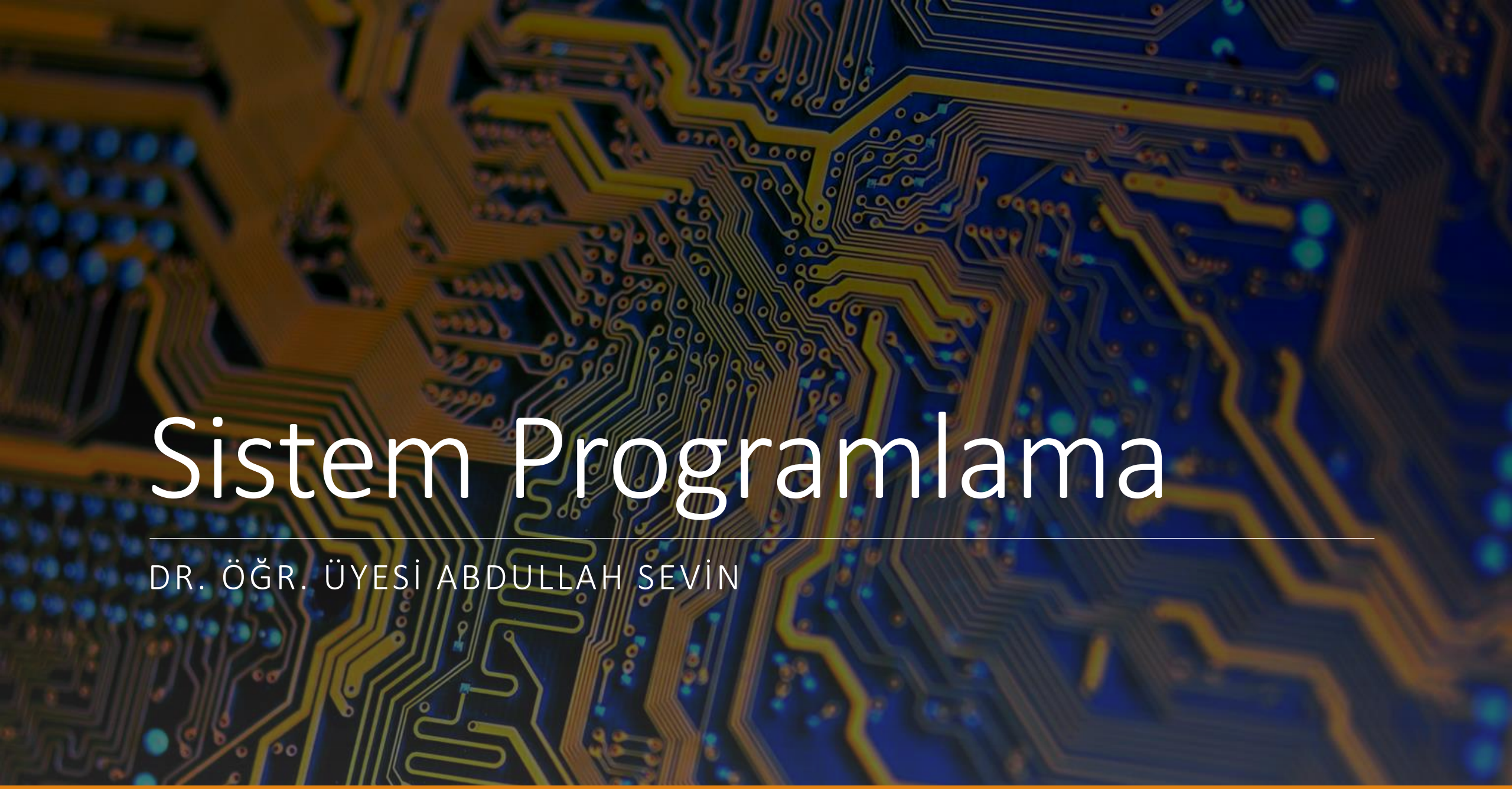


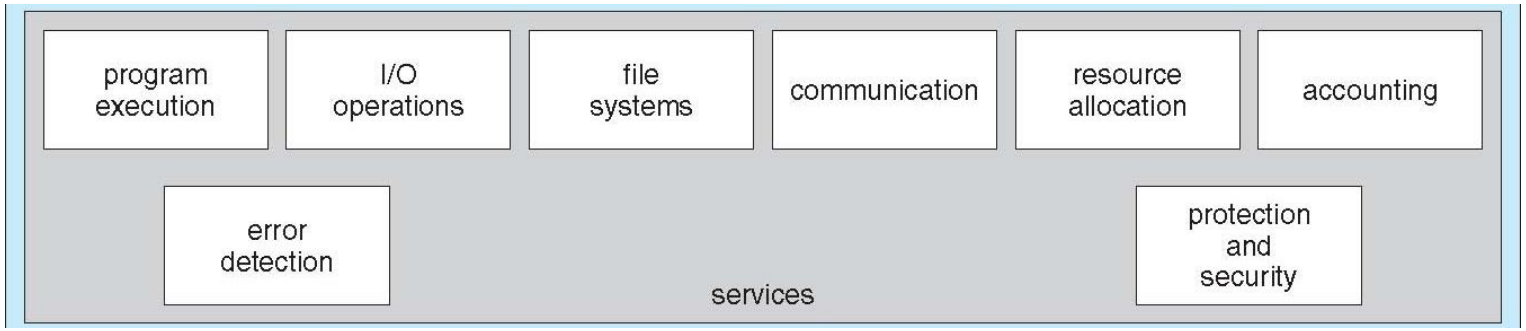
Sistem Programlama

DR. ÖĞR. ÜYESİ ABDULLAH SEVİN

Sistem Programlama nedir?

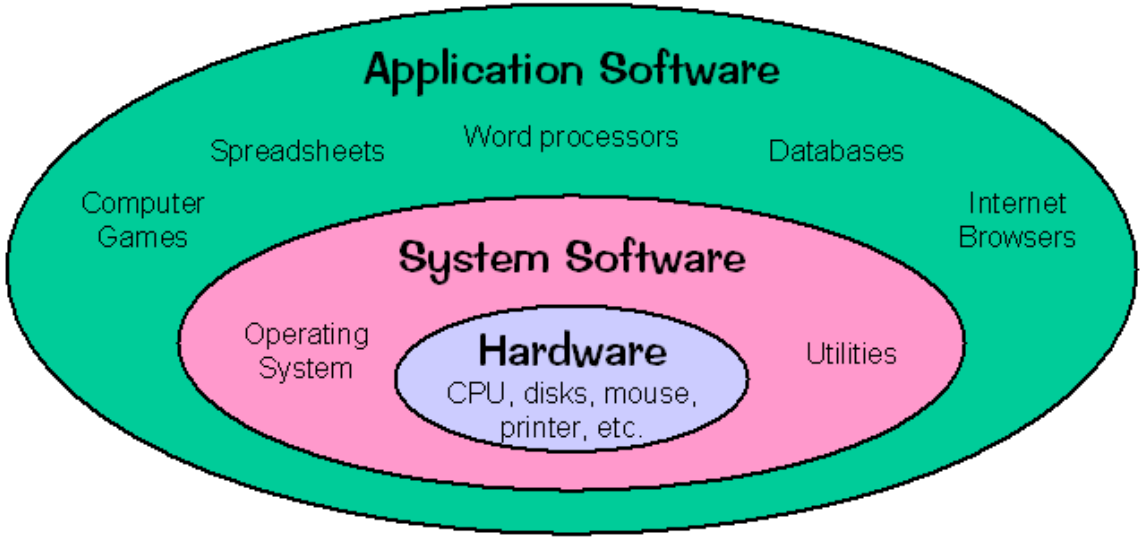
- **Sistem programlama:** program geliştirme için sistem araçlarının kullanılmasıdır.
- Programlama bilgilerine *ek* olarak, *bilgisayar mimarisi ve işletim sistemi* bilgisi
- **Uygulama programları** ile direkt olarak kullanıcıya servisler sağlanırken,

  Sistem programları ile *uygulama programlarına servisler sağlanır*.
- Sistem programlamada, işletim sistemi servisleri ile etkileşim kuran programlar yazılır.



