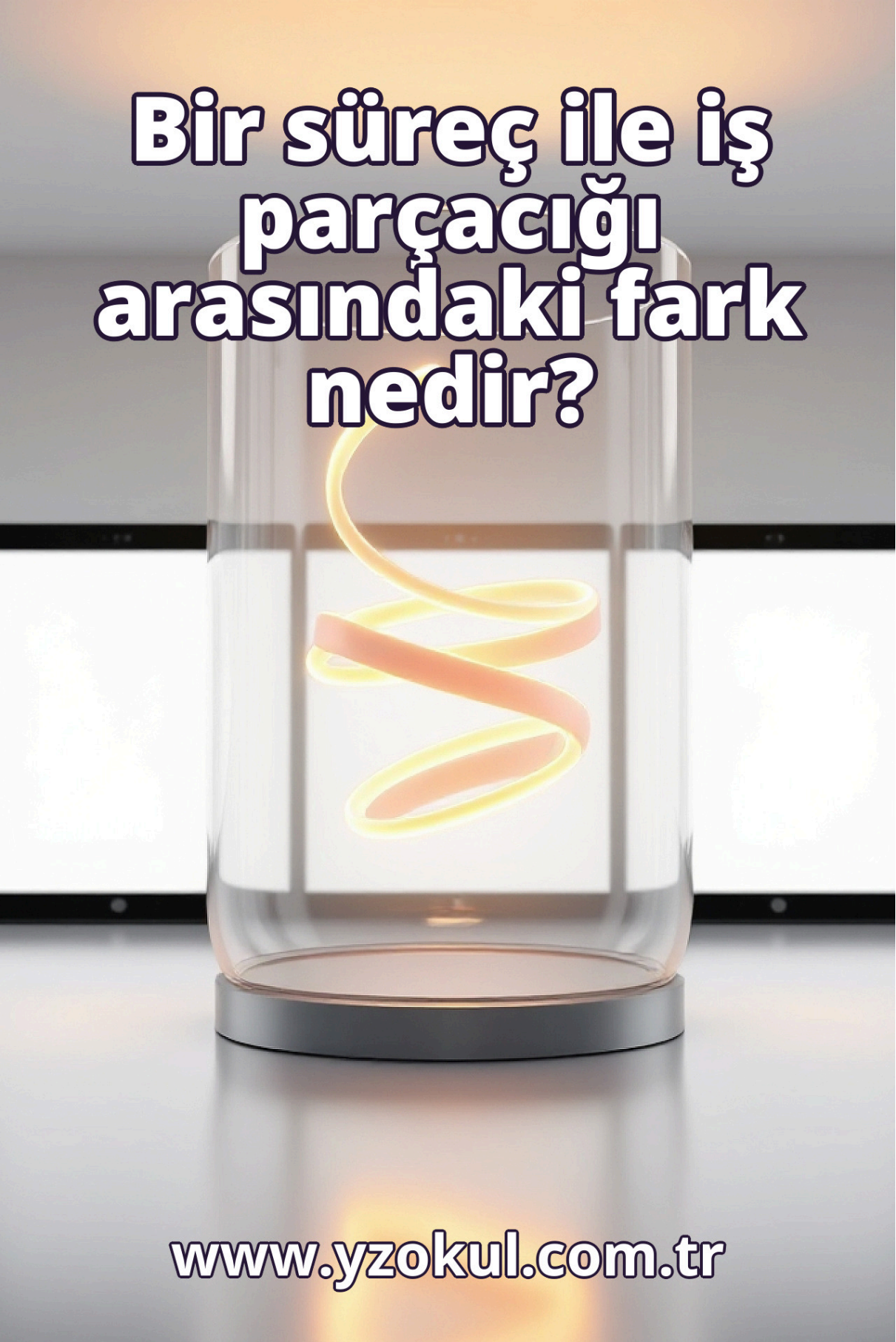
Bir süreç ile iş parçacığı arasındaki fark nedir?



www.yzokul.com.tr

Bir süreç ile iş parçacığı arasındaki fark nedir?



