Gömülü Sistemler İçin Yeni Bir İşletim Sistemi: eGİS
A New Operating System For Embedded Systems: eGİS

Sinan YILDIRIM, Aylin KANTARCI

Bilgisayar Mühendisliği Bölümü
Ege Üniversitesi, Bornova,İzmir
sinan.yildirim@mail.ege.edu.tr
aylin.kantarci@ege.edu.tr

Özetçe

Birçok gömülü uygulama daha rahat bir çalışma ortamı sunan gömülü ve gerçek zamanlı bir işletim sitemine gereksinim duymaktadır. Gömülü uygulamalara servis sunan gömülü işletim sistemleri, genel amaçlı işletim sistemlerinin aksine uygulama gereksinimlerine göre değiştirilebilir ve yapılandırılabilir olmalıdır. Bu çalışmada gerçek zamanlı, taşınabilir, yapılandırılabilir, nesneye yönelik ve gömülü bir işletim sistemi olan eGİS (Ege Gömülü İşletim Sistemi) geliştirilmiştir. Bu bildiride geliştirilen sistem tanıtılmış ve elde edilen bazı deneysel sonuçlar sunulmuştur.

Abstract

Many embedded applications need a real-time and embedded operating system that provides an simple operation environment. Embedded operating systems that give service to embedded applications must be configurable and restructurable in contrast to general purpose operating systems. In this work, eGİS (Ege Gömülü İşletim Sistemi) which is a real-time, portable, configurable, object oriented and embedded operating system is implemented. In this paper, the developed system is introduced and some experimental results are presented.

1. Giriş

Gömülü sistemler günümüzde giderek önem kazanmaktadır. Avuçiçi cihazlar ve mobil telefonlar gibi taşınabilir cihazlar, televizyon gibi büyük ve karmaşık sistemler gömülü yazılım içeren bazı sistemlerdir. Bu sistemler kısıtlı sistem kaynakları, gerçek zamanlılık ve çok değişken uygulama gereksinimlerine sahip olmaları ile geleneksel yazılım sistemlerinden ayrılmaktadır.

Bilgisayar donanımcıları uzun bir süre gömülü yazılımla ilgilenmemişlerdir. Bu sistemler için yazılımın genellikle alt seviye diller ile yazılması ve sistemin donanım maliyetinin daha ön planda oluşu, gömülü yazılımlara gereken önemin verilmemesine yol açmıştır. Modern yazılım bakış açısının sunduğu çözümler gömülü sistemler için pahalı ve gerçeklenemez olarak görünmüş ve bu alanla ilgili araştırmalar ikinci plana atılmıştır. Donanımın ucuzlamasıyla birlikte dikkatler yazılım üzerine çevrilmiş, gömülü sistemlerle ilgili yazılım araştırmaları önem kazanmaya başlamıştır [1].

Gömülü sistemler için gerçekleştirilen işletim sistemleri, genel amaçlı işletim sistemlerinden farklı tasarım amaçlarına ve servislere sahiptirler. Verimli bir kaynak yönetimi ve gerçek zamanlı sistem algoritmaları içeren gömülü işletim sistemleri, giderek modern yazılım kavramlarının sunmuş olduğu avantajlara gereksinim duymaktadır. Gömülü sistemler, çalışma zamanı gereksinimleri belirlenmiş ve özelleşmiş yazılımları içeren sistemler oldukları için gömülü sistemler için tasarlanmış olan işletim sistemleri uygulama istek ve kısıtlarının zorladığı değişimlerin kolay yapılabildiği sistemler olmalıdır. Büyük ve karmaşık olan günümüz gömülü uygulamalarının ihtiyaçlarını karşılayacak olan gömülü işletim sistemleri, iyi bir sistem başarımına sahip olmak ve verimli kaynak yönetimi algoritmaları içermek dışında yeniden yapılandırılabilme, taşınabilme ve dağıtıklık gibi modern yazılım özelliklerini destekleyebilmelidir.

Bu doğrultuda, gömülü işletim sistemlerinde modern bir yazılım kavramı olan tasarım desenlerinin [2] kullanılması, uygulama gereksinimlerine göre işletim sistemi yapılandırılmasını kolaylaştırır. Tasarım desenleri gömülü işletim sistemlerin özel tasarım problemlerini çözebilir ve nesneye yönelik tasarımları genişleyebilir, yeniden kullanılabilir ve daha düzenli hale getirir. Günümüzdeki gerçek zamanlı ve gömülü yazılım sistemleri, mimarisel ve genel tasarım desenleri ve bu desenlerin işaret ettiği tasarım problemleri göz önüne alınarak tasarlanılmalıdırlar.

