# 《Linux 操作系统设计实践》

**实验报告**

## 实验 5：图形介绍

**院 系： 数学与计算机科学学院**

**专 业： 计算机科学与技术**

**年 级： 2016 级计算机 5 班**

**学 号： 031602507**

**姓 名： 陈俞辛**

### 一、实验环境：Ubuntu 16.04

### 二、实验内容：

**（一）实验代码**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <gtk/gtk.h>

#define STRING\_LEN 64

#define FILE\_NAME "ToDoList.txt"

GtkWidget \*uiList;

GtkWidget \*textToDo;

GtkWidget \*textDeadLine;

char selectText[STRING\_LEN\*2] = {0};

const gchar \*list\_item\_data\_key="list\_item\_data";

typedef struct \_Node

{

char name[STRING\_LEN];

char num[STRING\_LEN];

struct \_Node \*next;

}Node;

Node \*head = NULL; //链表头

Node \*tail = NULL; //链表尾

void DelList() //清空链表

{

if(head==NULL)

return;

Node \*p = head;

while(p)

{

Node \*tmp = p->next;

free(p);

p = tmp;

}

head = NULL;

tail = NULL;

}

void InitList() //初始化

{

DelList();

head = (Node \*)malloc(sizeof(Node));

head->next = NULL;

tail = head;

}

void InsertNode(Node \*node)//添加结点

{

if(head==NULL)

return;

tail->next = node;

tail = node;

tail->next = NULL;

}

void DelNode(const char \*str)//删除结点

{

if(head==NULL)

return;

Node \*pl = head, \*pr = head->next;

while(pr)

{

char buf[STRING\_LEN\*2]={0};

sprintf(buf,"ToDo:%s \t DeadLine:%s",pr->name,pr->num);

if(strcmp(buf,str)==0)

{

if(pr->next==NULL)

{

tail = pl;

}

pl->next = pr->next;

free(pr);

pr = pl->next;

break;

}

else

{

pl = pr;

pr = pr->next;

}

}

}

void ShowTip(const char \*info)

{

GtkWidget \*dialog = gtk\_message\_dialog\_new(NULL,GTK\_DIALOG\_MODAL|GTK\_DIALOG\_DESTROY\_WITH\_PARENT,GTK\_MESSAGE\_INFO,GTK\_BUTTONS\_OK,"%s",info);

gtk\_dialog\_run(GTK\_DIALOG(dialog));

gtk\_widget\_destroy(dialog);

}

void ReadFile()//从文件读取ToDoList

{

FILE \*fp = fopen(FILE\_NAME,"r");

if(fp==NULL)

return;

DelList();

InitList();

char ToDoStr[STRING\_LEN],DDLStr[STRING\_LEN];

while(fscanf(fp,"%s %s",ToDoStr,DDLStr)==2)

{

Node \*pnew = (Node \*)malloc(sizeof(Node));

strcpy(pnew->name,ToDoStr);

strcpy(pnew->num,DDLStr);

InsertNode(pnew);

}

fclose(fp);

}

void WriteFile()//保存数据到ToDoList文件

{

FILE \*fp = fopen(FILE\_NAME,"w");

if(fp==NULL)

return;

Node \*p = head->next;

while(p)

{

fprintf(fp,"%s %s\n",p->name,p->num);

p = p->next;

}

fclose(fp);

}

void AddNodeToList(const char \*ToDoStr, const char \*DDLStr)//向uiList中加入元素

{

char str[STRING\_LEN\*2] = {0};

sprintf(str, "ToDo:%s \t DeadLine:%s",ToDoStr,DDLStr);

gchar \*string;

GtkWidget \*label=gtk\_label\_new (str);

GtkWidget \*list\_item = gtk\_list\_item\_new ();

gtk\_container\_add (GTK\_CONTAINER (list\_item), label);

gtk\_widget\_show (label);

gtk\_container\_add (GTK\_CONTAINER (uiList), list\_item);

gtk\_widget\_show (list\_item);

gtk\_label\_get (GTK\_LABEL (label), &string);

g\_object\_set\_data (G\_OBJECT (list\_item), list\_item\_data\_key, string);

}

void onBtnInsert(GtkWidget \*widget, gpointer data)//点击ADD按钮

{

const char \*ToDoStr = gtk\_entry\_get\_text((GtkEntry \*)textToDo);

const char \*DDLStr = gtk\_entry\_get\_text((GtkEntry \*)textDeadLine);

if(strlen(ToDoStr)==0 || strlen(DDLStr)==0)

return;

Node \*pnew = (Node \*)malloc(sizeof(Node));

sprintf(pnew->name,"%s",ToDoStr);

sprintf(pnew->num,"%s",DDLStr);

InsertNode(pnew);

AddNodeToList(ToDoStr,DDLStr);

gtk\_entry\_set\_text((GtkEntry \*)textToDo,"");

gtk\_entry\_set\_text((GtkEntry \*)textDeadLine,"");

ShowTip("ADD SUCCESSED");

}

void onBtnDelete(GtkWidget \*widget, gpointer data)//点击Finished按钮

{

gtk\_list\_clear\_items(GTK\_LIST(uiList),0,-1);

DelNode(selectText);

Node \*p = head->next;

while(p)

{

AddNodeToList(p->name,p->num);

p = p->next;

}

ShowTip("Congratulations!");

}

void onBtnSave(GtkWidget \*widget, gpointer data)//点击SAVE按钮

{

WriteFile();

if(widget!=0 || data!=0)

ShowTip("SAVE SUCCESSED");

}

void onBtnRead(GtkWidget \*widget, gpointer data)//READ按钮

{

gtk\_list\_clear\_items(GTK\_LIST(uiList),0,-1);

ReadFile();

Node \*p = head->next;

while(p)

{

AddNodeToList(p->name,p->num);

p = p->next;

}

if(widget!=0 || data!=0)

ShowTip("READ SUCCESSED");

}

void onBtnExit(GtkWidget \*widget, gpointer data)//点击EXIT按钮

{

gtk\_main\_quit();

onBtnSave(0,0);

}

void OnListSelectionChanged(GtkWidget \*gtklist, gpointer func\_data)//选择的uiList项目发生变化触发

{

GList \*dlist = GTK\_LIST(gtklist)->selection;

if (!dlist)

{

return;

}

else

{

if(dlist!=NULL)

{

const char \*buf = g\_object\_get\_data(G\_OBJECT(dlist->data),list\_item\_data\_key);

sprintf(selectText,"%s",buf);

}

}

}

void UI\_Init(int argc, char \*argv[])//初始化界面

{

gtk\_init(&argc,&argv);

GtkWidget \*window=gtk\_window\_new(GTK\_WINDOW\_TOPLEVEL);

gtk\_window\_set\_default\_size(GTK\_WINDOW(window),100,250);

g\_signal\_connect(G\_OBJECT(window),"delete\_event",G\_CALLBACK(gtk\_main\_quit),NULL);

gtk\_window\_set\_title(GTK\_WINDOW(window),"ToDoList");

gtk\_window\_set\_position(GTK\_WINDOW(window),GTK\_WIN\_POS\_CENTER);

gtk\_container\_set\_border\_width(GTK\_CONTAINER(window),10);

//--------------------垂直线性布局----------------------

GtkWidget \*vbox=gtk\_vbox\_new(0,0);

gtk\_container\_add(GTK\_CONTAINER(window),vbox);

//------------------------------------------------------

GtkWidget \*scrolled\_window = gtk\_scrolled\_window\_new (NULL, NULL);//可拖动窗口

gtk\_widget\_set\_size\_request (scrolled\_window, 250, 150);

gtk\_container\_add (GTK\_CONTAINER (vbox), scrolled\_window);

uiList = gtk\_list\_new();//放于可拖动窗口中的uiList

gtk\_scrolled\_window\_add\_with\_viewport (GTK\_SCROLLED\_WINDOW (scrolled\_window),uiList);

g\_signal\_connect (uiList, "selection-changed", G\_CALLBACK (OnListSelectionChanged), NULL);

//------------------水平布局---------------------------

GtkWidget \*hbox1 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox1,FALSE,FALSE,5);

GtkWidget \*lab\_name = gtk\_label\_new("ToDo");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),lab\_name,1,0,5);

textToDo = gtk\_entry\_new();

gtk\_box\_pack\_start(GTK\_BOX(hbox1),textToDo,1,0,5);

GtkWidget \*lab\_num = gtk\_label\_new("DeadLine");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),lab\_num,1,0,5);

textDeadLine = gtk\_entry\_new();

gtk\_box\_pack\_start(GTK\_BOX(hbox1),textDeadLine,1,0,5);

