**IN 2200- Software Engineering Methods Reg no. 094078G**

Large software systems may be the most complex things ever built and most of them are still growing. This places great weights on the software engineering procedure, which must be ordered and controlled. These software systems are not static; they change regularly both during development and after deployment. Major software can be categorized into 3 sections

1. Operating Systems
2. Application Programs
3. Programming software

**1) Linux Operating System**

Under the Operating Systems category one of the major projects was Linux Operating System. The development of Linux is one of the most conspicuous examples of free and open source software collaboration; naturally all the underlying source code can be used, easily changed, and reorganized, both commercially and non-commercially, by anyone under licenses such as the GNU General Public License. Typically Linux is boxed in a format known as a Linux distribution for desktop and server use. Some popular conventional Linux distributions include Debian (and its derivatives such as Ubuntu), Fedora and openSUSE. Linux distributions include the Linux kernel and auxiliary utilities and libraries to fulfill the distribution's anticipated use.

**Main Goals of Linux Operating System**

When considering this kind of open source software it is hard to define some particular goals, because it always grows up. When users feel needs then the operating system also has goals.

System goals are

1. Performance

Linux offers high performance on many networks. It has the capability to handle large numbers of users at once.

1. Security

Computer security is a much exposed topic in relative to the Linux kernel, because a large number of bugs in the kernel can be possible security defects, whether they allow for privilege growth or create denial of service attack paths. Over the years, numerous such flaws were found and fixed in the Linux kernel. Many new security features have continuously been implemented to address computer insecurity issues in the Linux kernel.

1. Portability

Linux is not initially designed to be portable, Linux is now one of the most broadly ported operating system kernels, running on a varied range of systems from the iPAQ (a handheld computer) to the IBM Z/Architecture (a massive mainframe server that can run hundreds or even thousands of concurrent Linux instances). Linux also runs as the main operating system on IBM's Blue Gene supercomputers. Linux is the OS on 91.8% of systems on the Top 500 supercomputers list (2010). Linux has also been ported to numerous handheld devices such as TuxPhone, Apple's iPod and iPhone. The Google Android, HP webOS, and Nokia Maemo operating systems, developed for mobile phone devices, all use improved versions of the Linux kernel.

1. Virtual Machines

The Linux kernel has wide provision for and runs on many virtual machine architectures both as the host operating system and as a guest operating system. The virtual machines usually follow Intel x86 families of processors, though in a few cases PowerPC or ARM processors are also emulated.

1. Stability

Linux has high constancy compared with other operating systems. There is no need to reboot the system to maintain performance levels. Rarely it stuck up or slows down and it has a continuous up time of hundreds of days or more.

1. Flexibility

Linux is very flexible and it can be used for high performance server applications, desktop applications, and embedded systems. We can install only the needed components for a specific use. We can also restrict the use of specific computers.

1. Compatibility

Linux system runs all common Unix software packages and can process all common file formats.

1. Networking

Linux provides a strong support for network functionality; client and server systems can be easily set up on any computer running Linux. It can perform tasks like network backup faster than other operating systems.

**Main Components**

1. Linux Kernel

Linux has the concept of Monolithic Kernels

- They are the different, old-fashioned concept. Here, the entire code of the kernel including all its sub systems such as memory management, ﬁle systems, or device drivers is filled into a single ﬁle. Each function has entrée to all other parts of the kernel; this can result in elaborately nested source code if programming is not done with great care

Simultaneous Processing

-Linux is a multitasking system; it supports what seems to be simultaneous execution of several processes. Since only as many processes as there are CPUs in the system can actually run at the same time, the kernel switches (unnoticed by users) among the processes at short breaks to give them the impression of simultaneous processing.

Page Tables

- Data structures known as page tables are used to map virtual address space to physical address space. The easiest way of implementing the association between both would be to use an array containing an entry for each page in virtual address space. This entry would point to the related page frame.

