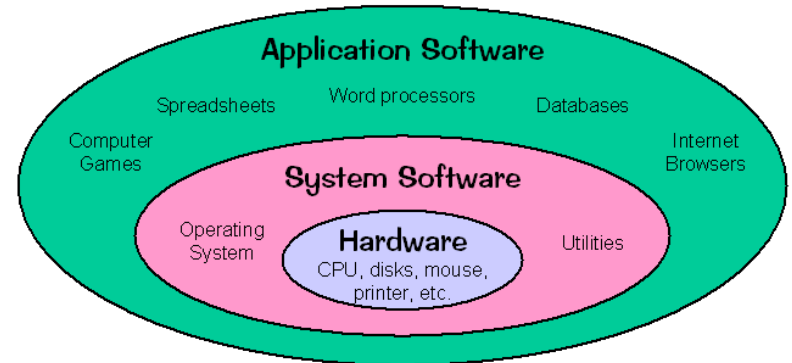


# Sistem Programlama nedir?

- ❑ **Sistem programlama:** program geliştirme için sistem araçlarının kullanılmasıdır.
- ❑ Programlama bilgilerine ek olarak, bilgisayar mimarisi ve işletim sistemi hakkında bilgi sahibi olmayı gerektirir.
- ❑ 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.

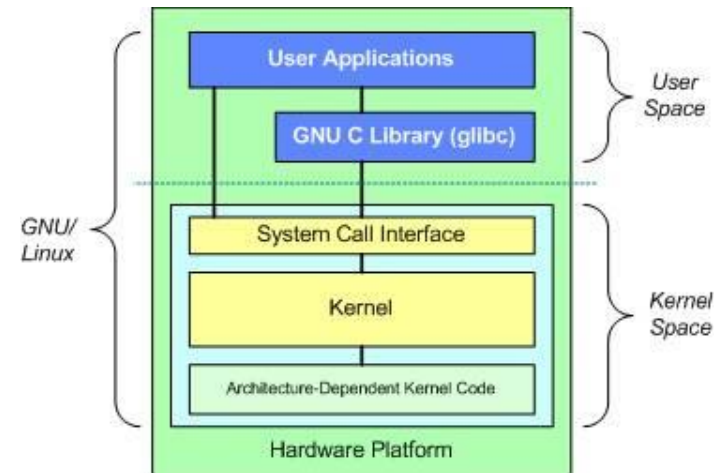
# Sistem Programlama

- 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.



# Sistem Programlama

- ❑ Sistem kütüphanelerinin oluşturduğu soyutlama katmanı (abstraction) ile bir fonksiyonu donanımın detaylarını bilmeden kullanabiliriz.
- ❑ Örneğin 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 edilir.



Kaynak: <http://www.ibm.com/developerworks/library/l-linux-kernel/>

# Sistem programlamada neden C

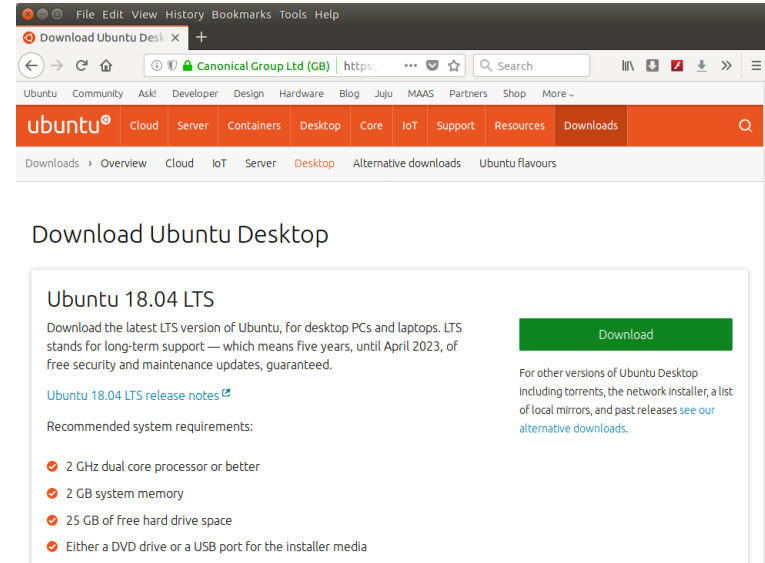
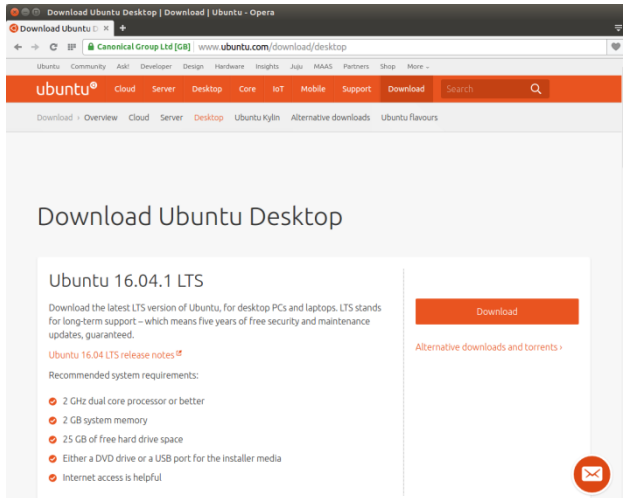
- Sistem programlamada C kullanılmasının sebepleri:
- C programı; işletim sistemleri, aygıt sürücüler, ağ sunucuları gibi uygulamalarda kullanılarak modern bilgisayarın temelini oluşturur.
- C programlama en az soyutlamaya sahiptir. Bundan dolayı donanıma daha yakındır.
- Birçok C ifadesi direkt olarak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

