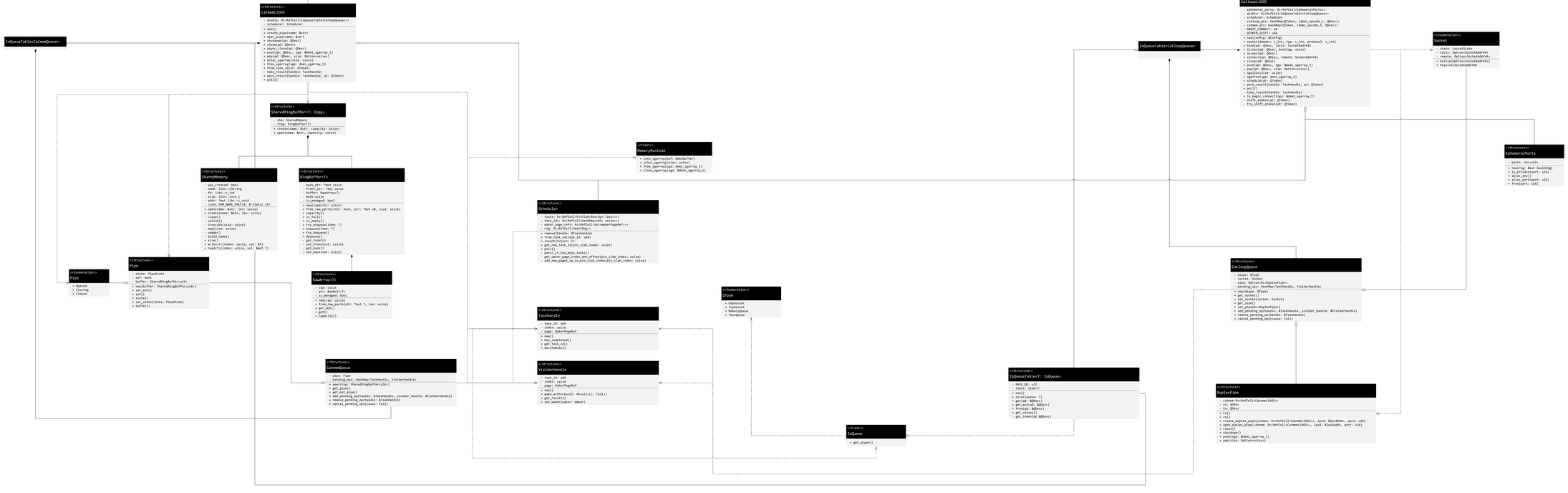
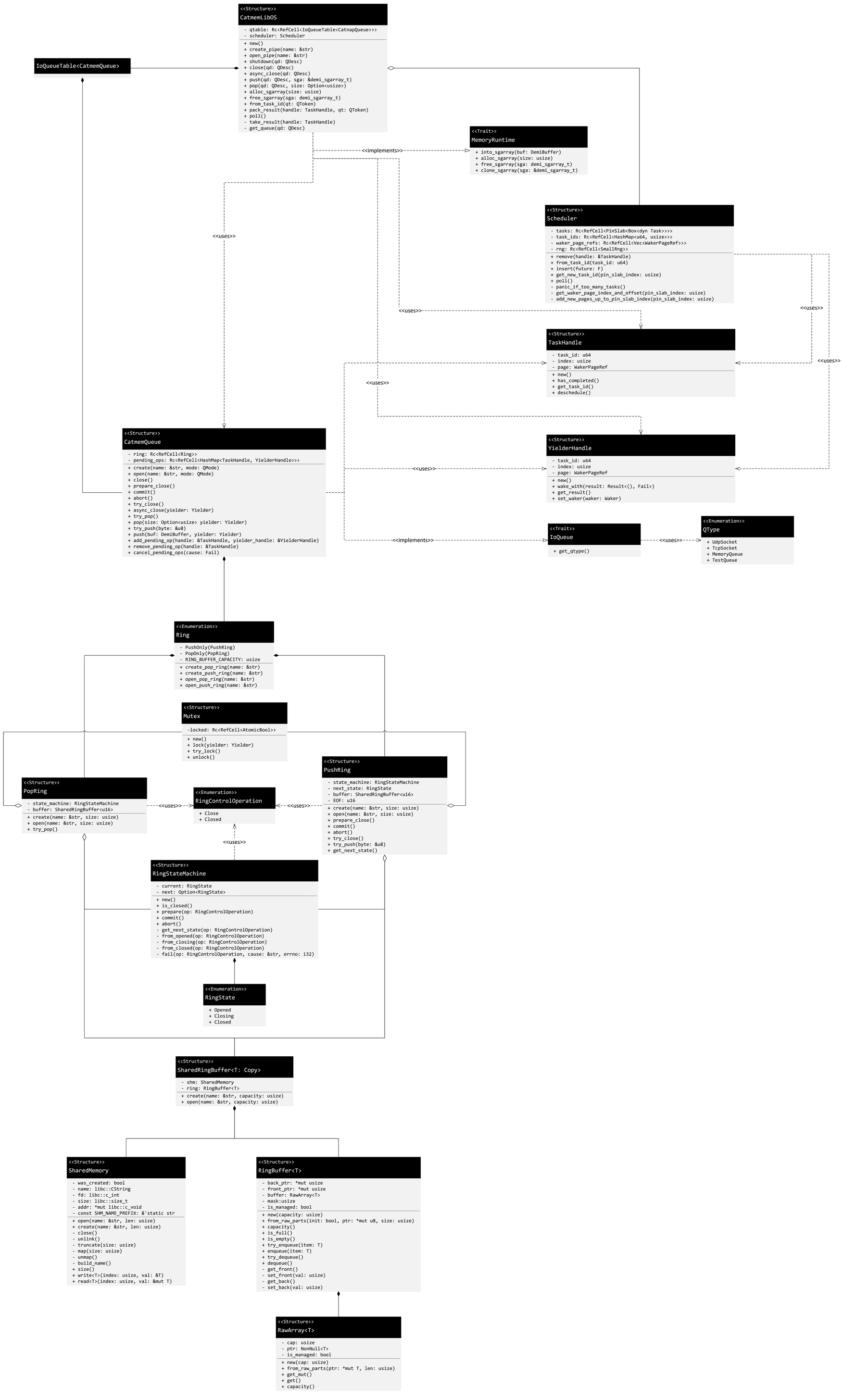

IoQueueTable<CatmemQueue>
<<Structure>>
CatmemLibOS
- static: RcRefCell<IoQueueTable<CatmemQueue>>
- scheduler: Scheduler
+ new()
+ create_pipe(name: &str)
+ open_pipe(name: &str)
+ shutdown(qd: QDesc)
+ close(qd: QDesc)
+ addr(qd: QDesc): QDesc
+ push(qd: QDesc, sga: &dmem_sgarray_t)
+ pop(qd: QDesc, size: Option<usize>)
+ alloc_sga(size: usize)
+ free_sgarray(sga: dmem_sgarray_t)
+ from_task_id(id: QToken)
+ take_result(handle: TaskHandle)
+ pack_result(handle: TaskHandle, qt: QToken)
+ poll()
<<Structure>>
SharedRingBufferT: Copy>
- shm: SharedMemory
- ring: RingBufferT>
+ create(name: &str, capacity: usize)
+ open(name: &str, capacity: usize)
<<Structure>>
SharedMemory
- was_created: bool
- name: CStrRefOwning
- fd: libc::c_int
- size: libc::size_t
- ptr: *mut u8, *const
- const SHM_NAME_PREFIX: &'static str
+ open(name: &str, len: usize)
+ create(name: &str, len: usize)
- unlink()
- unmap()
- truncate(size: usize)
- mapsize: usize)
- unmap()
- build_name()
- write<T>(index: usize, val: &T)
- read<T>(index: usize, val: &mut T)
<<Structure>>
RingBufferT>
- back_ptr: *mut usize
- front_ptr: *mut usize
- buffer: RawArray<T>
- mask:usize
- is_managed: bool
+ new(capacity: usize)
+ from_raw_parts(init: bool, ptr: *mut u8, size: usize)
+ capacity()
+ is_full()
+ is_empty()
+ enqueue(item: T)
+ unqueue(item: T)
+ try_dequeue()
+ dequeue()
+ get_front()
- set_front(val: usize)
- get_back()
- set_back(val: usize)
<<Enumeration>>
Pipe
+ Queue
+ Closing
+ Closed
<<Structure>>
Pipe
- state: PipeState
- eof: bool
- buffer: SharedRingBufferu16>
+ new(buffer: SharedRingBufferu16)
+ eof()
+ eof;()
+ state()
+ set_state(state: PipeState)
- buffer()
<<Structure>>
RawArray<T>
- cap: usize
- ptr: NonNull<T>
- is_managed: bool
+ new(capacity: usize)
+ from_raw_parts(ptr: *mut T, len: usize)
+ get_mut()
+ get()
+ capacity()
<<Structure>>
CatmemQueue
- state: Pipe
- pending_ops: HashMap<TaskHandle, YielderHandle>
+ new(ring: SharedRingBufferu16)
+ get_pipe()
+ get_mut_pipe()
+ add_pending_op(handle: &TaskHandle, yielder_handle: &YielderHandle)
+ remove_pending_op(handle: &TaskHandle)
+ cancel_pending_ops(cause: Fail)
<<trait>>
