Chapter 4 CIT 173

Using a linux desktop environment:

Another use of a lniux system is a network server. If youre using a desktop environment—chances are that youre doing so in order to run productivity software.

You might need to compile programs from source code.

A desktop environment is a set of programs that control the screen. It provides small utility program to perform task such as managing files.

Choosing a desktop environment:

KDE – the K desktop Environment or KDE, is a popular desktop environment for linux. It is the default DE for Mandriva and openSUSE. Its built using the Qt widget set.

GNOME – popular linux desktop ARENA. Default DE for Fedora and Debian distros. Built on the GIMP toolkit or GTK+ widget set.

LXDE – lightweight X11 DE. Intended to consume few resources. Works well on old or modest computers.

Unity – canonical, publisher of ubuntu distro. It became the ubuntu default DE in 2011. Aims for simplicity and provide consistent DE across various desktop and mobile plats.

XFCE – originally modeled on a commercial DE known as CDE, but is built using GTK+ widget set. Provides more config than gnome or unity and consume fewer resources.

Build your Own – you can make your own but it’s more complex. At a minimum you need a window manager. But for the config to be a true DE you need other components like file manager and small productivity tools. All the components need to be accessible from a sort of menu system.

New users from MAC and Windows will be happiest with the KDE environment.

Panels – panels are typically located on the side of the screen in which common apps appear. Unity uses a such a config by default, as does GNOME 3.

Context menu – sometimes right-click in an unsued part of the screen to obtain a context menu w/ a variety of options which may include the option to run programs.

Terminal – you can launch a program called a *terminal*, provides a text mode UI inside a window. Can either run text mode or GUI programs by typing their filenames in this window.

Only a handful of programs appear in GNOME 3 panel, u must either add programs to it or launch apps that fedora devs didn’t include by default in some other way.

You can open apps by finding it in the apps list. You can open this list by starting w/ the kickoff application launcher—accessible via SUSE chameleon icon on the Kicker pane’s far left. Select apps > internet > web browser > firefox.

File managers – Nautilus is GNOME’s default file manager. Because nautilus is similar to file managers in other OS, chances are that youll be able to use its main features quite easily.

The right pane of nautilus shows the home directory’s files and sub categories.

You can add bookmarks for locations not shown in the Places category. Simply traverse to the files within the desired bookmark location and click gear icon on nautilus’ menu bar. From the drop down menu select bookmark this location. Or press CTRL + D. to modify a bookmark do not use the nautilus window. Instead—with nautilus open, select files from top bar in the GNOME desktop and select bookmarks from the resulting drop down menu.

Document properties – you can right click a file and select properties from the resulting context menu. This produces a Properties dialog box.

Working with productivity software:

Area of productivity software is extremely broad – 100s if not 1000s of these type of apps exist and entire books written bout them. Common tool categories include web browsers, email clients, office tools, multimedia apps, cloud computing, and mobile apps.

You can use the menus or other app display tools on your DE to locate productivity apps.

Table of equivalence: if u normally use a particular windows app, u may be able to find a linux substitute by consulting a table of equivalent apps.

Browsers:

Chrome aims to be fast and easy to use. Introduced in 2008. An open source variant called chromium is also available.

Firefox – most popular browser for Linux. Consumes a lot of memory.

Web – originally called Epiphany. Browser for GNOME desktop—designed to be simple and easy.

Konqueror – serves a dual function: both web browser and file manager. Fairly lightweight.

Lynx – a text based web browser instead of a GUI. Useful if don’t want to be bothered with graphics. Useful as a test browser when developing own web pages.

Opera – claims to be unusually fast.

Dark sides of web browsing:

Can log user access data and for marketing or other ways.

Much content is dynamic and websites download small programs. It’s usually harmless but can be used to deliver malware.

Malicious websites can trick users into giving up sensitive data such as financial info by phishing.

Many sites are not secure. Data transferred can be read on intervening computers.

Passwords.

USING EMAIL CLIENTS:

Evolution – powerful GUI email client includes groupware and scheduling features.

KMail – The KDE project’s Kmail is a well-integrated DE

Mutt – one of several text based email readers. Mutt is quite capable despite being text based.

Thunderbird – an email client that’s closely associated w/ firefox web browser.

USING OFFICE TOOLS:

GNOME office: developed independently of each other, but GNOME Office attempts to link them into a coherent whole. AbiWord, Evince, Evolution, Gnumeric, Inkscape, and Ease are part of the package.

Calligra: was born out of a split earlier popular KDE office suite called KOffice. Calligra is thriving.

