Pardus Açılış Sistemi

1 Eylül 2006

İçindekiler

1 Giriş

İşletim sistemi çekirdeğinin çalışmaya başladığı andan, kullanıcının giriş yapabileceği ana kadar yapılan işlemler, açılış (init) sürecidir. Dosya sistemlerinin bağlanması, donanım sürücülerinin yüklenmesi, sistem servislerinin başlatılması ve grafik arabiriminin çalıştırılıp, giriş ekranının gösterilmesi bu sürecin kapsamındadır.

Bilgisayar kapatılacağı zaman, servislerin durdurulması, bağlanmış dosya sistemlerinin ayrılması da aynı sistem tarafından yürütülür.

Pardus 1.1 sürümü ile birlikte kabuk tabanlı klasik açılış sistemini terkedip, Müdür adıyla geliştirdiğimiz yeni bir sisteme geçtik. Bu belge, daha hızlı açılış sağlayan, Python diliyle yazıldığı için geliştirmesi ve bakımı daha kolay olan bu yeni sistemin kullanımını anlatmaktadır.

Açılış süreci, temel kullanım ve ayarlar bölümleri kullanıcılara yönelik olup, temel kavramları ve özel durumlarda gerekebilecek bilgileri anlatmaktadır.

Paketleme bilgileri ve servis betikleri bölümleri, sistem yöneticilerine ve entegratörlere yönelik olup, pisi paketi yaparken, yada üçüncü parti bir programa pardus desteği verirken gerekli olan bilgileri vermektedir.

Teknik yapı bölümü, programcılara yönelik olup, incelemek ve geliştirmek isteyenlere müdürün bileşenlerini ve teknik ayrıntılarını anlatmaktadır.

2 Açılış Süreci

Bilgisayar açılınca, anakart üzerindeki BIOS (Basic Input/Output System, Temel Giriş/Çıkış Sistemi) adı verilen yazılım çalışmaya başlar. Kendi iç denetleme ve donanım hazırlama sürecini tamamladıktan sonra, öntanımlı açılış aygıtından (bu bağlı bir harddisk, CD okuyucu, ya da USB disk olabilir), MBR (Master Boot Record, Ana Önyükleme Kaydı) adı verilen ve söz konusu kayıt ortamının en başında bulunan ufak önyükleyici yazılımı yükler ve çalıştırır.

Önyükleyici

Pardusun kurulumu sırasında, öntanımlı önyükleyici yazılımı Grub, kurulum yaptığınız diskin başına yerleştirilecek, ve diskte başka işletim sistemleri varsa, bunlar da Grub ayar dosyasına yazılacaktır. Böylece bilgisayarı açınca karşısınıza çıkacak Grub menüsünden hangi işletim sistemini açmak istediğinizi seçebilirsiniz.

Önyükleyici, BIOS yordamlarını kullanarak seçtiğiniz işletim sistemi çekirdeğini belleğe yükletir ve çalışmayı çekirdeğe devreder.

Linux çekirdeği, içereceği donanım sürücüleri seçilerek özelleştirilmiş biçimlerde derlenebilmektedir. Bu sürücüler çekirdeğin içine dahil edilebildiği gibi, gerektiği

anda yüklenecek modüller olarak da sistemde bulunabilirler. Kullanılmayan sürücüler çekirdeğin boyutunu ve bellek kullanımını arttırdıkları için, Pardusta donanım sürücülerini olabildiğince ayrı modüller halinde dağıtıyoruz. Bu şekilde, çalışan bir sistemde bir modülü çıkartıp, yeni sürümünü yükleyerek kolayca güncellemek de mümkün olmaktadır.

Önyükleyiciden çekirdeğe geçiş sırasında, depolama aygıtının sürücüsünü içermeyen bir çekirdek, önyükleyici, aygıtı BIOS aracılığıyla kullandığı için kolayca yüklenebilir, ama yönetimi BIOS'tan alınca bu aygıta erişemeyeceği için açılış sürecine devam edemez. Çok sayıda depolama aygıtının sürücülerini çekirdek içine koymak pratik olmadığı için, bu soruna çare olarak initrd (init ram disk, açılış bellek diski) denilen ikinci bir dosya, önyükleyici tarafından çekirdekle birlikte belleğe yüklenir. Bu dosya içinde, ufak bir dosya sistemi şeklinde, daha az yaygın olan depolama aygıtı sürücüleri bulunmaktadır. Çekirdek buradan gerekli sürücü varsa yükledikten sonra, bu dosyayı bellekten atar.

