Taşınabilir Kod Yazma

Temel İlkeler ve Problemler

Barış Şimşek
Bilgisayar Müh.
http://www.enderunix.org/simsek/

EnderUNIX Yazılım Geliştirme Takımı

Başlarken

Bu sunum yalnızca merak uyandırmak için hazırlanmıştır! Bir klavuz değildir.

1988'de POSIX olarak bilinen IEEE Std 1003.1-1988 bir Amerikan standardı olarak yayınlandı.

1990 yılında IEEE Std 1003.1-1990 bir dünya standardı olarak yeniden yayınlandı.

POSIX = IEEE Portable Operating System Interface for Computing Environments

POSIX

POSIX, işletim sistemi için temel bir arayüz tanımlar.

C programlama arayüzü, sistem dosyaları, dosya biçimleri...

POSIX.1 karakter tabanlı uygulamalar içindir. Ağ ve grafik gibi farklı alanlara ait standartları içermez. POSIX.8 bu alanlar için oluşturulmuştur.

POSIX sistem yönetimine dair standartlar içermez. Nasıl yedekleme yapılır, nasıl kullanıcı açılır, nasıl paket yüklenir...

POSIX ve UNIX

POSIX System V ve Berkeley UNIX üzerine inşa edilmiştir.

POSIX bir işletim sistemi sunmaz!

POSIX Geliştirme Ortamı

POSIX uyumlu kod yazmak için bir başlık dosyası (.h) veya bir kütüphane yüklemeniz gerekmez. POSIX geliştirme ortamına ulaşmak için:

```
#define _POSIX_SOURCE 1
```

Bunu işletim sistemi sahibi firma derleyicide uygular. Sizin detayları bilmenize gerek yoktur.

Port Etme – Kaynak Kod

```
#include <stdio.h>
#include <sys/time.h>

main(argc,argv)
int argc;
char **argv;
{
  struct timeval tv;
  struct timezone tz;

  gettimeofday(&tv,&tz);
  printf("The current time is:\n%s",
  ctime(&tv.tv_sec));
  if (tz.tz_dsttime)
    printf("Daylight savings time\n");
  else
    printf("Standard time\n");
  exit(0);

} /* POSIX Programmer's Guide,Donald A. Lewine, Oreilly */
```

Port Etme – Sorunlar

- `<sys/time.h>` başlık dosyası yok. (POSIX için)
- `struct timeval` ve `struct timezone` yapıları yok.
- POSIX `gettimeofday()` fonksiyonunu içermiyor.

Not: Hedef işletim sistemi, BSD 4.2 gibi yukarıdaki üç taşınabilirlik problemlerini içermiyorsa derleme sorunsuz gerçekleşir. Bu kodun taşınabilir olduğu anlamına gelmez. Bu durumları göz ardı ediyoruz.

Port etme - Çözümler

- **<sys/time.h>** yerine **<time.h>** yerleştirilir.
- **timeval** ve **timezone** yapılarına **gettimeofday()** fonksiyonu için ihtiyacımız var.
- **gettimeofday()'**un dönüş değeri, **ctime()** 'ın parametresi olarak kullanılıyor.
- Bu parametrenin eş değeri olarak **localtime(timer)** kullanılabilir. **localtime()**, zaman değerini **struct tm** yapısına çevirir.

Port etme - çözüm

tm yapısı <time.h> içinde şu şekilde tanımlanmıştır.

```
int tm_sec;     /* seconds (0 - 60) */
int tm_min;     /* minutes (0 - 59) */
int tm_hour;    /* hours (0 - 23) */
int tm_mday;    /* day of month (1 - 31) */
int tm_mon;     /* month of year (0 - 11) */
int tm_year;    /* year - 1900 */
int tm_wday;    /* day of week (Sunday = 0) */
int tm_yday;    /* day of year (0 - 365) */
int tm_isdst;   /* is summer time in effect? */
char *tm_zone;  /* abbreviation of timezone name */
long tm_gmtoff; /* offset from UTC in seconds */
```

isdst, 0'dan farklı ise yaz saati etkin demektir.

