Users@Planet CCRMA

(The Linux Survival Guide)

Juan Reyes
juanig@ccrma.stanford.edu

Abstract

This guide takes the approach of a traveler's survival companion and its sole purpose is to illustrate, show and inform new CCRMA users and visitors about computer resources and the Linux environment and applications which might be helpful for doing scientific research and compositional work at CCRMA. It also briefly describes the meaning of Open Source as a part of the laboratory and community philosophy here at CCRMA. Following is a brief history of hardware at CCRMA and descriptions of Linux as an operating system, the Unix environment, useful shell commands and many X windows, Gnome and KDE applications. In the applications section there will be useful descriptions of programs and information provided by developers' documentation but do not discard some of the tricks and options as to our knowledge we present to the reader in some of these sections. If additional information is required in the case of a particular command, program or application, the reader is encouraged to look for more in-depth information on the Unix manual pages, on the web or in the links to home pages which are also provided here. Furthermore, there is some effort in several sections, to present solutions to common problems or challenges the system presents to the common user, beware of them.

Contents

1 Preface to this Revision

This is a revision to the original "Users@PlanetCCRMA" page, result of a project to introduce CCRMA students, researchers, composers as well as interested people, part of the community or not. On this revision, features of previous editions are retained mainly for historical reasons, and for documentation. New and updated tricks and methods for current Linux Fedora workstations are part of these updates. Linux, being Unix for the masses, retains much of its model, thereby what has been learned before prevails and responds in similar ways. As with technology trends, most changes occur on hardware, therefore, new drivers and ways for accessing devices are always in need. In most respects main changes to this operating system are product of hardware and technological improvements. Furthermore, processors are speedier and available memory expands all the time, to the point that these are not a worries anymore. Linux being the quintessential development environment has projects coming out all the time, so new software and improvements to legacy applications are always happening. From musical or audio frameworks, paradigms remain the same, but in cases new methods and programs ease tasks, while permitting options not available years before. On this revision some of these changes are brought, in addition to a fresh new look at the interface, and updates to previous features. Well aware that after this guide was originally written, **Wikis** became the 'de facto' mode of documentation, at this time we still find that CCRMA's wiki complements this information and vice-versa. Since Linux continuously evolves, there is no option but to assert that this project remains a work in progress.

`Fri Jun 17 03:15:54 PM PDT 2022`

NOTES: This revision contains a tutorial and hands on approach to Snd audio editor which itself is an introduction to sound synthesis and signal processing, legacy topics at CCRMA for years.

2 Motivation: Computer Music

Intuition, insight, curiosity, discovery are qualities which help enhancing a lifestyle around various activities that keep CCRMA alive. Spirit of an explorer, to get around and finding out about the unknown, as well as artist intuition for knowing what is looking good, might be sound advice in order to be able expressing ideas as freely and as good as possible on the knoll surroundings but also within a established Computer Music community. The field of this subject matter is so dynamic that even meanings of what Computer Music might be are being changed as we speak. Perhaps Bill Schottstaedt's vision relating Computer Music as the production of expressive music by means of computational tools, systems or devices can be a bit too generalized these days however for most purposes outlines most goals pertaining research and composition, scientific or artistic, of work done at CCRMA for most part. Computers are just tools for creation and knowledge production. At CCRMA they are essential for most activities and needs. Here we present a guide with elements that hopefully will assist users getting around a myriad of technologies that help ideas, hypotheses or, whatever the goal is, to materialize and become realities. Linux world at CCRMA might be a path to great discoveries.

Curtis Roads (1985) points out about the following quote from Edgar Varèse (circa 1939), on his vision and quest for electronic musical instruments that well pertains as further motivation for Computer Music:

> Here are the advantages I anticipate from such a machine: liberation from the arbitrary, paralyzing tempered system; the possibility of obtaining any number of cycles or, if still desired, subdivisions of the octave, consequently the formation of any desired scale; unsuspected range in low and high registers; new harmonic splendors obtainable from the use of sub harmonic combinations now impossible; the possibility of obtaining any differentiation of timbre, of sound combinations; new dynamics far beyond the present human-powered orchestra; a sense of sound-projection in space by means of the emission of sound in any part or in as many parts of the hall as may be required by the score; cross rhythms unrelated to each other, treated simultaneously... all of these in a given unit of measure or time which is humanly impossible to obtain. (Varèse 1966)

3 Some of the History of Hardware at CCRMA

- AI Department at Stanford:

  In the keynote speech address at the International Computer Music Conference of 1993 in Tokyo John Chowning says:

  > The computer system to which I had access was powerful indeed. It comprised an IBM 7090 that had an enormous memory of 36k 36-bit words and a hard disk whose capacity was well over 500k words and about the size of a refrigerator. The hard disk was shared by a DEC PDP-1 computer. By September 1964, with a great amount of help from a young Stanford undergraduate mathematics major, we implemented the Music IV program, provided by Max Mathews on the IBM 7090 and used the PDP-1 as buffer memory to send the samples from the hard disk to the DEC scope. The x-y converters for deflecting the electron beam of the CRT were connected to separate channels of an audio system, thus providing for stereo output.
  >
  > The young undergraduate who engineered this system was David Poole. He was extraordinarily helpful and patient as I at age 30, learned from him about things that I would have never have thought to be important to music. He was not only responsible for this, the first

on-line computer music system, but later went on to design and build really large computers, one of which served many years as the platform for CCRMA's digital synthesizer. This synthesizer was fondly known as the "Samson Box " After its designer Peter Samson of System Concepts in San Francisco.

With this powerful sound synthesis concept at the center, the adventure of computer music began. Music IV was so well-conceived that it was the progenitor of a number of offspring on a variety of systems used by a number of people from a variety of disciplines who intermingled and conversed as they waited for jobs to run or terminals to become free. Quite by chance, surprising and serendipitous alliances were made. Seemingly disparate fields found substantive connections. Composer/musicians needed to know what engineering scientists and cognitive/perceptual psychologists knew and heard. Signal processing, acoustics, psycho-acoustics and computer programming became familiar and necessary terrains of knowledge for musicians and music became a rich domain of application for engineering and perceptual sciences. Escher, Sheperd, and Risset (graphic artist, cognitive psychologist, and composer/physicist) became linked through compositions in which powerful visual illusions suggested compelling auditory counterparts. Young composers were nurtured by scientists and engineers- Godfrey Winham at Princeton, David Poole and George Gucker at Stanford, Jim Beauchamp at Illinois and, of course, Max Mathews. What a beginning – the field of music would acquire another dimension. [2, Chowning, 1993].

- Samson BOX:

In October 1977, CCRMA took delivery of the Systems Concepts Digital Synthesizer affectionately known as the "Samson Box," named after its designer Peter Samson. The Samson Box resembled a green refrigerator in the machine room at the Stanford Artificial Intelligence Laboratory, and it cost on the order of 100,000. In its hardware architecture, it provided 256 generators (waveform oscillators with several modes and controls, complete with amplitude and frequency envelope support), and 128 modifiers (each of which could be a second-order filter, random-number generator, or amplitude-modulator, among other functions).[4, Loy, 1991]. Up to 64 Kwords of delay memory with 32 access ports could be used to construct large wave-tables and delay lines. A modifier could be combined with a delay port to construct a high-order comb filter or Schroeder all-pass filter–fundamental building blocks of digital reverberators. Finally, four digital-to-analog converters came with the Box to supply four-channel sound output. These analog lines were fed to a 16-by-32 audio switch that routed sound to various listening stations around the lab.

The Samson Box was an elegant implementation of nearly all known, desirable, unit-generators in hardware form, and sound synthesis was sped up by three orders of magnitude in many cases. Additive, subtractive, and nonlinear FM synthesis and wave-shaping were well supported. Much music was produced by many composers on the Samson Box over more than a decade. It was a clear success. [13, Smith,1991]

- The NeXT Machines:

The NeXT computer still found at CCRMA in its black or white architecture was the first computer with a DSP or signal processing chip which could be dedicated to sound or music. In 1989 and for the sole purpose of developing a music workstation NeXT Inc. of Redwood City CA., hired Stanford Graduate Julius O. Smith and composer David A. Jaffe as well as chief software engineer Lee Boynton to develop the NeXT Music Kit. The Music Kit is an object-oriented software system for building music, sound, signal processing and MIDI applications on the NeXT computer. It has been used in such diverse commercial applications as music sequencers, notation packages, computer games, and document processors. Professors and students in the academia have used the Music Kit in a host of areas, such as music performance, scientific experiments, computer aided instruction and physical modeling. The Music Kit was the first system to unify the MIDI and Music V paradigm, thus combining interaction with generality. It was developed by NeXT Computer, Inc. from 1986 to 1991, and by CCRMA at Stanford University from 1992 to 1996. It has also been supported by developers such as Pinnacle Research, Inc., as well as the Stanford University Office of Technology Licensing. Furthermore Julius Smith and David Jaffe designed and implemented DSP56001 software supporting the NeXT Music Kit, including the real-time DSP monitor and unit-generator modules for sound synthesis and signal processing and also wrote and supported The NeXT DSP Library. Helped support and debug the Sound/DSP Mach driver and the NeXT Sound Library.

The entire collection and algorithms of the Samson Box were translated to the NeXT by means of Common Lisp and a Music V dialect known as CLM or Common Lisp Music all done by hand by composer/scientist Bill Schottstaedt. Most of the composition work at CCRMA in the 1990's was done using home brewed software like Common Lisp Music, Common Music, the Music Kit and more. The CCRMA environment at the moment (circa 1993) consisted of an Ethernet network which connected workstations running the NeXTStep operating system and Macintosh computers plus a gateway that connected the workstations to the campus at large and also to national and international networks.

- The Lab and Open Source Community, Linux:

  At this time, the CCRMA computing environment is supported by more than 40 machines that include fast Pentium class PCs running Linux (some of them dual-booting Linux and NEXTSTEP), Silicon Graphics workstations, NeXT workstations (for old time's sake) and PowerPC Macintosh computers. All machines are connected through a switched high speed backbone and several servers provide shared services and resources to all computers in a way that is transparent to the users. A high speed connection to the Stanford University Network (SUNET) provides connectivity with the rest of the world, including direct access to the new Internet 2 network. Sound-file manipulation and MIDI input and output are supported on all platforms. Multichannel playback is supported on some Linux and SGI workstations and on the Macs through several Pro Tools systems. Digital audio processors include a Studer-Editech Dyaxis II system, two Digidesign Pro-Tools systems with CD-R drives, digital i/o cards on Linux systems, Singular Solutions analog and digital audio input systems for the NeXTs, and several Panasonic DAT recorders. Text and graphics are handled by an HP 4c color scanner on the Unix-based systems and by high resolution network connected printers

4 Down to specifics: Linux

4.1 What-is-Linux

Linux is an operating system, a software program that controls your computer. An operating system solves several problems arising from several hardware variations like computer keyboards, mice, monitors, graphic cards and even MIDI or sound input and output devices.The problem is solved by providing a standard way for applications (programs) to access hardware devices. When an operating system exists, applications can be more compact, because they share the commonly used code for accessing the hardware. The operating system can do many other things as well like for example, providing a file system so that you can store and retrieve data and very important, a user interface so that you can control the operation of your computer.

4.2 Open-Source

Some, including MIT scientist Richard Stallman, yearned for the return of those happier times and the mutual cooperation of programmers that existed when Unix was not commercial. So in 1983, Stallman launched the GNU (GNU's not Unix) project, which aimed at creating a free Unix-like operating system. Like early Unix, the GNU operating was to be distributed in source form so that programmers could read, modify, and redistribute it without restriction. Stallman's work at MIT had taught him that by using the Internet as a means of communication, programmers the world over could improve and adapt software at incredible speed, far out pacing the fastest rate possible using traditional software development models, in which few programmers actually see one another source code. [7, McCarty,1999]

4.3 Why-Linux at CCRMA

Linux is a real Unix, multiuser, multitasking and file sharing operating system and yes because Linux is a hacker's playground. Linux is a cross-platform operating system that runs on many computer models. Linux is free and many Linux applications are distributed in source making possible for you and others to modify or improve them. And because Linux is a good if not an ideal platform for developing your own software. Sound and music applications are maturing and provide an unprecedented flexibility only found years ago on the NeXT computers with the music kit. Linux offers a good choice of compilers for all the tastes around CCRMA. No other present popular operating system is more stable and reliable than Linux and it has a blazing fast performance.

4.4 Choices: Why RedHat,... Fedora

Choice is always difficult and trying to be as democratic as possible is not an easy job. Fernando Lopez-Lezcano responds at the issue of linux distributions as follows:

> There are several reasons, I guess mostly historical at this point. Please keep in mind that PlanetCCRMA as a project was not "designed" (ie: We did not say at some point "let's get a project started and let's base it on RedHat"). What you see now as "PlanetCCRMA" is just the result of initially making available to the world the custom-built packages that I was using to maintain a Linux based, music and audio-friendly environment at CCRMA. Through mostly user feedback the project has recently grown out of control, and now I can truly say: "I have created a monster" :-).
>
> The Linux environment at CCRMA has been RedHat based for quite a while (I think the first version I installed was RedHat version 4 or something like that back in '97). At that time one of the appeals of RedHat was RPM (the package manager) and an easy to use installer (I was previously using Slackware, first Linux install was done some time in '96). At several points in time (even before what I was doing internally at CCRMA became available as "PlanetCCRMA") I considered switching to a different distro, and I really considered Debian (and others), but Debian was not (at that time) as easy to install as RedHat and the stable release was _too_ stable (for my taste). For good or bad, the choice was made a long time ago and each time I have again considered switching, there have not been enough incentives to do so(*).
>
> Regarding "apt" (the almost automatic installer up-grader and updater), PlanetCCRMA originally did not use it, and, as you may guess, it was not very usable without it :-). Apt made a huge difference in making it easy to install and update for non-CCRMA users, but the choice of apt was not "to make the distro act more like Debian". I could have used another program other than apt, but at that time, apt was, I think, the best choice (for example, there was also a free implementation of the up2date server that I considered but it was too young in its development cycle and apt for rpm seemed like a better and potentially more stable choice that leveraged on Debian's experience with it). If I started from scratch today I could use yum, for example (and I will, when I have a slice of that mythical thing called "free time").
>
> Anyway, sorry for the long post, but hope it clarifies _some_ of the reasons. This is not to say that CCRMA and (as a side effect) PlanetCCRMA will stay forever with RedHat - Fedora Core. Who knows what the future will bring(**)– Fernando(*) there has to be a considerable advantage, I maintain a customized environment of 40+ machines so switching involves learning all the quirks of a new distro, plus customizing it so that it behaves in pretty much the same way as before and creates the same environment. Not impossible, of course, but a lot of work. Even a version transition within the same distro involves a lot of work! And it is harder now because it is not

just CCRMA anymore...(**) for example Progeny is porting the anaconda installer to Debian! So, Debian could, at some point, be as easy to install as RedHat-Fedora.

4.5 The Unix Environment

UNIX is an operating system consisting of three important features; a kernel, the shell and a file system.

As its name implies, the kernel is at the core of each UNIX system and is loaded in whenever the system is started up - referred to as a boot of the system. It manages the entire resources of the system, presenting them to you and every other user as a coherent system. You do not need to know anything about the kernel in order to use a UNIX system. Amongst the functions performed by the kernel are:

- managing the machine's memory and allocating it to each process.
- scheduling the work done by the CPU so that the work of each user is carried out as efficiently as is possible.
- organizing the transfer of data from one part of the machine to another.
- accepting instructions from the shell and carrying them out.
- enforcing the access permissions that are in force on the file system.

Whenever you login to a Unix system you are placed in a program called the **shell**. You can see its prompt at the bottom left of your screen. To get your work done, you enter commands at this prompt. The shell acts as a command interpreter; it takes each command and passes it to the operating system kernel to be acted upon. It then displays the results of this operation on your screen. The shell provides you with one or more of the following features. You can:

- create an environment that meets your needs
- write shell scripts
- define command aliases
- manipulate the command history
- automatically complete the command line
- edit the command line

A **file system** is a logical method for organizing and storing large amounts of information in a way which makes it easy manage. The **file system** is the smallest unit in which information is stored. The UNIX file system has several important features:

- Different types of files

  To you, the user, it appears as though there is only one type of file in UNIX - the file which is used to hold your information. In fact, the UNIX file system contains several types of files.

  - **Ordinary File:** This type of file is used to store your information, such as some text you have written or an image you have drawn. This is the type of file that you usually work with.
  - **Directories:** A directory is a file that holds other files and other directories. You can create directories in your home directory to hold files and other sub-directories.

    Having your own directory structure gives you a definable place to work from and allows you to structure your information in a way that makes best sense to you.

    Directories which you create belong to you - you are said to "own" them - and you can set access permissions to control which other users can have access to the information they contain.

  - **Special files:** This type of file is used to represent a real physical device such as a printer, tape drive or terminal.

    It may seem unusual to think of a physical device as a file, but it allows you to send the output of a command to a device in the same way that you send it to a file. For example:

    ```
    cat scream.au > /dev/audio
    ```

    This sends the contents of the sound file scream.au to the file /dev/audio which represents the audio device attached to the system. Guess what sound this makes?

    The directory /dev contains the special files which are used to represent devices on a UNIX system.
  - **Pipes:** UNIX allows you to link commands together using a pipe. The pipe acts a temporary file which only exists to hold data from one command until it is read by another.

- Structure of the File System
  - **Your home directory:** Any UNIX system can have many users on it at any one time. As a user you are given a home directory in which you are placed whenever you log on to the system.

    User's home directories are usually grouped together under a system directory such as /home. A large UNIX system may have several hundred users, with their home directories grouped in sub directories according to some schema such as their organizational department.
  - **Your current directory:** When you log on to the system you are always placed in your home directory. At first this is your current directory. If you then change to another directory this becomes your current directory. The command **pwd** displays the full path name to your current directory.
  - **Pathnames:** Every file and directory in the file system can be identified by a complete list of the names of the directories that are on the route from the root directory to that file or directory.
  - **Access permissions:** Every file and directory in your account can be protected from or made accessible to other users by changing its access permissions. You can only change the permissions for files and directories that you own.

5 Unix-software examples:

- Shell Commands:

  A good tour suggestion from Craig Sapp is a nice tutorial of the Unix shell which found at: **Linux-Command.org** http://www.linuxcommand.org/index.php

  Manual pages (or rather ...help) of shell commands can also be found at: **The Linux man-pages project** https://www.kernel.org/doc/man-pages/

  More information about Linux commands can be found at section: §9.4.

  A good palette of useful Unix at CCRMA commands which will safely help you navigate in a Unix/Linux environment might include:

5.1 General use

`man`	Get information or help about a command
`apropos`	Locate commands by keyword look up
`ls`	List directory contents
`pwd`	Path of working directory
`cd`	Change working directory
`mv`	Move file or change name
`cp`	Copy file
`mkdir`	Make directory
`rm`	Remove files and directories
`cat`	output contents of a file
`grep`	search for a string of text
`rsync`	synchronize and update two directories (first with second)
`mount /mnt/zip`	Mount zip disk on directory /mnt/zip
`mount /mnt/cdrom`	Mount cd disk on directory /mnt/cdrom

5.2 Archiving Commands

`tar -xvzf`	Uncompress a file-name.tar.gz
`tar -xvf`	Uncompress a file-name.tar
`gunzip`	Uncompress a file-name.zip
`tar -cvzf`	Compress a file-name.tar.gz from directory-name /

5.3 Communications:

`finger`	look for all ccrma users
`finger login@`	look for particular user
`ssh`	secure login on remote machine
`scp`	secure file copy from or to machine
`talk login@cmn`	talk to other user on this terminal
`find`	look for file

5.4 Sound-software and CCRMA related:

`sndplay`	play a sound file
`sndinfo`	information about a sound file
`sox`	different audio file utilities
`play`	part of sox; basic reading and playing most soundfile formats
`flac123`	play a "FLAC" lossless compressed soundfile
`ogg123`	play a compressed audio "OGG" soundfile
`mpg123`	play a compressed audio "mp3" soundfile
`ecasound`	high quality audio recording, filtering and playback
`aplay`	command-line sound recorder and player for ALSA soundcard driver
`aplaymidi`	play Standard MIDI Files
`timidity`	play MIDI Files, MIDI-to-WAVE converter and player
`lame`	compress and create mp3 audio files
`oggenc`	encode audio into the Ogg Vorbis format
`flac`	lossless compress of audio using Free Lossless Audio Codec (flac)
`cdrecord`	burn "Wav" tracks in current directory
`cdparanoia`	copy cd tracks to a sound file in current directory
`resample`	change sample rate in a soundfile
`sndrecord`	record sound
`sndsine`	generate a sine signal

5.5 Text-Graphics and Desktop Publishing

`vi`	text editor
`emacs`	development text editor
`latex`	compile a TeX file
`dvips`	convert a dvi file to postscript

5.6 Shell-printing printing by a shell command

`lpr`	print a document
`lpr -P np2`	print a document in the trailer
`psnup`	resize to 2 or more pages in one
`psresize`	change size of a ps file
`ps2pdf`	convert a ps file into a pdf file
`pdf2ps`	convert a pdf file into a ps file

More on printing commands on the typesetting and shell commands sections: §9.9, §9.4

5.7 SysAd-Commands: system administration commands

`chmod`	change permissions on files and directories
`yppasswd`	change password

5.8 The-X-Window System

Another important component of Linux is its graphical user interface, The X Window system. Unix was originally mouse less, text-based system that used noisy teletype machines rather than modern

CRT monitors. The Unix command interface is very sophisticated and, even today, some power users prefer it to a point-and-click graphical environment, using their monitor as though it were a noiseless Teletype. Nevertheless Unix now provides users a choice of graphical or command interfaces.

X is a unique graphical user interface in two major respects. First, X integrates with a computer network, letting users access local and remote applications. Second, X lets you configure its look and feel to an amazing degree. To do so you run a special application-called a window manager- on top of X.

All the windows on the screen, keyboard and mouse user interface on Unix systems are possible thanks to the X windows server. Must applications which run straight on an X window start with x or capital X.

Although you can run X windows by itself using another layer call desktop can help you work more efficiently. A desktop is a set of desktop tools and applications such as Netscape, Real Audio, and accessories such as note pads, calculators games, or dictionaries. You can choose window managers depending on your desktop. GNOME and KDE are some of the desktops managers used at CCRMA. Sawfish is the default window manager in GNOME and you can change the feel and looks of your windows and backgrounds by using the configuration tools of your desktop of choice.

5.9 Popular-Xwindow applications at CCRMA

`xemacs`,	X window and improved interface to emacs http://www.xemacs.org/
`xfig`,	Draw and manipulate objects interactively http://www.xfig.org/
`gv`,	An X user interface for GhostScript http://wwwthep.physik.uni-mainz.de/ plass/gv/
`xcircuit`,	Drawing program tailored to circuits http://xcircuit.ece.jhu.edu/
`xmms`,	X multimedia system http://www.xmms.org/
`xmixer`,	Mixer to control volume settings n/a
`xsane`,	User-interface to control an image acquisition device http://www.xsane.org/
`xcdroast`,	Front end for programs like cdrecord and mkisofs http://www.xcdroast.org/
`xpdf`,	Viewer for Portable Document Format (PDF) files n/a

6 GNOME: http://www.gnome.org/

As with most GNU programs, GNOME has been designed to run on all modern strains of Unix-like operating systems.

The initials stand for GNU Network Object Model Environment but this doesn't really help explain what it does, though. GNOME is a part of the GNU Project and an attempt to make a desktop environment which is free software and which runs on lots of platforms is consistent so that you use the same approach to do the same tasks in different programs, can be used by developers to develop software easily and is fun to use. GNOME can be used with several window managers including Sawfish http://sawmill.sourceforge.net/, Enlightment, http://www.enlightenment.org and Window Maker which allows a Next Step look like environment.

One of GNOME's most interesting features is session awareness. When you re-enter GNOME, it reconfigures your desktop to match the state at the time you exited. GNOME even restores each application to its former state. GNOME provides desktop tools including games, calendar, address book, etc, plus GIMP, the GNU Image manipulation program, office applications plus many more. Explore these application by clicking and holding the GNOME (footprint) icon. Once you are logged in.

The GNOME project acts as an umbrella, the major components of GNOME are:

- The GNOME desktop: an easy to use windows-based environment for users.
- The GNOME development platform: a rich collection of tools, libraries, and components to develop powerful applications on Unix.
- The GNOME Office: A set of office productivity applications.

GNOME is a large collection of software, created over the last years. It ranges in scope from small utilities to large, powerful systems, and from low-level development libraries to end-user applications. So what is it that all parts of GNOME have in common?

- The GNOME project was the first to provide a fully free desktop environment for Unix-like systems. Free Software is about empowering users, and about granting them rights over the software they use.
- The GNOME Usability Project aims to improve the ease-of-use of GNOME and make the GNOME experience as enjoyable and natural as possible.
- GNOME is chock-full of cutting edge technologies. Network transparent component technology using CORBA, extensive use of XML, and one of the most advanced imaging models on any platform are only some of the features that makes GNOME the closest to rocket science you're likely to run on your desktop. In addition, it's all implemented in extremely efficient C, which makes it fast, lean, and very portable.
- The GNOME developer community is vast, tightly knit, and very friendly. If you're a developer who wants to get started on modern Unix GUI applications, GNOME is the emerging standard, and the large amount of developer documentation and other resources will help you find your way quickly. The GNOME libraries and infrastructure take care of most of the boring work for you, and let you focus on the code that makes your application unique. For more developer information, go to the GNOME Developer's Site.
- GNOME is used, developed and documented in hundreds of countries across the globe, and with the new GNOME internationalization features, GNOME lets you work in your language, no matter if it's Japanese, Russian, Swahili or English, complete with documentation, help, and menus in your native language.
- Accessibility is about enabling people with disabilities to participate in substantial life activities that include work and the use of services, products, and information. The GNOME Accessibility Project is developing a suite of software services and support in GNOME that allows people with disabilities to utilize all of the functionality of the GNOME user environment.

Even though it's extremely user-friendly, GNOME is a large and complex system, and as such, requires some learning to utilize to the fullest. To make that easier, we've provided some pointers to useful documentation.

The first resource you should look at is the GNOME User's Guide. It contains a wealth of useful information for the novice and experienced GNOME user alike. If you'd like, you can download the HTML or PDF version from: http://www.gnome.org/

7 KDE http://www.kde.org/

KDE is a powerful Open Source graphical desktop environment for Unix workstations. It combines ease of use, contemporary functionality, and outstanding graphical design. KDE (the K Desktop Environment) includes KWM, the K Window Manager, as an integral component and provides a file manager, a help system, a configuration utility and a variety of accessories and applications, including: games such as Kmines, Kpoker, and Ktetris, graphical applications such as Kfract, a fractal generator and Kview, an image viewer, multimedia applications such as Kmix , a sound mixer and Kmedia, a media player. For more information point go to the KDE home page and if you have a question make sure you go through their FAQs or frequently asked questions section.

8 ALSA http://www.alsa-project.org/

is the Advanced Linux Architecture. It consists of a system of device drivers for different sound-cards, a library API, utilities and of course programs that run on top of it. In general it is a set of software tools for manipulating high quality audio and MIDI inside the computer and the external world. ALSA has been in development for the past years and because of its "state of art" it was recently adopted as part of the standard Linus Torwalds Kernel.

Given this state of excitement you might get interested in getting into this group of developers and improve on existing drivers or even write your own driver for your favorite sound-card using the ALSA API. Otherwise you might get tempted on developing your own ALSA audio application and added to a growing collection of programs which include a full feature sound editor, MIDI sequencers, software synthesizers and certainly, algorithmic composition packages.

Extensive documentation can be found at various places including the PlanetCCRMA@Home page, the ALSA home page as well as the Agnula Project home page. A detailed description is a bit beyond the scope of this guide but the reader is encouraged also to read the mail archives of the planetCCRMA list as well as various Linux Audio lists and of course the ALSA users or developers list.

At CCRMA ALSA runs smoothly in the background and in theory all your favorite applications should run without much knowledge of it. However minimal information might prove helpful in case of troubleshooting audio in Linux. One of this situations might arise if for instance you are trying to plug in a MIDI-USB device to one of CCRMA's workstations (if this is the case please read the section on USB further on below and consult the USB devices section of PlanetCCRMA@Home).

9 Applications

Following are descriptions in their original form found in the documentation of Linux software available at CCRMA. Wherever possible hints or examples of their use will be provided but the reader is encouraged to further enhance his or her knowledge with the full documentation either on man pages or on the web links provided here. Some of this sites have very complete and operational descriptions in addition to to FAQs or frequently asked questions and even user's groups and mailing lists. Most of the following applications have been tested and are working, however there is a slight chance that they might not work accordingly. Don't panic, most of the time is not your mistake but instead a configuration error. Kindly let know anyone at CCRMA staff of your problem or email local-users@ccrma.stanford.edu. There is a big chance that someone else at CCRMA had the same problem or better of that knows a solution to your problem.

9.1 Terminal "Window"

- GNOME Terminal
  http://www.gnome.org

  This is perhaps the most important item on your desktop. It is your window to Unix and the rest of the world. You can open it by clicking on the footprint icon on the lower left GNOME panel. The GNOME Terminal application starts the default shell(command line interpreter) for you. It can be configured to different colors, backgrounds, or sizes. Make sure you set the scrolling option to a value bigger than 100. Character sets, fonts can also be adjusted but keep in mind that most Unix commands are in English with ASCII characters. However, in special cases like when you are using pine or text editors like Vi or Joe and you want to type non English characters, for western languages use the ISO-8859-1 character set which includes special characters for languages like Spanish, French, Portuguese German or Italian. You can open as many terminals as necessary. At CCRMA we use the C-Shell as default for the terminal window.

  Craig Sapp suggests a nice collection of tutorials of the shell which can be found at:

  LinuxCommand.org
  http://www.linuxcommand.org/learning_the_shell.php

  Manual pages can also be found at:

  SuperMan pages
  http://www.linuxcommand.org/superman_pages.php

9.2 Electronic Mail - email

- Pine
  http://www.washington.edu/pine This is a Program for Internet News & Email - is a tool for reading, sending, and managing electronic messages. Pine was designed by the Office of Computing & Communications at the University of Washington specifically with novice computer users in mind, but it can be tailored to accommodate the needs of "power users" as well. Versions are available for various flavors of Unix as well as for personal computers running a Microsoft operating system.

- evolution
  http://www.ximian.com/devzone/projects/evolution-devel.html Evolution is the GNOME mailer, calendar, contact manager and communications tool.Perhaps NeXT mailer look-like, although some people might claim is Eudora-like if not others. Evolution represents the next step forward in GNOME applications. The tools which make up Evolution are tightly integrated with one another and act as a seamless personal information-management tool. Evolution is extensible and it will be possible to use it to solve a huge variety of information-sharing problems like contact makers, schedules, mailing lists and more. Evolution will import email and email addresses and contacts from older email clients like pine or Netscape.

  To setup evolution for the first time just type:

  ```
  evolution
  ```

  and follow the dialogs. The information you need to know as of now is your cram login when you get the email account information dialog. Make sure you type your account with the ccrma.stanford.EDU domain as follows:

  ```
  your-login@ccrma.stanford.edu
  ```

For receiving email select the mail server type as “Local Delivery”. Make sure your /var/spool/mail/ - directory is on the configuration path.

For sending email make sure you select the *“SMTP”* mail server type. In the Host section type *“localhost”*.

If you previously used pine or any other mail client at CCRMA it will ask you if you want to import your mailboxes and your email addresses. Just choose the appropriate selection.

Evolution has an extensive list of tool-tips, on-line help as well as FAQs and tips.

To export contacts (email-addresses) in Evolution Left-click on the very first contact, now hold down Shift and left-click on the very last contact. Now right click and select ”Save as VCard”. You’ll get a nice, 100% standard-compliant .vcf (palm compatibe) file which can be read in other address-book programs and even sync to hardware palms.

For Mailboxes also Left-click first, Shift-left-click last, Right-click Save As. It saves as standard mbox file with an .mbox extension which might be exported to other operating systems mail clients.

When in doubt in Evolution, highlight and right-click!

- Evolution: More than an Email Reader

  You can filter or rather direct your email to different folders. While procmail and spamassasin can filter unwanted spam mail (see below), you can still filter way more by using the filter utilities in Evolution. But let’s start on how to redirect email to a specific folder:

  Suppose you want to archive email from a particular list you are subscribed, let’s say the PlanetCCRMA list.

  1. Create a PlanetCCRMA folder
  2. Go to the Tools menu and select “Filters”
  3. On the filters window click “Add”
  4. Name the rule something like “planetcrma mailing list”
  5. On execute actions select “if all criteria met”
  6. Add action if there is not one
  7. Select the mailing list criteria
  8. Then on the whitespace write “planetccrma@ccrma.stanford.edu”
  9. On the “then” side below of the rule box select action: “move to folder”
  10. and then select “PlanetCCRMA in local folders”
  11. Email from this list should be redirected to the PlanetCCRMA folder the next time you receive email.

  It would be a good idea to take a look at different criteria in t he *rules* window for familiarizing with different filters if you want to classify your email and even if you want to have your own spam filters. Most email filters are created the way explained above just by changing rules and actions.

- Evolution Contacts

  Working with contacts is also a very useful feature. Contacts not only file email address information, they also can store telephones, fax numbers, birthday information, you name it. Each contact can act like a card and as your phone book directory. In fact each contact can be a Vcard and exported to contact programs in other operating systems’ software. You can attach a Vcard and send it over the Internet but perhaps the most useful feature is that you can synchronize your “palm pilot” with Evolution’s contacts.

- Synchronizing your Palm-Pilot

  Evolution has a built-in script which runs “gnome-pilot” to synchronize information to and from your pilot just by following these steps:

1. Select your Palm ID
2. Select the conduits of information you want to have between your desktop and the pilot.
3. Share calendar data, ToDo's, etc.

Whether you have a serial Palm or a USB you should not run into much trouble while trying the *HotSync* push button on your device. If synchronization is not working make sure that Pilot Link is properly configured. You will need to make sure that you have read and write permissions on the device, which is normally /dev/pilot. If that does not work, check /dev/ttyS0 (maybe /dev/ttyS1) if you have a serial connection, or /dev/ttyUSB0 for a USB connection. You can do this by asking your system administrator for the right permissions or in your own system by becoming root and running the command: chmod 777 /dev/ttyUSB0 (or chmod 777 /dev/ttyS0).

