Rebecca Katz

CS 340- HW1

Dr. Fluture

**Part A:**

MS Windows

The Windows operating system, distributed by Microsoft, was created by Bill Gates and Paul Allen when they realized that the future of computing was headed towards personal computers (PC). Their vision was to make computers accessible to the masses.

The original operating system that Microsoft developed was named MS-DOS (Microsoft Disc Operating System). Overall MS-DOS was found to be effective but difficult for the average user to operate. So on November 20, 1985 Microsoft released Windows 1.0 written in the languages: Assembly, C, and C++. In this version users can now use the simple mouse pointer to click through “screens” or “windows”, rather than confusing command lines. 1.0 used a 16-bit architecture.

Windows 95, released in August of 1995, features the first appearance of the Start menu, taskbar, and minimize/ maximize/ close buttons on each window. Windows 95 has built-in Internet support, dial-up networking, and new Plug and Play capabilities that make it easy to install hardware and software. The new hybrid architecture of a 16/32-bit operating system also offers enhanced multimedia capabilities, more powerful features for mobile computing, and integrated networking.

Windows XP offers much needed improvements upon its release in 2001, most notably the increase in speed and stability. Windows XP abandoned the long-used Windows 95 kernel (core software code) for a more powerful code base and offered a more practical interface and improved application and memory management.  XP offers intuitive navigation, remote desktop support, an encrypting file system, and system restore and advanced networking features. The system architecture is updated to support [IA-32](http://en.wikipedia.org/wiki/IA-32), [IA-64](http://en.wikipedia.org/wiki/IA-64), and [x86-64](http://en.wikipedia.org/wiki/X86-64).

Finally, the release of Windows 8 in late 2012 has dual-functionality as both a tablet for entertainment and a fully-featured PC. Windows 8 also includes enhancements of the familiar Windows desktop, with a new taskbar and streamlined file management. Windows 8 architecture supports [IA-32](http://en.wikipedia.org/wiki/IA-32), [x86-64](http://en.wikipedia.org/wiki/X86-64), [ARM architecture](http://en.wikipedia.org/wiki/ARMv7) (ARMv7).

UNIX

In 1969 Dennis Ritchie, Ken Thompson, and others began working on what would become UNIX operating system in AT&T's Bell Laboratories. UNIX was initially released in 1973. The UNIX systems have a graphical user interface (GUI) similar to Microsoft Windows which provides an easy to use environment. However, knowledge of UNIX is required for operations which aren't covered by a graphical program, or for when there is no windows interface available. The [C programming language](http://en.wikipedia.org/wiki/C_(programming_language)) was designed by [Dennis Ritchie](http://en.wikipedia.org/wiki/Dennis_Ritchie) as a systems programming language for UNIX. UNIX is written in a combination of C and Assembly languages.

Unix systems are characterized by a modular design known as the "[Unix philosophy](http://en.wikipedia.org/wiki/Unix_philosophy)": the OS provides a set of simple tools that each perform a limited, well-defined function, with a unified [filesystem](http://en.wikipedia.org/wiki/Unix_filesystem" \o "Unix filesystem) as the main means of communication and a [shell](http://en.wikipedia.org/wiki/Unix_shell) scripting and command language to combine the tools to perform complex workflows.

UNIX OS consists of 3 parts: the kernel, shell, and programs.

The kernel allocates time and memory to programs and handles the file-store and communications in response to system calls. The shell is a Command Line Interpreter (CLI) which acts as an interface between the user and kernel. After a user logs in and is authenticated, the shell program starts; it interprets and executes user-given commands which are programs themselves.

Everything in UNIX is either a file, a collection of data created by users using editors, or a process, an executing program identified by a unique PID (process identifier).

