Synopsis

Title

strtol(3) et al. shouldn't have a restricted first parameter.

Author

Alejandro Colomar Andres; maintainer of the Linux man-pages project.

Proposal category

Bug fix.

Cc

GNU C library
GNU Compiler Collection
Linux man-pages
Paul Eggert
Xi Ruoyao
Jakub Jelinek
Martin Uecker
LIU Hao
Jonathan Wakely
Richard Earnshaw
Sam James
Ben Boeckel
"Eissfeldt, Heiko"
David Malcolm

Description

restrict

The *restrict* qualifier is used in parameters by APIs to inform the callers that

- The API *may* copy from one of the objects to another one, or perform other operations that would cause similar problems. To avoid violations of for example C11::6.5.16.1p3, callers *must not* pass other references to the same object.
- The API *may* optimize based on the assumption that writing to such a parameter will not affect any of the other objects it received. This is actually just another way to express the previous point.

The definition of the *restrict* qualifier is in terms of accesses. As long as an object is only accessed via one restricted pointer, other restricted pointers are allowed to point to the same object. This is less strict than I think it should be, but this proposal doesn't attempt to change that definition.

A consequence of the definition is that the following program is correct:

```
int f(const char *restrict ca, char *restrict a);

int
main(void)
{
      char  x = 3;
      char  *xp = &x;

      f(xp, xp);
}

int
f(const char *restrict ca, char *restrict a)
{
      /*
       * We're not allowed to use '>' on pointers to distinct (array)
       * objects, but since we don't access the objects, they might
```

```
                 * point to the same object.  In fact, they _must_ point to the
                 * same one to avoid UB here.
                 */
                return ca > a;
        }
```

Diagnostics

While the above program is correct, it's a bad use of the qualifier: one can only guarantee that it's correct by inspecting the source code of both the caller and the callee. Compilers are far more conservative in what is considered a good use of the *restrict* qualifier.

Compilers emit diagnostics so that it is only necessary to inspect the source code of either the caller or the callee to detect what is likely a violation. Ideally, the definition of *restrict* could be tightened to match those diagnostics.

GCC emits diagnostics for the example program shown above:

```
$ gcc -Wall f.c;
f.c: In function main:
f.c:9:7: warning: passing argument 2 to restrict-qualified parameter aliases wit
    9 |       f(xp, xp);
      |       ^
```

Pointer-to-pointer

Here's another example program that is technically correct. The parameters in this example more closely resemble *strtol*(3), by using a pointer-to-pointer as the second parameter. Other than that, it's conceptually the same as the *f.c* program shown above.

```
int g(const char *restrict ca, char *restrict *restrict ap);

int
main(void)
{
        char  x = 3;
        char  *xp = &x;

        g(xp, &xp);
}

int
g(const char *restrict ca, char *restrict *restrict ap)
{
        return ca > *ap;
}
```

In this case, compilers are not smart enough to detect that both parameters alias each other. However, one could conceive a better compiler that would emit a diagnostic similar to the one in *f.c*.

strtol(3)

The standard prototype for *strtol*(3) is

```
long int
strtol(const char *restrict nptr, char **restrict endptr, int base);
```

strtol(3) accepts 4 objects via pointer parameters and global variables. Here's the list, together with the access mode of each, expressed in terms that mirror those of the GNU access function attribute.

nptr	access(read_only)
endptr	access(write_only)
errno	access(read_write)

**endptr*	access(none)

endptr

cannot alias any other object; it must use *restrict*.

This qualifier helps catch obvious bugs such as *strtol(p, p, 0)* and *strtol(&p, &p, 0)*.

For the other parameters, we need to distinguish between the caller and the callee.

Caller

The caller may be able to know the access mode of *nptr* via compiler-specific attributes. However, there's no current attribute that can specify the access mode of **endptr* nor *errno*, so the caller must assume the worst case: read-write.

The caller knows that *errno* doesn't alias any of the function arguments.

The caller (usually) knows if *nptr* and **endptr* alias each other. However, it has no way to know the access mode of the latter, and so it must assume the object is written to via that alias, which is the worst case.

Here's a list of what the caller can assume:

nptr	access(read_only)	alias *endptr
endptr	access(write_only)	unique
errno	access(read_write)	unique
**endptr*	access(read_write)	alias nptr

Callee

The access modes of every object are known.

The callee knows that **endptr* is not accessed.