– Evolution Composer: Emacs-like key bindings

You can have Evolution's composer (compose email window) Emacs-like key bindings. On older versions there is a tab in tools/ settings/ composer and even in the composer window menus over preferences. In newer versions this setup is done via your Gnome editor preferences. For this go to keyboard shortcuts on the Gnome-preferences menu and on the Text editing shortcuts tab select "emacs".

- Mozilla
  http://www.mozilla.org , is an open-source web browser, designed for standards compliance, performance and portability and is the browser of choice at CCRMA. You can also read email, news and transfer files plus it offers a very good multilingual interface. It works in a similar fashion or better than Netscape or eXplorer.

- Procmail
  http://www.procmail.org

  CCRMA is now processing email at the server level and the final delivery agent is *procmail.*

  Procmail can be used to create mailing lists, sort your incoming mail into separate folders/files (real convenient when subscribing to one or more mailing lists or for prioritising your mail), preprocess your mail, start any programs upon mail arrival (e.g. to generate different chimes on your workstation for different types of mail) or selectively forward certain incoming mail automaticallyto someone.

  Procmail should be invoked automatically over the .forward file mechanism as soon as mail arrives. Alternatively, when installed by a system administrator (and in the standard Red Hat Linux configuration), it can be invoked from within the mailer immediately. When invoked, it first sets some environment variables to default values, reads the mail message from stdin until an EOF, separates the body from the header, and then, if no command line arguments are present, it starts to look for a file named $HOME/.procmailrc. According to the processing recipes in this file, the mail message that just arrived gets distributed into the right folder (and more). If no rcfile is found, or processing of the rcfile falls off the end, procmail will store the mail in the default system mailbox.

  You're not supposed to start procmail from the command line. Procmail expects exactly one mail message to be presented to it on its stdin. Usually the mail system feeds it into procmail.

  There exists an excellent FAQ about Emailfilters (and procmail in particular), maintained by Nancy McGough which can be obtained and seen at: http://www.cis.ohio-state.edu/hypertext/faq/usenet/mail/filtering-faq/faq.html

- SpamAssassin
  http://spamassassin.taint.org

  If you are getting a lot of spam mail:

  SpamAssassin is a mail filter to identify spam.

Using its rule base, it uses a wide range of heuristic tests on mail headers and body text to identify ”spam”, also known as unsolicited commercial email.

The spam-identification tactics used include:

– header analysis: spammers use a number of tricks to mask their identities, fool you into thinking they’ve sent a valid mail, or fool you into thinking you must have subscribed at some stage. SpamAssassin tries to spotthese.
– text analysis: again, spam mails often have a characteristic style (to put it politely), and some characteristic disclaimers and CYA text. SpamAssassin can spot these,too.
– blacklists: SpamAssassin supports many useful existing blacklists, such as mail-abuse.org, ordb.org or others.
– Razor: Vipul’s Razor is a collaborative spam-tracking database, which works by taking a signature of spam messages. Since spam typically operates by sending an identical message to hundreds of people, Razor short-circuits this by allowing the first person to receive a spam to add it to the database – at which point everyone else will automatically block it.

Once identified, the mail can then be optionally tagged as spam for later filtering using the user’s own mail user-agent application.

SpamAssassin requires very little configuration; you do not need to continually update it with details of your mail accounts, mailing list memberships, etc. It accomplishes filtering without this knowledge, as much as possible.

So, SpamAssassin can be activated on your account by adding a filtering recipe to your ”.procmailrc” file, if you have one, or creating one if you don’t. Beware, you can lose email if you don’t configure your .procmail file properly.

Please contact your system’s administrator or staff for getting a current and correct procmail configuration and make sure you understand your regexps (regular expressions) and filters.

An example of a “$HOME/.procmailrc” configured to use SpamAssassin and which you can tailor to your specific requirements might be something like (procmail man page has more details on the syntax and structure of this file):

```
# SpamAssassin sample procmailrc
#
# Pipe the mail through spamassassin (replace ’spamassassin’ with ’spamc’
# if you use the spamc/spamd combination)
#
# The condition line ensures that only messages smaller than 250 kB
# (250 * 1024 = 256000 bytes) are processed by SpamAssassin. Most spam
# isn’t bigger than a few k and working with big messages can bring
# SpamAssassin to its knees.
#
# The lock file ensures that only 1 spamassassin invocation happens
# at 1 time, to keep the load down.
#
:0fw: spamassassin.lock
* < 256000
| spamassassin

# Mails with a score of 15 or higher are almost certainly spam (with 0.05%
# false positives according to rules/STATISTICS.txt). Let’s put them in a
# different mbox. (This one is optional.)
:0:
* ^X-Spam-Level: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
almost-certainly-spam

# All mail tagged as spam (eg. with a score higher than the set threshold)
# is moved to "probably-spam".
:0:
* ^X-Spam-Status: Yes
probably-spam

# Work around procmail bug: any output on stderr will cause the "F" in "From"
# to be dropped.  This will re-add it.
# NOTE: This is probably NOT needed in recent versions of procmail
:0
* ^^rom[ ]
{
  LOG="*** Dropped F off From_ header! Fixing up. "

  :0 fhw
  | sed -e '1s/^/F/'
}
```

- Using SpamAssassin inside Evolution:

  You can also use SpamAssassin directly inside Evolution. For this you will need to create a filter in order to process the message and then place it inside a "spam folder". Following are the steps needed while in evolution:

  1. Create a Spam Folder
  2. Go to the menu Tools —¿ and look for Filters
  3. Under Filter Rules, choose the "Add" button
  4. look for the "Add Rule" dialog and,
     (a) Under the Criterion, "IF," frame,
        i. Choose a rule name, something like "spam-filter"
        ii. Choose "Pipe Message to Shell Command" from the conditions list.
        iii. Write the command "spamassassin -e ¿ /dev/null" in the command text box.
        iv. Select "returns greater than" from the return drop down list.
        v. Set 0 to the number box.
     (b) Under the Action, "THEN," frame,
        i. Choose the "Move" action and set the destination to your new spam folder.
        ii. Click on the "Add Action" button.
        iii. Choose "Stop processing" from the new action list.
     (c) Move the new "spam rule" to the top of the filter list because Evolution performs filter rules sequentially.
  5. *Some known issues:*

     It is advisable to do email backups and testing before using a spam filter. Sometimes some messages will get trapped inside the spam folder. If this kind of situation repeats you might want to add another rule to your spam filter to fix the problem. Therefore, periodically reviewing your spam folder for legitimate mail is a good idea.

Be aware that no matter how fast your computer is, adding a spam filter might slow down your mail download from the server significantly. However, depending on the volume of mail you receive, adding this kind of filtering will significantly reduce dealing with junk mail.

9.3 Web Browsing

There are several Linux Web browsers each one wit its own very appealing features for browsing and file viewing. Most prominent candidates include Mozilla, Konqueror, Opera, and Netscape browsers. Out of curiosity you may want to know that can use any one of about 50 browsers in Linux, but of course many are experimental or immature. Alternatively you might want to be more conservative and choose Netscape which still available. No word about eXplorer in Linux.

All of the major browsers carry on exciting features that make browsing more pleasurable in ways only available to Linux and Unix users. For example, both Mozilla and Konqueror let you disable pop-up ads, and are smart on how they do it too. -The browser detects when the user clicks on a link that will open up in another window and allows that kind of action, while denying the ability of a web page to open its own window without user interaction.

Linux browsers pioneered true page scaling, allowing for the viewing of a web site as Small as a PDA screen up to resolutions so that a page can be comfortably read in a 1600x1200 pixel screen. This is a great feature in particular when you are projecting browser windows on a LCD projector.

- Mozilla
  http://www.mozilla.org , is an open-source web browser, designed for standards compliance, performance and portability and is the browser of choice at CCRMA. You can also read email, news and transfer files plus it offers a very good multilingual interface. It works in a similar fashion or better than Netscape or eXplorer.

  Mozilla is probably the most advanced Linux browser, considering the way that Red Hat has configured it in Red Hat 7.3. Everything works and works well. You can bet that most sites will work with Mozilla.

- Firefox
  http://www.mozilla.org/products/firefox/ , is the newest generation browsing based on technology developed over the years for Mozilla and Netscape.

  It is small, fast and easy to use and offers many advantages such as the ability to block pop-up windows and tab browsing. Firefox lets you to accomplish your on line activities faster, more safely and efficiently because it is light. The built-in Google bar provides convenient access to the best search engine on the web. With Tab browsing, there is a fast and convenient way to browse the web. This means you can open several pages in one window in separate browser tabs. Open links in the background while you read a web page, then continue to the links when you're done. Pages are available when you need them, making the web feel faster even over slow connections. Firefox shields you from unwanted popup windows and lets you allow certain sites to open popup windows ifnecessary.

- Konkeror
  http://www.konkeror.org , is an Open Source web browser based for the K Desktop Environment with features such as HTML4.0 compliance,Java applets support, JavaScript, CSS1 and (partially) CSS2, as well as Netscape plugins (for example, Flash or RealVideo plugins).

  Konqueror might also serve as the file manager for the K Desktop Environment. It supports basic file management on local UNIX filesystems, from simple cut/copy and paste operations to advanced remote and local network file browsing.

  Among its features if you where and know how to look for them, you can browse an Audio CD in your machine, and automatically rip tracks from CDs while surfing them.

- Amaya
  http://www.w3.org/Amaya/ , is a Web editor, (i.e. a tool used to create and update documents directly on the Web). Browsing features are seamlessly integrated with the editing and remote access features in a uniform environment. This follows the original vision of the Web as a space for collaboration and not just a one-way publishing medium.

  Work on Amaya started at W3C in 1996 to showcase Web technologies in a fully-featured Web client. The main motivation for developing Amaya was to provide a framework that can integrate as many W3C technologies as possible. It is used to demonstrate these technologies in action while taking advantage of their combination in a single, consistent environment.

  Amaya started as an HTML + CSS style sheets editor. Since that time it was extended to support XML and an increasing number of XML applications such as the XHTML family, MathML, and SVG. It allows all those vocabularies to be edited simultaneously in compound documents.

- Galeon
  http://galeon.sourceforge.net/ , is the Web browser created by Gnome and it is based on gecko (the Mozilla rendering engine). It's fast, it has a light interface and it is full standards compliant.

  It is appealing because of its slender design and speed. The icons on the browser itself are small and cleanly designed, and for some people look better than other cartoony buttons found on most Web browsers. The buttons take up little of the space that one would rather use for browsing. Pages load quickly and almost flawlessly, although some standard fonts looked a little less clear than one might like.

- Netscape
  http://www.netscape.com This is a Mail, Calendar, Instant Messenger, Search and web surfer application but it is not longer part of most Linux distributions.

9.3.1 Customizing Mozilla

Skins:
Dress Mozilla up with new appearances!. Yes, you can download new skins and install them so that next time you start Mozilla, it will have a different look.

Mozilla comes standard with two themes, Classic and Modern. These themes were designed for an enjoyable browsing experience but soon developers realized that people might desire to dress-up their browser to meet their own particular tastes and thus provided a means for independent designers to develop their own themes. You can take advantage of new on the edge state of the art themes to tailor your taste. To install new themes on Mozilla:

1. Before installing the themes, launch Mozilla.
2. JavaScript has to be enabled for Navigator in advanced preferences.
3. Software Installation has to be enabled in advanced preferences.
4. Themes can be automatically installed from the http://themes.mozdev.org/ site.
5. Alternatively, themes are supplied as xpi files; meaning they can be downloaded and installed later.

Tabbed Browsing:
Tabbed browsing gives you a better way to surf the net. You no longer have to open one page at a time. With tabbed browsing, open several pages at once with one click. And now your homepage can be multiple tabbed pages.

1. Ctrl-T opens a new tab.

2. Ctrl-PgUp and Ctrl-PgDn cycle between tabs.
3. Right-click on a link to Open Link In New Tab.
4. Save a window of tabs as a group bookmark by going to Bookmarks, File Bookmark and checking "File as group" when you go to the bookmark, all the tabs will open at once.
5. Set tabbed browsing preferences in Edit / Preferences / Navigator / Tabbed Browsing.

Pop up window blocking:
You can configure Mozilla so pop-up windows will not open. Popup blocker lets you surf the web without intrusion. Advanced popup blocker notifies you when popups are blocked. You can also block pop-ups on a site per site basis.

1. Open the Mozilla Preferences window.
2. Select the Advanced Category and then Scripts & Plugins.
3. In this window, uncheck (disable) the Open Unrequested Windows.
4. Disable Move or Resize Existing Windows.
5. Uncheck Raise or Lower Windows.

Plug-Ins:
If installing plug-ins without root permissions, use $home/.mozilla/plugins instead of the plug-ins subdirectory where Mozilla is installed. You may have to create this directory first. Mozilla Firefox also uses $home/.mozilla/plugins for this. Java and Flash are already installed system-wide at CCRMA. You should not need to re-install any plug-in. If that is the case please contact the system administrator. You can go to the

plugindoc.mozdev.org
http://plugindoc.mozdev.org/linux.html web page for instructions and additional support about Mozilla plug-ins.

Privoxy:
The privoxy daemon comes standard on most Linux distributions and chances are that it might be configured and running on your workstation. Privoxy can block annoying adds and animations but it can also prevent useful information on you browser so it must therefore be tested on favorite web sites so that it behaves accordingly.

> Privoxy is a web proxy with advanced filtering capabilities for pro- tecting privacy, modifying web page content, managing cookies, control- ling access, and removing ads, banners, pop-ups and other obnoxious Internet junk. Privoxy has a very flexible configuration and can be customized to suit individual needs and tastes. Privoxy has application for both stand-alone systems and multi-user networks.

These are the steps for using Privoxy in Mozilla:

1. In the Edit menu go to — Preferences — Advanced — Proxies —
2. Select "Manual Proxy configuration"
3. Type 127.0.0.1 on the "HTTP Proxy" space and 8118 on "Port" space.
4. Type 127.0.0.1 on the "SSL Proxy" space and 8118 on "Port" space.
5. After doing this, flush your browser's disk and memory caches to force a re-reading of all pages and to get rid of any ads that may be cached. You are now ready to start enjoying the benefits of using Privoxy!

Cookies:
A cookie is a small amount of information used by some web sites. A web site that sets cookies will ask your browser to place one or more cookies on your hard disk when you visit the site. Later, when you return to the site, your browser sends back the cookies that belong to the site. You can specify how cookies should be handled by setting your cookie preferences and by using the Cookie Manager. To change cookie preferences:

1. Open the Edit menu and choose Preferences.
2. Under the Privacy & Security category, choose Cookies. (If no subcategories are visible, double-click the category to expand the list.)
3. Click one of the radio buttons:
   - Disable cookies: Choose this to refuse all cookies.
   - Enable cookies for the originating web site only: Choose this if you don't want to accept or return Foreign cookies. Cookies received through email (when the message contains a web page) are treated as foreign cookies.
   - Enable all cookies: Choose this to permit all web sites to set cookies on your computer and receive them back during subsequent visits. Note: If you select this option, and later choose to reject all cookies, you may still have some older cookies stored on your computer (though no new ones will be set).

Viewing Cookies
To view detailed information about cookies:

1. Open the Tasks menu, choose Privacy & Security, and then choose Cookie Manager.
2. Choose View Stored Cookies from the submenu. The Cookie Manager window opens with a list of all the cookies stored on your computer.
3. To see details for a particular cookie, click on it:
   - *Name,* means the name assigned to the cookie by its originator.
   - *Information,* is a string of characters and the data a web site tracks for you. It might contain a user key or name by which you are identified to the web site, information about your interests, and so forth.
   - *Host or Domain*, this item tells you whether the cookie is a host cookie or a domain cookie.
   - *Path*, This is the file pathway. If a cookie comes from a particular part of a web site, instead of the main page, a path is given.
   - *Server Secure,* indicates whether the cookie was sent over a secure server. If a cookie is secure, it will only be sent over a secure (https) connection. Before sending a secure cookie, your browser checks the connection and will not send if the connection is not secure.
   - *Expires,* is the date and time at which the cookie is deactivated.

Using the Password Manager:
Password Manager can help you by storing your user names and passwords on your computer's hard disk, and entering them for you automatically when you visit such sites.

When you enter your user name and password at a web site a dialog box appears asking, "Do you want Password Manager to remember this logon?" You can choose the following options:

1. *Yes,* the next time you return to the web site you'll see that your user name and password are already filled in.
2. *Never for this site,* password Manager will not ask in the future if you want to save your user name and password for that site.

3. *No,* password Manager won't remember the user name and password, but will ask again the next time you visit the site.

Password Manager saves your user names and passwords on your own computer in a file that's difficult, but not impossible, for an intruder to read.

Becareful if you provide personal information such as your name, phone number, or email address to a web site, because it is free to store that information in its database and use it later. A web site might use this information to improve its service to you or target advertising to your interests. A web site could sell the information it has gathered to other companies.

To turn on encryption for your stored sensitive information:

1. Open the Edit menu and choose Preferences.
2. Under the Privacy & Security category, choose Web Passwords. (If no subcategories are visible, double-click the category to expand the list.)
3. In the Encrypting versus Obscuring section, select "Use encryption when storing sensitive data."
4. Click OK. If you haven't previously set a master password, a new dialog box appears and leads you through the process of setting it.

Useful Hints:

- Make Mozilla behave like Netscape:

  1. On the top [hhtp://] blank address space just type: [about:config]
  2. Scroll down to general and look for: "general.useragent.vendor"
  3. Change its value to: [netscape 7.0]
  4. Alternatively, you can look for the filter blank line and type: ["general.useragent"]
  5. Scroll down and look for: " general.useragent.vendor"
  6. If "general.useragent.vendor" is not in the list:
     – right-click and select new and type, [general.useragent.vendor]
     – Follow dialogs and add the value: [netscape 7.0]
     – click [ok]

9.4 Terminal shell commands

There is a lot of information inside the "manual" **man** pages for almost every Unix shell command. Also most commands suggested here should work on a (ssh) secure shell environment.

At CCRMA the shell of choice is the C-shell. There should be a `.cshrc` file in your home directory and familiarity with its meaning can be really helpful on getting acquainted with the behavior of your terminal window. You can write scripts or shortcuts which can be stored on your `.cshrc` file and therefore this means that you can have your own shell commands. You can find more information in the man pages or in
http://www.gnu.org/help/gethelp.html
or simply,
http://www.gnu.org/
A good source for information and tutorials on Unix commands can be found at,
http://www.geek-girl.com/Unixhelp/
If you have read the Unix section of this guide you will find the following commands 'very' useful:
- General and file manipulation commands:

man	Get information or help about a **command**
apropos	Locate commands by keyword look up
ls	List directory contents
pwd	Path of working directory
cd	Change working directory
mv	Move file or change name
cp	Copy file
mkdir	Make directory
rm	Remove files and directories
mount /mnt/zip	Mount zip disk on directory /mnt/zip

Commands for printing:

lpr file.ps	Print a file in default printer (knoll)
lpr -P np2 file.ps	Print a file in the trailer's printer
lpq	Show printer jobs in the queue
lprm	Remove the last job submitted if it is in the queue
lprm -np2 25	Remove job 25 in spool queue np2

WARNING !!! PLEASE, DO NOT use the *lpr* command to print Adobe Acrobat *"pdf'* files or postcript definition files. Open Acrobat Reader instead and print from within inside Acrobat Reader or use the command *pdf2ps* to convert to a "ps" postscript file.

- Postcript utilities:

As a wonderful ecological solution suggested by Julius Smith and to save some trees the following commands will let to print two pages in one from a postscript file. You type:

```
psnup -n 2 infile.ps > outfile.ps
psnup -pletter -n 2 infile.ps > outfile_2up.ps
```

To print an A4 formatted document you need to re size it to the American standard "letter-size". For this, you use the command **psresize** in the following way:

```
psresize -PA4 -pletter infile.ps > outfile.ps
```

- Synchronize a directory:

Suppose you have moved contents of one your local directories to CCRMA's "/zap" directory and make changes to one of the files in the "/zap" directory. To synchronize or update your changed file to the original file in the original directory, you use the `'rsync'` command as follows:

```
rsync -av /zap/dir/ ~/dir/
```

To update or synchronize just a file:

```
rsync -av /zap/dir/ ~/dir/yourfile
```

- Connectivity and file transfer and text manipulation:

`finger`	look for all ccrma users
`finger login@`	look for particular user
`ssh`	secure login on remote machine
`scp`	secure file copy from or to machine
`talk login@cmnXX`	talk to other user on this terminal
`find`	look for file
`cat`	output contents of a file
`grep`	search for a string of text

- Editors and publishing:

Command	Description
`vi`	text editor
`emacs`	development text editor
`latex`	compile a LaTeXfile
`dvips`	convert a dvi file to postscript
`ps2pdf`	convert a ps file into a pdf file

- Account Administration:

`chmod`	change permissions on files and directories
`yppasswd`	change password

9.5 Multimedia

Most of these programs can be accessed through the Gnome footprint print menu in the lower left corner of your Linux desktop. Mplayer can be started form the command line.

- XMMS, man-page, can play your favorite mp3 files but it can also play audio-stream radio stations with this feature. Just add the file name or the URL to its. play-list. URLs might be in a file with the .pls suffix like kcsm.pls and the file should be in the following format:

  ```
  [playlist]
  NumberOfEntries=1
  File1=http://hifi.kcsm.org:8002/
  ```

  The X Multimedia System is a player with the following features:

  1. Seeking in files Volume/Balance Shuffle play Repeat play
  2. Equalizer Playlist editor
  3. Spectrum Analyzer Oscilloscope
  4. User Interface: One line mode al'a WinShade in WinAmp Timer Elapsed/Timer Remaining Double Size option

     Full Winamp 2.0 skin support Gnome/Afterstep/WindowMaker dock app GTK Interface for requesters (with theme support) Auto remove borders if the WM has support for it Plugins:
  5. Visualization Effects Input Output

     Streaming/Shoutcast(1.0/1.1)/Icecast support Fast jump in playlist Scroll wheel support Save to wav option Saves http streams to HD HTTP authentication Plays mpeg layer 1/2/3 also wav and formats supported by mikmod Proxy authentication support Compiles on other systems (FreeBSD, Solaris, LinuxPPC, AIX, Irix)

- Mplayer
  http://www.mplayerhq.hu/homepage/

  You missed a class at Stanford and you want watch its video. Your friend just sent you a video clip of her birthday.

  Yes, you can watch it in your Linux workstation at CCRMA and Mplayer is your solution.

  "MPlayer is a movie player for LINUX (runs on many other Unices, and non-x86 CPUs, see the documentation). It plays most MPEG, VOB, AVI, VIVO, ASF/WMV, QT/MOV, FLI, NuppelVideo, yuv4mpeg, FILM, RoQ, OGG and some RealMedia files, supported by many native, XAnim, and

Win32 DLL codecs. You can watch VideoCD, SVCD, DVD, 3ivx, FLI, and even DivX movies too (and you don't need the avifile library at all!)."

You can launch the Mplayer application by typing something like (because you should read its man-page):

```
mplayer -vo xv yourfile &
```

or

```
mplayer -vo xv 'url of your videofile'
```

Well, you can also watch DVDs with Mplayer.

In some machines sampling rate playback is synchronized to the CD-Player / DVD-writer. If this is the case audio playback will be faster or slower. Please change the synchronization options with the *envy24control* command.

- xmixer
  http://www.hdk-berlin.de/ rasca/xmixer/

  This is a Gtk+ / Xaw-based soundcard mixer program for Linux.

- Cd Player
  http://www.gnome.org/

  A front end cd player for loading, playing, and skipping tracks of compact discs which are in the cd tray of your workststion.

- KDE Media Player
  http://www.kde.org/

  A MPEG file player application for KDE.

- Real Player is A MPEG file player application.

- KDE MIDI Player
  http://www.kde.org/

  A MIDI file player application for KDE. Files will be played on the soundcard internal synthesizer.

9.6 Word Processing, Spreadsheets, Presentations, Office Tools

9.6.1 Open-Office (OO)

OpenOffice
http://www.openoffice.org/ This is the real open source version of SUN's Star Office which is periodically updated at CCRMA. In modern RedHat versions you can run OpenOffice from Gnome's footprint main menu under applications. You can also run OpenOffice by invoking the "`openoffice`" command. If you are running the program for the first time, it will query some questions in regards to its license and it will also create an **OpenOffice.org** directory in your home directory.

OpenOffice.org is both an Open Source product and a project. The product is a multi-platform office productivity suite. It includes the key desktop applications, such as a word processor, spreadsheet, presentation manager, and drawing program, with a user interface and feature set similar to other office suites?. Sophisticated and flexible, OpenOffice.org also works transparently with a variety of file formats, including those of Microsoft Office.

StarDivision, the original author of the StarOffice suite of software, was founded in Germany in the mid-1980s. It was acquired by Sun Microsystems during the summer of 1999 and StarOffice 5.2 was released in

June of 2000. Future versions of StarOffice software, beginning with 6.0, will be built using the OpenOffice.org source, APIs, file formats, and reference implementation.

The OpenOffice.org source code initially includes the technology which Sun Microsystems has been developing for the future versions of StarOffice(TM) software. The source is written in C++ and delivers language-neutral and scriptable functionality, including Java(TM) APIs. This source technology introduces the next-stage architecture, allowing use of the suite as separate applications or as embedded components in other applications. Numerous other features are also present including XML-based file formats and other resources.

OO-Features The suite covers pretty much everything you need with a word processor, spreadsheet, presentation software, equation editor and a drawing program. It opens most major formats such as MS Office almost flawlessly (Macros aren't converted), saves to PDF* , has comprehensive help, and spellchecking in 15 languages. Having said that, let's detail some of the major features: The Entire OpenOffice Suite is a powerful yet simple environment for you to work. Allof the following features are used throughout the program.

- The ability to import and export many different types of documents and templates in a huge range of formats from HTML to MS Office XP. You can work seamlessly with your colleagues, school or friends, no matter what computer you use. Although we think the OpenOffice.org Open File Formats are technically superior many people are still locked into other products so you can set OpenOffice.org to use any other file format by defaultif you wish.
- Runs natively on almost every modern operating system except Mac OS X. This port is under development, and a volunteer porting team is working hard on it. You can give them a hand and get it running quicker by joining the project: http://mac.openoffice.org
- Import address books from Mozilla/ Netscape 6.x/7 or LDAP. You don't have to keep on typing in those contact details.
- Unicode support allows at least 23 localisations of OpenOffice.org. Not only will OpenOffice.org run on practically anything but it'll run in practically any language. There are more being added all the time. Even if yours isn't there yet we might be able to put you in touch will some fellow speakers to get the work done in double time. Together we work better.
- Vastly enhanced printing capabilities and options over all platforms, with extras like 'Print to PDF' under UNIX. If you want to see all that work in hard copy, OpenOffice.org will take care of it for you; or if you want to go for the 'paperless office' we can create 'digital paper' PDF files that can be read on practically anything. OpenOffice.org is all about bringing this sort of sophisticated, expensive functionality to the ordinary user.
- Autopilot guides you through creating complex document, clearly. Do you know what to do but aren't sure how to do it? Let the autopilot help. It'll ask you a few simple questions before creating a template for you to use again and again. Impress everyone with your brilliant skills, but don't let them know it was the autopilot!
- The Stylist allows you to change the entire look of your document instantly. It's that little box in the corner of the screen and it's an easy way to get a consistent look and apply formatting throughout your document. Just select some text, a cell or an object and choose a different option to move simply and easily between different styles. You can either use the built in defaults or create your own. And if it gets in your way just press F11 or click the little button near the top of your screen. You can get it back the same way or change styles in the box on the left hand side of the font selection.
- A native XML file format for small, yet powerful documents. File size is typically half that of MS Office formats. XML is the computer industry standard for data exchange. You're buying into the future and your documents will always be readable, even in a simple text editor.

- The powerful OpenOffice.org API allows you to create external functions to leverage the power of the suite just how you need it. Independent programmers can use our free software development kit (SDK) to extend the suite in ways we haven't even thought of. You can use OpenOffice.org Basic or Java and C/C++. More languages are being worked on at http://api.openoffice.org/.

OO-Writer Word Processing

- Form-letter management to send letters to addresses in a database. Automate chores and make complex things easy, it's the way we've tried to design OpenOffice.org and we think it shows through here.
- Compare changes and work collaboratively using the Versions system. If you need to work on a document together, but can't be at the same computer then OpenOffice.org's powerful 'Versioning' will come in handy. It lets you see what's changed, who's changed it and accept or reject the changes either individually or with a powerful series of filters. Currently versioning information doesn't export perfectly to some other formats.
- Direct connection with external email software to seamlessly send your documents worldwide. Cut down on the number of clicks and get your work into the hands of the people who need it.
- AutoCorrect word completion speeds up your writing in a non-obtrusive and powerful way. As you're typing it makes its best guess at what you mean to say. You can accept it by pressing enter or just keep on typing to ignore. If you make a mistake at any time just press Ctrl+Z for a quick and simple undo. Now I love this but I understand that these AutoCorrect features can drive some people up the wall. So there's a flexible menu where you can choose exactly what you want on or off. You won't find AutoCorrect word completion anywhere else, it's just another optional feature that makes OpenOffice.org unique.
- Indexing functions make moving around inside a document easy. A simple bibliography database makes it easy to store sources for essays.
- Sophisticated layout manager, with extensive Desktop Publishing (DTP) functionality that allows you to display everything you want simply and clearly.

OO-Calc: Spreadsheet

- You're also free to pull in data from corporate databases, and cross-tabulate, summarise, and manipulate with advanced DataPilot technology.
- Natural language formulas let you create formulas using words (e.g. "sales - costs").
- You can be your own spreadsheet expert thanks to templates with built-in functions, allowing you to concentrate on your real work.
- Scenario Manager allows "what if..." analysis at the touch of a button.
- Charting tools to visualise data in 2D or 3D.

OO-Draw: Drawing tool DRAW - from a quick sketch to a complex plan, DRAW gives you the tools to communicate with graphics and diagrams. It won't organise or edit your digital photo's as well as other programs but if you need to create a diagram, picture or illustration then you'll be glad to have it.

- FontWork allows you shape stunning 2D and 3D images from text for an extra effect.
- Quick and easy creation of complex 3D shapes. Sophisticated rendering let you create photo realistic images with your own texture, lighting effect, transparency, perspective, and so on.

- Bezier curves, for realistic smooth curves.
- Eyedropper to change any pixel colour into another.
- Raster tools and effects.
- Enjoy single click access to the objects you need for your diagrams.
- Arrange objects, rotate in two or three dimensions; the 3D controller puts spheres, rings, cubes, etc at your disposal.
- Smart connectors make short work of flowcharts, organisation charts, network diagrams etc.
- Import and Export graphics to and from all common formats (including BMP, GIF, JPEG, PNG, TIFF, and WMF).

OO-Impress: presentations IMPRESS is a truly outstanding tool for creating effective multimedia presentations. Your presentations will stand out with 2D and 3D clip art, special effects animation, and high-impact drawing tools.

- Exploits all the effects and graphics tools from Draw for the complete range of easy-to-use drawing and diagramming tools to spice up your presentation.
- AutoLayout makes sure your presentations look professional. Even quickly produced presentations look great, less time micromanaging bullet placement, more productivity.
- A complete range of Views are supported: Drawing / Outline / Slides / Notes / Handouts to meet all the needs of presenters and audiences.
- Slide show Animation and Effects bring your presentation to life.

9.6.2 StarOffice

StarOffice
http://www.sun.com/staroffice/

The program can be started from the command line by typing "soffice". If it is the first time you are running StarOffice you will be prompted to configure it (to be able to run, soffice needs to copy some files to your home directory). Just follow the prompts, you might want to put the local directory in, for example, Library/, so that it does not clutter your home directory. After you are done with the configuration you should be able to start the program by typing "`soffice`".

Staroffice is a huge program and rather slow to start, but quite complete. It has a decent word processor, spreadsheet, presentation editor, vector graphics editor and so on and so forth. It has some future because there's now an Open Source version (which at this point is not ready for everyday use), that means that you (hopefully) will not be locked into a proprietary document format.

For presentations in StarOffice follow the autopilot or the online help. You can start from templates also and to some extend you can import or export data to or from powerpoint.

9.6.3 LyX

LyX is a graphical front to the very capable LaTeX page description language. LaTeXis really a programming language that enables you to do almost anything you could think off, with the added advantage that the output of LaTeXis device independent. That is, no matter where you print it or how you view it, it will look exactly the same (within the hardware limitations of the printing or viewing device). But it is hard to learn. With Lyx you can easily access a subset of LaTeXby using a GUI. You will not see the exact rendering of the document in the Lyx window but it is very easy to render to dvi (the native output format of latex) or postscript. Lyx is described as a "WYSIWYM" editor (what you see is what you mean :-)

9.6.4 gnotepad

gnotepad
http://gnotepad.sourceforge.net/

It is an easy-to-use, yet fairly feature-rich, simple HTML and text editor. GNOME. gnotepad was designed to have as little bloat as possible, while still providing many of the common features found in a modern GUI-based text editor. It is still fairly light-weight, especially for the features it offers, and aims to remain that way. Additionally, gnotepad+ is not intended to be fancy, so it leverages its text editing capabilities on the GTK Text Widget. Hence, if you are looking for a programmer's editor, look for another text editor or help improve the GTK Text Widget.

9.6.5 Gnumeric

Gnumeric
http://www.gnome.org/

The Gnumeric spreadsheet which is part of the GNOME desktop environment and is intended to be a drop in replacement for commercial spreadsheets and it offers a customizable feel that attempts to minimize the costs and pain of transition. Gnumeric will import your existing Excel, 1-2-3, Applix, Sylk, XBase and Oleo files. You should be able to start the program by typing `"gnumeric"`. There is also good documentation online by clicking on the help menu.

9.6.6 AbiWord

AbiWord
http://www.abisource.com/

A lightweight word processor, can import most formats. Abiword's features include it's " look & feel ", the ability to format pages and paragraphs, a spell checker, an interactive rule, the integration of styles, the unlimited capacity to undo/redo , a find and replace function and the image insertion. It is also able to import documents from Microsoft Word 97 and rtf (Rich Text Format) and to save documents using Internet HTML format.

