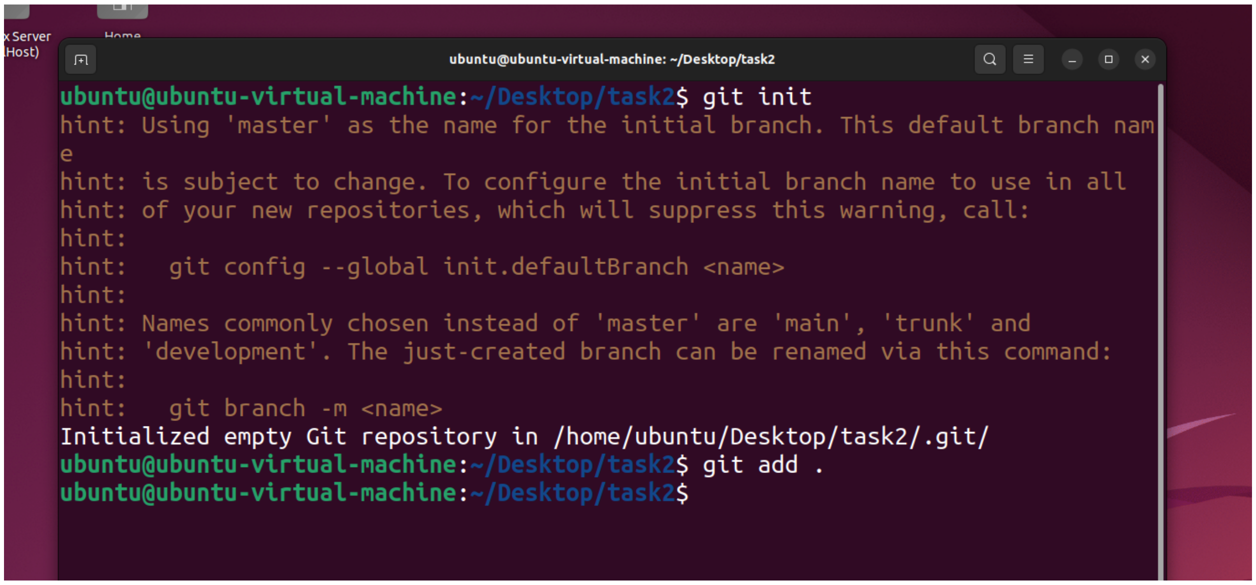
1. **Create GitLab repository**:

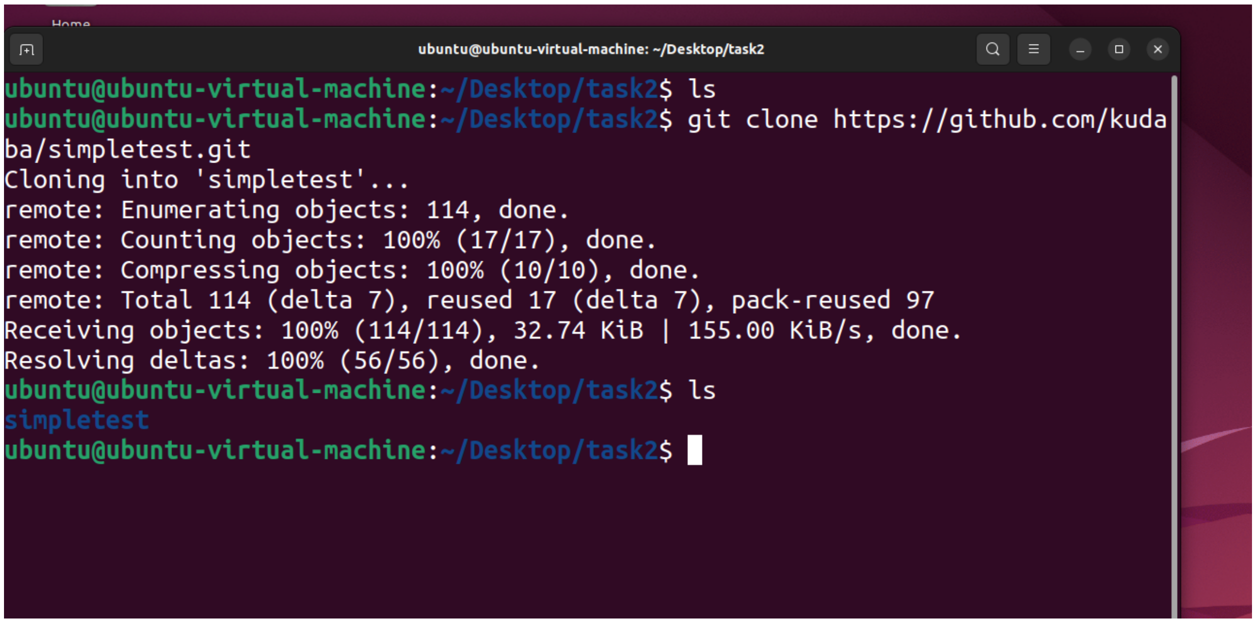
gitlab create-project packettool



```
ubuntu@ubuntu-virtual-machine:~/Desktop/task2$ git init
hint: Using 'master' as the name for the initial branch. This default branch nam
e
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/ubuntu/Desktop/task2/.git/
ubuntu@ubuntu-virtual-machine:~/Desktop/task2$ git add .
ubuntu@ubuntu-virtual-machine:~/Desktop/task2$
```

