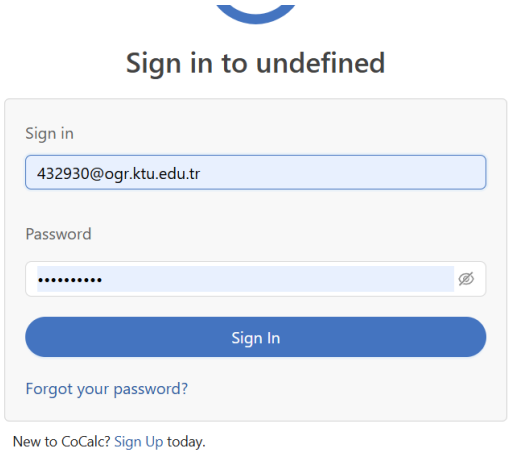
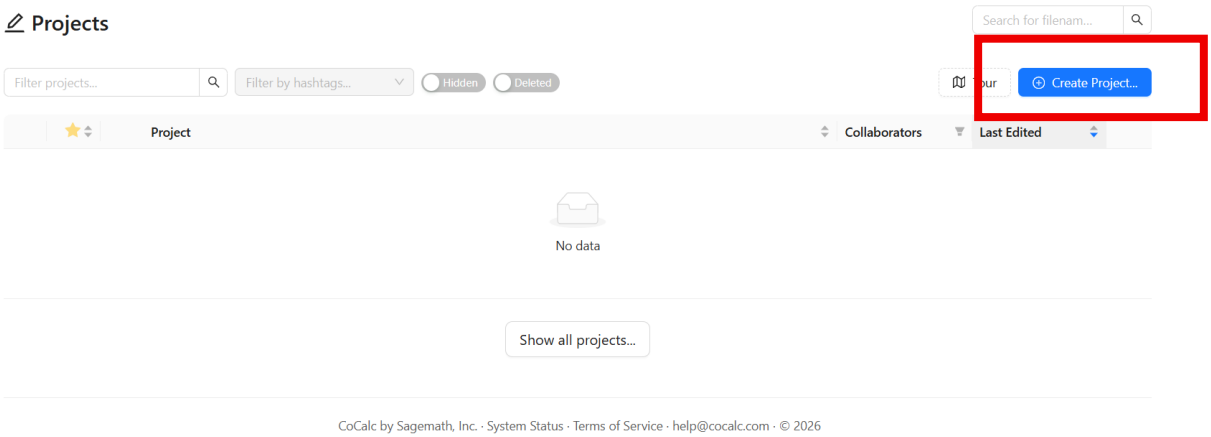
SÜREÇLERARASI HABERLEŞME DENEYİ YAPILIŞI

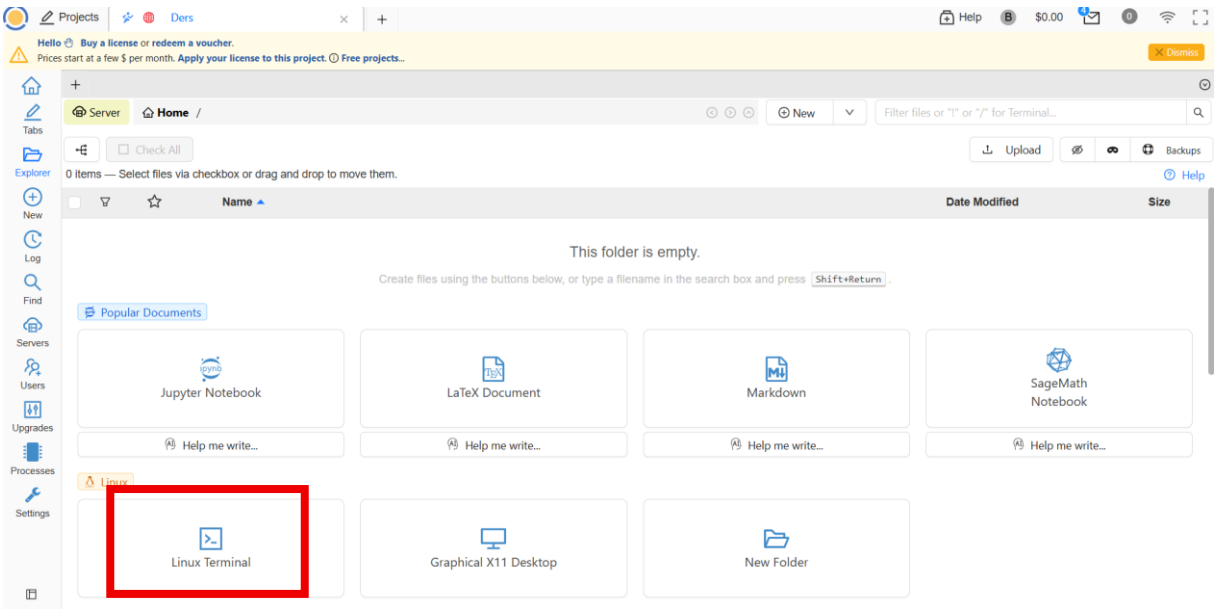
Öncelikle COCALC uygulamasına girip öğrenci maillerinizle üye olup uygulamaya giriş yapmalısınız.