A document saved under AbiWord has an extension by default of * abw and is written in XML and thus in ASCII format. These files can then be read by any text editor but that does not mean that AbiWord is a XML editor.

AbiWord might be open through the gnome menu (footprint) –¿ programs –¿ Applications –¿ AbiWord or you might simply type the command:

```
AbiWord &
```

on the shell.

9.6.7 WordPerfect

WordPerfect
http://linux.corel.com/

Word Perfect 8 is available by typing `wp8` at the command line. As it is a proprietary software package with uncertain future support I would not recommend it at this point.

9.6.8 GnomeCard

Address Book - GnomeCard
http://www.gnome.org/

GnomeCard is the combination of an address book (contact manager) and an electronic business card manager. At the moment, the program shows two views: a tree view and a WYSIWYG view. From the tree view, the user can visualize every contact registered in the loaded file, with the possibility to expand the record to the desired detail, and will be able to perform actions, like DnD, dialing and sending e-mail. From the WYSIWYG view, the user will be able to edit the presentation of the business card, including photos and logos that may come with the card record, and view its contacts in a more comfortable way. You should be able to start this program by typing ”`gnomecard`” on the command line.

9.7 Text-Editors

- vi (Vim)

  Even though Vi, man-page (or its enhanced version Vim) is somewhat awkward to use at first, it enables fast, simple, and effective editing once you get the hang of it. A key concept in Vi is combining a certain action (delete, copy to buffer, capitalize, etc.) with a movement (go to line 25, go to end of document, go to next occurrence of “foo,” go to 2ND occurrence of character “x” in this line, etc.). The action is performed on all lines or characters between the current cursor position and the destination cursor position. Vi is extremely powerful in moving around within (or between) files—Vim in particular is excellent. You can jump to a specific line, to the line where you were before jumping to the current line, to the line in the middle of the screen, to the line where you just changed “foo” into “bar,” etc. You’ll never have to mess with arrow keys to move around within a file. VIM is an improved version of the editor ”vi”, one of the standard text editors on UNIX systems.

  For more information on Vi commands see section:§9.7.1.

- Emacs

  To quote the emacs, man-page manual: Emacs is the extensible, customizable, self-documenting real-time display editor. If this seems to be a bit of a mouthful, an easier explanation is Emacs is a text editor and more. At its core is an interpreter for Emacs Lisp (“elisp”, for short), a dialect of the Lisp programming language with extensions to support text editing. Some of the features of GNU Emacs include:

  – Content sensitive major modes for a wide variety of file types, from plain text to source code to HTML files.
  – Complete online documentation, including a tutorial for new users.
  – Highly extensible through the Emacs Lisp language.
  – Support for many languages and their scripts, including all the European “Latin” scripts, Russian, Greek, Japanese, Chinese, Korean, Thai, Vietnamese, Lao, Ethiopian, and some Indian scripts.
  – A large number of extensions which add other functionality. The GNU Emacs distribution includes many extensions; many others are available separately–even a web browser.

  Emacs is the editor of choice of many developers, and even is used for word processing and email. Yo can even compile and debug inside Emacs.

- XEmacs
  XEmacs, man-page is a highly customizable text editor and application development system. Learning XEmacs is a lifelong activity and even people who have used Emacs for years keep discovering new features. You should be able to start Xemacs by typing `xemacs` on the command shell. Almost all features of Emacs are supported in XEmacs (the ones that aren’t supported are generally implemented in a better way in XEmacs). XEmacs has an extensive interactive help facility, but assumes that you know how to manipulate XEmacs windows and buffers. CTRL-h enters the Help facility. Help Tutorial `(CTRL-h t)` requests an interactive tutorial which can teach beginners the fundamentals of XEmacs in a few minutes. Help Apropos (CTRL-h a) helps you find a command given its functionality, Help Key Binding `(CTRL-h k)` describes a given key sequence’s effect, and Help Func tion `(CTRL-h f)` describes

a given Lisp function specified by name. All of these help functions, and more, are available on the Help menu if you are using a window system.

One of the interesting features that makes XEmacs useful at ccrma is that it can run lisp as a subprocess in one of the buffers so that all emacs editing commands can be applied to the lisp expressions you are evaluating. For this (and other features) to work you need to have the proper incantations in a file in your home directory called ".emacs". This file takes care of initializing things properly for all supported programming, typing or text modes.

- gedit

  If Vi or Emacs seem not too friendly, you might try Gedit.

  gedit is a full-featured text editor for the GNOME desktop environment. You can use it to prepare simple notes and documents, or you can use some of its advanced features, making it your own software development environment.

  gedit is written in C and Python and built on the GTK+ toolkit. It is part of the core application suite of GNOME. It has syntax-highlighting for a wide variety of different programming languages (such as C, C++, Java, JavaScript, Python, etc.)

  Some among others features include:

  1. Print and print preview support
  2. Clipboard support (cut/copy/paste)
  3. Go to specific line
  4. Auto indentation
  5. Bracket matching
  6. Full support for internationalized text (UTF-8)
  7. Text wrapping
  8. Line numbers

- gnotepad
  http://gnotepad.sourceforge.net/

  Please see the word processing section§9.6.

9.7.1 Vi commands

To Invoke vi on the command line type: *'vi filename'*

Command mode and input (insert) mode

Vi starts in command mode. The positioning commands operate only while vi is in command mode. You switch vi to input mode by entering any one of several vi input commands. Once in input mode, any character you type is taken to be text and is added to the file. You cannot execute any commands until you exit input mode.To exit input mode, press the escape (Esc) key.

Format of vi commands: *'vi [count][command]'* (count repeats the effect of the command)

Below some commands to get you started:

- Input commands

a	Append after cursor
A	insert (append) at the end of the line
i	Insert before cursor
o	Open line below
O	Open line above
: r file	Import a file into the current file
: 12r file	Import a file into the current file after line 12

- Cursor movement during insert mode

ctrl-h	Back one character
ctrl-w	Back one word
ctrl-u	Back to beginning of insert

- Cursor motion (command-mode)

H	Upper left corner (home)
M	Middle line
L	Lower left corner
h	Back a character
j	Down a line
k	Up a line
ˆ	Beginning of line
$	End of line
l	Forward a character
w	One word forward
b	Back one word
fc	Find c
;	Repeat find (find next c)

- Command mode

k	Move one line upwards
l	Move one character to the right
h	Move one character to the left
w	Move one word to the right
W	Move one word to the right past punctuation
b	Move one word to the left
B	Move one word to the left past punctuation
e	Move to the end of the current word
1G	Move to the beginning of the file
H	Move to the top of the current screen
M	Move to the middle of the current screen
L	Move to the bottom of the current screen
Ctrl-G	Move to the last line in the file
Ctrl-F	Move one screen towards the end of the file
Ctrl-D	Move 1/2 screen towards the end of the file
Ctrl-B	Move one screen towards the beginning of the file
Ctrl-U	Move 1/2 screen towards the beginning of the file
Ctrl-L	Refresh the screen
5G	Move to line 5 of the file (5 can be any line number)

- Save command mode

: w	Write out the file to save changes
: w file	Write the file to named file
: wq	Save the file exit vi
: w!	Force save the file
: q!	Quit vi but don't save changes
ZZ	Same as : wq
: sh	Execute shell commands (¡ctrl¿d)

- Deletion commands

dd or ndd	Delete n lines to general buffer
d)	Delete to end of sentence
db	Delete previous word
D	Delete to end of line
x	Delete character

- Undo commands

u	Undo last change
U	Undo all changes on line

- Window motion

ctrl-g	Display current line number
ctrl-l	Redraw screen
?string	Search backward
/	Search forward
ctrl-b	Page backward
ctrl-f	Page forward
ctrl-u	Scroll up (half a screen)
ctrl-d	Scroll down (half a screen)
G	Go to last line
nG	Go to line n
: n	Go to line n

- Search command mode

string	Find text string forward
?string	Find text string backward
n	Find forward next string instance after a string search
N	Find backward next string instance after a string search
: g/O/s//r/g	Global Search and substitute replace (O=search object r=replace object)

9.7.2 Emacs Xemacs cheatsheet

The following emacs commands can make your life much simpler if you are editing with emacs or Xemacs. Remember that these are text editors and their world is very dependent on the computer keyboard. Yeah right, by now you shouldn't' be using the mouse at all.

[c-x means control-x or the keyboard combination *¡ctrl¿-¡x¿*] [m-x means meta-x and at CCRMA is the keyboard combination *¡esc¿-¡x¿*]

```
c-w:      cut
m-w:      copy
c-y:      yank (paste)
c-k:      kill (delete) line

c-a:      move cursor to line beginning
c-e:      move cursor to line end
c-v:      move cursor to middle of file
c-l:      position window so cursor is in the middle
c-f:      move cursor ahead character
c-b:      move cursor back one character

c-s:      search for char-string (word) forward
c-r       search for char-string (word) backward
c-g:      quit emacs command (mini-buffer)

c-x-3:    start two windows (vertical)
c-x-2:    start two windows (horizontal)
c-x-1:    go back to one window
```

```
c-[space-bar]:  mark set (start selection)

c-a:       beginning of line
c-e:       end of line
m-a:       beginning of paragraph
m-e:       end of paragraph
c-x-[:     goto beginning of file
c-x-]:     goto end of file

c-x (:     start defining macro
c-x ):     end defining macro
c-x e:     execute macro

c-x z:     repeat last command
    z:     repeat second to last command
c-x m-z:   repeat last compex command

m-[shift]\%: replace character string (word)

c-[space-bar] c-e:       select a line
m-a c-[space-bar] m-e:   select a paragraph (region)

m-x -> ispell-buffer:  spell check file
m-x -> ispell-region:  spell check paragraph
m-x -> latex-mode:     go to \LaTeX mode
```

Make sure you also have a backup of your .emacs file or emacs preferences file. It would be a good idea to open it and understand it. You can add to this file your own command definitions and combinations. Pay good attention to the following lines and make sure the mode you are intending to use is part of this list.

```
(mapcar #'(lambda (element)  ;; emacs modes
            (setq auto-mode-alist
     (pushnew element auto-mode-alist)))
        '(("\\.pike$"                . pike-mode)
          ("\\.pmod$"                . pike-mode)
          ("\\.ats$"                 . lisp-mode)
          ("\\.clm$"                 . lisp-mode)
          ("\\.cm$"                  . lisp-mode)
          ("\\.cmn$"                 . lisp-mode)
          ("\\.ins$"                 . lisp-mode)
          ("\\.cl$"                  . lisp-mode)
          ("\\.lisp$"                . lisp-mode)
          ("\\.scm$"                 . scheme-mode)
          ("\\.sh$"                  . sh-mode)
          ("\\.bashrc"               . sh-mode)
          ("\\.gnus"                 . emacs-lisp-mode)
          ("\\.c$"                   . c-mode)
          ("\\.h$"                   . c-mode)
          ("\\.l$"                   . c-mode)
          ("\\.y$"                   . c-mode)
          ("\\.awk$"                  . c-mode)
          ("\\.cc$"                   . c-mode)
          ("\\.x$"                    . c-mode)
          ("\\.m$"                    . matlab-mode)
          ("\\.s$"                    . asm-mode)
          ("\\.p$"                    . pascal-mode)
          ("\\.txt$"                  . text-mode)
          ("\\.text$"                 . text-mode)
          ("\\.tex$"                  . latex-mode)
          ("\\.sm$"                   . latex-mode)
          ("\\.sty$"                  . latex-mode)
          ("\\.bib$"                  . bibtex-mode)
          ("\\.el$"                   . emacs-lisp-mode)
          ("\\.tcl$"                  . tcl-mode)
          ("\\.java$"                 . java-mode)
          ("\\.html$"                 . html-mode)
          ("\\.ohtml$"                . html-mode)
          ("[]>:/]Makefile"           . makefile-mode)
          ("[]>:/]\\..*emacs"         . emacs-lisp-mode)
          ("\\.rfc$"                  . rfc-mode)
          ("\\.id$"                   . rfc-mode)
          ("\\.isl$"                  . isl-mode)
          ("\\.php?$"                . php-mode)
          ("\\.Xdefaults$"            . xrdb-mode)
          ("\\.Xenvironment$"         . xrdb-mode)
          ("\\.Xresources$"           . xrdb-mode)
          ("*.\\.ad$"                 . xrdb-mode)
          ("\\.sql$"                  . sql-mode)
          ("\\.tbl$"                  . sql-mode)
          ("\\.sp$''                   . sql-mode)
          )
        )
```

Emacs is smart enough to recognize the file extension when you are opening a file and automatically will load the necessary mode so that your text will be formatted according to the text-mode you are going to edit. You turn-on this feature by adding file extensions and their relevant mode.

Suppose you want to use matlab mode. You also need to add the following code to you .emacs file:

```
;--------------------;
; Enable MATLAB mode ;
; -------------------;
(autoload 'matlab-mode "matlab" "Matlab Editing Mode" t)
(add-to-list 'auto-mode-alist '("\\.m$" . matlab-mode))
(setq matlab-indent-function t)

(defun my-matlab-mode-hook ()
  (setq matlab-indent-function t)  ; if you want function bodies indented
  (setq fill-column 76)            ; where auto-fill should wrap
  (turn-on-auto-fill))
```

```
(setq matlab-mode-hook 'my-matlab-mode-hook)
(autoload 'matlab-shell "matlab" "Interactive Matlab mode." t)
```

Emacs is very good for editing documents. The next lines will add some functionality to emacs so that you will feel at home while editing files in Emacs, feel free to add this code to your dot-emacs file.

```
;:* Initialise aucTeX
;:*(require 'tex-site)
;:*=======================
;:* load TeX-toolbar
;:*(require 'tex-toolbar)
;:*=======================
;:* autoload style files
(setq TeX-auto-save t)
(setq TeX-parse-self t)
;:*=======================
;; The documentstyle command is usually near the beginning.
(setq-default TeX-auto-parse-length 200)
(setq LaTeX-default-options "letterpaper,12pt,english,normalheadings")
(setq-default TeX-master t)
;:*
(add-hook 'latex-mode-hook
          (function (lambda ()
                      (local-set-key (quote [228]) (quote "\"a"))
                      (local-set-key (quote [246]) (quote "\"o"))
                      (local-set-key (quote [252]) (quote "\"u"))
                      (local-set-key (quote [223]) (quote "\"s"))
                      (local-set-key (quote [196]) (quote "\"A"))
                      (local-set-key (quote [214]) (quote "\"O"))
                      (local-set-key (quote [220]) (quote "\"U"))

(setq ispell-extra-args '("-t"))       ; start ispell in TeX mode
                      )))
```

In general before buying any books related to the subject matter of text editing, you should run the emacs tutorial inside emacs which will last about an hour but you will graduate as an “EmacSoologist”.

9.8 Image Manipulation

- GIMP Gnu Image Manipulation Program
  http://www.gimp.org/

  The GIMP is the GNU Image Manipulation Program. It is a piece of software suitable for such tasks as photo retouching, image composition and image authoring. For all of you who know the terminology, a brief list of features and capabilities which GIMP provides might be:

  – Full suite of painting tools including brushes, a pencil, an airbrush, an ink tool and cloning.
  – Advanced scripting capabilities provided by a procedural database so you can call internal GIMP functions from external scripts, such as Perl-Fu(Perl scripts and Python-Fu(Python scripts).

- Multiple undo and redo, limited only by disk space.
- Transformation tools including rotate, scale, shear, and flip.
- File formats supported include Post script, JPEG, GIF, PNG, XPM, TIFF, TGA, MPEG, PCX, BMP, and many others.
- Selection tools including rectangular, elliptical, free, fuzzy, paths, and intelligent scissors.
- Plug-ins that allow for easy addition of new functions, new file formats, and new effect filters.
- Tile-based memory management so image size is limited only by available space.
- Sub-pixel sampling for all paint tools, allowing for high-quality anti-aliasing.
- Full Alpha channel (transparency) support.
- Layers and channels.

Once in GIMP you open a new image (file) by choosing "new" from the File menu in the toolbox window. You can also scan images with GIMP.

9.9 Typesetting

- LaTeX, man-page is a high-quality typesetting system, with features designed for the production of technical and scientific documentation. LaTeX is the de facto standard for the communication and publication of scientific documents.

  LaTeXis not a word processor! Instead, LaTeXencourages authors not to worry too much about the appearance of their documents, but to concentrate on getting the right content. It is based on the idea that it is better to leave document design to document designers, and to let authors get on with writing documents. LaTeXis based on Donald E. Knuth's TeX typesetting language. LaTeXwas first developed in 1985 by Leslie Lamport, and is now being maintained and developed by the LaTeX3 Project.

  LaTeX here at ccrma is installed as part of the teTeX distribution. Latex is highly recommended for papers and other documents where very high quality and portability are essential. The TeXsystem generates device independent output so that not matter where you view or print your document, it will always look the same (up to the available resolution of the hardware device you are using, of course). You do have to learn a language, but it is worth it. See section §9.6.3 for LyX, a gui frontend that you can use to reap most of the benefits of LaTeX without the steep learning curve.

  Some of the commands that are part of the teTeX packages:

`tex`	tex formats the interspersed text and commands contained in the named files and outputs a typesetter independent file (called DVI, which is short for DeVice Independent)
`latex`	latex enables the LaTeXset of TeXmacros and outputs a dvi file
`dvips`	converts a dvi file into postscript
`xdvi`	a very simple dvi file viewer

  To get a postscript file you might want to type following commands,

```
latex  yourfile.tex
...
...
...
dvips -o yourfile.ps yourfile.dvi
...
...
...
```

where yourfile is the name of your file. Also please note that the `latex` command produces a dvi file which need further to be translated to ps format or Acrobat's pdf's format. To get a pdf format you type:

```
dvipdf yourfile.dvi yourfile.pdf
```

A good LaTeXreference manual can also be found at

- LatexHelp
  http://www.emerson.emory.edu/services/latex/latex_toc.html

  To save some trees and as suggested by Julius Smith the following commands will let to print two pages in one from a ps file:

  ```
  psnup -n 2 infile.ps > outfile.ps
  psnup -pletter -n 2 infile.ps > outfile_2up.ps
  ```

  To print an A4 formatted document you need to re size to "letter-size" by using `psresize` as follows:

  ```
  psresize -PA4 -pletter infile.ps > outfile.ps
  ```

- Producing PDF Output using pdfLaTeX, man-page:

  The Portable Document Format (PDF) is a-page oriented format available for electronic documents. It is based on Postscript and combines fixed textual contents with hypertext elements, compression elements and security measures.

  Adobe (developers of Postscript) has made the "PDF" specification public and freely available so that anyone is entitled to write software to produce and to read files in this format. They have also made available a free reader program called "Acrobat Reader" (also known in Linux as the "acroread" command). Today most graphical and text processing programs are capable of generating PDF output.

  While having a TeX or LaTeX file there are several ways the user can take. The choice depends more on the kind of graphics being used. As we know LaTeX can handle a long list of graphic formats including "ps" and "eps". If LaTeX is your choice for generating your "pdf" files, first you need to compile the device independent file or "dvi" (see previous section) and then use the "dvipdf" command as follows:

  ```
  dvipdf yourfile.dvi yourfile.pdf
  ```

  The other more direct method is to use the "pdflatex" program to get a "PDF" file right away. When started, the program reads a configuration file "pdf.cfg" which can set many parameter such as page size, offsets, compression level, dvi or pdf output, and can specify the names of the font mapping files.

  "pdflatex" handling of graphics is different from LaTeX. For best results graphics should be imported as "PDF" also before running the "pdflatex" command. For instance if you have "eps" graphic files you can use the "epstopdf" to get a graphic "PDF" file like,

  ```
  epstopdf your_graph.eps
  ```

  If you are using *Xfig* §5.9 for diagrams or graphics you can export directly to "PDF" format. In some cases you might need to adjust your figure by using the LaTeXcommands such as scalebox and rotatebox to place the figure as desired.

  As a matter of choice, here is an example of a LaTeX- file heading for better handling of graphics in both programs by means of "latex" or "pdflatex" commands:

```
\documentclass[letterpaper,11pt]{article}
\usepackage{times}

 \ifx\pdftexversion\undefined
    \usepackage[dvips]{graphics}
 \else
    \usepackage[pdftex]{graphics}
 \fi
```

- LaTeX2HTML (see man-page), is a convertor written in Perl that converts LATEX documents to HTML. This way e.g. scientific papers - primarily typeset for printing - can be put on the Web for online viewing.

  Very very useful stuff. The CCRMA Planet guide itself is written in LaTeXand plublished on the web through latex2html. At ccrma the complete latex2html online documentation can be found at file:///usr/lib/latex2html/docs/manual/index.html

9.10 Presentation-Applications

There are several presentation packages available for Linux aside from commercial pointing software. You can do your presentations by using HTML itself. You can also try LaTeXslide feature or better try latex2HTML, §9.9. Additionally you can try Open Office Impress (see the office software section §9.6.1). However, there is an extension of the LaTeXslide class which can give you outstanding results with effects that will not distract your audience. This is the prosper class.

9.10.1 Using Prosper for Presentations

You can use the LaTeXprosper class to take advantage of the quality of LaTeXformatting with colorful presentation styles that can be used to obtain high-quality PDF files. These files are web-friendly and can be displayed in full resolution quality, full-screen mode with a feeling much better than other point presentation packages on any computer using PDF readers or acroread (more at §9.22).

The prosper class was developed by Frédéric Goualard and it makes sense when you want to reuse some material of an article written in LaTeXfor papers or slides. Here, you can find more examples and prosper documentation courtesy of the math department at the University of Colorado.

At ccrma prosper documentation can be found at file:////usr/share/doc/tetex-prosper-1.5

Following there is an example of a TeXfile for a presentation (which you can copy-and-paste to file to see results). Notice preamble and document class definitions.

```
\documentclass[letterpaper,contemporain,pdf,colorBG,slideColor]{prosper}
\hypersetup{pdfpagemode=FullScreen}
%
% process with these commands:
% latex filename.tex
% dvipdf filename.dvi
%
\title{VictorBook of the Fourier Transform}
\subtitle{A  Musician's Approach}
\author{Juan Reyes}
\email{juanig@CCRMA.Stanford.EDU}
\institution{CCRMA\\Center for Computer Research in Music and Acoustics}
```

```
\newcommand{\abs}[1]{\ensuremath{\left|#1\right|}}
\newcommand{\ejoT}{\ensuremath{e^{j\omega T}}}
\newcommand{\defined}{\ensuremath{\stackrel{{\scriptscriptstyle\Delta}}{=}}}

\begin{document}
\maketitle

\begin{slide}{Fourier Series}

Fourier series are defined as:

\[ f_{per}(\omega_0t) = a_0 + \sum_{m=1}^{\infty} b_m\cos(2\pi
m\omega_0t) + c_m \sin(2\pi m \omega_0t) \]

\begin{itemize}
\item where $a_0,$ is the D.C. component and
\item where the coefficients $b_m,$ and $c_m,$ represent
  magnitudes of sine and cosine components for each frequency
  $\omega_0t,$ respectively.
\end{itemize}
\end{slide}


\begin{slide}[Dissolve]{Fourier Transforms}

  \begin{itemize}
  \item By using Euler's identity and in terms of sines
    and cosines the Fourier Transform is defined as:
    %\vspace{0.125in}
    \[S(\omega_0) = \int_{-\infty}^{\infty} s(t)
    [\cos(2\pi\omega_0t) + j\sin(2\pi\omega_0t)] \ ,dt., \]

  \item The Inverse Fourier Transform is defined as:
    %\vspace{0.125in}
    \[s(t) = \int_{-\infty}^{\infty}  S(\omega_0)
    [\cos(2\pi\omega_0t) + j\sin(2\pi\omega_0t)] \ ,dt., \]
  \end{itemize}
\end{slide}


\begin{slide}{IDFT}

The \emph{ IDFT} in terms of its angular frequency $\omega,$
is defined as:

\[ y(n) \defined \frac{1}{N} \sum_{n=0}^{N-1} Y(k) e^{j\omega_k n},
k=0,1,2,..., N-1  \]

\vspace{0.125in}
where $y(n) \leftrightarrow Y(k),$ conform a pair of transforms.
```

```
    The term $k,$ is the frequency index and  $n,$ is the time index.

    \end{slide}
  \end{document}
```

Once you have your TEXfile to make a presentation using the*prosper class,* type these commands to compile and process the postscript files:

```
latex filename.tex
```

and,

```
dvipdf filename.dvi
```

You can use acroread for your presentation:

```
acroread filename.pdf &
```

9.11 Music-Typesetting

- CMN
  http://www-ccrma.stanford.edu/software/cmn/

  CMN (Common Music Notation) is a Common Lisp based music notation package. Bill Schottstaedt describes it as “a simple little hack” but that would be an understatement, to say the least. It outputs directly to postscript so that the rendering quality is only limited by the resolution of the viewing or printing hardware. It includes its own music postscript font. The CMN manual is available online at “http://www-ccrma.stanford.edu/software/cmn/cmn/cmn.html” .

  If you are at ccrma, start the cm-clm lisp image, type “(in-package ’cmn)” to make all cmn symbols available and enter the following score (extracted from the cmn manual):

```
(cmn (output-file "test.eps")(size 24)
  (system brace
    (staff treble (meter 6 8)
      (c4 e. tenuto) (d4 s) (ef4 e sf)
      (c4 e) (d4 s) (en4 s) (fs4 e (fingering 3)))
    (staff treble (meter 3 4)
      (c5 e. marcato) (d5 s bartok-pizzicato) (ef5 e)
      (c5 e staccato tenuto) (d5 s down-bow)
      (en5 s) (fs5 e)))
  (system bracket
    (staff bar bass (meter 6 16)
      (c4 e. wedge) (d4 s staccato)
      (ef4 e left-hand-pizzicato)
      (c4 e tenuto accent rfz) (d4 s mordent)
      (en4 s pp) (fs4 e fermata))))
```

  LilyPond prints beautiful sheet music. It produces music notation from a description file. Lilypond uses LATEXtypesetting language as the basic rendering engine so it shares all of its advantages.

  Here’s a very short example you can test. In your favorite text editor, enter the following input, and save the file as test.ly:

```
\score {
  \notes { c'4 e' g' }
}
```

To run LilyPond, you invoke ly2dvi to compile your LilyPond source file:

```
ly2dvi -P test.ly
```

The results of the ly2dvi run are two files, test.dvi and test.ps. The postscript file (test.ps) is the one you can print. You can view the postscript file using ghostview by typing “gv test.ps”. The dvi (device-independent) file (test.dvi) is the output of LaTeXand can be viewed by typing “xdvi test.dvi”.

9.12 JACK

http://jackit.sourceforge.net is a low-latency audio server. It can connect several client applications to an audio device, and allow them to share audio with each other. Clients can run as separate processes like normal applications, or within the JACK server as ”plug-ins”.

Traditionally it has been hard if not impossible to write audio applications that can share data with each other. In addition, configuring and managing audio interface hardware has often been one of the most complex aspects of writing audio software.

JACK allows applications to send and receive audio data to/from each other as well as the audio interface. There is difference in how an application sends or receives data regardless of whether it comes from another application or an audio interface. From the programmers point of view it is an API that provides a high level abstraction and removes the audio interface hardware from the picture and allows them to concentrate on the core functionality of their software.

9.12.1 Qjackctl

QjackCtl is a simple application to control JACK. It provides a simple GUI dialog for setting several JACK daemon parameters, which are properly saved between sessions, and a way control of the status of the audio server daemon. With time, this primordial interface has become richer by including a enhanced patchbay and connection control features.

Most Pro-Audio applications require Jack for low-latency audio processing. QjackCtl is a handy tool for controlling and showing JACK status. Make sure you get familiar with this app and in particular if you need to toggle with different sampling rates on your audio hardware.

9.12.2 Using-JACK

In order to obtain high quality audio and avoid dropouts, under-run or overrun errors during audio recording or playback it is advisable to run JACK. Since it is a Unix process which acquires top priority you should use it only for sound purposes. You normally start JACK in a terminal window by issuing the following command:

```
jackstart -d alsa -d hw -r 44100
```

Please notice that the `-r` follows the audio sampling rate. If you are working with CLM instruments you might want to use a lower sampling rate like `22050`. Provided your sound-card will have this sampling rate available (not in the case of synchronized cards like the Delta 1010's) you might want to start jack by issuing the following command:

```
jackstart -d alsa -d hw -r 22050
```

In the case of a real-time situation like a concert situation, you can give JACK priority over all the other Linux processes with this options in the jackstart command:

```
jackstart --realtime --driver=alsa
```

WARNING: When you are done with your audio or sound job please make sure to quit or stop the JACK process by issuing:

```
C-c [control-c] sequence
```

Alternatively you might want to use the Qjackctl GUI (graphical user interface) for starting and stopping JACK manually with a mouse click. Qjackctl is found at the sound & video tab of the main menu. See here.

9.13 SND

> *Snd* is a sound editor and a composition tool with features not often found on other sound editors and commercial music software. By pinpointing a historical evolution of computer music and research through the years, most ideas unfold attributes behind Snd's design and prove its functional goals. A time-line of accomplishments at CCRMA are furthermore plentiful justification as to why Snd's development has been steered the way it has.
>
> Most of *Snd* development and its state of the art can be traced to the evolution of Computer Music and all of its branches since 1960. *Snd* inherits from knowledge built at CCRMA from the days the center was located at the *Stanford Artificial Intelligence Laboratory (SAIL)*, up to now, running on portable computer machines as well as in major operating systems including Linux. Since September 1996, *Snd* has been Open Source, with many people including musicians and engineers, contributing to its development geared primarily by its author Bill Shottstaedt. *Snd* can be thought of as last steps taken in the development of Computer Music systems, regardless of being a non real time sequential interactive system.
>
> http://ccrma.stanford.edu/software/snd/snd/snd.html Documentation describes as a sound editor modeled loosely after Emacs and an old, sorely-missed PDP-10 sound editor named *Dpysnd.* It can accommodate any number of sounds each with any number of channels, and can be customized and extended using the Scheme language implementation also developed by Bill Shottstaedt at CCRMA. On Unix systems *Snd* is started by typing the `snd` command on a terminal window. `NOTE: On Linux you might need to start Jack ( Qjackctl ) before invoking Snd.`

.

To get started, go to the File menu, and open a sound file. To hear the sound, click the 'play' button. To see an fft, click the 'f' button on the left. The left mouse button is used for most pointing operations; the middle button pastes in the current selection; the right button brings up the SND pop up menu. Make sure you go through the help pull down menu to find more about all the features SND. In addition to this there is an enormous amount of information in the snd manual available online.

Start **Snd** by typing:

```
[cmn19] snd &
```

.

on the command line. If you have a sound you want to listen, you can type for instance,

```
[cmn19] snd /zap/dog.snd &
```

To play a sound just click on the play click-box. Alternatively playing a sound from the listener is also an alternative.

```
(play)
```

.

Getting used to the *listener* is not a bad idea because a lot of embedded *Snd's* features are available this way. On the listener you type Scheme syntax commands which usually comprise functions that perform some procedure inside Snd affecting your sounds or not. For instance let's say you want to open a file called "oboe.snd". On the listener you will type:

```
(open-sound "oboe.snd")
```

.

On the edit menu, option "edit header" will pop up sound file characteristics like sampling-rate, file-type, number of channels and others. On this example we are opening a sound file of type "snd" also known as Sun/NeXT ".au" and sampling rate 22050. Perhaps we want to change its sampling rate to 48000 and get a new more compatible ".wav" sound file. After opening your sound, on the listener you will type:

```
(src-sound  (/ 22050 48000))
```

The function *'src-sound'* does sample rate conversion. "`(/ 22050 48000)`" is a division operation between both sample rates to get the exact amount of rate conversion. To get a new file with the new sample rate and with ".wav" riff format, on the listener you will type:

```
(save-sound-as "new.wav"  :sample-type mus-l24int :header-type mus-riff)
```

Notice that "new.wav" would be your new ".wav" sound file. *Snd* doesn't load it automatically after conversion. Text in quotes is conventional Unix file handling. " mus-l24int" means 24bit-little-endian-integer which is the standard format accepted by most audio editors and DAWs. Of course you can 'undo' changes by using the `(undo)` function. Now let's say you want to reverse your sound. On the listener you will type:

```
(reverse-sound)
```

.

To override changes just undo but to save changes use the `(update-sound)` function:

```
(update-sound)
```

.

The listener sometimes returns errors, sometimes with clues of what you did wrong. Don't be intimidated with errors or mistakes because there is always a make up. Finally try to make sense of the following Scheme expression and see what it returns.

```
(= (cos 0) (sin (/ pi 2)))
```

9.13.1 Editing soundfiles with SND

Sound editing on *Snd* is powerful but not as easy as with other simpler editors. Good advice is learning emacs like keyboard combinations (see below), and then try them on a SND window. To make things easier start with a short (3 secs) mono (one - channel) soundfile.

[c-x means control-x or the keyboard combination *¡ctrl¿-¡x¿*] [m-x means meta-x and at CCRMA is the keyboard combination *¡esc¿-¡x¿*]

Now, if you have a soundfile in the active window try the following combinations and watch the cursor in the displayed soundfile active window. If you don't have a soundfile you can use the **open** command in the file pull-down menu of SND.

```
c-a: move cursor to window start
c-e: move cursor to window end
c-v: move cursor to mid-window
c-l: position window so cursor is in the middle
c-f: move cursor ahead one sample
c-b: move cursor back one sample

c-q: play current channel starting at cursor
c-t: stop playing


c-[Space]: start selection definition


c-m: place (or remove) mark at cursor location
c-j: goto mark

c-w: delete (cut) current region
M-w: (copy) current region
c-y: paste in last deleted region

c-_: undo

c-x c-s: save your current edition.
```

TIP: To play a sound file from a place other than the beginning:

1. Activate sync or sync/unite modes in the lower right check-boxes
2. Create a synchronize marker by:

   ```
   c-s-M (control-shift capital)
   ```

   You should you get markers for all the channels.
3. To play from that location just click on the triangle in any of the markers.
4. Playback should start.

