İŞLETİM SİSTEMLERİ DÖNEM PROJESİ

Süreç Yönetim Sistemi – ProcX

Ders: İşletim Sistemleri
Öğrenci: Emirhan Başar
Öğrenci No: *2221221012*
Öğretim Üyesi: Dr. Öğr. Üyesi Samet Kaya

1. Projenin Amacı

Bu projenin amacı, işletim sistemlerinde **süreç yönetimi**, **çoklu thread kullanımı** ve **IPC (Inter-Process Communication)** mekanizmalarının gerçek bir uygulama üzerinden öğrenilmesini sağlamaktır.

Bu kapsamda geliştirilen **ProcX** uygulaması;

- Yeni süreç başlatabilen
- Çalışan süreçleri listeleyebilen
- Süreç sonlandırabilen
- Birden fazla terminal (instance) arasında **ortak süreç tablosu** paylaşabilen bir **nohup benzeri süreç yönetim sistemi** sunmaktadır.

Program aynı anda birden fazla terminalde çalıştırılabilmekte ve tüm instance'lar **shared memory** üzerinden senkron şekilde işlem yapabilmektedir. Bu sayede IPC mekanizmalarının gerçek zamanlı etkisi gözlemlenmektedir.

2. Sistem Mimarisi

ProcX uygulaması çalıştığında **üç ana thread** oluşturmaktadır:

2.1 Main Thread

- Kullanıcıdan menü seçimlerini alır.
- Süreç başlatma, listeleme ve sonlandırma işlemlerini yönetir.
- Fork + exec mekanizmasını tetikler.

2.2 Monitor Thread

- Belirli aralıklarla (`waitpid(WNOHANG)`) süreçleri kontrol eder.
- Sonlanan süreçleri otomatik olarak tespit eder.
- Shared memory üzerindeki süreç tablosunu günceller.
- "[MONITOR] Process XXXX sonlandı" mesajını üretir.

2.3 IPC Listener Thread

- POSIX Message Queue üzerinden gelen mesajları dinler.
- Diğer terminal instance'larında başlatılan veya sonlanan süreçler hakkında bilgi verir.
- Çoklu terminal senaryosunda senkron çalışmayı sağlar.

3. Kullanılan IPC Yapıları

Projede **POSIX tabanlı üç IPC mekanizması** birlikte kullanılmıştır:

3.1 Shared Memory

- Tüm terminal instance'ları tarafından ortak kullanılan süreç tablosu burada tutulur.
- PID, komut, mod, owner ve süre bilgileri saklanır.

3.2 Semaphore

- Shared memory erişimi semaphore ile korunur.
- Yarış koşulları (race condition) engellenmiştir.

3.3 Message Queue

- Instance'lar arası olay bildirimleri için kullanılır.
- START, TERMINATED ve KILLED mesajları gönderilir.

4. Süreç Modları

Yeni süreç başlatılırken kullanıcıdan mod seçimi istenir:

4.1 Attached Mode (0)

- Süreç ProcX instance'ına bağlıdır.
- ProcX kapatıldığında süreç otomatik olarak sonlanır.

4.2 Detached Mode (1)

- Süreç arka planda bağımsız çalışır.
- ProcX kapatılsa bile süreç çalışmaya devam eder.
- `setsid()` kullanılarak yeni bir session oluşturulur.

Bu fark test senaryolarında açıkça gözlemlenmiştir.

5. Programın Çalıştırılması

5.1 Derleme

make

5.2 Çalıştırma

./procx

5.3 Menü Yapısı

ProcX v1.0

1. Yeni Program Çalıştır

2. Çalışan Programları Listele

3. Program Sonlandır

0. Çıkış

6. Test Senaryoları ve Sonuçlar

Test 1 – Tek Terminalde Süreç Başlatma

Amaç:
Yeni bir sürecin başlatılması ve listede görünmesi.

İşlem:

- `sleep 100` detached modda başlatıldı.
- Listeleme menüsü seçildi.

Beklenen:

- PID, komut, mod, owner ve süre bilgileri doğru şekilde listelendi.

Test 2 – Çoklu Terminalde IPC Bildirimi

Amaç:
Bir terminalde başlatılan sürecin diğer terminalde bildirilmesi.

İşlem:

- Terminal A'da `sleep 200` başlatıldı.
- Terminal B'de IPC mesajı gözlemlendi.

[IPC] Yeni process başlatıldı: PID XXXX

Test 3 – Attached vs Detached Davranışı

Test 3A – Detached Mode

- Süreç detached modda başlatıldı.

- ProcX kapatıldı.

Beklenen:
Süreç çalışmaya devam etti.

Test 3B – Attached Mode

- Süreç attached modda başlatıldı.
- ProcX kapatıldı.

Beklenen:
Süreç otomatik olarak sonlandı.

Test 4 – Süreç Sonlandırma

Amaç:
PID girilerek sürecin sonlandırılması.

İşlem:

- Menüden “Program Sonlandır” seçildi.
- PID girildi.

Beklenen:

- SIGTERM gönderildi mesajı.
- Süreç sonlandı.

Test 5 – Monitor Thread Kontrolü

Amaç:
Kısa süreli sürecin otomatik tespiti.

İşlem:

- `sleep 7` başlatıldı.
- Sürecin bitmesi beklendi.

Beklenen:

[MONITOR] Process XXXX sonlandı

7. Sonuç ve Değerlendirme

Bu proje ile işletim sistemlerinde;

- Süreç yönetimi
- Fork ve exec kullanımı
- Çoklu thread yapısı
- IPC mekanizmaları

uygulamalı olarak öğrenilmiştir.

ProcX uygulaması aşağıdaki gereksinimleri başarıyla karşılamaktadır:

- Shared memory üzerinden ortak süreç tablosu
- Semaphore ile güvenli erişim
- Message queue ile instance'lar arası iletişim
- Çoklu terminal desteği
- Attached / Detached süreç modeli
- Süreç başlatma, listeleme, sonlandırma ve otomatik izleme

Testler sonucunda sistemin **kararlı ve doğru şekilde çalıştığı** gözlemlenmiştir