Open-source Video Recording Scheduler

by

Mohd Azhar bin Abd. Hadi

Dissertation is submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Information Technology)

JANUARY 2005

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Open Source Video Recording Scheduler

by

Mohd Azhar bin Abd. Hadi

A project dissertation submitted to the
Information Technology Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirements for the
BACHELOR OF TECHNOLOGY (HONS)
(INFORMATION TECHNOLOGY)

Approved by,

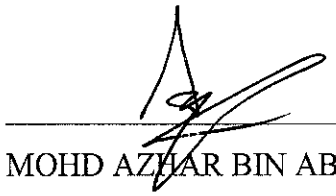
_____

(Mr. Izzatdin Abdul Aziz)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
JANUARY 2005

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



MOHD AZHAR BIN ABD. HADI

ACKNOWLEDGEMENT

First and foremost, I would like express my gratefulness to Allah S.W.T for His Bless and Guidance that strengthens me in facing every challenge in completing this project.

I would like to thanks my supervisor, Mr. Izzatdin bin Abdul Aziz for his understanding, morale support, and guidance that really motivates me to accomplish this project.

I would also like to thank all my families and friends for their advice and morale supports in this project.

I would also like to say "thank you" for everybody else that has contributed to this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND NOMENCLATURES

2-D: Two Dimensional

3-D: Three Dimensional

Apache: Open-source web server application

CLI: Command Line Interface

CCD: Charge-coupled Devices

DivX: New video compression standard that is evolved from the MPEG4

JPEG: Joint Picture Expert Group

Kernel: The core of operating system

Linux: An open-source operating system that is operating in similar way of UNIX.

MPEG: Motion Picture Expert Group

MJPEG: M-JPEG

GNOME: GNU Network Object Model Environment

GNU: Open-source Foundation Organizations

GUI: Graphical User Interface

NTSC: National Television System Committee

OS: Operating System

PAL: Phase Alternate Line

PCI: Peripheral Component Interconnect

V4L: Video For Linux

V4L2: Video For Linux version 2

Xvid: Open Source DivX

ABSTRACT

This report explains about the project on developing a video recording scheduler application on open-source platform. The scope of this project is developing a GUI application that allows users to set the video recording schedule, and users are capable to set the codec used for recorded video compression. This application would run on open-source platform, therefore the target users is the open-source user that have a problem in using current open-source applications especially novice users. This project would be completed phase by phase, and for application development, the method applied is evolutionary development approach. This project depends on open-source tools and platform for application development. The end product of this project is able to perform all targeted features without producing any errors or machine problems. As a conclusion, this project has achieved its objectives in order to solve some problems of open-source users in scheduling their video recording.

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Video is one of the essential technologies used in modern human life. The most widely-used video technology is a television, where the video signals are broadcasted through cable or radio wave, and the television box would capture the signals and convert it into the visual form. Television becomes essential in modern society where almost every family would have it. However, computer technology has changed this trend where personal computer owners normally consider on substituting the television with television capture card (TV tuner).

1.1.1 Video Capturing Application

Using video captured devices such as TV tuner has given some advantages to users. It is cheaper and requires smaller space, and users could easily manipulate the captured video signal according to what they want. Though, users will depend on the software for accessing the video devices and manipulating the captured video signals. If the software or application only has capability to capture the video signal, then users could do nothing except to capture the video.

Most video capturing applications are sold together with the video devices. Usually these applications are limited to specific brand of video devices. However, there are also other applications that are developed to support several types of video devices, according to certain specifications like chipset types and drivers. Among of these applications, there are proprietary applications and non-proprietary applications.

1.1.2 Open-source Application

Users actually have alternatives in choosing applications according to their requirements. There are proprietary applications available in the market, which need to be bought, registered, and sometimes need to have a contract before it is legally used. Users could also choose to use non-proprietary application or also known as an open-source application. Open-source applications are originally free and most of them could be downloaded directly from the internet. Open-source applications are also free in term of sharing the source code, where users could access the source code and modify it wherever necessary to suit their requirements.

Considering video-related aspect, open-source video applications have some advantages compared to the proprietary software since it is based on the free technology and developed with mass contributions of developers, including video experts. Starting from the release of popular open-source platform (Linux OS) in 1990s until now, developers had written and developed lots of drivers, modules, tools, and applications to support various types of video devices in the market.

1.2 Problem Statement

1.2.1 Problem Identification

Even though open-source video capturing applications offer flexible hardware supports and customizable options, it still has problems for certain users. Users have to deal with lots of configurations for video interface, drivers and codec in order to use the application. These configurations allow more technical customizations for the application, but it might be too complicated to be understood by certain people. Thus, it is important to ensure that the application is understandable and usable by target users.

Most of video capturing applications developed for Linux provide recording features, including the encoding (compression) and decoding (decompression) processes. Video recording activities normally come with the needs of scheduling. Users could use specific Linux utilities for scheduling their video recording. However, some users especially novice users know nothing about this utilities. Additionally, users would be preferred if they can set the schedule directly on video capturing applications.

1.2.2 Significance of the Project

The end product of this project does not target to replace current video capturing applications, but it is developed as an alternative for users who do not wish to deal much on command lines and use the interactive GUI application for scheduling their video recording tasks. At present, the number of Linux users has been rapidly increasing and the application might suit the needs of Linux novice users that might still unfamiliar with Linux commands. Further more, the application is considered as a contribution to the society since users could use the application freely and also improve it wherever necessary.

1.3 Objectives and Scope of Study

The objectives of this project are:-

- *develop an application to schedule video recording processes*

The application developed in this project would allow users to set the schedule of their video recording processes. The application would access some operating system utilities for scheduling purposes. For video recording process, the application would manipulate some video tools for capturing, encoding, and recording the video signal into video file. In order to achieve this objective, a small-scale study has been conducted on understanding the video capturing concept and manipulating video devices.

- *Develop an application to be used on open-source platform, using open-source development tools*

The application developed would target to be used on the open-source p latform such as Linux OS. Application development processes in this project would use the open-source tools such as Gnome development tools and libraries. The target group of this application is Linux novice users. Simple studies are conducted to understand Linux platform and to use Gnome programming tools.

1.4 Feasibility of the Project within Scope and Time Frame

This project has started with conducting several studies to identify the constraints, requirements and options for application development. After the studies are conducted, the author starts to configure the tools that would be used such as downloading, installation, and also include reconfiguration and recompilation processes. Since open-source tools are used, lots of documentations are available in the Internet and most of the problem faced by the author is already discussed in the online forums with the suggested solutions from advanced users and experts. All the information could also be found from the books in the library. With all the resources provided, the project would be feasible in the time frame given. To have a better view of the project timeline, please refer at Appendix A in Appendices.

CHAPTER 2

LITERATURE REVIEW

2.1 Video Capturing

Digital video is now an integral part of many aspects of business, education, and entertainment. In general, digital video means the visual information represented in a discrete form. This visual information might be captured from the real scene. Richardson (2000) has defined; "Video image is a projection of a 3-D scene onto a 2-D plane over a period of time" (p. 5). He also stated that generating a digital representation of a video scene can be considered in two stages: acquisition (sampling the projection of the scene into an electrical signal, for example via CCD array) and digitization (sampling the projection spatially and temporally, and converting each sample to a number or set of numbers).

According to Dixon (1999), PC users who want the best quality in capturing and recording video into their hard drives prefer to use PCI capture board rather than external video devices like television. Since PCI bus provides data rates of around 100 MB/sec, video could be captured in full-size and full-rate video and only limited only by users' hard disk speed.

2.2 Video Device Support on Linux

The video device used in this project depends on Bt878A chipset. Fernandez (2005) mentioned that Bt878A chip is the successor of the famous Bt848. When Brooktree (now Conexant) company designed this new chipset, they did include a build-in low frequency ADC, with the objective of digitalize the sound signals coming from the tuner to be later on demodulated by software (obtaining the FM Stereo, TV Stereo, RDS, MTS, etc.) using the OS driver. Unfortunately no manufacturer has used this possibility and they simply use the ADC to get mono sound from a tuner into the PC.

Bt878 and Bt848 chips would need bttv drivers in order to work on Linux platform. bttv driver is written by Gerd Knorr. Gerd Knorr also write V4L modules. V4L or Video For Linux is a well-known video interface modules for Linux OS and it eases the process of accessing and manipulating video devices. V4L is already embedded as a module in Linux since Linux kernel 2.0. After that, the V4L2 module is released with more optimization and better features than V4L.V4L2 is written by Bill Dirks. It is tested on Linux staring from kernel 2.4 and officially included in the current Linux kernel, version 2.6.

2.3 Benefits of Open-Source Platform and Tools

Linux is chosen as a platform for this project application development. According to Rehman (2003); "Linux is a very good operating system as it has all of the development tools available. Now users can install Linux on high-end workstations from Sun Microsystems, HP, and IBM as well as commodity PC hardware available everywhere. It provides stability and most of the people are familiar with development tools" (p11).

Linux is an OS developed under the open-source GPL license. In this project, open-source platform and development tools are used for certain reason. O'Riordan (2002) had discussed on the benefits of Free (Open-source) Software over Proprietary Software: "Free Software empowers it's users by allowing them to help themselves by making changes they want to the software (or getting someone else to make these changes). It allows people to help their neighbours by sharing the software, proprietary software does the opposite: it makes sharing illegal telling people that it is a criminal offense to say "yes" when someone asks for help. And Free Software allows people to help their community by distributing improved versions of the software."(p. 2).

Open-source software is beneficial for society since it gives more freedom to users to use and improves the software. The author finds a great chance in learning development techniques and methods from the open-source software.

2.4 Application Development

2.4.1 Software Engineering

The application development in this project would apply the method of evolutionary development approach. Under this approach, there are two types of prototyping, the evolutionary prototyping and the throw-away prototyping. Evolutionary prototyping objective is to deliver a working system to end-users, while throw-away prototyping objective is to validate the system requirements. The application of evolutionary approach for application development in this project is described in Section 3.1.3a.

Evolutionary prototyping would start the prototype development according to the user requirements that is best understood and have highest priority. This technique has much in common with techniques of rapid application development (RAD) and Joint Application Development (JAD) (Millington and Stapleton, 1995).

According to Sommerville (2000), there are two main advantages of adopting evolutionary prototyping to software development, one is the accelerated delivery of the system and the second is the engagement of users in system development.

2.4.2 Usability Engineering

Usability engineering is one of the essential parts of product development life cycle, since it will influence the designing process of product. Usability engineering objective is to ensure that application is designed according to user requirements and become usable by target users. It is useless to have great-functional application if it could not be used. Dr. Susan Dray had said "If you can not use it, it doesn't work".

Usability engineering consists of gathering the usability requirements in the initial phases of the project. According to Preece (1994), usability requirements processes consist of three analyses. The first one is task analysis, where the characteristics (particularly cognitive characteristic) of user requirements are determined. The second one is user analysis. This process would determine the scope of the users including their intellectual ability, cognitive processing ability and experience. The final process involved is environment analysis, which focuses on the real environment within which the system would operate.

2.5 Video Compression

2.5.1 Composite Video Encoding and Decoding

Most video capturing applications provide an option for users to select television standard, which are NTSC and PAL. According to Poynton (2003), insufficient channel capacity was available at the outset of television broadcasting to transmit three separate color components. The NTSC and PAL techniques were devised to combine (encode) the three color components into a single composite signal. (p. 62).

In the encoding process of NTSC and PAL, video components would be filtered to form luma and color difference signals. This video form is known as YUV (for PAL) and YIQ (for NTSC). The decoding processes of NTSC and PAL into video components would also use this kind of video form. (Poynton 103)

2.5.2 The Importance of Compression

Video in YUV, YIQ, and YUY2 form have a big number of bits. Therefore uncompressed captured video signal would consume a lot of disk space. Woodward (2003) stated that YUY2 color space is 16 bits per pixel and there are approximately 30 frames per second. By multiplying that all together (640 x 480 x 2 bytes x 30 fps), there are 18,432,000 bytes per second (or roughly 17.5MB per second) for the uncompressed video.

Therefore for video recording, video compression should be applied in order to avoid the problem of insufficient disk space and disk writing rate. Woodward (2003) did mention these in his article:

"Most modern hard disks can perform sustained writes at 17.5MB/sec, some older models typically cannot. Depending on the fragmentation of your hard disk, it is entirely possible that the hard disk and Windows buffering might not be able to keep up with your raw video capture. The result is lots of consecutively dropped frames, which is never desirable. But even if your hard drive could keep up a sustained write without dropping any frames, would it be able to store what you are trying to capture? At 17.5MB per second, 1 minute of video equals approximately 1 GB of data. One hour would be nearly 62GB of data!"

The common solution to this problem is using the real-time compressor such as Huffyuy or Motion JPEG (MJPEG). These codec can yield good compression ratios with no loss or virtually no loss of visual quality. (Woodward 2003)

2.6 Video Capturing Application

Most experts would recommend users to use xawtv for capturing V4L-supported devices. xawtv is developed by Gerd Knorr. It is originally developed only to view the television program. Then it had been developed into an integrated application that could scan the video devices, record the captured video, and also other features such as scanning the channel, capturing image and customize the video input format.

Even though it is recommended by experts and provides lots of features, some novice users prefer to use other TV applications such as TVTime. This is because xawtv is console-based application and does not have interesting interface. It does support mouse clicking for its menu, but the resolutions and the colors of the menu text could be considered as bad.

CHAPTER 3

METHODOLOGY/PROJECT WORK

3.1 Procedure Identification

This project life cycle consists of two major parts, which are *Project Feasibility* and *Project Acquisition*. Each part has two phases, and these four phases are related in certain order. Each phase, starting from the phase Concept, must be completed before the project could move to the next phase. Figure 1 illustrates the framework of project life cycle phases.

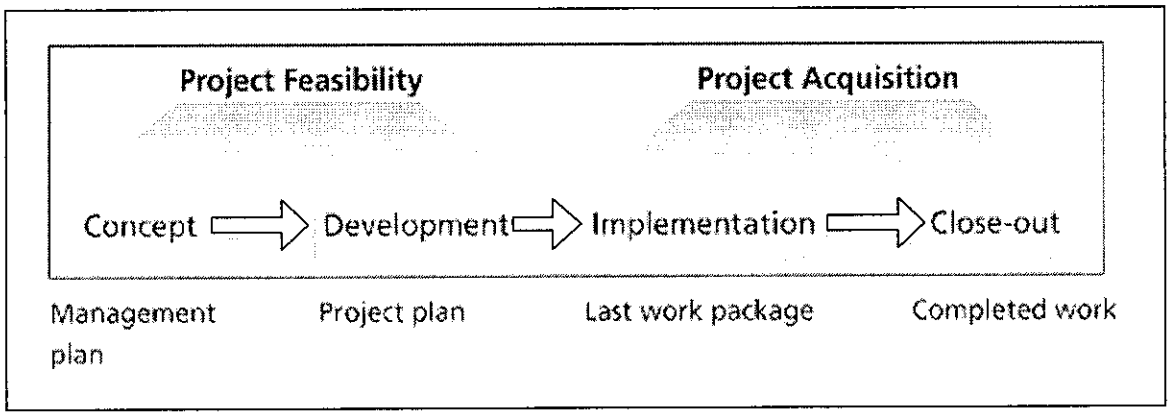


Figure 1: Project Life Cycle

The initial two phases (Concept and Development) which are under project feasibility focus on the project planning, while the last two phases (Implementation and Close-out) under project acquisition part would focus on delivering the actual work.

3.1.1 Concept Phase

In the concept phase, the whole project idea is determined. This process involves the resolving of underlying project concept, overview of project works, and defining the project scope. In this phase, the overall project works are identified.

3.1.2 Development Phase

In the development phase, the defined project idea, concept, and scope would be used to plan the project. The project works identified in concept phase would be reviewed to structure more detail activities or tasks involved in the project.

In planning this project, time constraint is concerned. Project scope and identified project works are used to develop the project schedule. In preparing project schedule, activities determined are sequenced, to see the connection between every task. Critical path is determined here.

3.1.3 Implementation Phase

In general, the task in project implementation phase is to execute the project plan. In project plan execution, every defined task would be completed according to the schedule. Here, the schedule is used in controlling the project plan execution, since the project progress would be compared with the schedule to determine any urgencies and necessary steps to be taken. The author decides not to have too detailed tasks in order to avoid the scheduling control process become complicated.

The application development and usability testing processes take place in this project phase.

3.1.3a Application Development

Evolutionary development approach is applied for the application development processes in this project. This approach is chosen since several technical constraints and user requirements are still not discovered or validated in the beginning of this project. There are two types of prototyping in evolutionary development methods. The prototyping type applied in this project is *throw-away prototyping*, since it is used to validate the requirements. Figure 2 illustrates the phases involved in evolutionary development approach.

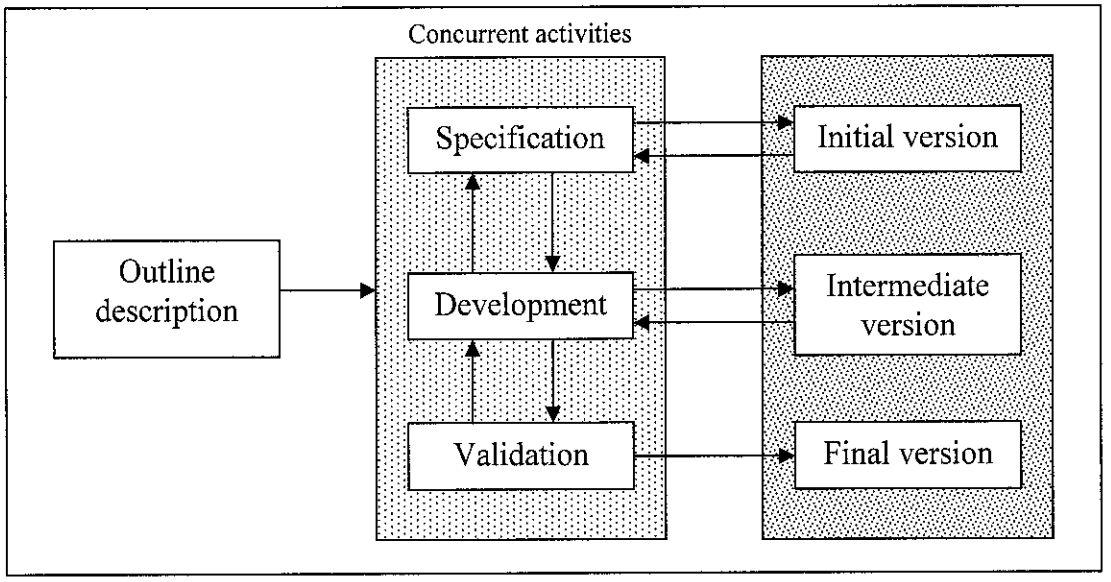


Figure 2: Product Development Life Cycle

By developing the simple prototype in the beginning of project development, new constraints and requirements could be identified. Additionally, any requirements that are unclear or not understood by users could be validated through prototyping.

Prototyping process would optimize and validate the user requirements. These optimized requirements then would be used to improve back the prototype design before it is redeveloped. In general, the prototype would 'evolve' from the simple application into several versions of functional applications.

Evolutionary development approach also help in reducing project time by overlapping most tasks in the processes of requirement analysis and product development. Thus, it is really suitable for this project that only requires 14 weeks to be accomplished.

