Taskmaster

Özet: *Bu projenin amacı, supervisor'a benzer özelliklere sahip bir iş kontrol daemon'u yapmaktır.*

Sürüm: 3

İçindekiler

Bölüm I Önsöz

İşte Sakinler hakkında küçük ama faydalı bir bilgi:

Sakinler gibi yaygın, uzun ömürlü, huysuz ve (işlerine geldiğinde) tek fikirli bir türle kavgaya tutuşmak çoğu , tam da tozun dumana karıştığını, geçmişin geçmişte kaldığını ve talihsiz anlaşmazlıkların artık tarih olduğunu düşündüğünüz anda (hatta jeolojik çağlar sonra), ana sisteminizde küçük bir gezegenin ansızın belirmesi anlamına geliyordu; bu gezegene bir uydu filosu eşlik ediyordu ve kendileri de çok sayıda asteroit büyüklüğünde parçalarla çevriliydi, Bunların her biri, sayısız iri kayadan oluşan bulanık bir kabuğun içinde kozalanıp, her biri daha küçük kaya ve çakıl taşlarından oluşan büyük bir heyelanla çevrili olarak seyahat ederken, tüm bu korkunç koleksiyon ışık hızına o kadar yakın bir hızda hareket ediyordu ki, özellikle dikkatli ve gözlemci bir türün bile uyarı miktarı genellikle yerel eşdeğer "Bu da ne--?" diye için yeterli bir süre olurdu.

Dwellers nedir? Google'da ara! Hayır, ama cidden, gidip Cebirci'yi okuyun. Eğer onu okursanız bu proje çok daha kolay olur.

Bölüm II Giriş

Unix ve Unix benzeri işletim sistemlerinde, iş kontrolü, işlerin bir kabuk tarafından, özellikle etkileşimli olarak kontrol edilmesini ifade eder; burada "iş" bir kabuğun bir işlem grubu için temsilidir. Temel iş kontrol özellikleri, iş/işlem grubundaki tüm işlemlerin askıya alınması, devam ettirilmesi veya sonlandırılmasıdır; daha gelişmiş özellikler işe sinyaller gönderilerek gerçekleştirilebilir. İş kontrolü, çoklu işlem nedeniyle Unix'te özellikle ilgi çekicidir ve genellikle sıralı yürütmeye (toplu işleme) uygulanan iş kontrolünden ayırt edilmelidir.

Bölüm III

Hedefler

Buradaki göreviniz tam teşekküllü bir iş kontrol daemon'u yapmaktır. Bunun için oldukça iyi bir örnek supervisor olabilir.

Basit tutmak adına, programınız root olarak çalışmayacaktır ve bir daemon olmak zorunda değildir. Kabuk aracılığıyla başlatılacak ve kullanıcıya bir kontrol kabuğu sağlarken işini yapacaktır.

Bölüm IV

Genel Talimatlar

IV.1 Dil kısıtlamaları

İstediğiniz dili kullanmakta özgürsünüz. Kütüphanelere yapılandırma dosyalarını ayrıştırmak ve eğer uygulamayı seçerseniz istemci/sunucu bonusu için izin verilir. Bunun dışında, kesinlikle dilinizin standart kütüphanesiyle sınırlısınız.

IV.2 Savunma oturumu

Savunma oturumu için aşağıdakilere hazırlıklı olun:

- Programınızın gerekli her bir özelliği doğru bir şekilde uyguladığını, sağlayacağınız bir yapılandırma dosyası ile çalıştırarak gösterin.
- Programınızın, denetlenen süreçleri manuel olarak öldürmek, hiçbir zaman doğru şekilde başlamayan süreçleri başlatmaya çalışmak, çok fazla çıktı üreten süreçleri başlatmak vb. dahil ancak bunlarla sınırlı olmamak üzere çeşitli şekillerde not vereniniz tarafından test edilmesini sağlayın.

Bölüm V

Zorunlu Kısım

Bu projenin bir Sanal Makine üzerinde yapılması gerekiyor.
Programınız işleri alt süreçler olarak başlatabilmeli ve gerektiğinde yeniden başlatarak canlı tutabilmelidir. Ayrıca, bu süreçlerin canlı mı yoksa ölü mü olduğunu da her zaman bilmelidir (Bu doğru olmalıdır).

Hangi programların başlatılacağı, nasıl başlatılacağı, kaç tane başlatılacağı, yeniden başlatılmaları gerekip gerekmediği gibi bilgiler, formatı size bağlı olan bir yapılandırma dosyasında yer alacaktır (örneğin YAML iyi bir fikirdir, ancak istediğinizi kullanın). Bu yapılandırma başlatma sırasında yüklenmeli ve taskmaster çalışırken ona bir SIGHUP gönderilerek yeniden yüklenebilir olmalıdır. Yeniden yüklendiğinde, programınızın çalışma durumunda gerekli tüm değişiklikleri yapması beklenir (Programları kaldırmak, bazılarını eklemek, izleme koşullarını değiştirmek, vb...), ancak yeniden yüklemede değiştirilmemiş süreçleri ortaya ÇIKARMAMALIDIR.

Programınız, olayları yerel bir dosyaya kaydeden bir günlük tutma sistemine sahip olmalıdır (Bir program başlatıldığında, durdurulduğunda, yeniden başlatıldığında, beklenmedik bir şekilde öldüğünde, yapılandırma yeniden yüklendiğinde, vb...)

Başlatıldığında, programınız ön planda kalmalı ve kullanıcıya bir kontrol kabuğu sağlamalıdır. 42sh gibi tam teşekküllü bir kabuk zorunda değildir, ancak en azından kullanılabilir olmalıdır (Satır düzenleme, geçmiş ... tamamlama da güzel olurdu). Supervisor'ın kontrol kabuğu supervisorctl'den ilham alın.

