Understanding the Linux Operating System.

Katelyn Stapleton

Southern New Hampshire University

IT-365-Q5210

June 21, 2020

Abstract

Microsoft Windows and Macintosh are not the only operating system available on the market. A well-known alternative is the Linux operating system. The Linux operating system is an open-source operating system with many versions available to select. Linux has distinct features that are important to understand. From the user’s perspective, the Linux operating system operates by text input/output instead of a graphic user interface. The operating system considers everything as a file. To help understand the Linux operating system better, an analysis was done covering the system components, file system, distribution systems (also networking), and program developer considerations. It does not cover a specific version of Linux. The study provides an analysis of the Linux OS in general.

Keywords: Linux, Operating System, File System, System Components, Distributed Systems, Networking, Program Development, Linux OS

Understanding the Linux Operating System.

Outside of the technological community, Linux is an operating system that many people have heard of, but do not know much more than “it is different from Windows and Mac”. Linux is an open-source operating system with many versions available. Before using, change or distributing the Linux operating system, it is important to understand the system components, file system, distribution systems (networking), and program development considerations.

**Linux System Components**

**Process Management**

Process management is controlling the processes including creation, removal, establishing communication, and synchronizing processes (Silberschatz et al., 2010). A process is set of instructions executing a specific task from a user, application, or the operation itself (Franklin & Coustan, 2000; Tutorialspoint, n.d.). Commonly, a process is also defined as a program in execution (Educba, 2019).

The Linux operating system employs two different forms of processes: the foreground process and the background process (Kili, 2017). The foreground processes is an interactive process between the user and the terminal (Ilyas, 2019). Foreground processes are visible to the user, require user to execute the program, and need user input to perform operations (Ilyas, 2019). A Linux shell can only support one active foreground process per terminal (Hass, 2019). A Linux that uses a graphic user interface (GUI) can support more than one active foreground process (Hass, 2019). A background process is a non-interactive or automatic process initiated by the system or user (Kili, 2017). However, the actual process is invisible to the user (Kili, 2017; Ilyas, 2019). Multiple background processes can run concurrently within memory limits (Hass, 2019). A special type of background process is the daemon process. Daemon processes are system-related background processes with authorized permissions of root and services requests. Daemons are not controlled by a terminal, but often are waiting for a given command or input (Tutorialspoint, n.d.). One simple example is a printer daemon is waits for printer commands before executing (Tutorialspoint, n.d.).

Linux assigns and tracks processes by assigning two five-digit identification number, the child process ID and the parent process ID (Tutorialspoint, n.d.). The child process ID (PID) is a unique ID number exclusive to the specific process (Tutorialspoint, n.d.). The parent process ID (PPID) is unique when compared to other parent processes. However, every child process has PPID to dictate its origin (Tutorialspoint, n.d.). After all PIDs are used, Linux reuses the ID number of discontinued processes that are no longer associated with the given ID (Tutorialspoint, n.d.). Linux uses a process table in kernel to document the process IDs and the additional corresponding data/information (Sharma, 2017).

Considering the cycle between the parent process and child process, the important of two separate IDs are clear (Tutorialspoint, n.d.). With the exception of the OS initialization (init) process, every process has a parent processes (LiveFire Labs, n.d; Tutorialspoint, n.d.).). The parent processes forks a new process, known as the child process (LiveFire Labs, n.d.). Initially, the child process is an exact copy of the parent process (LiveFire Labs, n.d.). The parent process is sleeping while the child process is working (LiveFire Labs, n.d.). Once the child process is finished its work, the child process dies and awakes the parent process to continue functioning (LiveFire Labs, n.d.). This cycle is known as the Fork-and-Exec (LiveFire Labs, n.d.).

Two additional processes exist in Linux system resulting from errors within the system or forking (Tutorialspoint, n.d.). A zombie process is a process that is closed and no longer is use, but still listed as an entry on the process table (Tutorialspoint, n.d.). An orphan process is process where the parent process was closed before the child process closed (Tutorialspoint, n.d.).

Overall, Linux processes have four statuses: running, waiting, stopped and zombie. Running is when a process is running or ready to run (Tutorialspoint, n.d.). Waiting is when a process is waiting for an event to occur or system resources (Tutorialspoint, n.d.). Stopped is when a process is stopped (Tutorialspoint, n.d.). Zombie is when a process is dead, but still listed on the process table (Tutorialspoint, n.d.).

**Memory Management**

Memory management is the regulating, allocating, and utilizing the memory within an operating system (Guru99, n.d.-a). Even within Linux, memory management is a vast topic with many variations available (GoLinuxCloud, 2020). Linux uses the concept of virtual memory through dynamic allocation (GoLinuxCloud, 2020; Jones, 2008). If installed within the system, Linux uses a memory management unit (MMU) to handle virtual memory including management (Ott, 2016).

