Sistem Programlama

Ders 13

Dr. Öğr. Üyesi Mehmet Dinçer Erbaş
Bolu Abant İzzet Baysal Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

İşlem kontrolü

- Yarış durumu (race conditions)
  - Birden fazla işlem ortak veri üzerinde bir şeyler yapmak istediğinde ve sonucun hangi işlemlerin çalışma sırasına göre değişeceği durumlarda yarış durumu oluşur.
- fork fonksiyonu nedeniyle yarış durumu oluşabilir.
  - Genelde hangi işlemin önce çalışacağını bilemeyiz.
  - Hangi işlemin önce çalışacağını bilsek bile bundan sonra ne olacağı sistemin yük durumuna ve kernel tarafından görevlendirme algoritmasına bağlıdır.

Yarışma durumu

- Bir işlem alt işleminin sonlanmasını beklemek isterse wait fonksiyonlarından birini kullanabilir.
- Bir işlem üst işleminin sonlanmasını beklemek isterse aşağıdakkine benzer bir döngü kullanılabilir.

```
while(getppid() != 1)
    sleep();
```

- Bu tip döngü kullanarak bir durumu kontrol etmeye polling (kuyruklama) denir.
  - Bu yöntem CPU zamanını boşa harcadığı için pek tercih edilmez.

Yarışma durumu

- Yarışma durumunu engellemek için, işlemler arasında bir iletişim yöntemi olmalıdır.
  - Sinyaller kullanılabilir.
  - Farklı işlemler arası haberleşme yöntemleri kullanılabilir.
- Üst ve alt işlemler için genellikle aşağıdakine benzer senaryolar gözlemlenir.
  - Fork işlemi sonrası hem alt hem üst işlemin yapacağı işler olur. Örneğin üst işlem alt işlemin işlem numarasını kullanarak bir kayıt dosyasını değiştirebilir ve alt işlem üst işlem için yeni bir dosya oluşturabilir. Bu örnekte, her iki işlemin birbirine yapacaklarını tamamladıklarını bildirebilmesi gerekir.
  - Örnek29
  - Örnek30.

Yarışma durumu

```
#include "apue.h"
TELL_WAIT();
/* set things up for TELL_xxx & WAIT_xxx */
if ((pid = fork()) < 0) {
        err_sys("fork error");

        } else if (pid == 0) {          * child *

        TELL_PARENT(getppid());

        WAIT_PARENT();

        exit(0);

}

TELL_CHILD(pid);
WAIT_CHILD();
exit(0);
```

- TELL_WAIT, TELL_PARENT, TELL_CHILD ve WAIT_PARENT fonskiyonları apue kütüphanesinde tanımlanmıştır.

exec fonksiyonları

- Daha önce bahsettiğimiz üzere fork fonksiyonu ile yeni bir işlem yaratıp bu işlemde exec fonksiyonları ile bir başka programı çalıştırabiliriz.
- Bir işlem exec fonksiyonunu çağırdığında, çağıran işlemin yerine yeni program geçer ve yeni programın main fonksiyonu çalışmaya başlar.
  - exec fonksiyonu çalışması sonrası işlem numarası değişmez. Çünkü yeni bir işlem yaratılmamıştır.
  - Yapılan işlemin hafızadaki yerine yeni veriler yazılır. Metin kısmı, heap ve yığın bölümlerine yeni programın verileri yazılır.

exec fonksiyonları

- 6 farklı exec fonksiyonu vardır.
- fork fonksiyonu ile yeni işlem yaratırken, exec fonksiyonu ile yeni programı başlatabiliriz.

```
#include <unistd.h>

int execl(const char *pathname, const char *arg0, ... /* (char *)0
*/ );

int execv(const char *pathname, char *const argv []);

int execle(const char *pathname, const char *arg0, .../* (char *)0,

char *const envp[] */ );

int execve(const char *pathname, char *const argv[], char *const
envp []);

int execlp(const char *filename, const char *arg0, ... /* (char *)0
*/ );

int execvp(const char *filename, char *const argv []);
```

Döünş: hata ise -1, OK ise dönüş yok.

exec fonksiyonları

- İlk dört fonksiyon yoladı argüman alırken son iki fonksiyon bir dosya ismi argüman alır.
- Bir dosya adı verildiğinde
  - Dosya adı / ile başlıyor ise yoladı olarak alınır.
  - Aksi halde çalıştırılabilir dosya PATH ismi ile tanımlı çevre değişkeninde belirtilen yerde aranır.
    - PATH=/bin:/usr/bin:/usr/local/bin/:.
- execlp veya execvp yoladı olarak verilen dosyayı bulunur ancak bu dosya çalıştırılabilir bir dosya değil ise, fonksiyon bu dosyayı bir kabuk komut dizisi (shell script) olarak kabul eder ve /bin / sh ile bu dosyayı çalıştırır.
- Diğer fark argüman listesinin verilme biçimidir.
  - l ise liste olarak, v ise vektor olarak argümanlar verilir.