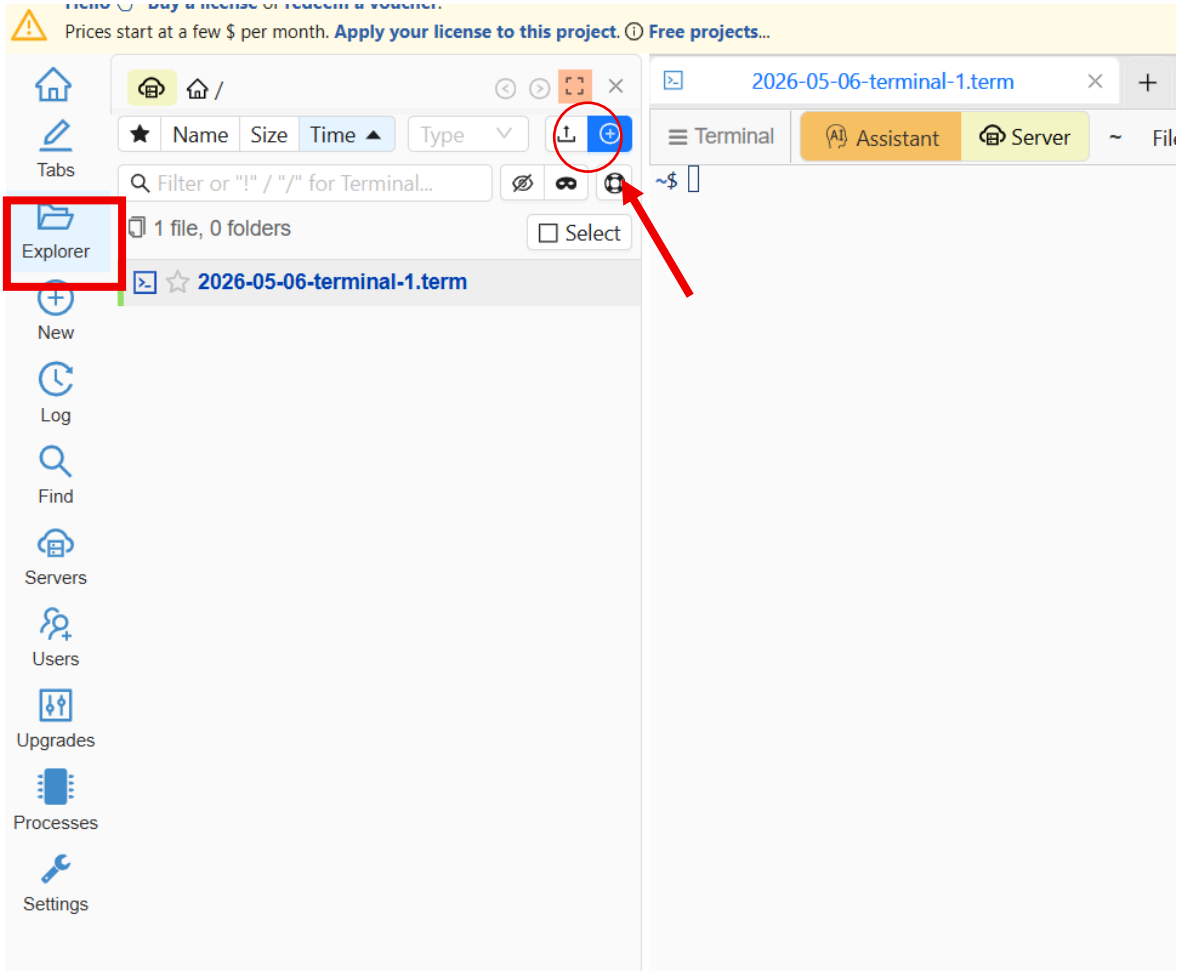
Üye olduktan sonra “Create Project” kısmından yeni proje oluşturmalısınız.



