P- 1181

CD BURNING TOOL

Project Report

Submitted in partial fulfillment of the
Requirement for the award of the degree of the

Bachelor of Engineering in Information Technology of
Bharathiar University, Coimbatore.

Submitted by

K. Senthil Kumar
0027S0104

R. Venkatesh
0027S0118

Under the guidance of

Mr.K.R.Baskaran , M.S.
Assistant Professor.

DEPARTMENT OF INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY,
COIMBATORE – 641006.

MARCH 2004.

DEPARTMENT OF INFORMATION TECHNOLOGY
KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University, Coimbatore)

CERTIFICATE

This is to certify that the project entitled

CD BURNING TOOL

Is done by

K. Senthil Kumar
0027S0104

R. Venkatesh
0027S0119

And submitted in partial fulfillment of the
Requirement for the award of the degree of the

Bachelor of Engineering in Information Technology of
Bharathiar University, Coimbatore.

[illegible]

Professor & Head of the Department
(Dr.S.THANGASAMY)

[illegible]

Project Guide
(Mr K.R.Baskaran M.S.)

Certified that the candidates were examined by us in the project work
Viva voce examination held on 26-03-2004

[illegible] 26/3

Internal Examiner

[illegible]

External Examiner

Declaration

We,

K.Senthilkumar	**0027S0104**
R.Venkatesh	**0027S0119**

declare that the project entitled "CD BURNING TOOL ", is done by us and to the best of our knowledge, a similar work has not been submitted earlier to the Bharathiar University or any other institution, for fulfillment of the requirement of the course study.

This project report is submitted on the partial fulfillment of the requirement for all awards of the degree of Bachelor of Engineering in Information Technology of Bharathiar University.

Place: Coimbatore.

Date : 26-05-2004.

[illegible]
[K.Senthilkumar]

[illegible]
[R.Venkatesh]

Project Guided by

____

Mr. K.R.Baskaran M.S.,
Assistant Professor, IT Department

ACKNOWLEDGEMENT

The exhilaration achieved upon the successful completion of any task should be definitely shared with the people behind the venture. This project is an amalgam of study and experience of many people without whose help this project would not have taken shape.

At the onset, we take this opportunity to thank the management of our college for having provided us excellent facilities to work with. We express our deep gratitude to our Principal Dr.K.K.Padmanabhan for ushering us in the path of triumph.

We are always thankful to our beloved Professor and the Head of the Department, Dr.S.Thangasamy whose consistent support and enthusiastic involvement helped us a great deal.

We are greatly indebted to our beloved guide Mr.K.R.Baskaran M.S., Assistant Professor, Department of Information Technology for his excellent guidance and timely support during the course of this project. As a token of our esteem and gratitude, we would like to honour him for his assistance towards this cause.

We also thank our project coordinator Mrs.S.Devaki M.S., and our beloved class advisor Ms.P.Sudha B.E., for their invaluable assistance.

We also feel elated in manifesting our deep sense of gratitude to all the staff and lab technicians in the Department of Information Technology.

We feel proud to pay our respectful thanks to our Parents for their enthusiasm and encouragement and also we thank our friends who have associated themselves to bring out this project successfully.

SYNOPSIS

The main objective of this project "CD Burning Tool" is to assure the efficiency in the data burning. The CD Burning Tool is a GUI and operative tool, which is developed for LINUX platform, this tool supports Burning CD in different file format of different operating system.

When ever the user has clicked the software which is written using "CD Burning Tool", a window will be opened .a wizard will started for selecting the options and formats such as audio, video, mixed etc.., and after completion of selecting formats, user can select list of the files will be explored to the user, the files may be in any storage device.

The existing system does not provide user-friendly tools, which can only used by expert users so this tool and also provides advanced compilation and option in CD Writing.

Some of the other features are burning with an existing data CD,
The tool Provides option for writing formats like audio mode,video mode, data mode, mixed mode.

