

## TEST 1-TEK INSTANCE PROCESS BAŞLATMA ve LİSTELEME

Bu testte ProcX tek bir instance olarak çalıştırılmıştır. sleep komutu DETACHED modda fork-exec mekanizması ile başlatılmış, process bilgileri shared memory üzerinde kaydedilmiş ve semaphore ile senkronize edilerek doğru şekilde listelenmiştir.

```
==== ProcX v1.0 ====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 1
Komut girin: sleep 100
Mod (0: Attached, 1: Detached): 1
[SUCCESS] Process başlatıldı: PID 6038 | Komut: sleep | Mod: DETACHED

==== ProcX v1.0 ====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 2

--- ÇALIŞAN PROGRAMLAR ---
PID: 6038 | Komut: sleep | Mod: DETACHED | Owner: 6000 | Süre: 3 sn
Toplam: 1 process

==== ProcX v1.0 ====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 0

[EXIT] Çıkış isteği alındı. Kapatma işlemleri başlıyor...
[EXIT] ATTACHED process temizliği başlıyor...
[EXIT] ATTACHED temizliği bitti. Kapatılan: 0
```

## TEST 2-ÇOKLU INSTANCE- IPC TESTİ

Bu testte ProcX birden fazla instance olarak çalıştırılmıştır. Bir instance tarafından başlatılan process için message queue üzerinden START olayı gönderilmiş, diğer instance bu olayı IPC listener thread aracılığıyla almış ve shared memory üzerindeki güncel process listesini doğru şekilde göstermiştir.

### Terminal 1

```
==== ProcX v1.0 ====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 1
Komut girin: sleep 200
Mod (0: Attached, 1: Detached): 1
[SUCCESS] Process başlatıldı: PID 6748 | Komut: sleep | Mod: DETACHED

==== ProcX v1.0 ====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 2

--- ÇALIŞAN PROGRAMLAR ---
PID: 6748 | Komut: sleep | Mod: DETACHED | Owner: 6683 | Süre: 27 sn
Toplam: 1 process
```

## Terminal 2

```
===== ProcX v1.0 =====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz:
[IPC] Olay: PID 6748 -> START
2

--- ÇALIŞAN PROGRAMLAR ---
PID: 6748 | Komut: sleep | Mod: DETACHED | Owner: 6683 | Süre: 32 sn
Toplam: 1 process
```

## TEST 3- ATTACHED Vs DETACHED

Bu test senaryosunda ProcX uygulamasının attached ve detached modlarda başlatılan süreçleri farklı şekilde yönetme ve Ctrl+C (SIGINT) sinyali karşısındaki davranışı test edilmiştir. İlk olarak ProcX üzerinden detached modda uzun süreli bir süreç (sleep 200) başlatılmıştır. Bu süreç terminalden bağımsız çalışacak şekilde setsid() kullanılarak başlatılmış ve ProcX kapatılsa dahi çalışmaya devam etmesi amaçlanmıştır. Ardından attached modda kısa süreli bir süreç (sleep 30) başlatılmıştır. Bu süreç çalışırken kullanıcı tarafından Ctrl+C (SIGINT) sinyali gönderilmiştir. SIGINT sinyali ProcX tarafından yakalanmış, program güvenli çıkış (graceful shutdown) moduna geçmiştir. Bu aşamada attached modda çalışan süreç, ProcX tarafından SIGTERM sinyali gönderilerek sonlandırılmış ve waitpid() ile sistemden tamamen temizlenmiştir.

```
===== ProcX v1.0 =====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 1
Komut girin: sleep 200
Mod (0: Attached, 1: Detached): 1
[SUCCESS] Process başlatıldı: PID 11525 | Komut: sleep | Mod: DETACHED

===== ProcX v1.0 =====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 1
Komut girin: sleep 30
Mod (0: Attached, 1: Detached): 0
[SUCCESS] Process başlatıldı: PID 11555 | Komut: sleep | Mod: ATTACHED
^C
[HANDLER] SIGINT alındı. Güvenli çıkış başlıyor...
[INFO] Attached process bitti: PID 11555
[EXIT] ATTACHED process temizliği başlıyor...
[EXIT] ATTACHED temizliği bitti. Kapatılan: 0
[CLEANUP] Diğer ProcX instance'leri acik. IPC kaynaklari korunuyor...
```

```
===== ProcX v1.0 =====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz:
[IPC] Olay: PID 11555 -> TERMINATE
2

--- ÇALIŞAN PROGRAMLAR ---
PID: 11525 | Komut: sleep | Mod: DETACHED | Owner: 11419 | Süre: 13 sn
Toplam: 1 process
```

```
● elif@elif:~/Desktop/enso$ ps aux | grep sleep
elif      11525  0.0  0.0   5256  1792 ?        Ss   14:31   0:00 sleep 200
elif      11661  0.0  0.0   5256  1792 ?        S    14:31   0:00 sleep 180
elif      11700  0.0  0.0   6140  2048 pts/1    S+   14:32   0:00 grep --color=auto sleep
```

