实验报告

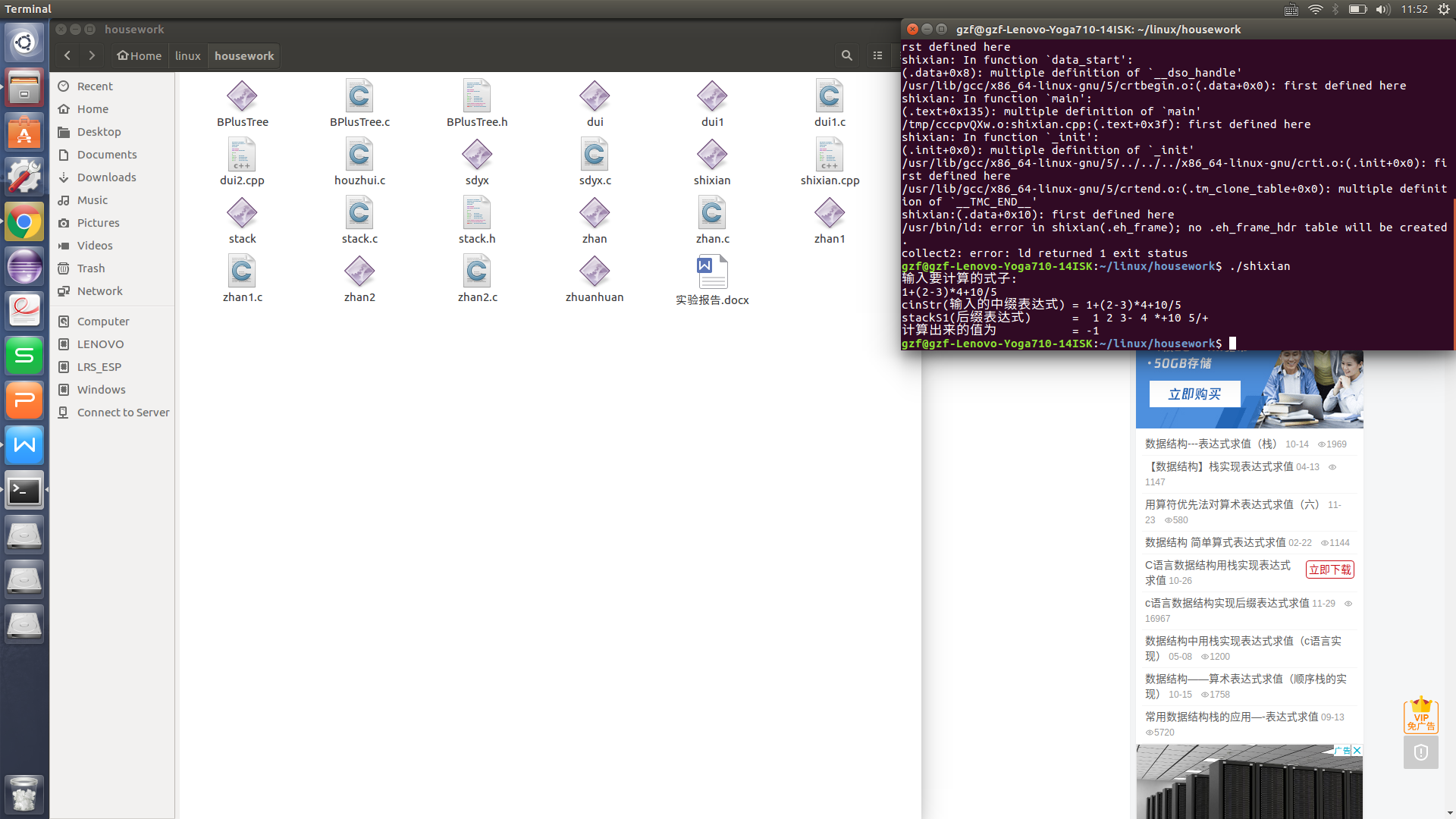
班级：硬件三班 学号：161403321 姓名：郭梓凡

1. **利用栈实现后缀表达式**
2. zhan.cpp
3. 输入形如1+(2-3)\*4+10/5的表达式，可包含‘+’，‘-’，‘\*’，‘/’，’(’,’)’。
4. 输出结果

