# Chapter 48: Atomics

## Section 48.1: atomics and operators

Atomic variables can be accessed concurrently between different threads without creating race conditions.

```
/* a global static variable that is visible by all threads */
static unsigned _Atomic active = ATOMIC_VAR_INIT(0);


int myThread(void* a) {
  ++active;         // increment active race free
  // do something
  --active;         // decrement active race free
  return 0;
}
```

All lvalue operations (operations that modify the object) that are allowed for the base type are allowed and will not lead to race conditions between different threads that access them.

- Operations on atomic objects are generally orders of magnitude slower than normal arithmetic operations. This also includes simple load or store operations. So you should only use them for critical tasks.
- Usual arithmetic operations and assignment such as `a = a+1;` are in fact three operations on `a`: first a load, then addition and finally a store. This is *not* race free. Only the operation `a += 1;` and `a++;` are.