Sonuç

```
#define _POSIX_SOURCE 1

#include <stdio.h>
#include <time.h>
main(argc,argv)
int argc;
char **argv;
{
        struct tm *tmptr;
        time_t timer;

        timer = time(NULL);
        tmptr = localtime(&timer);
        printf("The current time is:\n%s",
        ctime(&timer));
        if (tmptr -> tm_isdst)
                printf("Daylight savings time\n");
        else
                printf("Standard time\n");
        exit(0);
} /* POSIX Programmer's Guide,Donald A. Lewine, Oreilly */
```

%100 uyumlu

Koddaki büyük taşınabilirlik sorunlarını çözerek “taşınabilir kod” elde ettik diyebiliriz artık. Ancak biraz daha dikkatle bakarsak hala taşınabilir olmayan kod parçacıkları var.

- Gerekli tüm başlıkları eklediğimizden emin miyiz? **`exit()`** **`<stdlib.h>`** gerektirir.
- **`exit(0)`** da başarılı dönüş olarak 0 yazdık. POSIX bunu **`EXIT_SUCCESS`** olarak tanımlamıştır. **`exit(EXIT_SUCCESS)`**
- Biraz açıklama eklemeyi ihmal etmemeli.
- Artık ULTRIX’e hatta VAX/VMS’e bile port ettik diyebiliriz.

Masaüstü Karışık Olanlara

Fonksiyonları bir bir taşınabilir olanları ile değiştirmek yerine #ifdef yapıları kullanarak kodu işletim sistemine göre seçimlik hale getirebilirsiniz.

```
#ifdef BSD
struct timeval tv;
struct timezone tz;
#else
struct tm *tmptr;
#endif
```

Bu durumda yönetilmesi gereken iki kod dalı vardır. 10000 satır bir programda bu yöntemi kullanırsanız 10001.satırı nereye ekleyeceğinizi bulmakta zorlanacaksınızdır.

POSIX'in Sundukları

• Arayüzler semboliktir. İşletim sistemine göre gösterim yapar. Örneğin süreç ID'si pid_t veya dosya hakkı mode_t olarak tanımlanmıştır. Bunlar hedef sisteme göre int, unsigned short gibi değerler alırlar.

• ioctl() gibi çok amaçlı fonksiyonlar, belirli bir görevi gerçekleştiren hususi birkaç fonksiyonla değiştirilmiştir.

Şablon Kullanın

Kodlar yeniden kullanılabilir olsun.

Program, tek bir kaynak dosyada olmasın. Dosyalara bölünmeli veya modüler yazılmalı. Bu şekilde farklı kişilerin kodu yürütmesi mümkün hale gelir. Modülerlik aynı zamanda yeniden kullanılabilirliğin de şartıdır.

Bir şablon kullanın.

```
/* Feature test switches */
#define _POSIX_SOURCE 1
/* System headers */
#include <stdio.h>
/* Local headers */
#include "global.h"
/* Macros */
#define AVARAGE (SUM / NELEM)
/* File scope variables */
static int workdir;
/* External variables */
extern int errno;
/* External functions */
int getword(char *line);
/* Structures and unions */
/* Signal catching functions */
/* Functions */
/* Main */
```

Sorunlar: G/Ç

Printf:

• Ekrana basılan şeyler sisteme göre küçük değişiklik gösterebilir. Örneğin %e en az 2 hane üs içermeli.

• %d %i %o %u %x ve %X dönüştürdükleri değerin int boyutuna sahip olduğunu varsayar. Eğer parametre short (küçük tamsayı) ise %hd %hi %ho %hu %hx ve %hX kullanın. Büyük tamsayı (long) ise %ld %li %lo %lu %lx ve %lX kullanın.

• Bazı sistemlerde printf() in bir defada ekrana basacağı karakter 509 ile sınırlıdır. Bazı sistemlerde bu değer daha yüksektir. Bu nedenle büyük buffer'lar birkaç adımda ekrana basılmalı.