MemoryRuntime
+ into_sgarray(buf: DemiBuffer)
+ alloc_sgarray(size: usize)
+ free_sgarray(sga: dmem_sgarray_t)
+ clone_sgarray(sga: &dmem_sgarray_t)
<<Structure>>
Scheduler
- tasks: RcRefCell<FctSlab<Box<dyn Task>>>
- page_ref: RcRefCell<MemoryMapped>,
- waker_page_ref: RcRefCell<rc::WakerPageRef>>
- rng: RcRefCell<SmallRng>>
+ remove(handle: &TaskHandle)
+ from_task_id(task_id: u64)
+ insert(future: F)
+ get_new_task_id(pin_slab_index: usize)
+ poll()
+ panic_if_too_many_tasks()
- get_waker_page_index_and_offset(pin_slab_index: usize)
- add_new_pages_up_to_pin_slab_index(pin_slab_index: usize)
<<Enumeration>>
CType
+ IoSocket
+ TcpSocket
+ MemoryQueue
+ TestQueue
<<Structure>>
TaskHandle
- task_id: u64
- index: usize
- page: WakerPageRef
+ new()
+ has_completed()
+ get_task_id()
+ deschedule()
<<Structure>>
YielderHandle
- task_id: u64
- index: usize
- page: WakerPageRef
+ new()
+ wake_with(result: Result<(), Fail>)
+ get_result()
+ set_waker(waker: &Waker)
<<trait>>
IoQueue
+ get_qtype()
<<Structure>>
IoQueueTableT: IoQueue>
- ALG_QD: u32
- table: Slab<T>
+ new()
+ alloc(queue: T)
+ free(qd: QDesc)
+ get_mut(qd: &QDesc)
+ free(qd: &QDesc)
+ get_values()
- get_index(qd &QDesc)
IoQueueTable<CatloopQueue>
<<Structure>>
CatloopLibOS
- ephemeral_ports: RcRefCell<EphemeralPorts>>
- static: RcRefCell<IoQueueTable<CatloopQueue>>
- scheduler: Scheduler
- rating_qts: HashMap<Qtoken, (dmem_opcode_t, QDesc)>
- camem_qts: HashMap<Qtoken, (dmem_opcode_t, QDesc)>
- ALG_QD: u32
- QTOKEN_SHIFT: u64
+ new(config: &Config)
+ socket(domain: c_int, typ: c_int, protocol: c_int)
+ bind(qd: QDesc, local: SocketAddrV4)
+ listen(qd: QDesc, backlog: usize)
+ accept(qd: QDesc)
+ connect(qd: QDesc, remote: SocketAddrV4)
+ close(qd: QDesc)
+ push(qd: QDesc, sga: &dmem_sgarray_t)
+ pop(qd: QDesc, size: Option<usize>)
+ sgalloc(size: usize)
+ sgafree(sga: dmem_sgarray_t)
+ sgaheap_stat(): QToken
+ pack_result(handle: TaskHandle, qt: QToken)
+ poll()
- take_result(handle: TaskHandle)
+ is_magic_connect(sga: &dmem_sgarray_t)
- shift_ahead(qt: QToken)
- try_shift_qtoken(qt: QToken)
<<Enumeration>>
Socket
- state: SocketState
- local: Option<SocketAddrV4>
- remote: Option<SocketAddrV4>
+ Active(Option<SocketAddrV4>)