Ana bilgisayar sanal makinenizi kurmak için istediğiniz araçları kullanabilirsiniz.

Bu kabuk en azından kullanıcının şunları izin vermelidir:

- Yapılandırma dosyasında açıklanan tüm programların durumunu görün ("status" komutu)
- Programları başlatma / durdurma / yeniden başlatma
- Ana programı durdurmadan yapılandırma dosyasını yeniden yükleyin
- Ana programı durdurun

Yapılandırma dosyası, kullanıcının denetlenecek her program için aşağıdakileri belirtmesine izin vermelidir:

- Programı başlatmak için kullanılacak komut
- Başlatılacak ve çalışır durumda tutulacak işlem sayısı
- Bu programın açılışta başlatılıp başlatılmayacağı
- Programın her zaman mı, hiçbir zaman mı yoksa yalnızca beklenmedik çıkışlarda mı yeniden başlatılacağı
- Hangi dönüş kodları "beklenen" çıkış durumunu temsil eder
- Programın "başarıyla başlatılmış" sayılması için başlatıldıktan sonra ne kadar süre çalışması gerektiği
- İptal edilmeden önce kaç kez yeniden başlatma denemesi yapılmalıdır
- Programı durdurmak (yani zarif bir şekilde çıkmak) için hangi sinyal kullanılmalıdır
- Zararsız durdurmadan sonra programı öldürmeden önce ne kadar süre bekleneceği
- Programın stdout/stderr'lerini atma veya dosyalara yönlendirme seçenekleri
- Programı başlatmadan önce ayarlanması gereken ortam değişkenleri
- Programı başlatmadan önce ayarlanacak bir çalışma dizini
- Programı başlatmadan önce ayarlanacak bir umask

Bölüm VI Bonus kısmı

Projenizin faydalanacağını düşündüğünüz herhangi bir ek özelliği uygulamanız teşvik edilir. Doğru bir şekilde uygulandığında ve en azından belli belirsiz yararlı olduğunda puan alacaksınız.

İşte size başlamanız için bazı fikirler:

- Başlatma sırasında ayrıcalık azaltma (Kök olarak başlatılması gerekir).
- İstemci/sunucu mimarisi iki ayrı programa izin verir: Gerçek iş kontrolünü yapan bir daemon ve kullanıcı için bir kabuk sağlayan ve daemon ile UNIX veya TCP soketleri üzerinden iletişim kuran bir kontrol programı. (Supervisord ve supervisorctl'e çok benzer)
- Daha gelişmiş günlükleme/raporlama olanakları (E-posta/http/syslog/vb... yoluyla uyarılar)
- Kullanıcının denetlenen bir süreci konsoluna "eklemesine" izin verin, tıpkı tmux ya da ekranın yaptığı , ondan "ayırın" ve arka plana geri koyun.

Bölüm VII

Ekler

VII.1 Örnek yapılandırma dosyası

Bu, görev yöneticiniz için bir yapılandırma dosyasının nasıl görünebileceğidir:

```
programlar
        :
     nginx:
        cmd: "/usr/local/bin/nginx -c /etc/nginx/test.conf" numprocs: 1
        umask: 022 workingdir:
        /tmp autostart: true
        autorestart: beklenmeyen
        exitcodes:
            - 0
            - 2
        startretries: 3
        başlama zamanı: 5
        durma sinyali: TERM
        durma zamanı: 10
        stdout:   /tmp/nginx.stdout
        stderr:   /tmp/nginx.stderr
        env:
            STARTED_BY: taskmaster CEVAP:
            42
    vogsphere:
        cmd: "/usr/local/bin/vogsphere-worker --no-prefork" numprocs: 8
        umask: 077 workingdir:
        /tmp autostart: true
        autorestart: beklenmedik
        exitcodes: 0
        startretries: 3
        başlama zamanı: 5
        durma sinyali: USR1
        durma zamanı: 10
        stdout: /tmp/vgsworker.stdout
        stderr: /tmp/vgsworker.stderr
```

VII.2 Süpervizör denemesi

supervisor PyPI'da bir Python paketi olarak mevcuttur. Denemek için en basit yol, evinizde bir virtualenv oluşturmak, onu etkinleştirmek ve ardından "pip install supervisor" ile 'ı kurmaktır. Daha önce python yüklemeniz gerekebilir, Homebrew'da mevcuttur.

Daha sonra bir veya iki programı yönetmek için bir yapılandırma dosyası oluşturabilir, başlatabilir
supervisord -c myconfigfile.conf, ardından supervisorctl kullanarak onunla etkileşime geçin.

Unutmayın ki supervisor olgun, zengin özelliklere sahip bir programdır ve taskmaster ile yapmanız gerekenler daha az karmaşıktır, bu yüzden onu sadece bir ilham kaynağı olarak görmelisiniz. Örneğin, supervisor kontrol kabuğunu ana programla UNIX-domain soketi üzerinden iletişim kuran ayrı bir süreçte sunarken, sizin yalnızca ana programda bir kontrol kabuğu sağlamanız .

Programınızın belirli bir durumda nasıl davranması gerektiği veya bazı seçeneklere nasıl bir anlam vermeniz gerektiği konusunda şüpheleriniz varsa... şüpheye düştüğünüzde, süpervizörün yaptığı gibi yapın, yanlış yapamazsınız.

Bölüm VIII

Teslim ve akran düzeltmesi

Çalışmanızı her zamanki gibi GiT deponuza gönderin. Yalnızca deponuzdaki çalışmalar notlandırılacaktır.

Herkese iyi şanslar ve yazar dosyanızı unutmayın!