GtkWidget \*btn\_add = gtk\_button\_new\_with\_label("ADD");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),btn\_add,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_add),"clicked",G\_CALLBACK(onBtnInsert),NULL);

//-------放入Finished、SAVE、EXIT3个按钮--------

GtkWidget \*hbox2 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox2,FALSE,FALSE,5);

GtkWidget \*btn\_delete = gtk\_button\_new\_with\_label("Finished");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_delete,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_delete),"clicked",G\_CALLBACK(onBtnDelete),NULL);

GtkWidget \*btn\_save = gtk\_button\_new\_with\_label("SAVE");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_save,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_save),"clicked",G\_CALLBACK(onBtnSave),NULL);

GtkWidget \*btn\_exit = gtk\_button\_new\_with\_label("EXIT");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_exit,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_exit),"clicked",G\_CALLBACK(onBtnExit),NULL);

//------------------------------------------------------

onBtnRead(0,0);//从文件读取数据放入链表

gtk\_widget\_show\_all(window);//显示所有窗体

gtk\_main();

}

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

{

InitList(); //初始化链表

UI\_Init(argc, argv);//初始化UI

DelList(); //释放链表

return 0;

}

void InsertNode(Node \*node)//添加结点

{

if(head==NULL)

return;

tail->next = node;

tail = node;

tail->next = NULL;

}

void DelNode(const char \*str)//删除结点

{

if(head==NULL)

return;

Node \*pl = head, \*pr = head->next;

while(pr)

{

char buf[STRING\_LEN\*2]={0};

sprintf(buf,"ToDo:%s \t DeadLine:%s",pr->name,pr->num);

if(strcmp(buf,str)==0)

{

if(pr->next==NULL)

{

tail = pl;

}

pl->next = pr->next;

free(pr);

pr = pl->next;

break;

}

else

{

pl = pr;

pr = pr->next;

}

}

}

void ShowTip(const char \*info)

{

GtkWidget \*dialog = gtk\_message\_dialog\_new(NULL,GTK\_DIALOG\_MODAL|GTK\_DIALOG\_DESTROY\_WITH\_PARENT,GTK\_MESSAGE\_INFO,GTK\_BUTTONS\_OK,"%s",info);

gtk\_dialog\_run(GTK\_DIALOG(dialog));

gtk\_widget\_destroy(dialog);

}

void ReadFile()//从文件读取ToDoList

{

FILE \*fp = fopen(FILE\_NAME,"r");

if(fp==NULL)

return;

DelList();

InitList();

char ToDoStr[STRING\_LEN],DDLStr[STRING\_LEN];

while(fscanf(fp,"%s %s",ToDoStr,DDLStr)==2)

{

Node \*pnew = (Node \*)malloc(sizeof(Node));

strcpy(pnew->name,ToDoStr);

strcpy(pnew->num,DDLStr);

InsertNode(pnew);

}

fclose(fp);

}

void WriteFile()//保存数据到ToDoList文件

{

FILE \*fp = fopen(FILE\_NAME,"w");

if(fp==NULL)

return;

Node \*p = head->next;

while(p)

{

fprintf(fp,"%s %s\n",p->name,p->num);

p = p->next;

}

fclose(fp);

}

void AddNodeToList(const char \*ToDoStr, const char \*DDLStr)//向uiList中加入元素

{

char str[STRING\_LEN\*2] = {0};

sprintf(str, "ToDo:%s \t DeadLine:%s",ToDoStr,DDLStr);

gchar \*string;

GtkWidget \*label=gtk\_label\_new (str);

GtkWidget \*list\_item = gtk\_list\_item\_new ();

gtk\_container\_add (GTK\_CONTAINER (list\_item), label);

gtk\_widget\_show (label);

gtk\_container\_add (GTK\_CONTAINER (uiList), list\_item);

gtk\_widget\_show (list\_item);

gtk\_label\_get (GTK\_LABEL (label), &string);

g\_object\_set\_data (G\_OBJECT (list\_item), list\_item\_data\_key, string);

}

void onBtnInsert(GtkWidget \*widget, gpointer data)//点击ADD按钮

{

const char \*ToDoStr = gtk\_entry\_get\_text((GtkEntry \*)textToDo);

const char \*DDLStr = gtk\_entry\_get\_text((GtkEntry \*)textDeadLine);

if(strlen(ToDoStr)==0 || strlen(DDLStr)==0)

return;

Node \*pnew = (Node \*)malloc(sizeof(Node));

sprintf(pnew->name,"%s",ToDoStr);

sprintf(pnew->num,"%s",DDLStr);

InsertNode(pnew);

AddNodeToList(ToDoStr,DDLStr);

gtk\_entry\_set\_text((GtkEntry \*)textToDo,"");

gtk\_entry\_set\_text((GtkEntry \*)textDeadLine,"");

ShowTip("ADD SUCCESSED");

}

void onBtnDelete(GtkWidget \*widget, gpointer data)//点击Finished按钮

{

gtk\_list\_clear\_items(GTK\_LIST(uiList),0,-1);

DelNode(selectText);

Node \*p = head->next;

while(p)

{

AddNodeToList(p->name,p->num);

p = p->next;

}

ShowTip("Congratulations!");

}

void onBtnSave(GtkWidget \*widget, gpointer data)//点击SAVE按钮

{

WriteFile();

if(widget!=0 || data!=0)

ShowTip("SAVE SUCCESSED");

}

void onBtnRead(GtkWidget \*widget, gpointer data)//READ按钮

{

gtk\_list\_clear\_items(GTK\_LIST(uiList),0,-1);

ReadFile();

Node \*p = head->next;

while(p)

{

AddNodeToList(p->name,p->num);

p = p->next;

}

if(widget!=0 || data!=0)

ShowTip("READ SUCCESSED");

}

void onBtnExit(GtkWidget \*widget, gpointer data)//点击EXIT按钮

{

gtk\_main\_quit();

onBtnSave(0,0);

}

void OnListSelectionChanged(GtkWidget \*gtklist, gpointer func\_data)//选择的uiList项目发生变化触发

{

GList \*dlist = GTK\_LIST(gtklist)->selection;

if (!dlist)

{

return;

}

else

{

if(dlist!=NULL)

{

const char \*buf = g\_object\_get\_data(G\_OBJECT(dlist->data),list\_item\_data\_key);

sprintf(selectText,"%s",buf);

}

}

}

void UI\_Init(int argc, char \*argv[])//初始化界面

{

gtk\_init(&argc,&argv);

GtkWidget \*window=gtk\_window\_new(GTK\_WINDOW\_TOPLEVEL);

gtk\_window\_set\_default\_size(GTK\_WINDOW(window),100,250);

g\_signal\_connect(G\_OBJECT(window),"delete\_event",G\_CALLBACK(gtk\_main\_quit),NULL);

gtk\_window\_set\_title(GTK\_WINDOW(window),"ToDoList");

gtk\_window\_set\_position(GTK\_WINDOW(window),GTK\_WIN\_POS\_CENTER);

gtk\_container\_set\_border\_width(GTK\_CONTAINER(window),10);

//--------------------垂直线性布局----------------------

GtkWidget \*vbox=gtk\_vbox\_new(0,0);

gtk\_container\_add(GTK\_CONTAINER(window),vbox);

//------------------------------------------------------

GtkWidget \*scrolled\_window = gtk\_scrolled\_window\_new (NULL, NULL);//可拖动窗口

gtk\_widget\_set\_size\_request (scrolled\_window, 250, 150);

gtk\_container\_add (GTK\_CONTAINER (vbox), scrolled\_window);

uiList = gtk\_list\_new();//放于可拖动窗口中的uiList

gtk\_scrolled\_window\_add\_with\_viewport (GTK\_SCROLLED\_WINDOW (scrolled\_window),uiList);

g\_signal\_connect (uiList, "selection-changed", G\_CALLBACK (OnListSelectionChanged), NULL);

//------------------水平布局---------------------------

GtkWidget \*hbox1 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox1,FALSE,FALSE,5);

GtkWidget \*lab\_name = gtk\_label\_new("ToDo");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),lab\_name,1,0,5);

textToDo = gtk\_entry\_new();

gtk\_box\_pack\_start(GTK\_BOX(hbox1),textToDo,1,0,5);

GtkWidget \*lab\_num = gtk\_label\_new("DeadLine");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),lab\_num,1,0,5);

textDeadLine = gtk\_entry\_new();

gtk\_box\_pack\_start(GTK\_BOX(hbox1),textDeadLine,1,0,5);