+ Passive(SocketAddrV4)
<<Structure>>
EphemeralPorts
- ports: Vec<u16>
+ new(rng: &mut SmallRng)
+ is_private(port: u16)
+ alloc_any()
+ alloc_port(port: u16)
+ free(port: u16)
<<Structure>>
CatloopQueue
- qtype: QType
- socket: Socket
- pipe: Option<RcChainedPipe>>
- pending_ops: HashMap<TaskHandle, YielderHandle>
+ new(qtype: QType)
+ get_socket()
+ set_socket(socket: Socket)
+ get_pipe()
+ set_pipe(Rc<ChainedPipe>)
+ add_pending_op(handle: &TaskHandle, yielder_handle: &YielderHandle)
+ remove_pending_op(handle: &TaskHandle)
+ cancel_pending_ops(cause: Fail)
<<Structure>>
DuplexPipe
- catmem Rc<RefCell<CatmemLibOS>>
- rx: QDesc
- tx: QDesc
+ tx()
+ rx()
+ create_duplex_pipe(catmem: Rc<RefCell<CatmemLibOS>>, ipv4: &Ipv4Addr, port: u16)
+ open_duplex_pipe(catmem: Rc<RefCell<CatmemLibOS>>, ipv4: &Ipv4Addr, port: u16)
+ close()
+ shutdown()
+ push(sga: &dmem_sgarray_t)
+ pop(size: Option<usize>)

<<Structure>>
CatnapLibOS
- qtable: Rc<RefCell<IoQueueTable<CatnapQueue>>>
- runtime: PosixRuntime
+ new(config: &Config)
+ socket(domain: c_int, typ: c_int, protocol: c_int)
+ bind(qd: QDesc, local: SocketAddrV4)
+ listen(qd: QDesc, backlog: usize)
+ accept(qd: QDesc)
+ accept_coroutine(qd: QDesc, yielder: Yielder)
+ connect(qd: QDesc, remote: SocketAddrV4)
+ connect_coroutine(qd: QDesc, remote: SocketAddrV4, yielder: Yielder)
+ close(qd: QDesc)
+ async_close(qd: QDesc)
+ close_coroutine(qd: QDesc, yielder: Yielder)
+ push(qd: QDesc, sga: &demi_sgarray_t)
+ push_coroutine(qd: QDesc, mut buf: DemiBuffer, yielder: Yielder)
+ pushto(qd: QDesc, sga: &demi_sgarray_t, remote: SocketAddrV4)
+ pushto_coroutine(qd: QDesc, mut buf: DemiBuffer, remote: SocketAddrV4, yielder: Yielder)
+ pop(qd: QDesc, size: Option<usize>)
+ pop_coroutine(qd: QDesc, size: Option<usize>, yielder: Yielder)
+ poll()
+ schedule(qt: QToken)
+ pack_result(handle: TaskHandle, qt: QToken)
+ sgalloc(size: usize)
+ sgafree(sga: demi_sgarray_t)
- insert_task_into_scheduler(task_id: String, qd: QDesc, yielder_handle: &YielderHandle, coroutine: Pin<Box<Operation>>)
- take_result(handle: TaskHandle)
- get_queue(qd: QDesc)
IoQueueTable<CatnapQueue>
<<Trait>>
Runtime
<<implements>>
<<Structure>>
PosixRuntime
+ scheduler: Scheduler
+ new()