Çekirdek

Çekirdek donanım yönetimini ele alıp, hazırlıklarını tamamladıktan sonra, kök (root) dizin olarak belirtilen yerdeki dosya sistemini açarak, burada /sbin/init programını çalıştırır. Bu program temel süreç olarak, diğer süreçleri yönetir. Belli olaylar (açılış, kapatma komutu, güç yönetimi, vs) olduğunda ayar dosyasında belirtilen komutları çalıştırır.

Müdür

Müdürün ana kısmı olan /sbin/mudur.py komutu, açılış sırasında init tarafından çağrıldığında, aygıt yönetimi ve otomatik donanım tanıma (hotplug) için kullanılan udev servisini başlatır. Bu servis bilgisayara USB disk yada CD gibi bir şey taktığınızda, bu olayla ilgilenen programların otomatik çalıştırılması, takılan aygıta erişim için gereken /dev dizini altındaki dosyaların oluşturulması gibi işleri yapar. Müdür daha sonra depolama aygıtlarındaki dosya sistemlerini kontrol eder ve bağlar. Müdürün diğer bir parçası olan /sbin/muavin.py aracılığıyla, bilgisayarda hazır bulunan (coldplug) ve sürücüsü initrd içinde olmayan donanımların sürücülerini yükletir. Muavin yeni bir donanım takıldığında da udev tarafından çağrılmakta ve sürücü yükleme işini yapmaktadır. Müdür, sistem saatini donanım saatiyle senkronize ettikten ve temel ağ ayarlarını da yaptıktan sonra, Pardus'un sistem yapılandırma aracı olan Çomar'ı başlatır. En son olarak Çomar'a servisleri çalıştırması komutunu verir.

Kapanış sırasında çağrıldığında ise, önce Çomar'a servisleri durdurması komutunu verir. Sonra sistem saatini tekrar donanım saati ile senkronize eder. En son olarak bağlı dosya sistemlerini düzgün bir şekilde ayırarak, veri kaybı olmadan kapanmalarını sağlar. Müdür işini bitirdikten sonra init komutu bilgisayarı kapatması yada yeniden başlatması için gerekli komutu çekirdeğe verir.

Servisler

Arkaplanda çalışarak kullanıcıya yada diğer programlara bir takım hizmetler sunan yazılımlardır. Örneğin zemberek uygulamalara Türkçe yazım denetimi desteği sağlarken, DBus masaüstü uygulamalarının iletişimini, cups ise yazıcı desteğini sağlar. Uzaktan erişim sağlayan SSH gibi sunucu yazılımları da birer servis olarak çalışmakta ve yönetilebilmektedir.

Servisler başlatıldığında, KDM (KDE Desktop Manager, KDE Masaüstü Yöneticisi) servisi, görüntüye kullanıcı giriş ekranını getirir. Buradan kullanıcı ve parolası ile giriş yapıldığında da o kullanıcının masaüstü yüklenir ve çalışmaya başlar.

3 Temel Kullanım

Açılış ve donanım tanıma sistemi büyük ölçüde otomatik çalışmaktadır.

Bilgisayarınızda hangi servislerin kurulu olduğunu görmek, bunların açılışta başlayıp başlamayacağını ayarlamak, servislerin çalışma durumunu denetlemek, gerektiğinde yeniden başlatmak gibi işler için, Pardus Yapılandırma Merkezi içinde Sistem Seçeneklerinde bulunan Servis Yöneticisi yazılımını kullanabilirsiniz.

Komut satırından ve betiklerden servisleri yönetebilmek için /bin/service komutu da bulunmaktadır.

Kurulu servisleri ve durumlarını görmek için:

```
service
```

komutunu parametresiz olarak veya list parametresi ile verebilirsiniz.

```
service openssh start
service openssh stop
service openssh restart
```

komutları ilk parametre olarak verilen servisi başlatacak, durduracak, ve durdurup tekrar başlatacaktır.

```
service openssh reload
```

komutu bazı servislerde, sunucunun durdurulmadan, ayar dosyalarını yükleyip kendisine çeki düzen vermesini sağlar. Her servis için geçerli değildir.

```
service openssh on
service openssh off
```

komutları ile aynı şekilde servisleri açıp kapayabilirsiniz. Bu şekilde açıp kapattığınız servisler start ve stop'tan farklı olarak, bir dahaki açılışta aynı durumda başlayacaklardır.