2. **Clone the repository**:

git clone https://github.com/kudaba/simpletest.git



```
ubuntu@ubuntu-virtual-machine:~/Desktop/task2$ ls
ubuntu@ubuntu-virtual-machine:~/Desktop/task2$ git clone https://github.com/kuda
ba/simpletest.git
Cloning into 'simpletest'...
remote: Enumerating objects: 114, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 114 (delta 7), reused 17 (delta 7), pack-reused 97
Receiving objects: 100% (114/114), 32.74 KiB | 155.00 KiB/s, done.
Resolving deltas: 100% (56/56), done.
ubuntu@ubuntu-virtual-machine:~/Desktop/task2$ ls
simpletest
ubuntu@ubuntu-virtual-machine:~/Desktop/task2$
```

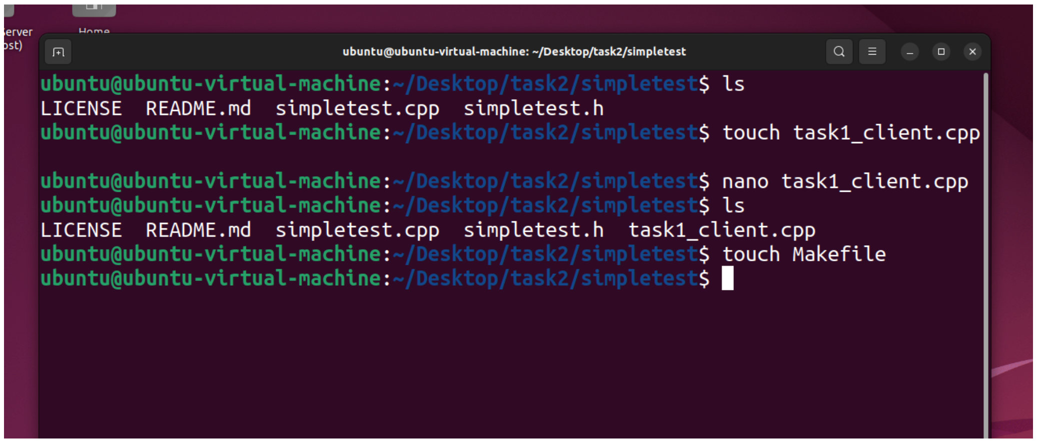
3. **Set up directory structure**:

cd packettoolmkdir packets

4. **Create C++ source file**:

touch task1_client.cpp

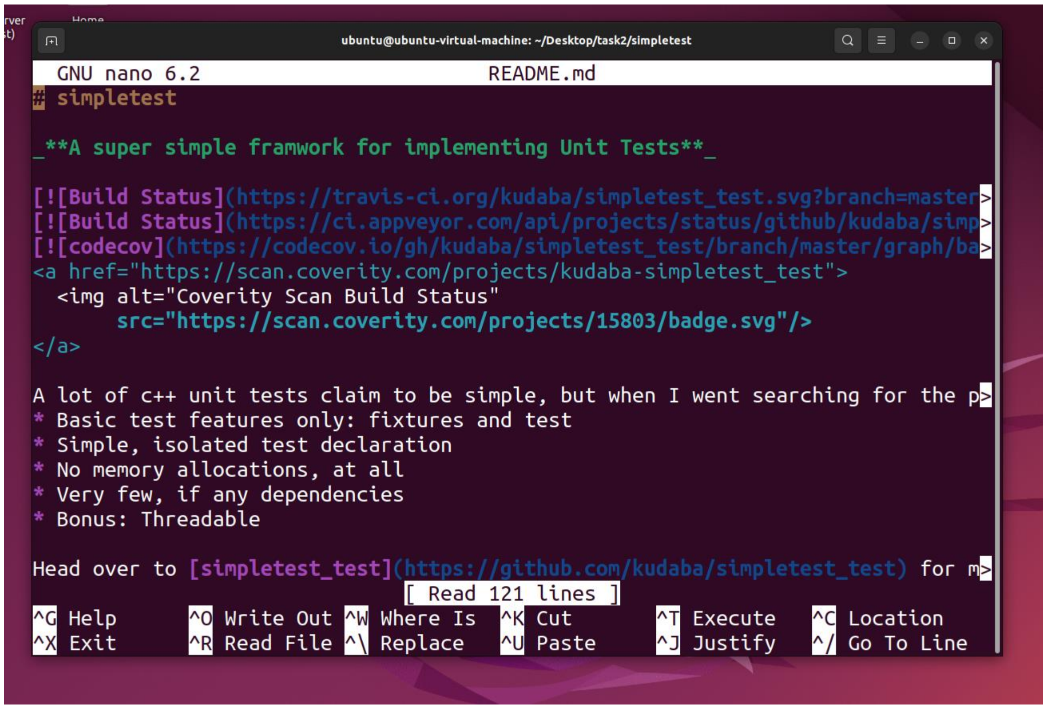
5. **Write the C++ code**: Edit **task1_client.cpp** and add the provided code snippet.
6. **Write Makefile**:



```
ubuntu@ubuntu-virtual-machine: ~/Desktop/task2/simpletest
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ ls
LICENSE  README.md  simpletest.cpp  simpletest.h
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ touch task1_client.cpp

ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ nano task1_client.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ ls
LICENSE  README.md  simpletest.cpp  simpletest.h  task1_client.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ touch Makefile
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ 
```

