"en_IE.iso885915@euro" (using the ISO/IEC 8859-15 character set and the euro currency), and "en_IE.utf8" (using the UTF-8 encoding of the Unicode character set).

UTF-8 ➤25.2

Linux and other versions of UNIX support the `locale` command, which can be used to get locale information. One use of the `locale` command is to get a list of all available locales, which can be done by entering

```
locale -a
```

at the command line.

Because locale information is becoming increasingly important, the Unicode Consortium created the Common Locale Data Repository (CLDR) project to establish a standard set of locales. More information about the CLDR project can be found at *www.unicode.org/cldr/*.

When a call of `setlocale` succeeds, it returns a pointer to a string associated with the category in the new locale. (The string might be the locale name itself, for example.) On failure, `setlocale` returns a null pointer.

`setlocale` can also be used as a query function. If its second argument is a null pointer, `setlocale` returns a pointer to a string associated with the category in the *current* locale. This feature is especially useful if the first argument is LC_ALL, since it allows us to fetch the current settings for all categories. A string returned by `setlocale` can be saved (by copying it into a variable) and then used in a later call of `setlocale`.

Q&A

The `localeconv` Function

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

localeconv

Although we can ask `setlocale` about the current locale, the information that it returns isn't necessarily in the most useful form. To find out highly specific information about the current locale (What's the decimal-point character? What's the currency symbol?), we need `localeconv`, the only other function declared in `<locale.h>`.

`localeconv` returns a pointer to a structure of type `struct lconv`. The members of this structure contain detailed information about the current locale. The structure has static storage duration and may be modified by a later call of `localeconv` or `setlocale`. Be sure to extract the desired information from the `lconv` structure before it's wiped out by one of these functions.

Some members of the `lconv` structure have `char *` type; other members have `char` type. Table 25.1 lists the `char *` members. The first three members describe the formatting of nonmonetary quantities, while the others deal with monetary quantities. The table also shows the value of each member in the "C" locale (the default); a value of "" means "not available."

The `grouping` and `mon_grouping` members deserve special mention.