Katedral ve Pazar

TMMOB
ELEKTRİK MÜHENDİSLERİ ODASI

KATEDRAL VE PAZAR

"The Cathedral and the Bazaar"

Eric Steven Raymond

Çevirenler: Cihan Gerçek; Aydın Bodur

EMO Yayın No: GY/2008/6

1.Basım, Ankara, Ağustos 2008

ISBN: 978 9944 89 598 9

Elektrik Mühendisleri Odası

Ihlamur Sokak No:10 06420 Kızılay/Ankara

Tel: (312) 4253272 **Faks:** (312) 4173818

e-posta: emo@emo.org.tr **web:** http://www.emo.org.tr

001.5 Ray
Ray, Eric Steven
Katedral ve Pazar = *The Cathedral and the Bazaar* : Çeviren Cihan Gerçek, Aydın Bodur-- 1.bs. -- Ankara : Elektrik Mühendisleri Odası Yayınları, 2008.
50 s. ; 24 cm
(Emo Yayınları ; - GY/2008/6) 978-9944-89-598-9

Program geliştirme

EMO adına Yayına Hazırlayan: Aydın Bodur
Kapak Tasarım: Nahide Akkuş

Baskı
Başak Matbaacılık ve Tanıtım Hiz. Ltd. Şti.
Tel. 0312 384 27 61 Ankara

BİLEŞİM YAYINEVİ

Katedral ve Pazar

Eric Steven Raymond
Thyrsus Enterprises [http://www.tuxedo.org/~esr/] *<esr@thyrsus.com>*
Version 3.0

Türkçe'ye Çevirenler: Cihan Gerçek, Aydın Bodur –Temmuz 2008

Orijinal Eser:
The Cathedral and the Bazaar
Eric Steven Raymond
Thyrsus Enterprises [http://www.tuxedo.org/~esr/]
<esr@thyrsus.com>
Version 3.0

Te'lif Hakları

$Tarih: 2002/08/02 09:02:14$

Revizyonlar

Revizyon 1.57 11 Eylül 2000 esr
Yeni Başlık "Karışıklığı kaç gözlem giderir"
Revizyon 1.52 28 August 2000 esr
MATLAB, Emacs'a paralel bir destektir. Corbatoó ve Vyssotsky bunu 1965'te anlamıştı.
Revizyon 1.51 24 Ağustos 2000 esr
İlk DocBook versiyonu. 2000 sonbaharında, zaman duyarlıklı malzeme üzerinde küçük değişiklikler.
Revizyon 1.49 5 Mayıs 2000 esr
Mühlet ve tarifelere HBS notu eklendi.
Revizyon 1.51 31 Ağustos1999 esr
O'Reilly'nin basıldığı ilk versiyon.
Revizyon 1.45 8 Ağustos 1999 esr
Snafu İlkesi, pazarın esası ve gelişimi ile ilgili (ön)tarihsel örneklere ait dipnotlar eklendi.
Revizyon 1.44 29 Temmuz 1999 esr
Tasarım alanının keşfine yönelik olarak pazarların yararlılığıyla ilgili incelikleri içeren "Yönetim ve Maginot Hattı Hakkında" bölüm eklendi; Epilog büyük ölçüde genişletildi.
Revizyon 1.40 20 Kasım 1998 esr
Halloween Belgeleri temelinde Brooks'la ilgili düzeltme eklendi.

Revizyon 1.39 28 Temmuz 1998 esr
RMS'in inandırıcı argümanları sonucunda, Paul Eggert'in GPL-pazar grafiğini kaldırdım.
Revizyon 1.31 10 Şubat 1998 esr
"Epilog: Netscape Pazar'ı Kucaklıyor" eklendi
Revizyon 1.29 9 Şubat 1998 esr
"Serbest yazılım" değiştirilerek "açık kaynak" yapıldı.
Revizyon 1.27 18 Kasım 1997 esr
Perl Konferansı anekdotu eklendi.
Revizyon 1.20 7 Temmuz 1997 esr
Bibliyografya eklendi.
Revizyon 1.16 21 Mayıs 1997 esr
Linux Konferansında ilk resmi sunum.