touch Makefile

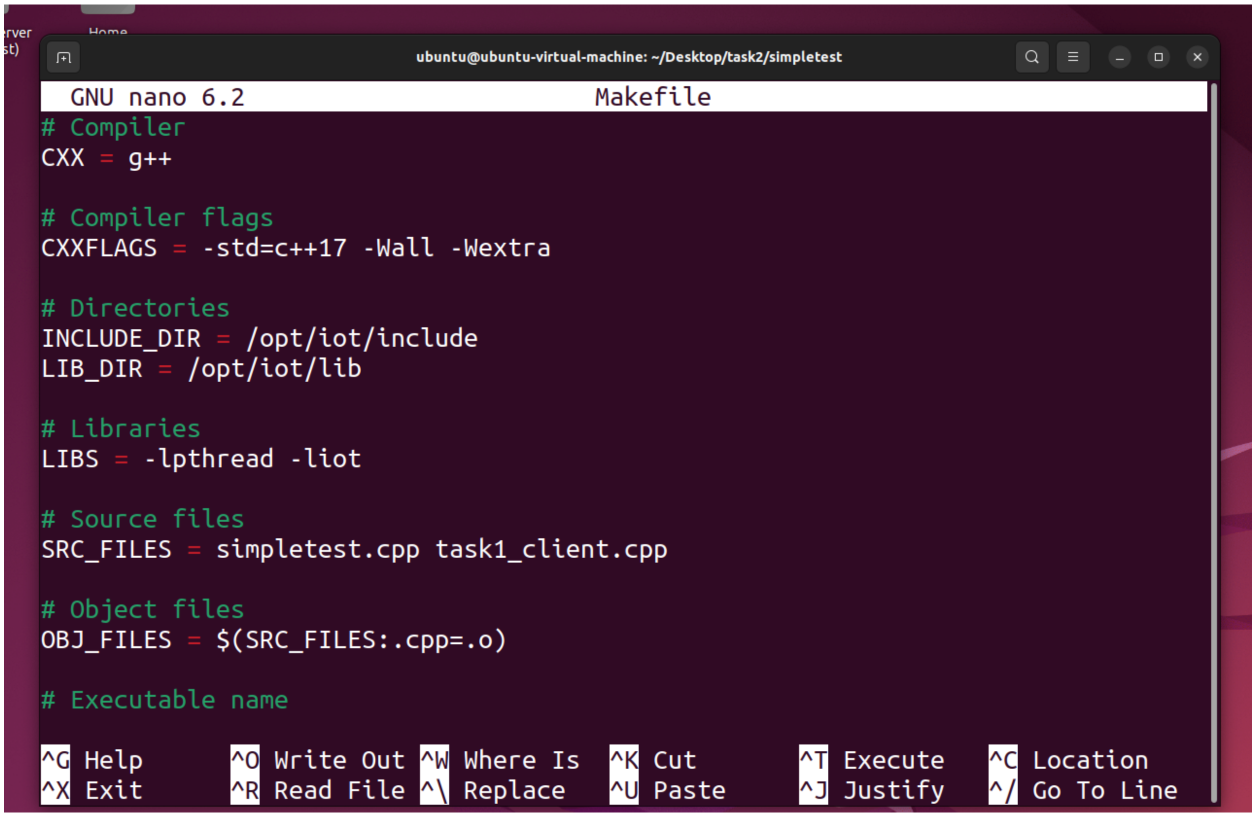
Edit **Makefile** and add the necessary compilation instructions.



```
ubuntu@ubuntu-virtual-machine: ~/Desktop/task2/simpletest
 GNU nano 6.2                          README.md
# simpletest

_**A super simple framwork for implementing Unit Tests**_

[![Build Status](https://travis-ci.org/kudaba/simpletest_test.svg?branch=master>
[![Build Status](https://ci.appveyor.com/api/projects/status/github/kudaba/simp>
[![codecov](https://codecov.io/gh/kudaba/simpletest_test/branch/master/graph/ba>
<a href="https://scan.coverity.com/projects/kudaba-simpletest_test">