Allocation of physical memory

- When it assigns RAM, the kernel must keep track of which pages have already been assigned and which are still permitted in order to avoid two procedures from using the same areas in RAM. Because memory allocation and release are very regular tasks, the kernel must also guarantee that they are finalized as quickly as possible. The kernel can assign only whole page frames.

Timing

- The kernel must be capable of calculating time and time variances at numerous points

System Calls

- System calls are the classical method of permitting user processes to cooperate with the kernel. The POSIX standard deﬁnes a number of system calls and their effect as applied on all POSIX compliant systems including Linux

Device Drivers

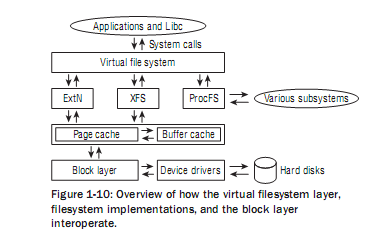
- The role of device drivers is to interconnect with I/O devices which are attached to the system; for example, hard disks, ﬂoppies, interfaces, sound cards, and so on. In accordance with the classical Unix maxim access is performed using device ﬁles that typically reside in the /dev directory and can be handled by programs in the same way as regular ﬁles.

Modules

-These Modules are used to dynamically add functionality to the kernel at runtime device drivers, ﬁle systems, network protocols, practically any subsystem of the kernel can be modularized.

Caching

-The kernel uses caching technique in order to improve system performance. Rather than reading from the RAM reading from the cache is faster. Caches are also organized into pages.



1. Graphical User Interface

Normally users work in a Linux based system in several ways such as a command line interface (CLI), a graphical user interface (GUI), or through controls attached to the related hardware, which is common for embedded systems. Normally for desktop systems, the default mode is usually a graphical user interface, by which the CLI is available through terminal emulator windows or on a distinct virtual console. Most low-level Linux components, including the GNU userland, use the CLI absolutely. The CLI is particularly matched for automation of tedious or delayed tasks, and provides very simple inter process communication. A graphical terminal emulator program is frequently used to access the CLI from a Linux desktop. A Linux system naturally implements a CLI by a shell, which is also the traditional way of interrelating with a Unix system. A Linux distribution dedicated for servers may use the CLI as its only interface. Linux employs the X Window rendering system for presenting data on screen. Because X Window delivers only the basic window formation, it relies on third party user interfaces to deliver the window borders and buttons, menus, icons and desktop that users can see and operate. KDE and GNOME are two of the most common (see KDE and GNOME). A Linux delivery may contain more than one user interface, all of which appear on screen somewhat similar to the Windows and Mac interfaces. See X Window.

1. GNU C library

The kernel delivers frequent secondary functions that resemble the standard C library found in user space, but things are much more economical in the kernel domain.

1. Kernel modules

In computing, a loadable kernel module (or LKM) is an object file that contains code to extend the running kernel, or so called base kernel, of an operating system. Most current Unix like systems, and Microsoft Windows, support loadable kernel modules. LKMs are normally used to add support for new hardware and/or file systems, or for adding system calls. When the functionality provided by an LKM is no longer required, it can be unloaded in order to free memory.

**Design & Functionality**

Separate projects that interface with the kernel deliver much of the system's higher-level functionality. The GNU userland is a significant part of most Linux based systems, providing the most common application of the C library, a common shell, and many of the common Unix tools which carry out many basic operating system tasks. The graphical user interface (or GUI) used by most Linux systems is built on top of an implementation of the X Window System.

