FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ

İŞLETİM SİSTEMLERİ

Dönem Projesi

ProcX - Gelişmiş Süreç Yönetim Sistemi

HAZIRLAYAN

Beyza YILMAZ

2321021036

DANIŞMAN

Dr. Öğr. Üyesi Samet KAYA

Fatih Sultan Mehmet Vakıf Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği Bölümü

Aralık, 2025

ProcX, Linux ortamında çalışan ve kullanıcıların terminal üzerinden yeni programlar başlatmasına, çalışan süreçleri listelemesine ve süreç sonlandırmasına olanak sağlayan nohup-benzeri bir süreç yönetim sistemidir. Proje; POSIX paylaşımlı bellek (shm_open/mmap) ile süreç bilgilerini ortak alanda tutar, semafor (sem_open) ile kritik bölgeleri korur ve System V mesaj kuyruğu (msgget/msgrcv/msgsnd) ile farklı terminal oturumları arasında olay bildirimleri iletir. Ek olarak, monitor thread arka planda detached süreçlerin yaşam durumunu izler ve tabloyu güncel tutar.

Sistem Genel Mimarisi

ProcX tek kaynak dosyalı (procx.c) bir CLI uygulamasıdır. Çalışma anında üç ana yürütme bileşeni vardır: (i) Main thread (menü ve kullanıcı etkileşimi), (ii) Monitor thread (detached süreçleri periyodik izleme), (iii) IPC Listener thread (mesaj kuyruğundan olay dinleme). Tüm instance'lar aynı named shared memory alanını map ederek ortak tabloya bakar.

Bileşenler

- Main Thread: Menü gösterimi, kullanıcıdan komut alma, süreç başlatma/listeleme/sonlandırma çağrıları.
- Monitor Thread: Sadece bu instance'ın başlattığı DETACHED süreçlerde waitpid(WNOHANG) ile bitişi yakalar, tabloyu günceller ve terminate bildirimi yollar.
- IPC Listener Thread: Mesaj kuyruğundan start/terminate olaylarını dinler ve kullanıcıya bildirim basar.

Veri Paylaşımı

Ortak süreç tablosu SharedData yapısı ile tutulur. Her ProcessInfo kaydı PID, owner PID, komut, mod, durum, başlangıç zamanı ve aktiflik bayrağı içerir. En fazla 50 kayıt desteklenir.

IPC Tasarımı

1. Shared Memory (POSIX)

Paylaşımlı bellek adı SHM_NAME ile tanımlanır (/procx_shm). init_ipc() fonksiyonu önce shm_open(O_CREAT|O_EXCL) ile kaynağı yaratmayı dener. Başarılı olursa creator instance olur ve ftruncate ile boyutu sizeof(SharedData) olacak şekilde ayarlar. Kaynak zaten varsa (errno=EEXIST) O_RDWR ile bağlanılır. Ardından mmap ile tüm instance'lar aynı SharedData alanını map eder.

2. Semaphore (POSIX Named Semaphore)

Ortak tabloya eşzamanlı erişim sem_open ile oluşturulan named semaphore üzerinden yapılır (/procx_sem). Kritik bölgelerde sem_wait/sem_post kullanılarak veri yarışları (race condition) engellenir. Örneğin add_process_record(), process_listele() ve process_sonlandir() sırasında tablo erişimleri kilit altındadır.

3. Message Queue (System V)

Instance'lar arası olay bildirimi System V mesaj kuyruğu ile yapılır. ftok için /tmp/procx_msgfile dosyası oluşturulur ve msgget ile kuyruk açılır/oluşturulur. send_notification() fonksiyonu start (1) ve terminate (2) olaylarını kuyruğa yazar. ipc_listener_thread() msgrcv ile bloklayarak bekler ve olayları ekrana basar.

Not: Kodda program her başladığında kuyrukta kalan eski mesajlar IPC_NOWAIT ile temizlenmektedir. Bu, önceki oturumdan kalan mesajların yeni oturumu kirletmesini önler.

