locale.h

目录

- 简介__
- setlocale()
- localeconv()

简介

`locale.h`是程序的本地化设置，主要影响以下的行为。

- 数字格式
- 货币格式
- 字符集
- 日期和时间格式

它设置了以下几个宏。

- LC_COLLATE: 影响字符串比较函数 `strcoll()`和 `strxfrm()`。
- LC_CTYPE: 影响字符处理函数的行为。
- LC_MONETARY: 影响货币格式。
- LC_NUMERIC: 影响 `printf()`的数字格式。
- LC_TIME: 影响时间格式 `strftime()`和 `wcsftime()`。
- LC_ALL: 将以上所有类别设置为给定的语言环境。

setlocale()

`setlocale()`用来设置当前的地区。

```
char* setlocale(int category, const char* locale);
```

它接受两个参数。第一个参数表示影响范围，如果值为前面五个表示类别的宏之一，则只影响该宏对应的类别，如果值为 `LC_ALL`，则影响所有类别。第二个参数通常只为 `"C"`（正常模式）或 `""`（本地模式）。

任意程序开始时，都隐含下面的调用。

```
setlocale(LC_ALL, "C");
```

下面的语句将格式本地化。

```
setlocale(LC_ALL, "");
```

上面示例中，第二个参数为空字符，表示使用当前环境提供的本地化设置。

理论上，第二个参数也可以设为当前系统支持的某种格式。

```
setlocale(LC_ALL, "en_US.UTF-8");
```

但是这样的话，程序的可移植性就变差了，因为无法保证其他系统也会支持那种格式。所以，通常都将第二个参数设为空字符串，使用操作系统的当前设置。

`setlocale()` 的返回值是一个字符串指针，表示已经设置好的格式。如果调用失败，则返回空指针 NULL。

`setlocale()` 可以用来查询当前地区，这时第二个参数设为 NULL 就可以了。

```
char *loc;

loc = setlocale(LC_ALL, NULL);

// 输出 Starting locale: C
printf("Starting locale: %s\n", loc);

loc = setlocale(LC_ALL, "");

// 输出 Native locale: en_US.UTF-8
printf("Native locale: %s\n", loc);
```

localeconv()

`localeconv()` 用来获取当前格式的详细信息。

```
struct lconv* localeconv(void);
```

该函数返回一个 Struct 结构指针，该结构里面包含了格式信息，它的主要属性如下。

- char* mon_decimal_point: 货币的十进制小数点字符，比如 `.`。
- char* mon_thousands_sep: 货币的千位分隔符，比如 `,`。
- char* mon_grouping: 货币的分组描述符。
- char* positive_sign: 货币的正值符号，比如 `+` 或为空字符串。
- char* negative_sign: 货币的负值符号，比如 `-`。
- char* currency_symbol: 货币符号，比如 `$`。
- char frac_digits: 打印货币金额时，十进制小数点后面输出几位小数，比如设为 `2`。
- char p_cs_precedes: 设为 `1` 时，货币符号 `currency_symbol` 出现在非负金额前面。设为 `0` 时，出现在后面。
- char n_cs_precedes: 设为 `1` 时，货币符号 `currency_symbol` 出现在负的货币金额前面。设为 `0` 时，出现在后面。
- char p_sep_by_space: 决定了非负的货币金额与货币符号之间的分隔字符。
- char n_sep_by_space: 决定了负的货币金额与货币符号之间的分隔字符。
- char p_sign_posn: 决定了非负值的正值符号的位置。
- char n_sign_posn: 决定了负值的负值符号的位置。
- char* int_curr_symbol: 货币的国际符号，比如 `USD`。
- char int_frac_digits: 使用国际符号时，`frac_digits` 的值。
- char int_p_cs_precedes: 使用国际符号时，`p_cs_precedes` 的值。
- char int_n_cs_precedes: 使用国际符号时，`n_cs_precedes` 的值。
- char int_p_sep_by_space: 使用国际符号时，`p_sep_by_space` 的值。
- char int_n_sep_by_space: 使用国际符号时，`n_sep_by_space` 的值。
- char int_p_sign_posn: 使用国际符号时，`p_sign_posn` 的值。
- char int_n_sign_posn: 使用国际符号时，`n_sign_posn` 的值。

下面程序打印当前系统的属性值。

```
#include <stdio.h>
#include <locale.h>
#include <string.h>

int main ()
{
    setlocale (LC_ALL,"zh_CN");
    struct lconv * lc;
    lc=localeconv();
    printf ("decimal_point: %s\n",lc->decimal_point);
    printf ("thousands_sep: %s\n",lc->thousands_sep);
    printf ("grouping: %s\n",lc->grouping);
    printf ("int_curr_symbol: %s\n",lc->int_curr_symbol);
    printf ("currency_symbol: %s\n",lc->currency_symbol);
    printf ("mon_decimal_point: %s\n",lc->mon_decimal_point);
    printf ("mon_thousands_sep: %s\n",lc->mon_thousands_sep);
    printf ("mon_grouping: %s\n",lc->mon_grouping);
    printf ("positive_sign: %s\n",lc->positive_sign);
```

```
    printf ("negative_sign: %s\n",lc->negative_sign);
    printf ("frac_digits: %d\n",lc->frac_digits);
    printf ("p_cs_precedes: %d\n",lc->p_cs_precedes);
    printf ("n_cs_precedes: %d\n",lc->n_cs_precedes);
    printf ("p_sep_by_space: %d\n",lc->p_sep_by_space);
    printf ("n_sep_by_space: %d\n",lc->n_sep_by_space);
    printf ("p_sign_posn: %d\n",lc->p_sign_posn);
    printf ("n_sign_posn: %d\n",lc->n_sign_posn);
    printf ("int_frac_digits: %d\n",lc->int_frac_digits);
    printf ("int_p_cs_precedes: %d\n",lc->int_p_cs_precedes);
    printf ("int_n_cs_precedes: %d\n",lc->int_n_cs_precedes);
    printf ("int_p_sep_by_space: %d\n",lc->int_p_sep_by_space);
    printf ("int_n_sep_by_space: %d\n",lc->int_n_sep_by_space);
    printf ("int_p_sign_posn: %d\n",lc->int_p_sign_posn);
    printf ("int_n_sign_posn: %d\n",lc->int_n_sign_posn);

    return 0;
}
```

□□□ #2
□ □□□ □□□ 2021-09-04 22:34:34 UTC
□ □□□ □□□ 2022-12-14 19:30:47 UTC