Within Linux, the types of memory used for allocation are physical memory and virtual memory (GoLinuxCloud, 2020). Physical memory is the actual main memory within the operating system. Physical memory is assigned to a physical address, used by the hardware (Ott, 2016; StudyTonight, 2019). Virtual memory uses the hard disk as an extension of the RAM to increase the size of memory available (Weeks, 2004). Virtual memory assigns a logical address generated by the CPU, which are simply pointers (Guru99, n.d.-b; GoLinuxCloud, 2020). Virtual memory provides built-in memory protection. The kernel’s RAM marked as invisible to user space processes (Ott, 2016). Processes can use different maps for allocating memory (Ott, 2016).

Linux manages the memory blocks by implementing paging (Duarte, 2009). Paging is a form a dynamic allocation with multiple possible structuring techniques (Silberschatz et al., 2010). The CPU maps the logical address to physical addresses using page tables. (Duarte, 2009; Silberschatz et al., 2010). Paging breaks physical memory into fixed blocks (frames) and breaks the virtual memory into matching fixed-size blocks (pages) (Silberschatz et al., 2010). Linux swaps memory through demand paging, where pages are only loaded on demand (Gupta, 2018). If the page is not already located in the page table, a page fault occurs. When a page fault occurs, the CPU attempts to make the page accessible to memory or terminates the page (when illegally accessed) (Gupta, 2018). Page hit occurs when the page is already listed in the table. Pages are eventually removed from the table, but the schedule varies based on the selected paging structure (Silberschatz et al., 2010). Overall, virtual memory is designed to utilize current memory, increase the amount of available memory, and optimize memory exchanges and usage.

**Linux File Management**

File management is one of the key features that separates the Linux operating systems (OS) from other operatins systems (Zand, 2019). In the Linux OS, all programs, directories and files are defined as files (Zand, 2019). The files are usually given names for users to comprehend the differences and application of the file. Linux manages files by using a single hierachiical directory structure, known the a tree file system (Jones, 2008; Zand, 2019). The Linux file system starts at the root directory with a certain preestablished directories. Directories can host subdirectories and files, continuing to branch as requested (Jones, 2008). All files and directories can be traced back to the root directory (Jones, 2008).

By default, Linux uses a terminal or shell interface to interaction with users (Buzdar, 2018). In the terminal, users inputs a command(s), the command is processed by the system, and the output (if applicable) is displayed to the user (Ellingwood, 2014). Users can navigate through the system, change permissions, view files, and modify the operating system (Ellingwood, 2014). Graphical user interfaces (GUI) are available for Linux operating systems, but usually will needed to be installed by the user first (Buzdar, 2018).

**System Resource Management**

System resources are any parts of the computer needed for the operating system to function properly (Koutoupis, 2018). Some examples of system resources are the CPU, graphic cards, memory, networks, memory, and input/output addresses ((Fisher, 2019; Oracle, 2020). The Linux Operating System manages the system resources through the Linux kernel (RedHat, 2020). The kernel is the base components of the operating system (RedHat, 2020). The kernel manages the system resources and communications with hardware (RedHat, 2020). It handles the memory, process, and file management (RedHat, 2020). To some extent, the administrative user can also manage the system resources (Oracle, 2020). The Linux operating system also includes an administrative layer for users to execute and configure system level tasks (RedHat, 2020). Users can entered commands or use diagnostic tools to monitor and analyze the usage of system resources (Oracle, 2020). Based on permissions and accessibility, users can tune the system resources as necessary (Oracle, 2020). An administrative user can access and modify control groups to perform resource limitation, prioritization, accounting, and control (Koutoupis, 2018). Other options available for users are to address or fix system errors, such replacing hardware or simply restarting the computer (Koutoupis, 2018).

# The Linux File System

**File structure**

Maintained by the Linux Foundation, the Linux operating system uses the filesystem hierarchy standard (FHS) to define the directory and directory contents (Linux Foundation Group, 2015). Everything is a file in Linux, including directories. Files often act as pointers, directing the command or operation to a certain location (Wong, 2019; Christensson, 2019). Overall, all file paths start in root directory, even when stored on different devices (Sohail, 2016). In Linux, the root of the file system is simply indicated by a / (forward slash). According to the Linux Foundation, the root file system consist of 15 primary directories with different purposes (See Table 1) (Linux Foundation Group, 2015; Both, 2016).

**Table 1**

***Primary Root Directories***

|  |  |
| --- | --- |
| **Directory** | **Purpose** |
| /bin | User command binanries. User executable files |
| /boot | Static bootload. Files to boot the system |
| /dev | Device files for hardware. (not device drivers) |
| /etc | Host-specific/local system configurations |
| /home | User home directory (optional) |
| /lib | Shared library files images need to boot the system |
| /lib<qual> | Alternare short for shared library files (optional) |
| /media | Removable media mount point |
| /mnt | Temporary mount point for file systems |
| /opt | Add-applications/software packages. Optional files |
| /root | Home directory for root user (optional). |
| /run | Run-time variable data |
| /sbin | System binanries |
| /srv | Data for services |
| /tmp | Temporary directory/files |

Linux is an open-source operating system. As a result, numerous file system types are available in the market. The file system types varying in the designs storage format, metadata structure, and other forms (Both, 2016). Some commonly mentioned types are ext2, ext3, ext4, BTRFS, exFAT or NTFS (Windows), and XFS (Garrison, 2017). Linux can operating using a single or multiple file systems, both old and new (Both, 2016; Garrison, 2017).

