BLGM 343 – DENEY 2*
İŞLEMLER

Amaçlar

1. POSIX tabanlı işletim sistemlerinde işlemler
2. Yeni işlem oluşturma
3. İşlemlerin tamamlanmasını bekleme

Ön Bilgi

Bu deneyin amacı UNIX üzerinde çalışan işlemleri detaylı olarak incelemektir. Basit programlar kullanarak öğrencinin konu üzerindeki bilgisinin arttırılması hedeflenmektedir. Bu deney üç kısımdan oluşmaktadır. İlk kısımda ana işlem ile alt işlemleri arasındaki bağlantı incelenmektedir.

İşlemler

POSIX tabanlı işletim sistemlerinde her çalışan program bir işlemdir. Her işlemin de bir işlem numarası vardır. Bu numaraya **getpid()** sistem çağrısıyla erişilebilir. Ayrıca her işlemin bir de ana işlemi vardır. Bu işlem numarasına da **getppid()** sistem çağrısıyla ulaşılabilir. Yalnızca **init**, başlangıç programının ana işlemi yoktur. Ana işlemi sonlanan programların ana işlemi init olur. Bir işlem istediği sayıda alt işlem açabilir. Bunun için **fork()** sistem çağrısını kullanması yeterlidir. Eğer ana işlem alt işlemlerini beklemek isterse **wait()** sistem çağrısını kullanabilir. Bir işlem **sleep()** sistem çağrısı kullanılarak belli bir süre duraklamaya alınabilir. Bir işlem **exit()** sistem çağrısıyla sonlandırılabilir.

Deney

Bu deneyde programımızı adım adım oluşturacağız. Her adımı yazdıktan sonra programınızı çalıştırarak test ediniz.

İlk adım olarak çalışan programımızın işlem numarasını yazdırmayı yapalım. Bunun için programımıza **<unistd.h>** kütüphanesini eklememiz gerekiyor. Bunun ardından aşağıdaki gibi

* BLGM 343 dersi için Güz 2013/2014 döneminde Gürcü Öz ve Cem Kalyoncu tarafından hazırlanmıştır

işlem numarasını yazdırabiliriz:

```
printf("İşlem numarası: %d\n", getpid());
```

Bundan sonraki adımda bizim işlemimizin üst işleminin numarasını yazdıralım. Bunun için **getppid()** sistem çağrını kullanmamız gerekiyor.

Bir sonraki adımda programınızın ana işleminin bir alt işlem oluşturmasını sağlayalım. Bunun için **fork()** sistem çağrısını kullanmanız gerekiyor. Bu çağrının ardından ekrana her hangi bir yazı yazdırarak bu yazının kaç kere yazdırıldığını görelim.

Bir sonraki adıma geçmeden önce bir başka test yapalım. İlk **fork()** sistem çağrısının ardından ikinci bir **fork()** daha ekleyip az önceki yazının kaç kere çağırıldığını görelim. Neden sonuç bu şekilde çıkıyor? Sonucu gördükten sonra, bir sonraki adıma geçmeden ikinci **fork()** çağrısını silelim.

Sıradaki adımda **fork()** sistem çağrısından sonra işlem ve üst işlem numaralarının gösterilmesini sağlayalım. Hangi işlemin alt işlem olduğunu anlayabiliyor musunuz?

Bir sonraki adım olarak alt işlem ve ana işleme farklı işler yaptırmayı deneyelim. Fark ettiğimiz üzere **fork()** sistem çağrısının hemen ardından ana işlem ve alt işlem programı aynı anda çalıştırmaya devam ederler. Bu noktada, iki işlemin değişkenleri, açtığı dosyalar ve çalışma satırları tam olarak aynıdır. Bu durumdan dolayı, alt işlemle ana işlemi ayırt etmek için özel bir mekanizma gerekmektedir. Bunun için **fork()** sistem çağrısından dönen değer kullanılmaktadır. **fork()** sistem çağrısı çalıştıktan sonra ana işlem ve alt işlem için farklı değerler döndürmektedir. Ana işleme alt işlem numarasını verirken, alt işleme ise 0 değerini vermektedir. Aşağıda bu duruma bir örnek verilmiştir:

```
int sonuc=fork();
if(sonuc==0) {
     printf("Alt işlem.\n");
}
else {
     printf("Üst işlem.\n");
}
```

