|  |
| --- |
| **Operating System Services** |
| * User Interface   + Command-line interface – text commands via keyboard or typing in commands   + Batch interface – commands/directives entered into files, and the files are executed   + Graphical user interface (GUI) - window system with pointing /touch device * Program Execution   + Load the program into memory and run the program   + Program ends normally or abnormally (indicating an error) * I/O operations * File-system manipulation   + CRUD   + Read/write/create/delete files   + Provide permissions management to allow/deny access * Communications   + Process needs to exchange information with another process   + Processes could be on the same computer, or different computers   + Implemented via shared memory     - Two processes read/write to shared section of memory   + Implemented via message passing     - Packets with predefined formats are moved between processes * Error detection   + Memory error, power failure, parity error on disk, network connection failure, printer out of paper, illegal memory location, too much CPU time used * Resource allocation   + CPU cycles, main memory, file storage * Accounting – what resources and how long are users using them for -> billing purposes, security, optimization/efficiency * System calls -> windows – windows API; Unix, Linux, Mac OSX – POSIX API; Java Virtual Machine – Java API |
| **Policy vs Mechanism**   * Mechanisms – how to do something * Policies – what will be done * Policies are flexible * Example: timer construct is a mechanism for ensuring CPU protection. Deciding how long the timer is set is a policy decision. |
| **Modules**   * Loadable kernel modules – modules that can be loaded at boot time OR during run time * Saves memory by only loading kernel modules that are needed (e.g., printing) |
| **Operating-System Debugging**   * Failure Analysis |
| **Strace vs Dtrace** |