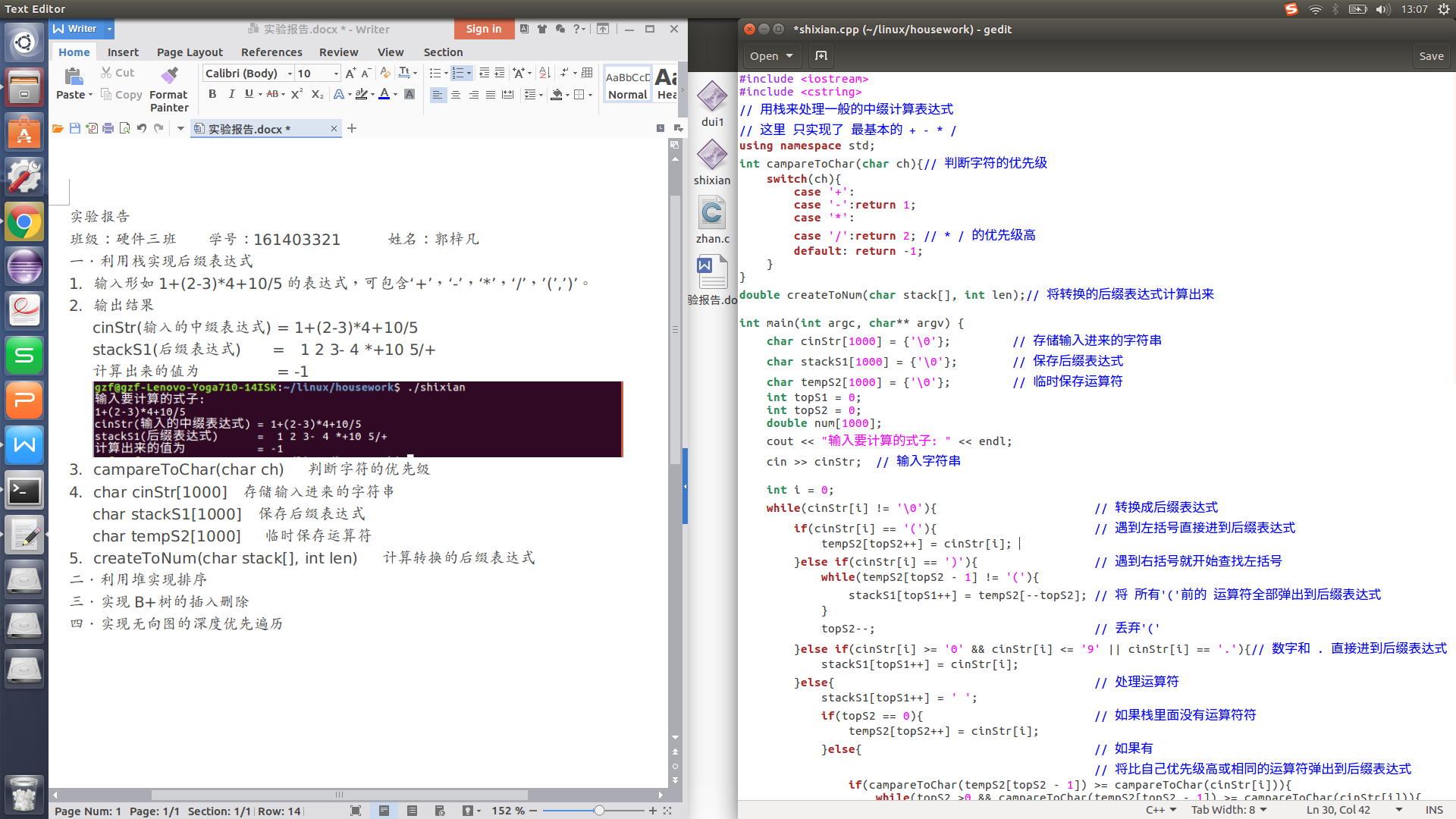
cinStr(输入的中缀表达式) = 1+(2-3)\*4+10/5

stackS1(后缀表达式) = 1 2 3- 4 \*+10 5/+

计算出来的值为 = -1



1. 先将中缀表达式转换为后缀表达式
2. 遇到操作数：直接输出（添加到后缀表达式中）。
3. 栈为空时，遇到运算符，直接入栈。
4. 遇到左括号：将其入栈。
5. 遇到右括号：执行出栈操作，并将出栈的元素输出，直到弹出栈的是左括号，左括号不输出。
6. 遇到其他运算符：加减乘除：弹出所有优先级大于或者等于该运算符的栈顶元素，然后将该运算符入栈。
7. 最终将栈中的元素依次出栈，输出。
8. campareToChar(char ch) 判断字符的优先级

