EECS3221 notes

Jerry Wu

Week II 2023/05/18

Operating system structures

Services

- Operating systems provide an environment for execution of programs and services to programs and users.
- Some other OS services include but are not limited to::
  - **UI** - Almost all operating systems have a user interface (cmd, GUI, touch screen, etc)
  - **Program execution** - The system must be able to load a program into memory and run the program, end execution either normally or abnormally (indicating an error)
  - **IO operations** - A running program may require IO which may involve a file or an IO device
  - **File system manipulation** - Programs need to read and write files and directories, create and delete them, search, list file info, and permission management
  - **Communications** - Processes may exchange information on the same computer or between computers over a network
    - Communications may be via shared memory or through message passing (packets moved by the OS)
  - **Error detection** - OS needs to be constantly aware of possible errors
    - May occur in the CPU and memory, IO devices, or in software
    - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
    - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system
  - **Resource allocation**

  - **Security**

- Our goal is to develop an understanding of OS in order to build scalable software

CLI (command line interpreter)

CLI allows direct command entry closer to the hardware which is much faster than a GUI

- Sometimes implemented in the kernel, sometimes by system programs
- Sometimes multiple flavours implemented (shells)
- Primarily fetches a command from the user and executes
- There are multiple build in commands, and some are just names of system programs
  - If the latter, adding new features doesn't require shell modification

GUI

First invented by Xerox. Made for ease of use for the user so they won't have to use commands.

- Controlled by keyboard and mouse with a monitor for viewing, or touch screens
- Icons represent files, programs, actions, etc
- Various mouse buttons over objects in the interface cause actions (providing information, options, function execution, opening directories (folders))

System calls

- Programming interface to the services provided by the OS
- Typically written in a high level language (C or C++)
- Mostly accessed by programs via a high level API rather than direct system call use
- Three most common APIs are Win32 for windows, POSIX for UNIX, mac, Linux, etc. and the Java API for the JVM.

Implementation

- Each system call is associated with a number
  - **System call interface** maintains a table indexed according to said numbers
- The system all interface invokes the intended system call in the OS kernel and returns the status of the system call and any return values
- To make system calls, we need to pass parameters into the OS
  - Simplest: pass parameters in registers (sometimes there are more parameters than there are registers)
  - Pass in blocks, or table in memory and address of the block passed as a parameter in a register (Linux/solaris)
  - Parameters are pushed and popped from the program stack

Types of system calls

System call API cheat sheet on **slide 2.24**

- Create/terminate processes
- end, abort, load, execute
- get process attributes, set process attributes
- wait for time/event/signal event
- allocate and free memory
- dump memory if error
- **debuggers** for determining bugs: **single step** execution
- **locks** for managing access to shared date between processes

Example:: Arduino

- Single task (one program at a time)
- No on board OS
- Programs (sketch) loaded via USB stick

System services

System programs provide a convenient environment for program development and execution. They can be divided into::

- File manipulation/management (text editors, file manager, etc)
- Status information sometimes stored in a file(s) (task manager)
- Programming language support (GCC, JDK, etc)
- Program loading and execution
- Communications
- Background services (daemons)
- Application programs (everything else)

Most users' view of the OS is deined by system programs and not the actual system calls.

Linkers and loaders

- Source code compiled into object files designed to be loaded into any physical memory location - **relocatable object file**
- **Linker** combines these into a single binary executable file.
-