Burada, başarılı bir açık_kaynak projesi olan fetchmail, yani Linux'un tarihinde ortaya konulan yazılım mühendisliğiyle ilgili, şaşırtıcı kuramların sınanması denilebilecek bir çalışma masaya yatırılıyor. Söz konusu kuramlar, iki gelişme tarzı temelinde ele alınıyor. Biri, ticari alemin büyük kısmının modeli olan "katedral"; diğeri de onun karşısında yer alan ve Linux dünyasının modeli olan "pazar". Önce, bu modellerin yazılıma özgü hata ayıklama (arıza giderme, *[:debuging]*) süreciyle ilgili karşıt varsayımlardan kaynaklandığı gösteriliyor. Ardından, Linux deneyiminden gelen ve bireycil unsurların kendini düzelten diğer sistemleriyle üretken benzerlikler gösteren "*yeterince gözlem yapıldığında, bütün hatalar yüzeyseldir*" şeklindeki tez ileri sürülerek; yazılımın geleceğiyle ilgili muhtemel gelişmelere yönelik sonuçlara varılıyor.

EMO'dan Türkçe Çevirisi için

Sunuş

"Richard Stallman, 1971 yılında MIT'nin "Artifical Intelligent" labarotuvarında çalışmaya başladığında, oradaki durumu şöyle anlatıyor"...Çalışmaya başladığımda hemen yazılım paylaşım grubunun bir parçası oldum ve uzun bir zaman birlikte çalıştık. Labarotuvarın şefi olan HACKER yapının assembler'da tasarım ve yazılım işini yapmaktaydı....." diye anlatır bu süreci.

Burada karşımıza iki kavram çıkmaktadır. Birincisi ***yazılım paylaşımı****, diğeri ise* ***HACKER****'dır. Bu kavramları inceleyerek bu günü irdelemek sanırım daha sağlıklı olacaktır.*

Yazılım paylaşımı ya da bugünkü ifadeyle ***özgür yazılım*** *[:Freesoftware] yazılım dünyasında devrim yaratan bir sürecin felsefesini içeren bir kavramdır. Bu olgunun kökeni yukarıda belirtildiği gibi 70'li yıllar hatta 40'lı yılların sonuna kadar gitmektedir. 40'lı yılların sonunda ağırlığını fizikçilerin oluşturduğu yazılım grupları İngiltere ve ABD'de çalışmalarını yürütürken kendilerini REAL PROGRAMMER olarak nitelemekteydiler ve en önemli özellikleri tüm bilgi birikimlerini sınırsız bir şekilde paylaşmalarıydı. Bu anlayış ve çalışma tarzı 80'li yılların ilk yarısında kendini freesoftware akımında buldu. Bu dönemde HACKER kavramı her yerde ve her düzeyde duyulmaya başlandı. Saygı duyulan ve yazılımcıların kendilerine zevkle yakıştırdıkları bu kavram CIA ve FBI'ın çarpıtmaları sonucu CRACKER'lar ile aynı kurguda anılır oldu.*

Neydi özgür yazılımı ve hacker kavramını geleneksel olandan ayıran? Hacker kavramı ve hacker etiği temelde kapitalizmin protestan etiğine kökten bir başkaldırıdır. Bilgi çağında yeni bilgi en etkili olarak, esprili ve insanın kendi bireysel/özgür ritminde çalışması ile üretilebiliyor. Bu nedenle açık model sadece etik anlamda doğru olmakla kalmıyor, uygulamada da çok güçlü ve üretken olarak kendini gösteriyor. Network ve İnterneti birleştirerek NET diye tanımlarsak, Net'i ve Linux'u olanaklı kılan işte bu özgür yazılım kavramıdır ve onun etik değerleridir.