Çalışma Seviyeleri

Temel süreç görevini yürüten init komutu bir çalışma seviyesi (runlevel) kavramına sahiptir. Tarihsel olarak 0 ile 6 arasındaki bu çalışma seviyesi sistemi çeşitli durumlara getirmek için kullanılır. Bu seviyeler,

0 Sistemi kapat

1 Sistemi tek kullanıcılı moda geçir

2-5 Kullanıcı tanımlı

6 Sistemi yeniden başlat

Çalışma seviyesini değiştirmek için /sbin/telinit komutu kullanılabilir. Hangi seviyede ne yapılacağı /etc/inittab dosyasında belirlenir. Pardus'ta bu seviyeler kendilerine karşılık gelen parametre ile müdürü çağırmakta ve kullanıcı tanımlı seviyelerden yalnız bir tanesi normal çalışma seviyesi olarak kullanılmaktadır. Bu müdür parametreleri,

sysinit Temel açılış işlemlerini yap

boot Geri kalan açılış işlemlerini yap

shutdown Kapanış işlemlerini yap (seviye 0)

single Tek kullanıcılı moda geç (seviye 1)

default Servisleri çalıştır (seviye 3)

reboot Kapanış işlemlerini yap (seviye 6)

Burada ihtiyaç duyabileceğiniz bir seviye, telinit S yada telinit 1 komutuyla geçebileceğiniz tek kullanıcılı seviyedir. Bu seviyede müdür servisleri kapatır, ve yetkili (root) kullanıcı olarak giriş yaparak sistemi onarma gibi işleri yapabilirsiniz. Normal çalışmaya dönmek için telinit 3 komutunu verebilirsiniz.

Kapanış için basitçe reboot veya halt komutlarını kullanabilirsiniz, daha esnek kapanış işlemleri için shutdown komutuna bakın.

Kayıtlar

Sistem dili, klavye haritası ve yereli müdürde öntanımlı değerlere sahiptir, /etc/conf.d/mudur dosyasından ayarlanabilmektedir, ve mudur çekirdek parametresi ile de değiştirilebilmektedir. Müdür bu değerleri belirledikten sonra, kdm gibi programların aynı yerleri dolaşmadan kolayca alabilmesi için, /etc/mudur/ dizini içindeki keymap, language ve locale adlı dosyalara yazar.

Müdür, /var/log/mudur.log dosyasına karşılaştığı olayları kaydetmektedir. Örnek bir dosyadan bir kesit:

```
Aug 26 13:55:08 (up 7.65) /sbin/mudur.py sysinit
Aug 26 10:55:15 (up 14.44) /sbin/mudur.py boot
Aug 26 10:55:16 (up 15.84) /sbin/mudur.py default
Aug 26 15:58:42 (up 18221.40) /sbin/mudur.py shutdown
```

Satırlar, tarih (çalışma süresi) bilgi biçimindedir. Çalışma süresi çekirdek çalışmaya başladığı andan itibaren geçen saniye olarak hesaplanmaktadır. Açılış sırasında sistem saati donanım saati ile senkronize edilirken tarih kısmında saatte bir düzeltme olduğu için olayın zamanını daha doğru göstermek için konmuştur.

Örnekte, sekizinci saniyeye doğru müdür, temel açılış işlemleri için init tarafından çağrılmakta, onbeşinci saniyeye doğru geri kalan açılış işlemleri için çağrılmakta, ve onaltıncı saniyeye doğru ise servisleri çalıştırması için çağrılmaktadır. Yaklaşık beş saatlik bir çalışmanın sonunda ise kapanış işlemleri için devreye girmiştir.

Normal çalışmada yalnızca uyarı ve hata mesajları ile çağrılma durumları bu dosyaya yazılmaktadır. Eğer müdürün tüm çıktısının kaydedilmesini istersenız, /etc/conf.d/mudur içinde bir debug=1 satırıyla, ya da önyükleyiciden vereceğiniz bir mudur=debug çekirdek parametresiyle bunu açabilirsiniz.

Diğer Komutlar

/sbin/muavin.py

Aygıt sürücülerini yükleyen muavin, –debug parametresiyle çağrıldığında bulduğu aygıtlar için hangi sürücüleri yükleyeceğini listeler. Donanım tanıma ile ilgili bir aksilik olduğunda, bilgi verici olabilir. Bu sürücüleri yüklemesini istiyorsanız ise –coldplug parametresini kullanabilirsiniz.

/sbin/update-environment

Normal bir çalışmada ihtiyacınız olmayacak bu komutu, eğer elle bir program derlemişseniz kullanmanız gerekebilir.

Bazı programlar, sistemde ve kullanıcı oturumlarında ayarlanması gereken çevre değişkenlerine ihtiyaç duyarlar. Örneğin Java, JAVA_HOME adlı bir değişkende sistemdeki Java işleticisinin bulunduğu dizinin adının bulunmasını ister. QT gibi, kitaplıklarını /usr/lib gibi standart konumların dışında /usr/qt/3/lib gibi dizinlere koyan uygulamalar ise, yüklenecek kitaplıkların bulunabileceği yerleri gösteren LDPATH gibi çevre değişkenlerine kendi dizinlerini eklemek isterler.

