

Drawn It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 9/21/25 | Cristiano Miranda | Update the various sections of the document using information gathered from clients, The Gaming Room. |
| 2.0 | 10/5/25 | Cristiano Miranda | Updated the evaluation section of the document. |
| 3.0 | 10/14/25 | Cristiano  Miranda | Update the recommendation section of the document. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room requires a web-based application that can run on any web browser, allowing for seamless play across multiple platforms. The game will utilize a multi-team-based system in which multiple players can join a single team. However, each game, each team within the game, and each player within a team must have unique names associated within their respective group. Only a singular instance of the main application must be running on the server, but multiple game instances must be active within the singular instance.

## Requirements

The Gaming Room requires a system that can be connected across multiple devices and continuously serve an interactive web application to users, facilitating interaction between multiple users without significant delay. The system must be performant enough to handle sending multiple images to multiple users without significant delay over the internet, and it needs to be secure while still allowing any user to access the web application it hosts. The web application itself must be supported on multiple web browsers across various devices, so multiple programming languages, a stylesheet language, and a markup language will need to be used to create the project. Additionally, the game will need indefinite access to a stock image library to utilize as clues in the game.

## [Design Constraints](#_2et92p0)

* Client requires a system to function as a server to continuously host their web application by sending and receiving data concurrently from multiple users without noticeable delay.
* Client requests scalability be available for the web-application and the server
* Client needs to provide access to the web application to unknown users
* Client requests that their web application be supported on a variety of web browsers
* Client needs to use HTML, CSS, and JavaScript to develop the front end and an additional programming language to develop the back end
* Client requires continuous access to a stock image library

## [System Architecture View](#_ilbxbyevv6b6)

Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

## [Domain Model](#_8h2ehzxfam4o)

The UML class diagram for this project represents the various classes that encapsulate specific logic related to certain elements of the project. To start, the *Entity* class houses multiple shared attributes and methods that are required for the other classes, such as the *Game*, *Team*, and *Player*, to function. The program utilizes the OOP principle of inheritance to allow the *Game*, *Team*, and *Player* classes to inherit these attributes and methods from the *Entity* class to avoid the need to rewrite the code in all classes, allowing the code to be less repetitive and removing the potential for errors to occur from having to manage the same code in multiple places. Additionally, the *Game*, *Team*, *Player*, and *GameService* classes are all involved in a zero-to-many relationship. The *GameService* class allows either zero or more *Game* objects to be instantiated, each *Game* object allows either zero or more *Team* objects to be instantiated, and each *Team* object allows zero or more *Player* objects to be instantiated. Through this, each of the classes that instantiate the objects is allowed to instantiate as many objects as they need to enable them to adapt to the current situation without having to program each class to handle specific variations for any situation that may arise. For instance, if three games need to be created, the *GameService* class can instantiate *Game* objects until three are created, and from here, each *Game* object can instantiate as many *Team* objects as needed. The same principle holds with the *Team* objects instantiating the *Player* objects. This implementation highlights the OOP programming principle of encapsulation by containing the various methods and attributes needed to create the different kinds of objects within their own classes. Rather than having to house all methods within the same class, and thus decreasing the readability of the code and making it harder to navigate the singular class, the methods and attributes related to each element are housed within their own class and called upon or accessed when a new object needs to be created. Additionally, this also highlights the OOP principle of abstraction by removing the need anyone working on project to be aware of how the other classes function internally. Instead, someone needs only to be familiar with the public methods and attributes they need to use to implement the class into the project effectively. Finally, the two other classes, *ProgramDriver* and *SingletonTester*, are related via the *ProgramDriver* class having a dependency on the *SingletonTester* class. Based on this dependency, the *ProgramDriver* class requires the *SingletonTester* class to operate and function within the program properly. Again, this relationship demonstrates the OOP principle of abstraction, as anyone working on the program in the future does not need to fully understand how the *SingletonTester* class functions to use the *ProgramDriver* class effectively. They would have to load it when using the *ProgramDriver* class, but do not need to be aware of how it is programmed or how it functions.