As well as those designed for general purpose use on desktops and servers, distributions may be focused for different purposes including: computer architecture support, embedded systems, stability, security, localization to a specific region or language, pointing of specific user groups, support for real time applications, or assurance to a given desktop environment. Furthermore, some distributions deliberately include only free software. Currently, over three hundred distributions are actively developed, with about a dozen distributions being most popular for general-purpose use. The popularity of Linux on standard desktop computers (and laptops) has been increasing over the years. Presently most distributions include a graphical user environment. The two most common such environments are GNOME and KDE Plasma Desktop, both of which are established and support a wide variety of languages

**2) Mozilla Firefox Browsers**

Mozilla Firefox is the next generation open source Internet browser from the Mozilla Foundation, and is set to flourish Mozilla Navigator as the default browser for the Mozilla set of applications at some point in the near future. Firefox and its parallel project Mozilla Thunderbird (the new Mozilla mail and news client) are standalone projects which can be run in isolation from one another and making it possible to substitute your tired standard browser with a fresh copy of Firefox. The Firefox project was started in 2003 with the aim of becoming the best browser for Microsoft Windows as a result of the disillusionment of a group of developers with the current Navigator program. This software belongs to the Application category.

**Main goals of Mozilla Firefox**

1. Reduce at least 50% of the I/O operations on the main thread (read, write, fast sync, fast stat) based on a certain set of criteria.
2. Obey the standard described in RFC 1945 and RFC 2068.
3. Don't break existing web servers.
4. Don't break existing log-file analysis software or user-agent parsing code.
5. Keep the User-Agent string reasonably short.
6. Use a consistent, obvious, and easy-to-parse format.
7. Easy Browsing

Browser has more features and tools included to make the browsing more efficient.

1. High performance

Page loading is much faster than the other browsers. When there are graphics and other multi-media, it will load them in a short time. This software will use the hardware very efficiently.

1. High Security

This includes the features like Anti-Malware, Secure Updates, Clear Recent History, Customized Security Settings, Anti-Phishing, Content Security Policy and Private Browsing.

1. Powerful Personalization

It gives the ability to customize the interface and lots of ways to customize it.

1. Use up to date technologies

It uses the advanced technologies like HTML5 (the new version).

**Main Components**

1. User Interface
2. Browser Engine
3. Rendering Engine
4. Networking
5. UI backend
6. JavaScript Interpreter
7. Data Storage

User Interface

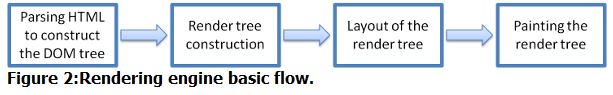
-Like other browsers Firefox also includes general features like address bar, back/forward button, options menu, place to view the requested page, search bar for easy search, Browse multiple sites at once(tabs), Download Manager, Firefox Support, RSS Feed Reader, Offline Browsing, Pop-Up Blocker.

Browser Engine

-This is the interface for querying and manipulating the rendering engine

Rendering engine

- Responsible for displaying the requested content. As an example if the requested content is HTML, it is in control for parsing the HTML and CSS and displaying the parsed content on the screen. The rendering engine will start getting the contents of the requested document from the networking layer.



The styling information together with visual commands in the HTML will be used to generate another tree (the render tree). The render tree encloses rectangles with visual features like color and dimensions. The rectangles are in the right order to be displayed on the screen. After the creation of the render tree it goes through a "layout" process. This means giving each node the exact coordinates where it should appear on the screen. The next stage is painting, the render tree will be traversed and each node will be painted using the UI backend layer.

Networking

-Mainly this component is used for the network calls, like HTTP requests. It contains a platform independent interface and beneath executions for each platform.

UI backend

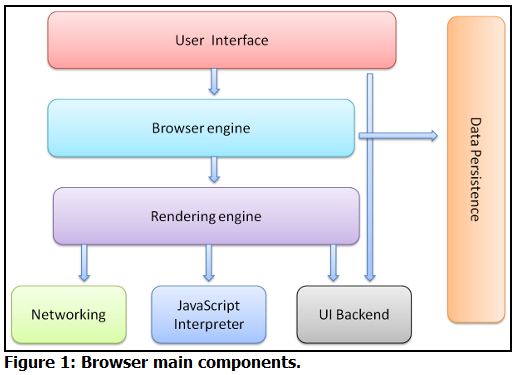
-This is used for drawing basic widgets like combo boxes and windows. It disclosures a common interface that is not platform specific. Below it uses the operating system user interface methods.