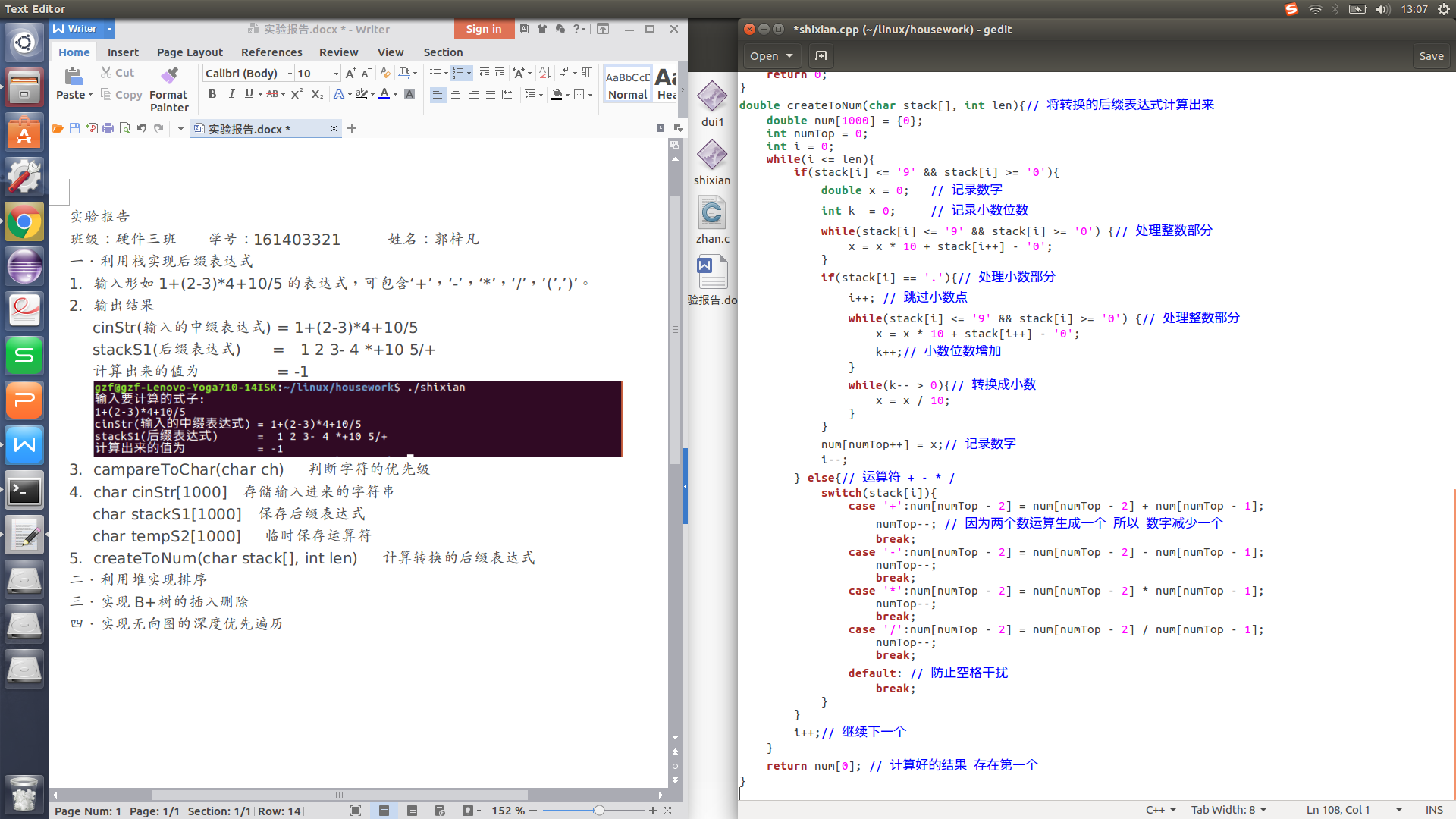


1. char cinStr[1000] 存储输入进来的字符串

char stackS1[1000] 保存后缀表达式

char tempS2[1000] 临时保存运算符

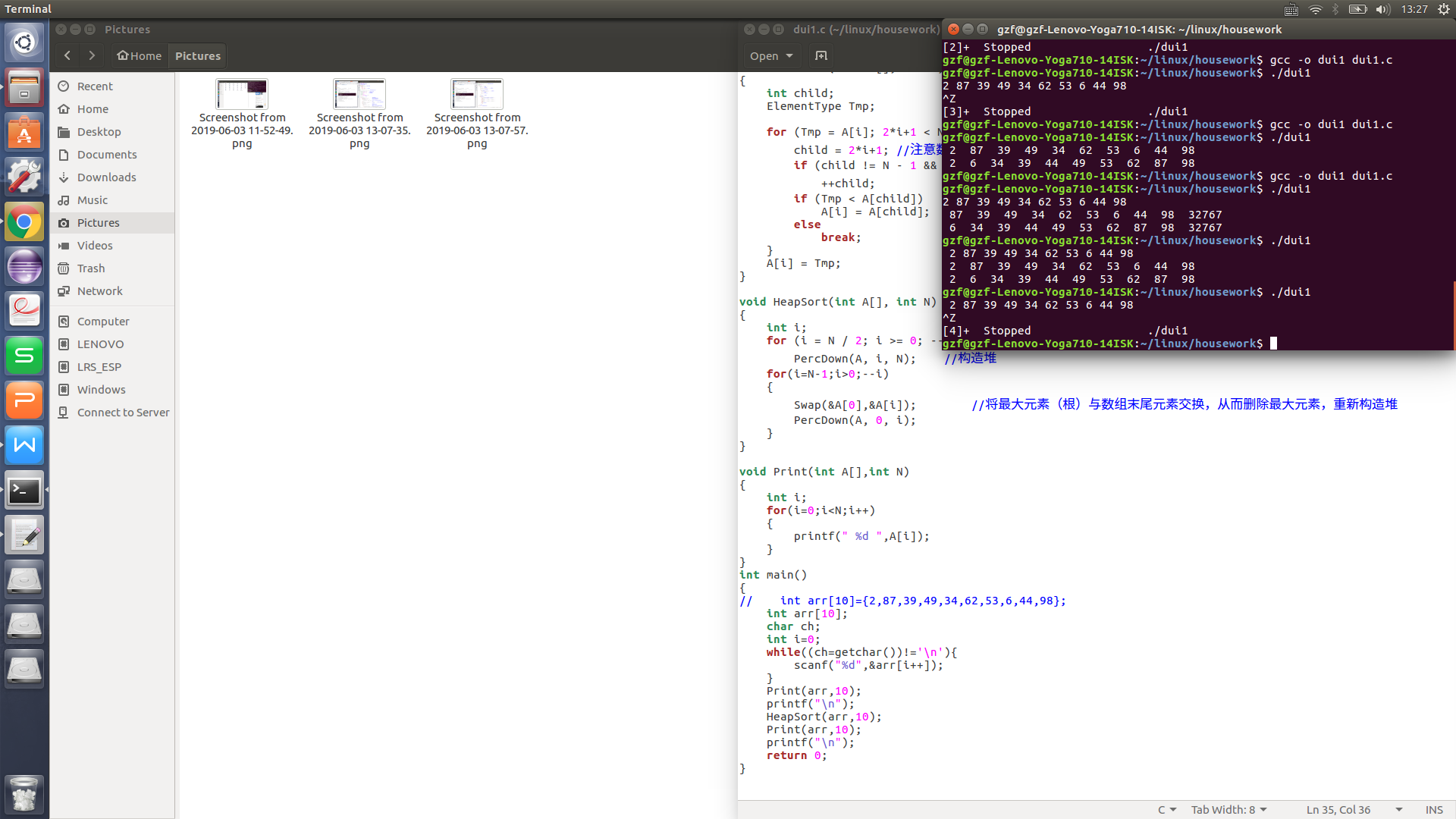
1. 计算后缀表达式
2. 如果是操作数，则放入栈中
3. 如果是操作符则取出栈中两个操作数，进行运算后，将结果放入栈中。
4. 直到最后栈中只有一个元素，此元素就是计算结果。
5. createToNum(char stack[], int len) 计算转换的后缀表达式