  <img alt="Coverity Scan Build Status"
       src="https://scan.coverity.com/projects/15803/badge.svg"/>
</a>

A lot of c++ unit tests claim to be simple, but when I went searching for the p>
* Basic test features only: fixtures and test
* Simple, isolated test declaration
* No memory allocations, at all
* Very few, if any dependencies
* Bonus: Threadable

Head over to [simpletest_test](https://github.com/kudaba/simpletest_test) for m>
                               [ Read 121 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```

7. **Compile the client**:

make

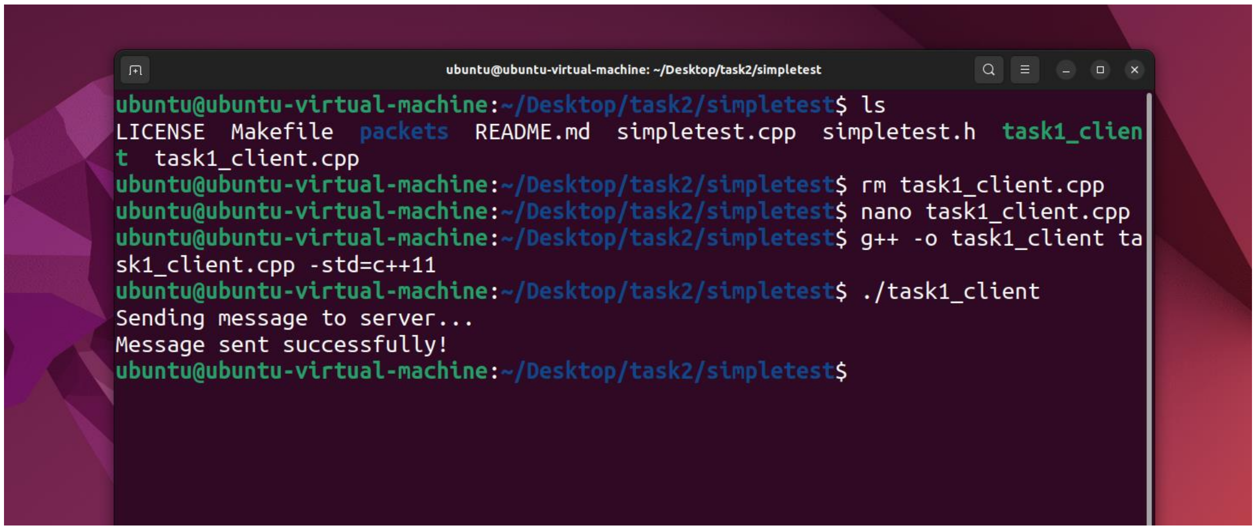