5. You can also move this markers and even 'rock the reels' by dragging the mouse.

A more complex but very useful combination for doing zero padding to your soundfile (i.e. adding silence) might be:

```
c-u 0.5 c-o
```

The real number in this case *0.5*, means that you will insert *0.5* seconds of silence after your cursor position. Of course you can use whatever value (in seconds) you might want.

Try to move forward or backward using the following combinations:

```
c-a            ;;; move cursor to the beginning
c-u 0.4 c-f    ;;; move cursor forward 0.4 seconds
c-e            ;;; move cursor to the end
c-u 1.12 c-b   ;;; nove cursor backwards 1.12 seconds
c-[Space]      ;;; start selection definition
c-e            ;;; a selection of 1.12 seconds
```

If,

```
c-u
```

is followed by a float number , the actual count is that number multiplied by the current sampling rate.

Note: These combinations and more can be viewed on the online Help menu at the overview command at the right side of the **SND** active window.

9.13.2 Digital-Signal-Processing with SND

As you might expect *Snd* is a full featured signal processing toolbox. Most standard techniques have been implemented and new ones can be programmed. As a starting point you might want to try GUI built-in DSP options on the control panel. To open the control you use keyboard key binding combination [C-x C-o]. To close the control panel you [C-x C-c]. On the control panel you will find scroll bars for amp, speed, expand, contrast, and reverb, in addition to a FIR filter (see below). Note that these effects are applied to a selected region or to a the whole file if there is no selection.

For more effects and to do customized digital signal processing you might want to use *Snd's S7's* Scheme features on the listener. A good start for this alternative could be copying **"new-effects.scm"** to your working or " /zap " directory and then load it in the scheme listener with the following steps:

```
cp  /usr/ccrma/lisp/src/snd/new-effects.scm /zap/
```

and then on the SND listener,

```
(load ‘‘new-effects.scm’’)
```

Not by magic notice that there is a new menu called **"effects"** on top of the "SND" GUI. This is a standard collection of DSP tools which might also help to understand and preview how audio signals can be manipulated. As a bonus just for trying, you get sliders to control most of these effects.

Of course you are encouraged to try all of these effects but make sure you already have an opened sound. If you are not happy with the processing you can undo it by **C-x C-u** or simply "undo" in SND's edit menu. If you are really curious about signal processing you can also look at the code and customize these filters or create new ones. Make sure you share them with the DSP and SND communities.

The effects include:

- Reverse soundfile.
- Normalize soundfile.
- Gain change
- Rubber soundfile
- Filters, bandpass, lo-pass, hi-pass and others.
- Flange effects
- Delay effects like echo, filtered echo and modulated echo
- Frequency effects
- Modulation effects
- Convolution
- and more

Most of these are either simple calls on SND functions or use functions in the other scm files. If you are using two or more channels sound-files the actual processing follows the sync chain or the currently active channels but aside from reverb it is good practice to work with DSP effects on each channel separately.

For example if you have a sound file loaded into Snd you can perform the following operations on the listener (among others):

```
(reverse-sound)               ;; reverse soundfile.
(normalize-sound 0.8)         ;; normalize soundfile.
(scale-to 0.6)                ;; gain change.
(rubber-sound 0.7)            ;; rubber soundfile.
(src-sound  (/ 22050 48000))  ;; change sample rate, 22.05k to 48k.
(update-sound)                ;; save changes to soundfile
```

Above features are an understatement of what can be accomplished with *Snd* if *S7* scheme functions are used for signal processing. If you are interested in exploring, there are examples on Snd's documentation but also worth looking examples and irections on files dsp.scm and new-effects.scm. Just copy these files to your working directory or download *Snd's* sources. -More on this below.-

-* A NOTE ABOUT SND'S SOURCE FILES: - On PlanetCCRMA Linux machines outside CCRMA network, Snd sources and documentation are located on,

```
/usr/lib64/snd/scheme \\
/usr/share/doc/snd
```

Alternatively, if you decide to download Snd sources yourself, or if you want to compile Snd, untar a snd-X.tar.gz file by using the command:

```
tar xvzf snd-X.tar.gz
```

your sources will be at

```
./snd-X
```

where X is the Snd version. Instructions for compiling Snd are part of the documentation described at the links on: §9.13

9.13.3 A-more-powerful SND

Snd can use either s7, (a Scheme interpreter), ruby and even Forth as internal scripting languages. For example to create a new sound from an s7 instrument, we can start with the outstanding *Birds* instrument written by Bill Shottstaedt, and also known as "bird.scm". Try typing the following commands on *Snd's* listener. You might need to copy "bird.scm" into your working directory.

1. `(load "bird.scm")`
2. `(make-birds)`
3. `(play)`

At first code might be intimidating but to get acquainted with *s7* language, a simple *"hello-world"* sine instrument in **s7-Scheme** might look like:

```
(define (sine-snd beg dur freq amp)
  (let ((start (seconds->samples beg))
(len (seconds->samples (+ beg dur)))
(phase-inc (hz->radians freq)))
    ;;
    (do ((i start (1+ i)))
((= i len))
      (outa i (* amp (sin (* i phase-inc)))))))
```

You can copy and paste this instrument to your favorite text editor although we recommend an emacs like editor (here you cab have SND as an inferior-lisp process). Once saved as **yoursine.scm** you can load it to the interpreter by typing:

1. `(new-sound "/zap/test.snd") or (open-sound "/zap/test.snd")`
2. `(load "yoursine.scm")`

   This definition of *sine-snd* has the following parameters in the lambda list: **beg** as for offset or beginning of sound,**dur** for durations of the sound **freq** for frequency and **amp** for amplitude or volume of the sound which in this case should be less than one. Therefore to create a sound file with a simple 800Hz sine tone of duration five seconds and amplitude 0.7, you type:

3. `(with-sound () (sine-snd 0 5 800 0.7))`

   To listen to the sound:

4. `(play)`

If you want successive beep tones then you can type something like:

```
(with-sound ()
    (do ((i 0 (+ i 1)))
((= i 5))
      (sine-snd i 0.5 777 0.5)))
```

and then:

```
(play)
```

Take a look at the code for another "sine" instrument, this time using *s7's* `oscil` unit generator, producing same results as above. -It might be easier to understand for people used to Music-N software synthesis packages-.

```
(define (simpler beg dur freq amp)
  (let* ((start (seconds->samples beg))
 (len (seconds->samples (+ beg dur)))
 (s (make-oscil :frequency freq)))
    ;;

    (do ((i start (1+ i)))
((= i len))
      (outa i (* amp (oscil s))))))
```

On the listener function calls for this instrument also involve beginning or offset of sound, duration, frequency, and intensity volume or amplitude. To get sound you use `''(with-sound ()())''` macro to trigger the the instrument. Copy-and-paste this code to a new file and then load it with the listener into *Snd's s7.* Once loaded:

1. (load "simpler.scm")
2. (with-sound () (simpler 0 3 800 0.7))
3. (with-sound () (simpler 0 10 1000 0.5)) ;;; 1000Hz. Tone

Look at the waveform of this sound and see if you can change its envelope. **HINT:** On the "edit" menu of Snd's main edit window, there is an "Edit envelope" option that opens another window with an envelope editor, see image below. In many cases is better to get an envelope to modify a sound this way. You can create fade-in or fade-out and even cross fades. Envelopes is a good way of manipulating a sound. reasons enough to get familiar with this kind of editing on *Snd.*

Convolution between two sound files:

A basic Signal Processing instrument to do "Fourier Convolution" instrument which can be found on **examples.scm** as part of the SND distribution is written as follows. Copy and paste this code to a .scm file in your home directory or in the /zap directory and save it as "**myconvolve.scm**"

```
(define (cnvtest snd0 snd1 amp)
  (let* ((flt-len (framples snd0))
 (total-len (+ flt-len (framples snd1)))
 (cnv (make-convolve :filter (channel->float-vector 0 flt-len snd0)
     :input (make-sampler 0 snd1))) )
    (do ((i 0 (+ i 1)))
((= i total-len))
      (outa i (* amp (convolve cnv)))
      )))
```

1. On the SND listener load myconvolve.scm by typing

   ```
   (load " /myconvolve.scm") or (load "/zap/myconvolve.scm")
   ```

2. Then go ahead and open your source sound file which will be sound 0 or (snd0) by,

   ```
   (open-sound "/zap/source.snd")
   ```

3. Open your impulse sound file which will be sound 1 or (snd1) typing,

   ```
   (open-sound "/zap/impulse.snd")
   ```

4. Perform "Fourier Convolution" on these two sounds as follows (pay attention to the amplitude factor which is the third field in the definition of **cnvtest**.

   ```
   (with-sound () (cnvtest 0 1 0.65))
   ```

5. To listen to your convolved file make sure that the region or soundfile selected on the SND window is the impulse soundfile window and on the listener type:

   ```
   (play)
   ```

9.13.4 Additive Synthesis on SND

Signal theory tells us that by means of the Fourier Transform, any periodic waveform can be represented as a sum of harmonically related sinusoids, each one with its own particular amplitude and phase. The timbre or the spectra of a sound can be viewed from two different perspectives known as the "time domain" and the "frequency domain". Depending on conditions and how a sound is being analyzed one view might be more useful than the other. For example in the case of additive synthesis, the frequency domain is more useful while if we were to edit a complete chunk of sound the time domain will certainly be more useful.

In theory the Fourier Transform (FFT) generates most of the information necessary to reconstruct a signal from a complex sound. Additive synthesis can be used for reconstructing a signal by synthesizing each partial as given on the FFT of the original signal but more generally speaking additive synthesis is used to sum and mix sinusoids to produce more complex sounds. This allows for control over the individual simple components by means of individual envelopes for amplitude and frequency.

Therefore one should be able to add up a bunch of sine waves and get any complex arbitrary signal. The simplest case is when all overtones are integer multiples of the fundamental frequency. In this simple case the waveform is periodic. As the periodic waveform repeats over time we can implement additive synthesis by using a table to store the values of one cycle instead of adding the output of all the equivalent sine oscillators (it is a lot more efficient). Here's a simple Snd instrument inspired on Fernando Lopez-Lezcano's *(circa 1996)* that implements additive synthesis by using the *table-lookup* unit generator. This is called harmonic synthesis by means of wave-table synthesis.

```
(define (dowave start-time dur frequency amplitude harmonics)
  (let* ((start (seconds->samples start-time))
 (len (seconds->samples (+ start-time dur)))
           ;;
           ;; create a table called "waveform" with the harmonics
           ;;
           (waveform (partials->wave harmonics ))
           ;;
           ;; create the table lookup unit generator
           ;;
           (s (make-table-lookup :frequency frequency :wave waveform))
           ;;
           ;; add a simple amplitude envelope...
           ;;
           (amp-env (make-env :envelope '(0 0 0.5 1 1 0)
                              :duration dur
                              :scaler amplitude))
  )
     (do ((i start (1+ i)))
         ((= i len))
       (outa i  (* (env amp-env) (table-lookup s)) ))
     ))
```

You can try the following function calls to listen to the tones.

- Load your instrument:

  ```
  (load "/zap/additive.cms")
  ```

- A simple function call to produce a single tone sound.

  ```
  (with-sound () (dowave 0 4 600 .7 '(1 .5 )))
  ```

- A function call to produce a “square-wave” sound.

  ```
  (with-sound () (dowave 0 1 400 .7 '(1 1 3 .33 5 0.2 7 0.14286 )))
  ```

  Open the frequencies view or F-view on Snd’s graphic interface to get a good look at the spectra of the sound. Move the sliders to see how the sound evolves.

- A simple function call to do a “Sawtooth” sound.

  ```
  (with-sound () (dowave 0 1 400 .7 '(1 1 2 .5 3 .333 4 .25)))
  ```

- A call to have a “triangular-wave” sound.

  ```
  (with-sound ()(dowave 0 1 400 .7 '(1 1 3 0.1111 5 0.04 7 0.20408)))
  ```

- Type the *(play)* function call to listen the sound (you can always type ’ C-x u ’ to undo or erase the active sound you just created on Snd’s editing window).

```
(play)
```

In Partial Synthesis the overtones are not integer multiples of a fundamental frequency and thus we do not resort to the previous shortcut of a table. Therefore, in this case each component (partial) needs to be implemented independently and separately and furthermore sum each signal component to get the more complex sound. This means a lot of computing resources which translate in more computations and time.

Here is a very simple instrument that implements additive synthesis with three partials.

```
(define* (doadd start-time duration frequency amplitude
(partial1 1.0)(amp1 0.3)
(partial2 2.0)(amp2 0.2)
(partial3 3.0)(amp3 0.1)
(ampenv '(0 0 0.5 1 1 0)))
  ;;
  (let* ((start (seconds->samples start-time))
 (len (seconds->samples (+ start-time duration)))
         (sine1 (make-oscil :frequency (* partial1 frequency)))
         (sine2 (make-oscil :frequency (* partial2 frequency)))
         (sine3 (make-oscil :frequency (* partial3 frequency)))
         (amp-env (make-env :envelope ampenv
                            :scaler amplitude
                            :end len)) )
    ;;
    (do ((i start (1+ i)))
        ((= i len))
      (outa i (* (env amp-env)
                 (+ (* amp1 (oscil sine1))
                    (* amp2 (oscil sine2))
                    (* amp3 (oscil sine3))))))
    ))
```

The function calls for this instruments are as follows:

- Load “doadd.scm”:

  ```
  (load "/zap/doadd.scm")
  ```

- A simple function call to the instrument with it default values:

  ```
  (with-sound() (doadd 0 1 660 0.75 ))
  ```

- A call with different factors on the second and third harmonics.

  ```
  (with-sound() (doadd 0 1 660 0.75 :partial2 4.0 :partial3 6.0))
  ```

  In the most general case all parameters of each sine wave are also a function of time (that is, they are controlled by envelopes). This is the most interesting case but also the most expensive computationally and the most difficult to control. But the problem is: ”how do we create or generate the enormous amount of data that we need to accurately represent hundreds of points in the envelopes or all partials?”

– Import from somewhere else ?
– Program generated ?
– High level musical concepts ?
– Synth parameters ?
– Manually ?
– Analysis –¿ Re-synthesis ?

Additive Synthesis of Inharmonic Tones.

Jean Claude Risset, among the pioneers of computer music, experimented exhaustively on the synthesis of natural sounds by means of a computer. He researched instrumental sounds such like a trumpet, by employing spectrum analysis tools at the time (circa 1964), revealing amplitudes and frequencies of partials of these instruments but most importantly, how each partial will differ depending on its frequency, duration and amplitude (i.e. the basis for additive synthesis). Later he joined John Chowning on persuit for complex spectra on FM Synthesis techniques.

Amid his work, there were synthesis of *Inharmonic* tones. For this he found that the higher the frequency of a partial component of a sound, the faster its decay, calling this a principle because a much more natural decay. In most natural sounds, evidence shows that higher frequency components tend to decay more rapidly than the lower frequency ones. But from a creative compositional perspective he mentioned:

> “ Because with additive synthesis one has complete control of decay rates, one can add interest to the computer timbres by occasionally violating the principle.”

Inharmonic tones evoke music played on gongs and bells. For a natural rendition of these sounds some envelopes are changed to non percussive envelopes, hence producing bell-like tones transformed into fluid textures while retaining same underlying pattern. In the first part of a synthesized sound as described above, the listener tends to perceive individual “sound objects” such as bells whereas in a second part of the sound, the fusion of partials into individual sound objects is hindered by asynchronous envelopes, helping hear each partial in successive patterns. Next is our take on the subject of synthesizing *Inharmonic* tones on *Snd’s s7.*

```
;;
;;; Bell Spectra
;;
(define bell-ratios '(.56 .563 .92 .923 1.19 1.7 2 2.74 3 3.74 4.07))
(define bell-amps   '(1 2/3 1 1.8 8/3 1.46 4/3 4/3 1 4/3 1))
;;
;;;  Inharmonic Spectra
;;
(define in-ratios '(.56 .92 1.19 1.71 2.0 2.74 3. 3.76 4.07 4.125))
(define in-amps '(2/3 1 .75 .5 .45 .4 .43 .15 .1 .07))
;;;

(define* (inharmonic beg dur freq att partials coeffs
     (envl '(0 0 .05 1  .85 0.07  1 0)))
  (let* ((start (seconds->samples beg))
 (len (length partials))
 (arr (make-vector len))
 (ratios (make-vector len))
 (amps (make-vector len))
 (envelopes (make-vector len))
 (end (seconds->samples (+ beg dur))) )
    ;;
    (set! ratios (list->vector partials))
    (set! amps (list->vector coeffs))
    ;;
    (do ((i 0 (1+ i)))                                ;; oscillator bank
((= i len))
      (set! (arr i) (make-oscil :frequency (* freq (ratios i))) ))
    ;;
    (do ((k 0 (1+ k)))                                ;; amplitude for each partial
((= k len))
      (set! (amps k) (/ 1 (1+ k))) )
    ;;
    (do ((k 0 (1+ k)))                                ;; envelope for each partial
((= k len))                                   ;; duration shorter on higher partials
      (set! (envelopes k) (make-env :envelope envl
   :base 28
   :duration (* dur (/ (- len k) len))
   :scaler (amps k))))
    ;;
    (do ((i start (1+ i)))                            ;; Output signal
        ((= i end))
      (let ((sum 0.0))
(do ((j 0 (1+ j)))
    ((= j len))
  (set! sum (+ sum
       (* (env (envelopes j)) (oscil (arr j)))))
  )
(if *reverb*
    (outa i (* (*  0.8 att  sum ) rev-amt) *reverb*)
    (outa i (*  0.8 att  sum )))
)) ))
```

Copy-and-paste -all-of-the-above- code into a new file and save it as “inharmonic.cms”.

- Load “inharmonic.cms”into *Snd:*

```
(load "inharmonic.cms")
```

- We can try several function calls to listen to Jean Claude Risset’s perennial bell and other *inharmonic* sounds. But first notice on the code above that we have four lists, couple with “ ratios” for each partial above or below a fundamental and other two with the amplitudes. Duration envelopes are calculated at “runtime” given that, the higher the partial, the shorter its duration. On the function definition for “inharmonic”, slot variables are provided for these lists as well as an overall default envelope which can also be changed. On the first function call, we are generating a bell-like tone on the lower end. Second call a higher bell tone. Third case is *inharmonic* spectra taken from Risset’s examples and for the purpose of comparison. Sounds might be similar, though they should be different. Aside from these examples, try changing overall envelope and other frequencies.

1. `(with-sound () (inharmonic 0 3 500 .4 bell-ratios bell-amps))`
2. `(with-sound () (inharmonic 0 6 1000 .4 bell-ratios bell-amps))`
3. `(with-sound () (inharmonic 0 3 1000 .4 in-ratios in-amps))`

9.13.5 FM synthesis on SND

1. Description:

   Frequency Modulation (or FM) synthesis was discovered and introduced by John Chowning at Stanford around 1973. FM is one of the pillars of CCRMA and was the underlying technology for a very successful synthesizer manufactured by Yamaha and known as the DX-7 during the 1980's. Many of its trademark sounds are still a novel but in the past FM has been an efficient and simple method for generating and simulating complex textures as the vibrato of the singing voice. FM at its core is a simple and powerful method for creating and controlling complex spectra and is also categorized as a worshiping method of audio synthesis. In its simplest form it involves a sine wave carrier whose instantaneous frequency is modulated according to another waveform called the modulator. If the mean frequency of the modulator reaches as high as audio frequency range, sidebands appear as partials of the new modulated sound. This model then is often called simple FM or sine-wave FM. Other forms of FM are extensions of the basic model.

   The position of each sideband is a function of the carrier modulator ratio or *C:M ratio*, and its number and amplitude change as function of the amplitude of the modulator or depth of the modulation. While the precise relationship between the sidebands, the *c:m* ratio and the amplitude of the modulator (better known as modulation index) is determined by the use of Bessel functions, it is common practice to say that the number of sidebands produced on either side of the carrier equals to the modulation index plus 2.

2. Implementation:

   FM synthesis is done by means of signal generators. In Snd *oscil* is Bill Shottstaedt's basic and outstanding signal generator among a very vast array of other generators. Bill's description of a signal generator is as follows:

   > "A generator is a function that returns the next sample in an infinite stream of samples each time it is called. An oscillator, for example, returns an endless sine wave, one sample at a time".

   For more on signal generators see Dick Moore's: *Elements of Computer Music*

   or Charles Dodge and T. Jerse's: *Computer Music.*

   > "Furthermore each generator is made of a set of functions: Make-[gen] sets up the data structure associated with the generator at initialization time; [gen] produces a new sample; [gen?] checks whether a variable is that kind of generator".

   [gen] is the name of a generator like *oscil* therefore *make-oscil* sets up a data structure (interpolation values) at initialization time. Then *oscil* produces a new sample in an stream of samples inside a loop which repeats itself the duration of the desired sound in seconds times the current sampling-rate.

   > "The initialization function normally takes a number of optional arguments, setting whatever state the given generator needs to operate on. The run-time (loop) function's first argument is always its associated structure. Its second argument is nearly always something like an FM input, whatever run-time modulation might be desired; in other cases it can be a function to provide input data or editing operations. Amplitude envelopes are handled with a separate

env generator. Frequency sweeps of all kinds (vibrato, glissando, breath noise, FM proper) are all forms of run-time frequency modulation".

In *Snd's s7's* scheme the initialization function might be something like:

```
(sine-wave (make-oscil :frequency frequency))
```

where frequency is in Hz or cycles per second (cps). There is an optional parameter to this function which is *initial-phase* or the initial angle of phase for the oscillator (oscil uses the Sine function for sine wave generation). The run-time (loop) function might look like:

```
 (oscil sine-wave )
```

or using the optional parameters a more complex *Oscil* function call with vibrato, glissando and frequency modulation value looks like:

```
(oscil sine-wave (+ vibrato glissando frequency-modulation))
```

The *Oscil* signal generator definition in *Snd* is:

"oscil produces a sine wave (using sin) with optional frequency change (i.e. fm). Its first argument is an oscil created by make-oscil. Oscil's second (optional) argument is the current (sample-wise) frequency change (it defaults to 0). The optional third argument is the (sample-wise) phase change (in addition to the carrier increment and so on). So the second argument can be viewed as FM, while the third is PM (phase modulation). The initial-phase argument to make-oscil is in radians. You can use degrees-¿radians to convert from degrees to radians. To get a cosine (as opposed to sin), set the initial-phase to (/ pi 2)".

As can be seen a simple signal generator like *Oscil* can be used for Frequency Modulation and its closely related cousin Phase Modulation.

The terms Frequency Modulation (FM) and Phase Modulation (PM) are source of some confusion and ambiguities. The main source of the diverging paths taken by FM implementers can be traced to a discrepancy between the standard analytical FM formula cited above and "Music-V" implementation which have appeared in most texts on the subject. As to this issue Bill Schottstaedt points out: "The difference is that FM integrates, whereas PM does not. – this does not make up any difference in computational speed, or musical results".

A better explanation in Bill Shottstaedt's *FM.html* should clear this confussion:

"The terms "angle modulation", "phase modulation", and "frequency modulation" (not to mention "exponential modulation" and "angular modulation") are used almost interchangeably in the radio literature, but the textbooks on the subject normally distinguish between PM and FM as follows: if f(t) is directly proportional to the modulating signal then it's PM; if f(t) is directly proportional to the derivative of the modulating signal, then it's FM. Of course, you can't tell which is in use either from the waveform or the mathematical expression of the waveform – you have to know what the modulating signal was. That is a roundabout way of saying that in computer music applications there is no essential difference between frequency and phase modulation. However, the fact that phase and frequency modulation are just two ways of interpreting the same thing should not mislead us into thinking "phase doesn't matter" – the initial phases of the parts of the FM formula do make a difference, as we'll see below".

As a benefit for the reader, this tutorial shows “both” an instrument doing simple FM and another doing simple PM. The reader should listen for any audible or perceptual differences.

3. Frequency Increment (Frequency change parameter)

   In order to understand some of the details in Frequency Modulation it is important to understand the notion of frequency increment. Frequency increment is expressed in terms of radians per second (RPS). In the case of vibrato, or FM or any time we need to specify values to the optional 2nd argument of some oscillator, it is necessary to deal explicitly in RPS. The Snd scheme function *hz-¿radians* converts converts frequencies expressed in cycles per second *“Hz”* to frequencies expressed in radians per second over the actual sampling rate.

   Since we know there are *2 pi* radians in one cycle and if we know Snd’s sampling rate is *22050* therefore, if we want to express this rate in terms of radians per second, the result should be also equal to *2 pi*. To make sure this is true you can try typing the following command on Snd’s listener window,

   ```
   (hz->radians 48000)
   ```

   and you should get,

   ```
   6.2831850751536
   ```

   which indeed is two*pi*. If you don’t get two*pi*, you are working with other sampling frequency on Snd. You might find your sampling frequency by simply issuing,

   ```
   (seconds->samples 1)
   ```

   This also means that the radian frequency of half the sampling rate is *pi,* the radian frequency of 1/4 the sampling rate or quarter cycle is *pi* over two and so on.

   - Several years ago composer Nicky Hind wrote a series of tutorials for composers to get around CLM (Common Lisp Music) on NeXT machines at CCRMA. CLM is Snd predecessor and still being used by some composers because of Lisp code and perhaps some easiness in contrast with other languages. Among those tutorials is a two-part FM Synthesis tutorial worth reading on CLM’s ditribution. Most of his examples can be translated into Snd with few differences, most notably CLM’s “RUN” loop. Below we are borrowing his code and paraphrasing some of his indications.

4. Frequency Increment Instrument

   A frequency increment instrument and other instruments illustrate this issue on *Snd’s s7’s Scheme.* (you can copy and paste them directly on Snd’s listener window or better to Emacs or your favorite editor then save as “fm.cms”.

```
(define (constant-freq-inc beg dur frequency amplitude freq-inc)
  (let* ((start (seconds->samples beg))
 (len (seconds->samples (+ beg dur)))
        (sine-wave (make-oscil :frequency frequency))
        (freq-inc-radians-per-sec (hz->radians freq-inc))
 )
    (do ((i start (1+ i)))
        ((= i len))
      (outa i (* amplitude (oscil sine-wave freq-inc-radians-per-sec ))
                  )) ))
```

- If you are not copying and pasting to the listener load your file "fm.cms":

  ```
  (load "fm.cms")
  ```

- Now you can try the following function calls on the constant-freq-inc instrument:
  - `(with-sound () (constant-freq-inc 0 1 440 0.3 0))`
    Will produce a single one second 440Hz sound.
  - `(with-sound () (constant-freq-inc 0 1 440 0.3 440))`
    Notice a frequency change of 440 which synthesizes a tone one octave higher than 440Hz at 880Hz.
  - `(with-sound ()(constant-freq-inc 0 1 440 0.3 220))`
    Will produce a tone a fifth higher than 440Hz at 660Hz.
  - `(with-sound () (constant-freq-inc 0 1 440 0.3 -220))`
    A frequency change of -220 will produce a tone an octave lower than 440Hz at 220Hz.

5. Sine-Wave Instrument with Enveloped Frequency-Increment

```
(define* (enveloped-freq-inc beg dur frequency amplitude (freq-inc 0))
  (let* ((start (seconds->samples beg))
 (len (seconds->samples (+ beg dur)))
        (freq-inc-func '(0 0  50 1  100 0))
        (sine-wave (make-oscil :frequency frequency))
        (freq-env (make-env :envelope freq-inc-func
                            :scaler (hz->radians freq-inc)
                              :duration dur)) )
    (do ((i start (1+ i)))
              ((= i len))
            (outa i (* amplitude (oscil sine-wave (env freq-env) ))
                      )) ))
```

- If you are not copying and pasting to the listener load your file "fm-2.cms":

  ```
  (load "fm-2.cms")
  ```

- Now try the following function calls on the enveloped-freq-inc instrument:
  - `(with-sound ()(enveloped-freq-inc 0 1 440 0.3))`
    Will produce a one second sound at 440Hz with no frequency increment.
  - `(with-sound ()(enveloped-freq-inc 0 1 440 0.3 :freq-inc 440))`
    This call produces a glissando up to one octave higher and back again.
  - `(with-sound ()(enveloped-freq-inc 0 1 440 0.3 :freq-inc -220))`
    This call produces a glissando down to one octave below and back again. Notice -220 frequency change.

Following we have *s7* Scheme code for a very simple FM instrument which can be used as a template for more complex instruments, and in fact more complex projects in Snd. Notice that *make-oscil* and *make-env* are the definitions for the the signal generators *oscil* and *env* respectively. Here we are creating two oscillators for the carrier and modulator plus one amplitude envelope and a modulation-index envelopes which changes as sound unfolds.

6. Simple FM Instrument

```
(define*  (simple-fm  start-time duration freq amp
       (ampenv '(0 0 25 1 75 1 100 0))
       (mratio 1) (index 1) (indenv '(0 1 100 1)))
  ;;
  (let* ((beg (seconds->samples start-time))
 (end (seconds->samples (+ start-time duration)))
 (car (make-oscil :frequency freq))
 (mod (make-oscil :frequency (* freq mratio)))
 (ampf (make-env :envelope ampenv :scaler amp
 :duration duration))
 (devf (make-env :envelope indenv
 :scaler (radians->hz (* freq mratio index))
 :duration duration)) )
    ;;

    (do ((i beg (1+ i)))
((= i end))
      (outa i (* (env ampf)
 (oscil car (* (env devf)
       (oscil mod))))) )
    ))
```

The function calls for this instrument are as follows:

- Load your code or file “simple-fm.cms”:

  ```
  (load "simple-fm.cms")
  ```

- A simple function call to the instrument with its default values:

  ```
  (with-sound () (simple-fm 0 3 400 0.5 :index 0.75))
  ```

  There is a carrier frequency of 400Hz and a sideband of 800Hz with 0.65 the value of the amplitude of the carrier. You can activate the FFT or (f) view of your Snd sound file to make sure these are the values of this spectra.

- Changing the modulation-index to 0.75

  ```
  (with-sound () (simple-fm 0 3 400 0.5 :mratio 4))
  ```

  Here we get a different spectrum with a carrier frequency at its maximum amplitude and a second sideband frequency which decreases 40% the value of the peak amplitude.

- In the next function call we are manipulating the modulation index as well as *C:M* ratio. In this case we are using a *C:M* ratio of 4:1

  ```
  (with-sound () (simple-fm 0 3 400 0.5 :index 0.75 :mratio 4))
  ```

  then we obtain sidebands on 100Hz with 0.95 amplitude and at 690Hz and the same 0.95 amplitude.

- Now if we change both the modulation-index to 1.75 and the *C:M* ratio also to 4,

```
(with-sound () (simple-fm 0 3 400 0.5 :index 0.75 :mratio 4 :indenv '(0 0.25 100 1.75)))
```

here we get a fundamental frequency around 200Hz at the peak amplitude and average sidebands at 200Hz and 975Hz each one with an average amplitude of 0.925 the peak amplitude.

-In this last example of function call we change the default modulation-index envelope to *'(0 0.25 100 1.*

- Also notice that we get a changing spectra with an average carrier frequency of 400Hz.

- You can always play an active sound window in Snd by using the *(play)* function call (you can always type *' C-x u '* to undo or erase the active sound you just created on Snd's editing window):

```
(play)
```

7. Simple Phase Modulation Instrument

As previously discussed if f(t) is directly proportional to the modulating signal then we have Phase Modulation. If f(t) is directly proportional to the derivative of the modulating signal, then we have Frequency Modulation or FM. This means that difference between PM and FM lies in that FM integrates, whereas PM does not. This does not mean that there is any difference in computational speed, musical or perceptual results. Snd provides tools to check this analytically with its spectrogram windows which can be dragged to change with time. Here we have a simple "Phase Modulation" instrument. Notice the modulating frequency at *oscil's* fourth parameter.

```
(define* (simple-pm start-time duration freq amp
                   (ampenv '(0 0 25 1 75 1 100 0))
                   (mratio 1) (index 1) (indenv '(0 1 100 1))
                   (fmp-inc 0.0)) ;; FM phase increment
  ;;
  (let* ((beg (seconds->samples start-time))
 (end (seconds->samples (+ start-time duration)))
        (freq-mod (hz->radians fmp-inc))
        (phase-mod 0.00)
        (car (make-oscil :frequency freq))
        (mod (make-oscil :frequency (* freq mratio)))
        (ampf (make-env :envelope ampenv
 :scaler amp
 :end end))
        (devf (make-env :envelope indenv
                        :scaler (radians->hz (* freq mratio index))
 :end end)) )
    ;;
    ;;   Notice below that the "phase-mod" part inside "oscil" is
    ;;   incremented on each pass. This means "oscil" is actually
    ;;   doing "Phase Modulation".
    ;;

    (do ((i beg (1+ i)))
        ((= i end))
      (outa i (* (env ampf)
                 (oscil car freq-mod (* (env devf)(oscil mod))))
    ))))
```

Function calls for the Phase Modulation instrument are:

- Load your code or file "phase-mod.cms":

  ```
  (load "phase-mod.cms")
  ```

- A simple function call to the instrument with it default values:

  ```
  (with-sound ()(simple-pm 0 3 400 0.5))
  ```

  Notice the carrier frequency of 400Hz and a sideband of 800Hz with 0.65 the value of the amplitude of the carrier. You can activate the FFT or (f) view of your Snd sound file to make sure these are the values of this spectra.
  The other function calls which should match the FM instrument calls are as follows:
- Changing the modulation-index to 0.75

  ```
  (with-sound ()(simple-pm 0 3 400 0.5 :index 0.75))
  ```

- Manipulating the modulation index as well as *C:M* ratio.

  ```
  (with-sound ()(simple-pm 0 3 400 0.5 :index 0.75 :mratio 4))
  ```

- Now if we change both the modulation-index to 1.75 and the *C:M* ratio also to 4,

  ```
  (with-sound ()(simple-pm 0 3 400 0.5 :index 1.75 :mratio 4))
  ```

- In this last example of function call we change the default modulation-index envelope to *'(0 0.25 100 1.75)* meaning that the value of the modulation index increases from 0.25 to 1.75 along the duration of the sound.

  ```
  (with-sound ()(simple-pm 0 3 400 0.5 :index 0.75 :mratio 4 :indenv '(0 0.25 100 1.75)))
  ```

  ```
  - We should get a changing spectra with an average carrier frequency of 400Hz similar to the FM instrum
  ```

8. Conclusion

   Synthesis of audio by means of Frequency Modulation and its relatives is still a powerful method for generating complex spectra. Many instruments using this principle as a basis have been developed and include violins, trumpets, plucked strings as well as bells, gongs, drums and more. As John Chowning points out, the richness of FM lies in the ability to manipulate on the time domain the spectrum of a sound. We have seen that this can be easily accomplished by using envelope generators which control the modulation-index over time.

