详细解析 rtknavi.c 文件中的 main 函数以及它调用的 rtksvrstart 函数（rtksvrstart 的定义在 RTKLIB 的 rtksvr.c 中，但我们可以从 rtknavi.c 的调用中精确分析其输入输出）。

---

rtknavi.c: main() 函数解析

main 函数是 rtknavi 程序的**入口点**。它的核心任务是：

1. **定义默认配置**：在代码中硬编码（Hardcode）所有默认的处理选项、解算选项和数据流路径。
2. **初始化服务结构**：初始化 rtksvr_t 服务控制结构体。
3. **启动服务**：将所有配置参数传递给 rtksvrstart 函数，以启动后台处理线程。

1. 变量赋有效值的过程

main 函数中的变量赋值主要分为两类：

A. 静态初始化（编译时确定）

在函数开始时，几个关键的结构体和变量被“硬编码”赋予了默认值：

- solopt_t SolOpt = {...}:
    - **作用**：定义解算的**输出格式**选项。
    - **赋有效值**：
        - posf = SOLF_LLH：输出格式为经纬高 (LLH)。
        - times = TIMES_GPST：时间格式为 GPST。
- prcopt_t PrcOpt = {...}:
    - **作用**：定义 GNSS **处理算法**的核心选项。这是 pntpos 和 relpos（RTK 解算）的主要配置来源。
    - **赋有效值**：
        - mode = PMODE_KINEMA：定位模式为动态（Kinematic）。
        - navsys = 51：导航系统掩码，51 = 1 (GPS) + 2 (GLO) + 16 (GAL) + 32 (BDS)。
        - elmin = 15.0*D2R：卫星截止高度角为 15 度。
        - ionoopt = 1 (IONOOPT_BRDC)：电离层改正使用广播星历模型

（Klobuchar）。

- tropopt = 1 (TROPOPT_SAAS)：对流层改正使用 Saastamoinen 模型。
- err, std, prn, thresar 等数组：被赋予了用于噪声模型和模糊度解算的默认统计值。

- char* paths[8] = {...}:
  - **作用**：定义输入和输出数据流的文件路径。
  - **赋有效值**：
    - paths[0] (流动站): "data\\Rover_20240520_082407.rtcm3::T::x60"
    - paths[1] (基站): "data\\Base_Station_20240520_082407.rtcm3::T::x60"
    - paths[3] (解算结果): "data\\Solution_%Y%m%d-%h%M%S.log" (文件名将包含时间戳)。
    - ::T::x60 是文件流选项，表示时间戳从路径中获取（::T）并以 60 倍速播放（::x60）。
- int Format[] = {1, 1, 0, 3, ...}:
  - **作用**：定义 paths 对应流的数据格式。
  - **赋有效值**：
    - Format[0] = 1：流动站输入格式为 STRFMT_RTCM3。
    - Format[1] = 1：基站输入格式为 STRFMT_RTCM3。
    - Format[3] = 3：解算结果输出格式为 SOLF_LLH (与 SolOpt.posf 对应)。

B. 动态/运行时赋值（运行时确定）

在 rtksvrstart 被调用之前，main 函数会执行一些运行时的初始化和配置覆盖：

1. rtksvrinit(&rtksvr);：调用 rtksvrinit (在 rtksvr.c 中定义)，将 rtksvr 结构体的所有字段清零或设为默认值。
2. strinit(&monistr);：初始化 monistr（监视流）结构体。
3. stropen(...): 尝试在 52001 端口（或偏移端口）打开一个 TCP 服务器流，用于实时监控解算状态。

4. traceopen(TRACEFILE);：如果 DebugTraceF 大于 0，打开一个追踪日志文件。
5. PrcOpt.rovpos = POSOPT_POS;：**覆盖 PrcOpt**。设置流动站位置模式为 POSOPT_POS（在此配置中通常意味着使用 PrcOpt.ru 中的先验值，或者如果 pntpos 先运行则动态计算）。
6. PrcOpt.refpos = POSOPT_RTCM;：**覆盖 PrcOpt**。设置基站位置模式为 POSOPT_RTCM，这意味着程序将**等待 RTCM 消息（如 MT1005/1006）来获取基站坐标**，而不是使用 PrcOpt.rb 中的硬编码值。
7. strs 数组：根据 Stream 和 StreamC 的配置，动态设置 strs 数组，指定哪些流是活动的（例如 STR_FILE 或 STR_NONE）。
8. solopt[i] = SolOpt; 和 solopt[i].posf = Format[i+3];：将 main 函数开头的静态默认值 SolOpt 复制到 solopt 数组中，并用 Format 数组中的值更新其 posf（位置格式）字段。

