**ACKNOWLEDGEMENT**

It is said that the gratitude is the best attitude. It gives us immerse pleasure in bringing out this project report entitled “Text Editor” taken up during the 5th semester of Bachelor of Engineering course.

We are thankful to our respected principal **Dr.Mallikarjuna K Babu** and our respected H.O.D **Dr. H S Guruprasad** for his constant support and guidance.

We are also thankful to **Mr. Vikrant B M,** who taught us the concepts underlying the project. Also we would like to thank our lab instructors **Mr. Syed Akram** and **Mr. Lohith J.J.** for their constant support, guidance and motivation throughout the project.

Finally, I am very much grateful to all my friends, who has given us generously all help, warm encouragement, moral support and kind co-operation at all point of time.

**ABSTRACT**

This project is a simple text editor developed on Linux platform and provides various options required for creating, editing, displaying and formatting text files. It has been named “**TEXT EDIT**”. It supports keyboard as well as mouse interactions to make it easier for the user to interact with and use it. The Graphical User Interface (GUI) provided is also easy to use and understand.

A toolbar is also provided to perform several task as shortcuts, this will save users time as well as allow him to use the editor features more efficiently. All the general features of a text editor have been included, such as copy/paste, delete/insert, saving/loading of a file, etc. is supported by the text editor.

TextEdit also support having multiple tabs for programming purposes and programmers have to work several files at once and allow users to type in several fonts to support proper formatting.

**CONTENTS**

1. **Problem Understanding 5  
   1.1** Problem Statement 5  
   **1.2** Product Perspective 5   
   **1.3** Hardware Requirements 6   
   **1.4** Software Requirements 7
2. **Introduction to LINUX 8  
   2.1** Operating System 8   
   **2.2** Advantages of LINUX 9   
   **2.3** LINUX Features 9
3. **Editors 11  
   3.1** Definition 11   
   **3.2** Types of Editors 11   
   **3.3** Overview of Editing Process 11   
   **3.4** Editor Structure 13
4. **System Design and Modeling 14  
   4.1** System Design 14   
   **4.2** Data Flow Diagram 15  
   **4.3** Data Structure used 16
5. **Introduction to GTK+ 17  
   5.1** About 17   
   **5.2** Features of GTK 17
6. **Implementation Details 20  
   6.1** Introduction 20   
   **6.2** Implementation 20
7. **Conclusion and Future Enhancements 22  
   7.1** Conclusion 22   
   **7.2** Limitations 22   
   **7.3** Future Enhancement 23

**APPENDIX A: Source code for the Text Editor 24**

**APPENDIX B: Snapshots 42**

**BIBLIOGRAPHY 45**

**CHAPTER-1: PROBLEM UNDERSTANDING**

**1.1 Problem Statement**

To develop a graphical text editor using in-built GTK library.

**Motivation:**

As a test editor is a very common tool, it is a good start for a programmer to test his/her programming skills by making his own text editor.

**Design Decision:**

C language is selected for designing his project. Several inbuilt functions and templates present in C language are used for making the coding easier. The idea of using GTK library for GUI purpose is that it supports: overall screen, window and pad manipulation. Output to windows and pads, reading terminal input, control terminals and input and output, color manipulation, use of soft label keys, term info capabilities and access to low level terminal manipulation routines.

**Platform:**

This editor is for LINUX operating system. Linux platform was chosen because for different GUI options which are present in GTK library are present in the LINUX operating system.

**Language:**

Language used is C Language.

**1.2 Product Perspective:**

The product is a full screen text editor for the LINUX operating system with the capabilities to edit existing files and also create new ones. To make editing simple is provided a pop own menu interface, as also are shortcut keys for command.

**Product Functions:**

* **File operations**

Create a new file

Open an existing file

Open in a new tab

Save a file

Delete a file

Exit from the editor

* **Editing operations**

Cut

Copy

Paste

* **Font Options**

Select font of the text

* **Help options**

About

**1.3 HARDWARE REQUIREMENTS**

The hardware requirements are minimum that is required to install the Linux OS

* MAIN PROCESSOR : Pentium IV (500Mhz)
* RAM SIZE : 128 MB
* CACHE MEMORY : 256KB
* DISKETTE DRIVE : 1.44MB,3.5inches
* OPTICAL DRIVE : 4X DVD-ROM DRIVE

**1.4 SOFTWARE REQUIREMENTS**

* OPERATING SYSTEM: Red Hat Linux 9.0 or Fedora core
* COMPILER USED: GCC version 3.2.2
* EDITOR: VI Editor version 6.1
* GTK library present in the Linux Operating System

**CHAPTER -2: INTRODUCTION TO LINUX**

**2.1 OPERATING SYSTEM**

An operating system (OS) is a resource manager. It takes the form of a set of software routines that allow users and application programs to access system resources (e.g. the CPU, memory, disks, modems, printers, network cards etc.) in a safe, efficient and abstract way. An OS encourages efficient use of the CPU by suspending programs that are waiting for I/O operations to complete to make way for programs that can use the CPU more productively. An OS also provides convenient abstractions (such as files rather than disk locations) which isolate application programmers and users from the details of the underlying hardware.

**Linux operating system**

Linux is a UNIX-based operating system originally developed for an Intel-compatible PC's. It is now available for most types of hardware platforms, ranging from PDAs to mainframes. Linux000 is a "modern operating system", meaning it has such features as virtual memory, memory protection, and preemptive multitasking.

Linux is built and supported by a large international community of developers and users dedicated to free, open-source software. This community sees Linux as an alternative to such proprietary systems as Windows and Solaris and as a platform for alternatives to such proprietary applications as MS Office, Internet Explorer, and Outlook.

Linux is an open-source operating system .The Linux operating system is touted for its speed, minimal hardware requirements, security and remote administration. Linux is a fully featured operating system which doesn't have to cost a dime. You download many different distributions directly from the Internet and the support community is huge. Because of Linux's speed and stability it has become the operating system of choice for most types of servers.

There are graphical environments (GUIs), office applications, developers' tools, system utilities, business applications, document publishing tools, network client and server applications.

Linux specifically refers to the Linux kernel. However, the kernel is useless without a set of tools and applications to run on the kernel. Linux is most commonly distributed with this toolset and a collection of applications in what is called a "distribution". The most common are Red hat, Mandrake, Suse, and Debian. Distributions differ in three basic ways: the process for installing the distribution, the applications available, and process for installing and managing these applications.

**2.2 ADVANTAGES OF LINUX**

* Configurability
* Convenience
* Stability
* Community
* Freedom

**2.3 LINUX FEATURES**

It is a sophisticated high performance Unix like operating system for multiple architectures

**Key Linux Features:**

* Per user file and application security.
* Can have applications from multiple machines on your workstation at the same time.
* Shared libraries for disk and memory efficiency.
* Unified memory/disk cache for high performance.
* Transparent software FPU emulation for integer only machines.
* POSIX.1 style API with both USL and BSD extensions. Porting almost all correctly written POSIX or Unix API based programs is trivial.
* TCP/IP networking including both protocol layers and the standard BSD tool set.
* Wide set of internet World Wide Web tools.
* NFS server and client - the standard Unix network file system.
* Internet SMTP mail server and clients - including MIME support.
* SLIP and PPP for internet networking over analog or digital modems.
* Wide range of high performance network drivers.
* Wide range of supported hardware.
* Multimedia soundcard and CD-ROM drivers.

**CHAPTER-3: EDITORS**

**3.1 Definition**

An editor is general-purpose system software, which can be used to enter and/or modify text, programs, graphics or data.

An editor may be a line editor, screen editor, graphics editor or word Processors. We will be discussing all these as we move on.

**3.2 Types of Editors**

Editors are broadly classified as

1. Line Editors

2. Screen Editors

3. Graphic Editors

**3.3 Overview of Editing Process**

An interactive editor is a software program that allows a user to create and revise a target document. The term document includes objects such as computer programs, text equations, tables, diagrams, line art and photographs – anything we find on a hard copy.

The text editor should be considered the primary interface to the computers for all types of “knowledge workers” as they compose, organize, study & manipulate computer based information.

The primary elements being edited in a text editor are character strings of the target text.