GtkWidget \*btn\_add = gtk\_button\_new\_with\_label("ADD");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),btn\_add,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_add),"clicked",G\_CALLBACK(onBtnInsert),NULL);

//-------放入Finished、SAVE、EXIT3个按钮--------

GtkWidget \*hbox2 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox2,FALSE,FALSE,5);

GtkWidget \*btn\_delete = gtk\_button\_new\_with\_label("Finished");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_delete,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_delete),"clicked",G\_CALLBACK(onBtnDelete),NULL);

GtkWidget \*btn\_save = gtk\_button\_new\_with\_label("SAVE");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_save,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_save),"clicked",G\_CALLBACK(onBtnSave),NULL);

GtkWidget \*btn\_exit = gtk\_button\_new\_with\_label("EXIT");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_exit,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_exit),"clicked",G\_CALLBACK(onBtnExit),NULL);

//------------------------------------------------------

onBtnRead(0,0);//从文件读取数据放入链表

gtk\_widget\_show\_all(window);//显示所有窗体

gtk\_main();

}

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

{

InitList(); //初始化链表

UI\_Init(argc, argv);//初始化UI

DelList(); //释放链表

return 0;

}

void ReadFile()//从文件读取ToDoList

{

FILE \*fp = fopen(FILE\_NAME,"r");

if(fp==NULL)

return;

DelList();

InitList();

char ToDoStr[STRING\_LEN],DDLStr[STRING\_LEN];

while(fscanf(fp,"%s %s",ToDoStr,DDLStr)==2)

{

Node \*pnew = (Node \*)malloc(sizeof(Node));

strcpy(pnew->name,ToDoStr);

strcpy(pnew->num,DDLStr);

InsertNode(pnew);

}

fclose(fp);

}

void WriteFile()//保存数据到ToDoList文件

{

FILE \*fp = fopen(FILE\_NAME,"w");

if(fp==NULL)

return;

Node \*p = head->next;

while(p)

{

fprintf(fp,"%s %s\n",p->name,p->num);

p = p->next;

}

fclose(fp);

}

void AddNodeToList(const char \*ToDoStr, const char \*DDLStr)//向uiList中加入元素

{

char str[STRING\_LEN\*2] = {0};

sprintf(str, "ToDo:%s \t DeadLine:%s",ToDoStr,DDLStr);

gchar \*string;

GtkWidget \*label=gtk\_label\_new (str);

GtkWidget \*list\_item = gtk\_list\_item\_new ();

gtk\_container\_add (GTK\_CONTAINER (list\_item), label);

gtk\_widget\_show (label);

gtk\_container\_add (GTK\_CONTAINER (uiList), list\_item);

gtk\_widget\_show (list\_item);

gtk\_label\_get (GTK\_LABEL (label), &string);

g\_object\_set\_data (G\_OBJECT (list\_item), list\_item\_data\_key, string);

}

void onBtnInsert(GtkWidget \*widget, gpointer data)//点击ADD按钮

{

const char \*ToDoStr = gtk\_entry\_get\_text((GtkEntry \*)textToDo);

const char \*DDLStr = gtk\_entry\_get\_text((GtkEntry \*)textDeadLine);

if(strlen(ToDoStr)==0 || strlen(DDLStr)==0)

return;

Node \*pnew = (Node \*)malloc(sizeof(Node));

sprintf(pnew->name,"%s",ToDoStr);

sprintf(pnew->num,"%s",DDLStr);

InsertNode(pnew);

AddNodeToList(ToDoStr,DDLStr);

gtk\_entry\_set\_text((GtkEntry \*)textToDo,"");

gtk\_entry\_set\_text((GtkEntry \*)textDeadLine,"");

ShowTip("ADD SUCCESSED");

}

void onBtnDelete(GtkWidget \*widget, gpointer data)//点击Finished按钮

{

gtk\_list\_clear\_items(GTK\_LIST(uiList),0,-1);

DelNode(selectText);

Node \*p = head->next;

while(p)

{

AddNodeToList(p->name,p->num);

p = p->next;

}

ShowTip("Congratulations!");

}

void onBtnSave(GtkWidget \*widget, gpointer data)//点击SAVE按钮

{

WriteFile();

if(widget!=0 || data!=0)

ShowTip("SAVE SUCCESSED");

}

void onBtnRead(GtkWidget \*widget, gpointer data)//READ按钮

{

gtk\_list\_clear\_items(GTK\_LIST(uiList),0,-1);

ReadFile();

Node \*p = head->next;

while(p)

{

AddNodeToList(p->name,p->num);

p = p->next;

}

if(widget!=0 || data!=0)

ShowTip("READ SUCCESSED");

}

void onBtnExit(GtkWidget \*widget, gpointer data)//点击EXIT按钮

{

gtk\_main\_quit();

onBtnSave(0,0);

}

void OnListSelectionChanged(GtkWidget \*gtklist, gpointer func\_data)//选择的uiList项目发生变化触发

{

GList \*dlist = GTK\_LIST(gtklist)->selection;

if (!dlist)

{

return;

}

else

{

if(dlist!=NULL)

{

const char \*buf = g\_object\_get\_data(G\_OBJECT(dlist->data),list\_item\_data\_key);

sprintf(selectText,"%s",buf);

}

}

}

void UI\_Init(int argc, char \*argv[])//初始化界面

{

gtk\_init(&argc,&argv);

GtkWidget \*window=gtk\_window\_new(GTK\_WINDOW\_TOPLEVEL);

gtk\_window\_set\_default\_size(GTK\_WINDOW(window),100,250);

g\_signal\_connect(G\_OBJECT(window),"delete\_event",G\_CALLBACK(gtk\_main\_quit),NULL);

gtk\_window\_set\_title(GTK\_WINDOW(window),"ToDoList");

gtk\_window\_set\_position(GTK\_WINDOW(window),GTK\_WIN\_POS\_CENTER);

gtk\_container\_set\_border\_width(GTK\_CONTAINER(window),10);

//--------------------垂直线性布局----------------------

GtkWidget \*vbox=gtk\_vbox\_new(0,0);

gtk\_container\_add(GTK\_CONTAINER(window),vbox);

//------------------------------------------------------

GtkWidget \*scrolled\_window = gtk\_scrolled\_window\_new (NULL, NULL);//可拖动窗口

gtk\_widget\_set\_size\_request (scrolled\_window, 250, 150);

gtk\_container\_add (GTK\_CONTAINER (vbox), scrolled\_window);

uiList = gtk\_list\_new();//放于可拖动窗口中的uiList

gtk\_scrolled\_window\_add\_with\_viewport (GTK\_SCROLLED\_WINDOW (scrolled\_window),uiList);

g\_signal\_connect (uiList, "selection-changed", G\_CALLBACK (OnListSelectionChanged), NULL);

//------------------水平布局---------------------------

GtkWidget \*hbox1 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox1,FALSE,FALSE,5);

GtkWidget \*lab\_name = gtk\_label\_new("ToDo");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),lab\_name,1,0,5);

textToDo = gtk\_entry\_new();

gtk\_box\_pack\_start(GTK\_BOX(hbox1),textToDo,1,0,5);

GtkWidget \*lab\_num = gtk\_label\_new("DeadLine");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),lab\_num,1,0,5);

textDeadLine = gtk\_entry\_new();

gtk\_box\_pack\_start(GTK\_BOX(hbox1),textDeadLine,1,0,5);

GtkWidget \*btn\_add = gtk\_button\_new\_with\_label("ADD");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),btn\_add,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_add),"clicked",G\_CALLBACK(onBtnInsert),NULL);

//-------放入Finished、SAVE、EXIT3个按钮--------

GtkWidget \*hbox2 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox2,FALSE,FALSE,5);

GtkWidget \*btn\_delete = gtk\_button\_new\_with\_label("Finished");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_delete,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_delete),"clicked",G\_CALLBACK(onBtnDelete),NULL);

GtkWidget \*btn\_save = gtk\_button\_new\_with\_label("SAVE");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_save,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_save),"clicked",G\_CALLBACK(onBtnSave),NULL);

GtkWidget \*btn\_exit = gtk\_button\_new\_with\_label("EXIT");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_exit,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_exit),"clicked",G\_CALLBACK(onBtnExit),NULL);

//------------------------------------------------------

onBtnRead(0,0);//从文件读取数据放入链表

gtk\_widget\_show\_all(window);//显示所有窗体

gtk\_main();

}

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

{

InitList(); //初始化链表

UI\_Init(argc, argv);//初始化UI

DelList(); //释放链表

return 0;

