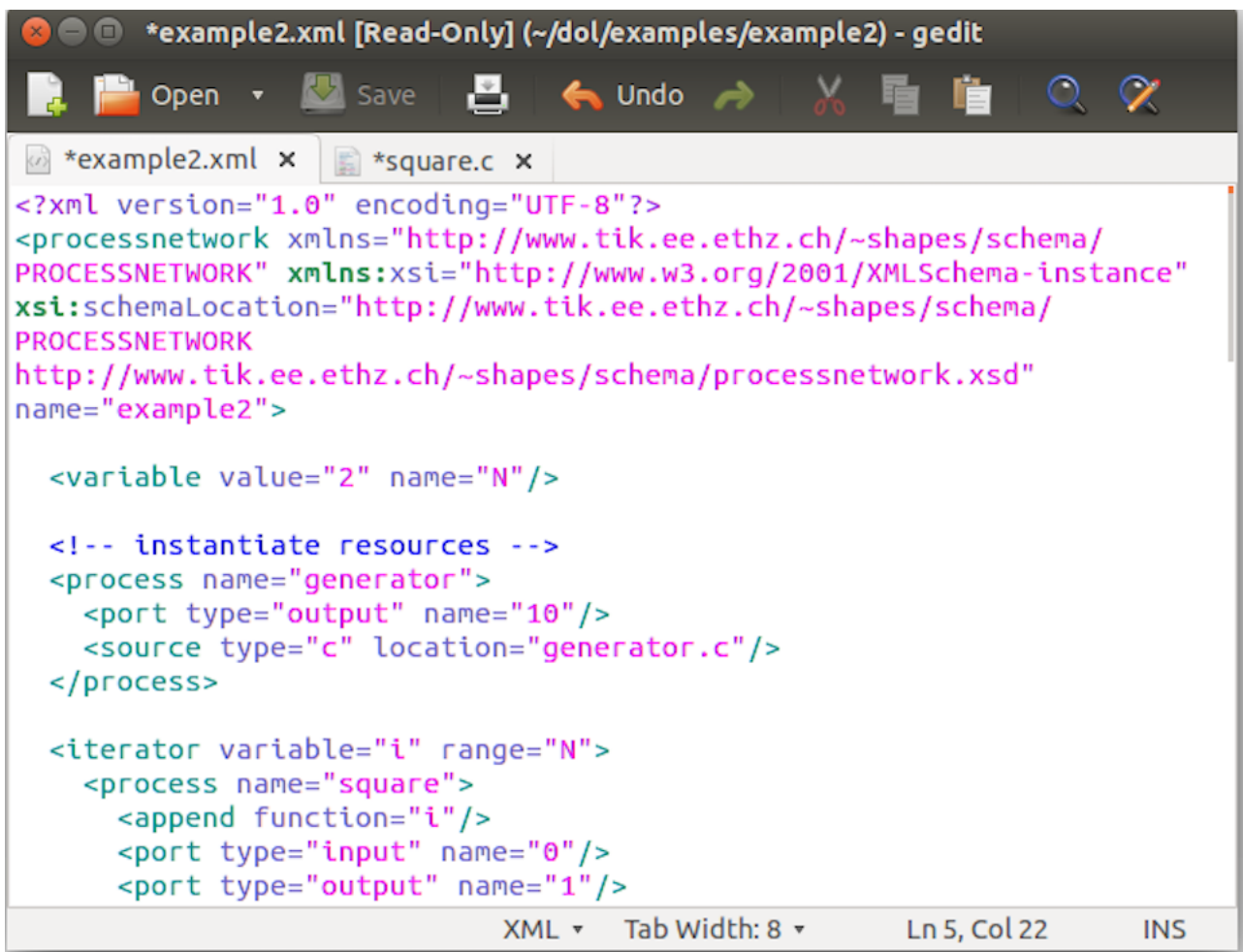


DOL实例分析与编程

DOL分析 Markdown

实验任务及实现：

- **任务1：** 修改example2，让3个square模块变成两个。
实现：修改examples文件夹下的example2.xml文件



```
<?xml version="1.0" encoding="UTF-8"?>
<processnetwork xmlns="http://www.tik.ee.ethz.ch/~shapes/schema/PROCESSNETWORK" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.tik.ee.ethz.ch/~shapes/schema/PROCESSNETWORK
http://www.tik.ee.ethz.ch/~shapes/schema/processnetwork.xsd"
name="example2">

