‘P’	`mpfr_prec_t`, integer conversions
‘R’	`mpfr_t`, float conversions

The ‘`type`’ specifiers have the same restrictions as those mentioned in the GMP documentation: see Section “Formatted Output Strings” in *GNU MP*. In particular, the ‘`type`’ specifiers (except ‘`R`’ and ‘`P`’) are supported only if they are supported by `gmp_printf` in your GMP build; this implies that the standard specifiers, such as ‘`t`’, must *also* be supported by your C library if you want to use them.

The ‘`rounding`’ field is specific to `mpfr_t` arguments and should not be used with other types.

With conversion specification not involving ‘`P`’ and ‘`R`’ types, `mpfr_printf` behaves exactly as `gmp_printf`.

Thus the ‘`conv`’ specifier ‘`F`’ is not supported (due to the use of ‘`F`’ as the ‘`type`’ specifier for `mpf_t`), except for the ‘`type`’ specifier ‘`R`’ (i.e., for `mpfr_t` arguments).

The ‘`P`’ type specifies that a following ‘`d`’, ‘`i`’, ‘`o`’, ‘`u`’, ‘`x`’, or ‘`X`’ conversion specifier applies to a `mpfr_prec_t` argument. It is needed because the `mpfr_prec_t` type does not necessarily correspond to an `int` or any fixed standard type. The ‘`precision`’ field specifies the minimum number of digits to appear. The default ‘`precision`’ is 1. For example:

```
mpfr_t x;
mpfr_prec_t p;
mpfr_init (x);
...
p = mpfr_get_prec (x);
mpfr_printf ("variable x with %Pu bits", p);
```

The ‘`R`’ type specifies that a following ‘`a`’, ‘`A`’, ‘`b`’, ‘`e`’, ‘`E`’, ‘`f`’, ‘`F`’, ‘`g`’, ‘`G`’, or ‘`n`’ conversion specifier applies to a `mpfr_t` argument. The ‘`R`’ type can be followed by a ‘`rounding`’ specifier denoted by one of the following characters:

‘U’	round toward positive infinity
‘D’	round toward negative infinity
‘Y’	round away from zero
‘Z’	round toward zero
‘N’	round to nearest (with ties to even)
‘*’	rounding mode indicated by the `mpfr_rnd_t` argument just before the corresponding `mpfr_t` variable.