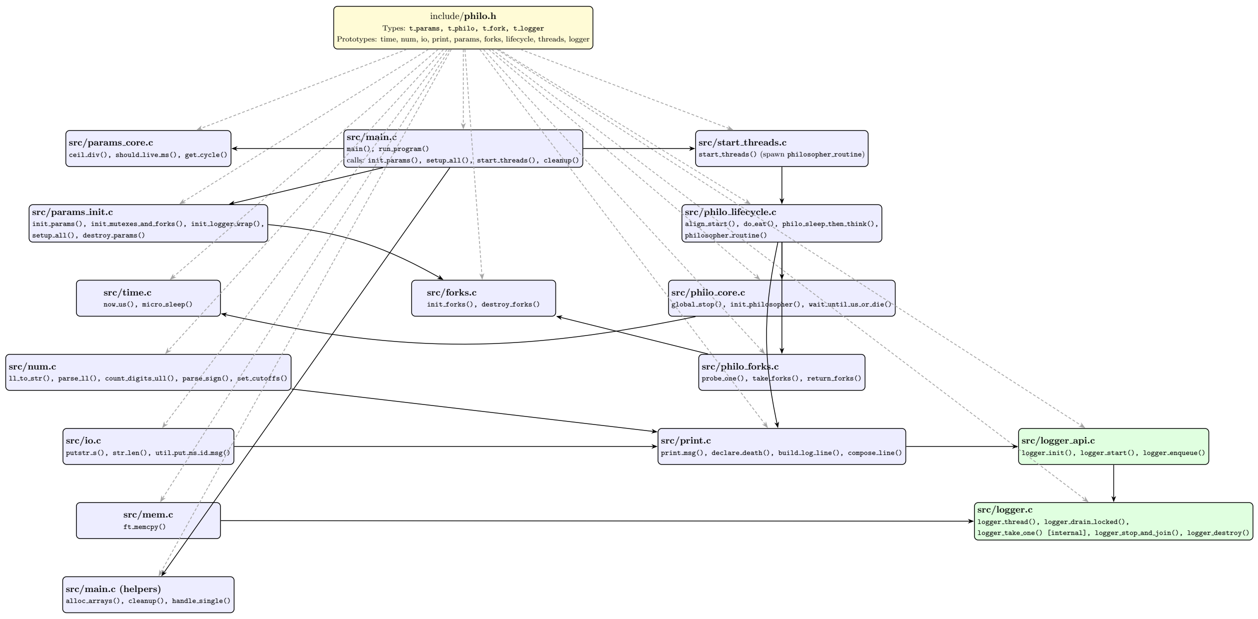
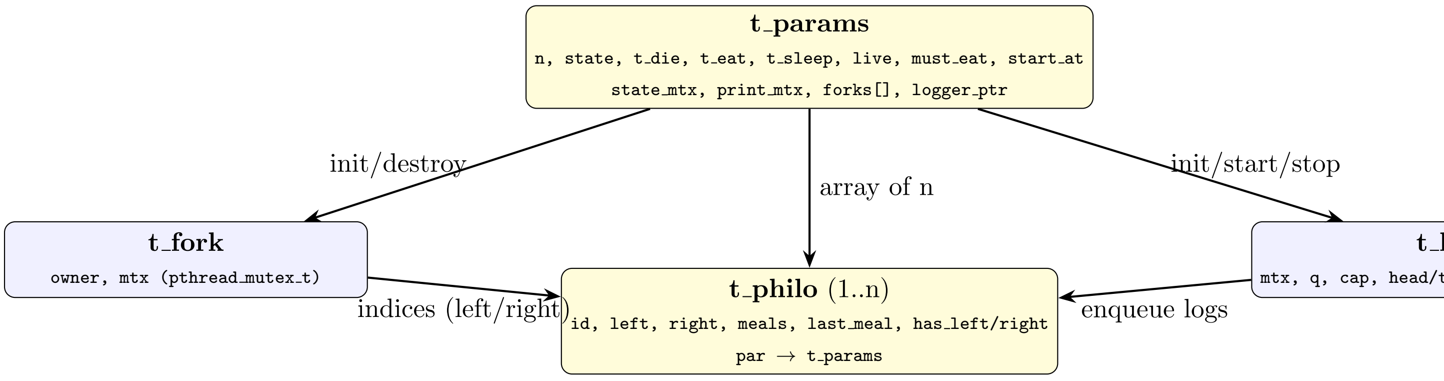
Philosophers (Mutex Version) — File Structure Flow