  <variable value="2" name="N"/>

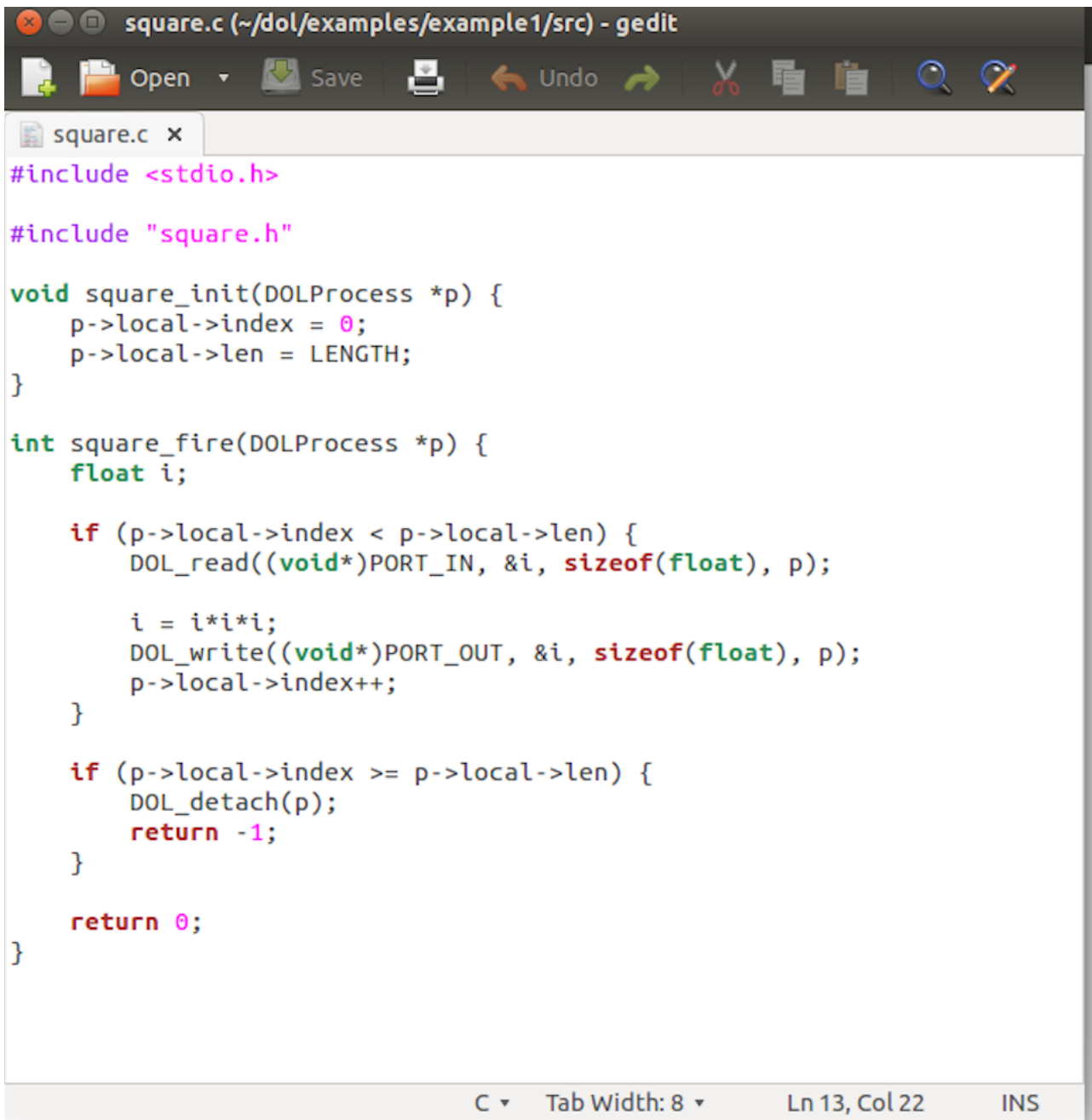
  <!-- instantiate resources -->
  <process name="generator">
    <port type="output" name="10"/>
    <source type="c" location="generator.c"/>
  </process>

  <iterator variable="i" range="N">
    <process name="square">
      <append function="i"/>
      <port type="input" name="0"/>
      <port type="output" name="1"/>
    </process>
  </iterator>
</processnetwork>
```

