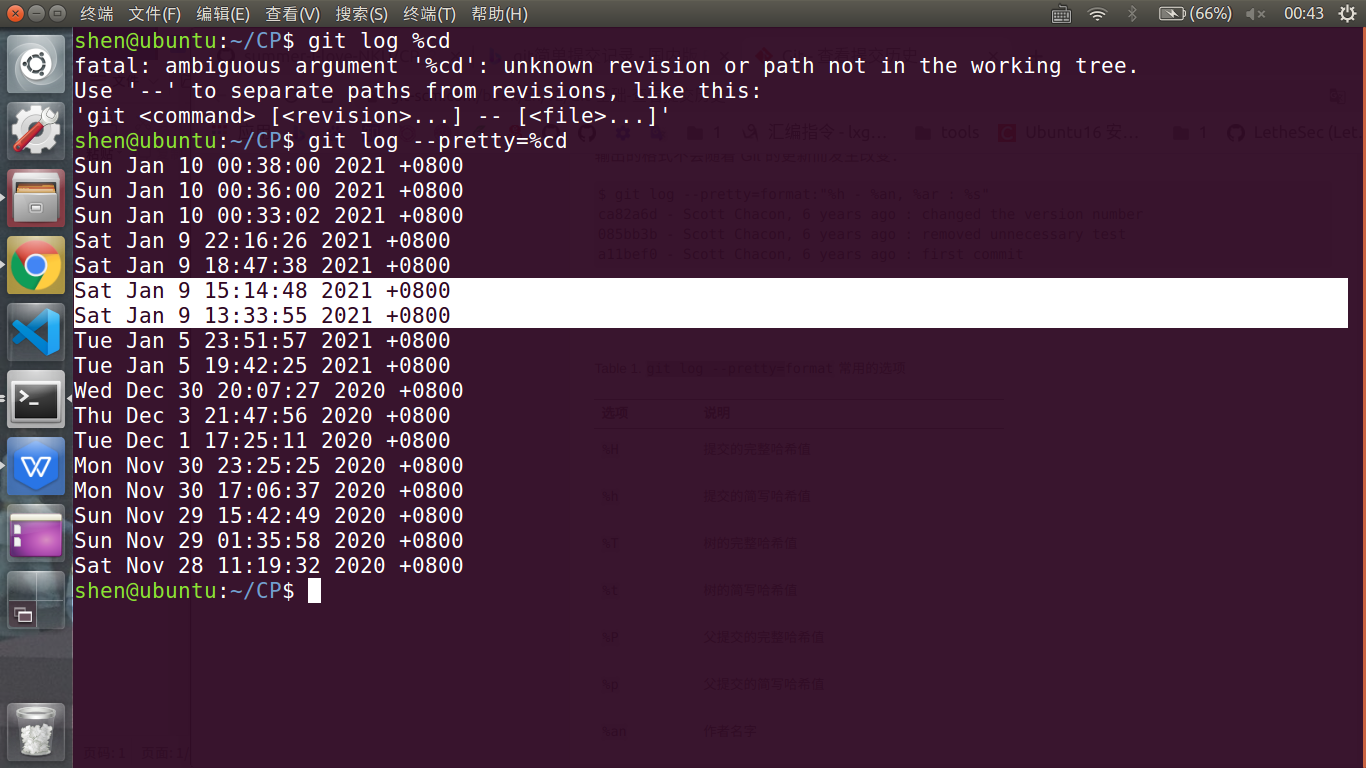
学长您好，我写这个不是为了确认我的分数我相信您已经记好了。

我只是想说明一下我真的是自己认真做的，虽然只完成了简单的一点功能用了好多天。我不是划水也没有欺骗。尊严问题啊！！！

我的很菜的代码在：<https://github.com/summer-I-love-NKU/CP_2020>



首先我今天七点就到了学院楼，但发现电脑clone下来的版本不是今下午两点多修改的最新版本（包含数组运算、数组默认初始化等等和printf完善的类型检查）

又回宿舍打开移动硬盘里的Ubuntu，但还是发现新版本不见了，最后才在垃圾桶找到，发现是删除新版本（15：44）而把旧版本（13：33）push了。

**其他没讲到的细节：**

1. 整体上

getlabel只是遍历一下声明语句，存到.data段。没有生成标签，因为if else while的标签直接在生成语句里写，逻辑运算也只是testl %eax %eax即可。