include/philo.h
Types: t_params, t_philo, t_fork, t_logger
Prototypes: time, num, io, print, params, forks, lifecycle, threads, logger
src/params_core.c
ceil_div(), should_live_ms(), get_cycle()
src/main.c
main(), run_program()
calls: init_params(), setup_all(), start_threads(), cleanup()
src/start_threads.c
start_threads() (spawn philosopher_routine)
src/params_init.c
init_params(), init_mutexes_and_forks(), init_logger_wrap(),
setup_all(), destroy_params()
src/philo_lifecycle.c
align_start(), do_eat(), philo_sleep_then_think(),
philosopher_routine()
src/time.c
now_us(), micro_sleep()
src/forks.c
init_forks(), destroy_forks()
src/philo_core.c
global_stop(), init_philosopher(), wait_until_us_or_die()
src/num.c
ll_to_str(), parse_ll(), count_digits_ull(), parse_sign(), set_cutoffs()
src/philo_forks.c
probe_one(), take_forks(), return_forks()
src/io.c
putstr_s(), str_len(), util_put_ms_id_msg()
src/print.c
print_msg(), declare_death(), build_log_line(), compose_line()
src/logger_api.c
logger_init(), logger_start(), logger_enqueue()
src/mem.c
ft_memcpy()
src/logger.c
logger_thread(), logger_drain_locked(),
logger_take_one() [internal], logger_stop_and_join(), logger_destroy()
src/main.c (helpers)
alloc_arrays(), cleanup(), handle_single()

Data Structures (from `philo.h`)


t_params
n, state, t_die, t_eat, t_sleep, live, must_eat, start_at
state_mtx, print_mtx, forks[], logger_ptr
init/destroy
array of n
init/start/stop
t_fork
owner, mtx (pthread_mutex_t)
t_philo (1..n)
id, left, right, meals, last_meal, has_left/right
par → t_params
indices (left/right)
enqueue logs
t_l
mtx, q, cap, head/t

1 Project Structure (Mutex Version)

1.1 main.c

Entrypoint: parses args, initializes params/mutexes/forks/logger, starts threads, handles single-philosopher case, and runs `cleanup()`.

1.2 params_core.c & params_init.c

Timing helpers (`ceil_div(), get_cycle(), should_live_ms()`), parameter parsing, and all mutex/fork allocation: `init_params(), init_mutexes_and_forks(), init_logger_wrap(), setup_all(), destroy_params()`.

1.3 start_threads.c

Creates philosopher threads and starts the background logger thread.

1.4 philo_lifecycle.c, philo_core.c, philo_forks.c

Implements the life cycle (`philosopher_routine()`), precise waiting with death checks, and fork acquisition/release with mutexes.

1.5 print.c, logger_api.c, logger.c

Builds formatted log lines and enqueues them to a dedicated logger thread that drains the queue and serializes output (guarded by `print_mtx`; internal helpers like `logger_take_one()` live in `logger.c`).

1.6 forks.c

Low-level fork table ownership and (de)initialization for `t_fork` entries.

1.7 time.c, num.c, io.c, mem.c

Time utilities in microseconds, number/string helpers (e.g., `ll_to_str()`), minimal I/O functions, and memory helpers.

1.8 include/philo.h

Single header with all types and prototypes (mutex-based API, logger primitives, timing utilities).