Mixed mode:

Mixed mode is compiling a CD of different files such as audio, video and data format

Audio and video mode:

MP3 audio files (extension .mp3) have become one of the most common file formats of the Internet community if compressed audio data is to be transmitted. So CD burning tool supports burning of MP3 files. These files can now be dragged and dropped into window for audio compilations just like wave files (.wav) or audio tracks (.cda) or other formats can be compiled

Data mode:

In this mode data of any file can be compiled for example a file of any operating system can be recognized and the data can be burned into the CD

Data is transferred and compiled using a driver, which is developed for our software.

Some of the other features:

CD Speed is a application that provides detailed information on the writing speed of the CD-R/RW installed, as well as on the reading rate and audio extraction quality of the built-in CD/DVD-ROM drives.

CD burning tool tries to determine the specific properties of a CD-ROM drive by transmitting some commands and check the drive's answer afterwards. Unfortunately this method is kind of risky because there is no standard command set supported by each and every CD-ROM model. The result of sending a not supported command to a CD-ROM drive might be a system crash. This phenomenon is already known from the hardware detection.

The bus type and the command set may be defined. If possible our tool presets these options with reasonable default values. If BUS is pre-selected as bus type it's most likely not a good idea to change this settings because our tool detected the bus type by checking the driver. The result therefore is most certainly correct. If you have an IDE CD-ROM then the command list in the dialog contains only a single entry. This is because IDE CD-ROM drives have a standard command set and the differences between CD-ROM models are relatively small. If you're an owner of a SCSI-CD-ROM you shouldn't need to change the pre-selected command set because again our tool has probably pre-selected a „reasonable" entry.

This tool supports DVD drives and DVD CD's.This tool is designed with QT-Designer as front end for designing GUI.

CONTENTS

1. INTRODUCTION

The main objective of this project "CD Burning Tool" is to assure the efficiency in the data burning. The CD Burning Tool is a GUI and operative tool which is developed for LINUX platform.

1.1 Existing System and Limitations

The existing system does not have a good graphical interface for CD recording i.e ease of use. It is a complex one consisting of several steps using the command mode, it is very difficult for a beginner to use such commands and its not user friendly, since all the commands cannot be easily remembered given at any time without any errors.

Limitation:

In the existing system is only in the command mode and only a experienced user can do the CD writing and not fit for a new user .

1.2 Proposed System and Advantages :

The proposed system provides a good GUI for the CD recording purpose, where any user can easily write into CDs.

Advantages:

➢ The GUI developed is

- Ease of use.
- Using only mouse clicks can into a CD.
- Less effort for writing into a CD.
- Even novice users of LINUX can easily write into a CD.

2. SOFTWARE REQUIREMENT ANALYSIS

System study is an activity that encompasses most of the tasks that we have collectively called computer system engineering. System study is conducted with the following objectives.

- Identify the needs.
- Evaluate the system concept for feasibility.
- Perform economic and technical analysis.
- Allocate function to hardware, software, people and other system elements.
- Create a system definition that forms the foundation for all subsequent engineering works.

2.1 Product Definition:

The process of writing into CDs is very much necessary. This writing software is very useful for all formats of file. This is flexible and easy to use. It will be of great use to beginners to easily write into CDs.

2.2 Project Plan:

This project entitled "CD Burning tool " is for developing a GUI package to achieve easy way of writing into CDs. This technique can be applied to text, data and audio file formats. The important steps in writing into CD using this GUI are:

- Selecting the format of the file to be written.
- Use the multi session option if there is a need to write into the same CD another time.
- Providing the SCSI driver numbers.
- Changing the format of the audio file to .wav if in other formats like mp3

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 Purpose:

The primary purpose of the Software Requirement Specification (SRS) is to document the previously agreed functionality, attributes and the performance of the project titled "CD BURNING TOOL". This specification is the primary document upon which all of the subsequent design, source code and test plan will be based.

3.1.1 Scope:

The scope of the document is to describe the requirements definition effort. The SRS is limited to description of the easier way of burning CDs in LINUX environment.

3.1.2 Acronyms and definitions

CD

Compact disk

GTK

GIMP TOOL KIT

GNU

GNU is not UNIX

OS

Operating system

SCSI

Small computer system interface

3.1.4 Overview

This document specifies the requirements for the product Interactive CD Burning tool. After review, no additions or modifications to these specifications can be made without the approval of the producers.