**File Type**

Linux considers everything on the operating system as a file (Anne, 2016). Linux does not differentiate between files and directories (Kili, 2016). By default, Linux has three primary file types (Anne, 2016). The primary files types are regular files, directory files, and special files containing five secondary file types (Anne, 2016). Regular files are files containing text, data, and program instructions (Kili, 2016). Common regular files are readable files, binary files, image images, and compressed files (Anne, 2016). Directory files are files that store regular and special files (Kili, 2016). The directory files are organized in the hierarchy of the Linux file system (Kili, 2016). The third main file type is special files, totaling to secondary five types (Anne, 2016). Special files do not have a specific definition, but can be categorized as either as device files or communication-related files. The secondary file types are block files, character device files, pipe files, symbolic link files, and socket files. Block files are device files that can transfer large data blocks and provide buffered access to hardware (Kili, 2016). Character device files provide the serial stream of the input or output, one character at a time (Jones, 2009; Kili, 2016). Pipe files, also known as named pipes, are files that redirect the output of a command(s) into the input to another command (Kili, 2016). Symbolic link files are files that serve as a reference link to a specific directory file or regular file (Kili, 2016). Socket files exchange data and information between applications and processes running in different environments (Kili, 2016).

**File Access**

Linux utilizes the Virtual File System (VFS), also known the Virtual File system Switch (VFS), as its application programing interface (API). On a simple level, the VFS is described as an interface or layer between the kernel and actual file system (Ebrahim, 2018). The VFS allows users applications to access and perform file input/output (I/O) on the multiple file systems located on the devices (Jones, 2009). It sends a single set of commands to the kernel to access the file system(s) (Both, 2016). The VFS calls the required device driver needed to interface with the file system(s) (Both, 2016). Once the device driver receives the general commands, the driver interprets the commands into a commands specific to file system type on the disk partition or logical volume (Both, 2016).

The Virtual File System is based on the concept of object-oriented programming (Silberschatz, Galvin, & Gagne, 2009). The VFS uses four object types as a set of operations to connect to the function (Silberschatz, 2009). Implementation of the file object is needed to perform any function specific to the file object type (Silberschatz, 2009). The first object type is an index node (inode) (Jones, 2009). Inodes contains the metadata about a file, but not the file name or actual data (Christensson, 2019). Each inode has a unique index (inode) number along with the file system ID (McKay, 2020). The inode is paired with a filename and acts as a pointer to the file data (Christensson, 2019). However, the number of inodes is limited in each storage device (Christensson, 2019). The second object type is the directory entry (dentry) (Jones, 2009). Every directory is considered a file in Linux (Silberschatz, 2009). Dentries connect the inode number, file name, and location together (Silberschatz, 2009; Jones, 2009). It contacts a component name, pointer to the parent dentry and a pointer to the inode (Wong, 2019). Dentries build a memory cache that represents a file directory (Jones, 2009). The third object type is the superblock (Jones, 2009). The superblock represents the entire file system, containing high-level metadata about the file system (Silberschatz, 2009; Jones, 2009). The superblock defines the managing parameters of the filing system and provides structural information to the kernel. The superblock provides access to the inodes (Silberschatz, 2009). It is mounted on each disk drive, each networked drive, and in memory (Silberschatz, 2009; Jones, 2009). It is a highly important object (Jones, 2009) for the file system. The final object type is the file object (Jones, 2009). For each file, a file object exists. It contains the metadata about the file including file history (Jones, 2009). The file object includes a set of file operations such as read, write, open, close and more (Jones, 2009). It also contacts the flags and permissions settings of the file (Jones, 2009). For a brief summary of how the objects connect, the file object refers to the dentry object (Jones, 2009). The dentry object refers to the inode object (Jones, 2009). Then the inode and dentry refer to the underlying superblock (Jones, 2009).

**Linux Operating System Protection**

As for protection, the VFS is an abstraction layer between the kernel and the system-call interface (Jones, 2009). The abstraction layer hides the information on how the file system is specifically executing the behavior. Users cannot directly interfere with or view the file system’s implementation of processes and/or behaviors (Jones, 2009). Besides the VFS, an additional abstraction layer exists that hides underlying physical devices from users, such as a storage, a disk, or a partition (Jones, 2009).

Linux has file system security model that is follows for ensuring proper file ownerships and access (Both, 2016). File ownership and permission is an important feature in providing protection and security, especially when local access is granted (Mfillpot, 2010). Everything is a file in the Linux operating system. A file contains operations for flags, permission settings, and file history on an individual basis (Jones, 2009). Every file has three user-based permission groups. The first group is the owner permissions, which apply to the owner only (Mfillpot, 2010). In general, other users cannot alter the permission set by the owner (Mfillpot, 2010). The second group is group permissions. Group permissions only apply to users associated with the group assigned to the file (Mfillpot, 2010). The third group is all user permissions. All user permissions apply to any other users on the system (Mfillpot, 2010). The owner needs to handle all user permissions careful due to the high vulnerability is can cause (Mfillpot, 2010). Every file has three basic permission types that related to each group. In general, the owner can add or remove the permissions to any group at any time (Mfillpot, 2010). The permissions types are read, write, and execute. Read refers to the permission read the file. Write refers the permission to write or modify the file (Mfillpot, 2010). Execute refers to the permission to execute a file or file the directory contents (Mfillpot, 2010).