9. Further information:

- John Chowning. *The synthesis of complex audio spectra by means of frequency modulation.* Journal of the Audio Engineering Society, 7:pp:526-534, 1973.
- William Schottstaedt. *The simulation of natural instrument tones using frequency modulation with a complex modulating wave.* Computer Music Journal, 1(4), 1977.
- Barry Truax. *Organizational techniques for c:m ratios in frequency modulation.* Computer Music Journal, pages 39-45, 1978. Reprinted in Foundations of Computer Music, C. Roads and J. Strawn (eds.). MIT Press, 1985.
- Dexter Morril. *Trumpet algorithms for computer composition.* Computer Music Journal, 1(1):46-52, 1977. Reprinted in Foundations of Computer Music, C. Roads and J. Strawn (eds.). MIT Press, 1985.
- James Beauchamp. *Letters to the editor: Will the real fm equation please stand up ?* Computer Music Journal, 16(4), 1992.

9.13.6 Simple Physical Modeling on *Snd's s7*

On physical modeling, it is the actual physics of the instrument, and most certainly, its playing technique what is modeled on the computer. Typically, a physical model of an instrument is an algorithm in which a delay line represents points of one period of its waveform at a desired frequency or pitch. Therefore the length of a delay line corresponds to the number of samples required to produce one period of the sound in addition to some type of modulation or modification process recurring on each cycle of the algorithm. Initial conditions of a given delay line, and parameters on the modification process play a key role determining the character of the sound produced. Research on Physical Modeling focuses on finding and bringing parameters to adjust then and tweak them the way the acoustics of a given instrument really operates. Julius Smith's book on Physical Modeling has explantations and technical details on research done by him and his students at CCRMA and other places in the world. Better explanations, part of his published book on the subject, are found online at *Smith, J.O., Introduction to Physical Signal Models*

For example a clarinet can be divided into two sections, namely excitation and resonant body. Performer blows and excites a reed that vibrates producing a wave of air that goes inside a pipe which is the resonant body that amplifies the sound of the instrument. Pitch depends on the length of this pipe and is manipulated by the registers of the instrument. As per basic physics, the longer the pipe, the lower the frequency. Here we can see that the behavior of the body can be modeled by how the air column behaves inside its pipe. Thereby air going-in one side of the pipe and going-out on the other side can help us visualize delay occurring between its inlet and outlet. Since we know that the column of air inside the pipe is vibrating at a given frequency, delay lines model this vibration by giving values or the differences on the vibrating column of air at every time the wave goes in one direction and in the opposite direction when is bouncing back. Typically two delay lines represent a wave guide that models a vibrating column of air inside a clarinet's body. The reed which excitates and moves the air inside the resonant body is also modeled by different means.

- A Model of a Plucked String: The Karplus-Strong Algorithm -

As and example of a delay line and may be one of the most basic incarnations of physical modeling is the Karplus-Strong algogorithm which models the "Plucked String." Basically this method loops a short waveform through a filtered delay line to simulate the sound of a hammered or plucked string or some types of percussion. Alex Strong at Stanford's CS department, invented the algorithm, and K. Karplus did the analysis of how it worked. Together they developed software and hardware implementations of the algorithm, including a custom VLSI chip. Furthermore Julius Smith explains the Karplus-Strong algorithm as:

- The Karplus-Strong algorithm, per se, is obtained when the delay-line initial conditions used to "pluck" the string consist of random numbers, or "White Noise".-

Here we have the code for a "plucked String" Karplus-Strong algorithm that should trigger insights on the subject of physical modeling.

```
(define* (karplus-strong beg dur freq amplitude (damp 0.5))
  (let* ((s-rate (seconds->samples 1))
 (len (+ 1 (floor (/ s-rate 100.0))))) ; 100 = lowest freq in samples
    (let ((delayline (make-delayl len (- (/ s-rate freq) 0.5)))
  (filt (make-onezero))
  (start (seconds->samples beg))
  (end (seconds->samples (+ beg dur)))
  (dout 0.0))
      (do ((i 0 (+ i 1)))
  ((= i len))
(set! dout (delayl delayline (+ (* 0.99 dout) (mus-random damp)))))
      (do ((i start (+ i 1)))
  ((= i end))
(set! dout (delayl delayline (one-zero filt dout)))
(outa i (* amplitude dout))))))
```

- Save this code on a file named "ks.cms", and then load it into *Snd:*

  ```
  (load "ks.cms")
  ```

- Now you can try a couple following function calls to get a taste of the instrument:

  1. `(with-sound () (karplus-strong 0 12 300 0.7))`
     Notice that this plucked sound decays by itself, There is no need for an amplitude envelope.
  2. `(with-sound () (karplus-strong 0 2 660 0.7 0.8))`
     Above we are trying to control decay of the sound with a dampening factor of "0.8".

- Physical Modeling Using the Synthesis ToolKit (STK) -

Perry Cook and Gary Scavone did research and their dissertation theses on the subject of Physical Modeling at CCRMA. A lot of their research, among others, is embedded into the *Synthesis Tool Kit* (STK). They describe **STK** as a set of audio signal processing C++ classes and instruments for music synthesis. **Snd's s7** counterpart are a set of instrument primarily following the K-S plucked string (previously described), a bowed string model , a flute and clarinet models as well as a handful of brass and percussion physical models. You can combine these instruments to create programs which make cool sounds using a variety of synthesis techniques.
More on STK
http://www-ccrma.stanford.edu/software/stk/

At CCRMA you can copy the file `prc95.scm` to your /zap directory by issuing the next commands:

```
scp /usr/ccrma/lisp/src/snd/prc95.scm /zap/prc95.scm
```

Alternatively you can look in the SND distribution for this file and also place it in your /zap or /tmp directories.
Start your STK or rather Physical modeling experimentation with the usual steps for working with SND and scheme

1. Open a new sound in *Snd,*

   ```
   (new-sound "/zap/test.snd") or (open-sound "/zap/test.snd")
   ```

2. And now load your STK instruments,

   ```
   (load "prc95.scm")
   ```

Careful analysis of the code will make you aware of these instruments and their calls:

- Plucked string –¿ plucky

- Bowed string –¿ bow
- Brass Horns –¿ brass
- Clarinet –¿ clarinet
- Flute winds –¿ flute

Also notice the parameters for the s7 function call.

```
(beg dur freq amplitude maxa)
```

Where **beg** is start time of sound, **dur** is the duration of the sound, **freq** is the frequency in Hertz of the sound, **amplitude** is attack intensity or initial pressure of the sound and **maxa** is the overall maximum amplitude of the Physical Model. Note that there is a ratio between amplitude and maxa. You can read more about these parameters on the STK web pages, in the various paper written on the subject by Perry Cook and Gary Scavone or you can simply read the code to figure them out.

You can listen to another rendition of the K-S plucked string algorithm by typing the following function call on Snd's listener:

```
(with-sound () (plucky 0 2 505 .7 1.0))
```

In this case the start time is zero, duration of sound is 2 seconds, frequency is 505Hz and amp is 0.7 while maximum output is function of 1.0.

Try the following plucked sound:

```
(with-sound () (plucky 0 0.3 643.4317 .2 1.0))
```

and try to perceive the difference. Now to get a natural clarinet sound we might type the following function call:

```
(with-sound () (clarinet 0 2 500 .5 0.6))
```

A very long and low clarinet sound might be:

```
(with-sound () (clarinet 0 3 220 .5 0.8))
```

If it sounds like a clarinet, try to play expression markings by changing the **amplitude** and **maxa** ratio. And what about if you want to add vibrato ?

Next, here is a sequence of pipes (may be flute) sounds:

```
(define (pipes)
  (with-sound ()
      (flute 0 .5 261.62555 .6 1.2)
      (flute 1 .8 391.99542 .4 1.1)
      (flute 2 1.4 523.2511 .4 1.1)
      (flute 3.5 .5 261.62555 .6 1.2)
      (flute 4 .8 391.99542 .4 1.1)
      (flute 5 1.4 523.2511 .7 1.1)
      (flute 7 .4 698.4565 .8 1.2)
      (flute 7.8 1.6 391.99542 .2 1.0)
      (flute 9 4 261.62555 .3 0.8)
      ))
```

In this case we are defining a scheme function called “pipes” which carries the score for the physical model. This is sort of like the “SKINI” protocol for the C++ version of STK. Please note the different start times and durations and very important!! don’t take for granted that start times are function of durations themselves.

In order to make sound you type the following function call on the scheme listener.

```
(pipes)
```

Try to experiment changing the order of those sound and make them sound more like a magic pan flute.
.

- Moving Delay Lines and the Leslie Effect -

Delay lines are the building stones for physical models of this kind. But in fact they can be used to model a variety of acoustical phenomena for example the Doppler Effect. For instance, this kind of sound is typical when you hear a train is getting closer to you, and when it passes by. In more technical terms, a Doppler shift is an apparent change in acoustic frequency content of a sound source due to motion of the source relative to the listener. It is known that if the length of a delay line is changed as time goes by, the frequency of the sound passing through is also changed. Therefore, we can also model Doppler effects with delay lines. Among the most interesting musical applications is the Leslie Speaker. The Leslie speaker is described as follows:

> The Leslie is a popular audio processor used with electronic organs and other instruments. It employs a rotating horn and rotating speaker port to “choralize” the sound. Since the horn rotates within a cabinet, the listener hears multiple reflections at different Doppler shifts, giving a kind of chorus effect. Additionally, the Leslie amplifier distorts at high volumes, producing a pleasing “growl” highly prized by keyboard players.

There is a Snd’s s7 implementation of the Leslie effect using time varying delay lines. In this case and to fully appreciate the effect Stereo sound must be used. On this mix there is a panned mixture of a sound source split into two channels, each one with its own stereo placement, path filtering, and Doppler shift. For demonstration purposes this s7 function, instrument or effect makes use of a “wave-train” unit generator to outline characteristics of this sound. Readers should be encouraged to grab this code (leslie.cms) and hack it to read input from an external sound file or for another types of synthesis. To try the Leslie intrument on Snd, here are the steps:

1. On the listener, load the instrument into Snd:

   ```
    (load "leslie.cms")
   ```

2. A simple function call to the instrument using default values:

   ```
   (with-sound (:channels 2) (rotates 0 1 800))
   ```

3. A function tweaking the speed parameter:

   ```
   (with-sound (:channels 2) (rotates 0 8 300 :speedsl 1.0))
   ```

4. A function call with an envelope changing speed of the rotating horn:

   ```
   (with-sound (:channels 2) (rotates 0 3 800 :velenv ’(0 0.05 100 1)))
   ```

5. Another call using a triangular envelope changing speed of the horn:

   ```
   (with-sound (:channels 2) (rotates 0 5 1000 :velenv '(0 0.25 50 1 100 0.3)))
   ```

   Although frequency shifts are subtle, sound tends to be perceived as spacious. Try other functions call of your choice with other parameters and perhaps look at the code to see how delay lines are implemented in Snd s7.

More to come !!! on Physical Modeling in *Snd.*

9.13.7 Bandpass and other Filters

Before digging into filters and digital signal processing, it might be worth to stop and reflect on the object of using a machine for music generation. Are we getting too mechanical that we need mechanisms for achieving ideas and creations?. There is a tendency to think that products resulting from a system, a machine, are molded to be mass produced and widely distributed. In this way an idea turns into a 'template' and its innovative features and success translate into patterns for replication of a hit. To think about a subject matter under this framework seems too constrained and thus limiting a wide array of features encompassed under the terms of computer music.

More than three decades have passed since our first international computer music conference (ICMC). Furthermore, The Computer Music Journal (CMJ) has been publishing papers for almost half a century. ICMS's are still happening every year and CMJ still publishing state of the art, stating and not being too idealistic, facts that a computer is far from being just a machine and probing music produced with computers is not just templates of mass produced patterns. Likewise papers keep publishing and conferences still going on periodically because in part early promises of computer technology have not materialized and remain as goals to be achieved for the faithful pursuer.

Even so, a lot has been achieved, hence horizons have widened for many. Instead of a machine, a computer is more like a *'lab'*, an environment to test ideas, procedures, methods, and more. With it, prototyping, modeling, arranging, editing and of course, creation of works among others are attained. Given this scope, a composer can question aesthetic issues, musical possibilities within a broad spectrum of past-present and future. Beyond analysis and synthesis techniques people can go further into frameworks encompassing new definitions of performance, interactions, listening spaces, just to name a few.

If music is a tradition, the use of a computer follows experiments and language searches on the real of the second half of the twentieth century not leaving ideas of the first half and even before. Recall the "Sound Houses" in the *"New Atlantis"* hinted by Sir Francis Bacon (circa 1627):

> " We have also sound-houses, where we practice and demonstrate all sounds and their generation. We have harmonies, which you have not, of quarter-sounds and lesser slides of sounds. Divers instruments of music likewise to you unknown, some sweeter than any you have, together with bells and rings that are dainty and sweet..."

It does not look computer music is coerced to "digital media" as some want to label it. It goes further beyond those bounds. An advantage of the LAB analogy expressed before, is that ideas on other domains of knowledge can be tested and confronted to the music domain by means of an algorithm. For instance the idea of digital filtering came from replicating a physical mechanism into a discrete computer model of its functionality. Namely, tape recorders of the fifties and sixties. On the issue of "Composers and the Computer" Curtis Roads (circa 1985) abetted hints for what computer techniques beyond a machine might certainly course people pursuing this practice:

> Some digital techniques are little more than precise versions of previous mechanical or analog processes. But precision is not the most important attribute of the computer. What the computer offers the composer is programmability. -The extension of functionality in any direction-...

... The computer can be used to control a synthesizer, to process sounds, to edit scores, to create scores according to composer-specified rules, to print music, to analyze music, and to act as a partner in improvisation.

Signal processing on a digital domain in a computer is an opportunity for ingenuity. Thereby, it is a method for testing ideas and for discoveries. Though, it can be challenging, it is usually rewarding. Filters process media and when in audio domains they become hammer and chisel for crafting sound in music works. A filter can remove, reduce or even strengthen certain bands of frequencies in a signal. There are *infinite impulse response* filters *(IIR)* and *finite impulse response* filters *(FIR).* On *IIR* filters output is recirculated into input in such a way that sound could keep recirculating in a loop forever, depending upon settings of the feedback volume control. Therefore, infinite repeats translate into infinite response. On *FIR* filters repeats are finite. Important to note here that 'repeats' mean 'delays' on DSP parlance. The *frequency response* of a filter shows which frequencies a filter passes and which it rejects. Filters are classified by the kind of frequencies they pass as follows:

- *Lowpass:* Low frequencies are allowed to pass through.
- *Highpass:* High frequencies are allowed to pass through.
- *Bandpass:* A band of frequencies between a lower and upper limit are allowed to pass through.
- *Bandreject:* A band of frequencies between a lower and upper limit are blocked.
- *Allspass:* All frequencies are allowed to pass equally. this kind of filter changes only the phase of the signal.

More complex filters can be made out of combinations of these simpler filters. Likewise, any complex filter can be broken down into combinations of above filters. In addition to modifying the amplitude of signals based on their frequency, filters commonly modify the phase of the signals, delaying phases of some frequencies more than others. We start our examples by trying out a *band pass* on Snd. Here we want to get a band of frequencies in-between lower and upper frequencies. In other words we want to isolate some frequencies given a center frequency and a bandwidth. There are several ways to design a *Bandpass* but here and for the purpose of getting your hands right into DSP *Bandpass* from a *FIR* filter design is going to be tried.

Snd's s7 is a full featured DSP environment stocking plenty of tools to do almost any kind of conceivable signal processing techniques. You can have a canonical filter of any order and type in addition to *IIR* and *FIR* filters of any order. You just need to supply filter coefficients accordingly into the filter's generator. It is not fair to think that DSP is just boiling down recipes for getting right coefficients for a given application. Nevertheless for the time being, it is worth thinking about coefficients for our filtering applications. For the implementation of a Bandpass we need to find "the recipe" for finding its coefficients. To understand this procedure it is helpful to understand the workings of an *FIR* filter.

Recall that a finite impulse response *FIR* filter is a filter whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time. An impulse response might be thought of as a single '1' (one) followed by many '0' (zeroes), on the digital discrete domain. Zeroes will come out after the one has made its way through the delay line of the filter. Most digital filters are function of a delay line. A discrete-time FIR filter of order N, each value of the output sequence is a weighted sum of the most recent input values:

For an analytical explanation of FIR filters or for further information go "HERE"

or, go to Julius Smith's General Filters page:

"Jos Webpage"

```
(define bandpass fir-filter)
```

Following are equations for calculating coefficients for this filter and where W_0 is lower frequency and W_1 is upper frequency. N is the order of the the filter and n is each coefficient number. Keep on mind that most filter definitions wor with radian frequencies. If Hertz are easier to handle then frequencies need to be converted.

$$b_0 = \frac{\omega_1 - \omega_0}{\pi}$$
$$b_n = \frac{\sin(\omega_1 n) - \sin(\omega_0 n)}{\pi n} \cos(\frac{\pi n}{N})$$

Each value is stored in an array so that we have a vector with all coefficients to be pass into the filter unit generator of Snd.. Below there is a function definition to see if the filter really works.

```
(define* (bpass-noise start dur amp cfq bwf
 (len 30))
  (let* ((beg (seconds->samples start))
 (q (hz->radians cfq))                          ;; center 'cut' frequency to radians
 (bw (hz->radians bwf))                         ;; bandwidth frequency to radians
 (loc (- q (/ bw 2)))                           ;; lower frequency bound
 (hic (+ q (/ bw 2)))                           ;; upper frequency bound
 (b0 (make-bandpass loc hic len))               ;; make bandpass structure
   (ra (make-rand :frequency sampling-rate
:amplitude (* 2 amp)))        ;; create random noise generator
 (end (+ beg (seconds->samples dur))))
    ;;
    (do ((i beg (+ i 1)))
((= i end))
      (outa i (bandpass b0 (rand ra)))
      ) ))
```

Copy-and-paste -all-of-the-above- code into a new file and save it as “bandpass.cms”.

- Load “bandpass.cms”into *Snd:*

  ```
  (load "bandpass.cms")
  ```

- Now we can try function calls to listen the effect of this bandpass on a noise source. On our function definition “bpass-noise” last couple of arguments are center frequency and bandwidth. On this case we have a center frequency of 3000 Hz. and a bandwidth of 1000 Hz. For this call we are using default filter order of 30. Look at Snd’s screenshot below.

  1. `(with-sound () (bpass-noise 0 2 0.5 3000 1000))`
  2. `(with-sound () (bpass-noise 0 2 0.5 1200 500 :len 60))`

     A function call above for “bpass-noise” with a center frequency of 1200 Hz., a bandwidth of 500 Hz., and filter order of 60. Listen and look at the new spectra. Band should be narrower.
  3. `(with-sound () (bpass-noise 0 2 0.5 3000 1000 :len 100))`

  And finally above, a function call with a filter order of 100. Frequency peak of 3000 Hz, should be sharper.

Next another rendition of a Bandpass but this time using the Butterworth type. Once again there is an example on dsp.scm also adapted and written for Snd by Bill Schottstaedt. More information about *Butterworth* filters: “Butterworth Wiki” . A thorough description of “dsp.scm” and its implementation is of course found on Snd’s “documentation” . Main characteristic of Butterworth filters is having a flat response and though suitable for occasions when a stable pass band is needed. Filter structure definition as well as its implementation on *Snd’s s7* is provided as follows:

```
(define (make-butter-band-pass fq bw)
  (let* ((c (/ 1.0 (tan (* 0.5 (hz->radians bw)))))
 (d (* 2.0 (cos (hz->radians fq))))
 (c1 (/ 1.0 (+ 1.0 c))) )
    (make-filter 3
 (float-vector c1 0.0 (- c1))
 (float-vector 0.0
       (* (- c) d c1)
       (* (- c 1.0) c1) )) ))


(define butter-band-pass filter)

(define* (butterw start dur amp cfq bw)
  (let* ((beg (seconds->samples start))
 (f0 (make-butter-band-pass cfq bw))
 (ra (make-rand :frequency sampling-rate
:amplitude (* 2 amp)))
         (end (+ beg (seconds->samples dur))))
    (do ((i beg (+ i 1)))
((= i end))
      (outa i (* (* 5 amp) (butter-band-pass f0 (rand ra))))
      )))
```

Copy-and-paste -all-of-the-above- code into a new file and save it as “butterworth.cms”.

- Load “butterworth.cms”into *Snd:*

  ```
  (load "butterworth.cms")
  ```

- Now we can try couple of function calls and listen the effect of a *Butterworth* on a noise source. On the function definition “butterw” last couple of arguments are center frequency and bandwidth. On the first case we have a center frequency of 1200 Hz. and a very narrow bandwidth of 80 Hz. On the second case we are again trying a a center frequency of 3000 Hz. and a narrow bandwidth of 80 Hz. Take a look at Snd’s spectrum screenshot below.

  1. `(with-sound () (butterw 0 2 0.5 1200 80))`
  2. `(with-sound () (butterw 0 2 0.5 3000 100))`

The canonical *'biquad'* filter is a second-order *IIR* filter with two poles and two zeros. In practical terms, it is a versatile filter with options for a wide variety of implementations such as lowpass, highpass, bandpass, notch filter, among others. As an *IIR* filter, recall that each application depends on the coefficients of its difference equation. Nonetheless, below we are applying coefficients such that we have a resonance filter, almost like a *Helmholtz* 'resonator', if we can call it like that.

```
(define (make-biquad a0 a1 a2 b1 b2)
  (make-filter 3
       (float-vector a0 a1 a2)
       (float-vector 0.0 b1 b2)))


(define (make-resonance-biquad freq r)                    ;; frequncies in Hz.
  (let* ((radius (- 1. (hz->radians (* r .225))))         ;; radius is bandwidth.
 (b1 (* -2 radius (cos (hz->radians freq))))
 (b2 (*  radius radius))
 (a0 (- 0.5 (* 0.5 b2)))
 (a1 0.0)
 (a2 (- a0)))
    (make-biquad a0 a1 a2 b1 b2)) )

(define resonance-biquad filter)


(define* (res-bq start dur freq amp fc ra)
  (let* ((beg (seconds->samples start))
         (s (make-pulse-train :frequency freq))
 (res (make-resonance-biquad fc ra))
 (end (+ beg (seconds->samples dur))) )
    (do ((i beg (+ i 1)))
((= i end))
      (outa i (* 14 amp (resonance-biquad res (pulse-train s))))
      )))
```

Copy-and-paste this code into a new file and save it as "resonance.cms".

- Load "resonance.cms"into *Snd:*

  ```
  (load "resonance.cms")
  ```

- Coming, two function calls to outline the effect of a *Resonance* on a pulse train sound source. On the function definition "res-bq" we have arguments for center frequency and bandwidth as well. On the first case we have a center frequency of 3000 Hz. and a bandwidth of 1200 Hz. for natural resonance response. On the second case we are again trying a resonance at 3000 Hz. using a narrower bandwidth of 200 Hz. Listen and take a look at *Snd's* spectrum screenshot below.

  1. `(with-sound () (res-bq 0 2 1000 0.5 3000 1200))`
  2. `(with-sound () (res-bq 0 2 600 0.5 3000 200))`

Last example is a feedback *Comb* filter. Comb filters are basic building blocks for digital audio effects. For example acoustic echo simulation is one instance of a comb filter. The feedback comb filter is a special case of an *IIR* "recursive" digital filter, since there is feedback from the delayed output to the input. The feedback comb filter can be regarded as a series of echoes, exponentially decaying and uniformly spaced in time. There can be time varying comb filters that produce "effects", such as flanging, phasing and chorusing among others. *Snd's* comb filter implementation is a delay line with a scaler on the feedback. On this example we are time changing frequencies of the combs so that a changing resonance is heard as time goes by.

```
(define* (comb-filter-tv start dur freq amp length feedback
(chg-env '(0 1 1 0)))
  (let* ((beg (seconds->samples start))
 (end (seconds->samples (+ start dur)))
         (s (make-pulse-train :frequency freq))     ;; pulse train sound source
         (del (make-comb :size length               ;; length of delay (samples)
:max-size 95
:scaler feedback))          ;; comb feedback scaler
         (chg (make-env :envelope chg-env           ;; time changing envelope
  :scaler length
  :base 10.0
  :duration dur)))
    (do ((i beg (+ i 1)))
((= i end))
      (outa i (* amp (comb del (pulse-train s) (env chg))))) ))
```

Copy-and-paste this code into a new file and save it as "combfilter.cms".

- Load "combfilter.cms"into *Snd:*

  ```
  (load "combfilter.cms")
  ```

- Right below, several function calls showing a comb effect while using a *Comb Filter* on a pulse train sound source. On the function definition "comb-filter-tv" (i.e. comb filter time variant) we have arguments for delay length, a feedback gain scaler and a time-changing envelope in addition to the other standard parameters. On case first we use a delay of 80 samples and a feedback gain of 0.75. On the second we are using same values but inverting the time envelope. On last couple of examples we tweaking the feedback gain parameter but decreasing delay. Keep on mind that filter gain can over blow since we are feeding back signal.Listen and take a look at *Snd's* spectrum screenshot below.

  1. `(with-sound () (comb-filter-tv 0 3 600 .3 80 .75))`
  2. `(with-sound () (comb-filter-tv 0 3 600 .3 80 .75 :chg-env '(0 0 1 1)))`
  3. `(with-sound () (comb-filter-tv 0 3 600 .4 40 .9 :chg-env '(0 0 1 1)))`
  4. `(with-sound () (comb-filter-tv 0 3 600 .6 45 .8 :chg-env '(0 0 0.5 .6 1 0)))`

222urlhttps://ccrma.stanford.edu/course-overviews/music-222

9.13.8 Multichannel Intensity Panning

Below is a rather complex, not so elementary example. It has to do with spatialization of sound sources. Here we are posting code mainly for legacy and historical reasons, in addition to outlining more features of *Snd's s7.* Complexity gives flexibility at the expense of a rather not so steep (hopefully) slope. With some time and insight a great deal can be learned from all programming shown here. If there is some intimidation, reader should feel free not to tackle this subject and implementation now, on the hope of coming back later.

On this web page, we are presenting an instance of a theory and its applications using a computer model. Though it should be acknowledged that a complete discussion of this subject matter is well beyond the scope of these introductory pages. Here we are presenting elementary concepts pertaining to spatialization and motion of a sound source, given a path in a two-dimensional plane with expectations that readers can build upon, and extend its use. Hopefully to go deeper on now accessible sound diffusion techniques such as Ambisonics, VBAP, and perhaps wave field synthesis. On understanding code here, reader would at least get some curiosity on topics as follows: spatial perception of sound, intensity panning, Doppler Effect, motion of sound sources, reverberation, Lissajous Figures, and localization of sources as well as structure and functional programming using Scheme.

Using *Lissajous Figures* in a musical context is a technique pursued by John Chowning and others on the days of SAIL, Stanford Artificial Intelligence Laboratory. A complete recount of this research is on a Computer Music Journal paper titled "The Simulation of Moving Sources" published by John Chowning. It is advisable for readers interested on this subject to get their hands on this paper. Insight on *Turenas*, among pioneering Computer Music pieces is also recommended. Although widely available as a stereo recording, multichannel renditions can also be obtained. Most certainly by listening to this piece people get acquainted with the nature of acoustical space manipulation.

Code below makes use of J. Chowning's Lissajous equations, but this time using a white noise sound source so the effect is better perceived. Doppler is added so that a person in a sweet spot listens to sounds coming-and-going. Keep on mind that Doppler is function of distance, as well as speed of source which is also dependent on time. Reverberation is also an intricate part thereby giving the illusion of an enclosed space as a listening environment. More on the subject is widely available on publications all around. There is even a Stanford course on the subject of "Sound in Space" taught by Fernando Lopez-Lezcano who has been researching the subject for years now. Among other people contributing deeply to the field worth mentioning are Dick Moore, Gary Kendall, Pablo di Liscia, Juan Pampin, Joe Anderson, Pablo Cetta, just to name a few. Before getting into the code, a bit of theory as outlined on Dick Moores's book "Elements of Computer Music" might prove helpful for better understanding these processes. Seems worth to remark that code for below application of intensity panning using J. Chowning's Lissajous functions was based on Dick Moore's *CMusic* panning unit generator written in "C" language.

> "In the production of computer music, the typical problem is not to simulate a particular concert hall or room with any precision but to impart a spatial quality to sounds generated either by modification of recorded sounds or by methods of pure synthesis."

Dick Moore's quote above outlines "spatial quality" to perception of sounds. This can also be portrayed as sound coming from everywhere on three dimensions. On this portrait localization of sounds becomes a composition parameter. For this it is necessary to consider the processing of signals both by digital signal processing(dsp) and by our own hearing mechanism. Paraphrasing Moore again, we define the quest for sound spatialization in various ways:

- To treat locations of speakers as the set of possible locations from which sound can emanate.
- Create the impression of sound coming from directions other than those of loudspeakers.
    1. Directions might be restricted to the horizontal plane.
    2. Directions might include a vertical dimension as well.

- Another possibility is to create impressions of both direction and distance, allowing a virtual sound source to come from a location within a "virtual sound space", generally lying beyond a perimeter outside by that of tying all loudspeaker on the listening space.
- If general control over sound position can be achieved in either or both of the horizontal or vertical planes, then it is also possible to create the impression of moving sound sources.

The problem of sound spatialization is then the problem of gaining prescriptive control over positions of virtual sound sources within an imaginary, virtual, or illusory space in which such events may occur. For this we need to keep in mind that:

- It is quite difficult to create sound sources that are located at positions that are closer to those of the loudspeakers.
- The "illusory space" is therefore typically defined by a closed polygon resulting by tying all loudspeakers with a string.
- Localization cues are needed to create virtual sound sources at arbitrary locations within a virtual acoustic space. For this purose, it is necessary to use the loudspeakers in a special configuration that provides strong localization cues to a listener located inside the listening space. Localization cues mightbe divided into these basic categories.
    1. Perceptual and cognitive cues that help determine the direction of the virtual sound source on the horizontal plane (azimuth).
    2. Perceptual and cognitive cues giving direction of a virtual sound source on the vertical plane (elevation).
    3. Perceptual and cognitive cues that help determine "distance" of the virtual sound source.

In regards to intensity it should be said that sounds in the real world coming from directly in front of, or behind the listener, reach both ears with equal intensity, while those coming from the right or left reach one ear with slightly more intensity than the other. A general impression of directional intensity may be simulated through the use of *"intensity panning"*. While using a multichannel speaker system, we can provide ideal intensity cues only for directions defined by positions of the speakers. At azimuth angles intermediate between any two of these directions, we can distribute sound between adjacent pairs of loudspeakers.

Generally, we can control the intensity of a sound in each playback channel by using a gain factor that is multiplied directly to the waveform of the sound undergoing spatialization. Because such a gain factor multiplies the waveform directly, it represents a direct control on the amplitude rather than its intensity. Recall that intensity is also a function of distance. Thus intensity changes with distance following the inverse square law of intensities, where sound is inversely proportional to the square of its distance from the source to the listener.. Therefore amplitude is proportional to the square root intensity for linear amplitudes.

To maintain a constant sound intensity at the listener's position for all intermediate positions of the virtual sound source as it pans from left to right, we require that the total intensity be constant. On mathematical terms we can think of a relation like $g_1^2 + g_2^2 = K$, where *'g'* and *'h'* are the respective gains of each speaker and where *'K'* is a constant. If we take look at, $sin^2(\theta) + cos^2(\theta) = 1$, we see that the squares of sine and cosine are equal to *'one'* which is always constant.

Therefore by using this relation, we can always guarantee that at every angle theta with the azimuth of a sound source, sum of its squares will always be one and a constant. But keep on mind that intensity proportional to the inverse square of distance. In order to create a realistic illusion while using intensity panning we need to add the distance component. Distance here is the distance between the listener and the loudspeaker. For a realistic pan we need to use the following equations:

$$g_1 = \frac{\sin^2(\theta)}{d^2}$$

$$g_2 = \frac{\cos^2(\theta)}{d^2}$$

where *'g'* is gain factor of each loudspeaker, *theta* azimuth angle and *'d'* is distance. A physicist description of *intensity* outlines:

> Energy from the motion of sound waves flows through the eardrums and into the inner ear where is registered as sound. Intensity *'I'* is the energy *'E'* per unit of time *'t'* that is flowing across a surface of a unit area *'a'*. Therefore $I = P/a^2$, where $P = E/t$. Power for this purpose is equivalent to the amplitude of a sound.

For a circular pan without a hole in the middle we can use the following equations:

$$g_a = \frac{\sqrt(2)}{2}[cos(\theta) + sin(\theta)]$$

$$g_b = \frac{\sqrt(2)}{2}[cos(\theta) - sin(\theta)]$$

Now above equations need to be implemented. For this purpose a stereo panning program needs to coded. Below is our first elementary *"stereo"* intensity panning program. Only hack here is that we need to adjust angular phase starting on *pi/4* so that sound starts moving from one loudspeaker to the other. A constant called *'cfactor'* stores the value of square root of two over two, which is also equal to *sin(pi/4)* and *cos(pi/4)*. Program is commented to help understanding what is going on.

```
(define* (pan-equal beg dur freq amp dist
    (cycles 1)   ;; number of rounds
    (dir #t))    ;; direction of rounds

  (let* ((start (seconds->samples beg))
 (s (make-oscil :frequency freq))
 (zeta (/ pi 4))
 (theta 0.)
 (cfactor (cos (/ pi 4)))
 (end (seconds->samples (+ beg dur)))
 )

    ;; Check for direction parameters clockwise start on -pi
    ;;                         counter-clockwise start on +pi

    (if dir (set! theta (- zeta))
(set! theta zeta))

    ;; main loop
    ;;

    (do ((i start (1+ i)))
((= i end))
      (let* ((gfa (* cfactor (+ (cos theta) (sin theta))))
     (gfb (* cfactor (- (cos theta) (sin theta))))
     (invsq (* (/ (* dist dist)) amp))
     (amp1 (* gfa invsq))
     (amp2 (* gfb invsq))
     (signal (oscil s))
     )
       (outa i (* amp1 signal))
(outb i (* amp2 signal))

;; update azimuth angle values
;
(if dir
    (begin
      (set! theta (+ theta (/ (* cycles pi) end)))
      (if (>= theta  (* 7 zeta)) (set! theta (- zeta))) )
    (begin
      (set! theta (- theta (/ (* cycles pi) end)))
      (if (<= theta  (- (* 7 zeta))) (set! theta zeta)) )
    )
))
    ))
```

Above program pans a simple sine wave into two *"Stereo"* channels. Here we are adding useful features which make it longer but they add more flexibility. Instead of going from left to right, we can make sound go around several times by toggling *'cycles'* parameter. A value of two is two complete rounds. A value of four is four rounds. Likewise a direction parameter can also be toggled. Notice that equations are implemented inside the main loop. Azimuth is incremented on a sample rate level so that signal smoothly goes from one channel to the other. Keep on mind that angular phase here is changing as time goes by.