This document is the authoritative source on all non-implementation specific matters concerning the finished product; thus any functional requirements not listed in this document are considered optional and may or may not be present in the finished product.

The rest of this document specifies the requirements for the Interactive CD Burning tool. Section 2 describes the situation where the Interactive CD Burning tool will operate, and briefly lists the requirements. This section is for those who want a quick overview of the Interactive CD Burning tool's requirements.

Section 3 lists the specific requirements for the software, when section 4 lists the general requirements for the whole product.

3.2 GENERAL DESCRIPTION

3.2.1 Project perspective

Interactive CD Burning tool's purpose is to design a tool for interactively write CD's in Linux OS.

The Interactive CD Burning tool 's primary use is to provide a good graphical User Interface for any Linux novice user who wish to burn CD's without taking much effort by using simple mouse clicks.

3.2.2 Product functions:

- Must able to select the CD writing modes through button clicks.
- Support both Gnome and KDE Support.
- Supports on the FLY mode of writing into CDs.

3.2.3 End users:

The end users will be any user of the LINUX environment. They may not be experts in the area of LINUX, and therefore a GUI will be necessary so that the process of writing into CDs can be done easily.

3.2.4 General constraints:

The Interactive CD Burning tool software will be implemented in UNIX-C. Since UNIX-C is an OS dependent language, the project members can't document, for example, the installation in all possible environments.

The CD WRITING software will be implemented in LINUX, version 8. Since LINUX is an OS that's getting wide use these days, the installation in all possible environments.

3.3. Interfaces

3.3.1 User interface

Though the user interface the user must be able to configure the following,

1. The Label of the CD.

2. The speed at with the CD writes to use for writing.

3. The Writer SCSI bus number.

4. The Reader SCSI bus number.

3.3.2 Performance

Interactive CD Burning tool must reach a certain performance, which is to be tested with the following test case:

- The audio mode is converted into the wave (.wav) format for writing into CD.
- The entire content of the CD will be erased not a specific file.
- The SCSI driver numbers should be provided, so that our software can locate the driver.

3.4 General requirements

3.4.1 Reliability

The producers will do their best, within the time limits imposed on them, to deliver a perfectly working product. No actual guarantees to the amount of software faults (bugs) in the finished product will be made, other than the fact that the product must pass a previously created set of tests. The details of this test bench are beyond the scope of this document, and will be specified in a separate document.

3.4.2 Installation

The required steps to install the project CDBURNING TOOL must be documented .No automatic installer program will be required.

3.4.3 Version control

The source code generated will be kept in some version control system. However, given this project's almost non-existent resources, the system may be simply an archive of the older versions of the source code.

4. SYSTEM DESIGN

The system design is the high level strategy for solving the problem and building a solution. System design includes decisions about the organization of the system into subsystems, allocation of subsystems to hardware and software components, and major conceptual and policy decisions that form the framework for the detailed design.

Design of user interface:

The user interface includes widgets, radio buttons for selecting various options and text box for input file selection. The user is given the option of selecting data, audio or erase operation with separate buttons. Everything can be selected using the mouse pointer. The user can also provide the SCSI bus numbers int the GUI itself. Separate window is provided for selecting the files to be written. When the writing operation is over the user is provided with a post message box indicating the end of the operation.

Interactive CD Burning Tool:

The following steps are involved in writing into CDs in various file formats.

- Collect data (i.e.) creation of image file.
- Make iso image of the data either audio CD or data CD.
- Provide the GUI for writing.

DATA CD:

This system provides the facility of writing the file that is in the data format. Here the data that needs to be written is first collected and the iso image file is being created .After that the selected contents are being burned into the CD.

AUDIO CD:

This system provides the facility for writing the files that are in the audio format. The tracks that needs to be written are extracted from the hard disk/CD ,they are converted into wave (.wav) file if they are in some other format .Then the selected files are being burned into the CD.

ERASING CD:

The contents present in the CD are erased if this option is selected from our GUI, Here the SCSI drive numbers should be provided, so that our GUI can know the location of the CD driver.