Torvalds, Linux üzerinde çalışmaya 1991'de, Helsinki üniversitesinde bir öğrenci iken başladı. Önce evindeki bilgisayara Andrew Tanenbaum'un minix'ini kurdu. Onun üzerinde çalışarak geliştirilebilir bir iskelet ortaya çıkardı. Bu

çalışmanın en önemli özelliği, başından itibaren projeye başkalarını da dahil etmesiydi. 25 Ağustos 1991'de, Net'te konu başlığı "Minix'te en çok neyi görmek istersiniz?" olan bir mesaj gönderdi ve bu mesajda "özgür bir işletim sistemi" yaptığını ilan etti. Cevap olarak bir sürü düşünce ve hatta programı test etmede yardım etme sözleri geldi. İşletim sisteminin ilk versiyonu, Eylül 1991'de Net'te kaynak kodu herkese açık olarak çıktı.

Ortak çalışma ile yeni sürümü Ekim başında hazırdı. Torvalds bunun ardından tüm yazılım dünyasına, yeni sistemi geliştirmede ona katılmaları için, daha doğrudan bir davette bulundu. Bir ay içinde diğer programcılar işe karışmışlardı. O günden bu yana, Linux network'ü şaşırtıcı ve yaratıcı bir süratle büyüdü.

Eric Raymond, ilk Net'te yayınlanmış olan ünlü denemesi "Katedral ve Pazar Yeri"nde Linux'un açık modeli ile çoğu şirket tarafından tercih edilen kapalı model arasındaki farkı, modelleri pazara ve katedrale benzeterek açıklıyor. Bir teknoloji uzmanı olmasına rağmen Raymond, Linux'un gerçek yeniliğinin teknik ile birlikte ve daha da önemli olarak sosyal yanı olduğunu vurgular. Yeni, tamamen açık ve sosyal bir şekilde geliştirilmiş olması önemli bir yenilikti. Onun kelimeleri ile katedralden pazara geçişti.

Raymond katedrali, içinde bir kişinin veya çok küçük bir grup insanın, her şeyi önceden planladığı ve sonra kendi gücüyle planı gerçekleştirdiği bir model olarak tanımlıyor. Gelişim, kapalı kapılar ardında oluşur ve diğer herkes, sadece "bitmiş" sonuçları görür. Pazar modelinde ise, düşünce üretme süreci herkese açık ve düşünce başından itibaren, test edilmek üzere diğerlerine dağıtılıyor.

Görüşlerin çeşitliliği en önemli şeydir ve düşünceler erken bir aşamada geniş çapta yayıldığından, dışarıdan yapılacak ekleme ve eleştirilere de açık bir süreç oluşur. Katedral bitmiş halde sunulduğunda ise, temelleri artık değişmez. Pazarda ise, insanlar değişik yaklaşımlar bulmaya çalışırlar ve birsinin harika bir düşüncesi olduğunda, diğerleri onu alıp üzerine inşa ederler.

Bu özgür ve açık-kaynaklı model, genel anlamda şöyle anlatılabilir: Her şey, bir sorunun ortaya çıkması veya birinin önemli bulduğu bir amaçla başlar. O sorunun çözümü veya kişinin amacını gerçekleştirmesi sadece bu düzeyde kalmaz aynı zamanda 0.1.1 sürümünü de oluşturur. Açık modelde alıcı, bu çözümü özgürce kullanma, test etme ve geliştirme hakkına sahiptir. Bu kurgu ancak ve ancak çözüme giden bilgi de (kaynak) beraberinde var ise olanaklıdır.

Özgür ve açık-kaynaklı modelde, bu hakların verilmesi iki yükümlülüğü de beraberinde getirir: aynı haklar orjinal çözüm veya iyileştirilmiş sürüm (0.1.2) paylaşıldığında devredilmelidir ve katkıda bulunanlar, herhangi bir sürümü

paylaşıldığında belirtilmelidir. Açık kaynaklı model bu haliyle Platon'un Akademia'sının bir devamıdır.