Sistem Programlama

- **Bilgisayar**; donanım, sistem yazılımı ve uygulama yazılımı olmak üzere üç ana parçaya ayrılabilir.
- **Donanım ve sistem yazılımı,** uygulama programlarını çalıştırmak için kullanılır.
- **Sistem yazılımı,** donanım ve uygulama yazılımları arasında bir soyutlama katmanı oluşturur.



https://www.cseworldonline.com/articles/difference-between-application-software-and-system-software.php

Sistem Programlama

- **Sistem yazılımı**, bilgisayar donanımının çalıştırılması için kullanılır.
- **Sistem yazılımı**, bir bilgisayar sisteminin donanım bileşenlerini kontrol etmek, bütünleştirmek ve yönetmekten sorumludur,
- böylece diğer yazılımlar ve sistem kullanıcıları, veri aktarımı gibi düşük seviyeli ayrıntılarla ilgilenmek zorunda kalmazlar. (Bellekten diske kopyalama veya bir ekrana yazı yazdırma vs.)

Sistem Programlama

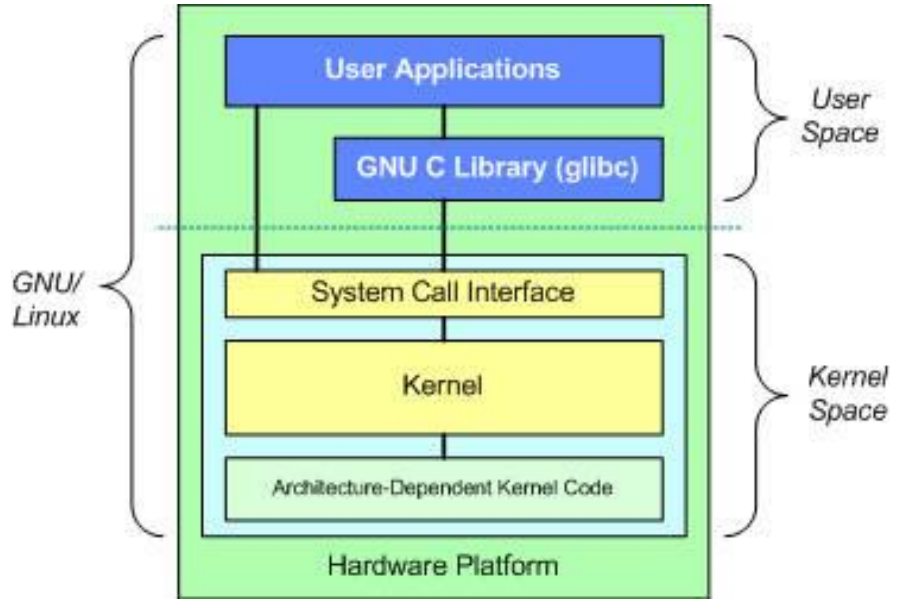
- Genel olarak, sistem yazılımı bir **işletim sisteminden** ve **disk biçimlendiriciler**, **dosya yöneticileri, görüntü yöneticileri, metin düzenleyiciler, kullanıcı kimlik doğrulaması (oturum açma)** ve **yönetim araçları ve ağ ve cihaz kontrol yazılımı** gibi bazı temel yardımcı programlardan oluşur.
- Genelde kullanıcılar, arka planda çalıştığı için sistem yazılımları ile etkileşime girmezler.