Linux offers simple methods for the owner to establish user names and passwords including for the owner. Many sources have recommended disabling root accounts or remote root access due to higher susceptible in brute force account attacks (Ethand, 2015). Though more challenging to operate, many Linux system support pluggable authentication modules (Linux-PAM, PAM) (Kili, 2018). PAM is large suite of shared libraries enabling owners to choose how applications authenticate users (Kili, 2018). Authentication can be enable that is separate from the standard system features (Kili, 2018).

Linux does not come with built-in anti-virus software. A variety of sources state that few Linux viruses exist in comparison to Windows or MacOS (Memon, 2020). Antivirus software is considered as unnecessary. However, it can still added an additional layer of protection (Memon, 2020). As a side note, users should be aware that Linux servers are more vulnerable the Linux desktops due to their higher level of usage and connectivity (Alu, 2019). For network, program, or system threats, multiple security tools and software specifically for Linux are available online (Alu, 2019).

Firewalls are available with the Linux operating system (Alu, 2019). The Linux kernel has a built-in active firewall (Alu, 2019). A second firewall is available in any Linux distributions (Alu, 2019). However, the additional firewall is usually disabled by default and needs to be activated manually (Alu, 2019).

**Distributed Systems Considerations**

**System Transfer**

A server application can moved between different systems use the Linux system operating system. One important consideration is the operating system itself. Many versions of the Linux operating system exists with some designed for different purposes such as enterprise servers, programming, or personal computers (Silberschatz et al., 2010). Both systems need the capability to host the mutual connection. Any connection may work, but it is advisable to use an operating system best suited for the applications and connection (TestOut Corporation, 2019). Another important consideration is the differences in hardware and firmware. The server application is still running on a Linux operating system. However, the new machine hosting the server may have different hardware, firmware, and software capabilities (TestOut Corporation, 2019). Differences can include memory, CPUs, directory/file layouts, and differences in the command language (Silberschatz et al., 2010). The different system should meet the minimum and/or ideal requirements to host the server application and establish a connection with the system hosting the client application (Jet Global, 2019). The new system needs enough memory, RAM, hard drive space (Jet Global, 2019). Execution and response packages are sent at certain times and need to communicate with the other application (Jet Global, 2019).

**Local Area Network**

A server application may need to move to different system within local area network (LAN). The multiple aspects of network topology should be considered. The server application needs to communicate with the client application. Network topology is describes how devices and nodes are connected, known the physical topology (TestOut Corporation, 2019). It also describes how messages flow within the network, known as the logical topology (TestOut Corporation, 2019). The two primary groups are the full-connected network and partially connected network (Silberschatz et al., 2010). In a fully connected network, the new location of the server should only have a little to no impact. In a partially connected network, moving the server application is an important consideration. Moving the server without reviewing the topology may result in communication failures if a section(s) in the network malfunctions (TestOut Corporation, 2019). As a result, the client and server applications may not properly connect, deliver, update, or close (Silberschatz et al., 2010). In a partially connected network, the client and server applications alter the time it takes to communicate. The messages may exchange faster or slower depending on the new server location (Silberschatz et al., 2010).

**Cloud Distribution**

Server application may be moved to a different system located within a cloud. Performance is an important consideration. Clouds are sometimes known for slower performance levels and lagging issues. Client-server application may occur at a slower rate and not always update as originally intended. Bandwidth, network usage, and range are some of the many factors that may result in a faulty communication or a low performance level (Dobran, 2018). A server and client application needs to be able to handle the variations in communication. Another consideration is the compatibility of client applications (Hester, 2015). A client application may require access and messaging through a local port (Hester, 2015).. The client application may not have an application version allows desired procedures to operate through a cloud network (Hester, 2015). The application may need to be rerouted or reprogrammed to work with the cloud server application. A third consideration is the type of cloud network. A cloud network or server can simply be described as a virtualized hosting platform (Dobran, 2018). Public and private clouds are two other types of clouds, but relates security differences (Hester, 2015). The types of clouds to examine are the full cloud and the hybrid cloud. With a full cloud, all applications and data are hosted completely in the cloud (Hester, 2015). The server application needs to be uploaded to the cloud also. The server and client applications will communicate through virtual ports, which may result in different setups and security measures (Dobran, 2018). A hybrid cloud uses a mix of both physical servers and cloud networking (Dobran, 2018). The network is partially clouded, so it may be hard to decide on the best location for the server application in comparsion to the client application (Hester, 2015). Factors to consider are the future/current location of client application(s), reliability and the new design of the network topology (Hester, 2015). Also, in general, clouds may require downtime for updates or other matters (Hester, 2015). The server and client application may need to communicate within certain time frames.