}

void onBtnInsert(GtkWidget \*widget, gpointer data)//点击ADD按钮

{

const char \*ToDoStr = gtk\_entry\_get\_text((GtkEntry \*)textToDo);

const char \*DDLStr = gtk\_entry\_get\_text((GtkEntry \*)textDeadLine);

if(strlen(ToDoStr)==0 || strlen(DDLStr)==0)

return;

Node \*pnew = (Node \*)malloc(sizeof(Node));

sprintf(pnew->name,"%s",ToDoStr);

sprintf(pnew->num,"%s",DDLStr);

InsertNode(pnew);

AddNodeToList(ToDoStr,DDLStr);

gtk\_entry\_set\_text((GtkEntry \*)textToDo,"");

gtk\_entry\_set\_text((GtkEntry \*)textDeadLine,"");

ShowTip("ADD SUCCESSED");

}

void onBtnDelete(GtkWidget \*widget, gpointer data)//点击Finished按钮

{

gtk\_list\_clear\_items(GTK\_LIST(uiList),0,-1);

DelNode(selectText);

Node \*p = head->next;

while(p)

{

AddNodeToList(p->name,p->num);

p = p->next;

}

ShowTip("Congratulations!");

}

void onBtnSave(GtkWidget \*widget, gpointer data)//点击SAVE按钮

{

WriteFile();

if(widget!=0 || data!=0)

ShowTip("SAVE SUCCESSED");

}

void onBtnRead(GtkWidget \*widget, gpointer data)//READ按钮

{

gtk\_list\_clear\_items(GTK\_LIST(uiList),0,-1);

ReadFile();

Node \*p = head->next;

while(p)

{

AddNodeToList(p->name,p->num);

p = p->next;

}

if(widget!=0 || data!=0)

ShowTip("READ SUCCESSED");

}

void onBtnExit(GtkWidget \*widget, gpointer data)//点击EXIT按钮

{

gtk\_main\_quit();

onBtnSave(0,0);

}

void OnListSelectionChanged(GtkWidget \*gtklist, gpointer func\_data)//选择的uiList项目发生变化触发

{

GList \*dlist = GTK\_LIST(gtklist)->selection;

if (!dlist)

{

return;

}

else

{

if(dlist!=NULL)

{

const char \*buf = g\_object\_get\_data(G\_OBJECT(dlist->data),list\_item\_data\_key);

sprintf(selectText,"%s",buf);

}

}

}

void UI\_Init(int argc, char \*argv[])//初始化界面

{

gtk\_init(&argc,&argv);

GtkWidget \*window=gtk\_window\_new(GTK\_WINDOW\_TOPLEVEL);

gtk\_window\_set\_default\_size(GTK\_WINDOW(window),100,250);

g\_signal\_connect(G\_OBJECT(window),"delete\_event",G\_CALLBACK(gtk\_main\_quit),NULL);

gtk\_window\_set\_title(GTK\_WINDOW(window),"ToDoList");

gtk\_window\_set\_position(GTK\_WINDOW(window),GTK\_WIN\_POS\_CENTER);

gtk\_container\_set\_border\_width(GTK\_CONTAINER(window),10);

//--------------------垂直线性布局----------------------

GtkWidget \*vbox=gtk\_vbox\_new(0,0);

gtk\_container\_add(GTK\_CONTAINER(window),vbox);

//------------------------------------------------------

GtkWidget \*scrolled\_window = gtk\_scrolled\_window\_new (NULL, NULL);//可拖动窗口

gtk\_widget\_set\_size\_request (scrolled\_window, 250, 150);

gtk\_container\_add (GTK\_CONTAINER (vbox), scrolled\_window);

uiList = gtk\_list\_new();//放于可拖动窗口中的uiList

gtk\_scrolled\_window\_add\_with\_viewport (GTK\_SCROLLED\_WINDOW (scrolled\_window),uiList);

g\_signal\_connect (uiList, "selection-changed", G\_CALLBACK (OnListSelectionChanged), NULL);

//------------------水平布局---------------------------

GtkWidget \*hbox1 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox1,FALSE,FALSE,5);

GtkWidget \*lab\_name = gtk\_label\_new("ToDo");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),lab\_name,1,0,5);

textToDo = gtk\_entry\_new();

gtk\_box\_pack\_start(GTK\_BOX(hbox1),textToDo,1,0,5);

GtkWidget \*lab\_num = gtk\_label\_new("DeadLine");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),lab\_num,1,0,5);

textDeadLine = gtk\_entry\_new();

gtk\_box\_pack\_start(GTK\_BOX(hbox1),textDeadLine,1,0,5);

GtkWidget \*btn\_add = gtk\_button\_new\_with\_label("ADD");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),btn\_add,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_add),"clicked",G\_CALLBACK(onBtnInsert),NULL);

//-------放入Finished、SAVE、EXIT3个按钮--------

GtkWidget \*hbox2 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox2,FALSE,FALSE,5);

GtkWidget \*btn\_delete = gtk\_button\_new\_with\_label("Finished");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_delete,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_delete),"clicked",G\_CALLBACK(onBtnDelete),NULL);

GtkWidget \*btn\_save = gtk\_button\_new\_with\_label("SAVE");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_save,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_save),"clicked",G\_CALLBACK(onBtnSave),NULL);

GtkWidget \*btn\_exit = gtk\_button\_new\_with\_label("EXIT");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_exit,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_exit),"clicked",G\_CALLBACK(onBtnExit),NULL);

//------------------------------------------------------

onBtnRead(0,0);//从文件读取数据放入链表

gtk\_widget\_show\_all(window);//显示所有窗体

gtk\_main();

}

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

{

InitList(); //初始化链表

UI\_Init(argc, argv);//初始化UI

DelList(); //释放链表

return 0;

}

void OnListSelectionChanged(GtkWidget \*gtklist, gpointer func\_data)//选择的uiList项目发生变化触发

{

GList \*dlist = GTK\_LIST(gtklist)->selection;

if (!dlist)

{

return;

}

else

{

if(dlist!=NULL)

{

const char \*buf = g\_object\_get\_data(G\_OBJECT(dlist->data),list\_item\_data\_key);

sprintf(selectText,"%s",buf);

}

}

}

void UI\_Init(int argc, char \*argv[])//初始化界面

{

gtk\_init(&argc,&argv);

GtkWidget \*window=gtk\_window\_new(GTK\_WINDOW\_TOPLEVEL);

gtk\_window\_set\_default\_size(GTK\_WINDOW(window),100,250);

g\_signal\_connect(G\_OBJECT(window),"delete\_event",G\_CALLBACK(gtk\_main\_quit),NULL);

gtk\_window\_set\_title(GTK\_WINDOW(window),"ToDoList");

gtk\_window\_set\_position(GTK\_WINDOW(window),GTK\_WIN\_POS\_CENTER);

gtk\_container\_set\_border\_width(GTK\_CONTAINER(window),10);

//--------------------垂直线性布局----------------------

GtkWidget \*vbox=gtk\_vbox\_new(0,0);

gtk\_container\_add(GTK\_CONTAINER(window),vbox);

//------------------------------------------------------

GtkWidget \*scrolled\_window = gtk\_scrolled\_window\_new (NULL, NULL);//可拖动窗口

gtk\_widget\_set\_size\_request (scrolled\_window, 250, 150);

gtk\_container\_add (GTK\_CONTAINER (vbox), scrolled\_window);

uiList = gtk\_list\_new();//放于可拖动窗口中的uiList

gtk\_scrolled\_window\_add\_with\_viewport (GTK\_SCROLLED\_WINDOW (scrolled\_window),uiList);

g\_signal\_connect (uiList, "selection-changed", G\_CALLBACK (OnListSelectionChanged), NULL);

//------------------水平布局---------------------------

GtkWidget \*hbox1 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox1,FALSE,FALSE,5);

GtkWidget \*lab\_name = gtk\_label\_new("ToDo");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),lab\_name,1,0,5);

textToDo = gtk\_entry\_new();

gtk\_box\_pack\_start(GTK\_BOX(hbox1),textToDo,1,0,5);

GtkWidget \*lab\_num = gtk\_label\_new("DeadLine");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),lab\_num,1,0,5);

textDeadLine = gtk\_entry\_new();

gtk\_box\_pack\_start(GTK\_BOX(hbox1),textDeadLine,1,0,5);

GtkWidget \*btn\_add = gtk\_button\_new\_with\_label("ADD");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),btn\_add,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_add),"clicked",G\_CALLBACK(onBtnInsert),NULL);

//-------放入Finished、SAVE、EXIT3个按钮--------

GtkWidget \*hbox2 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox2,FALSE,FALSE,5);