CD-CD WRITING:

This system is able to write contents from one CD into another CD. The contents need to be written are collected and burned. This method supports "on the fly mode "of writing into CDs (i.e.) the contents from one CD can be transferred directly from one CD into another without the use of any buffer so there is lot of time saved in this mode.

5. SYSTEM TESTING

Testing is an activity to verify that a correct system is being built and is performed with the intent of finding faults in the system. Testing is an activity, however not restricted to being performed after the development phase is complete. But this is to be carried out in parallel with all stages of system development, starting with requirement specification. Testing results once gathered and evaluated, provide a qualitative indication of software quality and reliability and serve as a basis for design modification if required.

System Testing is a process of checking whether the development system is working according to the original objectives and requirements. The system should be tested experimentally with test data so as to ensure that the system works according to the required specification. When the system is found working, test it with actual data and check performance.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and the attendant "cost" associated with a software failure are the motivation forces for a well planned, thorough testing.

5.1 Testing Objectives :

The testing objectives are summarized in the following three steps. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has high probability of finding an error. A successful test is one that uncovers as --yet-undiscovered errors.

5.2 Testing Principles:

All tests should be traceable to customer requirements. Tests should be planned long before testing begins, that is, the test planning can begin as soon as the requirements model is complete. Testing should begin "in the small" and progress towards resting "in large". The focus of testing will shift progressively from programs to individual modules and finally to the entire project. Exhaustive testing is not possible. To be more effective, testing should be one, which has highest probability of finding errors.

The following are the attributes of good tests:

- A good test has a high probability of finding an error.
- A good test is not redundant.
- A good test should be "best of breed"
- A good test should be neither too simple nor too complex.

5.3 Levels of Testing:

The details of the software functionality tests are given below. The testing procedure that has been used is as follow:

- ➢ Unit Testing
- ➢ Integration Testing
- ➢ Validation Testing
- ➢ Output Testing

Unit Testing:

Unit testing is carried out to verify and uncover errors within the boundary of the smallest unit or a module. In this testing step, each module was found to be working satisfactory as per the expected output of the module. In the package development, each module is tested separately after it has been completed and checked with valid data. Unit testing exercise specific paths in the modules control structure to ensure complete coverage and maximum error detection.

The project is divided into three modules. These three modules are developed separately and verified whether they function properly.

Integration Testing:

Integration Testing address the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of higher-order tests are conducted. The main objective in this testing process is to take unit-tested modules and build a program structure that has been dictated by design.

The following are the types of integrated testing:

Top Down Integration:

This method is an incremental approach to the construction of the program structure. Modules are integrated by moving downward the control hierarchy, beginning with the main program module. The module sub-ordinates to the main program module are incorporated to the structure in either a depth-first or breadth-first manner.

Bottom Up Integration:

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

The low level modules are combined in to clusters that perform a specific software sub-function.

A driver i.e. the control program for testing is return to coordinate test case input and output.

The cluster is tested and drives are removed and clusters are combined moving upward in the program structure.

The three modules which are developed during unit testing are combined together and they are executed and verified whether the linking of the files and the corresponding modules without any error.

Validation Testing:

At the end of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and correction testing begins.

Validation Test Criteria:

Software testing and validation is achieved through series of black box tests that demonstrate conformity with the requirements are achieved, documentation is correct and other requirement are met. Here the three modules which are checked in the integration testing are assembled and programmed in the code is testing for any correction.

Output Testing:

Output testing is series of different test whose primary purpose is to fully exercise the computer based system. Although each test has a different purpose, all the work should be verified so that all system element have properly integrated and perform allocated functions.

Output testing is the stage of implantation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. The input screens, output documents were checked and required modification made to suite the program specification. Then using rest data prepared, the whole system was tested and found to be a successful one.

Here giving checks the whole system sample inputs i.e. the password and it is checked for the whole function so that the system functions well and the requirements are met.

6. FUTURE ENHANCEMENT:

This project is developed in Linux 8, we used GTK environment for GUI design in this project. More feature can be included in this project as the future enhancement

The following characteristic features can be included

- CD extra traction modules can be included.
- Can play audio CDs .
- Progress bar can be included .
- Multi session with rewriteable option in readable CD s.
- Writers can be enhanced for DVD ROMs.
- Erasing of a particular file .