This means, for the callee, there are only 3 objects to consider:

nptr	access(read_only)	may alias errno if not restrict
endptr	access(write_only)	unique
errno	access(read_write)	may alias nptr if it's not restrict
*//*endptr*	access(none)	

Depending on if *nptr* is a restricted pointer or not, the calle will know if it can alias *errno* or not.

errno

It might seem that it's a problem that the callee doesn't know if *nptr* can alias *errno* or not.

However, the callee will not write to the latter directly until it knows it has failed; at which point it need not read *nptr* anymore. This means that the callee cannot apply any optimizations where that distinction could be useful.

Still, the callee can call helper functions which are allowed to set *errno* as long as they don't report an error. This might again seem like a reason why the aliasing information would be important.

But nothing prohibits those internal helper functions to specify that *nptr* is *restrict* and thus distinct from *errno*. That way, those optimizations can still be performed, even exposing an API that does not use the *restrict* qualifier on *nptr*.

Analyzer

nptr and **endptr* will often alias each other. Consider for example the case where two numbers are read from a string:

```
char  str[] = "1 2";

p = str;
n = strtol(p, &p, 0);
m = strtol(p, &p, 0);
```

An analyzer more powerful than the current ones could extend the current **−Wrestrict** diagnostic to also diagnose this case.

To prevent triggering diagnostics in a powerful analyzer that would be smart enough to diagnose the example function g(), the prototype of *strtol*(3) should be changed to

```
long int
strtol(const char *nptr, char **restrict endptr, int base);
```

Proposal (diff based on C23/N3220)

7.8.2.3p1 [The strtoimax and strtoumax functions]

```
-intmax_t strtoimax(const char * restrict nptr, char ** restrict endptr, int base);
+intmax_t strtoimax(const char *nptr, char **restrict endptr, int base);

-uintmax_t strtoumax(const char * restrict nptr, char ** restrict endptr, int base);
+uintmax_t strtoumax(const char *nptr, char **restrict endptr, int base);
```

7.8.2.4p1 [The wcstoimax and wcstoumax functions]

```
-intmax_t wcstoimax(const wchar_t *restrict nptr, wchar_t **restrict endptr, int base)
+intmax_t wcstoimax(const wchar_t *nptr, wchar_t **restrict endptr, int base);

-uintmax_t wcstoumax(const wchar_t *restrict nptr, wchar_t **restrict endptr, int base
+uintmax_t wcstoumax(const wchar_t *nptr, wchar_t **restrict endptr, int base);
```

7.24.1.5p1 [The strtod, strtof, and strtold functions]

```
-double strtod(const char *restrict nptr, char **restrict endptr);
+double strtod(const char *nptr, char **restrict endptr);

-float strtof(const char *restrict nptr, char **restrict endptr);
+float strtof(const char *nptr, char **restrict endptr);

-long double strtold(const char *restrict nptr, char **restrict endptr);
+long double strtold(const char *nptr, char **restrict endptr);
```

7.24.1.6p1 [The strtodN functions]

```
-_Decimal32 strtod32(const char * restrict nptr, char ** restrict endptr);
+_Decimal32 strtod32(const char *nptr, char **restrict endptr);

-_Decimal64 strtod64(const char * restrict nptr,char ** restrict endptr);
+_Decimal64 strtod64(const char *nptr, char **restrict endptr);

-_Decimal128 strtod128(const char * restrict nptr,char ** restrict endptr);
+_Decimal128 strtod128(const char *nptr, char **restrict endptr);
```

7.24.1.7p1 [The strtol, strtoll, strtoul, and strtoull functions]

```
-long int strtol(const char *restrict nptr, char **restrict endptr, int base);
+long int strtol(const char *nptr, char **restrict endptr, int base);

-long long int strtoll(const char *restrict nptr, char **restrict endptr, int base);
+long long int strtoll(const char *nptr, char **restrict endptr, int base);

-unsigned long int strtoul(const char *restrict nptr, char **restrict endptr, int base
+unsigned long int strtoul(const char *nptr, char **restrict endptr, int base);

-unsigned long long int strtoull(const char *restrict nptr, char **restrict endptr, in
+unsigned long long int strtoull(const char *nptr, char **restrict endptr, int base);
```

7.31.4.1.2p1 [The wcstod, wcstof, and wcstold functions]

```
-double wcstod(const wchar_t * restrict nptr, wchar_t ** restrict endptr);
+double wcstod(const wchar_t *nptr, wchar_t **restrict endptr);
```

```
−float wcstof(const wchar_t * restrict nptr, wchar_t ** restrict endptr);
+float wcstof(const wchar_t *nptr, wchar_t **restrict endptr);

−long double wcstold(const wchar_t * restrict nptr, wchar_t ** restrict endptr);
+long double wcstold(const wchar_t *nptr, wchar_t **restrict endptr);
```

