THE UNIVERSITY COLLEGE
OF THE CARIBOO

COMPUTING 253

Small Computer Systems: Organisation and Architecture

Workshop #3 –*"Sharing a waltz with UNIX"*

No required submission

UNIX/Linux

Developed in the early 1970's at Bell Labs, UNIX was created as an "open" operating system that provided end-users the ability to extensively customise it, as needed (but not open as in "free" to distribute and use). Because of its portability and low distribution cost, UNIX became very popular with universities, researchers, and businesses. It was eventually ported to most mainframe and mini-computers.

(Note: Although first coded in Assembly Language, UNIX was developed with the C programming language in mind. As long as a C compiler is available for a given platform *(CPU and bus architecture), UNIX can be compiled for it!)*

For microcomputers, full UNIX is far too large (resource-wise). To run on these machines, different *flavours* of UNIX were introduced, such as FreeBSD, SCO-UNIX, XENIX, and Minix (Mini-UNIX). In 1991, Linus Torvalds took the Minix kernel and crafted it for the i386 platform (based on the Intel 80386 CPU architecture, which includes all processors from 80386, 80486, and Pentium CPUs (I, II, III, and IV). He named the new, non-commercial flavour "Linux" (Little UNIX), and released it *freely* to the computing public for further development.

(Note: So why did Linus develop Linux? Most PC OSes were either not broad enough or too expensive (or both) for what he needed while going to school. His university's minicomputer was shared by all students and faculty, with never enough available user-time, time for Linus to just "play." He wanted the mini's OS at home...the rest is history.)

Today, many companies release their own Linux distributions ("distros") that include special configurations, software collections, and programming languages, with drivers for most new buses and devices (USB, IEEE1394, scanners, LCDs, etc.). Some popular i386 distros are available from RedHat, Mandrake, SUSE, Debian, and Caldera.

> *(Note: FreeBSD is closer to the true UNIX core than Linux, with a more stable platform. Yet Linux has become more popular because of totally open design, fully-user directed development, and timely interfaces for new devices.)*

For all practical purposes, Linux is essentially the same as UNIX. Mostly all user-commands, application programs, and general concepts apply to Linux as they do UNIX. Most users are equally comfortable in either operating system.

> *(Note: Linux is part of the GNU Software Development (GNU-"GNU is Not Unix") and most Linux open-source software is part of the GPL (GPL-"GNU Public License"). "Open-source"—the source code is open to all.)*

So which is the best Linux distribution? With so many available, and each almost specialised for specific tasks and environments (only all-purpose, or "generic," releases get media attention), the question should be: *which Linux distribution best fits the current needs?* For the moment, the answer is a personal one.

The Linux distribution used in this tutorial is CD-bootable **KNOPPIX**. It is based on the very popular **Debian** distribution.

CLI/GUI

UNIX is a *command-line interface* (CLI), with interaction accomplished through a "shell" (of which there are many) that behaves like DOS's command line (or Window's Command Prompt). Many UNIX commands are similar to DOS (since most DOS commands actually originated from UNIX). Directories function almost identically in UNIX as they do DOS, except for advanced security and the concepts of "symbolic links" and "ownership."

Not to be outdone by MacOS or Windows, UNIX incorporates a graphical shell called a *window manager* to provide GUI features. Window managers use X-Windows: a library of graphic functions providing graphical interfaces similar to Windows. Sitting atop X-Windows, the window manager provides a similar user- and application-interface to Windows and MacOS.

> *(Note: Popular desktop/window managers for Linux are KDE and GNOME. Both managers offer many popular GUI features like drag-n-drop, object-linking-and-embedding, as well as letting users open a* terminal window *and drop to a* command-line *at any time. KDE – "K Desktop Environment;" GNOME – "GNU Network Object Model Environment.")*

Future Focus of UNIX/Linux

Most computing professionals view UNIX as a "workhorse operating system," meaning that it is dependable, stable, and strong, but not very pretty, and this description has carried over to Linux. Most servers on the Internet, and in large businesses, run a *NIX (a form of UNIX or Linux) rather than another OS; even Microsoft maintains a set of *NIX-based servers for some web-services (of course, they don't advertise this fact).

But in providing a comfortable, intuitive environment for the average, end-user, Linux requires much improvement for installation/set-up and interaction tasks (relating equally to *software* and *hardware*). With periodic updates to the kernel, GUIs (*window managers*), and methods for installing new hardware, Linux has recently found itself being prematurely promoted as a "desktop operating system" rather than just a server-class operating system.