Yazılım Bileşenleri ve Fonksiyon Açıklamaları

ProcX sistemi, sorumlulukların ayrılması (separation of concerns) prensibine uygun olarak modüler fonksiyonlar halinde geliştirilmiştir. Sistemdeki fonksiyonlar ve görevleri aşağıdaki kategorilere ayrılmıştır:

1. IPC ve Kaynak Yönetimi Fonksiyonları

Bu fonksiyonlar, Shared Memory, Semaphore ve Message Queue gibi sistem kaynaklarının başlatılmasından ve temizlenmesinden sorumludur.

- **`init_ipc()`:** Programın başlangıcında çalışır. Paylaşımlı bellek, semafor ve mesaj kuyruğunu oluşturur. Eğer kaynaklar daha önce oluşturulmuşsa (başka bir ProcX instance'ı tarafından), mevcut kaynaklara bağlanır (attach). "Creator" (ilk başlatan) ve "Client" ayrımını yönetir.
- **`cleanup_ipc()`:** Program sonlanırken çağrılır. Paylaşımlı bellek bağlantısını keser (munmap), semaforları kapatır ve sistem kaynaklarını serbest bırakır.
- **`siginit_handler(int sig)`:** `SIGINT` (Ctrl+C) sinyali yakalandığında tetiklenir. Programın ani kapanması durumunda kaynakların temizlenmesini (`cleanup_ipc`) ve çocuk süreçlerin güvenle sonlandırılmasını sağlar.

2. Süreç Yönetimi (Process Management) Fonksiyonları

Süreçlerin oluşturulması, sonlandırılması ve takip edilmesi işlemlerini yürüten çekirdek fonksiyonlardır.

- **`process_baslat(const char *command, ProcessMode mode)`:** Sistemin kalbidir. `fork()` çağrısı ile yeni bir süreç oluşturur.
  - **Detached Mod:** `setsid()` ile süreci terminalden ayırır.
  - **Attached Mod:** `waitpid()` ile sürecin bitmesini bekler.
  - Son olarak `execvp()` ile kullanıcı komutunu çalıştırır.
- **`add_process_record(...)`:** Yeni oluşturulan sürecin bilgilerini (PID, komut, mod, vb.) paylaşımlı bellekteki (Shared Memory) tabloya yazar. Bu işlem sırasında veri bütünlüğü için semafor kilitleri kullanılır.
- **`process_sonlandir()`:** Kullanıcıdan alınan PID bilgisine göre, hedef sürece `SIGTERM` sinyali göndererek süreci sonlandırır.
- **`kill_child_process()`:** Program kapanmadan önce, bu oturuma bağlı (attached) çalışan alt süreçlerin açık kalmaması için hepsini sonlandırır.

3. Thread Fonksiyonları

Arka planda asenkron olarak çalışan görevleri yönetir.

- **`monitor_thread(void *arg)`:** "Garbage Collector" ve "Health Check" görevi görür. Periyodik olarak çalışır; sonlanan süreçleri (`waitpid` ile `WNOHANG` modunda) tespit eder, "Zombie Process" oluşumunu engeller ve paylaşımlı bellek tablosunu günceller.
- **`ipc_listener_thread(void *args)`:** Mesaj kuyruğunu (`msgrcv`) sürekli dinler. Başka bir süreçten gelen "Başladı" veya "Bitti" bildirimlerini yakalar ve kullanıcıya konsol üzerinden bilgi verir.