Program paketlerinin kolayca bu değişkenlere değer atayabilmesi için, /etc/env.d dizini kullanılmaktadır. Bir pisi paketi buraya bir dosya koyduğunda, Çomar aracılığıyla çalıştırılan update-environment komutu, bu dosyaları okuyup sistem kabuğu için bir profil dosyası oluşturmakta, eğer kitaplık yolları değişmişse, /etc/ld.so.conf dosyasını yeni yolları içerecek şekilde değiştirmekte, ve /sbin/ldconfig komutunu çağırarak bu yeni dizinlerdeki kitaplıkların taranarak dinamik kitaplık yükleyicinin /etc/ld.so.cache dosyasında indekslenmesini sağlamaktadır.

/sbin/update-modules

Normal bir çalışmada ihtiyacınız olmayacak bu komutu, eğer elle bir çekirdek modülü derlemişseniz kullanmanız gerekebilir.

Çekirdek modülleri içeren pisi paketleri kurulduktan sonra, Çomar aracılığıyla çağrılan bu komut, /sbin/depmod komutunu çalıştırarak, çalışmakta olan çekirdek için, bu modüllerin hangi aygıtları desteklediklerini ve birbirlerine olan bağımlılıklarını gösteren modules.*map, modules.dep, modules.alias dosyalarını oluşturmaktadır. Her bir çekirdek için /lib/modules/2.6.16.24-49/ gibi dizinlerde bulunan bu dosyalar, muavin ve modprobe komutu tarafından kullanılmaktadır.

Ayrıca eğer paket /etc/modules.d/ dizini altına bir dosya yerleştirmişse, bu dosyalar taranarak, modüller için yükleme şekil ve parametrelerini gösteren /etc/modprobe.conf dosyası güncellenmektedir.

4 Ayarlar

Önyükleyici Ayarları

Önyükleyicinin menüsündeki işletim sistemleri, ve çekirdeğin hangi seçeneklerle çalıştırılacağı, /boot/grub/grub.conf dosyasından ayarlanmaktadır. Grub programının belgelerinde bu ayarlarla ilgili detaylı bilgi alabilirsiniz. Bilgisayar açılırken, Grub menüsü çıktığında, açacağınız sistemin üzerindeyken e tuşuna basarak da, bu ayarları o açılışa mahsus olarak değiştirebilirsiniz.

İşletim sistemi girdilerindeki kernel satırları, yüklenecek Linux çekirdeğini ve parametrelerini belirtmektedir. Söz dizimi

```
kernel (aygıt)/dosya prm1=değer prm2=değer1,değer2 prm3 ...
```

biçimindedir. İlk bilgi yüklenecek çekirdeğin bulunduğu depolama aygıtı ve çekirdeğin dosya adıdır. Daha sonra bu çekirdeğe verilen parametreler gelmektedir. Parametreler tek başına, bir değere sahip, yada birden fazla değerli olabilir.

Gerekebilecek bazı çekirdek parametreleri:

quiet Öntanımlı gelen bu seçenek çekirdeğin donanımları tararken bulduğu teknik bilgileri ekrana basmasını önler, müdür çalışmadan önce oluşan bir donanım sorunu varsa, bu seçeneği kaldırarak daha fazla bilgi alabilirsiniz.

splash Açılışta grafik logo gösterilmesini sağlar, metin ekranda açılış için, bu seçeneği ve console seçeceğini kaldırın.

noacpi Standartlara uygun olmayan bazı BIOS'larda bu seçenekle çekirdeği çalıştırmak mümkün olmaktadır.

mudur Bu seçenekle müdürün bazı ayarlarını değiştirebilirsiniz. Müdür ayarları bölümünde daha ayrıntılı anlatılmıştır.

Tüm çekirdek parametrelerini ve açıklamalarını, kernel-source paketini kurduktan sonra /usr/src/linux/Documentation/kernel-parameters.txt dosyası içinde bulabilirsiniz.

Müdür Ayarları

Müdür init tarafından çalıştırılmaktadır, genellikle değiştirmenize gerek olmayan init ayarları /etc/inittab dosyasında bulunmaktadır.