- Save this code on a file named "panins.cms", and then load it into *Snd:*

  ```
  (load "panins.cms")
  ```

- Now we can try function calls to listen the effect of intensity panning:

  1. `(with-sound (:channels 2)(pan-equal 0 3 500 0.5 1))`

     - Listen to the sound going from one speaker to the other in a circular way.-
     - We can set the number of channels variable '*clm-channels* to default to two channels *(stereo sound):*

  2. `(set!  *clm-channels* 2)`

     Let's try to make the sound go around twice,

  3. `(with-sound ()(pan-equal 0 3 600 0.8 2 :cycles 2))`

     And finally here we are making several rounds on the opposite direction;

  4. `(with-sound () (pan-equal 0 6 800 0.7 1 :cycles 4 :dir #f))`

     Below a screenshot of *Snd's* showing a Stereo soundfile generated by function calls above.

Listen carefully to the sound generated in two channels and try to perceive its motion from one side to the other. Here we are creating the illusion of an imaginary space beyond loudspeakers and circular motion on headphones. Change parameters on function calls to hear different parameters. Further try to hack the above code so that sound is something different than pure sine wave. May be try to read a mono sound file and pan it around. Above code is basic and a stepping stone for our next multichannel sound diffusion application. Not that we are hoping that by hacking above code you will end up with next example.

But perhaps curiosity might lead you to something close. Question might be then, what about more than two-channels stereo?.

Lissajous Figures

——————————————————————————————– Lissajous Figures, familiar to most physical scientists and engineers, connotes harmony, order and stability. Lissajous figures are named after French mathematician Jules Antoine Lissajous, but are also known as Bowditch curves after Nathaniel Bowditch, a mathematician from Salem, Massachusetts, who discovered them around 1815.Lissajous figures were sometimes displayed on oscilloscopes meant to simulate high-tech equipment in science-fiction TV shows and movies in the 1960s and 1970s. Lissajous curves are the family of curves described by the parametric equations:

$$x(t) = asin(\omega_x t - \delta_x)$$
$$y(t) = bsin(t)$$

With these equations we get *x/y* pairs that plot on rectangular coordinates producing curves depending on parameters for factors and angles of the above equations. MatLab or Octave for that matter are very useful for plotting Lissajous Curve. See here.

- But, what about these figures?.- John Chowning had been experimenting with drawing tools on early computer devices to get points for creating motion of sound using azimuth and Doppler shift parameters. Motion of the source from one point to other will be give a difference in location. By using a mouse-like device he was able to control points that generated graphic output though no sound. In this way there was not near actual interaction with sound and therefore mapping parameter from graphs to composition tended to be a cumbersome method. But because of being in a laboratory environment such at SAIL, David Poole, another researcher at the time, pointed out that these drawing patterns looked like a Lissajous figure. Then curiosity was sparked by his comment leading Chowning to learn about and program Lissajous figures.

> “I quickly advanced through the well-known looping patterns and discovered that interesting figures could be generated, and whose sound manifestation possessed a graceful motion that seemed to me natural —as the sound followed the path of a Lissajous figure it decelerated and accelerated as it approached and left a change in direction.”

In regards to *Turenas* and spatial motion of sound sources Chowning goes further on Lissajous figures:

> “In Turenas I made full use of the newly acquired control of sounds in space. The spatial trajectories areboth curvilinear and linear motions. The linear trajectories are sometimes expressedby radical changes in timbre as the sounds pass through the listener space. Thus, computer synthesis allowed me to achieve synchronous control over spatial trajectories and timbral transformations.”

Here are John Chowning’s Lissajous equation having sine and cosine components:

$$y_\tau = cos(3\pi\tau) + cos(7\pi\tau)$$
$$x_\tau = sin(2\pi\tau) + sin(6\pi\tau)$$

Below is a graph of these Lissajous Equations on the *x/y* plane:

Coming back to the question posted before as how we go from two-channel “stereo” to four-channel and perhaps beyond, it is important to keep on mind that listening space is *360 deg.* and that any sound source will come somewhere around this circle. To cue an angular position an energy ratio (gain or attenuation) is applied to the direct signal on each loudspeaker pair. Since we have four quadrants, angles between loudspeaker pairs are *90 deg.* in relation to the listener. The obvious means of changing ratio of the direct signal for the moving source, is to make the energy applied to the loudspeakers pairs proportional to the angle of displacement. Very much like we did on the two channel example but after calculations for channel-one and channel-two, we calculate energy ratios for channel-two and channel-three, and so forth. For this purpose we need to come up with a function tha helps to find out energy ratios for each of the four loudspeaker. We can start with th following equation as stated on by Dick Moore on Elements of Computer music:

$$G_n(\theta) = \begin{cases} \cos(\theta - \theta_n) & \text{if } |\theta - \theta_n| < 90^\circ \\ 0 & \text{otherwise.} \end{cases}$$

Above *Gn* is gain-of-speaker-n, *theta* is position of sound source and *theta-n* is angle of loudspeaker-n. This function is implemented on s7 scheme as follows:

```
(define halfpi (* 2. (atan 1.)))

(define (posi rho spkpos distf)
  (let* ((diff (- rho spkpos))
 (in-between-angle halfpi))
    (if (>= (abs diff) in-between-angle)
(values 0)  (* (cos diff) distf))
    ))
```

Function above takes the angle where the sound source is, the angle of the standing speaker in addition to the inverse square of distance. From our main program we will call this function to calculate gain for each speaker depending upon position. Note that above function gives gains for positions in between loudspeakers. Recall that gain is proportional to distance, therefore gain here is among localization cues.

Furthermore, in order to simulate the distance cue, a reverberant signal should be synthesized in addition to a direct signal (as above), such that the intensity of the direct signal decreases more with distance than does the reverb signal. Thus, as more distance from the listener, overall gain or amplitude is attenuated. It is assumed that in a small space, the amplitude of reverb signal produced by the sound source at constant intensity but varying distances from the listener changes little, but in a large space it changes somewhat.

As for motion or sound source and velocity cues, we know that a static listener receives velocity information from a moving sound source at a rate of change proportional to speed. Similarly a frequency shift can also tell if source is getting closer or getting away. This because of Doppler shifts better known as Doppler Effect. The simulation of the Doppler effect is achieved, simply, by scaling the unit of distance in meters and making change in frequency proportional to the rate of change of distance over time. Our implementation of Doppler shifts is done by using delay lines. We get frequency changes by changing the length of a delay line. There is a delay line for each loudspeaker. Length is calculated from the distance of the sound source to the listener.

As pointed before reverberation is also an essential part of the distance cue. Perceived reverberation also gives size of room and space information, including shape. Plainly speaking reverberation, might be defined as the persistence of sound after its excitation. Quantitatively is regarded as the collection of reflected sounds from the surfaces in an enclosed space such like an auditorium. Direct sound received is followed by distinct early reflected sounds and then a collection of many tail reflected sounds which blend and overlap giving characteristics of an auditorium or even virtual space. John Chowning again points out: *" In simulating a sound source in an enclosed space, then, it is desirable for the artificial reverberation to surround the listener and to be spatially diffuse."*

Below is our intensity panning program written on *Snd's s7.* In addition to *Lissajous Figures* as a basis for spatial gestures, this code implements above and just described features for obtaining cues for localizing a sound around a space. It is assumed that listener is on the center (sweet spot) and surrounded by a set of loudspeakers, each at a fixed distance from the listening spot. It is important to point out that position of sound source changes with time and depends on angular position inside a 360-deg. circle. Sound-path gestures are performed beyond the perimeter outlined by loudspeakers on a bigger illusory space behind. Note that *Lissajous* image above shows several cycles (iterations) of *Lissajous equations.* In order to achieve full gestures we don't need complete iterations of the equations.

```
(define twopi (* 8. (atan 1.)))
(define halfpi (* 2. (atan 1.)))


(define samprate (seconds->samples 1))

(define sspeed 343)

(define (distn->samples dist)
  (floor (* dist (/ samprate sspeed)))
  )

(define nchans 4)


;;
;; ;;
;; ;;  Gain at position: quadrant in reference to speaker.
;;

(define (gainfn rho spkpos distf nchs)
  (let* ((diff (- spkpos rho))
 (in-between-angle (/ twopi nchs))
 )
    (if (>= (abs diff) in-between-angle)
(values 0)  (* (cos diff) distf))
    ))

;;
;; ;;  A moving sound source function definition.
;; ;;  ===========================================
;;

(define* (lissajous beg dur (frq 800) (cycles 1) (nch nchans) (rev-amt 0.025))
  (let* ((start (seconds->samples beg))
 (flt (make-two-pole :radius .998 :frequency frq))
 (ran1 (make-rand  8000 0.00750))                ;; get some noise
 (zeta (* cycles halfpi))                     ;; start angular position
 (theta zeta)                                 ;; position in time
 (dsize (distn->samples 10))                   ;; 2.0 Min distn
 (maxdllsz (distn->samples 1000))               ;; Max delay (distn)
 (dll (make-delay dsize :max-size maxdllsz))    ;; delay line
 (speaker (make-vector nch))                  ;; speaker position array
 (gains (make-vector nch))                    ;; gains array
 (outsig (make-vector nch))                   ;; final signals
 (end (+ start (seconds->samples dur))))

;
; Initialize speaker positions
; degrees: -45; 45; 135; -135;
```

```
;

    (do ((k 0 (1+ k)))
((= k nch ))
      (let ((rads (+ (/ pi nch) (* k (/ twopi nch))))
    (idx (- (1- nch) (modulo (1+ k) nch))))
(set! (speaker idx) (- pi rads))
))


    ;;
    ;; ; main " generate signal" loop
    ;; ; ---------------------------
    ;;

    (do ((i start (1+ i)))
((= i end))
      ;;
      (let* ((insig (rand ran1))                     ;; noise unit generator
     ;;
; J. Chowning Lissajous equations
     ;;
     (yt (+ (cos (* 3 pi theta)) (cos (* 7 pi theta))))
     (xt (+ (sin (* 2 pi theta)) (cos (* 6 pi theta))))
     ;;
; New position angle (polar coordinates)
     ;;
     (rho (atan yt xt))
     ;;
; distance of sound source from origin
     ;;
     (dfn0 (sqrt (+ (* xt xt) (* yt yt))))

;
; distance cannot be zero
; becasue we get infinite
; gain (signal blows)
;

     (distn (+ dfn0 1.998))
     ;;
; inverse squared distance
     (invsqd (/ (* distn distn))

; change filter center freq

     (cfq (+ (- frq 50) (* 256 (/ (1+ dfn0)))))

; two-pole filter

     (noi (two-pole flt insig))
     )
;;
;;
; change filter center freq
(set! (mus-frequency flt) cfq)

;;
;; ;;  calculate gain for each speaker
;;

(do ((j 0 (1+ j)))
    ((= j nch))
  (let ((gf (gainfn rho (speaker j) invsqd nch)))
    (set! (gains j) (* gf 5.998)) ))

;;
;; ;;  Generate Doppler motion plus output signal
;; ;;  ----------------------------------------
;;

(do ((j 0 (1+ j)))
    ((= j nch))
  (let ((gfn (gains j)))
    (set! (outsig j) (* gfn
(+ (* .075 noi)
   (* .375 (delay dll noi
     (* 4747 (/ (1+ dfn0))))
 )))
)))
;;
;; ;; --> Here we start output
;;

(do ((k 0 (1+ k)))
    ((= k nch))
  (out-any i (* .625 (outsig k)) k)

  ;;
  ;; ;; add reverb!
  ;;

  (if *reverb*
      (let ((dist-scaler (* 2.725 invsqd))
    )
(out-any i (* (outsig k)
      (*  rev-amt  dist-scaler))
 k *reverb*) ) )
  )

;;
```

```
;; ;; Increment grow angle function
;; ;; ----------------------------
;;

       (set! theta (- theta (/ (* .25 cycles pi) end)))
(if (< theta  (- zeta)) (set! theta zeta)) ))  ))



;;
;; ;;          - E N D -
;; ;; ----------------------------
;;
```

- Save this code on a file named "lissajous.cms", and then load it into *Snd:*

  ```
  (load "lissajous.cms")
  ```

  - Alternatively you can download it HERE.

- Now we can try function calls to listen Lissajous gestures on panning:

  1. `(with-sound (:channels 4) (lissajous 0 10 :nch 4))`

     - Listen to the sound going from one speaker to the other.-
     - We can set the number of channels variable **clm-channels** to default to four channels:

  2. `(set!  *clm-channels* 4)`

     Let's try to make more gestures. Keep on mind that duration's or length of traces are function of time. The more cycles, the more but shorter traces.

  3. `(with-sound () (lissajous 0 20 :nch 4 :cycles 2))`

     Now we need to add some "reverb". There are several kinds of reverbs on Snd. You can take a choice between "nrev", "jc-reverb" or "freeverb". For now let's choose "freeverb".

  4. `(load "freeverb.scm")`

     Set the variable **clm-reverb-channels** to default four channels.

  5. `(set!  *clm-reverb-channels* 4)`

     Listen to the full location cues by adding "freeverb" reverb:

  6. `(with-sound (:reverb freeverb) (lissajous 0 40 :nch 4))`

     J. Chowning *Lissajous* equations tend work better on lengthier sounds. Below an example of several traces.

  7. `(with-sound (:reverb freeverb) (lissajous 0 80 :nch 4 :cycles 2 :rev-amt 0.0125))`

Listen carefully to the sound generated in four channels and try to perceive its motion on the listening space. Here there is the illusion of an imaginary space beyond loudspeakers. Change parameters on function calls to hear different parameters. Furthermore, hack the above code so that sound is something different than just whistling noise. Read a mono sound file and diffuse it everywhere. But think spatialization and why does it need to fit in a particular composition.

9.13.9 Snd's Initialization File

Snd can be customized and extended; see here. There is an initialization file, called "$\sim$ /*.snd_s7*". This is an optional file containing any customizations or extensions that need to be loaded whenever Snd starts.

An example of a "$\sim$ /*.snd_s7*" init file with customization an extensions could be something like: (more below : §9.13.9)

```
;; options:
;; ========

(set! *load-path* (cons "." (cons "/usr/lib64/snd/scheme" *load-path*)))  ;; Snd's scheme path


(set! (window-width) 600)           ;these set the initial window size
(set! (window-height) 400)
(set! (window-x) 160)
(set! (window-y) 0)

(if (provided? 'snd-motif)          ;Motif and Gtk use different font naming conventions
    (begin
      (set! *listener-font* "9x15")
      (set! *axis-label-font* "-*-times-medium-r-normal-*-18-*-*-*-*-*-*-*")
      (set! *axis-numbers-font* "9x15"))
    (begin
      (set! *listener-font* "Sans 10")
      (set! *axis-label-font* "Times Medium 14")
      (set! *axis-numbers-font* "Sans 10")))

(set! *listener-prompt* "> ")        ;change listener prompt from the default ">" to ":"
;; (set! (show-listener) #t)         ;include the listener window initially


(define beige (make-color 0.96 0.96 0.86))
(define blue (make-color 0 0 1))
(set! *selected-graph-color* beige) ;selected graph background is beige
(set! *selected-data-color* blue)   ;selected graph data is blue

(set! *save-dir* "/zap/")        ;save-state files are placed in /zap/snd
(set! *temp-dir* "/zap/")        ;temp files are placed in /zap/tmp
;;; (set! *peak-env-dir* "/zap")


;; load extensions:
;; ================

;; (if (not (provided? 'snd-extensions.scm)) (load "extensions.scm"))
(if (not (provided? 'snd-selection.scm)) (load "selection.scm"))
;; (if (not (provided? 'snd-hooks.scm)) (load "hooks.scm"))

(load "hooks.scm")
(load "extensions.scm")
```

```
(load "dsp.scm")
hook-push after-open-hook          ;if sound has many chans, use just one pane for all
  (lambda (hook)
    (let ((snd (hook 'snd)))
      (if (> (channels snd) 4)
          (set! (channel-style snd) channels-combined)))))

;;
;;  customization:
;;  ==============
;;
;; (set! (show-controls) #t)

(set! *with-inset-graph* #t)        ;display an overview of the current window in the upper rig
(set! *with-pointer-focus* #t)      ;automatically focus on (activate) the widget under the mous

(set! *selection-creates-region* #f) ;turn off automatic region creation
;; (set! (enved-clip?) #t)
;; (set! (enved-wave?) #t)
(set! *show-y-zero* #t)

;;
;;;;  bound keys: these keys use the meta-key combination.
;;

(bind-key "End" 0
          (lambda()
            "view full sound"
            (set! (x-bounds) (list 0.0 (/ (framples) (srate))))))
;;
(bind-key #\r 0
          (lambda()
            "update sound"
            (update-sound)
            keyboard-no-action))
;;
(bind-key #\c 0
          (lambda()
            "appply controls"
            (apply-controls)
            keyboard-no-action))

;;
(bind-key #\n 0
          (lambda()
            "scale to normalize to 0.8"
            (scale-to 0.8)
            keyboard-no-action))
;;
(bind-key #\w 0
          (lambda()
            "save sound as wav-riff file"
             (save-sound-as "test.wav"  :sample-type mus-lshort :header-type mus-riff)
            ;; (scale-to 0.8)
            keyboard-no-action))
;;
(bind-key #\v 0
          (lambda()
            "save sound as wav-riff file 32 bits little endian"
             (save-sound-as "new.wav"  :sample-type mus-lint :header-type mus-riff)
            ;; (scale-to 0.8)
            keyboard-no-action))
;;
(bind-key #\u 0
          (lambda()
            "up scale sample rate and save sound as wav-riff file"
            (src-sound (/ 48000 44100) 5)
            (save-sound-as "upsrc.wav" :srate 48000 :sample-type mus-bfloat :header-type mus-ri
            keyboard-no-action))
```

If you dig into this file, there are three sections namely, options, extensions, and customization. Try to focus on the comments on the lines and options and extensions are self explanatory. However, the bound keys below customization deserve some explanation. Almost any function or any aspect on Snd can be customized. Here we are assigning keys to Snd functions, they work just like Emacs key bindings but in this case, they only use the meta-key combination. Below some descriptions to the assigned bind-key combinations:

1. meta-end :: view entire waverform of sound on screen
2. meta-r :: update changes to soundfile
3. meta-c :: apply manipulation on controls to soundfile
4. meta-n :: normalize or scaled entire amplitude to 0.8
5. meta-w :: save sound as a “WAV” (riff) soundfile.
6. meta-v :: save sound as a 32 bits little endian “WAV” (riff) soundfile
7. meta-u :: up-scale sample rate and save sound as wav-riff file

Tons of other functions on Snd can be customized to particular needs this way. Start from HERE.

9.13.10 More about SND

Anything inside SND can be customized. New commands and configuration options can be added in much the same way that emacs or xemacs can be extended through its own built in lisp interpreter. You can have several of your own customization in the .snd file in your home directory.

Make sure to look and pay **attention !** to the SND documentation for more illustrated and elaborate examples. We all encourage you to create your own “.snd” dot-snd file and place it in your home directory. The file examp.scm in the snd sources contains also many ways, explanations and of course examples on how to get around SND.

If you are used to Mac or Windoze "soundfile editors" or if you are somehow more curious about SND, Dave Phillips has published a good description, comparison and customization article about SND available online at:

https://web.archive.org/web/20011102184425/http://linux.oreillynet.com/pub/a/linux/2001/10/05/snd_partone.html

9.14 Sound-Utilities

- SoX
  http://sox.sourceforge.net/

  > is a sound file format converter for Unix.It also does sample rate conversion and some sound effects. It's the swiss army knife of sound tools: the interface isn't great, but it does almost everything.
  >
  > SoX uses file suffices to determine the nature of a sound sample file. If it finds the suffix in its list, it uses the appropriate read or write handler to deal with that file. SoX has an auto-detect feature that attempts to figure out the nature of an unmarked sound sample. This is the 'auto' file format.

  Here are some examples on how to use SoX at the shell prompt but first make sure you have the correct file suffix.

  SoX does not like the **".snd"** file suffix (extension), therefore you need to change your sound-file extension to Sun's **"au"** extension. You can do this with the `mv` command like,

  ```
  mv File.snd file.au
  ```

  To translate a sound-file in SUN Sparc ".AU" format into a Microsoft ".WAV" or "Riff" file you do,

  ```
  sox File.au file.wav
  ```

  The following example lowers the amplitude of "ifile.au" by half and establishes a sample rate of "22.05K" to the Macintosh compatible file "ofile.aiff"

  ```
  sox -v 0.5 ifile.au -r 22050 ofile.aiff
  ```

  You can use the '-V' option on all your command lines. It makes SoX print out its idea of what is going on. To add a low-pass filter (note use of stdout for output of the first stage and stdin for input on the second stage) you can use Unix pipes in the following fashion,

  ```
  sox infile.wav -t raw -s -w -c 1 - lowpass 3700  |
    sox  -t  raw -r 11025 -s -w -c 1 - -t au -r 8000 -U -b
    -c 1 ofile.au
  ```

  There are more SoX examples in the CD mastering section.

- sndlib
  http://ccrma-www.stanford.edu/software/snd/sndlib/

  > The "sndlib" sound library is a collection of sound file and audio hardware handlers written in C. It provides relatively straightforward access to many sound file headers and data types, and most of the features of the audio hardware. Its manual is available online at "http://www-ccrma.stanford.edu/software/snd/snd/sndlib.html" .

Some command line example programs written with sndlib (that are part of the snd package) are quite useful on their own:

sndplay	*play a soundfile*
sndinfo	*print information about a sound file*
sndrecord	*record a mono or multichannel sound*
sndsine	*generate a sinusoidal signal*

For example if you want to listen to a sound file you can type:

sndplay soundfile.snd
sndplay soundfile.wav

If you need information about a particular, let's say the number of channels, you type:

sndinfo soundfile.au
sndinfo soundfile.aiff

- resample
http://ccrma.stanford.edu/ jos/resample/

    is a much needed and useful tool for high quality sampling rate conversion that Julius Smith wrote a long time ago (during the NeXT days). You use resample in the command line by writing something like:

```
resample -to 44100 clm.snd cd-ready.snd
```

The resample program takes a 16-bit mono or stereo sound file and a "sampling- rate conversion factor" r, specified as a floating-point number, and produces an output sound file whose sampling rate is r times that of the input file.

The output file is in AIFF format. If you omit the output file name, resample will create a file using the input file name, with ".resamp" appended.

Use *resample* to create CD-ready soundfiles or transfer from one medium to another. Beware that the default sampling rate in *"clm"* is set to 22050 samples per second because in this way sounds are faster to render. You can use *resample* to get your soundfile up to CD rate and then use SOX to get the correct timing and WAV "riff" formats for burning a CD.

- Ecasound
http://www.eca.cx/ecasound/

    This is a full software package designed for multitrack audio processing. It can be used for simple tasks like audio playback, recording and format conversions, as well as for multitrack effect processing, mixing, recording and signal recycling. Ecasound supports a wide range of audio inputs, outputs and effect algorithms. Effects and audio objects can be combined in various ways, and their parameters can be controlled by operator objects like oscillators and MIDI-CCs. As most functionality is located in shared libraries, creating alternative user-interfaces is easy. A versatile console mode interface is included in the package.

Ecasound has a text user interface that includes an interactive interpreter, see the man page for details. The man page documents the some additional commands:

ecaplay	*plays input soundfiles*
ecaconvert	*converts a set of files to a given format*
ecafixdc	*fixes DC offsets in a soundfile*
ecanormalize	*normalizes a soundfile*
ecasignalview	*monitors input signal level and statistics*

Comprehensive customization can be done by creating a configuration file in your home directory, see the man page for details.

Since Ecasound is a synchronized ALSA application it might prove very useful in special audio situations which require a high level of audio performance. Perhaps in a real time "Tape" concert situation or if you find yourself in a situation where you want to transfer a multi-channel sound-file to tape media like ADAT or DTRS, and if you want to avoid digital dropouts, ecasound is like a Swiss Army audio tool. In addition to sound-file recording or playback you can also do digital signal processing.

Please make sure about the audio-header-type and the sampling rate of your sound-file. You can do this at CCRMA by issuing *ONE* of the following commands:

sndinfo soundfile.wav
sndinfo soundfile.snd

Ecasound only likes Riff or Wav files and obviously doesn't like NeXT's snd files. If you need to convert your sound-file you can use the `sox` command for up to 2-channel stereo files. If your sound-file is a snd file you need to rename it and convert it in the following way:

mv soundfile.snd soundfile.au
sox soundfile.au sox soundfile.wav

For multichannel sound-files you might want to use the srconvert functions of SND or CLM. A "sr-convert.ins" instrument invocation in CLM might look like this (for more information please consult CLM or SND documentation).

```
(with-sound (:output "new.wav"
             :header-type mus-riff
             :srate 44100
             :channels 4
             :play nil)
    (sr-convert 0 "~/zap/test.snd" :width 7))
```

Please notice the "header type" key inside with-sound it should be "mus-riff" to write multichannel wav files. Once you have your .wav file you just use ecasound in *interactive* mode with the following command:

```
ecasound -c -i new.wav -f 32,10,44100 -o alsahw,0,0
```

At this point and inside the ecasound command line (because of the `-i` switch) , you can start, forward and rewind your sound-file from almost any point with any offset.

Other examples of playback might be in the following renditions:

1. With JACK a two channel sound file and 44100 sampling rate.

   ```
   ecasound -f:32,2,44100 -i new.wav -o jack_alsa
   ```

2. A JACK client that has one input and one output port.

   ```
   ecasound -f:32,1,44100 -i jack -o jack \
      -ef3:800,1.5,0.9 -km:1,400,4200,74,0 -km:2,0.1,1.5,71,0
   ```

3. Realtime inputs (recording from soundcard)

```
ecasound -i alsahw,0,0 -o somefile.wav
```

4. Realtime Outputs (soundcard playback)

```
ecasound -i somefile.mp3 -o alsahw,0,0
```

More examples can be found in the Ecasound documentation or at its web page.

- ecawave
  http://www.wakkanet.fi/ kaiv/ecawave/welcome.html

  > is a simple graphical audio file editor. The user-interface is based on Qt libraries, while almost all audio functionality is taken directly from ecasound libraries. As ecawave is designed for editing large audio files, all processing is done direct-to-disk. Simple waveform caching is used to speed-up file operations. Ecawave supports all audio file formats and effect algorithms provided by ecasound libraries. This includes OSS, ALSA, aRts, over 20 file formats, over 30 effect types, LADSPA plugins and multi-operator effect presets."

- cdparanoia
  http://www.xiph.org/paranoia/

  > retrieves audio tracks from CDDA capable CDROM drives. The data can be saved to a file or directed to standard output in WAV, AIFF, AIFF-C or raw format. Most ATAPI, SCSI and several proprietary CDROM drive makes are supported; cdparanoia can determine if the target drive is CDDA capable.

  In addition to simple reading, cdparanoia adds extra-robust data verification, synchronization, error handling and scratch reconstruction capability.

- STK
  http://www-ccrma.stanford.edu/software/stk/

  > "The synthesis toolkit is a set of audio signal processing C++ classes and instruments for music synthesis. You can use these classes to create programs which make cool sounds using a variety of synthesis techniques. This is not a terribly novel concept, except that STK is very portable (it's mostly platform-independent C and C++ code) AND it's completely user-extensible. So, the code you write using STK actually has some chance of working in another 5-10 years. STK currently runs with realtime support (audio and MIDI) on SGI (Irix), Linux, and Windows computer platforms. Generic, non-realtime support has been tested under NeXTStep, but should work with any standard C++ compiler."

- SMS
  http://www.iua.upf.es/ sms/

  > "SMS is a set of techniques and software implementations for the analysis, transformation and synthesis of musical sounds. The aim of this work is to get general and musically meaningful sound representations based on analysis, from which musical parameters might be manipulated while maintaining high quality sound. These techniques can be used for synthesis, processing and coding applications, while some of the intermediate results might also be applied to other music related problems, such as sound source separation, musical acoustics, music perception, or performance analysis."

If you feel you are into Linux and Sound or simply for general knowledge make sure you take a look at Larry Ayers' *Soundings: Explorations In Linux Sound*
http://www.linuxgazette.com/issue47/ayers.html

9.15 Sound-Compression

The PCM (pulse-code modulation) established for CD-DA discs is not very compact and doesn't suit for delivering music via the net. That is why developers now are working on a number of complex compression algorithms. All of them differ very much in the sound quality, that is why a user has always to make a choice of an algorithm for its favorite music to be recorded.

- Flac:

  FLAC stands for Free Lossless Audio Codec, an audio format similar to MP3, but lossless, meaning that audio is compressed in FLAC without any loss in quality. This is similar to how Zip works, except with FLAC you will get much better compression because it is designed specifically for audio, and you can play back compressed FLAC files in your favorite player.

  If you have a "WAV" or "riff" file you can do lossless compression with this command:

  ```
  flac -7 yourfile.wav
  ```

  Note that the '-7' flag is high quality compression level. You can read more about FLAC on it (man-page).

  Metadata can be added to the FLAC file with something like:

  ```
  flac -T "TITLE=Freddie the freeloader" -T "ARTIST=Miles Davis" your.wav
  ```

- Ogg Vorbis
  http://www.vorbis.com

  > is a fully Open, non-proprietary, patent-and-royalty-free, general-purpose compressed audio format for high quality (44.1-48.0kHz, 16+ bit, polyphonic) audio and music at fixed and variable bitrates from 16 to 128 kbps/channel. This places Vorbis in the same class as audio representations including MPEG-1 audio layer 3, MPEG-4 audio (AAC and TwinVQ), and PAC.
  > Ogg Vorbis provides a high-quality format for you to listen to your music. Its file size is also smaller than MP3 and getting smaller as development continues. Vorbis already enjoys widespread player support and should be compatible with several major hardware players soon. With Vorbis, you can listen to your music with higher quality in less space. Also, using Vorbis means your player and encoder choices aren't bound by licensing terms.

  – *oggenc*
    is part of the "vorbis-tools" package. This program is a complete encoder that creates Ogg Vorbis compressed soundfiles. OggEnc input files must currently be 16 or 8 bit PCM WAV, AIFF, or AIFF/C files. Files may be mono or stereo (or more channels) and sampling rates between 8kHz and 56kHz. Call oggenc with the "-h" flag or see the oggenc man page for more details.
    This will create a "sample.ogg" output soundfile:

    ```
    oggenc sample.wav
    ```

    You can do a format conversion and pipe that to oggenc using unix pipes, for example the following command will convert a soundfile from .snd format to .wav and feed it to the encoder:

    ```
    sox -t .au sample.snd -t .wav - | oggenc -o sample.ogg -
    ```

Get high-quality encoding averaging 256 kbps (but still VBR):

```
oggenc infile.wav -b 256 -o out.ogg
```

VBR: Variable bit rate.

Metadata (title and author can be added to an Ogg file as follows:

```
 oggenc audio-cd.wav -q 8 -t "Freddie the freeloader" -a "Miles Davis" -o freddie.ogg
```

- *ogg123*
  is also part of the "vorbis-tools" package. In essence it is simple command line Ogg Vorbis decoder and player. You can play your "ogg files" with ogg123 with:

  ```
  ogg123 -d oss sample.ogg
  ```

- Play your Ogg files with xmms
  http://www.xmms.org
  The X Multimedia System (xmms) can recognize and play Ogg Vorbis encoded files. Just open xmms as follows:

  ```
  xmms sample.ogg &
  ```

- mp3
  http://www.mp3licensing.com

  does not need an introduction. It is a stereo compression standard that has become widespread in hardware and firmware based players, and is widely used for web distribution of music content. Regretfully even though it is a standard, it is not for free. Patent licenses under the combined patent portfolio of Fraunhofer IIS-A and Thomson multimedia are granted by Thomson multimedia exclusively and cover format encoders and decoders and commercial (revenue-generating) distribution of music. Read all the details in http://www.mp3licensing.com.

  According to the patent owners it is not possible to independently create an mp3 encoder without infringing on the patents. Recently they have started to crack down on open source projects that have created free alternatives to the commercial mp3 encoders. Read more about this in the lame (http://www.sulaco.org/mp3) and bladeenc (http://bladeenc.mp3.no/) web sites

  A couple of widely available encoders are available in PlanetCCRMA, but are only intended to be used as tools for learning about mp3. If you encode your own mp3s you should own a properly licensed encoder.

  - lame
    http://www.sulaco.org/mp3

    > means, Lame Ain't an MP3 Encoder and is an educational tool to be used for learning about MP3 encoding. The goal of the LAME project is to use the open source model to improve the psycho acoustics, noise shaping and speed of MP3. Run lame with the "–help" option or see the lame man page for more details.

    Audio files created with Lame can be played back by popular MP3 players such as Xmms mpg123 or madplay.
    For example a command for a fixed bit rate jstereo 128kbs encoding, highest quality :

    ```
     lame -h sample.wav sample.mp3
    ```

    A good choice for Music with a broad dynamic range might be :

```
lame -abr sample.wav sample.mp3
```

– Compressing audio with Lame presets

  The Lame –preset switches are designed to provide the highest possible quality. These are continually updated to coincide with the latest developments that occur and as a result should provide you with nearly the best quality currently possible from LAME.