**Server Applications**

The advantages are a wider network, higher performance, and better compatibility. Because the networking, the client-server application is no longer exclusive to only the original system. It may take longer, but is now accessible at another location (Silberschatz et al., 2010). The application is less likely to have compatibility issues. Ideally, moving the server application to another Linux system should only require minimal adjustments when comparing to moving to a completely different operating system (TestOut Corporation, 2019). In addition, the applications may perform better on independent systems. By placing the applications at separate locations, the applications may functions better. The client and server applications are using the hardware and firmware local to the system (Silberschatz et al., 2010).

The advantages of system transfer are accessibility, data security and transferability. With the proper internet connected and permissions, cloud-based applications are accessible from any location (IBM, 2019). Clouds are updated frequency, applying better and more-efficient technology to the system (IBM, 2019). Because the server application is on a virtual network, data is not necessarily lost during unforeseen delays or hardware issues (IBM, 2019). Many applications are cloud-capable (IBM, 2019). Both the client application and server application can be loaded into the cloud.

The advantages of LAN distribution are better transfer rates and a reduction of unexpected errors. With a fully connected LAN, the client-server applications have point-to-point access within a similar range (Silberschatz et al., 2010). Within a partially connected LAN, the client and server can be better locations in the network topology. The messages can exchange point-to-point or between a smaller number of nodes/devices (Silberschatz et al., 2010). If the server was moved outside of the LAN, the client-server applications may experience slower transfer rates and connection issues (TestOut Corporation, 2019). A LAN is adaptable in the moving the server to different locations will have a minimal amount of negative effects on the applications compared to moving the server outside of the LAN.

**Linux Program Development Considerations**

**Integrated Development Environments**

An Integrated Development Environment (IDE) or programming text editor is highly beneficial to a programmer. An IDE combines the important aspects of computer programming into one coding application (Codecademy, 2019). Text editors can summarized as a text editing system that may or may not contain IDE features. (Matthews, 2018). The terms of IDE and text editor are used interchangeably in the programming community.

Instead of just one IDE, Eclipse provides a variety of Linux-compatible IDEs (Eclipse Foundation, 2020). A programmer can select the IDE that best suits their needs. Some of available IDEs (and packages) include desktop, cloud, platforms enterprise, web, modeling, testers and more (Eclipse Foundation, 2020; Stackify, 2017). Most, if not all, IDE are compatible with Linux operating systems. (Eclipse Foundation, 2020). Eclipse is designed for Java development, but supports numerous plug-ins and packages for other languages (Sohail, 2019). Some notable languages that is can support are Java, Ruby, Python, C, C++, and more (Sohail, 2019). Eclipse also aims to keep their IDEs up to date with the developments in modern technology (Eclipse Foundation, 2020).

Beside Eclipse, another program to utilize for computer programming is Geany. Geany is a free text editor with many essential IDE features (Geany, 2020). It is designed to be a fast, versatile, and lightweight with a low dependency on other packages (Geany, 2020; Stackify, 2017). Geany has a wide range of features and plug-ins including auto-completion, code navigation, syntax highlighting, hints/call tips, and code folding (Troger et al., 2019; Stackify, 2017). It currently supports for over 50 programming languages, including Java, C, and Python (Geany, 2020; Stackify, 2017). Programmers can also “hack” and customize it to what suites their needs and/or preferences (Geany, 2020). Geany does not directly support compiling or interpretation, but can send external commands to execute the build system (Troger et al., 2019).

**Programming Languages**

Many languages are available to use on Linux operating systems. The language selected for building a program may vary based on the focus, level, popularity, and other criterion. C is considered a primary language in Linux operating systems (Anderson, 2015). In fact, Linux’s kernel is commonly programmed in C (Kili, 2016). Although it is a general-purpose language, C is mostly associated with system-level programming (Laura, 2020; Techopedia Inc, n.d.). It is categorized as a low-level procedural language (Laura, 2020). A few key features are that C high-speed and compiled language that uses less main memory than other languages (Munoz, 2015). C provides the capabilities of manual memory management, dynamic memory allocation functions, and pointer arithmetic (Munoz, 2015; Vartika02, 2019). However, a programmer needs to pay close attention to potential memory problems or defects (Laura, 2020).

The programming languages to consider for general usage are Python, Go, and Java. Python is a high-level open-source language (Python Software Foundation, 2019). Python is a versatile general-purpose language used in many technological areas (Jamsheer, 2020). Its advantages include object-orientation, large standard libraries, an unobstructed syntax, integration features, and programmer comprehension (Mindfire Solutions, 2017; Malik, 2019). Some of the disadvantages of Python are design restrictions and high memory consumption (Jamsheer, 2020). Python programs also execute at a slower speed because it is an interpreted language (Kili, 2016). Go is a newer open-source compiled language with the combination of C’s core features and newer modern-day features (Built In, 2019; Go, 2020). Go offers the benefits of “garbage collection” (automatic memory management), modernized object-oriented programming, structural programming, and focuses on efficient concurrency (Hasan, 2018; Patel, 2019; Techopedia Inc, 2017). On the negative side, Go is it still developing and has a smaller community in comparison to more established languages (Oamkumar. 2018). Another language to review is Java. Java is resourceful compiled language (Hasan, 2018). Java has features of object-orientation, distributed computing, memory allocation, and multithreading (MindsMapped Admin, 2019). For Linux operating system, it may be important to note that Java does not use pointers and may be less efficient in memory management due to its coding structure (MindsMapped Admin, 2019).