Müdürün kendi ayarları ise /etc/conf.d/mudur dosyasından ve önyükleyiciyle çekirdeğe verilen mudur parametresinden yapılabilmektedir. Çekirdekten verilen ayarlar dosyada yazanların yerine kullanılır.

safe Bu seçenek verilirse, sorun yaratabilecek bazı donanım işlemleri yapılmayacak, servislerden ise yalnızca giriş ekranını çıkartan kdm ve bağlı olduğu servisler çalıştırılacaktır.

language Açılışta ve giriş ekranında kullanılacak olan sistem dilini belirler. Öntanımlı değeri Türkçe'dir. Kurulum başka bir dilde yapılmışsa, ayar dosyasında bu dil seçilidir.

keymap Eğer sistem dilinin öntanımlı klavye haritasını kullanmak istemiyorsanız, bu seçenekle başka bir harita belirleyebilirsiniz.

clock Donanım saatiniz Greenwich Merkezi Zamanına (GMT) göre ayarlıysa bu değeri UTC yapın, eğer donanım saatiniz yerel zamana göre ayarlıysa da local yapın. Öntanımlı değeri yerele ayarlı saattir. Bu seçeneği yalnızca ayar dosyasından verebilirsiniz.

tty_number Kullanmak istediğiniz metin ekran konsol sayısı öntanımlı altı değerinden farklıysa bu seçenekten belirtebilirsiniz. Bu seçeneği yalnızca ayar dosyasından verebilirsiniz.

debug Bu seçenek verilirse müdür tüm çıktılarını /var/log/mudur.log dosyasına yazacak, hata düzeltme bilgileri sağlayacaktır.

Bazı örnekler:

```
mudur=safe,debug
mudur=language:tr,keymap:trf
```

Donanım Ayarları

Açılışta otomatik olarak yüklenemeyen, ya da ne olursa olsun yüklenmesini istediğiniz çekirdek modüllerini /etc/modules.autoload.d dizini içindeki kernel-x.y.z biçimindeki dosyalara her satıra bir modül ismi biçiminde yazabilirsiniz. Bu dosyalara bakılırken, mesela 2.6.16.24 çekirdeği çalışıyorsa sırasıyla varsa kernel-2.6.16 ve kernel-2.6 dosyalarına bakılacaktır.

Aynı şekilde, eğer otomatik yüklenmesi sorunlara yol açan modüller varsa, /etc/hotplug/blacklist dosyasına, ya da /etc/hotplug/blacklist.d/ dizini içindeki herhangi bir dosyaya, gene bir satıra bir modül ismi gelecek şekilde yazabilirsiniz.

Modüllere özel parametreler vermek için /etc/modules.d içinde bir dosya oluşturup, modprobe.conf belgesinde anlatılan söz dizimi ile alias, install, options gibi komutları verebilirsiniz. Müdürün bir parçası olan update-modules komutu bu dosyaları dolaşıp bir /etc/modprobe.conf dosyası oluşturmaktadır. Komutlarınızı bu dosyaya direk yazmayın, yeni bir modül paketi yada çekirdek kurulduğunda otomatik olarak üstüne yazılacaktır.

Bir donanım takıldığında, bununla ilgili bir ayar yapmak, isim vermek, özel bir program çalıştırmak gibi şeyler yapmak için, /etc/udev/rules.d/ dizinine bir kural dosyası koymanız yeterlidir. Buraya ismi .rules sonekiyle biten yeni bir dosya konduğunda otomatik olarak devreye girmektedir. Kural dosyasının söz dizimi için udev belgelerine bakın.

Müdür açılış sırasında bazı harddisk parametrelerini ayarlayabilmektedir. Bu tür bir ihtiyacınız varsa /etc/conf.d/hdparm dosyasında istediğiniz disklere istediğiniz parametreleri verebilirsiniz. Parametrelerin neler olduğunu öğrenmek için hdparm komutunun belgesine bakın. Bazı parametreler sorunlara yol açabileceğinden dikkatli olmanız tavsiye edilir. Dosyanın söz dizimi, aygıt adı= "parametreler" biçiminde satırlardan oluşmaktadır. Aygıt adı /dev/ dizini altındaki herhangi bir harddisk aygıtı yada bağlı bulunan tüm diskler için all olabilir.