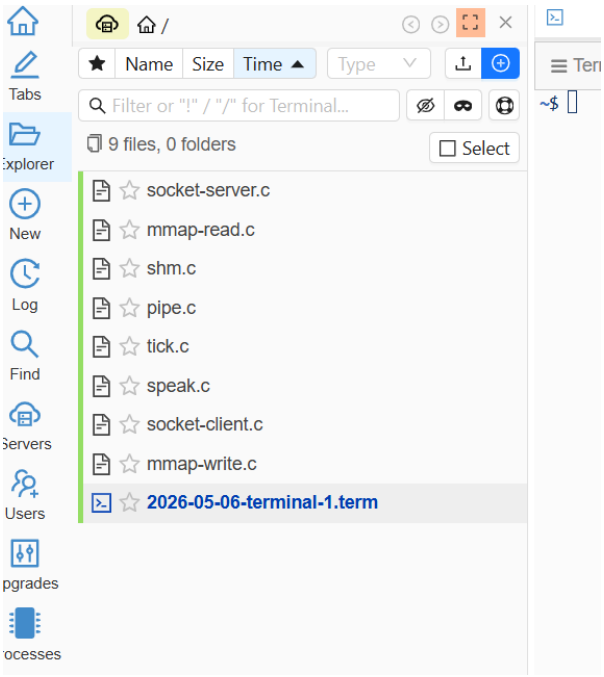
Projeyi oluşturduktan sonra sizi böyle bir ekran karşılayacak. Buradan “Linux Terminal” seçiyorsunuz.



“Explorer” kısmına girdikten sonra aşağıdaki ekran sizi karşılayacak. Buradan okla gösterdiğim dosya yükleme kısmına girip bölüm web sayfasına “https://hplab.ktu.edu.tr/bilgisayar/bilgisayar-sistemleri-laboratuvari “ deneyin olduğu yerden kodlarını indirip dosya yükleme kısmına yüklüyorsunuz.

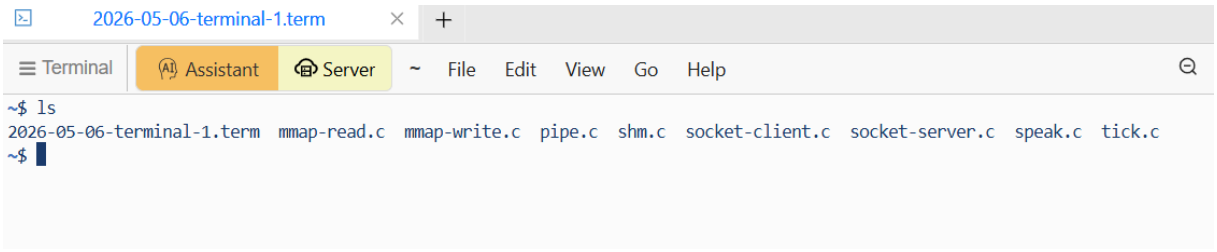


Yükleme işleminden sonra dosya kısmının aşağıdaki gibi olması lazım.



Bu noktadan sonra kodları çalıştırmaya başlayabilirsiniz.

Linux terminalinde" ls "komutu, bulunduğun dizindeki dosya ve klasörleri listelemek için kullanılır. Komutu yazarak dosyaları görüntüleyebilirsiniz.