  To activate these presets:

  1. `--preset standard`
     This preset should generally be transparent to most people on most music and is already quite high in quality.
  2. `--preset extreme`
     If you have extremely good hearing and similar equipment, this preset will generally provide slightly higher quality than the standard mode.
  3. `--preset insane`
     This preset will usually be to much for the common user and most situations, but if you must have the absolute highest quality with no regard to filesize, this is the way to go.
  4. `--preset  kbps`
     Using this preset will usually give you good quality at a speci- fied bitrate. Depending on the bitrate entered, this preset will determine the optimal settings for that particular situa- tion. While this approach works, it is not nearly as flexible as VBR, and usually will not attain the same level of quality as VBR at higher bitrates.

  For example to use the extreme preset you will use this command:

```
lame --preset extreme infile.wav outfile.mp3
```

  Add metadata tags to your mp3 file:

```
lame --preset standard --tt "Mars" --ta "Gustav Holst" cd.wav mars.mp3
```

– Converting MP3s to raw audio to create audio CDs

```
mpg123 -s file.mp3 | sox -c2 -s -w -t raw \-r 44100 - -t wav - > newsoundfile.wav
```

  mpg123 converts file.mp3 to the .wav file newsoundfile.wav and makes it ready to be part of audio CDs.

– bladeenc
  http://bladeenc.mp3.no/

  is a freeware MP3 encoder. It is based on the same ISO compression routines as mpegEnc, so you can expect roughly the same, or better, quality . The main difference is the appearance and speed. BladeEnc doesn't have a nice, user-friendly interface like mpegEnc, but it is more than three times faster, and it works with several popular front-end graphical user interfaces. Run bladeenc without arguments to get a usage description. Unfortunately BladeEnc is not longer in development but its features appear to be part of the Ogg team now.

9.16 Composition Environments

• CLM
  http://www-ccrma.stanford.edu/software/clm/

  Common Lisp Music is a music synthesis and signal processing package in the Music V family. CLM provides functions to experiment with sounds and although you can use CLM simply as a bunch of canned functions, it's a lot more fun to make your own. In CLM, these are called "generators" and "instruments", and a sequence of instrumental calls is a "note list". To create your own generators and instruments, you need to write the Lisp function that expresses in CLM's terms the sound processing actions you want.

- CM
  http://www-ccrma.stanford.edu/software/cm/cm.html

  Common Music is an object-oriented music composition environment. It produces sound by transforming a high-level representation of musical structure into a variety of control protocols for sound synthesis and display. Common Music defines an extensive library of compositional tools and an API through which the composer can easily modify and extend the system.

- CMN
  http://www-ccrma.stanford.edu/software/cmn/

  Common Music Notation is a free western music notation package written in Common Lisp that can create and display traditional western music scores. `cmn` is the main Lisp function and it reads in all its arguments, organizes the musical data into systems and staves, adds line and page breaks, beams, ties, slurs, dynamics, and so on, aligns and justifies the result, and as a result produces an "encapsulated Postscript" file.

- IMPROV
  http://improv.sapp.org/

  is a C++ environment for writing programs that enable musician/computer interaction using MIDI instruments. Improv programs can be written in special pre-defined environments, or they can be written from scratch using just the basic MIDI input and output classes.

  Example programs are provided which demonstrate how to use the Improv library. The programs range from simple programs such as one which switches the key number and attack velocity parameters of a MIDI message (switch1) to more complicated programs such as one which statistically analyzes the input notes to estimate the musical key of the performance (keyan).

- PD
  http://www.pure-data.org/

  stands for "pure data". Pd is a real-time software system for live musical and multimedia (video) performances. It is in active development by Miller Puckette, and perhaps others. The system is unfinished, but quite useable for sophisticated projects.The best documentation can only be viewed by running Pd. Click on the "Pure Documentation" menu item on the "Help" menu when you run Pd.

  There is official documentation as well as useful links for pd under the software guides section of the CCRMA home web page at pd guide@ccrma
  http://www-ccrma.stanford.edu/guides/package/pd/ To run Pd just type,

  ```
  pd -alsa &
  ```

  or just,

  ```
  pd &
  ```

  This will start Pd and a new Pd "GTK" box will appear. You can test you audio settings, MIDI settings and see if your audio system is running properly. DIO Digital input output errors might appear while you are running Pd, please pay attention to them since normally these mean audio input and output synchronization errors.

You can try various Pd switch options in order to tune and optimize your setup. Here are the most common flags which you can see with the command:

```
pd -help
```

-listdev	list audio and MIDI devices
-noadc	suppress audio input
-nodac	suppress audio output
-audioindev	audio in devices
-audiodev	specify input and output together
-audiobuf (n)	specify size of audio buffer in msec
-blocksize (n)	specify audio I/O block size in sample frames
-midiindev	midi in device list
-nomidiin	suppress MIDI input
-nomidi	suppress MIDI input and output
-oss	use OSS audio API
-alsa	use ALSA audio API
-jack	use JACK audio API (default for Linux)

Pd also works with video signals provided your workstation has a video capture board. You can combine MIDI, audio, signal processing and also video by using an extra library called GEM.

To run Pd with GEM just type,

```
pd -lib /usr/lib/pd/externs/Gem
```

(Thanks to Bill Verplank who first tested this command at CCRMA).

GEM is the Graphics Environment for Multimedia. It was written by Mark Danks to generate real-time computer graphics, especially for audio-visual compositions. Because GEM is a visual programming environment, users do not need any experience in traditional computer languages.

GEM is a collection of externals which allow the user to create OpenGL graphics within Pd, a program for real-time audio processing by Miller Puckette (of Max fame).

GEM currently has many different shapes and objects, including polygonal graphics, lighting, texture mapping, image processing, and camera motion. All of this is possible in real-time without any previous programming experience. Because GEM is an add-on library for Pd, users can combine audio and graphics, controlling one medium from another.

more information can be found at:

GEM SITE
http://iem.kug.ac.at/GEM/

9.17 MIDI-and-Sound

- JACK
  http://jackit.sourceforge.net

  is a started a terminal window by issuing one the following commands depending on the sample rate:

  jackstart -d alsa -d hw -r 44100
  jackstart -d alsa -d hw -r 22050

*Please see the **JACK section** for an explanation and the options in the jackstart command.*

In the case of a real-time situation like a concert situation, you can give JACK priority with the following options in the jackstart command:

jackstart –realtime –driver=alsa

When you are done with your audio or sound job make sure quit quit or stop the JACK process by issuing the: `C-c [control-c] sequence`

- PD
  http://www.pure-data.org/

  Please see the Composition Environments section.

- playmidi
  http://sourceforge.net/projects/playmidi/

  is a curses and X11-based MIDI file player for Linux. It supports playback to any OSS-supported synth device including external MIDI.

  To play a MIDI file through a external synthesizer you should type something like:

  ```
  playmidi -e /zap/test.mid
  ```

- Timidity
  Timidity is a software synthesizer that can play MIDI files by converting them into PCM waveform data.

  Timidity is a MIDI to WAVE converter and player that uses Gravis Ultrasound(*)-compatible patch files to generate digital audio data from general MIDI files. The audio data can be played through any sound device or stored on disk. On a fast machine, music can be played in real time. Type,

  ```
  timidity /zap/test.midi
  ```

  to hear a synthesized version for your MIDI file or to run timidity as a background process.

  ```
  timidity -iA &
  ```

  As being designed with "general MIDI" and with ALSA, Timidity can play in real time with more elaborate sounds and options as follows:

  ```
  timidity -Os -A130 -p3 -Ei5 -module=2 -q20 -T140 test.midi
  ```

  The options in this case mean:

  – -Os , use Alsa
  – -A100, , scale amplitude by factor(100)
  – -p3 , 3 voice polyphony

  - -Ei4 , sound (patch) 4
  - -module=2 , Roland 88 behaviour
  - -q20 , audio buffer size of 20/100 default 5/100
  - -T140 , tempo of 140 bps

- muse
  http://muse.seh.de/

  is a MIDI/Audio sequencer with recording and editing capabilities and the following features:

  1. standard midifile (smf) import-/export
  2. organizes songs in tracks and parts which you can arrange with the part editor
  3. midi editors: pianoroll, drum, list, controller
  4. score editor with high quality postscript printer output
  5. realtime: editing while playing
  6. unlimited number of open editors
  7. unlimited undo/redo
  8. realtime and step-recording
  9. multiple midi devices
  10. unlimited number of tracks
  11. audio tracks, LADSPA host for master effects
  12. multithreaded
  13. uses raw midi devices (ALSA, OSS & serial ports)
  14. XML project file
  15. project file contains complete app state (session data)
  16. Application spanning Cut/Paste Drag/Drop
  17. C++
  18. QT2 GUI Library
  19. STL

  Note: If you don't have a synthesizer on hand at your workstation you can take advantage of TiMidity to listen your MIDI file in the MusE time-line or track window. For this you need to have timidity as a background process as described above and while in MusE go to the "config" menu and select the "MIDI ports" command. A new window will appear and in any of the ports you will notice that if you select the device name pull down menu, there are two options for ports of timidity. Once selected just play your MIDI sequence and listen with your favorite TiMidity sound.

9.18 Graphics, vector-based

- dia
  http://www.lysator.liu.se/ alla/dia/
  can be used to draw many different kinds of diagrams. It currently has special objects to help draw entity relationship diagrams, UML diagrams, flowcharts, network diagrams, and simple circuits. It can load and save diagrams to a custom XML format (gzipped by default, to save space), can export diagrams to EPS or SVG formats and can print diagrams (including ones that span multiple pages).

– xfig
http://www.xfig.org/

is an interactive drawing tool which runs under the X Windows System. In xfig, figures may be drawn using objects such as circles, boxes, lines, spline curves, text, etc. It is also possible to import images in formats such as GIF, JPEG, EPSF (Postscript), etc. Those objects can be created, deleted, moved or modified. Attributes such as colors or line styles can be selected from various options. For text, various fonts are available.

You can start xfig on the command line by typing,

```
xfig &
```

– xcircuit
http://www.artsoft.com.br/linux/ldpp/en/app/xcircuit/

is flexible enough to be used as a generic program for drawing just about anything, and is competitive with powerful programs such as ”xfig”. It is especially good for any task requiring repeated use of a standard set of graphical objects, including architectural drawing, printed circuit board layouts, and music typography. XCircuit is a program for drawing publishable-quality electrical circuit schematic diagrams and related figures, and produce circuit net lists through schematic capture. XCircuit regards circuits as inherently hierarchical, and writes both hierarchical PostScript output and hierarchical SPICE net lists. Circuit components are saved in and retrieved from libraries which are fully editable.

You can start xcircuit by typing the command,

```
xcircuit &
```

– OpenOffice
http://www.openoffice.org/

Please see the word processing / presentation section §9.6.1

– StarOffice
http://www.sun.com/staroffice/

Please see the word processing / presentation section §9.6.2

– gnuplot
http://www.cs.uni.edu/Help/gnuplot/

is a command-driven interactive function plotting program. It can be used to plot functions and data points in both two- and three-dimensional plots in many different formats, and will accommodate many of the needs of today's scientists for graphic data representation. It features plotting of two-dimensional functions and data points in many different styles (points, lines, error bars). It allows computations in integer, float and complex arithmetic plotting of three-dimensional data points and surfaces in many different. Styles (contour plot, mesh) and support for complex arithmetic can also be added. It has self - defined functions support for a large number of operating systems, graphics file formats and devices. Please look for the extensive on-line help labels for title, axes, data points command line editing and history on most platforms.

9.19 Signal-Processing

– matlab
http://www.mathworks.com/

integrates mathematical computing, visualization, and a powerful language to provide a flexible environment for technical computing. MATLAB handles a range of computing tasks in engineering and science, from data acquisition and analysis to application development.

– octave
http://www.che.wisc.edu/octave/
is a high-level language, primarily intended for numerical computations. It provides a convenient command line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with Matlab. It may also be used as a batch-oriented language.
GNU Octave has extensive tools for solving common numerical linear algebra problems, finding the roots of nonlinear equations, integrating ordinary functions, manipulating polynomials, and integrating ordinary differential and differential-algebraic equations. It is easily extensible and customizable via user-defined functions written in Octave's own language, or using dynamically loaded modules written in C++, C, Fortran, or other languages.

– STK
http://www-ccrma.stanford.edu/software/stk/

Please see the MIDI and sound section for more information in regards to stk and don't forget to visit its web page.

– SMS
http://www.iua.upf.es/ sms/

Please see the sound utilities section for more information about sms. There is a front end web implementation that you should try in order to get more familiar with Spectral Models.

9.20 Development Environment

– gcc
http://gcc.gnu.org/

several versions of the compiler (C, C++, Objective C, Fortran, Java) are integrated in a set and this is why people use the name GCC or *"GNU Compiler Collection"*. GCC can compile programs written in any of these languages. The Fortran and Java compilers are described in separate manuals.

When you invoke GCC, it normally does preprocessing, compilation, assembly and linking. The "overall options" allow you to stop this process at an intermediate stage. For example, the '-c' option says not to run the linker. Then the output consists of object files output by the assembler.
Other options are passed on to one stage of processing. Some options control the preprocessor and others the compiler itself. Yet other options control the assembler and linker; most of these are not documented here, since you rarely need to use any of them.
Most of the command line options that you can use with GCC are useful for C programs; when an option is only useful with another language (usually C++), the explanation says so explicitly. If the description for a particular option does not mention a source language, you can use that option with all supported languages.
Please consult the man page for gcc.

– gdb
http://gdbwww.gdb.org/
is the GNU Project debugger, allows you to see what is going on 'inside' another program while it executes – or what another program was doing at the moment it crashed.
GDB can do four main kinds of things (plus other things in support of these) to help you catch bugs in the act:

1. Start your program, specifying anything that might affect its behavior.

2. Make your program stop on specified conditions.
3. Examine what has happened, when your program has stopped.
4. Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.

9.21 Spelling Checkers

– ispell
http://ficus-www.cs.ucla.edu/geoff/ispell.htm
is a fast screen-oriented spelling checker that shows you your errors in the context of the original file, and suggests possible corrections when it can figure them out. Compared to UNIX spell, it is faster and much easier to use. Ispell can also handle languages other than English.
Aspell is a spelling checker written by Kevin Atkinson. Its primary advantage is that it is better at making suggestions when a word is seriously misspelled. For example, when given ”trubble”, ispell will suggest only ”rubble”, where aspell suggests ”trouble” (as its first choice” as well as ”dribble”, ”rubble”, and a lot of other words. Its disadvantage is that the approximate-matching algorithm is specific to English.

You normally use ispell or aspell inside the Emacs environment, actually in the mini buffer. You can invoke spell checking with `M-x ispell-region` *or* `M-x ispell-buffer`*. The sequence* `M-s` *will usually do spell checking on a “marked” region. Please go to the silly emacs tricks section for more on these combinations.*

9.22 Postscript and other Graphic Formats

– gv
http://wwwthep.physik.uni-mainz.de/ plass/gv/
allows to view and navigate through PostScript and PDF documents on an X display by providing a user interface for the ghostscript interpreter.

```
gv /zap/test.ps &
```

will open the PostScript file /zap/test.ps

– ghostscript
http://www.cs.wisc.edu/ ghost/
is an interpreter for the PostScript (TM) language. A PostScript interpreter usually takes as input a set of graphics commands. The output is usually a page bitmap which is then sent to an output device such as a printer or display. PostScript is embedded in many printers.
Ghostscript has several main uses:

1. Display a PostScript file.
2. Display a PostScript file to decide if you really need to print it.
3. Print a PostScript file to a non-PostScript printer.

– acroread
http:// www.adobe.com/products/acrobat/
Acrobat Reader enables you to view, print, navigate, and browse PDF files seamlessly on its own or within leading Web browsers such as Netscape Navigator or Mozilla.
Adobe Portable Document Format (PDF) is a standard for electronic document distribution worldwide. Adobe PDF is a universal file format that preserves all the fonts, formatting, graphics, and color of any source document, regardless of the application and platform used to create it. Adobe PDF files are compact and can be shared, viewed, navigated, and printed exactly as intended by anyone with Acrobat® Reader® software.

9.23 USB External drives and Thumb drives

A USB "thumb drive" or sometimes called a "flash drive" is a data storage device that includes flash memory with an integrated USB interface. It is typically removable, re-writable and much smaller than CDs or DVDs, optical discs. Additionally, it is more flexible, portable, and holds more memory. Typically USB-Drives are used to backup contents of directories and files in order to have safe copies of information. Some people use them to transfer information from one computer to the other. Today most computer have USB ports on which these drives can be plugged in. USB drives are usually formatted with a windoze file system (e.g. FAT32) but can also be formatted to Linux ext-3 or ext-4 file systems. Of course Mac OSX also supports these drives. It important to note that Fedora Linux workstation at CCRMA can read and write data into windoze formmated thumb drives.

If you plug in a VFAT or even a FAT32 windoze filesytems USB thumb drive into a Linux workstation, as of Fedora 26 and above the drive will automatically mount into directory:

```
/run/media/your_user_name/usb-drive
```

- Gnome might also show an Icon of your drive on the desktop.

Once mounted, you have read and write permissions to your drive since Linux is seeing it as one of own your directories. You can go the content of your drive by changing directories such as:

```
cd /run/media/your_user_name/usb-drive
```

After your are done with the drive, you MUST unmounted. This is a bit tricky becuase depending on where your drive was mounted, on the command line you need to issue something like,

```
udisksctl unmount -b /dev/sdb1
```

You might need to change `'/dev/sdb1'` with whatever the device your drive is mount on. You can try finding about where the drive is with the `'fdisk -l'` command. Other options might be `'/dev/sdb2'` or `'/dev/sdc1'` and, so on. If none of these work, Gnome provides a way to unmount and eject portable drives by using Gnome's Nautilus folder and file manager.

But what about if you want to mimic one of your Linux directories or better, if you want to format a USB drive with a Linux file system?. If this is the case we can assume that you also have a Linux machine at home so that you administrative and root privileges. If this is not the case you might ask a staff member so that you can format your thumb drive with a Linux file system. Below we are showing how to format a drive with an ext-3 or ext-4 file system. However, you might need root privileges.

1. On the system identify the drive you want to FORMAT.
   Insert USB drive into your system and identify your USB drive correctly. This is a crucial step because formatting the wrong disk should be avoided at all costs. You can use the `'df -h'` command and it would show something like:

   ```
   df -h

   Filesystem      Size  Used Avail Use\%  Mounted on
   /dev/sda1       28G    24G  2.3G  92\%  /
   udev            1.4G   12K  1.4G   1\%  /dev
   ```

```
tmpfs           277M  1.2M  276M   1\%  /run
none            5.0M     0  5.0M   0\%  /run/lock
none            1.4G   34M  1.4G   3\%  /run/shm
/dev/sdb1      14.8G  1.4G  13.4G  10\% /run/media/cruzer}
```

2. Create a new partition with 'fdisk'.
   If your device is 'sdb1' as shown above run 'fdisk' like:

   ```
   fdisk /dev/sdb
   ```

   Once in 'fdisk', delete all partitions and create a new single partition. Linux ext3 partitions are the default in 'fdisk'. Write changes and quit 'fdisk'.
3. Format (create) the file system (for example ext-4).

   ```
   mkfs.ext4 /dev/sdb1
   ```

4. To label the drive (name it) and mount.

   ```
   e2label /dev/sdb1 cruzer
   ```

5. grant permissions on the thumb drive file system.

   ```
   cd /run/media/user_name/cruzer
   mkdir soundfiles
   chown user_name soundfiles
   ```

   'user_name' here is your login or username.

   You can organize your directories as above, but beware permissions and usernames.

NOTE: *Above directions also work for portable USB hard drives.*

9.24 Making-DVDs at CCRMA Workstations

You can backup a lot of data onto a single DVD. Perhaps sound-files, graphics, photos, presentations or even your $HOME directory. You can also author a digital movie on a DVD to play on household DVD players or on computer DVD drives. A Linux Journal article on DVD authoring might get you started on the subject of DVD video authoring. There are several tutorials and howto's on creating DVD-video. In general you might want to use dvdauthor. There is a tutorial on using dvdauthor at 'thoughts on DVD authoring'.

Since most of the time DVDs are used for data purposes here are some guidelines:

– DVD kinds, drives:
  There are several formats for DVDs. DVD-RAM, DVD-R, DVD+R, DVD-RW, DVD+RW plus dual layer versions of these. The distinctions are actually based on how data is written and read from the disk. Some manufacturers prefer one method over the other but in past months more and more drives are compatible with all standards. Other things to note is that DVD-R and DVD+R can only be recorded once while DVD-RW, DVD+RW can be recorded several times. In general if you are authoring and fixating movies, DVD-R are more compatible with home DVD players and legacy computer DVD drives.

DVD+R DL and DVD-R DL Dual layer are based on a technology which provides two individual recordable layers on a single-sided DVD disc. The dual layered discs can hold 7.95GB

DVD-RAM discs can be recorded and erased repeatedly but are compatible only with devices manufactured by the companies that support the DVD-RAM format. DVD-RAM discs are typically housed in cartridges.

– For burning DVDs

  ∗ "K3b is a CD and DVD burning application for Linux systems optimized for KDE window manager in Linux. It provides a comfortable user interface to perform most CD/DVD burning tasks like creating an Audio CD from a set of audio files or copying a CD. While the experienced user can take influence in all steps of the burning process the beginner may find comfort in the automatic settings and the reasonable K3b defaults which allow a quick start. The actual burning in K3b is done by the command line utilities cdrecord, cdrdao, and growisofs."
    With K3b you can create data CDs, audio Cds, Video Cds, Mixed Mode CDs, eMovix CD. You can do CD copy as well as CD ripping. Additionally you can do DVD Ripping and DivX/XviD encoding plus many other actions.
    K3b is on the Red Hat-Fedora or PlanetCCRMA menu under the Sound and Video applications. It can also be run on a terminal window with the 'k3b' command. Documentation for this program is huge and include several tutorials.

  ∗ "dvdauthor is a simple set of tools to help you author a DVD. The idea is to be able to create menus, buttons, chapters, etc, but for now you can just take an mpeg stream (as created by mplex -f 8 from mjpegtools 1.6.0) and write it to DVD."

  ∗ The man-page of growisofs states:
    " growisofs was originally designed as a frontend to mkisofs to facilitate appending of data to ISO9660 volumes residing on random-access media such as DVD+RW, DVD-RAM, plain files, hard disk partitions. In the course of development general purpose DVD recording support was implemented, and as of now growisofs supports not only random-access media, but even mastering of multi-session DVD media such as DVD+R and DVD-R/-RW. In addition growisofs supports first-/single-session recording of arbitrary pre-mastered image (formatted as UDF, ISO9660 or any other file system, if formatted at all) to all supported DVD media types."

  ∗ "X-CD-Roast provides a GUI interface for commands like cdrecord and mkisofs. X-CDRoast includes a self-explanatory X11 user interface, automatic SCSI and IDE hardware setup, support for mastering of new ISO9660 data CDs, support for production of new audio CDs, fast copying of CDs without hard disk buffering, and a log file option. You can use Xcdroast for graphical user interface to write DVDs but you need enter a ProDVD key in the setup." (see the Mastering Cd's section at §9.25).

– Commands for backing up and storing data on DVDs.

  The easiest way to burn or record data DVD's is with the growisofs command on a terminal window. Typically once you have your data directory burning is done in one step. Beware that permissions on the data directory are changed and therefore a 'tarball' is recommended provided that data size is not huge. You can have several tarballs though.

  The great advantage with the `'growisofs'` command is that you don't need to create an image of the data you want to store on a DVD. To master and burn an ISO9660 volume use the Rock-Ridge extensions in order to keep the filenames intact with a command that looks like:

```
growisofs -Z /dev/dvd -R dvd/
```

  Where `'dvd/'` is your data directory.

  If you have and iso image of the DVD type something like:

```
growisofs -dvd-compat -Z /dev/dvd=fc5.iso
```

Note: DVD ISO-images can be created with K3b.

9.25 Mastering Cd's

9.25.1 Xcdroast

Xcdroast
http://www.xcdroast.org/

X-CD-Roast is a graphical user interface (GUI) for the command-line set of cdr tools. With the X-CD-Roast front end things are a bit nicer and easier. The cdrtools set contains `cdrecord` which does the hard job supporting all the cd writers and also the actual writing of CDs, `readcd` which is a command line that reads data-tracks of CDs, `mkisofs` a utility that masters CD-/images from given file-trees on the hard disk and `cdda2wav` which reads audio-tracks.

You can master two kinds of CDs: audio or data. For audio you need to have **".wav"** format audio files in the /scratch directory. For data CDs you need to create an image of the files you want on your Cd also on /scratch . Once you have your audio files or images you can proceed to write them by accepting the track layout. Try to test your parameters by writing a Cd in simulation mode and once you are sure you can start duplicating your data or sound-files.

To burn an audio CD you can follow these procedures:

Make sure all the sound-files are in riff or **wav** format, and also 44.1KHz sampling rate (CD quality!). To find information about your sound-files you can type:

```
sndinfo yourfile.wav
```

or if you don't know the kind of sound-file you have, you can also type,

```
sndinfo yourfile.snd
```

Please make sure you find out about the sampling rate of your soundfile. At CCRMA you can use Julius Smith's high quality *resample* utility to get your sounfiles up to CD-rate. If you are not familiar make sure you read the sound utilities section or simply read the resample man page to create CD-ready soundfiles or if you transfer from one medium to another (i.e. dat, dtrs or adat to cd).

Beware that in general *"clm's"* default sampling rate is 22050 because of historical reasons and of course because of lower speed processors. Being the case with a stereo 22.05Ksr sound-file you can type the following options in order to have a CD-ready sound-file.

```
 resample -by 2.00 soundfile.snd cd.snd
```

Nevertheless, if you still need a quick solution you can use `sox` to do a reasonable job:

```
sox -t au -r 22050 yourfile.snd -r 44100 newfile.wav
```

or better

```
sox -t au -r 22050 yourfile.snd -r 44100 /zap/newfile.wav
```

To make sure your **wav** file has the correct timing for being a CD track, you also use `sox` with the next options:

```
sox oldfile.wav newfile.wav
```

or

```
sox -t au oldfile.snd newfile.wav
```

or

```
sox oldfile.aiff newfile.wav
```

Again, make sure the sampling rate is 44100 KHz.

Next, launch `xcdroast` on the command line and you should get its graphic interface. Make sure you have the correct cd-writer settings by clicking on the setup push button, and selecting the correct device for the machine you are using. Don't forget to save your configuration.

– To Master Audio Cd's

  Select the "Write Tracks" push button and you should get a new window with the list of tracks, provided they are on the **/scratch** directory. Notice there are two tabs above the track list and, select the layout tracks tab. Add the sound-files, change the order of your tracks and once done, click on the "Accept track layout" push button at the bottom. Now select the "Write tracks" tab to go back to the main Write tracks window and make sure you select the "track at once" (TAO) check mark or radio button. Finally click on the Write tracks push button to start burning your CD.

– To create Data Cd's

  Select the Create CD push button to start mastering your Data CD. A new window will appear with few more options. Select "Master Tracks" button and you will get yet another window with directories and may be information of your CD tracks. Explore this window and if necessary go to to the **/zap** or **/scratch** directories for the location of your files. Add directories or files and make sure they appear in the session list. Once done, you can select the "Create session/image" tab, "calculate the size" of your image and either "master to image file" or "Master and Write on-the-fly" push buttons.

  If you selected "Master and Write on-the-fly" the CD burning process will start, otherwise after the "image to file" process is finished, you can click on the big "Write Tracks" push button, select the layout tracks tab add your image file to the "tracks to write" list and finally click on the "accept this track layout" push-button. Select the write tracks tab and make sure your image file is on the tracks to write list. Make sure the Disk at Once radio button option is selected and finally click on the "Write tracks" button to start data CD burning.

The underlying tools that are called by xcdroast are sometimes easier to use standalone, follow to the next section for the glory of all the details:

9.25.2 cdrecord

cdrecord
http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html
, is used to record data or audio Compact Discs on an Orange Book CD-Recorder.

The easiest not so intuitive way for making or burning Cd's is to type the following commands:

```
mkisofs -J -v -o image.iso datadirectory/
```

and then,

```
cdrecord -v dev=0,0,0 speed=16 -data image.iso
```

Warning: Make sure dev=0,0,0 is your correct cd-burning device by issuing the command:

```
cdrecord -scanbus
```

In this case you are making a data CD from an image called "image.iso" which in turn is like a blueprint of the specified files you want to copy into a CD. *cdrecord* transfers that image to a standard CD tracks provided you have media in the CD drive of your computer. Some of the options here like (dev)ice and speed are dependent on the machine and drive you are using.

To make audio Cd's you don't need the *mkisofs* command, instead you probably would like to use the *sox* command which is explained in the following section. Once all your audio tracks are ready you can type something like:

```
cdrecord -v speed=1 dev=2,0 -audio track*.cdaudio
```

You can read more about *cdrecord* on its man page but if this doesn't seem so straightforward you can also try X-CD-roast as explained in the previous section.

If you already have an iso9660 image or cd tracks ready to burn it is quite easy to use, just type:

```
cdrecord -v speed=8 dev=0,0,0 iso_image_name
```

to burn an iso9660 disk image.

Cdrecord can give you a great deal of flexibility by using the command line instead of a graphical interface like XCDroast and it also provides useful information. Again, make sure you read the *man* page for a better understanding of the following commands.

Warning: Once again, make sure [`dev=0,0,0`] is your correct CD-burning device by issuing the command:

```
cdrecord -scanbus
```

Otherwise change [`dev=0,0,0`] accordingly and make sure is working by doing,

```
cdrecord -checkdrive dev=0,0,0
```

Note: *On newer Linux distributions, Fedora Core, etc., the cd-recorder device might be found as:*

```
/dev/cdrom
```

Following are useful examples for using the Cdrecord command line. (Please notice the *-dummy* mode).

The *-dummy* option will not write anything but it is very helpful for testing and see if the write speed is good enough and if the read/write buffer of the burning device are always filled with CD data. The *-dummy* mode is safe mode. Once you know everything is working just do not use when issuing the `cdrecord` command and you will have a burned CD ready for playback.

1. Rip an audio CD

   You might want to change your current directory to "/zap" by `cd /zap` . Then make sure your audio CD is "in" the CD drive. You can start ripping by,

   ```
   cdparanoia -X -B -w --force-read-speed 1
   ```

   This will rip the audio CD and deposit .wav files in the current directory (/zap normally). In audio CDs if the first track does not start at the beginning (sector 0) of the disc, the unused space before the first song will be written as a very short *track00.cdda.wav*. It is a good idea to delete that file.

2. Create an audio CD:

   ```
   cdrecord -v dev=0,0 -dummy -dao -useinfo -pad -audio *.wav
   ```

   This will write all the `.wav` files to the empty CD, creating a standard audio CD. Do not use CD-RW media because it can not be played by many audio CD players.

3. Converting MP3s to raw audio to create audio CDs

   ```
   mpg123 -s file.mp3 | sox -c2 -s -w -t raw \-r 44100 - -t wav - > newsoundfile.wav
   ```

   mpg123 converts file.mp3 to the .wav file newsoundfile.wav and makes it ready to be part of audio CDs.

4. Converting raw audio WAV files to MP3

   ```
   lame --preset extreme infile.wav outfile.mp3
   ```

   Please see the sound utilities section for more information about *Lame*. In this case *Lame* will encode the raw WAV file infile.wav to the MP3 file outfile.mp3 using high quality settings.

9.25.3 cdparanoia

cdparanoia
http://www.xiph.org/paranoia/ , retrieves audio tracks from CDDA capable CDROM drives.

Few examples:

1. Extract an entire disc, putting each track in a seperate file:

   ```
   cdparanoia -vsQ
   ```

2. Extract from the beginning of the disc up to track 3:

   ```
   cdparanoia -- "-3"
   ```

3. Extract from track 1, time 0:30.00 to 1:25.00:

```
cdparanoia "1[:30.00]-1[1:25]"
```

4. Ripping from the beginning of track 2 to the end of track 4:

```
cdparanoia "2-4"
```

5. Rip only track 5, from beggining the end.

```
 cdparanoia "5:[0.00]"
```

9.25.4 Printing-labels: cdlabelgen

cdlabelgen
http://www.red-bean.com/ bwf/soft-ware/cdlabelgen/
, was designed to simplify the process of generating labels for CD's. It originated as a program to allow auto generation of front- cards and traycards for CD's burned via an automated mechanism (specif- ically for archiving data), but has now become popular for labelling CD compilations of mp3's, and copies of CDs. Note that cdlabelgen does not actually print anything–it just spits out postscript, which you can then do with as you please.