**Part B:**

Android OS

The Android mobile Operating System is based on Google’s Linux kernel and was initially released in November 2008. The user interface is specifically designed for mobile devices with ARM architecture, using direct manipulation: user actions which loosely correspond to manipulation of [physical objects](http://en.wikipedia.org/wiki/Physical_object). Android utilizes hardware commonly found in mobile devices such as the accelerometer, gyroscope, proximity sensors and vibration motors. The code is open source, meaning it’s available for public viewing. The core of Android OS is written in C, other parts in C++, and the UI is written in Java and the OS platform is supported on [32-bit](http://en.wikipedia.org/wiki/32-bit) [ARM](http://en.wikipedia.org/wiki/ARMv7), [MIPS](http://en.wikipedia.org/wiki/MIPS_architecture), and [x86](http://en.wikipedia.org/wiki/X86). The applications are written primarily in the Java programming language along with the Android SDK. The application framework of Android allows for replacement and reuse of existing components.

The kernel type of the Android OS is a modified Linux kernel, called monolithic. A monolithic kernel is an operating system architecture where the entire operating system is working in [kernel space](http://en.wikipedia.org/wiki/Kernel_space) and is alone in [supervisor mode](http://en.wikipedia.org/wiki/Supervisor_mode). The monolithic model differs from other operating system architectures in that it alone defines a high-level virtual interface over computer hardware. A set of primitives or [system calls](http://en.wikipedia.org/wiki/System_call) implement all operating system services such as [process](http://en.wikipedia.org/wiki/Process_(computing)) management, [concurrency](http://en.wikipedia.org/wiki/Concurrency_(computer_science)), and [memory management](http://en.wikipedia.org/wiki/Memory_management).

To keep power consumption at a minimum, Android has a memory management system thatwill automatically suspend an Android app in memory when it is no longer in use. Suspended apps consume no resources, like battery power or processing power, and sit idly in the background until needed again. The benefits of this management are increased general responsiveness of Android devices and also ensuring that background applications do not needlessly consume power.

**Part C:**

Real-Time OS

A Real Time Operating System is a specific type of operating system that is intended to process incoming data without a buffer delay. Requirements for processing times are measured in 1/10 of a second. The time it takes to accept and complete an application’s task is extremely consistent in RTOS and the variability is called jitter. RTOS’s are categorized into soft and hard performance category which translates to a high vs low jitter respectively. The algorithm used on RTOS’s for scheduling is complex and allows for focusing on minimal interrupt and thread switching latency. The key quality of a RTOS is its speed/predictability of response rather than quantity of work accomplished. A strong emphasis is put on time constraints with RTOS’s especially with the hard RTOS category where it’s considered a failure if the system doesn’t complete a task within its constraint. Therefore RTOS’s require multitasking, process threads that can be prioritized on many interrupt levels and so allows programmers access to task prioritizations and deadline confirmation. Some examples of Real Time Operating Systems are Lynx OS, OSE, QNX, RT Linux, Vx Works and Windows CE.

Embedded Systems

Embedded systems are computer systems with a specific function sometimes within a larger system, hence the name embedded. In this way embedded systems are not flexible or very user oriented, although they could be part of a user-end device, such as a PC. Embedded systems usually don’t allow for user interactions as a computer does but still has the core components that a computer has namely a processor, software and I/O. In other words embedded systems are not always standalone devices, they often consist of small, computerized parts within a larger device. The fact that the embedded systems are specific function systems, the system design engineers are able to strongly focus on optimizing it to reduce cost and size while increases performance and reliability. Some embedded systems work within the real time operating system performance constraints for safety, security and usability. Embedded systems generally do not need entire operating systems to run and thus use program instructions called firmware which are stored in ROM or flash memory chips. The systems run with limited hardware resources and do not allow much direct user interaction. There are many examples of embedded system in modern life. They’re used in telephones, cell networks and Wi-Fi routers. Simple consumer electronics such as clock radios, DVD players, GPS receivers and printers include embedded systems as well. Medical devices are examples of embedded system use as well with defibrillators and insulin pumps.

Embedded vs Real-Time

Embedded systems are different from real-time systems. Many embedded systems require real-time system constraints (as discussed in the RTOS section above). This means that if the embedded system is designed in order to guarantee that real-time application requests will be met within the deadlines that are predefined than it is considered “real-time”. Thus some embedded systems require the constraint, such as aircraft control and pacemakers. Therefore real-time systems are actually a category within embedded systems.

