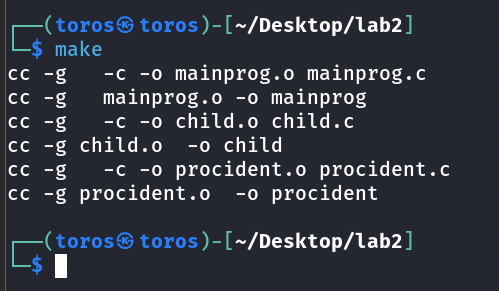
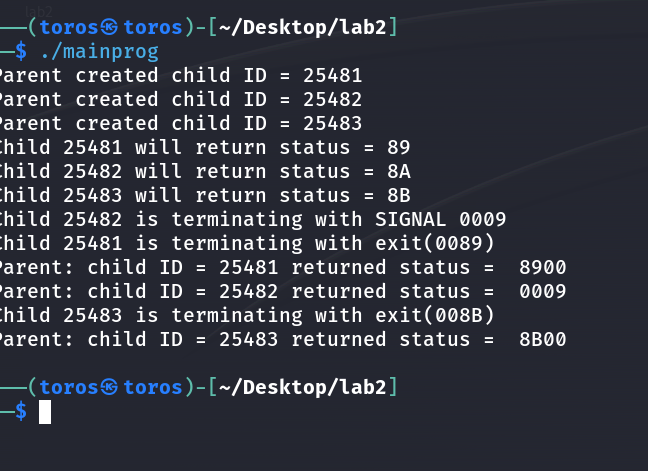
1.1)

1.2)



**mainprog**, 3 kez çocuk süreç oluşturan bir ebeveyn süreçtir. Bu çocuk süreçler, child adlı başka bir programa geçiş yapar ve her biri bir indeks değeri ile çağrılır. Bu indeks değeri, çocuk süreçlerin kaçıncı kez oluşturulduğunu temsil eder.

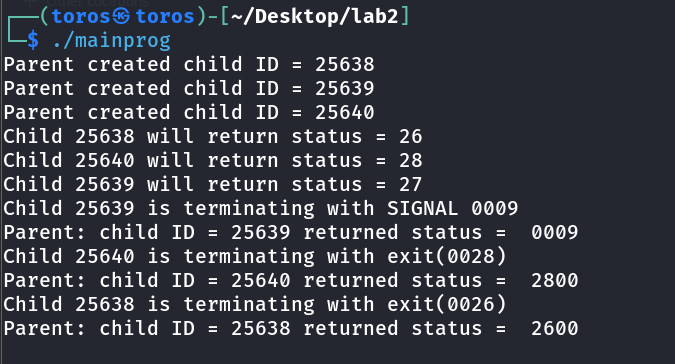
**fork()**: Bu sistem çağrısı, ebeveyn süreç tarafından çağrıldığında, yeni bir çocuk süreç oluşturur. Ebeveyn süreç, çocuk süreç oluşturulduktan sonra ve çocuk süreç, **execl()** veya benzeri bir çağrı ile başka bir programa geçtikten sonra eş zamanlı olarak devam eder.

**execl()**: Bu sistem çağrısı, belirtilen programı çalıştırmak için kullanılır. Mainprog içindeki çocuk süreçler, bu çağrıyı kullanarak **child** adlı başka bir programa geçerler. Bu durumda, child programına indeks değeri olarak bir argüman eklenir.

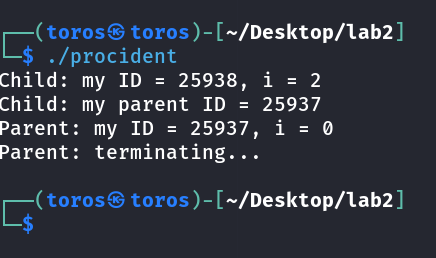
**wait()**: Bu sistem çağrısı, ebeveyn sürecin çocuk süreçlerinin tamamlanmasını beklemesini sağlar. Ebeveyn süreç, bu çağrı ile bir çocuk sürecin tamamlanmasını bekler ve çocuk sürecin çıkış durumuyla ilgili bilgileri alır. Bu durumda, wait**()** döngü içinde kullanılarak tüm çocuk süreçlerin tamamlanması beklenir.

