按代码写作的顺序：

1. 引用文件
2. 声明一个navigator的函数
3. 析构navigator的函数
4. Global位置的更新
5. Gps位置的更新
6. 传感器位置的更新
7. HOME点位置的更新
8. 飞行器状态的更新
9. 飞行器的地面检测函数
10. 飞行器的控制模式更新
11. 参数更新
12. task\_main\_trampoline的跳板函数
13. task\_main()函数
14. start()
15. status()
16. publish\_position\_setpoint\_triplet()
17. get\_default\_acceptance\_radius()
18. get\_acceptance\_radius()
19. get\_altitude\_acceptance\_radius()
20. get\_cruising\_speed()
21. get\_cruising\_throttle()
22. get\_acceptance\_radius()
23. add\_fence\_point()
24. load\_fence\_from\_file()
25. usage()
26. navigator\_main()
27. publish\_mission\_result()
28. publish\_geofence\_result()
29. publish\_att\_sp()
30. set\_mission\_failure(const char\* reason)

调用关系：

Check geofence violation

Gps\_pos

Sensor\_combined

Param\_update

Control\_mode

Vehicle\_status

Land\_detect

Home\_pos

Vehicle\_command

global\_pos

Global\_pos

Gps\_pos

Sensor\_combined

Vehicle\_status

Land\_detect

Control\_mode

Home\_pos

Onboard\_mission

Offboard\_mission

Param\_update

Vehicle\_command

更新topics

更新global\_pos和vehicle\_command

start

!thread\_should\_exit

订阅并更新 topics

Load geofence

Task\_main

Navigator\_main

fencefile

fence

stop

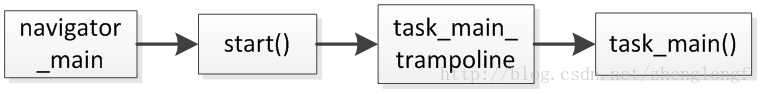
status

Task\_main\_trampoline

start

1. navigator线程是无人机自主飞行控制的核心所在，控制着导航模式、航点规划与控制模式的选择，其中主要包括自主起飞|、自主降落、自主返航、自主任务以及GPS失效保护等各个部分。该模块通过确定导航方式、地理围栏、失效保护等导航信号，最终确定目标航点，发送给位置控制器。位置控制器根据navigator指定的控制方式，实现对无人机位置的控制。

Navigator模块启动的流程如下。与其它线程一样都具有start、stop、status等供nuttx直接调用的接口。Task—main中包含着模块的主要流程。



navigator的主要流程图如下。地理围栏中包含预先设定好的位置点，可以划定安全飞行区域，避免无人机意外飞出计划空域。订阅信息来自urob提供的信息，不断的循环往复，处理不断更新的topics。不同的导航模式有不同的生成目标航点的方式，在选定导航模式后，通过\_navigation\_mode\_array中对应的run()函数运行相应的导航模式。