The interactive editing process is accomplished with the following tasks:

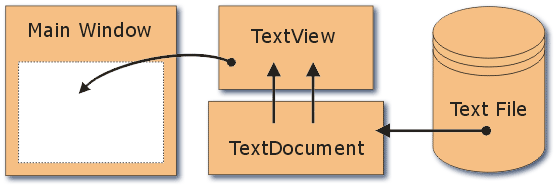
* The part of the target document that needs to be manipulated is viewed.
* The document is first displayed in the required format.
* Using the necessary designed tools, the document is modified.
* The view is updated appropriately and saved

In this above process, the selection of the part of the document to be viewed and edited involves first traveling through the document to locate the area of interest. This search is accomplished with operations such as – find pattern, move cursor & find & replace. Filtering controls the process of what is to be viewed & manipulated. Filtering extracts the relevant subset of the target document at the point of interest, such as the next screen of the text or the next statement.

In the actual editing phase, the target document is created or altered with a set of operations such as insert, delete, replace, move, copy. The editing functions are often specialized to operate on elements meaningful to the type of editor. For example, a manuscript-oriented editor might operate on elements such as single characters, words, lines, sentences, and paragraphs; a program-oriented editor might operate on elements such as identifiers, keywords, and statements.

In a simple scenario, the, the user might travel to the end of the document. A screenful of text would be filtered, this segment would be formatted, and the view would be displayed on an output device. The user could then, for example, delete the first three words of this view.

**3.4 EDITOR STRUCTURE**



Most text editors have a structure similar to that shown in the figure, regardless of the particular features they offer and the computers on which they are implemented. The command language processor accepts the input from the user’s input devices, and analyses the tokens and syntactic structure of the commands. In this sense, the command language processor functions much like the lexical and syntactic phases of a compiler. Just as in compiler, the command language processor may invoke semantic routines directly. In a text editor, these semantic routines perform functions such as editing and viewing.

**CHAPTER-4: SYSTEM DESIGN AND MODELING**

**4.1 SYSTEM DESIGN**

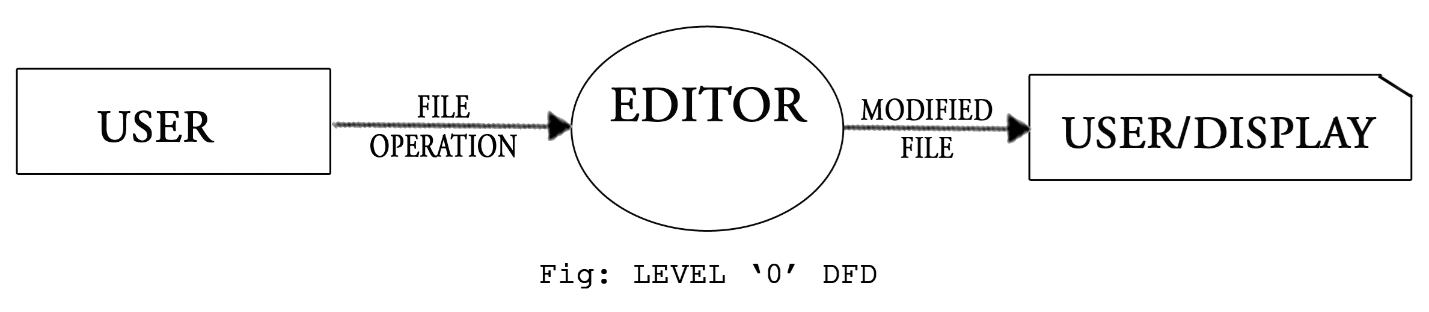
System design refers to formulating a precise description of the desired system in Software Development terms. The information contained in the system specification is similar to that contained in the required documents. The focus is on what functions the Editor must perform, rather than how the Editor performs them. The system specification can be considered as a computer oriented description of the Requirements Documents. This is represented by a **DATA FLOW DIAGRAM (DFD).**

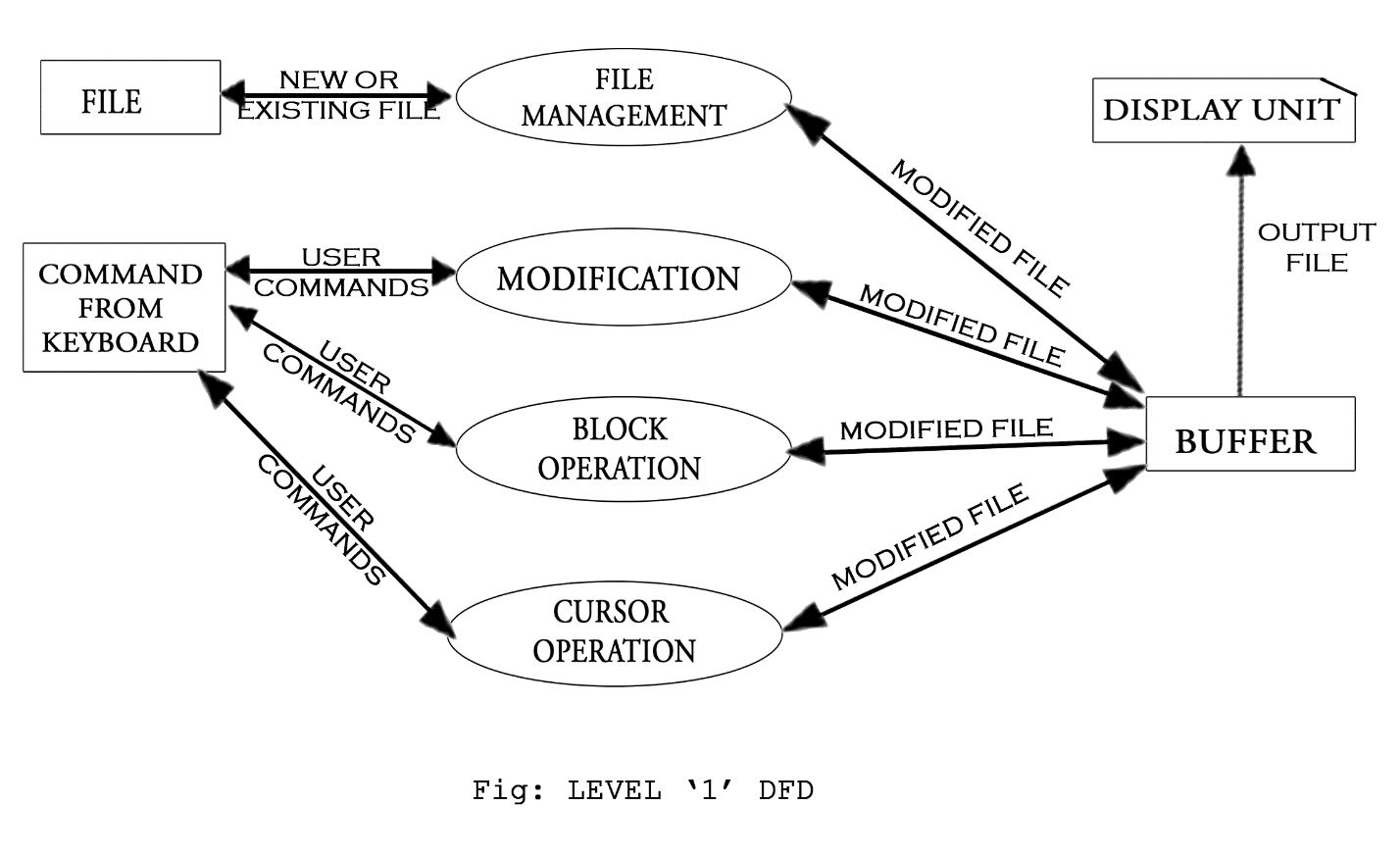
The documentation for the DFD is as follows:

* **FILE**: It is the Floppy Disk/Hard Disk drive where the required source information is stored. This data is to be retrieved into the Primary memory before any changes are done. It is stored in a named file
* **USER COMMAND**: This device provides information to modify/append the data. The command is entered through the keyboard/mouse.
* **BUFFER**: It is the primary memory where the actual modification of information is possible. Data is brought here from the file through commands.
* **DISPLAY UNIT**: Data can be seen here. It is also called Visual Display Unit. It is possible to see the changes made in this buffer and unit.
* **EDITOR**: This is the main part of the complete process. It has the actual program to retrieve data from the file and place it in the buffer.

The system specification can be described by following data flow diagrams (DFD).