Fork, ebeveyn süreç tarafından çağrıldığında yeni bir çocuk süreç oluşturulur. Ebeveyn süreç, çocuk süreç oluşturduktan sonra genellikle wait**()** çağrısı ile çocuk sürecin tamamlanmasını bekler. Bu şekilde, ebeveyn süreç ve çocuk süreçlerin eşzamanlı olarak çalışmasını sağlar.

1.3)



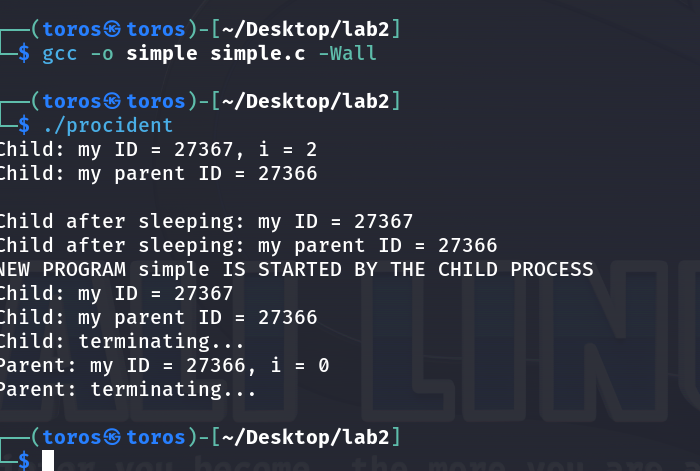
Ana sürecin çocuk süreçlerinin bitmesini beklemeden hemen sona erebileceğini gözlemleyebilirsiniz. Bu, wait() sistem çağrısını kaldırmak, yani ana sürecin çocuk süreçlerini beklemeden bağımsız olarak sonlanmasına neden olabilir.

2.2)

Programın çalışması sırasında, çocuk süreci (child) genellikle ebeveyn sürecinden (parent) önce tamamlanacaktır. Bunun nedeni, çocuk sürecin 10 saniyelik bir uyku (sleep(10)) süresine sahip olmasıdır. Ebeveyn süreç ise sadece 3 saniye uyuduktan sonra kendi işlemlerini tamamlayacaktır.

Çocuk sürecin farklı parent süreçlere sahip olmasının nedeni, execl("simple", "simple", 0) çağrısı nedeniyledir. Bu çağrı, çocuk süreci tarafından kullanıldığında, çocuk sürecin kendisini başka bir programa (simple) değiştirmesini sağlar. Bu nedenle, çocuk sürecin yeni bir programın süreç kimliğine sahip olması beklenir.

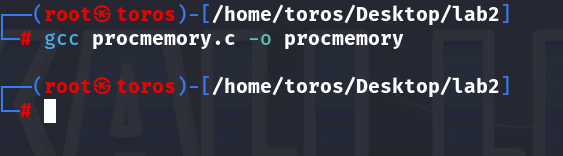
Çocuk sürecin programın sonlandığını görmemesi nedeniyle program sonlanmadan UNIX prompt görünür. Çocuk sürec, execl("simple", "simple", 0) çağrısı ile kendisini başka bir programa değiştirdiğinde, ebeveyn süreç bu değişikliği fark etmez ve kendi işlemlerini sürdürür. Bu nedenle, çocuk sürecin yeni programın sonlandığını görmemesi ve bu süreç tamamlanana kadar UNIX prompt hemen görünmez.

