Introduction

Overview

Supervisor is a client/server system that allows its users to control a number of processes on UNIX-like operating systems. It was inspired by the following:

Convenience

> It is often inconvenient to need to write `rc.d` scripts for every single process instance. `rc.d` scripts are a great lowest-common-denominator form of process initialization/autostart/management, but they can be painful to write and maintain. Additionally, `rc.d` scripts cannot automatically restart a crashed process and many programs do not restart themselves properly on a crash. Supervisord starts processes as its subprocesses, and can be configured to automatically restart them on a crash. It can also automatically be configured to start processes on its own invocation.

Accuracy

> It's often difficult to get accurate up/down status on processes on UNIX. Pidfiles often lie. Supervisord starts processes as subprocesses, so it always knows the true up/down status of its children and can be queried conveniently for this data.

Delegation

> Users who need to control process state often need only to do that. They don't want or need full-blown shell access to the machine on which the processes are running. Processes which listen on "low" TCP ports often need to be started and restarted as the root user (a UNIX misfeature). It's usually the case that it's perfectly fine to allow "normal" people to stop or restart such a process, but providing them with shell access is often impractical, and providing them with root access or sudo access is often impossible. It's also (rightly) difficult to explain to them why this problem exists. If supervisord is started as root, it is possible to allow "normal" users to control such processes without needing to explain the intricacies of the problem to them. Supervisorctl allows a very limited form of access to the machine, essentially allowing users to see process status and control supervisord-controlled subprocesses by emitting "stop", "start", and "restart" commands from a simple shell or web UI.

Process Groups

Processes often need to be started and stopped in groups, sometimes even in a “priority order”. It’s often difficult to explain to people how to do this. Supervisor allows you to assign priorities to processes, and allows user to emit commands via the supervisorctl client like “start all”, and “restart all”, which starts them in the preassigned priority order. Additionally, processes can be grouped into “process groups” and a set of logically related processes can be stopped and started as a unit.

Features

Simple

Supervisor is configured through a simple INI-style config file that’s easy to learn. It provides many per-process options that make your life easier like restarting failed processes and automatic log rotation.

Centralized

Supervisor provides you with one place to start, stop, and monitor your processes. Processes can be controlled individually or in groups. You can configure Supervisor to provide a local or remote command line and web interface.

Efficient

Supervisor starts its subprocesses via fork/exec and subprocesses don’t daemonize. The operating system signals Supervisor immediately when a process terminates, unlike some solutions that rely on troublesome PID files and periodic polling to restart failed processes.

Extensible

Supervisor has a simple event notification protocol that programs written in any language can use to monitor it, and an XML-RPC interface for control. It is also built with extension points that can be leveraged by Python developers.

Compatible

Supervisor works on just about everything except for Windows. It is tested and supported on Linux, Mac OS X, Solaris, and FreeBSD. It is written entirely in Python, so installation does not require a C compiler.

Proven

While Supervisor is very actively developed today, it is not new software. Supervisor has been around for years and is already in use on many servers.

Supervisor Components

supervisord

The server piece of supervisor is named **supervisord**. It is responsible for starting child programs at its own invocation, responding to commands from clients, restarting crashed or exited subprocesseses, logging its subprocess `stdout` and `stderr` output, and generating and handling "events" corresponding to points in subprocess lifetimes.

The server process uses a configuration file. This is typically located in `/etc/supervisord.conf`. This configuration file is a "Windows-INI" style config file. It is important to keep this file secure via proper filesystem permissions because it may contain unencrypted usernames and passwords.

supervisorctl

The command-line client piece of the supervisor is named **supervisorctl**. It provides a shell-like interface to the features provided by **supervisord**. From **supervisorctl**, a user can connect to different **supervisord** processes, get status on the subprocesses controlled by, stop and start subprocesses of, and get lists of running processes of a **supervisord**.

The command-line client talks to the server across a UNIX domain socket or an internet (TCP) socket. The server can assert that the user of a client should present authentication credentials before it allows him to perform commands. The client process typically uses the same configuration file as the server but any configuration file with a `[supervisorctl]` section in it will work.

Web Server

A (sparse) web user interface with functionality comparable to **supervisorctl** may be accessed via a browser if you start **supervisord** against an internet socket. Visit the server URL (e.g. `http://localhost:9001/`) to view and control process status through the web interface after activating the configuration file's `[inet_http_server]` section.

XML-RPC Interface

The same HTTP server which serves the web UI serves up an XML-RPC interface that can be used to interrogate and control supervisor and the programs it runs. See *XML-RPC API Documentation*.

Platform Requirements

Supervisor has been tested and is known to run on Linux (Ubuntu 9.10), Mac OS X (10.4/10.5/10.6), and Solaris (10 for Intel) and FreeBSD 6.1. It will likely work fine on most UNIX systems.

Supervisor will *not* run at all under any version of Windows.

Supervisor is known to work with Python 2.4 or later but will not work under any version of Python 3.