**System Resources and Program Size**

System resources are computer components controlled and assigned by the operating system (Fisher, 2019). Computers have four primary system resources: interrupt request (IRQ) lines, I/O ports, direct memory access (DMA) channels, and memory address ranges (Fisher, 2019). The usage and assignment of system resources allows the computer to communication between hardware and other computer components. Without the communication, the computer would not be able to run and perform tasks. A few simple examples are software execution or save files to a new location (Fisher, 2019). However, system resources have limitations.

In development, a programmer needs to rationalize the usage of system resources and the size of the program. Memory and system resources have limitations (Fisher, 2019). A programmer should consider the how the program operates (Rahman, 2018). A web browser requires a network connection. An internal word processor does not require a network, but needs the read, write, and/or execute the text files within the Linux system. Another consideration is the external interfaces that the program needs (Rahman, 2018). Common interfaces are user, hardware, software, and communication (Rahman, 2018). A standard user program needs user I/O, user-triggered interrupt requests, but a hardware program may not need any user interfaces. Next, the programming language is consideration in development (FreeCodeCamp, 2020). With compiled languages, a programmer has more control over the hardware aspects (FreeCodeCamp, 2020). Interpreted languages tend to be smaller in size, but execute slower (FreeCodeCamp, 2020). A programmer needs to consider the program’s ideal quality characteristics. One program ideally needs high process power while another program may need low memory consumption. A final consideration is the system requirements/available resources (Fisher, 2019). If the requirements cannot be determined, a programmer may unintentionally build a program that is incompatible with their system. If the program can estimate the requirements or has access to current system information, a programmer can build the program within the operating system’s limits.

References

Alu. (2019, February 9). *Do I need a Firewall in Linux?* Average Linux User. https://averagelinuxuser.com/linux-firewall/

Anderson, A. (2015, December 7). *The best programming languages for Linux lovers*. Kernel Mastery. https://kernelmastery.com/the-best-programming-languages-for-linux-lovers/

Anne, S. (2016, February 10). *File types In Linux/Unix explained in detail.* Linux. https://www.linux.com/training-tutorials/file-types-linuxunix-explained-detail/#

Bisht, A. (2018, December 30). *Logical and Physical Address in Operating System*. GeeksforGeeks. https://www.geeksforgeeks.org/logical-and-physical-address-in-operating-system/

Both, D. (2016, October 31). *An introduction to Linux file systems*. Opensource. https://opensource.com/life/16/10/introduction-linux-file systems

Built In. (2019, October 15). *Why Go? 8 engineers discuss Golang’s advantages & how they use it*. Built In. https://builtin.com/software-engineering-perspectives/golang-advantages

Buzdar, K. (2018, June 20). *Linux File Management from the Terminal*. VITUX. https://vitux.com/file-management-from-the-linux-terminal/

Christensson, P. (2019, December 20). *Inode Definition*. TechTerms. https://techterms.com/definition/inode

Codecademy. (2019). *What is an IDE?* Codecademy. https://www.codecademy.com/articles/what-is-an-ide

Dobran, B. (2018). *Cloud vs dedicated server: Which is best for your business?* PhoenixNAP Global IT Services. https://phoenixnap.com/blog/cloud-vs-dedicated-server

Duarte, G. (2009, January 27). *Anatomy of a Program in Memory*. Many But Finite. https://manybutfinite.com/post/anatomy-of-a-program-in-memory/

Ebrahim, M. (2018, June 22). *Linux Virtual File System*. Like Geeks. https://likegeeks.com/linux-virtual-file-system/

Eclipse Foundation. (2020). *Eclipse Downloads | The Eclipse Foundation*. Eclipse. https://www.eclipse.org/downloads/

Educba. (2019, October 10). *Linux process management*. EDUCBA. https://www.educba.com/linux-process-management/

Ellingwood, J. (2014, November 14). *Basic Linux Navigation and File Management*. DigitalOcean. https://www.digitalocean.com/community/tutorials/basic-linux-navigation-and-file-management

Ethand. (2015, February 2). *Linux Security Best Practices | IONOS DevOps Central*. Ionos. https://devops.ionos.com/tutorials/linux-security-best-practices-1/#ssh-key-authentication

Fisher, T. (2019a). *What Are a Computer’s System Resources and Why Do They Run Low?* Lifewire. https://www.lifewire.com/what-is-a-system-resource-2626016

Fisher, T. (2019b, November 3). *What are a computer’s system resources and why do they run low?* Lifewire. https://www.lifewire.com/what-is-a-system-resource-2626016

Franklin, C., & Coustan, D. (2000, August 14). *How operating systems work*. HowStuffWorks. https://computer.howstuffworks.com/operating-system5.htm

FreeCodeCamp. (2020, January 10). *Interpreted vs compiled programming languages: What’s the difference?* FreeCodeCamp. https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/

Garrison, J. (2017, July 10). *Which Linux File System Should You Use?* How-\ To Geek. https://www.howtogeek.com/howto/33552/htg-explains-which-linux-file-system-should-you-choose/