Hedefler

1. Bir işletim sisteminin temellerini kullanıcı açısından anlamak.
2. Neden C'de programlama yaptığımızı anlayacak kadar iyi C programlamak.
3. Sistem programlama hakkında bilgili olmak.
4. Temel tek işlemcili bilgisayar organizasyonunu anlamak.
5. Sistem hakkında donanımsal bilgi sahibi olmak
6. Donanımı gerektiğinde kontrol edebilmek

Sistem Programlama

- Sistem kütüphanelerinin oluşturduğu soyutlama katmanı (abstraction) ile bir fonksiyonu (Ör: printf) donanımın detaylarını bilmeden kullanabiliriz.
- Yanda verilen Linux mimarisinde görülen GNU C kütüphanesi gibi, bir sistem benzer kütüphaneleri içerdiği sürece uygulama o sistem üzerinde kullanılabilir.
- Sistem araçlarının kullanımı standartların oluşmasını sağlar ve böylece geliştirilen programlar diğer bilgisayarlara kolayca transfer edilebilir.



https://developer.ibm.com/articles/l-linux-kernel/

Neden C dili

- **Sistem programlamada C kullanılmasının sebepleri:**
- C programı; işletim sistemleri, aygıt sürücüleri, ağ sunucuları gibi uygulamalarda kullanılarak modern bilgisayarın temelini oluşturur.
- C programlama en az soyutlamaya sahiptir. Bundan dolayı donanıma daha yakındır.
- Birçok C ifadesi direkt olarak makine koduna dönüştürülebilir.
- C programlamada hafızaya işaretçiler (pointers) aracılığı ile erişilebilir ve böylece sistemin parçalarına ulaşma imkanı sağlanır.

Neden C

- Yalnızca dilin veya sistemin mevcut olduğu birçok durumda kullanılabilir.
  - Küçük, gömülü sistemler vs.
- "Üst düzey dilleri" desteği olmayan birçok "düşük düzey" durum var.
  - İşletim sistemleri, gerçek zamanlı sistemler, sürücüler, IoT cihazları..

✅ Cross-Platform Timing Solution

- This code works on Ubuntu, Raspberry Pi, and Windows without modifications:

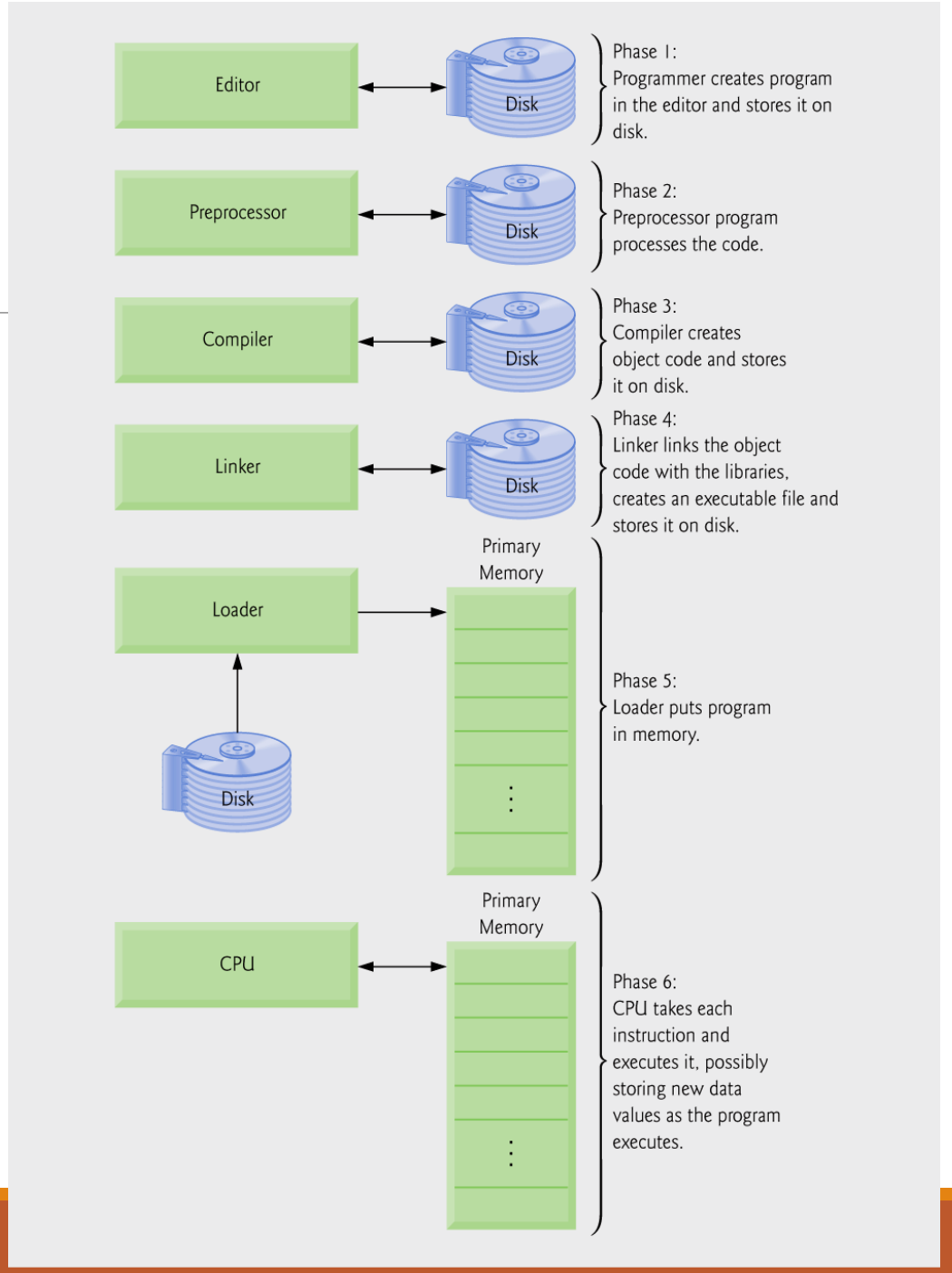
```
#include <iostream>
#include <chrono>
int main() {
    // Get current time point
    auto start = std::chrono::high_resolution_clock::now();
    // Simulate some work
    for (volatile int i = 0; i < 1000000; ++i);
    // Get end time
    auto finish = std::chrono::high_resolution_clock::now();
    // Calculate duration in nanoseconds
    auto duration = std::chrono::duration_cast<std::chrono::nanoseconds>(finish - start).count();
    std::cout << "Execution time: " << duration << " nanoseconds" << std::endl;
    return 0;
}
```

C prog. Geliştirme ortamı

Standard Adımlar

- Düzenleme
- Önişlemci
- Derleme
- Bağlama
- Yükleme
- Yürütme

Deitel & Deitel



C prog. Geliştirme ortamı

- Edit: Düzenleme
- Önişlemci: Derleme sürecinde ayrı bir adımdır. Basit bir ifadeyle, bir C Ön İşlemcisi yalnızca bir **metin değiştirme** aracıdır ve derleyiciye gerçek derlemeden önce gerekli ön işlemeyi yapmasını söyler. Tüm önişlemci komutları bir kare simgesiyle (#) başlar. (#define,..)
- Derleme…
- Bağlama…
- Yükleme…
- Yürütme…

