0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, /* 0x30-0x37 */
    0x38, 0x39, 0x3a, 0x3b, 0x3c, 0x3d, 0x3e, 0x3f, /* 0x38-0x3f */
    0x40, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, /* 0x40-0x47 */
    0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f, /* 0x48-0x4f */
    0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, /* 0x50-0x57 */
    0x78, 0x79, 0x7a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f, /* 0x58-0x5f */
    0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, /* 0x60-0x67 */
    0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f, /* 0x68-0x6f */
    0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, /* 0x70-0x77 */
    0x78, 0x79, 0x7a, 0x7b, 0x7c, 0x7d, 0x7e, 0x7f, /* 0x78-0x7f */
};
```

以ascii char为index，即可得到对应的low-case char。

只有'A'to 'Z' (0x41 to 0x5A)才有意义，其他不变。

比如，charset2lower['A'] = charset2lower[0x41] = 0x61 = 'a'

```
static const unsigned char charset2upper[256] = {
    0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, /* 0x00-0x07 */
    0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, /* 0x08-0x0f */
    0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, /* 0x10-0x17 */
    0x18, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x1e, 0x1f, /* 0x18-0x1f */
    0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, /* 0x20-0x27 */
    0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x2d, 0x2e, 0x2f, /* 0x28-0x2f */
    0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, /* 0x30-0x37 */
    0x38, 0x39, 0x3a, 0x3b, 0x3c, 0x3d, 0x3e, 0x3f, /* 0x38-0x3f */
    0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, /* 0x40-0x47 */
    0x48, 0x49, 0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f, /* 0x48-0x4f */
    0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, /* 0x50-0x57 */
    0x58, 0x59, 0x5a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f, /* 0x58-0x5f */
    0x60, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, /* 0x60-0x67 */
    0x48, 0x49, 0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f, /* 0x68-0x6f */
    0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, /* 0x70-0x77 */
    0x58, 0x59, 0x5a, 0x7b, 0x7c, 0x7d, 0x7e, 0x7f, /* 0x78-0x7f */
};
```

以ascii char为index，即可得到对应的upper-case char。

只有'a'to 'z' (0x61 to 0x7A)才有意义，其他不变。

比如，charset2upper['a'] = charset2lower[0x61] = 0x41 = 'A'