<<implements>>
<<Trait>>
MemoryRuntime
+ into_sgarray(buf: DemiBuffer)
+ alloc_sgarray(size: usize)
+ free_sgarray(sga: demi_sgarray_t)
+ clone_sgarray(sga: &demi_sgarray_t)
<<uses>>
<<uses>>
<<uses>>
<<Structure>>
CatnapQueue
- qtype: QType
- fd: Option<RawFd>
- socket: Socket
- pending_ops: HashMap<TaskHandle, YielderHandle>
+ new(qtype: QType, fd: Option<RawFd>)
+ local()
+ remote()
+ bind(local: SocketAddrV4)
+ listen(backlog: usize)
+ accept(coroutine: F)
+ do_accept(yielder: Yielder)
+ connect(coroutine: F)
+ do_connect(remote: SocketAddrV4, yielder: Yielder)
+ close()
+ async_close(coroutine: F)
+ do_close(yielder: Yielder)
+ push(coroutine: F)
+ do_push(buf: DemiBuffer, addr: Option<SocketAddrV4>, yielder: Yielder)
+ pop(coroutine: F)
+ do_pop(size: Option<usize>, yielder: Yielder)
+ do_generic_sync_control_path_call(coroutine: F, add_as_pending_op: bool)
+ do_generic_sync_data_path_call(coroutine: F)
+ add_pending_op(handle: &TaskHandle, yielder_handle: &YielderHandle)
+ remove_pending_op(handle: &TaskHandle)
+ cancel_pending_ops(cause: Fail)
<<uses>>
<<uses>>
<<Enumeration>>
QType
+ UdpSocket
+ TcpSocket
+ MemoryQueue
+ TestQueue
<<Structure>>
Scheduler
- tasks: Rc<RefCell<PinSlab<Box<dyn Task>>>>
- task_ids: Rc<RefCell<HashMap<u64, usize>>>
- waker_page_refs: Rc<RefCell<Vec<WakerPageRef>>>
- rng: Rc<RefCell<SmallRng>>
+ remove(handle: &TaskHandle)
+ from_task_id(task_id: u64)
+ insert(future: F)
+ get_new_task_id(pin_slab_index: usize)
+ poll()
- panic_if_too_many_tasks()
- get_waker_page_index_and_offset(pin_slab_index: usize)
- add_new_pages_up_to_pin_slab_index(pin_slab_index: usize)
<<uses>>
<<uses>>
<<Enumeration>>
SocketOp
+ NotBound
+ Bound
+ Listening
+ Accepting
+ Connecting
+ Connected
+ Closing
+ Closed
<<uses>>
<<uses>>
<<Enumeration>>
SocketState
+ NotBound
+ Bound
+ Listening
+ Accepting
+ Connecting
+ Connected
+ Closing
+ Closed
+ new(domain: c_int, typ: c_int)
+ bind(local: SocketAddrV4)
+ listen(backlog: usize)
+ prepare_accept()
+ try_accept()
+ prepare_connect()
+ try_connect(remote: SocketAddrV4)
+ try_close()
+ try_push(buf: &DemiBuffer, addr: Option<SocketAddrV4>)
+ try_pop(&DemiBuffer, size: usize)
+ local()
+ remote()
+ commit()
+ abort()
- prepare()