Bir süreç ile iş parçacığı arasındaki fark nedir? Kısa yanıt: süreç çalışan bir programdır, iş parçacığı ise onun içindeki bir yürütme akışıdır. İşletim sistemi her ikisini de yönetir. Bu kitap her fikri sıfırdan başlayarak adım adım açıklar; böylece tam olarak nasıl farklılaştıklarını ve nasıl birlikte çalıştıklarını görürsün.



Program, diskte saklanan bir dizi komuttur; henüz izlenmeyen bir tarif gibidir. Bu noktada henüz hiçbir şey çalışmıyordur. Komutlar öylece durur ve bekler. Örneğin bilgisayarına kurulu bir metin düzenleyici bir programdır. Ancak onu başlattığında etkin hale gelir.



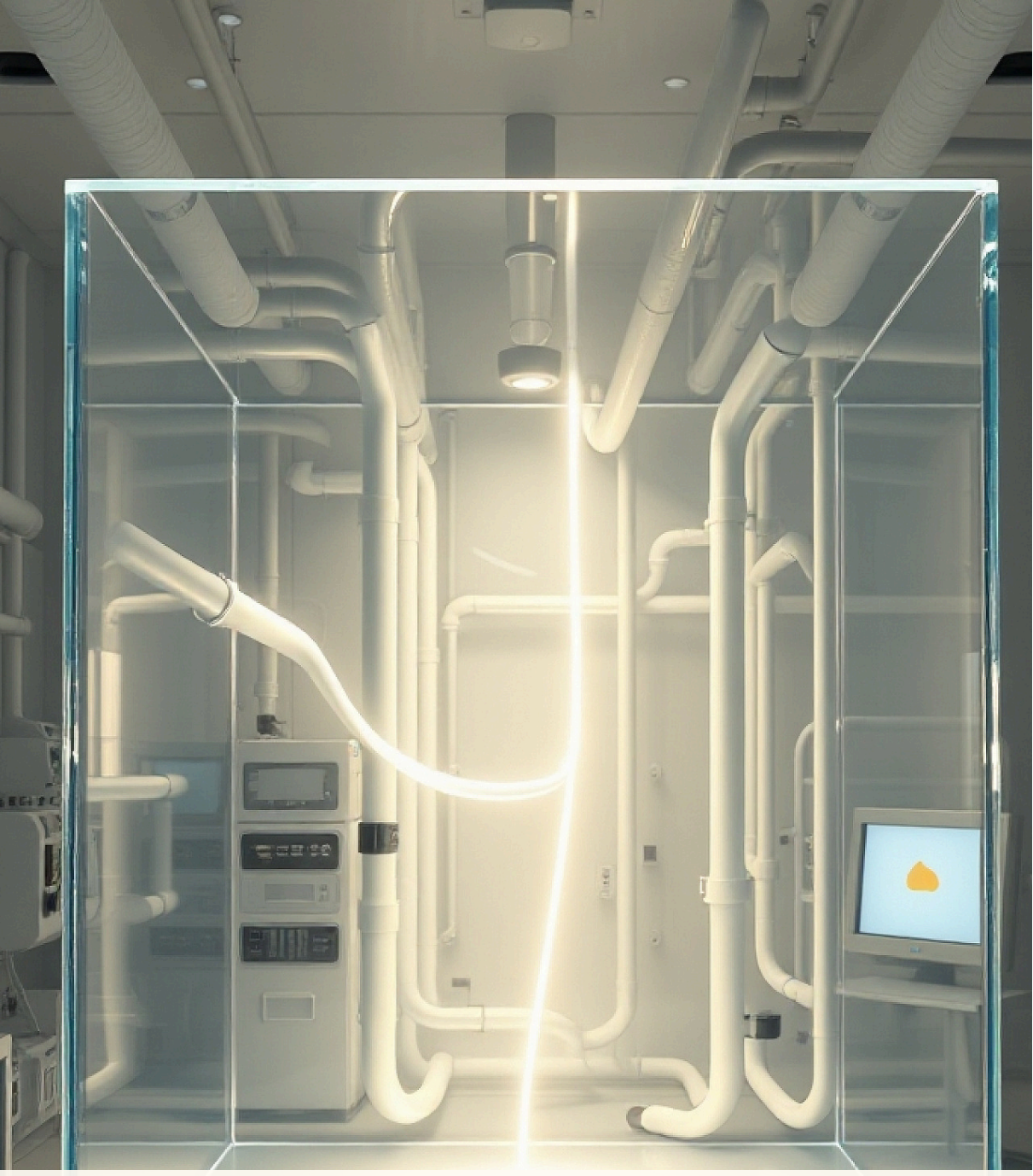
Bir program çalışmaya başladığında işletim sistemi onu bir sürece dönüştürür. Süreç, kendi belleği ve durumu olan, eylem halindeki bir programdır. İşletim sistemi komutları yükler, sürece bellek verir ve yürütmeye başlar. Örneğin metin düzenleyicini açtığında artık kullanabileceğin bir süreç oluşur.



