**1. Tutorial membuat workspace**

Workspace adalah repository/folder yang dibuat dengan maksud sebagai tempat kegiatan ang terkait dengan ROS berjalan. Semua kegiatan ROS harus berada dalam satu workspace.

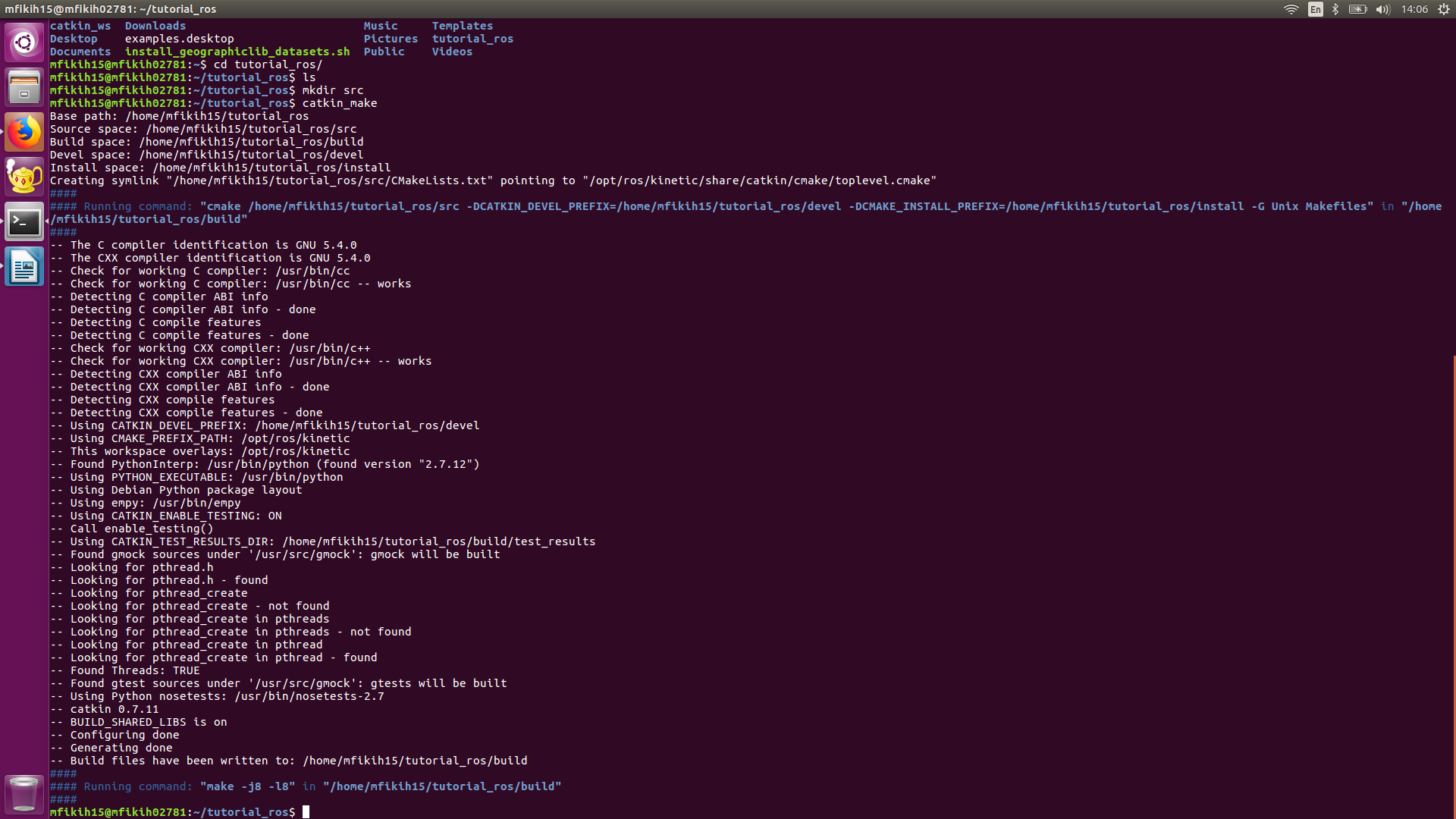
~$ mkdir tutorial\_ros ;membuat repo/folder <nama bebas>

~$ cd tutorial\_ros

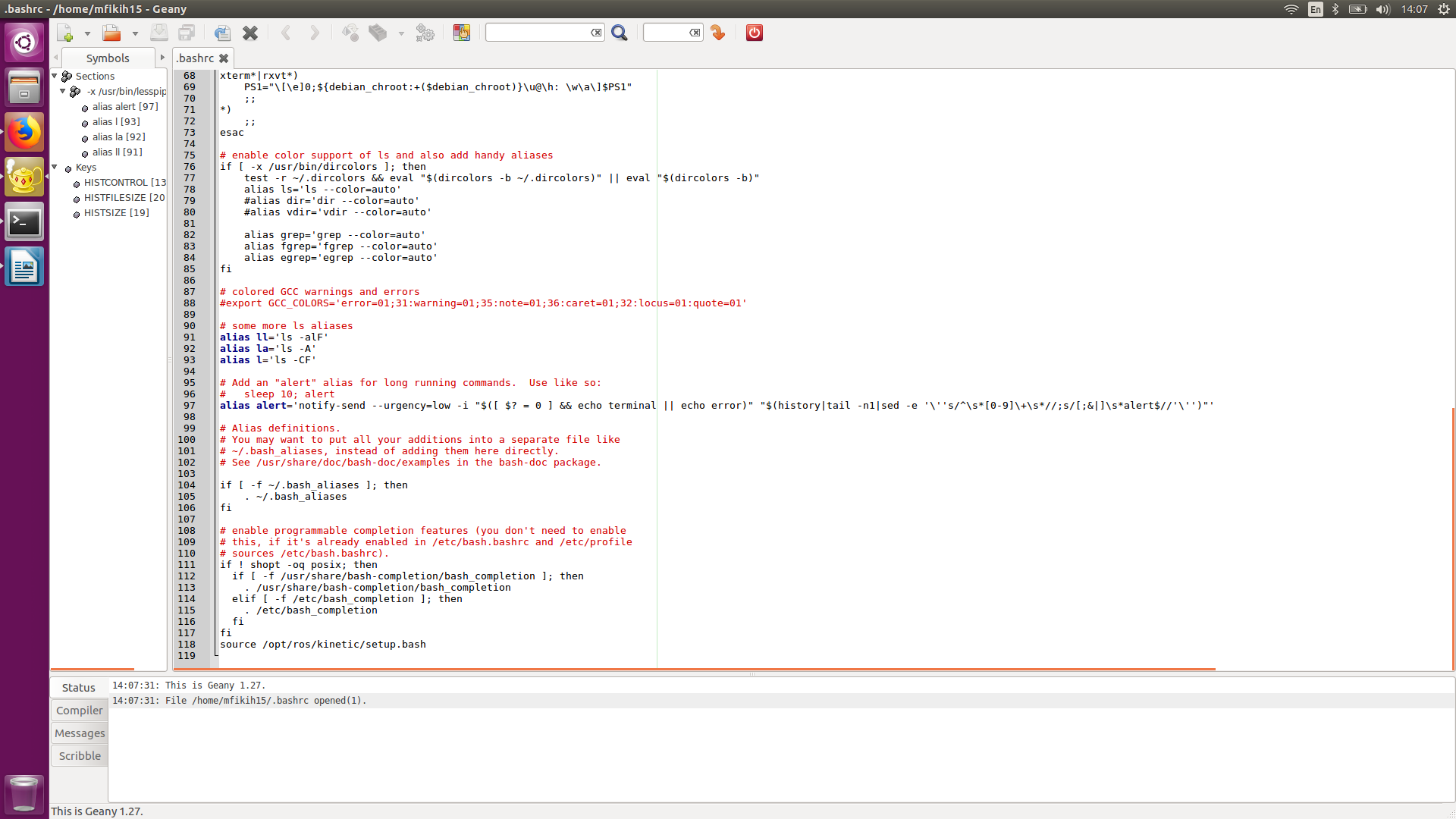
~/tutorial\_ros$ mkdir src ;buat folder src, nama folder harus src