4. Yardımcı ve Arayüz Fonksiyonları

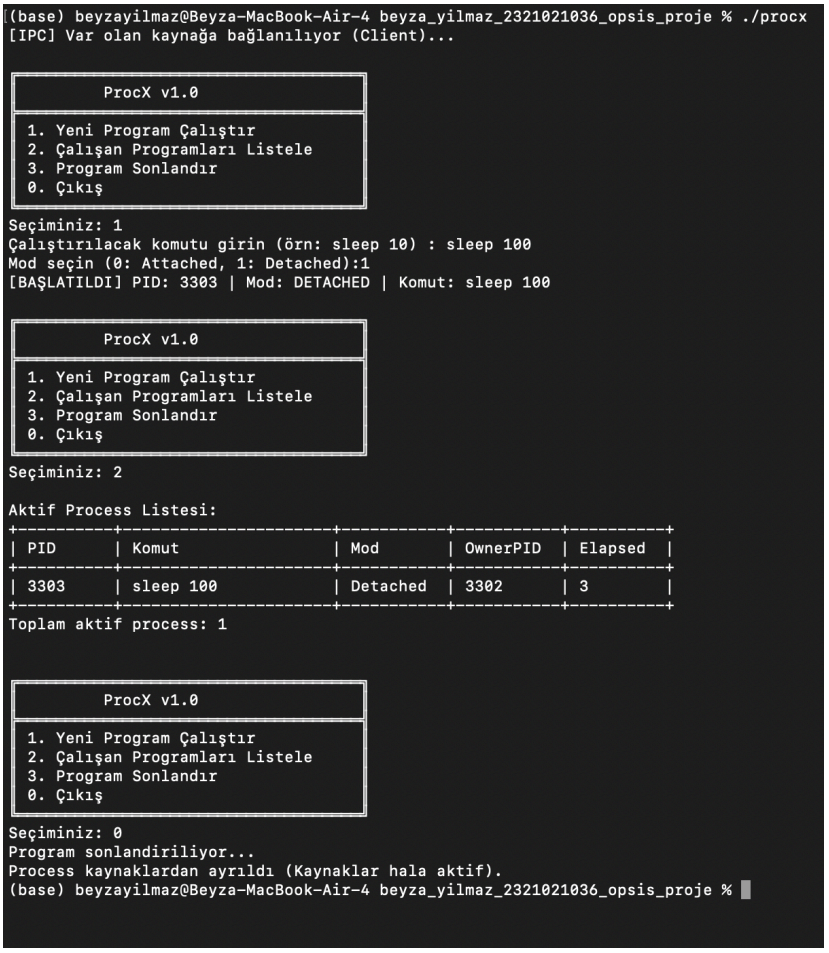
Veri işleme ve kullanıcı etkileşimi fonksiyonlarıdır.

- **`parse_command(...)`:** Kullanıcının girdiği komut satırını (string) parçalayarak `execvp` fonksiyonunun anlayacağı argüman dizisine (`argv`) dönüştürür. Ayrıca komutun sonunda `&` işareti olup olmadığını kontrol ederek "Detached" modunu belirler.