7.CONCLUSION:

Finally, let us summarize the features of the project and give proposals for future work that would give continuity and would enhance the functionality of the application. With these suggestions, the project will turn conceptually more robust and applicable for enterprises.

Outstanding features:

Among the main features of the techniques presented here, the following aspects stand out as implementation novelties:

The best known application that implements a easy way of writing contents into CDs in LINUX environment which is done by just mouse clicks on all types of file formats like data, audio etc. In this project the technique of on the fly is implemented. Hence the writing speed is improved .The multi session used here can be used to write into a CD already containing data's, where writing is done in the existing space in the CD.

8. BIBLIOGRAPHY:

References:

1. ROB PIKE, " UNIX PROGRAMMING ENVIRONMENT "

2. Christopher Negus , " REDHAT LINUX BIBLE "

Websites:

1. www.linux.org	DECEMBER-2003
2. www.gtk.org	JANUARY-2004
3. www.redhat.com	JANUARY-2004

```
#ifdef HAVE_CONFIG_H
#  include <config.h>
#endif

#include <gtk/gtk.h>

#include "callbacks.h"
#include "interface.h"
#include "support.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "cdburning.h"
#include "parse.h"

GtkWidget * dlg_bx;
GtkWidget * Gbutton;
GtkWidget * G_fs;
GtkWidget * G_mylist;
int dlgnumber=1;
int bFly;
int bMulti;
int G_row;
char  G_seleted_list_item[50];
int NewCopyErease = 1;

int  Format =1;
int  MultiorNew = 1;
void    on_window1_destroy                (GtkObject      *object,  gpointer        user_data)
                                                  {
          system("rm -fr /tmp/cdburningtool/source/*; rm -f /tmp/cdburningtool/dest/*;" );
          gtk_main_quit();
}
```

```
void
on_button_burn_clicked              (GtkButton       *button,
                                    gpointer         user_data)
{
        char cImg[300];
        char cDataCD[300];



        char cWav[200];
        char cAudioCD[300];

        printf("\nBurn clicked\n\n");
        fflush(stdout);



        NewCopyErease =1;
        Format = 1;
        MultiorNew = 1;





        if( (NewCopyErease == 1) && (Format == 1) && (MultiorNew == 1) ) {

                sprintf(cImg,"mkisofs -o  /tmp/cdburningtool/dest/datacd.iso -J -r -v -V %s
/tmp/cdburningtool/source/ ;",cdsetup.label);

                g_print("\n%s \n",cImg);

                system(cImg);

                if(bMulti)
```

```
    {

                        sprintf(cDataCD,"cdrecord -v speed=%s dev=%s -multi
   /tmp/cdburningtool/dest/datacd.iso",cdsetup.speed,cdsetup.cdwriter);


                        g_print("CD is not closed");
                        }
                  else {


                        sprintf(cDataCD,"cdrecord -v speed=%s dev=%s
   /tmp/cdburningtool/dest/datacd.iso",cdsetup.speed,cdsetup.cdwriter);
                        g_print("CD closed");
                  }


                  system(cDataCD);
                  g_print("\n%s \n",cDataCD);

           }




}


void
on_button_apply_clicked              (GtkButton       *button,
                                        gpointer         user_data)
{


     GtkWidget * cdw = lookup_widget(GTK_WIDGET(button),"entry_cdwriter");

       gchar * cdwtr = gtk_entry_get_text(GTK_ENTRY(cdw));

    GtkWidget * cdr = lookup_widget(GTK_WIDGET(button),"entry_cdreader");
```