3.1.3b Usability Testing

The usability testing is performed in order to obtain the user feedback on the usability of the product. The method applied for usability testing in this project is *heuristic evaluation* method.

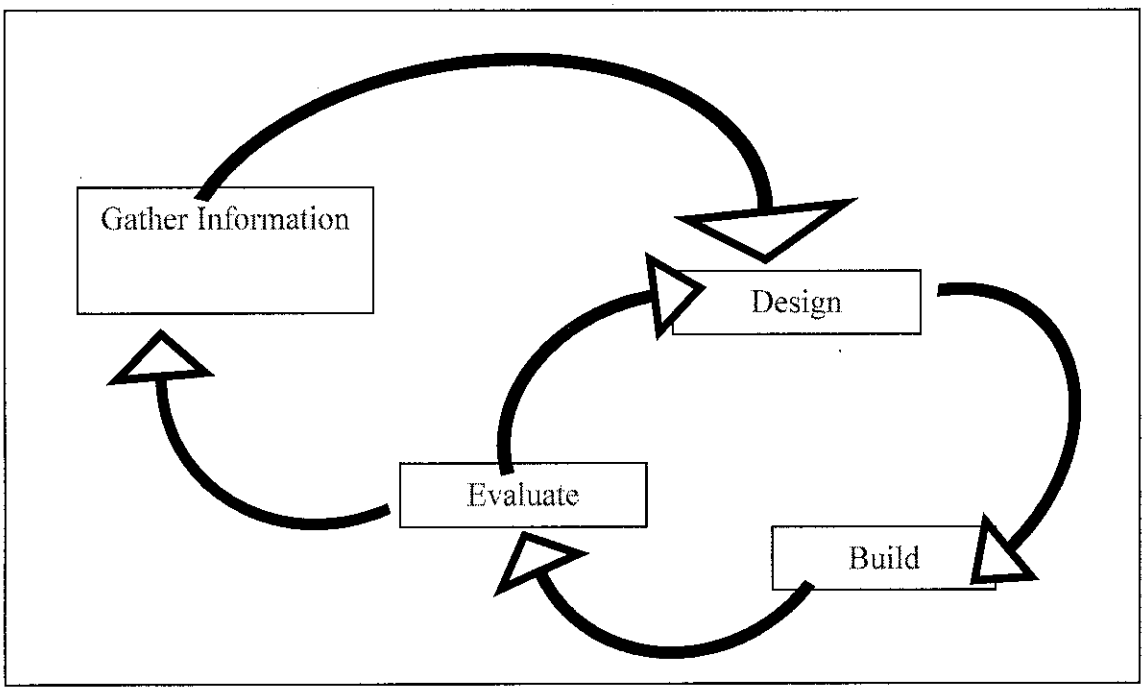


Figure 3: Iterative Design Lifecycle

Figure 3 illustrate the concept of iterative design lifecycle in application development. Application is designed and developed according to information (requirements) gathered and then it will be evaluated to gather new information (requirements). After that application is redesigned, redeveloped, and reevaluated until it matches all the requirements or overcome all the problems.

This concept is quite similar with prototyping, except it is specified for application's usability. In this project, most of the evaluation processes are done by the authors alone, which means the author as an application developer is testing the application as an evaluator.

However, the author also set a formal *heuristic evaluation* test, in order to identify the major usability flaw of the application. This test is just conducted once when the application prototype is almost completed. The test involves five evaluators who will inspect the user interface elements in the application and conform its compliance to the 10 Jacob Nielsen's heuristic principles. The reports would be given to the author. All problems mentioned would be combined to assign severity ratings to application and to rank the problem. The ranked problems then would be solved.

3.1.4 Close-out

In this project phase, all the project works have been completed. The author would need to complete all the documentation and reports regarding on the project. All of the related product development materials such as samples, manuals, tools would be archived.

3.2 Tools and Devices

3.2.1 Development Tools

The application is developed using the GNOME development tools, with C programming language. GNOME provides complete set of tools and libraries for application development. It has also cross-platform programming features, since any module that use GNU libraries could be compiled on Windows platform through the GNU port on Windows, Cygwin.

The tools used include the GCC for compilation, linking object file and building, GDB for debugging, automake for creating the makefile (for compiling and installation purposes) and emacs the integrated source editor that could be linked directly to the compiler and debugger tools. Other tool such as Glade is also used in designing the application GUI.

3.2.2 Modules and Libraries

In spite of GNOME development tools, GNOME libraries are also used. GNOME libraries used in the project are GTK, Glib, and ATK. GTK and ATK libraries is more on managing the application GUI resources including the event handling, while Glib consists on operating system operation such as memory management, file management, and standard C library like mathematical and date functions.

For accessing and manipulating video devices, V4L interface module is used. A lot of drivers are written to support V4L, and thus there are lots of V4L-supported devices available. Currently V4L2 is released with better features, but there are some drivers are not written to support V4L2. In order to overcome the driver-support problem, the video interface modules package that support both V4L and V4L2 drivers was released.

DivX compression would be used as a compression technology for recording video. It is selected since other compression formats like MPEG still produce undesired big file size. The current divx library used for project is DivX 5.

3.2.3 Platform

For the application development, Fedora Core 3 operating system kernel 2.6.9 has been used as a platform. Fedora Core 3 is chosen based on its reputations in application development projects and also based on the author's experience of compilation problem with other Linux OS.

3.2.4 Hardware/Devices

Application developed in this project is tested on two different Intel x86 machines. Specifically, the processor used in both machines is Intel Pentium 4 with i686 architecture. One processor has 1.7 GHz speed while the other one has 2.4 GHz.

Video capture card would be used for testing purpose as a video device. In this project, there are two video capture cards with different chipset have been tested. They are PixelView Play TV Pro and ASUS TV Pro.

PixelView Play TV Pro video capture card is based on the chipset Conexant Fusion 878A, which are also referred as Bt878A. This chipset designed by Brooktree (now known as Conexant) and requires bttv driver in Linux. ASUS TV Pro video capture card is based on the chipset Phillips SAA7134 which requires the saa1734 driver.

Both bttv and saa1734 driver are V4L2 supported, and already included in Linux kernel.

3.3 Project Works Details

3.3.1 Configuration

There are several requirements need to be fulfilled in this project before starting the development process. Those requirements are to set up the libraries and supporting tools for the application development.

As mentioned previously, Linux kernel 2.6 is embedded with V4L2 modules. However, in default Linux installation, this module is not enabled. Users need to configure the kernel to use the module and then recompile the kernel. Thus the kernel source code must be available for the recompilation process.

The application developed in this project actually uses the transcode tools for the process of capturing and recording the video input signal. Transcode installation would require ffmpeg video codec libraries and the new version of GNU libraries. It also requires the pre-installation of any video codec library that user would use with transcode tools. In this project, DivX 5 codec would be used and thus its codec library needs to be installed prior to the transcode installation. Then the transcode would be configured to enable the access of DivX 5 codec during its installation process.

3.3.2 Application Functional Design

The application developed in this project use the concept of manipulating other resources for achieving its objective. This is done through sending the bash shell command to the system. The program would manipulate the command string to include the command options and argument according to the user setting through the application GUI.

3.3.2a Capturing

Capturing processes is used in two different ways in this application. The first one is to capture and display the video signal directly from the video capture devices to the monitor. The second one is to capture the video signal and write it in the form of binary file and simultaneously implement the encoding process.

For capturing and displaying video signal, program would send the transcode command to run the xawtv program with all the video setting options which is the video sources, video devices, audio devices and television standard.

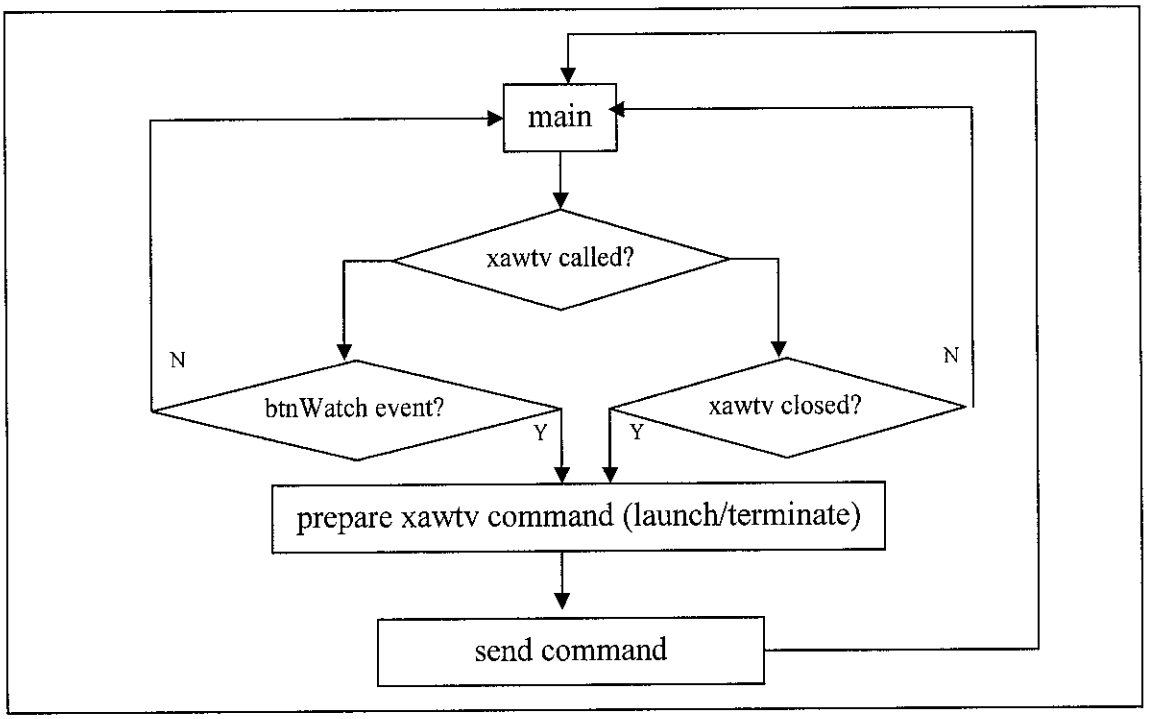


Figure 4: Application event handling for viewing video input.

For capturing and recording video, the program would manipulate the transcode tool. Transcode is linux text-console tools for video-stream processing. Transcode run on a platform that supports shared libraries and the processes of decoding/encoding is done by loading modules that are responsible for feeding transcode with raw video/audio streams (import modules) and encoding the frames (export modules).

For capturing the video signal, the program would send the shell command to activate the transcode with the setting of the video input devices and also other setting of the video options such as frame buffer and signal standards (still under consideration).

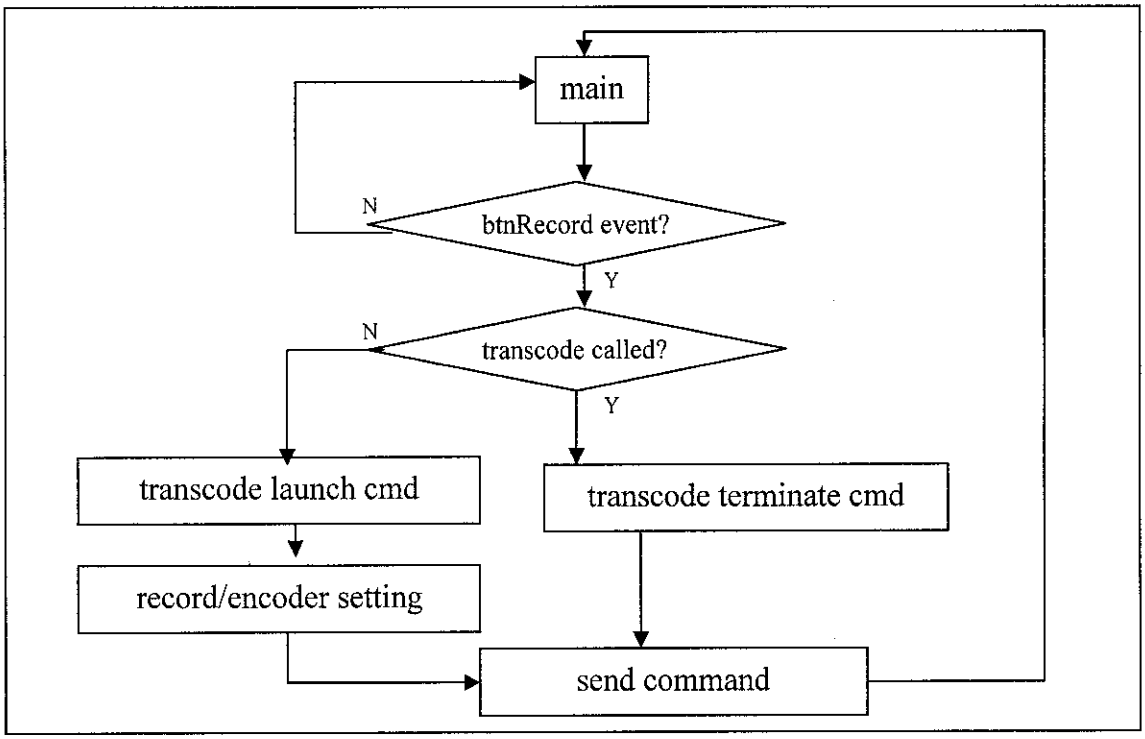


Figure 5: Application event handling for recording video input

3.3.2b Recording and Encoding

For recording the video signal captured, the shell command that activates transcode would be added with some arguments for the settings of video output file and also the video options such as codec type for encoding process. The locations and the filename might be set priorly by users but as a default the file would be saved in the /tmp directory and the recording time and date would be set as its name.

3.3.2c Scheduling

For scheduling process, this application uses the Linux utilities for scheduling, which is cron daemon. This program would set up the crontab file according to the schedules set by the user and when the setting need to be applied as a workable schedule, this program would just send the command to activate the crontab files. The schedule set would store the shell command of activating transcode tools for capturing and recording processes.

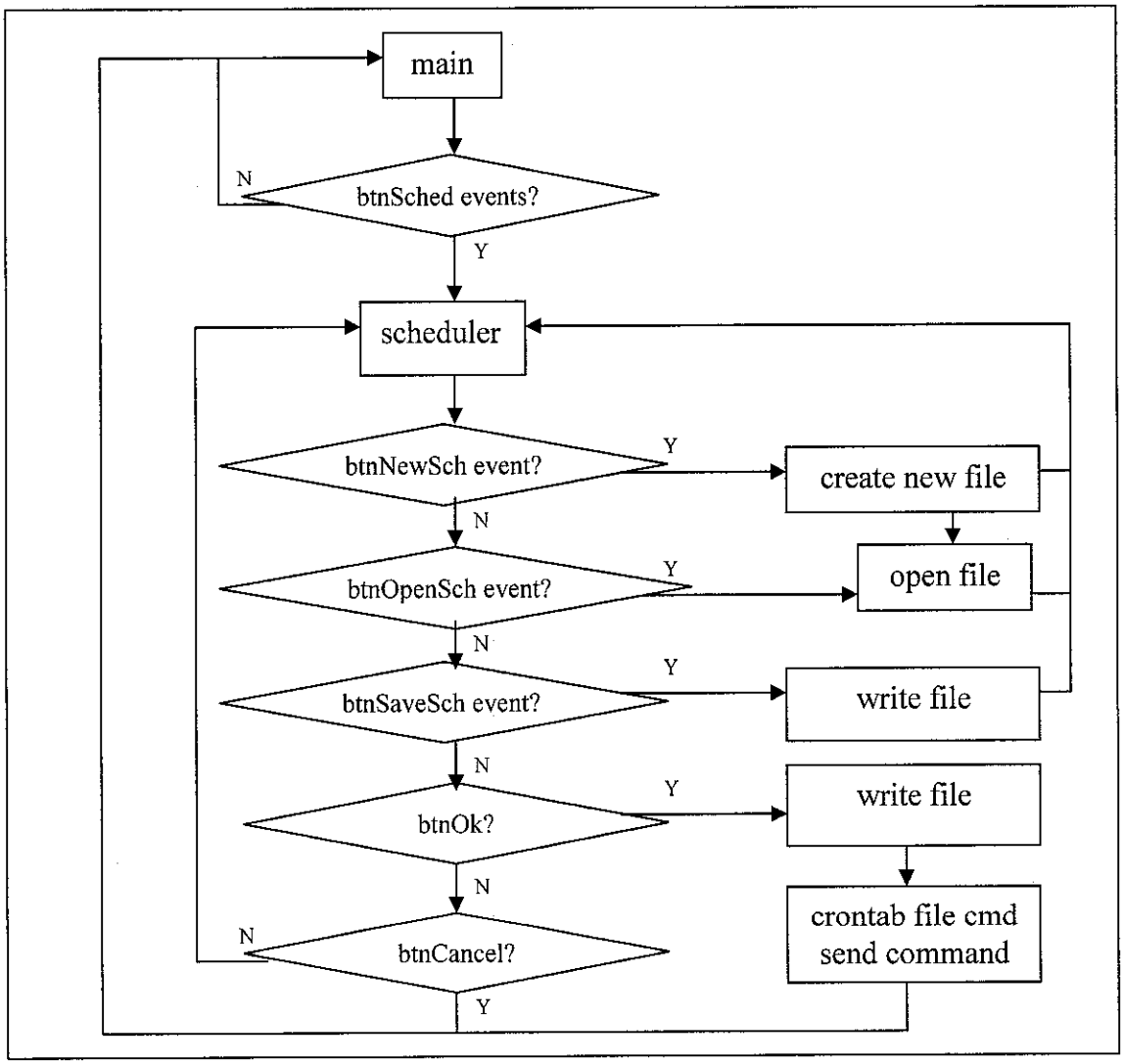


Figure 6: Application event handling for scheduling

3.3.3 Application User Interface

Graphical User Interface is chosen to be applied for this project's application in order to achieve higher degree in usability quality. Currently there are two forms used in the application. The first form is the main form, where users will see when the application start and from here users will select the option to view or record the video and also to configure the video input settings such as video devices. The second form is the schedule manager. Users will access this from the main form. The schedule manager allow user to create their schedule, customize and save it. Please refer to Appendix B to see the screenshot of both application forms.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Results and Findings

4.1.1 Product

The actual end product developed in this project is named as 'autovideo'. This name is decided by the author, to depict the concept of automation of video-related tasks.

As mentioned in Section 1.3, the target of this project is to develop the application that enables users to set the video recording activities on open-source platform. Therefore, this application is basically more on the 'schedule manager' and not necessarily does the video capturing and compression processes.

4.1.1a Features

This application provides several features to user. From this application, user could view the captured video just like watching television. Actually this program does not perform any capturing operation. It only runs the xawtv program to view the captured video (This is already mentioned in the Section 3.1.3a).

User could also record the captured video from this application and set the schedule for that processes in interactive way. This application also allows users to choose their preferred codec for encoding the recorded video. There are also additional features that are thought necessary such as channel scanning.

4.1.1b Comparison

The comparison that would be discussed here is not exactly comparing between this application and other applications, but it actually comparing between the how users perform the task with and without this application.

This application enables users to view the captured video by running the xawtv program from interactive GUI. User just needs to click a button to perform this task. Users could also set the option of video capturing processes such as the video devices, television standards, and video source types from this application with just a few clicks. Without this application, users need to run the xawtv program manually through command shell. Users also need to remember all the command required for setting the options of video capturing processes.