                Through these various relationships, the program can effectively run multiple games with multiple teams and multiple players without experiencing any interference, as every *Game* object will be used to store its associated *Team* objects, and each *Team* will store its associated *Player* objects. Additionally, these *Game* objects will be stored and managed within the *GameService* object that is only instantiated a single time, to ensure that any access to said object is consistent throughout the entire program. The *ProgramDriver* object will be used to instantiate *the GameService* object. From here, this singular object will be used to instantiate the various *Game* objects and store them within the list found within the *GameService* object. This approach allows for a more straightforward method to run multiple games simultaneously, while still making it easy to manage, maintain, and eventually remove these *Game* objects without disrupting other *Game* objects or closing the program.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client’s requirements and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

| **Development Requirements** | **Mac** | **Linux** | | **Windows** | **Mobile Devices** | |
| --- | --- | --- | --- | --- | --- | --- |
| **Server Side** | As of 2022, Mac no longer has an official server operating system. Instead, most of the server features are now integrated into macOS. It has multiple applications that can be used to host web applications, and some of them are open-source and free to utilize. MacOS is also known for its security and has fewer exploits than most other operating systems. Apple owns macOS, and thus, any updates, improvements, and fixes for the operating system are slower to release, as a single company works on them. However, hardware compatibility is strict and requires the use of proprietary systems, and these systems are no longer upgradable. Basic systems can be around $600 while their highest and most upgraded models can be $1000, and the increase price is not usually associated with increased performance. Thus, it isn't easy to scale as the hardware would become a limiting factor. Still, some third-party providers offer macOS hosting, and they provide tiers which can give the opportunity to upgrade later on. These plans range from a pay-as-you-go starting at $4 a day to $99 a month to run the highest end hardware at the current moment. | Linux is best known for being largely customizable through providing multiple options through its many distros, or a precompiled Linux operating system. Additionally, Linux houses many open-source and free web-hosting options across the various distros. It is also compatible with most hardware, allowing for easy upgrades and future scalability. Linux is also known for being secure, and most security practices are incorporated by default within the OS. Linux is also an open-source operating system, and thus, multiple communities are involved in providing updates, improvements, and fixes, allowing changes to be released quickly and issues to be fixed sooner. Numerous third parties also offer Linux-based web-application hosting, providing access to a support team in cases where problems arise. These plans vary greatly in price, with the most basic ones costing around $5 a month and more complex and powerful systems being priced at nearly $1000 a month. One caveat is that most Linux operating systems have a steep learning curve, and by default, most are GUI-less, so managing and working on them can be difficult for beginners. | | Windows offers a wide range of web hosting applications, many of which are open-source and free to use. Using Windows to host a web application also provides the benefit of allowing the web application access to various Microsoft-owned services, like the .NET Framework and MS SQL, which can be used in the development of the web application and for storing data generated by the web application's users. Still, Windows is known to have more exploits than other operating systems. Microsoft also owns Windows, and thus, updates, improvements, and fixes are slower to release because they all originate from a single company. Additionally, multiple third parties offer Windows as an option for running a server to host the web application. The price of these server hosting plans ranges from around $5 a month for basic and low-end hardware to potentially $400 a month for the most powerful hardware. In terms of hardware, Windows is compatible with a majority of hardware, so upgradability is possible. | Mobile devices can host a simple web application. Still, they are often not used, as the hardware in mobile devices is lower performing since these devices focus on portability. The Android operating system would be the leading choice for hosting a web application, as it provides access to multiple hosting applications. Still, most mobile devices utilize a wireless connection to access the internet, making it a potentially limiting factor while hosting the server. Additionally, most mobile devices are not upgradable, and thus, scalability will be greatly limited. | |