GtkWidget \*btn\_delete = gtk\_button\_new\_with\_label("Finished");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_delete,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_delete),"clicked",G\_CALLBACK(onBtnDelete),NULL);

GtkWidget \*btn\_save = gtk\_button\_new\_with\_label("SAVE");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_save,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_save),"clicked",G\_CALLBACK(onBtnSave),NULL);

GtkWidget \*btn\_exit = gtk\_button\_new\_with\_label("EXIT");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_exit,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_exit),"clicked",G\_CALLBACK(onBtnExit),NULL);

//------------------------------------------------------

onBtnRead(0,0);//从文件读取数据放入链表

gtk\_widget\_show\_all(window);//显示所有窗体

gtk\_main();

}

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

{

InitList(); //初始化链表

UI\_Init(argc, argv);//初始化UI

DelList(); //释放链表

return 0;

}

GtkWidget \*lab\_num = gtk\_label\_new("DeadLine");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),lab\_num,1,0,5);

textDeadLine = gtk\_entry\_new();

gtk\_box\_pack\_start(GTK\_BOX(hbox1),textDeadLine,1,0,5);

GtkWidget \*btn\_add = gtk\_button\_new\_with\_label("ADD");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),btn\_add,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_add),"clicked",G\_CALLBACK(onBtnInsert),NULL);

//-------放入Finished、SAVE、EXIT3个按钮--------

GtkWidget \*hbox2 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox2,FALSE,FALSE,5);

GtkWidget \*btn\_delete = gtk\_button\_new\_with\_label("Finished");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_delete,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_delete),"clicked",G\_CALLBACK(onBtnDelete),NULL);

GtkWidget \*btn\_save = gtk\_button\_new\_with\_label("SAVE");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_save,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_save),"clicked",G\_CALLBACK(onBtnSave),NULL);

GtkWidget \*btn\_exit = gtk\_button\_new\_with\_label("EXIT");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),btn\_exit,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_exit),"clicked",G\_CALLBACK(onBtnExit),NULL);

//------------------------------------------------------

onBtnRead(0,0);//从文件读取数据放入链表

gtk\_widget\_show\_all(window);//显示所有窗体

gtk\_main();

}

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

{

InitList(); //初始化链表

UI\_Init(argc, argv);//初始化UI

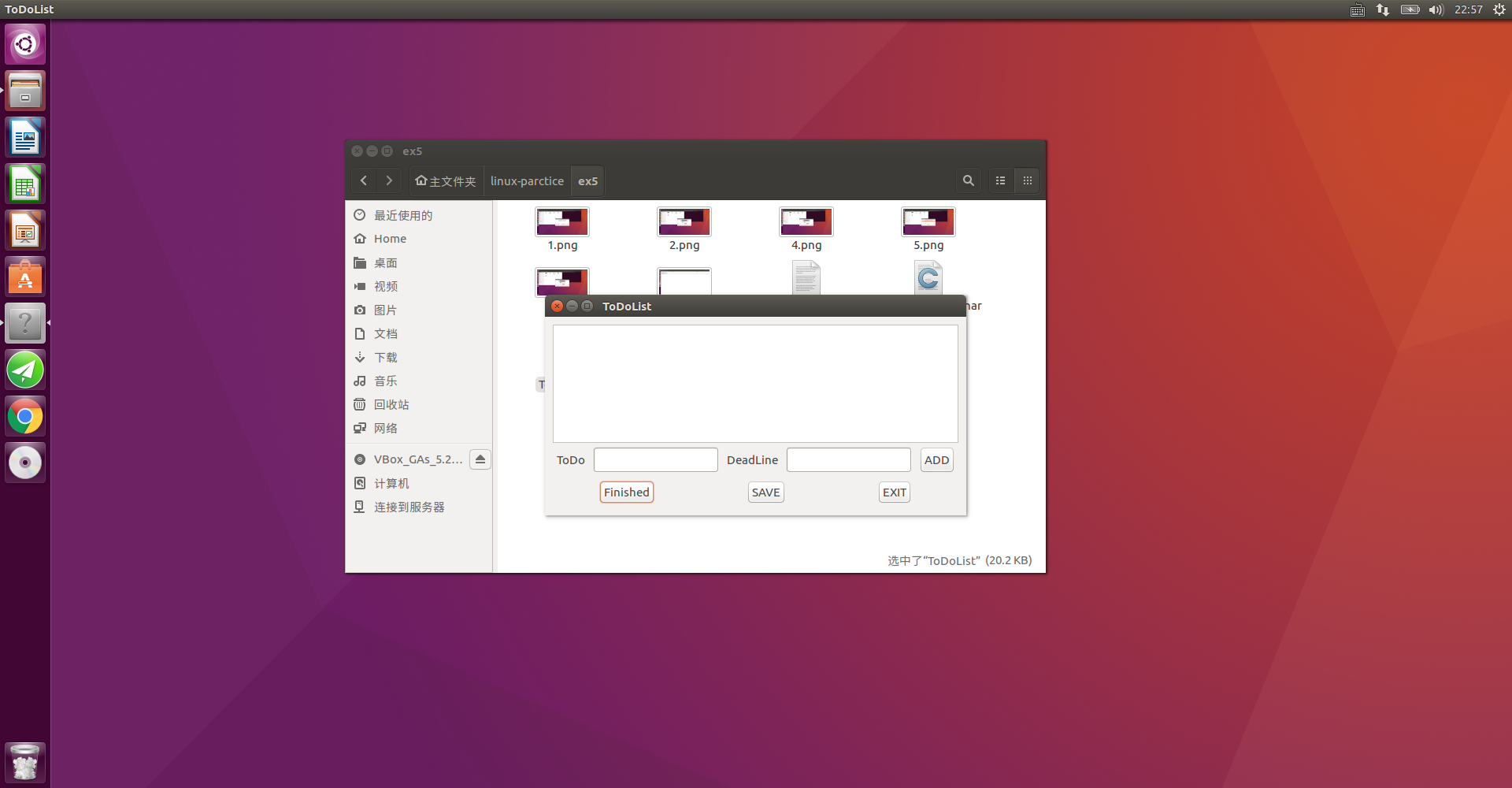
DelList(); //释放链表

return 0;