İşletim sistemi tüm süreçlerin yöneticisidir. Her süreci takip eder ve her birinin ne zaman çalışacağına karar verir. Her sürece kendi alanını verir; böylece süreçler birbirine karışmaz. Örneğin metin düzenleyicin ve web tarayıcın ayrı süreçler olarak çalışır, her birinin kendi belleği vardır ve işletim sistemi aralarında geçiş yapar.



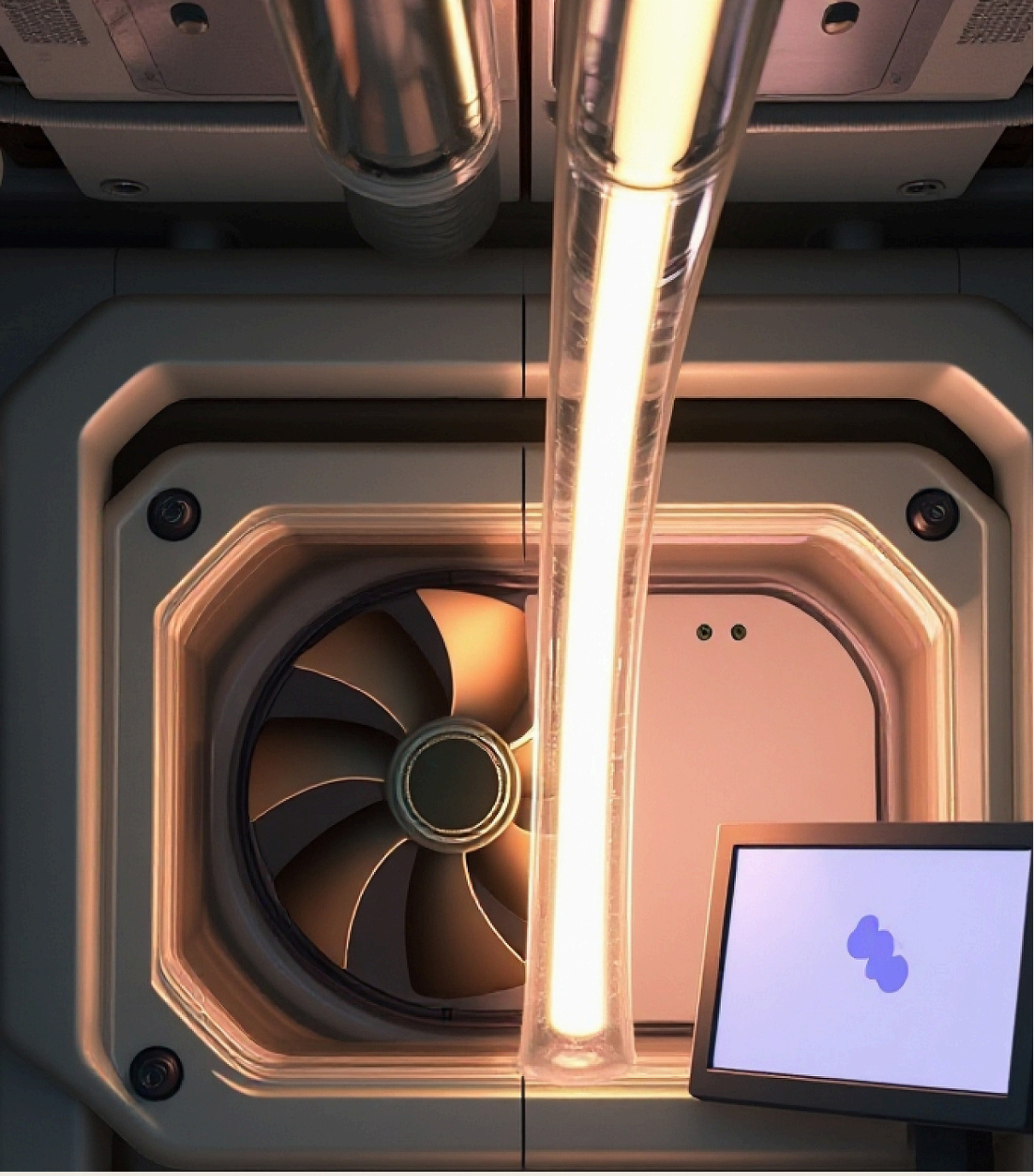
Tek bir süreç, içinde çalışan birkaç yürütme akışı barındırabilir. Bu akışların her birine iş parçacığı denir. Süreci bir atölye, iş parçacıklarını da içindeki, her biri komutları izleyen çalışanlar olarak düşün. Örneğin bir müzik çalar sürecinde sesi çalan bir iş parçacığı, ekranı güncelleyen başka bir iş parçacığı olabilir. Bir süreçte her zaman en az bir iş parçacığı bulunur.



İş parçacığı, bir süreç içindeki tek bir yürütme yoludur; aynı atölyede komutları izleyen tek bir çalışan gibidir. Programın komutları arasında her seferinde bir adım ilerler. Bir süreçte her zaman en az bir iş parçacığı bulunur ve birçok iş parçacığı olabilir. Örneğin basit bir hesap makinesini başlattığında, düğme basışlarını işleyen tek bir iş parçacığıyla başlar.



Bir iş parçacığının komutlar arasında baştan sona her seferinde bir adım ilerlediğini düşün. Bir komutu okur, işi yapar, sonra bir sonrakine geçer. Bu, iş parçacığının saklanan bir dosya değil, etkin bir akış olduğunu açıkça gösterir. Örneğin bir hesap makinesi uygulamasındaki iş parçacığı bir sayıyı okur, senin bir düğmeye basmanı bekler, sonra sonucu gösterir.



