The Life Cycle of a Daemon

Starting the User Environment

The root process on OS X is `launchd` (which replaces the `mach_init` and `init` processes used in previous versions of OS X and many traditional Unix systems). In addition to initializing the system, the `launchd` process coordinates the launching of system daemons in an orderly manner. Like the `inetd` process, `launchd` launches daemons on-demand. Daemons launched in this manner can be shut down during periods of inactivity and relaunched as needed. When a service request comes in, if the daemon is not running, `launchd` automatically starts the daemon to process the request.

Launching daemons on demand frees up memory and other resources associated with the daemon, which is worthwhile if the daemon is likely to be idle for extended periods of time. More importantly, however, this guarantees that runtime dependencies between daemons are satisfied without the need for manual lists of dependencies.

As the final part of system initialization, `launchd` launches `loginwindow`. The `loginwindow` program controls several aspects of user sessions and coordinates the display of the login window and the authentication of users.

For information about earlier portions of the boot process, see The Early Boot Process in *Kernel Programming Guide*.

Authenticating Users

OS X requires users to authenticate themselves prior to accessing the system. The `loginwindow` program coordinates the visual portion of the login process (as manifested by the window where users enter name and password information) and the security portion (which handles the user authentication). Once a user has been authenticated, `loginwindow` begins setting up the user environment.

In two key situations, `loginwindow` bypasses the usual login prompt and begins the user session immediately. The first situation occurs when the system administrator has configured the computer to automatically log in as a specified user. The second occurs during software installation when the installer program is to be launched immediately after a reboot.

Configuring User Sessions

Immediately after the user is successfully authenticated, `loginwindow` sets up the user environment and records information about the login. As part of this process, it performs the following tasks:

- Secures the login session from unauthorized remote access.
- Records the login in the system's `utmp` and `utmpx` databases.
- Sets the owner and permissions for the console terminal.
- Resets the user's preferences to include global system defaults.
- Configures the mouse, keyboard, and system sound using the user's preferences.

- Sets the user's group permissions (`gid`).
- Retrieves the user record from Directory Services and applies that information to the session.
- Loads the user's computing environment (including preferences, environment variables, device and file permissions, keychain access, and so on).
- Launches the Dock, Finder, and SystemUIServer.
- Launches the login items for the user.

Once the user session is up and running, `loginwindow` monitors the session and user applications in the following ways:

- It manages the logout, restart, and shutdown procedures. See Logout Responsibilities for more information.
- It manages the Force Quit window, which includes monitoring the currently active applications and responding to user requests to force-quit applications and relaunch the Finder. (Users open this window from the Apple menu or by pressing Command-Option-Escape.)

Logout Responsibilities

The procedures for logging out, restarting the system, or shutting down the system are similar. A typical logout/restart/shutdown takes place as follows:

1. The user selects Log Out, Restart, or Shut Down from the Apple menu.
2. The foreground application initiates the user request by sending an Apple event to `loginwindow`. (See Initiating a Logout, Restart, or Shutdown for a list of events.) For Cocoa applications, this is done by the Application Kit.
3. The `loginwindow` program displays an alert to the user asking for confirmation of the action.
4. If the user confirms the action, `loginwindow` quits every foreground and background user process.
5. Once all processes have quit, `loginwindow` ends the user session and performs the logout, restart or shutdown.

Terminating Processes

As part of a log out, restart, or shutdown sequence, `loginwindow` attempts to terminate all foreground and background user processes.

Your process should support sudden termination for the best user experience. See *NSProcessInfo Class Reference* for information on how to adopt this technology. If your process supports sudden termination, it is just sent a `SIGKILL` signal. If you have temporarily disabled sudden termination, the normal process below applies.

For Cocoa applications, termination is partly handled by the Application Kit, which calls the `applicationShouldTerminate:` delegate method. To abort the termination sequence, implement this method and return `NSTerminateCancel`; otherwise, termination of your application continues normally.

Non-Cocoa applications receive a "Quit Application" Apple event (`kAEQuitApplication`), as a courtesy, to give them a chance to shut down gracefully. The process should terminate itself immediately or post an alert dialog if a user confirmation is required (for example, if there is an unsaved document). As soon as that condition is resolved, the application should terminate. If the user decides to abort the termination sequence (by clicking Cancel in a Save dialog, for example) the application should respond to the event by returning a `userCanceledErr` error (`-128`).

If a foreground application fails to reply or terminate itself after 45 seconds, `loginwindow` automatically aborts the termination sequence. This safeguard is to protect data in various situations, such as when an application is saving a large file to disk and is unable to terminate in the allotted time. If a foreground application is unresponsive and not doing anything, the user must use the Force Quit window to quit it before proceeding.

For background processes, the procedure is a little different. The `loginwindow` program notifies the process that it is about to be terminated by sending it a Quit Application Apple event (`kAEQuitApplication`). Unlike foreground processes, however, `loginwindow` does not wait for a reply. It proceeds to terminate any open background processes by sending a `SIGKILL` signal, regardless of any returned errors.

If the system is being shut down or restarted, it sends a `SIGTERM` signal to all daemons, followed a few seconds later by `SIGKILL` signal.

Initiating a Logout, Restart, or Shutdown

To initiate a logout, restart, or shutdown sequence programmatically, the foreground application must send the appropriate Apple event to `loginwindow`. Upon receipt of the event, `loginwindow` begins the process of shutting down the user session.

The following list shows the preferred Apple events for logout, restart, and shutdown procedures. These events have no required parameters.

- `kAELogOut`
- `kAEShowRestartDialog`
- `kAEShowShutdownDialog`

In addition to the preferred Apple events, there are two additional events that tell `loginwindow` to proceed immediately with a restart or shutdown sequence:

Warning: If you send one of these events to `loginwindow`, the user does not have an opportunity to cancel the action, and unsaved data can be lost. These events should be used very seldom, if at all.

- `kAERestart`
- `kAEShutDown`