**4.2 Data Flow**

****

****

**4.3 Data Structure Used**

A Doubly Linked List (DLL) with a pointer to an array to store the characters has been chosen as the data structure to represent a single line in the file.

This data structure has been selected keeping in mind that whenever a line is inserted or deleted, all the nodes above and below the line need not be changed to point to their next/previous lines. This advantage scores over the disadvantage that random access of lines is not possible.

Every line is a stored as an array whose address is stored in the pointer field of the node. Here, the fact that random access is not possible is compensated by the advantage of highly efficient insertion and deletion options (they just involve changing of pointers). This is because insertion and deletion, being the most fundamental operation, need more focus than the lack of random access to a particular character.

As a whole, the string nodes in the DLL contain two links: one pointing to the previous node and one pointing to the next node and one pointer to the array which stores the characters entered.

**CHAPTER-5: INTRODUCTION TO GTK+**

**5.1 About**

GTK+, or the GIMP Toolkit, is a multi-platform toolkit for creating graphical user interfaces. Offering a complete set of widgets, GTK+ is suitable for projects ranging from small one-off tools to complete application suites. GTK+ is cross-platform and boasts an easy to use API, speeding up your development time.

GTK+ is written in C but has been designed from the ground up to support a wide range of languages, not only C/C++. Using GTK+ from languages such as Perl and Python (especially in combination with the Glade GUI builder) provides an effective method of rapid application development.

GTK+ is free software and part of the GNU Project. However, the licensing terms for GTK+, the GNU LGPL, allow it to be used by all developers, including those developing proprietary software, without any license fees or royalties.

**5.2 Features of GTK+**

## Stability

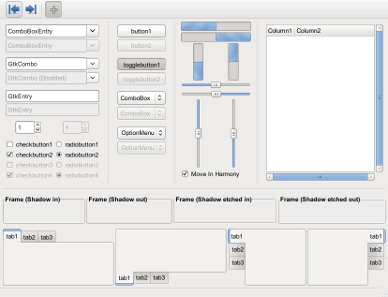
GTK+ has been developed for over a decade to be able to deliver the enticing features and superb performance that it brings to your application development. GTK+ is supported by a large community of developers and has core maintainers from companies such as [Red Hat](http://www.redhat.com/), [Novell](http://www.novell.com/), [Lanedo](http://www.lanedo.com/gtk+.html), [Codethink](http://www.codethink.co.uk/), [Openismus](http://www.openismus.com/) and [Intel](http://www.intel.com/).

## Language Bindings

GTK+ is available in many other programming languages thanks to the [language bindings](http://www.gtk.org/language-bindings.php) available. This makes GTK+ quite an attractive toolkit for application development.

## Interfaces

GTK+ has a comprehensive collection of core widgets and interfaces for use in your application.

[](http://www.gtk.org/images/features/twf.png)

* Windows (normal window or dialog, about and assistant dialogs)
* Displays (label, image, progress bar, status bar)
* Buttons and toggles (check buttons, radio buttons, toggle buttons and link buttons)
* Numerical (horizontal or vertical scales and spin buttons) and text data entry (with or without completion)
* Multi-line text editor
* Tree, list and icon grid viewer (with customizable renderers and model/view separation)
* Combo box (with or without an entry)
* Menus (with images, radio buttons and check items)
* Toolbars (with radio buttons, toggle buttons and menu buttons)
* GtkBuilder (creates your user interface from XML)
* Selectors (color selection, file chooser, font selection)
* Layouts (tabulated widget, table widget, expander widget, frames, separators and more)
* Status icon (notification area on Linux, tray icon on Windows)
* Printing widgets
* Recently used documents (menu, dialog and manager)

## Cross Platform

Originally GTK+ was developed for the X Window System but it has grown over the years to include backend support for other well known windowing systems. Today you can use GTK+ on:

* [GNU/Linux and Unix](http://www.gtk.org/download/linux.php)
* [Windows (32-bit)](http://www.gtk.org/download/win32.php) and [64-bit](http://www.gtk.org/download/win64.php)
* [Mac OS X](http://www.gtk.org/download/macos.php)

## Accommodating

GTK+ caters for a number features that today's developers are looking for in a toolkit including:

* Native look and feel
* Theme support
* Thread safety
* Object oriented approach
* Internationalization
* Localization
* Accessibility
* Bidirectional text support (LTR/RTL, Left To Right/Right To Left)
* UTF8 support
* Documentation

## Foundations

GTK+ is built on top of GLib. GLib provides the fundamental algorithmic language constructs commonly duplicated in applications. This library has features such as: (this list is not a comprehensive list)

* Object and type system
* Dynamic loading of modules (i.e. plug-ins)
* Thread support
* Timer support
* Memory allocator
* Threaded Queues (synchronous and asynchronous)
* Lists (singly linked, doubly linked, double ended)

**CHAPTER-6: IMPLEMENTATION DETAILS**

**6.1 Introduction**

This text editor supports many editing functions to the user. A user can edit a given text file. The different editing features are Open file, Save file, Delete file, cut copy and paste, delete line/word, insert line/word, Find and Replace.

The project is divided into three phases.

* Developing the graphical user interface.
* Writing the code for different functions.
* Code for user key functions.

**6.2 Implementation**

**Phase 1:**

**Developing the background window and menu bar.**

At first background window is generated using GTK library functions library command. Then the background color is set. Then on the top and bottom of this background window two sub windows of different colors are created. For each menu options different sub windows are created and appropriate strings of different names are added into those sub windows.

**Key Binding in each sub windows**

Here for each options in the sub windows , keys are binded and the keys include shortcut keys for each options in that window including the up-down motion of the curser in that sub window and again the left right motion of the sub windows in the menu bar.

**Phase 2:**

In this phase I have used several inbuilt functions in the list.h for buffers used in the editors. Doubly Linked List is the intermediate representation of the characters. I have connected both the front end and the back end using these functions.

**Phase 3:**

In this phase, I have binded the different user keys. The minimum keys needed are arrow keys. Then all the other shortcut keys are also binded.

**DETAIL:**

The constants TRUE and FALSE are set to 1 and 0 respectively.

Characters are inserted creating a new node which stores the character typed. The new node created will have links to its previous node and as the nodes are created, they link themselves to their predecessor. In effect, a doubly linked list is created.

The character index on the screen is maintained in the two variables. The variable lines keeps track of the current line on which the cursor is placed.

The cursor movements are defined in the functions. We traverse along the list using the links either in the left or the right direction at the same time decrementing or incrementing variable. We can also move the cursor up or down, at the same time decrementing or incrementing variable.

The menus in the editor are “File” “Edit” “Background” “Help”.

File contains options to create a new file, open an existing file, to save a file (also save as), and to quit the editor screen.

The Edit menu contains cut, copy and paste.

The Page Options contains options to indent the given file.

The Help menu contains information about the keys used in the editor, the shortcut keys and about the programmer.

The implementation details of all the above menus and their associated functions are in the source code.

**CHAPTER 7: CONCLUSION AND FUTURE ENHANCEMENTS**

**7.1 Conclusion**

Implementing a graphical text editor was very interesting experience and I gained a lot of knowledge about the editors as well as data structures through the project.

The graphical Text editor is mainly to create and revise a document.

The full screen text editor is designed to satisfy most of the user requirements such as creating a new file, opening an existing file, saving a file, deleting a file.

Editing a file is enabled by providing options for inserting or deleting character, word and line, Find and Replace options are also provided along with cut, copy and paste. I have tried to provide as many options as possible with user interface.

The Page Margin also can be adjusted.

Help options are provided for users, which also mentions the shortcut keys assigned for various frequently used functions. For all these functions to operate the GTK library present in the Linux platform is a must.

The project also has many defects in it. I have tried to implement user interface through mouse.

These defects can be corrected and a more comfortable document creating and revising environment for the user can be provided.

**7.2 Limitations**

This text editor has following imitations:

* Only text files can be edited.
* Few shortcut keys are provided.
* The source code is non portable to non-Linux platforms.
* Mouse support is only for single page files and for text selection only.
* Not much user interface options through mouse is provided.
* Source code is not helpful without GTK library routines.

**7.3 Future Enhancement**

* User interface can be improved.
* More shortcut keys can be assigned.
* More editing options must be provided.