TSR Processes

TSR (terminate and stay resident) processes refer to programs within DOS that can remain in memory once they’ve been loaded in order to be easily reactivated by a simple interrupt as opposed to the process being removed from the memory and having to be retrieved again. This was used in the times of DOS to overcome the limitation of single task execution at a time. The advantage of this is these processes are generally fast to load. The disadvantage is that many TSR’s at a time greatly reduce the memory availability to other processes and programs. Additionally, some TSR’s might not interact well with each other if loaded simultaneously in the memory.

A current example of a TSR process is a virus, which remains inside of the memory to infect the machine. Conversely a virus scanner remains loaded in memory to protect the computer from viruses.

**Part D:**

Interrupts and Interrupt Requests

An interrupt is a signal sent from either a hardware device or a program notifying the operating system that it needs to take care of something. Computers nowadays use interrupts all the time to ensure attention is given when needed. In this way a program that is currently running can be stopped in order for the operating system to be able to handle something else. This interrupt service is the key to multitasking nowadays. Although using interrupts isn’t exactly perfect multitasking, operating system engineers have designed them to act as though many things can happen at the exact same time but in reality there are many fast interrupts going on in the background. The process specifically starts with the current code or thread being suspended (temporarily saved) then an interrupt handler is executed to deal with the event. Once the event has been dealt with the processor resumes to execute the previous thread.

There are two possible types of interrupts:

* A hardware interrupt is an electronic alerting signal sent to the processor from an external device, either a part of the computer itself such as a [disk controller](http://en.wikipedia.org/wiki/Disk_controller) or an external [peripheral](http://en.wikipedia.org/wiki/Peripheral). For example from a keyboard or mouse press.
* A software interrupt is caused either by an exceptional condition in the processor itself, or a special [instruction](http://en.wikipedia.org/wiki/Instruction_(computer_science)) in the [instruction set](http://en.wikipedia.org/wiki/Instruction_set) which causes an interrupt when it is executed. For example, if the processor’s arithmetic logic unit is told to divide by zero.

All interrupts have a matching interrupt handler and all hardware interrupts initiations are called interrupt requests. Hardware interrupts are limited by the number of designated interrupt request lines to the processor while software interrupts don’t have a constraint. The interrupt request value is specific to each device thus originally had to be entered manually into the system in order to function properly because only one interrupt request can be handled at a time. So if there are identical values would cause a conflict. The interrupt requests are also assigned priorities in order to improve efficiency.

The “man” command is used to access the manuals for Linux commands. Executing the “man trap” command in the terminal returns the list of commands built into the Bash shell, including a detailed description of each command as well as the arguments they accept. For example, the operator “.”, which is used to execute a command from a shell script in the current directory, is actually a command that is built into the bash shell. The trap instruction is used to respond to hardware signals (interrupts). UNIX executes the trap commands based on alphabetic/numeric order. The trap command allows a user to specify how to respond to specific hardware signals. In other words the trap command handles certain types of hardware signals such as the escape key in ways that the programmer is able to choose. For example, a programmer could choose to handle a keyboard key with a pop up dialogue response.

**Sources:**

http://en.wikipedia.org/wiki/Microsoft\_Windows

http://www.britannica.com/EBchecked/topic/645197/Windows-OS

http://en.wikipedia.org/wiki/Comparison\_of\_Microsoft\_Windows\_versions

http://windows.microsoft.com/en-us/windows/history#T1=era0

http://en.wikipedia.org/wiki/Unix

http://www.unix.org/what\_is\_unix.html

http://www.ee.surrey.ac.uk/Teaching/Unix/unixintro.html

http://en.wikipedia.org/wiki/Android\_(operating\_system)

http://en.wikipedia.org/wiki/Monolithic\_kernel

http://en.wikipedia.org/wiki/Interrupt

http://www.computerhope.com/unix/utrap.htm