Sorunlar: G/Ç

- Format stringleri güvenilir şekilde kullanın.

```
printf(str); /* yanlış */

printf("%s", str);  /* doğru */
```

- Ekrana basılan değer ile belirtilen format string aynı mı kontrol edilmeli. printf("Integer %d\n", str);

- putc bazen fputc 'ye tanımlı bir makro olabilir. Bu nedenle putc(i, file++) gibi bir kullanımdan kaçının. Makro tanımına bağlı olarak çalışmayabilir.

- scanf için okunacak veri büyüklüğü belirtilmeli. Yoksa bellek taşması olur ve verilerimizi kaybederiz. Bu büyüklük otomatik olarak yerleştirilen NULL'ı da içermeli.

Sorunlar: G/Ç

• fread ve fwrite gibi doğrudan dosyaya yazan/okuyan fonksiyonlar tamamen taşınabilirdir. Ancak yazdıkları veya okudukları veri taşınabilir olmayabilir. Bu nedenle okunan değerler doğru yorumlanmalı.

• Geçici dosyalar için tmpnam ve tmpfile() kullanın.

• Dosya isimleri de taşınabilir olmalı:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9 0 . _ -
```

“-” sembolü en başta olmamalı. Ayrıca dosya ismi en fazla 14 karakter olmalı. Eğer dosya ismi dışardan okunuyorsa bunlar kontrol edilmelidir.

Sorunlar: Süreçler

fork()

• Yeni süreç oluşturmak için kullanılan fork çağrısı taşınabilir bir çağrıdır. BSD, bu işlemi hızlandırmak için vfork() çağrısına sahiptir. Bu tamamen taşınabilir değildir.

```
#ifdef BSD
pid = vfork();
#else
pid = fork();
#endif
```

• Değer vermeden return yapmak taşınabilir değildir.

• _exit yerine C kütüphanesi ile gelen exit() kullanmak daha taşınabilir özelliktedir.

Sorunlar: Süreçler

• Ölümcül hata oluşmadıkça abort() ile sonlandırma yapılmamalı. Tabi core dosyası oluşturmak için gerekli.

• Sinyal yönetiminde eski uygulamaların pek çok sorunu var. BSD bunları düzeltti. POSIX BSD sinyalleri üzerine bina edilmiştir.

• Sinyal yakalandığında sinyal işleyicisi çalışır. Sinyal geldiğinde program bir fonksiyonun ortasında olabilir. Bu nedenle güvenli olmayan fonksiyonlar sinyal işleyici tarafından çağrılmamalı. Örneğin printf ve türevleri güvenli değildir, unlink güvenlidir.

Sorunlar: Endians

Tek bir byte alana sığmayan sayılar bellekte belli bir sırayla ardışıl byte'lara yerleştirilir. Sayının en anlamlı bitlerinin önde olması veya en anlamsız bitlerinin önde olması platformlara göre değişir. Bu sıra little-endian ve big-endian olarak adlandırılır.

```
int i = 4;
char c = *(char *) i;
```

Sayılar bir dosyaya yazıldığında veya iki platform arasında ağdan geçtiğinde bu dönüşümlerin yapılması lazımdır. Yoksa en anlamlı byte ile en anlamsız byte yer değiştirmiş olur ve farklı bir sayı okunmuş olunur.

```
write (fd, &i, sizeof i);
```
/* Taşınabilir değil. */

Sorunlar: Endians

Yukarıdaki kod taşınabilir değildir. Çünkü ‘i’ nin her platformda boyutunun aynı olduğunu varsayıyor. Basit bir çözüm:

```
int j;
 char buf[4];
 for (j = 0; j < 4; ++j)
   buf[j] = (i >> (j * 8)) & 0xff;
 write (fd, buf, 4);
```

Diğer bir çözümde adres/port dönüşümleri için çokça kullanılan **`htons()`** ve **`ntohs()`** fonksiyonlarını kullanmaktır. TCP/IP big-endian kullanır. Dolaysıyla big-endian sistemlerde aldıkları parametreyi aynen döndürür, little-endian sistemlerde ise ardışıl byte’ların yerini değiştirir.

GNU Autotools

GNU Autotools (Autoconf, automake, libtool) yazılımları daha taşınabilir hale getirmek, farklı platformlarda kolaylıkla program yükleyebilmek için araçlar sunmaktadır.

Autotools

Autoconf, sisteminiz üzerinde bazı testler yaparak sisteminizin karakteristiklerini keşfetmeye çalışır. Bu işlemler kodun derlemesinden önce yapılır. Bu nedenle kaynak kodu sistemin karakteristiklerine göre değiştirebilir.

Automake, Makefile oluşturur.

Libtool, derleyici ve linker arasında bir arayüzdür. Üzerinde çalışılan platformun ne olduğunu bilmeden kolaylıkla taşınabilir statik ve shared kütüphane oluşturmaya yardımcı olur.

Makefile