**APPENDIX A:Source code for the Text Editor**

/\*Compile using

\*gcc –g –Wall textedit.c –o textedit `pkg-config –-cflags -–libs gtk+-2.0` \*/

#include <gtk/gtk.h>

#include <gdk/gdkkeysyms.h>

#include <string.h>

#include <libgen.h>

typedef struct {

GtkWidget \*menu\_label;

GtkWidget \*menu;

GtkWidget \*new;

GtkWidget \*open;

GtkWidget \*save;

GtkWidget \*close;

GtkWidget \*separator;

GtkWidget \*quit;

} FileMenu;

typedef struct {

GtkWidget \*menu\_label;

GtkWidget \*menu;

GtkWidget \*cut;

GtkWidget \*copy;

GtkWidget \*paste;

} EditMenu;

typedef struct {

GtkWidget \*menu\_label;

GtkWidget \*menu;

GtkWidget \*font;

} OptionsMenu;

typedef struct {

GtkWidget \*menu\_label;

GtkWidget \*menu;

GtkWidget \*about;

} HelpMenu;

typedef struct {

GtkWidget \*scrolled\_window;

GtkWidget \*textview;

GtkWidget \*tab\_label;

} FileObject;

typedef struct {

GtkWidget \*toplevel;

GtkWidget \*vbox;

GtkWidget \*notebook;

GtkWidget \*menubar;

GtkWidget \*toolbar;

FileMenu \*filemenu;

EditMenu \*editmenu;

OptionsMenu \*options\_menu;

HelpMenu \*helpmenu;

} TextEditUI;

typedef struct {

gchar \*filename;

gint tab\_number;

} FileData;

static int files\_open = 0;

static GList \*filename\_data = NULL; /\* Linked list of open file names \*/

static PangoFontDescription \*desc = NULL; /\* Global font for all tabs \*/

static void quit\_application(GtkWidget \*, gpointer);

static void text\_edit\_init\_GUI(TextEditUI \*);

static void text\_edit\_create\_menus(TextEditUI \*);

static void text\_edit\_create\_toolbar\_items(TextEditUI \*);

FileObject \*text\_edit\_file\_new(void);

static void text\_edit\_tab\_new\_with\_file(GtkMenuItem \*, GtkNotebook \*);

static void text\_edit\_select\_font(GtkMenuItem \*, gpointer);

static void text\_edit\_apply\_font\_selection(GtkNotebook \*);

static void text\_edit\_open\_file(GtkMenuItem \*, GtkNotebook \*);

static void text\_edit\_cut\_to\_clipboard(GtkMenuItem \*, GtkNotebook \*);

static void text\_edit\_copy\_to\_clipboard(GtkMenuItem \*, GtkNotebook \*);

static void text\_edit\_paste\_from\_clipboard(GtkMenuItem \*, GtkNotebook \*);

static void text\_edit\_show\_about\_dialog(GtkMenuItem \*, GtkWindow \*);

static void text\_edit\_close\_file(GtkMenuItem \*, GtkNotebook \*);

static void text\_edit\_save\_file(GtkMenuItem \*, GtkNotebook \*);

static void text\_edit\_register\_filename(gchar \*fname, gint tab\_num);

static gchar \*text\_edit\_get\_filename(gint tab\_num);

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

{

TextEditUI app;

gtk\_init(&argc, &argv);

app.toplevel = gtk\_window\_new(GTK\_WINDOW\_TOPLEVEL); /\* Main window \*/

gtk\_window\_set\_title(GTK\_WINDOW(app.toplevel), "TextEdit");

gtk\_window\_set\_default\_size(GTK\_WINDOW(app.toplevel), 650, 350);

/\*

\* Connect signal handler for destruction of

\* top-level window.

\*/

g\_signal\_connect(G\_OBJECT(app.toplevel), "destroy",

G\_CALLBACK(quit\_application), NULL);

text\_edit\_init\_GUI(&app); /\* Build interface \*/

gtk\_widget\_show\_all(app.toplevel);

gtk\_main();

return 0;

}

static void quit\_application(GtkWidget \*window, gpointer data)

{ gtk\_main\_quit(); }

static void text\_edit\_init\_GUI(TextEditUI \*app)

{

desc = pango\_font\_description\_from\_string("Progsole normal 12");

FileObject \*file = text\_edit\_file\_new();

text\_edit\_register\_filename("Untitled", 0); /\* Keep track of scratch buffer's filename \*/

app->vbox = gtk\_vbox\_new(FALSE, 0);

app->notebook = gtk\_notebook\_new();

app->menubar = gtk\_menu\_bar\_new();

text\_edit\_create\_menus(app);

text\_edit\_create\_toolbar\_items(app);

gtk\_notebook\_set\_show\_tabs(GTK\_NOTEBOOK(app->notebook), TRUE);

gtk\_notebook\_append\_page(GTK\_NOTEBOOK(app->notebook), file->scrolled\_window, file->tab\_label);

gtk\_box\_pack\_start(GTK\_BOX(app->vbox), app->notebook, TRUE, TRUE, 0);

gtk\_container\_add(GTK\_CONTAINER(app->toplevel), app->vbox);

}

static void text\_edit\_create\_menus(TextEditUI \*app)

{

FileMenu \*file;

EditMenu \*edit;

OptionsMenu \*options;

HelpMenu \*help;

GtkAccelGroup \*group = gtk\_accel\_group\_new();

app->filemenu = g\_new(FileMenu, 1);

app->editmenu = g\_new(EditMenu, 1);

app->options\_menu = g\_new(OptionsMenu, 1);

app->helpmenu = g\_new(HelpMenu, 1);

file = app->filemenu;

edit = app->editmenu;

options = app->options\_menu;

help = app->helpmenu;

gtk\_window\_add\_accel\_group(GTK\_WINDOW(app->toplevel), group);

file->menu\_label = gtk\_menu\_item\_new\_with\_label("File");

file->menu = gtk\_menu\_new();

file->new = gtk\_image\_menu\_item\_new\_from\_stock(GTK\_STOCK\_NEW, group);

file->open = gtk\_image\_menu\_item\_new\_from\_stock(GTK\_STOCK\_OPEN, group);

file->save = gtk\_image\_menu\_item\_new\_from\_stock(GTK\_STOCK\_SAVE, group);

file->close = gtk\_image\_menu\_item\_new\_from\_stock(GTK\_STOCK\_CLOSE, group);

file->separator = gtk\_separator\_menu\_item\_new();

file->quit = gtk\_image\_menu\_item\_new\_from\_stock(GTK\_STOCK\_QUIT, group);

gtk\_menu\_item\_set\_submenu(GTK\_MENU\_ITEM(file->menu\_label), file->menu);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(file->menu), file->new);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(file->menu), file->open);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(file->menu), file->save);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(file->menu), file->close);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(file->menu), file->separator);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(file->menu), file->quit);

gtk\_menu\_set\_accel\_group(GTK\_MENU(file->menu), group);

gtk\_widget\_add\_accelerator(file->new, "activate", group, GDK\_N,

GDK\_CONTROL\_MASK, GTK\_ACCEL\_VISIBLE);

gtk\_widget\_add\_accelerator(file->open, "activate", group, GDK\_O,

GDK\_CONTROL\_MASK, GTK\_ACCEL\_VISIBLE);

gtk\_widget\_add\_accelerator(file->save, "activate", group, GDK\_S,

GDK\_CONTROL\_MASK, GTK\_ACCEL\_VISIBLE);

gtk\_widget\_add\_accelerator(file->close, "activate", group, GDK\_W,

GDK\_CONTROL\_MASK, GTK\_ACCEL\_VISIBLE);

gtk\_widget\_add\_accelerator(file->quit, "activate", group, GDK\_Q,

GDK\_CONTROL\_MASK, GTK\_ACCEL\_VISIBLE);

edit->menu\_label = gtk\_menu\_item\_new\_with\_label("Edit");

edit->menu = gtk\_menu\_new();

edit->cut = gtk\_image\_menu\_item\_new\_from\_stock(GTK\_STOCK\_CUT, group);

edit->copy = gtk\_image\_menu\_item\_new\_from\_stock(GTK\_STOCK\_COPY, group);

edit->paste = gtk\_image\_menu\_item\_new\_from\_stock(GTK\_STOCK\_PASTE, group);

gtk\_menu\_item\_set\_submenu(GTK\_MENU\_ITEM(edit->menu\_label), edit->menu);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(edit->menu), edit->cut);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(edit->menu), edit->copy);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(edit->menu), edit->paste);