gencode分为语句和表达式（表达式计算结果都存在%eax），递归，类似先序遍历。

1. 字符赋值

输出，用movl会报错，使用的是movb赋值，输出是

movb $97, t

movzbl s, %eax

movsbl %al, %eax

subl $12, %esp

pushl %eax

call putchar

addl $16, %esp

3.数组结点的属性：

int arr\_len,init\_len,index;//声明的长度 初始化的长度 常量索引——a[5]

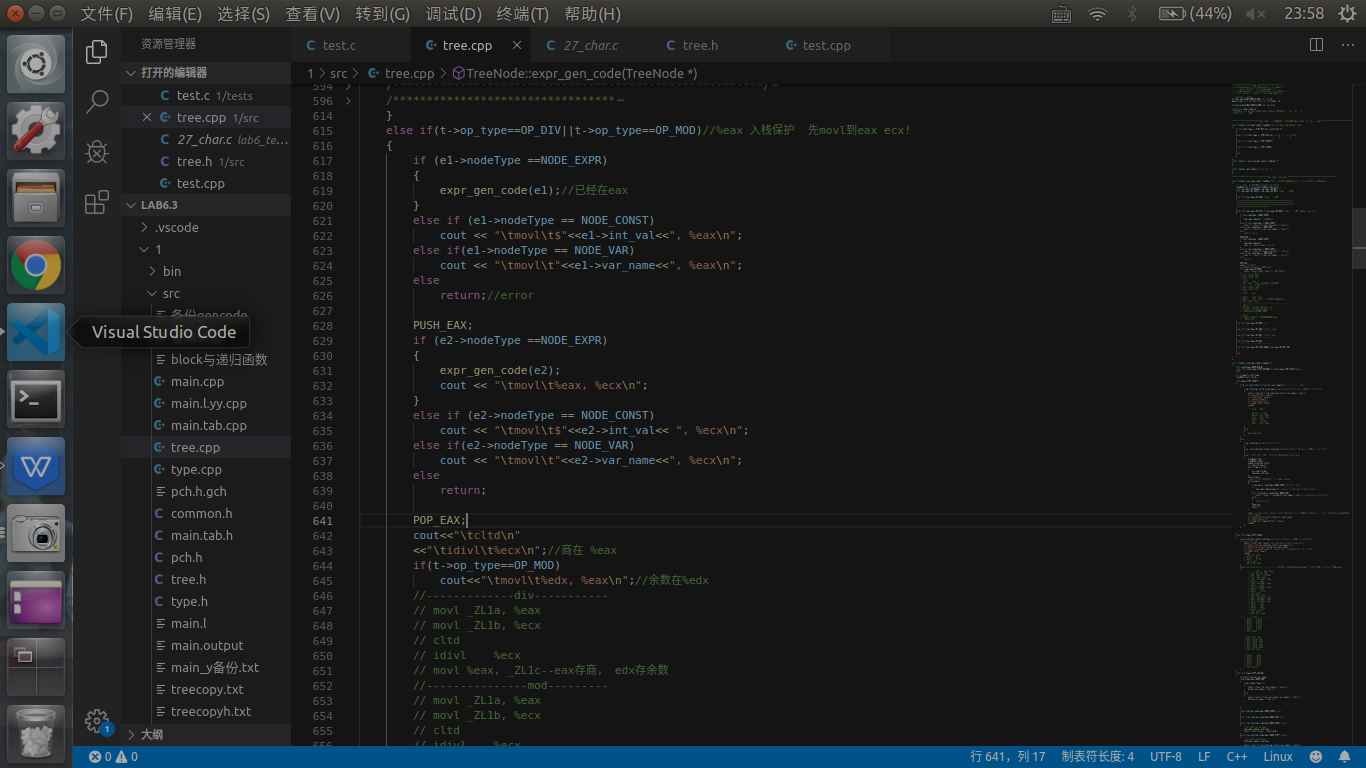
TreeNode\* var\_index; 变量索引——a[i]

char index\_flag='i'; 索引类型：默认是int 或者 变量结点

4.栈的使用和标签问题：

对%eax的压栈，我的表达式计算（无论是数值还是条件判断的逻辑值1或0）最终代码都将结果存在%eax，这时%eax就面临着被子表达式覆盖的风险

所以在表达式运算中，比如做除法（被除数在%eax,除数在%ecx），我就将被除数结果%eax先入栈，防止计算除数时如果除数为表达式也用到%eax。计算完除数再popl %eax。用%eax和%ecx作除法运算。

