main

init

system init() → coreDump → root_path → logger → iniConfig → workflow
(iniParser) (xml Parser)

server
config()
start()

param

IP
port
threads
max connection
wait_time epoll

讨论组 LOOP

LOOP: ↻

m.epoll → wait (wait_time)

① 有新socket连接
socket_poll. allocate()
↑
socket_fd

attach() 进入 epoller 监听

task
Void *data

<work_thread, task>

创建 work_thread
线程池 并
创建线程 detach

task_dispatcher
init (m_threads)

socket_handler

listen(ip,port)

handle (connects, wait time)

thread pool
○○○○
顺序线程
detach()

创建 server_socket (ip, port)

② 与服务端产生的别居
持持 socket_fd
(挂起信息) EPOLLHup → detach() → 从epoller删除
(挂 error) EPOLLER → detach() → remove()
[back to socket_pool]
EPOLLIN → detach()
包状态 task (socket_fd)

创建 event_poller, 将对机 socket_fd, 加入 epoller
创建 socket 对象地,用于存储 client socket fd

还关你
队列

assign

task_dispatcher

Signal

LOOP
task_dispatcher 线程

condition wait

handle (task)

thread pool 中含获用线程, 处理就队取出 task, 将有空闲线程 创
结队回归入
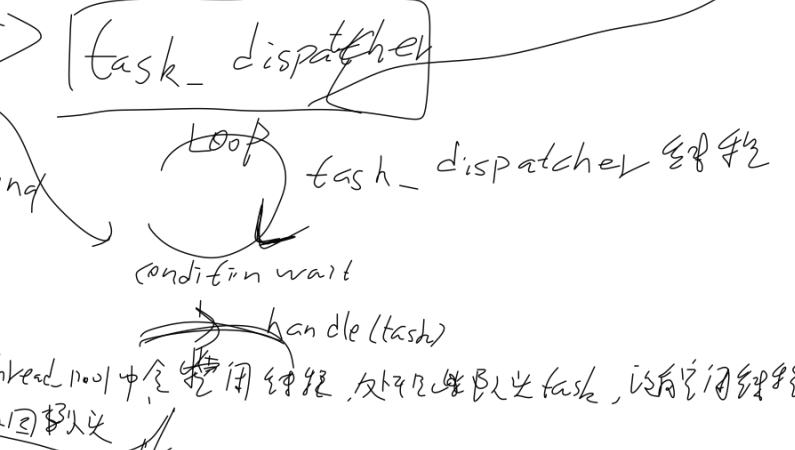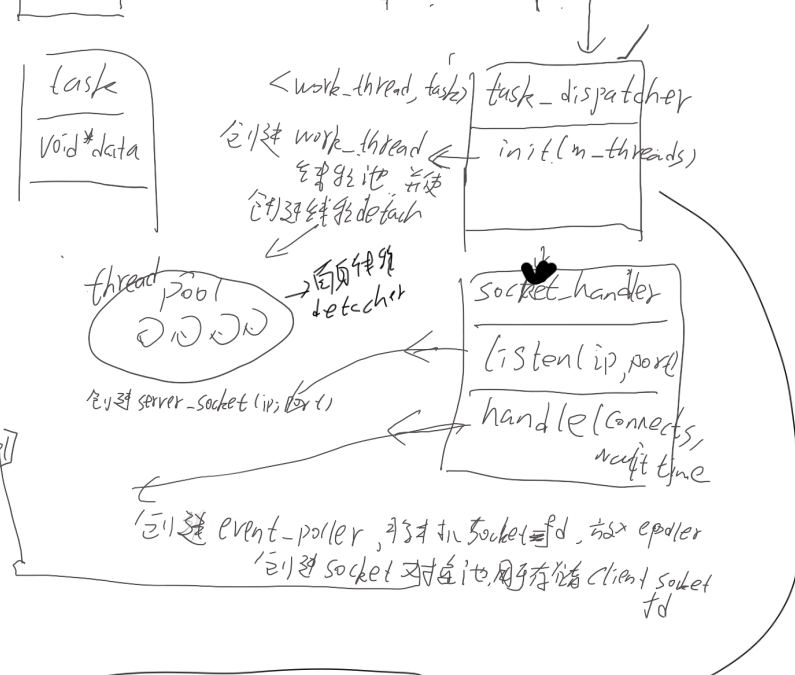
work_thread
m_cond wait()

set task, signal

task.run() → task → destroy() → 线程作为 空闲线程 归还到 thread pool

取从请求解析数据

通过 → 不通过 (内部进行, socket → close, 与与 socket 对象 归还到 socket 对象地)

无异常则取出
消息继续发送

socket_handler → remove (socketfd)
或将其重新送x epoller socket_handler → attach (socket)

input, ouput, work_cmd 引用 workflow

s.chetfd → send (ouput)

socket_handler → attach (socket) 进入 epoller

workflow→run
把握请更之 cmd, 找 work

work→run)) 运行 work下的
   所有 plugin, 如
   发生错误的地方
   追踪