gtk\_menu\_set\_accel\_group(GTK\_MENU(edit->menu), group);

gtk\_widget\_add\_accelerator(edit->cut, "activate", group, GDK\_X,

GDK\_CONTROL\_MASK, GTK\_ACCEL\_VISIBLE);

gtk\_widget\_add\_accelerator(edit->copy, "activate", group, GDK\_C,

GDK\_CONTROL\_MASK, GTK\_ACCEL\_VISIBLE);

gtk\_widget\_add\_accelerator(edit->paste, "activate", group, GDK\_V,

GDK\_CONTROL\_MASK, GTK\_ACCEL\_VISIBLE);

options->menu\_label = gtk\_menu\_item\_new\_with\_label("Options");

options->menu = gtk\_menu\_new();

options->font = gtk\_image\_menu\_item\_new\_from\_stock(GTK\_STOCK\_SELECT\_FONT, NULL);

gtk\_menu\_item\_set\_submenu(GTK\_MENU\_ITEM(options->menu\_label), options->menu);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(options->menu), options->font);

help->menu\_label = gtk\_menu\_item\_new\_with\_label("Help");

help->menu = gtk\_menu\_new();

help->about = gtk\_image\_menu\_item\_new\_from\_stock(GTK\_STOCK\_ABOUT, NULL);

gtk\_menu\_item\_set\_submenu(GTK\_MENU\_ITEM(help->menu\_label), help->menu);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(help->menu), help->about);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(app->menubar), file->menu\_label);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(app->menubar), edit->menu\_label);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(app->menubar), options->menu\_label);

gtk\_menu\_shell\_append(GTK\_MENU\_SHELL(app->menubar), help->menu\_label);

g\_signal\_connect(G\_OBJECT(file->new), "activate",

G\_CALLBACK(text\_edit\_tab\_new\_with\_file), (gpointer) app->notebook);

g\_signal\_connect(G\_OBJECT(file->open), "activate",

G\_CALLBACK(text\_edit\_open\_file), (gpointer) app->notebook);

g\_signal\_connect(G\_OBJECT(file->save), "activate",

G\_CALLBACK(text\_edit\_save\_file), (gpointer) app->notebook);

g\_signal\_connect(G\_OBJECT(file->close), "activate",

G\_CALLBACK(text\_edit\_close\_file), (gpointer) app->notebook);

g\_signal\_connect(G\_OBJECT(file->quit), "activate",

G\_CALLBACK(quit\_application), NULL);

g\_signal\_connect(G\_OBJECT(edit->cut), "activate",

G\_CALLBACK(text\_edit\_cut\_to\_clipboard), (gpointer) app->notebook);

g\_signal\_connect(G\_OBJECT(edit->copy), "activate",

G\_CALLBACK(text\_edit\_copy\_to\_clipboard), (gpointer) app->notebook);

g\_signal\_connect(G\_OBJECT(edit->paste), "activate",

G\_CALLBACK(text\_edit\_paste\_from\_clipboard), (gpointer) app->notebook);

g\_signal\_connect(G\_OBJECT(options->font), "activate",

G\_CALLBACK(text\_edit\_select\_font), (gpointer) app->notebook);

g\_signal\_connect(G\_OBJECT(help->about), "activate",

G\_CALLBACK(text\_edit\_show\_about\_dialog), (gpointer) app->toplevel);

/\* Add the menubar to the vertical container for the main window \*/

gtk\_box\_pack\_start(GTK\_BOX(app->vbox), app->menubar, FALSE, FALSE, 0);

}

static void text\_edit\_create\_toolbar\_items(TextEditUI \*app)

{

GtkWidget \*toolbar;

GtkToolItem \*new, \*open, \*save;

toolbar = app->toolbar = gtk\_toolbar\_new();

gtk\_toolbar\_set\_show\_arrow(GTK\_TOOLBAR(toolbar), TRUE);

new = gtk\_tool\_button\_new\_from\_stock(GTK\_STOCK\_NEW);

open = gtk\_tool\_button\_new\_from\_stock(GTK\_STOCK\_OPEN);

save = gtk\_tool\_button\_new\_from\_stock(GTK\_STOCK\_SAVE);

gtk\_toolbar\_insert(GTK\_TOOLBAR(toolbar), new, 0);

gtk\_toolbar\_insert(GTK\_TOOLBAR(toolbar), open, 1);

gtk\_toolbar\_insert(GTK\_TOOLBAR(toolbar), save, 2);

g\_signal\_connect\_swapped(G\_OBJECT(new), "clicked",

G\_CALLBACK(gtk\_menu\_item\_activate),

(gpointer) app->filemenu->new);

g\_signal\_connect\_swapped(G\_OBJECT(open), "clicked",

G\_CALLBACK(gtk\_menu\_item\_activate),

(gpointer) app->filemenu->open);

g\_signal\_connect\_swapped(G\_OBJECT(save), "clicked",

G\_CALLBACK(gtk\_menu\_item\_activate),

(gpointer) app->filemenu->save);

gtk\_box\_pack\_start(GTK\_BOX(app->vbox), toolbar, FALSE, FALSE, 0);

}

FileObject \*text\_edit\_file\_new(void)

{

FileObject \*new\_file = g\_new(FileObject, 1);

new\_file->scrolled\_window = gtk\_scrolled\_window\_new(NULL, NULL);

new\_file->textview = gtk\_text\_view\_new();

new\_file->tab\_label = gtk\_label\_new("Untitled");

gtk\_scrolled\_window\_set\_policy(GTK\_SCROLLED\_WINDOW(new\_file->scrolled\_window),

GTK\_POLICY\_AUTOMATIC, GTK\_POLICY\_AUTOMATIC);

gtk\_scrolled\_window\_set\_shadow\_type(GTK\_SCROLLED\_WINDOW(new\_file->scrolled\_window),

GTK\_SHADOW\_IN);

gtk\_container\_set\_border\_width(GTK\_CONTAINER(new\_file->scrolled\_window), 3);

gtk\_text\_view\_set\_left\_margin(GTK\_TEXT\_VIEW(new\_file->textview), 3);

gtk\_text\_view\_set\_right\_margin(GTK\_TEXT\_VIEW(new\_file->textview), 3);

gtk\_text\_view\_set\_pixels\_above\_lines(GTK\_TEXT\_VIEW(new\_file->textview), 1);

gtk\_widget\_modify\_font(new\_file->textview, desc); /\* desc is global font description \*/

gtk\_container\_add(GTK\_CONTAINER(new\_file->scrolled\_window), new\_file->textview);

return new\_file;

}

static void text\_edit\_tab\_new\_with\_file(GtkMenuItem \*menu\_item,

GtkNotebook \*notebook)

{

FileObject \*f = text\_edit\_file\_new();

gint current\_tab;

current\_tab = gtk\_notebook\_append\_page(notebook, f->scrolled\_window, f->tab\_label);

text\_edit\_register\_filename("Untitled", current\_tab);

gtk\_widget\_show\_all(GTK\_WIDGET(notebook));

}

static void text\_edit\_select\_font(GtkMenuItem \*menu\_item,

gpointer notebook)

{

GtkWidget \*font\_dialog = gtk\_font\_selection\_dialog\_new("Choose A Font");

gchar \*fontname;

gint id;

gtk\_font\_selection\_dialog\_set\_preview\_text(GTK\_FONT\_SELECTION\_DIALOG(font\_dialog),

"abcdefghijk ABCDEFHIJK");

id = gtk\_dialog\_run(GTK\_DIALOG(font\_dialog));

switch (id)

{

case GTK\_RESPONSE\_OK:

case GTK\_RESPONSE\_APPLY:

fontname = gtk\_font\_selection\_dialog\_get\_font\_name(GTK\_FONT\_SELECTION\_DIALOG(font\_dialog));

desc = pango\_font\_description\_from\_string(fontname);

break;

case GTK\_RESPONSE\_CANCEL:

break;

}

gtk\_widget\_destroy(font\_dialog);

text\_edit\_apply\_font\_selection(notebook);

}

static void text\_edit\_apply\_font\_selection(GtkNotebook \*notebook)

{

GList \*child\_list;

gint pages;

gint i;

GtkWidget \*swin;

pages = gtk\_notebook\_get\_n\_pages(notebook);

for (i = 0; i < pages; i++)