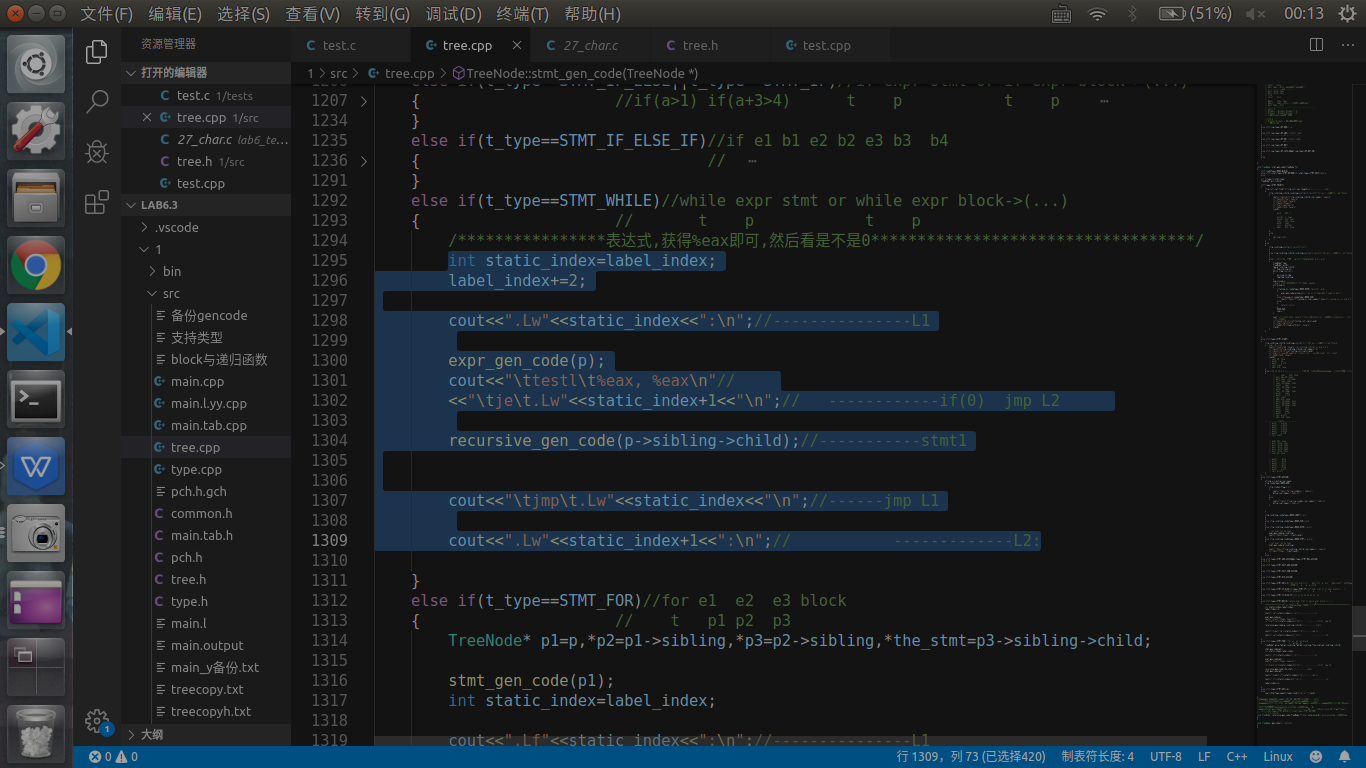


对于标签也面临着这样的问题，比如while(1){ while(1){a=1;} }

外层循环需要两个标签Lw1 Lw2 内层循环需要用Lw3 Lw4

防止外层是Lw1 Lw2 内层变成Lw2 Lw3

就在外面重新赋一个新的临时变量表示当前label，然后让label先自增2。再执行语句。



**最终结果实现：**

l1故障解释

我很自信说全过了是真的今下午甚至前两天就全部OK 了。但可能那时没发现下面两个l1中的错误

（1）表达式赋值：a=b=3;这种，原因是类型检查中当a为整型时只允许赋值为整数，而b=3是语句节点NODE\_STMT,这已经在代生成实现功能，但类型检查疏忽了。