Bilimsel etik, teorilerin ortaklaşa hazırlandığı ve hataların görülüp, tüm bilim camiasının eleştirileri yoluyla aşama aşama düzeltildiği bir model gerektirir. Elbette ki bilim adamları bu modeli, salt etik nedenlerden dolayı değil, aynı zamanda bilimsel bilgiye ulaşmanın en başarılı yolu olduğu için seçtiler. Doğa hakkında bildiklerimizin tümü, bu akademik veya bilimsel modele dayanır. İlk Hacker'ların açık kaynaklı modelinin bu kadar etkili biçimde çalışmasının nedeni tutkularını gerçekleştiriyor olmalarına ve diğer yazılımcılar tarafından motive edilmelerine ek olarak büyük ölçüde, bilgi üretimi için ideal açık akademik modele uymasıdır.

Manastır modelinde ise sadece bilgi erişimini kısıtlamakla kalmayıp, otorite yanlısı olan bir modeldir. Bu modele göre yapılandırılmış bir iş girişiminde otorite, amacı belirler ve gerçekleştirilmesi için küçük kapalı bir grup insan seçer. Grubun kendisi testini tamamladıktan sonra, diğerleri sonucu olduğu gibi kabul etmek zorundadır. Sonucu başka şekilde kullanmak, "yetki dışı kullanım"a girer. Kapalı model, bir faaliyeti daha yaratıcı ve kendi hatalarını düzeltici olmasına olanak verecek şekilde inisiyatif kullanımına veya eleştiriye izin vermez.

*Hacker öğrenim modelinin asıl gücü, hacker'ın öğrenirken aslında diğerlerine de öğretmesidir. Bir hacker, bir programın kaynak kodu üzerinde çalıştığında, genellikle onu geliştirir ve başkaları da onun çalışmasından faydalanır. Kaynaklarını kontrol ettiğinde, genellikle kendi tecrübesinden edindiği yardımcı bilgileri ekler. Çeşitli sorunlar etrafında devam eden, eleştirel ve geliştirici bir tartışma oluşur. Bu tartışmaya katılmak ve bilgiyi eklemenin ödülü ise diğer katılımcılar tarafından kabul görmektir." **

Elektrik Mühendisleri Odası başından beri destek olmaya çalıştığı açık kaynak kodlu yazılım alanında neredeyse bir klasik olan elinizdeki kitabı kamuoyu ile paylaşmaktan onur duymaktadır.

Amacına hizmet etmesi dileğiyle

Saygılarımızla
Ağustos 2008

Musa ÇEÇEN
TMMOB Elektrik Mühendisleri Odası
41.Dönem Yönetim Kurulu Başkanı

**Kaynaklar : http://www.ozguryazilim.net vd...

İçindekiler

Katedral ve Pazar

Linux yıkıcıdır. Kim derdi ki gezegenin orasına burasına dağılmış birkaç bin programcının birbirine Internet'in ince telleriyle bağlı bölük pörçük çalışmaları, büyü yapılmış gibi birleşsin; dünya çapında bir işletim sistemi olup çıksın!

Ben kesinlikle diyemezdim! Linux'un izinin radarıma yakalandığı 1993'ün başlarında, ben on yıldır Unix'e ve açık_kaynak geliştirmeye sardırmış durumdaydım. 1980'lerin ortalarında, GNU'da katkısı olan ilk kişilerdendim. Açık_kaynak yazılımı konusunda, nete (ağ ortamına) hatırı sayılır miktarda malzeme sunmuştum. Bunlar, tek başıma yaptığım veya ortaklaşa çalışmalar (nethack, Emacs's VC ve GUD modları, xlife ve diğerleri) şeklinde, bugün de yaygın kullanılan çeşitli programlardır. Neyin nasıl olduğunu bildiğimi sanıyormuşum!

Linux, bildiğimi sandığım çok şeyi altüst etti. Ben yıllarca Unix'in küçük gereçler, hızlı prototipleme ve evrimci programlama ile ilgili düsturunu vaz etmiştim. Ama öte yandan da, bunları aşan, nazik bir durum arz eden bir karmaşa yaşandığını, daha merkezi ve a priori (önceliklikli) bir yaklaşım gerektiğini düşünüyordum. Önemli yazılımların (Emacs programlama editörleri gibi gerçekten iri programların ve işletim sistemlerinin) izole bir ortamda çalışan ustalar yahut küçük bir üstatlar grubu tarafından özenle işlenmiş katedraller gibi inşa edilmesi ve bir beta sürümünün vaktinden önce yapılmaması gerektiği kanaatinde idim.