This application also enables users to set the schedule for video recording processes with customizable options including the video codec type. This is done through the manipulation of transcode tools and Linux cron daemon. Therefore users can perform this task interactively. Other applications such as xawtv and motv also provide scheduling features for video recording processes, which also by manipulating Linux cron daemon. Though, users would still need to run those applications from command shell, and users would also have to face the inappropriate user interface provided on the application, where sometimes the label is unreadable.

Other application like TVTime does not provide scheduling features. Linux users might be able to schedule the video recording on this application by using the Linux cron daemon. However, novice users might have no idea about this cron daemon and could not perform the task. Besides, sometime even the advanced users could forget the commands used to set the schedule on cron daemon. Therefore this application might help in term of scheduling purposes. Expert users could also utilize this application for setting the schedule to other tasks as long as they know how the cron daemon works.

4.1.2 Usability Testing

The method applied for application usability testing in this project is heuristic evaluation. This method is designed by Nielsen, J. (1992), where the test is conducted with the help from several evaluators for judging the application interface to conform it to the heuristic principles.

For this project, five people are chosen as evaluators. According to Nielsen, J. (1992), five people are sufficient for the heuristic evaluation method. All of chosen people are non-Linux users but familiar with video capturing applications. All of these evaluators are given two hours to try using the application and then they are filling the evaluation form given (Please refer to Appendix C to view the heuristic evaluation form).

Evaluator	Questions							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
User 1	3	2	4	4	2	5	2	2
User 2	5	2	3	5	3	5	4	3
User 3	4	0	3	4	3	4	4	2
User 4	3	1	2	4	2	5	3	3
User 5	3	2	3	5	3	4	2	3
Average	*3.6*	*1.4*	*3*	*4.4*	*2.6*	*4.6*	*3*	*2.6*

Table 1: Heuristic Evaluation Results

Table 1 show the results obtained from the evaluators' answers in the usability test. These results are generated from the structured questions only. For all questions except question number 2, lower rating means worse problem. Therefore, according to the Table 1, the problem asked in question number 5 could be said as the worst problem in this application. The question number 5 actually reflects on the effectiveness of the application's error message to prevent users from having a problem.

From the analysis done through the evaluators' answers (including their comments and suggestions), several problems have been listed out to be solved. One of the problems is regarding the schedule concept. Most evaluators would feel great when see that this application could create unlimited number of schedules and manage it in the form of files. Though, for the first time, all evaluators would just simply set the schedule and save it without even create the schedule file first. Regarding this problem, most of them suggest to disable all GUI controls in scheduler form if the application does not load any schedule file.

Since evaluators do not fully understand the concept of cron daemon, they tend to believe those schedule files are only manageable by this application. They say that the problem is this application does not provide a function to delete those files.

This is unexpected usability issue faced by the author since in the designing process, the author assume that most users have used software like office application and thus they should know that most software do not have a function to delete the file and users could do it externally.

Another major problem is regarding the customizable video-related options. Since all the evaluators are non-Linux users (so they become the novice when using it), they do not understand why the video devices is set '/dev/video0' as a default. The evaluators also complaint that the appearance of the buttons used is unattractive.

4.2 Discussion

This project has faced a lot of troubles in the configuration process, since the author need to learn the exact way of setting up all the video interface module and tools and also to find out the exact solutions whenever technical problems occur. The good example is the kernel configuration for V4L module. Actually the author just realized the needs of recompiling the kernel recently since initially the author assume that V4L module could be directly used since it is already embedded in the kernel. Therefore when any problem occurs such as incorrect video capturing, the author assume that it is caused by improper tools installation or program's bugs.

Another issue is the conflict between V4L and V4L2. The video device used in this project is supported by V4L2 interface (which is included in kernel), while most of tools used by the application only support V4L module. Thus the author tried to install the videodevX package, the module package that support both V4L and V4L2 modules. However when tools fail to capture the video the author could not tell whether it is caused by that conflict or by something else.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

This project does not target on substituting the video capturing applications available, but it targets on simplifying the processes of scheduling the video recording process on the open-source platform, which may cause difficulties to novice users. Thus the objective of the application is to allow users setting the video recording schedule on the open-source platform.

The application developed depends on external tools for manipulating the video devices, which is called *transcode*. *Transcode* is actually a powerful tool in Linux that can access various types of video device and perform the encoding and decoding processes using various types of compression format. However, *transcode* sometime might cause difficulties to users due to its instability in low performance machine. Therefore it is recommended if this application could have its own features to perform video related process and not depend on *transcode* tools.

The end product of this project has achieved the objectives, where it provides the targeted feature to users (scheduling) including other features that are thought necessary. The product has also been tested using heuristic evaluation method, in order to identify any design flaw. It is recommended to anyone that interested to develop video application to consider on the usability problem first before starting the development process. As a conclusion, the project has successfully achieved all of its objectives, and this is proven practically by its end product.

REFERENCES

1. Schwalbe, Kathy. (2004). *Information Technology Project Management*. Canada: Course Technology

2. Richardson, Iain E. G. (2000). *Video Codec Design*. New York: John Wiley & Sons.

3. Dixon, Douglas. June 1999, *Connecting Video To Your PC: Video Capture Hardware.*<http://www.manifest-tech.com/media_pc/pc_vcr.htm> Recent access: Feb 2005

4. Fernandez, J. D. 17 April 2005, *Analog to Digital Converter with 16 bits and 448000 Samples per second based in the Bt878A.*<http://www.domenech.org/bt878a-adc/index-e.htm> Recent access: 20 April 2005

5. Dirks, Bill. 26 June 2003 *Video For Linux Two.*<http://www.thedirks.org/v4l2/> Recent access: 8 May 2005

6. Knorr, Gerd. 2003, *video4linux* http://linux.bytesex.org/v4l2/ Recent access: 8 May 2005

7. Rehman, R. U. (2003) *The Linux Development Platform (Bruce Peren Series)*. New Jersey: Prentice Hall

8. O'Riordan, Ciaran. (2002) *Learning GNU C.*

9. Lowe, Kwan. 2004, *Kernel Rebuild Guide.* <http://www.digitalhermit.com/linux/Kernel-Build-HOWTO.html> Recent access: 8 May 2005

10. Sommerville, Ian. (2000) *Software Engineering*. Harlow, England: Addison-Wesley

11. Preece, Jenny. (1994) *Human Computer Interaction*. Harlow, England: Addison-Wesley

12. Nielsen, J. (1992) Finding usability problems through heuristic evaluation. *Proceedings ACM CHI'92 Conference* (Monterey, CA, May 3-7), 373-380.

13. Poynton, Charles. (2003) *Digital Video and HDTV*. San Francisco: Morgan Kaufmann Publishers

14. Woodward, Matt. 20 March 2003, *Guide to capturing, cleaning, and compressing video*.<http://arstechnica.com/guides/tweaks/vidcap.ars> Recent access: Feb 2005

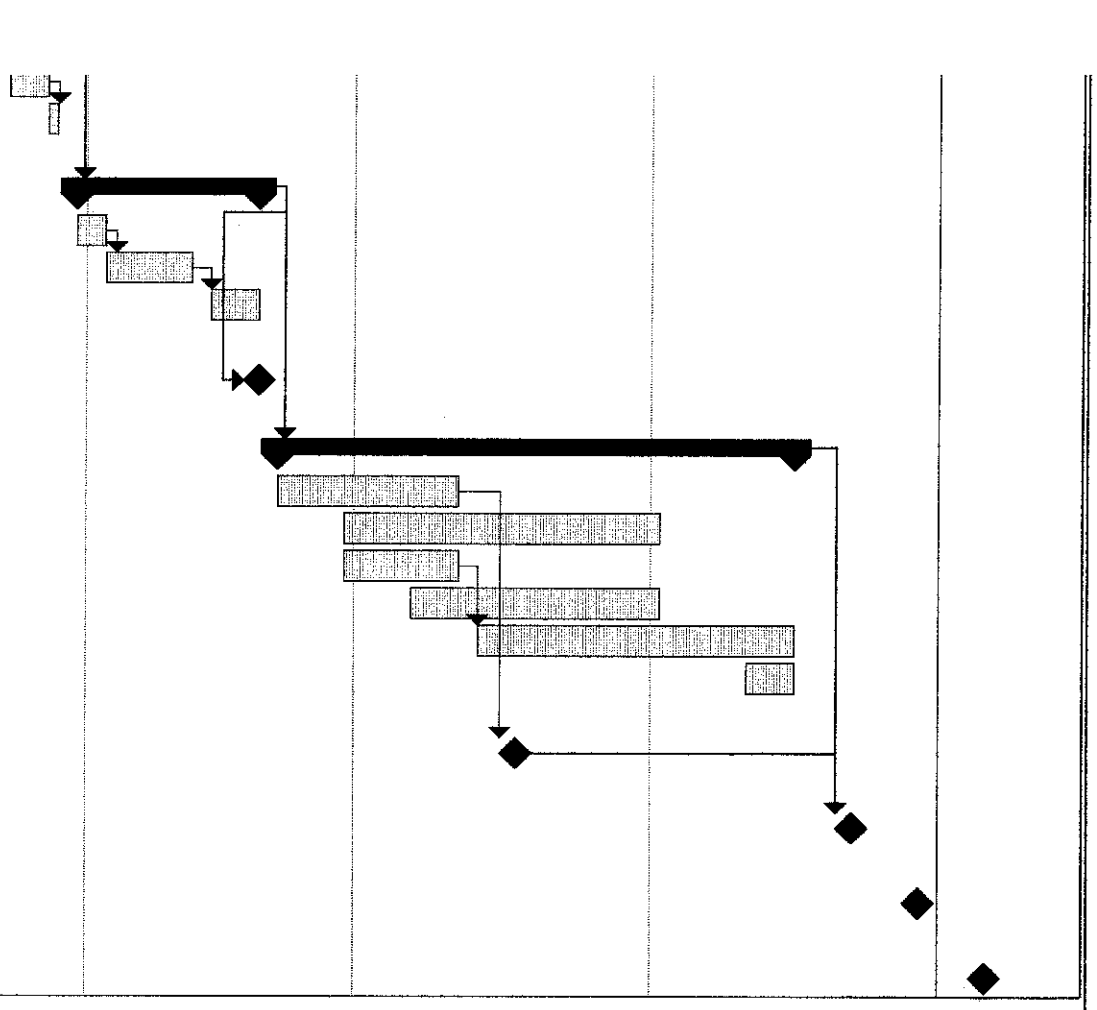
15. Prince, Sal. 2004, *Your Guide To Digital Video Recording* <http://dvr.about.com/od/tvcapturemethods/> Recent access: Feb 2005

APPENDICES

APPENDIX A: Project Schedule
APPENDIX B: Application Screenshot
APPENDIX C: Heuristic Evaluation Form
APPENDIX D: Source Code

APPENDIX A: Project Schedule

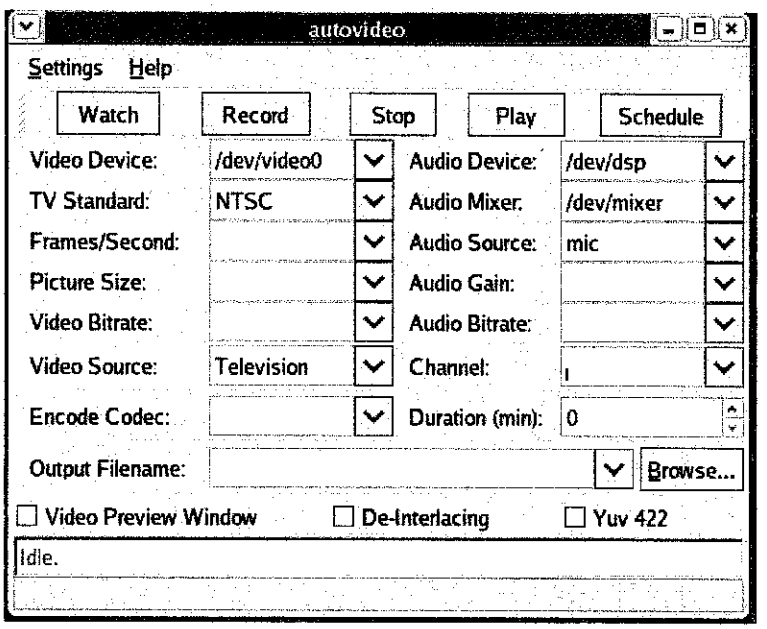
2		Topic Proposal	4 days
3		Topic Assignment	1 day
4			
5		**Preliminary Research/Design Work**	**15 days**
6		Determine project concept, objectives, and scope	3 days
7		Literature Studies	7 days
8		Project Planning	5 days
9			
10		Submission of Preliminary Report	0 days
11			
12		**Project Work**	**40 days**
13		Conducting Studies	3 wks
14		Defining Requirement	5 wks
15		Tools Setting	2 wks
16		Product Design	4 wks
17		Product Developmet	5 wks
18		Product Test	1 wk
19			
20		Submission of Progress Report	0 days
21			
22		Submission of Dissertation Final Draft	0 days
23			
24		Oral Presentation	0 days
25			
26		Submission of Project Dissertation	0 days

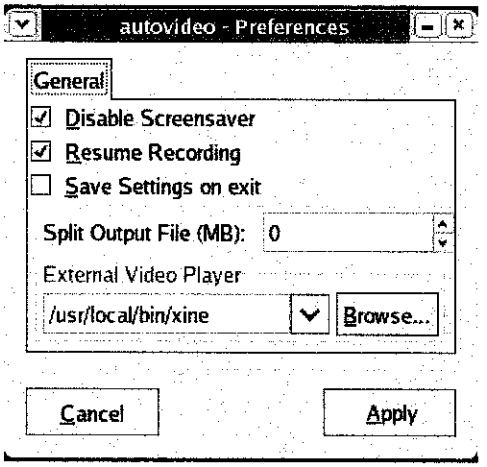


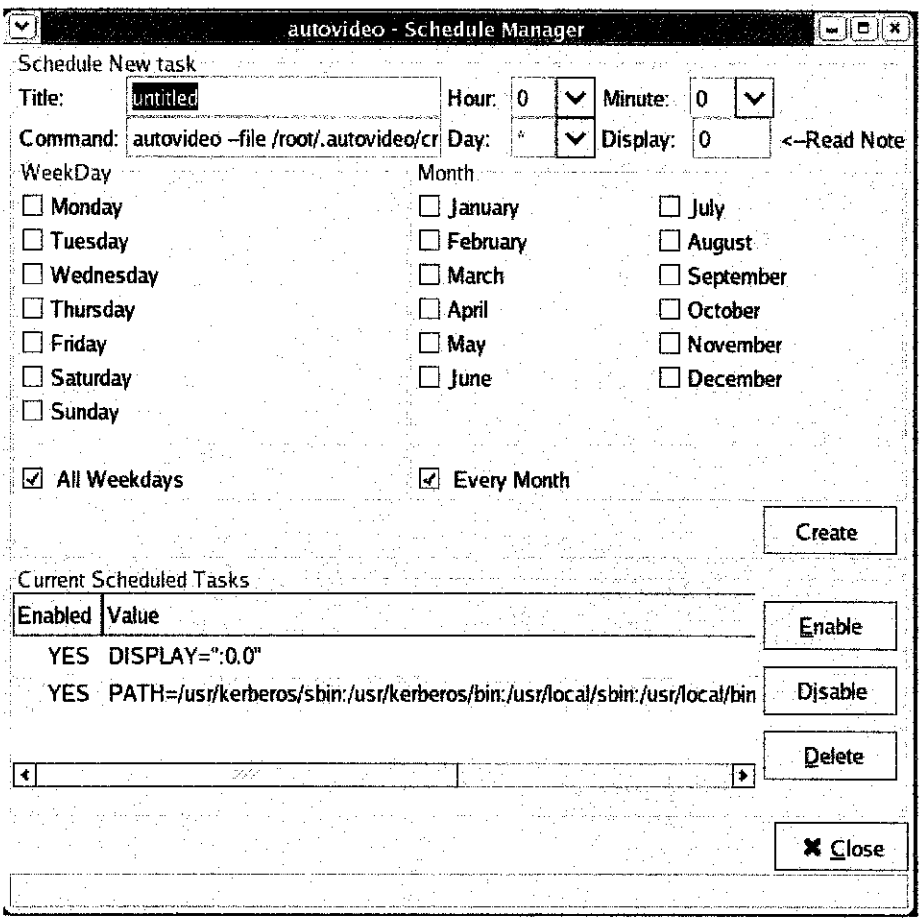
Project: Final Year Project Schedule
Date: 25 Jan 2005

Task		Rolled Up Task		External Tasks	
Progress		Rolled Up Milestone		Project Summary	
Milestone		Rolled Up Progress		Group By Summary	
Summary		Split		Deadline	

APPENDIX B: Application Screenshot


autovideo
Settings Help
Watch
Record
Stop
Play
Schedule
Video Device:
/dev/video0
Audio Device:
/dev/dsp
TV Standard:
NTSC
Audio Mixer:
/dev/mixer
Frames/Second:
Audio Source:
mic
Picture Size:
Audio Gain:
Video Bitrate:
Audio Bitrate:
Video Source:
Television
Channel:
Encode Codec:
Duration (min):
0
Output Filename:
Browse...
Video Preview Window
De-Interlacing
Yuv 422
Idle.


autovideo - Preferences
General
Disable Screensaver
Resume Recording
Save Settings on exit
Split Output File (MB):
0
External Video Player
/usr/local/bin/xine
Browse...
Cancel
Apply


autovideo - Schedule Manager
Schedule New task
Title:
untitled
Hour:
0
Minute:
0
Command:
autovideo –file /root/.autovideo/cr
Day:
*
Display:
0
<--Read Note
WeekDay
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
All Weekdays
Month
January
February
March
April
May
June
July
August
September
October
November
December
Every Month
Create
Current Scheduled Tasks
Enabled
Value
YES DISPLAY=":0.0"
YES PATH=/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin
Enable
Disable
Delete
Close

APPENDIX C: Heuristic Evaluation Form

APPLICATION USABILITY TESTING
HEURISTIC EVALUATION FORM

Name: _____ Date: _____

Please answer all related questions. Circle the numbered answer.

1) Do you have a problem to understand what the current application status/task is when you use this application?

1	2	3	4	5
Always	Quite Often	Sometime	Once or Twice	Never

Please explain more details about the problem and suggestion to improve it.

_____

_____

2) How many of labels used in this application that you do not understand? _____
**Label is the text used in the GUI object, such as 'Ok' or 'Cancel' on buttons, and 'File' on menu.*

Please state the labels and where it is used (e. g. menu item, button)

_____

3) Do you agree that you have the control on using this application?

1	2	3	4	5
Very Disagree	Disagree	Don't Know	Agree	Very Agree

Why?

_____

4) Do you agree that this application interface design is consistent?

1	2	3	4	5
Very Disagree	Disagree	Don't Know	Agree	Very Agree

Why?

_____

5) Do you agree that this application error message is efficient to prevent problems?

1	2	3	4	5
Very Disagree	Disagree	Don't Know	Agree	Very Agree

Why?

---

6) Do you agree that you could understand on how to use the software easily?

1	2	3	4	5
Very Disagree	Disagree	Don't Know	Agree	Very Agree

Why?