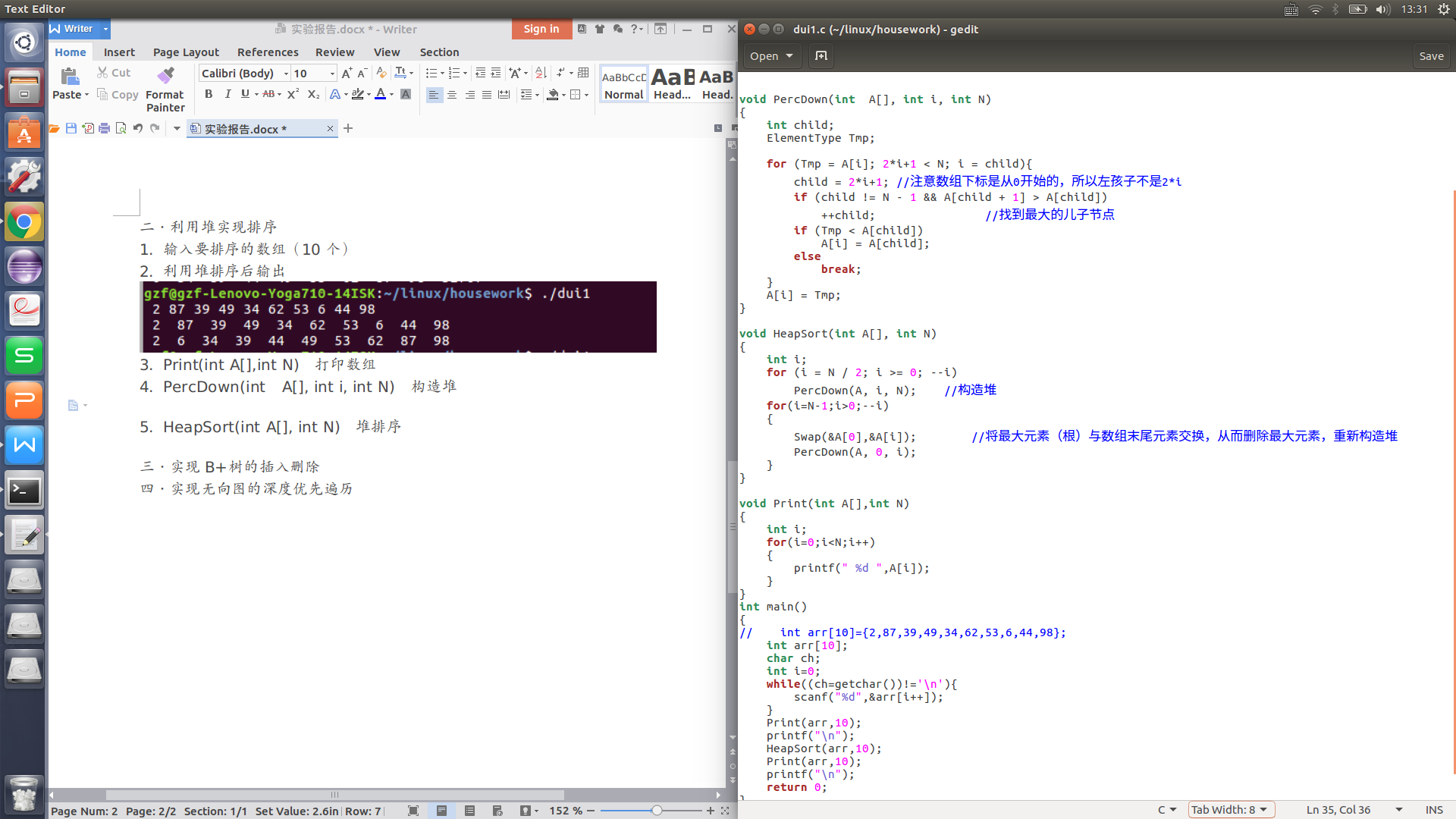


1. **利用堆实现排序**
2. dui.c
3. 堆实际上是一棵完全二叉树，其任何一非叶节点满足性质：

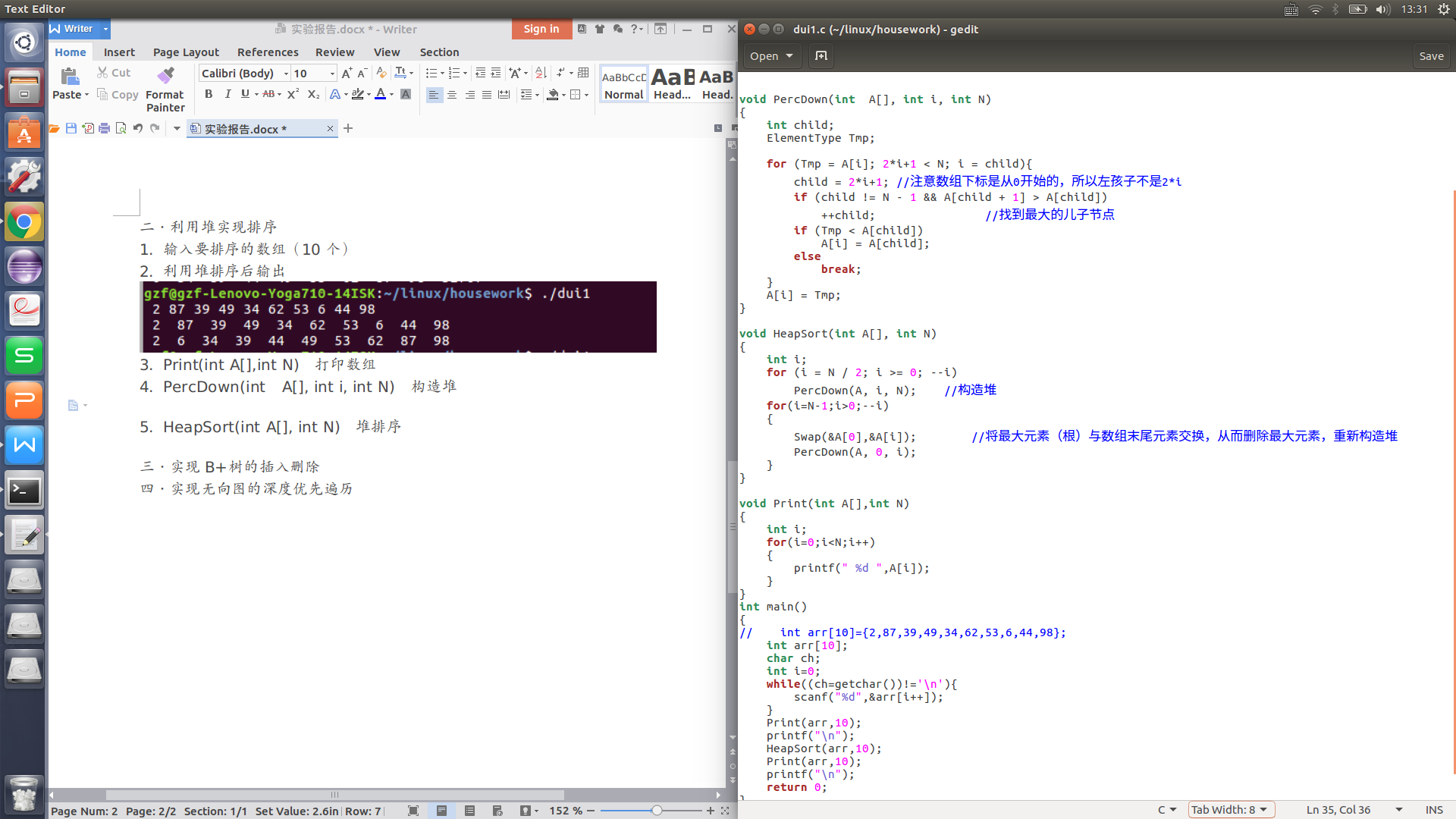
Key[i]<=key[2i+1]&&Key[i]<=key[2i+2]或Key[i]>=Key[2i+1]&&key>=key[2i+2]