Bunlara ek olarak küçük ve temel bir işletim sistemi çekirdeği içermeyi amaçlayan mikroçekirdek mimarileri [3] kullanmak, gömülü işletim sistemlerinin bakımını kolaylaştırmak ve hatalara karşı duyarlılığını arttırmak dışında yapılandırılabilme ve taşınabilme gibi özelliklerin sağlanması yönünde de katkıda bulunacaktır.

Bu çalışma kapsamında, gerçek zamanlı gömülü uygulamaların gereksinimlerini karşılayacak olan eGe Gömülü İşletim Sistemi (eGİS) [4], mikroçekirdek mimarisi göz önüne alınarak uygulama ihtiyaçlarına göre yapılandırılabilir ve taşınabilir olacak şekilde tasarlanmıştır. eGİS, C++ dili ve açık kaynak koda sahip ücretsiz yazılım geliştirme araçları olan Gcc derleyicisi, Gdb hata ayıklayıcısı, Ld bağlayıcısı ve Bochs emülatörü kullanılarak ve ise tamamiyle Türkçe geliştirilmiştir.

Literatürde UCOS-II, eCOS, PURE, CHORUS ve TINYOS gibi gömülü işletim sistemlerine rastlamak mümkündür. Bunlardan UCOS-II [5] işletim sistemi çok iyi sistem başarımına sahip ve tahmin edilebilir kaynak yönetim algoritmaları içermektedir. Ancak sistem içerisindeki öncelik tabanlı kesilebilir süreç yönetim algoritmasının değiştirilip yerine farklı bir yönetim algoritmasının eklenmesi işletim sisteminin önemli bir kısmının yeniden yazılmasını gerektirmektedir. Sistem algoritmalarının uygulama ihtiyaçlarına göre değiştirilebilmesi, bir başka deyişle işletim sisteminin uygulama ihtiyaçlarına göre yapılandırılabilmesi uCOS sistemi içerisinde mümkün değildir. Bu durum Nucleus [6] işletim sistemi için de geçerlidir. eCOS [7] ve PURE [8]

uygulama ihtiyaçlarına göre yapılandırılabilen işletim sistemleridir. Ancak eGİS daha küçük, sade ve daha temiz bir gerçekleştirime sahiptir ve tasarım desenlerine dayanan bir mimariye sahiptir.

TINYOS [9] yapılandırma işlemini kendine özgü NesC dilini kullanarak yapmaktadır. eGİS'te ise yapılandırma adımı herhangi bir üst seviye dil yardımı ile değil kod ile yapılmaktadır. EPOS [10] derleme anında kullanıcıdan elde ettiği yapılandırma bilgisi doğrultusunda sisteme ek yük getiren soyutlamalardan kurtularak işletim sistemini yapılandırmaktadır. Yani ek fonksiyon çağırımları ve sanal fonksiyonların getirdiği yükler ortadan kaldırılmıştır. eGİS'te ise henüz böyle bir çalışma yapılmamıştır.

Bu bildiride eGİS işletim sistemi mimarisi hakkında bilgi verilmekte ve sisteme ait bazı başarım ölçütleri sunulmaktadır. İkinci bölümde eGİS Sistem mimarisi açıklanmaktadır. Geliştirim detayları ve başarım sonuçları üçüncü bölümde verilmektedir. Genel yargılar ve ileride gerçekleştirilmesi planlanan çalışmalar dördüncü bölümde dile getirilmektedir.