So the question is: *will Linux replace Microsoft Windows and Apple MacOS as the common desktop operating system?*

Many *pro*-Linux groups enjoy dreaming such, but the reality is that Windows and MacOS will continue—at least for the short term. Yet Linux has changed how Apple and Microsoft design their operating systems. From gaining blatant X-Windows "look-and-feel" aspects, to more reliable operating system cores designed around *kernel-level interactions* rather than *linear-polling* structures, Apple and Microsoft have shifted towards the UNIX direction.

> *(Note: Apple MaxOS 10.x is built over a strong FreeBSD/X-Windows core, letting both Mac and UNIX?Linux applications to run. Windows 2000/XP also includes network components from FreeBSD, providing more reliable and faster TCP/IP transmission and security; 2000 Server also includes an old POSIX core to run UNIX-based utilities.)*

Topics of this tutorial

The breadth of topics related to learning UNIX/Linux is far beyond this one exercises. The goal of this tutorial is to introduce *NIX-style operating systems, and help summarise the important and essential aspects.

1. The *command shell* through a terminal window, and basic command syntax and purpose; access to built-in help.
2. Directories: creation, navigation, and destruction; copying/moving files in directories.
3. "Who's logged on?" and file/directory ownership.
4. Drive partitions and how <u>all</u> drives are directories off the "root" directory; and mounting partitions and drives
5. Editing files and language compilers: creating simple C++ and Java programs.
6. Examining KDE, the popular X-Windows window manager

7. The popular "open-source" office suite: OpenOffice
8. [Time permitting] Configuration of the system through KDE
9. [Time permitting] Accessing DOS/Windows diskettes and files with **Mtools**

Reference

As a resource, some UNIX and Linux reference books are available during the lab (ask your instructor). Further, your textbook includes a discussion in **Appendix G – Introducing Linux** (pg 1153-1165).

Finishing

When you have finished the observations and tasks section and logged out of Knoppix (the CD will eject), make sure that you have removed the CD. Return the CD, configuration diskette, and any books you have borrowed.

Topics not covered

The entire Linux operating system could not be covered in a single tutorial. With the focus placed on the important topics, in particular those that relate to DOS/Windows commands, the following areas have been left for later courses, or personal study,

- user configuration
- login parameters, environment variables, and initialisation configuration
- telnet services, remote login, and SSH
- network services and component configuration; remote X-Windows client access and SAMBA
- web-server, database server (SQL) installation, configuration, and usage
- command shell programming
- *process* activity management
- kernel upgrading and kernel component/module loading; use of RedHat's RPM format (RPM-"RPM Package Manager"; *defined by RedHat, so sometimes called, "RedHat Package Manager"*)
- multimedia and peripheral equipment configuration
- X-Windows and Window Manager configuration
- platform emulation: MS Windows and Apple MacOS
- software development for console and graphic mode *(via any programming language you want*—ya gotta love Linux!*)*

Observations and Tasks

Note: Unlike commands in the DOS/Windows CLI, **letter case matters** with Linux CLI commands.

Part 1: Getting Started with *KNOPPIX Linux*

1. Power up the system and **quickly** insert the KNOPPIX CD and configuration floppy diskette.
   At the "**boot:**" prompt, use <F2> or <F3> and examine the boot options.

   But to get started, type: **knoppix floppyconfig**
   (Note: The configuration floppy is not necessary, but used to provide some special settings for the lab's computers.)
2. The OS then continues to boot, autodetecting all the system devices, and reading the specific configurations from the floppy.
3. Along with detecting devices, a multitude of services are loaded. As this is a "desktop/workstation" installation, the services are intended for a single user. In the case of a "server" installation, a different number of services and programs would be loaded.

   Normally, a Linux installation will ask for a *username & password* before allowing access to the system. Since KNOPPIX is intended for use as a "single user," it bypasses the security prompt.