即任何一非叶节点的关键字不大于或者不小于其左右孩子节点的关键字。

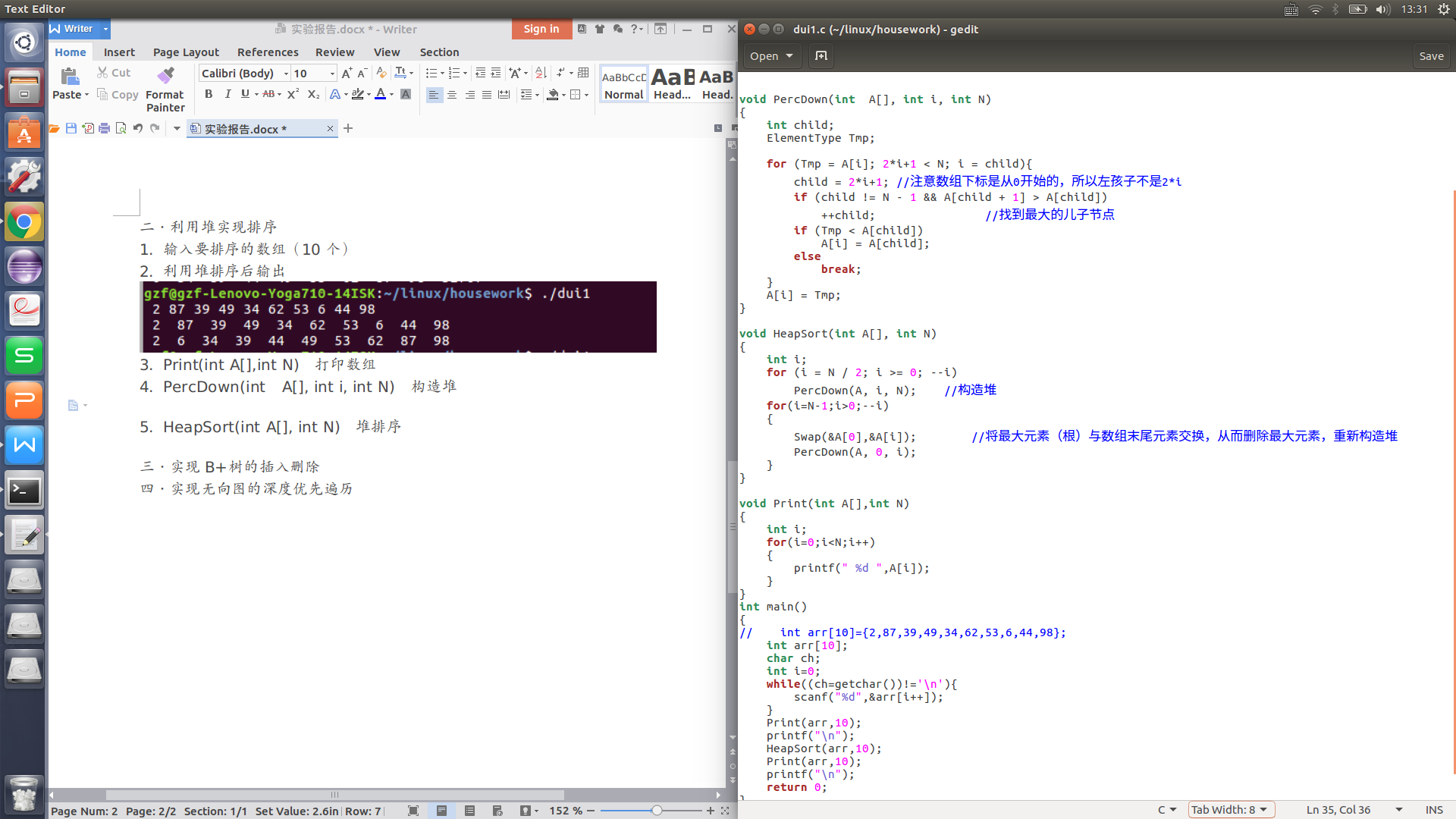
1. 基本思想
2. 将初始待排序关键字序列(R1,R2....Rn)构建成大顶堆，此堆为初始的无序区；
3. 将堆顶元素R[1]与最后一个元素R[n]交换，此时得到新的无序区(R1,R2,......Rn-1) 和新的有序区(Rn),且满足R[1,2...n-1]<=R[n];
4. 由于交换后新的堆顶R[1]可能违反堆的性质，因此需要对当前无序区(R1,R2,......Rn-1)调整为新堆，然后再次将R[1]与无序区最后一个元素交换，得到新的无序区(R1,R2....Rn-2)和新的有序区(Rn-1,Rn)。不断重复此过程直到有序区的元素个数为n-1，则整个排序过程完成。
5. 输入要排序的数组（10个）
6. 利用堆排序后输出
7. Print(int A[],int N) 打印数组