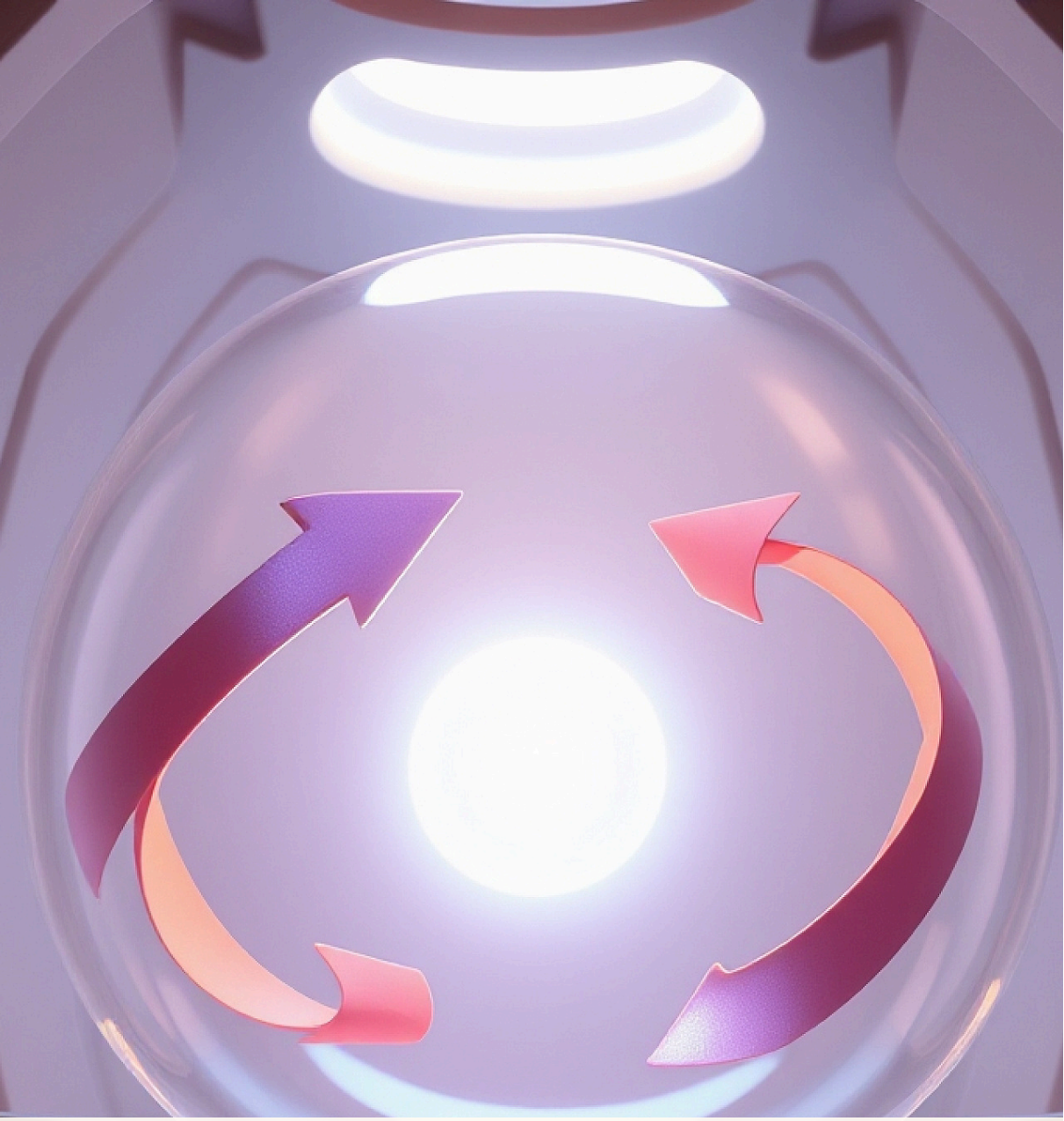
Aynı süreçteki iki iş parçacığı aynı anda farklı işler yapabilir. Bir iş parçacığı ekranla ilgilenirken bir diğeri arka plan işini yapabilir. Bu, bir sürecin neden birkaç iş parçacığına ihtiyaç duyabileceğini gösterir. Örneğin bir video oynatıcı, görüntüyü çizmek için bir iş parçacığı, video dosyasını diskten okumak için başka bir iş parçacığı kullanır; böylece görüntü akıcı kalır.