{

swin = gtk\_notebook\_get\_nth\_page(notebook, i);

child\_list = gtk\_container\_get\_children(GTK\_CONTAINER(swin));

if (GTK\_IS\_TEXT\_VIEW(child\_list->data))

gtk\_widget\_modify\_font(child\_list->data, desc);

}

}

static void text\_edit\_open\_file(GtkMenuItem \*menu\_item,

GtkNotebook \*notebook)

{

GList \*child\_list;

GtkWidget \*dialog;

GtkWidget \*save\_dialog, \*error\_dialog;

gint current\_page;

gint id;

gint offset;

gchar \*filename;

gchar \*contents;

GtkWidget \*scrolled\_win;

GtkWidget \*view;

GtkTextBuffer \*buffer;

GtkWidget \*tab\_name;

GtkTextIter start, end;

GtkTextMark \*mark;

GtkWidget \*prompt\_label;

GtkWidget \*content\_area;

current\_page = gtk\_notebook\_get\_current\_page(notebook);

scrolled\_win = gtk\_notebook\_get\_nth\_page(notebook, current\_page);

child\_list = gtk\_container\_get\_children(GTK\_CONTAINER(scrolled\_win));

view = (GTK\_IS\_TEXT\_VIEW(child\_list->data) ? child\_list->data : NULL);

if (view != NULL)

{

dialog = gtk\_file\_chooser\_dialog\_new("Open A File", NULL,

GTK\_FILE\_CHOOSER\_ACTION\_OPEN,

GTK\_STOCK\_OPEN, GTK\_RESPONSE\_ACCEPT,

GTK\_STOCK\_CANCEL, GTK\_RESPONSE\_REJECT,

NULL);

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

tab\_name = gtk\_notebook\_get\_tab\_label(notebook, scrolled\_win);

switch (id)

{

case GTK\_RESPONSE\_ACCEPT:

filename = gtk\_file\_chooser\_get\_filename(GTK\_FILE\_CHOOSER(dialog));

/\*\*

\* Check to see whether there is text in the buffer before

\* opening a file. If there is, prompt the user to save it.

\*/

buffer = gtk\_text\_view\_get\_buffer(GTK\_TEXT\_VIEW(view));

gtk\_text\_buffer\_get\_end\_iter(buffer, &end);

offset = gtk\_text\_iter\_get\_offset(&end);

if (offset > 0)

{

save\_dialog = gtk\_message\_dialog\_new(NULL, GTK\_DIALOG\_MODAL,

GTK\_MESSAGE\_INFO,

GTK\_BUTTONS\_NONE, NULL);

gtk\_dialog\_add\_buttons(GTK\_DIALOG(save\_dialog),

GTK\_STOCK\_SAVE, GTK\_RESPONSE\_ACCEPT,

GTK\_STOCK\_DELETE, GTK\_RESPONSE\_CLOSE,

GTK\_STOCK\_CANCEL, GTK\_RESPONSE\_CANCEL,

NULL);

prompt\_label = gtk\_label\_new("Save buffer contents?");

content\_area = gtk\_dialog\_get\_content\_area(GTK\_DIALOG(save\_dialog));

gtk\_box\_pack\_start(GTK\_BOX(content\_area), prompt\_label, FALSE, FALSE, 0);

gtk\_widget\_show\_all(save\_dialog);

gtk\_widget\_hide(dialog);

id = gtk\_dialog\_run(GTK\_DIALOG(save\_dialog));

switch (id)

{

case GTK\_RESPONSE\_ACCEPT:

text\_edit\_save\_file(NULL, notebook);

text\_edit\_register\_filename(filename, current\_page);

break;

case GTK\_RESPONSE\_CLOSE:

gtk\_text\_buffer\_get\_bounds(buffer, &start, &end);

gtk\_text\_buffer\_delete(buffer, &start, &end);

break;

case GTK\_RESPONSE\_CANCEL:

gtk\_widget\_destroy(save\_dialog);

return;

}

gtk\_widget\_destroy(save\_dialog);

}

if (g\_file\_test(filename, G\_FILE\_TEST\_EXISTS))

{

g\_file\_get\_contents(filename, &contents, NULL, NULL);

mark = gtk\_text\_buffer\_get\_insert(buffer);

gtk\_text\_buffer\_get\_iter\_at\_mark(buffer, &start, mark);

gtk\_text\_buffer\_set\_text(buffer, contents, -1);

text\_edit\_register\_filename(filename, current\_page);

gtk\_label\_set\_text(GTK\_LABEL(tab\_name), basename(filename));

}

else

{

/\* File does not exist - unknown file name \*/

error\_dialog = gtk\_message\_dialog\_new(NULL, GTK\_DIALOG\_MODAL,

GTK\_MESSAGE\_ERROR,

GTK\_BUTTONS\_OK, NULL);

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

gtk\_widget\_destroy(error\_dialog);

}

break;

case GTK\_RESPONSE\_REJECT:

break;

}

gtk\_widget\_destroy(dialog);

}

}

static void text\_edit\_cut\_to\_clipboard(GtkMenuItem \*menu\_item, GtkNotebook \*notebook)

{

GList \*child\_list;

GtkWidget \*textview;

GtkTextBuffer \*buffer;

GtkWidget \*scrolled\_win;

gint current\_page;

GtkClipboard \*clipboard = gtk\_clipboard\_get(GDK\_SELECTION\_CLIPBOARD);

current\_page = gtk\_notebook\_get\_current\_page(notebook);

scrolled\_win = gtk\_notebook\_get\_nth\_page(notebook, current\_page);

child\_list = gtk\_container\_get\_children(GTK\_CONTAINER(scrolled\_win));

textview = child\_list->data;

buffer = gtk\_text\_view\_get\_buffer(GTK\_TEXT\_VIEW(textview));

gtk\_text\_buffer\_cut\_clipboard(buffer, clipboard, TRUE);

}

static void text\_edit\_copy\_to\_clipboard(GtkMenuItem \*menu\_item,

GtkNotebook \*notebook)

{

GList \*child\_list;

GtkWidget \*textview;

GtkTextBuffer \*buffer;

GtkWidget \*scrolled\_win;

gint current\_page;

GtkClipboard \*clipboard = gtk\_clipboard\_get(GDK\_SELECTION\_CLIPBOARD);

current\_page = gtk\_notebook\_get\_current\_page(notebook);

scrolled\_win = gtk\_notebook\_get\_nth\_page(notebook, current\_page);

child\_list = gtk\_container\_get\_children(GTK\_CONTAINER(scrolled\_win));

textview = child\_list->data;

buffer = gtk\_text\_view\_get\_buffer(GTK\_TEXT\_VIEW(textview));

gtk\_text\_buffer\_copy\_clipboard(buffer, clipboard);

}

static void text\_edit\_paste\_from\_clipboard(GtkMenuItem \*menu\_item,

GtkNotebook \*notebook)

{

GList \*child\_list;

GtkWidget \*textview;

GtkTextBuffer \*buffer;

GtkWidget \*scrolled\_win;

gint current\_page;

GtkClipboard \*clipboard = gtk\_clipboard\_get(GDK\_SELECTION\_CLIPBOARD);

current\_page = gtk\_notebook\_get\_current\_page(notebook);

scrolled\_win = gtk\_notebook\_get\_nth\_page(notebook, current\_page);

child\_list = gtk\_container\_get\_children(GTK\_CONTAINER(scrolled\_win));

textview = child\_list->data;

buffer = gtk\_text\_view\_get\_buffer(GTK\_TEXT\_VIEW(textview));

gtk\_text\_buffer\_paste\_clipboard(buffer, clipboard, NULL, TRUE);

}

static void text\_edit\_show\_about\_dialog(GtkMenuItem \*menu\_item,

GtkWindow \*parent\_window)

{

const gchar \*authors[] = { "Abhishek and Ankur", NULL };

gtk\_show\_about\_dialog(parent\_window,

"program-name", "Text Edit",

"authors", authors,

"license", "GNU General Public License",

"comments", "A simple lightweight GTK+ text editor",

NULL);

}

static void text\_edit\_close\_file(GtkMenuItem \*menu\_item,

GtkNotebook \*notebook)

{

GtkWidget \*scrolled\_win;

gint current\_page;

current\_page = gtk\_notebook\_get\_current\_page(notebook);

scrolled\_win = gtk\_notebook\_get\_nth\_page(notebook, current\_page);

gtk\_widget\_destroy(scrolled\_win); /\* Remove current tab \*/

}

static void text\_edit\_save\_file(GtkMenuItem \*menu\_item,

GtkNotebook \*notebook)

{

GList \*child\_list;

GtkWidget \*dialog;

GtkWidget \*textview;

GtkTextBuffer \*buffer;

GtkWidget \*scrolled\_win;

gint current\_page;

gint response;

GtkWidget \*tab\_label;

GtkTextIter start, end;

gchar \*filename;

gchar \*contents;

current\_page = gtk\_notebook\_get\_current\_page(notebook);

scrolled\_win = gtk\_notebook\_get\_nth\_page(notebook, current\_page);

child\_list = gtk\_container\_get\_children(GTK\_CONTAINER(scrolled\_win));

textview = child\_list->data;

tab\_label = gtk\_notebook\_get\_tab\_label(notebook, scrolled\_win);

if (strcmp(gtk\_label\_get\_text(GTK\_LABEL(tab\_label)), "Untitled") == 0)

{

/\*\*

\* File currently has no name. Allow user to name the file.