Part 2: Configuring the Network Card Parameters

1. Before continuing, the network settings can be provided.
2. Select either the cute penguin icon on the task bar, or (K)→KNOPPIX; then Network/Internet→Network card configuration and provide:

"Use DHCP broadcast?" ***no*** (unless your network is using DHCP enabled)
"IP Address for eth0" ***192.168.100.xx*** (xx = the computer's number)
"Network Mask for eth0" ***255.255.255.0*** (just press Enter)
"Broadcast Address for eth0" ***192.168.100.255*** (just press Enter)
"Default Gateway" ***192.168.100.251***
"Nameservers(s)" ***192.146.156.163***

Part 3: Terminals and User Interaction and Basic commands

1. To open a *terminal window* (also called a "console" or "shell window"), click the taskbar icon that looks like a computer monitor.
2. Command line interaction with Linux is done through a *shell*, which performs the same function as *MS-DOS Command Prompt* in Windows: to provide a "command line interface" to the user.
3. Common shells: Bourne (bsh), Korn (ksh), C-shell (csh), tcsh (from C-shell), and zsh (from Korn shell), with the most popular Linux shell being bash (Bourne again shell).

   Along with command interaction, shells provide complete programming languages (programs written in shell languages are called shell scripts).

   Although not fully featured as common programming languages (C++, COBOL, Java, Pascal), shells have almost as much capability as other scripting languages such as VBscript or Javascript.
4. There is a small set of often-used commands that must become familiar to all users. These are related to file copying/moving, directory navigation, and logging in/logging out.
5. Obtain a listing of files in your directory, type: **ls**
   To see all files (including hidden files), type: **ls –a**
   For a longer, detailed display, type: **ls –al**
   (Some Linux even provide **dir** command, similar to DOS; *try it*.)
6. To check if another specific computer is available, use the **ping** command, type: **ping 192.168.100.xx** (**xx** = any other computer in the lab)
   Also,
   192.168.100.100 (Guru), and www.cariboo.bc.ca (UCC Webserver)

   ***Note:* <CTRL-C>** *can exit almost any command line application.*
   If this doesn't work, try quitting by just pressing the **"Q"** ***key***.

Part 4: Built-in Help

1. Most new Linux users (called *newbies*) find Linux very confusing because of the number of commands that must be learned: *which command to use*, and *what does it do*. *(this is a similar complaint of DOS)*
   Such as the strange sounding commands: **mv**, **grep**, and **chown**.
2. With hundreds of available commands (written by different people, over the decades of UNIX/Linux's history), confusion can set in quickly. To help, designers always include "online manuals" called **man** pages that index [almost] every available command.

   For programmers, man pages also include descriptions of major C-functions for use in relating to the OS, such as **scanf()** *and* **printf()**.
3. To use a man page for help, use the syntax: **man** *command*
   Use this method to get help on the commands: **date**, **grep**, and **top**.
4. Another help method is to ask for condensed help directly from the command (built-in help). The syntax is: *command* **--help**
5. Most Linux distributions also include the **info** command. This is a large interactive, summary document of important Linux commands.
   Info is used *alone* or *with a command*, use the syntax: **info** *command.*

   For newbies getting familiar with one of the help methods is a <u>must</u>.
6. If you have not done so already, obtain help for the commands you typed in **Part 2** (using one of the methods: **man**, **--help**, or **info**).
   Do all the commands have at least built-in **--help** available?

Part 5: Command Piping and Redirection

1. At the command line data can transferred from one command to another, or redirect input/output from/to files. These concepts are available in most operating systems because of the necessary programming required at the command line level.

   "piping" – pipe the output of one command as input to another;
   "redirection" – redirect output of a command to a file, or file content as input to a command
2. As an example of this technique, the following use more than one command per line, moving data from one to another.
3. To locate all the files in the running Linux operating system, and load the output into a buffer that you can scroll through (or page up/down),
   type: **locate / | less**

   Press 'Q' to quit the less *command.*
4. Output can also be sent to a file, rather than to the screen. To store a listing of <u>all</u> the files in the current directory (or 'folder' for GUI users),
   type: **ls -a > files.txt**
5. To examine the contents of the file, type: **less files.txt**
6. To produce a sorted listing of the files in the current directory,
   type either of the following:
   a) **ls –a | sort | less** <u>or</u>
   b) **ls –a | sort > files.txt** , followed by: **less files.txt**
7. The screen might be a little messy at this point, and it would be nice to clear the screen. To <u>clear the CLI screen</u>, type: **clear**

<u>Applying Piping and the</u> <u>cal</u> **<u>command</u>**

8. An interesting command is **cal**, which produces a calendar view of a year, or just a month, for the range of years *1-9999*.
9. Get some <u>help</u> on the **cal** command and produce a display showing which day of the week your birthday falls on in 2003, 2004, and 2055.
10. Determine the cal command to display only your <u>birthmonth</u> this year.
    <u>Pipe</u> the output to a file called, **mybirthday.txt**
    Use **less** to examine the **mybirthday.txt** file.

