What is an Operating System IV

4.1 Components III. An *operating system* (OS) is software that manages computer hardware and software resources and provides common services for computer programs. The operating system is an essential component of the system software in a computer system. Application programs usually require an operating system to function.

Networking

Currently most operating systems support a variety of networking protocols, hardware, and applications for using them. This means that computers running dissimilar operating systems can participate in a common network for sharing resources such as computing, files, printers, and scanners using either wired or wireless connections. Networks can essentially allow a computer's operating system to access the resources of a remote computer to support the same functions as it could if those resources were connected directly to the local computer. This includes everything from simple communication, to using networked file systems or even sharing another computer's graphics or sound hardware. Some network services allow the resources of a computer to be accessed transparently, such as SSH which allows networked users direct access to a computer's command line interface.

Client/server networking allows a program on a computer, called a client, to connect via a network to another computer, called a server. Servers offer (or host) various services to other network computers and users. These services are usually provided through ports or numbered access points beyond the server's network address. Each port number is usually associated with a maximum of one running program, which is responsible for handling requests to that port. A daemon, being a user program, can in turn access the local hardware resources of that computer by passing requests to the operating system kernel.

Many operating systems support one or more vendor-specific or open networking protocols as well, for example, SNA on IBM systems, DECnet on systems from Digital Equipment Corporation, and Microsoft-specific protocols (SMB) on Windows. Specific protocols for specific tasks may also be supported such as NFS for file access. Protocols like ESound, or esd can be easily extended over the network to provide sound from local applications, on a remote system's sound hardware.

Security

A computer being secure depends on a number of technologies working properly. A modern operating system provides access to a number of resources, which are available to software running on the system, and to external devices like networks via the kernel.

The operating system must be capable of distinguishing between requests which should be allowed to be processed, and others which should not be processed. While some systems may simply distinguish between "privileged" and "non-privileged", systems commonly have a form of requester *identity*, such as a user name. To establish identity there may be a process of *authentication*. Often a username must be quoted, and each username may have a password. Other methods of authentication, such as magnetic cards or biometric data, might be used instead. In some cases, especially connections from the network, resources may be accessed with no authentication at all (such as reading files over a network share). Also covered by the concept of requester **identity** is *authorization*; the particular services and resources accessible by the requester once logged into a system are tied to either the requester's user account or to the variously configured groups of users to which the requester belongs.

In addition to the allow/disallow model of security, a system with a high level of security will also offer auditing options. These would allow tracking of requests for access to resources (such as, "who has been reading this file?"). Internal security, or security from an already running program is only possible if all possibly harmful requests must be carried out through interrupts to the operating system kernel. If programs can directly access hardware and resources, they cannot be secured.

External security involves a request from outside the computer, such as a login at a connected console or some kind of network connection. External requests are often passed through device drivers to the operating system's kernel, where they can be passed onto applications, or carried out directly. Security of operating systems has long been a concern because of highly sensitive data held on computers, both of a commercial and military nature. The United States Government Department of Defense (DoD) created the *Trusted Computer System Evaluation Criteria* (TCSEC) which is a standard that sets basic requirements for assessing the effectiveness of security. This became of vital importance to operating system makers, because the TCSEC was used to evaluate, classify and select trusted operating systems being considered for the processing, storage and retrieval of sensitive or classified information.

Network services include offerings such as file sharing, print services, email, web sites, and file transfer protocols (FTP), most of which can have compromised security. At the front line of security are hardware devices known as firewalls or intrusion detection/prevention systems. At the operating system level, there are a number of software firewalls available, as well as intrusion detection/prevention systems. Most modern operating systems include a software firewall, which is enabled by default. A software firewall can be configured to allow or deny network traffic to or from a service or application running on the operating system. Therefore, one can install and be running an insecure service, such as Telnet or FTP, and not have to be threatened by a security breach because the firewall would deny all traffic trying to connect to the service on that port.

An alternative strategy, and the only sandbox strategy available in systems that do not meet the Popek and Goldberg virtualization requirements, is where the operating system is not running user programs as native code, but instead either emulates a processor or provides a host for a p-code based system such as Java.

Internal security is especially relevant for multi-user systems; it allows each user of the system to have private files that the other users cannot tamper with or read. Internal security is also vital if auditing is to be of any use, since a program can potentially bypass the operating system, inclusive of bypassing auditing.

User interface

A screenshot of the Bourne Again Shell command line. Each command is typed out after the 'prompt', and then its output appears below, working its way down the screen. The current command prompt is at the bottom.