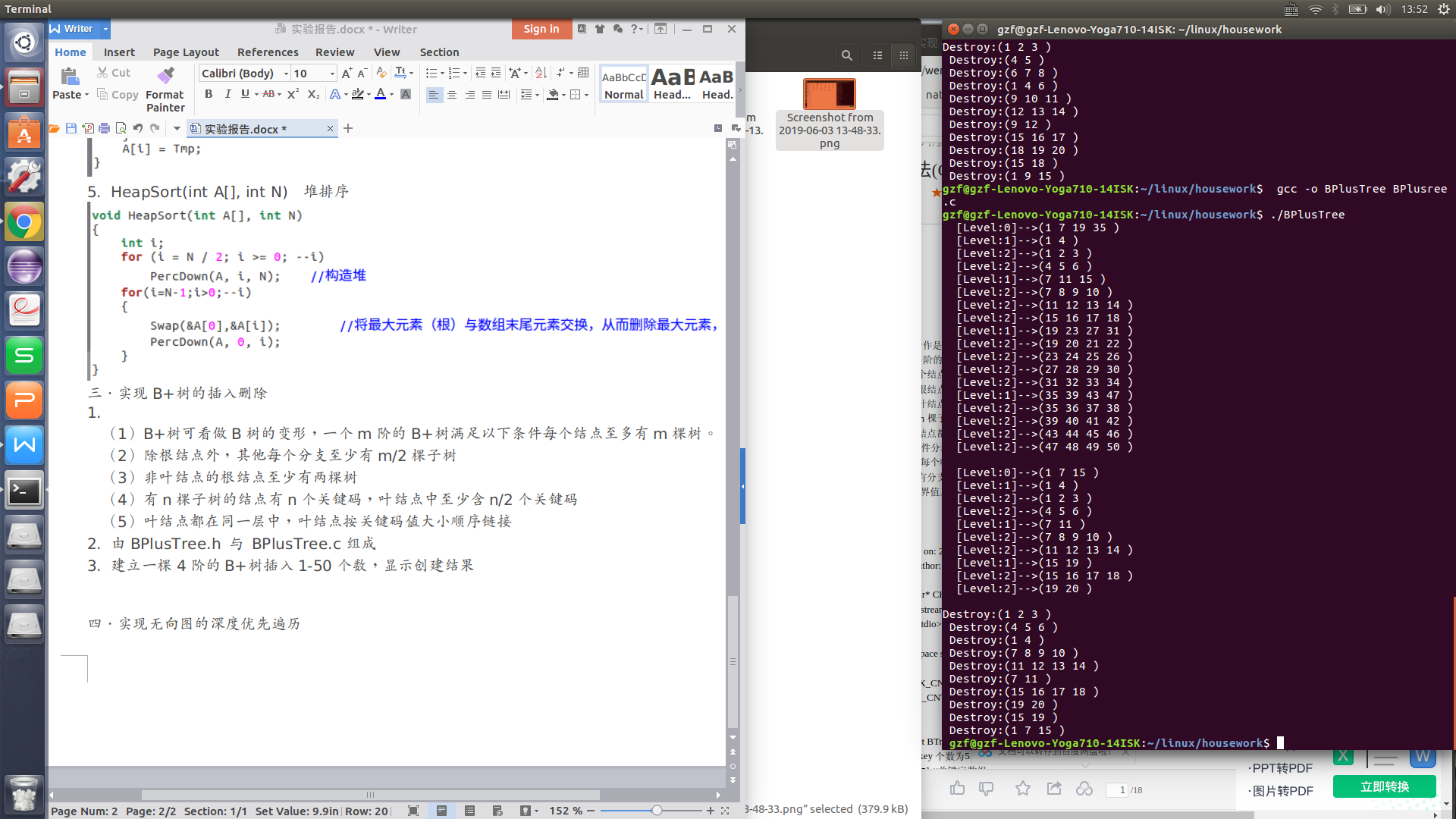
1. PercDown(int A[], int i, int N) 构造堆



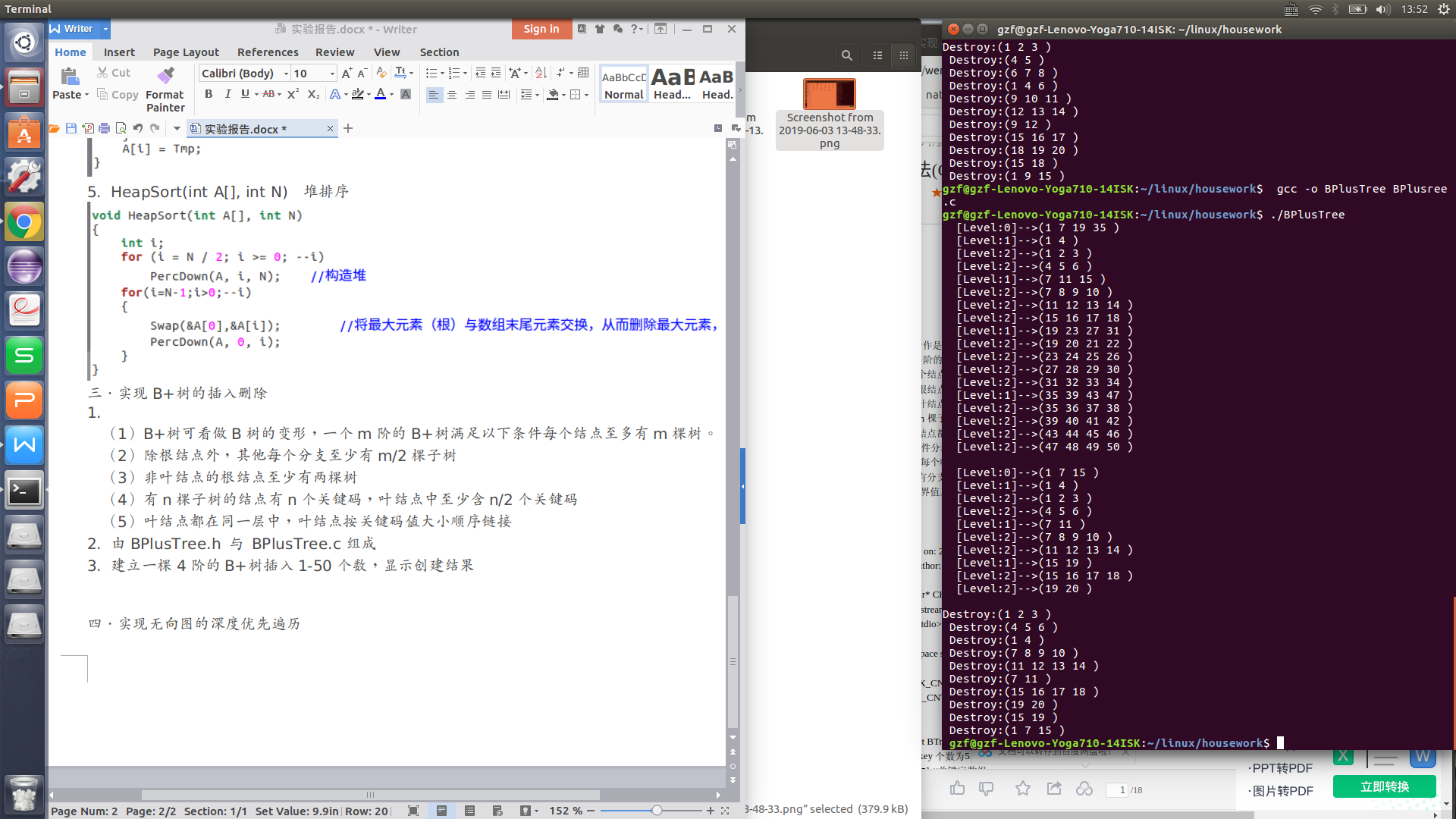
1. HeapSort(int A[], int N) 堆排序



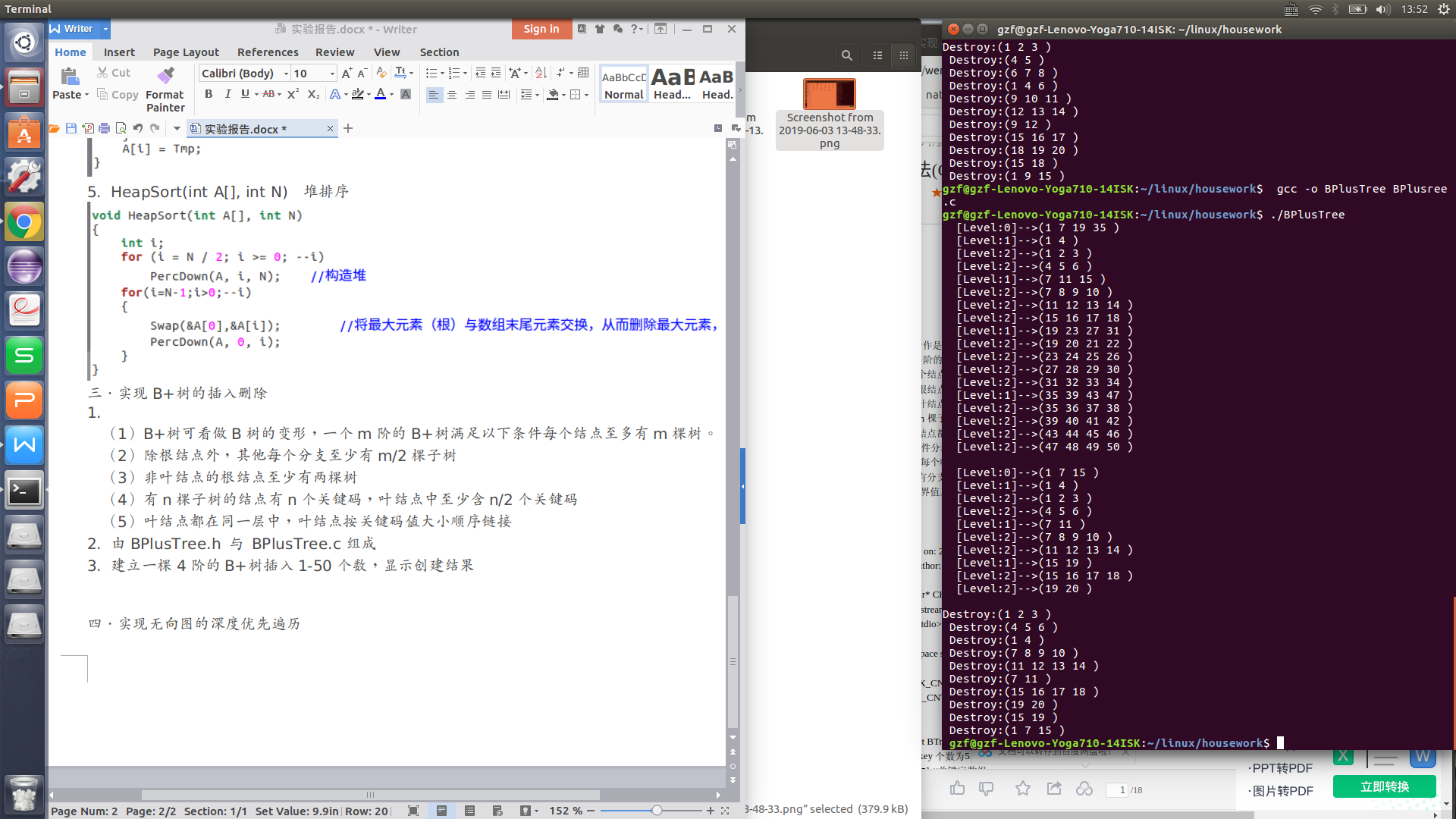
1. **实现B+树的插入删除**
2. BPlusTree.h BPlusTree.c
3. B+树
4. B+树可看做B树的变形，一个m阶的B+树满足以下条件每个结点至多有m棵树。
5. 除根结点外，其他每个分支至少有m/2棵子树
6. 非叶结点的根结点至少有两棵树
7. 有n棵子树的结点有n个关键码，叶结点中至少含n/2个关键码
8. 叶结点都在同一层中，叶结点按关键码值大小顺序链接
9. 建立一棵4阶的B+树插入1-50个数，显示创建结果