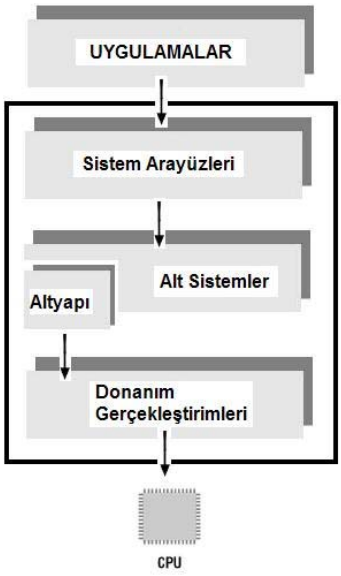
2. eGİS Sistem Mimarisi

eGİS gerçek zamanlı gömülü sistemler için nesneye yönelik olarak geliştirilmiş bir mikroçekirdektir. Şu anda süreç yönetimi, kesme yönetimi ve süreçler arası haberleşme işlevlerini yerine getiren üç alt sistemden oluşmaktadır. eGİS, katmanlı bir mikroçekirdek olup temel mimarisi Şekil 1'de gösterilmiştir.

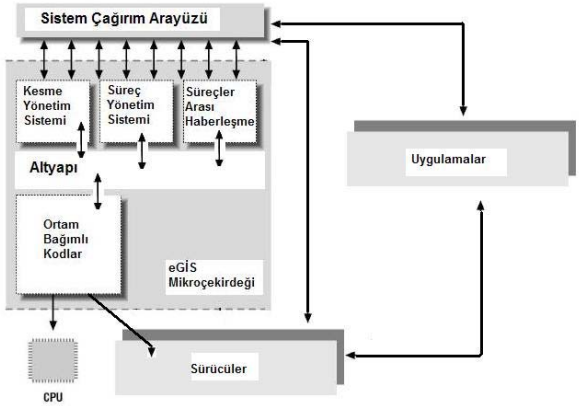
AltYapı katmanı sistemin çekirdeğini oluşturmakta ve tüm sistem yönetim algoritmalarına ait arayüzler ve çok temel sistem gerçekleştirimleri bu katmanda yer almaktadır. Sistem yapılandırılması esnasında birbirleri ile yer değiştirilen alt sistemler AltYapı katmanı üzerine oturmaktadır. *AltSistemler* katmanı AltYapı katmanında belirtilen sistem soyutlamalarına ait gerçekleştirimleri içermektedirler. Örneğin süreç yönetimi için gerekli olan temel soyutlamalar ve algoritmik arayüzler AltYapı katmanında yer alırken, bu arayüzlerin gerçekleştirimi AltSistemler katmanında yer almaktadır.

Platform katmanı yine AltYapı katmanı içerisinde yer alan donanımsal soyutlamalara ait gerçekleştirimleri içermektedir. Bunun sonucunda sistemin donanım ortamına ait bağımlılıklar tek bir katmanda toplanmış olmakta ve donanım ortamı değişikliklerinden sadece tek bir katman etkilenmektedir. Bu da sistemin taşınabilirliğini olumlu yönde etkilemektedir.

Uygulama Arayüzleri katmanı ise uygulamaların sistem yapılandırılmasından ve iç yapılardan haberdar olmadan, sade ve temiz bir arayüzle sistem servislerinden faydalanmalarını sağlamaktadır. Sonuç olarak, sistemin içsel değişiklikleri uygulamaları etkilememektedir.



Şekil 1*: eGİS Sistemi Mimarisi*



Şekil 2: eGİS Mikroçekirdeği

eGİS mikroçekirdeğinin daha detaylı gösterimi Şekil 2'de verilmiştir. **eGe_Surec**, **eGe_Kesme** ve **eGe_Haberlesme** sırası ile süreç, kesme ve süreçler arası haberleşme servislerinin gerçekleştirimlerini içeren alt sistemlerdir. eGe_Surec öncelik tabanlı kesilebilir süreç yönetim algoritmasını içermektedir. Şu anda 64 adet iş parçacığını yönetebilen bu alt sistem; iş parcacığı yaratımı, durdurulması, sonlandırılması, öncelik değiştirme, bekletme ve sonlandırma servislerini sunmaktadır. eGe_Kesme alt sistemi kesme kesmeleri yönetebilmek için kesmelere kayıt olmak, kayıtlı kesmeden çıkmak için basit servisler sunmaktadır. eGe_Haberlesme iş parçacıkları arasındaki haberleşmeyi ve eşzamanlamayı sağlayan semafor sinyalleme/yakalama ve kilit açma/kapama servislerini sunmaktadır.