- get_next_state(op: SocketOp)
- not_bound_state(op: SocketOp)
- bound_state(op: SocketOp)
- listen_state(op: SocketOp)
- accepting_state(op: SocketOp)
- connecting_state(op: SocketOp)
- connected_state(op: SocketOp)
- closing_state(op: SocketOp)
- closed_state(op: SocketOp)
- fail(op: SocketOp, cause &str, errno: i32)
- retry_errno(errno: i32)
<<uses>>
<<Structure>>
Socket
- state: SocketState
- next_state: Option<SocketState>
- fd: RawFd
- local: Option<SocketAddrV4>
- remote: Option<SocketAddrV4>
+ new(domain: c_int, typ: c_int)
+ prepare_bind()
+ bind(local: SocketAddrV4)
+ prepare_listen()
+ listen(backlog: usize)
+ prepare_accept()
+ prepare_accepted()
+ try_accept()
+ prepare_connect()
+ prepare_connected()
+ try_connect(remote: SocketAddrV4)
+ prepare_close()
+ prepare_closed()
+ try_close()
+ try_push(buf: &DemiBuffer, addr: Option<SocketAddrV4>)
+ try_pop(&DemiBuffer, size: usize)
+ local()
+ remote()
+ commit()
+ abort()
+ rollback()
- retry_errno(errno: i32)
<<Structure>>
SocketState
+ previous: Option<SocketState_>
+ current: SocketState_
+ next: Option<SocketState_>
+ new_unbound()
+ new_connected()
+ commit()
+ abort()
+ rollback()
+ prepare()
- get_next_state(op: SocketOp)
- not_bound_state(op: SocketOp)
- bound_state(op: SocketOp)
- listen_state(op: SocketOp)
- accepting_state(op: SocketOp)
- connecting_state(op: SocketOp)
- connected_state(op: SocketOp)
- closing_state(op: SocketOp)
- closed_state(op: SocketOp)
- fail(op: SocketOp, cause &str, errno: i32)
<<uses>>
<<Structure>>
TaskHandle
- task_id: u64
- index: usize
- page: WakerPageRef
+ new()
+ has_completed()
+ get_task_id()
+ deschedule()
<<uses>>
<<Structure>>
YielderHandle
- task_id: u64
- index: usize
- page: WakerPageRef
+ new()
+ wake_with(result: Result<(), Fail>)
+ get_result()
+ set_waker(waker: Waker)
<<uses>>
<<uses>>
<<implements>>
<<Trait>>
IoQueue
+ get_qtype()
<<implements>>
<<Structure>>
IoQueueTable<T: IoQueue>
- BASE_QD: u32
- table: Slab<T>
+ new()
+ alloc(queue: T)
+ get(qd: &QDesc)
+ get_mut(qd: &QDesc)
+ free(qd: &QDesc)
+ get_values()
- get_index(qd &QDesc)


IoQueueTable<CatmemQueue>
<<Structure>>
CatmemLibOS
- qtable: Rc<RefCell<IoQueueTable<CatnapQueue>>>
- scheduler: Scheduler
+ new()