| **Client Side** | MacOS offers its own native IDE for developing applications for MacOS. However, in this case, third-party options will have to be used to allow development for cross-platform support. Still, Mac is mainly focused on creating programs for macOS, so while there are third-party options for developing cross-platform applications, they are not supported by default. Hardware is another limiting factor, as macOS is tied to specific hardware that is usually not upgradable. In terms of hosting the web-application to MacOS users, MacOS's default browser, Safari, supports a majority of features that most other web browsers support as well, so no additional considerations will be needed to allow MacOS users to access the web-application. | Linux houses support for most programming languages, and its open-source nature has created multiple communities that create and develop numerous options for designing and maintaining web applications. Still, Linux is more complicated to set up and get functioning, making it less beginner-friendly. Linux supports running on a majority of hardware, and it is known to require fewer resources to run, leaving more computing power available for developing applications. Linux supports multiple web browsers, so no additional considerations will be needed for Linux OS users to access the web application. | | Windows is widely used due to the plethora of applications available for designing, maintaining, and creating web applications and applications in general. Many of these tools are created and maintained by Microsoft itself, but there are also multiple third-party options available. In terms of hardware, Windows is supported by a majority of available hardware, so accessing a computer system capable of running and developing with Windows is straightforward. Still, even though Windows is the easiest operating system to use, many of the Microsoft-owned tools require the purchase of licenses to utilize. Windows provides support for multiple web browsers; however, Windows by default houses an older web browser called Internet Explorer, and the features available to it are minimal. Considerations will need to be taken for users who attempt to access the web application using this web browsers. | While it is possible to develop applications on mobile devices, it is not practical due to the limited user interface and its significant reliance on touch screens, as well as the usual lack of a physical keyboard. Additionally, several tools are available for developing applications, but many of these are third-party and often lack essential features. Hardware is also limited and often underperforming due to the mobile device's compact nature and its reliance on battery power. However, considerations will need to be made as mobile devices utilize touch screens for user input, and their screen sizes are often smaller and more compact than those on laptop or desktop systems. The web application will need to scale its size to fit onto the screens of these smaller devices while still being easy to read and interact with to avoid limiting the user experience. | |
| **Development Tools** | MacOS offers a native IDE called Xcode, which is primarily used to develop macOS-only applications, and its costs around $99 per year. Still, a different IDE would be necessary for this program as an IDE that supports web development is required. Thus, other third-party IDEs are available for creating cross-platform applications, and many of them are free and open source, such as Visual Studio Code. Additionally, macOS supports multiple programming languages, and installing additional tools can further increase this number. No additional team would be required to develop the web application specifically for macOS as its native browser, Safari, is compatible with most web-features. | Multiple open-source and free-to-use IDEs are available for use on Linux; thus, no fee would be required to develop on Linux OS. Additionally, Linux offers various tools that aid in application development and support most programming languages by default. However, installing these tools and the IDE will take some effort due to Linux's learning curve. No extra focus would be required to develop the web application specifically for Linux OS users as most of the web browsers available on it are compatible with most web-features. | Windows offers multiple IDEs, such as Visual Studio Professional and Visual Studio Enterprise, and provides some free-to-use IDEs, such as Visual Studio Code and Visual Studio Community. Still, for Visual Studio, a Microsoft-owned product, teams larger than five people or companies that generate a large amount of revenue are required to purchase a license to use their industry-grade IDE. The cost of this license is around $99 a month for the professional edition and around $500 a month for the enterprise edition. An additional team might be required to develop the web-application to account for users utilizing the old Microsoft web browser internet explorer. While it does support some web-features, a majority are incompatible and could cause issues if not account for. | | There are a few IDEs available for mobile devices, but one of the best known is AIDE, or Android IDE which is free to use. However, it is greatly limited and primarily focuses on teaching people how to code. There is a significant lack of programming tools for mobile devices, as they are limited in terms of user input and hardware performance. However, the most notable aspect of mobile devices is that they will require a team dedicated to developing the web-application to function properly for the plethora of screen sizes and different device types available. While the main fundamentals of the various devices will function the same, such as user inputs and interaction with the web page, the different resolutions for the various screens can become a potential problem if not account for. To help ensure that the user experience is decent across different mobile devices, a team should be responsible for adjusting the web-application to behave differently by scaling specific aspects of the web page to best adjust to the screen size of the user’s device, such as enlarging buttons or decreasing text size to avoid it wrapping or extending out of frame. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