```
  GNU nano 6.2                          Makefile
# Compiler
CXX = g++

# Compiler flags
CXXFLAGS = -std=c++17 -Wall -Wextra

# Directories
INCLUDE_DIR = /opt/iot/include
LIB_DIR = /opt/iot/lib

# Libraries
LIBS = -lpthread -liot

# Source files
SRC_FILES = simpletest.cpp task1_client.cpp

# Object files
OBJ_FILES = $(SRC_FILES:.cpp=.o)

# Executable name

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

8. **Set up packet streams**:

/opt/iot/bin/create_packetfile 192.168.1.7 1001 /opt/iot/bin/create_packetfile 192.168.1.8 8877

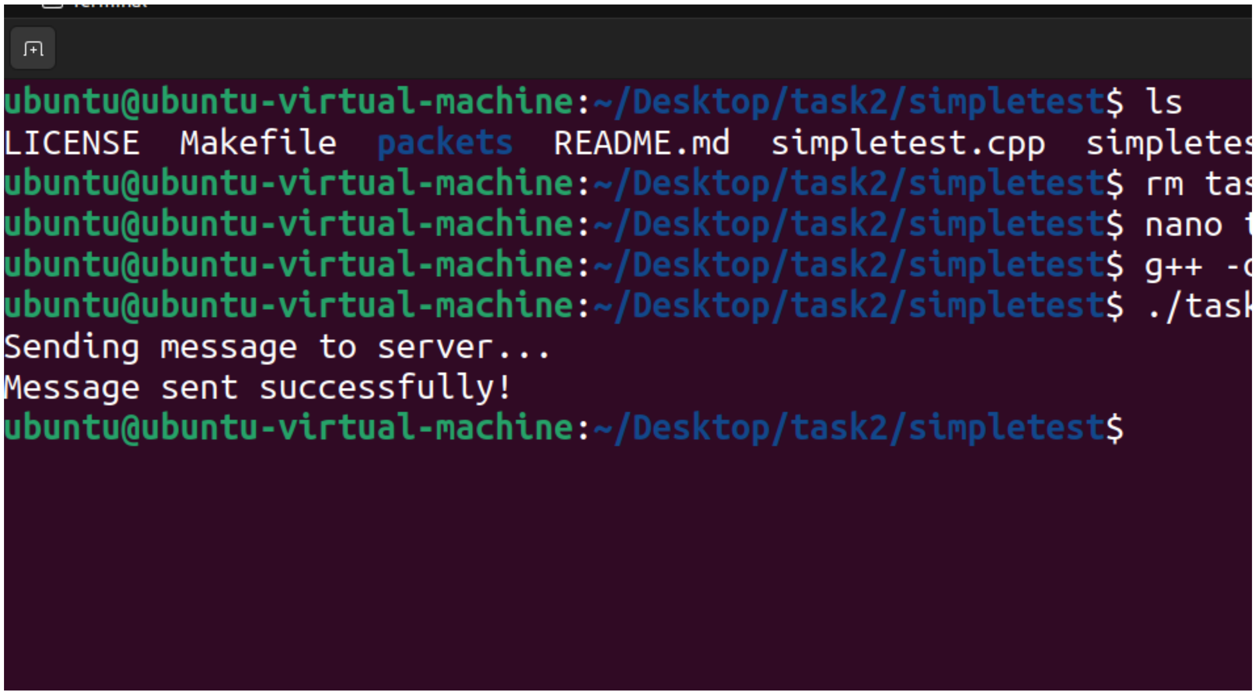
9. **Test the client**:



```
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ ls
LICENSE  Makefile  packets  README.md  simpletest.cpp  simpletest.h  task1_clien
t  task1_client.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ rm task1_client.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ nano task1_client.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ g++ -o task1_client ta
sk1_client.cpp -std=c++11
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ ./task1_client
Sending message to server...
Message sent successfully!
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$
```