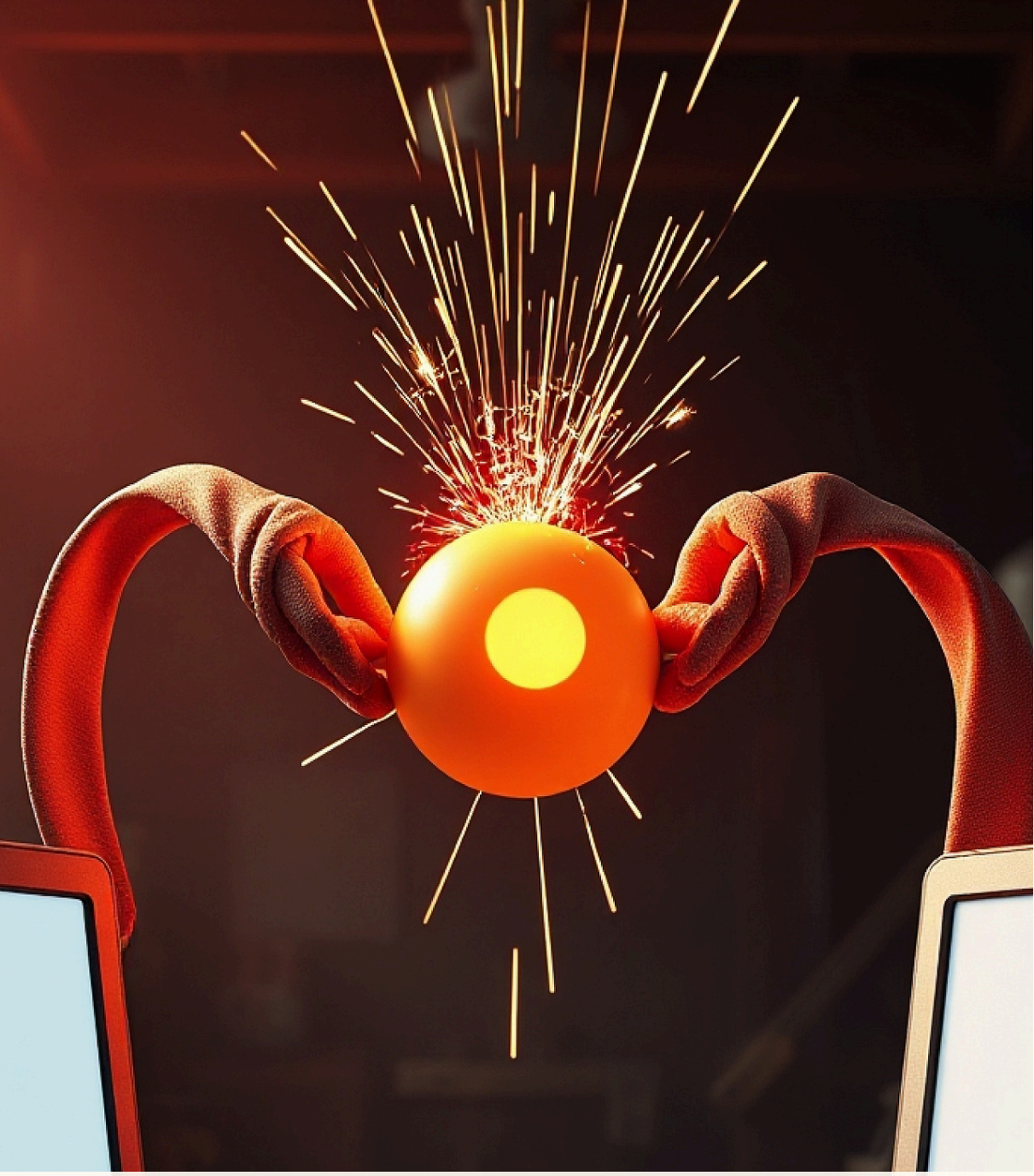
Aynı süreçteki iş parçacıkları sürecin belleğini ve kaynaklarını paylaşır. Aynı veriyi kopya oluşturmadan okuyabilir ve kullanabilirler. Buna bellek paylaşımı denir. Örneğin bir müzik çalardaki iki iş parçacığı, belleğe zaten yüklenmiş şarkı listesine birlikte erişebilir; böylece bir iş parçacığı şarkıyı çalarken bir diğeri başlığını gösterebilir.



