New in 6.d: Thread-Safe, Atomic Operations

We now have a new type—\texttt{atomicint)—and a new set of atomic routines for it. Atomic operations can be safely performed from multiple threads at the same time, without any locking. The new functions are \texttt{atomic-assign}, \texttt{atomic-fetch}, \texttt{atomic-fetch-inc}, \texttt{atomic-fetch-dec}, \texttt{atomic-fetch-add}, \texttt{atomic-fetch-sub}, \texttt{atomic-inc-fetch}, \texttt{atomic-dec-fetch}, and \texttt{cas}.

Unicode lovers rejoice! These are also available as fancy-Unicode operators that use \texttt{⚛} \U+269B ATOM SYMBOL: infix \texttt{⚛=}, prefix \texttt{⚛}, prefix \texttt{⚛++}, postfix \texttt{⚛++}, prefix \texttt{⚛--}, postfix \texttt{⚛-}, infix \texttt{⚛+=}, infix \texttt{⚛-=}, and infix \texttt{⚛−=}

```perl
# Regular ops: not thread safe! Wrong result!
my int $total = 0;
await start { for ^20000 { $total++ } } xx 10;
say $total; # OUTPUT: «188758␤»

# Using atomic operations: thread-safe and correct
my atomicint $atotal = 0;
await start { for ^20000 { $atotal⚛++ } } xx 10;
say $atotal; # OUTPUT: «200000␤»
```