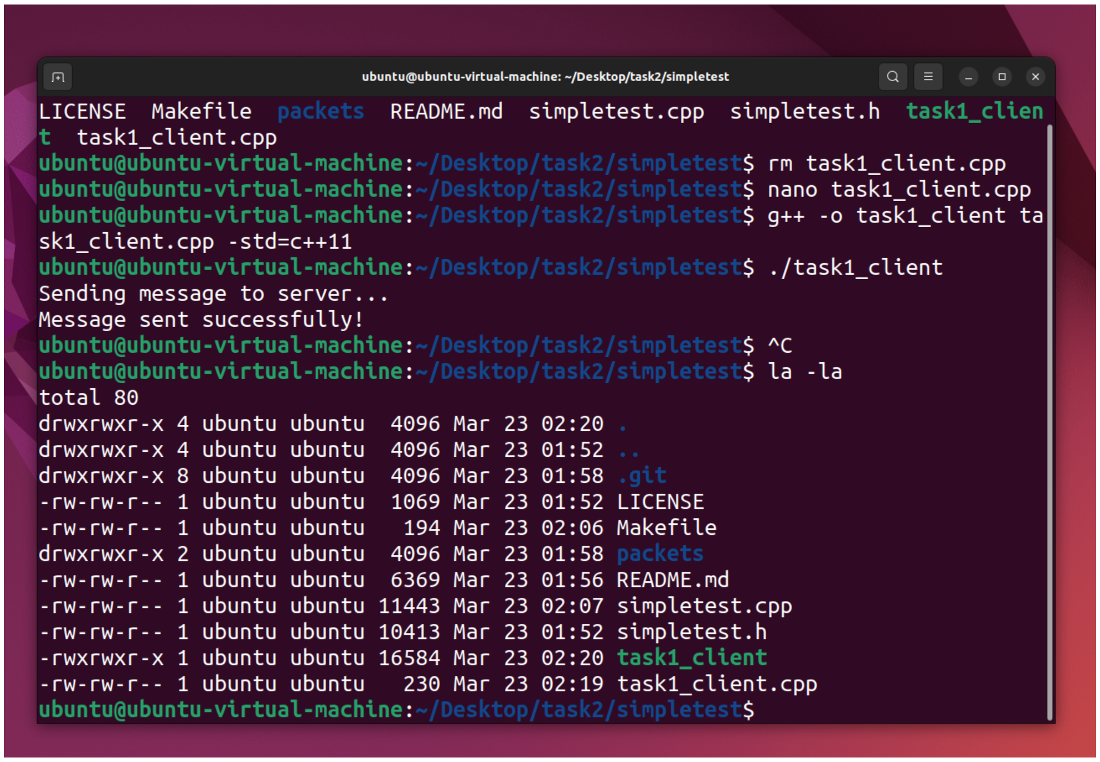
./task1_client

10. **Check packet streams**:



```
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ ls
LICENSE  Makefile  packets  README.md  simpletest.cpp  simpletes
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ rm tas
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ nano t
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ g++ -o
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ ./task
Sending message to server...
Message sent successfully!
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$
```

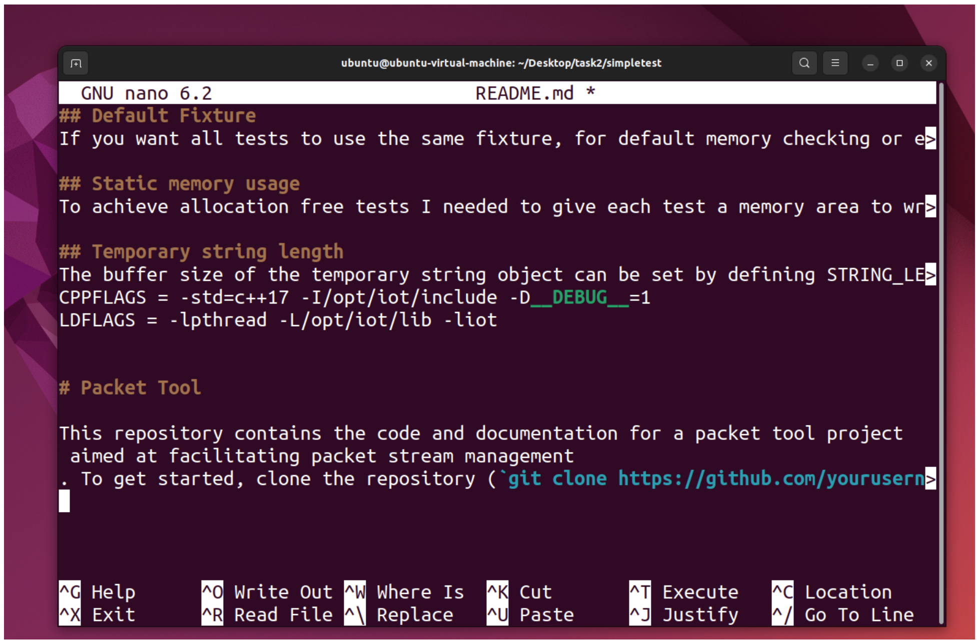
ls -l packets/



```
LICENSE  Makefile  packets  README.md  simpletest.cpp  simpletest.h  task1_clien
t  task1_client.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ rm task1_client.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ nano task1_client.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ g++ -o task1_client ta
sk1_client.cpp -std=c++11
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ ./task1_client
Sending message to server...
Message sent successfully!
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ ^C
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ la -la
total 80
drwxrwxr-x 4 ubuntu ubuntu  4096 Mar 23 02:20 .
drwxrwxr-x 4 ubuntu ubuntu  4096 Mar 23 01:52 ..
drwxrwxr-x 8 ubuntu ubuntu  4096 Mar 23 01:58 .git
-rw-rw-r-- 1 ubuntu ubuntu  1069 Mar 23 01:52 LICENSE
-rw-rw-r-- 1 ubuntu ubuntu   194 Mar 23 02:06 Makefile
drwxrwxr-x 2 ubuntu ubuntu  4096 Mar 23 01:58 packets
-rw-rw-r-- 1 ubuntu ubuntu  6369 Mar 23 01:56 README.md
-rw-rw-r-- 1 ubuntu ubuntu 11443 Mar 23 02:07 simpletest.cpp
-rw-rw-r-- 1 ubuntu ubuntu 10413 Mar 23 01:52 simpletest.h
-rwxrwxr-x 1 ubuntu ubuntu 16584 Mar 23 02:20 task1_client
-rw-rw-r-- 1 ubuntu ubuntu   230 Mar 23 02:19 task1_client.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$
```

Verify that the packet streams have been created and contain data.