Diğer Ayarlar

Çekirdeğin kapatılınca kaybolan ve /proc/sys dosya sistemi aracılığıyla yapılan ayarları için, /etc/sysctl.conf dosyasını kullanabilirsiniz. Örneğin ağ ayarlarından tüm makinelere gönderilen pinglere cevap vermeyi kapatmak için:

```
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

gibi bir satır ekleyebilirsiniz. Açılışta bu dosya okunurken, /proc/sys/net/ipv4/ dizinindeki icmp_echo_ignore_broadcasts dosyasına 1 değeri yazılarak çekirdeğin bu özelliği çalıştırılacaktır. Daha fazla bilgi için sysctl komutuna ve çekirdek belgelerine bakabilirsiniz.

Servislerin ayar dosyaları da genellikle /etc/conf.d/ dizini içinde durmaktadır.

Bunlardan önemli bir tanesi local.start ve local.stop dosyalarıdır. İlki açılışta, ikincisi ise kapanışta çalıştırılan birer kabuk betiğidir. Pardus tarafından kullanılmayan ve tamamen kullanıcılara ait olan bu dosyaların içine, İhtiyaç halinde kendi özel komutlarınızı koyabilirsiniz.

5 Paketleme Bilgileri

Pisi paketi yaparken, paketlediğiniz programı açılış sistemine entegre etmenizi gerektiren çeşitli durumlarla karşılaşabilirsiniz.

Çekirdek Modülü Paketleri

Çekirdek modülleri içeren bir paket yaparken, paketinize /lib/modules/2.6.xx.xx/ dizininin hemen altındaki modules.* dosyalarını almamalısınız. Aksi takdirde dosya çakışmaları ortaya çıkabilir. Modül paketleri kurulurken bu dosyalar sistemdeki modüller taranarak update-modules komutu tarafından düzgün bir şekilde oluşturulacaktır.

Eğer modülün inşası sırasında bu dosyalar paket kurulum dizinde oluşuyorsa, actions.py içinde basitçe

```
pisitools.remove("/lib/modules/*/modules.*")
```

ile silebilirsiniz.

Eğer bir modülün her açılışta mutlaka yüklenmesini istiyorsanız, Çomar System.Package betiğinizde, ihtiyacınıza göre /etc/modules.autoload.d/ dizini içinde kernel-2.x yada kernel-2.x.y dosyasının sonuna bir satır olarak ekletebilirsiniz. Dosyada daha önce zaten böyle bir satır olup olmadığını da kontrol edin!

Bir modülün donanım varsa bile otomatik yüklenmesinin önüne geçmek gerekiyorsa ise, /etc/hotplug/blacklist.d/ dizini içine kendi paket adınızda bir dosyaya modül adlarını satır satır yazıp koymalısınız.

Modüllerinizin parametreleri, gene paket adınızı taşıyan bir dosyada, /etc/modules.d/ dizinine koymalısınız. Kurulumda update-modules, gene otomatik olarak /etc/modprobe.conf dosyasını güncelleyecektir.

Donanımla İlgili Paketler

Donanım takılma olaylarını yakalamak ve bunun üzerine bir şeyler yapmak için udev sistemini kullanın. Hazırlayacağınız kural dosyasını /etc/udev/rules.d/ dizinine koymanız yeterli olacaktır.

Servis Paketleri

Eğer arkaplanda çalışıp hizmet verecek bir program paketliyorsanız, ya da açılış sırasında çalıştırmanız gereken komutlar varsa, bir Çomar System.Service betiği yazıp paketinize eklemeniz gerekmektedir. Bu betikle ilgili bilgiler bir sonraki bölümde anlatılmıştır.

Çevre Değişkenleri

Çevre değişkenlerinizi kabuğa vermek ve kitaplık, komut gibi yollara eklemeler yapmak istiyorsanız, /etc/env.d/ dizinine,

```
LALA=''deneme''
LDPATH=''/usr/lala/lib''
```

biçiminde satırlardan oluşan bir dosya koymanız yeterlidir. Değişkenlerinizin önceliğini belirlemek için dosya adını XX, 00 ile 99 arasında bir sayı olacak şekilde XXpaketadı biçiminde verin. Bu dosyalar işlenirken küçük sayıdan büyüğe gidilmekte, ve sonra gelen değer öncekinin üstüne yazılmaktadır. Fakat yol belirten PATH, LDPATH, MANPATH, vb değişkenlerde farklı bir mantık izlenmekte, bu değişkenler birbirlerine : işareti ile eklenmektedir.

6 Servis Betikleri

Çomar System.Service Sınıfı

Müdür için yazılmış servis betikleri Çomar'da bu sınıfa kayıt olurlar ve aşağıdaki metotları sağlarlar:

System.Service.info

Servis hakkında bilgi döndürür. İlk satır servisin tipi, ikinci satır durumu, son satır da yerelleştirilmiş olarak servisin adını verir.

Üç tip servis vardır:

server Apache, OpenSSH, Postfix gibi dış bilgisayarlara hizmet veren web, kabuk, eposta sunucu gibi servislerin başlatma betikleri.

local Pardus masaüstünün çalışabilmesi için yerel programlara hizmet veren DBus, HAL, Zemberek gibi servislerin başlatma betikleri.

script Bir servis başlatmayan, yalnızca açılışta belli işlemleri yerine getiren betikler.

Bir servis dört durumda olabilir:

on Servis sürekli olarak açıktır ve şu anda çalışmaktadır.

started Servis kapalıdır ama kullanıcı tarafından çalıştırılmıştır.

stopped Servis açıktır ama bir sebepten dolayı durdurulmuş yada çalışamamıştır.

off Servis kapalıdır.

Servis tipi ve adı, kullanıcı arayüzlerinde göstermek için tasarlanmıştır.

System.Service.start

Servisi başlatır.

System.Service.stop

Servisi durdurur.

System.Service.reload

Eğer servis destekliyorsa durdurmadan ayarlarını tekrar yükleyip güncellemesini sağlar.

System.Service.setState (state)

Servisin durumunu değiştirir. Eğer state parametresi "on" verilirse, servis çalıştırılır ve sürekli olarak açık konuma getirilir. Böylece bilgisayarı yeniden başlattığınızda servis otomatik olarak açılacaktır. State "off" verilerek servis kapatılıp çalışmayacak şekilde ayarlanır.

System.Service.ready

Servis eğer "on" olarak ayarlanmışsa, start metodunu çağırarak başlatır. Müdür açılış sırasında tüm servislere bu çağrıyı yaparak servisleri başlatmaktadır.

System.Service.changed

Bu uyarı bir servisin durumu değiştiğinde gönderilir. Parametre olarak "started" ve "stopped" değerlerini verir.

Çomar comar.service Modülü

Servis betiklerinde sık kullanılan fonksiyonlar comar-api paketi ile gelen bu Python modülünde sağlanmıştır.

Betiklerde:

```
from comar.service import *
```

şeklinde kullandığınızda info, setState, ready metotları sizin için tanımlanmış olacaktır. Böylece yalnızca start ve stop metotları ile bir servis betiği yazabilirsiniz. Bu hazır fonksiyonlar, betik tipi ve adı bilgisini betiğinizin içinde tanımlayacağınız serviceType ve serviceDesc değişkenlerinden alacaktır.

Eğer servisinizin çalışıp çalışmadığına göre True/False döndüren bir status fonksiyonu yazarsanız, bu da servisinizin durumunu info çağrısında raporlarken kullanılacaktır.

Bu modülü import ettiğiniz halde, bu fonksiyonları kendiniz de tanımlayabilirsiniz, bu durumda betiğiniz içindekiler çalışacaktır.

Modülde ayrıca şu yardımcı fonksiyonlar bulunmaktadır:

run()

Parametre olarak verilen komutu yeni bir kabuk açmadan çalıştırır. Komutun dönüş değerini döndürür.

Örnek:

```
run("/sbin/start-stop-daemon --start --quiet --exec /usr/sbin/cpufreqd")
```

checkDaemon()

Verilen pid dosyasının gösterdiği servisin çalışıp çalışmadığına bakar.

Örnek:

```
def status():
    return checkDaemon("/var/run/kdm.pid")
```

waitBus()

Dosya adı verilen unix sokete bağlanılabiliyorsa True aksi halde False değerini döndürür. Bir servisin başlatılması ile istemcileri dinlemeye başlaması arasında bir süre geçiyorsa bu komutla bekleyebilirsiniz.

Opsiyonel timeout parametresi ile maksimum bekleme süresini (öntanımlı 5 saniye), yine opsiyonel wait parametresi ile de deneme aralıklarını (öntanımlı 0.1 saniye) ayarlayabilirsiniz. Eğer stream değil de datagram tipinde bir sokete bağlanmanız gerekiyorsa False değerli bir stream parametresi vermelisiniz.

Örnek (kütük servisi hazır mı?):

```
waitBus("/dev/log", stream=False)
```

Örnek (dbus servisi hazır mı?):