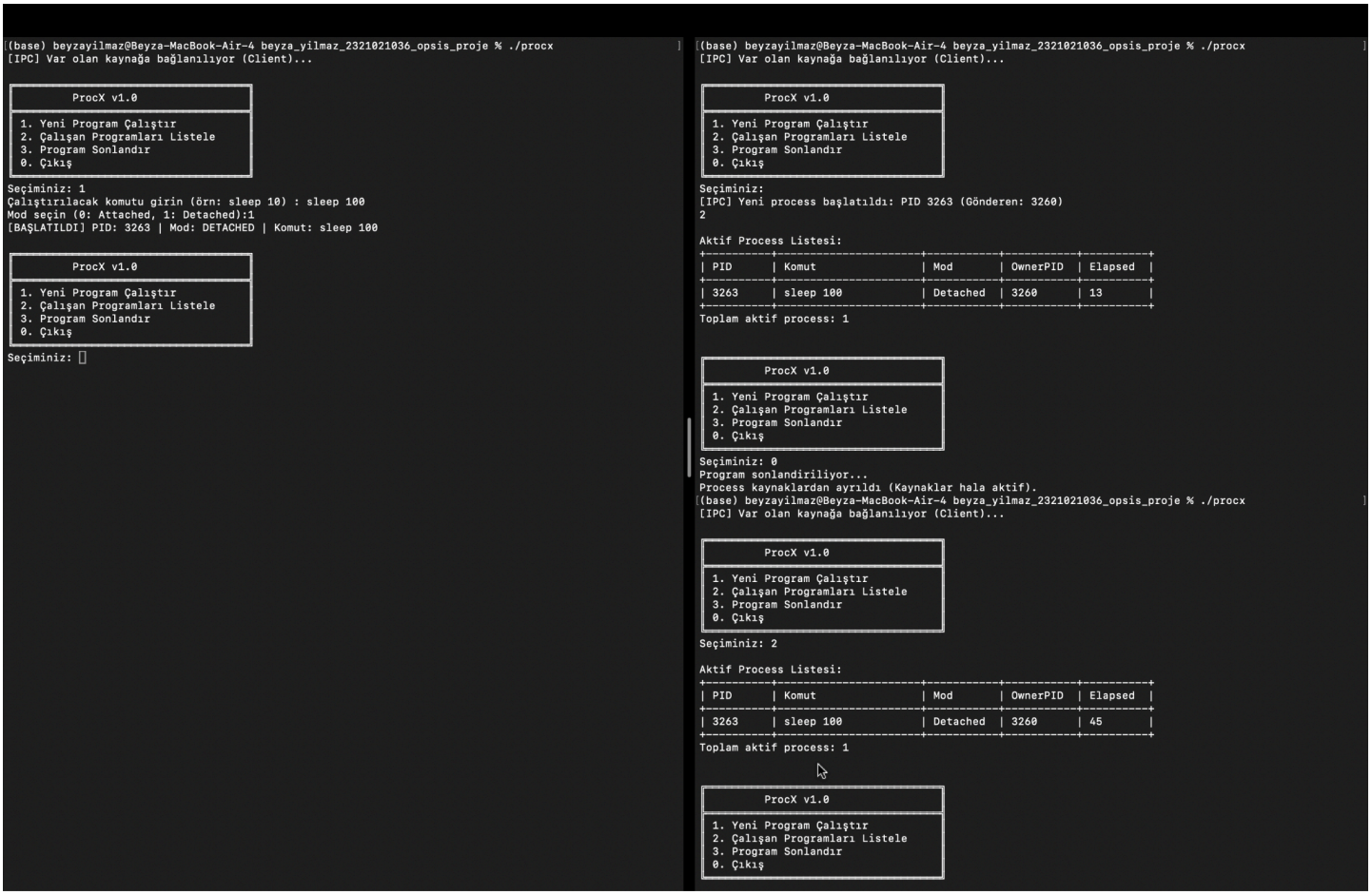
- **`process_listele()`:** Paylaşımlı bellekteki süreç tablosunu okur. `kill(pid, 0)` yöntemiyle süreçlerin hala hayatta olup olmadığını kontrol eder ve formatlı bir tablo şeklinde ekrana basar.
- **`display_menu()`:** Kullanıcıya ana menü seçeneklerini sunar ve güvenli veri girişi (input validation) sağlar.

Test Senaryoları Çıktıları

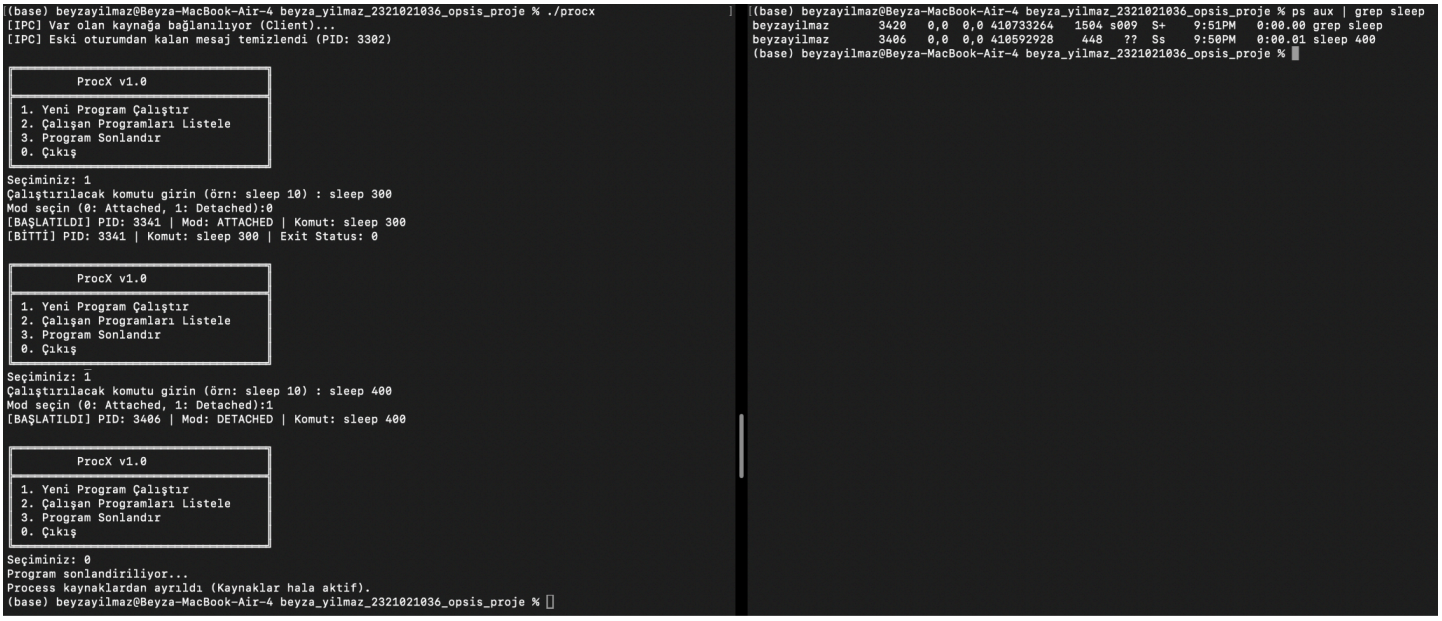
Test 1: Tek Instance - Process Başlatma ve Listeleme



