CS360 Ders Notları – Prosesler Hakkında Daha Fazlası

- Jian Huang
- Dizin: **~huangj/cs360/notes/ProcMisc**

Unix; **fork**, **exec** ve **wait** sistem çağrılarını kullanır. Bunların hepsi sahip olunan proseslerin kontrolüdür. Aslında, bunlar oldukça yeterlidir. Burada son olarak, yaygın olarak kullandığımız Unix proses kontrolü hakkında bir ANSI C standart fonksiyonundan bahsedeceğiz.

System

ANSI C'de, bir program içerisinden komut string'lerinin çalıştırılması için stdlib.h'ta tanımlandı. Örneğin, *system("ps x")* gibi. Ancak, *system*'in tam olarak çalıştırılması sistem bağımlıdır.

```
int system(const char *cmdstring);
```

Unix üzerinde, *system;* **fork**, **exec** ve **wait** çağrılarının kullanımıyla uygulanmıştır. Diğer bir ekstra bilgi ise; *system* hata kontrolünü ve sinyal kullanımını da dahil eder. Henüz sinyaller hakkında konuşmadığımızdan, system'in çağrıldığı proseslerin icrasında ortaya çıkabilecek hatalara odaklanalım:

Birincisi, *fork* başarısız olabilir ya da *wait* bir hata geri döndürebilir. Bu durumda, *system* hatayı belirtmek için errno ile -1 değerini geri döndürür.
İkincisi, *exec* başarısız olabilir (kabuk çalıştırılmamış olabilir), bu kez geri dönüş değeri 127 olur. Aksi halde, hepsi başarılı olursa, o zaman system'in geri dönüş değeri kabuğun sonlandırma durumudur. Aşağıda sinyal'lerin dikkate alınmadığı örnek bir system uygulaması görülmektedir.

```
#include < sys/types.h >
#include < sys/wait.h >
#include < errno.h >
#include < unistd.h >

int system (const char * cmdstring)
{
  pid_t pid;
  int   status;

  if (cmdstring == NULL)
     return (1);   /* Her zaman bir komut yapar */

  if ( (pid = fork() ) < 0) {
      status = -1;  /* Muhtemelen prosesler dışında */
```

```
    }
    else if (pid == 0) {  /* Yavru proses */
        execlp("/bin/sh","-c",cmdstring, (char *)0);
        _exit(127);      /* Hata! */
    }
    else {
        /* Her şey iyiyse… */
        while (wait(&status) < 0)
          if (errno != EINTR) {
              /* Normal bir kesme değil */
              status = -1;
              break;
          }
    }

    return (status);
  }
```

Proses Zamanı

Bugüne kadar, hakkında bahsettiğimiz **takvim zamanı** kategorisi altına giren tüm zaman ölçümleri, Epoch'tan(*) beri system tarafından korunmuş saniye numarasıdır, (* Epoch bir zaman referans tarihidir.) 00:00:00 1 Ocak 1970, Eşgüdümlü Evrensel Zaman (UTC). UTC aynı zamanda Greenwich Mean Time (Greenwich Ortalama Güneş Saati) olarak bilinir.

Prosesler hakkında tüm bunları bildikten sonra, **proses zamanı** hakkında konuşalım ki aynı zamanda **CPU zamanı** olarak da anılır. **Proses zamanı** bir proses tarafından kullanılan ne kadar merkezi işlemci kaynağı bulunduğunu ifade eden bir kavramdır. Takvim zamanı gibi, proses zamanı da tarihsel olarak işlem süresi saniyede 50, 60 ya da 100 tik-tak olan saat tik-tak'larıyla ölçülür.

Her proses için, system üç değer korur:

```
- clock zamanı (duvar saati zamanı)
- kullanıcı CPU zamanı
- system CPU zamanı
```

Unix'te, proses zamanı çağırmayla toplanır.

```
#include < sys/times.h >
clock_t times (struct tms * buf);
```

tms structure takip eden tanımlaya sahiptir;

```
struct tms {
  clock_t tms_utime;  /* kullanıcı CPU zamanı */
  clock_t tms_stime;  /* system CPU zamanı */
  clock_t tms_cutime; /* kullanıcı CPU zamanı, sonlandırılmış çocuk */
  clock_t tms_cstime; /* system CPU time, sonlandırılmış çocuk */
}
```

Not olarak, *time*'ın geri dönüş değer duvar saati zamanıdır, geçmişteki keyfi bir noktadan ölçülmüştür. Kimse bu mutlak değeri kullanmaz, ama göreceli olanı kullanmak daha iyidir. Bu, o kod parçasının duvar saati çalışma zamanını elde etmek için programda iki noktadan

elde edilen iki geri dönüş değeri arasında farklı değerler alır. ANSI C standart *time.h* duvar saati zamanını ölçmek için kullanılmış olabilir, fakat CPU zamanı için kullanılmaz.

u/stime sırasıyla, mevcut proseslerin harcadığı kullanıcı uzayındaki ve ya sistem çağrılarındaki CPU zamanını ölçer. cu/stime mevcut proseslerin tüm çocuk prosesleri için beklenilen zamanı aynı miktarda ölçer.

Her saniyede kaç adet tıkırtı(saat sesi) olduğunu bulmak için, bunu kullanın;

```
#include < unistd.h >
long clktck = 0;
clktck = sysconf(_SC_CLK_TCK);
```