1. **Operating Platform**:

I recommend utilizing the Linux operating system. While in this case, development and hosting of an online web-application can be performed on any of the three major operating systems: Windows, macOS, or Linux, the various advantages of Linux, such as the significant focus on community-driven and open-source development and the plethora of well-priced third-party hosts, make it the best option for this scenario. In this scenario, the primary focus is to enable multiple games to run concurrently, allowing various users, each with different computer systems, to connect to and join any of the currently running games or even create their own. Powerful hardware will be required to ensure that all these games have the necessary resources to run concurrently, and that the network is not overloaded by the number of simultaneous users connecting to the server. By utilizing third-party hosts, these two issues will no longer be the client's responsibility. If any bottleneck occurs from the physical hardware, rather than having to handle upgrading the hardware, the client can pay to increase their server's available resources. Additionally, the upkeep of the server's physical hardware will be delegated to the other company, allowing the client to focus solely on web-application development and not server maintenance. Finally, knowing that multiple users will connect to the server and sensitive user data will be stored there makes it essential that security is at the forefront of development. Linux has the fewest known security vulnerabilities. Even before installing and implementing additional security measures, the operating system itself will help reduce the risk of unauthorized users gaining access to or compromising the server's security.

1. **Operating Systems Architectures**:

Linux's support for a wide range of hardware makes it compatible with multiple operating system architectures, such as ARM, x86, and x86-64. Compatibility with these various architectures enables the server to utilize different types of hardware if needed, making it easier to find powerful and affordable options. Additionally, major programs and tools that are compatible with other operating systems, such as Windows or macOS, are also compatible with Linux. In terms of the operating system's software, Linux uses a layered system in which different layers act as interfaces to one another. Like with most operating systems, the innermost layer is the physical hardware of the system, and to allow users to connect with and utilize it, multiple outer layers are required to build this connection. In Linux's case, the layer above the hardware is called a kernel, which was designed independently from other operating systems and is named the Linux Kernel. This kernel is the central element of the Linux Operating System that handles various essential operations, such as controlling CPU processes and managing memory. From here, above the kernel is called the system library. As Abraham Silberchatz et al. write, the system libraries contain a "standard set of functions through which applications can interact with the kernel" (2008). Rather than requiring users to interact with harder-to-understand processes directly, these libraries allow users to perform desired actions without directly accessing the kernel. This method helps create abstraction by enabling tasks to be performed without the user fully understanding how a system library interacts with the kernel. Plus, because Linux is open source and free to use, any customization, whether to the kernel layer or the outer layers, allows the operating system to be explicitly tailored to any task. In this case, since the client requires a server that will primarily handle multiple user connections and process data, we can allocate more resources to these processes by choosing a Linux operating system that is barebones and utilizes the command line interface, thereby avoiding the waste of resources on loading a GUI or other unwanted applications.

1. **Storage Management**:

In the context of this web application, data storage will primarily be limited to the server, with little to no data stored on client systems. Still, most web browsers allow for storing data locally, known as local storage. A string can be stored and accessed with a key value, meeting the requirements, as it can store basic data on client systems if necessary. In contrast, the server will be responsible for storing user data and the various files for the web application itself. While the web-application files should be quick to access since they will be used at nearly any point the server is active, accessing user data presents a challenge because only certain instances will require specific user data. It is unknown how many users will access the application and store data on the server. Thus, for user data specifically, the storage method must be quick to access without requiring data to be stored in server memory, as memory will primarily be used for web application data. Additionally, the storage medium must be scalable if the current storage capacity begins to run low. The best option to solve this issue would be to utilize a Database, as it allows for quick data access via indexing and can scale to increase storage availability without compromising the currently stored data. In terms of the database that should be used, PostgreSQL is the ideal database management system because it supports multiple data types commonly used in web applications, such as JSON and XML.

1. **Memory Management**:

Linux's memory management is highly advanced and effective, ensuring that the appropriate amount of memory is allocated to software and deallocated when no longer needed. The ability to deallocate or move unused memory to storage helps avoid performance problems and resource wastage from unnecessary processing. For instance, while most web-application files need to be loaded into memory to ensure the system responds quickly and is not halted by accessing data from storage, some files or processes, such as some of the clue images, are not always active or required. If a clue image is no longer needed, the application can instruct the operating system to remove it from memory, freeing up space for the next clue and ensuring that no memory is wasted on unused or unnecessary data. Linux also utilizes virtual memory to enable multiple concurrent processes by determining which processes are currently inactive and can be moved from memory. This process, called demand paging, allows currently running applications to access the now free physical memory. As Abraham Silberschatz et al. write, by utilizing demand-paged virtual memory, only certain "pages are loaded when they are demanded during program execution…" (2008). Rather than having to limit the number of processes running, the server can continue to run multiple processes by simply halting those that are inactive in favor of active ones. This ability will be particularly beneficial if the number of games starts to exceed the space available in physical memory. Still, while this will be slower to run and potentially slow down the games as they are swapped in and out of memory, it prevents the application from halting any game until memory is freed, either by users leaving the application or by games ending, both of which are not ideal. Rather than losing players due to insufficient physical memory, games would continue running, albeit more slowly, allowing users to continue playing.