~$ catkin\_make ;compiler

akan muncul tampilan sebagai berikut yang menandakan workspace berhasil dibuat

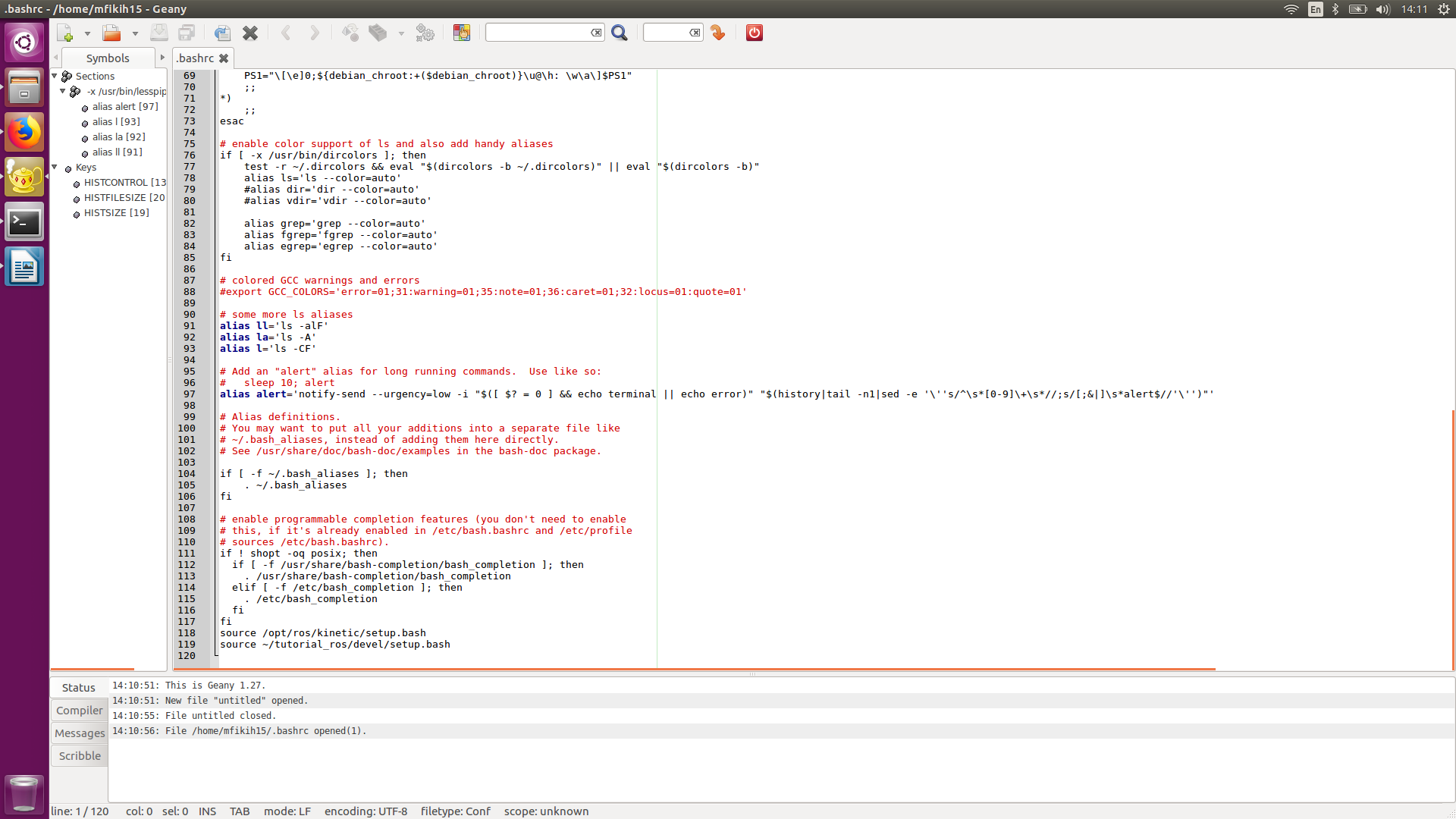


berikutnya memberitahukan bahwa kita memiliki workspace didalam ubuntu kita, caranya dengan membuka bashrc

~$ geany ~/.bashrc ;membukanya bisa menggunakan geany atau gedit

kemudian pada baris paling bawah tambahkan lokasi workspace yang dibuat ditambahkan lokasi devel/setup.bash

source ~/tutorial\_ros/devel/setup.bash



terakhir adalah restart terminal

**2. Membuat package pada workspace**

package adalah folder didalam workspace tempat dimana program berjalan.

Masuk kedalam workspace yang sudah dibuat pada tutorial 1

~$ cd tutorial\_ros/src

buat package dengan command

catkin\_create\_pkg <nama package> <depedency1> <depedency2> <depedency3>

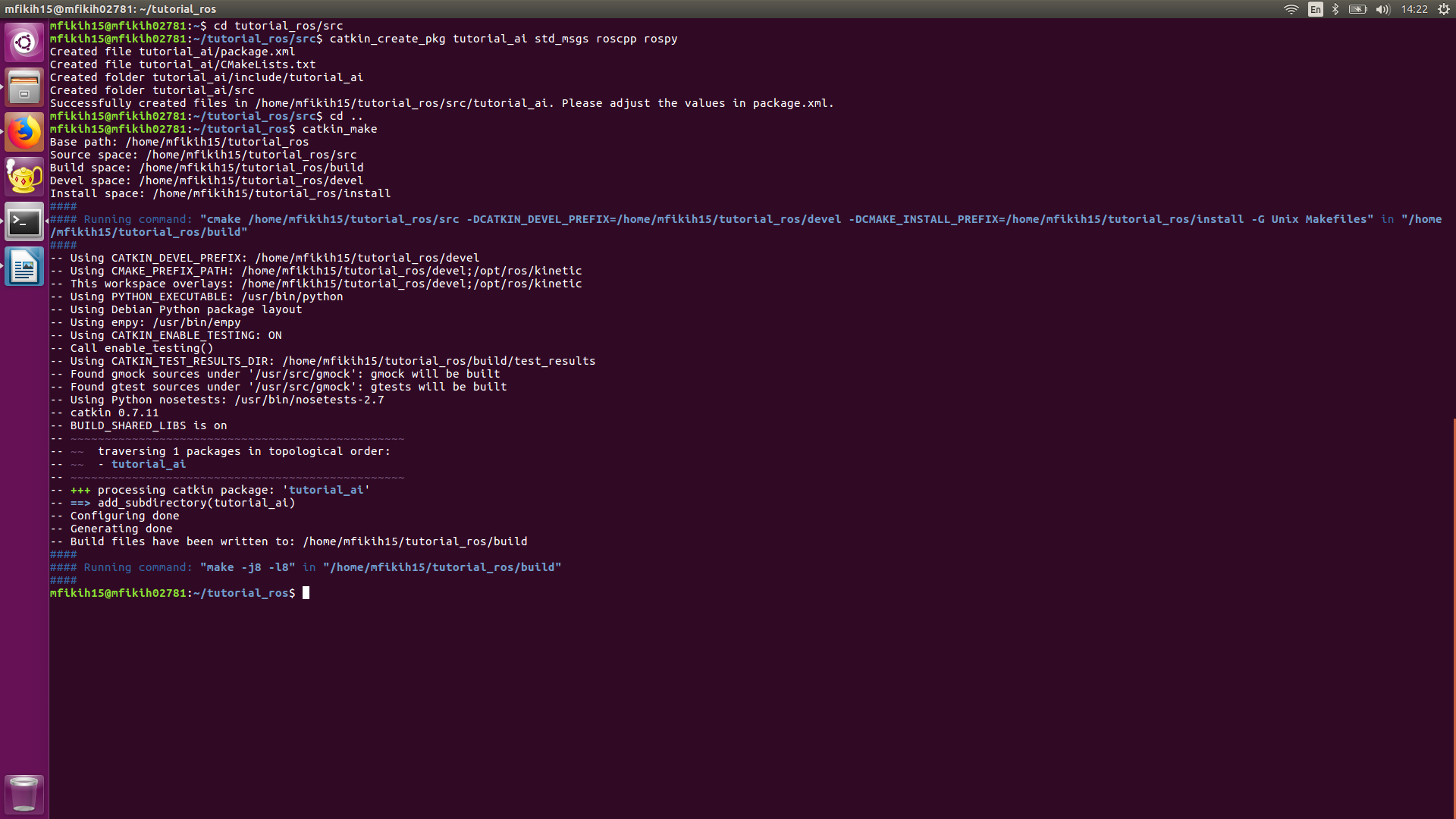
~/tutorial\_ros/src$ catkin\_create\_pkg tutorial\_ai std\_msgs roscpp rospy

kemudian kembali ke workspace dan compile dengan catkin make

~/tutorial\_ros/src$ cd ..