```
target1:  dep1 dep2 ... depN
<tab>   cmd1
<tab>   cmd2
<tab>   ...
<tab>   cmdN

target2:  dep4 dep5
<tab>   cmd1
<tab>   cmd2

dep4 dep5:
<tab>   cmd1
```

Örnek

```
qsheff: spool
        @echo "$@, djb gerektirir..."

spool:
        mkdir $@

# make qsheff
mkdir spool
qsheff, spool gerektirir...
```

Configure.in

Sistem karakteristiğini test etmek için yapılacak işlemler merkezi olarak configure.in den yönetilir. Configure.in geliştirici tarafından oluşturulur. Bu dosya kullanılarak configure betiği oluşturulur. Programı kuran sistem yöneticileri bu betiği kullanır.

```
AC_CHECK_FUNCS(inet_aton inet_addr, break)
#if HAVE_INET_ATON
  inet_aton kullanilacak kod parçası
#else
#if HAVE_INET_ADDR
  inet_addr kullanılacak kod parçası
#else
#error Function missing!
#endif
#endif
```

UNIX'ler Arası Fonksiyonlar

POSIX.1 'in tanımlamadığı fonksiyonlar tüm UNIX'lerde olmayabilir. Bunları kontrol etmek için configure.in içerisinde AC_CHECK_FUNCS yapısı kullanılabilir. Tüm sistemlerde olmayan bazı fonksiyonlar:

Alloca, dlopen, getline, getpagesize, gettimeofday, mmap, ptrace, setuid, **snprintf**, **strcasecmp**, strncasecmp, strdup, valloc, vfork

Configure.in

Configure.in:

```
AC_CHECK_FUNCS(strcpy bcopy)
```

*.h (kaynak kod tarafından dahil edilen):

```
#if !HAVE_STRCPY
#  if HAVE_BCOPY
#    define strcpy(dest, src)   bcopy (src, dest, 1
+ strlen (src))
#  else /* !HAVE_BCOPY */
     error no strcpy or bcopy
#  endif /* HAVE_BCOPY */
#endif /* HAVE_STRCPY */
```

autoscan

Kod içerisindeki tüm kütüphane fonksiyonlarını gözden geçirip taşınabilir/taşınamaz şekilde geliştiricinin işaretlemesi zor olacaktır. Autotools bunun için ‘autoscan’ aracını sunmaktadır. Autoscan, sizin için tüm kaynak kodları inceleyerek bilenen taşınabilirlik problemleri için bir configure.in oluşturur. Geliştirici kendi kurallarını buna ilave eder

autoscan

```
AC_INIT(sic/eval.h)

# Checks for programs.

# Checks for libraries.

# Checks for header files.
AC_HEADER_STDC
AC_CHECK_HEADERS(strings.h unistd.h)

# Checks for typedefs, structures, and compiler
characteristics.
AC_C_CONST
AC_TYPE_SIZE_T

# Checks for library functions.
AC_FUNC_VPRINTF
AC_CHECK_FUNCS(strerror)
AC_OUTPUT()
```

Kaynaklar

Bu doküman aşağıdaki kaynaklar esas alınarak hazırlanmıştır.

POSIX Programmer's Guide, Writing Portable UNIX Programs with the POSIX.1 Standard, Donald A. Lewine

ClamAv Projesi Configure.in dosyasi, http://www.clamav.net/

- Son -

AÇIK AKADEMİ

endersys
Danışmanlık, Yazılım ve İletişim Teknolojileri

Bu sunumu aşağıdaki adresten temin edebilirsiniz:

http://www.enderunix.org/slides/