```
waitBus("/var/lib/dbus/system_bus_socket")
```

loadEnvironment()

Çevre değişkenlerini /etc/profile.env dosyasından okuyup çalışan süreç için geçerli kılar. Böylece buradaki değişkenleri kullanan servisleri, ayrı bir kabuk başlatmadan çalıştırabilirsiniz.

config

Bu bir Python sözlük değişkeni olup, /etc/conf.d/betikpaketadı dosyasındaki isim=değer biçimindeki ayarları içerir. Eğer kullandığınız ayar dosyası adı betiğinizin paket adından farklıysa serviceConf değişkenine dosya adını verebilirsiniz (gene /etc/conf.d/ içinde bir dosya olmak üzere).

7 Teknik Yapı

Pardus açılış sisteminde, olabildiğince modern bir yapı kullanıyoruz. Aygıt dosyası oluşturma, donanım tanıma programlarını tetikleme işleri tamamen udev tarafından yapılmakta, çekirdek ile udev arasında eski hotplug sistemi yerine netlink soket bağlantısı kullanılmakta, donanıma ait modülün bulunması sysfs dosya sisteminden gelen üretici/ürün gibi bilgilerle, kurulu olan modüllerin desteklediklerinin listesi karşılaştırılarak dinamik olarak yapılmakta, açılış sistemi ve servis betikleri yüksek seviyeli Python diliyle yazılmakta.

Temel yaklaşımımız

- Donanım tanıma dinamik olmalı, kurulu sistem donanımda değişiklikler olduğunda, yeniden elle ayar istemeden, çalışmaya devam edebilmelidir,
- Programların sistemle entegrasyonu kolay ve problemsiz olmalıdır,
- Açılıştaki erken bir sorun sistemi kurtarmayı çok zorlaştırmakta olduğu için, temel işlemler çok sağlam olmalı, bir hata çıktığında bunu düzeltmek için çaba göstermeli, daha güvenli açılış seçenekleri sağlamalıdır,
- Sistem esnek olmakla birlikte, yeterli performansı da sağlamalıdır.

Tasarım Kararları

Python Nedeni çok açık. Kabuk, awk, perl, sed betiklerinden oluşan çorba yerine, bakımı ve okuması kolay, ihtiyacımız olan karakter dizisi (string) ve liste, sözlük gibi veritiplerini çok güçlü bir şekilde destekleyen, fonksiyonel ve nesne tabanlı, hızlı ve temiz program geliştirilebilen bir dil seçtik. Kod büyüklüğü büyük ölçüde kısalırken, özelliklerimiz ve hızımız arttı.

Disk hızı Harddisklerin oldukça yüksek olan okuma hızı, harddiskin okuyucu kafasının gereksiz dolaşımıyla büyük ölçüde düşüyor. Bu aşırı dolaşımın sebepleri arasında dosya sisteminin fragmentasyonu, çok sayıda dosyaya erişim, okuma/yazma işlemlerinin farklı dosyalar üzerinde içiçe yapılmaları, okuma işleminin tek seferde değil, ara ara parçalar halinde yapılması var. Bunu önlemek için, müdürün ana kısmını tek bir dosya (mudur.py) olarak tuttuk, dışardan çağırdığımız programları tam yolları ile (/sbin/mount şeklinde) çağırdık, Python yorumlayıcısının bir defada yüklenince birçok modülü (os, sys, string, time, glob, vb) sağlıyor olmasının avantajını kullandık, kabuğun zayıflığı nedeniyle çok kullanılmak zorunda kalınmış geçici dosyalardan kaçındık.

Çomar Klasik /etc/init.d/ altındaki kabuk betikleri yerine Python ile yazılmış Çomar betikleri kullanmayı seçtik. Dezavantajı servis betiklerinin yeniden yazılmasının gerekmesi. Dağıtımda servis betiği gerektiren uygulama

sayısının %5 altında olması ve bu betikleri yazmanın, Python gibi kolay bir dille birer start ve stop metodu yazmaktan ibaret olması, ve Pisi paketçilerinin zaten Python ile çalışıyor olması, bu dezavantajı önemsiz kılıyor. Hazır Çomar yapısını kullanmak, servisleri yöneten araçlar yapmayı, ve servis yönetimi yetkilerini kullanıcılara dağıtabilmeyi çok kolaylaştırmakta.

init Başka açılış sistemi denemelerinin (initng, depinit, vs) tersine, temel süreci değiştirmekten kaçındık. Bu süreç kendi işini çok iyi yapıyor ve yılların sağlamlığına sahip. Paralelliği temel açılış bitmeden zaten başlatamıyoruz, şu anda Çomar ile servisleri paralel çalıştırıyoruz, müdür de init tarafından çalıştırılıyor.

sleep En çok kaçındığımız komut. Kafadan bir süre uyuyup, beklediğimiz olayın tamamlandığını varsaymak yerine, o olayın gerçekten olup olmadığını kontrol ediyoruz. Mesela syslogd başlattıktan sonra, işlemlere devam etmeden bir saniye beklemek yerine, /dev/log soketinin açılmasını bekliyoruz, böylece süreçlerimiz hem daha sağlam, hem de daha hızlı oluyor.

udev Hemen hemen tüm dağıtımlar tarafından kullanılıyor. Klasik diskte tutulan sabit /dev dizinine ve bir süre çekirdeğe dahil olup sonra atılan devfs sistemine göre, çok temiz ve esnek.

muavin Donanım tanıma sırasında udev ile modprobe arasında bizim muavin programımız bulunuyor. Çekirdekten gelen MODALIAS değerini direk modprobe etmemekteki amacımız, araya daha esnek bir kontrol sokabilmek. MODALIAS, özellikle ISAPNP gibi veri yollarında henüz başarılı değil. Muavin ayrıca, CPU frekans modülleri gibi, daha sezgisel metotlarla bulunan modüllerle de ilgileniyor.

8 Belge Geçmişi

- İlk sürüm (2006-08-25), Gürer Özen