~/tutorial\_ros$ catkin\_make

jika berhasil maka akan muncul tampilan berikut



**3. Membuat simple publisher and subscriber**

publisher dan subscriber adalah metode pengiriman data antar codingan pada ROS

istilah yang harus diketahui :

- node : merupakan codingan yang dijalankan. Dalam tutorial ini nodenya ada 2 yaitu talker dan listener

- topic : merupakan topik yang dijadikan jalur pesan. Dalam tutorial ini topiknya adalah “chatter”. Topik antar node harus sama jika ingin berkomunikasi

- message : pesan yang digunakan pada topik. Dalam tutorial ini menggunakan std\_msgs/String berisi “hello world”

masuk kedalam folder src pada package yang sudah dibuat dan buat source code bernama talker.cpp dan listener.cpp

~$ cd tutorial\_ros/src/tutorial\_ai/src

~/tutorial\_ros/src/tutorial\_ai/src$ geany talker.cpp listener.cpp

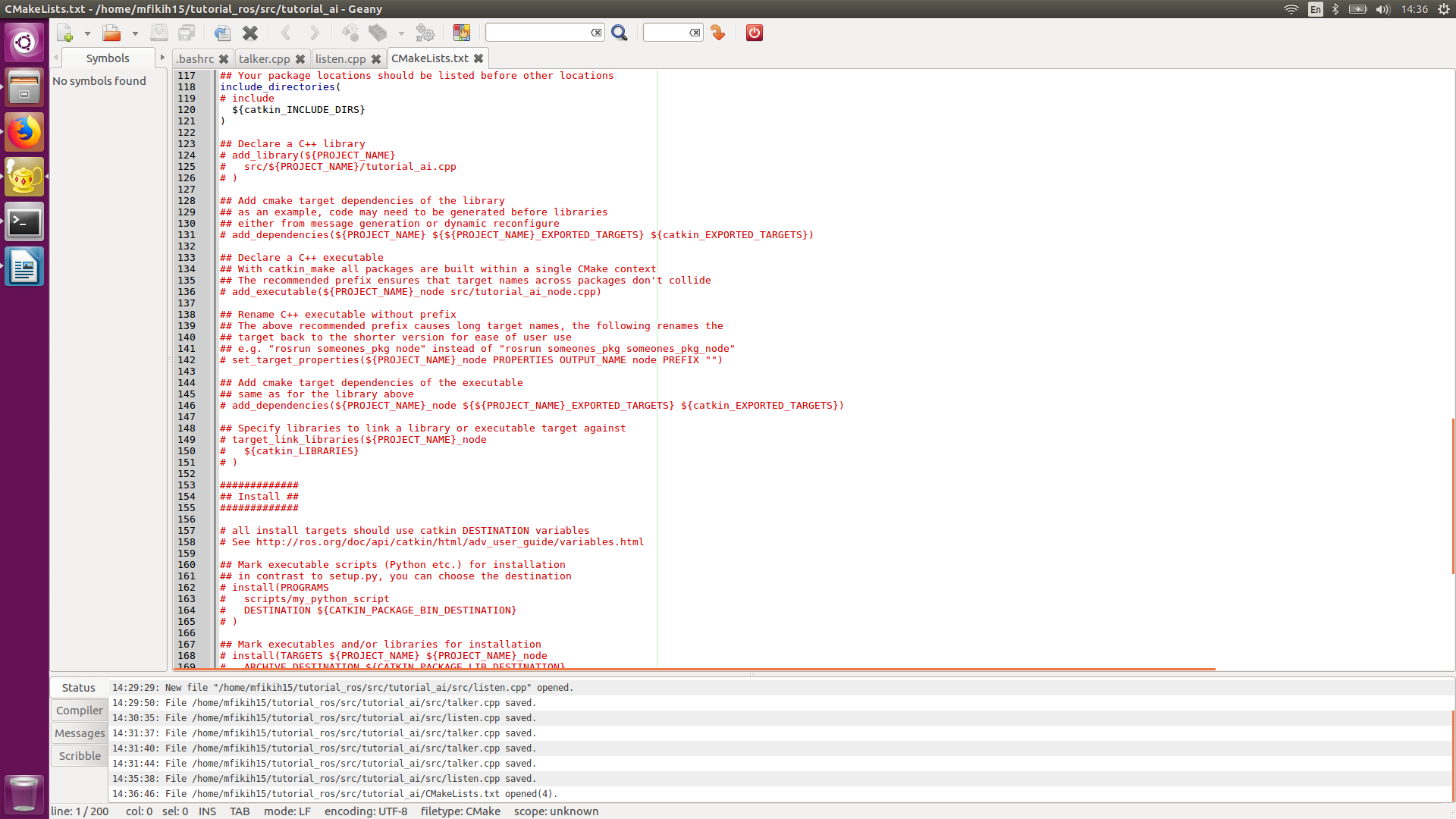
pada talker.cpp isis source code sebagai berikut :

pada listen.cpp isi source code sebagai berikut

kemudian buka file Cmakelists.txt pada /<package>/src/Cmakelists.txt

~/tutorial\_ros/src/tutorial\_ai/src$ cd ..

~/tutorial\_ros/src/tutorial\_ai$ geany CMakeLists.txt

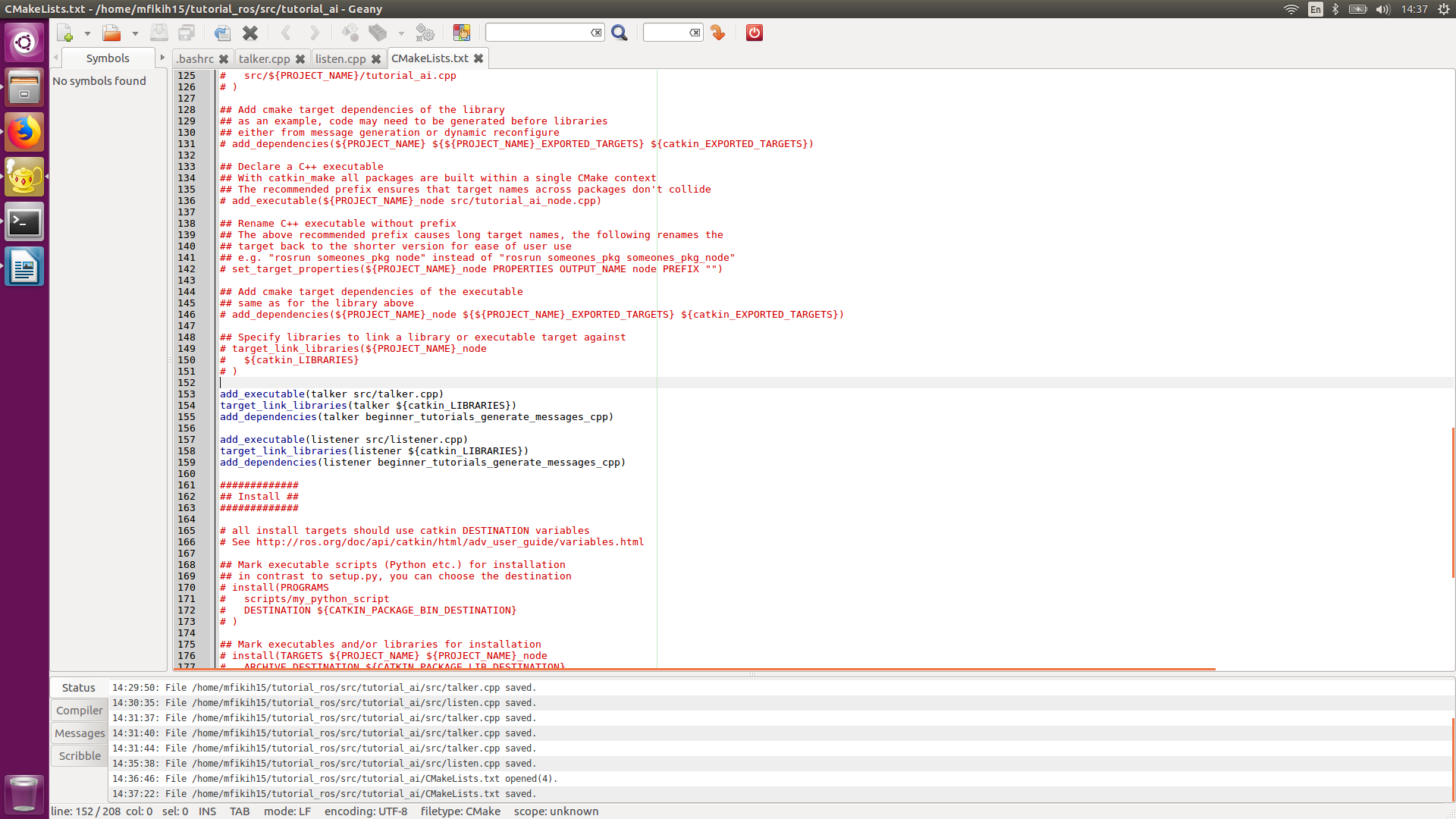


pada bagian diatas tulisan INSTALL, isi codingan berikut

add\_executable(talker src/talker.cpp)

target\_link\_libraries(talker ${catkin\_LIBRARIES})

add\_executable(listener src/listener.cpp)

target\_link\_libraries(listener ${catkin\_LIBRARIES})

kembali pada workspace dan compile

~/tutorial\_ros/src/tutorial\_ai$ cd ../..

