Server.c nasıl çalışıyor adım adım açıklama:
1) main açıklayarak başlayalım
struct sigaction act:
signal.h kütüphanesiyle gelen bir yapıdır. Sigaction fonksiyonunun gelen sinyalleri nasıl ele alacağını özelleştirmek için act'i kodumuzda tanımlıyoruz ve ardından istediğimiz atamaları yapıyoruz.
act.sa_sigaction = ft_handler ile gelecek sinyallerin hangi handler ile işleneceğini belirtiyoruz.
act.sa_flags = SA_SIGINFO ile XXXXXXXXXXX
sigemptyset(&act.sa_mask) ile XXXXXXXXXXX

if ifadelerinin içinde sigaction fonksiyonunu çağırıyoruz. Sigaction server'a gelen sinyalleri belirttiğimiz özel durumlara göre işleyecek ve sonucunda sayısal bir değer döndürecek. Bu değer -1 ise bir hata olduğunu belirten yazı yazdırıyoruz. Belirttiğimiz özel durumlara göre gelen sinyaller ft_handler da işlenecekti bu fonksiyonun detaylarına sonra geleceğim.

If kondisyonlarından sonra getpid ile PID numarasını ekrana yazdırıyorum.
While loop ile sonsuz bir döngü açıyorum ve pause() fonksiyonu ile sinyal alana kadar işlemleri askıya alıyorum.

2) ft_handler nasıl çalışır?
siginfo_t nedir?
SA_SIGINFO bayrağı sayesinde siginfo_t *info parametresi ile gönderen işlemin PID'si gibi detaylı bilgilere erişebiliyoruz.

fonksiyonuma kullanıcı tarafından gelebilecek 2 tür sinyalim var bunlar SIGUSR1 ve SIGUSR2.
Bu fonksiyon client tarafından gönderilen her bir karakter için tek tek bitleri işleyecek.

(void)content yapmamızın sebebi unused variable uyarısı almamak.
KULLANMAYACAKSAK O ZAMAN NEDEN FONKSİYONA VERİYORUZ????
çünkü sigaction'ın standart handler'ı şu şekilde çalışıyor:
void handler(int signum, siginfo_t *info, void *ucontext)

ilk if kondisyonunun adım adım çalışma şekli:
İlk sinyal `SIGUSR1` → Bit 0: **1** → **`i |= (1 << 0)`** ile `i = 00000001`.

- İkinci sinyal `SIGUSR2` → Bit 1: **0** → Hiçbir şey yapılmaz.
- Üçüncü sinyal `SIGUSR2` → Bit 2: **0** → Hiçbir şey yapılmaz.
- Dördüncü sinyal `SIGUSR1` → Bit 3: **1** → **`i |= (1 << 3)`** ile `i = 00001001`.
- Beşinci sinyal `SIGUSR2` → Bit 4: **0** → Hiçbir şey yapılmaz.
- Altıncı sinyal `SIGUSR1` → Bit 5: **1** → **`i |= (1 << 5)`** ile `i = 00101001`.
- Yedinci sinyal `SIGUSR1` → Bit 6: **1** → **`i |= (1 << 6)`** ile `i = 01101001`.
- Sekizinci sinyal `SIGUSR2` → Bit 7: **0** → Hiçbir şey yapılmaz.

Ayrıca her bit işlendiğinde kill ile server'dan client'a bit'in alındığına dair bir onay sinyali gönderilir.

*ÖNEMLİ SORU: Server'dan client'a gönderilen bu onay sinyali client tarafından nasıl işleniyor?

Client.c nasıl çalışıyor adım adım açıklama:
1) Main içeriği ile başlayalım:
signal(SIGUSR1, flag_handler) ile flag_handler'e SIGUSR1 sinyalini gönderiyoruz.
signal == SIGUSR1 yani 1 ise g_flag = 1 yapıyoruz. Bunu daha sonrasında send_signal içinde kullanacağız.
While loop ile mesajımız bitene kadar her bir karakteri send_signal'e gönderiyoruz ve en son null terminator gönderiyoruz.
2) send_signal çalışma mantığı
server.c dekine benzer bir şekilde bit shifting sonrası karakterin binary karşılığındaki mevcut bit konumuyla kesişim var ise 1 yok ise 0 sinyali gönderiyoruz.

Burada while loop ve g_flag mantığı şu şekilde:
Her bir bit gönderilirken, client server'dan bir SIGUSR1 sinyali bekliyor.
while (!g_flag) döngüsü, server'dan onay sinyali (SIGUSR1) gelinceye kadar client'ın beklemesini sağlıyor.
flag_handler() fonksiyonu SIGUSR1 sinyalini aldığında g_flag'i 1 yapıyor.
Bu durumda while (!g_flag) döngüsü duruyor ve bir sonraki bite geçiliyor.
Sonra g_flag tekrar 0'a reset ediliyor.
---kill ile ilk bitimi gönderdim, g_flag'im başta 0 olduğu için while loop burda duruyor. Serverdan SIGUSR1 sinyali geldiği anda g_flag 1 oluyor ve bir sonraki bit'e geçiyoruz.

volatile sig_atomic_t g_flag = 0 ne demek?

`volatile sig_atomic_t g_flag = 0`

- **Tanım:** `volatile`, değişkenin birden fazla işlemci veya donanım tarafından değiştirilebileceğini ifade eder. `sig_atomic_t` ise atomik bir işlemle okunan ve yazılan bir türdür (kesintisiz).
- **Kütüphane:** `signal.h`.
- **Nerede Kullanılıyor:**
  - `flag_handler` fonksiyonunda `g_flag` ayarlanır.
  - `send_signal` içinde sinyal onayının alınmasını kontrol etmek için kullanılır.
- **İşlev:** İstemcinin, server tarafından gönderilen sinyal onayını beklemesini sağlar.

`volatile` Anahtar Kelimesi:

- Derleyiciye, bu değişkenin beklenmedik şekilde değişebileceğini bildirir.
- Derleyicinin optimize etme işlemlerini engeller ve her okuma/yazma işleminde değişkenin güncel değerini kullanmasını sağlar.
- Sinyaller gibi harici olaylarla değişebilen değişkenlerde kritik öneme sahiptir.

`sig_atomic_t` Türü:

- Sinyaller tarafından güvenli bir şekilde okunup yazılabilen bir tür.

- Atomik işlemler garanti eder, yani değişken üzerindeki işlemler bölünemez.
- Çoklu thread veya sinyal işleme senaryolarında veri bütünlüğünü korur.