Linux'ta "cat" komutu dosya içeriğini görüntülemek, birleştirmek ve çıktı vermek için kullanılır. ("cat dosya_adı.c")

```
~$ cat shm.c

/*************************************************************************
* Code listing from "Advanced Linux Programming," by CodeSourcery LLC  *
* Copyright (C) 2001 by New Riders Publishing                           *
* See COPYRIGHT for license information.                                *
*************************************************************************/

#include <stdio.h>
#include <sys/shm.h>
#include <sys/stat.h>

int main ()
{
  int segment_id;
  char* shared_memory;
  struct shmid_ds shmbuffer;
  int segment_size;
  const int shared_segment_size = 0x6400;

   unsigned short pid;

  /* Allocate a shared memory segment.  */
  segment_id = shmget (IPC_PRIVATE, shared_segment_size,
                       IPC_CREAT | IPC_EXCL | S_IRUSR | S_IWUSR);

  /* Attach the shared memory segment.  */
  shared_memory = (char*) shmat (segment_id, 0, 0);
  printf ("shared memory attached at address %p\n", shared_memory);
  /* Determine the segment's size.  */
  shmctl (segment_id, IPC_STAT, &shmbuffer);
  segment_size = shmbuffer.shm_segsz;
  pid  = shmbuffer.shm_cpid;
  printf ("segment size: %d\n", segment_size);
  printf("pid of creator: %d\n", pid);
  /* Write a string to the shared memory segment.  */
  sprintf (shared_memory, "Hello World");
```

Pico komutu , Linux sistemlerinde kullanılan basit bir metin editörüdür. Terminal üzerinden dosya oluşturup düzenlemeyi sağlar.

(pico shm.c)

Düzenleme yaptıktan sonra (ctrl + o → enter → ctrl+x) yapıp düzenleme kısmından çıkabilirsiniz.

İLK UYGULAMA PAYLAŞIMLI BELLEK (SHARED MEMORY) UYGULAMASI

Kodu shm.c

Kodları çalıştırmak için “gcc program.c -o program” şeklinde çalıştırmanız lazım.

- program.c → kaynak kod
- -o program → çıktı dosyasının adı (kısaltma gibi düşünebilirsiniz)
- ./(kısaltma adı)’nı yazıp derliyorsunuz.

```
~$ gcc shm.c -o sh
~$ ./sh
shared memory attached at address 0x7fe344532000
segment size: 25600
pid of creator: 493
shared memory reattached at address 0x5000000
Hello World
~$
```

Paylaşımlı bellek sanal bellek olduğundan bilgileri kaybediyoruz. Fiziki bir dosyadan bahsetmiyoruz burada.

İKİNCİ UYGULAMA HARİTALANMIŞ BELLEK UYGULAMASI (SHARED MEMORY)

Kodları Mmap-write.c ve mmap-read.c

Haritalanmış bellekte ise fiziki bir dosya oluşturacağız. Mmap-write ile dosyaya belleğe (-100 ile 100 arasında) rastgele sayı üretiyor. Mmap-read ile dosyaya erişme işlemi yapıyoruz. O yüzden iki kod birlikte kullanılıyor.

Kodda bulununan Sscanf'te bellekten okuma demek. -100 ile 100 arasında aşağıdaki gibi -96 üretti, bunu okuyor. Sonra ekrana yazdırıyor. Sprintf ile de belleğin içerisindeki sayıyı 2 ile çarpıyor. Yani belleğe -192 ekrana -96 yazıyor.

Burada fiziki dosya oluşturmanız gerekecek. Örnekteki gibi ./kk dosya diyip fiziki bir dosya oluşturuyoruz.

