## Synchronization Primitives

### Lock `process` `thread`

*acquire*

*Thread* → *Lock*

*release*

### RLock `process` `thread`

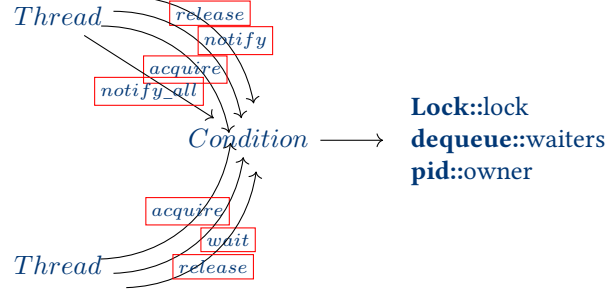*Thread* *acquire*

*RLock* → **pid**::owner **int**::count

*Thread* *release*
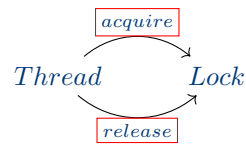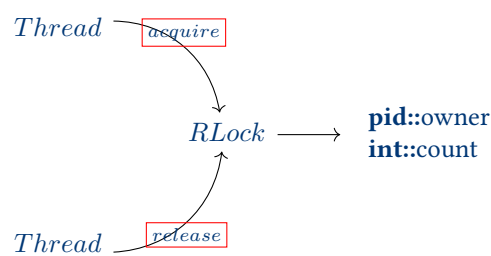
### Condition `process` `thread`

1 or more wait until notified

*Thread* *release* *notify* *acquire* *notify_all*

*Condition* → **Lock**::lock **dequeue**::waiters **pid**::owner

*acquire* *wait* *release*

*Thread*

### Semaphore `process` `thread`

*Thread* *acquire*

*Semaphore* → **Condition**::condvar **int**::value

*Thread* *release*
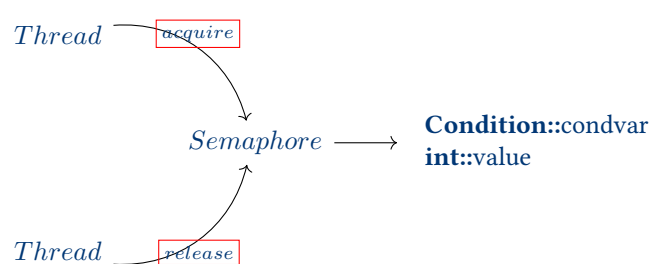
### BoundedSemaphore `process` `thread` `no MacOS`

ValueError is raised if value > initial

---

*Thread* *acquire*

*BoundedSemaphore* → **Condition**::condvar **int**::initial **int**::value

*Thread* *release*

### Event `process` `thread`

*Thread* *set* *clear* *wait* *is_set*

*Event* → **Condition**::condvar **bool**::flag

### Timer `thread`

*Thread* *run* *cancel*

*Timer* → **Event**::finished **int**::interval **function**::f **list**::args **dict**::kwargs
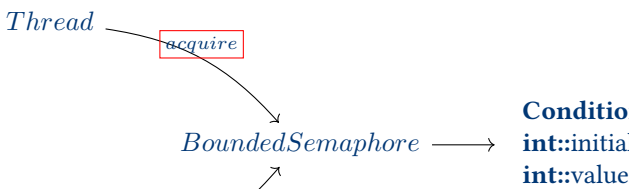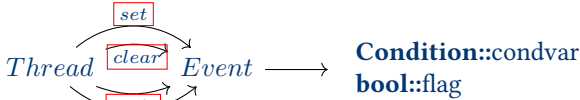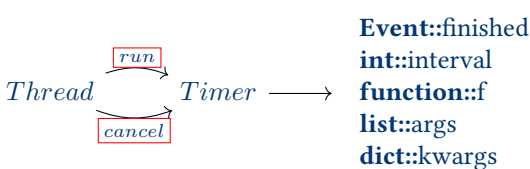
### Barrier `process` `thread`