Part 6: Resource Identification and Environment Variables

1. A main concern for servers is the availability of resources—the fewer resources available, the lower the overall system performance.
2. The following commands display the status of the essential resources available on the server, try them,

free –o	– memory allocation on the computer
df	– free disk space on main partitions (shown as devices)
top	– *top* running processes (measure of CPU utilisation); "q" to quit
date	– current date/time (not a resource, but useful)

3. To examine the current network configuration, use the command,
   ifconfig – network card configuration; IP address & subnet mask
   Can you see the settings you provided earlier?

4. Each login session obtains its own configuration environment.
   To display the current environment variables, in sorted order,
   type: **printenv | sort | less**

5. From this display, look up the <u>values</u> of the these <u>environment variables</u>, (*these are* <u>not</u> *commands*)

home	- path for user's home directory
hz	- length of command history (previous commands used)
language	- CLI interface language (keyboard, display characters)
logname	- login name (default in this case is "knoppix")
oldpwd	- previous current directory (see **pwd**)
path	- directory path names to search for executing commands
pwd	- path of current working directory (where you are *now*)
shell	- directory path location for active shell program files
user	- name of user logged in (usually similar to **logname**)

Part 7: Directories

1. Directories in Linux behave similarly to directories/folders in Windows, except that Linux provides "symbolic links" (virtual directories pointing elsewhere) and "owned directories" (belonging to particular users).

2. In Linux, <u>all</u> devices (drives) and every user directory "hang" off the **root directory** (/). Unlike Windows, Linux <u>does not have drive letters</u>, only directories, even different drives (or partitions) are treated as directories.

[The following steps guide you through a little directory exercise.]

3. In your current knoppix user-directory, create a new directory called "mydir," type: **mkdir mydir**.

4. Produce a listing with both **ls** and **ls –l** to see the directory you created.

   (Note: As you work through the following, produce a listing <u>after each command</u> to see how things change: **ls –l** *)*

5. Use the NEDIT GUI editor to create the following file. When finished, select File→Save then File→Quit. Type: **nedit myfile.file**

```
Linux is really neat.  Linux is different.
Linux is not Windows.
```

6. Copy the file into **mydir**, type: **cp myfile.file mydir/myfile.file**

7. Move the current file to another name (i.e., rename the file), type: **mv myfile.file other.file**

8. Change into the new directory, type**: cd mydir**

9. Move the file from the subdirectory to the parent directory, type: **mv myfile.file ..**

10. Change back to the parent directory, type: **cd ..**

11. With the subdirectory now empty, remove it, type: **rmdir mydir**

12. Remove the two files you created, type: **rm *.file**

[Optional]   Part 8: Ownership of Files and Directories

1. With the exception of a few system devices, all resources "belong," to either a user, or group; this is called the "ownership" of a resource (file, directory, etc.). With ownership are other attributes (permissions).

2. To examine permissions, produce a long listing of files in the current directory, type: **ls –al | less**

3. In the **ls** display, starting at the left side of the listing is a series of letters, with some dashes. Each file has its own permission entry, with each entry composed of specific fields. In order, each field (column) represents,

   1 – file type ('-' regular file, 'l' symbolic link, 'd' directory, 'c' device)
   2->4 – user permissions (what can the user do with the item)
   5->7 – group permissions (what can the user's group do)
   8->9 – other/world permissions (what can everyone else do)
   Permissions:
   "r"–read, "w"–write, "x"–execute, "d"–directory, "s"–root level required

4. Permissions can be changed using the **chmod** command.

   (Note: Permissions are beyond the scope of this tutorial. You are encouraged to research the concept further.)

Part 9: Partitions and Drives

[For the following, you must have **root** privilege; root is the ultimate/admin user.]