Biz de bu bilgiyi kullanarak hangi işlemin ana işlem hangisinin alt işlem olduğunu anlayabiliriz. Bu bilgileri kullanarak ana işlemin ve alt işlemin farklı bilgiler yazdırmasını sağlayalım.

sleep() sistem çağrısı bir işlemin çalışmasını belli bir süre durdurur. Örneğin aşağıdaki program işlem tamamlandı yazısını, işlem yapılıyor yazısından 4 saniye sonra yazacaktır.

```
printf("İşlem yapılıyor...\n");
sleep(4);
printf("İşlem tamamlandı.\n");
```

Biz de programımızın yalnızca ana işlem kodunun en başına 5 saniyelik bir gecikme ekleyelim. Programımız sonlanmadan önce 5 saniye bekliyor mu? Bu geçikmeyi alt işlem koduna taşıyalım. Program hala 5 saniye bekliyor mu? Neden? Not: Programın sonlandığını, komut istemcisinin (`[kullanıcı@host dizin]$`) ekranda görünmesinden anlayabiliriz .

Yukarıdaki adımda yazdırdığımız bilgilere ek olarak, hem ana işlemde hem de alt işlemde, işlem tipini belirterek işlem numarasını ve üst işlem numarasını yazdıralım. Bu yazdırma işlemi, alt işlemdeki bekleme kodundan sonra eklenmelidir. Programımızı çalıştırdığımızda alt işlemin üst işlem numarası kaç olmakta? Alt işlemin, ana işlem tarafından açılmasına rağmen alt işlemin üst işlem numarası ana işlem değildir, neden?

Ana işlemin alt işlemi beklemesini **wait()** sistem çağrısıyla sağlayabiliriz. Ana işlemin yapmakta olduğun işlemin en sonuna **wait()** sistem çağrısını ve ondan sonra "işlem tamamlandı" yazısını yazdırmak için gerekli kodu ekleyelim. Ana işlem "işlem tamamlandı" yazısını ne zaman yazmakta? Alt işlemin üst işlem numarasından bir değişiklik oldu mu? Neden? Burada unutmamız gereken nokta her **wait()** sistem çağrısının bir tane alt işlemi bekleyeceğidir.

İstenmesi durumunda **wait()** sistem çağrısı, alt işlemin sonlanırken belirttiği çıkış numarasını alabilir. Böyle bir durumda **wait()** sistem çağrısına int tipinde bir işaretçi verilmesi gerekir. Aşağıda bu duruma örnek bir program verilmektedir:

```
if(fork()) {
     int no, durum;
     printf("Alt işlem bekleniyor...\n");
     no=wait(&durum);
     printf("%d numaralı alt işlem %d değerini döndürdü.\n",
            no, WEXITSTATUS(durum));
}
else {
     printf("Alt işlem çıkış değeri olarak 2 döndürecek!\n");
     exit(2);
}
```

Programımızda son adım olarak, programa verilen komut satırı parametresini alt işlemin çıkış değeri olarak vermesini, ve bu değerin ana işlem tarafından ekranda gösterilmesini sağlayalım. Bunun için yukarıdaki programdan faydalanabilirsiniz. Bir alt programın çıkış değerinin hangi amaç için kullanılabileceğini düşünüp listeleyiniz.

Çalışma

Bu bölüm kendi çalışmanız için verilmiştir. Bir programın bütün alt işlemlerini beklemesini sağlayan kodu yazarak çalıştırınız. Sistemin testi için komut satırından belirtilen sayıda alt işlem açınız. Her alt işlem geriye rastgele bir çıkış değeri döndürmesini, ana işlemin de hangi alt işlemin hangi çıkış değeriyle sonlandığını ekrana göstermesini sağlayınız.