Every computer that is to be operated by an individual requires a user interface. The user interface is usually referred to as a shell and is essential if human interaction is to be supported. The user interface views the directory structure and requests services from the operating system that will acquire data from input hardware devices, such as a keyboard, mouse or credit card reader, and requests operating system services to display prompts, status messages and such on output hardware devices, such as a video monitor or printer. The two most common forms of a user interface have historically been the command-line interface, where computer commands are typed out line-by-line, and the graphical user interface, where a visual environment (most commonly a WIMP) is present.

Graphical user interfaces

A screenshot of the KDE Plasma Desktop graphical user interface. Programs take the form of images on the screen, and the files, folders (directories), and applications take the form of icons and symbols. A mouse is used to navigate the computer.

Most of the modern computer systems support graphical user interfaces (GUI), and often include them. In some computer systems, such as the original implementation of Mac OS, the GUI is integrated into the kernel.

While technically a graphical user interface is not an operating system service, incorporating support for one into the operating system kernel can allow the GUI to be more responsive by reducing the number of context switches required for the GUI to perform its output functions. Other operating systems are modular, separating the graphics subsystem from the kernel and the Operating System. In the 1980s UNIX, VMS and many others had operating systems that were built this way. Linux and Mac OS X are also built this way. Modern releases of Microsoft Windows such as Windows Vista implement a graphics subsystem that is mostly in user-space; however the graphics drawing routines of versions between Windows NT 4.0 and Windows Server 2003 exist mostly in kernel space. Windows 9x had very little distinction between the interface and the kernel.

Many computer operating systems allow the user to install or create any user interface they desire. The X Window System in conjunction with GNOME or KDE Plasma Desktop is a commonly found setup on most Unix and Unix-like (BSD, Linux, Solaris) systems. A number of Windows shell replacements have been released for Microsoft Windows, which offer alternatives to the included Windows shell, but the shell itself cannot be separated from Windows.

Numerous Unix-based GUIs have existed over time, most derived from X11. Competition among the various vendors of Unix (HP, IBM, Sun) led to much fragmentation, though an effort to standardize in the 1990s to COSE and CDE failed for various reasons, and were eventually eclipsed by the widespread adoption of GNOME and K Desktop Environment. Prior to free software-based toolkits and desktop environments, Motif was the prevalent toolkit/desktop combination (and was the basis upon which CDE was developed).

Graphical user interfaces evolve over time. For example, Windows has modified its user interface almost every time a new major version of Windows is released, and the Mac OS GUI changed dramatically with the introduction of Mac OS X in 1999.

4.2 Operating Systems

Real-time operating systems

A real-time operating system (RTOS) is an operating system intended for applications with fixed deadlines (real-time computing). Such applications include some small embedded systems, automobile engine controllers, industrial robots, spacecraft, industrial control, and some large-scale computing systems.

An early example of a large-scale real-time operating system was Transaction Processing Facility developed by American Airlines and IBM for the Sabre Airline Reservations System.

Embedded systems that have fixed deadlines use a real-time operating system such as VxWorks, PikeOS, eCos, QNX, MontaVista Linux and RTLinux. Windows CE is a real-time operating system that shares similar APIs to desktop Windows but shares none of desktop Windows' codebase. Symbian OS also has an RTOS kernel (EKA2) starting with version 8.0b.

Some embedded systems use operating systems such as Palm OS, BSD, and Linux, although such operating systems do not support real-time computing.

Operating system development as a hobby

Operating system development is one of the most complicated activities in which a computing hobbyist may engage. A hobby operating system may be classified as one whose code has not been directly derived from an existing operating system, and has few users and active developers.

In some cases, hobby development is in support of a "homebrew" computing device, for example, a simple single-board computer powered by a 6502 microprocessor. Or, development may be for an architecture already in widespread use. Operating system development may come from entirely new concepts, or may commence by modeling an existing operating system. In either case, the hobbyist is his/her own developer, or may interact with a small and sometimes unstructured group of individuals who have like interests.

Examples of a hobby operating system include ReactOS and Syllable.

Diversity of operating systems and portability

Application software is generally written for use on a specific operating system, and sometimes even for specific hardware. When porting the application to run on another OS, the functionality required by that application may be implemented differently by that OS (the names of functions, meaning of arguments, etc.) requiring the application to be adapted, changed, or otherwise maintained.

This cost in supporting operating systems diversity can be avoided by instead writing applications against software platforms like Java or Qt. These abstractions have already borne the cost of adaptation to specific operating systems and their system libraries.

Another approach is for operating system vendors to adopt standards. For example, POSIX and OS abstraction layers provide commonalities that reduce porting costs