You can try to generate your label with:

```
cdlabelgen -c "title of cd" \
           -s "subtitle" \
           -i "Piece 1%Piece 2%a third song %and more" \
           -e recycle.eps > bar.ps
```

Here is a more elaborate example which you can try in a terminal window (all the following options should go as a single line command or you might opt to put them in a script). Notice that the "%" character means a new line inside the "-i" option.

```
cdlabelgen -c "John Chowning" \
    -s "http://www-ccrma.stanford.edu/~jc"\
    -i " %  % 1- Turenas (1972) % %%\
     2- Phone (1981) % %%\
     3- Sabelithe (1971)% %%\
     4- Stria (1977) % %%\
    -e cd.eps > chowning-cd.ps
```

Make sure you have right path for the file "cd.eps". Once you have the file "chowning-cd.ps" you can try to print it in one of the CCRMA printers and do the folding and unfolding for standard "Norelco" CD trays but first test your output with:

```
gv chowning-cd.ps
```

If you are satisfied with the aesthetics you can print on a standard letter size paper by issuing:

```
lpr chowning-cd.ps
```

9.25.5 Stamping a CD with LaTeX

Here is some LaTeXcode which might be copied to a file for generating the actual CD-Label but you will need clear or white self adhesive CD-labels which are available at *Arcal* or Fry's. For LaTeXinstructions go to the typesetting section §9.9.

```
\documentclass[12pt]{article}

% funky manipulation of text.
%only useful for producing ps not pdf

\usepackage{pstricks}

% extra pstricks pieces
\usepackage{pst-text}
\usepackage{pst-eps}

% allows \includegraphics
% most likely, we'll be using dvips for output
\usepackage[dvips]{graphicx}

% allows \so  (stretched-out)
\usepackage{soul}

% set default pstricks unit to 1cm
\psset{unit=1cm}

% don't do page numbers, etc
\pagestyle{empty}


% environment for actual content
\begin{document}

% The rest is compressed, because you can't have any
% blank lines. they will start a new paragraph
% which isn't something we want in the middle of
% a picture..

% A pstricks trick, to help produce the right
% bounding box for eps
\TeXtoEPS

% pstricks pspicture environment. width/height/offset etc.
\begin{pspicture}(-1,-11)(11,1)

% default font attributes
\Huge\ttfamily\bfseries

% draw a little cross at the center (5,-5) (two straight lines)
\qline(4.6,-5)(5.4,-5)\qline(5,-4.6)(5,-5.4) % mark center
```

```
    % the text "\so{CCRMA Open House}" drawn around in an arc.
    \pstextpath[c](0,0){
      \psarcn[linestyle=none](5,-5){3.5}{270}{0}}
    {\so{CCRMA Open House}}

    % put an .eps at (7.7, -7.7)
    \rput(7.7,-7.7){\includegraphics[width=3cm]{ccrma.eps}}

    % put some text (in a table, to get layout correct)
    \rput(3.5,-8.5){\mdseries\sffamily\small%
     \begin{tabular}{r}
     Friday May 9, 2003 \\
     The Knoll, Stanford University \\
     U.S.A.
     \end{tabular}}

    % uncomment for cd border:
    %\pscircle(5,-5){2}
    %\pscircle(5,-5){6}

    \end{pspicture}
    \endTeXtoEPS

  \end{document}
```

Make sure you have the right path the logo file "ccrma.eps".

9.26 Coming or untested applications

- Gmix
  http://www.gnome.org/
  A front end Gnome audio mixer application for manipulating sound levels of all audio devices in a Linux workstation.
- GTV MPEG Player
  http://www.gnome.org/
  A MPEG file player application.
- G Sound Recorder
  http://www.gnome.org/
  A simple sound recorder and sound player for Gnome.
- ardour
  http://ardour.sourceforge.net/
  This is a "professional" multitrack, multichannel audio recorder and DAW (digital audio workstation) application. It supports up to 32 bit samples, 24+ channels at up to 96kHz, full MMC control, non-destructive, non-linear editor and LADSA plugins. Ardour is intended to be a full, professional replacement for recording equipment such as multiple Alesis ADAT or Tascam MDM tape systems, and the new HDR platforms.
- musickit
  http://musickit.sourceforge.net/
  The MusicKit is an object-oriented software system for building music, sound, signal processing, and MIDI applications. It has been used in such diverse commercial applications as music

sequencers, computer games, and document processors. Professors and students in academia have used the MusicKit in a host of areas, including music performance, scientific experiments, computer-aided instruction, and physical modeling. The MusicKit was the first to unify the MIDI and Music V paradigms, thus combining interaction with generality.

The NeXT MusicKit was first demonstrated at the 1988 NeXT product introduction and was bundled in NeXT software releases 1.0 and 2.0. Beginning with NeXT's 3.0 release, the MusicKit was no longer part of the standard NeXT software release. Instead, it was being distributed and supported by the Center for Computer Research in Music and Acoustics (CCRMA) of Stanford University. Since 1999 and version 5.0, the most recent releases run on several more popular operating systems.

– jazz
  http:// www.jazzware.com/cgi-bin/Zope.cgi/jazzware/

  Jazz is a full size midi sequencer with audio support allowing record/play and many edit functions as quantize, copy, transpose ..., multiple undo; two main windows operating on whole tracks and single events; graphic pitch and controller editing, GS and XG sound editing and more ...

10 Embedded Systems:

An embedded system is a controller with a dedicated function within a mechanical or electrical system. Modern embedded systems are often based on micro controllers (i.e. microprocessors with integrated memory and peripheral interfaces), but ordinary microprocessors are also common, especially in more complex systems. In either case, application types may range from general purpose to those specialized applications, and even custom designed for a purpose at hand. Later on micro-computers are also used to embed applications for specific objectives.

Furthermore, a microcontroller is essentially a small computer on a chip. Like any computer, it has memory, and can be programmed to do calculations, receive input, and generate output. Unlike a PC, it incorporates memory, a CPU, peripherals and I/O interfaces into a single chip. Microcontrollers are ubiquitous, found in almost every electronic device from ovens to iPods. More on the subject HERE.

Examples of embedded systems at CCRMA among others, include:

- Basic Stamp
- Pascal Stang's AVR Minis
- Arduino
- Wiring
- Pic Based systems
- Buddha Board
- Raspberry Pi

10.1 AvrLib

Avrlib is a library of easy-to-use C functions for a variety of tasks on the Atmel AVR microcontroller. It gives high-level functions for accomplishing tasks that might otherwise be tedious to program. These include serial communication, analog-to-digital conversion, displaying text on LCD displays and interfacing to a whole range of other devices such as hard drives, GPS units, Ethernet devices and sensors. The avrlib is distributed under a GNU Public License. For more information read the AVR @ CCRMA Programming guide HERE.

10.2 AVRLib on Wiring and Arduino

Wiring is an open project initiated by Hernando Barragan currently on development at la Universidad de Los Andes in Colombia. It consists of an IDE, a programming environment, and an electronics board with an AVR micro-controller for projects surrounding electronic arts, tangible media, teaching and learning computer programming or prototyping with electronics. It illustrates the concept of programming with electronics and the physical realm of hardware control which are necessary to explore physical interaction design and tangible media aspects.

Arduino, based on Wiring, is an open-source electronics platform also based on Atmel micro-controllers, providing easy-to-use hardware and software implementations. Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming.

Avrlib can also be used on Wiring and Arduino since both are based on Atmel AT-Mega chipsets. A tutorial with information and instructions can be found at AVRLIB on Wiring and Arduino.

There are tons of information on the subject matter of "Physical Interaction Design" (PID) at CCRMA on the PID Wiki. Recall PID is about the relationship between users, the devices they use, the tasks they accomplish, and people who receive this output. PID is used for new music controllers (NMC).

11 Hardware:

- Workstations Most computers at CCRMA are running Linux and therefore are Linux workstations. All the peripherals around your system are part of the workstation configuration. These might include screen monitor, computer keyboard, CPU, hard drive, compact disc units, zip drives as well as sound cards, input output devices such as MIDI interfaces or audio boxes. A Multimedia (audio and video as well as storage media) setup with speakers, microphones and even video camera is also part of the workstation configuration. Take time to familiarize yourself with the operation and handling of all these devices since you will be using them on a daily basis. Try backing up your data and sound-files to zip discs, cd-r or cd-rw's. A recordable dvd can also store your home directory and even your composition sounfiles.
- Audio I/O Most CCRMA workstations are configured to play and record audio. The applications in the sound and composition sections make extensive use of audio inputs and outputs. Audio connectors for headphones or recorders can be plugged in through the outputs of the OMNI I/O interfaces which are part of the system, in this way you are "monitoring" your audio through headphones or speakers. The OMNI I/O can also be used to record or process your audio signal through its balanced or unbalanced input. Make sure you adjust your recording levels in order to avoid clipping or saturation.
- ALSA is The Advanced Linux Sound Architecture project, please see the MIDI and Sound section for more information.
- MIDI In addition to audio input and output some sound-cards like Creative Labs Sound-Blaster or compatible provide a joystick- like MPU-401 connector that is used for MIDI IN and OUT when hooked up with a Y shaped adapter cable. Some CCRMA workstations are equipped with this feature allowing MIDI in and OUT. Programs like playmidi or aplay only support MIDI out from the computer, while PD and Jazz allow MIDI in. Make sure the cables are connected properly since this is the most popular source of MIDI malfunctioning.

  The input port on the interface card, the adapter cable and the instrument act as receiving bins. Before your MIDI instrument can process the information sent to it by the computer, it has to have this information sent in through its IN port. When you want to communicate your performance to your computer from your instrument, the computer must receive the information into its IN port (the MIDI IN port on your interface). The output ports work in the same way. Your instrument's MIDI OUT port sends OUT information to the interface card (your interface receives the data through it's MIDI IN port). The interface's MIDI OUT port sends out information to your instrument (your instrument receives the data through the MIDI IN port).

- OMNI I/O Audio Interfaces

  The Omni i/O by M Audio is a record/playback "front end" for the M-Audio Delta 44, Delta 66 audio interface cards, combining together to form a complete system capable of adapting to many studio scenarios.

  More than simply adding mic preamps to an audio card, the Omni I/O emulates a "split console" mixer design, making any routing scenario possible. The unit contains everything needed to record mics, guitars, keyboards and all other instruments, monitor playback, add effects, and mixdown.

  Two mic/instr pre-amps can be switched to line input for a total of four line inputs. Use the direct outs, or route to the Omni i/O?s extremely quiet mixer to add effects, blend in keyboards using the unity gain aux ins, and even reroute them to the inputs for recording. Monitor and mix with separate control room level, plus two headphone outs with individual level controls. A detailed user guide takes you step by step, making it easy to get the most out of the Omni I/O and your recording system.

12 The OS-X Linux Connection

Mac OS X is evolved enough so that its features and what you can accomplish is far above better and beyond NEXTSTEP / OPENSTEP. Some regard that it might be an understatement to say that OS X is derived from NEXTSTEP / OPENSTEP. In many respects it's functionality resembles the same and one might think of it as OpenStep 5 or 6. However, the similarities should not mislead you. Mac OS X nowadays is based on Darwin and chunks of existing open source software from a large number of sources like BSD, GNU, Mach, ... and even Linux.

Although Darwin is an operating system in itself, it can be best understood as a collection of technologies that have been integrated by Apple to form a major, central part of Mac OS X. Critical application environments of Mac OS X, such as Cocoa and Carbon, are not part of Darwin. Furthermore Aqua which is the standard graphical interface of Mac OS X including the Windowing System, and several other components are neither part of Darwin.

The OS-X Linux connection exists for several reasons:

You only have or like Apple hardware.

Interaction with commercial software.

Input / Output devices only compatible with Apple hardware.

Audio or Video interfaces that don't have Linux drivers.

Customized software that only works in one platform.

Perhaps you are not used to compile Linux kernels.

File transfer and exchanging.

The missing link between OS X and Linux in many respects is X11 or the X window system referenced above in its own section. Many of the commands only available in Unix systems before are becoming more and more available in OS X in particular because of the Fink project which is also an Open Source initiative. Therefore it should not be hard to have that same Linux functionality with Apple hardware.

If you want to have that Apple's ***geek*** machine you need to get some packages that Apple offers for free download until now in addition to Fink. The Apple OS X developer tools, X11 for Mac OS X and of course Xemacs for Apple's X11. You get Xemacs after you have installed Fink.

As suggested by Rick Taube, following are the steps for having a "ready made" OS-X Linux like Apple hardware machine.

- If you don't have Fink installed already, you need to do that first. You can download Fink from:

  fink
  http://fink.sourceforge.net/download/index.php

- Download and install Apple's X11 server from:

  X11
  http://www.apple.com/macosx/x11/download/

  Note: *If you are using Panther (OSX-10.3x) or above X11 comes as part of its distribution CDs which you can install at your own discretion.*

- Next you might want to intall Apple's Developer Tools as well:

  Note: *Apple's new Xcode (developer) tools are also included in every copy of Mac OS X v10.3 or above and should be installed from its CD (2).*

  If still you need to download Apple's developer tools, they might be found at:

  Developer Tools
  http://developer.apple.com/tools/index.html

- Now, to get the CCRMA Lisp world running on OS X you should download OpenMCL from:

  openmcl
  http://openmcl.clozure.com/

- SBCL now seems to be a good option for OS-X and can be found at:

  sbcl
  http://www.sbcl.org/

  Instructions for compiling the CCRMA Lisp world on OpenMCL or other OS-X Lisps like SBCL can be found in the documentation of each package (CM, CLM, CMN).

12.1 Fink

Fink is a project that wants to bring more Unix software to Mac OS X, which results in two main goals:

1. Porting software to Mac OS X. Meaning that the Fink project takes commodity Open Source Unix software and fixes whatever is necessary so that it will compile and run on Mac OS X.
2. To make the results available to casual users. For this, the Fink people builds a distribution using package management tools like dpkg and apt-get ported from Linux and the Debian GNU/Linux project. The binary distribution uses the .deb package format.

These are the options you can use when using "fink" on OS X:

`fink list`	list all packages in database
`fink install foo`	install package "foo"
`fink update-all`	update all installed packages
`fink info foo`	more info about package foo
`fink list "xemacs*"`	list all packages that start with 'xemacs'
`fink help`	list more fink options and help

12.2 SND-on-OSX

For more information and tutorials on SND please look at the Snd section: §9.13.

Cristopher Pierson Ewing has provided the steps for installing SND on OS-X (thanks Chris !)

1. The first step (and it is a vital one) is to make sure that your version of fink is fully up-to-date. Start by running:
   - fink selfupdate
   - fink update-all
2. Verify that you have a working version of X11 on your machine. This must include the development libs found in the X11SDK. If you are using apple's version of X11, the SDK package is _not_ installed by default when you install X11. You have to do a custom install from the X Code disk to get the package.

   Once all this is in place, use fink to install openmotif3
3. fink install openmotif3

   Next, install guile16 and its development libraries:
4. fink install guile16
5. fink install guile16-dev

   You can also use fink to install the fftw package. It really takes a longtime to build, but helps to speed up the fft-based functions in snd.
6. fink install fftw

   Beware that installation of fftw takes a while because it needs to compile itself and several dependencies.
7. Download and untar the latest SND sources from ccrma-ftp

   Cris Ewing suggests:

   > "If you have trouble with the configure process, you may want to check to make sure what the guile binaries are named. All the information you will need to get configure to work successfully is in README.Snd inside the snd distribution. It seems important to use the switch –disable-nls. I'm not sure why, but I couldn't get the build to work with nls enabled. This is the configuration command that I used after following the instructions above:"
8. On the SND directory you can try this options with the configure command on the **tsch** before making its binaries (all in one line or using backslash):

   ```
    ./configure CFLAGS=-I/sw/include LDFLAGS=-L/sw/lib \
    GUILE_CONFIG_path=/sw/bin/ --disable-nls \
   --with-guile --with-motif --with-static-xm  \
   --with-motif-prefix=/sw
   ```

   This config was run from a bash shell. If you are using **tcsh** (and you might be if you upgraded to panther) then you will have to omit the part about CFLAGS and LDFLAGS above. Instead, you can set these through the shell by running:
9. setenv LDFLAGS "-L/sw/lib -lmx -bind_at_load""

10. setenv CFLAGS ”-I/sw/include”

    After this, you’re nearly home:

11. make

12. sudo make install

    (you need to sudo make install, because it will install a Snd executable in /usr/local/bin)

12.3 SND-binary-on-OSX

If you don’t want to start from scratch or if you want to install an already statically compiled binary version of SND, follow the next steps: *(more information about these steps can be found on the previous section)*

1. Install the developer package for OS-X 10.2, 10.3, 10.4, or above on the OS-X CDs.

2. Install the ’X-window’ application on the OS-X CDs or download it if needed.

3. Download Fink for 10.2, 10.3, 10.4 or above.

4. fink selfupdate

5. fink update-all

   Once all this is in place, use fink to install guile-1.6 and its development libraries:

6. fink install guile16

7. fink install guile16-dev

8. Go to your home “bin” directory or if you are a super user, go to “/usr/local/bin” by:

   ```
   cd $HOME/bin
   ```

   or alternatively by

   ```
   cd /usr/local/bin
   ```

9. In this directory now download the SND binary from ccrma-ftp.stanford.edu

   SND binaries are in pub/Lisp/ and should be one of osx-10.2-snd, osx-10.3-snd, osx-10.4-snd, or above, depending on the OS-X version you are running.

10. Make sure you are on a terminal window and running the **tcsh** shell. The xterm application on X11 would be a good idea at this point.

11. Provided that guile was installed on **/sw/** which is the Fink standard, type the following command on the terminal:

    ```
    setenv GUILE_LOAD_PATH /sw/share/guile/1.6
    ```

12. Now on the terminal and on the “bin” directory where SND is type the following command in order to make SND an executable application,

    (make sure about the SND version you are running)

```
chmod 777 osx-10.3-snd
```

13. You need to run SND under X11 using the "xterm" application. Make sure you set your environment "PATH" variables so that the bin diretories are on the the binary path. To run SND just type:

    (Again, make make sure about your SND version)

    ```
    ./osx-10.3-snd
    ```

    or if they are on the binary path,

    ```
    osx-10.3-snd
    ```

13 USB-MIDI:

The Advanced Linux Sound Architecture (ALSA) is a major component in most Linux distributions. At PlanetCCRMA is an essential part since most audio and MIDI applications used at CCRMA depend on it. Extensive research and testing is done on a daily basis to be up-to date with the latest drivers and kernel related ALSA API. Recently many improvements have been done to many old and newer hardware sound-cards but perhaps attention should be paid toward USB related audio and MIDI devices because newer hardware seems to be geared to this direction in these days.

Testing was performed with various commercial USB-MIDI and USB-AUDIO interfaces to the extend that we are able to consider USB as good as an alternative (or better in some cases) to serial and parallel interfaces and even to audio sound-cards. The "usb-audio" module of ALSA has been updated and improved to the extend that most hardware tested, behaved in a reliable fashion with standard Linux and PlanetCCRMA applications. In our tests hardware included not only interfaces but controllers such as keyboards and wheels.

MIDI is a very important part at CCRMA because of its close relationship with human computer interface (HCI). Our testing focused not only on ALSA sequencers but also with the PD Linux environment and more specialized software such as Max Matthews' Scanned synthesis program and Craig Sapp's Improv. In most cases USB-Audio performed accurately even with the combination of Scanned synthesis audio and Radio Baton MIDI input using M-Audio's Quattro interface. With the Quattro we were also able to listen to four discrete 44.1KHz audio channels and record audio at 96KHz sampling rate.

For more updated information on the ALSA "usb-audio" module and for additional USB audio an related issues, point to the USB audio web pages on the Nano HOWTOs section at PlanetCCRMA @ Home:
http://ccrma.stanford.edu/planetccrma/software/

14 CCRMA-lize your computer

If you want to mimic your $HOME/ directory at your dorm, or somewhere in some far away island, dessert or galaxy using CCRMA's Linux environment (A.K.A. PlanetCCRMA). Good news, it can be done. Furthermore, it might be that you want to "CCRMALIZE" your laptop or workstation with the Linux operating system. It can also be done! §14.2.

14.1 Mimic your CCRMA Home Directory

If you already have Linux installed on your computer you can follow the next steps using unison or rsync. If you don't have Linux installed on your machine you might consider installing Linux §14.3 and PlanetCCRMA.

- With the 'unison' command: Suppose you want to sychronize the directory 'doc' in your computer with the 'doc' in your CCRMA $HOME/ directory. You simply type a 'unison' command like,

```
unison doc ssh://ccrma-gate/doc
```

Make sure you are in the $HOME/ directory in both machines.

- With the 'rsync' command: Suppose you want to sychronize the directory 'mail' in your computer with the 'mail' in your CCRMA $HOME/ directory. You type a 'rsync' command like,

  ```
  rsync -a -vv -e ssh mail/ ccrma-gate:mail/
  ```

- To sychronize a single file, use the 'rsync' command like:

  ```
  rsync -vv -e ssh myfile.mat ccrma-gate:matlab/myfile.dat
  ```

- You can find more information at the MAN pages for unison and more rsync.

14.2 PlanetCCRMA-DVD

In case of an out-of-the box alternative, Fernando Lopez-Lezcano has made available a PlanetCCRMAFedora-Core(N) Linux distribution DVD with many of the standard audio and Video applications and a lot more. In addition to the standard fedora disks, the planetCCRMA DVD contains a low-latency Linux kernel optimized for audio applications in real-time, RPM packages with many Computer Music applications as well as popular open source music packages all tied together and as a Linux workstation ready to go. For most situations no compilation and tedious configurations are needed for this Linux to be used by a computer music composer or musician. The sources for the planetCCRMA DVD are located at CCRMA-FTP.

Instead of using the standard Red Hat Fedora Core installation CDs or DVDs, you can use the PlanetCCRMA DVD in the very same fashion. Once you have your Linux partition boot your computer with the PlanetCCRMA DVD. Make sure you read the "README" and related documentation files after you have downloaded the sources for the DVD.

14.3 Custom-Linux-Installation

Note: This is almost a verbatim of a RHFC Linux installation. For detailed information please consult the main Fedora site at RedHat.

1. Make sure you have a Linux Partition on your Laptop or Workstation. (Before creating a partition on a windoze (vfat) drive make sure you backup sensitive data). People use commercial software like Partition Magic to get a Linux partition. You can also use the program. If you are not so familiar with the command window it might be worth to ask for some help.

   It is recommended to have at least 3 partitions:

   - A 'home' partition where your home directories and files are installed. The larger the better but keep in mind that this is the partition you most often backup.
   - A root or '/' partition where system files are installed. A complete Fedora Core installation require almost 10GB plus the extras you might want to install.
   - A 'swap' partition which should be at least as large as your physical memory. If you have 1GB of ram memory your 'swap' partition should be at least 1GB.

2. Download or get Linux DVDs or CDs from the main Fedora site at RedHat or at one of its mirrors.

3. Boot your computer with Fedora Core CD-1 or the DVD and follow its graphical instructions (default values are appropriate for most common installations. If you have a laptop make sure to use LCD screen when asked about your monitor type).

   If you can't get to a graphical mode installation, the graphics might not be supported on X.ORG windows yet. Check the chipset while boot on windoze and got to X.ORG to see how can it be configured.

   By experience it might be a good idea that you do a full (Install Everything) installation.

4. Get fedora updates (beware of Kernel version numbers and good bandwidth). You can use the yum command:

   ```
   yum -y update
   ```

   At this time it is a good idea to get familiar with the yum command or read about it at the yum site.

5. Complete the Fedora Core installation with Fedora Extras (see section §14.4).

6. Last but not least! Immerse yourself into PlanetCCRMA @ Home

14.4 After-a-Fedora-Linux Installation

For functionality and compatibility with other operating systems, after you have done a full Fedora Core installation some additional packages might be also considered. Most of these packages are available at every workstation in CCRMA but they don't come standard in Fedora Core distributions.

Most of these packages are found at Fedora Extras site.

and may be installed just by issuing the yum command like,

```
yum install 'name-of-package'
```

There should be an '/etc/yum.repos.d/fedora-extras.repo' file with Urls pointing to fedora-extras sites. If this is not the case YUM might have to be installed or re-installed as these files are part of the yum install package.

It might be useful to install the following packages (please consult PlanetCCRMA @ Home for more):

- Xemacs,§14.4
- XMMS,§14.4
- Mozilla, Firefox plug-ins,§14.4
  - Macromedia Flash
  - Java runtime Plugin j2re
  - Real Player and Acroread (see below)
- RealPlayer,§14.4
- Acroread,§14.4
- Prosper,§14.4
- Ncftp,§14.4
- Octave,§14.4
- Octave-forge,§14.4

- Xemacs:

  XEmacs (more at §9.7) is a version of Emacs, compatible with and containing many improvements over GNU Emacs. XEmacs is a highly customizable open source text editor and application development system. To install XEmacs use yum as follows:

  ```
  yum install xemacs
  ```

- XMMS:
  XMMS (read more at §9.5), is the X Multimedia System. It is used to play audio and other kinds of media files. By default XMMS can play MPEG audio, Ogg Vorbis, RIFF wav, most module formats, and a few other for- mats. XMMS can be extended through plug-ins to play a number of other audio and video formats.

  Fedora Core does NOT ship with XMMS, you must install from the Fedora Extras site or issuing the following yum command:

  ```
  yum install xmms xmms-devel
  ```

  There are not MP3 plug-ins at he Fedora Extras site. The RPMs at freshrpms work and include MP3 support. You must download each RPM and install it manually by using the RPM command accordingly:

  Look for xmms-mp3 at freshrpms and also download xmms-skins from also at freshrpms.

  Then install as root with these commands:

  ```
  rpm -Uvh xmms-mp3 xmms-skins
  ```

- Mozilla, Firefox plug-ins:

  - *Macromedia Flash:*
    The Flash RPM plug-in can be downloaded from Macromedia. Read more about Linux web browsers at §9.2.
    Install the RPM by

    ```
    rpm -Uvh flash-plug in
    ```

    Make sure you are logged into the X server (init 5, runlevel 5) and do not have Mozilla or FireFox open when you install the RPM. Make sure to (read) accept the agreement. Close the terminal and reboot your machine.
  - *Java runtime Plug-in JRE:*
    Download J2SE from SunSite

    There are RPM and self extracting files. If you download the RPM, just install it by typing in the terminal window something like (*you might have different version numbers*):

    ```
    sh jre-1_5_0_07-linux-i586-rpm.bin
    ```

    or

    ```
    rpm -Uvh jre-1_5_0_07-linux-i586-rpm
    ```

Then link the library to the Mozilla plug-ins directories with a command like *(it should be a one-line-command)*:

```
ln -s /usr/java/jre1.5.0_03/plugin/i386/ns7/libjavaplugin_oji.so \
/usr/lib/mozilla/plugins/libjavaplugin_oji.so
```

If you download the binary installer as root follow these steps:

∗ `mkdir /opt`

∗ `mv *.bin /opt`

∗ `cd /opt`

∗ `chmod +x *-linux-i586.bin`

∗ `sh *-linux-i586.bin`

∗ Link the library to the Mozilla plug-ins with a command like *(it might not be the same version numbers. Make sure type all in a one line command)*:

```
ln -s /opt/jre1.5/plugin/i386/ns7/libjavaplugin_oji.so \
/usr/lib/mozilla/plugins/libjavaplugin_oji.so
```

– *Real Player and Acroread (see below): §14.4*

- RealPlayer:
  - RealPlayer, plays streaming audio and video over the Internet in real-time. It plays RealAudio, RealVideo, MP3, 3GPP Video, Flash, SMIL 2.0, JPEG, GIF, PNG, RealPix, RealText, Ogg Vorbis, and Ogg Theora (more at section §9.5).
  - Download RealPlayer GOLD from RealSite.
  - Install RealPlayer GOLD as root with something like:

    ```
    rpm -Uvh RealPlayerXGOLD.rpm
    ```
  - Try the KQED or the NPR site to test RealPlayer.
- Acroread:
  - Adobe Acroread, lets you view and print documents in Portable Document Format (PDF). PDF files retain all the formatting, fonts, and graphics of the original document, and virtually any Postscript document can be converted into PDF. LaTeXcan also generate PDF files (see section §9.22 for using acrobat reader and postscript formats).
  - Download the Acroread RPMs from dag site
  - Install Acroread RPMs by:

    ```
    rpm -Uvh acroread-7.0 mozilla-acroread-7.0
    ```
- Prosper:
  - Prosper is a LaTeXclass for writing transparencies and presentations by using the PDF file format. It is a much better and cleaner alternative to other presentation programs by offering an environment for writing high-quality slides for both printing an displaying with a video-projector. Prosper is in the Fedora Extras repository.
  - To install Prosper just issue the 'yum' command:

    ```
    yum install tetex-prosper
    ```

- Ncftp:
  - "The purpose of ncftp is to provide a powerful and flexible interface to the Internet standard File Transfer Protocol. It is intended to replace the stock ftp program that comes with the system".
  - "Some of the cooler features include progress meters, filename com- pletion, command-line editing, background processing, auto-resume downloads, bookmarking, cached directory listings, host redialing, working with firewalls and proxies, downloading entire directory trees, etc".
  - To install Ncftp simply use the yum command:

    ```
    yum install ncftp
    ```

- Octave:
  - "GNU Octave, is a high-level language, primarily intended for numerical computations. It provides a convenient command line interface for solving linear and nonlinear problems numerically. Octave can be useful for performing numerical experiments using a language that is mostly compatible with Matlab".
  - Octave can also be installed with yum.

    ```
    yum install octave
    ```

- Octave-forge:
  - Octave Forge has the signal processing toolbox items for Octave. It serves as an add-on utility collection that includes many functions compatible with the Matlab Signal Processing Tool Box and others.
  - Both octave and octave-forge are included in Fedora Extras and therefore can be installed with:

    ```
    yum install octave-forge
    ```

15 Wireless-Tools and Networking with Linux at CCRMA

2200urlhttp://atrpms.net/name/ipw2200/

For using the wireless (guestnet) network at CCRMA please follow the instructions at CCRMA guestnet.

If you have a laptop with a wireless card the probability that you might use Linux and all the PlanetCCRMA features is very high. Many standard cards already have drivers which work on many Linux kernels. Perhaps the only issue is to let the Linux Kernel guess what kind of wireless card your laptop is using in case it has not recognized it.

The best way to find out about a wireless card is by issuing the command:

```
lspci
```

and look for a wireless chip-set. There are several sites which document wireless network cards for Linux, utilities as well as Howto's. If you find you need more information, kindly look at the Wireless LAN resources for Linux at wireless tools.

A very popular wireless card in many laptops today is the 'Intel Pro Wireless' (IPW) which is part of most Centrino system's configurations and packages. Therefore as an example of getting your wireless card to work in a Linux laptop setup at CCRMA, the driver for an IPW card namely ipw2200 will configured.

First of all you might need to find out if the package wireless-tools is already installed in your computer. Type the following command:

```
rpm -q wireless-tools
```

If you don't have wireless tools installed you can use yum to install them although they are part of a full Fedora Core installation:

```
yum install wireless-tools
```

Furthermore, the ipw2200 is already compiled in many kernels but in case you need a fresher version or for some other reason ipw2200 RPMs can be found at ATrpms:

- IPW2200 driver
- IPW2200 firmware

Download the ipw2200- rpm and the ipw2200-firmware- rpm.
Install both packages by issuing a rpm command like.

```
rpm -Uvh ipw2200-* ipw2200-firmware-*
```

Firmware for the IPW2200 is constantly updated and usually is the cause for not loading the driver modules into the kernel. Try to have the most up-to-date version.
Once you have successfully installed the above packages you can proceed configuring you wireless setup.

- Do a

  ```
  dmesg | grep ipw2200
  ```

  to see how the kernel is detecting your IPW card.

- Do

  ```
  iwconfig
  ```

  to see how your IPW card is bound to the hardware.

- If eth1 is found to be bound as a wireless network interface, to see what networks are available type:

  ```
  iwlist eth1 scan
  ```

  – this will scan for all available networks.

- Since the CCRMA guestnet wlan network is protected find with someone at staff or with the network administrator what is the "SSID" (access point name) and also the "WEP" key. Beware that your card must be registered in guestnet and that you must also have a CCRMA login.

- Now configure your wireless card "eth1" or other as follows:

  –
    ```
    'iwconfig eth1 essid YOURSSID
    ```

  –
    ```
    iwconfig eth1 key xxxxxxxxxx
    ```

- For a DHCP client you can type:

  ```
  dhclient eth1
  ```

- To see the status of your wireless setup you can type,

```
iwconfig
```

- To start a network connection type:

```
ifup eth1
```

- To see the network status type:

```
ifconfig
```

- then do a

```
ping cmn10.stanford.edu
```

to see how the connection is working.

References

[1] Chowning, J. M. *The Simulation of Moving Sources,* Journal of the Audio Engineering Society, Vol. 19, No. 1. 1971.

[2] Chowning, J.M., *Computer Music: A Grand Adventure and Some thoughts About Loudness,* Proceedings of the 1993 International Computer Music Conference, pp-2 -3, Tokyo, Japan, International Computer Music Association, San Francisco, USA, 1993.

[3] Kendall, G. A 3-D *Sound Primer: Directional Hearing and Stereo Reproduction,* Computer Music Journal, Vol 19, No. 4, pp. 23-46, 1995.

[4] Loy, D. G., *Notes on the implementation of MUSBOX: A compiler for the Systems Concepts digital synthesizer,* In Roads, C., editor, The Music Machine, pages 333-349. MIT Press, Cambridge, MA., 1981.

[5] Loy, D. Gareth, *Musimathics, The Mathematical Foundations of Music,* MIT Press, 55 Hayward Street, Cambridge MA, USA, 2007.

[6] Mathews, Max, and John R. Pierce, eds. *Current Directions in Computer Music Research,* MIT Press, 55 Hayward Street, Cambridge MA, USA, 1991.

[7] McCarty, B., *Learning DEBIAN GNU/Linux,* O'Reilly & Associates, Inc., Sebastopol CA,USA, 1999.

[8] Moore, F. Richard, *A General Model for Spatial Processing of Sounds,* Computer Music Journal, Vol 7, No. 3, pp. 6-15, 1983.

[9] Moore, F. Richard, *Elements of Computer Music,* Prentice Hall, Englewood Cliffs, NJ. USA. 1990.

[10] Raymond, Eric S., *A Hacker's Dictionary,* MIT Press, Cambridge, MA., USA, 1994.

[11] Roads, C., ed. *Composers and The Computer,* Willian Kaufmann, Inc. Los Altos, CA, USA, 1985

[12] Sobell, M., *A practical Guide to Linux. Commands, Editors and Shell Programming. 2nd.,* ed. Pearson Education Inc., 501 Boylston St. Suite 900, Boston, MA 02116, USA, 2010.

[13] J.O Smith, *Viewpoints on the History of Digital Synthesis,* Proceedings of the International Computer Music Conference (ICMC-91, Montreal), pp. 1-10, Computer Music Association, October 1991.

[14] Smith, Julius O., *Physical Audio Signal Processing,* W3K Press, CCRMA, Stanford CA, USA, 2004.

[15] Stallman, Richard M. and Joshua Gay ed., *Free Software, Free Society,* GNU Press, Boston, MA. USA, 2002.

[16] Steiglitz, Ken, *A Digital Signal Processing Primer with Applications to Digital Audio and Computer Music,* Addison-Wesley Publishing Company, 2725 Sand Hill Road, Menlo Park CA, 1995.

[17] Varèse, Edgar. 1966, *The Liberation of sound.* in M. Boretz and E. Cone, eds., *Perspectives on American Composers.* Norton, New York, NY. USA, 1971.