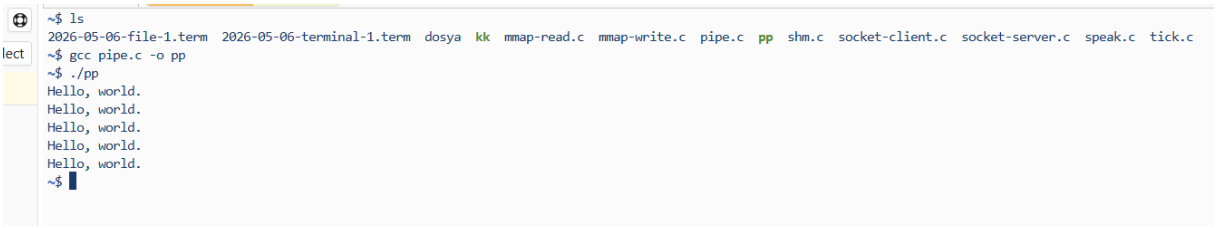
```
~$ ls
2026-05-06-terminal-1.term  mmap-read.c  mmap-write.c  pipe.c  shm.c  socket-client.c  socket-server.c  speak.c  tick.c
~$ gcc mmap-write.c -o kk
~$ ./kk dosya
~$ ls
2026-05-06-terminal-1.term  dosya  kk  mmap-read.c  mmap-write.c  pipe.c  shm.c  socket-client.c  socket-server.c  speak.c  tick.c
~$ cat dosya
-96
```

```
~$ gcc mmap-read.c -o pp
~$ ./pp dosya
value: -96
~$ cat dosya
-192
 ~$
```

ÜÇÜNCÜ UYGULAMA PİPE UYGULAMASI

Kodu Pipe.c

Kodda kaç sn aralıklarla kaç tane Hello world yazacağını buluyorsunuz. Kodun içeriğini değiştirip düzenleyebilirsiniz. Aşağıdaki kodda 3 sn aralıklarla 5 adet hello world yazıyor.