Farklı süreçlerin kendi ayrı bellek alanları vardır. Bir süreç, başka bir sürecin verisine öylece ulaşamaz. İşletim sistemi bu ayrımı uygular. Örneğin web tarayıcın metin düzenleyicinin belleğini doğrudan okuyamaz. Bilgi alışverişi yapmaları gerekiyorsa işletim sisteminin sağladığı özel bir kanalı kullanmaları gerekir.



Paylaşım elverişlidir çünkü iş parçacıkları bilgiyi hızla aktarabilir ve aynı görev üzerinde birlikte çalışabilir. Veriyi kopyalamaları veya işletim sisteminden yardım istemeleri gerekmez. Bu, iş parçacıklarını ayrı süreçlere göre daha hafif ve koordine edilmesi daha hızlı kılar. Örneğin bir fotoğraf düzenleyicideki iki iş parçacığı aynı görüntü üzerinde aynı anda çalışabilir; biri filtre uygularken diğeri önizlemeyi günceller.



İki iş parçacığı aynı veriye aynı anda dokunduğunda sonuç yanlış veya kafa karıştırıcı olabilir. Bir iş parçacığı, diğeri hâlâ okurken bir değeri değiştirebilir. Paylaşılan veriye dikkatli ve düzenli bir şekilde erişilmelidir. Örneğin bir bankacılık uygulamasındaki iki iş parçacığı aynı bakiyeyi aynı anda güncellerse son tutar yanlış olabilir.



Paylaşılan veriye erişim, iş parçacıkları çarpışmak yerine sırayla hareket edecek şekilde yönetilmelidir. Yaygın bir yol kilit kullanmaktır: bir iş parçacığı çalışırken veriyi kilitler, diğer iş parçacıkları o kilidi açana kadar bekler. Bu, paylaşılan belleği hatalara yol açmadan yararlı tutar. Örneğin bir alışveriş sepetini güncelleyen iş parçacığı sepeti kilitleyebilir, miktarı değiştirebilir, sonra başka bir iş parçacığı güvenle okuyabilsin diye kilidi açabilir.



Yaygın bir hata, iş parçacığını süreçle aynı şey sanmaktır. İş parçacığı bir sürecin içinde yaşar; süreç kaptır, iş parçacığı ise onun içindeki akıştır. Bir süreçte birçok iş parçacığı olabilir, ancak her iş parçacığı tam olarak bir sürece aittir. Örneğin bir web tarayıcısı tek bir süreçtir ve her sekme bu süreç içinde kendi iş parçacığında çalışabilir.



Bir diğer yaygın hata, paylaşılan belleğin her zaman zararsız olduğunu varsaymaktır. Paylaşım hız getirir ama çatışmalardan kaçınmak için dikkatli yönetim de gerektirir. Dikkat edilmezse iş parçacıkları birbirlerinin değişikliklerinin üzerine yazabilir. Örneğin aynı listeye aynı anda öğe ekleyen iki iş parçacığı bir öğeyi kaybedebilir. İş parçacıkları veri paylaştığında her zaman uygun koordinasyon kullan.



Özetle: süreç, işletim sistemi tarafından yönetilen çalışan bir programdır; iş parçacığı ise onun içindeki bir yürütme akışıdır. Aynı süreçteki iş parçacıkları belleği paylaşır; bu elverişlidir ama dikkatli yönetilmelidir. Süreçlerin ayrı belleği vardır, iş parçacıkları ise belleklerini paylaşır. Unutma: süreç kaptır, iş parçacığı ise onun içindeki akıştır.