#6 | @17 stmt stmt\_type: assign children: @17 -> [ @18 @19 ]

#6 | @18 var var\_name: b

#6 | @19 stmt stmt\_type: assign children: @19 -> [ @20 @21 ]

#6 | @20 var var\_name: c

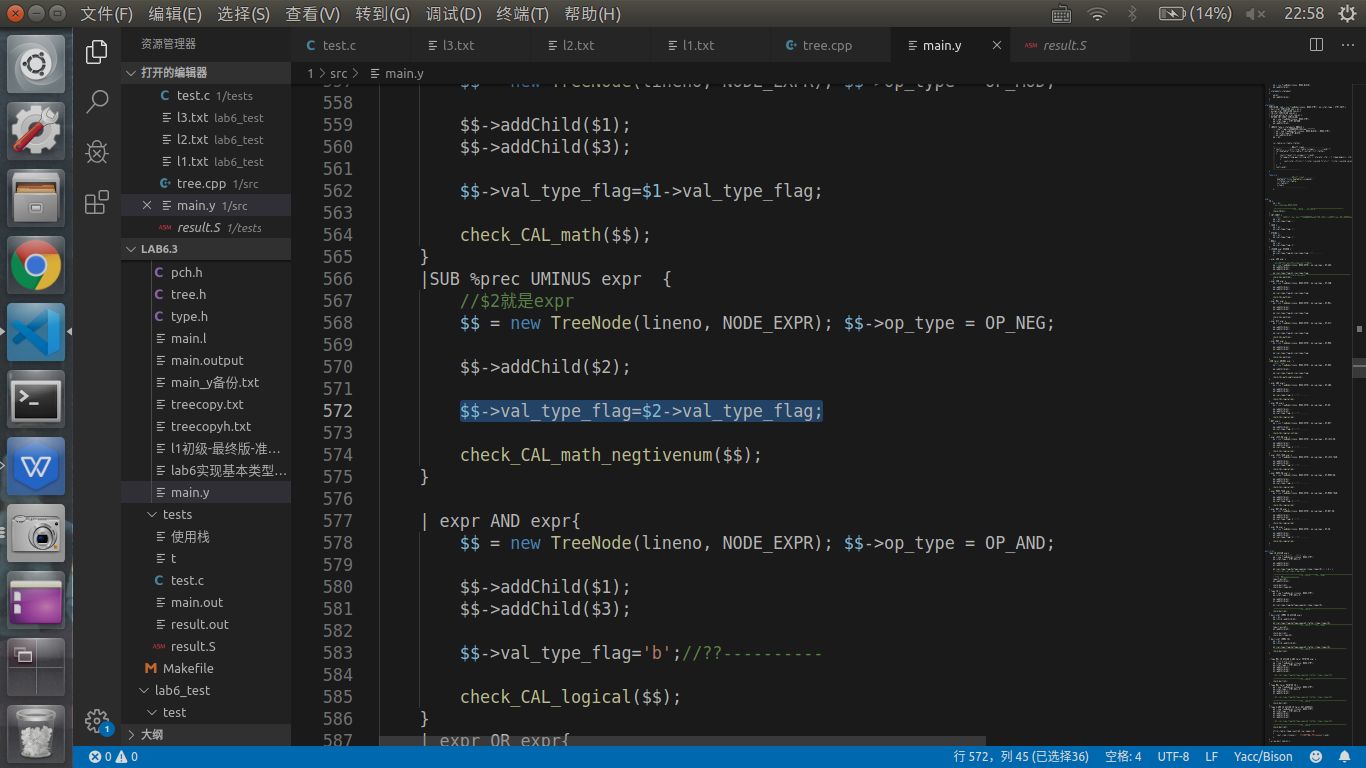
#6 | @21 const const\_type: int const\_value: 3