1. All commands that access system resources must be issued from a user with the correct privilege—and no user has more privilege than **root**.
2. Spawn a sub-shell with a *substitute user* to **root**, type: **su**
   The default substitute user is **root** the "ultimate" (admin) user.
   Notice that the prompt has changed from knoppx@.. *to* root@..

fdisk

3. Like DOS, Linux also has a fixed-disk partition table program called **fdisk**. Unlike Windows' version, Linux's fdisk is more powerful and more complex; *be careful not to save any changes you might make*.
4. To open fdisk on the only hard disk (called **hda**--hard disk 'a'),
   type: **fdisk /dev/hda** (or: **/sbin/fdisk /dev/hda**)
5. To display the current partition table, press **p**. Note the device names, start/end cylinders, sizes (in units), and system partition type (with ID).
6. Linux can recognise many system partition file system types. To see a list, press **L**. (The ID from the partition list should match the partition table entry type.)
7. Press **q** to quit the program

cfdisk

8. Linux users decided that fdisk needed a friendlier face, and wrote **cfdisk** (*curses-graphics library* based fdisk).
9. To run cfdisk, type: **cfdisk** (or: **/sbin/cfdisk**)
   The partition listing should match the one from **fdisk**, except nicer.
10. Record the device name (**hda##**) and which partition each points to (*this data may be important in the next parts*).
11. Press **q** to quit the program.
12. To exit the **root** shell, and return to the **knoppix@..** account, type: **exit**

Part 10: Mounting and Unmounting Drives

[The following, requires **root** privilege; *do you remember how to obtain it ?*]

1. Each partition and drive in Linux must be mounted: "attach a device to the subdirectory tree," *a joining that provides access to the device*.
2. Although mounting seems to relate to physical drives (and devices), mounting is also required for remote, or virtual, directories--allowing local directories to span across a network to another server.

Mounting Hard Disk partitions

3. The following is a simple example of mounting a partition (of any operating system) to a current (existing) directory.
4. Make a new directory in the current directory, type: **mkdir win**
   There should be no files in the directory, type: **ls win**
5. The partition structure of the lab's system should have (see ***Part 9***):
   hda2 – Fat32 (which is the Windows98 C:\ drive)
6. To mount the C:\ Fat32 partition, provide the partition and the directory to attach to, type: **mount /dev/hda2 win**
7. If successful, change into the **win** directory (cd) and take a listing (ls).
8. To display a current listing of mounted drives & devices, type: **mount**
9. Change back to the parent directory of **win**, type: **cd ..**
10. Dismounting the partition with the **umount** command (for **u**n**mount**), type either: **umount /dev/hda2** or **umount win**
    (Note: if an error occurs, it is probably that you are still in the ***win*** *directory, and therefore the system will not let you unmount.)*

Mounting Floppy drives and CD-ROMS

11. Floppy drives and CD-ROMS rarely contain the same volumes (disks) for a long time. As they are changed, they must be mounted and unmounted.
12. For this reason, some default directories are usually placed in the **/mnt** directory, but not usually mounted automatically, type: **ls /mnt**

13. On a standard Linux install, the following are example of mounting floppies and CDs *(do not try these now, since both are already mounted)*:
    mount –t msdos /dev/fd0 /mnt/floppy
    mount /dev/hdc /mnt/cdrom
14. Before continuing, make sure you have exited from root privilege, type: **exit** and the prompt should show knoppix@... instead of root@…

Part 11: Writing a small program in C

(Ensure you are not in a root shell; root user should never develop code.)

1. Except for dedicated languages (such as for databases), there exists a Linux compiler, or interpreter, for almost all programming languages.
2. With the popularity of UNIX (and its various flavours) for the last 25 years, C/C++ has become *the* standard language for most all OSes.
3. Open a text editor to begin a new file, type: **nedit smallprogram.c**
4. Code the C program below. When finished, save and quit the editor.

```
#include <stdio.h>
#define MAX 40     // maximum name length

//function prototype
void display (char n[], int times);

//main function (does not need prototype)
int main (void)
{
    char name[MAX];   // user's name
    int loop=0;       // number of loops

    printf ("What is your name? ");
    gets(name);           // get user's name
    printf ("How many times shall I print it? ");
    scanf ("%d",&loop);  // read no. of times

    display (name, loop);
    return (0);
} // end of main()
```

```
//function display (n)ame so many (t)imes
void display (char n[], int t)
{
   int i=0;
   for (i=0; i<t; i++)
       printf ("%s ",n);

   printf ("\n");
}// end of display()
```

5. Quit the editor and return to the command line window.
6. To compile the program and generate the executable, type: **gcc smallprogram.c –o smallprogram.run** *(ignore the warning about* **gets()** *)*
7. If there are syntax errors, edit the file again and fix them.
   To run the program, type: **./smallprogram.run**

   Note: The name of the executable program, in this case **smallprogram.run**, *could be any name. If no output executable name is provided, the gcc compiler creates a default executable file called:* **a.out**