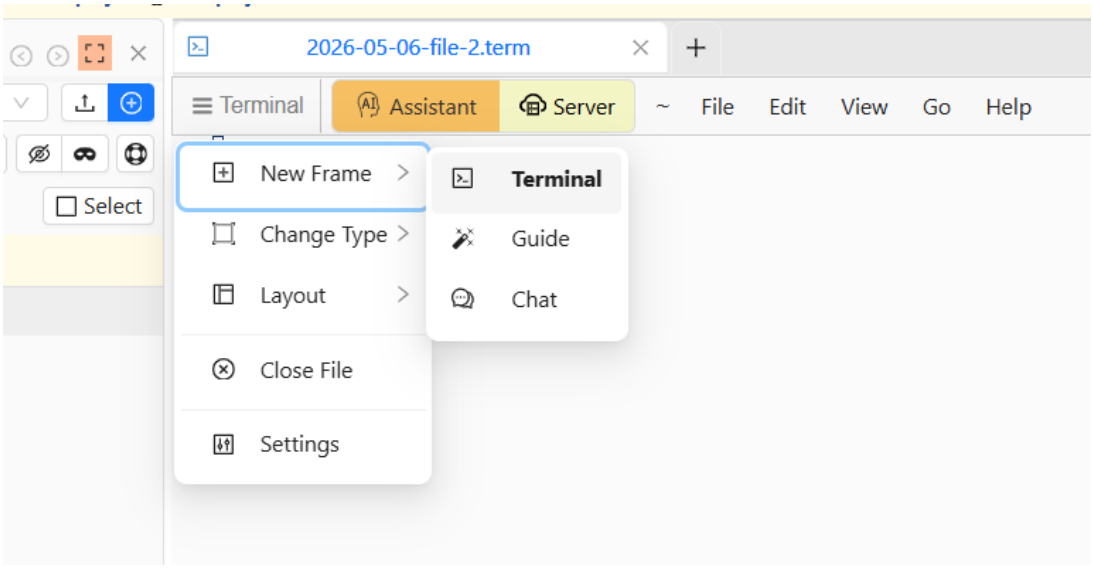


DÖRDÜNCÜ UYGULAMA FIFO UYGULAMASI

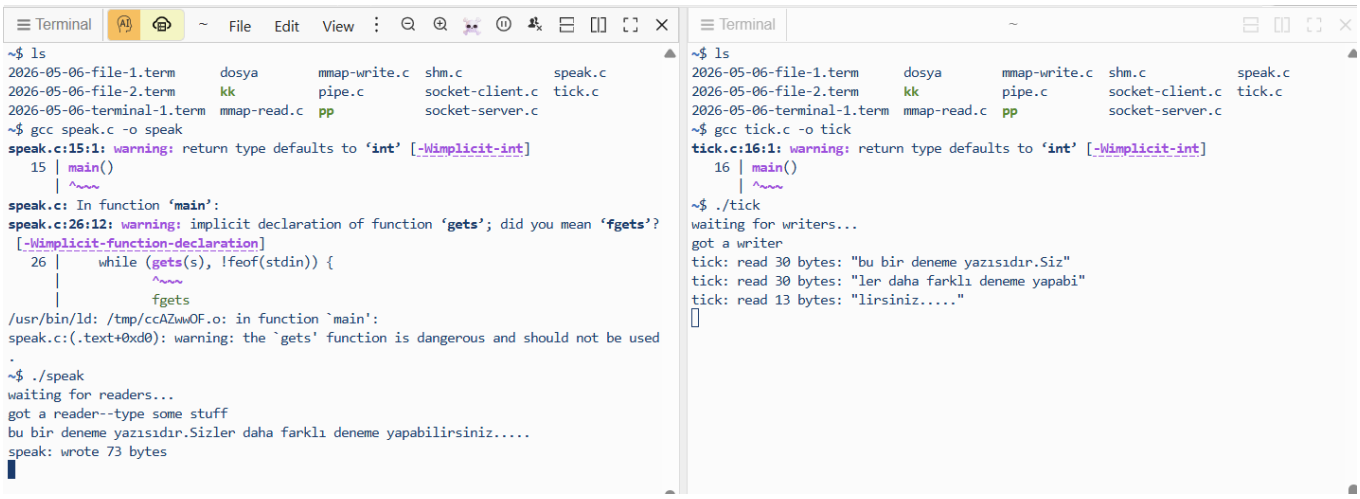
Kodları speak.c ve tick.c

Kodlar iki farklı süreçler arası iletişim yöntemini gösteriyor. Yazdığın yazıyı byte byte bölme işlemi yapıyor.

Burada iki farklı terminalde sırasıyla işlem yapıyoruz. Yani speak.c ve tick.c ayrı ayrı terminalde çalışacak. Aşağıdaki gibi yeni bir terminal açıp ekranı 2 ye bölüyorsunuz.



Ekran bu şekilde gözükecek.



Aşağıda numaralandırdığım sıraya göre kodları yazabilirsiniz.

(İLK TERMİNAL)

```
~$ ls
2026-05-06-file-1.term •    dosya        mmap-write.c  shm.c            speak.c
2026-05-06-file-2.term      kk           pipe.c        socket-client.c  tick.c
2026-05-06-terminal-1.term  mmap-read.c  pp            socket-server.c
~$ gcc speak.c -o speak
speak.c:15:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   15 | main()
      | ^~~~
speak.c: In function ‘main’:
speak.c:26:12: warning: implicit declaration of function ‘gets’; did you mean ‘fgets’?
 [-Wimplicit-function-declaration]
   26 |     while (gets(s), !feof(stdin)) {
      |            ^~~~
      |            fgets
/usr/bin/ld: /tmp/ccAZwwOF.o: in function `main':
speak.c:(.text+0xd0): warning: the `gets' function is dangerous and should not be used
.
~$ ./speak
waiting for readers...
got a reader--type some stuff
bu bir deneme yazısıdır.Sizler daha farklı deneme yapabilirsiniz.....
speak: wrote 73 bytes
```

(İKİNCİ TERMİNAL)

```
~$ ls
2026-05-06-file-1.term      dosya        mmap-write.c  shm.c            speak.c
2026-05-06-file-2.term      kk           pipe.c        socket-client.c  tick.c
2026-05-06-terminal-1.term  mmap-read.c  pp            socket-server.c
~$ gcc tick.c -o tick
tick.c:16:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   16 | main()
      | ^~~~
~$ ./tick
waiting for writers...
got a writer
tick: read 30 bytes: "bu bir deneme yazısıdır.Siz"
tick: read 30 bytes: "ler daha farklı deneme yapabi"
tick: read 13 bytes: "lirsiniz....."
```

BEŞİNCİ UYGULAMA SOKET UYGULAMASI

Bu uygulamayı sizlerin yapmasını istiyorum. Raporda nasıl yapıldığını detaylı şekilde açıklayınız.