Geany. (2020). *Geany - the flyweight IDE*. Geany. https://www.geany.org/

Go. (2020). *The Go Programming Language*. Go. https://golang.org/

GoLinuxCloud. (2020, March 24). *Tutorial: Beginners guide on linux memory management*. GoLinuxCloud. https://www.golinuxcloud.com/tutorial-linux-memory-management-overview/

Gupta, A. (2018, February 6). *Linux OS: Demand Paging, Page Faults and Working Set*. Medium. https://medium.com/software-under-the-hood/under-the-hood-os-demand-paging-page-faults-and-working-set-82849bb6b404

Guru99. (n.d.-a). *Memory Management in OS*. Guru99. Retrieved June 14, 2020, from https://www.guru99.com/os-memory-management.html

Guru99. (n.d.-b). *Virtual Memory in OS: What is, Demand Paging, Advantages*. Guru99. Retrieved June 14, 2020, from https://www.guru99.com/virtual-memory-in-operating-system.html

Hasan, M. (2018, November 12). *Top 20 most popular programming languages to learn for your open-source project*. UbuntuPIT. https://www.ubuntupit.com/top-20-most-popular-programming-languages-to-learn-for-your-open-source-project/

Hass, J. (2019, December 8). *Background and Foreground Process in Linux*. Lifewire. https://www.lifewire.com/multitasking-background-foreground-process-2180219

Hester, M. (2015). *5 critical facts every business must know before moving their network to the cloud*. LAN Sytems. https://www.lansystems.com/wp-content/uploads/2015/05/5-Critical-Facts-You-Must-Know-Before-Moving-Your-Computer-Network-To-The-Cloud-by-LAN-Systems.pdf

IBM. (2019). *Benefits of cloud computing*. IBM. https://www.ibm.com/cloud/learn/benefits-of-cloud-computing

Ilyas. (2019, January 26). *LINUX Process Management*. Medium. https://medium.com/@ilyash/linux-process-management-eef45f3c198e

Jamsheer. (2020, April 21). *Top advantages and disadvantages of Python: A 2020 guide*. Acodez. https://acodez.in/advantages-and-disadvantages-of-python/

Jet Global. (2019, January 31). *Server Hardware Sizing Recommendations*. Jet Global Insights. https://support.jetglobal.com/hc/en-us/articles/219401657-Server-Hardware-Sizing-Recommendations

Jones, M. (2008, December 20). *Anatomy of Linux process management*. IBM Developer. https://developer.ibm.com/technologies/linux/tutorials/l-linux-process-management/

Jones, M. (2009, July 31). *Anatomy of the Linux virtual file system switch*. IBM Developer. https://developer.ibm.com/technologies/linux/tutorials/l-virtual-file system-switch/

Kernel Development Community. (n.d.). *Concepts overview — The Linux Kernel documentation*. Kernel Development Community. Retrieved June 14, 2020, from https://www.kernel.org/doc/html/latest/admin-guide/mm/concepts.html

Kili, Aaron. (2016, May 26). *Top 5 programming languages for developing Linux desktop applications*. FOSSMint. https://www.fossmint.com/best-programming-languages-for-developing-linux-desktop-applications/

Kili, Aaron. (2017, April 5). *All You Need To Know About Processes in Linux*. Tecmint. https://www.tecmint.com/linux-process-management/

Kili, Anon. (2016, July 15). *Explanation of “Everything is a File” and Types of Files in Linux*. Tecmint. https://www.tecmint.com/explanation-of-everything-is-a-file-and-types-of-files-in-linux/

Kili, Anon. (2018, December 11). *How to Configure and Use PAM in Linux*. Tecmint. https://www.tecmint.com/configure-pam-in-centos-ubuntu-linux/

Koutoupis, P. (2018, August 21). *Everything You Need to Know about Linux Containers, Part I: Linux Control Groups and Process Isolation | Linux Journal*. Linux Journal. https://www.linuxjournal.com/content/everything-you-need-know-about-linux-containers-part-i-linux-control-groups-and-process

Laura. (2020, January 23). *C vs C++ comparison: Find out the difference between C and C++*. BitDegree Tutorials. https://www.bitdegree.org/tutorials/c-vs-c-plus-plus/#C

Linux Foundation Group. (2015, March 19). *File system Hierarchy Standard*. Refspecs.Linuxfoundation.Org. https://refspecs.linuxfoundation.org/FHS\_3.0/fhs/index.html

LiveFire Labs. (n.d.). *UNIX Process Management - Part II (parent-child, fork-and-exec, daemons)*. LiveFire Labs. Retrieved June 14, 2020, from https://www.livefirelabs.com/unix\_tip\_trick\_shell\_script/unix\_system\_administration/unix-process-management-part-2-parent-child-fork-exec-daemons.htm

Malik, U. (2019, July 16). *Advantages and disadvantages of the Python programming language*. Vertabelo Academy. https://academy.vertabelo.com/blog/python-programming-advantages-disadvantages/

Matthews, K. (2018, September 4). *Which are better for developers, IDEs or text editors?* JAXenter. https://jaxenter.com/ides-vs-text-editor-148936.html

