Table 6-3. *Standard GLib Macros*

Macro	Description
`ABS (a)`	Return the absolute value of argument `a`. This function simply returns any negative number without the negative sign and does nothing to positive numbers.
`CLAMP (a, low, high)`	Make sure that `a` is between `low` and `high`. If `a` is not between `low` and `high`, the returned value will be the closest of the two. Otherwise, the returned value will be left unchanged.
`G_DIR_SEPARATOR` `G_DIR_SEPARATOR_S`	On UNIX machines, directories are separated by a slash (`/`), and on Windows machines, they are separated by a backslash (`\`). `G_DIR_SEPARATOR` will return the appropriate separator as a character, and `G_DIR_SEPARATOR_S` will return the separator as a string.

`GINT_TO_POINTER (i)`
`GPOINTER_TO_INT (p)`
	Convert an integer to a `gpointer` or a `gpointer` to an integer. Only 32 bits of the integer will be stored, so you should avoid using integers that will take up more than that amount of space when using these macros. Remember that you cannot store pointers in integers. This only allows you to store an integer as a pointer.

`GSIZE_TO_POINTER (s)`
`GPOINTER_TO_SIZE (p)`
	Convert a `gsize` value to a `gpointer` or a `gpointer` to `gsize` value. The `gsize` data type must have been stored as a pointer with `GSIZE_TO_POINTER()` to convert it back. See `GINT_TO_POINTER()` for more information.

`GUINT_TO_POINTER (u)`
`GPOINTER_TO_UINT (p)`
	Convert an unsigned integer to a `gpointer` or a `gpointer` to an unsigned integer. The integer must have been stored as a pointer with `GUINT_TO_POINTER()` to convert it back. See `GINT_TO_POINTER()` for more information.

`G_OS_WIN32`
`G_OS_BEOS`
`G_OS_UNIX`
	These three macros allow you to define code that will only be run on a specific platform. Only the macro corresponding to the user's system will be defined, so you can bracket code specific to the user's operating system with `#ifdef G_OS_*`.

`G_STRUCT_MEMBER (type, struct_p, offset)`	Returns the member of the structure located at the specified `offset`. This `offset` must be within `struct_p`. `type` defines the data type of the field you are retrieving.
`G_STRUCT_MEMBER_P (struct_p, offset)`	Returns an untyped pointer to the member of the structure located at the specified `offset`. The `offset` must be within `struct_p`.
`G_STRUCT_OFFSET (type, member)`	Returns the byte offset of a member within a structure. The structure type is defined by `type`.

`MIN (a, b)`
`MAX (a, b)`
	Calculates the minimum or maximum value of the two arguments `a` and `b` respectively.

`TRUE` and `FALSE`	`FALSE` is defined as zero, and `TRUE` is set to the logical not of `FALSE`. These values are used for the `gboolean` type.