+ create_pipe(name: &str)
+ open_pipe(name: &str)
+ shutdown(qd: QDesc)
+ close(qd: QDesc)
+ async_close(qd: QDesc)
+ push(qd: QDesc, sga: &demi_sgarray_t)
+ pop(qd: QDesc, size: Option<usize>)
+ alloc_sgarray(size: usize)
+ free_sgarray(sga: demi_sgarray_t)
+ from_task_id(qt: QToken)
+ pack_result(handle: TaskHandle, qt: QToken)
+ poll()
- take_result(handle: TaskHandle)
- get_queue(qd: QDesc)
<<Trait>>
MemoryRuntime
+ into_sgarray(buf: DemiBuffer)
+ alloc_sgarray(size: usize)
+ free_sgarray(sga: demi_sgarray_t)
+ clone_sgarray(sga: &demi_sgarray_t)
<<implements>>
<<uses>>
<<Structure>>
Scheduler
- tasks: Rc<RefCell<PinSlab<Box<dyn Task>>>>
- task_ids: Rc<RefCell<HashMap<u64, usize>>>
- waker_page_refs: Rc<RefCell<Vec<WakerPageRef>>>
- rng: Rc<RefCell<SmallRng>>
+ remove(handle: &TaskHandle)
+ from_task_id(task_id: u64)
+ insert(future: F)
+ get_new_task_id(pin_slab_index: usize)
+ poll()
- panic_if_too_many_tasks()
- get_waker_page_index_and_offset(pin_slab_index: usize)
- add_new_pages_up_to_pin_slab_index(pin_slab_index: usize)
<<uses>>
<<uses>>
<<Structure>>
TaskHandle
- task_id: u64
- index: usize
- page: WakerPageRef
+ new()
+ has_completed()
+ get_task_id()
+ deschedule()
<<uses>>
<<uses>>
<<Structure>>
CatmemQueue
- ring: Rc<RefCell<Ring>>
- pending_ops: Rc<RefCell<HashMap<TaskHandle, YielderHandle>>>
+ create(name: &str, mode: QMode)
+ open(name: &str, mode: QMode)
+ close()
+ prepare_close()
+ commit()
+ abort()
+ try_close()
+ async_close(yielder: Yielder)
+ try_pop()
+ pop(size: Option<usize> yielder: Yielder)
+ try_push(byte: &u8)
+ push(buf: DemiBuffer, yielder: Yielder)
+ add_pending_op(handle: &TaskHandle, yielder_handle: &YielderHandle)
+ remove_pending_op(handle: &TaskHandle)
+ cancel_pending_ops(cause: Fail)
<<uses>>
<<Structure>>
YielderHandle
- task_id: u64
- index: usize
- page: WakerPageRef
+ new()
+ wake_with(result: Result<(), Fail>)
+ get_result()
+ set_waker(waker: Waker)
<<implements>>
<<Trait>>
IoQueue
+ get_qtype()
<<uses>>
<<Enumeration>>
QType
+ UdpSocket
+ TcpSocket
+ MemoryQueue
+ TestQueue
<<Enumeration>>
Ring
- PushOnly(PushRing)
- PopOnly(PopRing)
- RING_BUFFER_CAPACITY: usize
+ create_pop_ring(name: &str)
+ create_push_ring(name: &str)
+ open_pop_ring(name: &str)