7.31.4.1.3p1 [The wcstodN functions]

```
−_Decimal32 wcstod32(const wchar_t * restrict nptr, char ** restrict endptr);
+_Decimal32 wcstod32(const wchar_t *nptr, char **restrict endptr);

−_Decimal64 wcstod64(const wchar_t * restrict nptr,char ** restrict endptr);
+_Decimal64 wcstod64(const wchar_t *nptr, char **restrict endptr);

−_Decimal128 wcstod128(const wchar_t * restrict nptr,char ** restrict endptr);
+_Decimal128 wcstod128(const wchar_t *nptr, char **restrict endptr);
```

7.31.4.1.4p1 [The wcstol, wcstoll, wcstoul, and wcstoull functions]

```
−long int wcstol(const wchar_t * restrict nptr, wchar_t ** restrict endptr, int base);
+long int wcstol(const wchar_t *nptr, wchar_t **restrict endptr, int base);

−long long int wcstoll(const wchar_t * restrict nptr, wchar_t ** restrict endptr, int
+long long int wcstoll(const wchar_t *nptr, wchar_t **restrict endptr, int base);

−unsigned long int wcstoul(const wchar_t * restrict nptr, wchar_t ** restrict endptr,
+unsigned long int wcstoul(const wchar_t *nptr, wchar_t **restrict endptr, int base);

−unsigned long long int wcstoull(const wchar_t * restrict nptr, wchar_t ** restrict en
+unsigned long long int wcstoull(const wchar_t *nptr, wchar_t **restrict endptr, int b
```

B.7 [Format conversion of integer types <inttypes.h>]

```
−intmax_t strtoimax(const char * restrict nptr, char ** restrict endptr, int base);
+intmax_t strtoimax(const char *nptr, char **restrict endptr, int base);

−uintmax_t strtoumax(const char * restrict nptr, char ** restrict endptr, int base);
+uintmax_t strtoumax(const char *nptr, char **restrict endptr, int base);

−intmax_t wcstoimax(const wchar_t *restrict nptr, wchar_t **restrict endptr, int base)
+intmax_t wcstoimax(const wchar_t *nptr, wchar_t **restrict endptr, int base);

−uintmax_t wcstoumax(const wchar_t *restrict nptr, wchar_t **restrict endptr, int base
+uintmax_t wcstoumax(const wchar_t *nptr, wchar_t **restrict endptr, int base);
```

B.23 [General utilities <stdlib.h>]

```
−double strtod(const char *restrict nptr, char **restrict endptr);
+double strtod(const char *nptr, char **restrict endptr);

−float strtof(const char *restrict nptr, char **restrict endptr);
+float strtof(const char *nptr, char **restrict endptr);

−long double strtold(const char *restrict nptr, char **restrict endptr);
+long double strtold(const char *nptr, char **restrict endptr);

−long int strtol(const char *restrict nptr, char **restrict endptr, int base);
+long int strtol(const char *nptr, char **restrict endptr, int base);

−long long int strtoll(const char *restrict nptr, char **restrict endptr, int base);
+long long int strtoll(const char *nptr, char **restrict endptr, int base);
```

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

−unsigned long long int strtoull(const char *restrict nptr, char **restrict endptr, in
+unsigned long long int strtoull(const char *nptr, char **restrict endptr, int base);

−_Decimal32 strtod32(const char * restrict nptr, char ** restrict endptr);
+_Decimal32 strtod32(const char *nptr, char **restrict endptr);

−_Decimal64 strtod64(const char * restrict nptr, char ** restrict endptr);
+_Decimal64 strtod64(const char *nptr, char **restrict endptr);

−_Decimal128 strtod128(const char * restrict nptr, char ** restrict endptr);
+_Decimal128 strtod128(const char *nptr, char **restrict endptr);
```

B.30 [Extended multibyte/wide character utilities <wchar.h>]

```
−double wcstod(const wchar_t * restrict nptr, wchar_t ** restrict endptr);
+double wcstod(const wchar_t *nptr, wchar_t **restrict endptr);

−float wcstof(const wchar_t * restrict nptr, wchar_t ** restrict endptr);
+float wcstof(const wchar_t *nptr, wchar_t **restrict endptr);

−long double wcstold(const wchar_t * restrict nptr, wchar_t ** restrict endptr);
+long double wcstold(const wchar_t *nptr, wchar_t **restrict endptr);