~/tutorial\_ros$ catkin\_make

jika berhasil akan muncul tampilan berikut

Base path: /home/mfikih15/tutorial\_ros

Source space: /home/mfikih15/tutorial\_ros/src

Build space: /home/mfikih15/tutorial\_ros/build

Devel space: /home/mfikih15/tutorial\_ros/devel

Install space: /home/mfikih15/tutorial\_ros/install

####

#### Running command: "make cmake\_check\_build\_system" in "/home/mfikih15/tutorial\_ros/build"

####

-- Using CATKIN\_DEVEL\_PREFIX: /home/mfikih15/tutorial\_ros/devel

-- Using CMAKE\_PREFIX\_PATH: /home/mfikih15/tutorial\_ros/devel;/opt/ros/kinetic

-- This workspace overlays: /home/mfikih15/tutorial\_ros/devel;/opt/ros/kinetic

-- Using PYTHON\_EXECUTABLE: /usr/bin/python

-- Using Debian Python package layout

-- Using empy: /usr/bin/empy

-- Using CATKIN\_ENABLE\_TESTING: ON

-- Call enable\_testing()

-- Using CATKIN\_TEST\_RESULTS\_DIR: /home/mfikih15/tutorial\_ros/build/test\_results

-- Found gmock sources under '/usr/src/gmock': gmock will be built

-- Found gtest sources under '/usr/src/gmock': gtests will be built

-- Using Python nosetests: /usr/bin/nosetests-2.7

-- catkin 0.7.11

-- BUILD\_SHARED\_LIBS is on

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

-- ~~ traversing 1 packages in topological order:

-- ~~ - tutorial\_ai

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

-- +++ processing catkin package: 'tutorial\_ai'

-- ==> add\_subdirectory(tutorial\_ai)