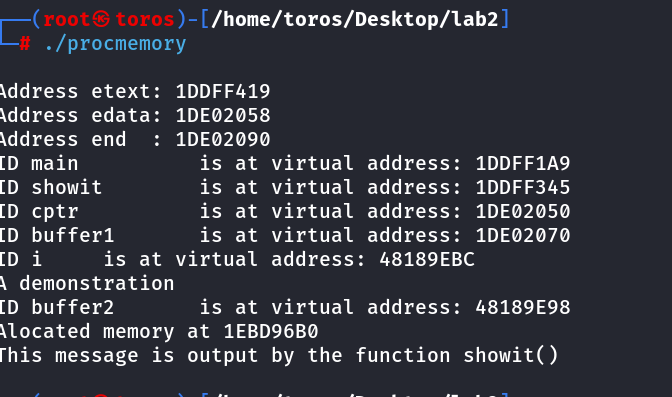
2.3) 

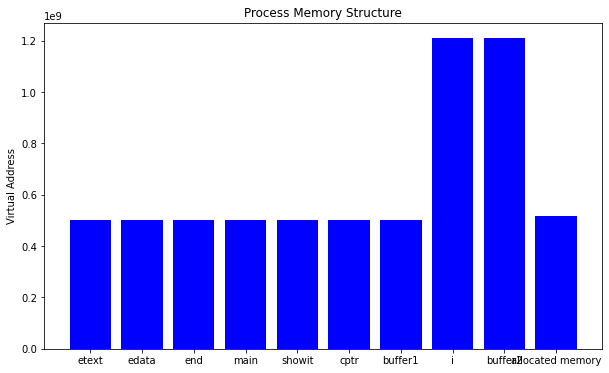
Çocuk süreci (simple.c'yi çalıştıran süreç) artık 3 saniye boyunca uyuyacak, bu nedenle ebeveyn sürecinden (procident.c) önce tamamlanacaktır.

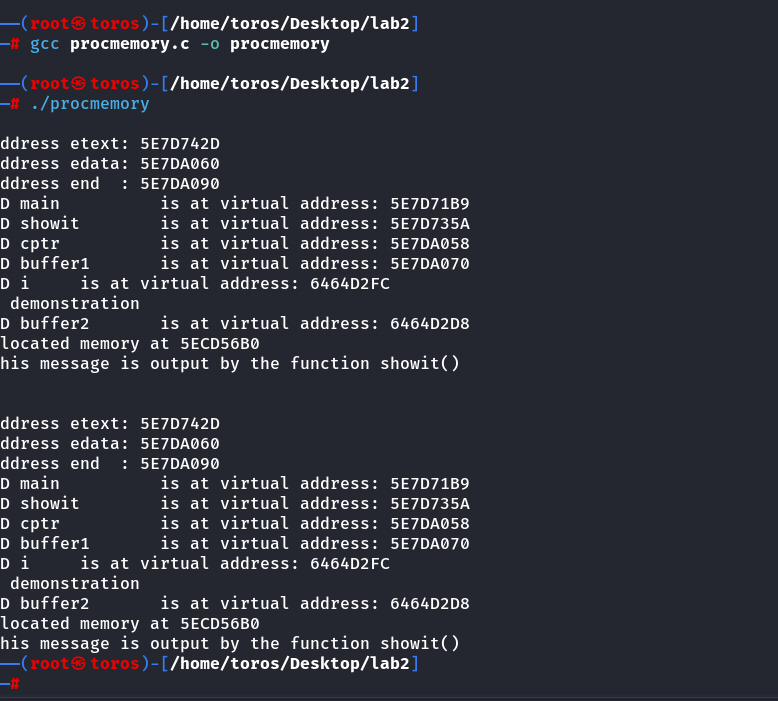
Ebeveyn süreci, çocuk sürecinden önce sonlanacak ve bu durum ebeveyn sürecin PID değerini etkilemeyecektir. PID değeri, sürecin kimliğini belirtir ve süreç sona erdiğinde bu kimlik serbest bırakılmaz veya değiştirilmez.

3.1)



3.2)



3.3)

fork() fonksiyonu çağrıldığında, yeni bir süreç oluşturulur ve bu yeni süreç, çağrıldığı sürecin tam bir kopyasıdır. her iki süreç de aynı sanal adres uzayına sahiptir, ancak kendi bağımsız bellek kopyalarına sahiptirler. her iki süreçte de i değişkeni aynı sanal adresi paylaşacak ancak kendi değerini tutacaktır. Bu durum, eğer bir süreç i'yi değiştirirse, bu değişiklik diğer süreci etkilemeyecek ve her iki süreç bağımsız olarak devam edecektir.