# Sistem programlamada 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 işletim sistemini kullanabilirsiniz.



<https://www.ubuntu.com/download/desktop>

# Temel Başlıklar

**Derse ait temel başlıklar aşağıdaki gibi sıralanabilir:**

- C programlama dili hakkında temel bilgiler
- C İşaretçiler, malloc, stringler, vb...
- Dosyalar ve dizin dosyaları, Sinyaller
- Bağlantılar (linkler), Shell yönlendirme, Shell Script
- 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

Çalışma Tipi	Oran
1. Ara Sınav	%75
1. Proje / Tasarım	%10
1. Performans Görevi (Uygulama)	%10
2. Performans Görevi (Uygulama)	%5
1. Final	%50

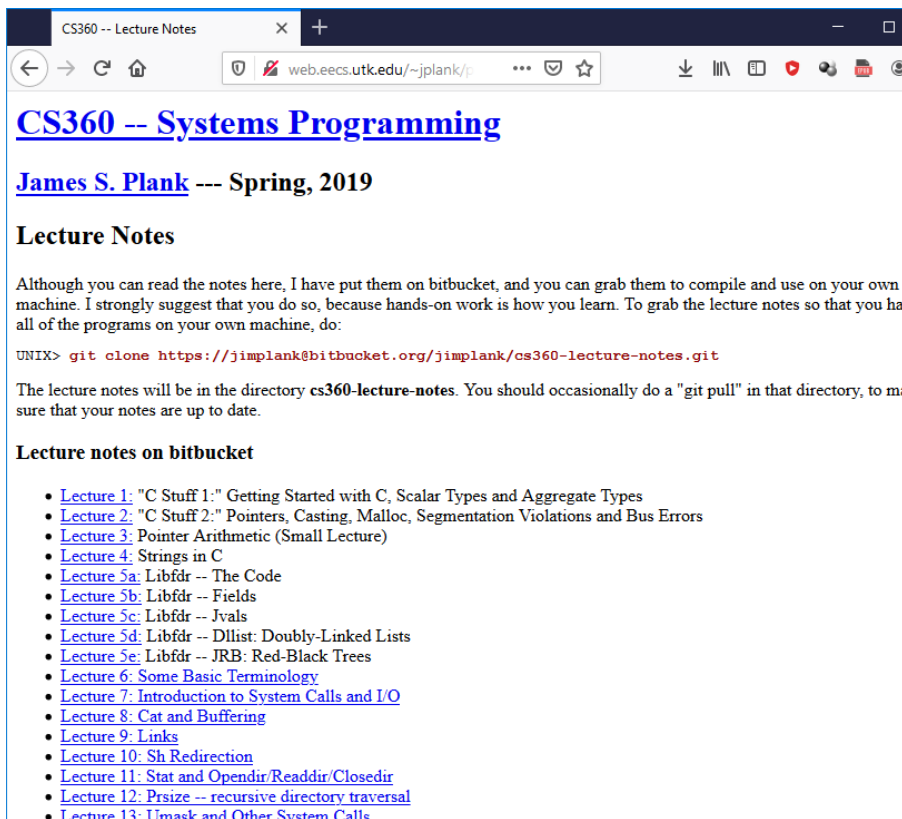


# Kaynaklar

- **Sabis ders dokümanları**

 DOKÜMANLAR

- [http://web.eecs.utk.edu/~plank/plank/classes/cs360/lecture\\_notes.html](http://web.eecs.utk.edu/~plank/plank/classes/cs360/lecture_notes.html)
- [http://web.eecs.utk.edu/~huangj/cs360/lecture\\_notes.html](http://web.eecs.utk.edu/~huangj/cs360/lecture_notes.html)