eGİS işletim sisteminin çekirdek sınıfı tanımlaması Şekil 3'te gösterilmiştir. Bu sınıfın sistemde sadece bir adet örneği bulunmakta ve bu mikroçekirdek nesnesi uygulama ihtiyaçları doğrultusunda seçilmiş alt sistemlere ait referansları barındırmaktadır. Alt sistemler arasındaki haberleşmeler de yine bu nesne üzerinden yapılmaktadır.

```
class eGIS_Cekirdek : public eGIS_Nesne
{
  public:

    virtual ~eGIS_Cekirdek();

    static eGIS_Cekirdek *cekirdekDondur();

    void surecSistemiKaydet(eGIS_SurecSistemi *sistem);
    void haberlesmeSistemiKaydet(eGIS_HaberlesmeSistemi *sistem);
    void kesmeSistemiKaydet(eGIS_KesmeSistemi *sistem);

    eGIS_SurecSistemi *surecSistemiDondur();
    eGIS_HaberlesmeSistemi *haberlesmeSistemiDondur();
    eGIS_KesmeSistemi *kesmeSistemiDondur();

  protected:

    eGIS_Cekirdek();

  protected:

    static eGIS_Cekirdek *_cekirdek;

    eGIS_SurecSistemi *_surecSistemi;
    eGIS_HaberlesmeSistemi *_haberlesmeSistemi;
    eGIS_KesmeSistemi *_kesmeSistemi;
};
```

Şekil 3: Mikroçekirdek Sınıfı

Genel sistemin doğru bir şekilde çalışabilmesi için ilgili alt sistemlerin mikroçekirdek nesnesine kaydedilmesi ve alt sistemlerin doğru bir şekilde ilklenmesi gerekmektedir. Bu adımlardan sonra işletim sistemi ana iş parçacığını yaratılarak ve o iş parçacığına geçiş yaparak çalışmaya başlar. Bu adımlar Şekil 4'te gösterilmiştir.

```
extern "C" int eGIS_main()
{
  plt_intel386_BaglamFabrikasi baglamFabrikasi;
  plt_intel386_KesmeSistemi pltKesmeSistemi;
  plt_intel386_KritikBolge kritikBolge;

  eGe_Surec::SurecSistemi ege_surec(&baglamFabrikasi);
  eGe_Kesme::KesmeSistemi ege_kesme(&pltKesmeSistemi);
  eGe_Haberlesme::HaberlesmeSistemi ege_haberlesme;

  /* surec yonetim alt sistemi olan Zigana kaydediliyor */
  eGIS_CEKIRDEK->surecSistemiKaydet(&ege_surec);

  /* kesme yonetim sistemini ata */
  eGIS_CEKIRDEK->kesmeSistemiKaydet(&ege_kesme);

  /* haberlesme sistemini ata */
  eGIS_CEKIRDEK->haberlesmeSistemiKaydet(&ege_haberlesme);

  ege_kesme.ilkle();
  ege_surec.ilkle();

  /* isletim sistemini baslat */
  ege_kesme.kesmeleriAc();
  ege_surec.yonetimiAc();

  return 0;
}
```

Şekil 4: Mikroçekirdeğin Yapılandırılması

eGİS işletim sisteminin katmanlı yapısı taşınabilirlik, genişleyebilirlik, bakım ve yapılandırılabilme bakımından genel sisteme katkıda bulunmuştur. İçsel mimaride görüldüğü gibi alt sistemler içindeki gerçekleştirimler temel altyapı soyutlamalarına bağlanmaktadır. Donanımsal gerçekleştirimler mikroçekirdeğin dışında tutuluğu için alt sistemler donanımdan tamamiyle soyutlanmışlardır. Ayrıca eGİS'in uygulama ihtiyaçlarına göre şekillenebilmesi soyutlamalar ve gerçekleştirimlerin birbirlerinden ayrılması ile kolaylaşmıştır. Uygulama ihiyaçlarına uygun servisleri sunan değişik alt sistemler aynı soyutlamalara ve arayüzlere uygun oldukları sürece birbirleri ile yer değiştirebilirler.

eGİS mimarisinin esnekliğinde, katmanlı yapının getirdiği avantajların yanı sıra, alt sistem tasarımlarında önemli bir yazılım mühendisliği aracı olan *tasarım desenlerinin* kullanılmasının da büyük bir rolü vardır. Desenlerin eGİS sistemine kattığı özellikler Tablo 1'de gösterilmiştir. Sistemin taşınabilirliği, sistem nesnelerini somut sınıf isimlerini açık bir şekilde belirterek yaratmadan Köprü ve *Soyut Fabrika* desenleri kullanımı ile sağlanmıştır. *Tekil* deseni mikroçekirdek nesnesini kontrollü erişimi sağlamıştır. *Önyüz* deseni eGİS işletim sisteminin içsel yapısını istemcilerden soyutlar ve temiz bir uygulama arayüzü sunar. *Strateji* deseni yönetim algoritmalarının birbirleri ile değiştirilebilmesini ve eGİS sisteminin yönetim algoritmalarından bağımsız olması sağlanmıştır [11].