Linus Torvald'ın tarzı -yani, erken ve sık aralıklarla sürüm yapmak; mümkün olduğu ölçüde çok şeyi vermek; karışıklıklara açık olmak- epey şaşırtıcı olmuştu. Linux çevresinde, katedral tarzının ihtişamından ziyade, tutarlı ve kararlı bir sistemin doğmasının birtakım mucizelere bağlı olduğu imajını veren, çeşitli gündemlerin ve yaklaşımların konu edildiği gürültülü bir pazar yeri havası vardı (bu en iyi, Linux'un, *alelade, herhangi birinin* fikirlerini sunduğu arşiv siteleriyle sınırlanır).

Bu pazar tarzının işleyecek gibi durması ve iyi işlemesi de bir sarsıntı yarattı. Aslında, sırf bireysel projeler için değil; ayrıca Linux dünyasının katedral kurarken hayal bile edilmeyecek hızlarda karmaşaya düşmemesinin; hatta tersine gücüne güç katmasının sebebini anlamak için de çok kafa patlattım.

1996 ortasında, durumu anlamaya başladığımı düşünüyordum. Şans eseri açık_kaynak projesi formunda karşıma çıkan harika bir fırsat,

pazar tarzında bilinçli bir çabaya girmemi sağladı. Projeyi yaptım -sonuç, anlamlı bir başarıydı.

O projenin öyküsü olan bu yazıda, açık_kaynak temelinde proje geliştirme konusunda hisseler çıkartacağım. Linux dünyasında öğrendiklerim bu ilk bilgilerden ibaret değildir. Ayrıca, Linux dünyasının bu şeylere neler kattığı da görülecektir. Bunların, Linux çevresini bir çeşit yazılım hayratı haline getirenin ne olduğunu kavramanıza yardım edeceklerini sanıyorum - hatta kim bilir, daha verimli olmayı bile sağlayabilirler.

Mektup Adresine Varmalı

1993'ten itibaren, Pennsylvania'daki Batı Chester'da serbest erişimli bir Internet Servis Sağlayıcı olan CCIL'in (Chester County InterLink) teknik tarafını yürütüyordum. Kurucu ortağıydım ve çok kullanıcılı bülten paneli yazılımı da benim üstümdeydi - *telnet://locke.ccil.org* adresi üzerinden bakabilirsiniz. Bugün otuz hat üzerinde üç bin kullanıcıya hizmet veriyor. Ben bu sayede CCIL'in 56K hattı üzerinden günde 24 saat nete erişim sağlıyordum - aslında bu, bir yerde işin gereğiydi.

Internet e-postayı sıkça kullanmam gerekiyordu. *locke* sitesinde postama bakmak için telnet'e girme mecburiyeti, illet bir şeydi. Benim istediğim, gelen bir mektuptan haberdar olmak ve kendi yerel gereçlerimle ilgilenmek üzere mektubun (kendi ev sistemime) teslimi idi.

Internet'in malum posta transfer protokolü, yani SMTP (Basit Posta Transfer Protokolü, *[:Simple Mail Transfer Protocol]*) uygun değildi. Çünkü, makinelerin gereğince işlemesi, tam zamanlı bağlantıda olmalarına bağlıdır. Oysa benim cihazım, sürekli Internet'te değildi ve statik IP adresi yoktu. İhtiyacım olan şey, çevirmeli bağlantı üzerinden, postamın yerel olarak teslimini sağlayacak bir programdı. Bu türden bir şeylerin olduğunu ve pek çoğunun POP (Postahane Protokolu, *[:Post Office Protocol]*) denilen basit bir uygulama protokolüne dayandığını biliyordum. POP, bugün pek çok posta istemcisi [*:client*] tarafından tanınmasına rağmen, o zamanlar benim kullandığım okuyucuda kurulu değildi.