Words, stage, sheets, flow, and Kexi. Calligra also offers graphics and project management software products.

Apache OpenOffice: until early 2011 this office suite was the most prominent suite for linux. Its sponsor Oracle stopped supporting commercial dev of the project. Now it is ran by Apache Group and includes 6 apps: writer, calc, impress, base, draw, and math.

LibreOffice: created as a fork of the older pre-apache. Becoming the most popular linux suite. Provides 6 apps: writer, calc, impress, base, draw, and math.

Most of these office suite apps support the OpenDocument format or ODF. Which is an open set of file formats that’s slowly making inroads as a standard for word processing spread sheet, and other office files. Intended to enable easy transfer of files across apps, app specific assumptions often hinder such transfers, especially complex docs.

Lyx for instance can take the place of a word processor, but it’s built as a way to create and edit LaTeX documents. LaTeX is a document format that’s popular in computer science, mathematics, and other technical fields.

Multimedia Apps:

Up until recently Linux’s multimedia capacity has been lacking.

Audacity.

Blender – create 3D images including stills and animations.

GIMP, or GNU image manipulation program – a still image manipulation program like Photoshop.

ImageMagick – Suite of graphics programs. You use the programs from the command line. Convert file formats, add frames to images, and so on.

HandBrake – Program provides an easy way to convert between video formats into formats that use the efficient H.264 encoding.

MythTV – turn a regular PC into a digital video DVR using this software.

Cloud computing: storage of computer software are or data over the internet rather than locally storing it on you computer. In practice – complications can arise. For instance, might require that you use a particular web browser or have a specific browser plug in installed. If provider supports a wide range of browsers as clients you shouldn’t have problems with cloud.

On demand streaming media like Netflix

File storage devices like google drive

Office productivity suites like Zoho office or Google Docs

Web-based email like Gmail.

Using mobile applications

Although android is a linux based OS, for the most part it runs entirely different applications than do desktop or server implications of Linux. Mobile computing typically focuses on small programs known as apps. You can use Google Play for installing apps on android based.

Using server programs:

Identifying common server protocols and programs:

Networks including the internet function by means of network *protocols*. They are clearly defined descriptions of how two computers should exchange data to achieve a particular end like transferring email or delivering a file to be printed. Most protocols are described in one or more standards documents known as Request for Comments documents RFC, each of which has a number. These defines the protocol, each of which has a #. Most network protocols involve transferring data over one or more ports. You can think of a port as being something like a telephone extension number, the main number or the IP address identifies the computer as a whole and the port identifies the protocol being used. Server program attaches itself to a port # and receives all incoming request on that port.

Port # 20-21, protocol: FTP.

Port # 22, protocol: SSH (remote access tool and supports file transfers)

# 23, telnet, old unencrypted remote access tool.

25, SMTP, moving email on the internet

53 DNS, enables PC to look up IP by providing hostname or vice versa.

67 BOOTP, DHCP, enable computer on local network to help automatically config other computers to use a network.

80 HTTP basis of the web

109-110 POP2,POP3, enables recipients to initiate an email transfer so it’s used as the last leg in email delivery

118, SQL – a network-enabled database interface language.

137-139, SMB, file and printer sharing

143-220 IMAP, another email…. protocol

389 LDAP, for accessing directories or database

443 HTTPS, secure version of HTTP

2049 NFS, a protocol and a server of the same name for file sharing between unix and unix like os’s.

Focusing on web servers:

A web server delivers web pages to internal and or external network users.

Apache HTTPD: part of the prevalent Linux Apache MyQL PHP (LAMP) stack for web apps. Original package released in 1995.

Nginx – released in 2002. Can retrieve resources on behalf of a client from one or more servers, as well as operate as a mail server.

Installing and launching servers:

You can install servers in the same way that you install other software. Once the software is installed you must launch a server. You launch a server by configuring the computer to run it automatically whenever it boots and runs in the background—as a *daemon*—that is, as a process that runs unattended.

Most servers are started automatically when linux boots. You can also open a terminal program and type a text-mode command along with a keyword such as start or stop. Various distros use a particular initialization daemon both to start and to manage the various server daemons.

Which initialization daemon it uses: System V init (SysVinit), Upstart, system

Super server programs run constantly, keeping the servers they manage unloaded except when they’re seldom-used servers. Can also function as a security feature.

Securing servers:

Servers can contain bugs that enable outsiders to abuse the software to run programs locally

You can misconfigure a server, granting an outsider greater access to your system than you had intended.

Users with accounts and remote access via server can abuse this trust. Risk is worse when combined with a server bug or misconfig.