exec fonksiyonları

- execl, execlp ve execle fonksiyonları argümanların ayrı ayrı verilmesini bekler. Argümanların bittiği null işaretçi ile belirtilir.
- Diğer üç fonksiyon için ise (execv, execvp ve execve) için ise argümanlar bir diziye yazılmalı ve bu dizinin adresi bu fonksiyonlara verilmelidir.
- Fonksiyonlar arasındaki son fark çevre listesinin yeni programa gönderilme şekli konusundadır.
  - Sonu e harfi ile biten iki fonksiyon (execle ve execve) çevre metinlerine bir işaretçi göndermemize izin verir.
  - Diğer dört fonksiyon ise environ değişkenini kullanır ve işlemin yeni programa iletilmesi için bulunan çevre değişkenlerini kopyalar.

exec fonksiyonları

Function	*pathname*	*filename*	*fd*	Arg list	*argv*[ ]	environ	*envp*[ ]
execl	•			•		•	
execlp		•		•		•	
execle	•			•			•
execv	•				•	•	
execvp		•			•	•	
execve	•				•		•
fexecve			•		•		•
(letter in name)		p	f	l	v		e

exec fonksiyonları

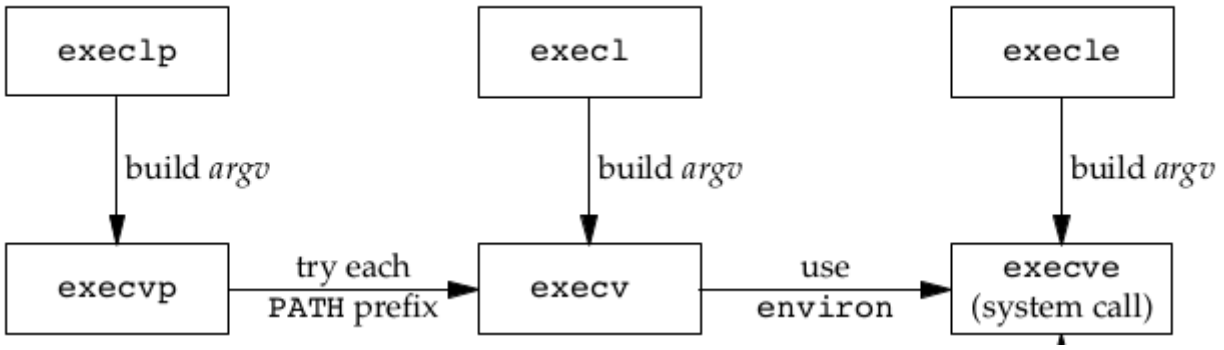
- Yeni çalışacak olan program bu programı başlatan işlemden aşağıdaki bilgileri alır.
  - Işlem numarası ve üst işlem numarası
  - Gerçek kullanıcı numarası ve gerçek grup numarası
  - Destekleyici grup numarası
  - Oturum numarası
  - Kontrol edilen terminal
  - Alarm saatine kalan zaman
  - Kök klasör
  - Dosya modu ve yaratma maskesi
  - Dosya kilitleri
  - Işlem sinyal maskesi
  - Bekleyen sinyaller
  - Kaynak limitleri
  - tms_utime, tms_stime, tms_cutime ve tms_cstime değerleri

exec fonksiyonları

- Açık dosyaların durumu close-on-exec bayrağının değerine göre belirlenir.
  - Bayrak ayarlanmamış ise açık dosyalar açık olarak kalır.
  - Bayrağın ayarlanması için fcntl fonksiyonu ile ayarlamak gerekir.
- POSIX.1 standartlarına göre exec fonksiyonu çalıştırıldığında açık klasör akışlarının kapanması gereklidir.
  - Bu genellikle opendir fonksiyonunun çağırılması ile yapılır.
  - opendir fcntl kullanarak açık olan klasör akışları için close-on-exec bayrağını ayarlar.
- Gerçek kullanıcı numarası ve gerçek grup numarası exec fonksiyonu ile değişmez. Ancak efektif kullanıcı numarası set-user-ID ve set-group-ID bayraklarının değerine göre değişiklik gösterebilir.

exec fonksiyonları

- Unix sistemlerinde genellikle bu altı fonksiyonun sadece biri, execve, kernel içerisinde çalışan bir sistem çağrısı olarak tanımlanır. Diğer fonksiyonlar ise sadece kütüphane fonksiyonlarıdır ve sonuçta execve fonksiyonunu çağırır.



- Örnek31.