Bana bir POP3 istemcisi gerekiyordu. Internet'te aradım ve buldum. Aslında üç-dört tane bulmuştum. İçlerinden birini bir süre kullandıysam da; olması gereken özelliklerinden biri, yani, yanıtlamanın çalışabilmesi için çekilip alınmış [*:fetched*] mektuptaki adreslerle başa çıkma [*:hack*] yeteneği yoktu.

Problem şuydu: Diyelim 'joe' diye biri *locke*'da bana mektup göndermiş olsun. Eğer mektubu yerel sistemime alır da yanıtlamaya kalkarsam; postacım [:*mailer*], tüm iyi niyetiyle, bu cevabı sistemimde mevcut olmayan bu 'joe' adlı kişiye göndermeye kalkışır. Ayrıca <@ccil.org> üzerinde dönüş adreslerini elle düzenlemek de ciddi sıkıntı demektir.

Oysa, bilgisayarın yapması gereken şeylerden biri olan bu iş konusunda, mevcut POP istemcilerinden hiçbiri bir şey bilmiyordu. Böylece ilk dersimize gelmiş oluyoruz:

1. Her iyi yazılım, programcısının kendi yarasını kaşımasıyla başlar.

Aslında bu herkesin malumu olmakla birlikte ("İcatların anası zaruretlerdir" ünlü bir deyiştir), yazılımcılar günlerini genelde ihtiyaç duymadıkları ve sevmedikleri programlarla uğraşarak geçirirler.

Ama Linux dünyasında öyle değildir - nitekim Linux çevresinde üretilen yazılımların ortalama kalitesinin yüksek olması da muhtemelen bu nedenledir.

Bu durumda derhal mevcutlarla yarışacak yepyeni bir POP3 sunumcu kodu yazmaya mı giriştim? Olacak iş değil! Ama, elimin altındaki POP uygulamalarını inceledim ve "Bunlardan hangisi benim istediğime yakın?" diye sordum. Çünkü:

2. İyi programcı, ne yazacağını bilen programcıdır. Ama, neyi yeniden yazacağını (ve kullanacağını) bilen programcı, büyük programcıdır.

Büyük programcı olduğumu söylemiyorum fakat öyle olmaya özeniyorum. Büyük olmanın ayırt edici karakteristiği, yapıcı tembelliktir. En yüksek not, sonuca değil çabanıza verilir. Ayrıca, eldekiyle başlamak ta, daima sıfırdan başlamaktan iyidir.

Örneğin Linus Torvalds da [http://www.tuxedo.org/~esr/faqs/linus] yazmaya sıfırdan başlamaya girişmemiş; tersine elinin altındaki bir programı, PC klonlarına yönelik Unix benzeri ufak bir işletim sistemi olan Minix kodunu kullanmıştı. Yani bu kod, büyüyüp Linux olacak bebek için, bir çıkış noktası olmuştu. Benim hazır bir POP işlevini kalkış noktası olarak kullanmam da, aynı hikayedir.

Unix dünyasının kod paylaşma geleneği, kodun yeniden kullanımı konusunda dostane bir tutum sergiler (nitekim bu, GNU projesinde işletim sistemi olarak, çeşitli çekincelerine rağmen, neden Unix'in seçildiğini de açıklar). Linux dünyası da bu geleneği teknolojik sınırların

sonuna kadar kullanmıştır. Çoğu durumda terabaytlar ölçüsünde kaynak, kullanıma açık durumdadır. Yani, birinin Linux dünyasındaki eli yüzü düzgün denebilecek çalışmasına ulaşmak için sarf edilen çaba, başka yerlere göre çok daha iyi sonuç verir.

Benim için de tam olarak böyle oldu. İşin başında bulduklarım sayesinde, ikinci adımdaki araştırmalarım sonunda dokuz aday tespit ettim: fetchpop, PopTart, get-mail, gwpop, pimp, pop-perl, popc, popmail ve upop. İlki, yani fetchpop, Seung-Hong Oh'nun geliştirdiği bir işti. İçine, mevcut başlık-yeniden yazımı [*:header-rewrite*] özelliğinden başka eklemeler de yaptım. Zaten o da bunların 1.9 sürümünde yer almasını uygun gördü.