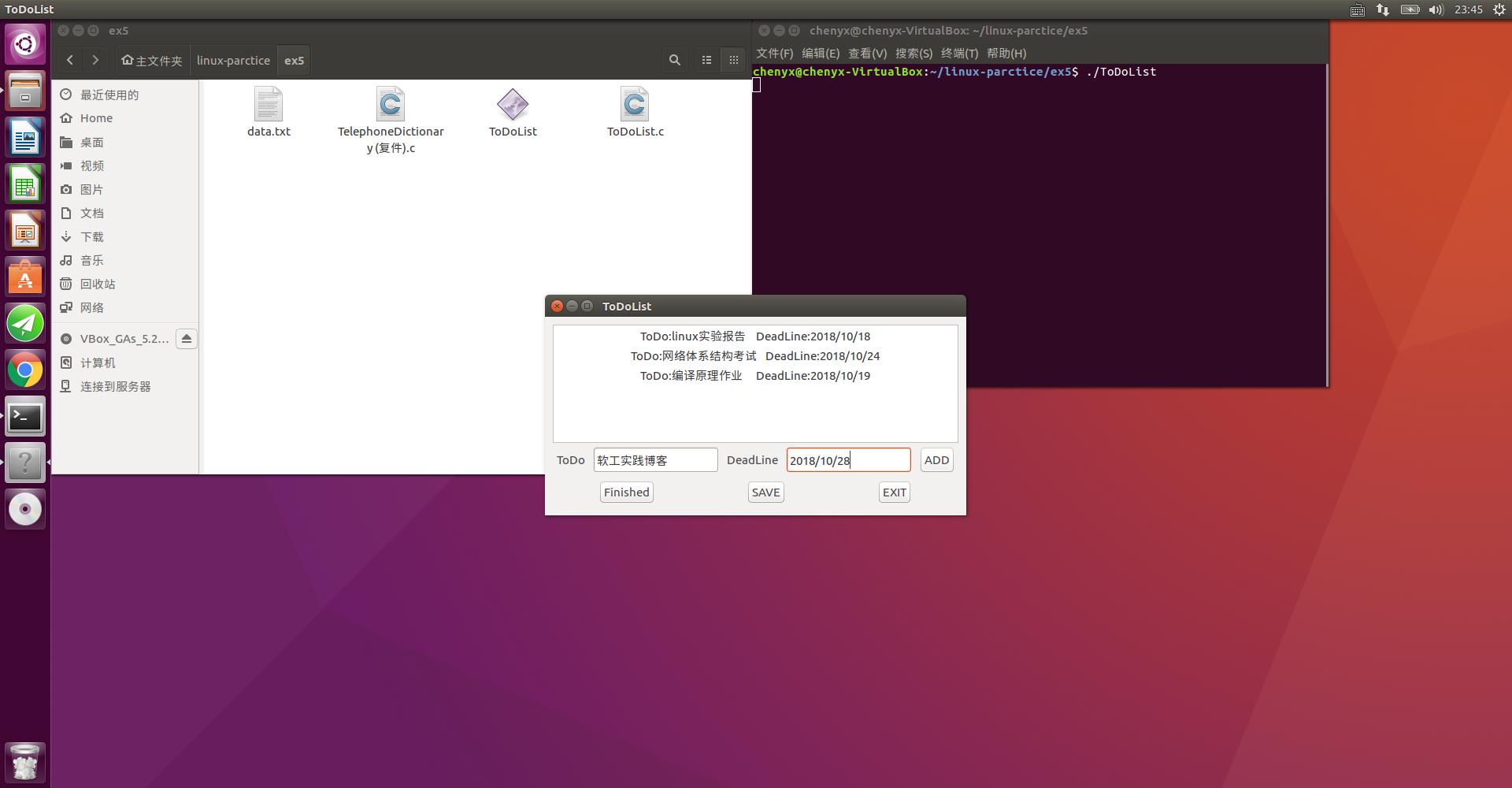
}

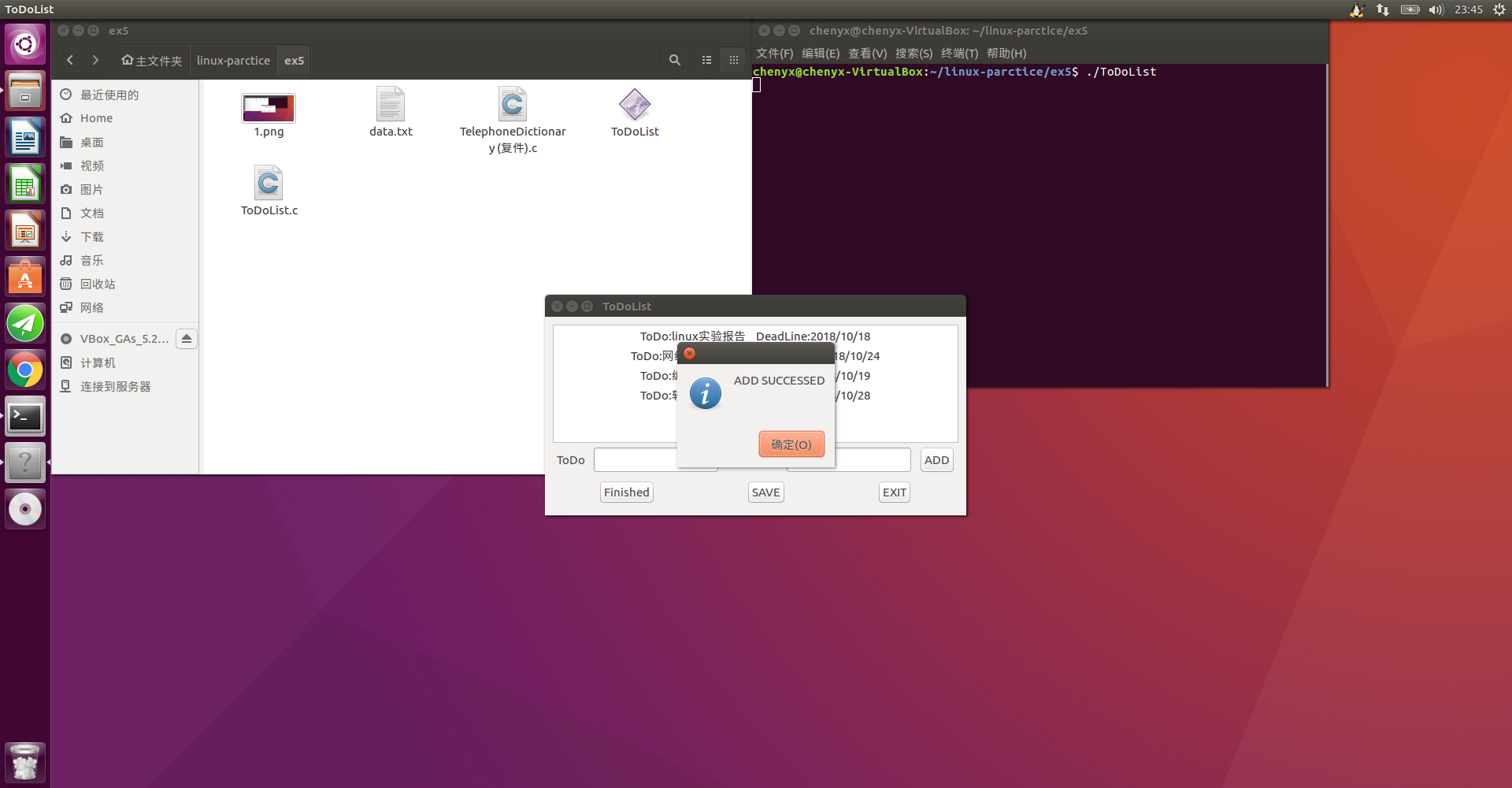
**（二）实验结果**

1.原始界面，此时还没有任何的待办事项

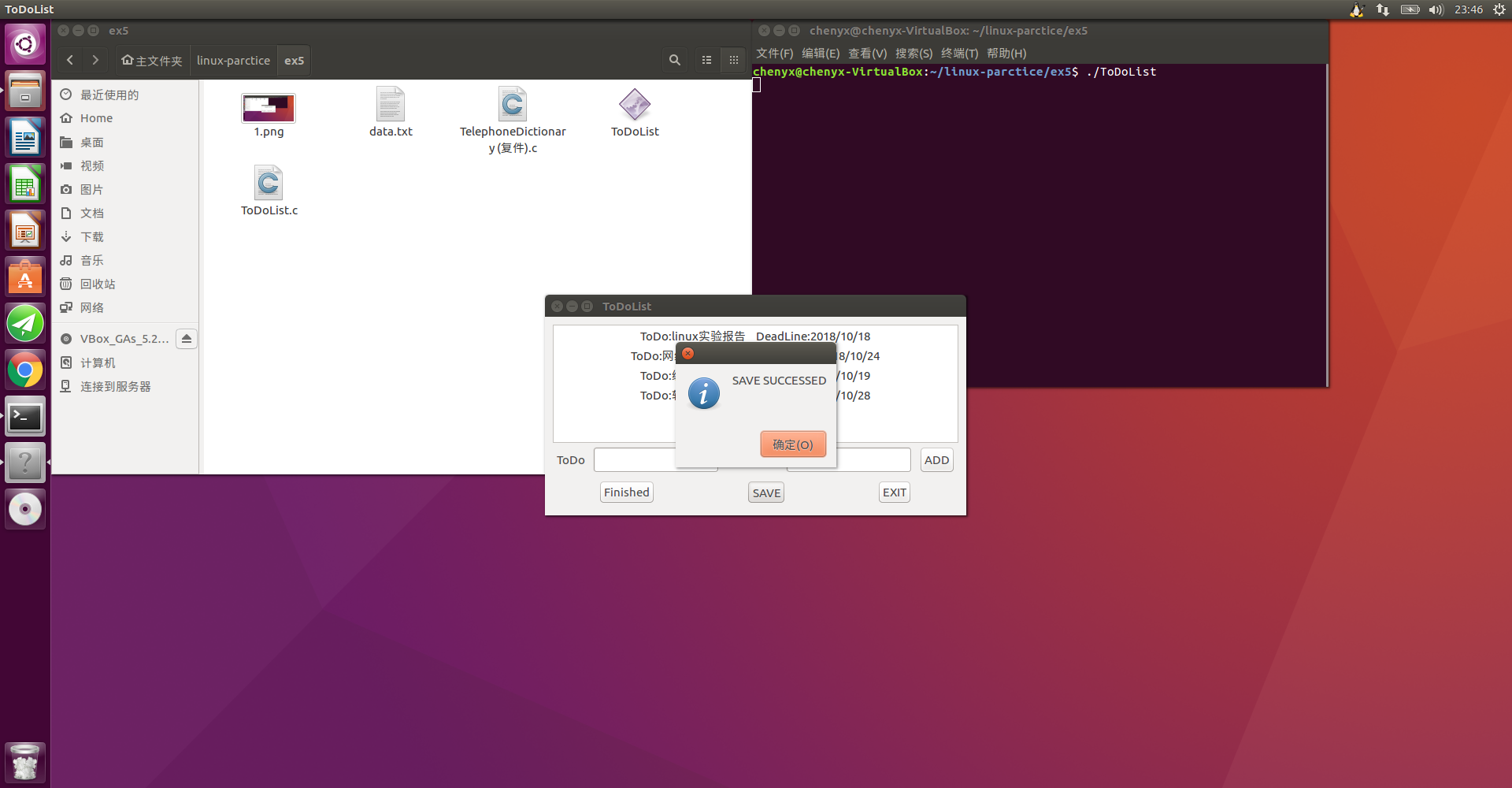


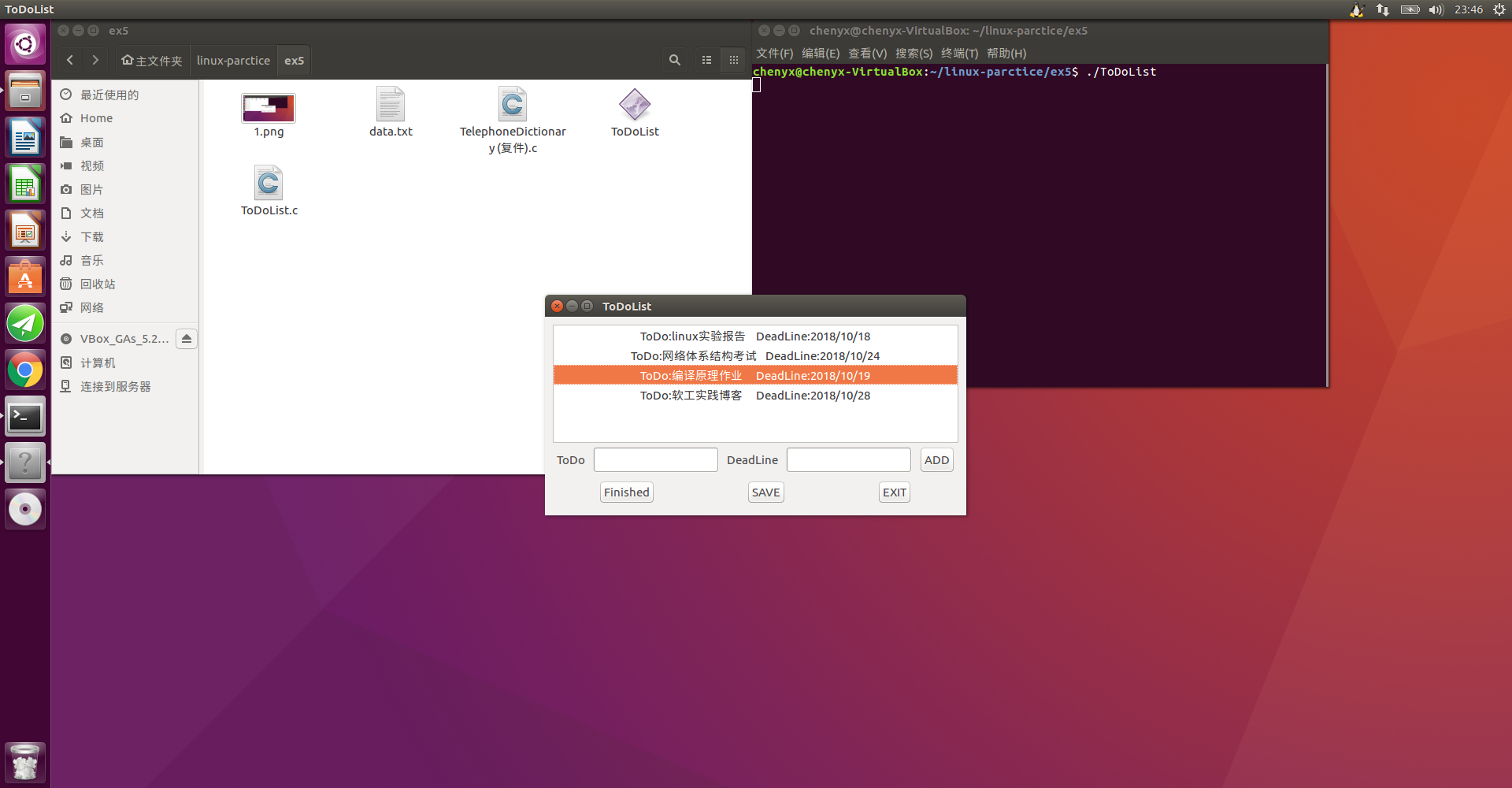
2.加入待办事项，添加成功会提示信息

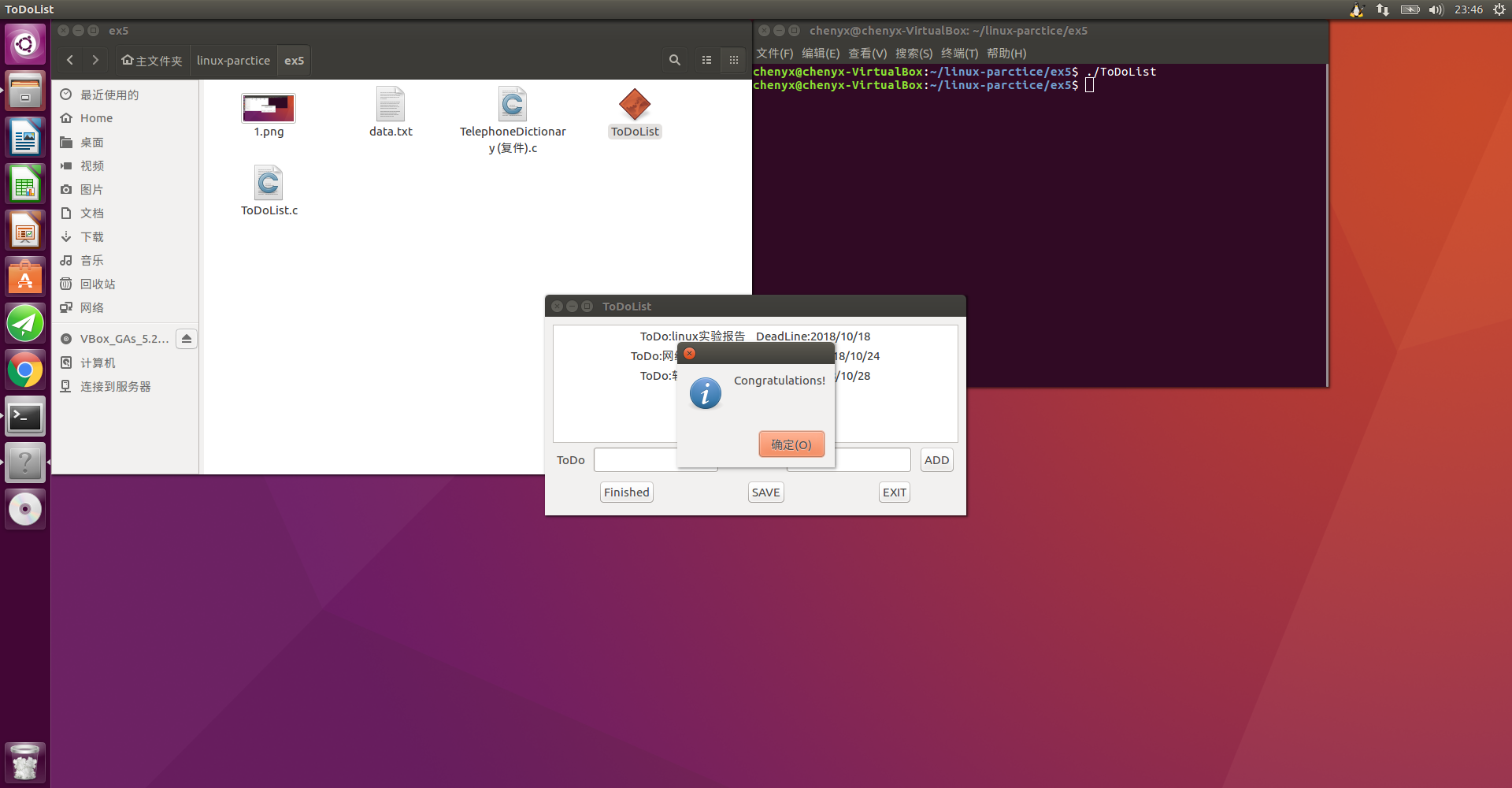




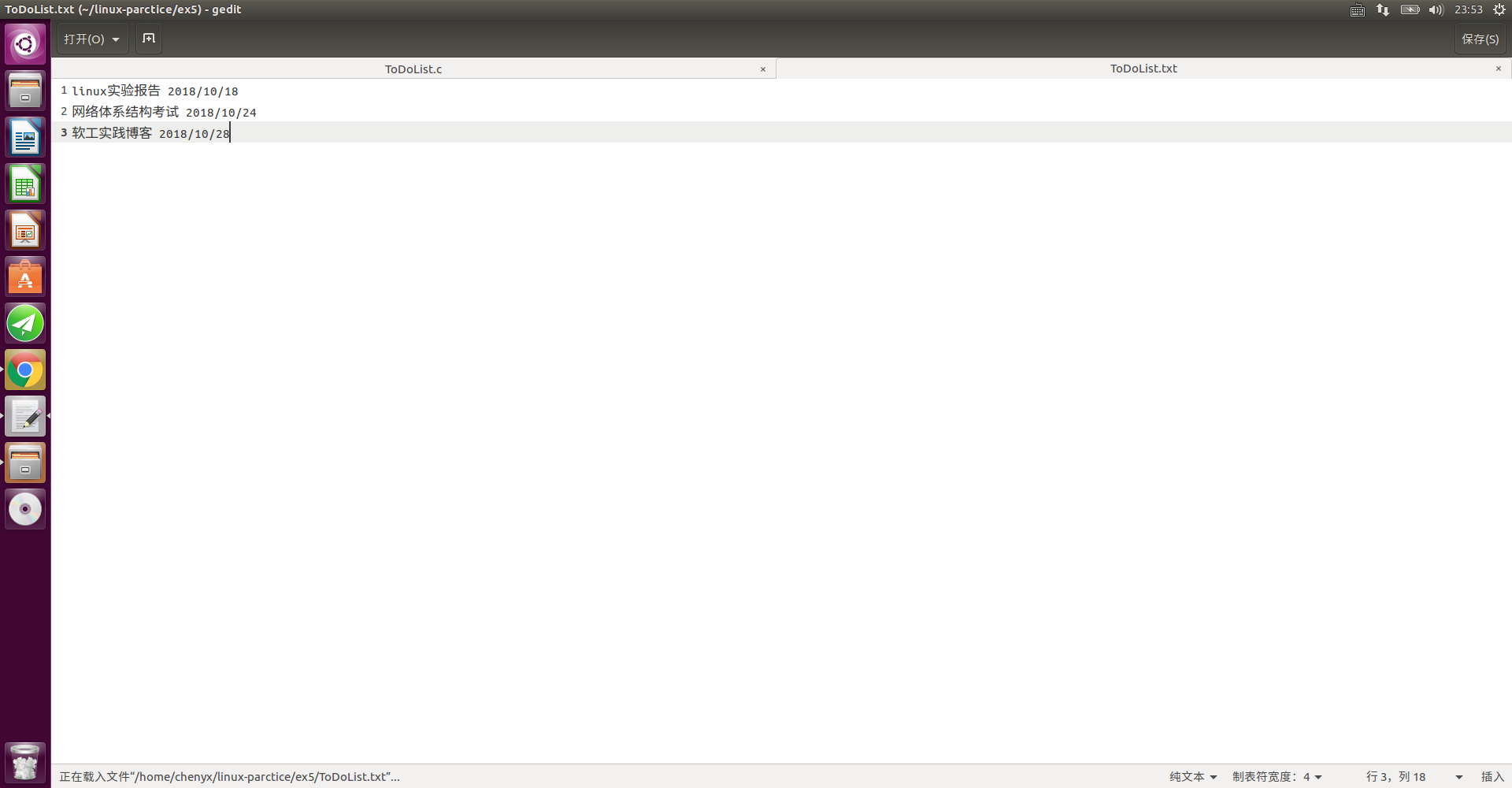
3.加入待办事项之后保存至ToDoList.txt文件，保存成功会有提示信息

 4.可以选中已完成的待办事项可以从中ToDoList中删除，删除成功会有提示信息





5.保存的ToDoList可以在ToDoList.txt文件中查看



6.其他无法截图展示的功能有：点击 EXIT 按钮退出后会自动将ToDoList的内容保存至ToDoList.txt文件；每次打开程序，会自动从ToDoList.txt中读取内容并展示。

**三、 实验总结**

**（一）编程思想**

程序实现的功能主要有以下几个：添加待办事项、删除已完成的待办事项、将待办事项保存至txt文件、打开程序读取txt文件中待办事项。

考虑到删除已完成的待办事项的时候，未必是按序删除，因此采用链表的数据结构来存储数据。维持一个链表，表中各个结点保存各个待办事项。

添加待办事项——向链表内添加一个结点，点击 ADD 按钮时触发

删除已完成的待办事项——从链表中将对应的结点删除，点击 Finished 按钮时触发

将待办事项保存至txt文件——遍历链表将各节点的数据存入文件，点击 SAVE 或者 EXIT 按钮时触发

打开程序读取txt文件中待办事项——读取文件将数据存入链表，打开程序时自动触发

