ITTI工具

1. 初始化

**int itti\_init(task\_id\_t task\_max, thread\_id\_t thread\_max, MessagesIds messages\_id\_max,**

**const task\_info\_t \*tasks\_info, onst message\_info\_t \*messages\_info)**

参数：

Task\_id\_T task\_max 传入最大任务数

const task\_info\_t \*tasks\_info 传入初始化信息结构体

其他参数 ：未使用

功能：

全局遍历 使用tasks[].admin 信息存入

{

循环遍历初始化任务，为每一个任务都创建epoll句柄和信号量描述符

tasks[i].epoll\_fd = epoll\_create1(0)

创建一个非0的创建epoll句柄

tasks[i].sem\_fd = eventfd(0, EFD\_SEMAPHORE)

参数EFD\_SEMAPHORE创建一个类似信号量的描述符 0为计数值初始值。

itti\_subscribe\_event\_fd((task\_id\_t)i, tasks[i].sem\_fd);

将sem\_fd添加入epoll句柄 集合中

}

**void itti\_subscribe\_event\_fd(task\_id\_t task\_id, int fd)**

参数 ：

task\_id\_t task\_id 输入任务id

int fd 输入需要添加入任务epoll句柄的描述符

功能：初始化监听的描述符加入任务唯一的epoll句柄

event.events = EPOLLIN | EPOLLERR;//设置描述符可读和描述符出错时触发

1. 使用ITTI数据发送

**int itti\_send\_msg\_to\_task(task\_id\_t destination\_task\_id, instance\_t instance, MessageDef \*message)**

参数：

task\_id\_t destination\_task\_id 目标任务ID

instance\_t instance

MessageDef \*message 发送信息

功能：

加锁

调用int ret=itti\_send\_msg\_to\_task\_locked(destination\_task\_id, instance, message);发送数据的主要函数

解锁

**static inline int itti\_send\_msg\_to\_task\_locked(task\_id\_t destination\_task\_id, instance\_t instance, MessageDef \*message)**

参数 ：

task\_id\_t destination\_task\_id 目标任务ID

instance\_t instance

MessageDef \*message 发送信息

功能:

t->message\_queue.insert(t->message\_queue.begin(), message);

Task-> message\_queue 添加消息使用c++标准库中的vector进行存储

write(t->sem\_fd, &sem\_counter, sizeof(sem\_counter))

将信号量描述符写入信号通知监听函数通过write函数

1. 使用ITTI数据接受

**void itti\_receive\_msg(task\_id\_t task\_id, MessageDef \*\*received\_msg)**

参数：

task\_id\_t task\_id 需要监听的任务ID号

MessageDef \*\*received\_msg 获取任务消息地址

功能：

加锁

当 t->message\_queue.empty() 消息队列为空的时候

执行itti\_get\_events\_locked(task\_id, &t->events);

进行阻塞

当获得消息时会通过阻塞 执行\*received\_msg=t->message\_queue.back();获取消息队列尾部信息

再通过t->message\_queue.pop\_back();去除该消息

解锁

**static inline int itti\_get\_events\_locked(task\_id\_t task\_id, struct epoll\_event \*\*events)**

函数

task\_id\_t task\_id 任务ID

struct epoll\_event \*\*events 触发事件的消息描述符集合

功能：

通过t->nb\_events = epoll\_wait(t->epoll\_fd,t->events,t->nb\_fd\_epoll, epoll\_timeout);

监听epoll\_fd是否有事件触发，epoll\_timeout=-1时如果没有消息触发的时候该函数是阻塞态的

但触发到消息的时候会将触发的epoll事件集存入t->events结构体中，最后放回监听到的信号数量t->nb\_events；

通过遍历 t->events 结构体数组 t->nb\_events次后

通过read (t->sem\_fd, &sem\_counter, sizeof(sem\_counter))去接受获得到的信号量，并且计数值再底层自己实现

t->events[i].events &= ~EPOLLIN;通过清除该标志位