Ancak birkaç hafta sonra Carl Harris'in popclient/popistemci (posta istemcisi, postahane protokolü istemcisi, protokol istemcisi [*:popclient*]) koduyla tanıştığımda, bir sorunum olduğunu fark ettim. Fetchpop (örneğin arkaplan-daemon modu gibi), iyi sayılabilecek özgün fikirler içermekle birlikte, sadece POP3'le baş edebiliyordu ve oldukça amatörce yazılmıştı (o sıralar, Seung-Hong parlak fakat deneyimsiz bir programcıydı ve bu iki nitelik hemen göze çarpıyordu). Carl'ın yazdıkları daha iyi, profesyonel ve derlitoplu idi. Ama uygulamaya özgü hileler de denilebilecek bazı önemli fetchpop özelliklerine ihtiyacı vardı (ki onları da bizzat ben yazdım).

Gitmek mi zor, kalmak mı? Gitsem, yani diğerine geçsem; hazırladığım kod, daha iyi bir geliştirme zemini uğruna heba olacaktı. Fakat öte yandan da gitme yönünde, çoklu-protokol desteğinin varlığından doğan güçlü bir dürtü vardı. Postahane sunucu protokollerinin en yaygın olanı, tek olmamakla birlikte, POP3'tür. Fetchpop ve diğer rakipleri POP2, RPOP yahut APOP işini görmezler. Ayrıca ben de, sırf eğlence olsun diye, IMAP [http://www.imap.org] (Internet Mesajı Erişim Protokolü, *[:Internet Message Access Protocol]*, yani en son ve güçlü postahane protokolü) eklemek konusunda kararsızlık içindeydim. Yine de gitmenin iyi bir fikir olacağı konusunda teorik sayılabilecek bir gerekçem vardı ki daha Linux'a gelmeden epey önce öğrendiğim bir şeydi:

> 3. "Heba etmek hesap dahilindeyse, bir şekilde heba edersin." (Fred Brooks, *The Mythical Man-Month, Chapter 11*)

Bunu şöyle de formüle edebiliriz: Bir çözüm bulana kadar, sorunun ne olduğu genelde anlaşılmaz. İşi hakkıyla yapmayı öğrenmek, belki ikinci

denemede mümkün olacaktır. Bir işi hakkıyla yapma arzusundaysanız, bir kere daha başlamaya hazır olacaksınız [JB].

Neyse, fetchpop'a geçişlerim ilk denememde oldu (diyordum kendime). Böylece diğerine geçtim.

İlk grup popclient yamalarını [*:patch*] 25 Haziran 1996'da Carl Harris'e yolladığımda, kendisinin popclient konusundaki ilgisini epeyce yitirmiş olduğunu keşfettim. Kod, bir parça tozlanmıştı ve de ufak tefek arızalar (hatalar) içeriyordu. Değiştirilecek çok şey vardı. Ayrıca, yapacağım en mantıklı şeyin, programı bitirmeyi üstlenmek olduğunda derhal anlaşmıştık!

Ben henüz farkına varmadan proje ısınmıştı. İş, var olan bir POP istemcisine ufak tefek yamalar tasarlamaktan çıkmıştı; projenin arkasını getirmekle uğraşıyordum ve zaten önemli değişiklikler yapmam icap edeceğine dair bir takım kuşku bulutları da şekillenmeye başlamıştı. Kod paylaşımını öne çıkaran bir yazılım kültüründe evrim, projenin doğal sürecidir. Şu ilkeye göre hareket ediyordum:

> 4. Doğru yoldaysan, birbirinden ilginç problemlerle karşılaşacaksın.

Ama Carl Harris'in durumu daha önemliydi. Çünkü o, şunun farkındaydı:

> 5. Bir programa olan ilgini yitirdiysen; geliştirmek için rakip programı seçmek, en son yapacağın şey olmalı.