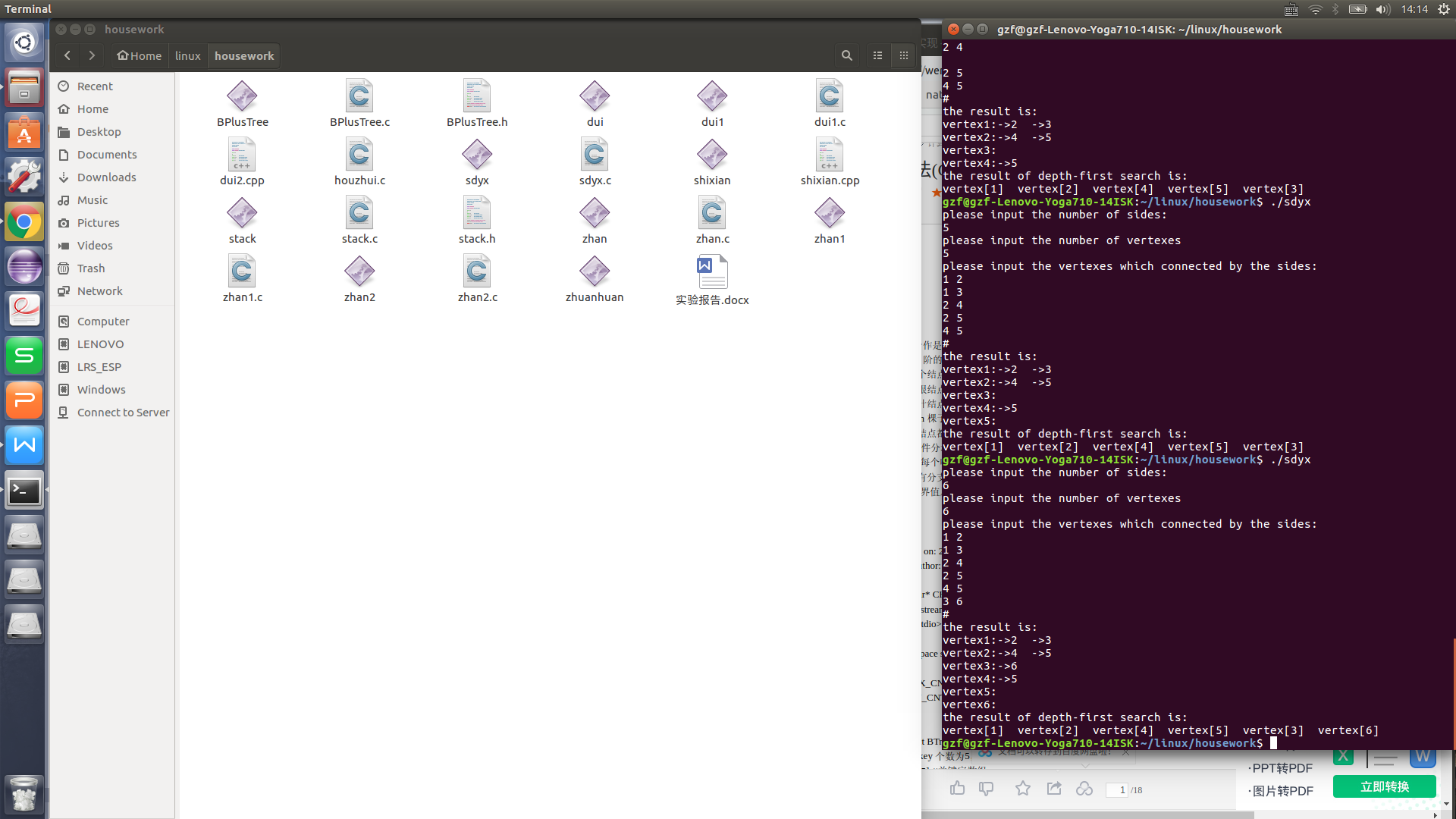
1. 删除20-50的数，显示结果



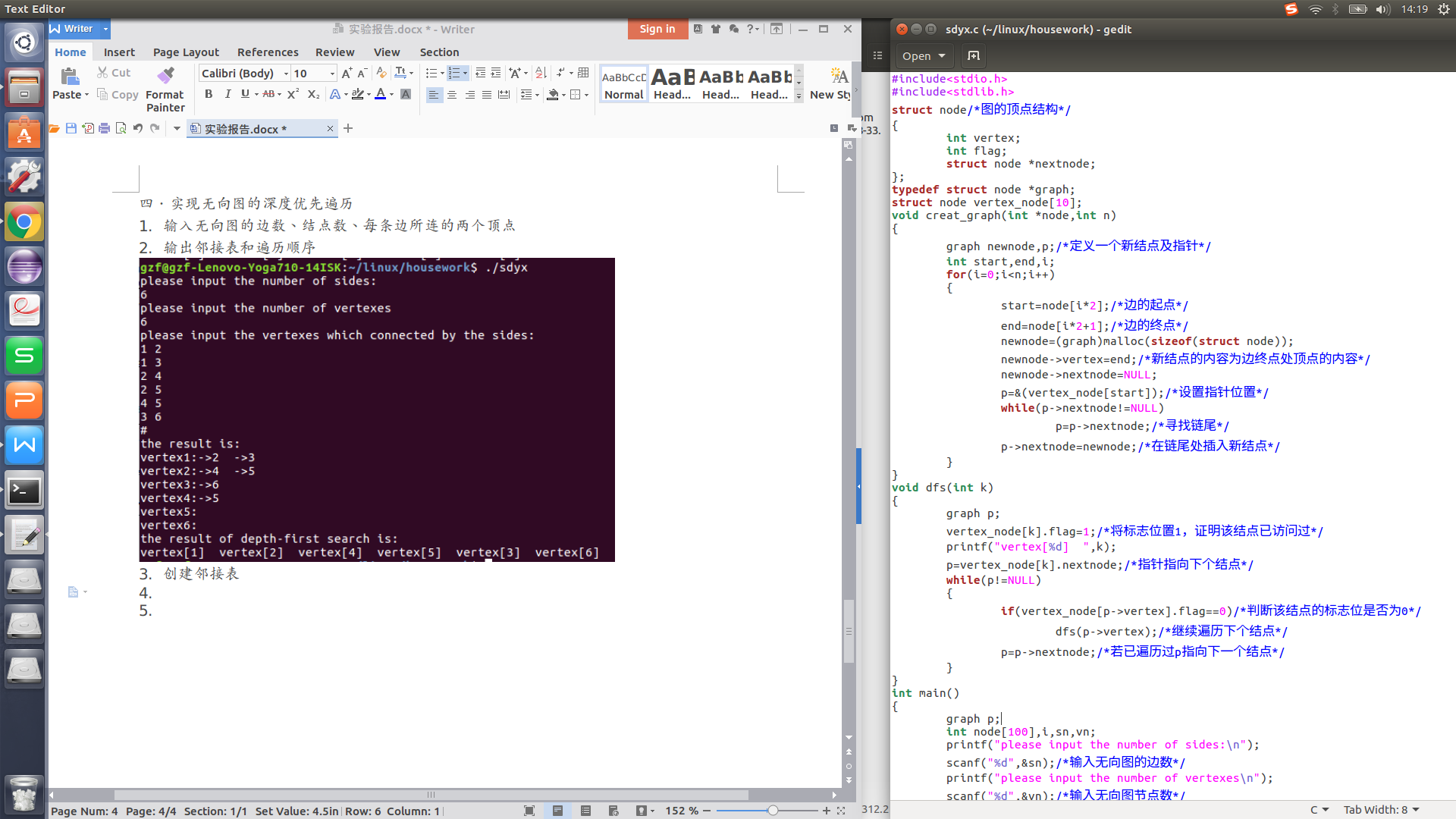
1. 显示销毁过程



1. **实现无向图的深度优先遍历**
2. sdyx.c
3. 输入无向图的边数、结点数、每条边所连的两个顶点
4. 输出邻接表和遍历顺序



1. creat\_graph(int \*node,int n) 创建邻接表



1. dfs(int k) 进行深度优先遍历（递归实现）

