```
long long int strtoll(const char * restrict nptr,
                      char ** restrict endptr,
                      int base);
unsigned long int strtoul(
  const char * restrict nptr,
  char ** restrict endptr, int base);
unsigned long long int strtoull(
  const char * restrict nptr,
  char ** restrict endptr, int base);
```

The numeric conversion functions (or "string conversion functions," as they're known in C89) convert strings containing numbers in character form to their equivalent numeric values. Three of these functions are fairly old, another three were added when the C89 standard was created, and five more were added in C99. C99

All the numeric conversion functions—whether new or old—work in much the same way. Each function attempts to convert a string (pointed to by the `nptr` parameter) to a number. Each function skips white-space characters at the beginning of the string, treats subsequent characters as part of a number (possibly beginning with a plus or minus sign), and stops at the first character that can't be part of the number. In addition, each function returns zero if no conversion can be performed (the string is empty or the characters following any initial white space don't have the form the function is looking for).

atof
atoi
atol

The old functions (`atof`, `atoi`, and `atol`) convert a string to a `double`, `int`, or `long int` value, respectively. Unfortunately, these functions lack any way to indicate how much of the string was consumed during a conversion. Moreover, the functions have no way to indicate that a conversion was unsuccessful. (Some implementations of these functions may modify the `errno` variable when a conversion fails, but that's not guaranteed.)

errno variable ➤24.2

strtod
strtol
strtoul

The C89 functions (`strtod`, `strtol`, and `strtoul`) are more sophisticated. For one thing, they indicate where the conversion stopped by modifying the variable that `endptr` points to. (The second argument can be a null pointer if we're not interested in where the conversion ended.) To check whether a function was able to consume the entire string, we can just test whether this variable points to a null character. If no conversion could be performed, the variable that `endptr` points to is given the value of `nptr` (as long as `endptr` isn't a null pointer). What's more, `strtol` and `strtoul` have a `base` argument that specifies the base of the number being converted. All bases between 2 and 36 (inclusive) are supported.

Besides being more versatile than the old functions, `strtod`, `strtol`, and `strtoul` are better at detecting errors. Each function stores `ERANGE` in `errno` if a conversion produces a value that's outside the range of the function's return type. In addition, the `strtod` function returns plus or minus `HUGE_VAL`; the

HUGE_VAL macro ➤23.3