Test 2: Çoklu Instance - IPC Testi



Test 3: Attached vs Detached Modu



Test 4: Process Sonlandırma

```
(base) beyzayilmaz@Beyza-MacBook-Air-4 beyza_yilmaz_2321021036_opsis_proje % ./procx
[IPC] Var olan kaynağa bağlanılıyor (Client)...

        ProcX v1.0
 1. Yeni Program Çalıştır
 2. Çalışan Programları Listele
 3. Program Sonlandır
 0. Çıkış
Seçiminiz: 1
Çalıştırılacak komutu girin (örn: sleep 10) : sleep 500
Mod seçin (0: Attached, 1: Detached):1
[BAŞLATILDI] PID: 3450 | Mod: DETACHED | Komut: sleep 500

        ProcX v1.0
 1. Yeni Program Çalıştır
 2. Çalışan Programları Listele
 3. Program Sonlandır
 0. Çıkış
Seçiminiz: 3
Sonlandirilacak process PID: 3460
Bu PID'e ait aktif process bulunamadi.

        ProcX v1.0
 1. Yeni Program Çalıştır
 2. Çalışan Programları Listele
 3. Program Sonlandır
 0. Çıkış
Seçiminiz: 3
Sonlandirilacak process PID: 3450
[INFO] Process 3450'e SIGTERM sinyali gönderildi.
# Monitor thread sonlanmayı tespit edip raporlayacak...

        ProcX v1.0
 1. Yeni Program Çalıştır
 2. Çalışan Programları Listele
 3. Program Sonlandır
 0. Çıkış
Seçiminiz:
[MONITOR] Process 3450 sonlandı. Exit Code: 0
^C
[Sinyal] Ctrl+C algılandı. Kaynaklar temizleniyor...
Process kaynaklardan ayrıldı (Kaynaklar hala aktif).
```

Test 5: Monitor Thread Testi

```
(base) beyzayilmaz@Beyza-MacBook-Air-4 beyza_yilmaz_2321021036_opsis_proje % ./procx
[IPC] Var olan kaynağa bağlanılıyor (Client)...
[IPC] Eski oturumdan kalan mesaj temizlendi (PID: 3446)
[IPC] Eski oturumdan kalan mesaj temizlendi (PID: 3446)

        ProcX v1.0
 1. Yeni Program Çalıştır
 2. Çalışan Programları Listele
 3. Program Sonlandır
 0. Çıkış
Seçiminiz: 1
Çalıştırılacak komutu girin (örn: sleep 10) : sleep 5
Mod seçin (0: Attached, 1: Detached):1
[BAŞLATILDI] PID: 3484 | Mod: DETACHED | Komut: sleep 5

        ProcX v1.0
 1. Yeni Program Çalıştır
 2. Çalışan Programları Listele
 3. Program Sonlandır
 0. Çıkış
Seçiminiz:
[MONITOR] Process 3484 sonlandı. Exit Code: 0
```