-- Configuring done

-- Generating done

-- Build files have been written to: /home/mfikih15/tutorial\_ros/build

####

#### Running command: "make -j8 -l8" in "/home/mfikih15/tutorial\_ros/build"

####

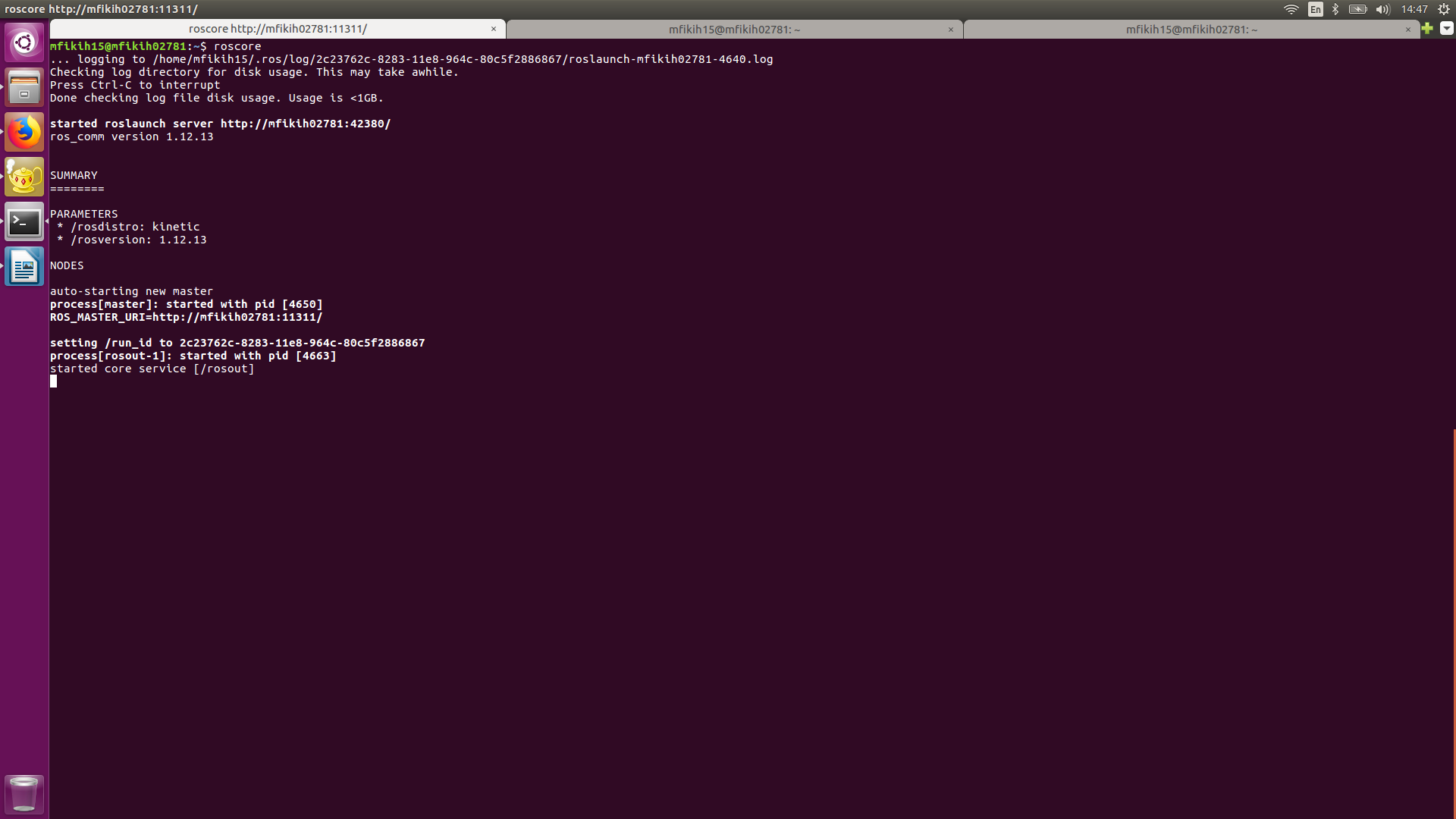
[100%] Built target listener

[100%] Built target talker

untuk menjalankannya, kembali ke home

~/tutorial\_ros$ cd ~

~$ roscore

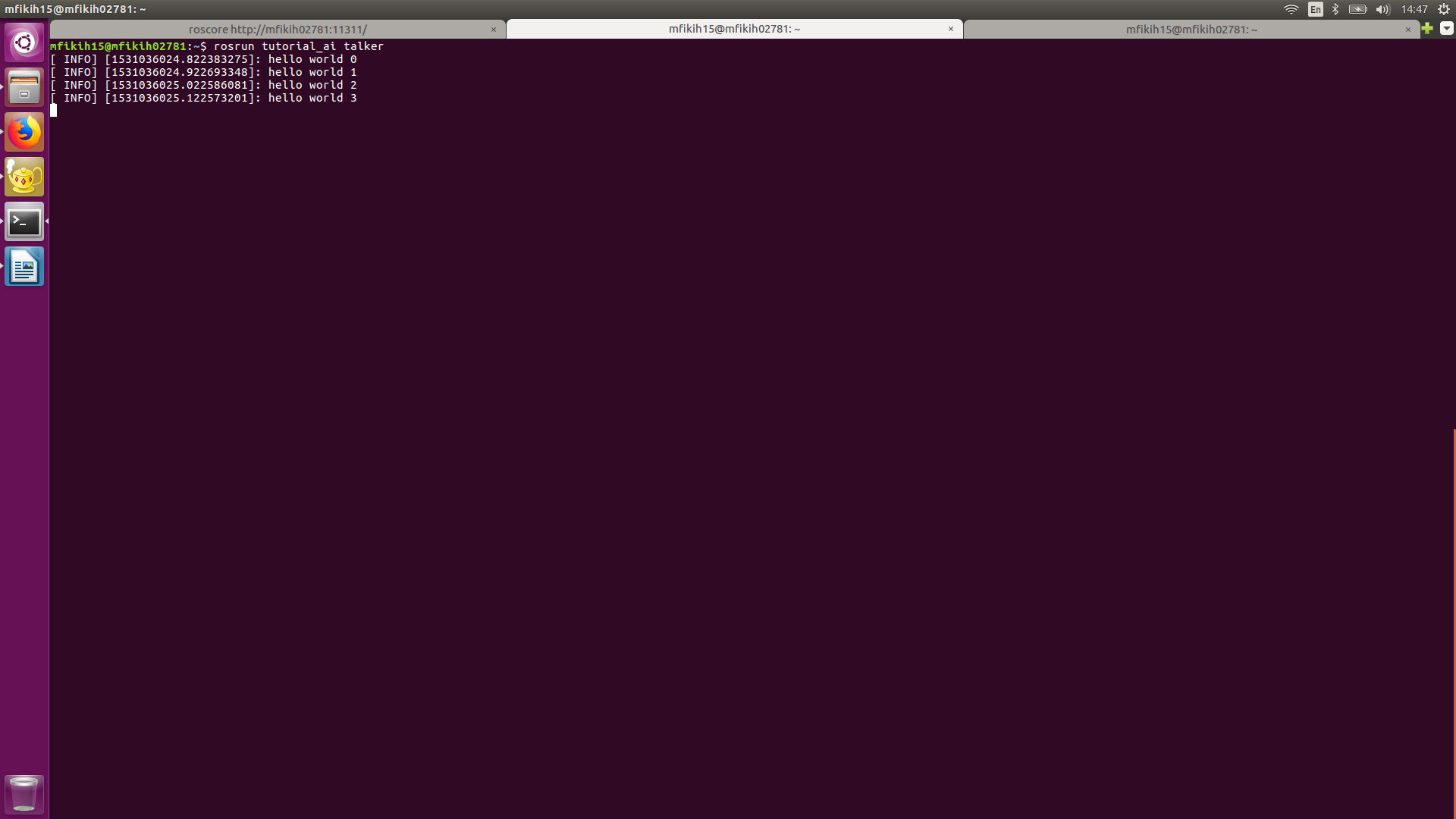


buka terminal baru

masukkan perintah

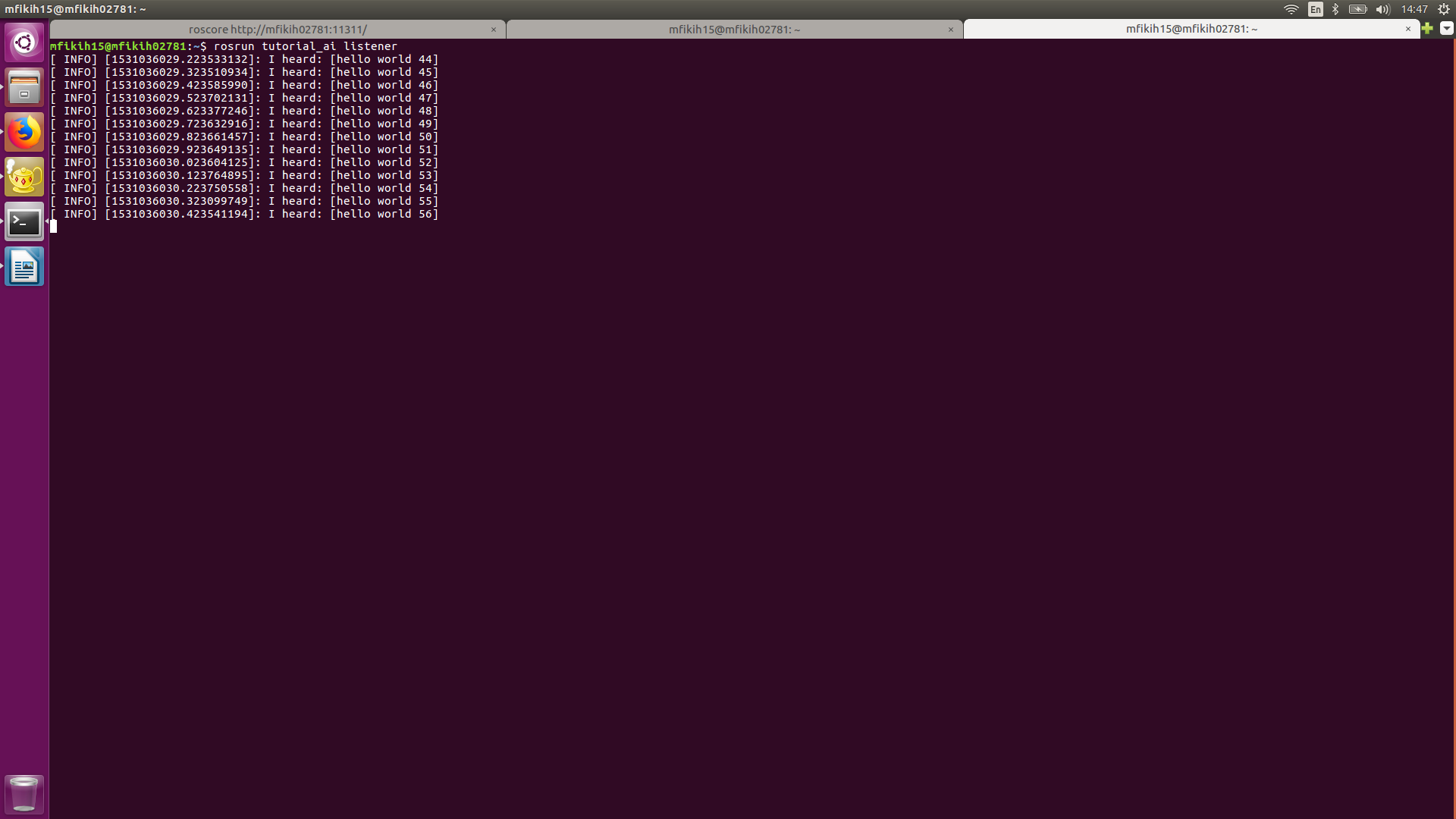
rosrun <nama package> <nama node>

~$ rosrun tutorial\_ai talker



buka terminal baru

~$ rosrun tutorial\_ai listener



**Penjelasan kodingan**

**pada Cmakelists.txt**

add\_executable(talker src/talker.cpp)

target\_link\_libraries(talker ${catkin\_LIBRARIES})

add\_executable(<nama node> <lokasi codingan>)

target\_link\_libraries(<nama node> $<{<compiler>})

nama node sebaiknya dibuat sama dengan nama kodingan

**pada talker.cpp**

#include "ros/ros.h"

#include "std\_msgs/String.h"

#include <sstream>

- ini adalah preprocessor, selalu gunakan ros/ros.h

- std\_msgs/String.h adalah standar message yang disediakan oleh ros untuk mengirim data berupa string

- include <sstream> untuk memasukkan string kedalam message

int main(int argc, char \*\*argv){

ros::init(argc, argv, "talker");

ros::NodeHandle n;

menginisialisasi codingan sebagai bagian dari ROS dan membuat node handle bernama n

ros::Publisher chatter\_pub = n.advertise<std\_msgs::String>("chatter", 1000);

membuat publisher dengan nama chatter\_pub, node handle n, message, topic, data antrian

ros::Publisher <nama publisher> = <node handle>.advertise<<message>>(<topic>,<data antrian>)

ros::Rate loop\_rate(10);

memberi delay pada program, 10 Hz

while (ros::ok()){

bernilai true saat ros berjalan

std\_msgs::String msg;