1. **Distributed Systems and Networks**:

Linux relies on the TCP/IP connection protocols, which are multiple protocols that work in tandem to pass data over a network to other devices. They utilize various layers to handle different aspects of data transfer, such as security, communication, and transport. The main benefit of using this family of protocols is that essentially all operating systems utilize it. Therefore, although the data for the web application will be hosted on a Linux-based system, it will be accessible from any device with a web browser and internet access. As Abraham Silberschatz et al. describe in their work, the IP part of TCP/IP is responsible for "[implementing] routing between different hosts…" while TCP ensures that the connection is "reliable… between hosts with guaranteed in-order delivery of packets…" (2008). By using this pair of protocols, the server can ensure it routes clients correctly and that once data begins to transfer to and from the server, it is validated before being processed to avoid potential errors. In terms of serving the web application to multiple users, a program called Apache will be used to open a connection to a port on the server, allowing any user who connects through this port to access the web application. Once the web application is hosted, any changes made to update it will require a reboot. Thus, the web application will experience some downtime, which will also occur if any server updates, changes, or outages happen. To help combat this, we can look for a third party that specifically offers backups and redundancies in case an issue arises. Having access to this option allows the main server to be taken offline while a reserve is activated to continue hosting the web application. Again, while this will still result in potential downtime, it will be more effective than solely relying on fixing the main server, as the issue could take a long time to solve.

1. **Security**:

Protecting user information will be essential for this project. While no security measures are fully unbreachable, we will ensure that proper security measures are in place to minimize the chance of unauthorized users accessing the server. To help with this, layering security measures will ensure that even if one method is broken or bypassed, another will still be there to protect the server and user data. To start, using Linux as the server's operating system will serve as the first security measure. Fewer vulnerabilities are present in Linux-based systems, and even if one is discovered, it is quickly patched due to the involvement of multiple communities in the operating system's development. Additionally, the Linux operating system has rigid authentication and access control to ensure that no unauthorized users can access the server or any data housed within it. Even when issues occur, Linux utilizes a sandboxing feature in which most processes are isolated from one another. As Brittany Day details in her work, the Linux operating system has many features that help to protect against viruses, and features such as "robust permission settings and sandboxing features… ensures that the impact is usually contained if malware slips through" (2024). No security measures are perfect, so accounting for issues when they occur can significantly help in preserving and protecting the user data housed on the server. These features can add additional security and provide ways to remove or isolate any malware or hackers in the system. To further this, adding extra security measures or layers will also be of great help. One such layer will be that the server will use two-factor authentication or an SSH key instead of relying solely on passwords to access user accounts. In doing this, even if a password is compromised, the hacker would need to gain access to another system to obtain an SSH key or to utilize a two-factor authenticator. This additional layer of security will provide more protection and potentially stop the hacker from accessing the server, even though they have already breached the security layer. Still, the server will need to allow external users to access the system to connect to the web application. Thus, to ensure a secure connection with users, the server will utilize HTTPS. This decision ensures that all data transferred to and from the server is secured via SSL/TLS, which encrypts data sent from the website to the user and vice versa once the connection is validated. This choice will prevent issues with eavesdroppers, as even if they intercept any data, it will be encrypted. To further improve this measure, we will use encryption to secure sensitive user data before sending it to users. This additional action ensures that if the SSL/TLS security method is bypassed, sensitive data will still be encrypted by other means. The same practice can be used when storing user data, as we will encrypt any sensitive user data even before storing it in the secure database. In the event the database is breached, the sensitive user data will still have an extra layer of security through encryption. To secure the database itself, multiple security measures will be used, such as controlling access and ensuring that those with access have secure passwords or access keys.References

*Day, B. (2024, December 30). Linux Security Overview: User Savvy, Architecture & Malware Defense.*

*Linux Security; LinuxSecurity.com.* [*https://linuxsecurity.com/features/linux-vs-viruses*](https://linuxsecurity.com/features/linux-vs-viruses)

*Silberschatz, A., Galvin, P. B., & Gagne, G. (2008). Operating System Concepts, 8th Edition.*

*John Wiley & Sons.* [*https://learning.oreilly.com/library/view/operating-system-concepts/9780470128725/*](https://learning.oreilly.com/library/view/operating-system-concepts/9780470128725/)