2. 输入与输出

- **输入**：
  - rtknavi.c 的 main 函数没有命令行输入（argc, argv 未被使用）。
  - 所有配置均来自 rtknavi.c 文件内部的硬编码值。
- **输出**：
  - **返回值**：1（成功）或 0（失败）。
  - **主要动作**：调用 rtksvrstart，成功则启动后台处理线程。
  - **日志**：如果 DebugTraceF > 0，会输出一个 rtknavi_...trace 文件。

---

rtksvrstart 函数解析 (在 rtknavi.c 中被调用)

此函数（在 rtksvr.c 中定义，未提供）是启动 RTK 服务的核心。main 函数是“配置中心”，而 rtksvrstart 是“执行中心”，它负责接收所有配置，设置 rtksvr_t 结构，并启动真正的工作线程（rtksvrthread）。

1. 输入 (来自 main 函数的调用)

rtksvrstart 接收 main 函数准备好的所有变量作为参数：

- &rtksvr：指向已 rtksvrinit 但未配置的服务器结构体。
- SvrCycle (10), SvrBuffSize (32768)：服务器配置。

- strs：[STR_FILE, STR_FILE, STR_NONE, STR_FILE, ...] (流类型数组)
- paths：["data\\Rover...", "data\\Base_Station...", ...] (流路径数组)
- Format：[STRFMT_RTCM3, STRFMT_RTCM3, ...] (流格式数组)
- &PrcOpt：指向在 main 中配置好的**处理选项**结构体。
- solopt：指向在 main 中配置好的**解算输出选项**数组。
- &monistr：指向已打开的 TCP 监视流。
- errmsg：用于返回错误信息的缓冲区。
- (以及 NavSelect, cmds, rcvopts, NmeaCycle 等其他配置)

2. 变量赋有效值的过程 (在 rtksvrstart 内部的预期动作)

1. **保存配置**：rtksvrstart 首先会将 main 传递过来的 PrcOpt 和 solopt 指针的内容，**复制**到 rtksvr 结构体内部的 svr->prcopt 和 svr->solopt 成员中。这是 PrcOpt 最终传递给处理线程的方式。
2. **初始化流**：rtksvrstart 会循环 strs 和 paths 数组，为 rtksvr->stream[i]（最多 MAXSTRRTK 个流）设置路径、类型和模式（读/写）。
3. **初始化 RTCM 结构**：对于每个输入流（流动站和基站），它会调用 init_rtcm(&svr->rtcm[i])。
    - init_rtcm 内部（如 rtcm.c 所示）：会为 rtcm->obs.data (观测数据缓冲区)、rtcm->nav.eph (星历缓冲区) 和 rtcm->nav.geph (GLONASS 星历缓冲区) **分配内存**。
4. **初始化导航结构**：rtksvrstart 还会初始化 svr->nav（一个 nav_t 结构体），这是解算时所有卫星星历的中心存储库。
5. **启动线程**：最后，rtksvrstart 创建一个新线程（通常是 rtksvrthread），并将配置完毕的 rtksvr 结构体指针作为该线程的唯一参数。

3. 输出 (从 rtksvrstart 返回)

- **返回值**：返回一个 int 状态（1 表示线程启动成功，0 表示失败）。
- **主要动作（副作用）**：
    - 一个**后台处理线程** (rtksvrthread) 开始运行。
    - 这个线程现在接管了 rtksvr_t 结构体，它将：
        1. 循环从 svr->stream[0] (Rover) 和 svr->stream[1] (Base) 读取

数据。

2. 调用 input_rtcm3 将数据送入 svr->rtcm[0] 和 svr->rtcm[1]。
3. input_rtcm3 调用 decode_rtcm3，后者调用 decode_type1004 (观测值), decode_type1019 (GPS 星历), decode_type1005 (基站坐标) 等。
4. 解码后的观测值存入 svr->rtcm[i].obs.data，星历存入 svr->rtcm[i].nav.eph，基站坐标存入 svr->rtcm[i].sta.pos。
5. rtksvrthread 随后调用 pntpos (使用 svr->rtcm[0].obs.data 和 svr->prcopt) 计算初始位置。
6. rtksvrthread 接着调用 relpos (使用 svr->rtcm[0].obs.data, svr->rtcm[1].obs.data, svr->nav, svr->sta.pos 和 svr->prcopt) 计算 RTK 解。
7. relpos 内部会调用 lambda 进行模糊度固定。
8. 解算结果 sol 被写入 svr->stream[3] (日志文件)。