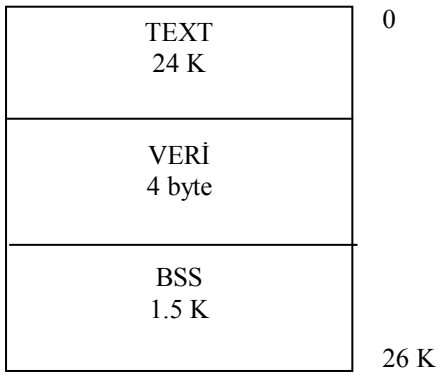
Tablo 1: Tasarım desenlerinin eGİS mikroçekirdeğine etkileri

Desen	Desenin Sunduğu Avantaj
Tekil	Mikroçekirdek nesnesine kontrollü erişim
Soyut Fabrika	Taşınabilirlik
Köprü	Taşınabilirlik
Önyüz	Uygulama bağımsızlığı
Strateji	Algoritma Değiştirilebilirliği

3. Gerçekleştirim ve Deneysel Sonuçlar

eGIS, C++ dili ile açık kaynak kodlu Gcc [12] derleyici ailesi ve Bochs [13] i386 PC emulatörü kullanılarak geliştirilmiştir. Şu anda eGİS i386 ortamında çalışabilir 3 alt sistem ve 50'nın üzerinde sınıftan oluşmaktadır.

eGİS mikroçekirdeği uygulama kodu ile beraber bağlanarak tek bir ikili ortam kodu elde edilir. Şekil 5'de eGİS mikroçekirdeğinin bellek kullanımı gösterilmiştir. Günümüz gömülü sistemlerinin eGIS'in ihtiyaç duyduğu bellek miktarını karşılayabileceği görülmektedir. Bu durum, işletim sisteminin taşınabilir ve yeniden yapılandırılabilir olması için eklenen katmanların getirmiş olduğu ek bellek kullanımının kabul edilebilir ölçülerde olduğunu göstermektedir.



Şekil 5: eGİS mikroçekirdeğinin bellek izi

Tablo 2'de işletim sisteminde yer alan bazı işlemlerin getirdiği yükler, saat darbesi sayısı cinsinden gösterilmektedir. Mikroçekirdek sınıfı tüm kayıtlı sistemleri tuttuğundan, kaydedilmiş sistemlere erişmek için mikroçekirdekten ilgili sistem istenmelidir. Bu işlem için 12 saat darbesi kadar bir ek yük ortaya çıkmaktadır. Aynı şekilde bir sonraki adımda hangi sürecin çalışacağına karar verme işlemi 168 saat darbesi sürmektedir. Bu sürenin yaklaşık 29 saat darbelik bölümü, süreç yönetim algoritmasının sistem içerisindeki değişebilirliğini sağlamak için koyulan ek

soyutlama katmanında harcanmaktadır. Benzer bir durum iş parçacığı yaratımı esnasında da görülmektedir. 1146 saat darbesi süren bir iş parçacığı yaratım işleminin en fazla 234 saat darbelik kısmı soyutlama katmanlarında harcanmaktadır.

Tablo 2: eGİS mikroçekirdeğine ait bazı başarım ölçütleri

İşlem	Ara katmanlarda harcanan saat darbesi	Geçen toplam saat darbesi
Mikroçekirdek nesnesine erişim	12	12
Süreç yönetim algoritmasının çalıştırılması	29	168
İş parçacığı yaratılması	<234	1146

Görüldüğü gibi işletim sisteminin uygulama ihtiyaçlarına göre şekillendirilebilmesi ve taşınabilir olması için mimaride yer alan ek katmanlar sistem başarımını bir miktar azaltmaktadır. Ancak gömülü işlemcilerin daha da güçlendiğini göz önüne alırsak, elde edilen avantajlara göre başarım kaybının kabul edilebilir boyutlardadır. Bunun yanı sıra, kullanılan modern yazılım mühendisliği teknikleri ile gömülü sistemlere getirilen esnekliğin sisteme kazandırdığı avantajların, küçük başarım kayıplarından çok daha büyük değer taşıdığı bir gerçektir.