**（二）相关知识**

本次实验用到大量GTK函数，这些列出一些重要的。

**1、gtk\_window\_new（GTK\_WINDOW\_TOPLEVEL）**

函数创建一个窗口并返回这个窗口的控件指针。

参数GTK\_WINDOW\_TOPLEVEL 指明窗口的类型为最上层的主窗口，它最常用。还可以取另一个值 GTK\_WINDOW\_POPUP 指明窗口的类型为弹出式的无边框的窗口。

**2、g\_signal\_connect（）**

使用这个宏为窗口或控件加回调函数。

g\_signal\_connect 宏有 4 个参数，分别是：

连接的对象，就是要连接信号的控件的指针(注意：必须是已创建完的控件的指针)，需要用 G\_OBJECT 宏来转换；

信号名称，就是要连接的信号名称，为字符串形式，用双引号引起来。不同的控件拥有的信号名称是不一样的；

回调函数，指信号发生时调用的函数，这里只用到函数名称，需要用 G\_CALLBACK宏来转换一下；

传递给回调函数的参数，它的值类型应该为 gpointer。如果不是这一类型可以强制转换，如果没有参数则为 NULL。这里只能传递一个参数，如果有多个参数，可以先将多个参数定义为一个结构，再将此结构作为参数传递过来

**3、改变窗口外观的几个函数**

设定窗口的标题：  
gtk\_window\_set\_title(window,const gchar\* title);

设定窗口的默认宽高：  
gtk\_window\_set\_default\_size(window,int width,int height);

设定窗口的位置：  
gtk\_window\_set\_position(window,GtkWindowPosition position);

其中 position 可以取如下值：

GTK\_WIN\_POS\_NONE 不固定

GTK\_WIN\_POS\_CENTER 居中

GTK\_WIN\_POS\_MOUSE 出现在鼠标位置