Part 12: Writing a small program in Java

1. Linux provides more than just older programming languages.
   New languages also exist in Linux, such as: Java.
2. Use **nedit** to code the following, and save it as **smallprogram.java**

```
import java.*;
import java.io.*;

public class smallprogram
{
public static void main(String[] args)
{
   String opinion = " likes Linux! ";
   char first=' ', last=' ';   // user's initials
   int loop = 20;      // number of times to repeat
```

```
      try
      {
         System.out.print("Your first initial? ");
           first = (char)System.in.read();
           System.in.read();

         System.out.print("Your last initial? ");
           last = (char)System.in.read();
           System.in.read();
      }
      catch (IOException e)
      {
      }
      System.out.println("Repeating "+loop+" times.");

      for (int i=0; i<loop; i++)
         System.out.print(first+"."+last+"."+opinion);

      System.out.println();

   } // end of main()
   } // end of smallprogram
```

3. Quit the editor and return to the command line window.
4. To compile the program, type: **javac smallprogram.java**
5. If there are errors, return to the editor, and fix them.
   To run the program, type: **java smallprogram**

Part 13: The GUI window manager: KDE

1. Minimise the terminal window to the taskbar.
2. **KDE** (K Desktop Environment) is probably the most popular Linux desktop/window manager for X-Windows, with **GNOME** being a very close second. The one installed with KNOPPIX is KDE.
3. Some other desktop/window managers for X-Windows are:
   WindowMaker, IceWM, FVWM'95, AfterStep, Sawfish, Blackbox, Fluxbox, XFCE, Enlightenment, MWM, WM2, and CDE (similar to KDE).
4. Experiment with the GUI (have fun):
   - activate programs through the taskbar (the gear with the "K")
   - examine the taskbar animations (icons and arrows at the taskbar ends)
   - right-click on the desktop and "Configure Desktop"—change the settings

Web-browsing: GUI

5. Since the network settings were established at the beginning of the tutorial, full Internet access via the CANLab and UCC networks.
6. Open the Mozilla web-browser from the icon on the taskbar.
   Go to **www.cariboo.bc.ca** and a favourite website, or check your e-mail.
7. Much like other types of applications in Linux, there are many available for the same task. Close Mozilla and open the KDE-integrate browser Konqueror (another icon to the taskbar). Go to same sites.

   Are there any differences in how the two browsers display content?

Web-browsing: CLI

8. It is not a necessary requirement that browsing be graphical.
9. Restore the terminal window and type: **lynx http://www.yahoo.ca**
   Examine how the CLI browser Lynx differs from Mozilla.
   Press "Q" to quit Lynx, then minimise the terminal window.

Game Playing

10. Like any good operating system, Linux and the GUI allow for games, games, and more games.
11. For the default-loaded games, select from (K)→Games.
    Some classic games are installed: KAsterioids, Galaga, Penguin Solitaire, Tetris, Chess, and Taipei.
12. Popular favourites are KBounce, Frozen Bubble, and Potato Guy.

Basic User Productivity Utilities

13. Like Windows and MacOS, KDE also comes with some basic utilities:
KCalc – on-screen calculator (K→Utilities)
KHexEdit – very cool fike editor! (K→Utilities→More Programs)
Editors – a bunch of text editors (K→Editors)
XMMS – Winamp-like Mp3 player (K→Multimedia→Sound→XMMS)
Palm Pilot utils – (K→Office→Pilot)
14. A very powerful program for graphic editing is **GIMP** -"GNU Image Manipulation Program." (K→Graphics->The GIMP)

WINE (WiNdows Emulator)

15. Although still in development, the WINE Consortium is building a completely emulated environment for running Windows-based programs (for Win3.1 up to Windows XP).
16. The power of WINE is that it integrates a Windows-based application into the Linux GUI, without requiring any core software from Microsoft.

    Currently, it is still difficult to install/configure, but when running correctly, it has been reported to perform applications as though they were running natively in Windows.
17. WINE is not completely set in KNOPPIX, but is available: (K)→WINE

Part 14: OpenOffice – *The* Linux Office Suite

1. Since Microsoft has not yet released a Linux version of MS Office XP, other organisations have produced equivalent (and compatible) suites.