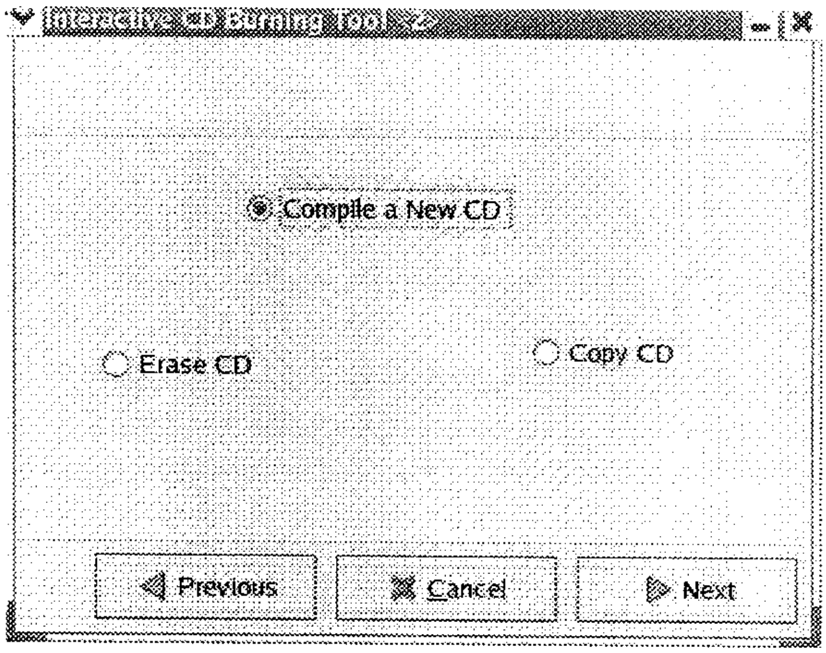
```
gchar * cdrdr = gtk_entry_get_text(GTK_ENTRY(cdr));

GtkWidget * volume = lookup_widget(GTK_WIDGET(button),"entry_label");

gchar * cvol = gtk_entry_get_text(GTK_ENTRY(volume));


GtkWidget * cdw_dev = lookup_widget(GTK_WIDGET(button),"entry_dev_writer");

gchar * cdwtr_dev = gtk_entry_get_text(GTK_ENTRY(cdw_dev));

GtkWidget * cdr_dev = lookup_widget(GTK_WIDGET(button),"entry_dev_reader");

gchar * cdrdr_dev = gtk_entry_get_text(GTK_ENTRY(cdr_dev));



strcpy(cdsetup.cdwriter,cdwtr);
strcpy(cdsetup.cdreader,cdrdr);
strcpy(cdsetup.label,cvol);

strcpy(cdsetup.cdwriter_dev,cdwtr_dev);
strcpy(cdsetup.cdreader_dev,cdrdr_dev);

g_print("\nApplying\n");

}
```