membuat objek bernama msg, isi defaultnya “data”

std::stringstream ss;

ss << "hello world " << count;

membuat objek bernama ss dengan isi hello world

msg.data = ss.str();

mengisi member data pada msg dengan isi pada ss

ROS\_INFO("%s", msg.data.c\_str());

fungsinya mirip seperti printf()

chatter\_pub.publish(msg);

mempublish objek msg menggunkan publisher

ros::spinOnce();

menjalankan node sebanyak satu kali

loop\_rate.sleep();

menjalakan delay yang sudah dibuat

++count;

}

return 0;

}

**pada listener.cpp**

#include "ros/ros.h"

#include "std\_msgs/String.h"

sama seperti pada talker

void chatterCallback(const std\_msgs::String::ConstPtr& msg){

ROS\_INFO("I heard: [%s]", msg->data.c\_str());

}

respon yang diberikan oleh subscriber jika dia menerima topik

chatterCallback(const <message>ConstPtr& <objek message>)

responnya berupa tulisan pada terminal:

I heard: hello world

int main(int argc, char \*\*argv)

{

ros::init(argc, argv, "listener");

ros::NodeHandle n;

sama seperti pada talker

ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);

membuat subscriber dimana jika dia menerima topik yang sama maka akan menjalankan chatterCallback

ros:Subscriber <nama subscriver> = <node handle>.subscriber(<topic>,<data antrian>,<nama callback>)

ros::spin();

menjalankan node sampai node dimatikan

return 0;

}

**4. membuat custom messages**

messages yang biasa digunakan di ROS adalah std\_msgs (common messages). Namun user dapat membuat sendiri messagenya sesuai dengan kebutuhan dari user itu sendiri.

1. buka workspace yang sudah dibuat sebelumnya, masuk kedalam package dan buat repo bernama “msg”

~$ cd tutorial\_ros/src/tutorial\_ai

~/tutorial\_ros/src/tutorial\_ai$ mkdir msg

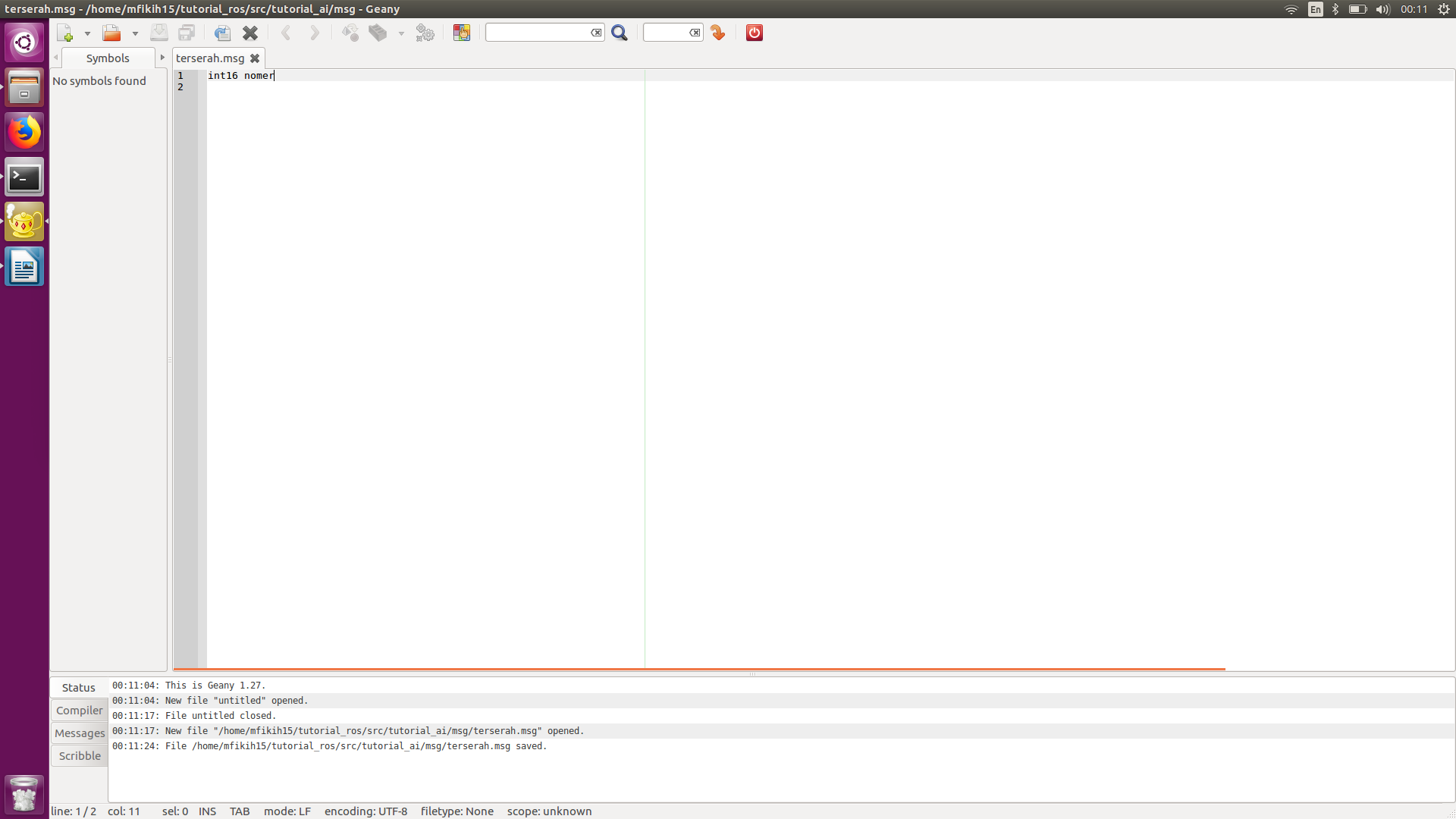
buat file dengan format <apapun>.msg pada repo msg