1. 负数

设置变量值的类型时写成$1了（$$->val\_type\_flag=$1->val\_type\_flag;

），因为之前二元运算都是这样设置的。因此也导致类型检查错误，赋值类型不是int

这里改为$$->val\_type\_flag=$2->val\_type\_flag;就好了



l2不同进制常量，数组各种方式初始化和计算，变量赋值(很简单的功能）

不同进制这个比如0xffffffff,按照进制逐位计算就行，刚刚试了一下15\*(16的7次方)是一个负数，这才使得结果为-1,C++代码也是这样，支持的是原码，所以虽然刚刚又疑惑了一下，我是实现的应该是对的

for(int i=0;i<res\_str.length();i++)

{

if(res\_str[i]>='a')

res=res\*16+res\_str[i]-'a'+10;

else

res=res\*16+res\_str[i]-'0';

}

node->int\_val=res%4294967296;//2\*\*32 默认int

node->val\_type\_flag='i';

yylval = node;

return INTEGER\_H;

}

*数组结点的属性：*

*int arr\_len,init\_len,index;//声明的长度 初始化的长度 常量索引——a[5]*

*TreeNode\* var\_index; 变量索引——a[i]*

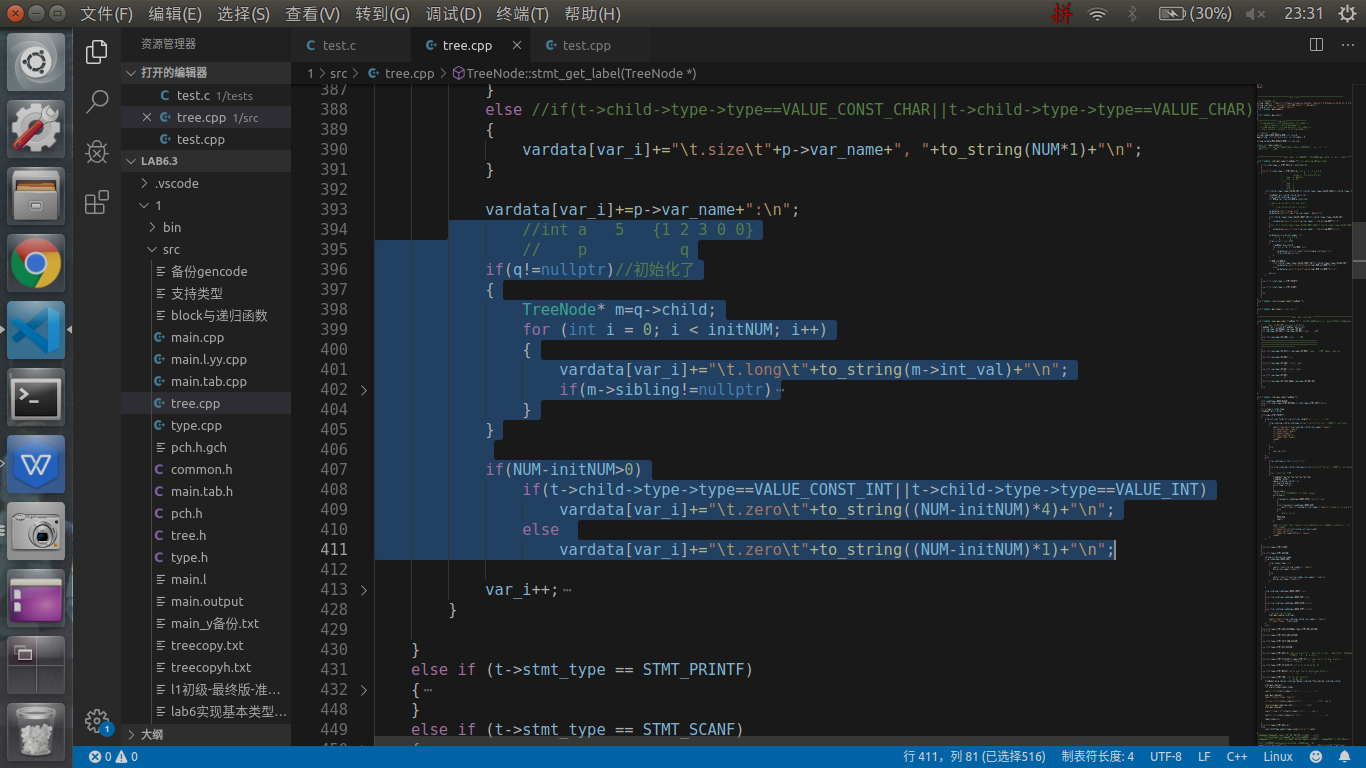
*char index\_flag='i'; 索引类型：默认是int 或者 变量结点*