CS360 -- Lecture Notes

[web.eecs.utk.edu/~jplank/](http://web.eecs.utk.edu/~jplank/)

## CS360 -- Systems Programming

James S. Plank --- Spring, 2019

### Lecture Notes

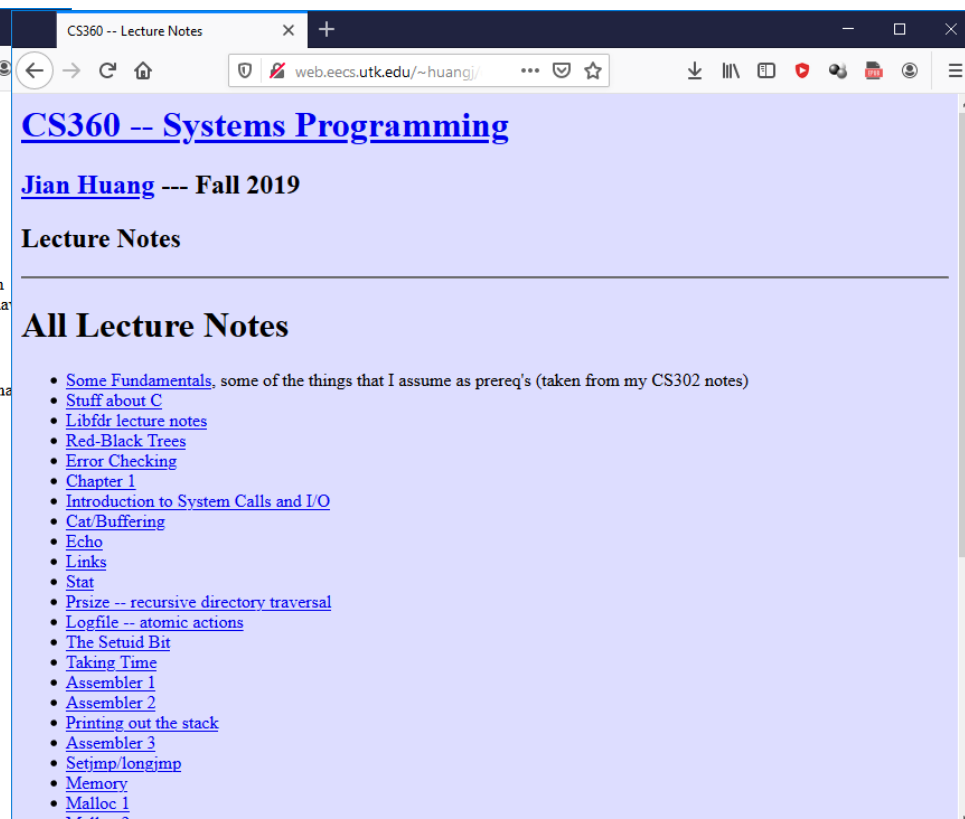
Although you can read the notes here, I have put them on bitbucket, and you can grab them to compile and use on your own machine. I strongly suggest that you do so, because hands-on work is how you learn. To grab the lecture notes so that you have all of the programs on your own machine, do:

```
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 -- Dlist: 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
- [Lecture 12](#): Prsize -- recursive directory traversal
- [Lecture 13](#): Umask and Other System Calls



CS360 -- Lecture Notes

[web.eecs.utk.edu/~huangj/](http://web.eecs.utk.edu/~huangj/)

## CS360 -- Systems Programming

Jian Huang --- Fall 2019

### Lecture Notes

---

### All Lecture Notes

- [Some Fundamentals](#), some of the things that I assume as prereq's (taken from my CS302 notes)
- [Stuff about C](#)
- [Libfdr lecture notes](#)
- [Red-Black Trees](#)
- [Error Checking](#)
- [Chapter 1](#)
- [Introduction to System Calls and I/O](#)
- [Cat/Buffering](#)
- [Echo](#)
- [Links](#)
- [Stat](#)
- [Prsize -- recursive directory traversal](#)
- [Logfile -- atomic actions](#)
- [The Setuid Bit](#)
- [Taking Time](#)
- [Assembler 1](#)
- [Assembler 2](#)
- [Printing out the stack](#)
- [Assembler 3](#)
- [Setjmp/longjmp](#)
- [Memory](#)
- [Malloc 1](#)
- [Malloc 2](#)