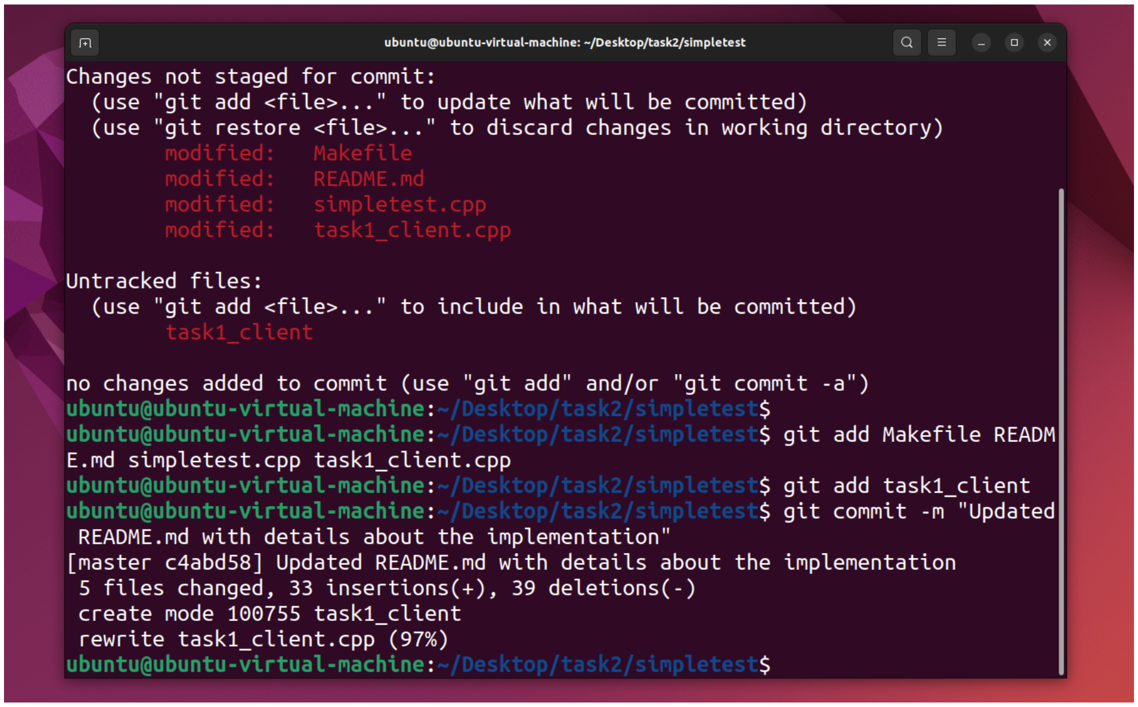
11. **Document your work**: Update **README.md** with details about your implementation.



```
  GNU nano 6.2                    README.md *
## Default Fixture
If you want all tests to use the same fixture, for default memory checking or e>

## Static memory usage
To achieve allocation free tests I needed to give each test a memory area to wr>

## Temporary string length
The buffer size of the temporary string object can be set by defining STRING_LE>
CPPFLAGS = -std=c++17 -I/opt/iot/include -D__DEBUG__=1
LDFLAGS = -lpthread -L/opt/iot/lib -liot


# Packet Tool

This repository contains the code and documentation for a packet tool project
 aimed at facilitating packet stream management
. To get started, clone the repository (`git clone https://github.com/yourusern>


^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```

12. **Push changes to GitLab**:

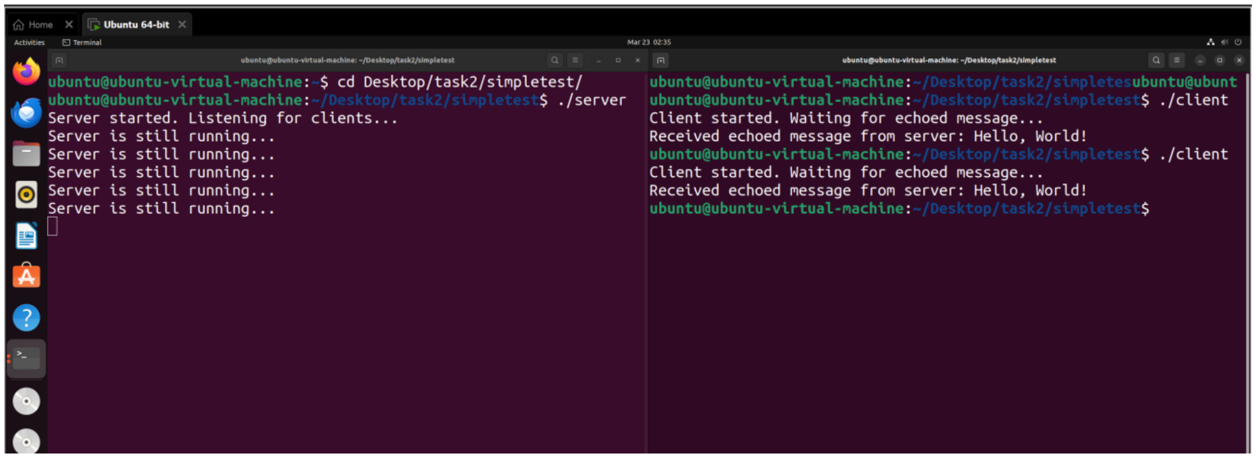
git add . git commit -m "Completed Task 1" git push origin master