---

7) Do you agree that this application does not have unnecessary information?

1	2	3	4	5
Very Disagree	Disagree	Don't Know	Agree	Very Agree

Why?

---

8) Do you agree that this application helps you to recover from any problem occurred?

1	2	3	4	5
Very Disagree	Disagree	Don't Know	Agree	Very Agree

Why?

APPENDIX D: Source Code

//autovideo.c

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

#include <gnome.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include "autovideo.h"
#include "interface.h"
#include "support.h"
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/videodev.h>


#define VIDEO_MODE_PAL_Nc	3
#define VIDEO_MODE_PAL_M	4
#define VIDEO_MODE_PAL_N	5
#define VIDEO_MODE_NTSC_JP	6
#define VIDEO_MODE_PAL_60	7


extern GtkWidget *autovideo;
extern char conf_File[250];
extern gboolean record;
extern GtkWidget *pref_Window;

gchar arecVal[250] = "";
gint visible[3] = { 0, 0, 0 };
gchar v4lInfo[9][255];
gint screenSaver = 0;
gint resume = 1;
gint saveExit = 0;
gchar temp[250] = "", temp1[10];


void on_autovideo_show (GtkWidget * widget, gpointer user_data)
{

	FILE *inFile;
	FILE *outFile;

	gchar home[250];		// = g_get_home_dir();
	gchar xawtv[250];
	gchar player[250] = "";
	gchar split_val[250] = "0";
	gchar rec_dur[250] = "";

	gchar conf_line[250] = "";
	gchar *pos;

	gchar vdev_val[250] = "", temp[250] = "";
	gchar vid_val[9][250], aud_val[10][250], mix_val[10][250];
	gchar adev_val[250] = "";
	gchar p_size[250] = "";
	gchar fps_val[250] = "";
	gchar cod_val[250] = "";
	gchar src_val[250] = "default_unknown";
	gchar chan_val[250] = "";
	gchar file_val[250] = "";
	gchar vbit_val[250] = "";
	gchar abit_val[250] = "";
```

```
        gchar agn_val[250] = "";
        gchar amix_val[250] = "/dev/mixer";
        gchar norm_val[10] = "NTSC";
        gint deinter;
        gint prev_val;
        gint yuv_val;
        gint i = 0;


        GtkWidget *vdev = lookup_widget (GTK_WIDGET (widget), "vdev_val");
        GtkWidget *adev = lookup_widget (GTK_WIDGET (widget), "adev_val");
        GtkWidget *size = lookup_widget (GTK_WIDGET (widget), "p_size");
        GtkWidget *fps = lookup_widget (GTK_WIDGET (widget), "fps_val");
        GtkWidget *cod = lookup_widget (GTK_WIDGET (widget), "combo_entry1");
        GtkWidget *src = lookup_widget (GTK_WIDGET (widget), "src_val");
        GtkWidget *chan = lookup_widget (GTK_WIDGET (widget), "chan_val");
        GtkWidget *file = lookup_widget (GTK_WIDGET (widget), "file_val");
        GtkWidget *vbit = lookup_widget (GTK_WIDGET (widget), "vbit_val");
        GtkWidget *abit = lookup_widget (GTK_WIDGET (widget), "abit_val");
        GtkWidget *agn = lookup_widget (GTK_WIDGET (widget), "agn_val");
        GtkWidget *prev = lookup_widget (GTK_WIDGET (widget), "prev_val");
        GtkWidget *inter = lookup_widget (GTK_WIDGET (widget), "de_inter");
        GtkWidget *yuv = lookup_widget (GTK_WIDGET (widget), "yuv_val");
        GtkWidget *rec = lookup_widget (GTK_WIDGET (widget), "recTime");
        GtkWidget *status = lookup_widget (GTK_WIDGET (widget), "status");
        GtkWidget *norm = lookup_widget (GTK_WIDGET (widget), "norm_val");
        GtkWidget *amix = lookup_widget (GTK_WIDGET (widget), "amix_val");

        GtkWidget *vdev_cbo = lookup_widget (GTK_WIDGET (widget), "vdev_cbo");
        GtkWidget *adev_cbo = lookup_widget (GTK_WIDGET (widget), "adev_cbo");
        GtkWidget *amix_cbo = lookup_widget (GTK_WIDGET (widget), "amix_cbo");
        GtkWidget *chan_cbo = lookup_widget (GTK_WIDGET (widget), "chan_cbo");

        GtkWidget *split = lookup_widget (GTK_WIDGET (pref_Window), "split_val");
        GtkWidget *vid_player = lookup_widget (GTK_WIDGET (pref_Window), "vid_player");

        GList *vid_lst = NULL, *aud_lst = NULL, *mix_lst = NULL;

        GtkStyle *style = gtk_style_new ();

        GdkColor green = { 0, 0, 35335, 0 };


        strcpy (home, g_get_home_dir ());
        system ("rm -rf /tmp/player-autovideo");


        strcpy (xawtv, home);
        strcat (xawtv, "/.xawtv");
        if ((inFile = fopen (xawtv, "r")) == NULL)
        {
                GtkWidget *no_xawtv;
                GList *chan_lst = NULL;
                chan_lst = g_list_append (chan_lst, "0\0");
                gtk_combo_set_popdown_strings (GTK_COMBO (chan_cbo), chan_lst);
                g_list_free (chan_lst);

                no_xawtv = create_no_xawtv ();
                gtk_widget_show (no_xawtv);
                gtk_window_set_transient_for (GTK_WINDOW (no_xawtv), GTK_WINDOW
(autovideo));
        }
        else
        {
                gchar ch_val[1000][100], ch[1];
                GList *chan_lst = NULL;
                gint i = 0;

                strcpy (ch_val[i], "0\0");
                chan_lst = g_list_append (chan_lst, ch_val[i]);
                i++;
```

```
		while (i != 1000)
		{
			fgets (conf_line, 250, inFile);
			if (feof (inFile))
				break;

			strncpy (ch, conf_line, 1);
			if (strchr (conf_line, '[') && (!strchr (ch, '#')) &&
				(!strstr (conf_line, "[global]")) && (!strstr (conf_line,
"[defaults]")) &&
				(!strstr (conf_line, "[launch]")) && (!strstr (conf_line,
"[eventmap]")))
			{
				pos = strrchr (conf_line, ']');
				strncpy (pos, "\0", 1);
				pos = strstr (conf_line, "[") + 1;
				sprintf (ch_val[i], "%s", pos);

				chan_lst = g_list_append (chan_lst, ch_val[i]);
				i++;
				strncpy (ch_val[i], "\0", 1);
			}
		}
		fclose (inFile);
		gtk_combo_set_popdown_strings (GTK_COMBO (chan_cbo), chan_lst);
		g_list_free (chan_lst);
	}


	chdir (home);
	mkdir (".autovideo2", S_IRUSR | S_IWUSR | S_IXUSR);
	mkdir (".autovideo2/cron", S_IRUSR | S_IWUSR | S_IXUSR);
	mkdir (".autovideo2/tmp", S_IRUSR | S_IWUSR | S_IXUSR);
	mkdir ("videos", S_IRUSR | S_IWUSR | S_IXUSR);


	if (fopen (".autovideo2/config", "r") == NULL && fopen (".autovideo", "r") !=
NULL)
		system ("xterm -e \"echo 'autovideo will start after the
convertion.';autovideo-convert.sh\"");

	if (fopen (conf_File, "r") != NULL)
	{
		inFile = fopen (conf_File, "r");
		// fscanf (inFile, "%s%s%s%s%s%s%s%s%d%s%s%s", vdev_val, adev_val,
p_size, fps_val, cod_val, vbit_val, abit_val, agn_val, chan_val, file_val, &prev_val,
player, deinter, rec_dur);
		while (feof (inFile) == 0)
		{
			int i = 0;

			fgets (conf_line, 250, inFile);
			if (feof (inFile))
				break;
			if (strstr (conf_line, "="))
			{
				pos = strstr (conf_line, "\n");
				strncpy (pos, "\0", 1);
				pos = strstr (conf_line, "=") + 1;
				while (conf_line[i] && i < (strlen (conf_line) - strlen
(pos)))
				{
					conf_line[i] = tolower (conf_line[i]);
					i++;
				}
			}

			if (strstr (conf_line, "videodev="))
				strcpy (vdev_val, pos);
			if (strstr (conf_line, "audiodev="))
				strcpy (adev_val, pos);
```

```
                if (strstr (conf_line, "picsize="))
                        strcpy (p_size, pos);
                if (strstr (conf_line, "fps="))
                        strcpy (fps_val, pos);
                if (strstr (conf_line, "codec="))
                        strcpy (cod_val, pos);
                if (strstr (conf_line, "vbitrate="))
                        strcpy (vbit_val, pos);
                if (strstr (conf_line, "abitrate="))
                        strcpy (abit_val, pos);
                if (strstr (conf_line, "again="))
                        strcpy (agn_val, pos);
                if (strstr (conf_line, "source="))
                        strcpy (src_val, pos);
                if (strstr (conf_line, "channel="))
                        strcpy (chan_val, pos);
                if (strstr (conf_line, "file="))
                        strcpy (file_val, pos);
                if (strstr (conf_line, "preview="))
                        prev_val = atoi (pos);
                if (strstr (conf_line, "player="))
                        strcpy (player, pos);
                if (strstr (conf_line, "deinter="))
                        deinter = atoi (pos);
                if (strstr (conf_line, "yuv="))
                        yuv_val = atoi (pos);
                if (strstr (conf_line, "duration="))
                        strcpy (rec_dur, pos);
                if (strstr (conf_line, "standard="))
                        strcpy (norm_val, pos);
                if (strstr (conf_line, "mixer="))
                        strcpy (amix_val, pos);
                if (strstr (conf_line, "audiosrc="))
                        strcpy (arecVal, pos);
                if (strstr (conf_line, "screensaver"))
                        screenSaver = atoi (pos);
                if (strstr (conf_line, "resume"))
                        resume = atoi (pos);
                if (strstr (conf_line, "saveonexit"))
                        saveExit = atoi (pos);
                if (strstr (conf_line, "split="))
                        strcpy (split_val, pos);
        }
        fclose (inFile);

}
else
{
        //strcpy (vdev_val, "/dev/video0");
        strcpy (adev_val, "/dev/dsp");
        strcpy (p_size, "384x288");
        strcpy (fps_val, "29.970");
        strcpy (cod_val, "xvid");
        strcpy (vbit_val, "1800,250,100");
        strcpy (abit_val, "96,0,5");
        strcpy (agn_val, "5");
        //strcpy (src_val, "0");
        strcpy (chan_val, "0");
        strcpy (norm_val, "NTSC");

        strcpy (file_val, home);
        strcat (file_val, "/videos/video.avi");

        prev_val = 1;
        deinter = 0;
        yuv_val = 0;
        //system ("echo none > /tmp/player-autovideo");
}


gtk_entry_set_text (GTK_ENTRY (vdev), "");
```

```
for (i = 0; i < 9; i++)
{
        sprintf (vid_val[i], "/dev/video%d", i);
        if ((inFile = fopen (vid_val[i], "r")) != NULL)
        {
                vid_lst = g_list_append (vid_lst, vid_val[i]);
                fclose (inFile);
                get_v4l_info (vid_val[i], i);
        }
}
gtk_combo_set_popdown_strings (GTK_COMBO (vdev_cbo), vid_lst);
if (strcmp (vdev_val, "") != 0)
        gtk_entry_set_text (GTK_ENTRY (vdev), vdev_val);

if ((inFile = fopen ("/dev/dsp", "r")) != NULL)
{
        aud_lst = g_list_append (aud_lst, "/dev/dsp");
        fclose (inFile);
}
for (i = 0; i < 10; i++)
{
        sprintf (aud_val[i], "/dev/dsp%d", i);
        if ((inFile = fopen (aud_val[i], "r")) != NULL)
        {
                aud_lst = g_list_append (aud_lst, aud_val[i]);
                fclose (inFile);
        }
}
gtk_combo_set_popdown_strings (GTK_COMBO (adev_cbo), aud_lst);
if (strcmp (adev_val, "") != 0)
        gtk_entry_set_text (GTK_ENTRY (adev), adev_val);


if ((inFile = fopen ("/dev/mixer", "r")) != NULL)
{
        mix_lst = g_list_append (mix_lst, "/dev/mixer");
        fclose (inFile);
}
for (i = 0; i < 10; i++)
{
        sprintf (mix_val[i], "/dev/mixer%d", i);
        if ((inFile = fopen (mix_val[i], "r")) != NULL)
        {
                mix_lst = g_list_append (mix_lst, mix_val[i]);
                fclose (inFile);
        }
}
gtk_combo_set_popdown_strings (GTK_COMBO (amix_cbo), mix_lst);
gtk_entry_set_text (GTK_ENTRY (amix), amix_val);

on_amix_val_changed ((GtkEditable *) widget, widget);

gtk_entry_set_text (GTK_ENTRY (size), p_size);
gtk_entry_set_text (GTK_ENTRY (fps), fps_val);
gtk_entry_set_text (GTK_ENTRY (cod), cod_val);

for (i = 0; i <= 9; i++)
{
        sprintf (temp, "=%s\0", src_val);
        if (strstr (v4lInfo[i], vdev_val) && strstr (v4lInfo[i], temp))
                gtk_entry_set_text (GTK_ENTRY (src), src_val);
}

gtk_entry_set_text (GTK_ENTRY (chan), chan_val);
gtk_entry_set_text (GTK_ENTRY (file), file_val);
gtk_entry_set_text (GTK_ENTRY (vbit), vbit_val);
gtk_entry_set_text (GTK_ENTRY (abit), abit_val);
gtk_entry_set_text (GTK_ENTRY (agn), agn_val);
gtk_entry_set_text (GTK_ENTRY (norm), norm_val);
if (deinter == 1)
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (inter), TRUE);
```

```
        else
                gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (inter), FALSE);

        if (!strcmp (rec_dur, "\0"))
                strcpy (rec_dur, "0\0");
        gtk_entry_set_text (GTK_ENTRY (rec), rec_dur);

        if (prev_val == 1)
                gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (prev), TRUE);
        else
                gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (prev), FALSE);
        if (yuv_val == 1)
                gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (yuv), TRUE);
        else
                gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (yuv), FALSE);
        if (screenSaver == 0)
                system ("xscreensaver-command -exit >/dev/null 2>&1 &");

        gtk_entry_set_text (GTK_ENTRY (split), split_val);
        gtk_entry_set_text (GTK_ENTRY (vid_player), player);

        //set Idle status
        if (fopen ("/tmp/trr", "r") == NULL)
        {
                style->text[GTK_STATE_NORMAL] = green;
                // style->font = gdk_font_load("10x20");
                gtk_entry_set_text (GTK_ENTRY (status), "Idle.");
                gtk_widget_set_style (GTK_WIDGET (status), style);
        }

        if (record == 1)
        {
                on_start_clicked_event ((GtkButton *) widget, widget);
                record = 0;
        }
        g_list_free (vid_lst);
        g_list_free (aud_lst);
        g_list_free (mix_lst);
}




void on_norm_val_changed (GtkEditable * editable, gpointer user_data)
{
        gchar norm_val[250];
        //gchar fps_val[250];
        GtkWidget *norm = lookup_widget (GTK_WIDGET (editable), "norm_val");
        GtkWidget *fps = lookup_widget (GTK_WIDGET (editable), "fps_val");
        strncpy (norm_val, gtk_entry_get_text (GTK_ENTRY (norm)), 250);
        //strncpy (fps_val, gtk_entry_get_text (GTK_ENTRY (fps)), 250);

        //set the fps value
        if (strcmp (norm_val, "NTSC") == 0 || strcmp (norm_val, "NTSC-JP") == 0)
                gtk_entry_set_text (GTK_ENTRY (fps), "29.970");
        else if (strstr (norm_val, "PAL") || strstr (norm_val, "SECAM"))
                gtk_entry_set_text (GTK_ENTRY (fps), "25");
}



void on_amix_val_changed (GtkEditable * editable, gpointer user_data)
{
        #include <stdio.h>
        #include <string.h>
        #include <fcntl.h>
        #include <sys/ioctl.h>
        #include <sys/soundcard.h>
```

```
        gchar amix_val[250] = "";
        GList *arec_lst = NULL;

        char *dname[SOUND_MIXER_NRDEVICES] = SOUND_DEVICE_NAMES;
        int recmask = 0, recsrc = 0;
        int mixer_fd;
        int i = 0;

        GtkWidget *arec = lookup_widget (GTK_WIDGET (editable), "arecVal");
        GtkWidget *amix = lookup_widget (GTK_WIDGET (editable), "amix_val");
        GtkWidget *arec_cbo = lookup_widget (GTK_WIDGET (editable), "arec_cbo");
        strncpy (amix_val, gtk_entry_get_text (GTK_ENTRY (amix)), 250);

        if ((mixer_fd = open (amix_val, O_RDWR)) < 0)
        {
                arec_lst = g_list_append (arec_lst, "");
                gtk_combo_set_popdown_strings (GTK_COMBO (arec_cbo), arec_lst);
        }
        else
        {
                if (ioctl (mixer_fd, SOUND_MIXER_READ_RECMASK, &recmask) == -1)
                {
                        arec_lst = g_list_append (arec_lst, "");
                        gtk_combo_set_popdown_strings (GTK_COMBO (arec_cbo), arec_lst);
                }
                else
                {
                        ioctl (mixer_fd, SOUND_MIXER_READ_RECMASK, &recmask);
                        for (i = 0; i < SOUND_MIXER_NRDEVICES; i++)
                                if ((1 << i) & recmask)
                                        arec_lst = g_list_append (arec_lst, dname[i]);
                        gtk_combo_set_popdown_strings (GTK_COMBO (arec_cbo), arec_lst);
                        if (strcmp (arecVal, "") != 0)
                                gtk_entry_set_text (GTK_ENTRY (arec), arecVal);
                        else
                        {
                                ioctl (mixer_fd, SOUND_MIXER_READ_RECSRC, &recsrc);
                                for (i = 0; i < SOUND_MIXER_NRDEVICES; i++)
                                if ((1 << i) & recsrc)
                                        gtk_entry_set_text (GTK_ENTRY (arec), dname[i]);
                        }
                }
                close (mixer_fd);
        }

        g_list_free (arec_lst);
}


void rec_src_set (GtkEditable * editable, gpointer user_data)
{
        #include <stdio.h>
        #include <string.h>
        #include <fcntl.h>
        #include <sys/ioctl.h>
        #include <sys/soundcard.h>

        gchar amix_val[250] = "";

        char *dname[SOUND_MIXER_NRDEVICES] = SOUND_DEVICE_NAMES;
        int recmask = 0, recsrc = 0;
        int igain = -1;
        int mixer_fd;
        int i = 0;

        GtkWidget *arec = lookup_widget (GTK_WIDGET (editable), "arecVal");
        GtkWidget *amix = lookup_widget (GTK_WIDGET (editable), "amix_val");

        strncpy (amix_val, gtk_entry_get_text (GTK_ENTRY (amix)), 250);
        strncpy (arecVal, gtk_entry_get_text (GTK_ENTRY (arec)), 250);
```