A server can be used as a stepping-stone to attack others, making it appear as if an attack originated from your computer.

Even without breaking into a computer, an attacker can swamp a server with bogus data, thus shutting it down. This technique is called a DOS.

A server requires paying attention to each of the risk factors. Steps to take to secure server include:

You should keep your server programs up to date by using your package mgmt tools to upgrade servers whenever upgrades become available.

You should learn enough about server config to be sure that you can config your servers properly.

You should remove unused accounts and audit necessary accounts to be sure that they use strong pwd.

Use a firewall config to restrict outsiders access to server computers. Can also minimize the risk of one of your computers being used to attack others.

Managing programming languages.

Choosing a compiled v s an interpreted language:

Compiled: programmers convert (or compile) a program written in a high level language from its OG source code form into the machine code form. Can take some time. From seconds to hours. Compilation can also fail because of errors in the program. When the compilation succeeds, the resulting machine code executes quickly.

Interpreted languages: programs written in interpreted languages are converted to machine code at the time they are run, by a program known as an interpreter. Happens on line by line basis. Meaning the program is never completely converted to machine code; the interpreter figures out what each line does and then does that one thing. This means that interpreted programs run much slower than compiled programs. Interpreted programs are much easier to develop though and easier to modify. Just open the program file in a text editor and save it back.

In addition to compiled and interpreted, there is also assembly language. I is a simple one to one correspondence between machine code #’s and symbols that the programmer uses. A skilled assembly language programmer can produce compact and efficient programs. Assembly is harder than writing programs in most high level languages.

In theory, most languages can be implemented either in compiled or INT form. Burt most languages are commonly used in just one form or the other.

Some languages don’t fit neatly into either category. Some fall in between like java.

Identifying common programming languages:

Assembly: difficult to write and nonportable. Each architecture has its own language.

C: the most important compiled language for linux, since most of the linux kernel as well as apps are written in C. c is efficient and easy to write. However it’s easy to write buggy programs bc it lacks error checking features. In linux, c programs are generally compiled with the gcc program, the GNU compiler collection package. File names can end in .c or .h

C++: an extension to C that adds object oriented features meaning greater emphasis is given to data structures and their interactions than to the procedures used to control the flow of the program. KDE and Apache openoffice/libre office are written largely in c++. C++ filenames can end in .cc, .cpp, .cxx, or .c++.

Java – created by Sun Microsystems, now Oracle as a cross platform language that’s somewhere between being compiled and interpreted. Java source code usually has a name that ends in .java

Perl – interpreted language that is designed for easy manipulation of text, but it’s a general purpose language that can be used for many other tasks as well. End in .pl, .pm, or .t

PHP, hypertext preprocessor was created for use on web servers in order to generate dynamic content—that is, content that varies depending on the user, the time of day, or some other criterion. An interpreted language and requires PHP aware web server like Apache. Can support user logins, shopping carts, diff content based on user location, etc. most often have endings in .php

Python: interpreted language makes code readability a major goal. Supports object orientation. Used for scripting purposes but can be used to write more complex programs too. Often use .py extensions.

Shell scripting: most linux text mode shells – the programs that enable entirely keyboard based use of the computer—provide their own interpreted languages. The BASH or (born again shell) is the most common, so bash scripting is quite common. Many of the files that control the linux process are in fact bash scripts. No unique filename extensions although some DO use .sh

Handling software packages:

Understanding software packages:

Software programs are bundled into a prebuilt package that has simplified their installation and mgmt. Uses a package mgmt. system (PMS). Theses packages are stored on repositories which are official software storage servers on the internet. Can be accessed over internet via your linux system’s local PMS utilities. Repositories have a lot of software packages stored on them ready to be explored and installed.

Common package tools:

Dpkg: low level package tool used as foundation of Debian based family of PMS tools. Can be used directly to install, manage, and remove software packages.

Rpm: is also a low level package tool similar in function to the dpkg utility. Used as foundation of the Red Hat Linux package mgmt. sys. It’s best to use a higher level PMS utility.

Apt-get: text mode tool for the Debian PMS. With apt-get, you can install from repositories and remove software packages from your local Linux sys. Can perform package upgrades for individual packages, all of packages on sys, or entire distro.

Yum: text mode tool for Red Hat PMS. Used on distro such as Red Hat Enterprise Linus of RHEL, fedora, and centOS. Can install from repositories, remove software packages from your local linux sys, upgrade packages, and so on. Can use for determining various pieces of info concerning packages and their mgmt., such as displaying a list of the PMS’s configured repositories.