Proposal for C2Y
WG14 N3704

Title:	Clean up frexp, ldexp, and scalbn prototypes
Author, affiliation:	CFP group
Date:	2025-09-09
Proposal category:	Editorial
Reference:	N3550

Rationale:

A review of C2Y draft N3550 has revealed several uses of the identifier `exp` (as in "exponent") in function prototypes for Annex H extensions to `<math.h>`. This is an unfortunate choice because

```
double exp(double x);
```

is the well-known exponential function. Its name is reserved for any use, per 7.1.3. This change simply replaces such uses of `exp` with one or another identifier consistent with usage in existing prototypes of the corresponding principal functions.

As a bonus item, the survey revealed a use of `exp` in an example as an abbreviation of "expected". That example is changed to align with similar prototypes and code in the same section.

Kudos to Vincent Lefèvre for calling these out.

This change requires no implementation changes. It does change function prototypes for Annex H extensions to `<math.h>`.

Suggested changes:

Typographic note: To emphasize the nature of parameterized identifiers like **_Float***N*, the convention is that the parameter *N* be in italic text (not program) font with plain (not bold) weight. Annex H in draft N3550 does not fully conform to the typographical convention, but these proposals do.

Changes in B.11 and H.11.3 (identical changes in two sections):

From:

```
_FloatN frexpfN(_FloatN value, int *exp);
_FloatNx frexpfNx(_FloatNx value, int *exp);
_DecimalN frexpdN(_DecimalN value, int *exp);
_DecimalNx frexpdNx(_DecimalNx value, int *exp);
```

To:

```
_FloatN frexpfN(_FloatN value, int *p);
_FloatNx frexpfNx(_FloatNx value, int *p);
_DecimalN frexpdN(_DecimalN value, int *p);
_DecimalNx frexpdNx(_DecimalNx value, int *p);
```

Changes in B.11 and H.11.3 (identical changes in two sections):

From:

```
_FloatN ldexpfN(_FloatN value, int *exp);
_FloatNx ldexpfNx(_FloatNx value, int *exp);
_DecimalN ldexpdN(_DecimalN value, int *exp);
_DecimalNx ldexpdNx(_DecimalNx value, int *exp);
```

To:

```
_FloatN ldexpfN(_FloatN value, int *p);
_FloatNx ldexpfNx(_FloatNx value, int *p);
_DecimalN ldexpdN(_DecimalN value, int *p);
_DecimalNx ldexpdNx(_DecimalNx value, int *p);
```

Changes in B.11 and H.11.3 (identical changes in two sections):

From:

```
_FloatN scalbnfN(_FloatN value, int exp);
_FloatNx scalbnfNx(_FloatNx value, int exp);
_DecimalN scalbndN(_DecimalN value, int exp);
_DecimalNx scalbndNx(_DecimalNx value, int exp);
_FloatN scalblnfN(_FloatN value, long int exp);
_FloatNx scalblnfNx(_FloatNx value, long int exp);
_DecimalN scalblndN(_DecimalN value, long int exp);
_DecimalNx scalblndNx(_DecimalNx value, long int exp);
```

To:

```
_FloatN scalbnfN(_FloatN value, int n);
_FloatNx scalbnfNx(_FloatNx value, int n);
_DecimalN scalbndN(_DecimalN value, int n);
_DecimalNx scalbndNx(_DecimalNx value, int n);
_FloatN scalblnfN(_FloatN value, long int n);
_FloatNx scalblnfNx(_FloatNx value, long int n);
_DecimalN scalblndN(_DecimalN value, long int n);
_DecimalNx scalblndNx(_DecimalNx value, long int n);
```

Change in F.10.4.9#1:

From:
On a binary system, `ldexp(x, ~~exp~~)` is equivalent to `scalbn(x, ~~exp~~)`.

To:
On a binary system, `ldexp(x, n)` is equivalent to `scalbn(x, n)`.

Change in 7.17.7.5#7:

From:

```
~~exp~~ = atomic_load(&~~cur~~);
do {
      ~~des~~ = function(~~exp~~);
} while (!atomic_compare_exchange_weak(&~~cur~~, &~~exp~~, ~~des~~));
```

To:

```
expected = atomic_load(&current);
do {
      desired = function(expected);
} while (!atomic_compare_exchange_weak(&current, &expected, desired));
```