Önişlemci

timing.h

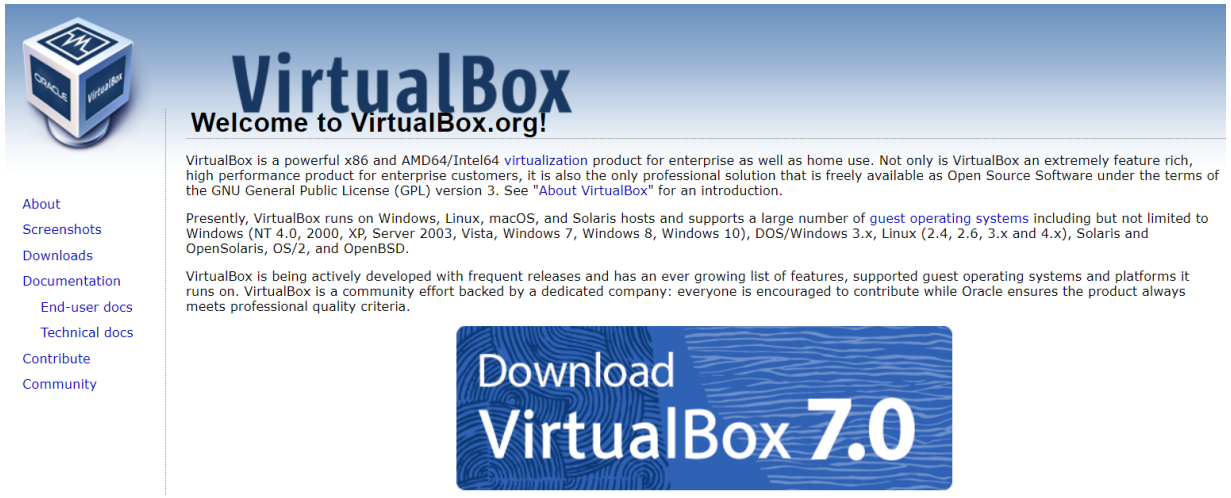
```
#pragma once

typedef unsigned int uint_32t;


/************** Timing routine (for performance measurements) ***********/
/* By Doug Whiting */
/* unfortunately, this is generally assembly code and not very portable */
#if defined(_M_IX86) || defined(__i386) || defined(_i386) || defined(__i386__) || defined(i386) ||
    defined(_X86_)   || defined(__x86_64__) || defined(_M_X64) || defined(__x86_64)
#define _Is_X86_    1
#endif

#if  defined(_Is_X86_) && (!defined(__STRICT_ANSI__)) && (defined(__GNUC__) || !defined(__STDC__))
    (defined(__BORLANDC__) || defined(_MSC_VER) || defined(__MINGW_H) || defined(__GNUC__))
#define HI_RES_CLK_OK         1          /* it's ok to use RDTSC opcode */

#if defined(_MSC_VER) // && defined(_M_X64)
#include <intrin.h>
#pragma intrinsic(__rdtsc)         /* use MSVC rdtsc call where defined */
#endif

#endif
```

Neden Linux?

- **Açık kaynak** olması sebebiyle, sistem programlama dersinde Linux tabanlı işletim sistemini tercih edilmektedir.
- Linux tabanlı işletim sisteminin çalışmasıyla ilgili detaylar incelenebilir. **Parçalar eklenebilir veya değiştirilebilir**.
- Ticari olmamasından dolayı kapalı bir sistem değildir.

İşletim Sistemi

- Ders ile ilgili uygulamaları gerçekleştirmek için Linux tabanlı Ubuntu İşletim sistemini kullanabilirsiniz.
- Farklı İşletim Sistemi üzerine kuracak iseniz Sanal Makine kullanmalısınız. (Örn: Windows üzerinde Oracle VM VirtualBox)



Konular

Derse ait temel başlıklar aşağıdaki gibi sıralanabilir:

-C programlama dili hakkında temel bilgiler

-C işaretçiler, malloc, stringler, vb...

-Dosyalar ve dizin dosyaları, Sinyaller

-Bağlantılar (linkler), Shell yönlendirme (Shell Redirection)

-Dosyadan okuma/yazma (File I/O), sistem çağrıları ve tampon bellek (buffer) kullanımı

-Sistem çağrıları ve giriş çıkış

-Simgesel dil (Assembly) (yerel değişkenler, fonksiyonlar, dallanma)

-Prosesler (süreçler) ve ilgili sistem çağrıları (fork, exec, dup, pipe)

-Prosesler arası iletişim, sinyaller

Değerlendirme

1. Ara Sınav	%60
1. Kısa Sınav	%10
1. Proje / Tasarım	%30
1. Final	%50

Kaynaklar

http://web.eecs.utk.edu/~huangj/cs360/index.html

CS360 -- Systems Programming

Jian Huang --- Fall 2022

CS360 LINKS

- General Information
- Syllabus
- Homeworks
- Labs
- Lecture Notes
- The Lab Home Page (maintained by the TA's)