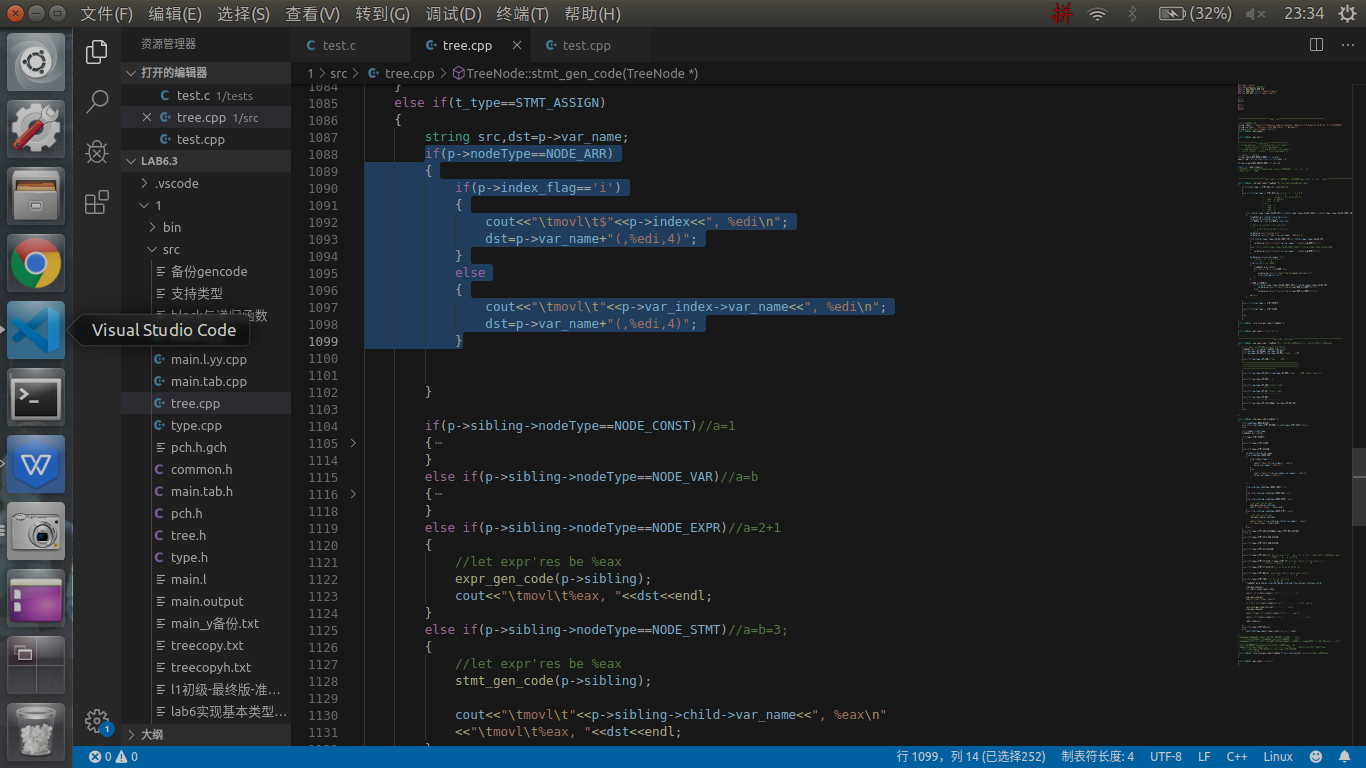
数组实现了int a[10];int a[10]={1,2}等等都可以在zero区域声明未初始化多少。