GTK\_WIN\_POS\_CENTER\_ALWAYS 窗口改变尺寸仍居中

GTK\_WIN\_POS\_CENTER\_ON\_PARENT 居于父窗口的中部

**4、gtk\_container\_add ()**

功能是将另一控件加入到容器中来。

它的第一参数是 GtkContainer 型的指针，这就需要将窗口控件指针用宏GTK\_CONTAINER 转换一下，即 GTK\_CONTAINER(window)。它的第二参数是要容纳的控件的指针，即 button。

**5、向盒状容器添加并排列控件**

用 gtk\_box\_pack\_\*系列函数向盒状容器添加并排列控件，这样的函数一共有2个，分别是： gtk\_box\_pack\_start、 gtk\_box\_pack\_end。

gtk\_box\_pack\_start、 gtk\_box\_pack\_end 分别表示按顺序从前到后依次排列控件和从后到前依次排列控件。这两个函数都有 5 个参数，依次是 GTK\_BOX(box)，要容纳控件的容器对象； button，被容纳控件的指针；是否扩展，是否添充和与前一控件的间隔。例如本次实验中用到的：gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox1,FALSE,FALSE,5);

**6、gtk\_widget\_show\_all（）**

原本每一个控件都要用函数 gtk\_widget\_show 来显示，而这个函数显示容器中所有控件。

它的参数是一个容器控件的指针，例如本次实验中用到的：

gtk\_widget\_show\_all(window);

**（三）收获及展望**

一开始看到实验题目觉得很困难，后来仔细阅读了老师提供的资料之后，做了一个简单的图形化界面。学会了如何布局，如何添加控件并为其绑定相应的动作等等。当然因为时间有限，程序还有很多不足，例如没办法设置优先级等。如果接下来有机会希望能够完善这个程序。