   The most popular is OpenOffice.org's **OpenOffice**. It is popular because concurrent versions exist for UNIX/Linux, MacOS, <u>and</u> Windows.
2. OpenOffice is available through the "seagull" icon on the taskbar, or K→Office->OpenOffice.org

Part 15: System Configuration

1. UNIX and Linux configuration activities used to be the domain of gurus—individuals that have *mastered* a particular operating system.
2. The entire Linux operating system is configured through the definitions within the files stored in the **/etc** directory (and other directories below it). To list some of the configuration files, type: **ls /etc/*.conf**
3. To make Linux life easier, a number of utilities are available that provide a nice interface to manipulating this files.
4. Some utilities are:
Control Center – the "circuit board" taskbar icon (or K→Settings)
User Management – since Linux is multi-user (K→System→KUser)
Service Run Level Editor – (K→System→SysV-Init Editor) **do not use!!**
Software Install Manager – (K→System→KPackage)
5. Examine the information present in **Control Center**.

Interrupt (IRQ) and I/O Address Listings

6. Using Control Center, and similar utilities, in the GUI, Information about the diagnostic details of the system's hardware is available. Similar information (although not so visual) can be found via the CLI.
7. Restore the terminal window and type: **less /proc/interrupts**
for the device I/O addresses, type: **less /proc/ioports**
8. Some other information is also available through the **/proc** directory.
for the device memory addresses, type: **less /proc/iomem**
for CPU information, type: **less /proc/cpuinfo**
for the Linux version, type: **less /proc/version**

Part 16: Mtools

Caution*: Mtools is useful only if the computer you are using the* server. *In the case of a remote login, the computer you are using is only an access terminal to the actual server, and the terminal knows nothing about *you* floppy disks.*

1. To provide easy access to DOS/Windows volumes in Linux (usually for diskettes), a library of tools were created: **mtools**.
Examples of commands in mtools: **mcopy, mformat, mdir**.

   *(Note**: By using** **mtools*****, diskettes <u>do not</u>*** *require mounting/unmounting, but can be used directly as though in DOS or Windows.)*

2. Although not available in every Linux distribution, mtools can be added separately via a free download: http://mtools.linux.lu

   Use one of the help methods, or just type: **mtools**, to examine the types of commands available.

3. Replace the Configuration floppy in the disk drive with an experimental diskette from your instructor, use the mformat command to format the new floppy, type: **mformat a:**
4. Copy the program files you wrote to A:, type: **mcopy small*.* a:**
5. Display the files on the floppy, type: **mdir a:**
6. After you have finished the tutorial, restart the computer into Windows 2000 Pro. and examine the source code files using Wordpad.

Part 17: Finishing Up

1. Since KNOPPIX loads from the CD-ROM, there is nothing to save regarding this tutorial session on the harddisk, we just need to finish.
2. Select: (K)→Logout.
   Follow any on-screen instructions, and the system will be shutdown.
3. Once the CD is ejected, remove it and the floppy and return them to your instructor. Press <ENTER> and the system will power down automatically.

***DOS/Windows Command Prompt* <u>versus</u> *UNIX/Linux Command* List**

Command Purpose	**DOS/Windows**	**UNIX/Linux**
Change file attributes	attrib	chmod
Clear screen	cls	clear
Common ASCII editor	edit	pico, emacs/xemacs, jed, joe, kedit, nedit, vi/vim, zile, *etc.*
Copy files	copy	cp –i (-i – *interactive prompt*)
Delete files and subdirectories	del or deltree	rm –R
Dir: Change to a directory	cd or chdir	cd
Dir: Display current directory	cd	pwd
Dir: Make a directory	md or mkdir	mkdir
Dir: Remove a directory	rd or rmdir	rmdir
Directory listing	dir	ls
Display ASCII file	type	cat
Display ASCII file (controlled)	more	cat, more, *or* less (less – *most advanced*)
Display command help	*command* /?	*command* --help | man *command* | info *command*
Display disk space	at the end of dir	df
Display environment vars.	set	printenv
Display memory usage	mem	free –t
Display/set date-time	date	date
Echo string to console	echo	echo
Erase files	del or erase	rm –i (-i – *interactive prompt*)
Exit prompt window, shell, or logout	exit	exit
Format a DOS diskette	format	mke2fs *or* mformat
Move files	move	mv –i (-i – *interactive prompt*)
Partition management	fdisk	fdisk *or* cfdisk
Rename file	ren	mv

***Linux <u>was not</u> named after its creator, Linus Torvolds. And he pronounces it* "Lin-ix," *<u>not</u>* "Line-ix."**