\*/

dialog = gtk\_file\_chooser\_dialog\_new("Save File", NULL,

GTK\_FILE\_CHOOSER\_ACTION\_SAVE,

GTK\_STOCK\_SAVE, GTK\_RESPONSE\_APPLY,

GTK\_STOCK\_CANCEL, GTK\_RESPONSE\_CANCEL,

NULL);

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

if (response == GTK\_RESPONSE\_APPLY)

{

filename = gtk\_file\_chooser\_get\_filename(GTK\_FILE\_CHOOSER(dialog));

buffer = gtk\_text\_view\_get\_buffer(GTK\_TEXT\_VIEW(textview));

gtk\_text\_buffer\_get\_bounds(buffer, &start, &end);

contents = gtk\_text\_buffer\_get\_text(buffer, &start, &end, FALSE);

g\_file\_set\_contents(filename, contents, -1, NULL);

text\_edit\_register\_filename(filename, current\_page); // ADDED

gtk\_label\_set\_text(GTK\_LABEL(tab\_label), basename(filename));

}

else if (response == GTK\_RESPONSE\_CANCEL)

{

gtk\_widget\_destroy(dialog);

return;

}

gtk\_widget\_destroy(dialog);

}

else

{

/\*\*

\* Editing a named file so just write textview contents

\* to the existing name.

\*/

filename = text\_edit\_get\_filename(current\_page);

buffer = gtk\_text\_view\_get\_buffer(GTK\_TEXT\_VIEW(textview));

gtk\_text\_buffer\_get\_bounds(buffer, &start, &end);

contents = gtk\_text\_buffer\_get\_text(buffer, &start, &end, FALSE);

g\_file\_set\_contents(filename, contents, -1, NULL);

}

}

static void text\_edit\_register\_filename(gchar \*fname, gint tab\_num)

{

/\*\*

\* Manage the full path to the open file names

\* with a linked list. Keeping the full path around

\* is important so that saving functionality puts the

\* file in its proper place (rather than using only the

\* name in the tab label, which would put it in the current

\* working directory)

\*/

gint found = FALSE;

FileData \*f = g\_new(FileData, 1);

GList \*node = g\_list\_alloc();

f->filename = fname;

f->tab\_number = tab\_num;

node->data = f;

if (filename\_data == NULL) filename\_data = node; /\* First node in list \*/

else

{

/\*\*

\* Go through the list of names and set a full file

\* path for the tab # we are working with. If the tab number

\* passed in is not found, assume we are creating a new name.

\*/

GList \*list = filename\_data;

while (list != NULL)

{

if (((FileData \*) list->data)->tab\_number == tab\_num)

{

found = TRUE;

((FileData \*) list->data)->filename = fname;

break;

}

else list = g\_list\_next(list);

}

if (!found) g\_list\_append(filename\_data, node);

}

}

static gchar \*text\_edit\_get\_filename(gint tab\_num)

{

GList \*list = filename\_data;

while (list != NULL)

{

if (((FileData \*) list->data)->tab\_number == tab\_num)

return ((FileData \*) list->data)->filename;

else

list = g\_list\_next(filename\_data);

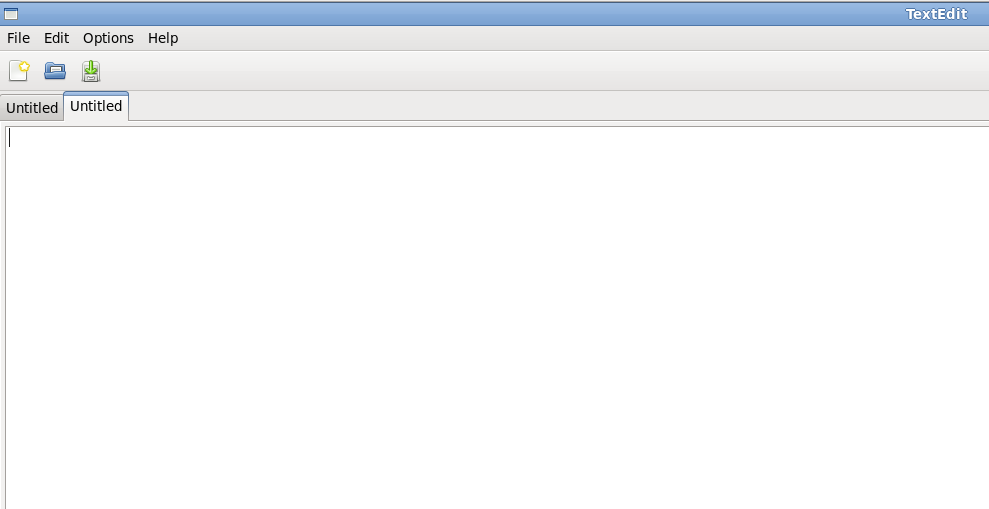
}

return NULL;