JavaScript Interpreter

- This is used to parse and execute the JavaScript code.

Data Storage

- This is a perseverance layer. The browser needs to save all sorts of data on the hard disk, for examples, cookies. The new HTML specification (HTML5) defines 'web database' which is a complete (although light) database in the browser.



**Design and Functionality**

Firefox supports most elementary Web standards including HTML, XML, XHTML, CSS (with extensions), JavaScript, DOM, [MathML](http://en.wikipedia.org/wiki/MathML), SVG, XSLT and [XPath](http://en.wikipedia.org/wiki/XPath). Firefox's standards support and growing acceptance have been credited.

Cross platform support

- Mozilla Firefox runs on definite platforms which has the OS versions in use at the time of release. In 2004 Version 1 supported older OS such as Windows 95 and Mac OS X 10.1, by 2008 version 3 required at least OS X 10.4 and even Windows 98 support ended. Firefox uses the same profile arrangement on the different platforms, so a profile may be used on multiple platforms, if all of the platforms can access the same profile; this includes, for example, profiles stored on an NTFS (via FUSE) or FAT32 partition reachable from both Windows and Linux, or on a USB flash drive. This is useful for users who dual-boot their machines. However, it may cause a few problems, especially with additions.

Aero peek capability

- Mozilla has incorporated aero peek ability for each tab on Windows 7. This feature was before not been enabled by default (but can be user enabled), but now is included as a full feature of Firefox, this causing in a displayed thumbnail image of the tab.

Security

- Firefox contains many features designed to increase security. Key features contain a sandbox security model, same origin policy, external protocol whitelisting, a phishing detector, and an option to clear all private data, such as browser history and cookies.

Tabbed Browsing

- Firefox supports tabbed browsing, which allows users to open several pages in one window. This feature was accepted over from the Mozilla Application Suite, which in turn had borrowed the feature from the popular [MultiZilla](http://multizilla.mozdev.org) extension for Mozilla.

Pop-up blocking

- Firefox also includes integrated customizable pop up blocking. It can also be turned off completely to allow pop ups from all sites. Firefox's pop up blocking can be problematic at times it avoids JavaScript based links opening a new window while a page is loading unless the site is added to a "safe list" found in the options menu.

Private Browsing

- Private Browsing is the latest feature inclusion in Mozilla Firefox. It was introduced in Firefox 3.5, which released on June 30, 2009. This feature lets users to browse the Internet without leaving any traces in the browsing history.

Download manager

- An integrated customizable download manager is also included. Downloads can be opened automatically depending on the file type, or they can be saved directly to disk. By default, Firefox downloads all files to a user's desktop on Mac and Windows or to the user's home directory on Linux, but it can be configured to prompt for a specific download location.

Live bookmarks

- Powered by RSS or Atom feeds, "Live Bookmarks", allow users to dynamically monitor changes to their favorite news sources. When this feature was first introduced in version 1.0 PR, there were a few worries that Firefox was beginning to include non-essential features and that it was beginning to bloat the browser much like the Mozilla Suite.

**References**

**For Linux**

1. wikipedia.org/wiki/Linux
2. www.linux.com
3. www.linuxfoundation.org
4. www.kernel.org
5. linux.co.uk
6. linuxtoday.com
7. www.gnu.org/gnu/linux-and-gnu.html
8. www.linuxdoc.org
9. en.wikipedia.org/wiki/Linux\_kernel
10. www.linux.com/directory/Components

**For Mozilla Firefox**

1. en.wikipedia.org/wiki/Firefox
2. www.mozilla.com
3. www.mozilla.org
4. www.spreadfirefox.com
5. forums.mozillazine.org
6. developer.mozilla.org
7. welloiledpc.com/firefox.htm
8. www.lesliefranke.com
9. www.lesliefranke.com
10. www.expertreviewnow.com