代码分析： **iterator**部分用一个循环建立N个square模块，而只需要修改n的定义式即可。定义n=2。

-任务2: 修改example1, 使其输出三次方数

实现: 修改examples/example1/square.c



```
square.c (~/.dol/examples/example1/src) - gedit

#include <stdio.h>

#include "square.h"

void square_init(DOLProcess *p) {
    p->local->index = 0;
    p->local->len = LENGTH;
}

int square_fire(DOLProcess *p) {
    float i;

    if (p->local->index < p->local->len) {
        DOL_read((void*)PORT_IN, &i, sizeof(float), p);

        i = i*i*i;
        DOL_write((void*)PORT_OUT, &i, sizeof(float), p);
        p->local->index++;
    }

    if (p->local->index >= p->local->len) {
        DOL_detach(p);
        return -1;
    }

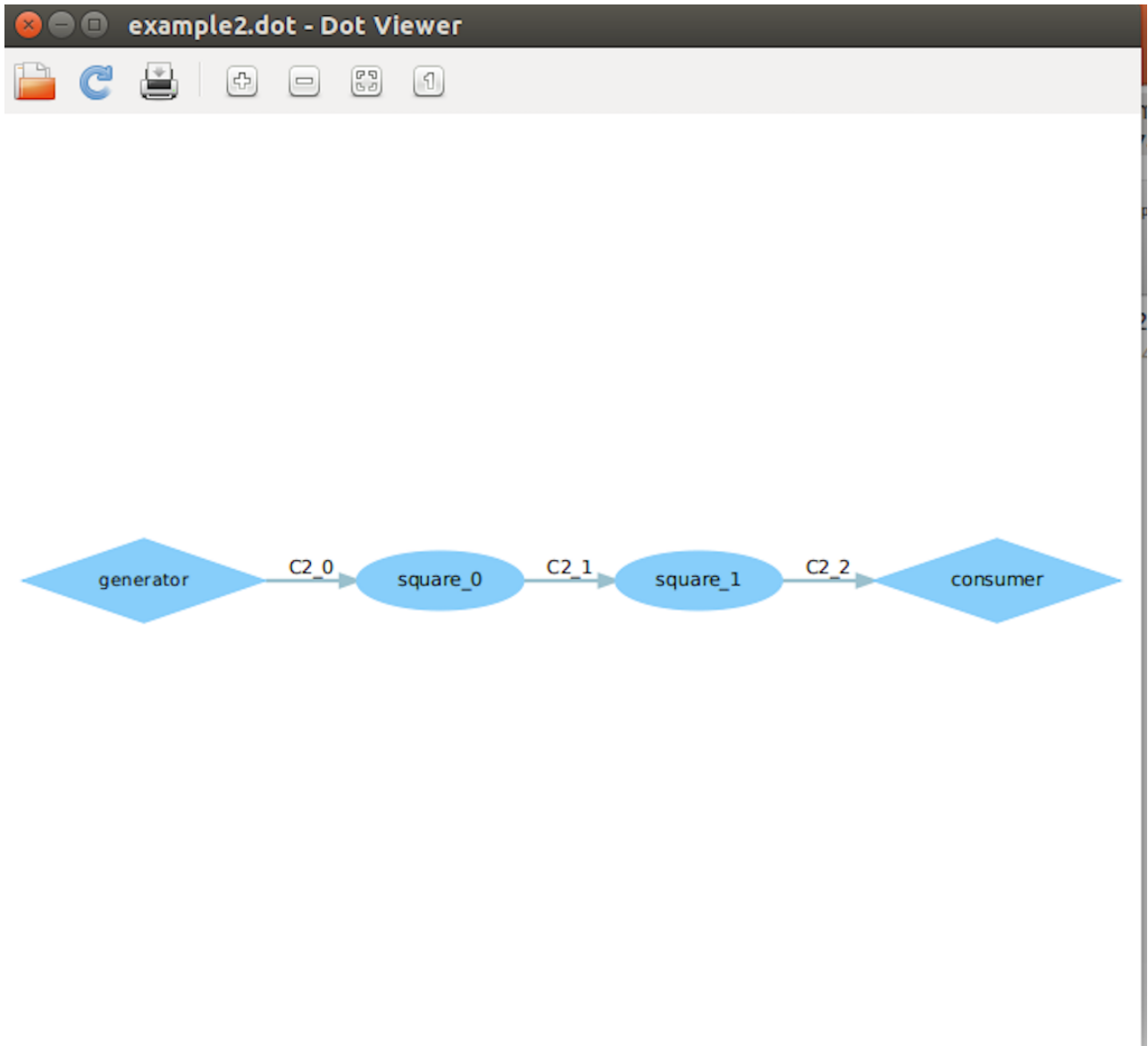
    return 0;
}
```

C ▾ Tab Width: 8 ▾ Ln 13, Col 22 INS

代码分析: 程序在`i=i*i*i`这一部分定义了代码的乘方逻辑, 因此只需要修改`i=i*i`至`i=i*i*i`即可

实验结果展示:

- 实验任务1:



- 实验任务2:

```
root@ubuntu: /home/guanzhuoqun/dol/build/bin/main

dol2:
execute:
  [echo] Make HdS application.
  [exec] make: Nothing to be done for `all'.
  [echo] Run HdS application.
  [concat] consumer: 0.000000
  [concat] consumer: 1.000000
  [concat] consumer: 4.000000
  [concat] consumer: 9.000000
  [concat] consumer: 16.000000
  [concat] consumer: 25.000000
  [concat] consumer: 36.000000
  [concat] consumer: 49.000000
  [concat] consumer: 64.000000
  [concat] consumer: 81.000000
  [concat] consumer: 100.000000
  [concat] consumer: 121.000000
  [concat] consumer: 144.000000
  [concat] consumer: 169.000000
  [concat] consumer: 196.000000
  [concat] consumer: 225.000000
  [concat] consumer: 256.000000
  [concat] consumer: 289.000000
  [concat] consumer: 324.000000
  [concat] consumer: 361.000000

BUILD SUCCESSFUL
Total time: 9 seconds
```

实验感想：

这次实验让我可以读懂了流程图的代码编写逻辑，以及中间的处理过程和最终结果的代码如何编写。知道了如何产生一个dol的运行界面。虽然只修改了简单的几个地方，但是对我读懂代码的帮助依然很大。

另外：这次实验很大的一个问题是这次实验修改的可能是系统文件，我一直权限不够，只能百度很多教程修改权限，然后在命令行里面修改代码，但是命令行里面又是用vim编译器，真的蛮难写的。