−long int wcstol(const wchar_t * restrict nptr, wchar_t ** restrict endptr, int base);
+long int wcstol(const wchar_t *nptr, wchar_t **restrict endptr, int base);

−long long int wcstoll(const wchar_t * restrict nptr, wchar_t ** restrict endptr, int
+long long int wcstoll(const wchar_t *nptr, wchar_t **restrict endptr, int base);

−unsigned long int wcstoul(const wchar_t * restrict nptr, wchar_t ** restrict endptr,
+unsigned long int wcstoul(const wchar_t *nptr, wchar_t **restrict endptr, int base);

−unsigned long long int wcstoull(const wchar_t * restrict nptr, wchar_t ** restrict en
+unsigned long long int wcstoull(const wchar_t *nptr, wchar_t **restrict endptr, int b

−_Decimal32 wcstod32(const wchar_t * restrict nptr, wchar_t ** restrict endptr);
+_Decimal32 wcstod32(const wchar_t *nptr, wchar_t **restrict endptr);

−_Decimal64 wcstod64(const wchar_t * restrict nptr, wchar_t ** restrict endptr);
+_Decimal64 wcstod64(const wchar_t *nptr, wchar_t **restrict endptr);

−_Decimal128 wcstod128(const wchar_t * restrict nptr, wchar_t ** restrict endptr);
+_Decimal128 wcstod128(const wchar_t *nptr, wchar_t **restrict endptr);
```

H.12.3p2 [String to floating]

```
−_FloatN strtofN(const char * restrict nptr, char ** restrict endptr);
+_FloatN strtofN(const char *nptr, char **restrict endptr);

−_FloatNx strtofNx(const char * restrict nptr, char ** restrict endptr);
+_FloatNx strtofNx(const char *nptr, char **restrict endptr);

−_FloatN wcstofN(const wchar_t * restrict nptr, wchar_t ** restrict endptr);
+_FloatN wcstofN(const wchar_t *nptr, wchar_t **restrict endptr);
```

```
−_FloatNx wcstofNx(const wchar_t * restrict nptr, wchar_t ** restrict endptr);
+_FloatNx wcstofNx(const wchar_t *nptr, wchar_t **restrict endptr);

−_DecimalN strtodN(const char * restrict nptr, char ** restrict endptr);
+_DecimalN strtodN(const char *nptr, char **restrict endptr);

−_DecimalNx strtodNx(const char * restrict nptr, char ** restrict endptr);
+_DecimalNx strtodNx(const char *nptr, char **restrict endptr);

−_DecimalN wcstodN(const wchar_t * restrict nptr, wchar_t ** restrict endptr);
+_DecimalN wcstodN(const wchar_t *nptr, wchar_t **restrict endptr);

−_DecimalNx wcstodNx(const wchar_t * restrict nptr, wchar_t ** restrict endptr);
+_DecimalNx wcstodNx(const wchar_t *nptr, wchar_t **restrict endptr);
```

H.12.5.2p1 [The strtoencfN functions]

```
−void strtoencfN(unsigned char encptr[restrict static N/8], const char * restrict nptr
+void strtoencfN(unsigned char encptr[restrict static N/8], const char *nptr, char **r
```

H.12.5.3p1 [The wcstoencfN functions]

```
−void wcstoencfN(unsigned char encptr[restrict static N/8], const wchar_t * restrict n
+void wcstoencfN(unsigned char encptr[restrict static N/8], const wchar_t *nptr, wchar
```

H.12.5.4p1 [The strtoencdecdN and strtoencbindN functions]

```
−void strtoencdecdN(unsigned char encptr[restrict static N/8], const char * restrict n
+void strtoencdecdN(unsigned char encptr[restrict static N/8], const char *nptr, char

−void strtoencbindN(unsigned char encptr[restrict static N/8], const char * restrict n
+void strtoencbindN(unsigned char encptr[restrict static N/8], const char *nptr, char
```

H.12.5.5p1 [The wcstoencdecdN and wcstoencbindN functions]

```
−void wcstoencdecdN(unsigned char encptr[restrict static N/8], const wchar_t *nptr, wc
+void wcstoencdecdN(unsigned char encptr[restrict static N/8], const wchar_t *nptr, wc

−void wcstoencbindN(unsigned char encptr[restrict static N/8], const wchar_t *nptr, wc
+void wcstoencbindN(unsigned char encptr[restrict static N/8], const wchar_t *nptr, wc
```