```
        if ((mixer_fd = open (amix_val, O_RDWR)) < 0)
                strcpy (amix_val, "");
        else
        {
                ioctl (mixer_fd, SOUND_MIXER_READ_RECMASK, &recmask);
                ioctl (mixer_fd, SOUND_MIXER_READ_RECSRC, &recsrc);
                ioctl (mixer_fd, SOUND_MIXER_READ_IGAIN, &igain);
                for (i = 0; i < SOUND_MIXER_NRDEVICES && strcmp (dname[i], arecVal); i++)
                {
                        if ((1 << (i + 1)) & recmask)
                        {
                                recsrc = (1 << (i + 1));
                                ioctl (mixer_fd, SOUND_MIXER_WRITE_RECSRC, &recsrc);
                        }
                        if (igain >= 0 && igain < 257)
                        {
                                igain = 257;
                                ioctl (mixer_fd, SOUND_MIXER_WRITE_IGAIN, &igain);
                        }
                }
                close (mixer_fd);
        }
}



gboolean on_combo_entry1_focus_out_event (GtkWidget * widget,

        GdkEventFocus * event, gpointer user_data)
{
      gchar file_val[250], tmp_file[250], cod_val[250], *noext, *ext, *pos, notilde[250],
extn[5];
        gint i;
        gchar home[250];
        FILE *inFile;

        GtkWidget *file = lookup_widget (GTK_WIDGET (widget), "file_val");
        GtkWidget *cod = lookup_widget (GTK_WIDGET (widget), "combo_entry1");
        strncpy (cod_val, gtk_entry_get_text (GTK_ENTRY (cod)), 250);
        strncpy (file_val, gtk_entry_get_text (GTK_ENTRY (file)), 250);

        strcpy (home, g_get_home_dir ());

        if (strchr (file_val, '~'))
        {
                strcpy (notilde, file_val);
                strcpy (file_val, home);
                strcat (file_val, "/");
                noext = strstr (notilde, "~");
                strncpy (noext, ".", 1);
                strcat (file_val, noext);
                gtk_entry_set_text (GTK_ENTRY (file), file_val);
        }


        if (strlen (file_val) == 0)
        {
                strcpy (file_val, home);
                strcat (file_val, "/videos/untitled");
        }

        if (strrchr (file_val, '.') == NULL)
        {
                strcat (file_val, ".avi");
                gtk_entry_set_text (GTK_ENTRY (file), file_val);
        }
```

```
if (strstr (file_val, ".avi") == NULL && strstr (file_val, ".mpg") == NULL)
{
        strcat (file_val, ".avi");
        gtk_entry_set_text (GTK_ENTRY (file), file_val);
}


ext = strrchr (file_val, '/');
ext = strrchr (file_val, '.');


if (strlen (ext) != 4)
{

        if (strlen (ext) == 0)
                strcat (file_val, "untitled");
        strcat (file_val, ".avi");
        gtk_entry_set_text (GTK_ENTRY (file), file_val);
        ext = strrchr (file_val, '/');
        ext = strrchr (file_val, '.');
}



if (strcmp (cod_val, "mpeg1") != 0 && strcmp (ext, ".mpg") == 0)
{
        noext = strstr (file_val, ".mpg");
        strncpy (noext, ".avi", 4);
        gtk_entry_set_text (GTK_ENTRY (file), file_val);
        //return 0;
}

else if (strcmp (cod_val, "mpeg1") != 0 && strcmp (ext, ".avi") == 0);


else if (strcmp (cod_val, "mpeg1") != 0)
{
        strncpy (file_val, ".avi", 4);
        gtk_entry_set_text (GTK_ENTRY (file), file_val);
        //return 0;
}

else if (strcmp (cod_val, "mpeg1") == 0 && strcmp (ext, ".avi") == 0)
{
        noext = strstr (file_val, ".avi");
        strncpy (noext, ".mpg", 4);
        gtk_entry_set_text (GTK_ENTRY (file), file_val);
        //return 0;
}

else if (strcmp (cod_val, "mpeg1") == 0 && strcmp (ext, ".mpg") == 0);

else if (strcmp (ext, ".mpg") == 0)
{
        noext = strstr (file_val, ".mpg");
        strncpy (noext, ".avi", 4);
        gtk_entry_set_text (GTK_ENTRY (file), file_val);
        //return 0;
}


if (resume == 0)
{
        if ((inFile = fopen (file_val, "r")))
        {
                fclose (inFile);
                strcpy (notilde, file_val);
                pos = strrchr (notilde, '.');
                strcpy (extn, pos);
                extn[4] = '\0';
                if (strrchr (notilde, '-') && isdigit (*(pos - 1)))
```

```
                        pos = strrchr (notilde, '-');
                strncpy (pos, "\0", 1);
                for (i = 1; i < 10000; i++)
                {
                        sprintf (tmp_file, "%s-%d%s\0", notilde, i, extn);

                        if ((inFile = fopen (tmp_file, "r")) == NULL)
                        {
                                strcpy (file_val, tmp_file);
                                i = 10000;
                        }
                        else
                                fclose (inFile);
                }
                gtk_entry_set_text (GTK_ENTRY (file), file_val);
                return 0;
        }
    }

    return FALSE;
}



void on_play_clicked (GtkButton * button, gpointer user_data)
{

    FILE *inFile;
    gchar file_val[250];
    gchar player[250];
    gchar command[250];

    GtkWidget *file = lookup_widget (GTK_WIDGET (button), "file_val");
    GtkWidget *vid_player = lookup_widget (GTK_WIDGET (pref_Window), "vid_player");
    strncpy (file_val, gtk_entry_get_text (GTK_ENTRY (file)), 250);
    strncpy (player, gtk_entry_get_text (GTK_ENTRY (vid_player)), 250);

    /*inFile = fopen ("/tmp/player-autovideo", "r");
    fscanf (inFile, "%s", player);
    fclose (inFile);
    */

//  if (strcmp (player, "none") != 0 || fopen (player, "r") != NULL)
    if (strlen (player))
    {
            strcpy (command, player);
            strcat (command, " \"");
            strcat (command, file_val);
            strcat (command, "\"&");
            system (command);
    }
    else
    {
            GtkWidget *noplay;
            noplay = create_noplay ();
            gtk_widget_show (noplay);
            gtk_window_set_transient_for (GTK_WINDOW (noplay), GTK_WINDOW
(autovideo));
            //exit;
    }
}


void on_play_activate (GtkMenuItem * menuitem, gpointer user_data)
{
    on_play_clicked ((GtkButton *) menuitem, menuitem);
}
```

```
void on_select_player_activate (GtkMenuItem * menuitem, gpointer user_data)
{
        on_preferences_activate ((GtkMenuItem *)menuitem, menuitem);
}



void on_save_settings1_activate (GtkMenuItem * menuitem, gpointer user_data)
{

        FILE *outFile;
        FILE *inFile;
        //gchar player[250] = "none";

        gchar vdev_val[250];
        gchar adev_val[250];
        gchar p_size[250];
        gchar fps_val[250];
        gchar src_val[250];
        gchar cod_val[250];
        gchar chan_val[250];
        gchar file_val[250];
        gchar vbit_val[250];
        gchar abit_val[250];
        gchar agn_val[250];
        gchar norm_val[10];
        gchar amix_val[250];
        gboolean prev_val;
        gchar rec_dur[250];
        gboolean deinter;
        gboolean yuv_val;


        GtkWidget *vdev = lookup_widget (GTK_WIDGET (menuitem), "vdev_val");
        GtkWidget *adev = lookup_widget (GTK_WIDGET (menuitem), "adev_val");
        GtkWidget *size = lookup_widget (GTK_WIDGET (menuitem), "p_size");
        GtkWidget *fps = lookup_widget (GTK_WIDGET (menuitem), "fps_val");
        GtkWidget *cod = lookup_widget (GTK_WIDGET (menuitem), "combo_entry1");
        GtkWidget *src = lookup_widget (GTK_WIDGET (menuitem), "src_val");
        GtkWidget *chan = lookup_widget (GTK_WIDGET (menuitem), "chan_val");
        GtkWidget *file = lookup_widget (GTK_WIDGET (menuitem), "file_val");
        GtkWidget *vbit = lookup_widget (GTK_WIDGET (menuitem), "vbit_val");
        GtkWidget *abit = lookup_widget (GTK_WIDGET (menuitem), "abit_val");
        GtkWidget *agn = lookup_widget (GTK_WIDGET (menuitem), "agn_val");
        GtkWidget *prev = lookup_widget (GTK_WIDGET (menuitem), "prev_val");
        GtkWidget *rec = lookup_widget (GTK_WIDGET (menuitem), "recTime");
        GtkWidget *inter = lookup_widget (GTK_WIDGET (menuitem), "de_inter");
        GtkWidget *yuv = lookup_widget (GTK_WIDGET (menuitem), "yuv_val");
        GtkWidget *norm = lookup_widget (GTK_WIDGET (menuitem), "norm_val");
        GtkWidget *amix = lookup_widget (GTK_WIDGET (menuitem), "amix_val");
        GtkWidget *arec = lookup_widget (GTK_WIDGET (menuitem), "arecVal");


        strncpy (vdev_val, gtk_entry_get_text (GTK_ENTRY (vdev)), 250);
        strncpy (adev_val, gtk_entry_get_text (GTK_ENTRY (adev)), 250);
        strncpy (p_size, gtk_entry_get_text (GTK_ENTRY (size)), 250);
        strncpy (fps_val, gtk_entry_get_text (GTK_ENTRY (fps)), 250);
        strncpy (cod_val, gtk_entry_get_text (GTK_ENTRY (cod)), 250);
        strncpy (src_val, gtk_entry_get_text (GTK_ENTRY (src)), 250);
        strncpy (chan_val, gtk_entry_get_text (GTK_ENTRY (chan)), 250);
        strncpy (file_val, gtk_entry_get_text (GTK_ENTRY (file)), 250);
        strncpy (vbit_val, gtk_entry_get_text (GTK_ENTRY (vbit)), 250);
        strncpy (abit_val, gtk_entry_get_text (GTK_ENTRY (abit)), 250);
        strncpy (agn_val, gtk_entry_get_text (GTK_ENTRY (agn)), 250);
        strncpy (rec_dur, gtk_entry_get_text (GTK_ENTRY (rec)), 250);
        strncpy (norm_val, gtk_entry_get_text (GTK_ENTRY (norm)), 10);
        strncpy (amix_val, gtk_entry_get_text (GTK_ENTRY (amix)), 250);
        strncpy (arecVal, gtk_entry_get_text (GTK_ENTRY (arec)), 250);
        deinter = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (inter));
        prev_val = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (prev));
```

```
        yuv_val = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (yuv));


        outFile = fopen (conf_File, "w");
        fprintf (outFile,

"[Default=autovideo]\n\nVideoDev=%s\nAudioDev=%s\nPicSize=%s\nFPS=%s\nCodec=%s\nVBitrate=
%s\nABitrate=%s\nAGain=%s\nSource=%s\nChannel=%s\nFile=%s\nPreview=%s\nDeInter=%s\nDurati
on=%s\nYuv=%s\nStandard=%s\nMixer=%s\nAudioSrc=%s\n\nScreenSaver=%d\nResume=%d\nSaveOnExi
t=%d\n\n[End autovideo]\n\n",
                vdev_val, adev_val, p_size, fps_val, cod_val, vbit_val, abit_val,
                agn_val, src_val, chan_val, file_val, prev_val == TRUE ? "1" : "0",
                deinter == TRUE ? "1" : "0", rec_dur,
                yuv_val == TRUE ? "1" : "0", norm_val, amix_val, arecVal,
                screenSaver, resume, saveExit);
        fclose (outFile);
}




void on_revert_to_defaults1_activate (GtkMenuItem * menuitem, gpointer user_data)
{
        char config[250];             //config file

        gchar status_val[250] = "";
        GtkWidget *status = lookup_widget (GTK_WIDGET (menuitem), "status");
        strncpy (status_val, gtk_entry_get_text (GTK_ENTRY (status)), 250);
        if (strcmp (status_val, "Idle.") != 0)
        {
                GtkWidget *unsafe;
                unsafe = create_unsafe ();
                gtk_widget_show (unsafe);
                gtk_window_set_transient_for (GTK_WINDOW (unsafe), GTK_WINDOW
(autovideo));
        }
        else
        {

                strcpy (config, conf_File);

                strcpy (conf_File, "");

                on_autovideo_show ((GtkWidget *) menuitem, menuitem);

                strcpy (conf_File, config);
        }
}



void on_reload_saved_settings1_activate (GtkMenuItem * menuitem,

        gpointer user_data)
{
        gchar status_val[250] = "";
        GtkWidget *status = lookup_widget (GTK_WIDGET (menuitem), "status");
        strncpy (status_val, gtk_entry_get_text (GTK_ENTRY (status)), 250);
        if (strcmp (status_val, "Idle.") != 0)
        {
                GtkWidget *unsafe;
                unsafe = create_unsafe ();
                gtk_widget_show (unsafe);
                gtk_window_set_transient_for (GTK_WINDOW (unsafe), GTK_WINDOW
(autovideo));
        }
        else
                on_autovideo_show ((GtkWidget *) menuitem, menuitem);
```

```
}



void on_about_activate (GtkMenuItem * menuitem, gpointer user_data)
{
        GtkWidget *about;
        about = create_about ();
        gtk_widget_show (about);
}



gboolean
on_autovideo_delete_event (GtkWidget * widget,
                                                GdkEvent * event, gpointer user_data)
{

        system ("killall -2 transcode 2>/dev/null");
        system ("killall -9 autovideo-sleep.sh 2>/dev/null");

        system ("rm -rf /tmp/autovideo*");
        system ("rm -rf /tmp/player-autovideo");
        system ("rm -rf /tmp/trr /tmp/tr-out*");

        system ("rm -rf \"c:\\trace_b.txt\"");

        if (screenSaver == 0)
                system ("xscreensaver -no-splash >/dev/null 2>&1 &");

        if (saveExit == 1)
                on_save_settings1_activate ((GtkMenuItem *) widget, widget);
        //exit
        on_btnClose_clicked ((GtkButton *) widget, widget);
        gtk_exit (0);
        return 1;
}



void on_exit1_activate (GtkMenuItem * menuitem, gpointer user_data)
{

        system ("killall -2 transcode 2>/dev/null");
        system ("killall -9 autovideo-sleep.sh 2>/dev/null");

        system ("rm -rf /tmp/autovideo*");
        system ("rm -rf /tmp/player-autovideo");
        system ("rm -rf /tmp/trr /tmp/tr-out*");

        system ("rm -rf \"c:\\trace_b.txt\"");

        if (screenSaver == 0)
                system ("xscreensaver -no-splash >/dev/null 2>&1 &");

        if (saveExit == 1)
                on_save_settings1_activate ((GtkMenuItem *) menuitem, menuitem);
        //exit
        on_btnClose_clicked ((GtkButton *) menuitem, menuitem);
        gtk_exit (0);
}