```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Makefile
        modified:   README.md
        modified:   simpletest.cpp
        modified:   task1_client.cpp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        task1_client

no changes added to commit (use "git add" and/or "git commit -a")
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ git add Makefile READM
E.md simpletest.cpp task1_client.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ git add task1_client
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ git commit -m "Updated
 README.md with details about the implementation"
[master c4abd58] Updated README.md with details about the implementation
 5 files changed, 33 insertions(+), 39 deletions(-)
 create mode 100755 task1_client
 rewrite task1_client.cpp (97%)
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$
```

Task 2

1. Extend your server to run continually and support any number of clients.

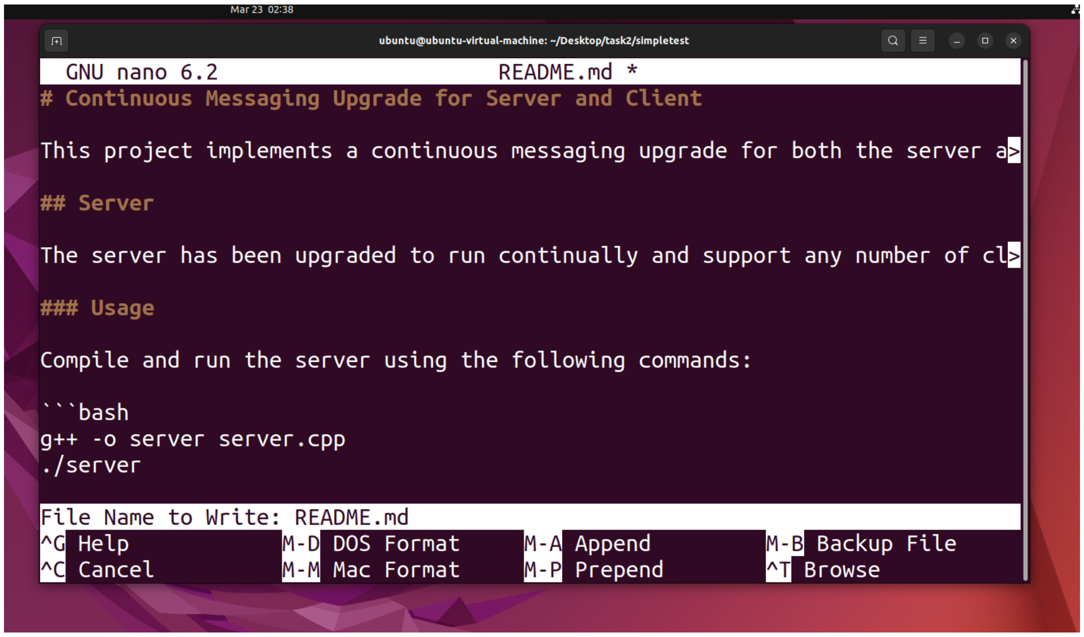


```
ubuntu@ubuntu-virtual-machine:~$ cd Desktop/task2/simpletest/
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ ./server
Server started. Listening for clients...
Server is still running...
Server is still running...
Server is still running...
Server is still running...
Server is still running...
```

```
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletesubuntu@ubunt
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ ./client
Client started. Waiting for echoed message...
Received echoed message from server: Hello, World!
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$ ./client
Client started. Waiting for echoed message...
Received echoed message from server: Hello, World!
ubuntu@ubuntu-virtual-machine:~/Desktop/task2/simpletest$
```

2. Add support for the server to send back the message received, i.e. echo it back, and for the client to wait until it receives the echoed message and prints it to the console.

Document your work in the README.md.



~~~
  GNU nano 6.2                    README.md *
# Continuous Messaging Upgrade for Server and Client

This project implements a continuous messaging upgrade for both the server a>

## Server

The server has been upgraded to run continually and support any number of cl>

### Usage

Compile and run the server using the following commands:

```bash
g++ -o server server.cpp
./server

File Name to Write: README.md
^G Help              M-D DOS Format       M-A Append           M-B Backup File
^C Cancel            M-M Mac Format       M-P Prepend          ^T Browse
~~~

Conclusion:

In this lab, we established a GitLab repository, developed a C++ client for packet handling, and configured compilation via Makefile. We also set up packet streams, verified their creation, and documented our process. Additionally, we extended the server to support continuous operation and implemented echo functionality between server and client, with corresponding documentation.