依靠的是数组结点的两个属性：声明的数组长度NUM和初始化的数组长度initNUM

数组元素赋值，实现了常量索引和变量索引：a[i] a[3]这种，依靠的是寻址方式：

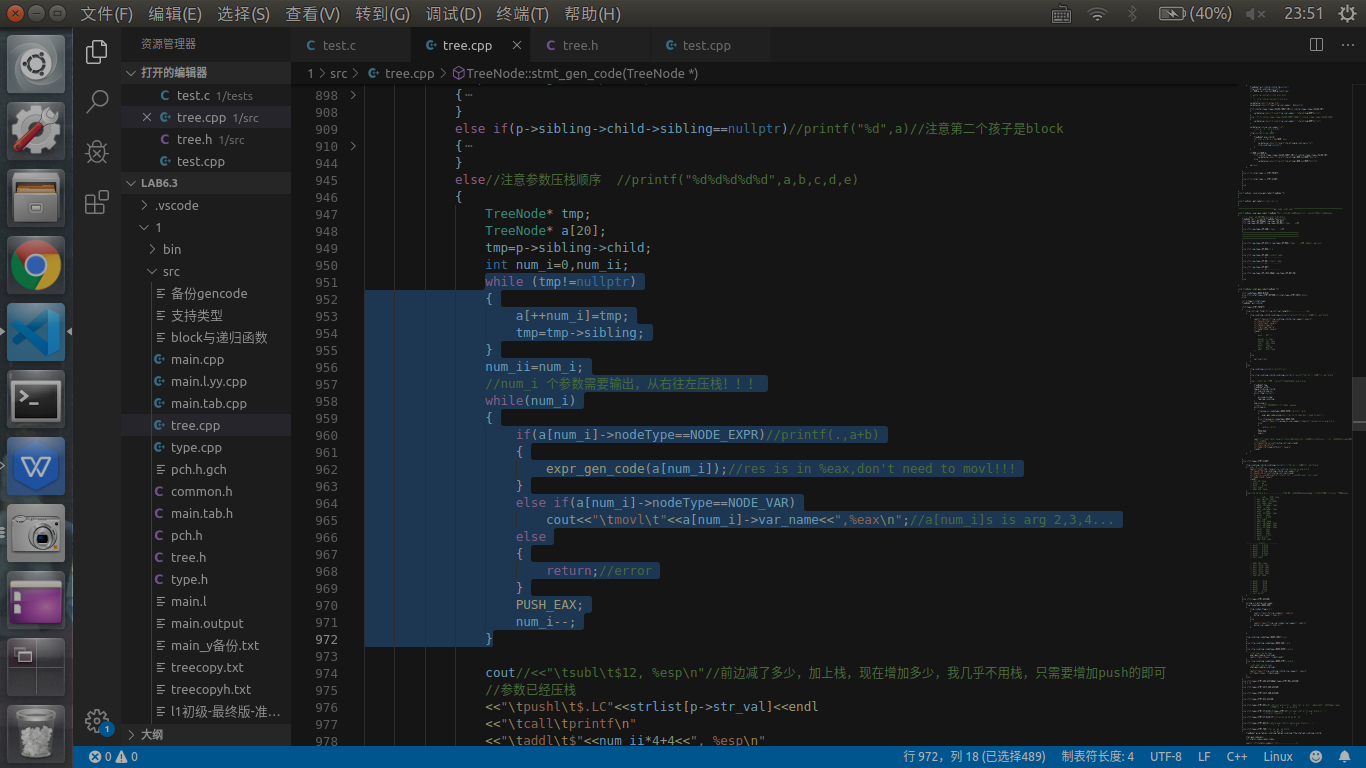
movl i %edi move $100 a(,%edi,4) 实现了a[i]=100;





l3多参数输入输出

也很简单，遍历孩子有几个，第一个while用个大数组存一下，第二个while倒着压栈



测试结果截图：