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

```
        system ("killall -2 transcode 2>/dev/null");
        system ("killall -9 autovideo-sleep.sh 2>/dev/null");

        system ("rm -rf /tmp/autovideo*");
        system ("rm -rf /tmp/player-autovideo");
        system ("rm -rf /tmp/trr /tmp/tr-out*");

        system ("rm -rf \"c:\\trace_b.txt\"");

        if (screenSaver == 0)
                system ("xscreensaver -no-splash >/dev/null 2>&1 &");
        //exit
        on_btnClose_clicked ((GtkButton *) button, button);
        gtk_exit (1);
}


void on_vdev_val_changed (GtkEditable * editable, gpointer user_data)
{
        struct video_capability cap;
        struct video_channel chan;
        int v4l_fd, i = 0;
        gchar vdev_val[250] = "";
        gchar src_val[10][250];

        GtkWidget *vdev = lookup_widget (GTK_WIDGET (editable), "vdev_val");
        GtkWidget *src = lookup_widget (GTK_WIDGET (editable), "src_val");
        GtkWidget *src_cbo = lookup_widget (GTK_WIDGET (editable), "src_cbo");
        GList *src_lst = NULL, *none = NULL;

        gtk_entry_set_text (GTK_ENTRY (src), "");
        none = g_list_append (none, "");
        gtk_combo_set_popdown_strings (GTK_COMBO (src_cbo), none);

        strncpy (vdev_val, gtk_entry_get_text (GTK_ENTRY (vdev)), 250);
        v4l_fd = open (vdev_val, O_RDONLY);
        ioctl (v4l_fd, VIDIOCGCHAN, &chan);
        ioctl (v4l_fd, VIDIOCGCAP, &cap);

        for (i = 0; i < cap.channels; i++)
        {
                memset (&chan, 0, sizeof (chan));
                chan.channel = i;
                if (-1 == ioctl (v4l_fd, VIDIOCGCHAN, &chan))
                        break;
                sprintf (src_val[i], "%s", &chan.name);
                src_lst = g_list_append (src_lst, src_val[i]);
                gtk_combo_set_popdown_strings (GTK_COMBO (src_cbo), src_lst);
        }
        close (v4l_fd);
        g_list_free (src_lst);
}


void get_v4l_info (char v4l_dev[250], int v4l_id)
{
        struct video_capability cap;
        struct video_channel chan;
        int v4l_fd, i = 0;

        v4l_fd = open (v4l_dev, O_RDONLY);
        ioctl (v4l_fd, VIDIOCGCHAN, &chan);
        ioctl (v4l_fd, VIDIOCGCAP, &cap);

        sprintf (v4lInfo[v4l_id], "%s=%s, ", v4l_dev, &cap.name);
        for (i = 0; i < cap.channels; i++)
        {
                memset (&chan, 0, sizeof (chan));
                chan.channel = i;
```

```
                if (-1 == ioctl (v4l_fd, VIDIOCGCHAN, &chan))
                        break;
                sprintf (v4lInfo[v4l_id], "%s%d=%s, ", v4lInfo[v4l_id], i,
                                &chan.name);
        }
        //printf ("%s\n", v4lInfo[v4l_id]);
        close (v4l_fd);
}


void set_norm (char v4l_dev[250], char norm[20])
{
        struct video_capability cap;
        struct video_channel chan;
        int v4l_fd, i = 0;

        v4l_fd = open (v4l_dev, O_RDONLY);
        ioctl (v4l_fd, VIDIOCGCHAN, &chan);
        ioctl (v4l_fd, VIDIOCGCAP, &cap);

        for (i = 0; i < cap.channels; i++)
        {
                memset (&chan, 0, sizeof (chan));
                chan.channel = i;
                if (strcmp (norm, "NTSC") == 0)
                        chan.norm = VIDEO_MODE_NTSC;
                else if (strcmp (norm, "PAL") == 0)
                        chan.norm = VIDEO_MODE_PAL;
                else if (strcmp (norm, "SECAM") == 0)
                        chan.norm = VIDEO_MODE_SECAM;
                else if (strcmp (norm, "PAL-Nc") == 0)
                        chan.norm = VIDEO_MODE_PAL_Nc;
                else if (strcmp (norm, "PAL-M") == 0)
                        chan.norm = VIDEO_MODE_PAL_M;
                else if (strcmp (norm, "PAL-N") == 0)
                        chan.norm = VIDEO_MODE_PAL_N;
                else if (strcmp (norm, "NTSC-JP") == 0)
                        chan.norm = VIDEO_MODE_NTSC_JP;
                else if (strcmp (norm, "PAL-60") == 0)
                        chan.norm = VIDEO_MODE_PAL_60;
                else if (strcmp (norm, "Current") == 0)
                        break;
                ioctl (v4l_fd, VIDIOCSCHAN, &chan);

                if (-1 == ioctl (v4l_fd, VIDIOCGCHAN, &chan))
                        break;
        }
        close (v4l_fd);
}


void on_preferences_activate (GtkMenuItem * menuitem, gpointer user_data)
{
        GtkWidget *scrsav = lookup_widget (pref_Window, "screenSaver");
        GtkWidget *res = lookup_widget (pref_Window, "resume");
        GtkWidget *sav = lookup_widget (pref_Window, "save_conf");
        GtkWidget *player = lookup_widget (pref_Window, "vid_player");
        GtkWidget *split = lookup_widget (pref_Window, "split_val");
        strcpy (temp, gtk_entry_get_text (GTK_ENTRY (player)));
        strcpy (temp1, gtk_entry_get_text (GTK_ENTRY (split)));
        gtk_widget_show (pref_Window);
        gtk_window_set_transient_for (GTK_WINDOW (pref_Window), GTK_WINDOW (autovideo));
        //gtk_window_set_decorated (gtk_widget_get_root_window (pref_Window), TRUE);

        if (screenSaver == 0)
                gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (scrsav), TRUE);
        else
                gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (scrsav), FALSE);
        if (resume == 1)
                gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (res), TRUE);
        else
```

```
		gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (res), FALSE);
	if (saveExit == 1)
		gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (sav), TRUE);
	else
		gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (sav), FALSE);
}

gboolean on_pref_delete_event (GtkWidget * widget,
                                                        GdkEvent * event, gpointer
user_data)
{
	on_pref_stop_clicked (NULL, NULL);
	//return FALSE;
}


void on_pref_stop_clicked (GtkButton * button, gpointer user_data)
{
	GtkWidget *player = lookup_widget (pref_Window, "vid_player");
	GtkWidget *split = lookup_widget (pref_Window, "split_val");
	gtk_entry_set_text (GTK_ENTRY (player), temp);
	gtk_entry_set_text (GTK_ENTRY (split), temp1);
	temp[0] = '\0';
	temp1[0] = '\0';
	gtk_widget_hide (pref_Window);
}


void on_pref_apply_clicked (GtkButton * button, gpointer user_data)
{
	GtkWidget *scrsav = lookup_widget (pref_Window, "screenSaver");
	GtkWidget *res = lookup_widget (pref_Window, "resume");
	GtkWidget *sav = lookup_widget (pref_Window, "save_conf");
	GtkWidget *split = lookup_widget (pref_Window, "split_val");
	GtkWidget *player = lookup_widget (pref_Window, "vid_player");
	gchar split_val[10] = "";
	gchar player_val[250] = "";
	gboolean scrsav_val = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (scrsav));
	gboolean resume_val = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (res));
	gboolean save_val = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (sav));

	FILE *inFile;
	FILE *outFile;
	gchar conf_File_old[250], conf_line[250];
	gint found[5] = { 0, 0, 0, 0, 0 };

	strcpy (split_val, gtk_entry_get_text (GTK_ENTRY (split)));
	strcpy (player_val, gtk_entry_get_text (GTK_ENTRY (player)));

	if (scrsav_val == TRUE && screenSaver == 1)
		system ("xscreensaver-command -exit >/dev/null 2>&1 &");
	else if (scrsav_val != TRUE && screenSaver == 0)
		system ("xscreensaver -no-splash >/dev/null 2>&1 &");
	screenSaver = (scrsav_val == TRUE ? 0 : 1);
	resume = (resume_val == TRUE ? 1 : 0);
	saveExit = (save_val == TRUE ? 1 : 0);
	sprintf (conf_File_old, "%s.old", conf_File);
	rename (conf_File, conf_File_old);
	inFile = fopen (conf_File_old, "r");
	outFile = fopen (conf_File, "w");
	for (;;)
	{
		int i = 0;

		fgets (conf_line, 250, inFile);
		if (feof (inFile))
			break;
		if (strstr (conf_line, "ScreenSaver="))
		{
			fprintf (outFile, "ScreenSaver=%d\n", screenSaver);
			found[0] = 1;
```

```
		}
		else if (strstr (conf_line, "Resume="))
		{
			fprintf (outFile, "Resume=%d\n", resume);
			found[1] = 1;
		}
		else if (strstr (conf_line, "SaveOnExit="))
		{
			fprintf (outFile, "SaveOnExit=%d\n", saveExit);
			found[2] = 1;
		}
		else if (strstr (conf_line, "Split="))
		{
			fprintf (outFile, "Split=%s\n", split_val);
			found[3] = 1;
		}
		else if (strstr (conf_line, "Player="))
		{
			fprintf (outFile, "Player=%s\n", player_val);
			found[4] = 1;
		}
		else if (strstr (conf_line, "[End autovideo]"))
		{
			if (found[0] == 0)
				fprintf (outFile, "ScreenSaver=%d\n", screenSaver);
			if (found[1] == 0)
				fprintf (outFile, "Resume=%d\n", resume);
			if (found[2] == 0)
				fprintf (outFile, "SaveOnExit=%d\n", saveExit);
			if (found[3] == 0)
				fprintf (outFile, "Split=%s\n", split_val);
			if (found[4] == 0)
				fprintf (outFile, "Player=%s\n\n", player_val);
			fprintf (outFile, "%s", conf_line);
		}
		else
			fprintf (outFile, "%s", conf_line);
	}
	fclose (inFile);
	fclose (outFile);
	unlink (conf_File_old);
	gtk_widget_hide (pref_Window);
	temp[0] = '\0';
	temp1[0] = '\0';
}

void on_chan_val_changed	(GtkEditable *editable,
							gpointer user_data)
{
	GtkWidget *status = lookup_widget (GTK_WIDGET (editable), "status");
	GtkWidget *chan = lookup_widget (GTK_WIDGET (editable), "chan_val");

	if (strstr (gtk_entry_get_text (GTK_ENTRY (status)), "Playing") &&
		strlen (gtk_entry_get_text (GTK_ENTRY (chan))))
		on_watch_click_event ((GtkButton *) editable, editable);
}


void on_src_val_changed		(GtkEditable *editable,
							gpointer user_data)
{
	GtkWidget *src = lookup_widget (GTK_WIDGET (editable), "src_val");
	if (strlen (gtk_entry_get_text (GTK_ENTRY (src))))
		on_chan_val_changed ((GtkEditable *) editable, editable);
}
```

//transcode.c

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

#include <gnome.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "autovideo.h"
#include "interface.h"
#include "support.h"

extern GtkWidget *autovideo;
extern GtkWidget *pref_Window;
extern char conf_File[250];
extern gboolean rec;
extern gint visible[3];
extern gchar v4l_info[9][255];
extern ver0613;
GtkWidget *tr_output;

#define BUFFER_SIZE 8192


void on_watch_click_event (GtkButton * button, gpointer user_data)
{

       FILE *outFile;

       gchar vdev_val[250];
       gchar adev_val[250];
       gchar p_size[250] = "-g ";
       gchar fps_val[250] = "-f ";
       gchar src_val[250], vsrc_val[250], temp[250], *pos;
       gint i = 0;
       gchar chan_val[250];
       gchar norm_val[10];
       gboolean deinter;
       gboolean prev_val;
       gboolean yuv_val;

       GtkWidget *vdev = lookup_widget (GTK_WIDGET (button), "vdev_val");
       GtkWidget *adev = lookup_widget (GTK_WIDGET (button), "adev_val");
       GtkWidget *size = lookup_widget (GTK_WIDGET (button), "p_size");
       GtkWidget *fps = lookup_widget (GTK_WIDGET (button), "fps_val");
       GtkWidget *src = lookup_widget (GTK_WIDGET (button), "src_val");
       GtkWidget *chan = lookup_widget (GTK_WIDGET (button), "chan_val");
       GtkWidget *prev = lookup_widget (GTK_WIDGET (button), "prev_val");
       GtkWidget *inter = lookup_widget (GTK_WIDGET (button), "de_inter");
       GtkWidget *yuv = lookup_widget (GTK_WIDGET (button), "yuv_val");
       GtkWidget *norm = lookup_widget (GTK_WIDGET (button), "norm_val");

       GtkWidget *status = lookup_widget (GTK_WIDGET (button), "status");
       GtkStyle *style = gtk_style_new ();
       GtkStyle *style0 = gtk_style_new ();
       GdkColor blue = { 0, 0, 0, 65335 };
       GdkColor green = { 0, 0, 35335, 0 };

       strncpy (vdev_val, gtk_entry_get_text (GTK_ENTRY (vdev)), 250);
       strncpy (adev_val, gtk_entry_get_text (GTK_ENTRY (adev)), 250);


       strcat (p_size, gtk_entry_get_text (GTK_ENTRY (size)));
       if (strcmp (p_size, "-g ") == 0)
              strcpy (p_size, " ");


       strcat (fps_val, gtk_entry_get_text (GTK_ENTRY (fps)));
```

```
	if (strcmp (fps_val, "-f ") == 0)
		strcpy (fps_val, " ");


	strncpy (vsrc_val, gtk_entry_get_text (GTK_ENTRY (src)), 250);
	strncpy (chan_val, gtk_entry_get_text (GTK_ENTRY (chan)), 250);
	strncpy (norm_val, gtk_entry_get_text (GTK_ENTRY (norm)), 10);
	prev_val = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (prev));
	deinter = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (inter));
	yuv_val = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (yuv));


	if (strlen (vdev_val) == 0)
	{
		GtkWidget *novid;
		novid = create_novid ();
		gtk_widget_show (novid);
		gtk_window_set_transient_for (GTK_WINDOW (novid), GTK_WINDOW (autovideo));
		on_stop_clicked ((GtkButton *) button, button);
		return;
	}

	if (strlen (adev_val) == 0)
	{
		GtkWidget *noaud;
		noaud = create_noaud ();
		gtk_widget_show (noaud);
		gtk_window_set_transient_for (GTK_WINDOW (noaud), GTK_WINDOW (autovideo));
		on_stop_clicked ((GtkButton *) button, button);
		return;
	}

	if (strlen (vsrc_val) == 0)
	{
		GtkWidget *nosrc;
		nosrc = create_nosrc ();
		gtk_widget_show (nosrc);
		gtk_window_set_transient_for (GTK_WINDOW (nosrc), GTK_WINDOW (autovideo));
		on_stop_clicked ((GtkButton *) button, button);
		return;
	}
	for (i = 0; i < 9; i++)
	{
		sprintf (temp, "=%s\0", vsrc_val);
		if (strstr (v4l_info[i], vdev_val) && strstr (v4l_info[i], temp))
		{
			pos = strstr (v4l_info[i], temp) - 1;
			strncpy (src_val, pos, 1);
			src_val[1] = '\0';
		}
	}


	system ("killall -2 transcode 2>/dev/null || echo Done > /tmp/autovideo.temp");
	system ("killall -9 autovideo-sleep.sh 2>/dev/null");


	do
		system ("sleep 0.5 2>/dev/null || sleep 0,5");
	while (fopen ("/tmp/autovideo.temp", "r") == NULL);
	system ("sleep 0.1 2>/dev/null || sleep 0,1");


	system ("rm -rf /tmp/autovideo*");


	outFile = fopen ("/tmp/autovideo.sh", "w");
	fprintf (outFile, "#!/bin/bash\n");
	//	fprintf (outFile, "v4lctl -c %s setnorm %s\n", vdev_val, norm_val); //set
the TV standard (norm)
	if (ver0613 == TRUE)
```

```
                fprintf (outFile,
                        "transcode --progress_off -i %s -p %s -u 100 %s %s --
import_v4l %s,\"%s\" %s %s %s\n",
                        vdev_val, adev_val, p_size, fps_val, src_val, chan_val,
                        yuv_val == TRUE ? "-x v4l=yuv422" : "-x v4l",
                        deinter == TRUE ? "-I 5" : "",
                        prev_val == TRUE ? "-J pv" : "");
        else
                fprintf (outFile,
                        "transcode --progress_off -i %s -p %s -V -u 100 %s %s --
import_v4l %s,\"%s\" %s %s %s\n",
                        vdev_val, adev_val, p_size, fps_val, src_val, chan_val,
                        yuv_val == TRUE ? "-x v4l=yuv422" : "",
                        deinter == TRUE ? "-I 5" : "",
                        prev_val == TRUE ? "-J preview" : "");
        fprintf (outFile, "echo -e '\n\n\'\n");
        fprintf (outFile, "echo Done > /tmp/autovideo.temp\n");
        fclose (outFile);


        set_norm (vdev_val, norm_val);

        system ("chmod +x /tmp/autovideo.sh");
        system ("/tmp/autovideo.sh >>/tmp/tr-output 2>&1 &");
        sleep (1);
        output_show ();


        system ("rm -rf /tmp/trr");

        system ("ps -A|grep transcode >/tmp/trr || rm -rf /tmp/trr");


        style->text[GTK_STATE_NORMAL] = blue;
        // style->font = gdk_font_load("10x20");
        gtk_entry_set_text (GTK_ENTRY (status), "Playing Stream from Device...");
        gtk_widget_set_style (GTK_WIDGET (status), style);


        if (fopen ("/tmp/trr", "r") == NULL)
        {
                style0->text[GTK_STATE_NORMAL] = green;
                // style->font = gdk_font_load("10x20");
                gtk_entry_set_text (GTK_ENTRY (status), "Idle.");
                gtk_widget_set_style (GTK_WIDGET (status), style0);
        }
}


void on_watch_activate_event (GtkMenuItem * menuitem, gpointer user_data)
{
        on_watch_click_event ((GtkButton *) menuitem, menuitem);
}




void on_start_clicked_event (GtkButton * button, gpointer user_data)
{

        FILE *outFile;

        gchar vdev_val[250];
        gchar adev_val[250];
        gchar p_size[250] = "-g ";
        gchar fps_val[250] = "-f ";
        gchar cod_val[250];
        gchar src_val[250], vsrc_val[250], temp[250], *pos;
        gint i = 0;
```

```
gchar chan_val[250];
gchar file_val[250];
gchar vbit_val[250] = "-w ";
gchar abit_val[250] = "-b ";
gchar agn_val[250] = "-s ";
gchar norm_val[10];
gboolean deinter;
gchar rec_dur[250];
int duration;
gboolean prev_val;
gboolean yuv_val;

gchar split_val[20] = "--avi_limit ";


GtkWidget *vdev = lookup_widget (GTK_WIDGET (button), "vdev_val");
GtkWidget *adev = lookup_widget (GTK_WIDGET (button), "adev_val");
GtkWidget *size = lookup_widget (GTK_WIDGET (button), "p_size");
GtkWidget *fps = lookup_widget (GTK_WIDGET (button), "fps_val");
GtkWidget *cod = lookup_widget (GTK_WIDGET (button), "combo_entry1");
GtkWidget *src = lookup_widget (GTK_WIDGET (button), "src_val");
GtkWidget *chan = lookup_widget (GTK_WIDGET (button), "chan_val");
GtkWidget *file = lookup_widget (GTK_WIDGET (button), "file_val");
GtkWidget *vbit = lookup_widget (GTK_WIDGET (button), "vbit_val");
GtkWidget *abit = lookup_widget (GTK_WIDGET (button), "abit_val");
GtkWidget *agn = lookup_widget (GTK_WIDGET (button), "agn_val");
GtkWidget *prev = lookup_widget (GTK_WIDGET (button), "prev_val");
GtkWidget *rec = lookup_widget (GTK_WIDGET (button), "recTime");
GtkWidget *inter = lookup_widget (GTK_WIDGET (button), "de_inter");
GtkWidget *yuv = lookup_widget (GTK_WIDGET (button), "yuv_val");
GtkWidget *norm = lookup_widget (GTK_WIDGET (button), "norm_val");

GtkWidget *split = lookup_widget (pref_Window, "split_val");

GtkWidget *status = lookup_widget (GTK_WIDGET (button), "status");
GtkStyle *style = gtk_style_new ();
GtkStyle *style0 = gtk_style_new ();
GdkColor red = { 0, 45335, 0, 0 };
GdkColor green = { 0, 0, 35335, 0 };

strncpy (vdev_val, gtk_entry_get_text (GTK_ENTRY (vdev)), 250);
strncpy (adev_val, gtk_entry_get_text (GTK_ENTRY (adev)), 250);
strncpy (rec_dur, gtk_entry_get_text (GTK_ENTRY (rec)), 250);

duration = atoi (rec_dur);

strcat (p_size, gtk_entry_get_text (GTK_ENTRY (size)));
if (strcmp (p_size, "-g ") == 0)
       strcpy (p_size, " ");

strcat (fps_val, gtk_entry_get_text (GTK_ENTRY (fps)));
if (strcmp (fps_val, "-f ") == 0)
       strcpy (fps_val, " ");
else if (strcmp (fps_val, "-f 23.976") == 0)
       strcpy (fps_val, "-f 0,1");
else if (strcmp (fps_val, "-f 24") == 0)
       strcpy (fps_val, "-f 0,2");
else if (strcmp (fps_val, "-f 25") == 0)
       strcpy (fps_val, "-f 0,3");
else if (strcmp (fps_val, "-f 29.970") == 0)
       strcpy (fps_val, "-f 0,4");
else if (strcmp (fps_val, "-f 30") == 0)
       strcpy (fps_val, "-f 0,5");
else if (strcmp (fps_val, "-f 50") == 0)
       strcpy (fps_val, "-f 0,6");
else if (strcmp (fps_val, "-f 59.940") == 0)
       strcpy (fps_val, "-f 0,7");
else if (strcmp (fps_val, "-f 60") == 0)
       strcpy (fps_val, "-f 0,8");
else if (strcmp (fps_val, "-f 1") == 0)
       strcpy (fps_val, "-f 0,9");
```

```
else if (strcmp (fps_val, "-f 5") == 0)
        strcpy (fps_val, "-f 0,10");
else if (strcmp (fps_val, "-f 10") == 0)
        strcpy (fps_val, "-f 0,11");
else if (strcmp (fps_val, "-f 12") == 0)
        strcpy (fps_val, "-f 0,12");
else if (strcmp (fps_val, "-f 15") == 0)
        strcpy (fps_val, "-f 0,13");

strncpy (cod_val, gtk_entry_get_text (GTK_ENTRY (cod)), 250);
strncpy (vsrc_val, gtk_entry_get_text (GTK_ENTRY (src)), 250);
strncpy (chan_val, gtk_entry_get_text (GTK_ENTRY (chan)), 250);
on_combo_entry1_focus_out_event ((GtkWidget *) button, NULL, button);
strncpy (file_val, gtk_entry_get_text (GTK_ENTRY (file)), 250);
strncpy (norm_val, gtk_entry_get_text (GTK_ENTRY (norm)), 10);


strcat (vbit_val, gtk_entry_get_text (GTK_ENTRY (vbit)));
if (strcmp (vbit_val, "-w ") == 0)
        strcpy (vbit_val, " ");

strcat (abit_val, gtk_entry_get_text (GTK_ENTRY (abit)));
if (strcmp (abit_val, "-b ") == 0)
        strcpy (abit_val, " ");

strcat (agn_val, gtk_entry_get_text (GTK_ENTRY (agn)));
if (strcmp (agn_val, "-s ") == 0)
        strcpy (agn_val, " ");

deinter = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (inter));


prev_val = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (prev));
yuv_val = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (yuv));

if (atoi (gtk_entry_get_text (GTK_ENTRY (split))) > 0)
        strcat (split_val, gtk_entry_get_text (GTK_ENTRY (split)));
else
        split_val[0] = '\0';


system ("killall -2 transcode 2>/dev/null || echo Done > /tmp/autovideo.temp");
system ("killall -9 autovideo-sleep.sh 2>/dev/null");

do
        system ("sleep 0.5 2>/dev/null || sleep 0,5");
while (fopen ("/tmp/autovideo.temp", "r") == NULL);
system ("sleep 0.1 2>/dev/null || sleep 0,1");

system ("rm -rf /tmp/autovideo*");



if (strlen (vdev_val) == 0)
{
        GtkWidget *novid;
        novid = create_novid ();
        gtk_widget_show (novid);
        gtk_window_set_transient_for (GTK_WINDOW (novid), GTK_WINDOW (autovideo));
        on_stop_clicked ((GtkButton *) button, button);
        return;
}
if (strlen (adev_val) == 0)
{
        GtkWidget *noaud;
        noaud = create_noaud ();
        gtk_widget_show (noaud);
        gtk_window_set_transient_for (GTK_WINDOW (noaud), GTK_WINDOW (autovideo));
        on_stop_clicked ((GtkButton *) button, button);
        return;
}
```