4. Sonuçlar ve Gelecek Çalışmalar

Özet olarak, bu çalışmada tanıtılan yeni bir gerçek zamanlı ve gömülü işletim sistemi çekirdeği olan eGİS'in, uygulama gereksinimlerine göre değişebilir, şekillendirilebilir ve taşınabilir bir sistem olacak şekilde tasarlanmasına dikkat edildiğini söyleyebiliriz.

Gelecekte yapılacak çalışmalarda, eGİS'e bellek ve disk üzerinden çalışabilen bir dosya sistemi, TCP/IP yığıtı, gömülü kullanıcı arayüzü sistemi gibi alt sistemlerin, yeni süreç yönetim algoritmalarının ve farklı donanım ortam gerçekleştirimlerinin eklenmesini planlamaktayız. Ayrıca eGİS mimarisinin daha da iyileştirilmesi ve en iyilenmesi, soyutlama katmanlarının getirmiş olduğu ek yüklerin ortadan kaldırılıp sistem başarımının daha arttırılması da amaçlanmaktadır.

eGİS işletim sistemi, açık kaynak koda sahip ve GNU lisansı ile diğer mühendislerin de katkılarına açık bir işletim sistemidir. eGİS diğer mühendislerin katılımı ile, daha hızlı ve sağlam bir genişleme imkanına sahip olacaktır. Bu amaçla eGİS açık kaynak kodlu projelerin yer aldığı www.sourceforge.net sitesinde paylaşıma açılmıştır [14].

Sonuç olarak yapılan çalışma ile, bu konu hakkında çalışma yapmak isteyenler için bir kaynak ortaya konulmuştur. Bu kaynağın, işletim sistemleri ve gömülü sistemler ile ilgilenenlere faydalı olacağı umulmaktadır.

5. Kaynakça

[1] E.A. Lee, "What's Ahead for Embedded Software? ", IEEE Computer Society Press, Computer, v.33 n.9 pp.18-26, 2000.

[2] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional Computing Series, 1994.

[3] J. Liedtke, "On micro-kernel construction", *In Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, 1995.

[4] K. S. Yıldırım, "Gömülü Sistemler İçin Tasarım Desenleri Kullanarak Nesneye Yönelik, Gerçek Zamanlı Bir Mikroçekirdek Tasarımı", Yüksek Lisans Tezi, Ege Üniversitesi Bilgisayar Mühendisliği Bölümü, 2005.

[5] J. J. Labrosse, *MicroC/OS-II: The Real-Time Kernel, 2nd Ed.*, CMP Books, 2003.

[6] The Nucleus Plus RealTime Operating System anasayfası: www.acceleratedtechnology.com/embedded/nuc_rtos.html

[7] The eCOS Operating System anasayfası: http://ecos.sourceware.org/

[8] U. Haack, W. Schröder-Preikschat, F. Schön, O. Spinczyk, "Design Rationale of the PURE Object-Oriented Embedded Operating System", *In Proceedings*

[9] Gay, D., Levis, P., Culler, D., 2004, Software Design Patterns for TinyOS

[10] Fröhlich, A.A., Schröder-Preikschat, W., EPOS: an Object-Oriented Operating System; In Proceedings of the Workshop on Object-Oriented Technology, Lecture Notes In Computer Science Volume 1743

[11] K. S. Yıldırım, A. Kantarci"eGİS: Gömülü Sistemler İçin Tasarım Desenleri Tabanlı, Nesneye Yönelik ve Gerçek Zamanlı Bir Mikroçekirdek", Ulusal Yazilim Mimarisi Sempozyumu, 2006.

[12] The GNU Compiler Collection anasayfası: http://gcc.gnu.org/

[13] Bochs IA32 PC Emulator homepage anasayfası: http://bochs.sourceforge.net/

[14] eGİS İşletim Sistemi kod ambarı adresi: http://sourceforge.net/projects/egis-eeos/