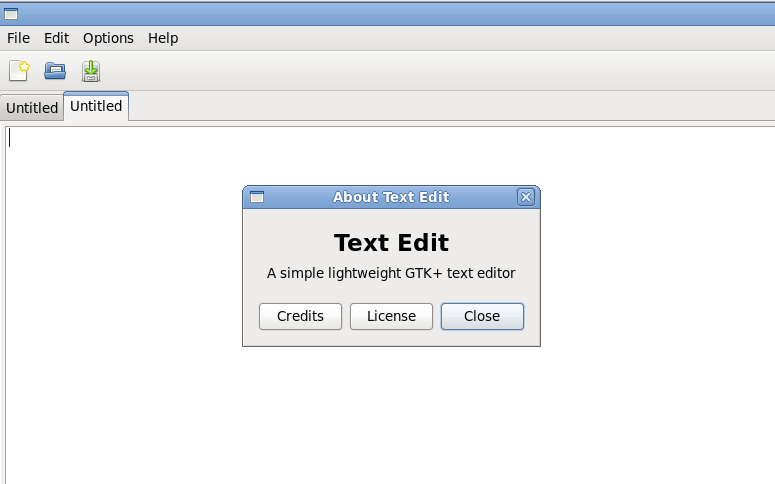
}

**APPENDIX B: SNAPSHOTS**

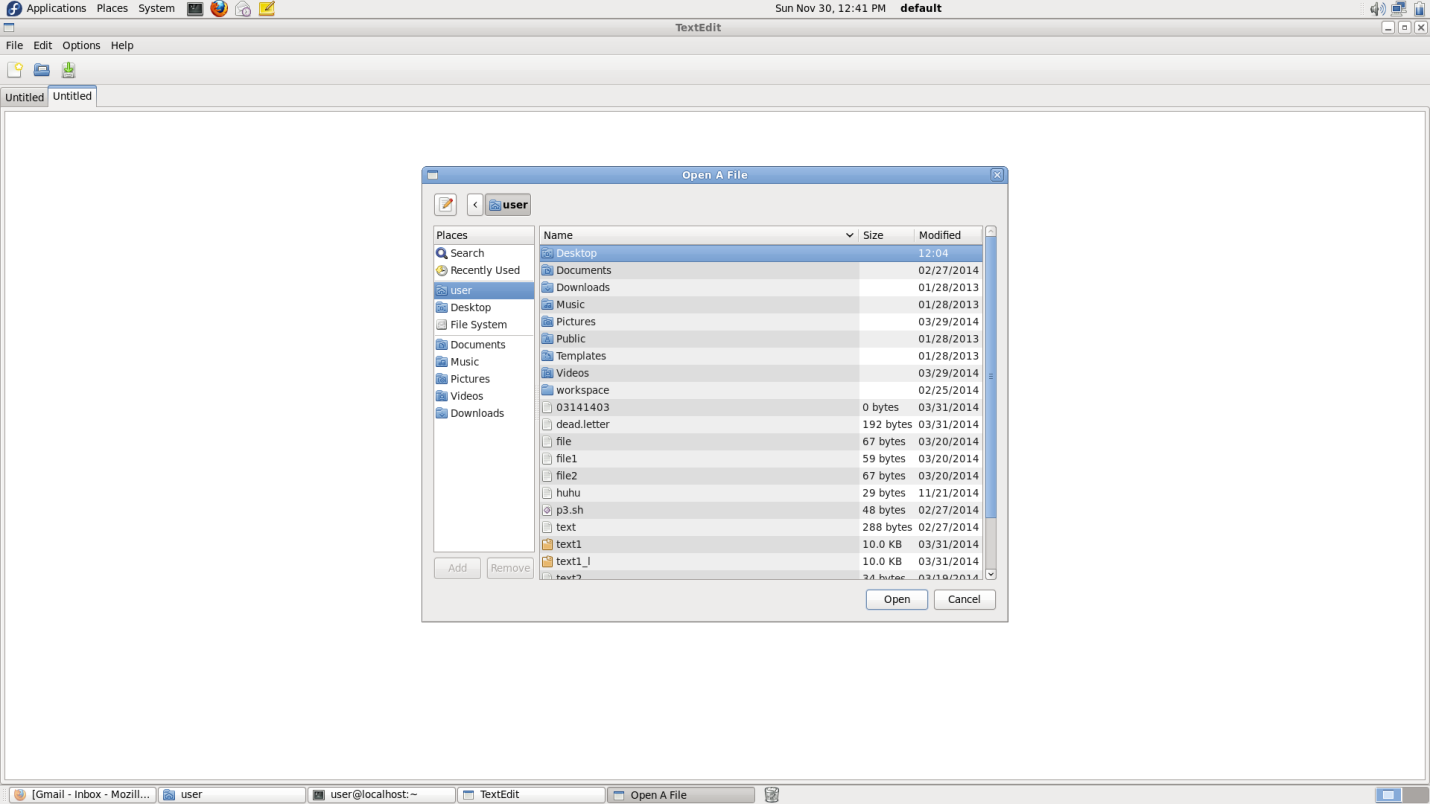
**FIG 8.1**: Snapshot showing multiple tabs feature in TextEdit.



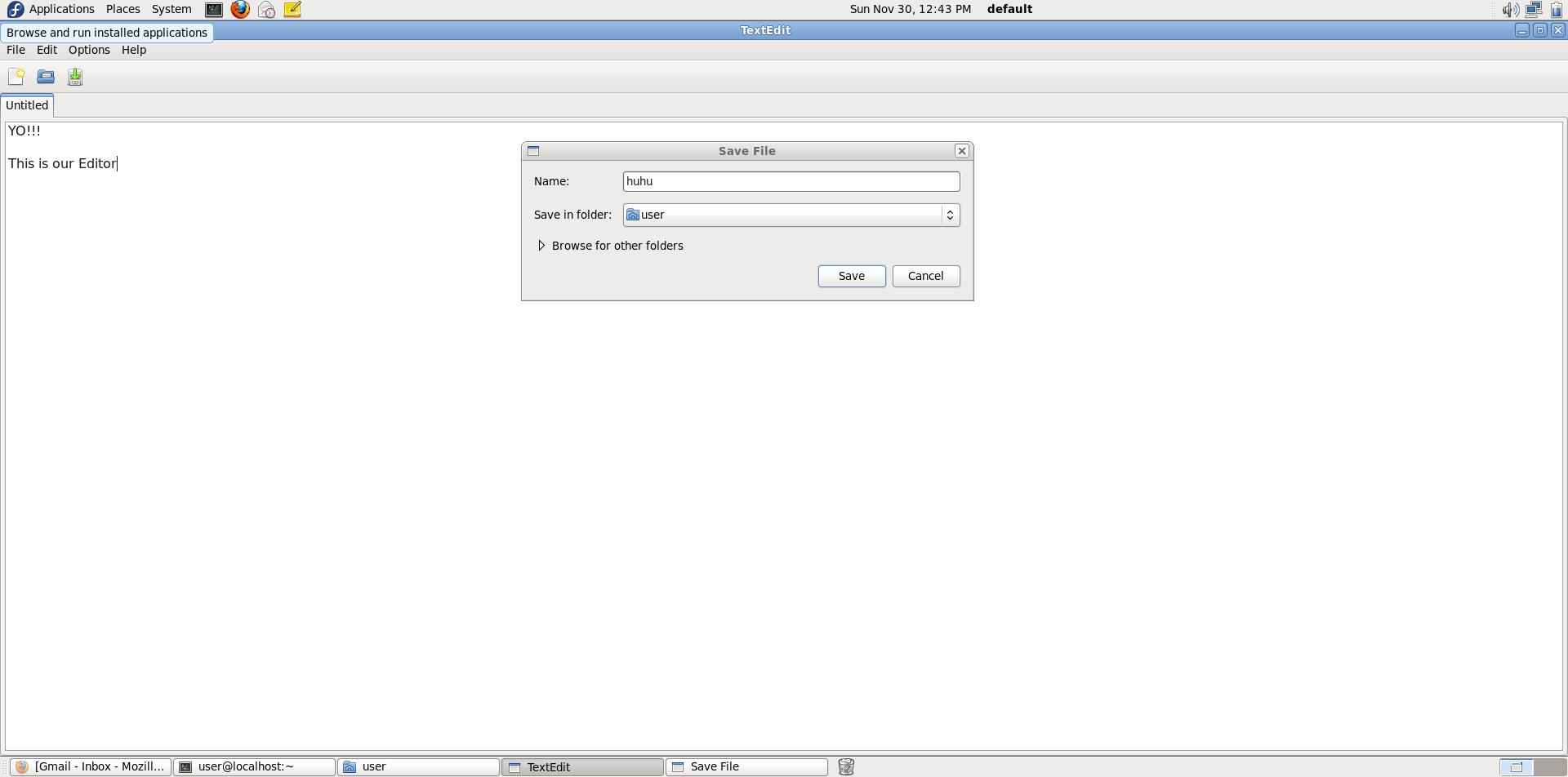
**FIG 8.2**: Snapshot showing About options menu in TextEdit.



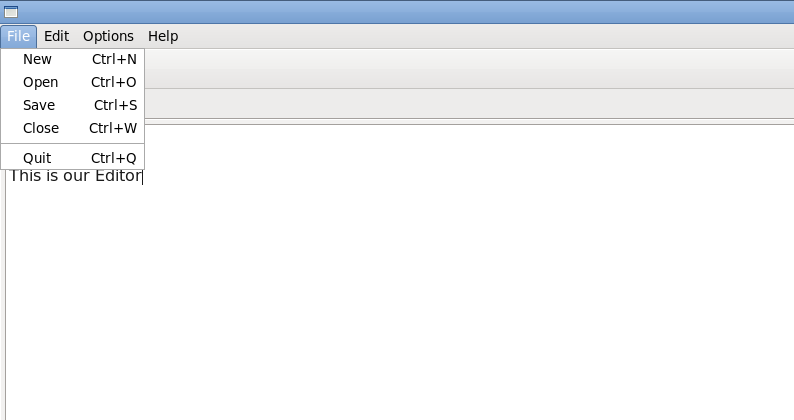
**FIG 8.3**: Snapshot showing Opening file option in TextEdit.



**FIG 8.4**: Snapshot showing Save option in TextEdit.



**FIG 8.5:** Snapshot showing File menu in TextEdit.



**BIBLIOGRAPHY**

* "Data Structures using C and C++", by Aaron M. Tenenbaum, Moshe J. Augenstein, Yedidyah Langsam

* Let Us C by Yashwanth Kanetkar
* Understanding the Linux Kernal by Daniel P.Bovet and Marco Cesati
* Leland L.Beck "An Introduction To System Programming",3rd Edition.
* The GTK+ Project : www.gtk.org