Run a script on start up

Sometimes it's useful to automatically run a script on a Pi when it boots up. You might want to start a script that monitors sensors, a Bittorrent client, or a program to back up your Pi to the internet. There are several ways of doing this, and each one has it's benefits.

In the following examples I'm going to start a program called servod which manages a servo. The program runs in the background and uses the Pi's GPIO pins to set the position of a servo.

/etc/rc.local

The simplest method is to put a command in /etc/rc.local. This file is executed while Linux is booting up and before any users have logged in.

When rc.local executes, the path environmanet variable (or any other environment variable) may not have been set up so it's important to use the program or script's full path. It's also important to make sure the program doesn't prompt for user input, as users won't have logged in when rc.local runs. If the program waits for user input, it won't complete, and the boot process will grind to a halt before users can login.

Similar problems can occur with programs that aren't meant to complete like servod, which runs continuously as a service or daemon. The solution in this instance is to run servod in the background using the '&' symbol. This tells the bash shell to execute the program without waiting for it to complete.

```
/home/pi/servoblaster/PiBits/ServoBlaster/user/servod &
```

Crontab

The cron daemon is a Linux thread that runs once a minute. It checks a file called the crontab file to see if there are any jobs that need to be executed. Usually entries in crontab specify a time to run tasks, but '@reboot' can be used to tell cron to run a task as soon as your Pi boots. You can edit a crontab file by typing this command at a terminal:

```
sudo crontab -e
```

http://raspberrywebserver.com/serveradmin/run-a-script-on-start-up.html

Your crontab file will be opened in the nano editor. Use the arrow keys to move the cursor to the end of the file and add the command that you want to run:

```
@reboot /home/pi/servoblaster/PiBits/ServoBlaster/user/servod
```

To save changes to the file, press control-o. You will be prompted to save to a temporary file name. Press return, and the crontab program will install the temporary file in the correct place. It's best practice to use a program or script's full path in crontab files.

/etc/init.d

Sometimes it's necessary to restart a service if it's configuration files have changed. For example, if Apache's configuration files are modified, it needs to be restarted using this command:

```
sudo /etc/init.d/apache2 restart
```

This command executes a script in /etc/init.d/ which can start, stop or restart the Apache web server. You can create your own init.d script for programs that you want to run at boot time. It should be put in /etc/init.d. It must start with a shebang (#!/bin/bash) followed by a LSB header, a set of specially formatted comments that contain information about the service.

This is an init script that I created to control servod:

```
#!/bin/bash
# /etc/init.d/servoblaster

### BEGIN INIT INFO
# Provides:          servoblaster
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Example initscript
# Description:       This service is used to manage a servo
```

```
### END INIT INFO



case "$1" in
    start)
        echo "Starting servoblaster"
        /home/pi/servoblaster/PiBits/ServoBlaster/user/servod
        ;;
    stop)
        echo "Stopping servoblaster"
        killall servod
        ;;
    *)
        echo "Usage: /etc/init.d/servod start|stop"
        exit 1
        ;;
esac

exit 0
```

An example of the header can be found in /etc/init.d/skeleton.

The code that follows the LSB is usually a bash case statment for processing the start/stop command line option. The option is stored in $1, and the case statement tests to see if the option is start or stop. If no option (or the wrong option) was supplied the default case executes, and a usage message is printed. If the start option is used, the script runs servod, and if the option is stop, the script kills the servod process.

Sometimes when a computer is shutdown, services need to be stopped in a controlled way. Programs may need to finish logging data or release lock files before they are terminated. If a script is started using cron or rc.local, it will simply stop when the Pi is shutdown. Using the init.d method to control scripts means that when your Pi shuts down, Linux will gracefully shutdown the script so that it has a chance to finish off any final tasks.

I used chmod to make the script executable:

```
sudo chmod +x /etc/init.d/servod
```

http://raspberrywebserver.com/serveradmin/run-a-script-on-start-up.html

I can start the servod program with this command:

```
sudo /etc/init.d/servod start
```

...and stop it again with this one:

```
sudo /etc/init.d/servod stop
```

In order to make the program run on start up, it's necessary to run this command:

```
sudo update-rc.d /etc/init.d/servod defaults
```

This creates a link to /etc/init.d/servod in directories from /etc/rc0.d through to /etc/rc6.d. When Linux boots up or shuts down, it looks in these folders to see if any scripts or programs need to be run. When I restart my Pi the servod program starts automatically.