McKay, D. (2020, January 17). *Everything You Ever Wanted to Know About inodes on Linux*. How To Geek. https://www.howtogeek.com/465350/everything-you-ever-wanted-to-know-about-inodes-on-linux/

Memon, Z. (2020, February). *Why You Still Don’t Need Antivirus Software on Linux in 2020*. Linux Hint. https://linuxhint.com/why\_no\_antivirus\_linux/

Mfillpot. (2010, May 18). *Understanding Linux File Permissions*. Linux.Com. https://www.linux.com/training-tutorials/understanding-linux-file-permissions/

Mindfire Solutions. (2017, April 24). *Advantages and disadvantages of Python programming language*. Medium. https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121

MindsMapped Admin. (2019, February 18). *Java advantages and disadvantages*. MindsMapped. https://www.mindsmapped.com/java-advantages-and-disadvantages/

Munoz, D. (2015). *After all these years, the world is still powered by C programming*. Toptal LLC. https://www.toptal.com/c/after-all-these-years-the-world-is-still-powered-by-c-programming

Oamkumar, R. (2018, June 11). *Advantages and disadvantages of Golang (Go)*. Software Developer India. https://www.software-developer-india.com/advantages-and-disadvantages-of-golang-go/

Oracle. (2020). *Oracle Linux 7: Administrator’s Guide*. Oracle. https://docs.oracle.com/en/operating-systems/oracle-linux/7/admin/ol7-monitune.html

Ott, A. (2016). *Virtual Memory and Linux*. Konsulko Group. https://elinux.org/images/b/b0/Introduction\_to\_Memory\_Management\_in\_Linux.pdf

Patel, K. (2019, May 7). *Why should you learn Go?* Medium. https://medium.com/@kevalpatel2106/why-should-you-learn-go-f607681fad65

Python Software Foundation. (2019). *Welcome to Python.org*. Python. https://www.python.org/about/

Rahman, A. (2018, February 20). *The Importance Of Software Requirements Specification | Nascenia*. Nascenia. https://www.nascenia.com/the-importance-of-software-requirements-specification/

RedHat. (2020). *What is Linux?* Redhat.Com. https://www.redhat.com/en/topics/linux/what-is-linux

Sharma, G. (2017, June 12). *PID Allocation in Linux Kernel*. Medium. https://medium.com/@gargi\_sharma/pid-allocation-in-linux-kernel-dc0c78d14e77

Silberschatz, A., Galvin, P. B., & Gagne, G. (2010). *Operating system concepts with Java* (8th ed.). John Wiley & Sons. https://edugen.wileyplus.com/edugen/lti/main.uni

Sohail. (2016, August 21). *The Linux File System Structure Explained*. LinuxAndUbuntu. http://www.linuxandubuntu.com/home/the-linux-file-system-structure-explained

Sohail. (2019, December 7). *8 best IDEs or code editors For Linux*. LinuxAndUbuntu. http://www.linuxandubuntu.com/home/8-best-ides-or-code-editors-for-linux

Stackify. (2017, March 2). *Top integrated developer environments (IDEs)*. Stackify. https://stackify.com/top-integrated-developer-environments-ides/

StudyTonight. (2019). *Introduction to Memory Management in Operating System | Studytonight*. Studytonight. https://www.studytonight.com/operating-system/memory-management

Taylor, T. (2019, May 8). *Cloud networking: Critical decisions when migrating from on-premises*. TechGenix. http://techgenix.com/cloud-networking/

Techopedia Inc. (n.d.). *What is system programming? - Definition from Techopedia*. Techopedia. https://www.techopedia.com/definition/9616/system-programming

Techopedia Inc. (2017, February 7). *What is Garbage Collection (GC)? - Definition from Techopedia*. Techopedia. https://www.techopedia.com/definition/1083/garbage-collection-gc-general-programming

TestOut Corporation. (2019). *TestOut PC Pro* (6.0.6). TestOut. https://cdn.testout.com/client-v5-1-10-620/startlabsim.html?ls6link=true

Troger, E., Treleaven, N., Lanitz, F., Wendling, C., & Brush, M. (2019, April 28). *Geany: Manual [Version 1.35]*. Www.Geany.Org. https://www.geany.org/manual/current/

Tutorialspoint. (n.d.). *Operating System - Processes - Tutorialspoint*. Tutorialspoint. Retrieved June 13, 2020, from https://www.tutorialspoint.com/operating\_system/os\_processes.htm

Vartika02. (2019, January 8). *Difference between C and C++* (S. Kumar (ed.)). GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-c-and-c/

Weeks, A. (2004). *What is virtual memory?* Www.Tldp.Org. https://www.tldp.org/LDP/sag/html/vm-intro.html#:~:text=Linux%20supports%20virtual%20memory%2C%20that

Wong, D. (2019, May 8). *Pathname lookup*. Kernel Foundation. https://www.kernel.org/doc/html/latest/file systems/path-lookup.html

Zand, M. (2019, September 16). *An intro to Linux file system management*. DesigSpark. https://www.rs-online.com/designspark/an-intro-to-linux-file-system-management

‌