

依据 RTKLIB 的标准结构和您提供的 rtknavi.c、pntpos.c 等文件的上下文，详细解释这个线程的流程、输入、输出和关键变量。

rtksvrthread 函数解析

1. 总体结构与作用

- **函数原型：** void \*rtksvrthread(void \*arg)
- **作用：** 作为守护进程在后台持续运行。它实现了 **读-解码-同步-解算-写** 的完整 RTK/GNSS 处理循环。
- **输入：**
  - arg：类型为 void \*，实际传递的是一个指向 **rtksvr\_t 结构体**（即 rtknavi.c 中配置的 rtksvr 变量）的指针。
- **输出：**
  - 函数本身返回值通常为 NULL，因为这是一个不退出的无限循环线程。
  - 主要输出是**通过服务器结构体写入到外部数据流**（例如，将定位结果写入文件，如 rtknavi.c 中配置的 paths[3]）。

2. rtksvrthread 详细流程（核心循环）

rtksvrthread 函数主要包含一个无限循环，按配置的周期（在 rtknavi.c 中配置为 SvrCycle=10 毫秒）执行以下步骤：

步骤	流程描述	关键操作/子函数调用	输入变量（来自 rtksvr）	赋值有效值的变量
0. 准备	线程初始化： 线程启动后，首先从 arg 中取出 rtksvr_t 结构体指针，	stropen()	svr->stream (包含路径和类型)	svr->stream[i].fp (文件指针)

步骤	流程描述	关键操作/子函数调用	输入变量（来自 rtksvr）	赋值有效值的变量
	并执行必要的初始化（如打开所有配置的流）。			
1. 数据输入	<b>读取原始数据：</b> 循环读取所有活动的输入流（流动站、基站、修正数据等）。	strread()	svr->stream[i]	svr->raw[i] 或 svr->rtcm[i].buff (缓冲区)
2. 数据解码	<b>解码并存储数据：</b> 将读取到的原始字节数据解码成 GNSS 观测值和导航电文。	input_rtc3() / input_raw() (在 rtcm.c 中)	svr->rtcm[i].buff 或 svr->raw[i].buff	svr->rtcm[i].obs.data (观测值)  svr->rtcm[i].nav (导航电文/星历)  svr->rtcm[i].sta.pos (基站坐标)

步骤	流程描述	关键操作/子函数调用	输入变量（来自 rtksvr）	赋值有效值的变量
3. 数据同步	<b>观测值同步：</b> 检查流动站和基站是否有最新且时间上接近的观测数据，以及是否有最新的星历可用。	rtksyncedge() (自定义函数)	svr->rtcm[0].obs, svr->rtcm[1].obs	svr->obs[0], svr->obs[1] (解算用的观测值)  svr->nav (合并后的星历库)
4. 定位解算	<b>执行定位：</b> 如果数据同步成功，则开始定位解算。	<b>A. pntpos()</b> (在 pntpos.c 中)  <b>B. relpos()</b> (未在列表中，但依赖 lambda.c)	svr->obs[0] (流动站观测值)  svr->obs[1] (基站观测值)  svr->nav (星历)  svr->prcopt (处理选项)	svr->sol (解算结果)

步骤	流程描述	关键操作/子函数调用	输入变量（来自 rtksvr）	赋值有效值的变量
5. 结果输出	<b>写入解算结果：</b> 将计算出的最新解算结果写入到输出流。	out_stat() / out_sol()	svr->sol (最新解)  svr->soloapt (输出选项)  svr->rtk.sol (RTK 解算器的状态)	svr->stream[3] (解算日志文件)
6. 循环控制	<b>周期等待：</b> 根据配置的解算周期等待，然后进入下一个循环。	sleep(SvrCycle)	SvrCycle (10ms)	-

### 3. 关键变量的赋值和传递

在 rtksvrthread 内部，核心变量的传递和赋值是通过 **rtksvr\_t 结构体** 的成员完成的。

变量成员	赋值/初始化来源	传递/使用到的函数	变量的作用
svr->prcopt	<b>赋值：</b> 来自 main 函数传入的 PrcOpt，在 rtksvrstart 中	pntpos(): 控制 SPP 的改正模型和滤波	核心处理选项，如电离层模型、截止角、模式 (PMODE_KINEMA)。

变量成员	赋值/初始化来源	传递/使用到的函数	变量的作用
	被复制。	器参数。  relpos(): 控制 RTK 模糊度固定和解算 模式。	
svr->sol	<b>初始化：</b> 在 rtksvrinit 中清 零。  <b>赋值：</b> 由 pntpos() 和 relpos() 函数 修改。	pntpos(): <b>首次写入</b> SPP 解 (初始位置)。  relpos(): <b>覆盖写入</b> RTK 高精度解。	存储当前的定位结果 (位置 rr、速度 rr+3、状态 stat)。
svr->rtcm[i].obs.data	<b>赋值：</b> 由 rtcm.c 中的 input_rtc3 解 码后写入。	pntpos(): SPP 解算 的观测值 输入。  relpos(): RTK 解算 的观测值	存储原始的伪距、载 波相位和多普勒观测 值。

变量成员	赋值/初始化来源	传递/使用到的函数	变量的作用
		输入。	
svr->rtcm[i].nav	<b>赋值：</b> 由 RTCM 消息 (如 MT1019) 解码后写入。	pntpos(): 传递给 satpos() 计算卫星状态。	存储流动站/基站接收到的广播星历等导航数据。
svr->nav	<b>赋值：</b> 由 rtksvrthread 合并来自所有输入流的星历数据。	pntpos() / relpos(): 作为计算卫星位置时查找星历的最终数据源。	全局导航数据结构, 包含所有系统的最新星历。
svr->rtcm[1].sta.pos	<b>赋值：</b> 由基站 RTCM 消息 (如 MT1005/1006) 解码后写入。	relpos(): 作为相对定位的 <b>固定参考点</b> 。	基准站的精确三维坐标。

#### 4. 总结

rtksvrthread 函数是 RTKLIB 的**神经中枢**。它持续不断地从各种输入流中提取数据, 通过 input\_rtc3 等函数将原始数据转化为可用的观测值和星历, 然后交由 pntpos 和 relpos 进行定位解算, 最后将高精度的结果写入输出文件。所有数据的流动和状态的维护都围绕着其核心变量——**rtksvr\_t** 结构体进行。