*Thread* *wait* *reset* *abort*

*Barrier* → **Condition**::condvar **function**::action **int**::timeout **int**::parties **int**::n_waiting **int**::count **int**::state ∈ { Filling Draining Resetting Broken }
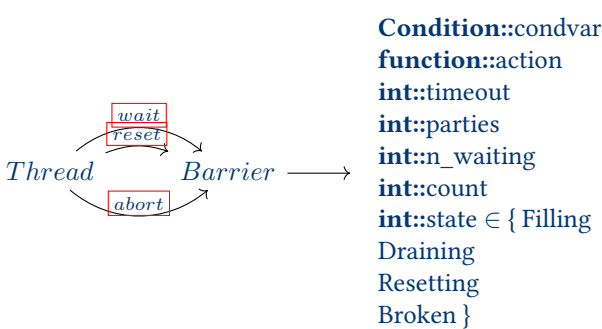
### Thread-Local Data `thread`

*Thread* *getattr* *setattr* *setdefault*
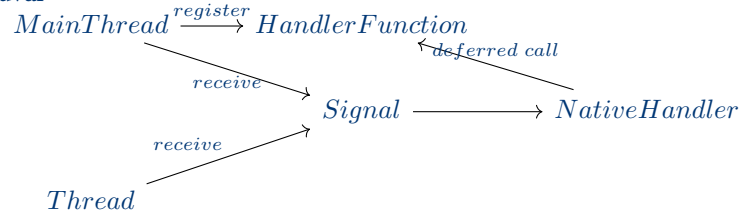
*local* → **dict**::__dict__

### Exception

Convert an Exception instance to string before passing to the different thread

---

## Signal

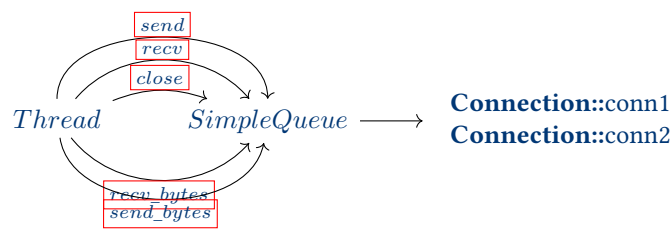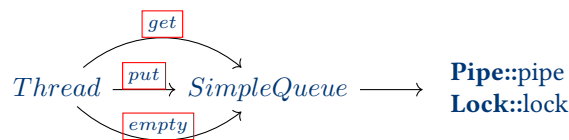Signal can be caught inside any thread, but handling is only allowed within Main Thread

*MainThread* —*register*→ *HandlerFunction*

*deferred call*

*receive*

*Signal* → *NativeHandler*

*receive*

*Thread*

**Condition**::condvar **int**::initial **int**::value

## Inter-Process Communication

### Queue `process`

*Thread* *get* `no MacOS` *qsize* *empty* *get_nowait* *put_nowait* *close* *join_thread*

*SimpleQueue* → **Pipe**::pipe **int**::maxsize **Lock**::lock **pid**::owner **BoundedSemaphore**::sem

### Pipe `process`

*Thread* *send* *recv* *close* *recv_bytes* *send_bytes*

*SimpleQueue* → **Connection**::conn1 **Connection**::conn2

### SimpleQueue

*Thread* *get* *put* *empty*

*SimpleQueue* → **Pipe**::pipe **Lock**::lock

### JoinableQueue `process`

*Thread* *get* *put* *empty* *task_done* *join*

*JoinableQueue* → **Queue**::self **Semaphore**::unfinished_ta Condition::condvar
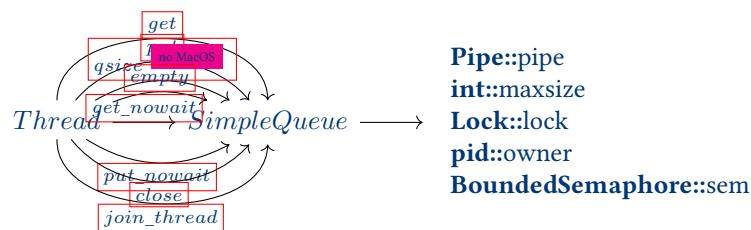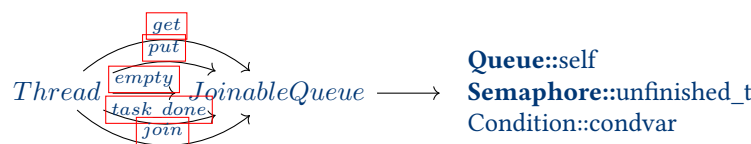
# How Condition Works



# Clusters and Pools