Değerlendirmesine bile girmemiş olmakla birlikte, Carl da ben de biliyorduk ki; en iyi çözüme varmak ikimizin de ortak hedefiydi. Yegâne sorun, benim sağlam pabuç olarak iş görüp göremeyeceğimdi. Ama (Carl), benim işi kotardığımı gördükten sonra, gayet zarif ve yardımcı bir tavır sergiledi. Umarım sıram gelince ben de onun gibi olurum.

Kullanıcı Sahibi Olmanın Önemi

Böylece popclient, bana miras kaldı. Ayrıca, kullanıcı havuzu da miras kalmıştı. Kullanıcının olması, harika bir şeydir. Bu, sadece bir ihtiyacı gördüğünü değil; işi doğru yaptığını da gösterdiği için böyledir. Gereğince donatılırlarsa, programın oluşumuna katkıları olabilir.

Unix geleneğinin bir başka kuvveti, -ki Linux çok hoş bir uç noktaya vardırmıştır, kullanıcıların çoğunun programcılar olmasıdır. Kaynak

kodu el altında olduğu için *etkili* programcılar olmaları mümkündür. Arıza (hata) giderme süresini kısaltmakta müthiş yararlı oabilirler. Kullanıcılar biraz teşvikle, sorunları teşhis eder; onarım için öneriler getirir ve kodu tek başınıza yapmaya kıyasla çok daha hızlı geliştirmenize yardımcı olurlar.

> 6. Programın oluşumunda kulanıcıların katkısına başvurmak, kod geliştirmede ve arıza (hata) gidermede en etkili ve en rahat yoldur.

Bu etkiyi göz ardı etmek gayet kolaydır. Aslında bizler, -yani açık_kaynak dünyasında yer alanların hatırı sayılır kısmı, Linus Torvald tersini gösterene kadar, bu etkinin, sistemin karmaşıklığıyla ve kullanıcı sayısıyla artacağını ciddi ölçüde göz ardı ettik.

Nitekim, bence Linus'un en zekice ve mühim eseri, inşa ettiği Linux çekirdeği değil; fakat icat ettiği Linux geliştirme modelidir. Bu fikrimi onun da bulunduğu bir sırada açıkladığımda, gülümsemiş ve sıkça söylediği şu noktayı yinelemişti: "Ben aslında, başkalarının yaptıklarını kendime yontmaktan hoşlanan aşırı tembel biriyim." *Tilki* gibi tembel. Yahut, Robert Heinlein'in, karakterlerinden biri için dediği gibi, başarısız olamayacak kadar tembel.

Geçmişe bakılarak, GNU Emacs Lisp kitaplığının ve Lisp kod arşivlerinin geliştirilmesi, Linux'un başarısına ve yöntemlerine örnek gösterilebilir. Emacs C kalbinin ve diğer bütün GNU gereçlerinin oluşumundaki katedral tarzının tersine, LISP kod havuzunun evrimi akışkandı ve yönlendireni de kullanıcıydı. Fikirler ve prototipler, kararlı bir duruma bürünmeden önce üç –belki de beş kez, yeniden yazılmışlardı. Genelde, Internet üzerinden *a la* Linux, gevşek bağlara sahip bir işbirliği geçerliydi.

Esasen, fetchmail öncesinde yaptığım bana ait en başarılı tekil çalışma, muhtemelen Emacs VC (sürüm kontrolü, [*version control*]) moduydu ki üç kişiyle e-posta üzerinden yürüttüğüm Linux-vari bir işbirliği idi. Bugüne kadar da içlerinden sadece biriyle, Richard Stallman (Emacs'ın yazarı ve Free Software Foundation'ın [http://www.fsf.org] kurucusu) ile tanıştım. Çalışma, Emacs içinden SCCS, RCS ve sonradan da CVS için "tek-tuş" kontrol operasyonları sağlayan bir ön-uç [:*front-end*] idi.

Bu çalışma, başka birinin yazdığı ufak, ham sccs.el modundan türemişti. VC'nin geliştirilmesi başarı getirdi. Çünkü, Emacs Lisp kodu, bizzat