+ open_push_ring(name: &str)
<<Structure>>
Mutex
-locked: Rc<RefCell<AtomicBool>>
+ new()
+ lock(yielder: Yielder)
+ try_lock()
+ unlock()
<<Structure>>
PopRing
- state_machine: RingStateMachine
- buffer: SharedRingBuffer<u16>
+ create(name: &str, size: usize)
+ open(name: &str, size: usize)
+ try_pop()
<<uses>>
<<Enumeration>>
RingControlOperation
+ Close
+ Closed
<<uses>>
<<Structure>>
PushRing
- state_machine: RingStateMachine
- next_state: RingState
- buffer: SharedRingBuffer<u16>
- EOF: u16
+ create(name: &str, size: usize)
+ open(name: &str, size: usize)
+ prepare_close()
+ commit()
+ abort()
+ try_close()
+ try_push(byte: &u8)
+ get_next_state()
<<uses>>
<<Structure>>
RingStateMachine
- current: RingState
- next: Option<RingState>
+ new()
+ is_closed()
+ prepare(op: RingControlOperation)
+ commit()
+ abort()
- get_next_state(op: RingControlOperation)
- from_opened(op: RingControlOperation)
- from_closing(op: RingControlOperation)
- from_closed(op: RingControlOperation)
- fail(op: RingControlOperation, cause: &str, errno: i32)
<<Enumeration>>
RingState
+ Opened
+ Closing
+ Closed
<<Structure>>
SharedRingBuffer<T: Copy>
- shm: SharedMemory
- ring: RingBuffer<T>
+ create(name: &str, capacity: usize)
+ open(name: &str, capacity: usize)
<<Structure>>
SharedMemory
- was_created: bool
- name: libc::CString
- fd: libc::c_int
- size: libc::size_t
- addr: *mut libc::c_void
- const SHM_NAME_PREFIX: &'static str
+ open(name: &str, len: usize)
+ create(name: &str, len: usize)
- close()
- unlink()
- truncate(size: usize)
- map(size: usize)
- unmap()
- build_name()
+ size()
+ write<T>(index: usize, val: &T)
+ read<T>(index: usize, val: &mut T)
<<Structure>>
RingBuffer<T>
- back_ptr: *mut usize
- front_ptr: *mut usize
- buffer: RawArray<T>
- mask:usize
- is_managed: bool
+ new(capacity: usize)
+ from_raw_parts(init: bool, ptr: *mut u8, size: usize)
+ capacity()
+ is_full()
+ is_empty()
+ try_enqueue(item: T)
+ enqueue(item: T)
+ try_dequeue()
+ dequeue()
- get_front()
- set_front(val: usize)
- get_back()
- set_back(val: usize)
<<Structure>>
RawArray<T>
- cap: usize
- ptr: NonNull<T>
- is_managed: bool
+ new(cap: usize)
+ from_raw_parts(ptr: *mut T, len: usize)
+ get_mut()
+ get()
+ capacity()