~/tutorial\_ros/src/tutorial\_ai$ cd msg

~/tutorial\_ros/src/tutorial\_ai/msg$ geany terserah.msg

isi dengan

int16 nomer



kemudian buka repo Cmakelists.txt dan package.xml di package

~/tutorial\_ros/src/tutorial\_ai/msg$ cd ..

~/tutorial\_ros/src/tutorial\_ai$ geany CMakelists.txt package.xml

pada Cmakelists

isi message\_generation pada find\_package

isi terserah.msg pada generate message

isi CATKIN\_DEPENDS message\_runtime pada catkin\_package

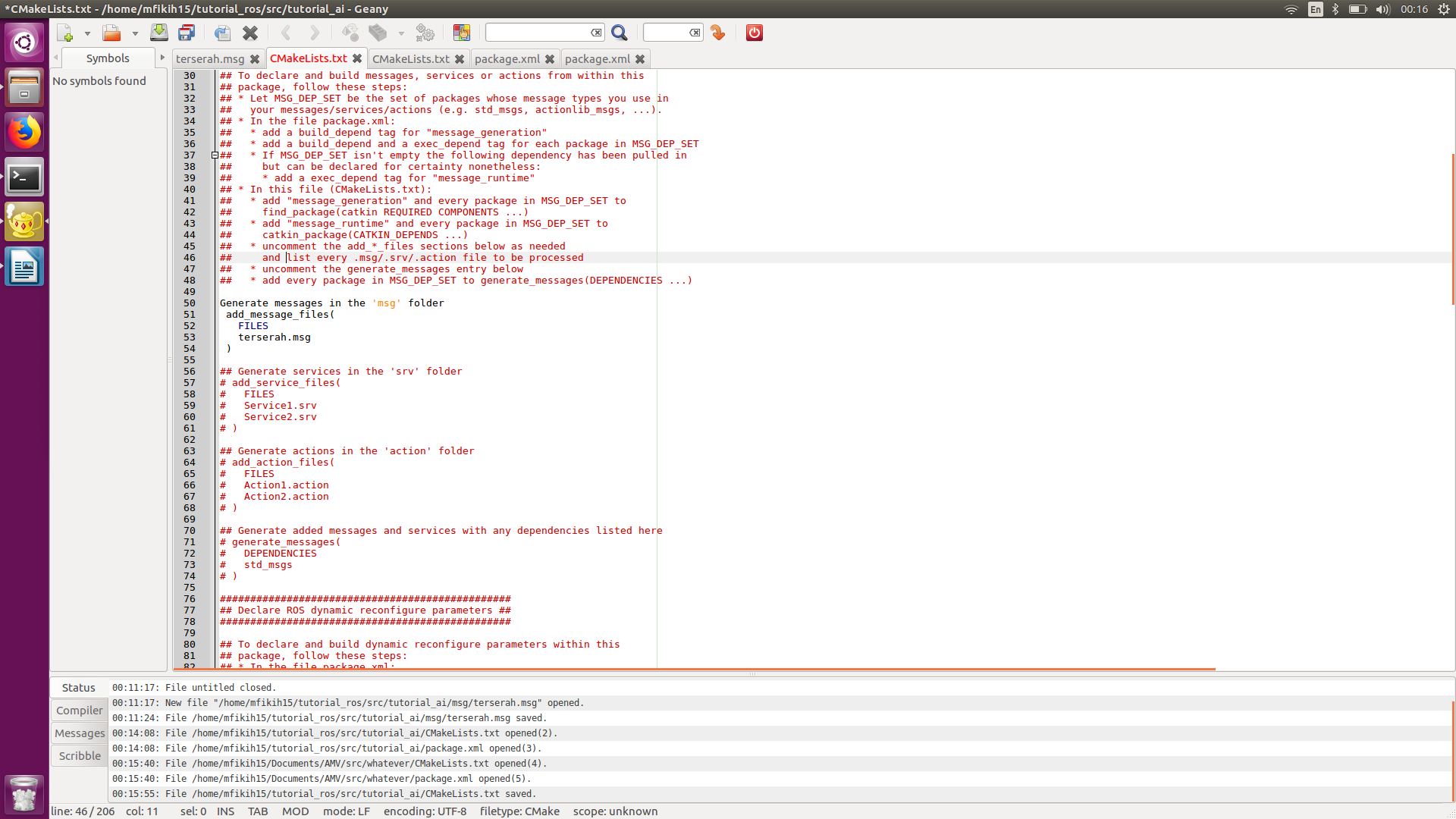
isi ini juga

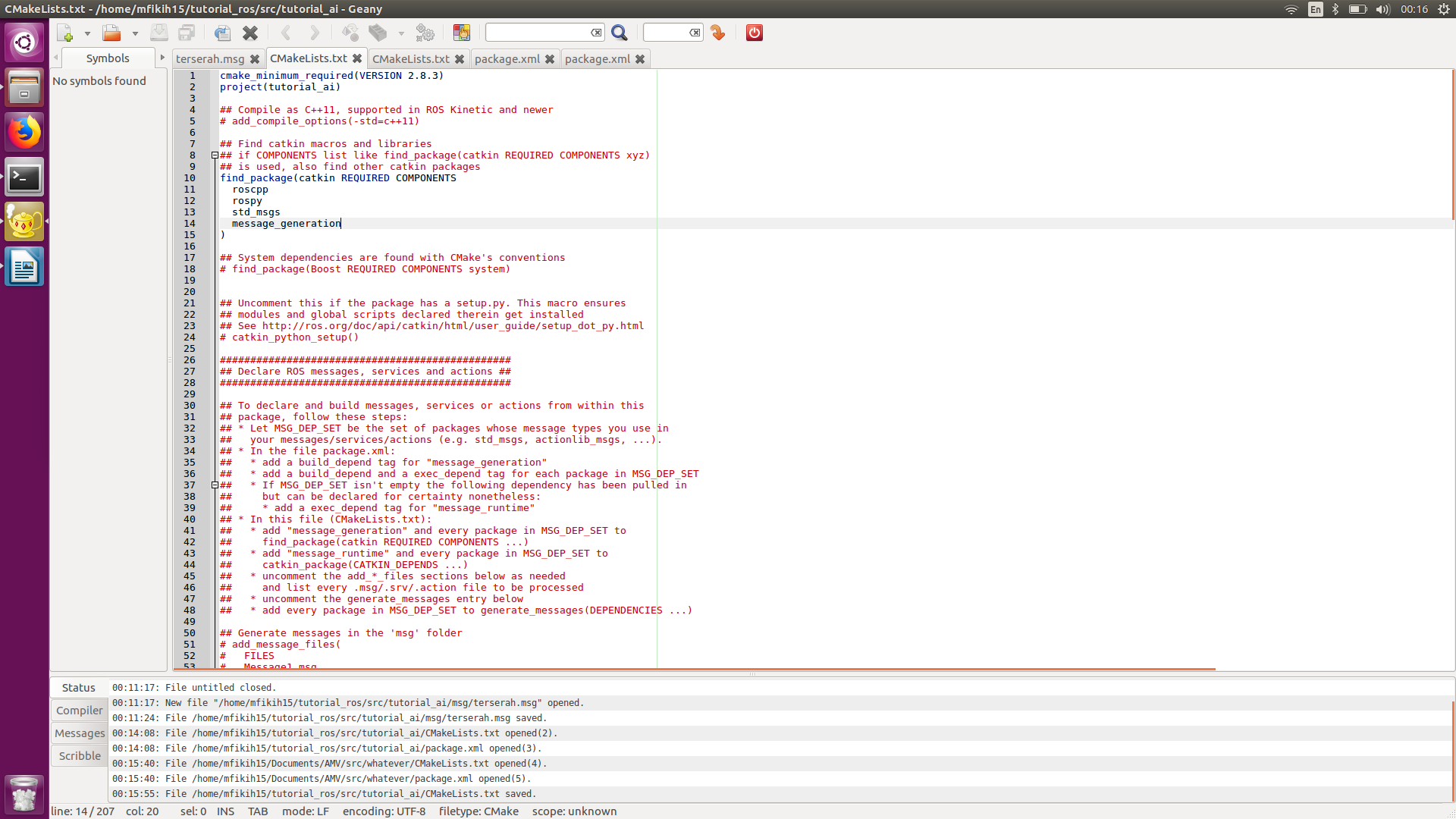
generate\_messages(

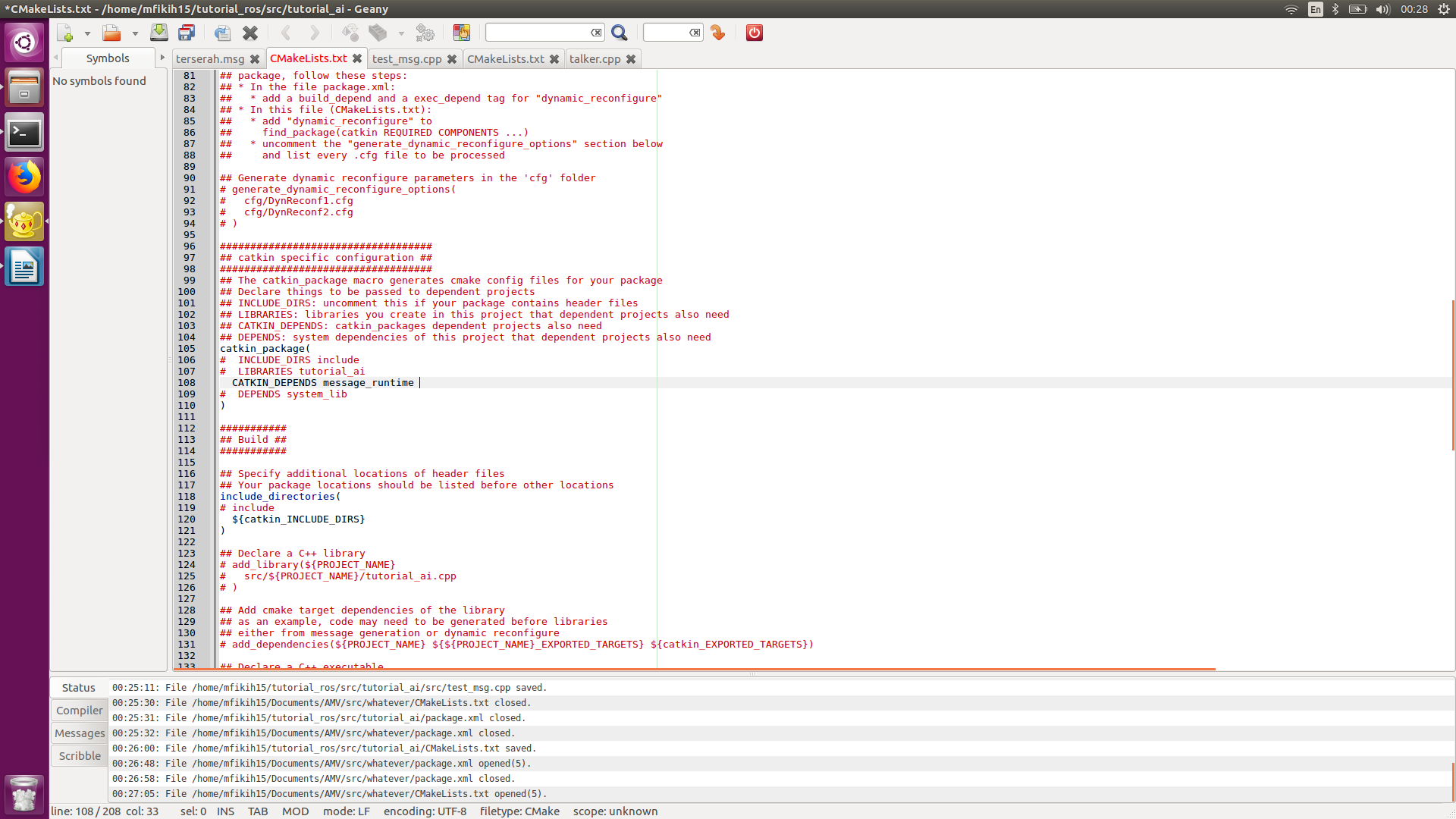
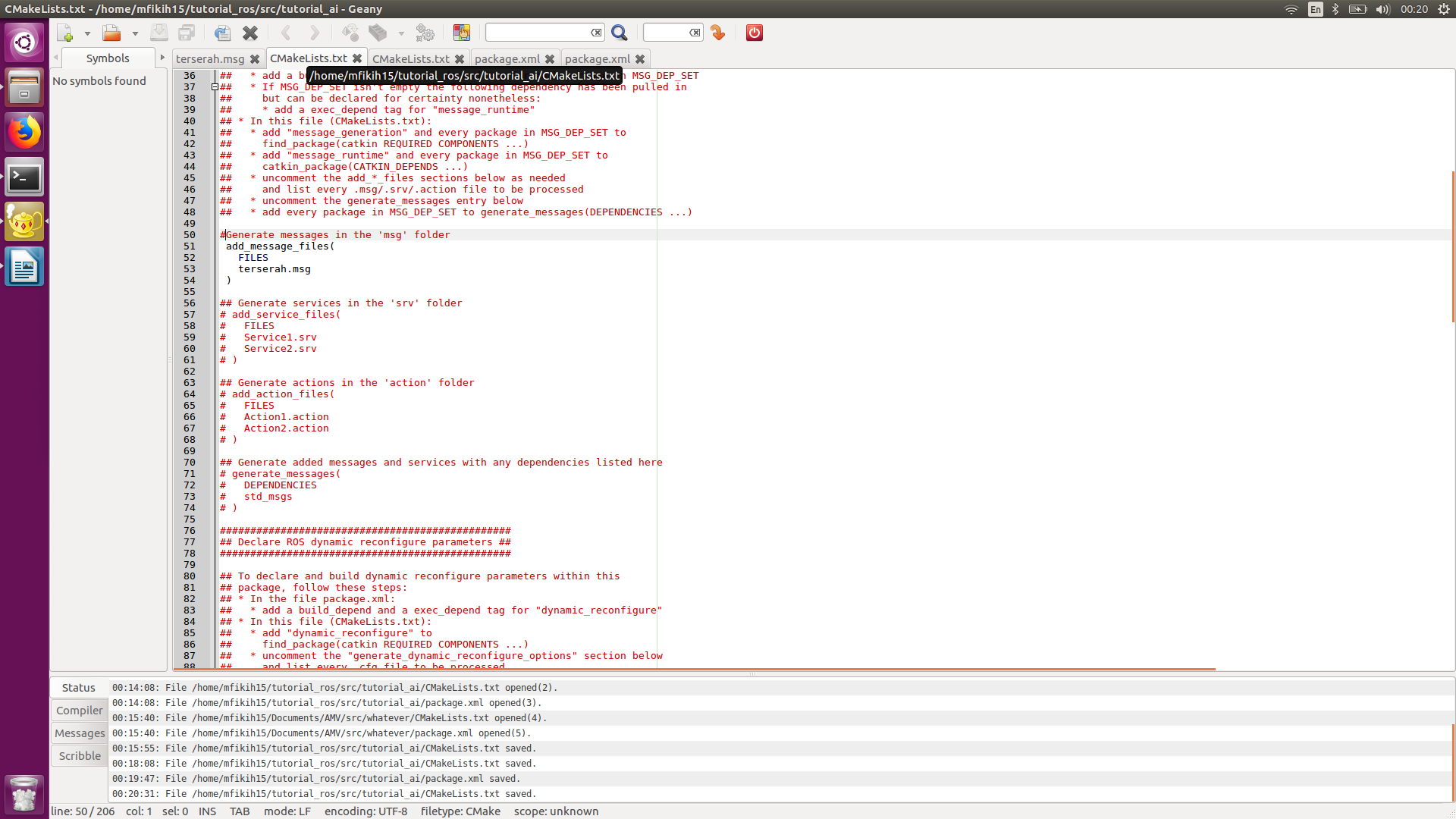
DEPENDENCIES

std\_msgs

)







tambahakan juga

add\_executable(test\_msg src/test\_msg.cpp)