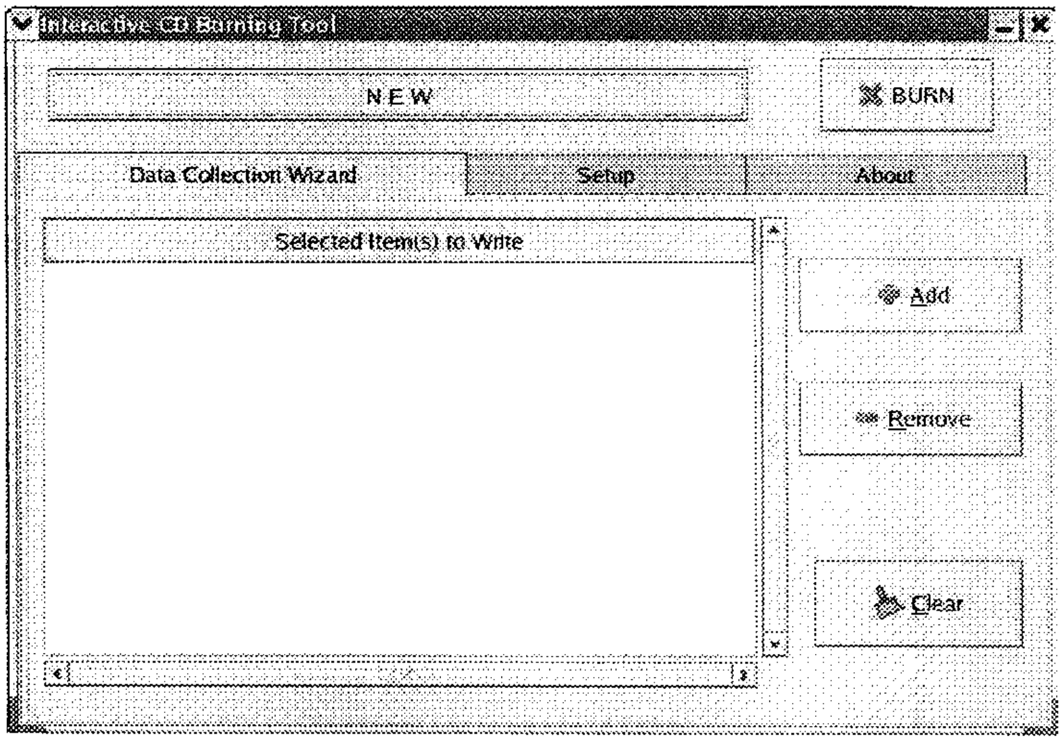
```
void
on_button_remove_dataed_clicked     (GtkButton      *button,
                                     gpointer       user_data)
{

        char cstring[200];

        GtkWidget * list = lookup_widget(GTK_WIDGET(button),"clist1");
        gtk_clist_remove(list,G_row);
        printf("\n%s removed",G_seleted_list_item);
        sprintf(cstring,"cd /tmp/cdburningtool/source;rm -fr %s",G_seleted_list_item);
        system(cstring);

        g_print("\nchecking...");
        fflush(stdout);

}
void
on_button_add_dataed_clicked        (GtkButton      *button,
                                     gpointer       user_data)
{


        GtkWidget * list = lookup_widget(GTK_WIDGET(button),"clist1");

        G_mylist = list;
        g_print("\nADD clicked");


        GtkWidget * fileselection1 = create_fileselection1 ();
        gtk_widget_show (fileselection1);
        G_fs = fileselection1;

}
```

```
void
on_button_clear_datacd_clicked      (GtkButton      *button,
                              gpointer        user_data)
{
        GtkWidget * list = lookup_widget(GTK_WIDGET(button),"clist1");
        g_print("\nClear clicked");
        gtk_clist_clear( (GtkCList * ) list);

        system("rm -fr /tmp/cdburningtool/source/* ");
}

void on_ok_button_added_clicked      (GtkButton      *button,
                              gpointer        user_data)
{
        gchar * items[1] ;

        int mode=0;


        gchar mystring[500];
        gchar cstring[500];
        gchar result[100];
        g_print("\nOk clicked:");

        strcpy(mystring,gtk_file_selection_get_filename(GTK_FILE_SELECTION(G_fs)));

        mode = parse(mystring,result);
                if(mode){ //dir
                sprintf(cstring,"cp -fr %s  /tmp/cdburningtool/source/",mystring);
                printf("\nDirectory\n");
        }
```

```
          else {



                   sprintf(cstring,"cp -f %s  /tmp/cdburningtool/source/",mystring);
          }


          system(cstring);
          items[0] = result;
          g_print("\n%s",cstring);





          gtk_clist_append((GtkCList *)G_mylist,items);


          gtk_widget_destroy(G_fs);




}



void
on_cancel_button_added_clicked        (GtkButton       *button,
                                       gpointer         user_data)
{
         printf("\nCancel clicked...");
         gtk_widget_destroy(G_fs);
}
```

```
void
on_clist1_select_row           (GtkCList      *clist,
                        gint          row,
                        gint          column,
                        GdkEvent      *event,
                        gpointer      user_data)
{
        gchar * text;
        g_print("\nSelected row");
        fflush(stdout);
        G_row= row;
        g_print("selected row:%d",row);
        gtk_clist_get_text(GTK_CLIST(clist), row, column, &text);
        g_print("Text :%s",text);

        strcpy(G_selcted_list_item, text);
        g_print("\n\nSelected string....:%s\n\n",text);
        fflush(stdout);
        g_print("\nselected............\n");

        fflush(stdout);


}
```


Interactive CD Burning Tool <2>
[✓ Compile a New CD]
Erase CD
Copy CD
Previous
Cancel
Next


Interactive CD Burning Tool
N E W
BURN
Data Collection Wizard
Setup
About
Selected Item(s) to Write
Add
Remove
Clear