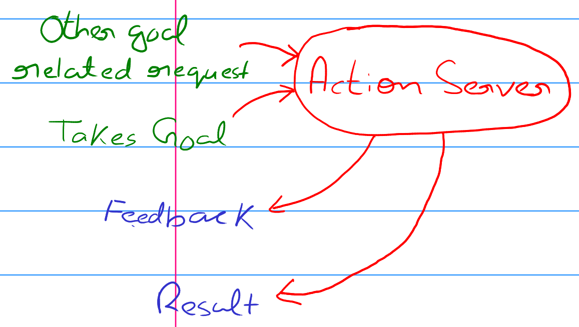
Actionlib tutorial

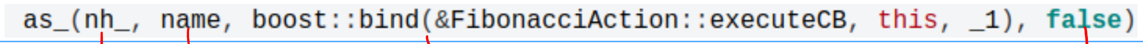
(1) Writing a Simple action Server using Execute callback



Fibonacci number :

0 1 2 3 4 5 6 7 . . .

1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 . . .



```
as_(nh_, name, boost::bind(&FibonacciAction::executeCB, this, _1), false)
```

node handle

name of action

function pointer

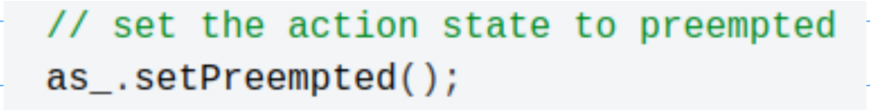
You need to address callbacks manually (auto_Start)

as_.Start( );

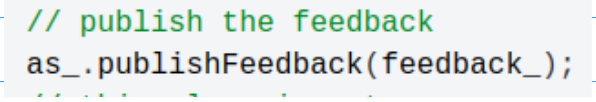
Explicitly start the action server, used if auto_Start is set to false.

as_.isPreemptRequested()

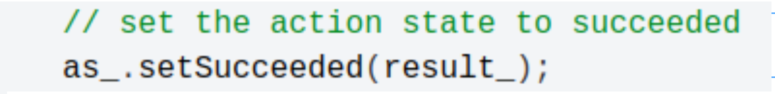
Allow polling implementation to query about preempt request.



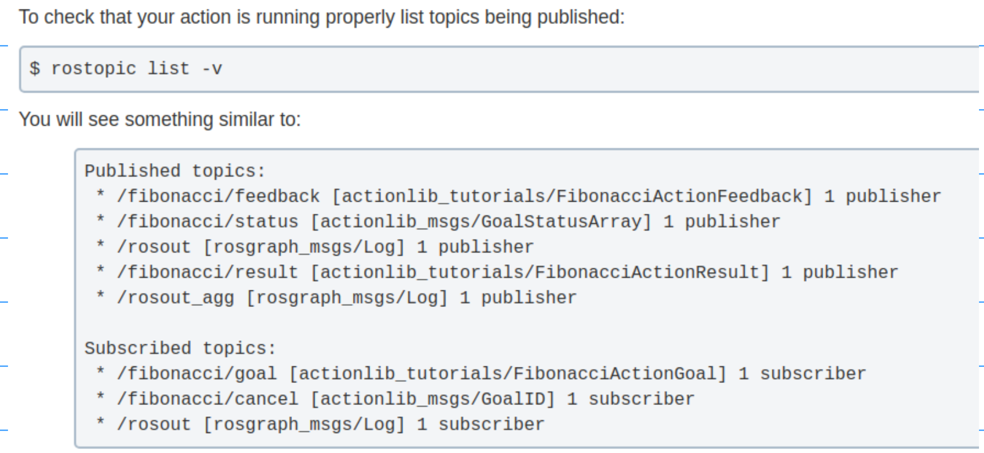
```
// set the action state to preempted
as_.setPreempted();
```



```
// publish the feedback
as_.publishFeedback(feedback_);
```



```
// set the action state to succeeded
as_.setSucceeded(result_);
```



To check that your action is running properly list topics being published:

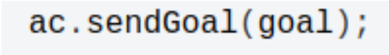
```
$ rostopic list -v
```

You will see something similar to:

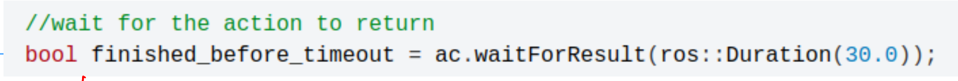
```
Published topics:
 * /fibonacci/feedback [actionlib_tutorials/FibonacciActionFeedback] 1 publisher
 * /fibonacci/status [actionlib_msgs/GoalStatusArray] 1 publisher
 * /rosout [rosgraph_msgs/Log] 1 publisher
 * /fibonacci/result [actionlib_tutorials/FibonacciActionResult] 1 publisher
 * /rosout_agg [rosgraph_msgs/Log] 1 publisher

Subscribed topics:
 * /fibonacci/goal [actionlib_tutorials/FibonacciActionGoal] 1 subscriber
 * /fibonacci/cancel [actionlib_msgs/GoalID] 1 subscriber
 * /rosout [rosgraph_msgs/Log] 1 subscriber
```

2 Writing a Simple action Client

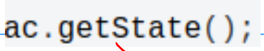


```
ac.sendGoal(goal);
```



```
//wait for the action to return
bool finished_before_timeout = ac.waitForResult(ros::Duration(30.0));
```

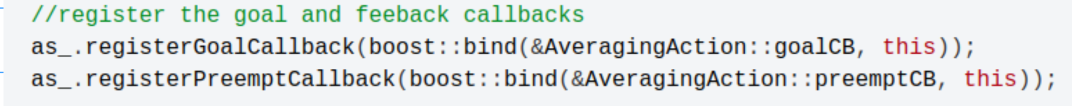
The timeout on the wait is set to 30 seconds, this means after 30 seconds the function will return with false if the goal has not finished.



```
ac.getState();
```

Returns string info about the state.

3 Writing a Simple action server using the goal callback method



```
//register the goal and feeback callbacks
as_.registerGoalCallback(boost::bind(&AveragingAction::goalCB, this));
as_.registerPreemptCallback(boost::bind(&AveragingAction::preemptCB, this));
```