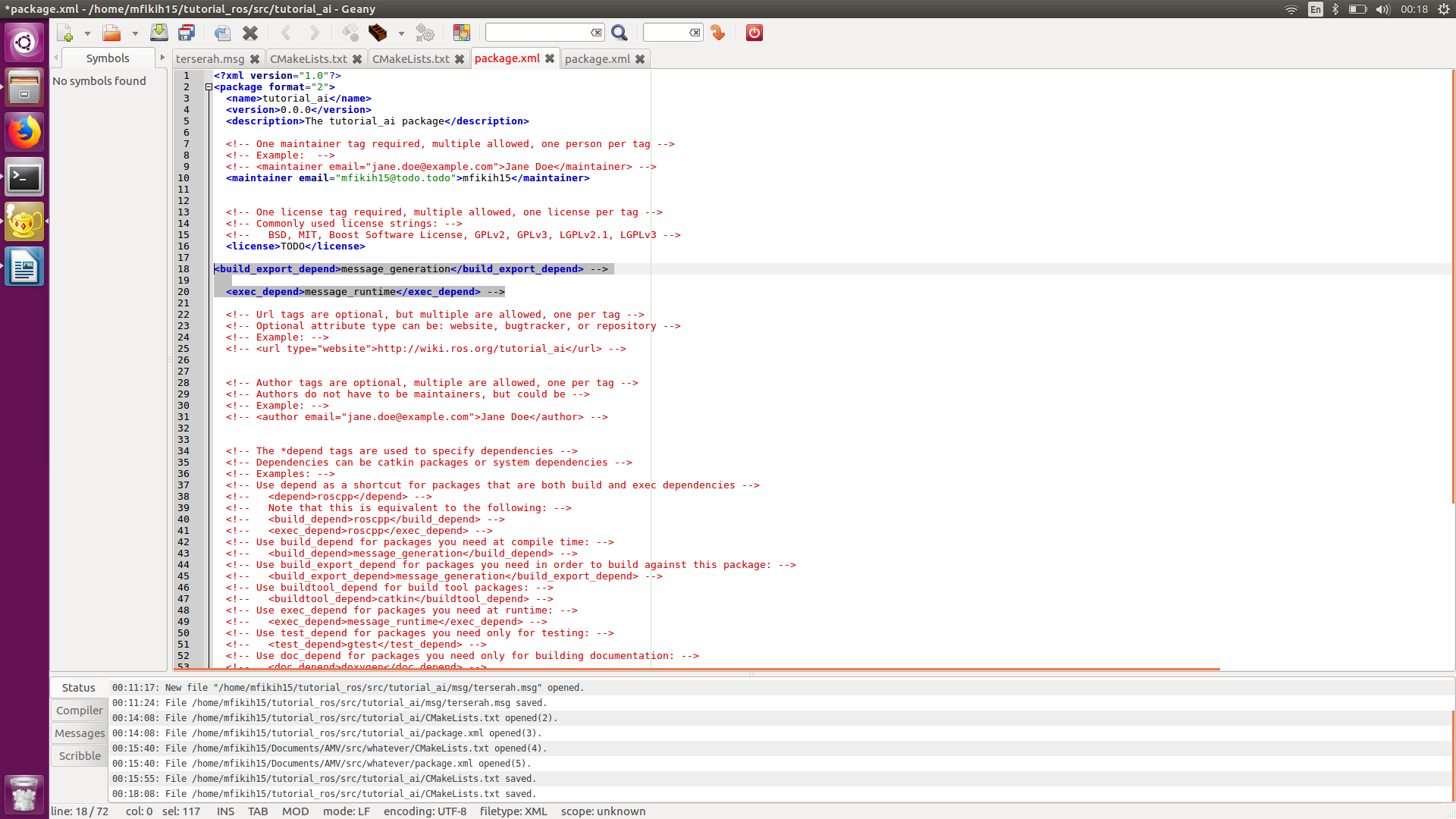
target\_link\_libraries(test\_msg ${catkin\_LIBRARIES})

diatas [install] seperti pada tutorial simple publisher and subscriber

pada package.xml isi dengan

<build\_export\_depend>message\_generation</build\_export\_depend> -->

<exec\_depend>message\_runtime</exec\_depend> →



buat lah test\_msg.cpp

~/tutorial\_ros/src/tutorial\_ai$ cd src

~/tutorial\_ros/src/tutorial\_ai/src$ geany test\_msg.cpp

#include "ros/ros.h"

#include "tutorial\_ai/terserah.h"

#include <sstream>

int main(int argc, char \*\*argv){

ros::init(argc, argv, "test\_msg");

ros::NodeHandle n;

ros::Publisher chatter\_pub = n.advertise<tutorial\_ai::terserah>("whatever", 1000);

while (ros::ok()){

tutorial\_ai::terserah msg;

msg.nomer = 1;

chatter\_pub.publish(msg);

ros::spinOnce();

}

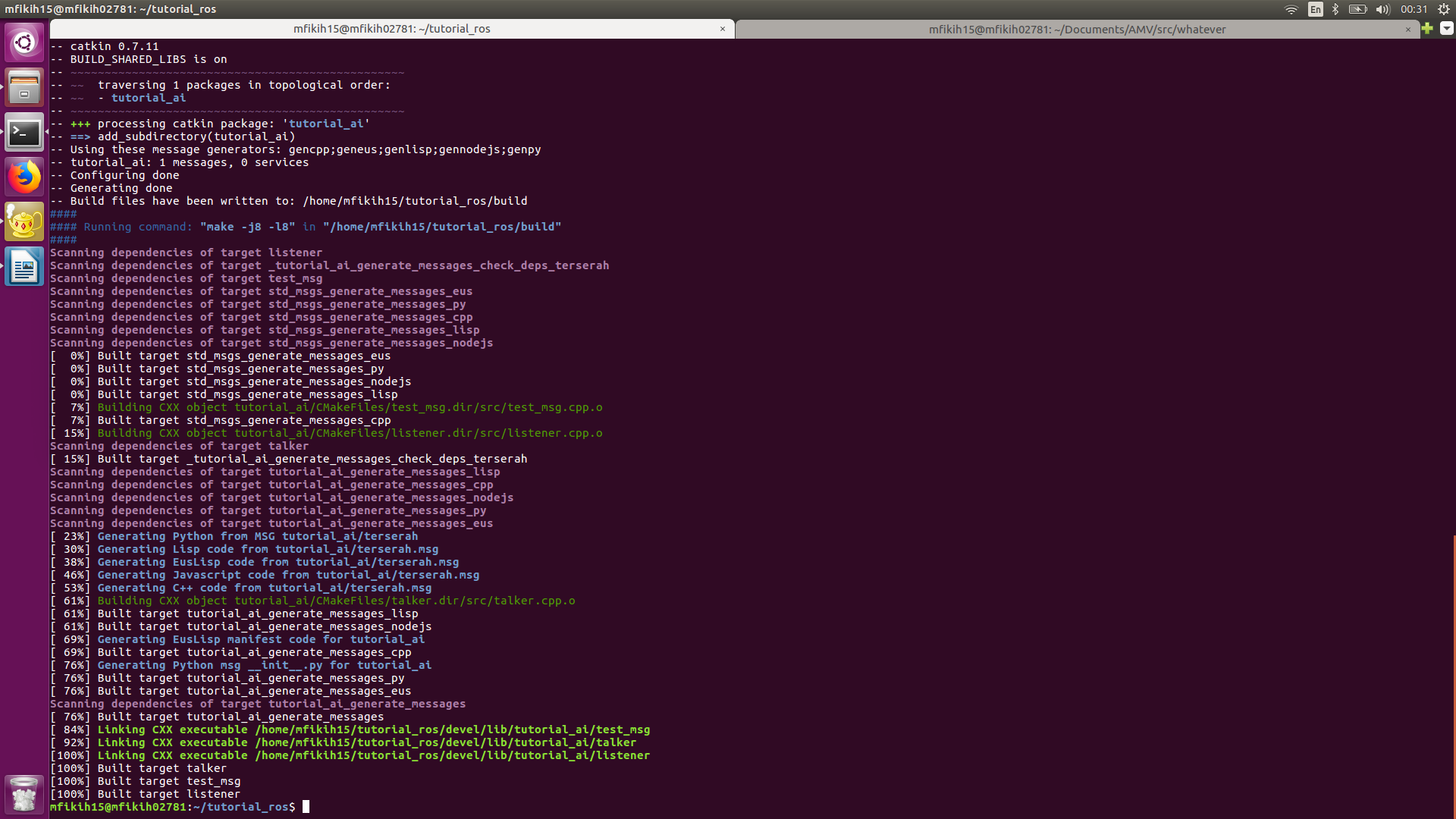
return 0;

}

kemudian catkin\_make

cd ~/tutorial\_ros

~/tutorial\_ros$ catkin\_make



jalankan roscore

~/tutorial\_ros$ roscore

buka terminal baru dan jalankan node test\_msg

rosrun tutorial\_ai test\_msg

buka terminal baru dan jalankan rostopic untuk melihat topic

rostopic echo /<nama topic>

**5. Image Transport**

image transport digunakan untuk memisahkan antara capture gambar dengan image processing

masih menggunakan workspace dan package yang sama

pada src di package tutorial\_ai, buatlah codingan bernama talker\_video.cpp dan listen\_video.cpp

~/tutorial\_ros/src/tutorial\_ai/src$ geany talker\_video.cpp listen\_video.cpp

pada talker\_video.cpp