```
        if (strlen (cod_val) == 0)
        {
                GtkWidget *nocod;
                nocod = create_nocod ();
                gtk_widget_show (nocod);
                gtk_window_set_transient_for (GTK_WINDOW (nocod), GTK_WINDOW (autovideo));
                on_stop_clicked ((GtkButton *) button, button);
                return;
        }
        else if (strcmp (cod_val, "ffmpeg4") == 0)
                strcpy (cod_val, "ffmpeg -F mpeg4");
        else if (strcmp (cod_val, "ffmjpeg") == 0)
                strcpy (cod_val, "ffmpeg -F mjpeg");
        else if (strcmp (cod_val, "msmpeg4") == 0)
                strcpy (cod_val, "ffmpeg -F msmpeg4");
        else if (strcmp (cod_val, "msmpeg4v2") == 0)
                strcpy (cod_val, "ffmpeg -F msmpeg4v2");


        if (strlen (vsrc_val) == 0)
        {
                GtkWidget *nosrc;
                nosrc = create_nosrc ();
                gtk_widget_show (nosrc);
                gtk_window_set_transient_for (GTK_WINDOW (nosrc), GTK_WINDOW (autovideo));
                on_stop_clicked ((GtkButton *) button, button);
                return;
        }
        for (i = 0; i < 9; i++)
        {
                sprintf (temp, "=%s\0", vsrc_val);
                if (strstr (v4l_info[i], vdev_val) && strstr (v4l_info[i], temp))
                {
                        pos = strstr (v4l_info[i], temp) - 1;
                        strncpy (src_val, pos, 1);
                        src_val[1] = '\0';
                }
        }



        rec_src_set ((GtkEditable *) button, button);

        if (fopen (file_val, "r") == NULL)
        {
                outFile = fopen ("/tmp/autovideo.sh", "w");
                fprintf (outFile, "#!/bin/bash\n");
                //fprintf (outFile, "v4lctl -c %s setnorm %s\n", vdev_val, norm_val);
         //set the TV standard (norm)
                if (ver0613 == TRUE)
                        fprintf (outFile,
                                 "transcode --progress_off -i %s -p %s -u 100 %s %s -y %s -
-import_v4l %s,\"%s\" %s -o \"%s\" %s %s %s %s %s %s %s",
                                 vdev_val, adev_val, p_size, fps_val, cod_val, src_val,
                                 chan_val, yuv_val == TRUE ? "-x v4l=yuv422" : "-x v4l",
file_val,
                                vbit_val, abit_val, agn_val, deinter == TRUE ? "-I 5" : "",
                                 prev_val == TRUE ? "-J pv" : "", split_val);
                else
                        fprintf (outFile,
                                 "transcode --progress_off -i %s -p %s -V -u 100 %s %s -
y %s --import_v4l %s,\"%s\" %s -o \"%s\" %s %s %s %s %s %s",
                                 vdev_val, adev_val, p_size, fps_val, cod_val, src_val,
                                chan_val, yuv_val == TRUE ? "-x v4l=yuv422" : "", file_val,
                                vbit_val, abit_val, agn_val, deinter == TRUE ? "-I 5" : "",
                                 prev_val == TRUE ? "-J preview" : "", split_val);

                if (duration != 0)
                {
                        fprintf (outFile, "&\n/tmp/autovideo-sleep.sh\n");
```

```
                fprintf (outFile, "\necho Done > /tmp/autovideo.temp\n");
        }
        else
                fprintf (outFile, "\necho Done > /tmp/autovideo.temp\n");

        fprintf (outFile, "echo -e '\n\n\'\n");
        fclose (outFile);

        if (duration != 0)
        {
                outFile = fopen ("/tmp/autovideo-sleep.sh", "w");
                fprintf (outFile, "#!/bin/bash\n");
                fprintf (outFile,
                        "sleep %dm\nkillall -2 transcode\nkillall -3 autovideo\nrm
-rf /tmp/autovideo*\n",
                        duration);
                fclose (outFile);
        }


        set_norm (vdev_val, norm_val);

        system ("chmod +x /tmp/autovideo.sh");
        system ("chmod +x /tmp/autovideo-sleep.sh 2>/dev/null");
        system ("/tmp/autovideo.sh >>/tmp/tr-output 2>&1 &");
    }
    else
    {
        outFile = fopen ("/tmp/autovideo.sh", "w");
        fprintf (outFile, "#!/bin/bash\n");
        //fprintf (outFile, "v4lctl -c %s setnorm %s\n", vdev_val, norm_val);
     //set the TV standard (norm)
        if (ver0613 == TRUE)
                fprintf (outFile,
                        "transcode --progress_off -i %s -p %s -u 100 %s %s -y %s -
-import_v4l %s,\"%s\" %s -o \"%s.append\" %s %s %s %s %s %s",
                        vdev_val, adev_val, p_size, fps_val, cod_val, src_val,
                        chan_val, yuv_val == TRUE ? "-x v4l=yuv422" : "-x v4l",
file_val,
                       vbit_val, abit_val, agn_val, deinter == TRUE ? "-I 5" : "",
                        prev_val == TRUE ? "-J pv" : "", split_val);
        else
                fprintf (outFile,
                        "transcode --progress_off -i %s -p %s -V -u 100 %s %s -
y %s --import_v4l %s,\"%s\" %s -o \"%s.append\" %s %s %s %s %s %s",
                        vdev_val, adev_val, p_size, fps_val, cod_val, src_val,
                       chan_val, yuv_val == TRUE ? "-x v4l=yuv422" : "", file_val,
                       vbit_val, abit_val, agn_val, deinter == TRUE ? "-I 5" : "",
                        prev_val == TRUE ? "-J preview" : "", split_val);
        if (duration != 0)
                fprintf (outFile, "&\n/tmp/autovideo-sleep.sh\n");
        else
                fprintf (outFile, "\n");

        fprintf (outFile,
                 "avimerge -o \"%s.new\" -i \"%s\" \"%s.append\" 2>/dev/null||echo
Done > /tmp/autovideo.temp|exit\n",
                 file_val, file_val, file_val);
        fprintf (outFile, "rm \"%s\" \"%s.append\"\n", file_val, file_val);
        fprintf (outFile, "mv \"%s.new\" \"%s\"\n", file_val, file_val);
        fprintf (outFile, "echo Done > /tmp/autovideo.temp\n");

        fprintf (outFile, "echo -e '\n\n\'\n");
        fclose (outFile);

        if (duration != 0)
        {
                outFile = fopen ("/tmp/autovideo-sleep.sh", "w");
                fprintf (outFile, "#!/bin/bash\n");
                fprintf (outFile,
```

```
                                "sleep %dm\nkillall -2 transcode\nkillall -3 autovideo\nrm
-rf /tmp/autovideo*\n",
                                duration);
                        fclose (outFile);
                }


                set_norm (vdev_val, norm_val);

                system ("chmod +x /tmp/autovideo.sh");
                system ("chmod +x /tmp/autovideo-sleep.sh 2>/dev/null");
                system ("/tmp/autovideo.sh >>/tmp/tr-output 2>&1 &");
        }


        style->text[GTK_STATE_NORMAL] = red;
        //      style->font = gdk_font_load("10x20");
        gtk_entry_set_text (GTK_ENTRY (status), "Recording...");
        gtk_widget_set_style (GTK_WIDGET (status), style);
        sleep (1);
        output_show ();


        system ("rm -rf /tmp/trr");

        system ("ps -A|grep transcode >/tmp/trr || rm -rf /tmp/trr");


        if (fopen ("/tmp/trr", "r") == NULL)
        {
                style0->text[GTK_STATE_NORMAL] = green;
                // style->font = gdk_font_load("10x20");
                gtk_entry_set_text (GTK_ENTRY (status), "Idle.");
                gtk_widget_set_style (GTK_WIDGET (status), style0);
        }
}


void on_start_activate (GtkMenuItem * menuitem, gpointer user_data)
{
        on_start_clicked_event ((GtkButton *) menuitem, menuitem);
}



void on_stop_clicked (GtkButton * button, gpointer user_data)
{

        GtkWidget *status = lookup_widget (GTK_WIDGET (button), "status");
        GtkStyle *style = gtk_style_new ();
        GdkColor green = { 0, 0, 35335, 0 };


        system ("killall -2 transcode 2>/dev/null || echo Done > /tmp/autovideo.temp");
        system ("killall -9 autovideo-sleep.sh 2>/dev/null");

        do
                system ("sleep 0.5 2>/dev/null || sleep 0,5");
        while (fopen ("/tmp/autovideo.temp", "r") == NULL);
        system ("sleep 0.1 2>/dev/null || sleep 0,1");


        system ("rm -rf /tmp/autovideo*");
        system ("rm -rf /tmp/trr");
        output_show ();


        style->text[GTK_STATE_NORMAL] = green;
        //style->font = gdk_font_load("10x20");
        gtk_entry_set_text (GTK_ENTRY (status), "Idle.");
        gtk_widget_set_style (GTK_WIDGET (status), style);
```

```
}


void on_stop_activate (GtkMenuItem * menuitem, gpointer user_data)
{
        on_stop_clicked ((GtkButton *) menuitem, menuitem);
}



void on_transcode_output_activate (GtkMenuItem * menuitem, gpointer user_data)
{

        if (visible[1] == 0)
        {
                tr_output = create_transcode_output ();
                gtk_widget_show (tr_output);
                visible[1] = 1;
        }
        output_show ();
}


void output_show ()
{
        GtkWidget *view;
        GtkTextIter start, end;
        GtkTextBuffer *buffer;
        FILE *inFile;
        const gchar *filename;
        gint bytes_read;
        gchar bufsize[BUFFER_SIZE];

        //GString *new_command;


        if (visible[1] == 0)
                tr_output = create_transcode_output ();
        view = lookup_widget (tr_output, "textview");
        buffer = gtk_text_view_get_buffer (GTK_TEXT_VIEW (view));

        //system ("iconv -t=utf-8 /tmp/tr-output >/tmp/autovideo-tr-out-utf 2>/dev/null");
        inFile = fopen ("/tmp/tr-output", "r");
        if (inFile == NULL)
        {
                gtk_text_buffer_set_text (buffer, "", -1);
                return;
        }

        gtk_text_buffer_set_text (buffer, "", -1);

        for (;;)
        {
                bytes_read = fread (bufsize, sizeof (gchar), BUFFER_SIZE, inFile);

                if (bytes_read > 0)
                {
                        gtk_text_buffer_get_end_iter (buffer, &end);
                        gtk_text_buffer_insert (buffer, &end,

                        g_get_charset (NULL) ? bufsize : g_locale_to_utf8 (bufsize, -1,
NULL, NULL,
                                                                NULL), bytes_read);
                }
                if (bytes_read != BUFFER_SIZE && (feof (inFile) || ferror (inFile)))
                        break;
        }
        if (ferror (inFile))
        {
                fclose (inFile);
```

```
                gtk_text_buffer_get_bounds (buffer, &start, &end);
                gtk_text_buffer_delete (buffer, &start, &end);
                gtk_text_buffer_set_text (buffer, "Unable to load file", -1);
                return;
        }

        fclose (inFile);
}

gboolean on_transcode_output_delete_event (GtkWidget * widget,

        GdkEvent * event, gpointer user_data)
{
        tr_output = create_transcode_output ();
        visible[1] = 0;
        return FALSE;
}


void on_output_hide_clicked (GtkButton * button, gpointer user_data)
{
        gtk_widget_destroy (tr_output);
        on_transcode_output_delete_event ((GtkWidget *) button, NULL, button);
}