## TEST 4-PROCESS SONLANDIRMA

Bu testte ProcX üzerinden DETACHED modda sleep 100 çalıştırılmıştır. Ardından “Çalışan Programları Listele” menüsü ile süreç listede doğrulanmış (PID: 12373) ve “Program Sonlandır” seçeneğiyle bu PID’ye SIGTERM sinyali gönderilmiştir. ProcX, sinyal gönderiminin başarılı olduğunu raporlamış ve sonrasında sürecin gerçekten sonlandığını waitpid kontrolü ile doğrulamıştır .

```
===== ProcX v1.0 =====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 1
Komut girin: sleep 100
Mod (0: Attached, 1: Detached): 1
[SUCCESS] Process başlatıldı: PID 13794 | Komut: sleep | Mod: DETACHED

===== ProcX v1.0 =====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 2

--- ÇALIŞAN PROGRAMLAR ---
PID: 13794 | Komut: sleep | Mod: DETACHED | Owner: 13758 | Süre: 8 sn
Toplam: 1 process

===== ProcX v1.0 =====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 3
Sonlandırılacak PID: 13794
[TERM] PID 13794 için sonlandırma isteği gönderiliyor...
[INFO] Process 13794'e SIGTERM gönderildi.
[INFO] Process 13794 sonlandı (waitpid).

===== ProcX v1.0 =====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 2

--- ÇALIŞAN PROGRAMLAR ---
Toplam: 0 process
```

## TEST 5 -Monitor Thread Test

Bu testte ProcX üzerinde DETACHED modda kısa süreli bir process başlatılmıştır. Process çalışırken çalışan programlar listesinde doğru şekilde görüntülenmiştir. Süre dolduktan sonra process kullanıcı müdahalesi olmadan sonlanmış ve monitor thread tarafından otomatik olarak tespit edilmiştir. Monitor thread, sonlanan süreci shared memory üzerinden temizleyerek tabloyu güncellemiş ve “[MONITOR] Process ... sonlandı.” mesajını üretmiştir.

Ardından ATTACHED modda bir process çalıştırılmış, bu süreç ProcX tarafından waitpid ile doğrudan takip edilmiş ve tamamlandığında “[INFO] Attached process bitti” mesajı verilmiştir. Bu test sonucunda, monitor thread’in detached süreçleri otomatik olarak izleyip temizlediği, attached süreçlerin ise ana thread tarafından senkron biçimde yönetildiği doğrulanmıştır.

```
===== ProcX v1.0 =====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 1
Komut girin: sleep 5
Mod (0: Attached, 1: Detached): 1
[SUCCESS] Process başlatıldı: PID 13486 | Komut: sleep | Mod: DETACHED

===== ProcX v1.0 =====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 2

--- ÇALIŞAN PROGRAMLAR ---
PID: 13486 | Komut: sleep | Mod: DETACHED | Owner: 13420 | Süre: 1 sn
Toplam: 1 process

===== ProcX v1.0 =====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz:
[MONITOR] Process 13486 sonlandı.
```

```
===== ProcX v1.0 =====
1. Yeni Program Çalıştır
2. Çalışan Programları Listele
3. Program Sonlandır
0. Çıkış
Seçiminiz: 1
Komut girin: sleep 10
Mod (0: Attached, 1: Detached): 0
[SUCCESS] Process başlatıldı: PID 13622 | Komut: sleep | Mod: ATTACHED
[INFO] Attached process bitti: PID 13622
```