INTRODUCTION AND CLASS GOALS

Kaynaklar

http://web.eecs.utk.edu/~jplank/plank/classes/cs360/lecture_notes.html

CS360 -- Systems Programming

James S. Plank

Lecture Notes

Although you can read the notes here, I have put them on bitbucket, and you can grab them to compile and use on your own machine. I strongly suggest that you do so, so that you have all of the programs on your own machine, do:

```
UNIX> git clone https://jimplank@bitbucket.org/jimplank/cs360-lecture-notes.git
```

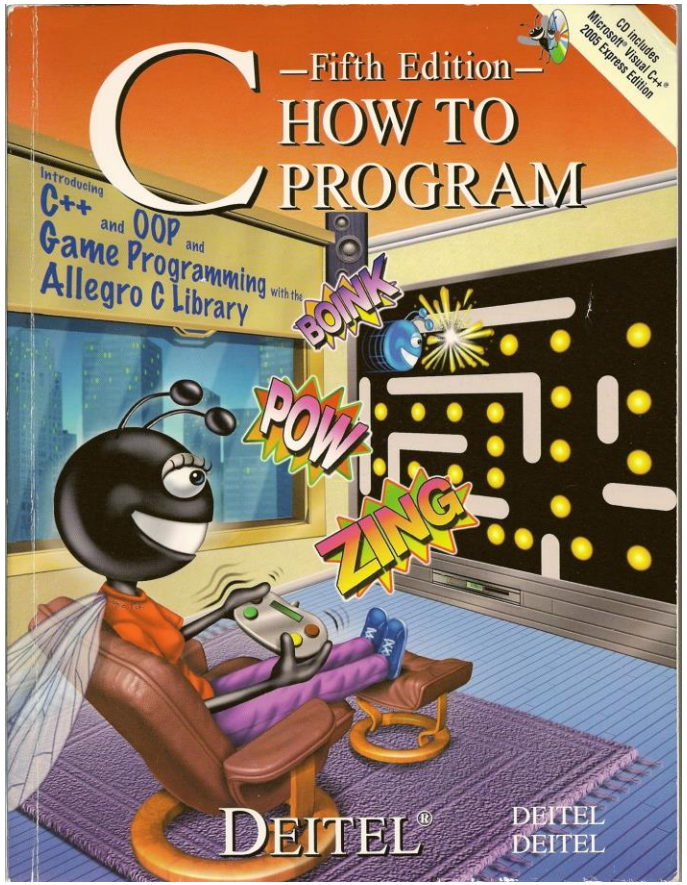
The lecture notes will be in the directory **cs360-lecture-notes**. You should occasionally do a "git pull" in that directory, to make sure that your notes are up to date.

Lecture notes on bitbucket

- Lecture 1: "C Stuff 1:" Getting Started with C, Scalar Types and Aggregate Types
- Lecture 2: "C Stuff 2:" Pointers, Casting, Malloc, Segmentation Violations and Bus Errors
- Lecture 3: Pointer Arithmetic (Small Lecture)
- Lecture 4: Strings in C
- Lecture 5a: Libfdr -- The Code
- Lecture 5b: Libfdr -- Fields
- Lecture 5c: Libfdr -- Jvals
- Lecture 5d: Libfdr -- Dllist: Doubly-Linked Lists
- Lecture 5e: Libfdr -- JRB: Red-Black Trees
- Lecture 6: Some Basic Terminology
- Lecture 7: Introduction to System Calls and I/O
- Lecture 8: Cat and Buffering
- Lecture 9: Links
- Lecture 10: Sh Redirection
- Lecture 11: Stat and Opendir/Readdir/Closedir

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Ek Kaynak



Deitel & Deitel, 5th ed.

C & C++

Paul Deitel (Author), Harvey Deitel (Author)

Ek Kaynak

Deitel & Deitel, 6th ed.

C & C++

Similar to 5th edition.