Çoklu İşlem

Şimdiye kadar, Biz zaten bir program çalıştığında bunu biliyoruz, bu daha sonra bir proses şeklini alır. (programı bir çalıştırma örneği).Gerçekten, işletim sistemi çekirdeği bir grup proseslerden oluşur. Detaylara girmeden önce proses kontrolü nasıl yapılmaktadır, Bir çoklu işletim sistemi içeriğinde ne yapar önce ona göz atalım.

İşletim sistemi temelde hiç çoklu görevlendirme özelliği bulunmayan sırala program yükleyicisidir. (DOS ya da CP/M gibi). Sonraki seviye, bir işletim sistemi çoklu işleme yardımcı olabilir. Bu sistemlerde, bir proses sonraki çalıştırmadan önce işlemci üzerinde beklemekten gönüllü olarak vazgeçmek zorunda (tek bir program hatası tüm makineleri etkileyebilir). Çoklu işlem yardımlaşması zaten kullanılmıyor, bu sadece bir periyodik saat kesintisi, bir bellek yönetimi veya her ikisi için donanım eksikliği olan makinelerde geçici bir uyum oldu.

Unix bir çoklu işlem önleyici kullanır, hangi zaman dilimleri bir zamanlayıcı tarafından tahsis edildiği veya diğerine el ile yapacağı kontrol çalışan proses önceden sahip olur. Hemen hemen tüm modern işletim sistemleri önlemeyi destekler.

Donanım kısıtlamaları bir yazılımı nasıl etkilediği kanaatine varmak için, en kolayı Microsoft'un tüm kötü yanları hakkında konuşmaktır. 1981'de, Microsoft yeni IBM PC üzerinde IBM ile tarihi anlaşma yaptı. Bill Gates 50.000$'a Seattle Computer Product QDOS satın aldı(Seattle Bilgisayar Ürünleri). QDOS 6 hafta içerisinde SCP Tim Paterson tarafından yazılmış ve gerçek bir işletim sistemi aldı. Aslında, QDOS açılımı Quick and Dirty Operating System. Bir çok ucuz işlemci, Intel 8086, işlemci olarak seçildi çünkü, IBM gerçek bir işletim sistemini göze alamazdı. Aslında, Intel, 1985 yılında 386 yayımlanana kadar, IBM bilgisayarlarında düz bir adres alanı ve periyodik bir donanım saat kesintisi yoktu.

Çok ironik, IBM çok güçlü olduğu 386 bulundu çünkü 1986 yılında IBM'in hakim olmasını serbest kıldı (bu gücünü, fiyatını vs etkileyebilirdi IBM bu yüzden bunu korumak istedi dolayısıyla daha çok yüksek sonlu mainframelerini satarlar). Compaq çok yakında 386 bilgisayarları bırakmasıyla IBM'e baskın geldi ve sonra geçmişte rol aldı.

ps

Unix'de ,şu anda ps komutu ile çalışan tüm işlemlerin bir listesini yapabilirsiniz. Aşağıdakini deneyin:

```
UNIX> ps x
25919 co IW    0:00 -csh (csh)
29620 co IW    0:00 xinit
29621 co S     0:32 X :0
29645 p0 S     0:01 xclock -geometry 100x100-0-0 -update 60
29646 p0 S     0:01 spermwatch -newmail xloadimage -onroot /blugreen/homes/plan
29647 p0 IW    0:00 /bin/sh /blugreen/homes/plank/bin/xyobiff
29669 p0 S     0:01 twm
29676 p1 IW    0:00 yobiff plank
  381 p2 S     0:01 -sh (csh)
```

```
  713 p2 R     0:00 ps x
  693 p3 S     0:00 vi lecture
29678 p3 IW    0:00 -sh (csh)
29684 p4 IW    0:00 -sh (csh)
29686 p5 IW    0:00 -sh (csh)
29685 p6 IW    0:00 -sh (csh)
UNIX>
```

Yukarıdaki gibi bir şey göreceksiniz. Dikkatinizi çekmiş olmalı, her **csh**'ın kendi süreci vardır. İlk Sayıların bulunduğu ilk sutuna "pid"ler adı verilir, "Proses ID" numaraları vardır . Çalışan her bir proses olduğunda bilgisayarında bu çalışan proses için benzersiz negatif olmayan bir pid vardır .

Proses No. 0,1 ve 2

Pid'ler için 0,1 ve 2 ile üç özel proses vardır. Proses 0 genellikle **swapper** olarak bilinen programlayıcı (yukarıya bakınız). Bu çekirdek bir prosestir ve disk üzerinde bu işleme karşılık hiçbir program bulunmaz (diğer bir değişle çekirdeğin bir parçası). Bazı sistemlerde , swapper **sched** olarak gösterilmiştir.

Proses 1 **init** program**. init** çekirdek prosesi değildir. Ön yükleme prosedürü sonunda çekirdek tarafından çalıştırılır . Program dosyaları **init** için /sbin/init ya da /etc/init (eski sistem üzerinde). Proses çekirdek önyükleyicisinden sonra bir Unix sistemini getirir . **init** sistemi okuma-bağımlı başlatma dosyaları (/etc/rc*) ve sistemi belli bir duruma getirir. **init** admin ayrıcalığı ile normal bir prosestir ve geçerliliğini asla kaybetmez.

Sanal bellek işe bazı Unix gerçeklenimlerinin, proses ID 2 sanal bellek sürecini destekleyen belleği konumundadır. Bu işlemin mümkün isimleri **pagedaemon** ya da **pageout**, vs içerir. Bu çekirdek prosesidir.

Aslında, bütün prosesler tek haneli pid ile tümü özel proseslerdir. Ancak, hangi proses hangi sistemle daha bağımlıdır.

Onların kullanımına bakabilirsin **ps aux**: (Bazı makineler **ps** için farklı ayarlar kullanır – Eğer "**ps aux**" çalışmazsa, bunu dene "**ps -ef**").

```
UNIX> ps aux | grep root
...
root       3  0.1  0.0    0    0 ?       S   Oct 22  4:14 fsflush
...
root       0  0.0  0.0    0    0 ?       T   Oct 22  0:17 sched
root       1  0.0  0.3  848  656 ?        S   Oct 22  0:04 /etc/init -
root       2  0.0  0.0    0    0 ?       S   Oct 22  0:00 pageout
...
UNIX>
```

---

getpid, getppid

Her işlemin bir "parent" prosesi vardır. Bunu oluşturan işlem budur. **getpid()** ve **getppid()** komutlarını kullanımıyla pid ve parent pid'i görebilirsiniz . **showpid.c** pid ve onun parent pid'ini yazdıran basit bir program :

```
main()
{
  printf("My pid = %d.  My parent's pid = %d\n", getpid(), getppid());
}
```

Bunu her çalıştırdığımızda, farklı bir pid, ama hep aynı parent pid gösterilecektir :

```
UNIX> showpid
My pid = 854.  My parent's pid = 381
UNIX> showpid
My pid = 855.  My parent's pid = 381
UNIX> showpid
My pid = 856.  My parent's pid = 381
UNIX> showpid
My pid = 857.  My parent's pid = 381
UNIX> ps x
...
  381 p2 S     0:01 -sh (csh)
...
UNIX>
```

Gördüğünüz gibi, **csh** bunların her birinin üstü konumundadır. Bunun nedeni **csh** içine tiplerinin komutları yazılmış olmasıdır, daha bu sonra **showpid** prosesleri hazırlanır.