void
on_clear_output_clicked (GtkButton * button, gpointer user_data)
{
        system ("rm -rf /tmp/tr~out*");
        output_show ();
}
```

//cron.c

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

#include <gnome.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "autovideo.h"
#include "interface.h"
#include "support.h"

extern GtkWidget *autovideo;
extern char conf_File[250];
extern gboolean rec;
extern gint visible[3];

GtkWidget *sched;
gchar sch_file[250] = "/tmp/autovideo-schedule";

//used in deletion confirmation box
char delete[500];


void on_schedule_click_event (GtkButton * button, gpointer user_data)
{
        gchar home[250];
        gchar tmp_file[250];   //temp file
        gchar config[250];             //config file
        gchar command[250];            //command to be written
        gint i;

        //GtkWidget *schedule;
        GtkWidget *cmd;
        //GtkWidget *wdsp;

        //call auto file extension
        //on_combo_entry1_focus_out_event ((GtkWidget *) button, NULL, button);

        //create window
        if (visible[2] == 1)
                return;
        sched = create_schedule ();
        gtk_widget_show (sched);
        visible[2] = 1;

        //lookup widget after it is created
        cmd = lookup_widget (sched, "txtCmd");
        //wdsp = lookup_widget (sched, "txtDisplay");

        strcpy (home, g_get_home_dir ());     //copy home dir value

        //set the default schedule file
        strcpy (sch_file, home);
        strcat (sch_file, "/.autovideo2/cron/schedule");
        //see if we have to create non-default schedule file
        if (fopen (sch_file, "r"))
        {
                for (i = 1; i < 10000; i++)
                {
                        sprintf (tmp_file, "%s-%d", sch_file, i);
                        if (fopen (tmp_file, "r") == NULL)
                        {
                                strcpy (sch_file, tmp_file);
                                i = 10000;
                        }
                }
        }
```

```
        //store config file to other variable
        strcpy (config, conf_File);
        //set the schedule file as config file
        strcpy (conf_File, sch_file);
        //run program init function
        on_save_settings1_activate ((GtkMenuItem *) button, button);
        //restore config file value
        strcpy (conf_File, config);

        //write the command to the appropriate textbox
        sprintf (command, "autovideo --file %s --record", sch_file);
        gtk_entry_set_text (GTK_ENTRY (cmd), command);
        //gtk_entry_set_text (GTK_ENTRY (wdsp), g_getenv ("DISPLAY"));
}

void on_schedule_activate_event (GtkMenuItem * menuitem, gpointer user_data)
{
        on_schedule_click_event ((GtkButton *) menuitem, menuitem);
}


void on_btnClose_clicked (GtkButton * button, gpointer user_data)
{
        //vars
        FILE *inFile;
        gchar cron_line[500];
        gchar schfile[500];
        GtkWidget *sched = lookup_widget (GTK_WIDGET (button), "schedule");

        strcpy (schfile, sch_file);
        //make sure we got " " after the filename in cron
        strcat (schfile, " ");
        //put cron to temp file
        system ("crontab -l > /tmp/autovideo-cron");
        inFile = fopen ("/tmp/autovideo-cron", "r");
        do
        {
                strcpy (cron_line, "unknown");
                fgets (cron_line, 500, inFile);

                //preserve schedule file if found in cron
                if (strstr (cron_line, schfile))
                        strcpy (sch_file, "/tmp/autovideo_needs_this_file");
                //remove schedule file if not on cron
                if (feof (inFile) != 0)
                        remove (sch_file);
        }
        while (feof (inFile) == 0);

        fclose (inFile);
        remove ("/tmp/autovideo-cron");

        //close window
        gtk_widget_destroy (sched);
        visible[2] = 0;
}

gboolean on_schedule_delete_event (GtkWidget * widget,
                                                                GdkEvent * event,
gpointer user_data)
{
        on_btnClose_clicked ((GtkButton *) widget, widget);
        return FALSE;
}


void on_chkMon_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllDays = lookup_widget (GTK_WIDGET (button), "chkAllDays");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllDays), FALSE);
}
```

```
void on_chkTue_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllDays = lookup_widget (GTK_WIDGET (button), "chkAllDays");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllDays), FALSE);
}


void on_chkWed_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllDays = lookup_widget (GTK_WIDGET (button), "chkAllDays");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllDays), FALSE);
}


void on_chkThu_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllDays = lookup_widget (GTK_WIDGET (button), "chkAllDays");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllDays), FALSE);
}


void on_chkFri_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllDays = lookup_widget (GTK_WIDGET (button), "chkAllDays");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllDays), FALSE);
}


void on_chkSat_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllDays = lookup_widget (GTK_WIDGET (button), "chkAllDays");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllDays), FALSE);
}


void on_chkSun_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllDays = lookup_widget (GTK_WIDGET (button), "chkAllDays");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllDays), FALSE);
}


void on_chkAllDays_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *Mon = lookup_widget (GTK_WIDGET (button), "chkMon");
        GtkWidget *Tue = lookup_widget (GTK_WIDGET (button), "chkTue");
        GtkWidget *Wed = lookup_widget (GTK_WIDGET (button), "chkWed");
        GtkWidget *Thu = lookup_widget (GTK_WIDGET (button), "chkThu");
        GtkWidget *Fri = lookup_widget (GTK_WIDGET (button), "chkFri");
        GtkWidget *Sat = lookup_widget (GTK_WIDGET (button), "chkSat");
        GtkWidget *Sun = lookup_widget (GTK_WIDGET (button), "chkSun");

        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Mon), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Tue), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Wed), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Thu), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Fri), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Sat), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Sun), FALSE);
}




void on_chkJan_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllMths = lookup_widget (GTK_WIDGET (button), "chkAllMths");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllMths), FALSE);
```

```
}


void on_chkJul_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllMths = lookup_widget (GTK_WIDGET (button), "chkAllMths");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllMths), FALSE);
}


void on_chkFeb_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllMths = lookup_widget (GTK_WIDGET (button), "chkAllMths");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllMths), FALSE);
}


void on_chkAug_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllMths = lookup_widget (GTK_WIDGET (button), "chkAllMths");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllMths), FALSE);
}


void on_chkMar_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllMths = lookup_widget (GTK_WIDGET (button), "chkAllMths");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllMths), FALSE);
}


void on_chkSep_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllMths = lookup_widget (GTK_WIDGET (button), "chkAllMths");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllMths), FALSE);
}


void on_chkApr_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllMths = lookup_widget (GTK_WIDGET (button), "chkAllMths");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllMths), FALSE);
}


void on_chkOct_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllMths = lookup_widget (GTK_WIDGET (button), "chkAllMths");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllMths), FALSE);
}


void on_chkMay_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllMths = lookup_widget (GTK_WIDGET (button), "chkAllMths");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllMths), FALSE);
}


void on_chkNov_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllMths = lookup_widget (GTK_WIDGET (button), "chkAllMths");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllMths), FALSE);
}


void on_chkJun_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllMths = lookup_widget (GTK_WIDGET (button), "chkAllMths");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllMths), FALSE);
}
```

```
void on_chkDec_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *AllMths = lookup_widget (GTK_WIDGET (button), "chkAllMths");
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (AllMths), FALSE);
}


void on_chkAllMths_pressed (GtkButton * button, gpointer user_data)
{
        GtkWidget *Jan = lookup_widget (GTK_WIDGET (button), "chkJan");
        GtkWidget *Feb = lookup_widget (GTK_WIDGET (button), "chkFeb");
        GtkWidget *Mar = lookup_widget (GTK_WIDGET (button), "chkMar");
        GtkWidget *Apr = lookup_widget (GTK_WIDGET (button), "chkApr");
        GtkWidget *May = lookup_widget (GTK_WIDGET (button), "chkMay");
        GtkWidget *Jun = lookup_widget (GTK_WIDGET (button), "chkJun");
        GtkWidget *Jul = lookup_widget (GTK_WIDGET (button), "chkJul");
        GtkWidget *Aug = lookup_widget (GTK_WIDGET (button), "chkAug");
        GtkWidget *Sep = lookup_widget (GTK_WIDGET (button), "chkSep");
        GtkWidget *Oct = lookup_widget (GTK_WIDGET (button), "chkOct");
        GtkWidget *Nov = lookup_widget (GTK_WIDGET (button), "chkNov");
        GtkWidget *Dec = lookup_widget (GTK_WIDGET (button), "chkDec");

        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Jan), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Feb), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Mar), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Apr), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (May), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Jun), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Jul), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Aug), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Sep), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Oct), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Nov), FALSE);
        gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (Dec), FALSE);
}
```

```
void on_btnCreate_clicked (GtkButton * button, gpointer user_data)
{
        gchar cron_job[500];
        int display;

        gchar month[250] = "", wkday[250] = "";
        gchar ttl[250], hr[250], min[250], day[250], cmd[250], dsp[250];
        gboolean jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec,
                                mon, tue, wed, thu, fri, sat, sun;

        GtkWidget *wttl = lookup_widget (GTK_WIDGET (button), "txtTitle");
        GtkWidget *whr = lookup_widget (GTK_WIDGET (button), "cbohrs");
        GtkWidget *wmin = lookup_widget (GTK_WIDGET (button), "cbomins");
        GtkWidget *wday = lookup_widget (GTK_WIDGET (button), "cboday");
        GtkWidget *wjan = lookup_widget (GTK_WIDGET (button), "chkJan");
        GtkWidget *wfeb = lookup_widget (GTK_WIDGET (button), "chkFeb");
        GtkWidget *wmar = lookup_widget (GTK_WIDGET (button), "chkMar");
        GtkWidget *wapr = lookup_widget (GTK_WIDGET (button), "chkApr");
        GtkWidget *wmay = lookup_widget (GTK_WIDGET (button), "chkMay");
        GtkWidget *wjun = lookup_widget (GTK_WIDGET (button), "chkJun");
        GtkWidget *wjul = lookup_widget (GTK_WIDGET (button), "chkJul");
        GtkWidget *waug = lookup_widget (GTK_WIDGET (button), "chkAug");
        GtkWidget *wsep = lookup_widget (GTK_WIDGET (button), "chkSep");
        GtkWidget *woct = lookup_widget (GTK_WIDGET (button), "chkOct");
```

```
GtkWidget *wnov = lookup_widget (GTK_WIDGET (button), "chkNov");
GtkWidget *wdec = lookup_widget (GTK_WIDGET (button), "chkDec");
GtkWidget *wMon = lookup_widget (GTK_WIDGET (button), "chkMon");
GtkWidget *wTue = lookup_widget (GTK_WIDGET (button), "chkTue");
GtkWidget *wWed = lookup_widget (GTK_WIDGET (button), "chkWed");
GtkWidget *wThu = lookup_widget (GTK_WIDGET (button), "chkThu");
GtkWidget *wFri = lookup_widget (GTK_WIDGET (button), "chkFri");
GtkWidget *wSat = lookup_widget (GTK_WIDGET (button), "chkSat");
GtkWidget *wSun = lookup_widget (GTK_WIDGET (button), "chkSun");
GtkWidget *wcmd = lookup_widget (GTK_WIDGET (button), "txtCmd");
GtkWidget *wdsp = lookup_widget (GTK_WIDGET (button), "txtDisplay");


strncpy (ttl, gtk_entry_get_text (GTK_ENTRY (wttl)), 250);
strncpy (hr, gtk_entry_get_text (GTK_ENTRY (whr)), 250);
strncpy (min, gtk_entry_get_text (GTK_ENTRY (wmin)), 250);
strncpy (day, gtk_entry_get_text (GTK_ENTRY (wday)), 250);
jan = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wjan));
feb = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wfeb));
mar = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wmar));
apr = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wapr));
may = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wmay));
jun = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wjun));
jul = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wjul));
aug = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (waug));
sep = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wsep));
oct = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (woct));
nov = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wnov));
dec = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wdec));
mon = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wMon));
tue = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wTue));
wed = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wWed));
thu = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wThu));
fri = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wFri));
sat = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wSat));
sun = gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (wSun));
strncpy (cmd, gtk_entry_get_text (GTK_ENTRY (wcmd)), 250);
strncpy (dsp, gtk_entry_get_text (GTK_ENTRY (wdsp)), 250);


if (jan == TRUE && strcmp (month, "") == 0)
       strcpy (month, "1");
else if (jan == TRUE)
       strcat (month, ",1");

if (feb == TRUE && strcmp (month, "") == 0)
       strcpy (month, "2");
else if (feb == TRUE)
       strcat (month, ",2");

if (mar == TRUE && strcmp (month, "") == 0)
       strcpy (month, "3");
else if (mar == TRUE)
       strcat (month, ",3");

if (apr == TRUE && strcmp (month, "") == 0)
       strcpy (month, "4");
else if (apr == TRUE)
       strcat (month, ",4");

if (may == TRUE && strcmp (month, "") == 0)
       strcpy (month, "5");
else if (may == TRUE)
       strcat (month, ",5");

if (jun == TRUE && strcmp (month, "") == 0)
       strcpy (month, "6");
else if (jun == TRUE)
       strcat (month, ",6");
```

```
if (jul == TRUE && strcmp (month, "") == 0)
        strcpy (month, "7");
else if (jul == TRUE)
        strcat (month, ",7");

if (aug == TRUE && strcmp (month, "") == 0)
        strcpy (month, "8");
else if (aug == TRUE)
        strcat (month, ",8");

if (sep == TRUE && strcmp (month, "") == 0)
        strcpy (month, "9");
else if (sep == TRUE)
        strcat (month, ",9");

if (oct == TRUE && strcmp (month, "") == 0)
        strcpy (month, "10");
else if (oct == TRUE)
        strcat (month, ",10");

if (nov == TRUE && strcmp (month, "") == 0)
        strcpy (month, "11");
else if (nov == TRUE)
        strcat (month, ",11");

if (dec == TRUE && strcmp (month, "") == 0)
        strcpy (month, "12");
else if (dec == TRUE)
        strcat (month, ",12");

if (strcmp (month, "") == 0
        || strcmp (month, "1,2,3,4,5,6,7,8,9,10,11,12") == 0)
        strcpy (month, "*");


if (mon == TRUE && strcmp (wkday, "") == 0)
        strcpy (wkday, "1");
else if (mon == TRUE)
        strcat (wkday, ",1");

if (tue == TRUE && strcmp (wkday, "") == 0)
        strcpy (wkday, "2");
else if (tue == TRUE)
        strcat (wkday, ",2");

if (wed == TRUE && strcmp (wkday, "") == 0)
        strcpy (wkday, "3");
else if (wed == TRUE)
        strcat (wkday, ",3");

if (thu == TRUE && strcmp (wkday, "") == 0)
        strcpy (wkday, "4");
else if (thu == TRUE)
        strcat (wkday, ",4");

if (fri == TRUE && strcmp (wkday, "") == 0)
        strcpy (wkday, "5");
else if (fri == TRUE)
        strcat (wkday, ",5");

if (sat == TRUE && strcmp (wkday, "") == 0)
        strcpy (wkday, "6");
else if (sat == TRUE)
        strcat (wkday, ",6");

if (sun == TRUE && strcmp (wkday, "") == 0)
        strcpy (wkday, "7");
else if (sun == TRUE)
        strcat (wkday, ",7");

if (strcmp (wkday, "") == 0 || strcmp (wkday, "1,2,3,4,5,6,7") == 0)
```

```
                strcpy (wkday, "*");


        display = atoi (dsp);

        sprintf (cron_job, "# %s\n%s\t%s\t%s\t%s\t%s\t%s\n\0", ttl, min, hr, day,
                        month, wkday, cmd);

        update_cron ("create", cron_job, display);

        on_schedule_show ((GtkWidget *) button, button);
}



void on_schedule_show (GtkWidget * widget, gpointer user_data)
{
        GtkWidget *treeview = lookup_widget (GTK_WIDGET (sched), "treeview");
        GtkCellRenderer *render;
        GtkTreeModel *treemodel;

        GtkTreeStore *treestore = gtk_tree_store_new (3, G_TYPE_STRING, G_TYPE_STRING,
G_TYPE_BOOLEAN);
        GtkTreeIter iter, parent, child;
        FILE *inFile;
        gchar cron_line[500];
        gchar enabled[500], disabled[500], value[500];
        gchar *pos;


        system ("crontab -l > /tmp/autovideo-cron-r");
        inFile = fopen ("/tmp/autovideo-cron-r", "r");
        while (feof (inFile) == 0)
        {
                strcpy (cron_line, "#");
                fgets (cron_line, 500, inFile);

                if (strstr (cron_line, "\n"))
                {
                        pos = strstr (cron_line, "\n");
                        strncpy (pos, "\0", 1);
                }

                strncpy (disabled, cron_line, 2);
                if (strstr (disabled, "#\\"))
                {
                        strcpy (enabled, "NO");
                        pos = cron_line + 2;
                        sprintf (value, "%s", pos);
                        gtk_tree_store_append (treestore, &parent, NULL);
                        gtk_tree_store_set (treestore, &parent, VALUE, value, ENABLED,
                                                        enabled, EDITABLE, TRUE, -1);
                }
                else if (strstr (disabled, "#"));
                else
                {
                        strcpy (enabled, "YES");
                        sprintf (value, "%s", cron_line);
                        gtk_tree_store_append (treestore, &parent, NULL);
                        gtk_tree_store_set (treestore, &parent, VALUE, value, ENABLED,
                                                        enabled, EDITABLE, TRUE, -1);
                }
        }
        fclose (inFile);
        remove ("/tmp/autovideo-cron-r");



        treemodel = GTK_TREE_MODEL (treestore);
        gtk_tree_view_set_model (GTK_TREE_VIEW (treeview),
```

```
                                                        GTK_TREE_MODEL (treemodel));
        if (!gtk_tree_view_get_columns (GTK_TREE_VIEW (treeview)))
        {
                render = gtk_cell_renderer_text_new ();
                //g_signal_connect (G_OBJECT(render), "edited",
G_CALLBACK(lastname_edited), treemodel);
                gtk_tree_view_insert_column_with_attributes (GTK_TREE_VIEW (treeview),

         -1, "Enabled", render,

         "text", ENABLED,


         NULL);
                render = gtk_cell_renderer_text_new ();
                gtk_tree_view_insert_column_with_attributes (GTK_TREE_VIEW (treeview),

         -1, "Value", render,

         "text", VALUE, NULL);
        }


        //gtk_container_add (GTK_CONTAINER (sched), treeview);

        gtk_tree_model_get_iter_first (GTK_TREE_MODEL (treemodel), &iter);
}



void on_enable_task_clicked (GtkButton * button, gpointer user_data)
{
        GtkWidget *treeview = lookup_widget (GTK_WIDGET (button), "treeview");
        GtkTreeModel *model;
        GtkTreeIter iter;
        if (!gtk_tree_selection_get_selected
                (gtk_tree_view_get_selection (GTK_TREE_VIEW (treeview)), &model, &iter))
        {
                GtkWidget *no_selection = create_no_selection ();
                gtk_widget_show (no_selection);
                gtk_window_set_transient_for (GTK_WINDOW (no_selection), GTK_WINDOW
(sched));
                return;
        }
        else
        {
                gchar *enabled;
                gchar *val;
                int i;
                gtk_tree_model_get (model, &iter, ENABLED, &enabled, -1);
                gtk_tree_model_get (model, &iter, VALUE, &val, -1);
                if (strstr (val, "DISPLAY=") || strstr (val, "PATH="))
                {
                        GtkWidget *variables = create_variables ();
                        gtk_widget_show (variables);
                        gtk_window_set_transient_for (GTK_WINDOW (variables), GTK_WINDOW
(sched));
                        return;
                }

                if (strcmp (enabled, "YES") == 0)
                        return;
                else
                        update_cron ("enable", val, 0);

                on_schedule_show ((GtkWidget *) button, button);
                g_free (enabled);
                g_free (val);
        }
}
```

```
void on_disable_task_clicked (GtkButton * button, gpointer user_data)
{
        GtkWidget *treeview = lookup_widget (GTK_WIDGET (button), "treeview");
        GtkTreeModel *model;
        GtkTreeIter iter;
        if (!gtk_tree_selection_get_selected
                (gtk_tree_view_get_selection (GTK_TREE_VIEW (treeview)), &model, &iter))
        {
                GtkWidget *no_selection = create_no_selection ();
                gtk_widget_show (no_selection);
                gtk_window_set_transient_for (GTK_WINDOW (no_selection), GTK_WINDOW
(sched));
                return;
        }
        else
        {
                gchar *enabled;
                gchar *val;
                gtk_tree_model_get (model, &iter, ENABLED, &enabled, -1);
                gtk_tree_model_get (model, &iter, VALUE, &val, -1);
                if (strstr (val, "DISPLAY=") || strstr (val, "PATH="))
                {
                        GtkWidget *variables = create_variables ();
                        gtk_widget_show (variables);
                        gtk_window_set_transient_for (GTK_WINDOW (variables), GTK_WINDOW
(sched));
                        return;
                }

                if (strcmp (enabled, "NO") == 0)
                        return;
                else
                        update_cron ("disable", val, 0);

                on_schedule_show ((GtkWidget *) button, button);
                g_free (enabled);
                g_free (val);
        }
}


void on_delete_task_clicked (GtkButton * button, gpointer user_data)
{
        GtkWidget *treeview = lookup_widget (GTK_WIDGET (button), "treeview");
        GtkTreeModel *model;
        GtkTreeIter iter;
        if (!gtk_tree_selection_get_selected
                (gtk_tree_view_get_selection (GTK_TREE_VIEW (treeview)), &model, &iter))
        {
                GtkWidget *no_selection = create_no_selection ();
                gtk_widget_show (no_selection);
                gtk_window_set_transient_for (GTK_WINDOW (no_selection), GTK_WINDOW
(sched));
                return;
        }
        else
        {
                GtkWidget *del = create_delete_task ();
                gchar *enabled;
                gchar *val;
                gtk_tree_model_get (model, &iter, VALUE, &val, -1);
                if (strstr (val, "DISPLAY=") || strstr (val, "PATH="))
                {
                        GtkWidget *variables = create_variables ();
                        gtk_widget_show (variables);
                        gtk_window_set_transient_for (GTK_WINDOW (variables), GTK_WINDOW
(sched));
                        return;
                }
```

```
                strcpy (delete, val);
                gtk_widget_show (del);
                gtk_window_set_transient_for (GTK_WINDOW (del), GTK_WINDOW (sched));
                g_free (val);
        }
}



void on_delete_task_yes_clicked (GtkButton * button, gpointer user_data)
{
        GtkWidget *del = lookup_widget (GTK_WIDGET (button), "delete_task");
        update_cron ("delete", delete, 0);
        delete[0] = '\0';

        on_schedule_show ((GtkWidget *) button, button);
        gtk_widget_destroy (del);
}



void on_delete_task_no_clicked (GtkButton * button, gpointer user_data)
{
        GtkWidget *del = lookup_widget (GTK_WIDGET (button), "delete_task");
        gtk_widget_destroy (del);
}



void update_cron (char action[20], char cron_job[500], int display)
{
        FILE *outFile;
        FILE *inFile;
        gchar cron_line[500], cron_tit[500] = "", disabled[500] = "";
        int fnd_dis = 0;
        int fnd_path = 0;

        system ("crontab -l > /tmp/autovideo-cron");

        inFile = fopen ("/tmp/autovideo-cron", "r");
        outFile = fopen ("/tmp/autovideo-cron-new", "w");

        while (feof (inFile) == 0)
        {
                cron_line[0] = '\0';
                fgets (cron_line, 500, inFile);
                if (feof (inFile))
                {
                        if (fnd_dis == 0)
                        {
                                sprintf (cron_line, "# X display
variable\nDISPLAY=\":%d.0\"\n", display);
                                fputs (cron_line, outFile);
                        }
                        break;
                }
                cron_line[499] = '\0';

                if (strstr (cron_line, cron_job))
                {
                        //if (strstr (action, "create"))
                        //sprintf (action, "no-create-exists");
                        if (strstr (action, "delete"))
                                cron_line[0] = '\0';
                        else
                                fputs (cron_tit, outFile);
                        if (strstr (action, "enable"))
                                fprintf (outFile, "%s\n", cron_job);
                        if (strstr (action, "disable"))
                                fprintf (outFile, "#\\%s\n", cron_job);
```

```
                        cron_line[0] = '\0';
                        cron_tit[0] = '\0';
                        //continue;
                }


                if (strstr (cron_line, "DISPLAY="))
                {
                        if (strstr (action, "create"))
                        {
                                sprintf (cron_line, "# X display
variable\nDISPLAY=\":%d.0\"\n", display);
                                fputs (cron_line, outFile);
                        }
                        fnd_dis = 1;
                }


                if (strstr (cron_line, "PATH="))
                        fnd_path = 1;


                strncpy (disabled, cron_line, 2);
                if (strstr (disabled, "#\\"))
                {
                        fputs (cron_tit, outFile);
                        //strcpy (cron_tit, "");
                        cron_tit[0] = '\0';
                        fputs (cron_line, outFile);
                }
                else if (strstr (disabled, "#"))
                        strcpy (cron_tit, cron_line);
                else
                {
                        fputs (cron_tit, outFile);
                        //strcpy (cron_tit, "");
                        cron_tit[0] = '\0';
                        fputs (cron_line, outFile);
                }
        }

        fclose (inFile);
        fclose (outFile);

        if (fnd_path == 0)
        {
                system ("echo \"# PATH variable for cron\" >> /tmp/autovideo-cron-new");
                system ("echo \"PATH=$PATH\" >> /tmp/autovideo-cron-new");
        }

        outFile = fopen ("/tmp/autovideo-cron-new", "a");

        if (strcmp (action, "create") == 0)
                fputs (cron_job, outFile);
        fclose (outFile);


        system ("crontab /tmp/autovideo-cron-new");
        remove ("/tmp/autovideo-cron-new");
        remove ("/tmp/autovideo-cron");
}
```